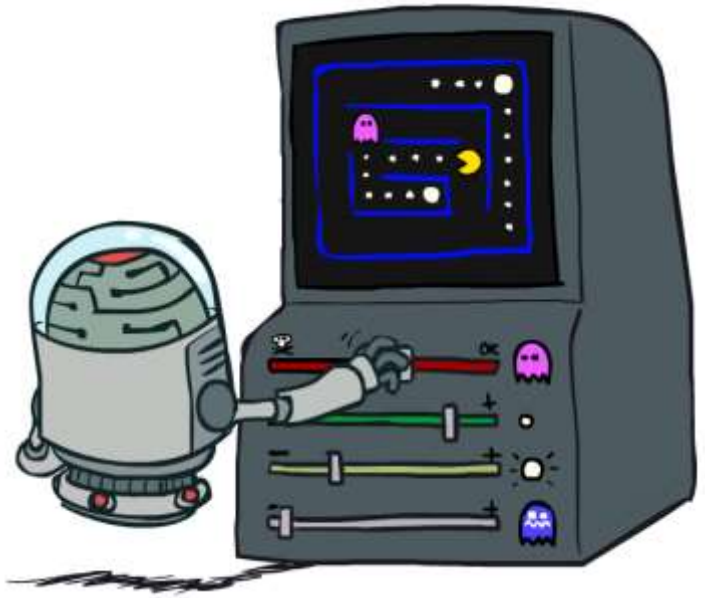
Osnovi Računarske Inteligencije

Učenje Uslovljavanjem II

Reinforcement Learning II



Predavač: Aleksandar Kovačević

Slajdovi preuzeti sa kursa CS188, University of California, Berkeley

http://ai.berkeley.edu/

Šta smo do sada naučili: MDP i RL

MDP sa svim podacima: Offline Rešenje

Cij	Tehnika
Izračunati V*, Q*, π^*	Iteriranje vredsnoti (IV) / politike (IP)
Evaluacija fiksne politike π	Evaluacija politike (EP)

MDP bez podataka: Zasnovano na Modelu

Cilj	Tehnika
Izračunati V*, Q*, π^*	IV/IP na aproks. MDP
Evaluacija fiksne politike π	PE na aproks. MDP

MPD bez podataka: Bez Modela

Cilj	Tehnika
Izračunati V*, Q*, π^*	Q-učenje
Evaluacija fiksne politike π	Učenje Vrednosti

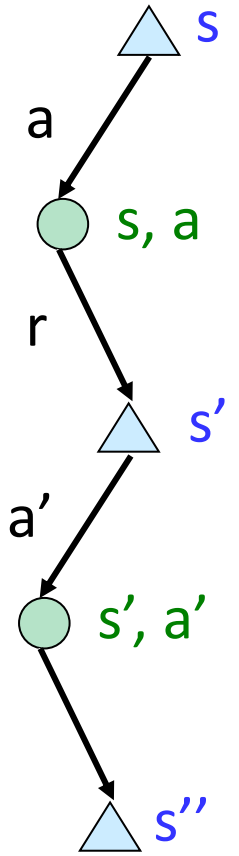
Učenje Bez Modela

- Učenje Zasnovano na Vremenskoj Razlici
  - Stičemo iskustva kroz akcije u okruženju

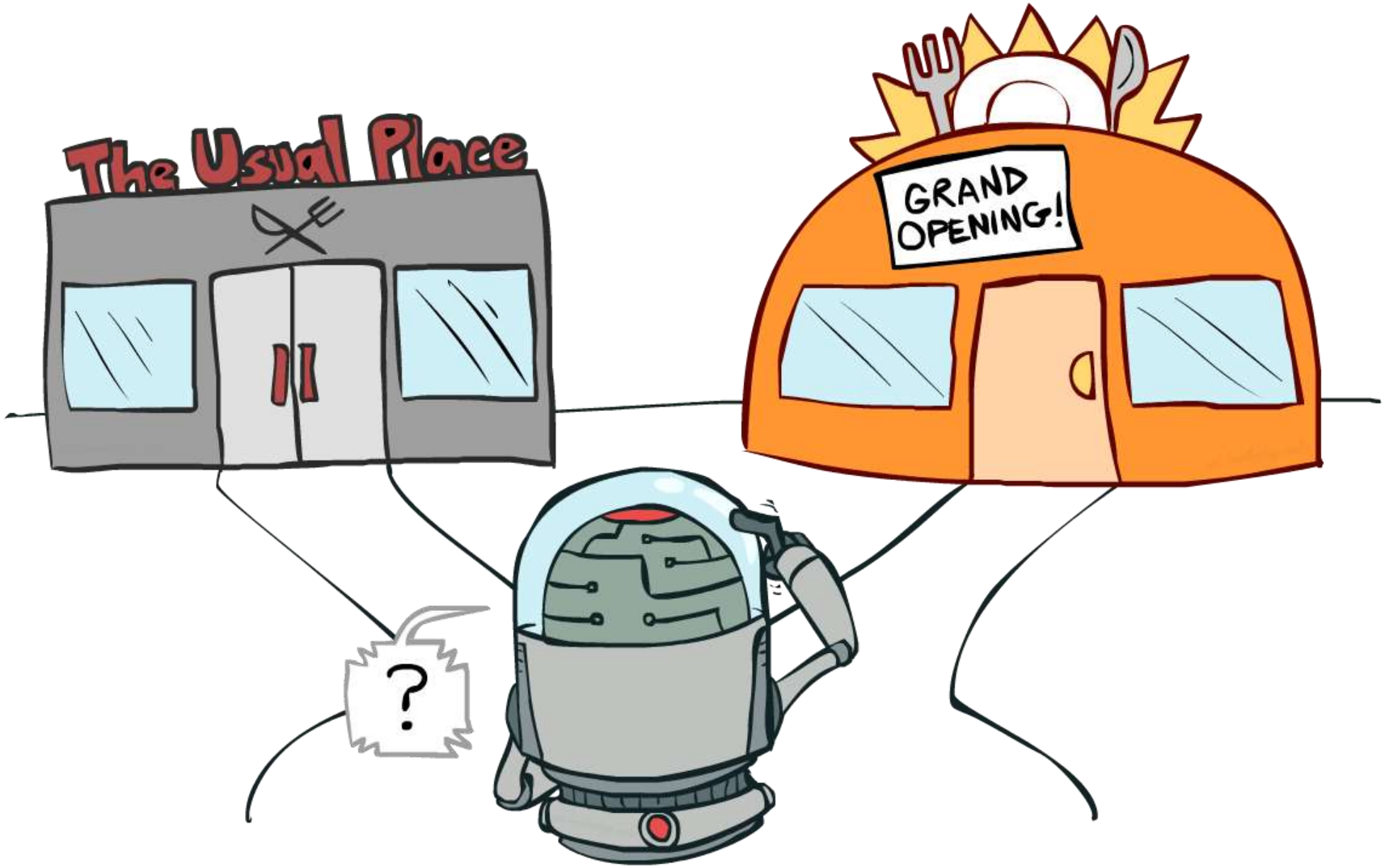
$$(s, a, r, s', a', r', s'', a'', r'', s'''' \ldots)$$

