Introducere în Reinforcement Learning

Cursul #1

Ștefan Iordache, Cătălina Iordache, Ciprian Păduraru

Cuprins

01 Organizatorice

Desfășurare & Examinare

02 Introducere

Ce înseamnă Reinforcement Learning (RL)?

01

Organizatorice

Desfășurare &
Examinare

Organizatorice #1

Structură

Curs

- 2 ore/săptămână
- Joi: 18-20

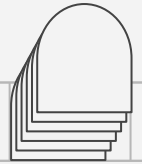
Laborator

- 4 laboratoare/săptămână
- 2 ore/laborator

Detalii

- Prezență obligatorie?
  Nu!
- Activitate cât mai mare?
  Da!
- Examen teoretic?
  Nu!
- Proiect?
  Da!
- Când & cum?
  La finalul semestrului, în sesiune, în echipe (**3-5**)

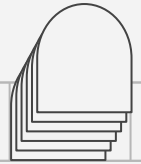
Organizatorice #2



Tehnologii folosite

- **Python (3.6-3.x)**
- **Jupyter Notebook**
- **OpenAI Gym**

Organizatorice #3



- *”Biblia” Reinforcement Learning*

  “Reinforcement Learning – An Introduction” – Richard S. Sutton & Andrew G. Barto

- *Referințe*

1. Stanford CS234 & Waterloo CS885
2. Deep RL Course from Berkeley CS285

02

Introducere

Ce înseamnă Reinforcement Learning (RL)?

Premisa

Provocarea cea mai mare în inteligența artificială este ***deducerea*** unor ***decizii bune*** sub spectrul ***incertitudinii***.

Obiective & Metodologie

Orice problemă de RL → 01 Explorare → 02 Optimizare → 03 Generalizare

Metodologie:

- *Explorarea mediului*
- *Folosirea experienței pentru decizii viitoare*

Cum funcționează deciziile?

Impact imediat sau întârziat?

Ambele! În cazul oricărei decizii luate impactul va fi atât *imediat* cât și pe *termen lung*. Este necesar să cântărim beneficiile acțiunilor în ambele cazuri!

Ce înseamnă o decizie bună?

Problemele din lumea reală **nu** au întotdeauna o ”cea mai bună soluție”, în practică având nevoie de să definim ***calitatea*** unei *acțiuni* sau a unei *decizii*.

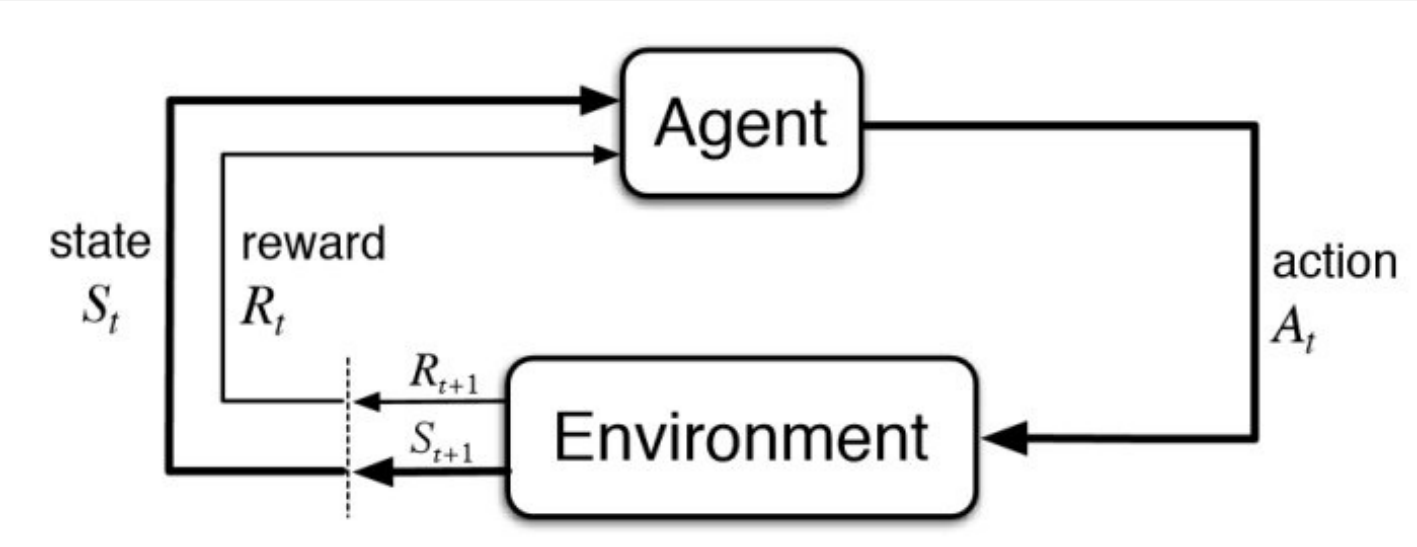
Avem la dispoziție toate datele?

Niciodată! În cazul problemelor de Reinforcement Learning **nu** avem un set de date complet, prestabilit, ci acesta ***este dedus din interacțiunea cu mediul***.

Mică comparație!

	Învățare supervizată	Învățare nesupervizată	Reinforcement Learning
Explorare	**NU**	**NU**	**DA**
Generalizare	**DA**	**DA**	**DA**
Optimizare	**NU**	**NU**	**DA**
Folosește un set de experiențe	**DA**	**DA**	**DA**

Diagrama generică - RL



Domenii de aplicabilitate - RL

1. Jocuri video
2. Mașini autonome
3. Robotică
4. Planificatoare
5. Domeniul Financiar
6. Sisteme de recomandare
7. Medicină

Thanks!

Este timpul pentru întrebări!!!

stefan.iordache10@s.unibuc.ro
catalina.patilea@s.unibuc.ro
ciprian.paduraru@fmi.unibuc.ro

+40 7.. ... ...

Introducere în Reinforcement Learning

Cursul #2

Ștefan Iordache, Cătălina Iordache, Ciprian Păduraru

Cuprins

01 Concepte generale

Primii pași în RL

02 Procese Markov

Baza algoritmilor noștri!

03 Algoritmi RL

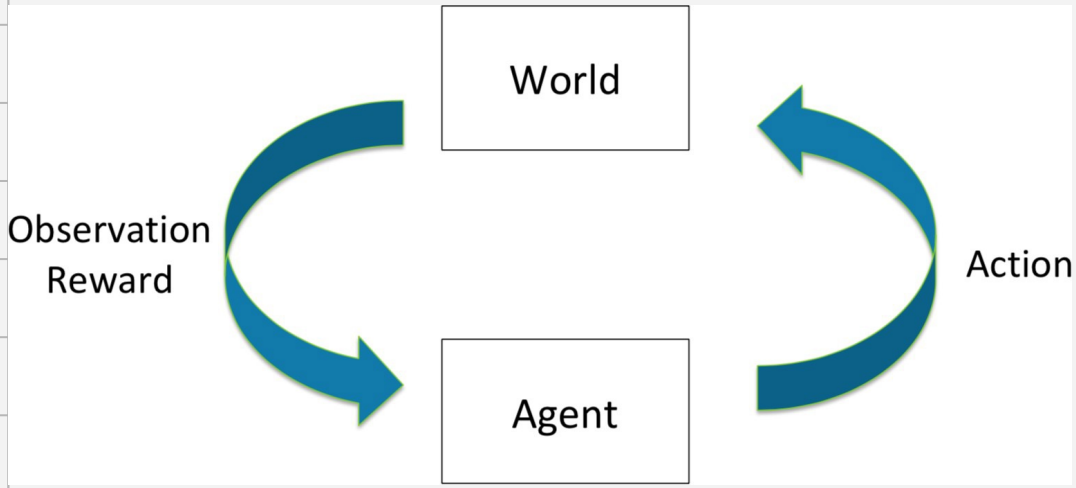
Începe distracția!

01

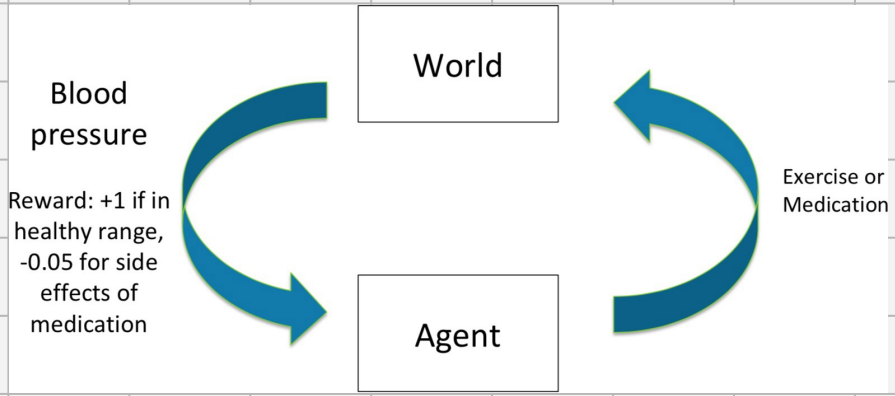
Concepte generale

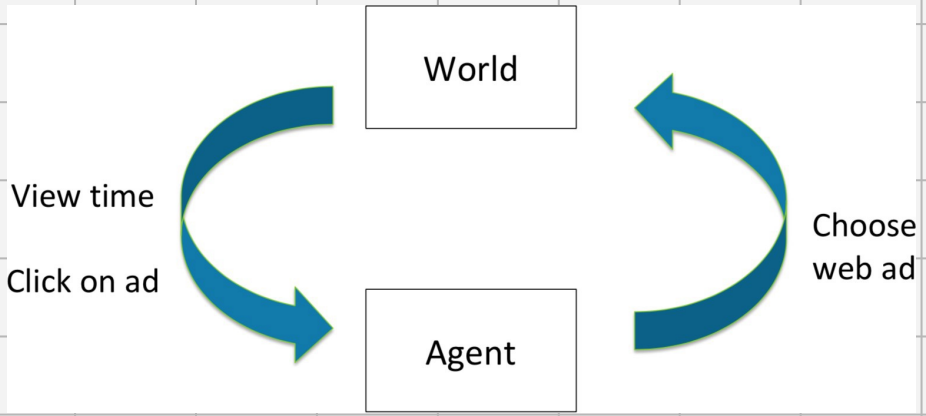
Primii pași în RL

Diagrama generică - RL

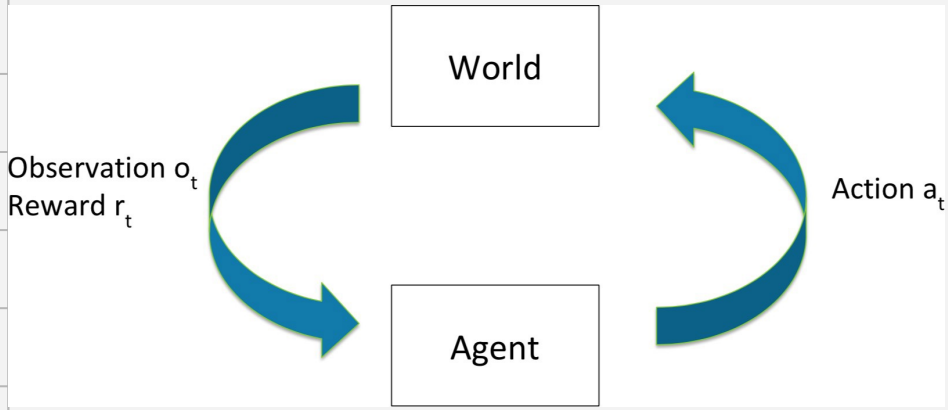


- **Scopul**: alegerea acțiunilor care *măresc recompensele viitoare*.
- Necesită ***balansarea*** recompenselor pe *termen scurt și lung*.


World
Blood pressure
Reward: +1 if in healthy range, -0.05 for side effects of medication
Exercise or Medication
Agent


World
View time
Click on ad
Choose web ad
Agent

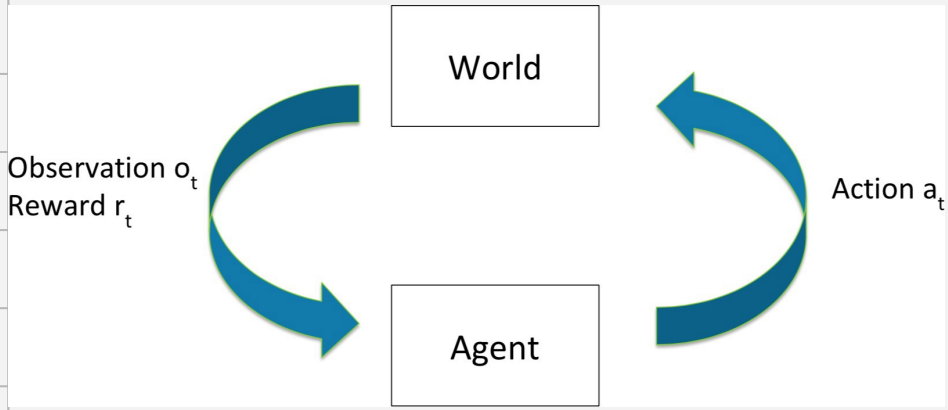
Procesul decizional: Agentul & Mediul



Pentru fiecare moment **t** de timp:

- Agentul execută o **acțiune a_t**
- Mediul este actualizat în urma acțiunii a_t și emite **observația o_t**, respectiv **recompensa r_t**
- Agentul **primește** cele două rezultate din mediu: observația și recompensa

Istoricul observațiilor



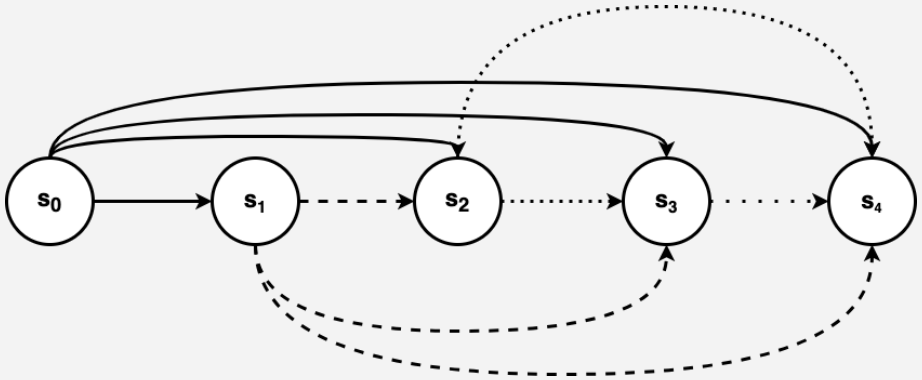
Istoricul la un moment de timp determinat, t:

$$h_t = \{a_1, o_1, r_1, \dots, a_t, o_t, r_t\}$$

- **Acțiunile sunt alese în baza istoricului. Cum?**

În baza unei funcții $\mathbf{s_t = f(h_t)}$

Procese Stochastice



Dinamica unui astfel de proces:

$$Pr(s_t | s_{t-1}, s_{t-2}, \dots, s_0)$$

- **Setul de stări este notat cu S.**
- **Problema: orizont infinit al funcției Pr.**

Care este soluția???

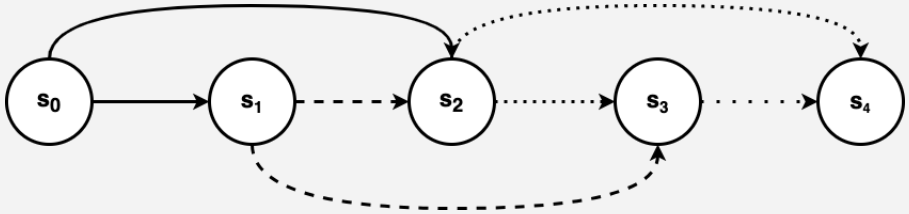
- **Procese Markov**

02

Procese Markov

Baza algoritmilor
noştri!

Procese Markov



Starea curentă depinde doar de un set finit de stări din trecut.

- **Procese Markov de ordinul I**

$$Pr(s_t|\ s_{t-1}, \dots, s_0) = Pr(s_t|\ s_{t-1})$$

- **Procese Markov de ordinul II**

$$Pr(s_t|\ s_{t-1}, \dots, s_0) = Pr(s_t|\ s_{t-1}, s_{t-2})$$

Procese Markov – staționaritate

În mod implicit, un proces Markov se referă la cel de ordinul I.

$$Pr(s_t|\, s_{t-1}, \dots, s_0) = Pr(s_t|\, s_{t-1})\ \forall t$$

- Extindem reprezentarea de mai sus către termenul de ***proces staționar:***

$$Pr(s_t|\, s_{t-1}) = Pr(s_{t'}|\, s_{t'-1})\ \forall t'$$

Avantajul unui proces staționar:

reprezentarea simplă! $Pr(s'|s)$

Întrebări naturale...

- **Cum reprezentăm stările? Cât de multe caracteristici adăugăm?**
  - Răspuns: Până când procesul poate fi considerat Markovian, respectiv staționar.

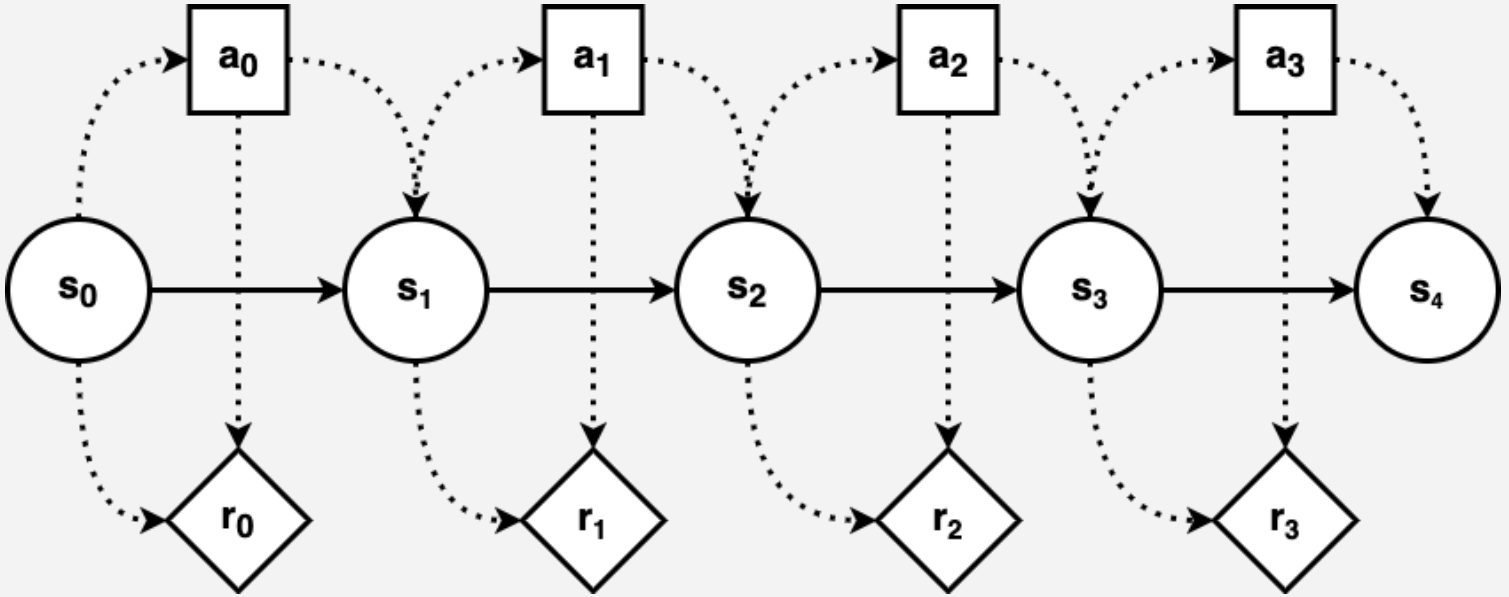
- **Există posibilitatea să adăugăm prea multe componente unei stări?**
  - Răspuns: Da! Adăugarea de componente va crește complexitatea calculelor, implicit necesarul computațional.
  - Soluția: Căutam cel mai mic subset de caracteristici care descrie procesul complet procesul Markovian!

- **Ce utilizăm în practică?**
  - Răspuns: Cel mai frecvent vom utiliza următoarea presupunere -> $s_t = o_t$

Despre decizii

- **Considerăm faptul că simplele predicții sunt inutile. De ce???**
  - Răspuns: Dorim să obținem informații care vor influența alegerile viitoare, nu simple predicții utile unui singur moment.
- **Astfel, sarcina noastră constă în conceperea unor algoritmi capabili să furnizeze decizii!**
- **Dar, cum influențăm deciziile? Cum construim acest algoritm?**
  - Răspuns: ***Procese Decizionale Markov!***

Procese Decizionale Markov



Întrebări naturale... (partea 2)

- **Stările sunt în continuare de tip Markov?**
- **Este lumea parțial observabilă?**
- **Dinamica este deterministă sau stochastică?**
- **Acțiunile influențează recompensa imediată sau afectează recompensele și starea următoare?**

03

Algoritmi RL

Începe distracția!

Algoritmi RL

Cuprind una sau mai multe dintre următoarele componente:

- **Model**
- **Politică (Policy)**
- **Value Function**

Modelul

- **Este reprezentarea agentului pentru felul în care mediul se va schimba în urma unei anumite acțiuni.**

- **Tranzițiile (felul în care agentul prezice următoarea stare):**

$$p(s_{t+1} = s' | s_t = s, a_t = a)$$

- **Metodologia de predicție a recompenselor:**

$$r(s_t = s, a_t = a) = E[r_t | s_t = s, a_t = a]$$

Politica (Policy) - π

- **O politică determină felul în care agentul alege acțiunile pe care le execută. Este o funcție de forma**

$$\pi: S \rightarrow A$$

- **Politici deterministe:** $\pi(s) = a$
- **Politici stochastice:** $\pi(a|s) = Pr(a_t = a|s_t = s)$

Value Function– V^{π}

- **Reprezintă suma recompenselor (cu discount), sub o anumită politică aplicată.**

$$V^{\pi}(s_t = s) = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots | s_t = s]$$

- **Factorul de discount (γ) va stabili importanța recompensei imediate și a celor viitoare.**
- **Metoda poate fi utilizată pentru a decide calitatea anumitor stări și acțiuni, ulterior stabilind o metodă de comparație între diverse politici.**

Thanks!

