Učenje podrškom

Matko Bošnjak

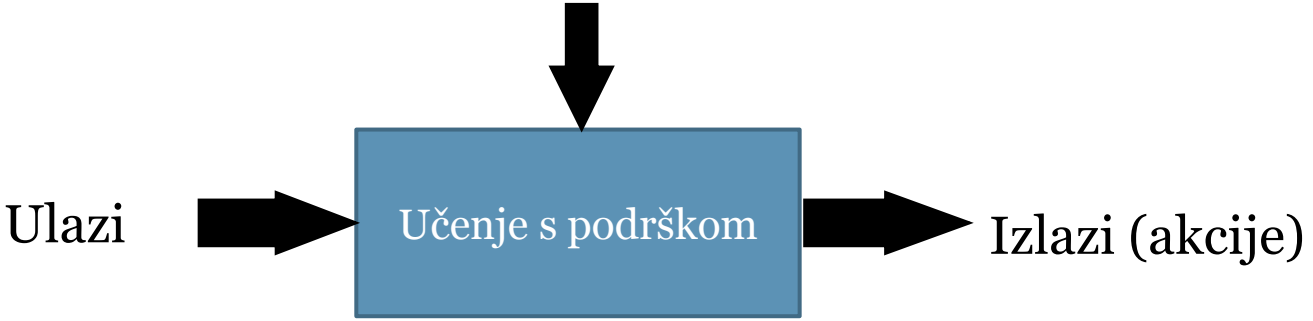
Uvod

- 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

Informacija za učenje = željeni (ciljni) izlazi

Greška = (ciljni izlaz – stvarni izlaz)

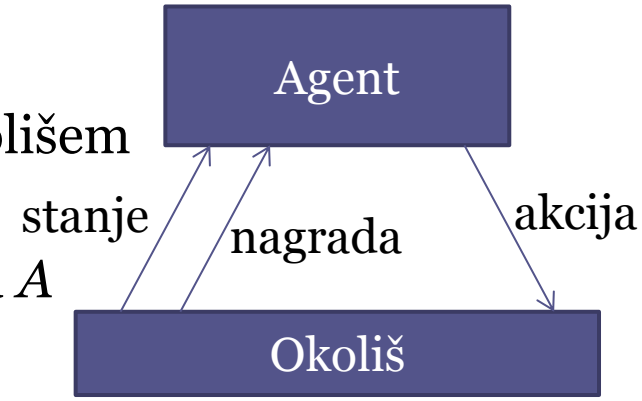
Informacija za učenje = ocjene (nagrade/kazne)



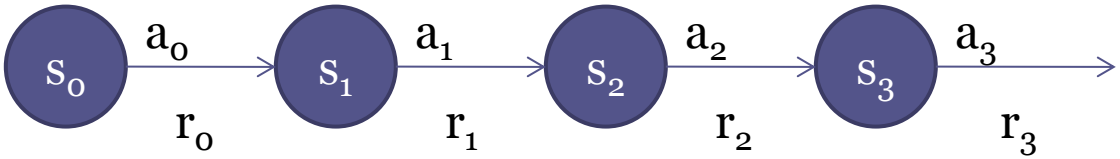
Cilj: dobiti što više nagrade

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: naučiti izabrati akcije koje max. očekivanu kumulativnu nagradu

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots, \qquad 0 \leq \gamma < 1$$

- =Zadatak agenta je naučiti kontrolnu strategiju $\pi : S \rightarrow A$

Uvod

- Cilj
  - optimizirati interakciju s okolišem kroz proces pokušaja i pogrešaka
  - izvesti seriju akcija, proučiti posljedice, naučiti kontrolnu strategiju za maksimiziranje kumulativnu nagrade

- Općenita formulacija problema učenja s podrškom
  - agent uči slijed akcija koje mijenjaju stanje u njegovoj okolini
  - kumulativna nagrada definira kvalitetu danog slijeda akcija
  - akcije mogu imati deterministične i nedeterministične ishode
  - agent može, a ne mora znati nešto o efektima svojih akcija

