Uvod u umjetnu inteligenciju

13. Podržano učenje

Bojana Dalbelo Bašić, Marko Čupić, Jan Šnajder

Sveučilište u Zagrebu
Fakultet elektrotehike i računarstva

Ak. god. 2021./2022.

v3.10

Vrste učenja - podsjetnik

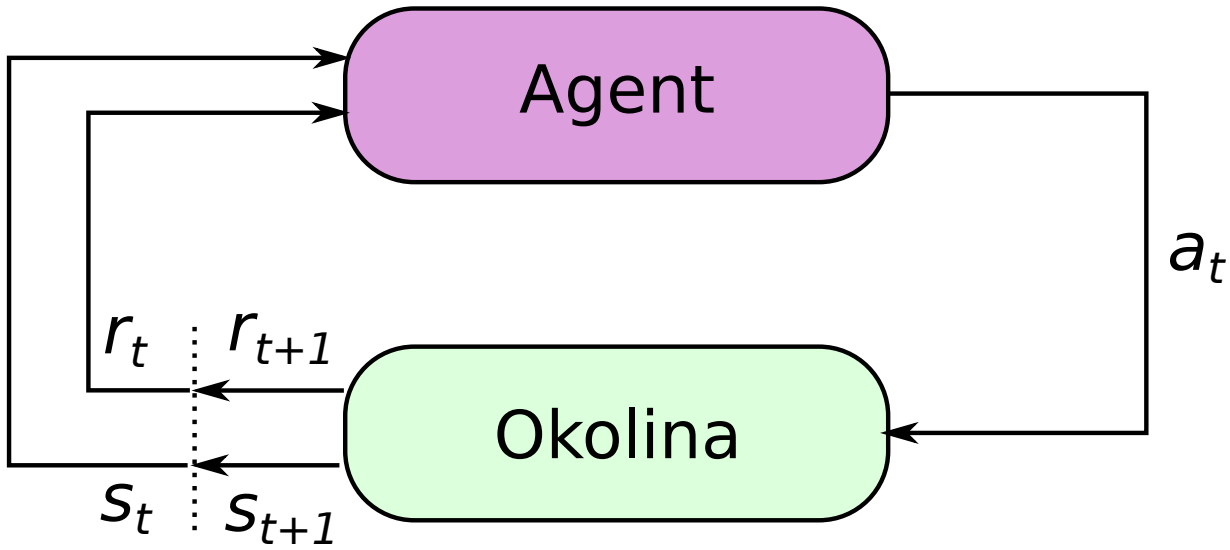
- *Nenadzirano učenje* - primjerci su oblika $(\vec{x})$ - grupiranje
- *Nadzirano učenje* - primjerci su oblika $(\vec{x}) \rightarrow (\vec{y})$ - klasifikacija, funkcijska aproksimacija
  - $\vec{x}$ je čitav primjerak; u slučaju igranja igre, to bi bili svi odigrani potezi tijekom igre
  - $\vec{y}$ je oznaka razreda ili konačna vrijednost funkcije; u slučaju igranja igre, to bi bio osvojeni broj bodova nakon što je igra završila
  - U slučaju igre, problem je utvrditi koji je od poteza tijekom igre koliko zaslužan za bodove koji su osvojeni na kraju.
- Ljudi ne uče tako! Primjerice, malo dijete koje uči hodati brzo počinje trčati - nagrada je ushićenje. No brzo nakon toga, slijedi pad i bol, pa dijete nauči biti opreznije. Učenje se odvija na temelju dobivene nagrade tijekom akcije ili s nekon odgodom.
- *Podržano učenje* oponaša ovakav način učenja.

Sadržaj

Model

Model na kojem je primjenjivo podržano učenje sastoji se od:

- *Agenta* - to je programski sustav koji uči optimalno ponašanje u okolini na temelju viđenih stanja okoline s (engl. *state*) i primljenih nagrada r (engl. *reward*)
- *Okoline* - agent u njoj poduzima akciju a_t na temelju koje mjenja stanje okoline u s_{t+1} te prima nagradu r_{t+1} od okoline



Slika: Model okoline koji razmatramo

Okolina

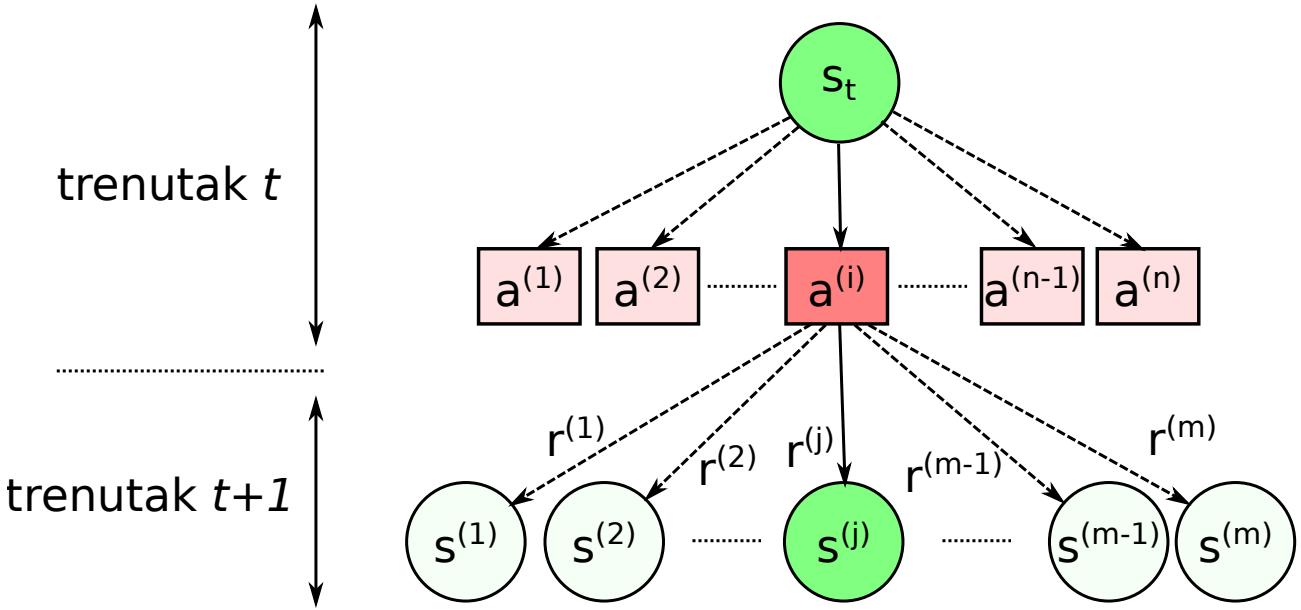
Okolina u kojoj agent djeluje može biti deterministička ili pak stohastička.