Este timpul pentru întrebări!!!

stefan.iordache10@s.unibuc.ro
catalina.patilea@s.unibuc.ro
ciprian.paduraru@fmi.unibuc.ro

+40 7.. ... ...

Introducere în Reinforcement Learning

Cursul #3

Ștefan Iordache, Cătălina Iordache, Ciprian Păduraru

Cuprins

01 Recapitulare

Repetiția: mama învățăturii!

02 Markov!

Da, insistăm!

01

Recapitulare

Repetiția: mama învățăturii!

Întrebare rapidă

Discount Factor

Într-un *proces decizional Markov*, un **factor de discount γ mai mare** (apropiat de 1) implică o ***importanță mai mare a recompenselor obținute pe termen scurt***, în comparație cu cele obținute pe *termen lung*?

Variante posibile

- **Adevărat**
- **Fals**

Întrebare rapidă

Discount Factor

Într-un *proces decizional Markov*, un **factor de discount γ mai mare** (apropiat de 1) implică o ***importanță mai mare a recompenselor obținute pe termen scurt***, în comparație cu cele obținute pe *termen lung*?

II

Răspuns

FALS! Doar valoarea 0 implică valorificarea exclusivă a recompensei imediate.

$$V^{\pi}(s_t = s) = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots | s_t = s]$$

Ne aducem aminte

- **MODELUL**
  - Un procedeu matematic pentru exprimarea dinamicii mediului și a recompenselor.
- **POLITICA (POLICY)**
  - Funcție utilizată de agent pentru a realiza asocieri între stări și acțiuni.
- **VALUE FUNCTION**
  - Funcție ce oferă drept răspuns suma recompensele viitoare, folosind drept parametri starea sau/și acțiunea (curente), aplicată sub o anumită politică.

02

Markov!

Da, insistăm!

O proprietate de bază

- **Starea = informație statistică a istoriei interacțiunilor cu mediul.**
- **Condiția unei stări s_{t+1} pentru a fi considerată de tip Markov:**

$$p(s_{t+1}|s_t, a_t) = p(s_{t+1}|h_t, a_t)$$

Lanțuri Markov

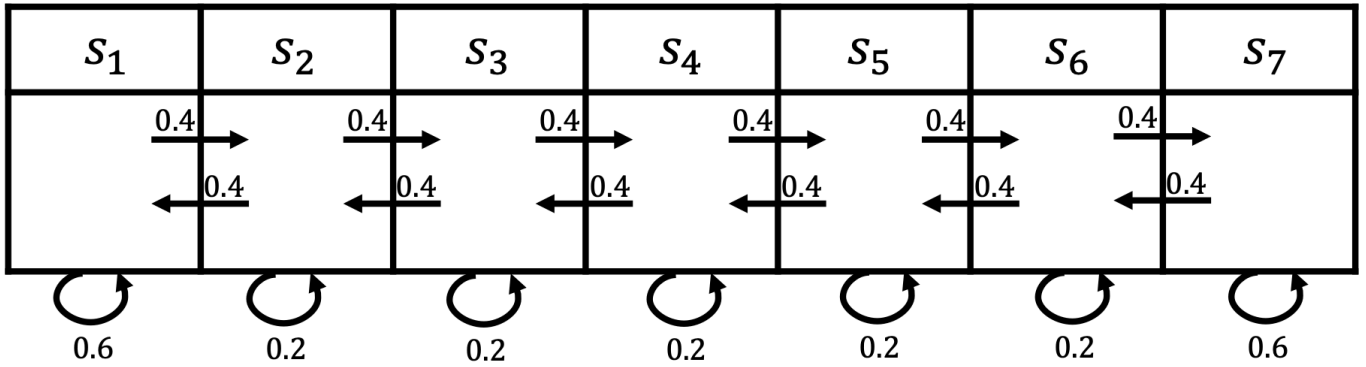
- **Proces aleatoriu** de tip **memoryless** (secvență de stări cu proprietatea Markov îndeplinită)
- Setul stărilor (S) este finit.
- P este modelul dinamic, ce explică tranzițiile

$$p(s_{t+1} = s' | s_t = s)$$

- **Atenție! Nu avem recompense sau acțiuni.**

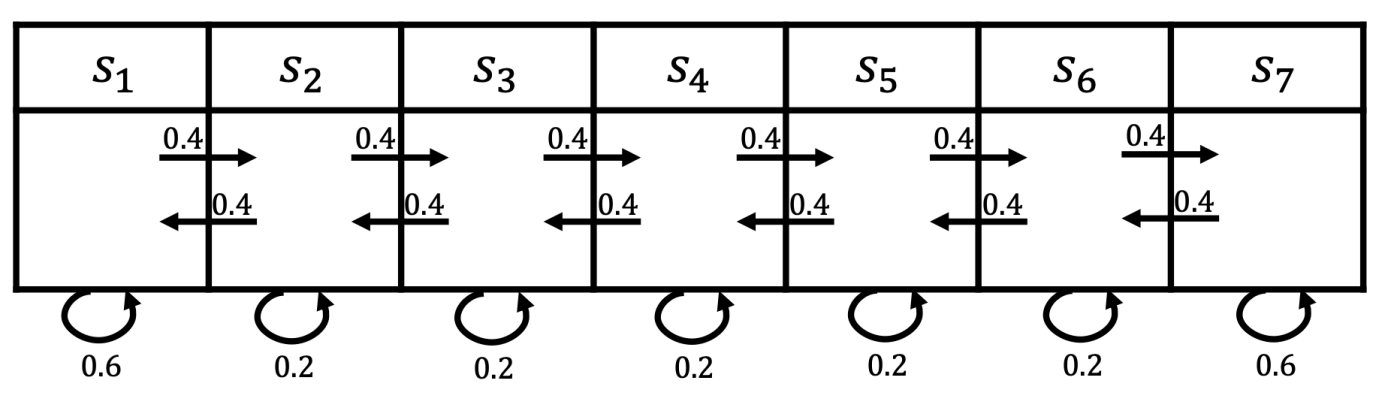
$$P = \begin{bmatrix} P(s_1|s_1) & \cdots & P(s_n|s_1) \\ \vdots & \ddots & \vdots \\ P(s_1|s_n) & \cdots & P(s_n|s_n) \end{bmatrix}$$

Mars Rover – matrice tranziții



$$P = \begin{pmatrix} 0.6 & 0.4 & 0 & 0 & 0 & 0 & 0 \\ 0.4 & 0.2 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0.4 & 0.2 & 0.4 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0.2 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0.2 & 0.4 & 0 \\ 0 & 0 & 0 & 0 & 0.4 & 0.2 & 0.4 \\ 0 & 0 & 0 & 0 & 0 & 0.4 & 0.6 \end{pmatrix}$$

Mars Rover – episoade



Exemple episoade (stare inițială s_4):

- $s_4, s_4, s_4, s_4, s_4, \ldots$
- $s_4, s_5, s_6, s_5, s_4, s_3, s_2, s_1, s_2, \ldots$
- $s_4, s_5, s_6, s_7, s_7, s_7, \ldots$

Să adăugăm recompense: MRP

- **MRP** (**Markov Reward Process**) = **Lanțuri Markov** + **Recompense**
- R reprezintă funcția de acordare a recompenselor:

$$R(s_t = s) = E[r_t |\ s_t = s]$$

- **Atenție! Nu avem acțiuni.**

$$P = \begin{bmatrix} P(s_1|s_1) & \cdots & P(s_n|s_1) \\ \vdots & \ddots & \vdots \\ P(s_1|s_n) & \cdots & P(s_n|s_n) \end{bmatrix}$$

$$R = (r_1, r_2, \dots, r_n)$$

Return & State Value Function

- **ORIZONT (HORIZON)**

1. Reprezintă numărul de momente de timp dintr-un episod.
2. Poate fi ***infinit*** sau ***finit (finite MRP)***

- **RETURN (G_t)** – Suma de recomponse (cu discount), de la momentul t către orizont.
- **STATE VALUE FUNCTION (V(s))** – Return-ul așteptat pornind din starea s.

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots$$

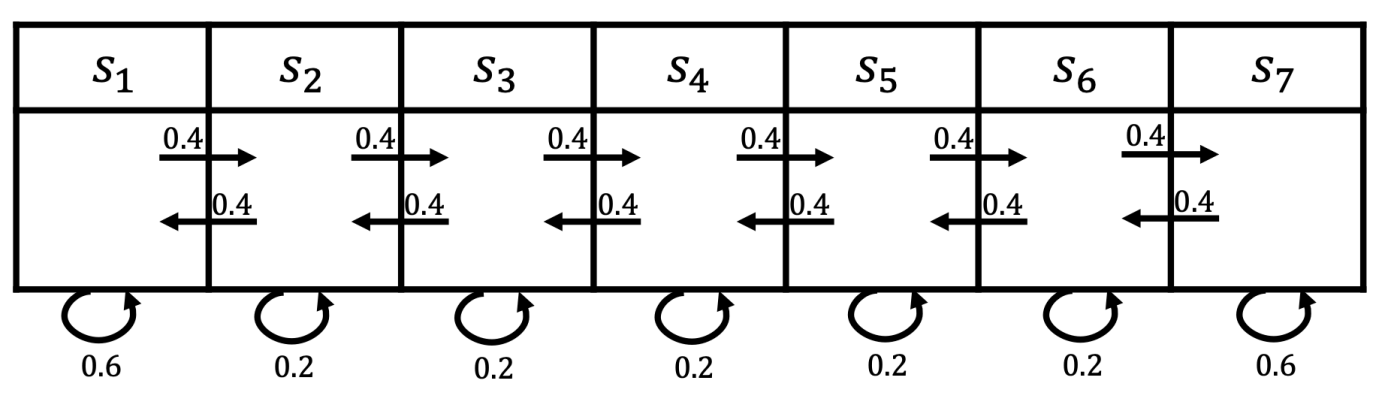
$$\boldsymbol{V(s) = E[G_t | s_t = s]}$$

$$E[G_t | s_t = s] = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots | s_t = s]$$

Observații

- **Tot ce am discutat generează un procedeu matematic convenabil, în condițiile în care evităm cazurile infinite.**
- **În mod natural, oamenii acționează sub un factor mereu mai mic decât 1.**

Mars Rover – Exemplu



Alocare recompense:

- +1 în starea s_1
- +10 în starea s_7
- 0 în orice altă stare

Exemplu (start s_4, $\gamma = \frac{1}{2}$, 4 pași):

- $s_4, s_5, s_6, s_7 \Rightarrow 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{8} * 10 = 1.25$
- $s_4, s_4, s_5, s_4 \Rightarrow 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{8} * 0 = 0$
- $s_4, s_3, s_2, s_1 \Rightarrow 0 + \frac{1}{2} * 0 + \frac{1}{4} * 0 + \frac{1}{8} * 1 = 0.125$

Simulare!!!

Putem evalua algoritmii de tip MRP prin generarea unui set suficient de mare de episoade, astfel încât să satisfacem următoarea ecuație:

$$V(s) = \underbrace{R(s)}_{\text{Immediate reward}} + \underbrace{\gamma \sum_{s' \in S} P(s'|s)V(s')}_{\text{Discounted sum of future rewards}}$$

Formă matriceală – calculul V

$$\begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix} = \begin{pmatrix} R(s_1) \\ \vdots \\ R(s_N) \end{pmatrix} + \gamma \begin{pmatrix} P(s_1|s_1) & \cdots & P(s_N|s_1) \\ P(s_1|s_2) & \cdots & P(s_N|s_2) \\ \vdots & \ddots & \vdots \\ P(s_1|s_N) & \cdots & P(s_N|s_N) \end{pmatrix} \begin{pmatrix} V(s_1) \\ \vdots \\ V(s_N) \end{pmatrix}$$

$$\begin{aligned} V &= R + \gamma PV \\ V - \gamma PV &= R \\ (I - \gamma P)V &= R \\ V &= (I - \gamma P)^{-1} R \end{aligned}$$

Complexitate $O(N^3)$ – datorată calculului inversei

Un calcul mai rapid

- **PROGRAMARE DINAMICĂ!!!**

1. **Inițializăm $V_0(s) = 0, \forall s$**
2. **Pentru k = 1, până la convergență:**
   a. Pentru fiecare s din S:
      - $V_k(s) = R(s) + \gamma \sum_{s' \epsilon S} P(s'|s) V_{k-1}(s')$

- **Complexitate: $O(|S|^2)$, *pentru fiecare iterație* ($|S| = N$).**

Să adăugăm acțiuni: MDP

- **MDP (Markov Decision Process) = MRP + Acțiuni**
- A este un set finit de acțiuni.
- **Discount factor $\gamma \epsilon [0, 1]$.**
- $R(s_t = s, a_t = a) = E[r_t | s_t = s, a_t = a]$

- $\boldsymbol{MDP = (S, A, P, R, \gamma)}$

Matricea P devine tri – dimensională: $\boldsymbol{S\ X\ S\ X\ A}$

Mars Rover – Exemplu

MDP cu 2 acțiuni.

s_1	s_2	s_3	s_4	s_5	s_6	s_7
						

$$P(s'|s,a_1) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad P(s'|s,a_2) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Thanks!

Este timpul pentru întrebări!!!

stefan.iordache10@s.unibuc.ro
catalina.patilea@s.unibuc.ro
ciprian.paduraru@fmi.unibuc.ro

+40 7.. ... ...

Introducere în Reinforcement Learning

Cursul #4

Ștefan Iordache, Cătălina Iordache, Ciprian Păduraru

Cuprins

01 Recapitulare

Repetiția: mama învățăturii!

02 Politici MDP

03 Control MDP

04 În căutarea politicii

01

Recapitulare

Repetiția: mama învățăturii!

Recapitulare

- Procese Markov
- Lanțuri Markov
- Mars Rover – matrice tranziții, episoade
- MRP
- MDP

02

Politici MDP

Politici MDP

- **Politica** specifică acțiunea care trebuie luată în fiecare stare în parte.

→ Poate fi **deterministă** sau **stochastică**.

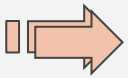
- Pentru generalitate, se consideră a fi o ***distribuție conditionată***.

→ Fiind dată **o stare S, se specifică distribuția peste acțiuni.**

- **Politica:**

→ $\boldsymbol{\pi(a|s) = P(a_t = a \,|s_t = s)}$

MDP + Politica

- **MDP + $\boldsymbol{\pi(a|s)}$**  **Markov Reward Process (MRP)**
- **MRP** $(\mathbf{S}, \boldsymbol{R^{\pi}}, \boldsymbol{P^{\pi}}, \boldsymbol{\gamma})$
  - $\boldsymbol{R^{\pi}} = \sum_{a \in A} \boldsymbol{\pi(a|s)} * \mathrm{R(s, a)}$
  - $\boldsymbol{P^{\pi}}\mathrm{(s'|s)} = \sum_{a \in A} \boldsymbol{\pi(a|s)} * \mathrm{P(s'|s , a)}$

PS: Totuși, putem folosi aceleași tehnici a evalua o politică pentru un MDP (algoritmul de programare dinamică).

MDP – Evaluarea politcii

- Algoritm Iterativ -

1. **Inițializăm $V_0(s) = 0, \forall s$**
2. **Pentru k = 1, până la convergență:**
   a. Pentru fiecare s din S:
      - $V_k^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s' \epsilon S} P(s'|\pi(s)) V_{k-1}^{\pi}(s')$

- **“Bellman Backup” – se aplică unei politici particulare.**

? Întrebare rapidă ?

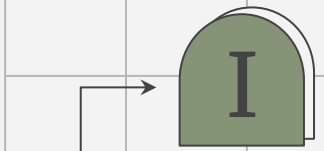
MDP – Evaluarea politicii

Care este diferența față de algoritmul anterior?

? Întrebare rapidă ?

MDP – Evaluarea politicii

Care este diferența față de algoritmul anterior?



Răspuns:

- **De această dată, se evaluează V sub o politică π.**

Exemplu MDP – Iterație a politicii de evaluare - Mars Rover

- **Dinamica:**
  - $p(s_6 \mid s_6, a_1) = 0.5$
  - $p(s_7 \mid s_7, a_1) = 0.5$, ...
- **Reward:**
  - +1, în starea s_1
  - +10, în starea s_7
  - 0, altfel
- **Fie: $\pi(s) = a_1$, $\forall s$, presupunem că $V_k = [1\ 0\ 0\ 0\ 0\ 0\ 10]$, $k = 1$, $\gamma = 0.5$**

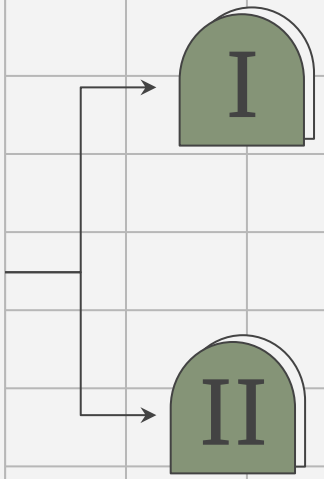
Exemplu MDP – Iterație a politicii de evaluare - Mars Rover – continuare

- **Fie: $\pi(s) = a_1$, $\forall s$, presupunem că $V_k = [1\ 0\ 0\ 0\ 0\ 0\ 1\ 0]$, $k = 1$, $\gamma = 0.5$**
  - Pentru fiecare s din S:
    - $V_k^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s' \epsilon S} P(s'|\pi(s)) V_{k-1}^{\pi}(s')$
  - $V_{k+1}(s_6) = r(s_6, a_1) + \gamma * 0.5 * V_k(s_6) + \gamma * 0.5 * V_k(s_7)$
  - $V_{k+1}(s_6) = 0 + 0.5 * 0.5 * 0 + \mathbf{0.5} * 0.5 * 10$
  - $V_{k+1}(s_6) = 2.5$

? Întrebări și mai rapide ?

Ipoteze:

- 7 stări discrete (locații ale roverului)
- 2 acțiuni (stânga/dreapta)



Q1: Câte politici deterministe există?

Q2: Politica optimă pentru un MDP este unică?

? Întrebare rapidă ?

Q1: Câte politici deterministe există?

2^7

Q2: Politica optimă pentru un MDP este unică?

Nu! pot exista 2 acțiuni cu aceeași valoare optimă (conformvalue function)

03

Control MDP

Politica optimă

$$\pi^*(s) = arg\ max_\pi V^\pi(s)$$

Despre controlul MDP

- **Există o singură funcție optimă de tip value function, dar multiple politici cu aceeași valoare optimă!**
- **Politica optimă pentru MDP cu orizont infinit:**
  - Este deterministă;
  - Este staționară;
  - Nu este mereu unică!

Brute-force

- Putem utiliza o metodă de tip brute-force pentru căutarea politicii optime.
- Complexitate: $|A|^{|S|}$

Policy Iteration

1. **Setăm i = 0**
2. **Inițializăm $\pi_0(s)$ aleatorie pentru fiecare stare s.**
3. **Iterăm cât timp $||\pi_i - \pi_{i-1}|| > 0$ $(L1 - norm)$:**
   a. $V^{\pi_i} \leftarrow MDP\ value\ function\ (evaluarea\ politicii\ \pi_i)$
   b. $\pi_{i+1} \leftarrow Îmbunătățirea\ politicii$
   c. $i \leftarrow i + 1$

04

În căutarea politicii

State-Action Value -> Q

- **Avansăm cu pași repezi și ajungem la ceea ce vom numi drept „State-Action Value” (Q).**

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi}(s')$$

- **Traducere: În starea s, executăm acțiunea a, apoi urmăm politica π.**

Cum actualizăm? Policy Iteration

- **Calculăm Q pentru o politică π_i pentru fiecare stare s din S și fiecare acțiune a din A.**

$$Q^{\pi_i}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi_i}(s')$$

- **Generăm politica nouă π_{i+1} pentru fiecare stare s din S.**

$$\pi_{i+1}(s) = \arg\max_a Q^{\pi_i}(s,a)$$

Aprofundare!

- **Îmbunătățirea politicii are loc la fiecare iterație!**

$$Q^{\pi_i}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi_i}(s')$$

$$\max_a Q^{\pi_i}(s,a) \geq R(s,\pi_i(s)) + \gamma \sum_{s' \in S} P(s'|s,\pi_i(s)) V^{\pi_i}(s') = V^{\pi_i}(s)$$

$$\pi_{i+1}(s) = \arg\max_a Q^{\pi_i}(s,a)$$

Cum demonstrăm algoritmul?

- **Presupunere:**
  - $V^{\pi_{i+1}} \geq V^{\pi_i}$

  unde π_i *nu este optimă, iar* π_{i+1} *este noua politică obținută prin îmbunătățirea* π_i

$$
\begin{aligned}
V^{\pi_i}(s) &\leq \max_a Q^{\pi_i}(s,a) \\
&= \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi_i}(s') \\
&= R(s,\pi_{i+1}(s)) + \gamma \sum_{s' \in S} P(s'|s,\pi_{i+1}(s)) V^{\pi_i}(s') \ \text{//by the definition of } \pi_{i+1} \\
&\leq R(s,\pi_{i+1}(s)) + \gamma \sum_{s' \in S} P(s'|s,\pi_{i+1}(s)) \left(\max_{a'} Q^{\pi_i}(s',a') \right) \\
&= R(s,\pi_{i+1}(s)) + \gamma \sum_{s' \in S} P(s'|s,\pi_{i+1}(s)) \\
&\quad \left(R(s',\pi_{i+1}(s')) + \gamma \sum_{s'' \in S} P(s''|s',\pi_{i+1}(s')) V^{\pi_i}(s'') \right) \\
&\quad \vdots \\
&= V^{\pi_{i+1}}(s)
\end{aligned}
$$

Ce am aflat?