- Primjena na mnogo problema

Specifičnosti problema

- Odgođeno nagrađivanje
  - Jedina informacija za učenje su trenutne nagrade, nemamo par $\langle s, \pi(s) \rangle$
  - 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 nepoznatih stanja (prikupljanje novih informacija) i iskorištavanja starih informacija o dobivanju nagrade za prethodno naučena stanja i akcije
- 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

Neke primjene učenja s podrškom

- Igranje raznih igara na ploči (npr. backgammon, Teasauro‘95)
- Problemi raspoređivanja na strojevima (npr. job-shop problem, Zhang/Dietterich’95)
- Raspoređivanje i upravljanje putničkim dizailma , Crites/Barto
- Ulaganje u dionice, Moody’01
- Upravljanje autonomnim vozilima (npr. helikopter, Ng‘04)
- Upravljanje robotima u raznim okruženjima: RoboCup (npr. Soccer)
- Power TAC (Reddy ‘11)
- Bežičnog umrežavanja, Yau’11
- Optimizacija energetske potrošnje u računalnim sustavima
- Autonomno upravljanje Cloud računalnim sustavima

- Popis uspjeha primjene učenja s podrškom
- Demos: http://www-all.cs.umass.edu/rlr/

Primjer

- Agent/robot vozi bicikl
- U jednom trenutku se nađe pod kutem od 45° udesno
  - Desno → pad (negativna podrška)
- Ponovno ista situacija
  - Ovaj puta lijevo → pad
  - Stanje 45° vodi u propast kako god
- U jednom trenutku se nađe pod kutem od 40° udesno
  - Skrenuti lijevo?
  - Skrenuti desno?
  - Lijevo → 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 zasnovanu 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, agentu stanju s_t odabire i izvodi akciju a_t
  - Okoliš odgovara trenutnom 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
  - Ovise samo o trenutnom stanju i akciji! (Markovljevo svojstvo)

- Proučavat ćemo samo konačne S i A
- Koncentrirati ćemo se na deterministčke MDP

Markovljevi procesi odlučivanja

- Skup stanja $s \in S$
- Skup akcija $a \in A$
- Funkcija prijelaza $\delta: S \times A \rightarrow S$ => model svijeta
  - Nedeterministički slučaj: $\delta: S \times A \rightarrow Prob(S)$
  - vjerojatnostna distribucija $P(s'|s,a)$
  - $\delta(s,a,s') = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}, \forall s, s' \in S, a \in A$
- Funkcija nagrade $r(s,a)$ => nagrada za akciju a iz stanja s
  - Često uzimamo samo $r(s) = r$ =>koliko je dobro neko stanje
  - Nedeterministički slučaj: $r(s,a,s') = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}, \forall s, s' \in S, a \in A$
- Markovljevo svojstvo: utjecaj akcije u nekom stanju ovisi samo o tom stanju
- Kontrolna strategija: $\pi : S \rightarrow A$
- Evaluacijska funkcija strategije: koliko je dobra stategija?
  - kumulativna nagrada (konačni broj, umanjivanje budućih nagrada)
  - Nedeterministički slučaj: očekivanje ukupne kumulativne nagrade

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:

