Primena Sarsa algoritma za obučavanje veštačkog agenta za igranje retro igara

MASTER RAD

Student:

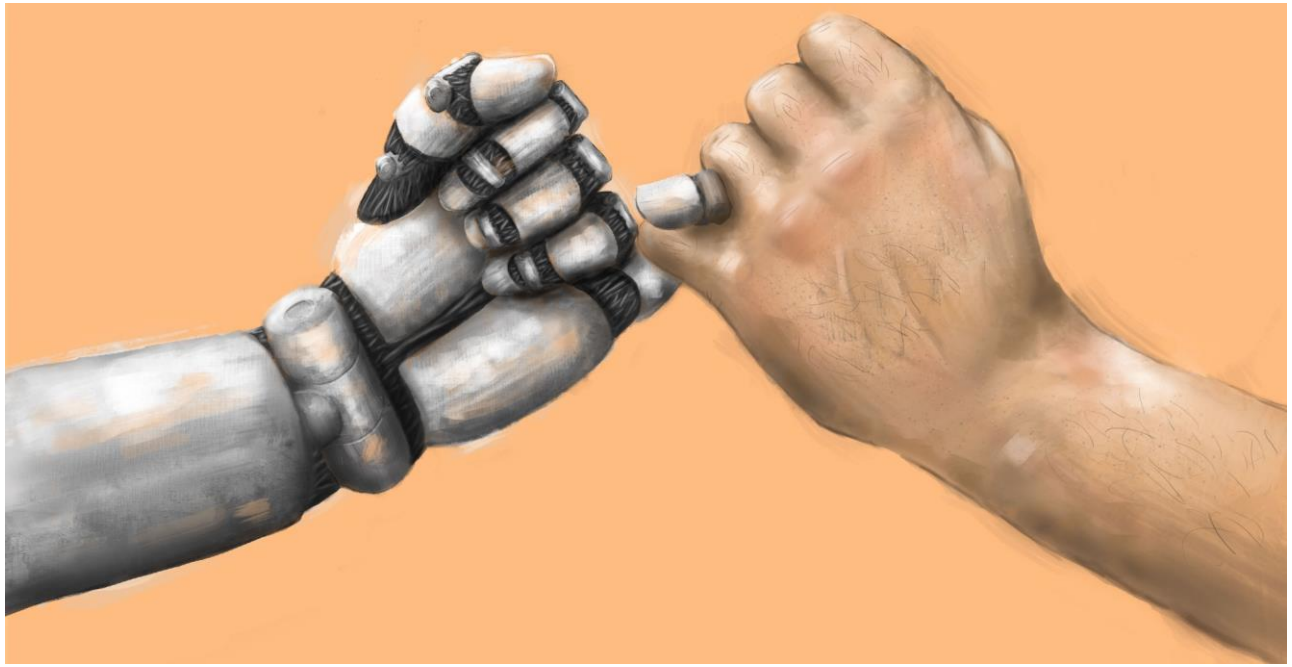
Filip Stamenković

Mentor:

