Introducere în Reinforcement Learning

Cursul #3

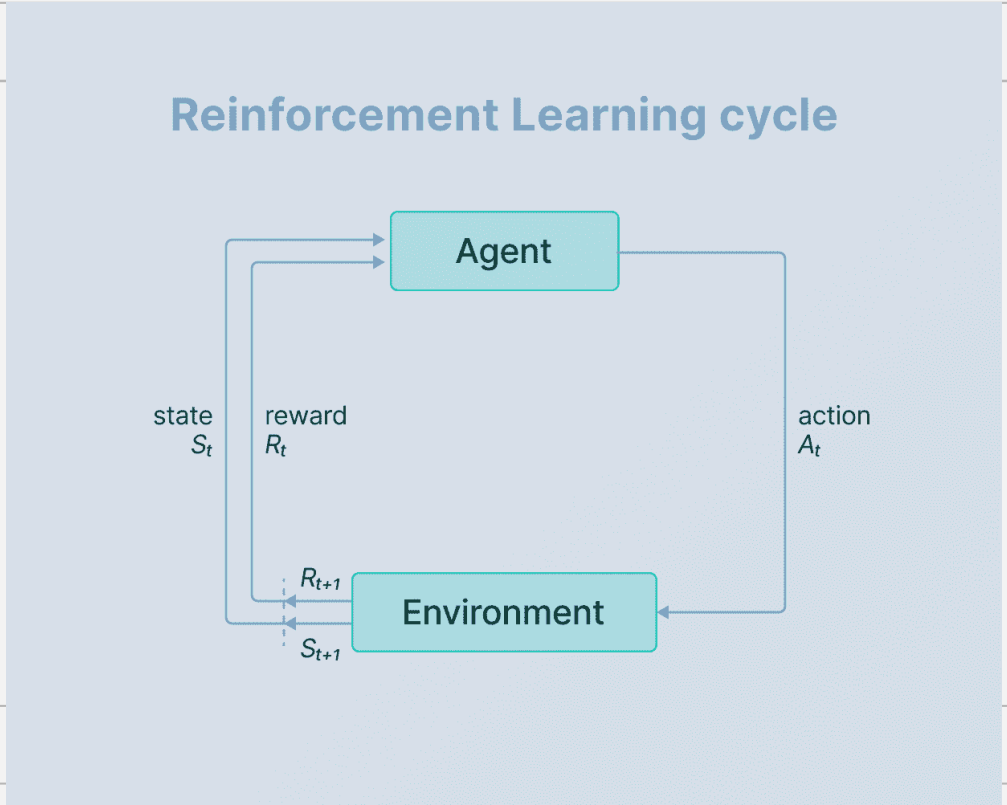
2025 – 2026 (Sem. I)
Drd. Ștefan Iordache
Conf. Dr. Ciprian Păduraru
Facultatea de Matematică și Informatică
Universitatea din București

Cuprins

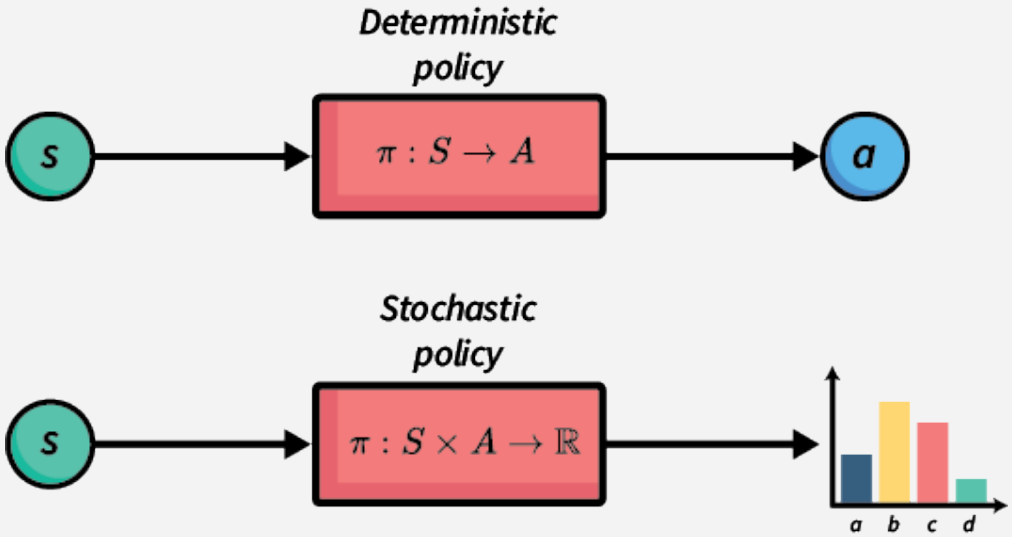
01

Recapitulare

Diagrama generică - RL



Procese Stochastice



Ecuația Bellman – V* (optim)

$$V(s) = \overbrace{\max_{a \in A(s)}}^{\text{best action from } s} \overbrace{\underbrace{\sum_{s' \in S}}_{\text{for every state}} P_a(s' \mid s)\left[\underbrace{r(s,a,s')}_{\text{immediate reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{V(s')}_{\text{value of } s'} \right]}^{\text{expected reward of executing action } a \text{ in state } s}$$

02

Algoritmi Value-Based

Value-Based vs. Policy-Based vs. Hybrid

Value-Based RL

- Scopul: învățarea valorilor stărilor – V(s)
- Extragem politica din V(s)
- Când? Medii finite / de dimensiune mică

Policy-Based RL

- Scopul: învățarea directă a politicii
- Facem by-pass învățării V(s)
- Când? Medii infinite / de dimensiune mare

Hybrid RL

- Scopul #1: antrenare eficientă și rezultate mai bune!
- Scopul #2: depășirea limitărilor impuse de metodele anterioare (varianța mare pentru policy-based / lucrul cu spațiile continue pentru value-based)

Algoritmul #1 – Value Iteration

- **Programare Dinamică!**
- Căutăm V* (value function optim) rezolvând ecuația Bellman, iterativ.
- **V aproximează V***
- Funcția V este îmbunătățită iterativ, scopul fiind convergența spre V*.
- Pot exista mai multe funcții V optime, cu politici rezultate diferite!