$$V^{\pi}(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$$

$$\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

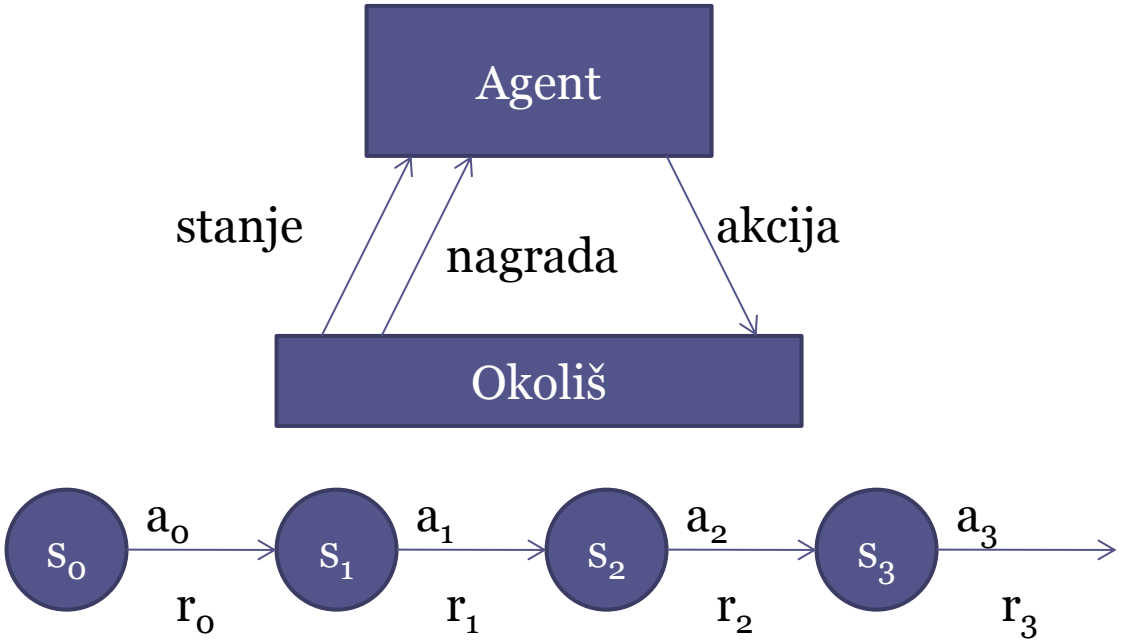
- $0 \leq \gamma < 1$ konstanta koja određuje relativnu vrijednost odgođene prema trenutnoj nagradi
- Razumno je umanjiti (eksponencijalno) značaj budućih nagrada prema trenutnim jer preferiramo nagradu sada, a ne kasnije

Definicija zadatka agenta

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

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

- 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*



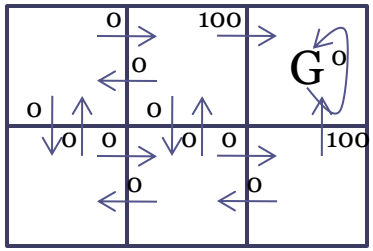
Cilj: naučiti optimalnu kontrolnu strategiju $\pi : S \rightarrow A$
(koja daje najveću očekivanu kumulativnu nagradu)

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

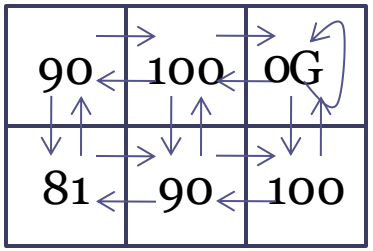
$$V^{\pi}(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$$

$$\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

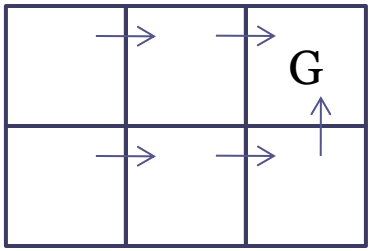
Primjer



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* $<s, a>$ (nadzirano učenje)
  - Sva informacija koju imamo je sekvenca trenutnih nagrada $r(s_i, a_i)$ za $i=0, 1, 2, \ldots$
  - 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?
  - Želimo funkciju koja direktno uči dobre parove <stanje, akcija> odnosno koju akciju treba izvršiti u kojem stanju
  - To zovemo Q funkcija

Q funkcija

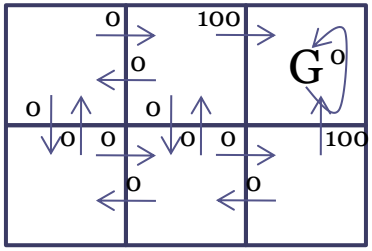
- Definirajmo evaluacijsku funkciju *Q(s, a)*
  - Vrijednost jednaka maksimalnoj prigušenoj kumulativnoj nagradi koja se može dostići iz stanja *s* čineći akciju *a*
- Trenutna nagrada + prigušena vrijednost koju dobivamo sljedeći optimalnu strategiju