doc. dr Aleksandar Milosavljević

Reinforcement Learning

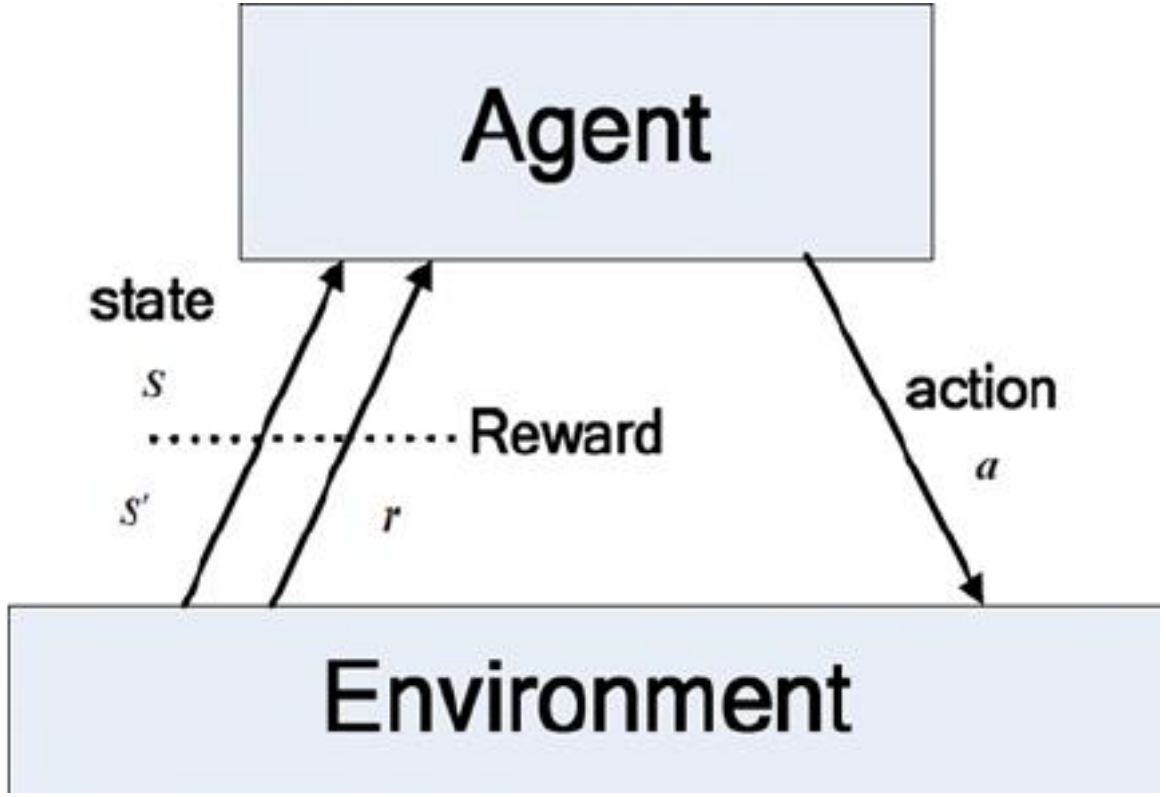
- Podela mašinskog učenja:
  - Nadgledano učenje
  - Nenadgledano učenje
  - *Reinforcement Learning*
- Jednostavan koncept, agent uči samo na osnovu nagrade i kazne.
- Retro video igre su veoma jednostavne i odličan su *benchmark* za RL algoritme.



Artwork by Robert Aguilera

Reinforcement Learning

- Glavni elementi RL-a:
  - Stanje – stanje u kom se agent nalazi
  - Akcija – izvršava je agent nad okruženjem
  - Nagrada – nagrada koju agent dobija
- Cilj agenta je da maksimizuje nagradu koju dobije od okruženja.



Komponente RL-a

- Komponente RL-a (agent može imati jednu ili više):
  - Funkcija vrednosti
    - Stanja – definiše koliku nagradu agent očekuje za dato stanje.
    - Akcije – definiše koliku nagradu agent očekuje za dato stanje i akciju.
  - Polisa – opisuje ponašanje agenta.
  - Model – u zavisnosti od toga da li je agentu model okruženja poznat, algoritmi se mogu podeliti na
    - Model-Based – Agentu je poznat model okruženja, pomoću dinamičkog programiranja ili *planning* algoritama agent može rešiti RL problem
    - Model-Free – Agentu nije poznat model okruženja

Nagrade i kazne

- Cilj agenta da maksimizuje nagradu koju dobija.
- γ – discount faktor, govori koliko agent vrednuje buduće nagrade u odnosu na trenutnu.
- $G_t = \sum_{i=0} \gamma^i R_{t+i}$
- Funkcija vrednosti stanja:
  - $V(S_t) = E[G_t]$
  - $V(S_t) = E[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots]$
  - $V(S_t) = E[R_t + \gamma V(S_{t+1})]$
  - Stanje je „dobro“ koliko nagrada dobijena u tom stanju + očekivana nagrada iz narednog stanja sa uračunatim *discount* faktorom.