- Kada agent poduzme neku akciju, okolina će dojaviti kako je ta akcija promijenila stanje okoline te koliku nagradu agent prima
- Oboje mogu biti stohastički procesi - moguće je da iz istog stanja s istom akcijom agenta okolina ne prelazi uvijek u isto sljedeće stanje, kao i da svaki puta daje neku drugačiju nagradu.

Razmatramo samo okoline koje zadovoljavaju svojstvo Markovljevog procesa odlučivanja, tj. kod kojih je uvjetna vjerojatnost prelaska okoline u sljedeće stanje odrediva samo na temelju podataka dostupnih u trenutnom stanju:

$$p(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, s_{t-2}, a_{t-2}, \ldots, s_0, a_0) = p(s_{t+1}|s_t, a_t)$$

Okolina



trenutno stanje: s_t

moguće akcije u tom stanju

Moguća nova stanja i nagrade kao rezultat izvođenja akcije a_i

Slika: Djelovanje akcije na okolinu

Djelovanje agenta

Način na koji agent odabire akciju naziva se politika (engl. *policy*) i označava slovom π.

- Politika agenta je funkcija koja preslikava stanje s u odabranu akciju a. Jedan od mogućih zadataka podržanog učenja jest naučiti optimalnu politiku za problem s kojim je agent suočen.

Zadaća agenta je birati akcije na način koji maksimizira sumu primljenih nagrada, odnosno:

$$R_t = r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} + \gamma^3 \cdot r_{t+4} + \cdots + \gamma^{T-t-1} \cdot r_T = \sum_{k=0}^{T-t-1} \gamma^k \cdot r_{t+k+1} \tag{1}$$

pri čemu je γ faktor kojim se balansira između želje da se maksimizira isključivo trenutna nagrada poteza (za $\gamma = 0$) odnosno suma svih dugoročno primljenih nagrada (uz $\gamma = 1$).

Djelovanje agenta

Vrijedi:

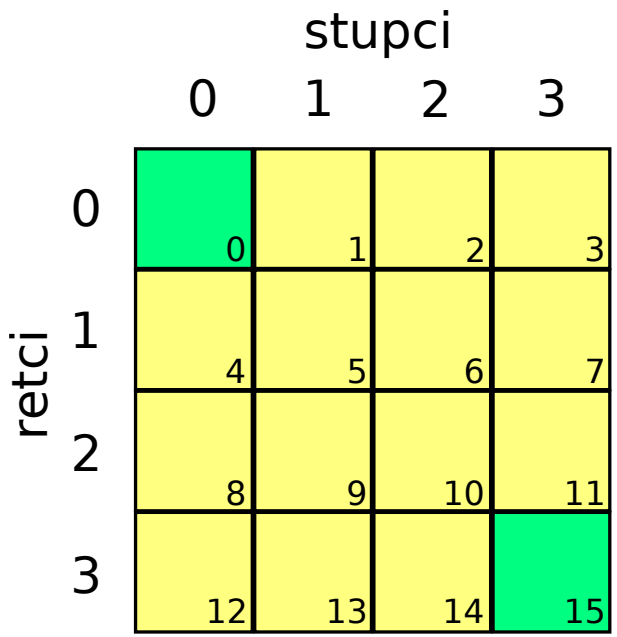
$$\begin{aligned} R_t &= r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} + \gamma^3 \cdot r_{t+4} + \cdots + \gamma^{T-t-1} \cdot r_T \\ &= r_{t+1} + \gamma \cdot \left(r_{t+2} + \gamma^1 \cdot r_{t+3} + \gamma^2 \cdot r_{t+4} + \cdots + \gamma^{T-t-2} \cdot r_T \right) \\ &= r_{t+1} + \gamma \cdot R_{t+1} \end{aligned}$$

Problem koji agent rješava može biti:

- *epizodički* - ako je gotov u konačnom broju koraka, što podrazumijeva da u okolini mora postojati terminalno stanje
- *kontinuirani* - ako agent akcije može obavljati do u beskonačnost

Primjer: Rešetkasti svijet 1

Robot je u rešetkastom svijetu. Izlazi su zeleni kvadratići. Svaki korak robota košta 1 jedinicu energije (nagrada je -1). Neka je $\gamma = 1$.



Slika: Rešetkasti svijet 1

Uz zadanu politiku želimo odrediti ukupnu očekivanu nagradu koju će robot dobiti ako krene iz svake od ćelija.

Izvođenje jedne epizode

```
public static <S,A> double play(DiscreteEnvironment<S, A>
    world, Policy<S, A> policy, S startState, double gamma)
    {

  world.setCurrentState(startState);
  double totalReward = 0;
  double scaler = 1;

  while(!world.isFinished()) {
    S currentState = world.getCurrentState();
    A selectedAction = policy.pickAction(currentState);
    double reward = world.applyAction(selectedAction);
    totalReward += scaler * reward;
    scaler *= gamma;
  }

  return totalReward;
}
```

Primjer: Rešetkasti svijet 1

Ako je politika "slučajno s jednolikom razdiobom odaberi smjer pomaka":

```
Vrijednosti stanja:
   0.000  -13.958  -20.076  -21.990
 -14.002  -17.986  -19.875  -20.009
 -20.136  -19.973  -17.992  -14.000
 -22.142  -19.971  -13.889    0.000
```

Ako je politika "u 70% gore, u preostalih 10%+10%+10% lijevo/desno/dolje":

```
Vrijednosti stanja:
   0.000  -29.911  -48.873  -58.008
  -5.658  -30.672  -48.666  -57.477
 -10.724  -31.219  -47.268  -51.167
 -14.545  -31.685  -41.595    0.000
```

Funkcije vrijednosti i vrijednosti akcije

Funkcija vrijednosti pod politikom π:

$$v_\pi(s) = \mathbb{E}_\pi[R_t|s_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{k+t+1}|s_t = s\right] \tag{2}$$

Funkcija vrijednosti akcije pod politikom π:

$$q_\pi(s,a) = \mathbb{E}_\pi[R_t|s_t = s, a_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{k+t+1}|s_t = s, a_t = a\right] \tag{3}$$

Vrijedi:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)\left[r(s,a,s') + \gamma \cdot v_\pi(s')\right] \tag{4}$$

odnosno ako uočimo da unutarnja suma zapravo predstavlja q-vrijednosti:

$$v_\pi(s) = \sum_a \pi(a|s) \cdot q_\pi(s,a) \tag{5}$$

Vrednovanje politike

Ako na raspolaganju imamo model politike i model okoline funkciju vrijednosti pod politikom možemo odrediti puno učinkovitije postupkom vrednovanja politike (engl. *policy evaluation*).
Postupak je iterativni.

