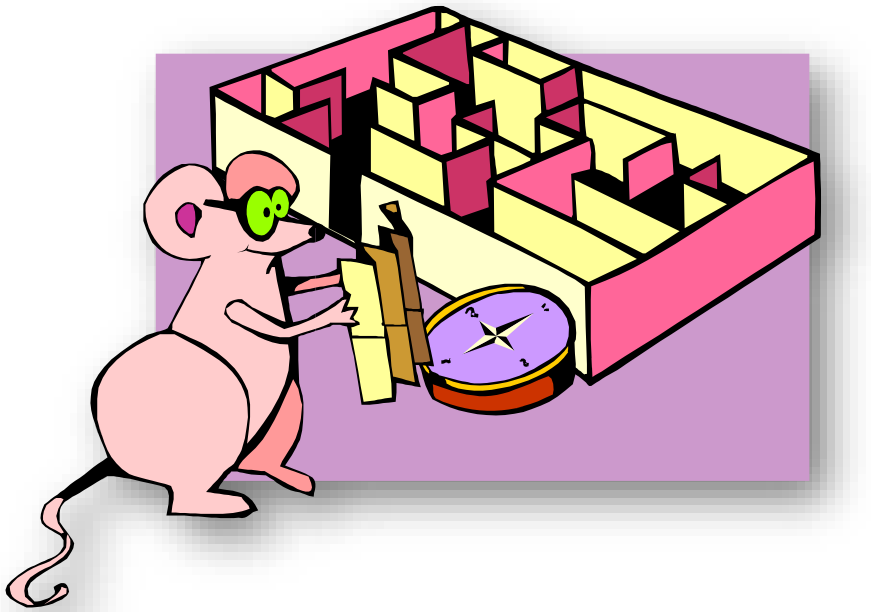
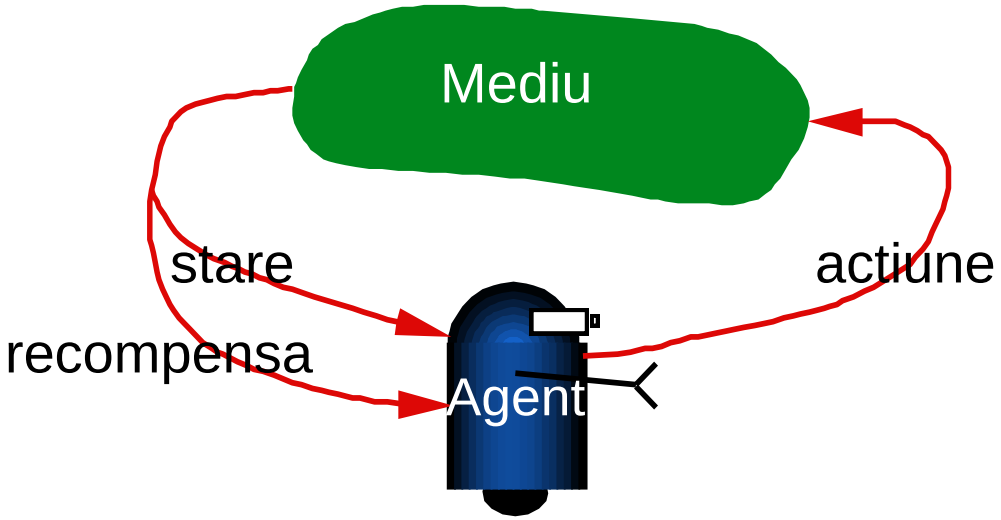


Invatare reimprospatata

Catalin Stoean

Ce este invatarea reimprospatata?

- Este **invatarea din interactiuni**.
- Avem un agent care
  - Invata si planifica permanent
  - Afecteaza mediul inconjurator
  - Are o multime de sarcini
  - Invata in urma a multiple mutari de genul incercare-si-eroare.



