

Učenje podrškom

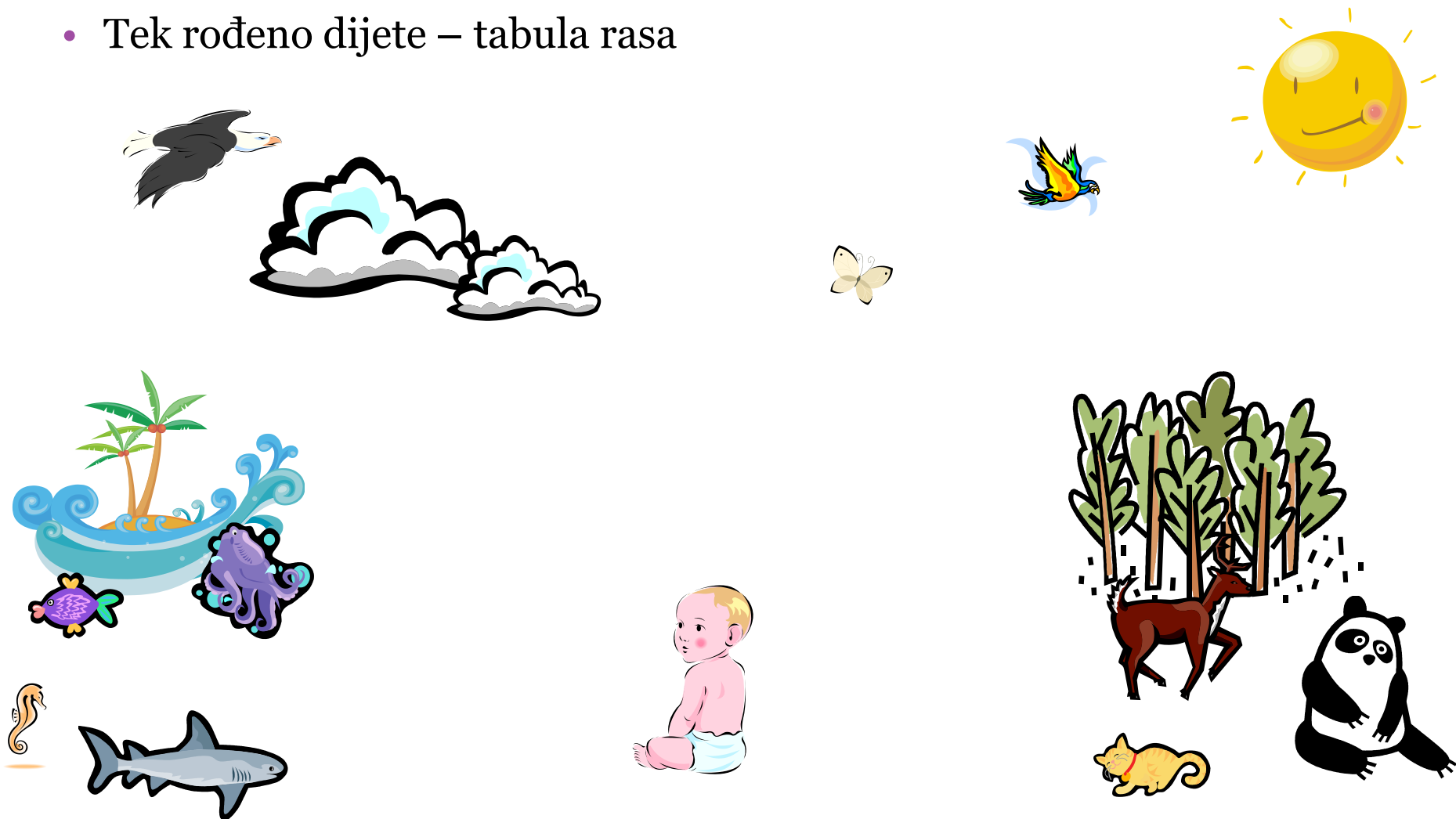
Matko Bošnjak, 2011

Rekapitulacija

- Nadzirano učenje (eng. supervised learning)
 - Učitelj nam daje primjere i njihove vrijednosti
 - Učimo funkciju klasifikacije ili regresije
- Nenadzirano učenje (eng. unsupervised learning)
 - Nema kritičara / učitelja
 - Sustav se samoorganizira
 - Učimo pravilnosti u podacima
- Učenje podrškom (eng. reinforcement learning)
 - Učenje se odvija kroz interakciju s okolišem