1. Inicijaliziraj vrijednosti na početne (primjerice 0) u polju v. Terminalna polja moraju biti 0.
2. Ponavljaj dok je ukupna promjena veća od nekog praga
   1. Koristeći polje v i izraz (4) nanovo izračunaj vrijednost za svako stanje i rezultat pohrani u v'
   2. $v \leftarrow v'$

Postupak dokazano konvergira i učinkovit je.

Primjer: Rešetkasti svijet 1 - vrednovanje politike

Ako je politika "slučajno s jednolikom razdiobom odaberi smjer pomaka":

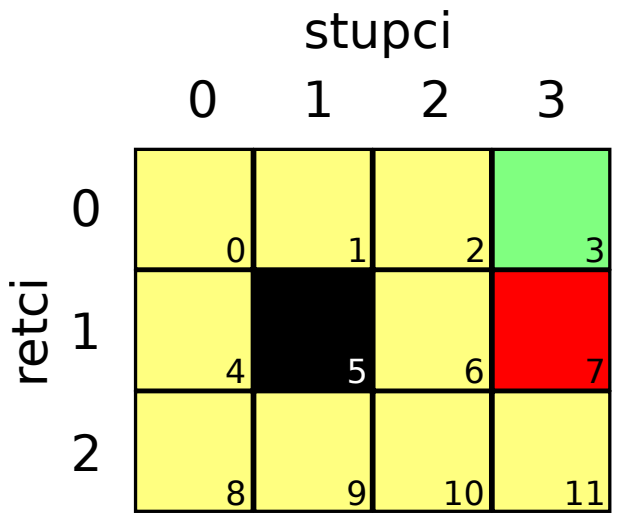
```
Vrijednosti stanja:
   0.000  -14.000  -20.000  -22.000
 -14.000  -18.000  -20.000  -20.000
 -20.000  -20.000  -18.000  -14.000
 -22.000  -20.000  -14.000    0.000
```

Ako je politika "u 70% gore, u preostalih 10%+10%+10% lijevo/desno/dolje":

```
Vrijednosti stanja:
   0.000  -29.784  -48.928  -58.205
  -5.670  -30.423  -48.796  -57.481
 -10.607  -31.280  -47.557  -51.103
 -14.490  -31.675  -41.620    0.000
```

Primjer: Rešetkasti svijet 2

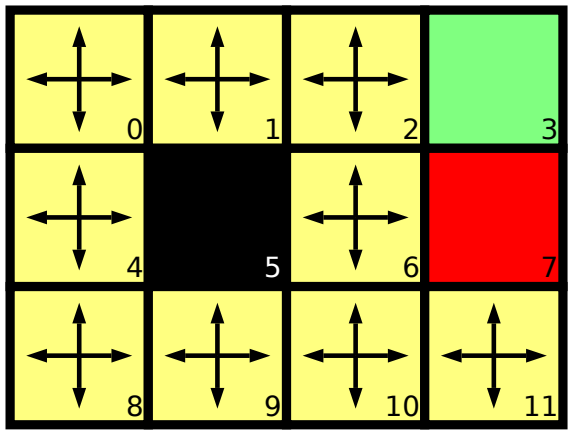
Svijet je nedeterministički. Zadana akcija provodi se uspješno u 80% slučajeva, a u 10%+10% provodi se jedna od "ne-suprotnih" akcija. Npr. akcija "gore" će u 80% slučajeva pomaknuti robota gore, u 10% slučajeva lijevo i u 10% slučajeva desno. Polje 5 je zid, polje 7 je oganj koji uništava robota (nagrada -1), polje 3 je završno (nagrada +1). Svi ostali prijelazi daju nagradu r (takozvanu nagradu življenja, engl. *living reward*, negativnog iznosa).

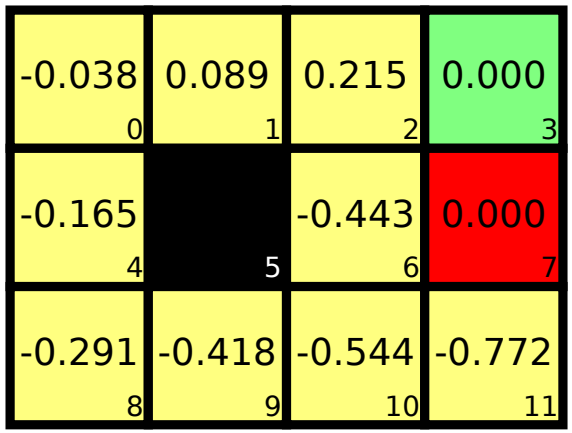


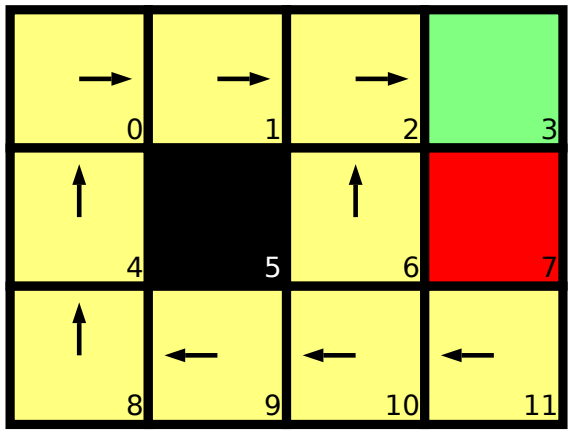
Slika: Rešetkasti svijet 2

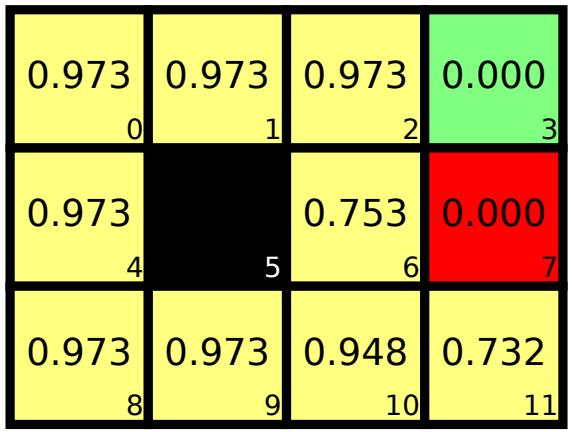
Primjer: Rešetkasti svijet 2

Funkcija vrijednosti pod politikama uz $r = 0$ i $\gamma = 1$
(lijevo: politika; desno: funkcija vrijednosti):