- Formulu optimalne strategije prepisujemo

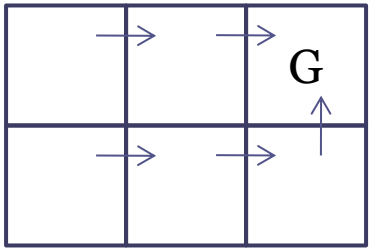
$$\pi^*(s) = \arg\max_a Q(s,a)$$

- 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

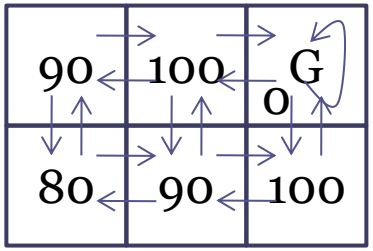
Primjer



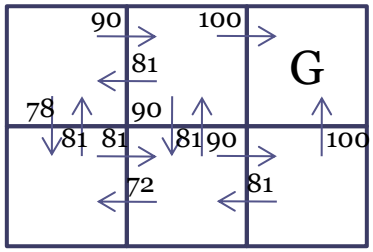
r(s,a)



Jedna optimalna strategija



$V^*(s)$



$Q(s,a)$

$$V^*(s) \leftarrow \max_a \left[r(s,a) + \gamma V^*(\delta(s,a)) \right]$$

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

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

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

Učenje Q funkcije

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

$$V^*(s) = \max_{a'} Q(s,a') \quad \leftarrow \quad V^*(s) \leftarrow \max_a \left[r(s,a) + \gamma V^*(\delta(s,a)) \right]$$

- Koji nam dozvoljava zapis:

$$Q(s,a) = r(s,a) + \gamma \max_{a'} Q(\delta(s,a),a') \quad \leftarrow \quad Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$

- Rekurzivna definicija daje podlogu za algoritme koje iterativno aproksimiraju Q
- Za opis algoritma koristimo zapis
  - *Q* – prava Q funkcije
  - *Q'* – procjena (hipoteza koja se uči)
- Ovdje je hipoteza reprezentirana velikom tablicom Q' vrijednosti
  - za par <s, a> → Q'(s, a) (trenutna hipoteza o stvarnoj ali nepoznatoj Q vrijednosti)
  - Obično ih sve inicijaliziramo na 0 ili neke slučajne vrijednosti

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*
  - Može se promatrati kao uzorkovanje 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

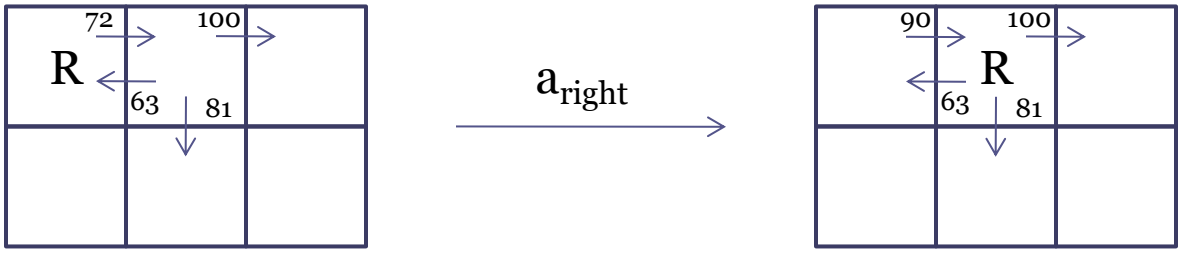
Čini zauvijek

- 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 ← s'

Q learning – primjer



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

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

- Pošto okolina ima apsorbirajuće stanje treniranje se provodi kroz epizode
- Tokom svake epizode, agent započinje u slučajno odabranom stanju i čini akcije sve dok ne dohvati apsorbirajuće stanje
- Nakon toga, sve ponovno ispočetka (novom epizodom)

Q learning – komentar

- Kako vrijednosti *Q'* evoluiraju?
- Sve *Q'* su ispočetka jednake nuli
- Agent ne osvježavaa 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
  - 2. Vrijednosti nagrade su ograničene: $|r(s,a)| < c$
  - 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 vrijednostiu tablici *Q'(s, a)* s najvećom greškom reducira istu tu pogrešku s faktorom γ pri svakom osvježavanju
- Nova vrijednost ovisi samo o *Q'* estimatoru koji čini pogrešku i o točnoj vrijednosti trenutne nagrade (koja u sebi nema grešku)