Istraživanje vs Iskorišćavanje

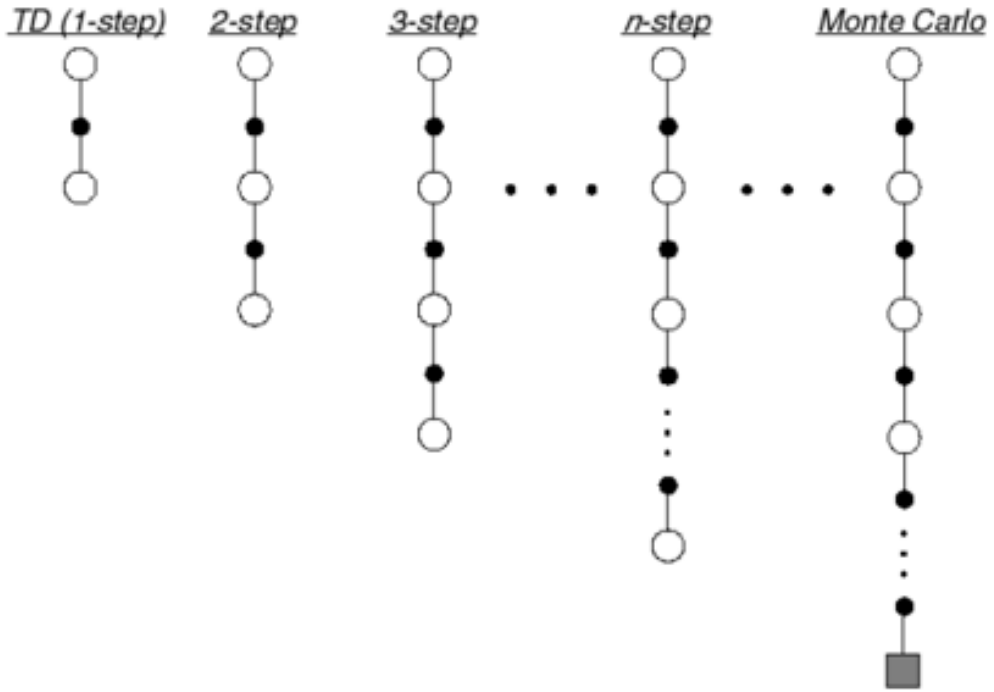
- Istraživanje – obilaženje neposećenih stanja.
- Iskorišćavanja – ići sigurnim putem zarad veće nagrade.
- Agent mora istražiti što više stanja ali da ne gubi previše nagrade usput.
- Analogija za ovaj problem može se naći i u realnom životu, npr. odabir restorana za ručak
  - Istraživanje – ići u novi neposećeni restoran
  - Iskorišćavanje – ići u omiljeni restoran
- ε-*greedy* mehanizam rešava ovaj problem.

Monte Carlo

- MC uči direktno iz interakcije sa okruženjem.
- MC je *Model-Free* algoritam.
- MC uči na osnovu celih epizoda.
- MC ima jednostavnu ideju: vrednost stanja jednaka je prosečnoj vrednosti.
- MC formula za predikciju:
  - $\mathrm{V}(S_t) = \mathrm{V}(S_t) + \frac{1}{\mathrm{N}(S_t)} \left(G_t - \mathrm{V}(S_t)\right)$

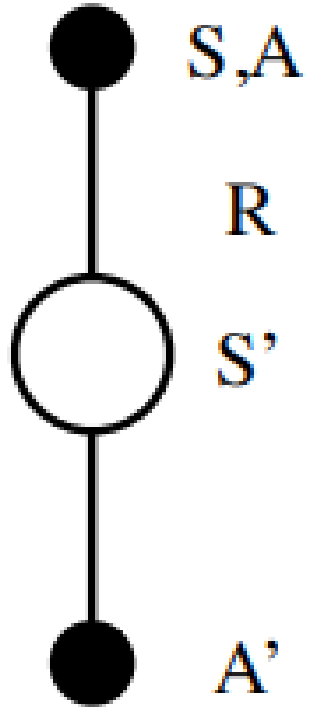
Temporal Difference

- TD isto kao i MC uči direktno iz interakcije sa okruženjem i *Model-Free* je algoritam, ali ne uči na osnovu celih epizoda, nego nakon **n** koraka.
- TD formula za predikciju u 1 korak:
  - $\mathrm{V}(S_t) = \mathrm{V}(S_t) + \alpha\left(R_t + \gamma V(S_{t+1}) - \mathrm{V}(S_t)\right)$
- TD formula za predikciju u **n** koraka:
  - $\mathrm{V}(S_t) = \mathrm{V}(S_t) + \alpha\left(G_t^{(n)} - \mathrm{V}(S_t)\right)$
  - $G_t^{(n)} = \sum_{i=0}^{n-1} \gamma^i R_{t+i} + \gamma^n V(S_{t+n})$
  - α – definiše u kojoj meri novi podaci prebrisuju stare.