- **Policy Iteration** ne ajută să ajungem atât la valori optime, cât și la politici asociate!

Value Iteration

$$V^{\pi}(s) = R^{\pi}(s) + \gamma \sum_{s' \in S} P^{\pi}(s'|s) V^{\pi}(s')$$

- **Introducem un nou concept: „Bellman Backup Operator”**
  - Se aplică asupra functiei V.
  - Ne returnează o nouă funcție V.
  - Îmbunătățește valoarea dacă este posibil acest lucru.

$$BV(s) = \max_{a} R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V(s')$$

Aprofundare

- **Cum aplicăm operațiunile de tip Bellman asupra unei politici?**

$$B^{\pi}V(s) = R^{\pi}(s) + \gamma \sum_{s' \in S} P^{\pi}(s'|s)V(s)$$

- **Cum evaluăm o politică? Repetat!, până când valoarea V nu se mai schimbă și nu observăm creșteri sau scăderi.**

$$V^{\pi} = B^{\pi}B^{\pi} \cdots B^{\pi}V$$

Value Iteration

- Algoritm Iterativ -

1. **Setăm k = 1**
2. **Inițializăm $V_0(s) = 0, \forall s$**
3. **Repetăm până la convergență:**
   a. Pentru fiecare s din S:

$$V_{k+1}(s) = \max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')$$

Avem condiții de convergență?

- ***Da! Dacă factorul de discount este mai mic strict decât 1!*** $\gamma < 1$

Thanks!

Este timpul pentru întrebări!!!

stefan.iordache10@s.unibuc.ro
catalina.patilea@s.unibuc.ro
ciprian.paduraru@fmi.unibuc.ro

+40 7.. ... ...

Introducere în Reinforcement Learning

Cursul #5

Ștefan Iordache, Cătălina Iordache, Ciprian Păduraru

Cuprins

01 Recapitulare

Repetiția: mama învățăturii!

02 Monte-Carlo

03 Temporal-Difference

04 TD(λ)

01

Recapitulare

Repetiția: mama învățăturii!

Recapitulare

- Politici MDP
- Control MDP
- În căutarea politicii

02

Monte Carlo

Monte-Carlo în Reinforcement Learning

- 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***

Evaluarea policitii – Monte Carlo

- **Scop**: învățarea v_π din episoade sub o politică π.
- **Return**
- **Value function**
- **Policy evaluation**: se folosește *empirical mean return* în loc de *expected return*.

- $S_1, A_1, R_2, \dots, S_k \sim \pi$
- $G_t = R_{t+1} + \gamma * R_{t+2} + \dots + \gamma^{T-1} * R_T$
- $v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$

First-Visit Monte Carlo

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

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

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

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

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

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

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Every-visit Monte-Carlo Policy Evaluation

Pentru fiecare moment de timp t în care starea s este vizitată într-un episod:

- Counter: $N(s) \leftarrow N(s) + 1$
- Incrementăm return-ul total, sub formă de sumă:
  $S(s) \leftarrow S(s) + G_t$

- Valoarea este estimată în funcție de *"**răspunsul mediu**"* *(mean return):*

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

$V(s) \rightarrow v_\pi(s)$, **deoarece** $N(s) \rightarrow \infty$

Bias, varianță & MSE (Mean Squared Error)

- Se consideră un model statistic care este parametrizat de $\boldsymbol{\theta}$ și determină o probabilitate de distribuție peste datele observate $P(x|\boldsymbol{\theta})$
- Se consideră statistica $\widehat{\boldsymbol{\theta}}$, care oferă o estimare a lui $\boldsymbol{\theta}$ și este o funcție a datelor observate.

Bias, Varianță și MSE

Bias-ul unui estimator $\widehat{\boldsymbol{\theta}}$:
$$\boldsymbol{Bias_{\theta}}(\widehat{\boldsymbol{\theta}}) = \mathbb{E}_{\boldsymbol{x}|\boldsymbol{\theta}}[\widehat{\boldsymbol{\theta}}] - \boldsymbol{\theta}$$

Varianța unui estimator $\widehat{\boldsymbol{\theta}}$:
$$\mathbf{Var}(\widehat{\boldsymbol{\theta}}) = \mathbb{E}_{\boldsymbol{x}|\boldsymbol{\theta}}[(\widehat{\boldsymbol{\theta}} - \mathbb{E}[\widehat{\boldsymbol{\theta}}])^2]$$

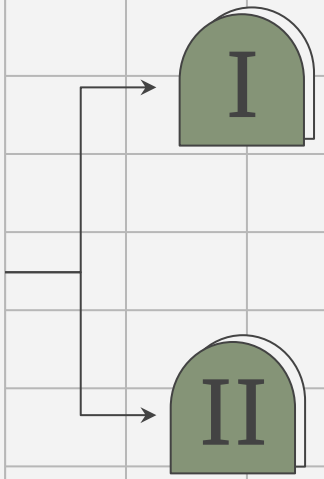
MSE pentru un estimator $\widehat{\boldsymbol{\theta}}$:
$$\boldsymbol{MSE}(\widehat{\boldsymbol{\theta}}) = \mathbf{Var}(\widehat{\boldsymbol{\theta}}) + \boldsymbol{Bias_{\theta}}(\widehat{\boldsymbol{\theta}})^2$$

First-Visit Monte Carlo on Policy Evaluation

- Inițializare N(s) = 0; G(s) = 0; oricare ar fi s ∈ S
- Pentru:
  - Extragem un episod i = $s_{i,1}$, $a_{i,1}$, $r_{i,1}$, $s_{i,2}$, $a_{i,2}$, $r_{i,2}$, ……. $s_{i,T}$
  - Definim $G_{i,t} = r_{i,t} + \gamma * r_{i,t+1} + \gamma^2 * r_{i,t+2} + \ldots + \gamma^{T_i-1} * r_{i,T_i}$ ca return începând cu pasul t, în cadrul episodului selectat i.
  - Pentru fiecare stare s vizitată într-un episod i:
    - Pentru **primul** timp t în care o stare s este vizitată într-un episod i:
      - N(s) = N(s) +1 (incrementare counter pentru toate vizitele)
      - G(s) = G(s) + $G_{i,t}$ (incrementare return total)
      - V^{π}(s) = G(s) / N(s) (actualizăm estimarea)

First-Visit Monte Carlo on Policy Evaluation

Proprietăți



Estimatorul V^{π} este un estimator 'unbiased'

$\mathbb{E}_{\pi}[G_t \mid s_t = s]$

Din teorema numerelor mari, N(s) -> ∞

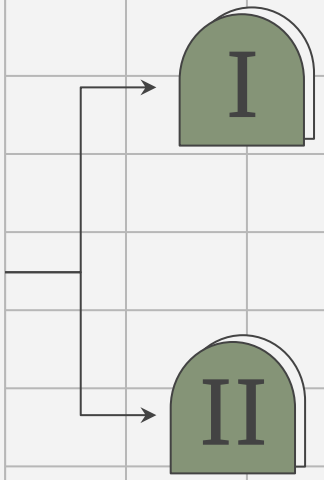
V^{π} -> $\mathbb{E}_{\pi}[G_t \mid s_t = s]$

Every-Visit Monte Carlo on Policy Evaluation

- Inițializare N(s) = 0; G(s) = 0; oricare ar fi s ∈ S
- Pentru:
  - Extragem un episod i = $s_{i,1}$, $a_{i,1}$, $r_{i,1}$, $s_{i,2}$, $a_{i,2}$, $r_{i,2}$, ……. $s_{i,T}$
  - Definim $G_{i,t} = r_{i,t} + \gamma * r_{i,t+1} + \gamma^2 * r_{i,t+2} + \ldots + \gamma^{T_i-1} * r_{i,T_i}$ ca return începând cu pasul t, în cadrul episodului selectat i.
  - Pentru fiecare stare s vizitata într-un episod i:
    - Pentru **fiecare** timp t în care o stare s este vizitata intr-un episod i:
      - N(s) = N(s) +1 (incrementare counter pentru toate vizitele)
      - G(s) = G(s) + $G_{i,t}$ (incrementare return total)
      - V^{π}(s) = G(s) / N(s) (actualizăm estimarea)

Every-Visit Monte Carlo on Policy Evaluation

Proprietăți

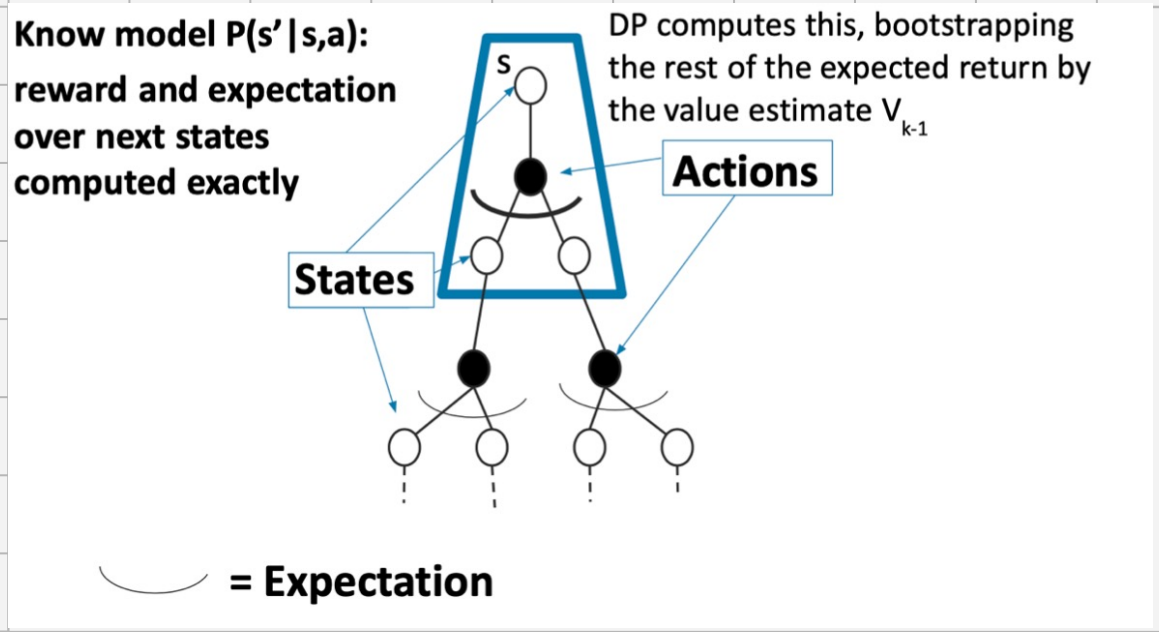


Estimatorul V^{π} **"every-visit" MC este un estimator 'biased' al lui** V^{π}

Dar!!! În practică este consistent și obține des o valoare MSE mai bună!

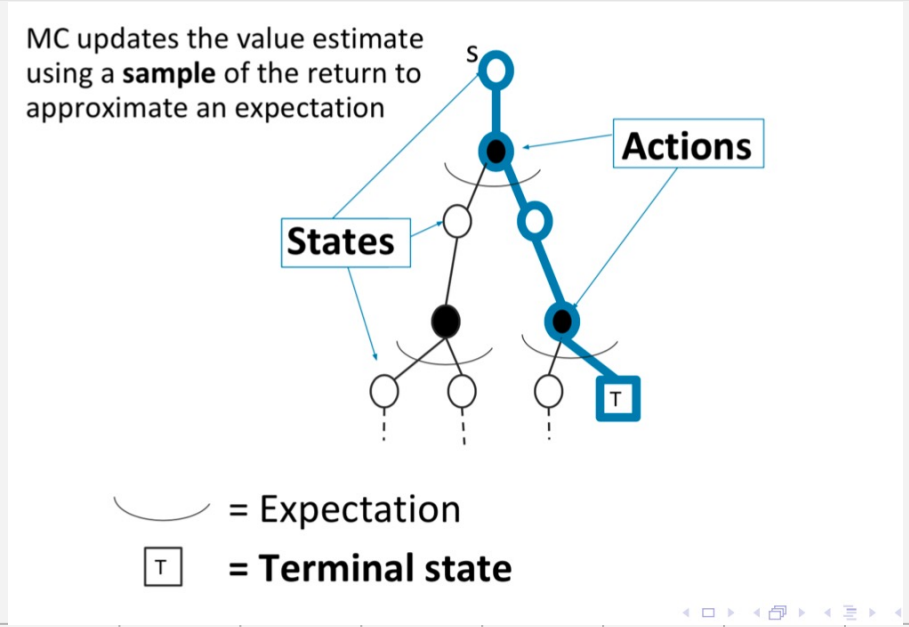
Evaluarea politicii cu programare dinamică

$$V^{\pi}(s) \leftarrow \mathbb{E}_{\pi}[r_t + \gamma * V_{k-1} | s_t = s]$$



Bootstrapping: Update-ul pentru V folosește o estimare.

Evaluarea politicii Monte Carlo



- **Metoda MC actualizează valoarea estimată folosind o probă (sample) a recompenselor pentru a aproxima ce se va întâmpla.**

Media incrementala

Media $\mu_1, \mu_2, \ldots$ *a unei secvențe* $x_1, x_2, \ldots$ poate fi calculată în mod incremental:

$$
\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)
\end{aligned}
$$

Actualizări incrementale pentru Monte-Carlo

- Considerăm episodul: S_1, A_1, R_2, ..., S_T. Actualizăm incremental V(s) după acest episod.
- Pentru fiecare stare S_t cu return-ul G_t:
  - $N(St) \leftarrow N(S_t) + 1$
  - $V(St) \leftarrow V(St) + (G_t - V(St)) * \frac{1}{N(St)}$
- În problemele non-staționare, poate fi util să urmăm o **medie ajustabilă** (**running**), adică să uităm de vechile episoade:
  - $V(S_t) <- V(S_t) + \propto (G_t - V(St))$

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

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

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

Algoritm TD

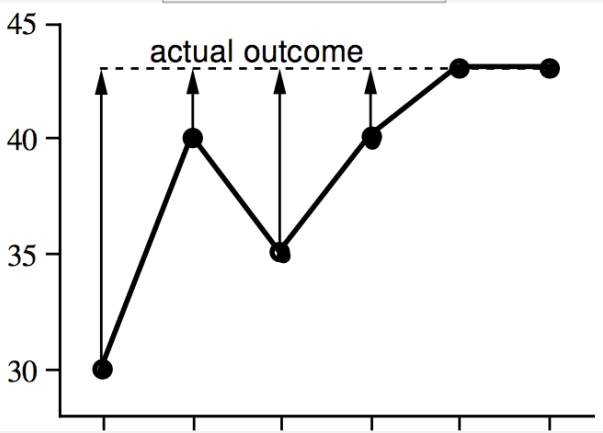
- Input: politica π ce urmează să fie evaluată
- Parametri: mărimea pasului $\alpha \in (0, 1]$
- Inițializăm V(s) arbitrat, pentru toate stările, exceptând V(terminal) = 0
- Iterăm pentru fiecare episod:
  - Inițializăm S
  - Iterăm pentru fiecare pas din episod, până la starea terminală:
    - $A \leftarrow$ *acțiunea dată de politica* π *pentru* S
    - Executăm acțiunea A, observăm R, S'
    - $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
    - $S \leftarrow S'$

Exemplu TD

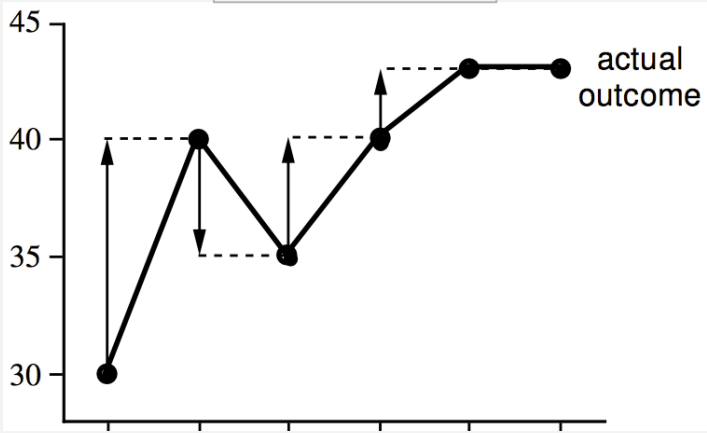
Stare	Timp	Timp prezis (pentru plecare)	Timp total prezis
Plecare din birou	0	30	30
Ajungem la mașină, plouă	5	35	40
Ieșire autostradă	20	15	35
Vehicul încet în față	30	10	40
Stradă rezidențială	40	3	43
Am ajuns acasă	43	0	43

Exemplu TD vs. MC

MC



TD



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.

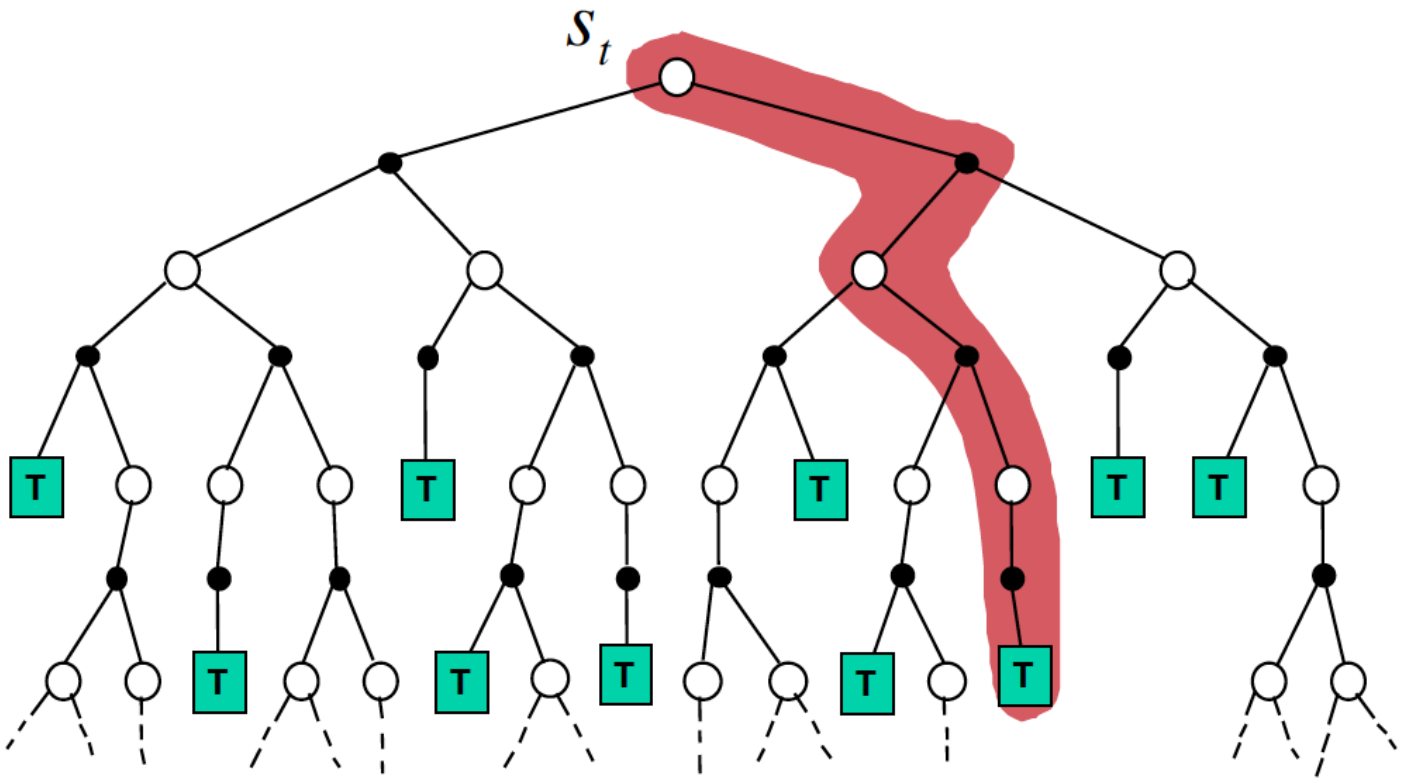
MC

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

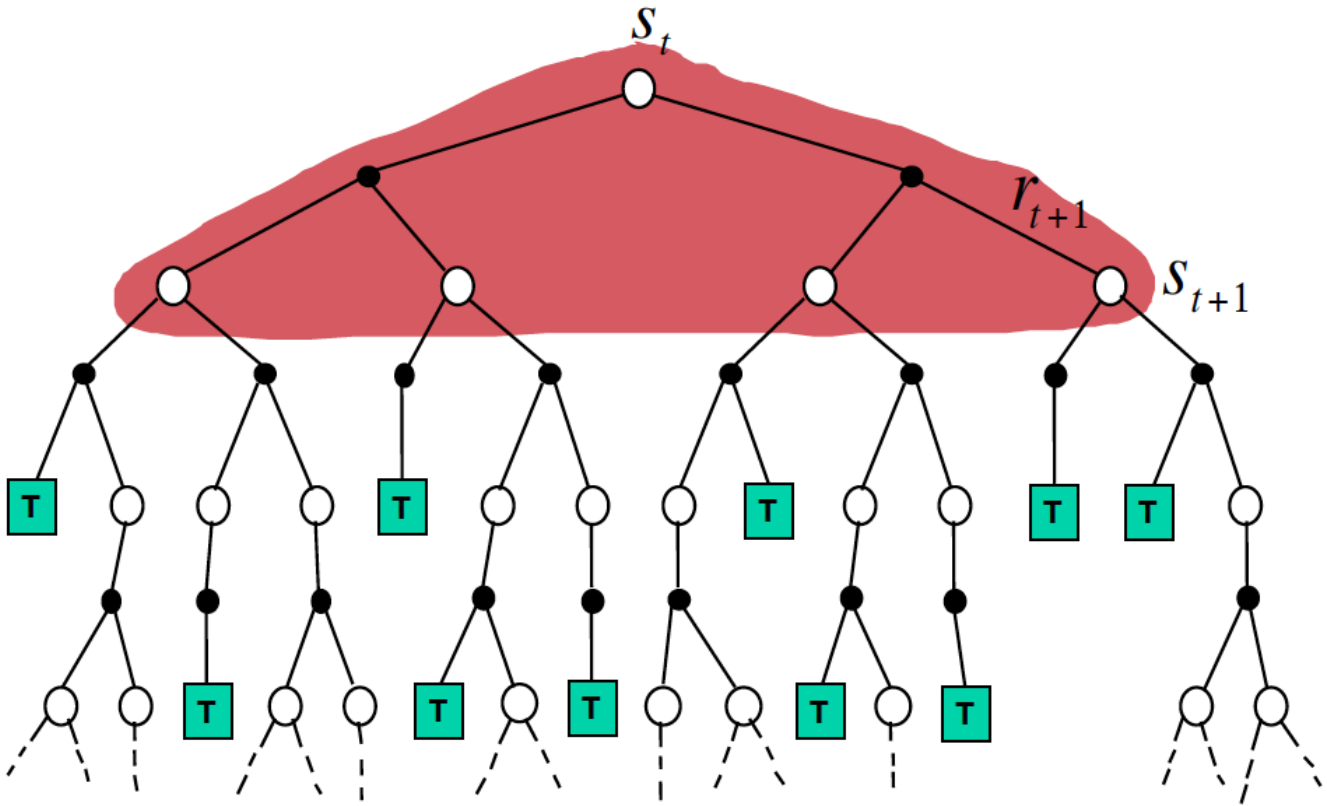
TD

- Mai eficient fată de MC!
- Sensibil la punctul de plecare (valoare inițială)!
- Convergență pentru TD(0)!