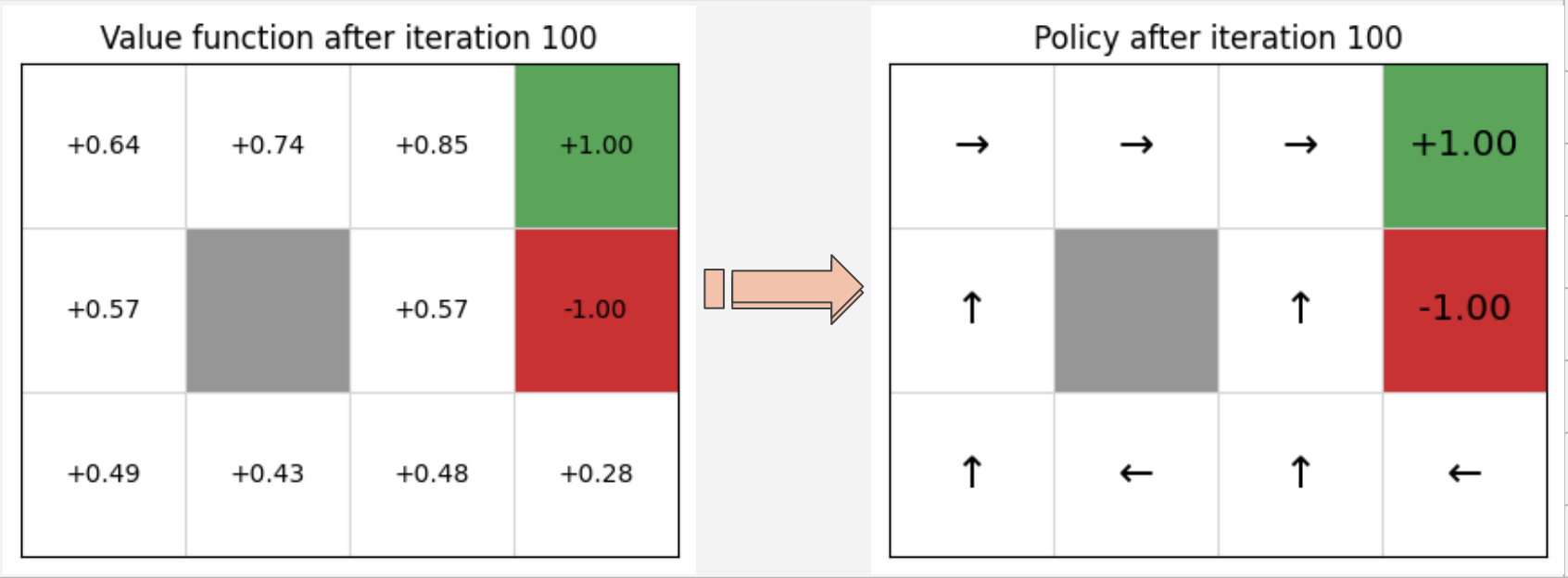
Algorithm 1 (Value iteration)

Input : MDP $M = \langle S, s_0, A, P_a(s' \mid s), r(s,a,s') \rangle$
Output : Value function V

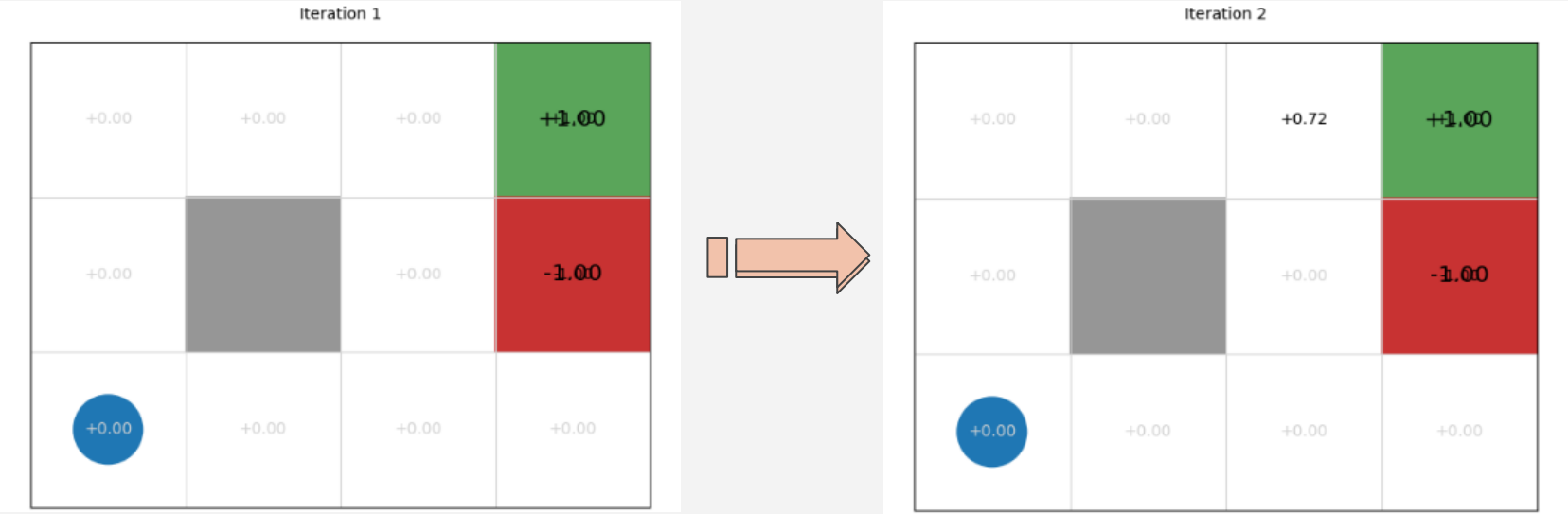
Set V to arbitrary value function; e.g., $V(s) = 0$ for all s

repeat
  $\Delta \leftarrow 0$
  for each $s \in S$
    $V'(s) \leftarrow \max_{a \in A(s)} \sum_{s' \in S} P_a(s' \mid s) \, [r(s,a,s') + \gamma \, V(s')]$ (Bellman equation)
    $\Delta \leftarrow \max(\Delta, |V'(s) - V(s)|)$
  $V \leftarrow V'$
until $\Delta \leq \theta$