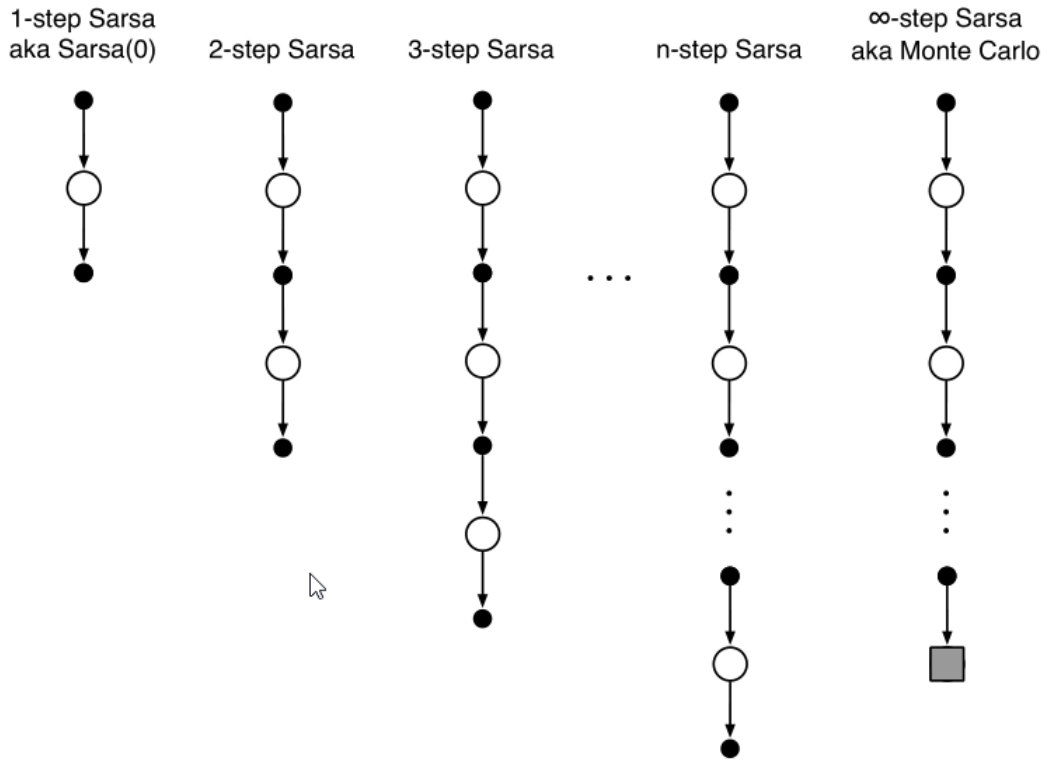
Sarsa

- Sarsa je *on-line* TD algoritam za kontrolu.
- Koristi funkciju vrednosti akcije, a ne stanja.
- Ime je skraćenica koja opisuje način rada algoritma (***S**tate-**A**ction-**R**eward-**S**tate-**A**ction*).
- *On-line* algoritmi svaki put kada izračunaju novu Q vrednost modifikuju i polisu.