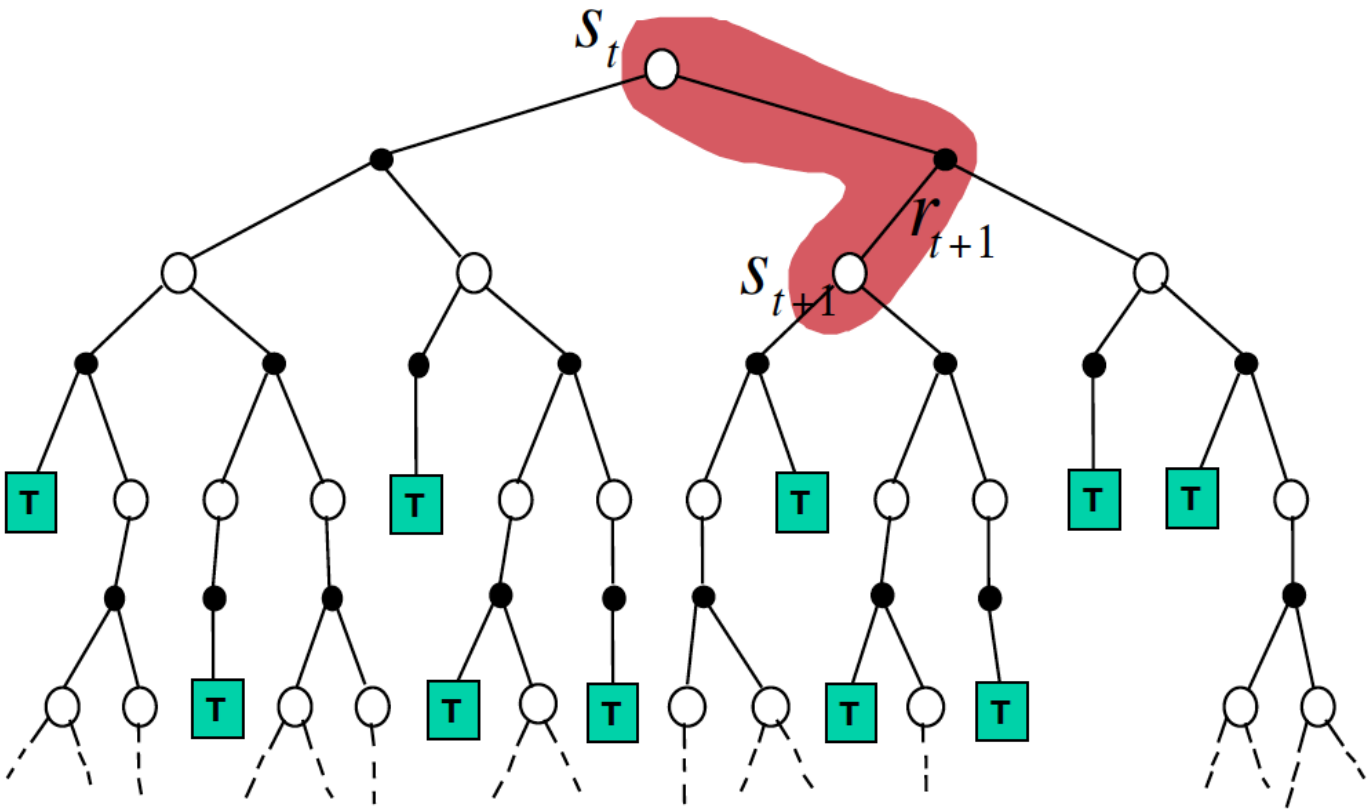
$$V(S_t) \leftarrow V(S_t) + \alpha\,(G_t - V(S_t))$$



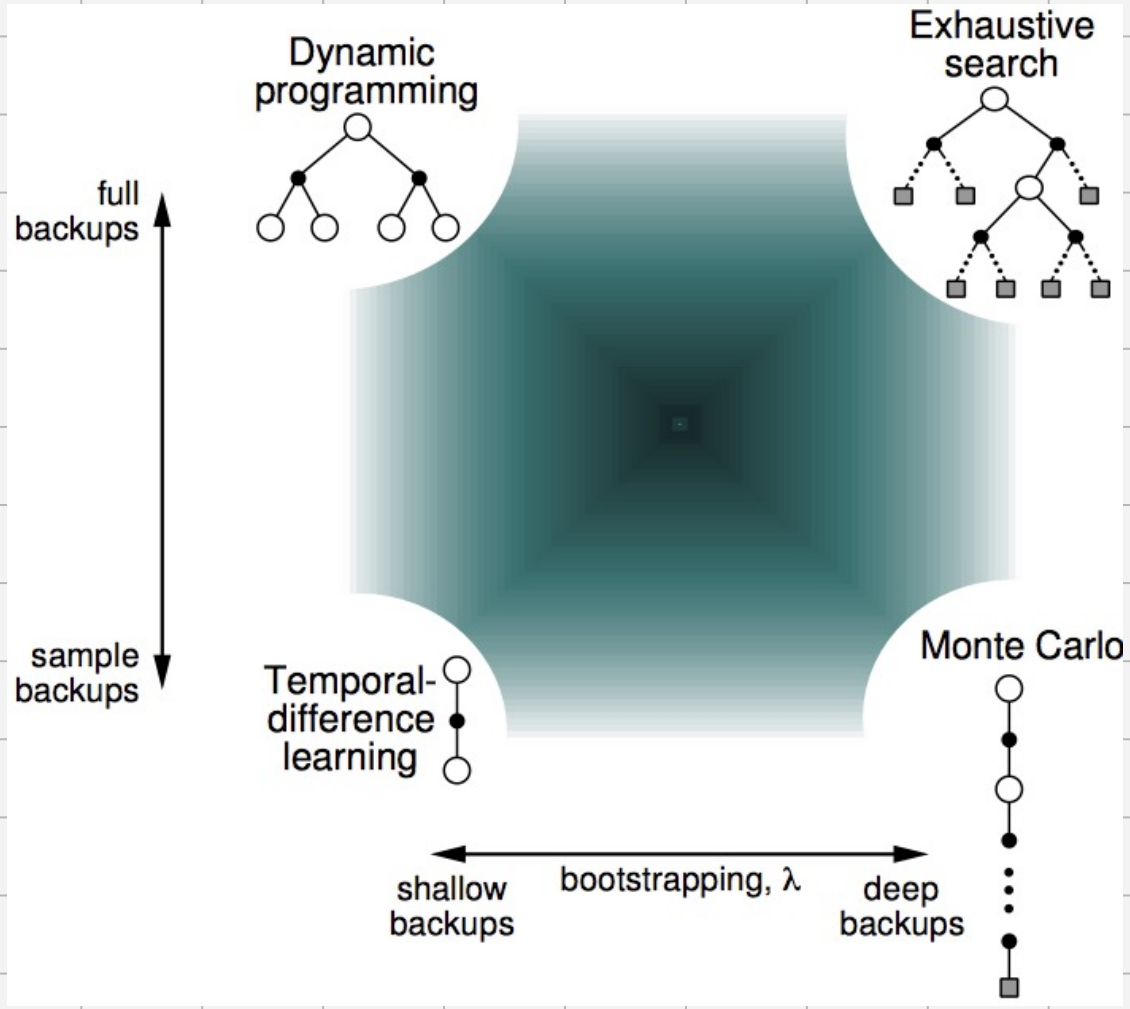
$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1}) \right]$$



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

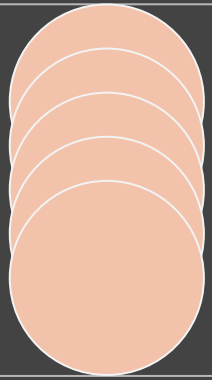


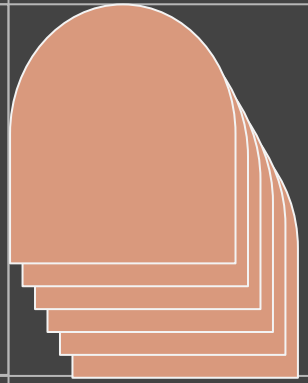
Marea pictogramă a RL-ului!


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

04

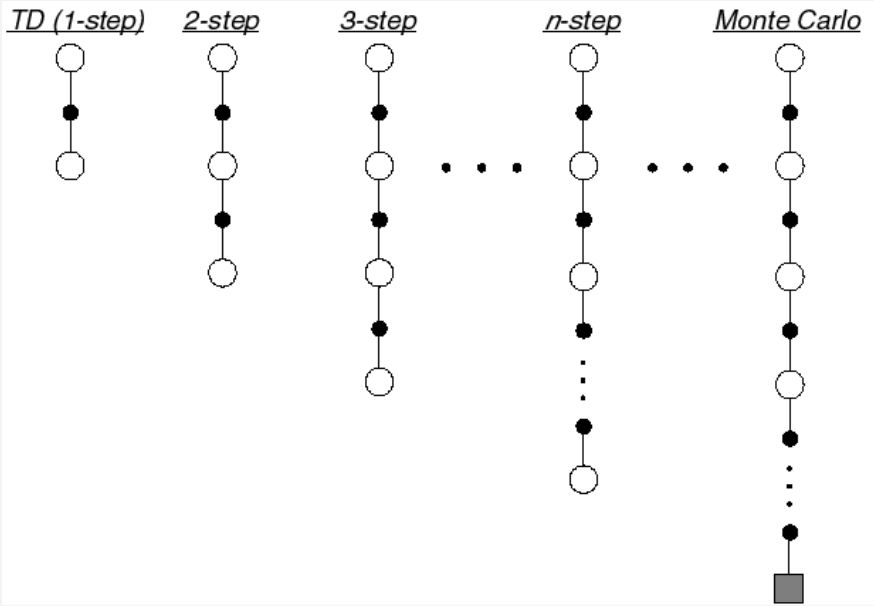
TD(λ)





”n-Step Prediction” – $TD(\lambda)$

- **Considerăm posibilitatea de boostrapping pentru n pași în viitor, aplicație pentru algoritmul TD.**



”n-Step Return” – $TD(\lambda)$

- “n-Step Return” pentru n = 1, 2, ...

$$
\begin{aligned}
n = 1 \quad (TD) \quad & G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1}) \\
n = 2 \quad & G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2}) \\
\vdots \quad & \vdots \\
n = \infty \quad (MC) \quad & G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T
\end{aligned}
$$

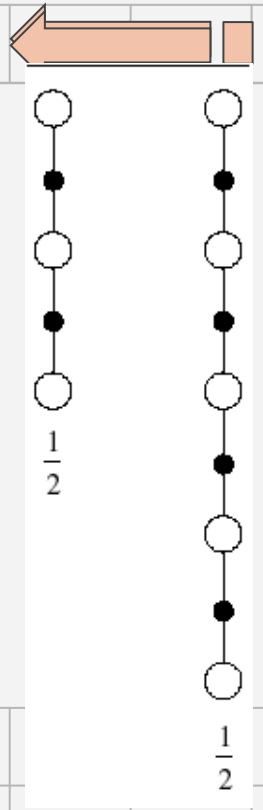
$$
G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})
$$

”n-Step Temporal Difference Learning” – $TD(\lambda)$

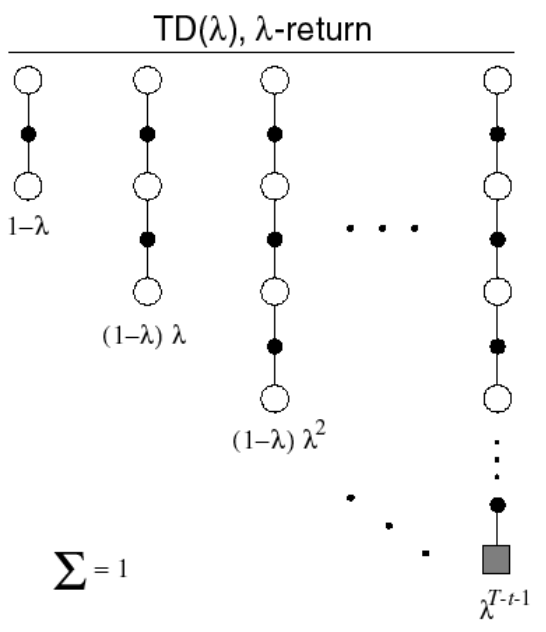
$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$$

”n-Step Returns” – aplicarea mediei

- **Da, putem realiza o medie între return-uri, pentru valori diferite ale lui n!**
- **Exemplu:** $\frac{1}{2}G^{(2)} + \frac{1}{2}G^{(4)}$



λ-return



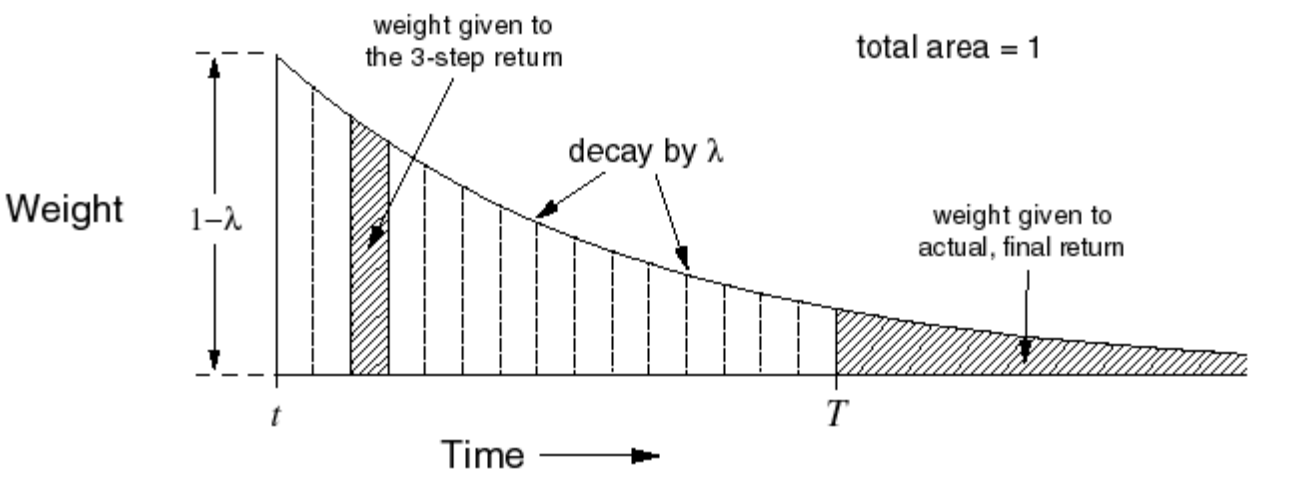
- **λ-return** este definit sub forma G_t^λ și constă în combinarea n-step return-urilor $G_t^{(n)}$.
- Folosim pentru calcul $(1-\lambda)\lambda^{n-1}$, astfel:

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

- Mai departe, definim $forward-view\ TD(\lambda)$:

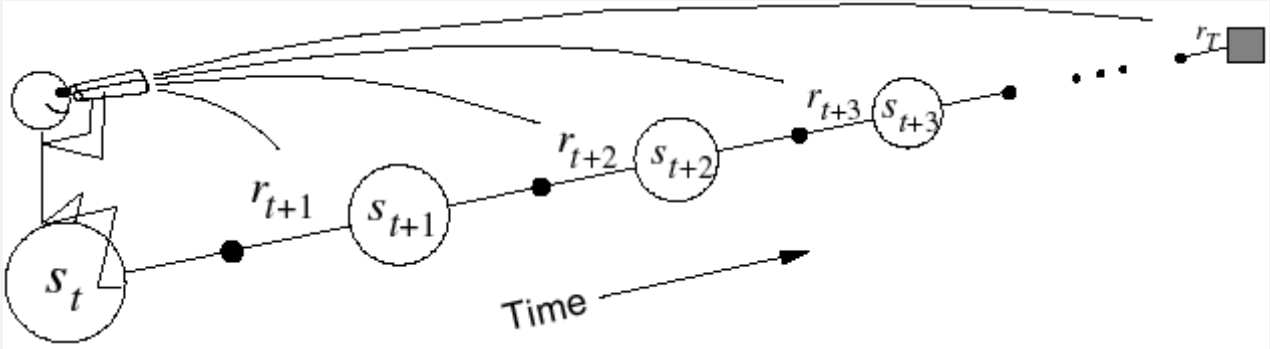
$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t^\lambda - V(S_t)\right)$$

λ-return



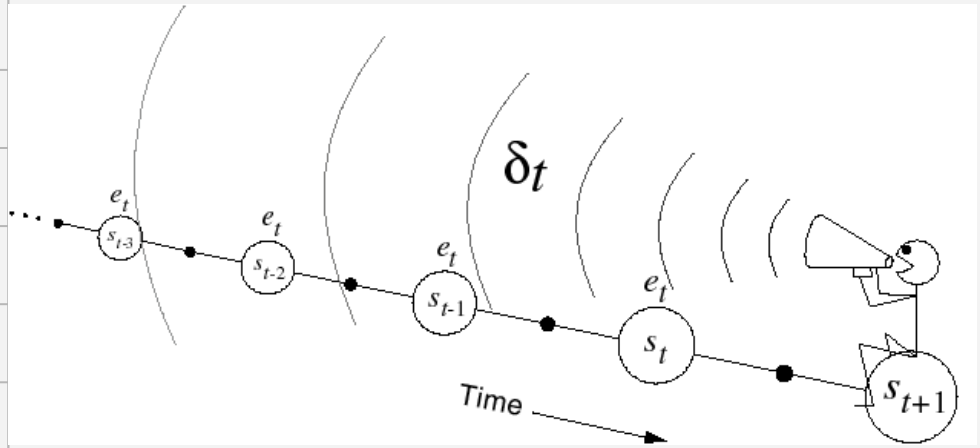
$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

Forward View – $TD(\lambda)$



- ESTE O FORMĂ TEORETICĂ A CEEA CE SE VA ÎNTÂMPLA!
- Se aseamănă cu Monte Carlo, datorită calculării folosind episoade complete!

Backward View – $TD(\lambda)$



- REPREZINTĂ MECANISMUL NECESAR ÎNVĂȚĂRII!

- **Eligibility trace!!! $E_t(s)$**

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$
$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$

$TD(\lambda) - TD(0) - \mathrm{TD}(1)$

- Dacă $\lambda = 0$, atunci doar starea curentă primește actualizări!
- Echivalent??? TD(0)

$$E_t(s) = \mathbf{1}(S_t = s)$$
$$V(s) \leftarrow V(s) + \alpha\delta_t E_t(s)$$

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

- Dacă $\lambda = 1$, recompensele vor fi migrate către finalul episodului! Numărul de actualizări în acest caz este egal cu cel realizat de MC (every-visit).

Teoremă!