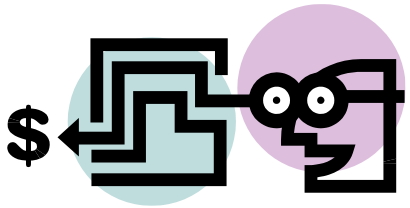
Caracteristici ale invatarii reimprospatate

- *Invatarea reimprospatata inseamna a invata cum sa actionezi pentru a maximiza o recompensa numerica.*

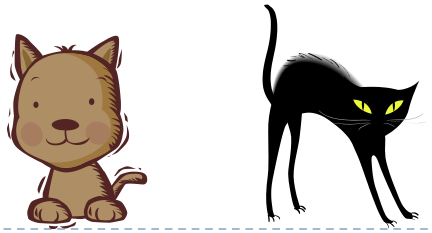
- Invatare din recompense numerice
- Interactionare cu sarcinile
  - Secvente de stari, actiuni si recompense
- Lumi incerte si nedeterministe
- Consecinte intarziate
- Invatare directionata catre tinta
- Echilibru intre explorare si exploatare

Puncte de vedere

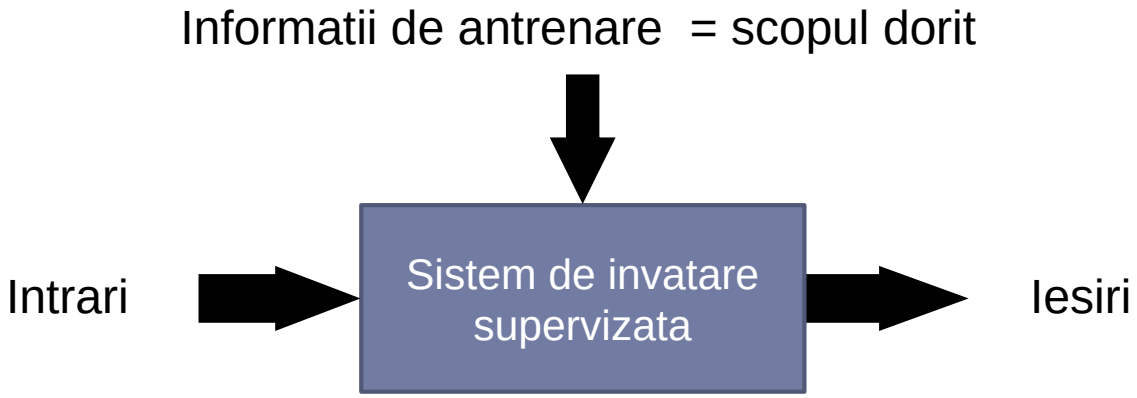
- Din punctul de vedere al agentului care invata:
  - Invatarea reimprospatata este invatare din interactiunea cu mediul inconjurator prin incercare si eroare
  - Ex: ce recompensa primesc daca fac acest lucru?
- Invatarea reimprospatata ca o unealta
  - Invatarea reimprospatata din recompense si pedepse
  - Antrenarea calculatorului in acelasi fel in care antrenezi un caine
- Aplicabilitate: probleme cu interactiune continua
  - Robotica
  - Invatarea la animale
  - Planificare
  - Jocuri
  - Sisteme de control



Invatare supervizata



- Pasul 1
  - Profesorul: Ce avem in imaginea 1, un caine sau o pisica?
  - Elevul: O pisica.
  - Profesorul: Nu, este un caine.
- Pasul 2
  - Profesorul: Imaginea 2 contine un caine sau o pisica?
  - Elevul: O pisica.
  - Profesorul: Da, este o pisica.
- Pasul 3 …