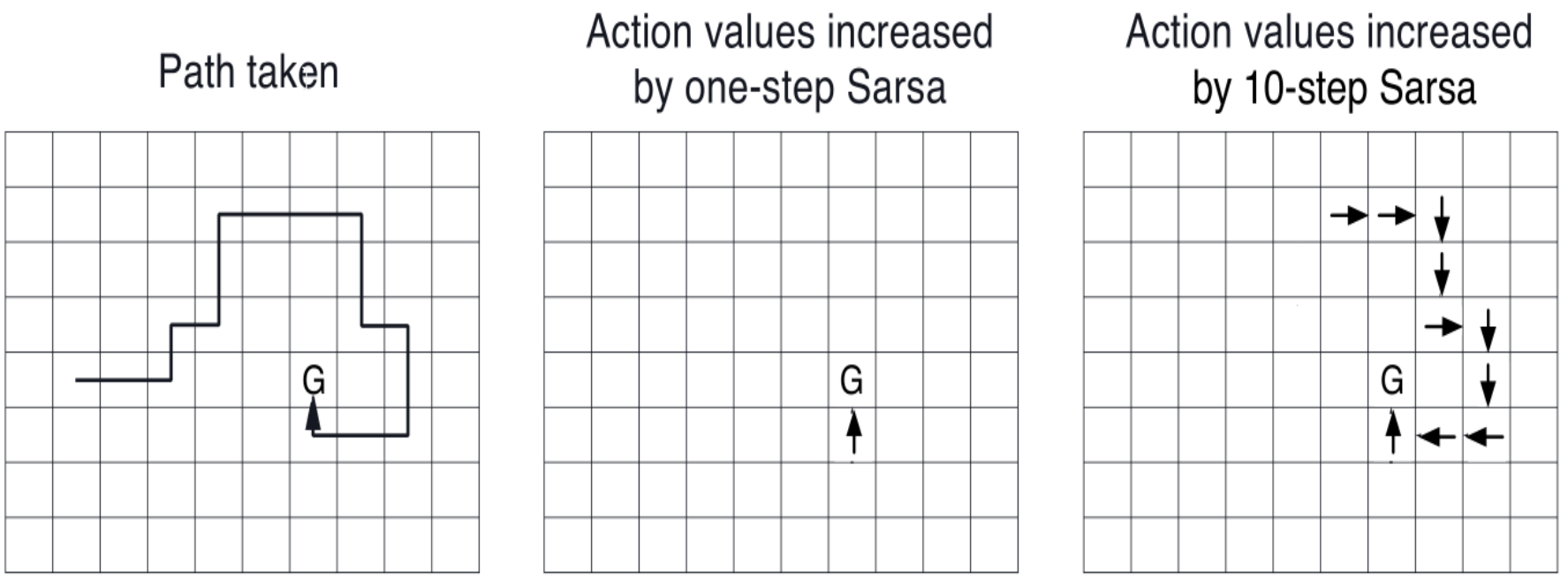


Sarsa u **n** koraka

- Sarsa u **n** koraka je algoritam za kontrolu.
- Funkcija akcije i polisa se modifikuju tek posle **n** koraka.
- Rešava problem „pristrastnosti“.
- Formula za kontrolu u **n** koraka:
  - $\mathrm{Q}(S_t, A_t) = \mathrm{Q}(S_t, A_t) + \alpha\left(q_t^{(n)} - \mathrm{Q}(S_t, A_t)\right)$
  - $q_t^{(n)} = \sum_{i=0}^{n-1} \gamma^i R_{t+i} + \gamma^n Q(S_{t+n}, A_{t+n})$



Sarsa u **n** koraka

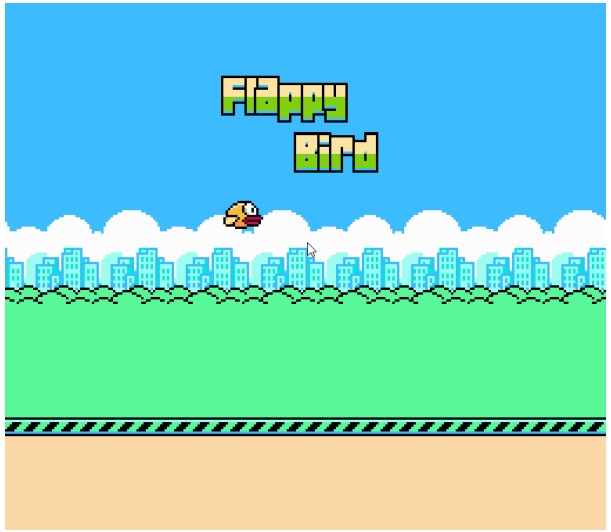


Arcade Learning Environment

- Objektno-orijentisan framework za testiranje veštačkog agenta koji igra Atari 2600 igre.
- ALE je razvijen 2013. godine na Univerzitetu u Alberti.
- Koristi Stella emulator.
- Podrška za preko 50 Atari igre.
- Lako dodavanje nove igre.
- Jezgro emulatora nezavisno od renderovanje slike i zvuka.
- Lak pristup ekranu igre (niz piksela) i ROM-u same igre.
- Mnoga druga okruženja se baziraju na ALE-u: RLE, Xitari, OpenAI Gym...

Flappy Bird

- Besplatna mobilna igra napravljena za Android i iOS platforme.
- Nastala 2013. godine, pravu popularnost stekla 2014.
- Igru je razvio Dong Nguyen u kompaniji dotGEARS.
- Protagonista je ptica koja konstantno leti s leva na desno i na koju konstantno utiče gravitacija.
- Pritiskom na ekran, ptica „poleti na gore“.
- Ptica „gine“ kada udari u tlo ili u prepreku.
- Cilj igre je proći što više prepreka.



Flappy Bird Sarsa

- Postoje samo dve akcije: Skoči (Jump) i bez operacije (NOOP).
- Cilj agenta je: naći optimalnu polisu tako da ptica ne pogine.
- Nagrade:
  - +1 za svaki frame kada je ptica živa
  - -100 kada ptica pogine
- Stanje u kome se ptica može naći se određuje pomoću:
  - Visina ptice: 1-180
  - Smer ptice: 1-5
  - Visina prepreke: 1-180
- Broj različitih „stanje-akcija“ para: 180 * 5 * 180 * 2 = 324.000.
- Ako se visina ptice i visina prepreke zamene jednom karakteristikom: **relativna udaljenost ptice od prepreke**, broj „stanje-akcija“ para: 180 * 5 *2 = 1.800.