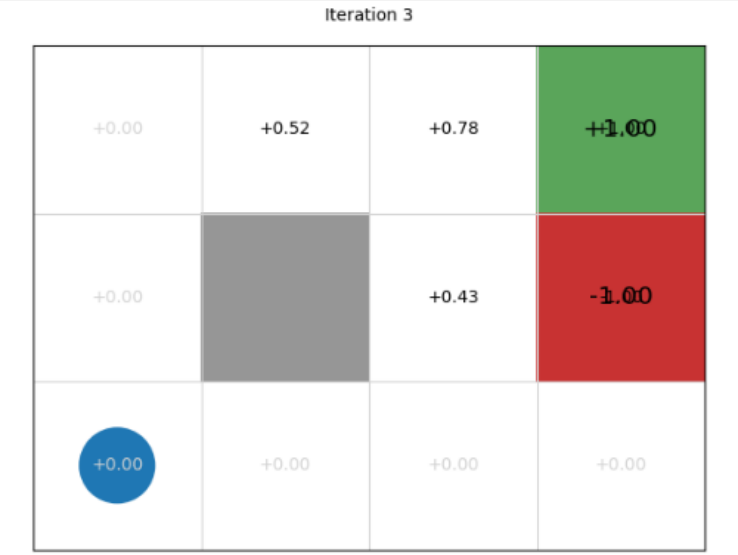
Value Iteration – Grid World

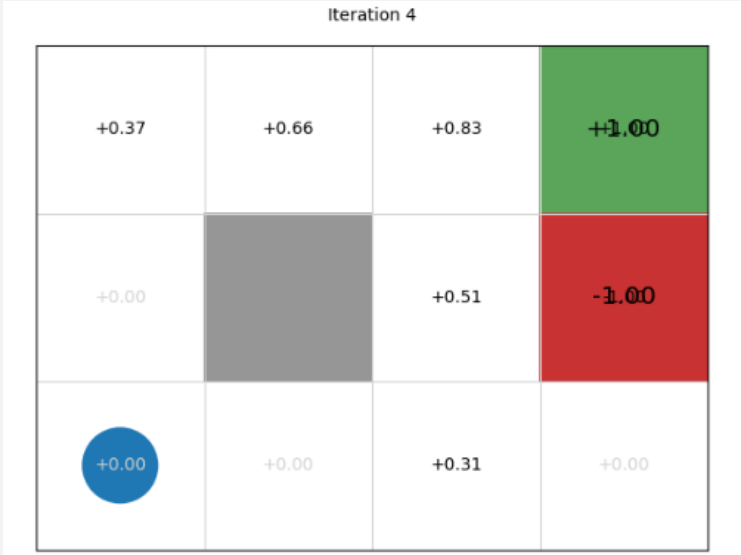


Value Iteration – Grid World...But How???

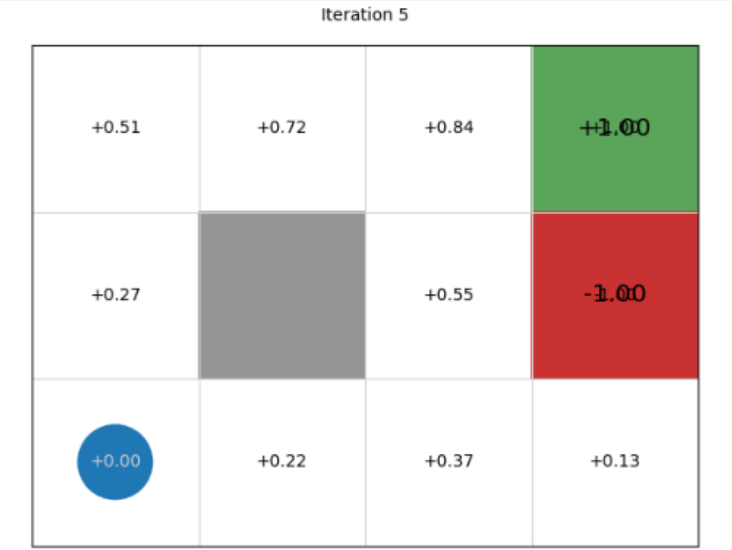


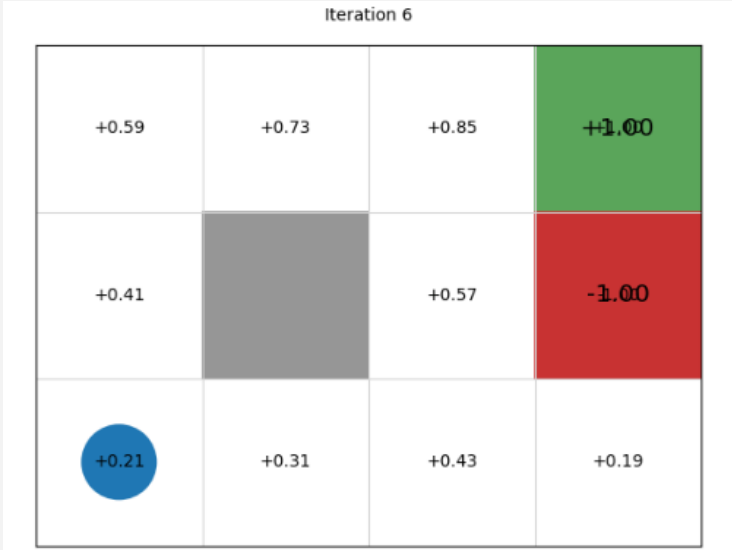
Value Iteration – Grid World...But How???