Bellmanove jednadžbe

Ako se agent nalazi u stanju s, kako bi igrao optimalno, treba odabrati onu akciju koja maksimizira sumu nagrada koje će dobiti uz odabir te akcije i dalje ponovno igrajući optimalno. No tada za funkciju vrijednosti vrijede Bellmanove jednadžbe:

$$v^*(s) = \max_a \left(\sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma \cdot v^*(s') \right] \right) \tag{6}$$

$$v^*(s) = \max_a q^*(s,a) \tag{7}$$

pri čemu smo oznakom $v^*(s)$ označili optimalne vrijednosti stanja, a oznakom $q^*(s,a)$ optimalne q-vrijednosti.

Bellmanove jednadžbe

Optimalne vrijednosti stanja odnosno q-vrijednosti su formalno definirane ovako:

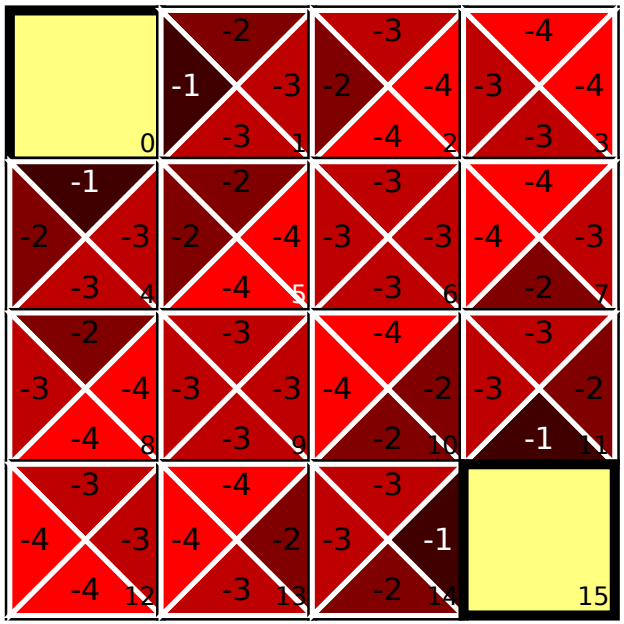
$$v^*(s) = \max_\pi v_\pi(s)$$

$$q^*(s,a) = \max_\pi q_\pi(s,a)$$

gdje maksimum pretražujemo po svim mogućim politikama. Politiku π za koju se taj maksimum postiže označavat ćemo s π^* i zvati optimalnom politikom.

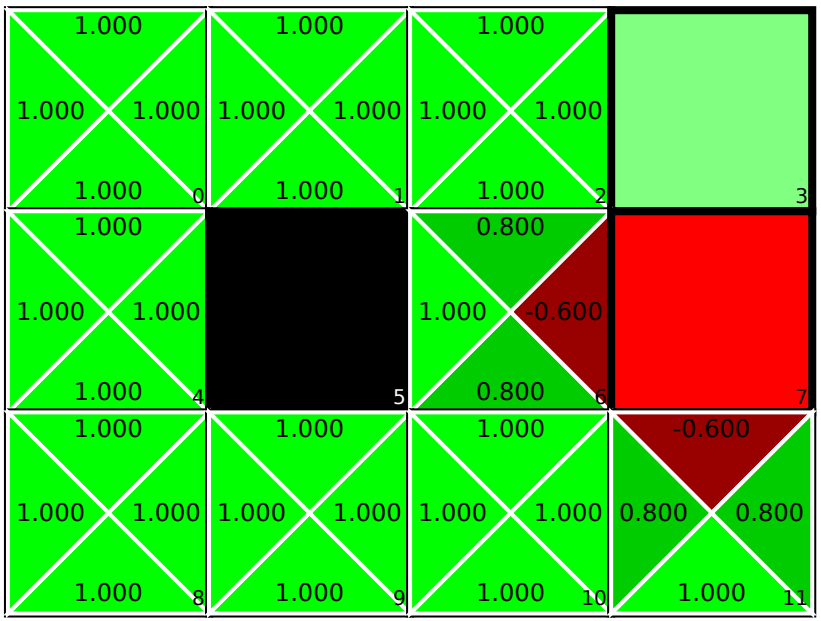
Rešetkasti svijet 1: Iteracija vrijednosti

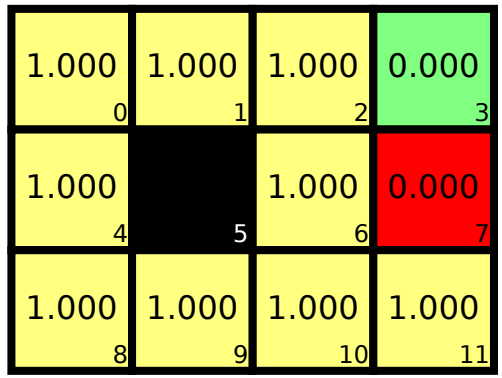
Iteracija vrijednosti (engl. *Value iteration*) je postupak sličan vrednovanju politike, samo što se vrijednosti ažuriraju prema Bellmanovoj jednadžbi (6). Lijevo: q-vrijednosti; desno: funkcija vrijednosti.



Rešetkasti svijet 2: Iteracija vrijednosti

Iteracija vrijednosti (engl. *Value iteration*) je postupak sličan vrednovanju politike, samo što se vrijednosti ažuriraju prema Bellmanovoj jednadžbi (6). Lijevo: q-vrijednosti; desno: funkcija vrijednosti.