Rezultati

Epizoda	Skor
10	0
15	1
21	5
24	21
25	48
30	20
33	102
42	682 (optimalna polisa)

Breakout

- Atari 1976. razvio Breakout arkadnu igricu.
- Na ekranu se nalaze 108 cigle podeljene u 6 reda i 18 kolona, jedna loptica i lopatica.
- Cilj igre je razbiti što više cigle a ne „poginuti“.
- Igra se završava kada igrač „očisti 2 ekrana cigli“ ili pogine 5 puta.
- Igrač kontroliše lopaticu, i njegov zadatak je da lopticu odbije od svoje lopatice kako bi cigle srušio.

Breakout – stanja i akcije

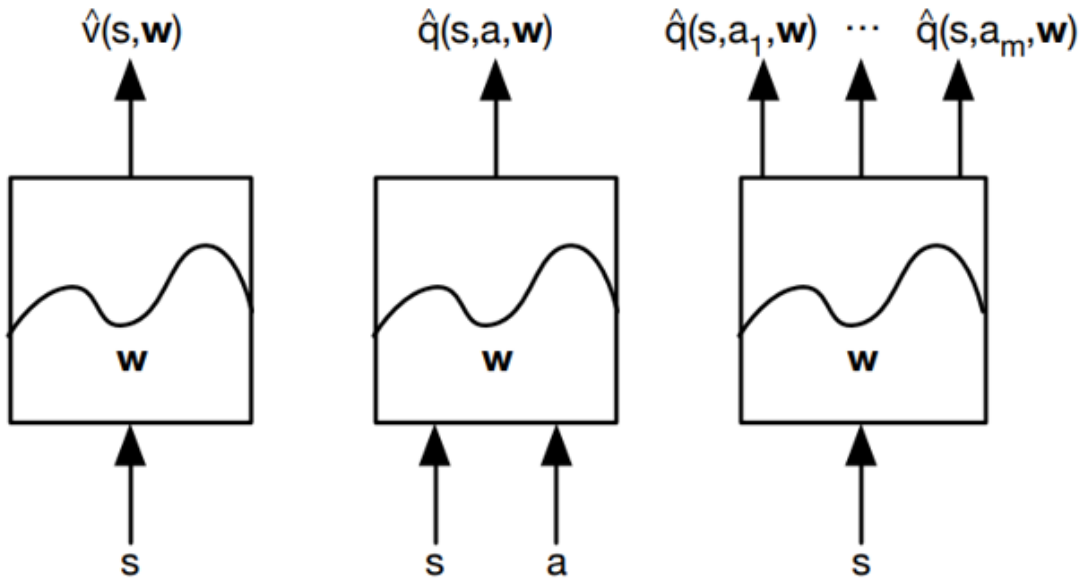
- Akcije koje agent može da izvrši:
  - Levo (LEFT)
  - Desno (RIGHT)
  - Bez akcije (NOOP)
- Stanje u kome se agent može naći je opisano:
  - X pozicija loptice: 250 (mogućih različitih vrednosti)
  - Y pozicija loptice: 250
  - Pozicija lopatice: 250
  - X brzina loptice: 7
  - Y brzina loptice: 7
  - Širina lopatice: 2
  - Promljive koje opisuju koje cigle su srušene: 2^{108}

Breakout – tabelarni pristup

- Broj različitih „stanje-akcija“ para je:
  - $N = 2^{108} * 250^3 * 7 * 7 * 2 * 3 \approx 1.5 * 10^{42}$
- Ako se vrednosti pamte pomoću *float* (4B) promenljivih, potreban RAM je:
  - $4B * N \approx 5.6 * 10^{33}$ GB
- ALE najbrže može da procesira 6.000 *frame*-a u sekundi, tako da minimalno potrebno vreme da se obiđe svaki različit par „stanje-akcija“ poseti jednom je:
  - $1.5 * 10^{42} / (6000\ Hz) \approx 2.6 * 10^{30}$ godina
- Breakout je nemoguće rešiti tabelarno ako se koriste sve neophodne informacije.
- Moguće je napraviti kompromis, i stanje predstaviti samo pomoću **relativne udaljenosti između loptice i lopatice**.
- Agent bi naučio da ne pogine, ali cigle bi rušio „slučajno“.