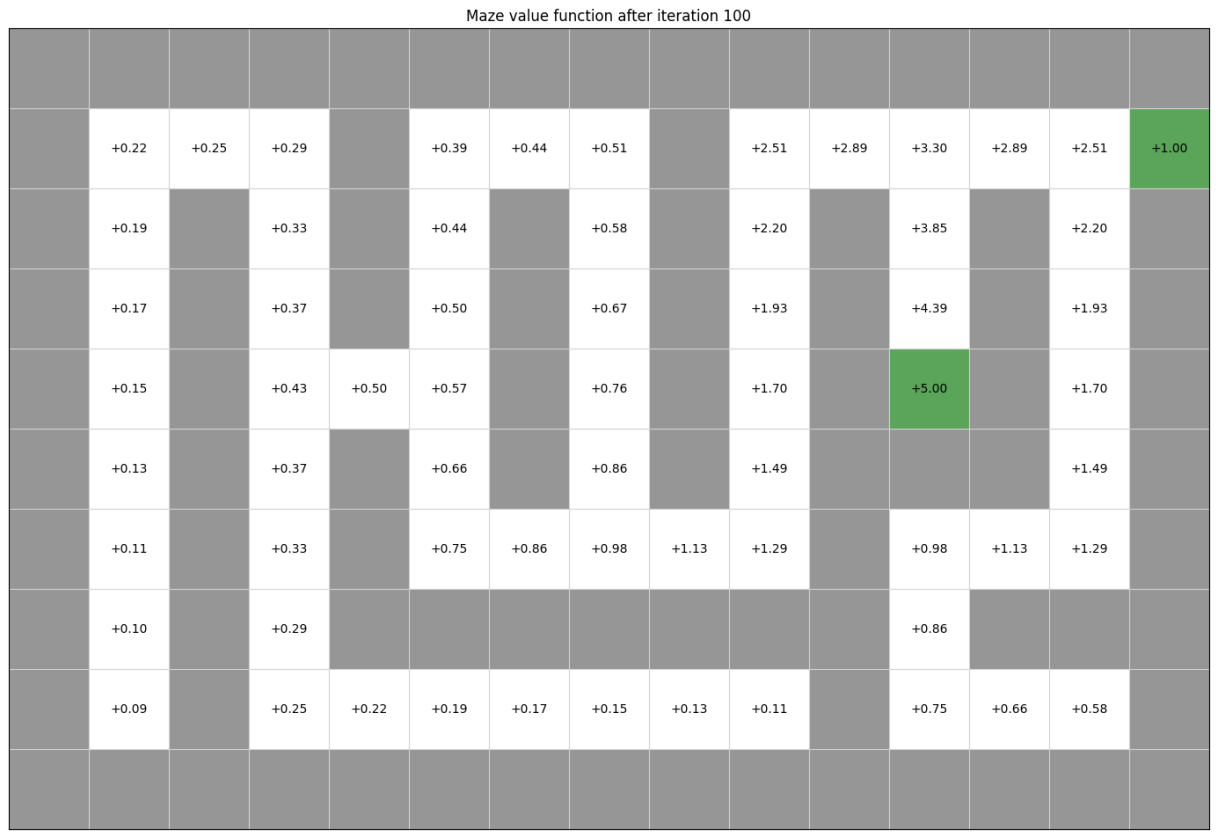


Value Iteration – Grid World...But How???

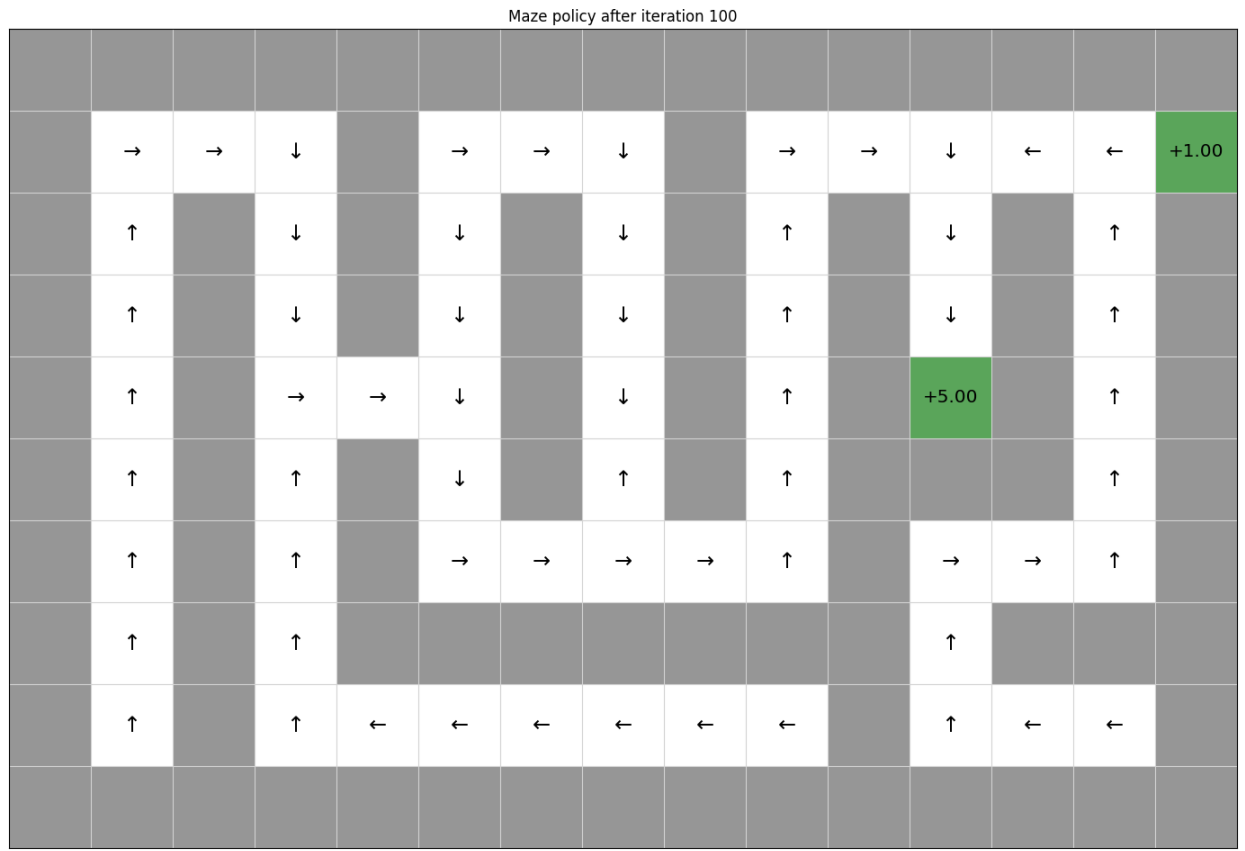




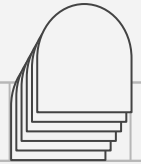
Value Iteration – Labirint



Value Iteration – Labirint



Value Iteration - Concluzii

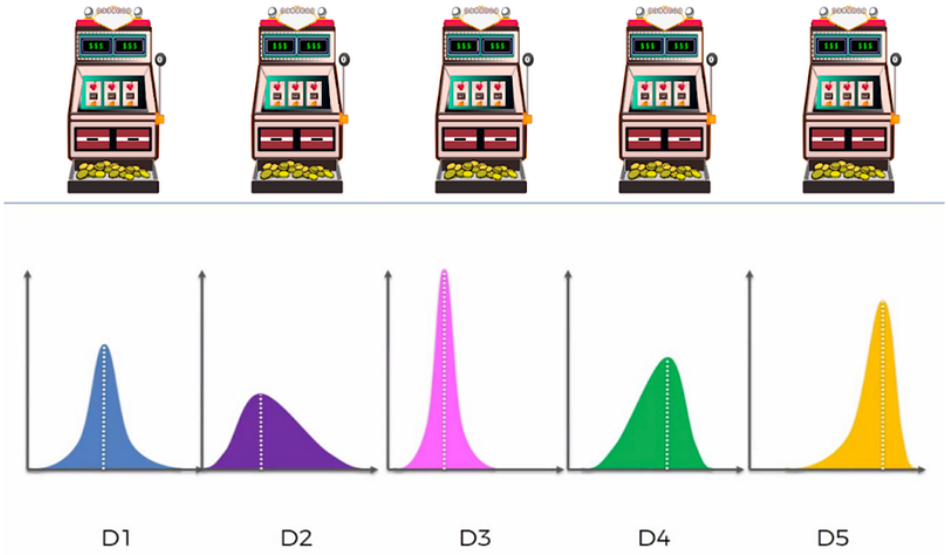


- Complexitate? $O(|S|^2 * |A|)$
- De câte iterații este nevoie? **Depinde...în funcție de θ! Nu putem ști!**
- Avantajul? **Produce mereu o politică optimă, dar teoretic necesită un număr infinit de iterații.**
- Când utilizăm Value Iteration? **Pentru problemele mici-medii, unde putem determina o politică optimă într-un timp rezonabil.**

Există ceva mai bun?