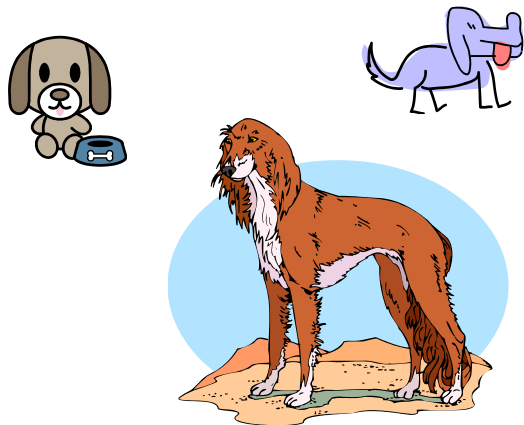
Učenje bez nadzora - analogija

- Tek rođeno dijete – tabula rasa



Učenje s nadzorom - analogija

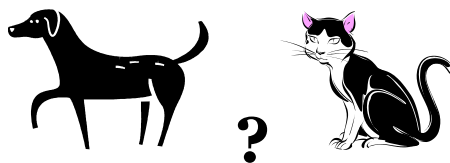
- Nakon što ovladamo komunikacijom...



PAS



MAČKA



Učenje s podrškom - analogija

- Zatim nam se život dodatno zakomplicira...

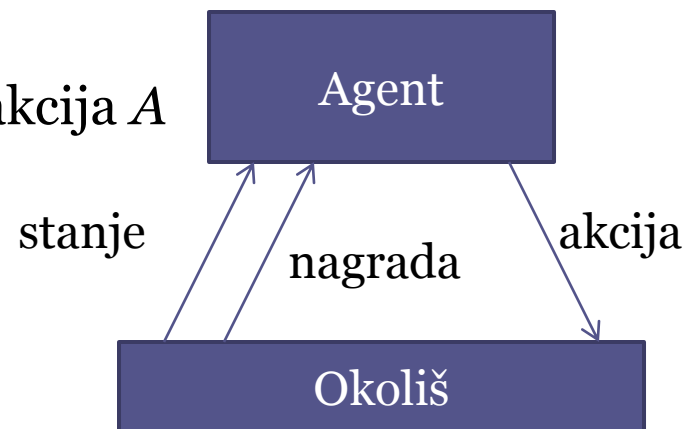


Uvod

- Agent se nalazi u interakciji s dinamičkim okolišem

- Okoliš je opisan skupom stanja S
- Agent može učiniti neku akciju iz skupa akcija A

- Izvođenjem akcije a u stanju s agent dobiva trenutnu nagradu r (vrijednost prelaska stanje-akcija)

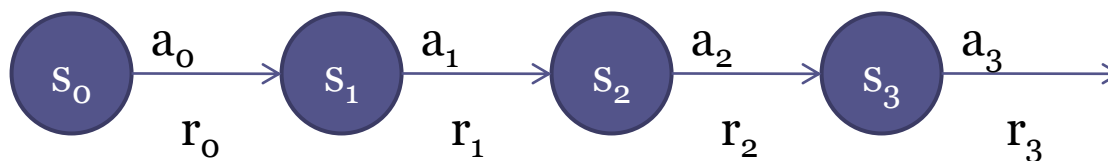


- Cilj: optimizirati interakciju s okolišem kroz proces pokušaja i pogrešaka
- Prirodan pristup učenju

Uvod

- Cilj
 - optimizirati interakciju s okolišem kroz proces pokušaja i pogrešaka
 - izvesti seriju akcija, proučiti posljedice, naučiti kontrolnu strategiju
 - naučiti kontrolirati sekvencijalne procese
 - maksimizirati kumulativnu nagradu
- Akcije mogu imati determinističke i nedeterminističke ishode
- Agent može, a ne mora znati nešto o efektima svojih akcija
- Zadatak agenta je naučiti kontrolnu strategiju

$$\pi : S \rightarrow A$$



Problemi

- Odgođeno nagrađivanje
 - Jedina informacija su trenutne nagrade
 - Temporal credit assignment – koje akcije dovode do eventualne nagrade
- Pretraživanje
 - Agent utječe na distribuciju primjera za učenje sekvencom akcija koje odabire
 - Koja strategija dovodi do najefektivnijeg učenja?
 - Tradeoff pretraživanja novog i iskorištavanja starog
- Djelomično vidljiva stanja
 - Senzori agenta često pružaju djelomičnu informaciju
 - Nekada je potrebno upotpuniti tu informaciju
- Cjeloživotno učenje
 - Od agenta se često očekuje učenje nekoliko povezanih zadataka unutar istog okruženja
 - Korištenje prethodnog iskustva