Otkrivanje optimalne politike

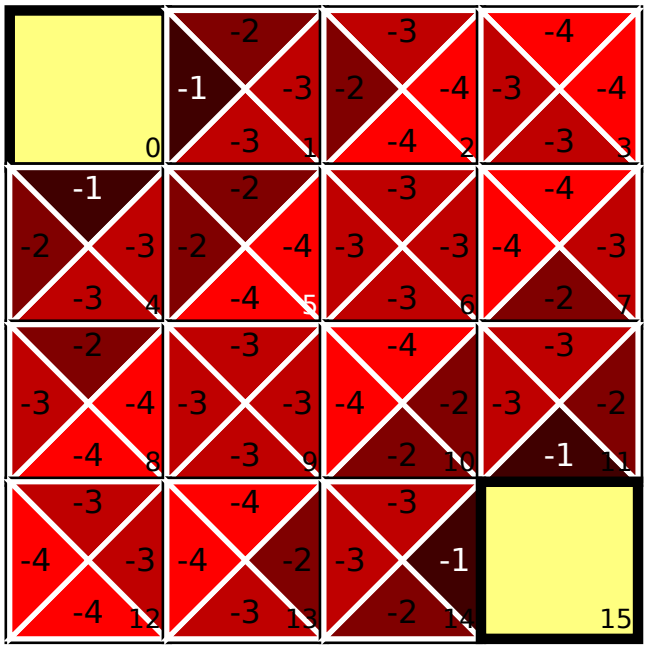
Jednom kad smo odredili optimalne iznose funkcije vrijednosti, možemo rekonstruirati i optimalnu politiku:

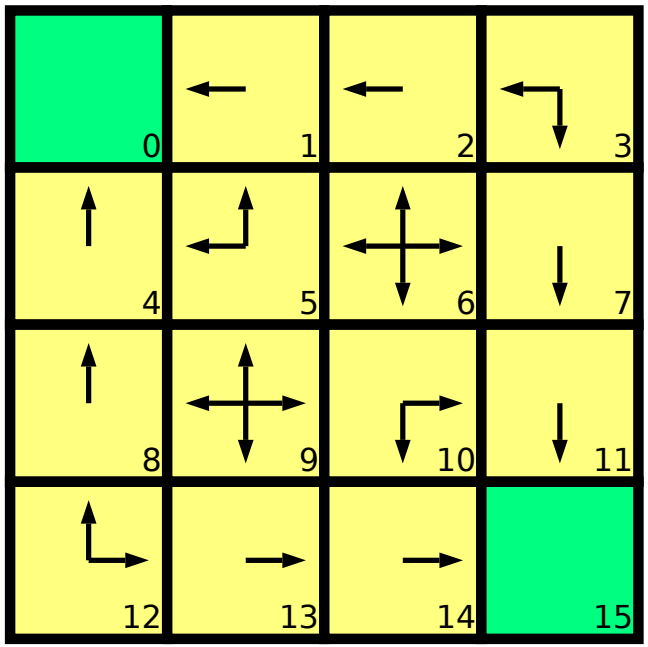
$$\pi^*(s) = \arg\max_a \left(\sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma \cdot v^*(s') \right] \right) \tag{8}$$

$$\pi^*(s) = \arg\max_a \left(q^*(s,a) \right) \tag{9}$$

Rešetkasti svijet 1: Optimalna politika

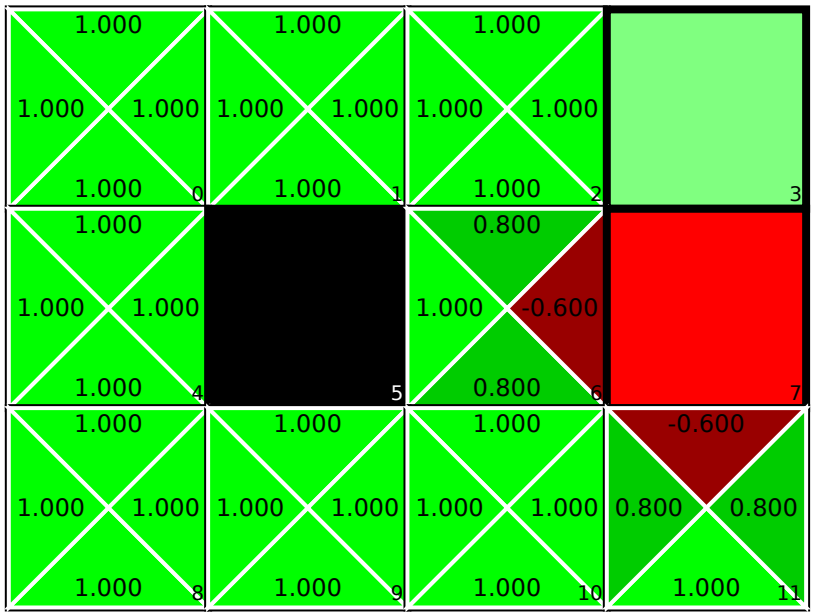
Optimalna politika očitana iz naučenih q-vrijednosti postupkom iteracije vrijednosti.

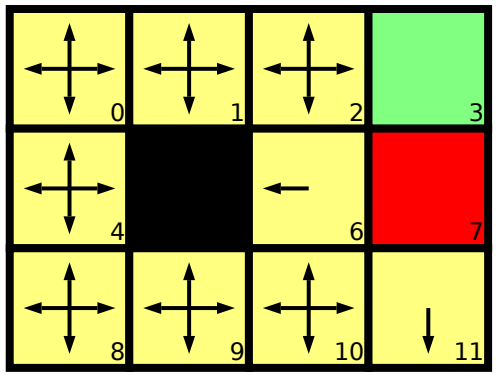




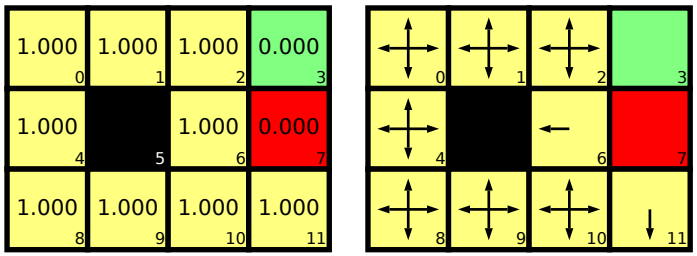
Rešetkasti svijet 2: Optimalna politika

Optimalna politika očitana iz naučenih q-vrijednosti postupkom iteracije vrijednosti.