  - Poravljamo procene sa svakim (s, a, r, s')

  - Za veliki broj koraka, dobijamo suštinski Belmanove Popravke (*Bellman Updates*)



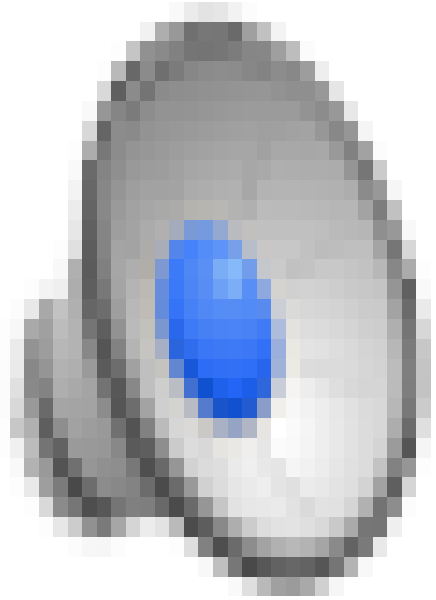
Istraživanje vs. Eksploatacija



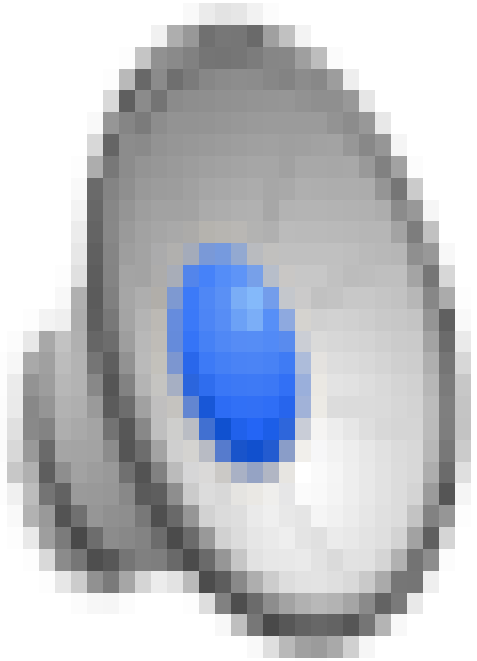
Kako Istraživati?

- Različiti načini da „forsiramo“ istraživanje
  - Najlakši: slučajne akcije (ε-greedy)
    - U svakom stanju baci novčić, tačnije:
    - Za neku malu verovatnoću ε, akciju odaberi slučajno
    - Za neku veliku verovatnoću 1-ε, radi ono što ti kaže trenutna politika za to stanje
  - Problemi sa slučajno odabranim akcijama?
    - Jedno vreme istražujemo, ali kad je učenje završeno bacamo se po okruženju nepotrebno
    - Jedno rešenje: smanjivati ε vremenom
    - Drugo rešenje: funkcije istraživanja (*exploration functions*)

Demo Q-učenje – Ručno Istraživanje – Primer Most



Demo Q-učenje – Epsilon-Greedy – Robot Koji Puzi



Funkcije Istraživanja

- Gde istraživati?
  - Slučajne akcije: svuda po malo
  - Bolja ideja: istraživati stanja za koja još uvek nismo sigurni kakva su, vremenom prestati sa istraživanjem

- Funkcije istraživanja
  - Koristimo procenu Q-vrednosti u i broj poseta tom stanju n da bi dobili optimističnu (opada sa brojem poseta) vrednost $f(u, n) = u + k/n$

može i n+1 da bi izbegli deljenje sa 0.

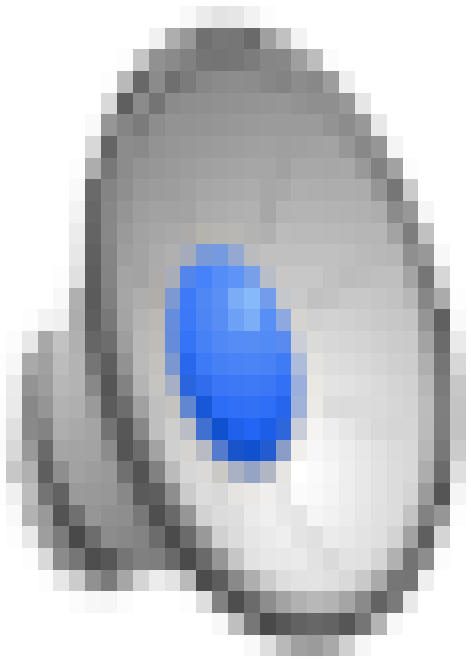
Običan Q-popravak: $Q(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} Q(s',a')$

koliko puta smo do sada u stanju s' uradili akciju a'

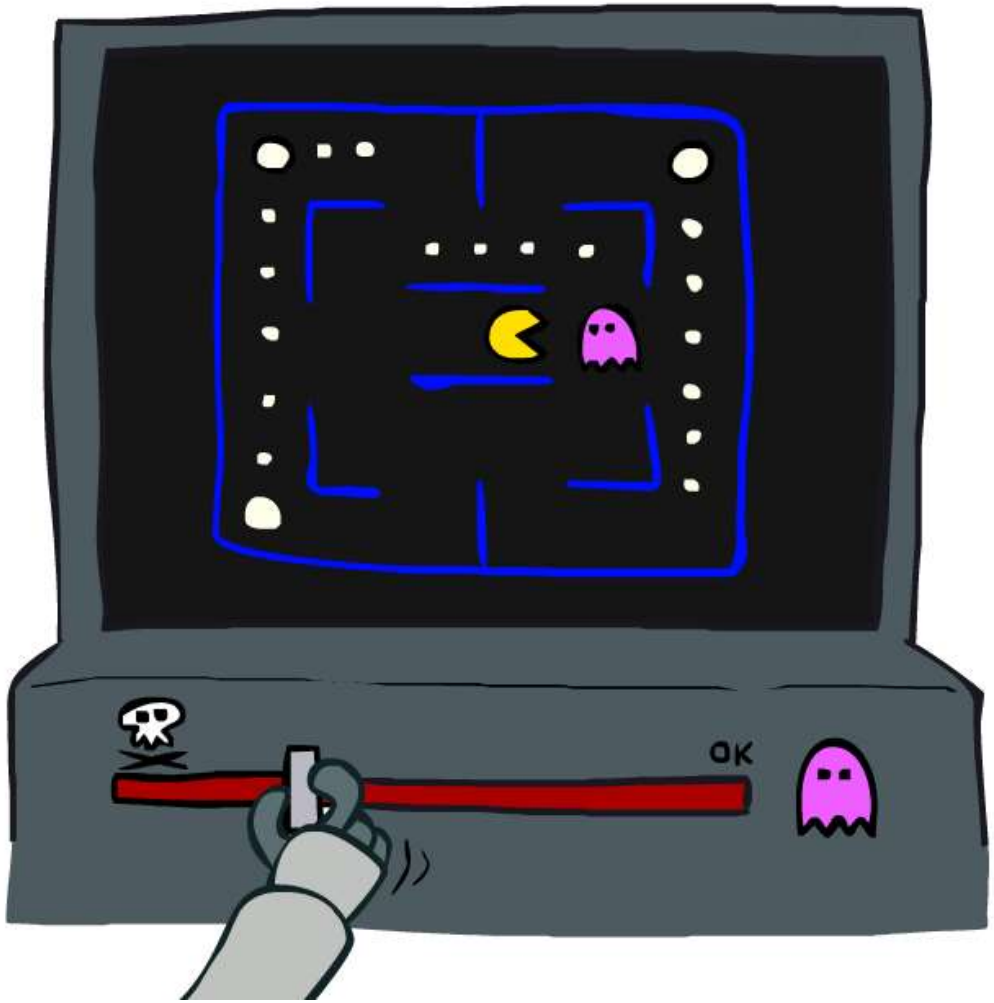
Modifikovan Q-popravak : $Q(s,a) \leftarrow_\alpha R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$