Strategije eksperimentiranja

- Kako agent odabire akcije?
  - uvijet konvergencije zahtjeva da se validne akcije odabiru beskonačno često
- 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:

$$P(a_i \mid s) = \frac{k^{Q'(s,a_i)}}{\sum_j k^{Q'(s,a_j)}}$$

  - Veće vrijednosti *k* → veća vjerojatnost Q' iznad prosjeka → iskorištavanje
  - Manja vrijednost *k* → pretraživanje
  - *k* se može povećavati kroz iteracije (pretraživanje na početku, kasnije iskorištavanje)

Strategije poboljšanja brzine konvergencije

- 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
  - Pohranjujemo čitavu epizodu kako bi mogli raditi višestruke korekcije zaredom, umjesto jedne po epizodi
  - Pohranjujemo prethodne prijelaze zajedno s dobivenim trenutnim nagradama i periodički ih ponovno istreniramo kako ne bi trebali izvoditi skupe akcije za stanja koja samo prošli
- Ukoliko agent zna δ i r, može jednostavno simulirati okoliš!
  - Efikasni algoritmi zasnovani na dinamičkom programiranju

Temporal difference learning

- Q learning algoritam uči iterativnom redukcijom odstupanja između procjene Q vrijednosti susjednih stanja
- On predstavlja poseban slučaj generalne klase Temporal difference algoritama koji uče redukcijom odstupanja između procjena u različitim vremenima

- Q learning pravilo reducira razliku između Q' stanja i njegovog susjednog stanja
- Isto tako možemo oblikovati 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:

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

TD(λ)

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

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

- 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

MDP i RL: Što bi trebali znati

- Učenje kako bi se izabrale optimalne akcije A
- Iz kumulativne (umanjene faktorom) nagrade
- Učenjem evaluacijskih funkcija: V(S), Q(S,A)

- Osnovne ideje
  - Ako funkcija prijelaza (model svijeta) $\delta: S \times A \rightarrow S$ poznata
    - može se koristiti dinamičko programiranje kako bi se naučila funkcija V(S)
    - nakon što je naučena, izabere se akcija a_t koja daje najveći $V(s_t)$
  - Ako funkcija prijelaza (model svijeta) $\delta: S \times A \rightarrow S$ nepoznata
    - uči se $Q(S, A)$
    - kako bi se učilo uzorkuje (engl. sampling) se funkcija prijelaza djelovanjem u stvarnom svijetu
    - nakon što je naučena, izabere se akcija a_t koja daje najveći $Q(s_t, a_t)$

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

Osnovna literatura

- Machine Learning
  - T. Mitchell
- Reinforcement Learning: An Introduction
  - R. S. Sutton, A. G. Barto
  - http://www.incompleteideas.net/sutton/book/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

Dodatna literatura

- Reinforcement Learning: A Tutorial Survey and Recent Advances
  - http://web.mst.edu/~gosavia/joc.pdf
- An Introduction to Reinforcement learning
  - http://ccc.inaoep.mx/~emorales/Papers/2011/Chap4.pdf
- A Tutorial for Reinforcement Learning
  - http://web.mst.edu/~gosavia/tutorial.pdf
- Algorithms for Reinforcement Learning
  - http://www.sztaki.hu/~szcsaba/papers/RLAlgsInMDPs-lecture.pdf
- Model Selection in Reinforcement Learning
  - http://www.sztaki.hu/~szcsaba/papers/RLModelSelect.pdf

Dodatna literatura

- Reinforcement learning page of the American Association for Artificial Intelligence (AAAI)
  - http://aitopics.org/topic/reinforcement-learning
- Reinforcement Learning Repository
  - http://www-anw.cs.umass.edu/rlr/
- Reinforcement Learning FAQ
  - http://webdocs.cs.ualberta.ca/~sutton/RL-FAQ.html
- RL community – Successes of RL
  - http://rl-community.org/wiki/Successes_Of_RL

Alati

- PyBrain: The Python Machine Learning Library
  - http://pybrain.org/pages/features
  - http://www.youtube.com/watch?v=fEM7YDNonSE&feature=player_embedded
- RL-Toolkit, RL-Library, RL-Glue
  - http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rlai.html
  - http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/RLtoolkit/RLtoolkit1.0.html
- Maja Machine Learning Framework (MMLF)
  - http://mmlf.sourceforge.net/
- MATLAB Repository for Reinforcement Learning
  - http://web.mst.edu/~gosavia/mrrl_website.html