NORMAL!!!
Altfel terminam materia și plecam acasă!

Multi-Armed Bandits (MAB)



- **Scenario:** Pull machine k → sample from ***unknown*** reward distribution D_k → observe reward.
- **Problem:** Given a finite number of pulls T, how can I optimize my winnings?
- How much should I **explore**? How much should I **exploit**?

Multi-Armed Bandits (MAB)

- Ce ne dorim? **Să executăm doar acțiunile bune!**
- Dar inițial nu știm nimic despre mediu, nu cunoaștem distribuțiile recompenselor.
- Pentru a **exploata (exploit)** trebuie să **explorăm (explore)** prima dată!
- Dar cât explorăm? Dacă pierdem ceva și nu mai câștigăm suficient?
- **”Fear of Missing Out (FOMO)”** – da, și algoritmii au sentimente!
- Scopul nostru devine **minimizarea regretului**

$$\mathcal{R}(\pi, t) = t \cdot \max_{a} Q^*(a) - \mathbb{E}[\sum_{k=1}^{t} X_{\pi(k),k}]$$

Q*(a) – recompensa medie pentru acțiunea a

Algoritmul #2 – Abstract Multi-Armed Bandit

Algorithm 2 (Abstract multi-armed bandit)

Input : Multi-armed bandit problem $M = \langle \{X_{i,k}\}, A, T \rangle$
Output : Q-function Q

$Q(a) \leftarrow 0$ for all arms $a \in A$
$N(a) \leftarrow 0$ for all arms $a \in A$

$k \leftarrow 1$
while $k \leq T$ **do**
    $a \leftarrow \text{select}(k)$
    Execute arm a for round k and observe reward $X_{a,k}$
    $N(a) \leftarrow N(a) + 1$
    $Q(a) \leftarrow Q(a) + \frac{1}{N(a)} [X_{a,k} - Q(a)]$
    $k \leftarrow k + 1$

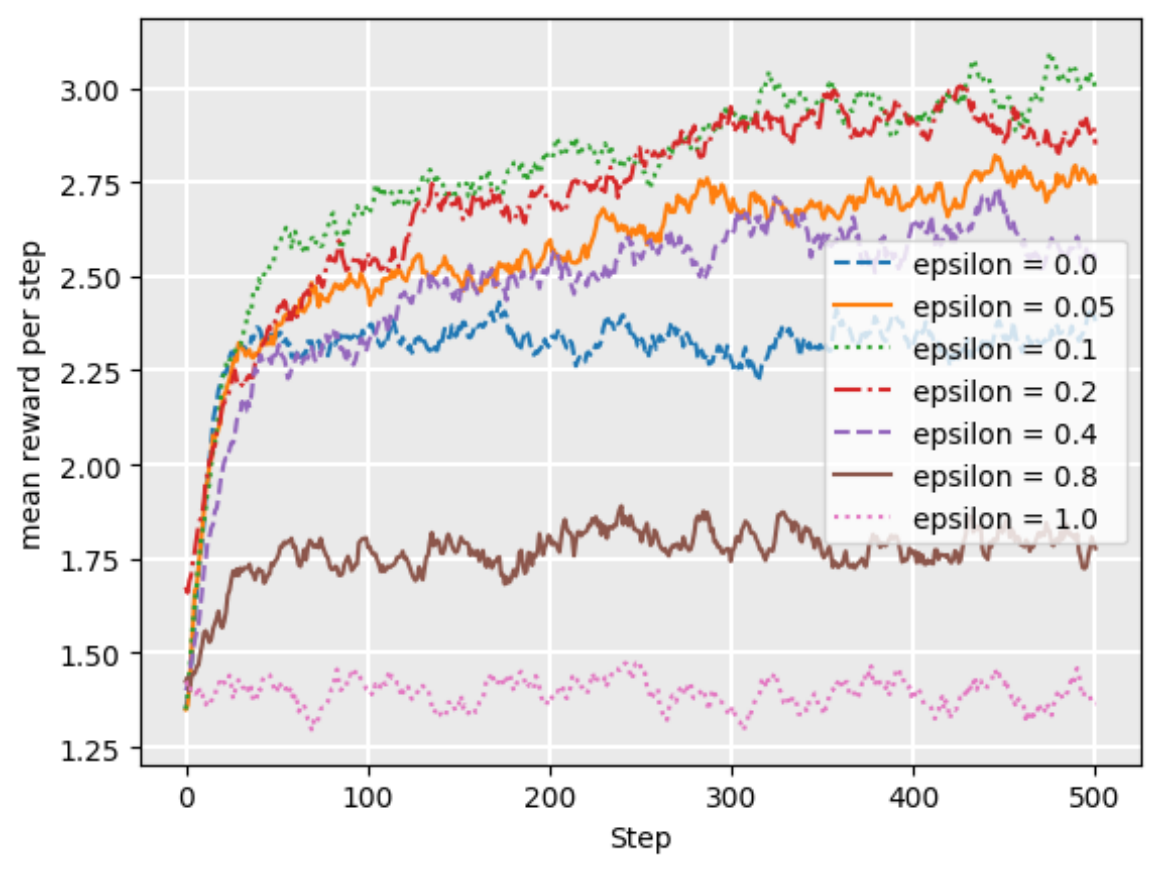
ϵ-Greedy

ϵ-Greedy -> Explore vs. Exploit !!!

- Toate cele **m** acțiuni sunt încercate cu probabilitate diferită de zero.
- Alegem cu probabilitate **1 - ϵ** acțiunea greedy.
- Cu probabilitate **ϵ** alegem o acțiune aleatorie.
- Există o valoare câștigătoare? **Nu!** Dar în practică 0.05 – 0.2 funcționează foarte bine.