  - Napomena: ovako nagrađujemo stanja koja su nas odvela u nepoznata stanja koja će nam vremenom dati nagradu

Demo Q-učenje – Funkcija Istraživanja – Robot koji puzi



Q-Učenje sa Aproksimacijom
Approximate Q-Learning

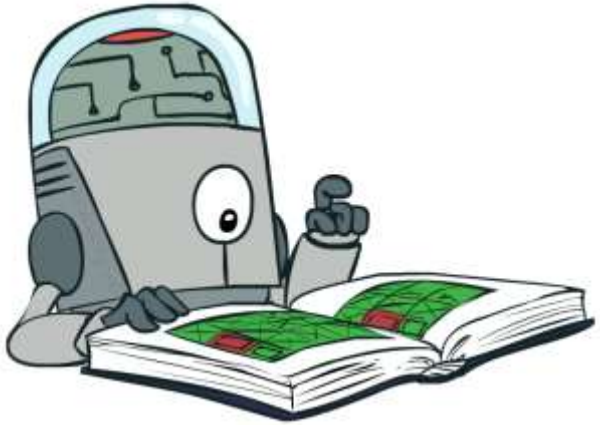


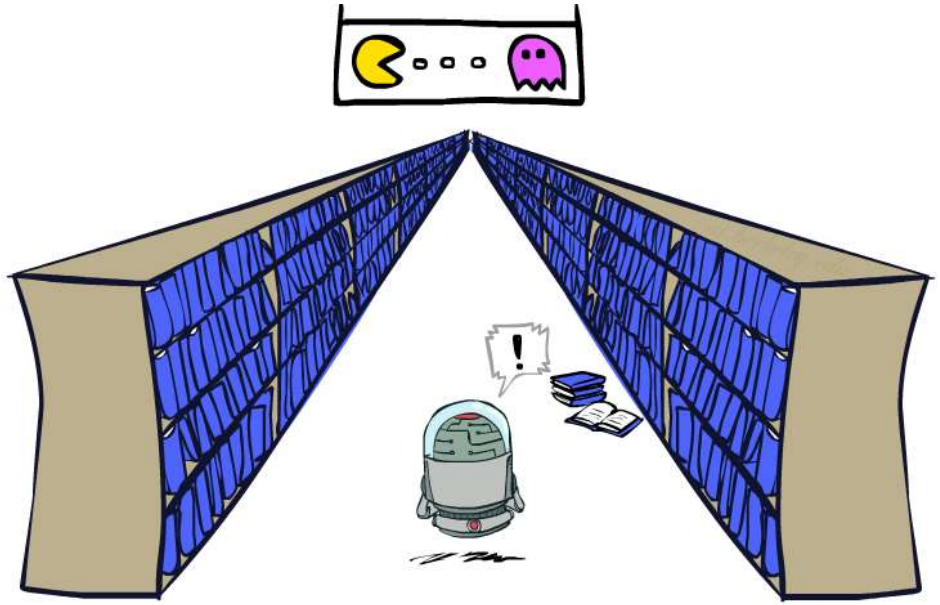
Šta je problem sa Q-učenjem?

- Osnovni algoritam popunjava matricu Q-vrednosti (vrste su stanja, a akcije kolone)
- Mora da je popuni celu!

- Za realne probleme popunjavanje cele matrice nije izvodljivo!
  - Previše stanja i akcija da bih ih sve postetili / isprobali više puta tokom učenja
  - Prevelika matrica Q-stanja da bi je čuvali u memoriji

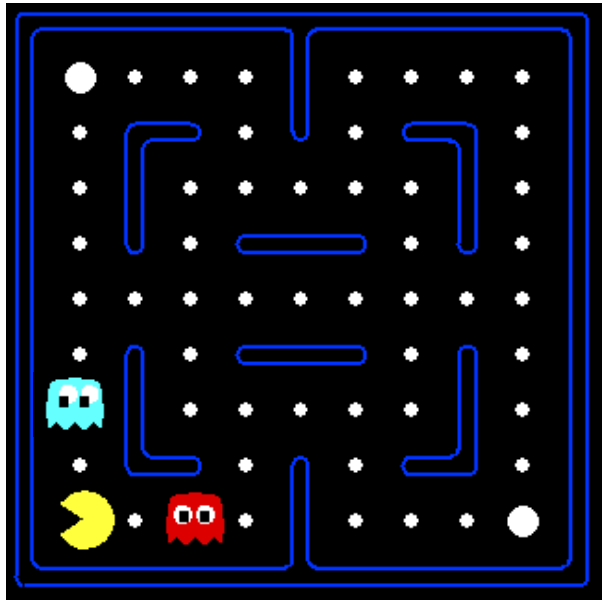
- Umesto toga, generalizujemo:
  - Naučimo mali broj Q-vrednosti klasično
  - Generalizujemo to iskustvo na slične situacije
  - Ovo je fundamentalna ideja Mašinskog Učenja, koju ćemo sresti još jako puno do kraja ovog kursa!