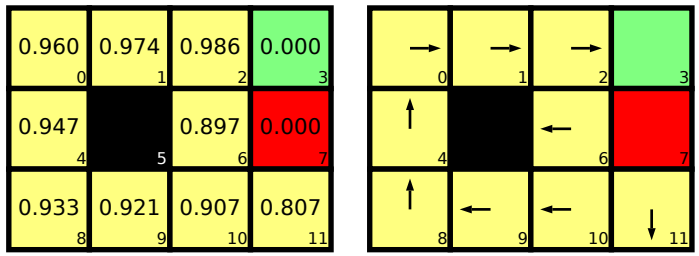




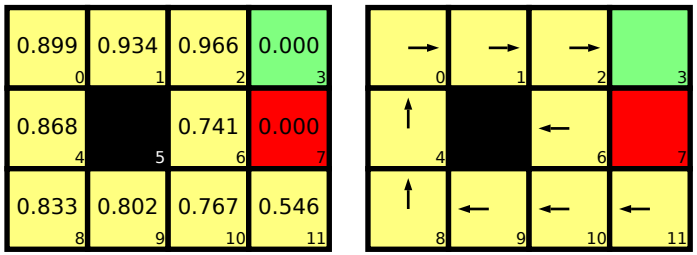
Rešetkasti svijet 2: Utjecaj nagrade življenja



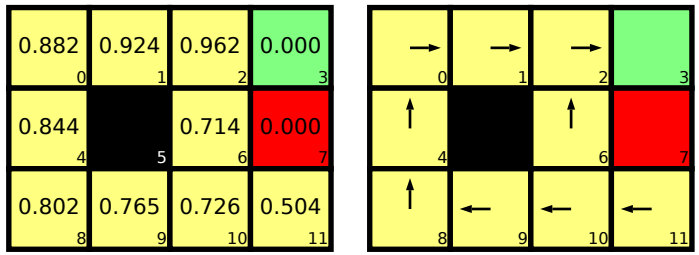
r=0, γ=1



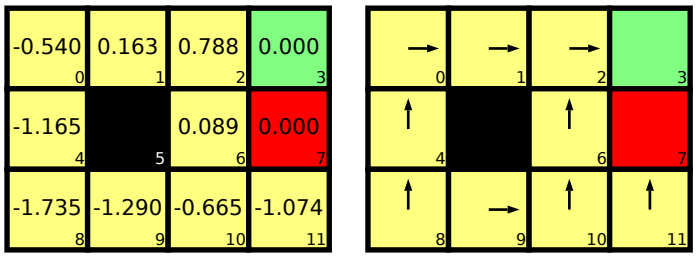
r=-0.01, γ=1



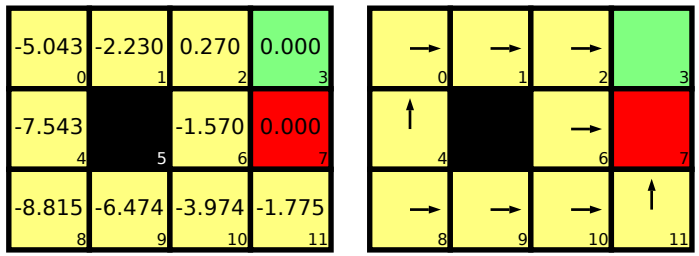
r=-0.025, γ=1



r=-0.03, γ=1



r=-0.5, γ=1



r=-2, γ=1

Rekapitulacija

1. Ako imamo model okoline i model politike, možemo odrediti funkciju vrijednosti
   - simulacijom igre: neučinkovito
   - vrednovanjem politike: učinkovitije
2. Ako imamo model okoline, možemo odrediti optimalne iznose funkcije vrijednosti iteracijom vrijednosti i tada rekonstruirati optimalnu politiku.
   - Postoji alternativni način *Iteracijom politike*, no taj postupak nećemo obraditi.

No što ako nemamo model okoline?

Sadržaj

Učenje tijekom epizode

Sada ćemo razmotriti slučaj kada agent prolazi epizodu po epizodu, i učenje obavlja izravno tijekom interakcije s okolinom čiji nema model. Obradit ćemo algoritam Q-učenja koji uči q-vrijednosti na temelju kojih je potom moguće izravno odrediti optimalnu politiku.
Algoritam učenja je iterativan, i kreće od početne inicijalizacije q-vrijednosti (primjerice, sve vrijednosti mogu biti 0). Agent prolazi kroz svaku epizodu i u svakom koraku koristi ϵ-pohlepnu politiku koju istovremeno i uči.
ϵ-pohlepna znači da u ϵ posto slučajeva agent odabire slučajno akciju, a u $(1-\epsilon)$ posto slučajeva bira najbolju akciju prema trenutno naučenoj politici. U ranim fazama ϵ bi trebao biti velik kako bi agentu omogućio da hrabro istražuje; kasnije ϵ treba smanjivati kako bi se agent više oslanjao na naučenu politiku.

Q-učenje

Algoritam Q-učenja izravno uči q-vrijednosti koje su definirane kao:

$$q(s,a) = r(s,a,s') + \gamma \cdot v_\pi(s')$$

Koristi činjenicu da je:

$$v_\pi(s') = \max_{a'} q(s',a')$$

pa ažuriranje q-vrijednosti radi prema izrazu:

$$q(s,a) \leftarrow (1-\alpha) \cdot q(s,a) + \alpha \cdot \left(r(s,a,s') + \gamma \cdot \max_{a'} q(s',a') \right) \qquad (10)$$

Kod ovog pristupa, optimalna vrijednost stanja s' procjenjuje se na temelju svih procijenjenih q-vrijednosti akcija u stanju s'.