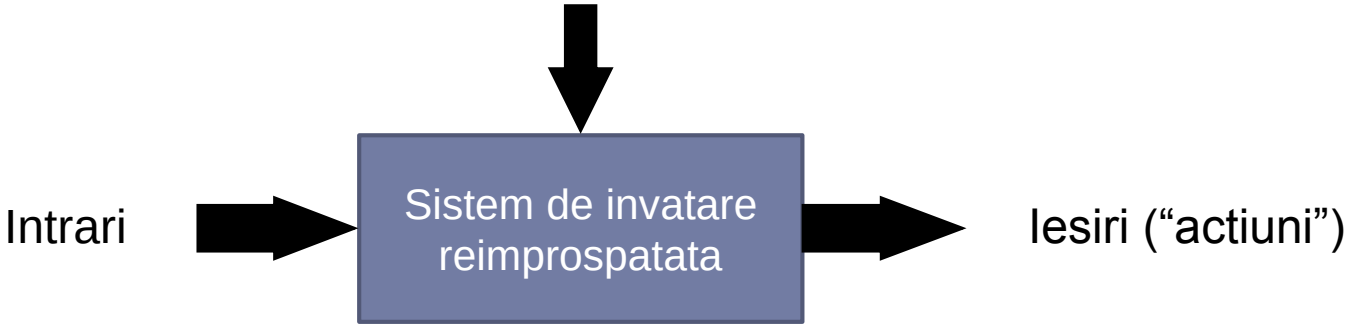


Eroare = (iesirea gasita – iesirea efectiva)

Invatarea reimprospatata

- Pasul 1
  - Mediul: Te afli in starea 8. Alege intre actiunile A sau C.
  - Elevul: Actiunea C.
  - Mediul: Recompensa ta este de 100.
- Pasul 2
  - Mediul: Te afli in starea 17. Alege intre actiunile B sau F.
  - Elevul: Actiunea B.
  - Mediul: Recompensa ta este de 50.
- Pasul 3 …

Informatii de antrenare = evaluari (“recompense” / “penalizari”)

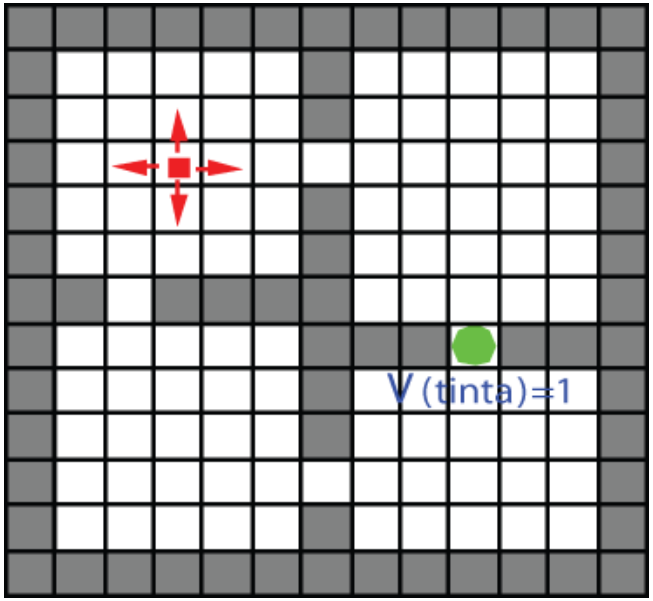


Obiectiv: cat mai multe recompense posibile

Invatarea reimprospatata

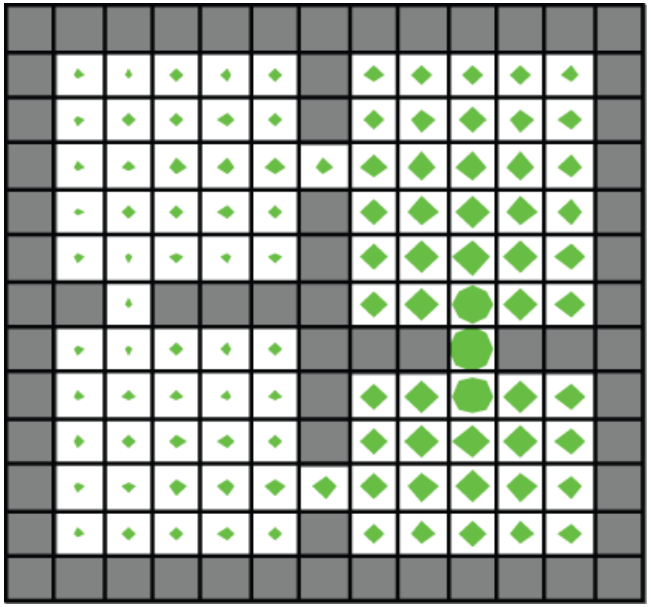
Formularea problemei

- Pornind din casuta cu patratul rosu, sa se ajunga la tinta (cercul verde).