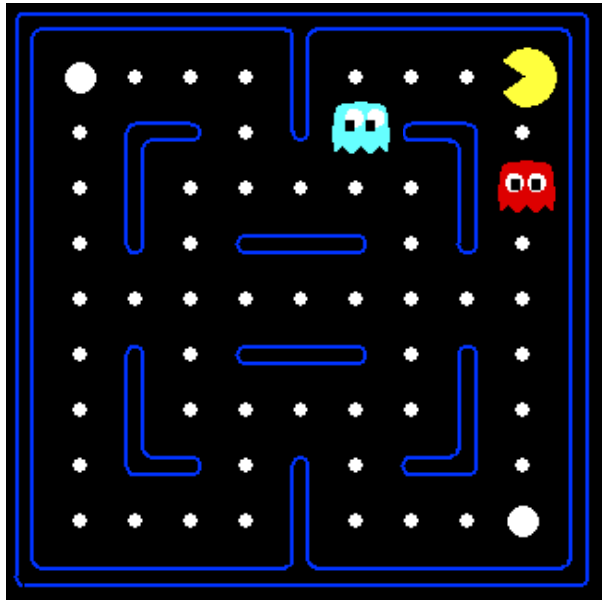


Primer: Pacman

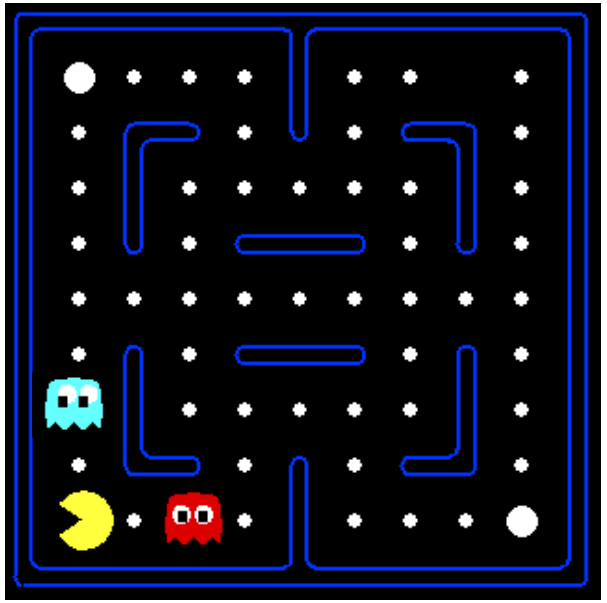
Recimo da iz iskustva naučimo da je ovo stanje loše:



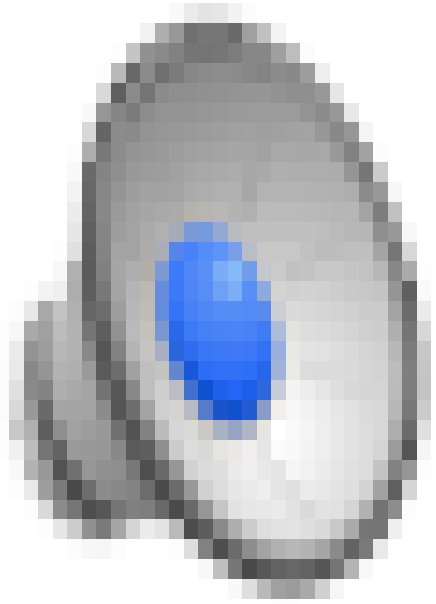
U osnovnom (naivnom) Q-učenju to nam ne pomaže mnogo za ovo stanje:



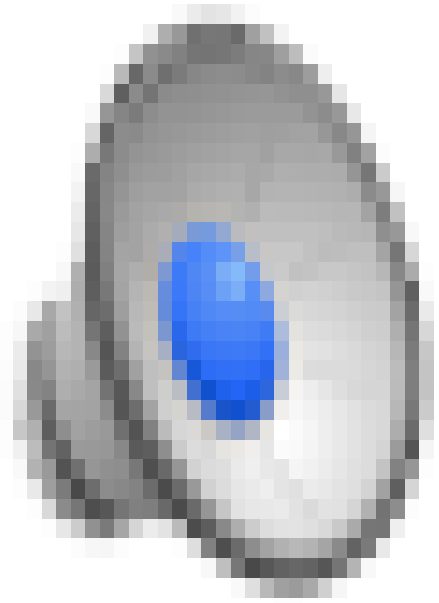
Čak ni za ovo!



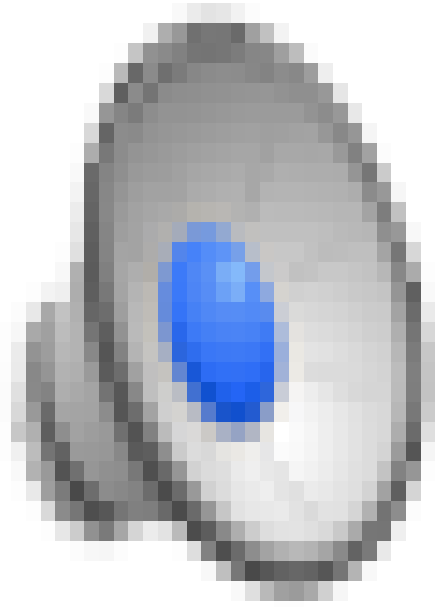
Demo Q-učenje Pacman – Mali Laviritin



Demo Q-učenje Pacman – Mali Laviritin – posle 2000 iteracija Q-učenja

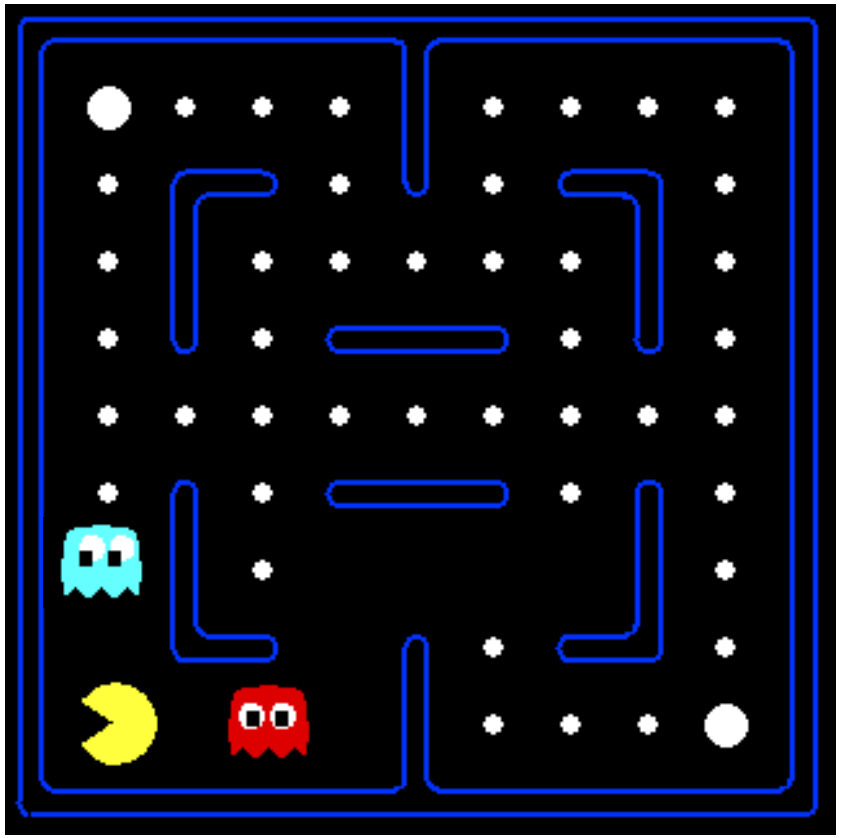


Demo Q-učenje Pacman – Malo komplikovaniji lavirint – jako puno iteracija da bi nešto naučili