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q(s,a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

ϵ-Greedy


3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
mean reward per step
0
100
200
300
400
500
Step
epsilon = 0.0
epsilon = 0.05
epsilon = 0.1
epsilon = 0.2
epsilon = 0.4
epsilon = 0.8
epsilon = 1.0

ϵ-Greedy - Decay

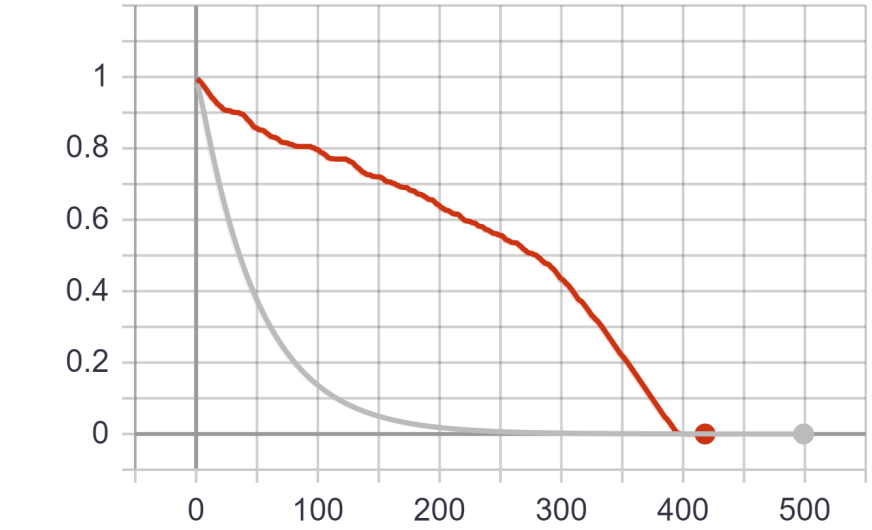
ϵ-Greedy -> Explore vs. Exploit !!!

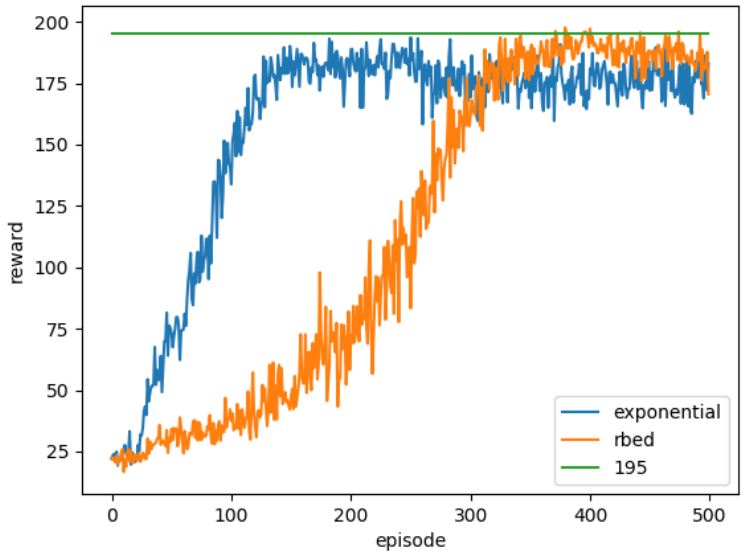
- Explorarea este preferată atunci când agentul nu știe prea multe despre mediu.
- Alegem exploatarea spre finalul sesiunilor de antrenare.
- Putem introduce un **factor de decay / degradare α (între 0 și 1), pe care îl înmulțim cu ε după fiecare episod.**
- Pentru un control mai avansat există și **Reward Based Decay.**

```
if EPSILON > MINIMUM_EPSILON and LAST_REWARD >= REWARD_THRESHOLD:
    EPSILON = DECAY_EPSILON(EPSILON)
    REWARD_THRESHOLD = INCREMENT_REWARD(REWARD_THRESHOLD)
```

ϵ-Greedy - Decay

- **Reward Based Decay**
- Exponential Decay





Model-Based vs. Model-Free

Model-Based

- Cunoaștem complet modelul lumii / mediul.
- $P_a(s' \mid s)$ și $r(s, a, s')$
- Putem aplica mereu această idee? **NU!**

Model-Free

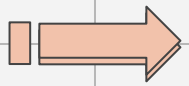
- Acționăm fără să cunoaștem mediul.
- Învățăm prin experiențe / explorare!
- Ne interesează învățarea V(s) / Q(s,a) sau a politicii.

Cine este Q???

- Funcție similară cu V(s), **Q : S x A -> R**
- **Calculează calitatea (quality) combinațiilor stare-acțiune.**

Q-Tables

State	Up	Down	Right	Left
(0, 0)	0.00	0.00	0.00	0.00
(0, 1)	0.00	0.00	0.00	0.00
(0, 2)	0.00	0.00	0.00	0.00
(1, 0)	0.00	0.00	0.00	0.00
(1, 1)	0.00	0.00	0.00	0.00
(1, 2)	0.00	0.00	0.00	0.00
(2, 0)	0.00	0.00	0.00	0.00
(2, 1)	0.00	0.00	0.00	0.00
(2, 2)	0.00	0.00	0.00	0.00
(3, 0)	0.00	0.00	0.00	0.00
(3, 1)	0.00	0.00	0.00	0.00
(3, 2)	0.00	0.00	0.00	0.00



State	Up	Down	Right	Left
(0, 0)	0.50	0.42	0.39	0.42
(0, 1)	0.56	0.44	0.51	0.51
(0, 2)	0.58	0.51	0.63	0.57
(1, 0)	0.09	0.18	0.06	0.43
(1, 1)	0.00	0.00	0.00	0.00
(1, 2)	0.64	0.65	0.74	0.59
(2, 0)	0.41	0.00	0.00	0.00
(2, 1)	0.69	0.09	-0.24	0.24
(2, 2)	0.79	0.61	0.90	0.65
(3, 0)	-0.02	0.00	0.00	0.00
(3, 1)	0.00	0.00	0.00	0.00
(3, 2)	0.00	0.00	0.00	0.00

Algoritmul #3 – Monte-Carlo

Algorithm 3 (Monte-Carlo reinforcement learning)

Input : MDP $M = \langle S, s_0, A, P_a(s' \mid s), r(s, a, s') \rangle$
Output : Q-function Q

Initialise Q arbitrarily; e.g., $Q(s, a) \leftarrow 0$ for all s and a
$N(s, a) \leftarrow 0$ for all s and a

repeat
  Generate an episode $(s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T)$;
    e.g. using Q and a multi-armed bandit algorithm such as ϵ-greedy
  $G \leftarrow 0$
  $t \leftarrow T - 1$
  while $t \geq 0$ **do**
    $G \leftarrow r_{t+1} + \gamma \cdot G$
    if s_t, a_t does not appear in $s_0, a_0, \ldots, s_{t-1}, a_{t-1}$ **then**
      $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{1}{N(s_t, a_t)} [G - Q(s_t, a_t)]$
      $N(s_t, a_t) \leftarrow N(s_t, a_t) + 1$
    $t \leftarrow t - 1$
until Q converges