Primjer

- Agent/robot vozi bicikl
- U jednom trenutku se nađe pod kutem od 45° udesno
 - Desno \rightarrow pad (negativna podrška)
- Ponovno ista situacija
 - Ovaj puta lijevo \rightarrow pad
 - Stanje 45° vodi u propast kako god
- U jednom trenutku se nađe pod kutem od 40° udesno
 - Skrenuti lijevo?
 - Skrenuti desno?
 - Lijevo \rightarrow Stanje 45°
- Robot će u konačnici naučiti jednostavno ne dolaziti više u stanja koja rezultiraju padom niti poduzimati akcije koje bi vodile takvim stanjima
- Kako bi ovo nadzirano naučili?

Markovljevi procesi odlučivanja

- Definiramo generalnu formulaciju problema učenja baziranu na Markovljevim procesima odlučivanja (eng. Markov Decision Process, MDP)
 - Agent percipira skup distinktivnih stanja S te na raspolaganju ima skup akcija A
 - U diskretnom vremenskom trenutku t , agent u stanju s_t odabire i izvodi akciju a_t
 - Okoliš odgovara nagradom $r_t = r(s_t, a_t)$ te proizvodi naredno stanje $s_{t+1} = \delta(s_t, a_t)$
 - δ i r su dio okoliša, uopće ne moraju biti poznate agentu
 - Ovisе samo o trenutnom stanju i akciji! (Markovljevo svojstvo)
- Proučavat ćemo samo konačne S i A

Definicija zadatka

- Zadatak agenta je naučiti strategiju $\pi : S \rightarrow A$
- Za odabir iduće akcije a_t na temelju trenutnog stanja $s_t : \pi(s_t) = a_t$
- Koju strategiju želimo naučiti?
 - Onu koja donosi najveću moguću kumulativnu nagradu tokom vremena
- Kumulativna vrijednost $V^\pi(s_t)$ koju postiže proizvoljna strategija π iz proizvoljnog početnog stanja s_t glasi:

$$\begin{aligned} V^\pi(s_t) &\equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \\ &\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i} \end{aligned}$$

- $0 \leq \gamma < 1$ konstanta koja određuje relativnu vrijednost odgođene prema trenutnoj nagradi
- Razumno je umanjiti (eksponencijalno) značaj budućih nagrada prema trenutnim jer preferiramo nagradu sada, a ne kasnije

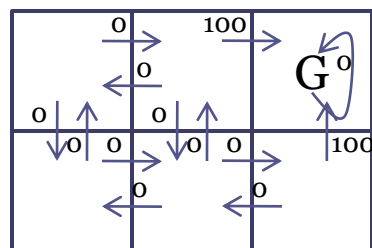
Definicija zadatka

- Želimo strategiju koja maksimizira $V^\pi(s)$ za sva stanja s
- Optimalna strategija:

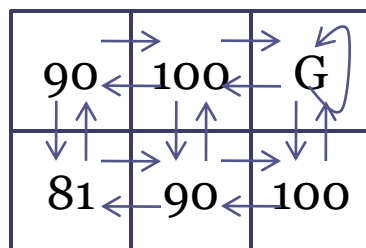
$$\pi^* \equiv \arg \max_{\pi} V^\pi(s) \quad , (\forall s)$$

- Funkcija vrijednosti optimalne strategije π : $V^*(s)$
- $V^*(s)$ daje maksimalnu kumulativnu nagradu (s umanjenim značajem budućih nagrada) koju agent može postići počevši od stanja s

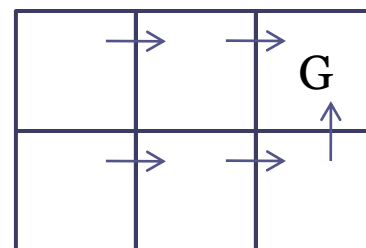
Primjer



$r(s,a)$



$V^*(s)$



Jedna moguća optimalna strategija

- G – apsorbirajuće stanje, stanje u kojemu agent ostaje
- Optimalna strategija vodi agenta najkraćim putem u G
- $\gamma = 0.9$