Repezentacije Zasnovane Na Osobinama
Feature-Based Representations

- Rešenje: stanje reprezentujemo kao vektor osobina (*features*)
  - Osobine su funkcije koje mapiraju stanje na realan broj (često 0/1) tako da taj broj oslikava jednu ili više karakteristika tog stanja
  - Primeri osobina:
    - Udaljenost od najbližeg duha
    - Udaljenost od najbliže tačke
    - Broj „živih“ duhova
    - 1 / (udaljenost do najbliže tačke)^2
    - Da li je pacman u tunelu? (0/1)
    - …… itd.
    - da li stanje izgleda baš kao stanje na slici? (Napomena: ovo je sad loše, ali kad dođemo do Deep RL biće vrlo korisno!)
  - Naravno, na ovaj način reprezentujemo i Q-stanja (s, a) npr. akcija a nas približava tačkici itd.



Linearne Fukcije Vrednosti
Linear Value Functions

- Kada definišemo funkcije osobina Q-vrednosti i V-vrednosti možemo da reprezentujemo kao linearnu kombinaciju nekih težina i vrednosti funkcija:

$$V(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

- Prednost: svoje iskustvo sumiramo (reprezentujemo) kroz težine

- Mana: u zavisnosti od izbora funkcija osobina, moguće je da imamo stanja koja su vrlo slična (po osobinama), ali realno treba da imaju vrlo različite Q i V vrednosti.

Q-Učenje sa Aproksimacijom

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

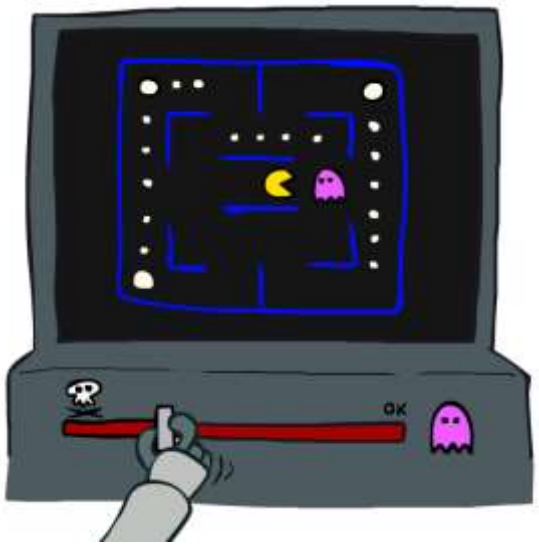
- Q-učenje sa linearnim Q-funkcijama (formula iznad):

$$\text{transition} = (s, a, r, s')$$

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s,a)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[\text{difference}] \qquad \text{Tačne Q}$$

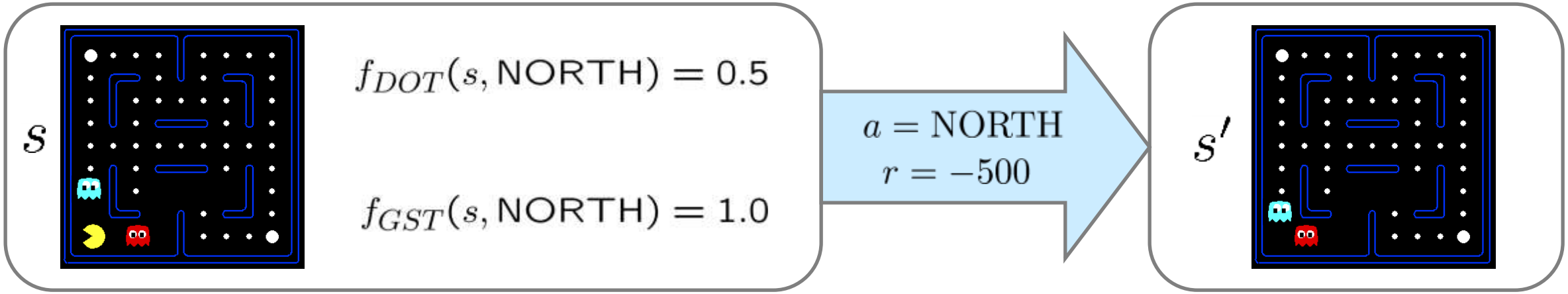
$$w_i \leftarrow w_i + \alpha\,[\text{difference}]\, f_i(s,a) \qquad \text{Aproksimirane Q}$$



- Intuitivna interpretacija:
  - Praktično, menjamo težine samo za osobine koje su aktivne (imaju veliku vrednost)
  - Na primer, ako nam se dogodi nešto jako loše „okrivićemo“ osobine koje su tada bile aktivne – učimo da „ne volimo“ stanja sa sličnim osobinama
- Formalna interpretacija: regresija pomoću MNK

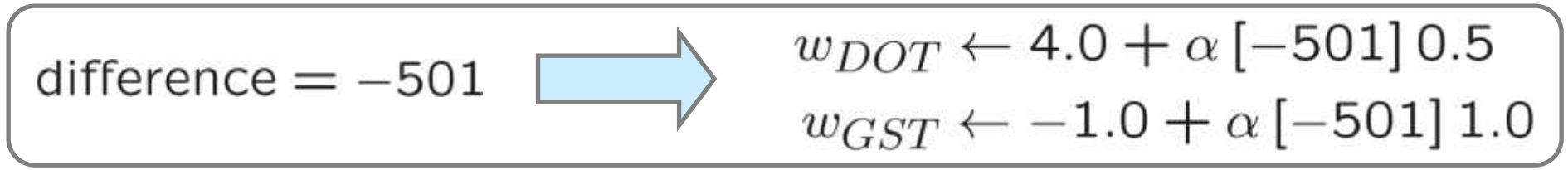
Primer: Q-Pacman

$$Q(s,a) = 4.0 f_{DOT}(s,a) - 1.0 f_{GST}(s,a)$$



$Q(s, \text{NORTH}) = +1$ $\qquad$ $Q(s', \cdot) = 0$

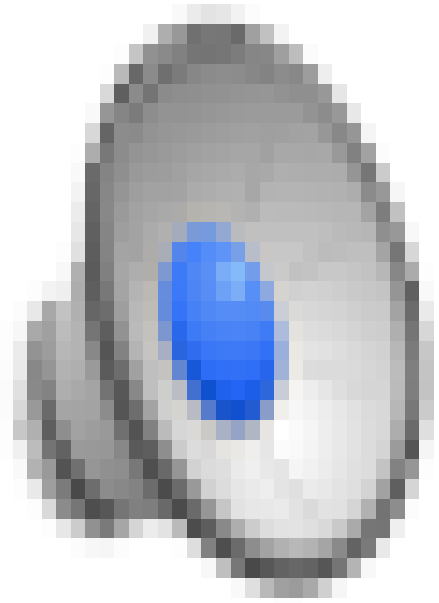
$r + \gamma \max_{a'} Q(s', a') = -500 + 0$ difference = – 500 – Q(s, NORTH)= – 500 – 1 = – 501