Suma actualizărilor offline este identică pentru forward-view și back-ward $TD(\lambda)$.

$$\sum_{t=1}^{T} \alpha \delta_t E_t(s) = \sum_{t=1}^{T} \alpha \left(G_t^{\lambda} - V(S_t) \right) \mathbf{1}(S_t = s)$$

λ diferit de 1?

$$
\begin{aligned}
G_t^\lambda - V(S_t) &= -V(S_t) \quad + \quad (1-\lambda)\lambda^0 \left(R_{t+1} + \gamma V(S_{t+1})\right) \\
&\quad + \quad (1-\lambda)\lambda^1 \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})\right) \\
&\quad + \quad (1-\lambda)\lambda^2 \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 V(S_{t+3})\right) \\
&\quad + \quad \ldots \\
&= -V(S_t) \quad + \quad (\gamma\lambda)^0 \left(R_{t+1} + \gamma V(S_{t+1}) - \gamma\lambda V(S_{t+1})\right) \\
&\quad + \quad (\gamma\lambda)^1 \left(R_{t+2} + \gamma V(S_{t+2}) - \gamma\lambda V(S_{t+2})\right) \\
&\quad + \quad (\gamma\lambda)^2 \left(R_{t+3} + \gamma V(S_{t+3}) - \gamma\lambda V(S_{t+3})\right) \\
&\quad + \quad \ldots \\
&= \quad (\gamma\lambda)^0 \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right) \\
&\quad + \quad (\gamma\lambda)^1 \left(R_{t+2} + \gamma V(S_{t+2}) - V(S_{t+1})\right) \\
&\quad + \quad (\gamma\lambda)^2 \left(R_{t+3} + \gamma V(S_{t+3}) - V(S_{t+2})\right) \\
&\quad + \quad \ldots \\
&= \delta_t + \gamma\lambda\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \ldots
\end{aligned}
$$

λ egal cu 1? → $Every - Visit\ Monte\ Carlo$

$$
\begin{aligned}
&\delta_t + \gamma\delta_{t+1} + \gamma^2\delta_{t+2} + \ldots + \gamma^{T-1-t}\delta_{T-1} \\
&= R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \\
&+ \gamma R_{t+2} + \gamma^2 V(S_{t+2}) - \gamma V(S_{t+1}) \\
&+ \gamma^2 R_{t+3} + \gamma^3 V(S_{t+3}) - \gamma^2 V(S_{t+2}) \\
&\quad\vdots \\
&+ \gamma^{T-1-t} R_T + \gamma^{T-t} V(S_T) - \gamma^{T-1-t} V(S_{T-1}) \\
&= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots + \gamma^{T-1-t} R_T - V(S_t) \\
&= G_t - V(S_t)
\end{aligned}
$$

Thanks!

Este timpul pentru întrebări!!!

stefan.iordache10@s.unibuc.ro
catalina.patilea@s.unibuc.ro
ciprian.paduraru@fmi.unibuc.ro

+40 7.. ... ...

Introducere în Reinforcement Learning

Cursul #6

Ștefan Iordache, Cătălina Iordache, Ciprian Păduraru

Cuprins

01 On-Policy MC

02 On-Policy TD

03 Off-Policy Learning

On-Policy vs Off-Policy

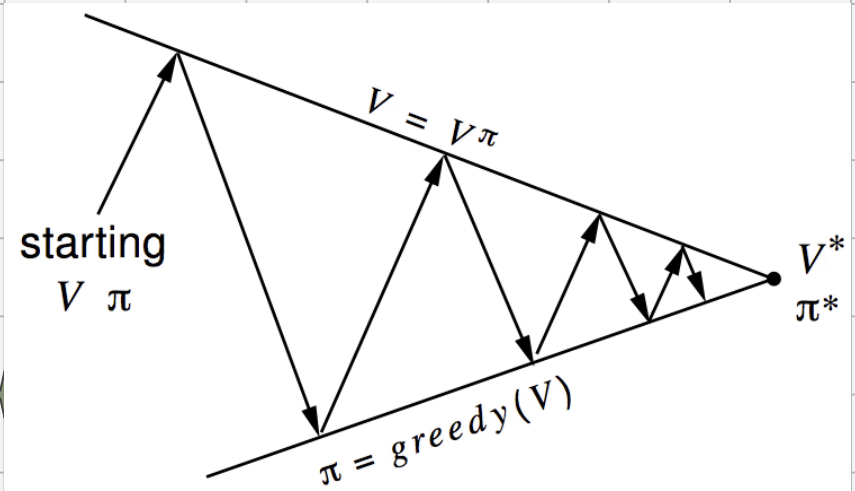
- **On-Policy**
  - ”Calificare la locul de muncă”
  - Învățăm politica π din experiențe extrase cu ajutorul aceleiași politici.
- **Off-Policy**
  - ”Învățăm uitându-ne la altcineva”
  - Învățăm politica π din experiențe extrase cu altă politică (μ).

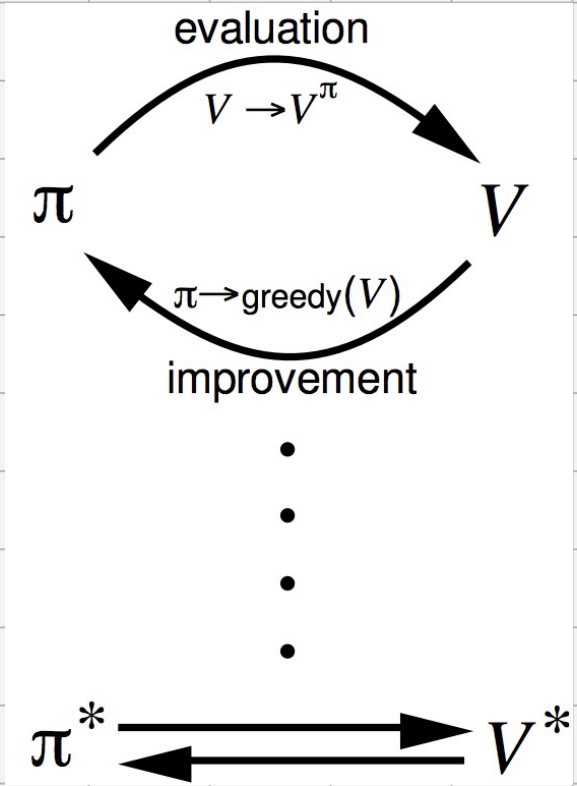
01

On-Policy MC

Generalizare!

- **Evaluarea politicii:** Estimarea V_π
- **Îmbunătățire politicii:** Generarea $\pi' \geq \pi$





Îmbunătățiri: V(s) vs. Q(s, a)

Îmbunătățirea politicii cu ajutorul V(s)

- **Necesită existența modelului din spatele MDP-ului (procesul decizional Markov)**

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s')$$

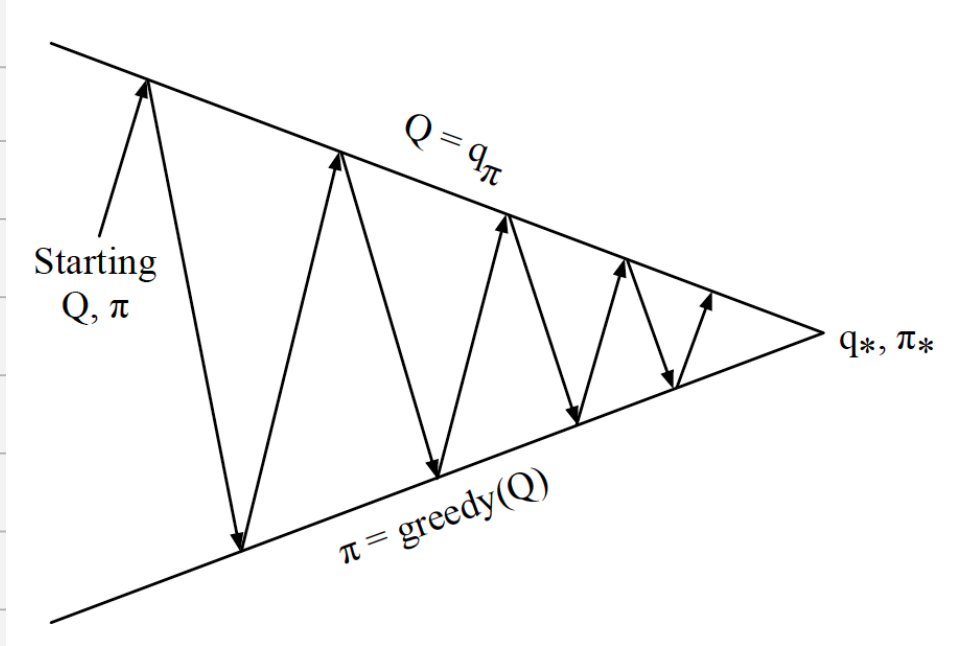
Îmbunătățirea politicii cu ajutorul Q(s, a)

- **Este model-free!**

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q(s, a)$$

Alegem ”Action-Value Function”

- **Evaluarea politicii:** **Estimarea Q = q_π**



Dar cum alegem acțiunile?

GREEDY!!!!

Exemplu: Avem două cutii (stânga și dreapta) din care extragem obiecte. Se execută următoarele acțiuni conform strategiei greedy:

- Deschidem cutia din dreapta => reward 0 = V(dreapta) = 0
- Deschidem cutia din stânga => reward +1 = V(stânga) = +1
- Deschidem cutia din stânga => reward +3 = V(stânga) = +3
- Deschidem cutia din stânga => reward +2 = V(stânga) = +2

Suntem siguri că alegem cea mai bună cutie???

Soluția pentru Greedy? ϵ-Greedy

ϵ-Greedy -> Explorare!!!

- 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.

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

Teoremă!

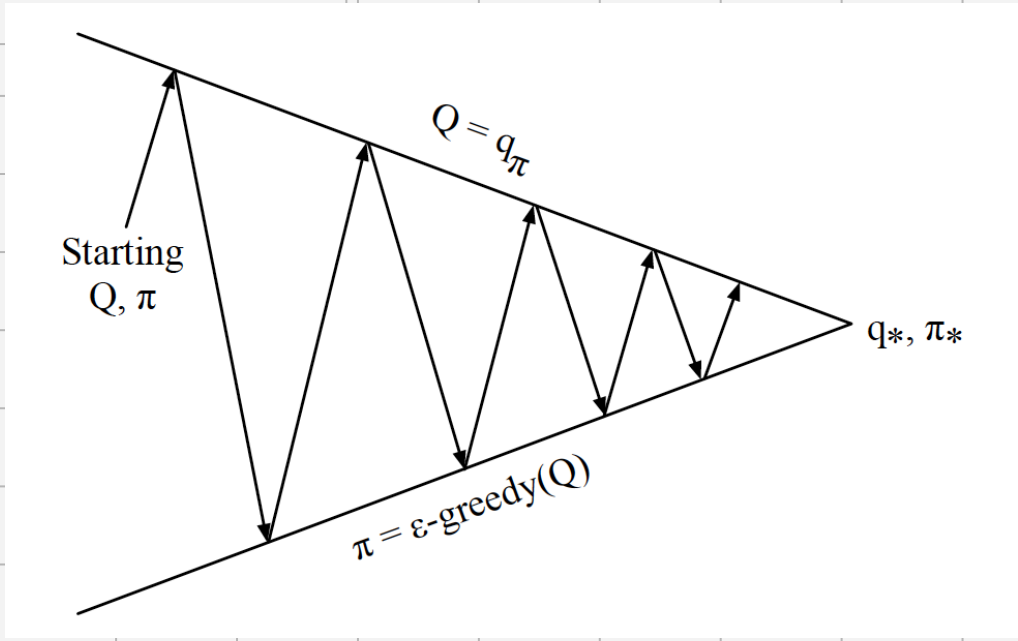
Pentru orice politică ε-greedy π, politica π’ obținută cu ajutorul q_π este o îmbunătățire față de politica anterioară, $v_{\pi'}(s) \geq v_\pi(s)$.

Mai detaliat...

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

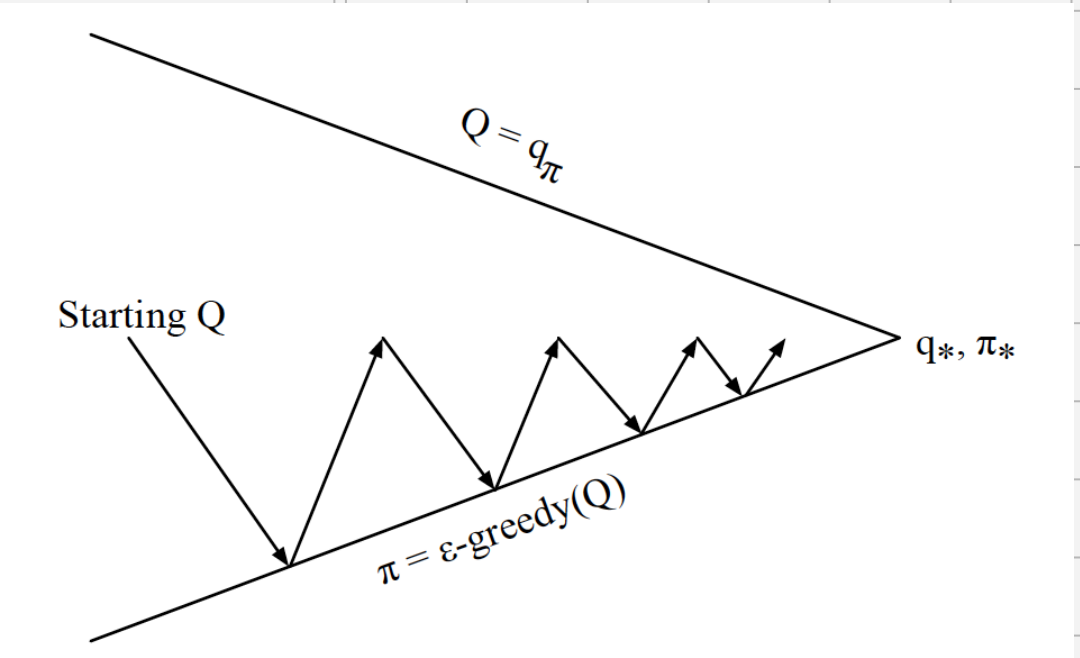
Un portret final!

- **Evaluarea politicii:** Estimarea $Q = q_\pi$
- **Îmbunătățirea politicii:** ε-greedy



Dar putem simplifica!

- **Evaluarea politicii:** **Estimarea Q $\approx q_\pi$**
- **Îmbunătățirea politicii:** **ε-greedy**



GLIE (Greedy in the Limit with Infinite Exploration)

- **Toate perechile stare-acțiune sunt explorate “la infinit”.**

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

- **Policita converge către una de tip greedy.**

$$\lim_{k \to \infty} \pi_k(a|s) = \mathbf{1}(a = \operatorname*{argmax}_{a' \in \mathcal{A}} Q_k(s, a'))$$

GLIE Monte-Carlo

- Extragem episodul cu indicele k, folosind π: $\{\boldsymbol{S_1}, \boldsymbol{A_1}, \boldsymbol{R_1}, \ldots, \boldsymbol{S_T}\} \sim \boldsymbol{\pi}$
- Pentru fiecare stare S_t și acțiune A_t din episod:

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} (G_t - Q(S_t, A_t))$$

- Îmbunătățim politica folosind valorile noi pentru “action-value function”:

$$\epsilon \leftarrow 1/k$$

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

Teoremă!

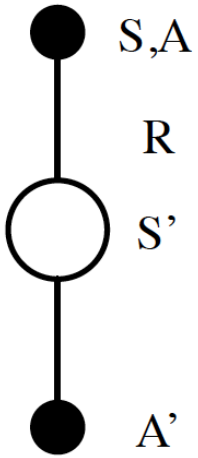
GLIE Monte-Carlo converge către zona optimă a funcției valoare-acțiune, $Q(s,a) \rightarrow q_*(s,a)$.

02

On-Policy TD

MC vs. TD

- **Ne reamintim avantajele TD:**
  - Varianță mai mică!
  - Online!
  - Învață din secvențe incomplete!
- **Ce putem face în continuare?**
  - **Aplicăm TD pentru Q(s, a) cu ε-greedy, la fiecare pas de timp t**



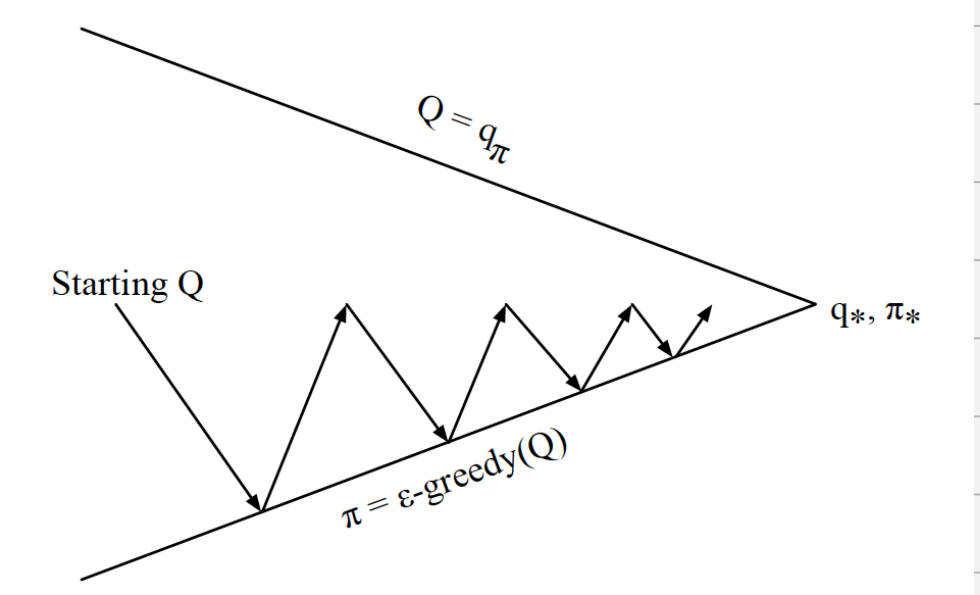
SARSA(λ)

$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma Q(S',A') - Q(S,A) \right)$$

Cum funcționează SARSA?

La fiecare pas de timp!!!

- **Evaluarea politicii:** **Estimarea Q** $\approx q_\pi$
- **Îmbunătățirea politicii:** **ε-greedy**



Algoritm - SARSA On-Policy Control

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Repeat (for each step of episode):
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

n-Step SARSA

- **“n-Step Return” pentru n = 1, 2, ...**

$$
\begin{aligned}
n = 1 \quad (Sarsa) \quad & q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1}) \\
n = 2 \quad & q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}) \\
\vdots \quad & \vdots \\
n = \infty \quad (MC) \quad & q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T
\end{aligned}
$$

n-Step SARSA

- **n-Step Q-Return**

$$q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n})$$

- **n-Step SARSA update**

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(q_t^{(n)} - Q(S_t, A_t) \right)$$

03

Off-Policy Learning

Off-Policy Learning - Introducere

- Evaluăm politica $\pi(a|s)$ pentru a calcula $v_\pi(s)$ sau $q_\pi(s,a)$.
- Dar! Urmăm comportamentul politicii $\mu(a|s)$.

- De ce este important acest tip de învățare în domeniu?
  - Este o practică bună!
  - Putem reutiliza experiențele din politici mai vechi.
  - Putem învăța *politica optimă* folosind o *politică exploratorie*.

Importance Sampling

Putem estima totul folosind distribuția unei alte politici.

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

Importance Sampling - MC

- **Folosim return-urile generate de politica μ pentru a evalua politica π.**

- **Va trebui să ajustăm G^t și $V(S_t)$.**

- **Atenție!!! Putem introduce varianță foarte mare!**

$$G_t^{\pi/\mu} = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})} \cdots \frac{\pi(A_T|S_T)}{\mu(A_T|S_T)} G_t$$

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{\pi/\mu} - V(S_t) \right)$$

Q-Learning

Importance Sampling - TD

- **Folosim target-urile generate de politica μ pentru a evalua politica π.**
- **Va trebui să ajustăm $V(S_t)$.**
- **Varianță mai mică față de versiunea cu Monte Carlo.**

$$V(S_t) \leftarrow V(S_t) + \alpha \left(\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} (R_{t+1} + \gamma V(S_{t+1})) - V(S_t) \right)$$

Ce este Q-Learning?

- Aplicăm tehnica off-policy learning pentru Q(s, a).
- Nu este necesar să folosim importance sampling!
- Următoarea acțiune este aleasă folosind comportamentul politicii:

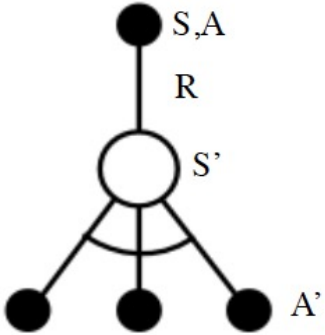
$$A_{t+1} \sim \mu(\cdot \mid S_t)$$

  - Considerăm posibilitatea unei acțiuni alternative:

$$A' \sim \pi(\cdot \mid S_t)$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t) \right)$$

Ce este Q-Learning?



$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma \max_{a'} Q(S',a') - Q(S,A) \right)$$

Thanks!

Este timpul pentru întrebări!!!

stefan.iordache10@s.unibuc.ro
catalina.patilea@s.unibuc.ro
ciprian.paduraru@fmi.unibuc.ro

+40 7.. ... ...

Introducere în Reinforcement Learning

Cursul #7

Ștefan Iordache, Cătălina Iordache, Ciprian Păduraru

Cuprins

01 Q-Learning

02 Deep Q-Networks

01

Q-Learning

De ce să folosim Q-Learning?

- **Cel mai simplu algoritm, de înțeles și aplicat!**
- **Off-Policy & Model-Free!**
- **Idee simplă: Învățăm o politică care maximizează recompensa totală.**

De ce se numește Q-Learning?

Q = Quality (cât de utilă este o acțiune pentru rezultate viitoare?)

Formula magică!