Q-učenje

α je stopa učenja i kreće se između 0 i 1. Ako je 0, nema ažuriranja vrijednosti; ako je 1, bitna je samo novoizračunata vrijednost. Vrijednosti između toga rade linearnu interpolaciju, što možemo pokazati ovako:

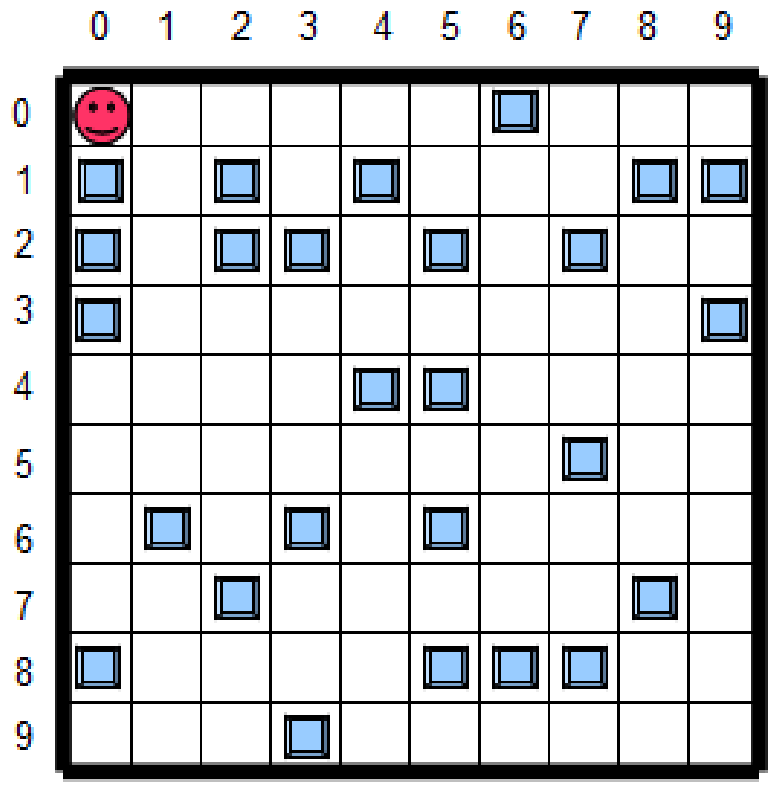
$$(1-\alpha)\cdot q(s,a)+\alpha\cdot\left(r(s,a,s')+\gamma\cdot\max_{a'}q(s',a')\right)=$$

$$q(s,a)+\alpha\cdot\left\{\left(r(s,a,s')+\gamma\cdot\max_{a'}q(s',a')\right)-q(s,a)\right\}$$

Primjer: Robot Robby

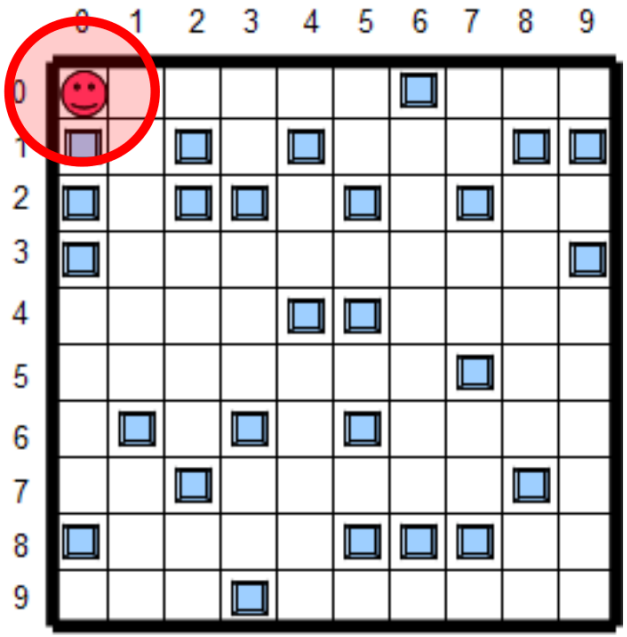
Prisjetimo se robota Robby:

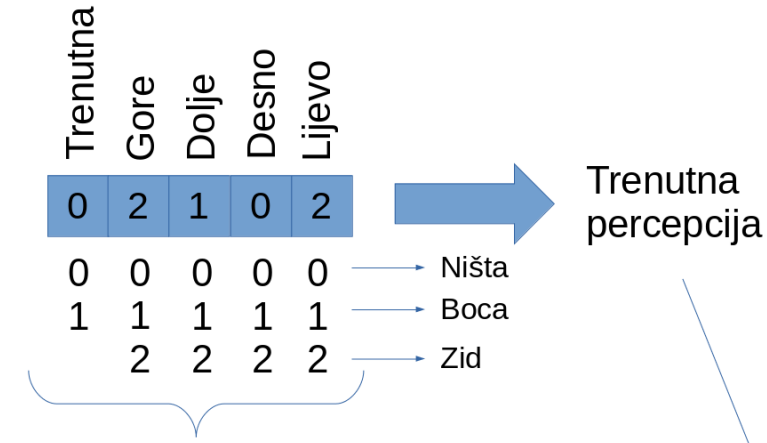
- zadaća je pokupiti boce u 150 koraka
- na temelju percepcije okoline robot treba odabrati akciju koju će napraviti



Primjer: Robot Robby

Percepciju Robbyja možemo kodirati cijelim brojem - postoje ukupno 162 različite percepcije:





$$2 \cdot 3^3 + 1 \cdot 3^2 + 2 \cdot 3^0 = 65_{10}$$

Primjer: Robot Robby

Postoji 7 akcija koje Robby može poduzeti:

- 0: ne radi ništa
- 1: pokupi bocu
- 2: idi gore
- 3: idi dolje
- 4: idi desno
- 5: idi lijevo
- 6: idi slučajno

U ovom primjeru stanja su različite percepcije. Stoga imamo 162 različita stanja, i u svakom mogućih 7 akcija. Pamtimo qTable[162][7].