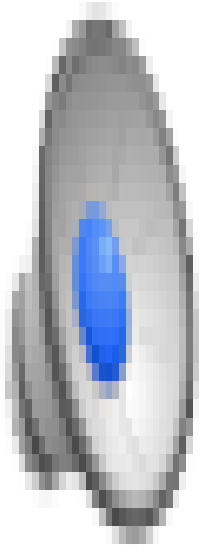
$$Q(s,a) = 3.0 f_{DOT}(s,a) - 3.0 f_{GST}(s,a)$$

alfa=0.004

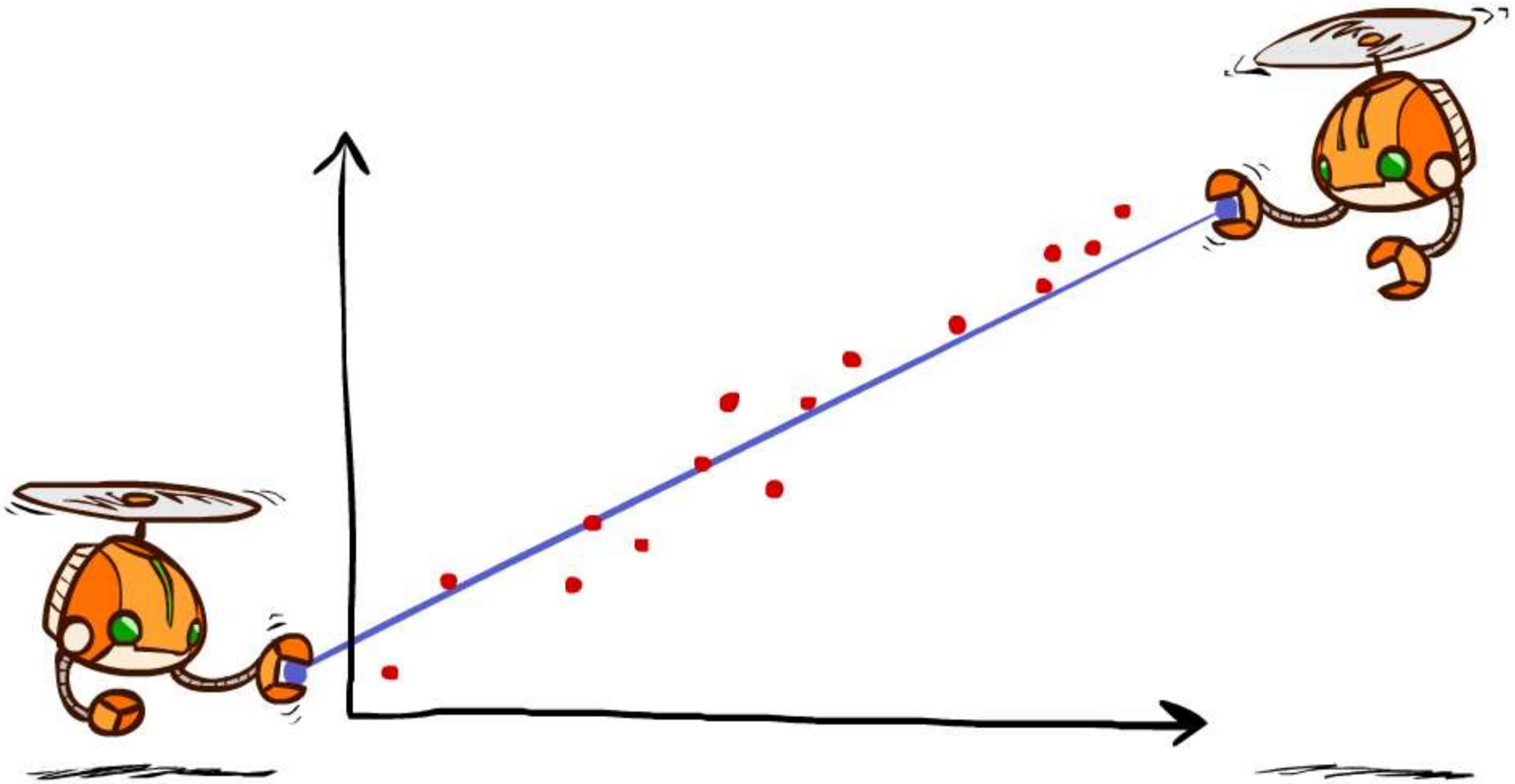
Demo – Q-učenje sa aproksimacijom - Pacman



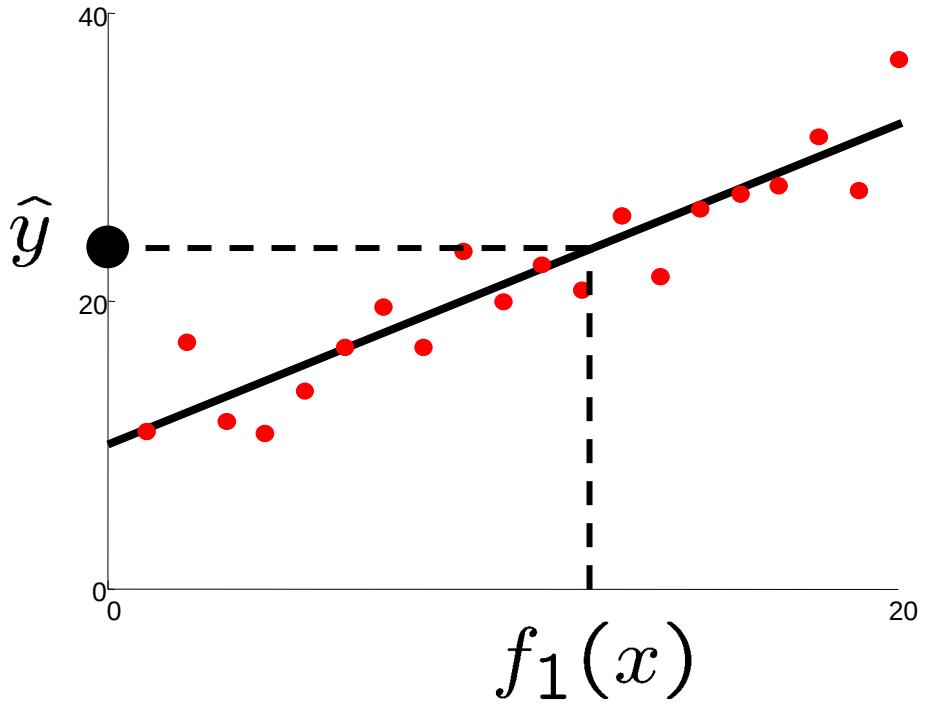
DeepMind Atari (©Two Minute Lectures) Demo – Q-učenje sa aproksimacijom pomoćue neuronskih mreža