Dupa aplicarea invatarii reimprospatate

- Urmele mai intense de verde simbolizeaza o recompensa mai mare pentru agent.



Invatare reimprospatata

- Agentul invata prin interactiunea cu mediul si prin observarea rezultatelor obtinute din aceste interactiuni.
  - Este vorba de “cauza si efect” si acesta este si modul in care noi ne formam cunoasterea aupra mediului pe parcursul vietii.

- Ideea de “cauza si efect” se traduce in pasii urmatori pentru un agent din cadrul invatarii reimprospatate:
  1. Agentul observa o stare de intrare
  2. Actiunea sa este determinata de o functie de luare de decizie (o **strategie**).
  3. Actiunea este indeplinita
  4. Agentul primeste o recompensa numerica de la mediu
  5. Informatia despre recompensa primita pentru starea/actiunea respectiva este retinuta

Invatare reimprospatata

- Prin executarea de actiuni si observarea recompenselor obtinute, strategia folosita pentru a determina cea mai buna actiune pentru o stare poate fi imbunatatita.
- Daca suficiente stari sunt observate, o strategie de decizie optimala va fi generata si vom obtine un agent care actioneaza perfect in mediul sau.
- Asadar, agentul invata din recompensele primite de la mediu, fara sa existe vreo alta forma de supervizare in afara de propria strategie de a isi alege deciziile.
- Este *aruncat* in mediul sau si lasat sa se descurce singur, din propriile greseli si succese.

Explorare si exploatare

- Daca agentul a incercat o actiune in trecut si a primit o recompensa potrivita, atunci repetarea acestei actiuni va reproduce aceeasi valoare.
  - Agentul **exploateaza** ceea ce stie pentru a primi recompensa.
- Pe de alta parte, agentul poate incerca alte posibilitati si ar putea obtine acolo recompense mai bune, deci **explorarea** este o tactica buna deseori.
- Fara un echilibru intre explorare si exploatare, agentul nu va invata eficient.

Functii Valoare

- Sunt functii de perechi stare-actiune care estimeaza cat de buna o anumita actiune va fi intr-o stare data sau care este rezultatul asteptat pentru acea actiune.

- $V^{\pi}(s)$ – valoarea unei stari s sub strategia π.
  - Recompensa asteptata cand se incepe in s, urmand strategia π.

- $Q^{\pi}(s, a)$ – valoarea pentru luarea actiunii a in starea s sub strategia π
  - Recompensa asteptata cand se incepe din s, se ia actiunea a si apoi se urmeaza strategia π.

Invatarea bazata pe diferente temporale (DT)

- Se foloseste pentru a estima aceste functii valoare.
- Daca nu se estimeaza functia valoare, agentul trebuie sa astepte pana se primeste recompensa finala pentru a actualiza valorile pentru perechi stare-actiune.
  - Pentru acest caz in care se merge pana la tinta pentru evaluare, se foloseste formula:

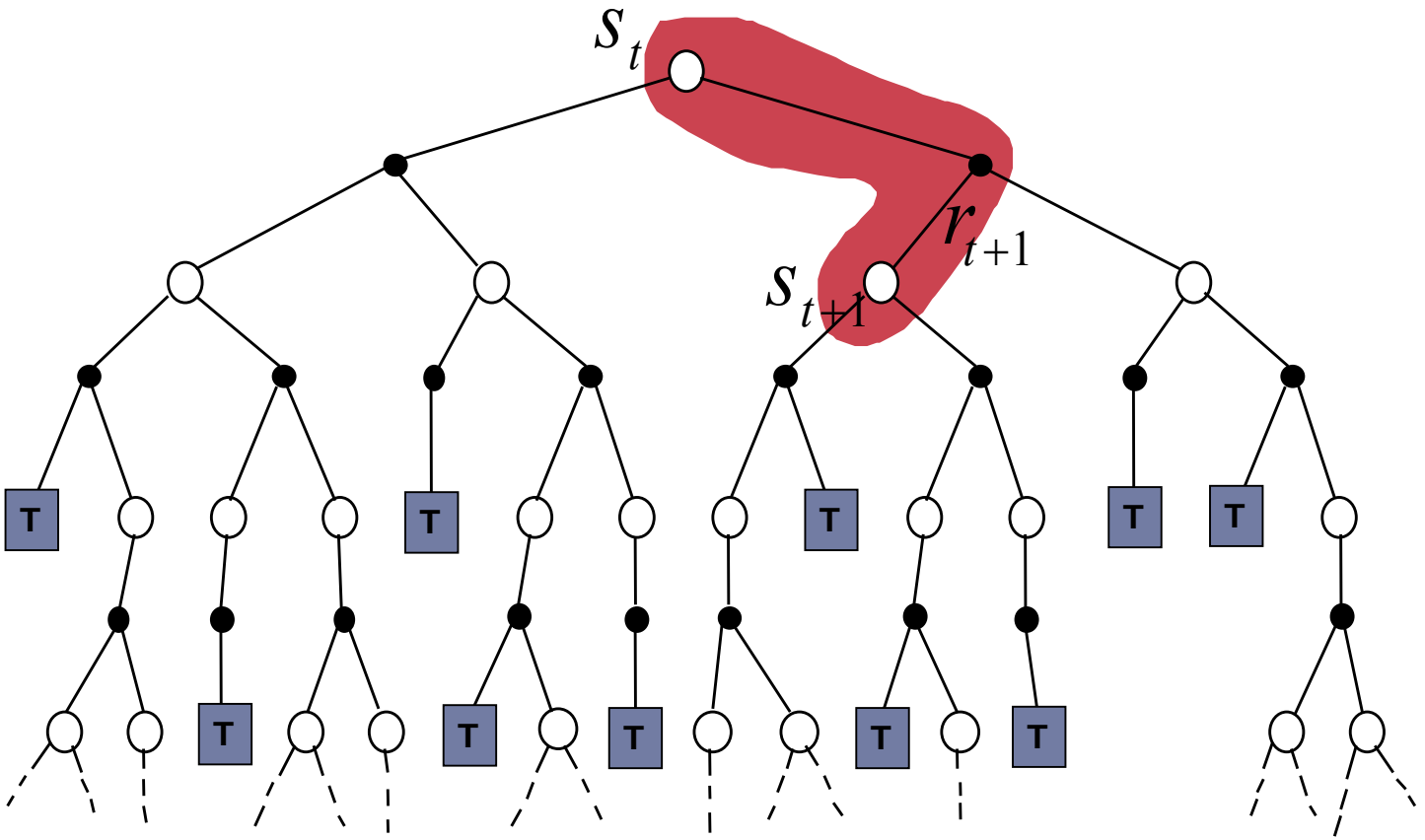
$$V(s_t) = V(s_t) + \alpha(R_t - V(s_t))$$

- s_t este starea vizitata la momentul t
- R_t – recompensa dupa momentul t
- α – parametru constant

Invatarea bazata pe DT

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

- r_{t+1} este recompensa observata la momentul t+1.



Strategii de selectare a actiunilor

- In functie de strategie, se controleaza echilibrul intre explorare si exploatare.
  - **ε-greedy**
    - De cele mai multe ori, actiunea care intoarce cea mai mare recompensa estimata este selectata.
    - Cu o mica probabilitate, ε, se alege o actiune in mod aleatoriu, independent de estimarile pentru recompense.
  - **Softmax**
    - Se ataseaza o pondere pentru fiecare actiune relativ la estimarea starii in care se ajunge.
    - Alegerea actiunilor se face in mod aleatoriu, insa proportional cu ponderea fiecarei actiuni.
    - Cele mai bune actiuni au sanse mari sa fie selectate, iar cele mai proaste au sanse foarte mici.