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{current value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{current value}}\Big)}^{\text{temporal difference}}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{new value (temporal difference target)}}$$

Schelet Q-Learning

- **Inițializarea tabelului Q cu valoare zero (peste tot), urmând să ajustăm valorile în pașii de antrenare.**

```python
import numpy as np

# state_size -> retine numarul de stari
# action_size -> retine numarul de actiuni posibile
Q = np.zeros((state_size, action_size))
```

Schelet Q-Learning

- **Explorare & Exploatare**

```python
import random

epsilon = 0.3

if random.uniform(0, 1) < epsilon:
    """
    EXPLORARE: selectăm o acțiune aleatorie
    """
else:
    """
    EXPLOATARE: selectăm acțiunea cu valoare maximă
    Valoare maximă = recompensa cea mai mare
    """
```

Schelet Q-Learning

- **Actualizare valori Q**

```
current_value = Q[state, action]
new_value = reward + gamma * np.max(Q[new_state, :])
temporal_difference = new_value - current_value

Q[state, action] = (1 - lr) * current_value + lr * temporal_difference
```

Antrenare

```
%%time
"""Antrenare agent"""

import random
from IPython.display import clear_output

# Hiperparametri
lr = 0.1
gamma = 0.6
epsilon = 0.1

# Colectii epoci si penalizari
all_epochs = []
all_penalties = []

for i in range(1, 100001):
    state = env.reset()

    epochs, penalties, reward, = 0, 0, 0
    done = False

    while not done:
        if random.uniform(0, 1) < epsilon:
            action = env.action_space.sample() # Explorare
        else:
            action = np.argmax(q_table[state]) # Exploatare

        next_state, reward, done, info = env.step(action)

        old_value = q_table[state, action]
        next_max = np.max(q_table[next_state])

        new_value = (1 - lr) * old_value + lr * (reward + gamma * next_max)
        q_table[state, action] = new_value

        if reward == -10:
            penalties += 1

        state = next_state
        epochs += 1

    if i % 100 == 0:
        clear_output(wait=True)
        print(f"Episodul: {i}")

print("Antrenare finalizata.\n")
```

Evaluare

```python
"""Evaluarea performanței Q-learning (după antrenare)"""

total_epochs, total_penalties = 0, 0
episodes = 100

for _ in range(episodes):
    state = env.reset()
    epochs, penalties, reward = 0, 0, 0

    done = False

    while not done:
        action = np.argmax(q_table[state])
        state, reward, done, info = env.step(action)

        if reward == -10:
            penalties += 1

        epochs += 1

    total_penalties += penalties
    total_epochs += epochs

print(f"Număr mediu de pași per episod: {total_epochs / episodes}")
print(f"Număr mediu de penalizări per episod: {total_penalties / episodes}")
```

Hiperparametri - Descriere

- **Alpha** (**α**) – Learning rate. Trend descrescător pe parcursul învățării.
- **Gamma** (**γ**) – Valoare cu trend descrescător. Către finalul episoadelor preferăm să luăm în calcul tot mai mult recompensa imediat următoare în locul celor pe termen lung.
- **Epsilon** (**ε**) – Spre finalul etapei de învățare nu mai avem nevoie de explorare, ci de exploatare. Experimentarea repetată conduce către o valoare cu trend descrescător.

Hiperparametri - Tuning

- **Grid search** – Foarte utilă la începutul experimentării cu orice algoritm de inteligență artificială.
- **Random search**
- ***Ray.tune: Hyperparameter Optimization Framework***

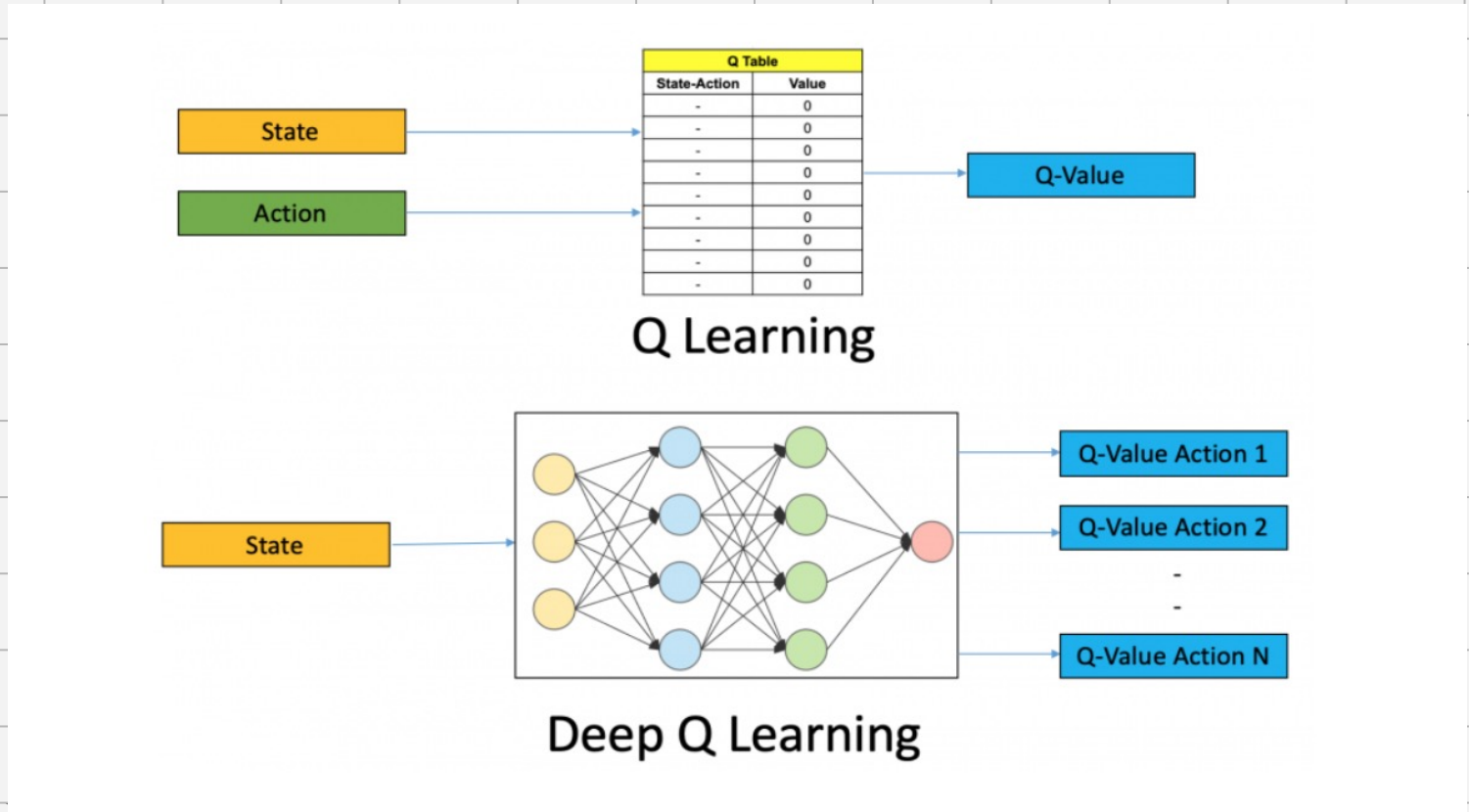
02

Deep Q-Networks

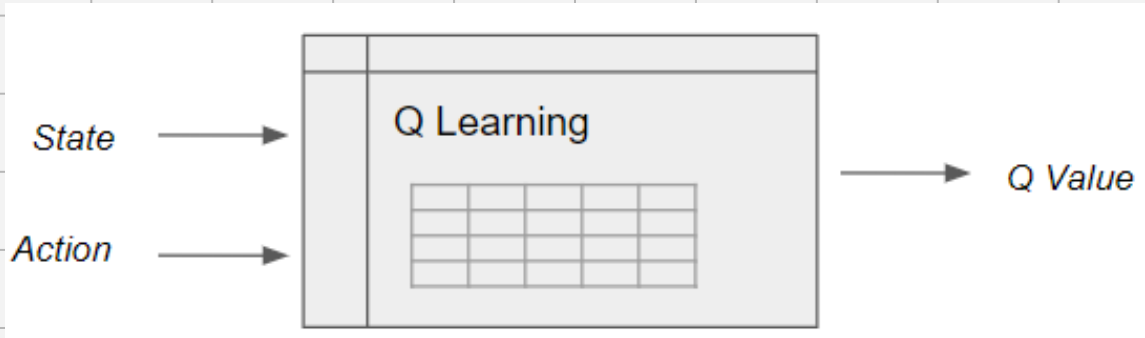
De ce avem nevoie de mai mult?

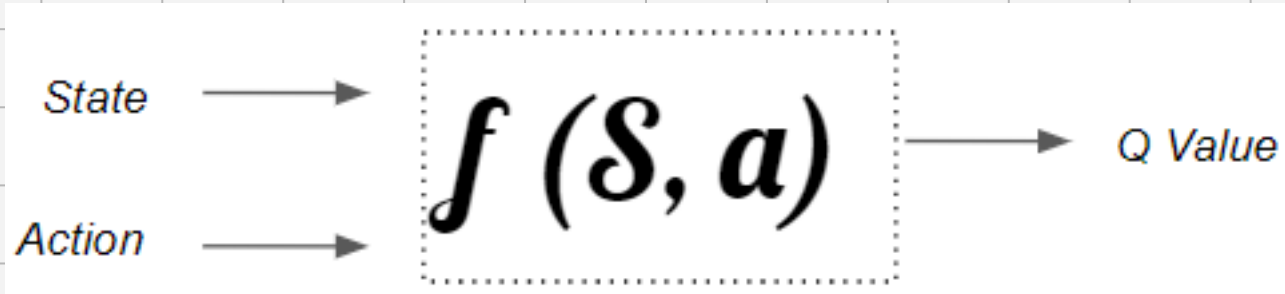
- **Mediile foarte mari sunt problematice!**
  - Necesarul de memorie pentru salvarea și actualizarea tabelului este foarte mare.
  - Timpul de explorare este nerealist.

Cum funcționează DQN/DQL?



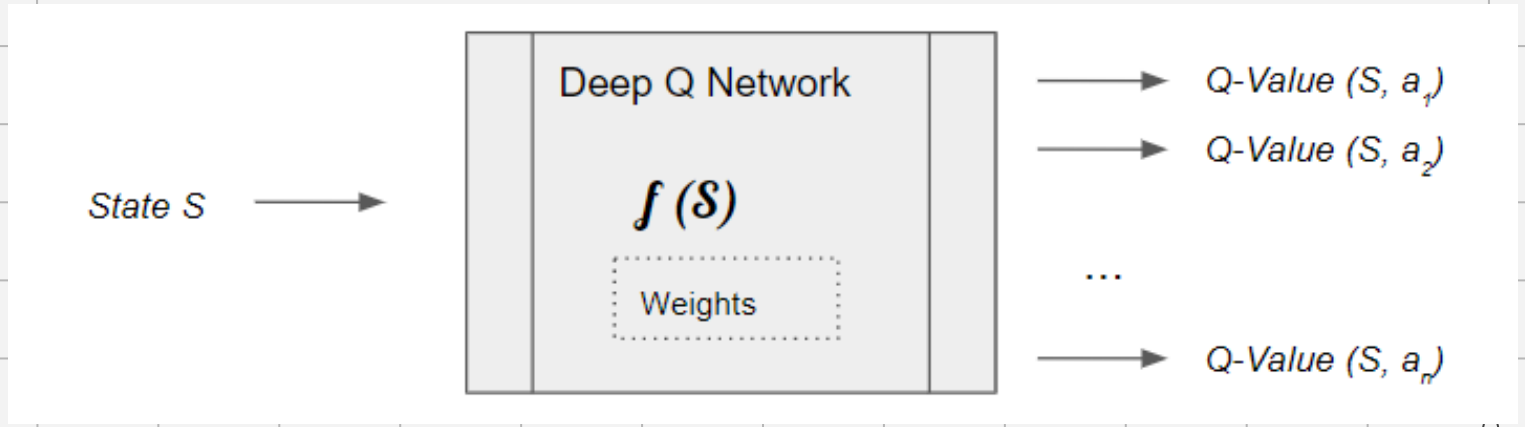
Cum funcționează DQN/DQL?



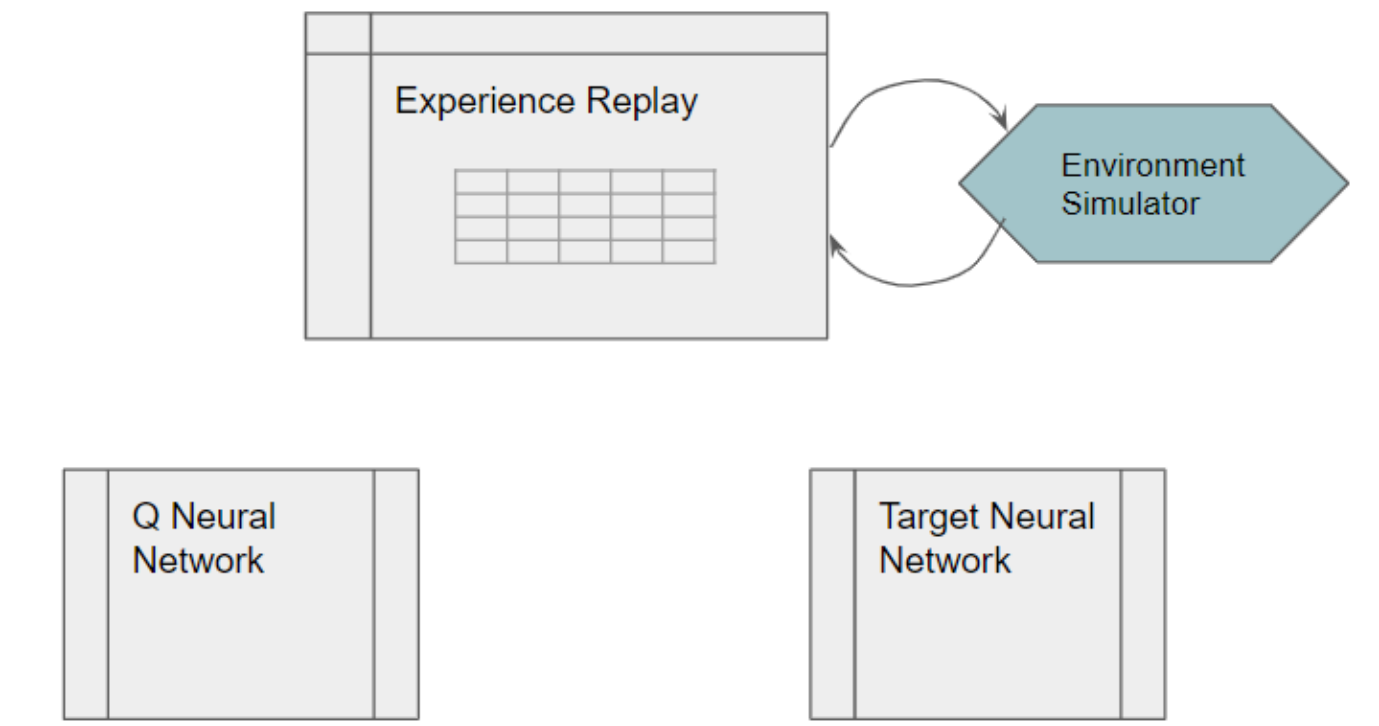


Neural Networks

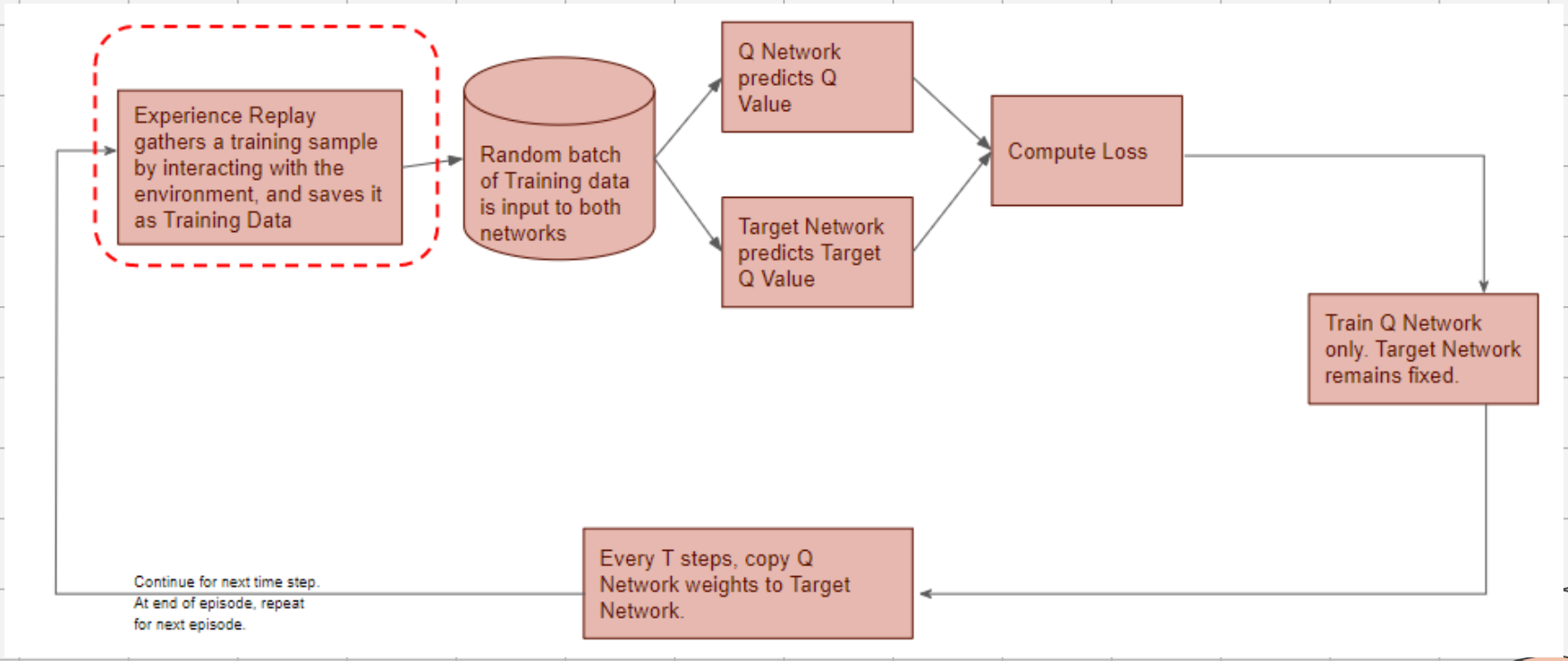
Cea mai bună metodă de aproximare a funcțiilor complexe!