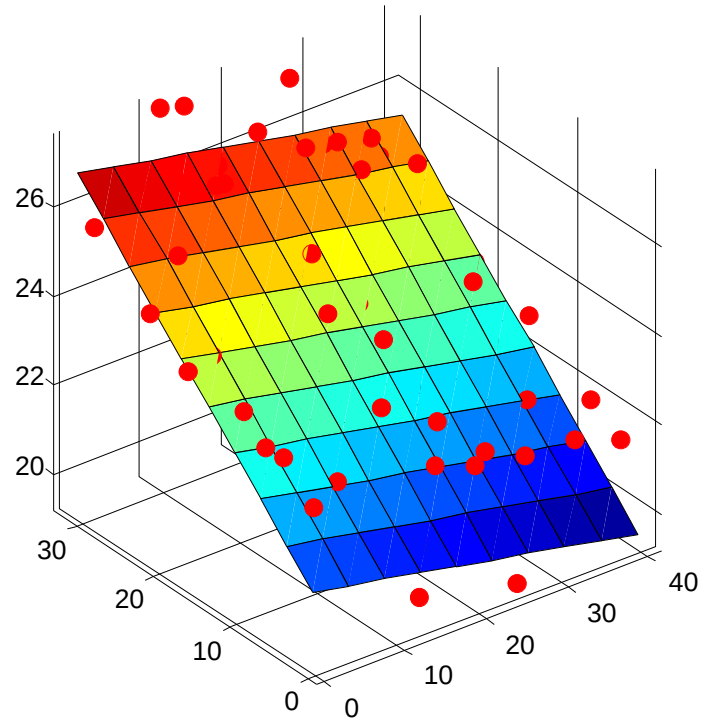


Q-Učenje i Metoda Najmanjih Kvadrata



Linearna Aproksimacija: Regresija





Predikcija:

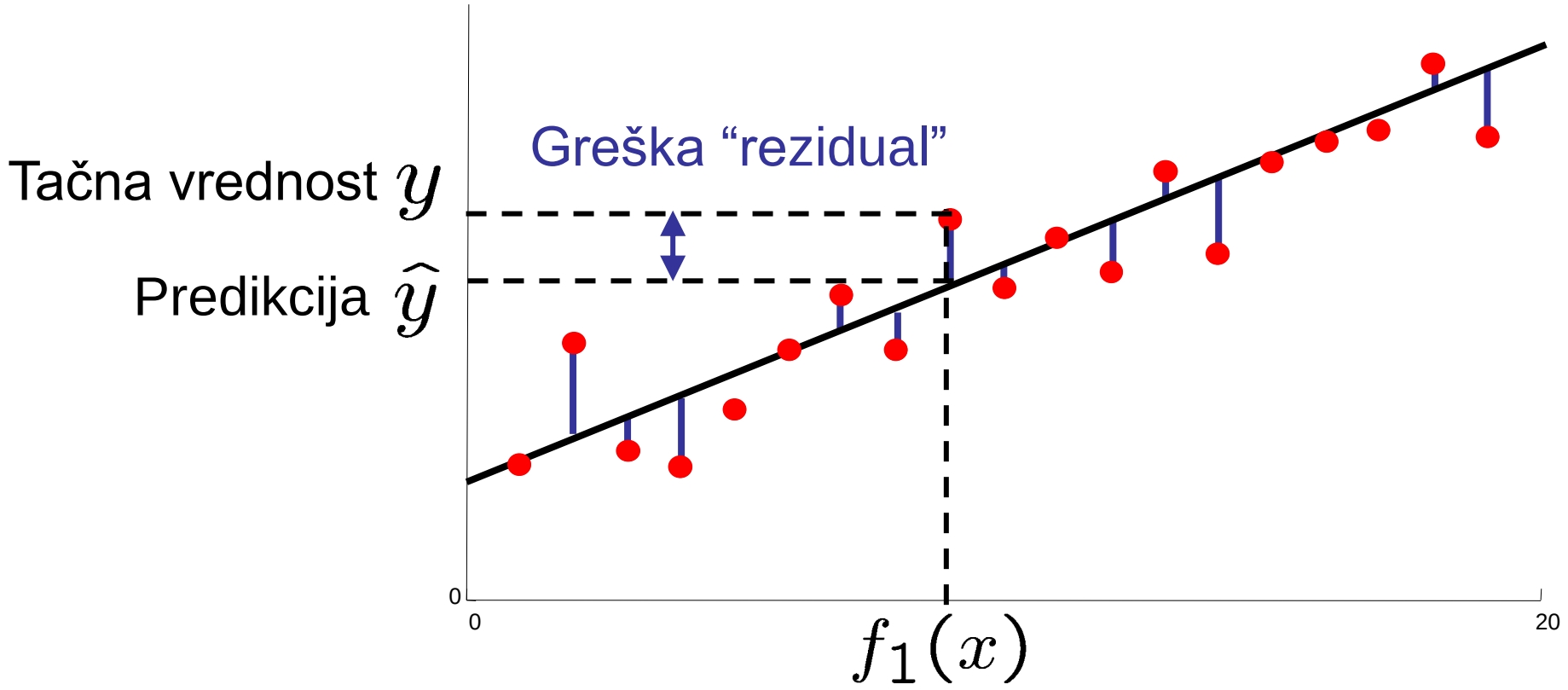
$$\hat{y} = w_0 + w_1 f_1(x)$$

Predikcija:

$$\hat{y}_i = w_0 + w_1 f_1(x) + w_2 f_2(x)$$

Kvadratna Greška

$$\text{total error} = \sum_i (y_i - \hat{y}_i)^2 = \sum_i \left(y_i - \sum_k w_k f_k(x_i) \right)^2$$