Q learning

- Kako agent može naučiti optimalnu strategiju π^* za proizvoljni okoliš?
 - Teško je direktno učiti funkciju $\pi^* : S \rightarrow A$, primjeri za učenje ne dolaze u obliku $\langle s, a \rangle$ (nadzirano učenje)
 - Sva informacija koju imamo je sekvenca trenutnih nagrada $r(s_i, a_i)$ za $i=0, 1, 2, \dots$
 - Jednostavnije je naučiti numeričku evaluacijsku funkciju definiranu nad stanjima i akcijama nego implementirati optimalnu strategiju kao funkciju
- Koju evaluacijsku funkciju naučiti?
 - Očiti izbor je V^*
 - Agent treba preferirati stanje s_1 nad stanjem s_2 kadgod je $V^*(s_1) > V^*(s_2)$
 - Buduća kumulativna nagrada je veća u s_1
 - Strategija mora odabirati akcije, ne stanja!
 - V^* se može koristiti za odabir akcije

Q learning

- Optimalna akcija u stanju s je akcija a koja maksimizira sumu trenutne nagrade $r(s, a)$ i vrijednosti V^* neposrednog stanja, prigušenog za γ

$$\pi^*(s) = \arg \max_a [r(s, a) + \gamma V^*(\delta(s, a))]$$

- Agent može doći do optimalne strategije učeći V^* uz uvjet da zna funkciju r i funkciju prijelaza stanja δ
- Nažalost, to je problem...agent uopće ne mora znati te funkcije (obično i ne zna!)
 - Podrazumijeva savršenu predikciju sljedećeg rezultata (nagrade i stanja sljedbenika)
 - U mnogim slučajevima to je nemoguće
- Koju funkciju učiti u općem slučaju?
 - Q funkciju

Q funkcija

- Definirajmo evaluacijsku funkciju $Q(s, a)$
 - Vrijednost jednaka maksimalnoj prigušenoj kumulativnoj nagradi koja se može doći iz stanja s čineći akciju a
- Trenutna nagrada + prigušena vrijednost koju dobivamo slijedeći optimalnu strategiju

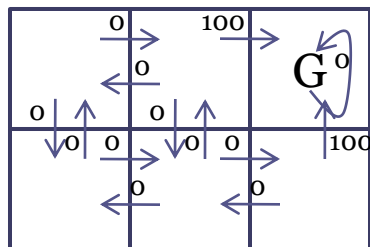
$$Q(s, a) \equiv r(s, a) + \gamma V^*(\delta(s, a))$$

- Formulu optimalne strategije prepisujemo

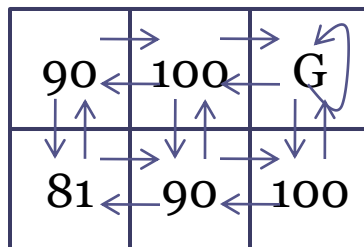
$$\pi^*(s) = \arg \max_a Q(s, a)$$

- Agent uči Q funkciju umjesto V^* funkcije
 - Za stanje s potrebno znati koje su akcije na raspolaganju i odabrati onu koja maksimizira $Q(s, a)$
 - Do globalnog optimuma sekvenci akcija lokalnim vrijednostima Q -a

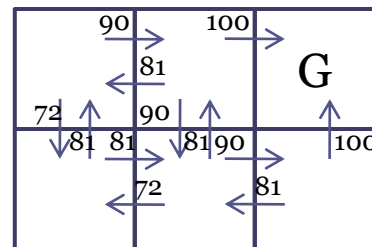
Primjer



$r(s,a)$



$V^*(s)$



$Q(s,a)$

Učiti Q

- Naučiti optimalnu strategiju → naučiti Q funkciju
- Ključno je pronaći pouzdan način procjene vrijednosti Q-a
- Proces iterativne aproksimacije
- Primjetimo odnos:

$$V^*(s) = \max_{a'} Q(s, a')$$

- Koji nam dozvoljava zapis:

$$Q(s, a) = r(s, a) + \gamma \max_{a'} Q(\delta(s, a), a')$$

- Q – prava Q funkcija
- Q' – procjena
- Ovdje je hipoteza velika tablica Q' vrijednosti
 - $\langle s, a \rangle \rightarrow Q'(s, a)$
 - Obično ih sve inicijaliziramo na 0

Q learning algoritam

- Agent proučava svoje stanje s , odabire akciju a , proučava nagradu $r = r(s, a)$ i novo stanje $s' = \delta(s, a)$
- Nakon toga osvježava $Q'(s, a)$ na svakom prijelazu

$$Q'(s, a) \leftarrow r + \gamma \max_{a'} Q'(s', a')$$

- Navedeno pravilo koristi trenutnu Q' vrijednost novog stanja s' za bolju procjenu Q' vrijednosti prethodnog stanja s
- Možda nam izgleda da je potrebno znati δ i r , međutim nije tako
 - Agent izvršava akciju u okolišu, proučava novo stanje s' i nagradu r
 - Svojevrsno simpliranje tih funkcija za trenutne vrijednosti s i a