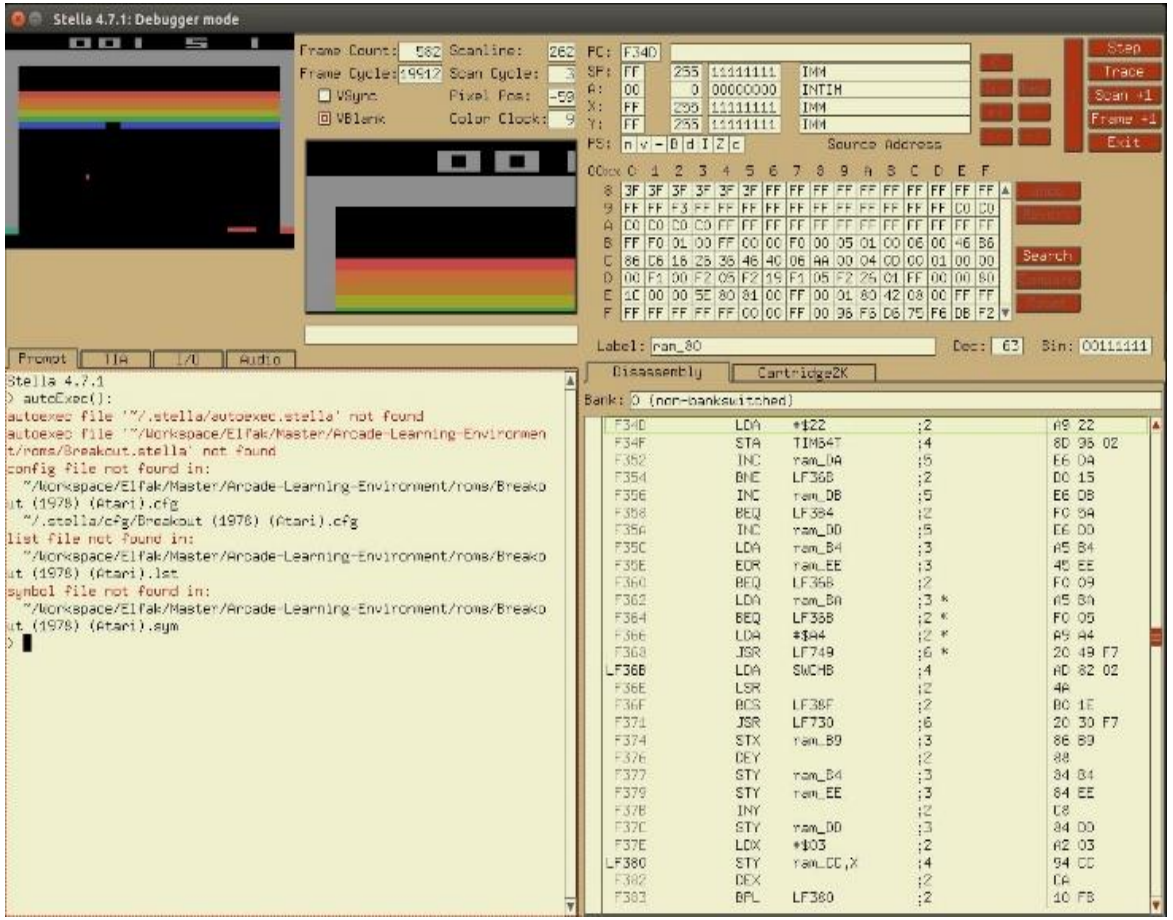
Aproksimator funkcije

- Zbog prevelikog broja stanja i zbog potrebe agenta da generalizuje iz sličnih situacija koristi se aproksimator funkcije.
- Aproksimizacija funkcije se koristi u primenjenoj matematici, u računarstvu kod nadgledanog učenja, prepoznavanja uzorka itd.
- Sve tehnike za aproksimizaciju funkcije iz pomenutih oblasti se mogu koristiti i kod *Reinforcement Learning*-a.
- Najčešće korišćeni aproskimatori kod RL-a su:
  - Linearni – vektor karakteristika (*feature vector*)
  - Nelinearni – neuronska mreža