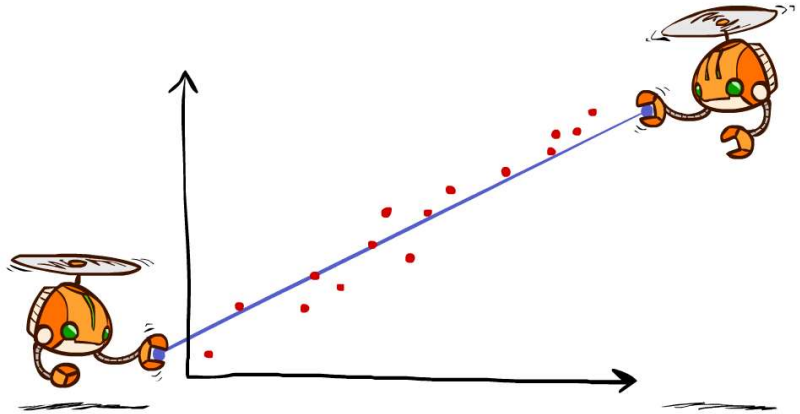
Minimizacija Greške

Ako bi imali samo jedno x, sa funkcijama osobina f(x), ciljnom vrednosti y, i težinama w:

$$\text{error}(w) = \frac{1}{2}\left(y - \sum_k w_k f_k(x)\right)^2$$

$$\frac{\partial\ \text{error}(w)}{\partial w_m} = -\left(y - \sum_k w_k f_k(x)\right) f_m(x)$$

$$w_m \leftarrow w_m + \alpha\left(y - \sum_k w_k f_k(x)\right) f_m(x)$$



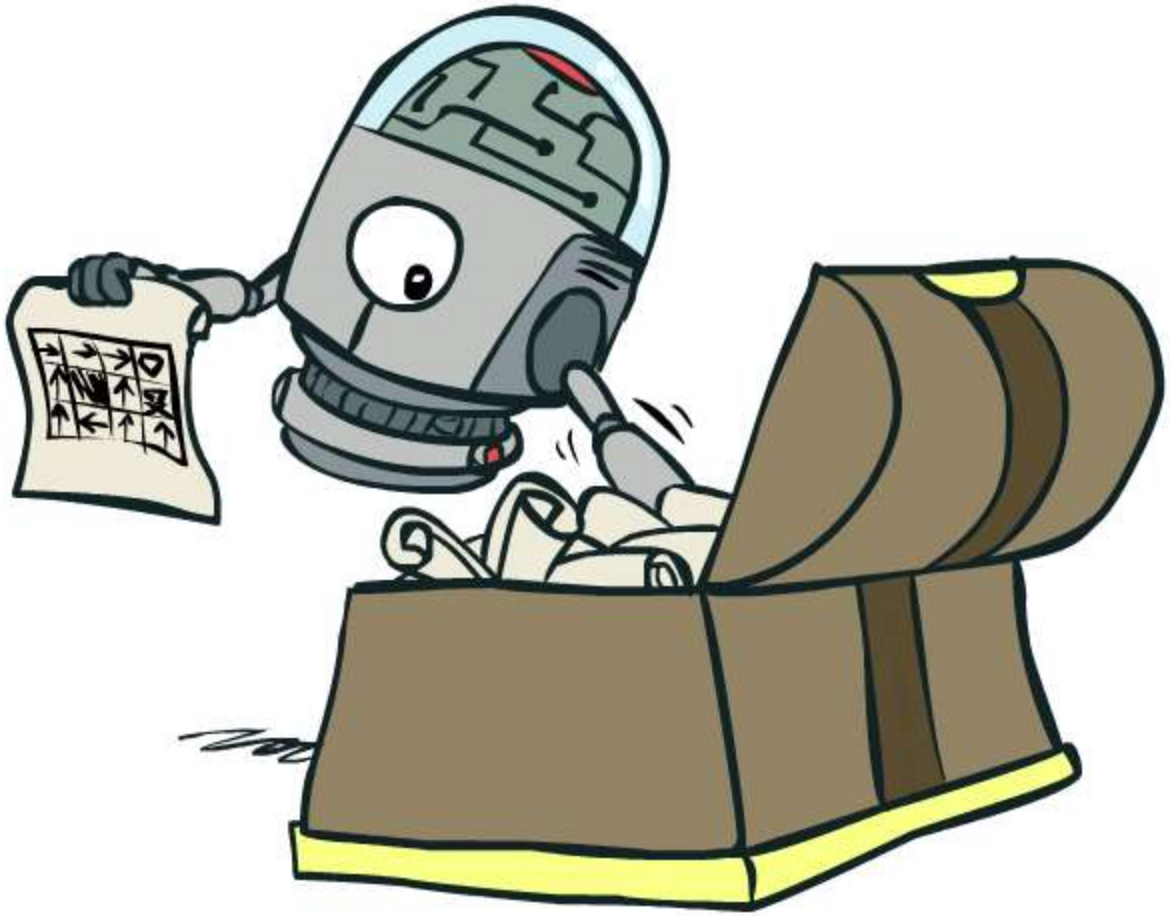
Objašenje metoda za promenu težina kod Q-učenja sa aproksimacijom:

$$w_m \leftarrow w_m + \alpha\left[r + \gamma \max_a Q(s', a') - Q(s, a)\right] f_m(s, a)$$

“cilj” “predikcija”

Napomena: za razliku od klasičnog nadgledanog ML cilj se ovde za isto x menja kroz vreme.

Traženje Politike - Policy Search



Traženje Politike

- Problem: često politike zasnovane na osobinama koje pobeđuju u igrama nisu one koje daju jako dobre aproksimacije V i Q
  - Tačnije, funkcije osobina mogu da budu loše procene očekivane nagrade ali da pomoću njih donosimo dobre odluke (radimo dobre akcije)
  - Prioritet Q-učenja je da priđemo što bliže najboljim Q-vrednostima
  - Prioritet pri odabiru akcija nisu baš same Q-vrednosti već njihov redosled po kvalitetu koja akcija je bolja od koje, nije bitno za koliko

- Rešenje: učimo baš politiku koja maksimizuje nagrade, a ne Q-vrednosti koje će nam dati tu politiku

- Traženje politike: počinjemo sa nekom „OK“ politikom (koja je recimo rezultat određenog broja iteracija Q-učenja) i onda „štelujemo“ (*fine tune*) politiku
- Štelovanje radimo malim promenama težina, forsiramo one koje nam daju sve bolju i bolju politiku

Traženje Politike

- Najjednostavniji algoritam:
  - Krećemo sa nekom inicijalnom linearnom Q-funkcijom
  - „Čačkamo“ težine gore-dole i gledamo da li nam to popravlja politiku
- Problemi:
  - Kako znamo da li je nova politika bolja?
  - Moramo da pustimo agenta da radi po njoj!
  - Ako ima puno osobina i puno „čačkanja“ ovo je skupo
- Postoje bolje metode, ali izlaze iz materije koju radimo na ovom krusu…

Traženje Politike – Demo Helikopter

[Andrew Ng]