Q learning algoritam

Q learning

Za svaki s , a inicijaliziraj $Q(s, a)$ na nulu

Promatraj trenutno stanje s

Čini zauvijek

Odaberi i izvrši akciju a

Primi trenutnu nagradu r

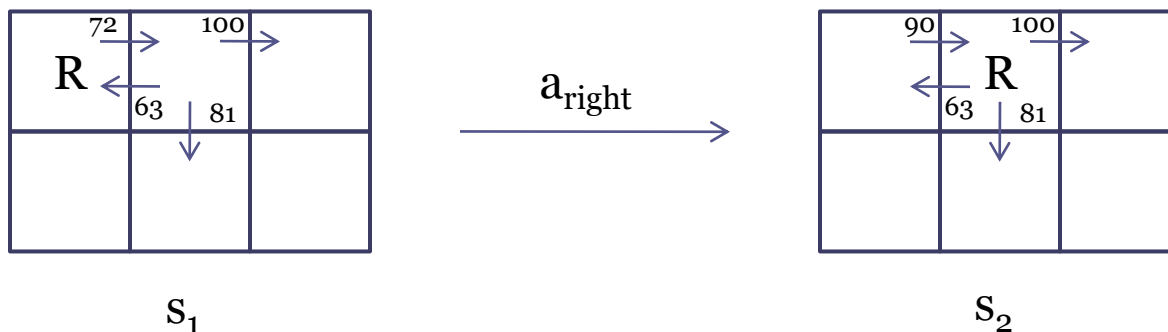
Promatraj novo stanje s'

Ažuriraj $Q(s, a)$ na sljedeći način:

$$Q'(s, a) \leftarrow r + \gamma \max_{a'} Q'(s', a')$$

$s \leftarrow s'$

Q learning - primjer



$$\begin{aligned}\hat{Q}(s_1, a_{right}) &\leftarrow r + \gamma \max \hat{Q}(s_2, a') \\ &\leftarrow 0 + 0.9 \max\{63, 81, 100\} \\ &\leftarrow 90\end{aligned}$$

- Treniranje se sastoji od serije epizoda
- Tokom svake epizode, agent započinje u slučajno odabranom stanju i čini akcije sve dok ne dohvati apsorbirajuće stanje
- Nakon toga, sve ponovno ispočetka

Q learning - komentar

- Kako vrijednosti Q' evoluiraju?
- Sve Q' su ispočetka jednake nuli
- Agent ne osvježava niti jednu vrijednost sve dok ne dohvati konačno stanje i dobije nagradu različitu od nule
- Tada će osvježiti Q' vrijednost prijelaza u konačno stanje
- U narednoj epizodi, prolazak kroz isto to stanje omogućava agentu osvježavanje Q' vrijednosti stanja prije
- Uz dovoljan broj epizoda treniranja, informacija unazadno propagira od stanja s nagradama različitima od nule po čitavom prostoru stanje-akcija raspoloživo agentu
- U konačnici Q' tablica dolazi do vrijednosti Q
 - Konvergencija?

Q learning - konvergencija

- Da li će Q learning algoritam konvergirati prema pravim vrijednostima funkcije Q?
- Hoće, ako:
 - 1. Sustav je deterministički markovljev proces odlučivanja
 - Stohastički kontrolni proces u diskretnom vremenu za koji vrijedi da buduće stanje ovisi samo o trenutnom stanju
 - 2. Nagradne vrijednosti su ograničene
 - 3. Agent odabire akcije tako da posjećuje svaki par stanje-akcija beskonačno mnogo puta
- Ovi su nam uvjeti dovoljno generalni, ali nas 3. ograničava
- Ideja iza dokaza konvergencije je da $Q'(s, a)$ s najvećom greškom reducira istu tu pogrešku s faktorom γ pri svakom osvježavanju
- Nova vrijednost samo djelomično ovisi o Q' estimatoru koji čini pogrešku