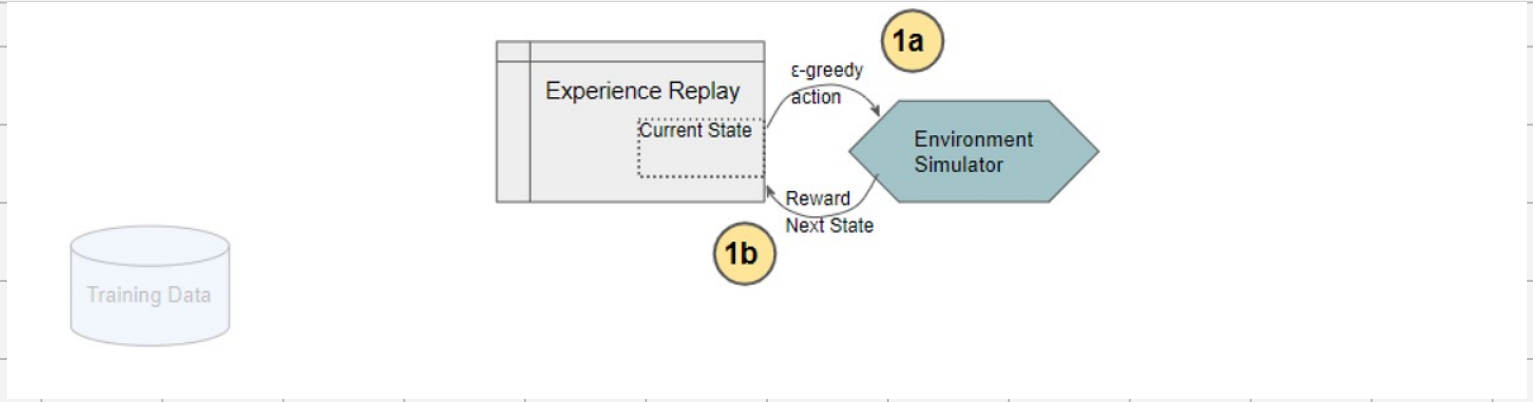
Arhitectura DQN/DQL

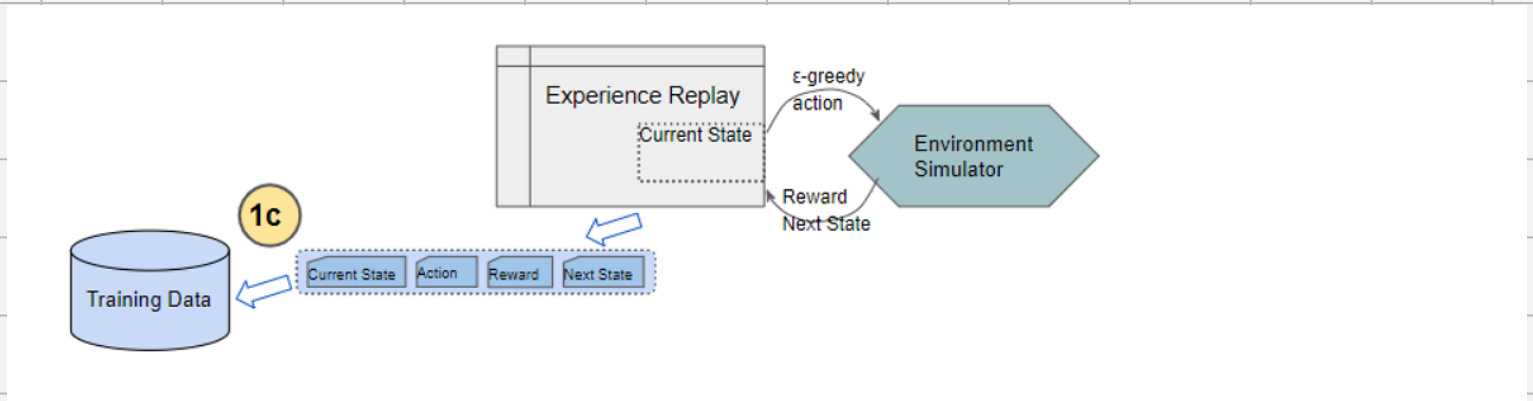


Extragerea datelor de antrenare

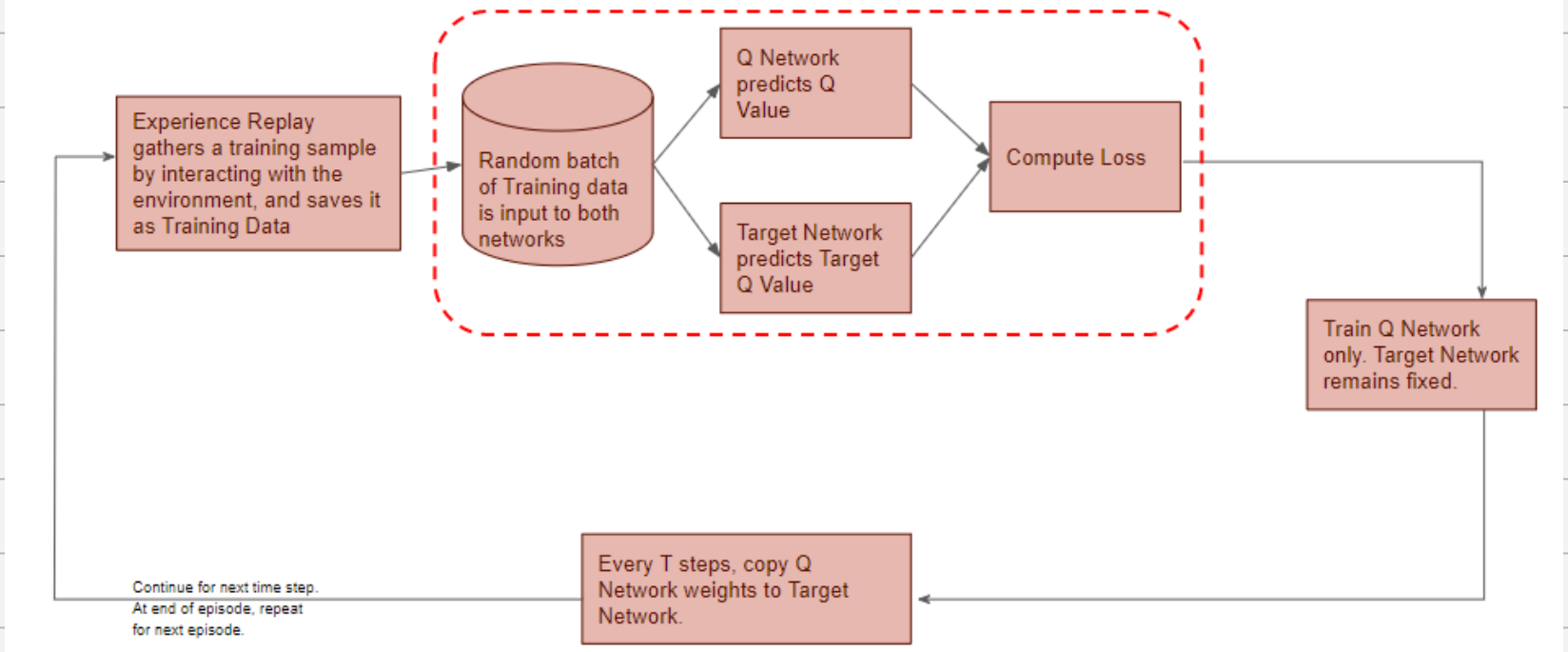


Extragerea datelor de antrenare

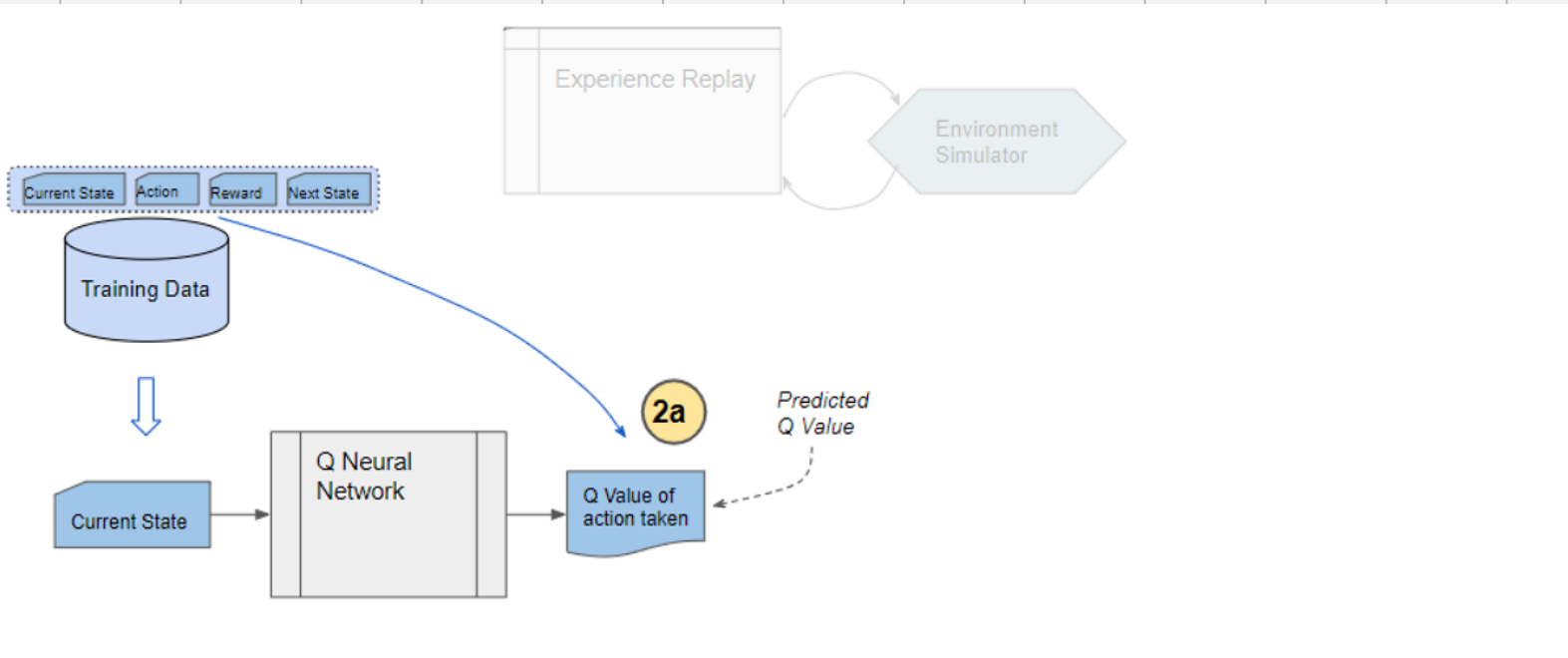




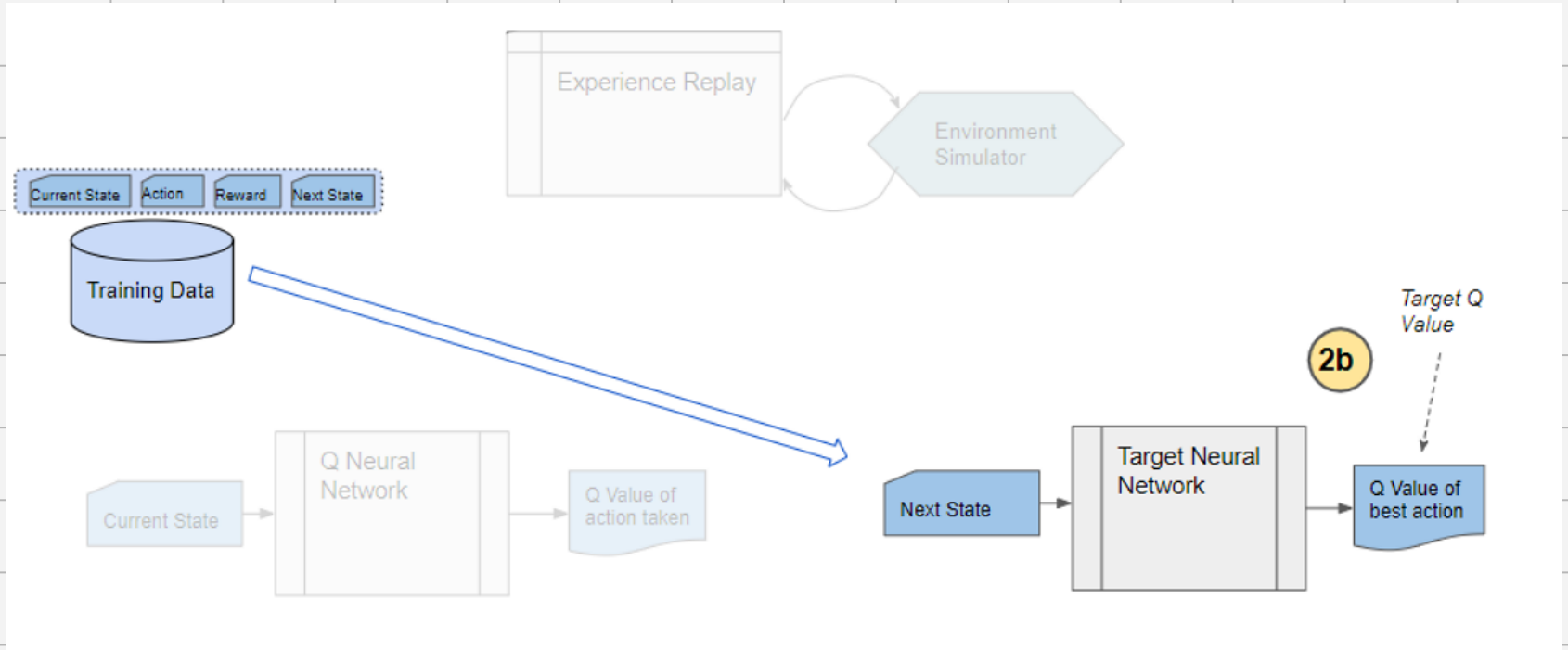
Predicții


Experience Replay gathers a training sample by interacting with the environment, and saves it as Training Data
Random batch of Training data is input to both networks
Q Network predicts Q Value
Target Network predicts Target Q Value
Compute Loss
Train Q Network only. Target Network remains fixed.
Every T steps, copy Q Network weights to Target Network.
Continue for next time step.
At end of episode, repeat for next episode.

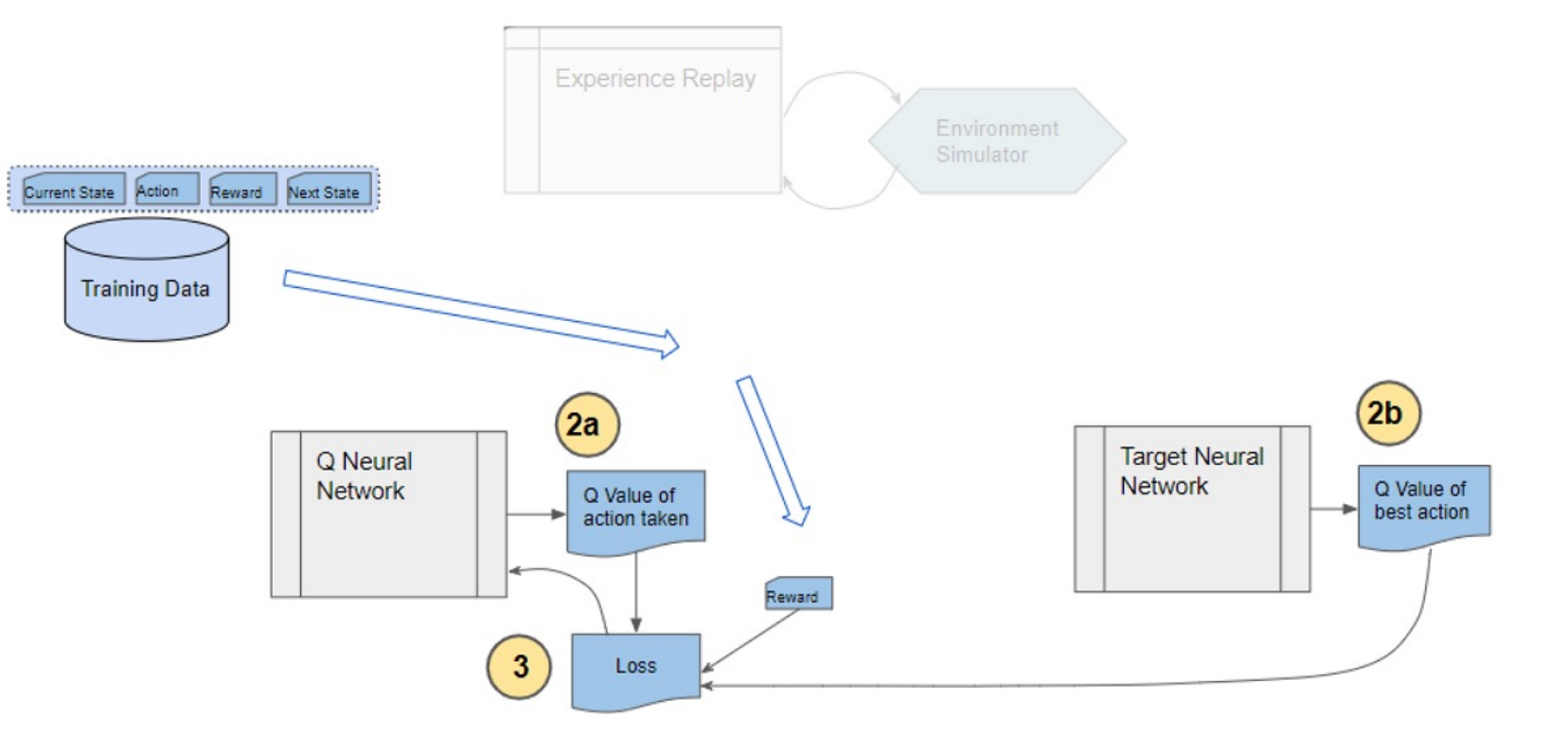
Predicții – Q-value



Predicții – Target Q-value



Loss & Q-Network Train



Thanks!

Este timpul pentru întrebări!!!

stefan.iordache10@s.unibuc.ro
catalina.patilea@s.unibuc.ro
ciprian.paduraru@fmi.unibuc.ro

+40 7.. ... ...

Introducere în Reinforcement Learning

Cursul #8

Ștefan Iordache, Cătălina Iordache, Ciprian Păduraru

Cuprins

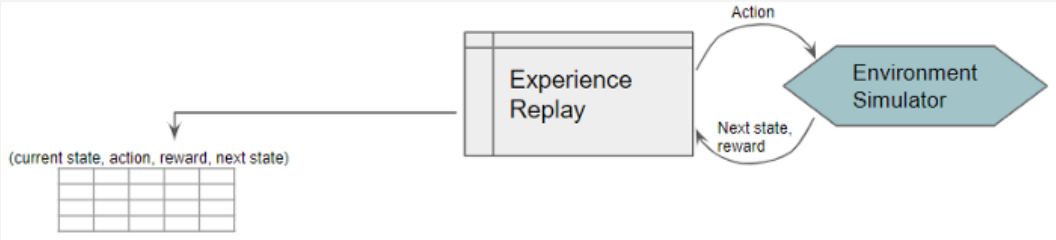
01 DQN In-depth

02 Policy Gradients

01

DQN In-depth

DQN – Inițializarea

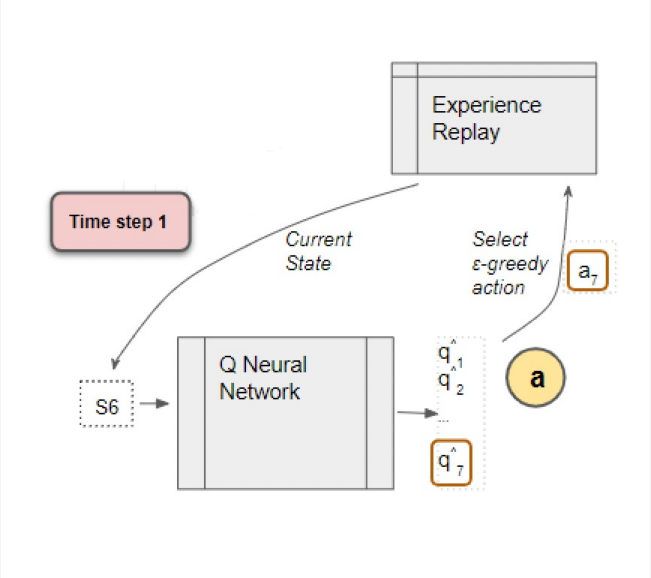


- Executăm un număr mare de pași pentru a construi setul de antrenare.

- Inițializăm Q Network cu ponderi aleatorii și le copiem către Target Network.

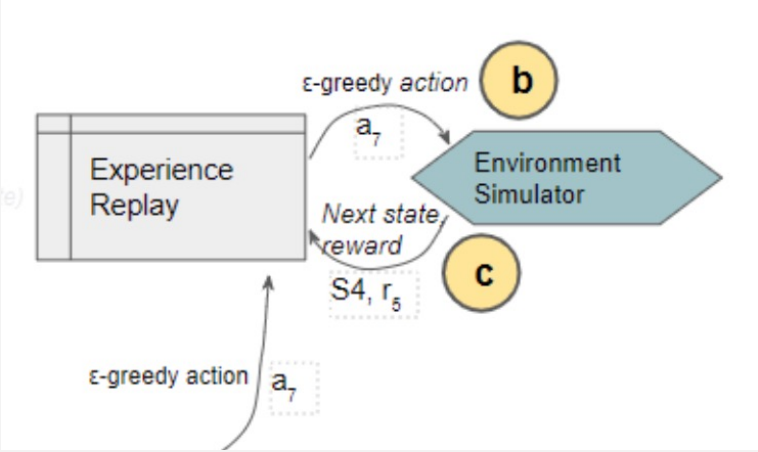
DQN – Experience Replay

- Q Network selectează o acțiune în mod **ε**-greedy, acționând precum agentul în timp ce interacționează cu mediul pentru generarea unui eșantion de antrenare.
- Nu se întâmplă partea de învățare aici!



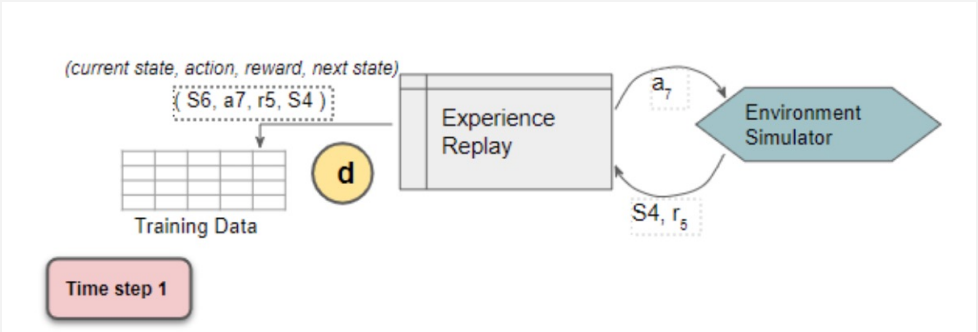
DQN – Experience Replay

- Executăm acțiunea **ε**-greedy și obținem perechea formată din (următoarea stare, reward).



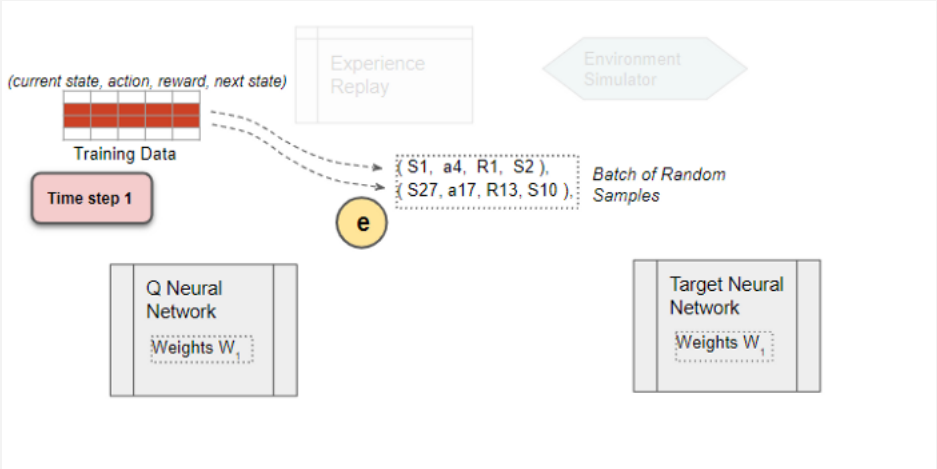
DQN – Experience Replay

- Stocăm tuplul obținut, fiind folosit mai târziu drept eșantion pentru partea de antrenare.