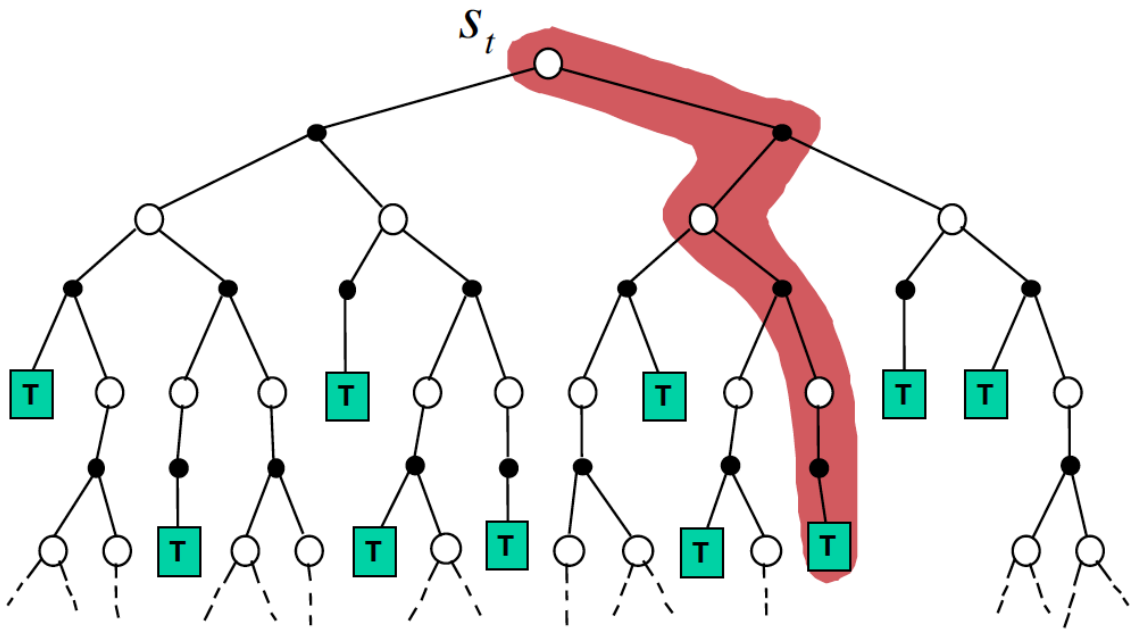
Algoritmul #3 – Monte-Carlo

- Metodele MC învață **direct din experiențe (episodice).**
- MC este **model-free:** nu cunoaște tranzițiile sau recompensele procesului decizional Markov (MDP).
- MC invață din **episoade complete.**

- MC folosește *cea mai simplă idee*: **mean return**
- MC poate fi aplicat pe **MDP-urile episodice** → ***pentru toate episoadele există final***

Algoritmul #3 – Monte-Carlo - Sampling

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



03

Temporal Difference

Ce este Temporal Difference?

TD are la bază strategia de învățare din ***episoade incomplete prin bootstrapping.***

MC vs. TD

Monte Carlo

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

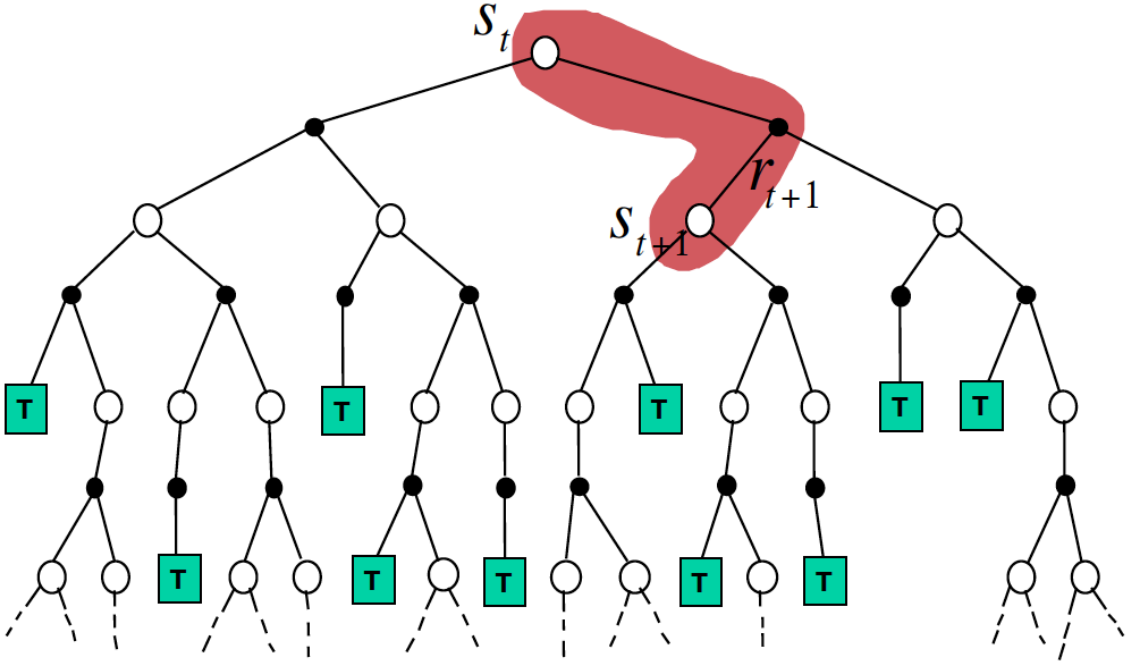
Temporal Difference

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

- $R_{t+1} + \gamma V(S_{t+1})$ poartă denumirea de "temporal difference target".
- $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ reprezintă eroarea TD.