Algoritmi de invatare
Invatarea Q

1. Se initializeaza **Q(s, a)** in mod aleatoriu sau cu zero
2. Repeta
   1. Initializeaza starea **s**
   2. Repeta
      1. Alege actiunea **a** in functie de strategia aleasa (ε-greedy sau Softmax)
      2. Executa actiunea **a**, observa **r** si **s'**
      3. $\mathbf{Q(s, a)} = \mathbf{Q(s, a)} + \alpha[\mathbf{r(s, a)} + \gamma \max_{a'} \mathbf{Q(s', a')} - \mathbf{Q(s, a)}]$
      4. $\mathbf{s} = \mathbf{s'}$
   3. Pana cand s este stare terminala
3. Pana cand se intalneste conditia de oprire (un numar de iteratii)

- α – rata de invatare, cu valori in [0, 1].
  - Daca luam α = 0, valorile pentru Q nu se modifica, deci nu se invata nimic.
  - Daca luam α = 0.9, invatarea are loc foarte rapid.
- γ factorul de reducere cu valori tot in [0, 1].
  - Face ca recompensele urmatoare sa conteze mai putin decat cele imediate.
- $\max_{a'}$ – recompensa maxima ce poate fi obtinuta in starea care urmeaza starii actuale, adica recompensa daca se ia cea mai buna actiune apoi.

Algoritmi de invatare
Invatarea Q

1. Se initializeaza **Q(s, a)** in mod aleator sau cu zero
2. Repeta
   1. Initializeaza starea **s**
   2. Repeta
      1. Alege actiunea **a** in functie de strategia aleasa (ε-greedy sau Softmax)
      2. Executa actiunea **a**, observa **r** si **s'**
      3. $\mathbf{Q(s, a)} = \mathbf{Q(s, a)} + \alpha[\mathbf{r(s, a)} + \gamma \max_{a'} \mathbf{Q(s', a')} - \mathbf{Q(s, a)}]$
      4. $\mathbf{s} = \mathbf{s'}$
   3. Pana cand s este stare terminala
3. Pana cand se intalneste conditia de oprire (un numar de iteratii)

- Se initializeaza tabela de valori Q
- Se observa starea curenta s
- Se alege o actiune a din starea s
  - Actiunea se alege in functie de strategia folosita
- Se executa actiunea, se ajunge la o noua stare s' si se observa recompensa r care se obtine din starea s, daca se ia actiunea a.
- Se actualizeaza valoarea Q pentru starea curenta folosind recompensa obtinuta si cea maxima posibila pentru starea urmatoare (linia 2.2.3).
- Se trece la urmatoarea stare s'.

- α – rata de invatare
- γ factorul de reducere
- $\max_{a'}$ – recompensa maxima ce poate fi obtinuta in starea urmatoare

Algoritmi de invatare
Invatarea SARSA

- Nu se foloseste recompensa maxima a starii urmatoare ca la Q.
- In schimb, se alege o noua actiune (si astfel o noua recompensa se obtine) folosind aceeasi strategie.
- Numele “sarsa” vine de la faptul ca actualizarile sunt realizate folosind tuplul Q(s, a, r, s’, a’)
  - s si a sunt starea si actiunea initiale
  - r este recompensa obtinuta in starea s daca se ia actiunea a
  - s’ si a’ sunt noua pereche stare-actiune

Algoritmi de invatare
Invatarea SARSA

1. Se initializeaza **Q(s, a)** in mod aleator sau cu zero
2. Repeta
   1. Initializeaza starea **s**
   2. Repeta
      1. Alege actiunea **a** in functie de strategia aleasa (ε-greedy sau Softmax)
      2. Alege actiunea **a'** din **s'** folosing aceeasi strategie
      3. $Q(s, a) = Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
      4. **s = s'; a = a';**
   3. Pana cand s este stare terminala
3. Pana cand se intalneste conditia de oprire (un numar de iteratii)

Exemplu

- Exista 6 incaperi etichetate de la A la F.
- Agentul care va trebui sa invete din acest mediu este soarecele, care se afla in imagine in incaperea D.
- Soarecele este pus in diverse incaperi si trebuie sa se adapteze in asa fel incat sa ajunga la cascaval si, evident, sa nu ajunga la pisica.
- Cascavalul si pisica nu isi pot schimba incaperile de care apartin, dar soarecele da.

Aplicati algoritmul bazat pe invatarea Q pentru aceasta problema.

Rezolvarea poate fi gasita aici.