Strategije eksperimentiranja

- Kako agent odabire akcije?
- Odabir akcije koja maksimizira $Q'(s, a)$
 - Iskoristiti trenutnu aproksimaciju Q'
 - Problem: zapadamo u akcije s visokim Q' vrijednostima koje poznajemo od samog starta, ne pretražujemo ništa novo
- Probabilistički pristup selektiranju akcije
 - Akcije s višom Q' vrijednosti dobivaju veću vjerojatnost odabira
 - Npr:
$$P(a_i | s) = \frac{k^{Q'(s, a_i)}}{\sum_j k^{Q'(s, a_j)}}$$
 - Veće vrijednosti $k \rightarrow$ veća vjerojatnost Q' iznad prosjeka \rightarrow iskorištavanje
 - Manja vrijednost $k \rightarrow$ pretraživanje
 - k se može povećavati kroz iteracije (pretraživanje na početku, kasnije iskorištavanje)

Iteracije algoritma

- Q learning nije potrebno učiti na optimalnim sekvencama akcija
- Prisjetimo se uvjeta konvergencije algoritma
- Treniramo agenta slijedom epizoda
- U svakoj epizodi, agenta stavljamo u nasumično početno stanje i dozvoljavamo mu da dođe do apsorbirajućeg stanja
- Nakon toga vršimo korekciju
- Strategije poboljšanja brzine konvergencije
 - Ukoliko zapamtimo čitavu epizodu, možemo raditi višestruke korekcije zaredom, umjesto jedne po epizodi
 - Čuvamo prethodne prijelaze zajedno s trenutnim nagradama i periodički ih ponovno istreniramo
- Ukoliko agent zna δ i r , može jednostavno simulirati okoliš!

Temporal difference learning

- Q learning algoritam uči iterativnim smanjivanjem odstupanja između procjene Q vrijednosti susjednih stanja
- On predstavlja poseban slučaj generalne klase Temporal difference algoritama koji uče smanjivanjem odstupanja između procjena u različitim vremenima
- Q learning pravilo reducira razliku između Q stanja i njegovog susjednog stanja
- Isto tako možemo dizajnirati algoritam koji reducira razliku između trenutnog stanja i nekog udaljenijeg stanja (ne nužno susjednog)

TD(λ)

- Do sada smo imali:

$$Q^{(1)}(s_t, a_t) \equiv r_t + \gamma \max_a Q'(s_{t+1}, a)$$

- $Q(s_t, a_t)$ možemo izračunati i na temelju nagrada iduća dva koraka:

$$Q^{(2)}(s_t, a_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 \max_a Q'(s_{t+2}, a)$$

- Ili na temelju idućih n koraka:

$$Q^{(n)}(s_t, a_t) \equiv r_t + \gamma r_{t+1} + \dots \gamma^{n-1} r_{t+n-1} + \gamma^n \max_a Q'(s_{t+n}, a)$$

TD(λ)

- Ideja TD(λ)
 - Koristiti konstantu $0 \leq \lambda \leq 1$ za kombinaciju estimatora dobivenih od više koraka unaprijed

$$Q^\lambda(s_t, a_t) \equiv (1 - \lambda)[Q^{(1)}(s_t, a_t) + \lambda Q^{(2)}(s_t, a_t) + \lambda^2 Q^{(3)}(s_t, a_t) + \dots]$$

- Ili rekurzivno:

$$Q^\lambda(s_t, a_t) \equiv r_t + \gamma \left[(1 - \lambda) \max_a Q'(s_t, a_t) + \lambda Q^\lambda(s_{t+1}, a_{t+1}) \right]$$

- Motivacija ove metode
 - U nekim okruženjima treniranje će biti učinkovitije ako se promatra više prijelaza unaprijed

Generalizacija

- Imamo eksplicitnu tablicu Q' vrijednosti – štreberluk
- Uvjet konvergencije – beskonačno posjećivanje svih prijelaza
- Gdje je tu učenje?
- Mnogi sustavi koriste aproksimacije uz Q learning algoritam
- Npr. Backpropagation algoritam
 - Neuronska mreža nadomješta Q' lookup tablicu
 - Svako $Q'(s, a)$ osvježavanje koristi kao primjer za učenje
- No nije lijek za sve
 - U nekim sustavima, algoritam ne uspijeva konvergirati čim se uvede aproksimacijska funkcija
 - Promjene u težinama uzrokuju promjenu Q' procjena za druge prijelaze

Literatura

- Machine Learning
 - T. Mitchell
- Reinforcement Learning: An Introduction
 - R. S. Sutton, A. G. Barto
 - <http://webdocs.cs.ualberta.ca/~sutton/book/ebook/the-book.html>
- A Short Introduction to Reinforcement learning
 - S. T. Hagen, B. Kröse
 - <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.4031>