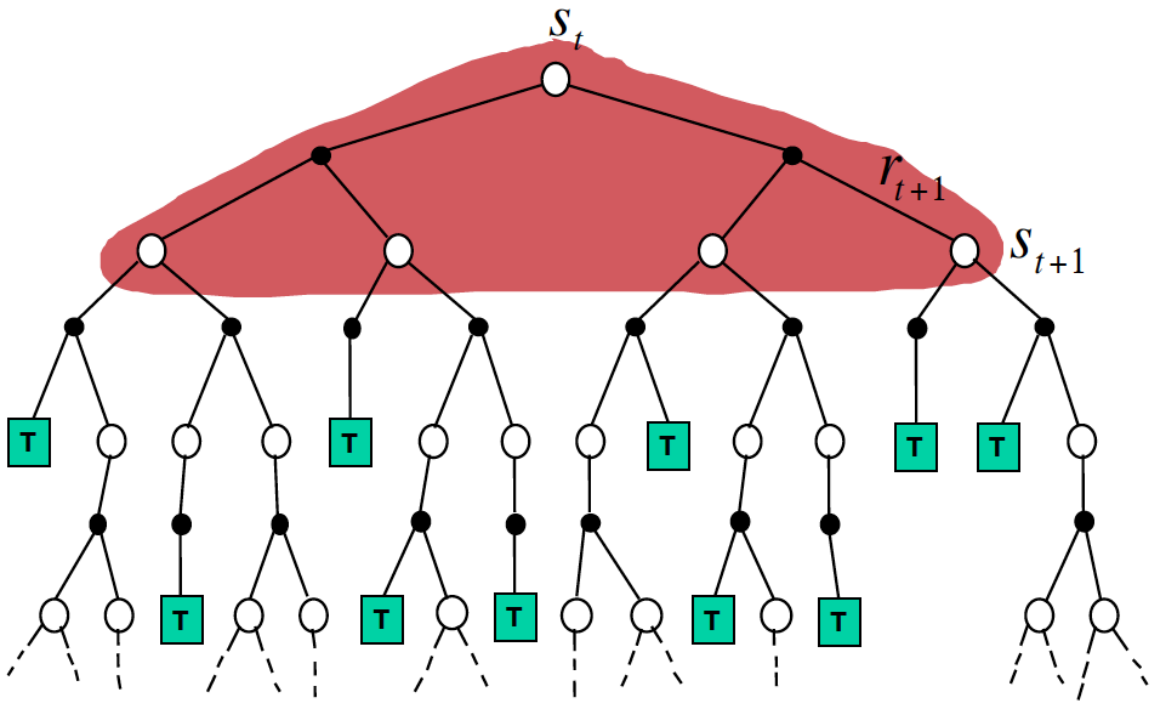
Temporal Difference - Bootstraping

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



Căutarea Exhaustivă

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1}) \right]$$



Avantaje & Dezavantaje TD vs. MC

TD învață înainte de a ști rezultatul final! Nu are nevoie de return/ outcome.

- Învățare “online” – TD.
- MC așteaptă până la finalul episodului pentru a afla return-ul.

- TD învață din secvențe parțiale, iar MC doar din episoade complete.
- TD funcționează în medii continue, fără stări terminale. MC nu poate ajunge la această performanță.

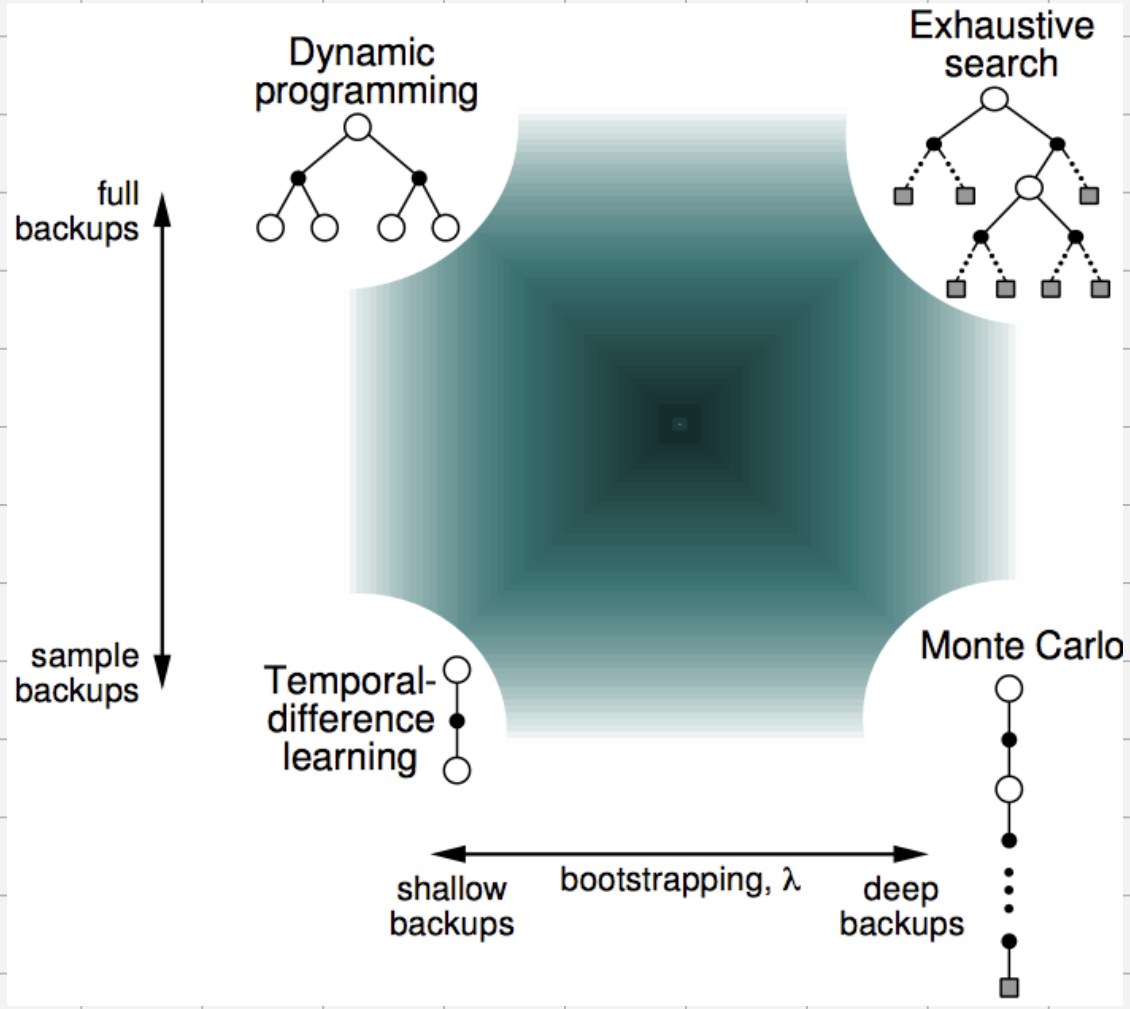
MC

- Varianță mare, bias zero!
- Convergență bună!
- Simplu de înțeles și aplicat!

TD

- Mai eficient fată de MC!
- Sensibil la punctul de plecare (valoare inițială)!

Marea pictogramă a RL-ului!


Dynamic programming
Exhaustive search
full backups
sample backups
Temporal-difference learning
Monte Carlo
shallow backups
bootstrapping, λ
deep backups

Thanks!

Este timpul pentru întrebări!!!

Acum…sau pe email:

stefan.iordache10@s.unibuc.ro