Primjer: Robot Robby

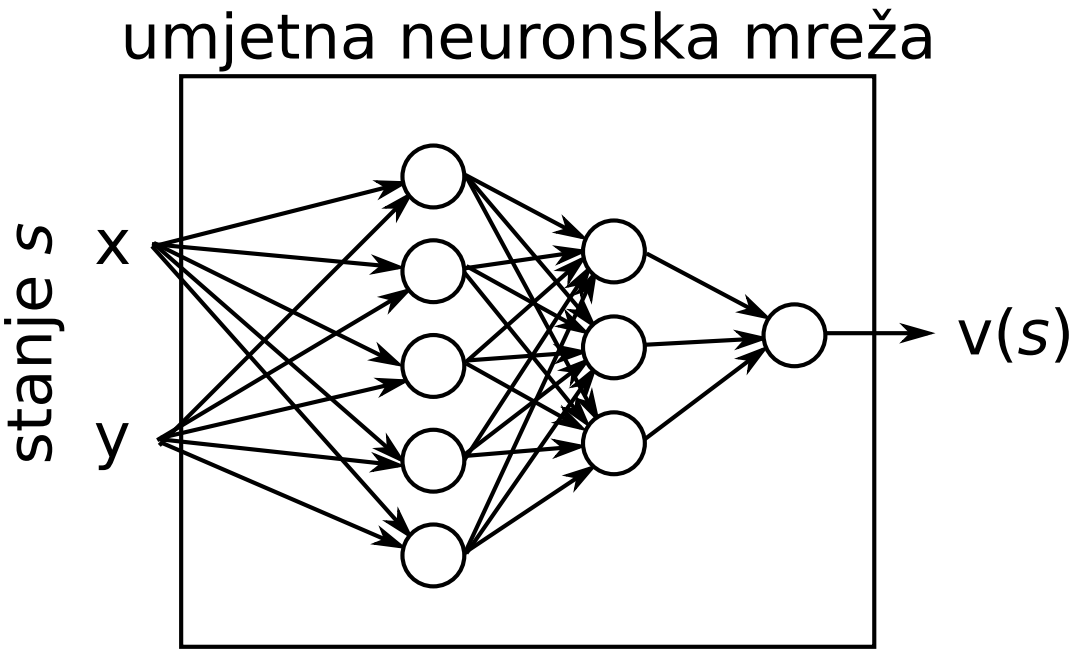
- Učenje provodimo kroz 100 000 epoha.
- U svakoj epohi prolazimo kroz 20 svjetova.
- Svjetove regeneriramo svakih 101 epohu (kako bi spriječili prenaučenost).
- Parametar α linearno mijenjamo od 0.1 do 0.001 tijekom prvih 60% epoha i dalje ostaje fiksan na minimumu.
- Parametar ϵ linearno mijenjamo od 0.1 do 0.001 tijekom prvih 60% epoha i dalje ostaje fiksan na minimumu.
- Parametar $\gamma = 0.9$
- Svijet je dimenzija 10×10 i popunjenost bocama je 35%.
- Nagrade: +10 za skupljenu bocu, -5 ako nema boce, -10 ako se zabije u zid, nagrada življenja je 0.

Primjer: Robot Robby

(DEMONSTRACIJA)

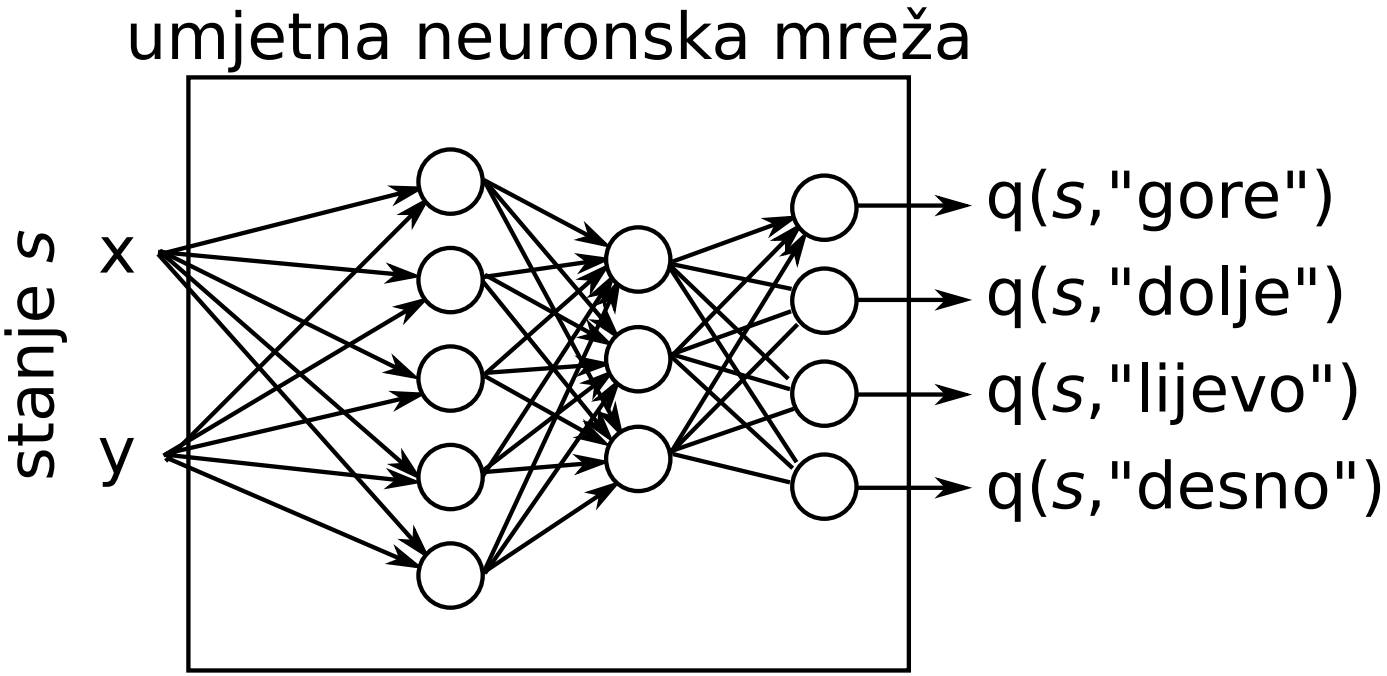
Veliki i/ili kontinuirani prostori stanja

Ako su prostori stanja veliki i/ili kontinuirani, tada funkciju vrijednosti kao niti q-vrijednosti ne možemo tabelirati (odnosno čuvati u polju u memoriji).



Veliki i/ili kontinuirani prostori stanja

Radimo li s q-vrijednostima, tada bismo koristili aproksimacijski model bi imao više izlaza: za svaku od akcija postojao bi jedan izlaz koji bi generirao q-vrijednost odgovarajuće akcije. Slika u nastavku ovo ilustrira, ponovno na primjeru rešetkastog svijeta gdje su moguće akcije "gore", "dolje", "lijevo" i "desno".



Veliki i/ili kontinuirani prostori stanja

U oba slučaja umjetna neuronska mreža koristi se za aproksimaciju funkcije vrijednosti odnosno q-vrijednosti.
Ažuriranje se potom radi na uobičajeni način koristeći unatražni prolaz.

Zaključak

- Dali smo samo najelementarniji uvod u podržano učenje
- Postoji još niz algoritama koji se danas koriste i koji rješavaju nedostatke osnovnih algoritama
- Danas se podržano učenje spaja s dubokim modelima koji služe za aproksimaciju funkcija koje se uče