Breakout – linearni aproksimator

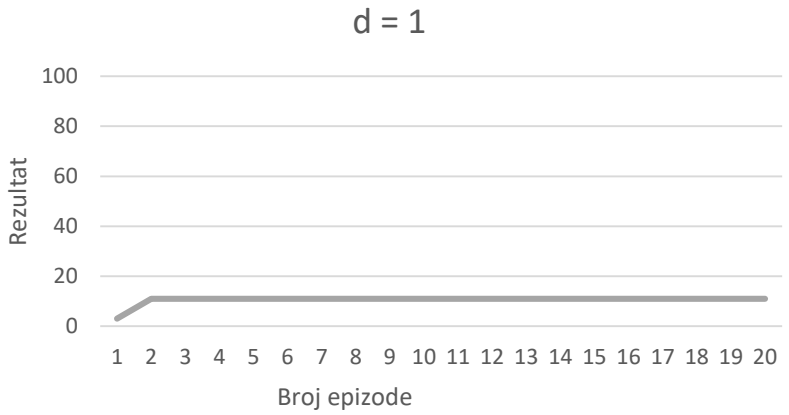
- Analizom igre pomoću Stella *debugger*-a, uočava se da se sva potrebna stanja mogu predstaviti pomoću 45 različitih karakteristika.
- Svaka karakteristika predstavlja vrednost ROM memorije na određenoj lokaciji.

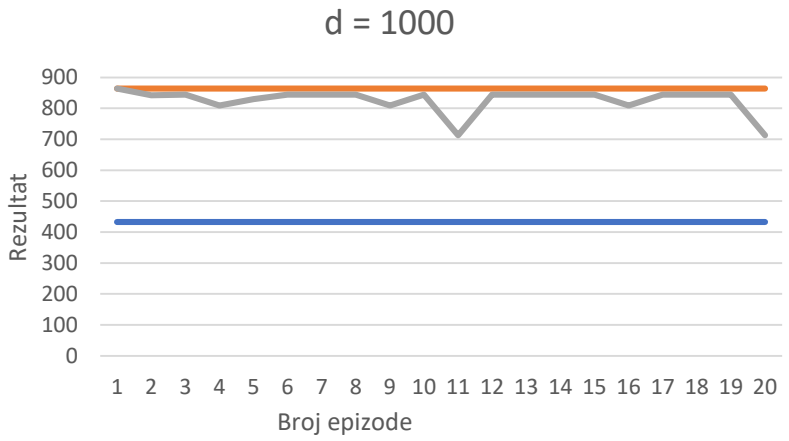


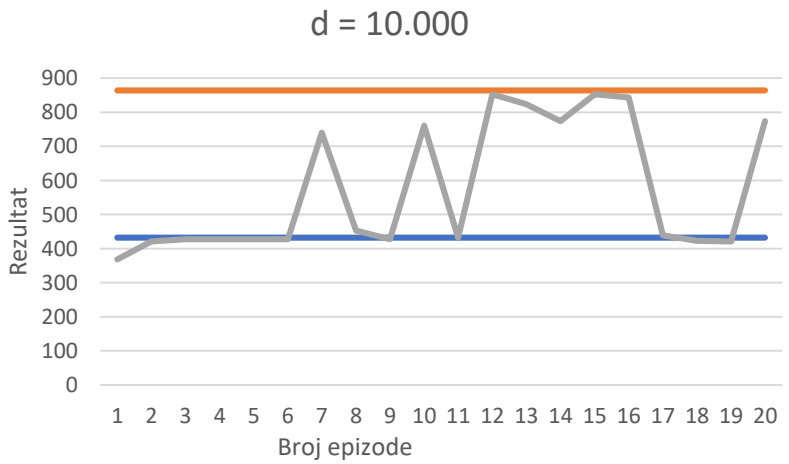
Breakout – linearni aproksimator

- Agent ne uspeva da uoči bitne karakteriste i uvek bira akciju **Levo**.
- Problem rešava nova dodata karakteristika: **relativna udaljenost između loptice i lopatice**.
- Nova karakteristika mora da ima veći opseg vrednosti od ostalih, kako bi agent naučio da je ta karakteristika najvažnija.

Breakout – rezultati







Rezultati

- Rezultat koji prosečan čovek može ostvariti u igri Flappy Bird je ~ 100.
- Sarsa agent je nakon 42. epizode naučio optimalnu polisu tako da ptica nikad ne „gine“.
- Za Breakout igru, prosečan čovek nakon 2 sata igranja igre ostvaruje prosečan rezultat od 31 poen.
- Sarsa agent je već u prvoj epizodi ostvari maksimalan rezultat od 864.
- Oba Sarsa agenta su ostvarila njihov zadatak, ostvarila su bolje rezultate bolje nego prosečan čovek.

Pitanja?