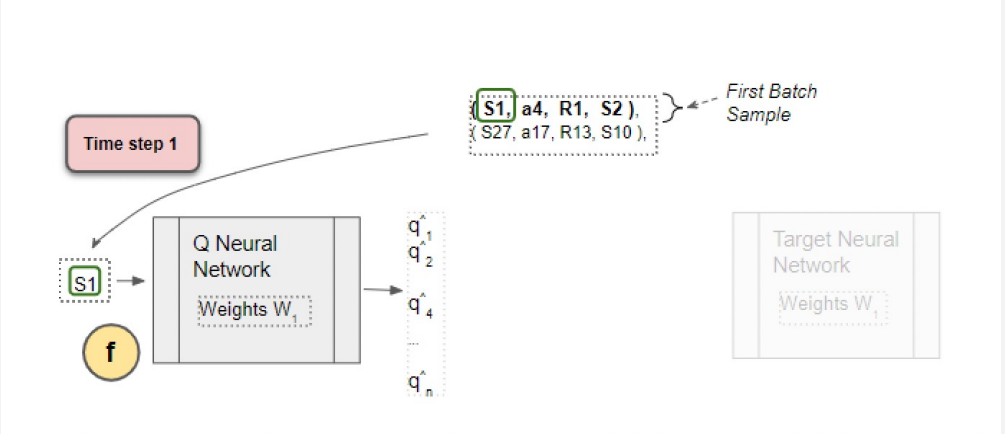
DQN – Q-Network

- Selectăm un batch aleatoriu drept input pentru cele două rețele.



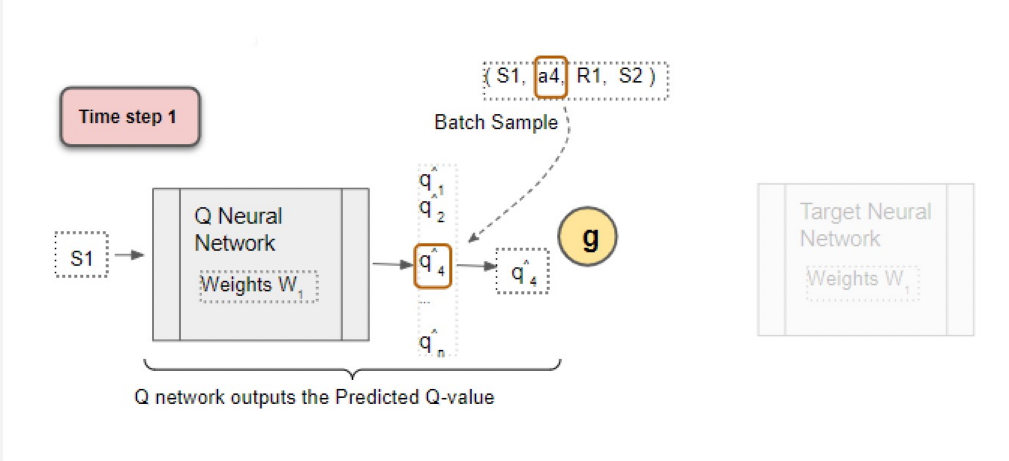
DQN – Q-Network

- Folosim starea curentă (S1, S27) din sample pentru a prezice valoarea Q pentru toate acțiunile care pot fi realizate din starea respectivă.



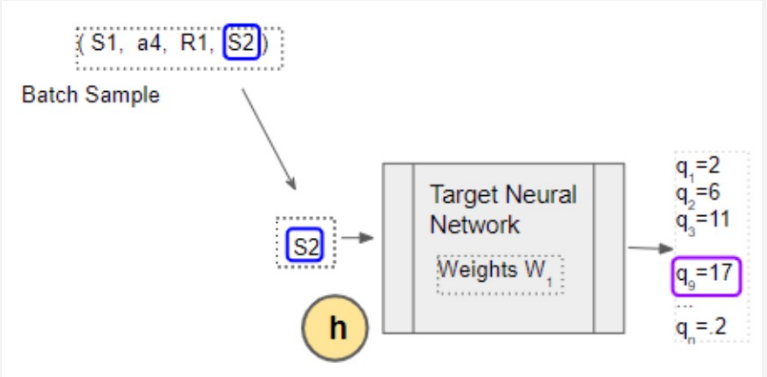
DQN – Q-Network

- Selectăm acțiunea prezisă cu ajutorul Q-value (în cazul expus a4, cu ajutorul q4).

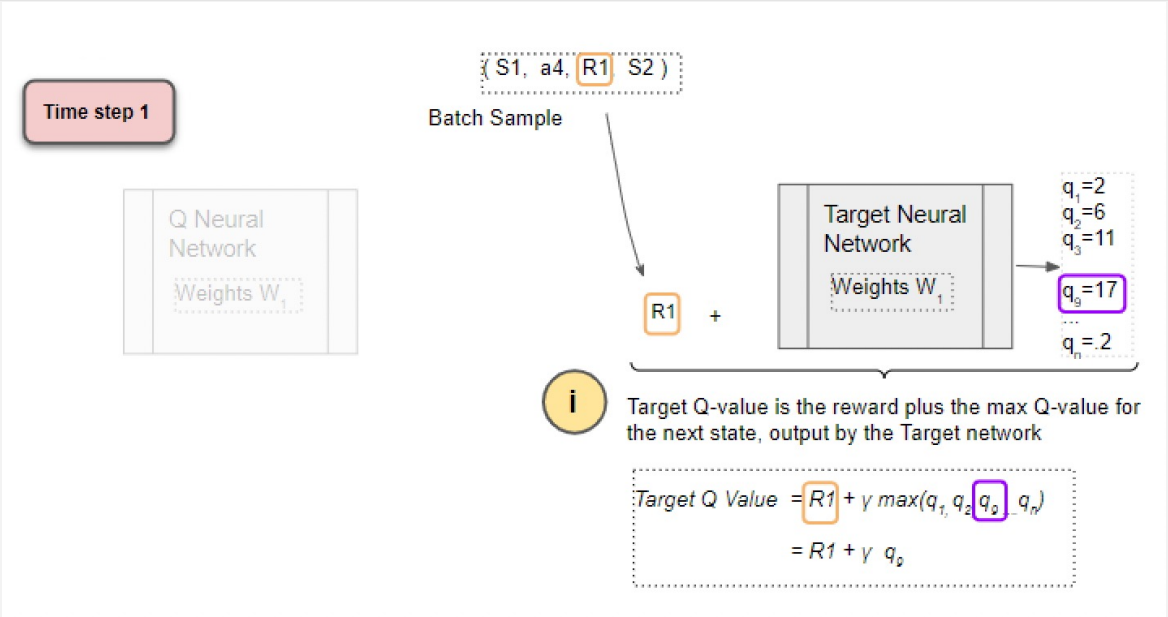


DQN – Target Network

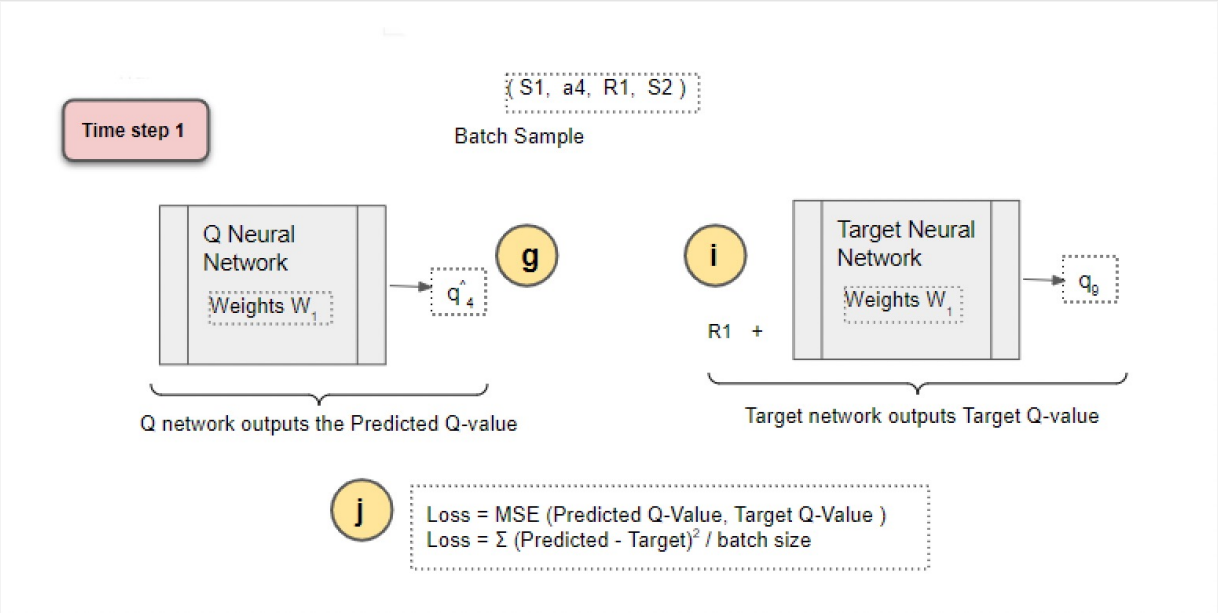
- Folosim starea următoare drept input pentru Target Network, astfel prezicând valorile Q pentru toate acțiunile care pot fi luate din starea următoare.
- Selectăm acțiunea cu Q-value maxim.



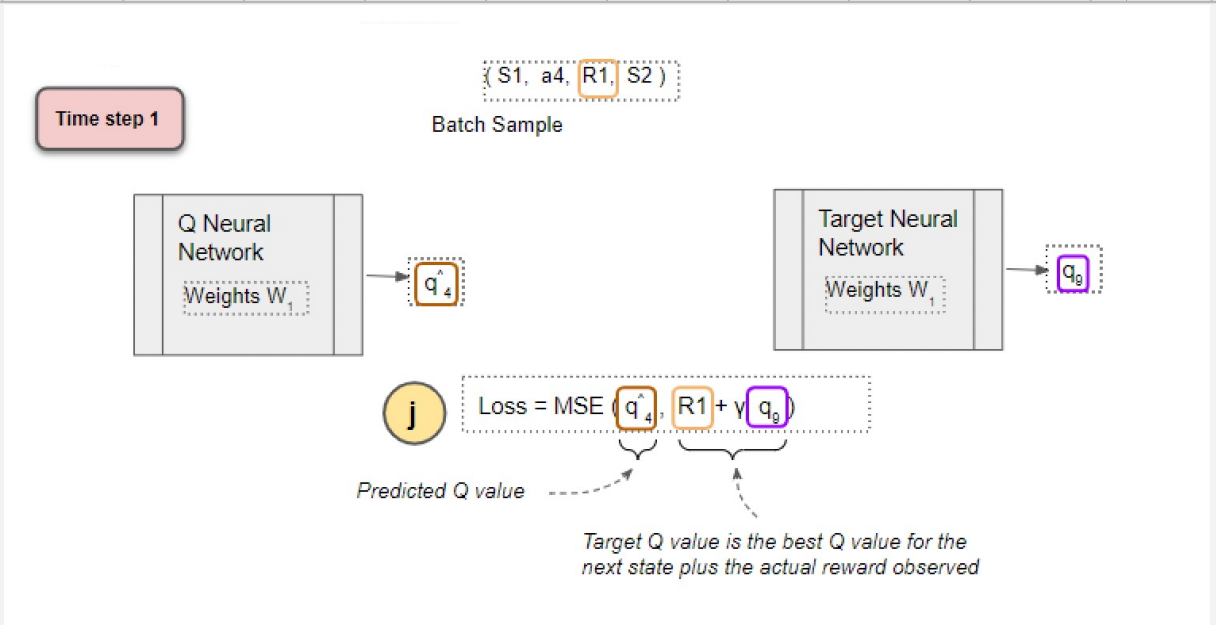
DQN – Target Q-Value



DQN – Loss

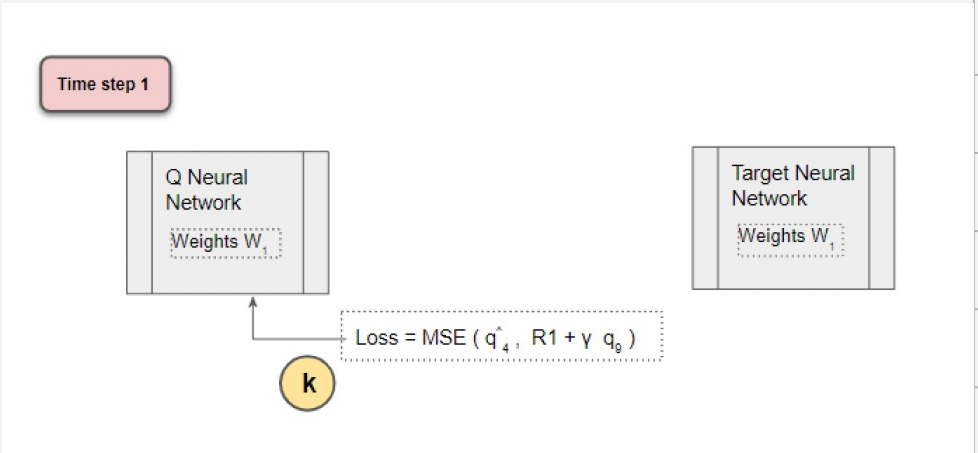

(S1, a4, R1, S2)
Batch Sample
Time step 1
Q Neural Network
Weights W1
g
i
Target Neural Network
Weights W1
R1 +
Q network outputs the Predicted Q-value
Target network outputs Target Q-value
j
Loss = MSE (Predicted Q-Value, Target Q-Value)
Loss = Σ (Predicted - Target)2 / batch size

DQN – Loss


(S1, a4, R1, S2)
Batch Sample
Time step 1
Q Neural Network
Weights W1
Target Neural Network
Weights W1
j
Loss = MSE (, R1 + γ)
Predicted Q value
Target Q value is the best Q value for the next state plus the actual reward observed

DQN – Back-Propagation – Q-Network

- Actualizăm ponderile pentru Q-Network prin procedeul denumit Back-Propagation, asistat de Gradient Descent.
- Nu sunt actualizate ponderile pentru Target Network!



DQN – Update Target Network

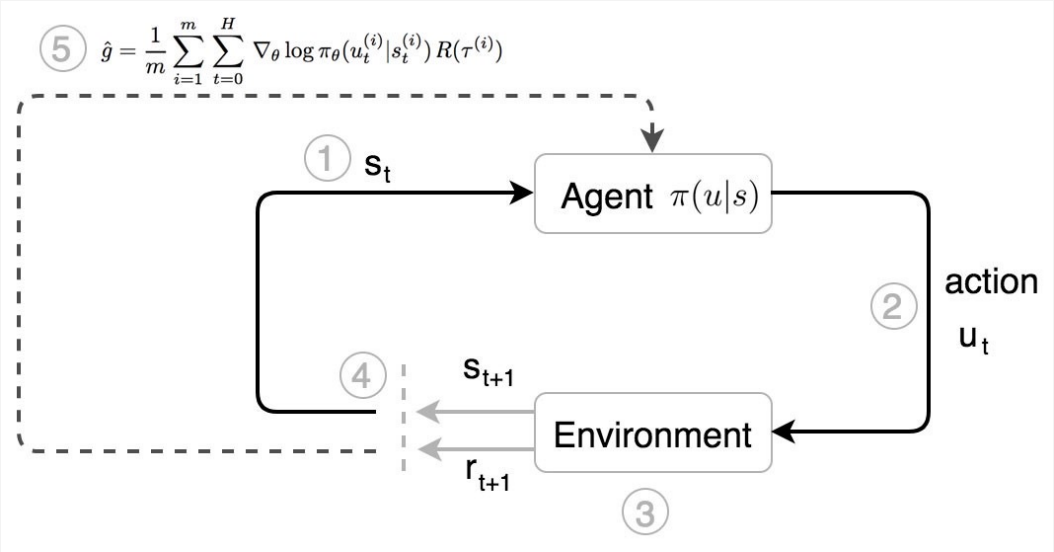
- Actualizarea Target Network se realizează prin copierea Q-Network după T momente de timp.

02

Policy Gradients

Cum funcționează? Ce e diferit?

- **Pasul 5 este cel mai important!**
- **Considerând traiectoria τ vom ajusta politica folosind totalul de recompense R(τ).**



$$(s_1, u_1, s_2, u_2, \ldots, s_H, u_H)$$

Policy Gradients?

Încercăm direct să maximizăm rezultatul prin pași mici în direcția gradientului.

Obiectiv!

- **Expected reward = suma probabilității unei anumite traiectorii x recompensă**

$$J(\theta) = \mathrm{E}\Big[\sum_{t=0}^{H} R(s_t, u_t); \pi_\theta\Big] = \sum_{\tau} P(\tau;\theta) R(\tau)$$

$$\max_{\theta} J(\theta) = \max_{\theta} \sum_{\tau} P(\tau;\theta) R(\tau)$$

Orizontul (H)

- **Introducem recompensele în calcul și exprimăm prin H numărul maxim de pași.**
- H poate tinde către infinit, până la o stare terminală, dar în practică nu folosim această ipoteză.

$$(s_1, u_1, r_1, s_2, u_2, r_2, \dots, s_H, u_H, r_H)$$

Optimizări!

- **Folosim derivata parțială a unei funcții f(x)!!!**

$$f(x)\,\nabla_\theta \log f(x) = f(x)\,\frac{\nabla_\theta f(x)}{f(x)} = \nabla_\theta f(x)$$

- **Înlocuim f(x) cu politica π!!!**

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \nabla_\theta \pi_\theta(\tau)$$

Multe calcule mai târziu...

- **Formula finală pentru Policy Gradient!**

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

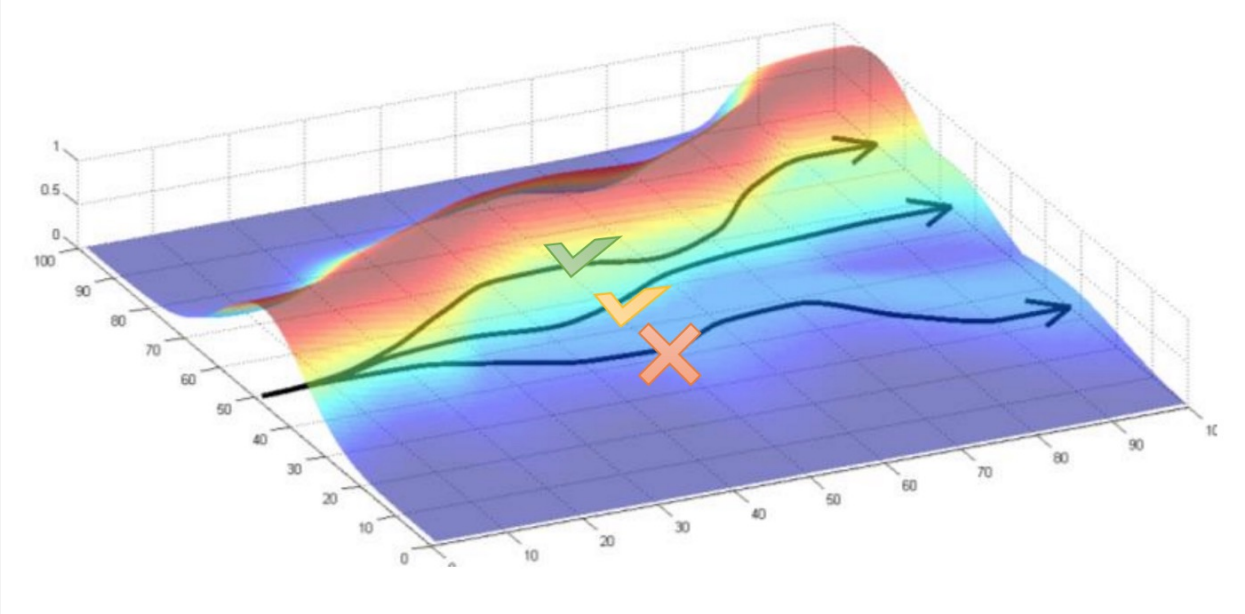
Dar...nu are logică!!!

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

Termenul subliniat este numit drept “maximum log likelihood”. În contextul nostru măsurăm în ce grad traiectoria aleasă se află sub politica curentă. Multiplicând cu reward-ul dorim să creștem acest grad daca traiectoria rezultă în reward-uri bune și contrar dacă scade recompensa.

Pe scurt: **Păstrăm ceea ce funcționează, aruncăm tot ce nu este bun!**

Să vizualizăm!



Un nou algoritm: REINFORCE!

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Defectul Policy Gradients

- **Varianță mare!**
- **Covergență scăzută!**

- Varianța crescută provoacă o coborâre anormală pe gradient, astfel modelul fiind incapabil să învețe pe deplin și să atingă convergența.
- Cum rezolvăm???

Baseline

- **Introducem V(s), denumit și baseline.**

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\underbrace{\left(\sum_{t'=1}^{T} r(\mathbf{s}_{i,t'},\mathbf{a}_{i,t'})\right)}_{Q(s,\,a)}$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\left(Q(\mathbf{s}_{i,t},\mathbf{a}_{i,t}) - V(\mathbf{s}_{i,t})\right)$$

- **Continuăm și introducem ceva special pentru Policy Gradients: Advantage**

$$A^\pi(\mathbf{s}_t,\mathbf{a}_t) = Q^\pi(\mathbf{s}_t,\mathbf{a}_t) - V^\pi(\mathbf{s}_t)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})A^\pi(\mathbf{s}_{i,t},\mathbf{a}_{i,t})$$

Cum explicăm mai departe?

- **Exemple:**
  - Situația I: Traiectoria A primește recompensa +20 și traiectoria B primește recompensa -10. => **Creștem probabilitatea pentru A, o scădem pentru B.**
  - Situația II: Traiectoria A primește recompensa +20 și traiectoria B primește recompensa +5. => **Creștem probabilitatea atât pentru A, cât și pentru B.**

Thanks!

Este timpul pentru întrebări!!!

stefan.iordache10@s.unibuc.ro
catalina.patilea@s.unibuc.ro
ciprian.paduraru@fmi.unibuc.ro

+40 7.. ... ...