Elektrotehnički fakultet Beograd

Statistička klasifikacija signala

Projekat

Primena *policy gradients* metoda u igrici Pong

Doc. dr Predrag Tadić

Student: Srđan Jovanović

Jul 2018

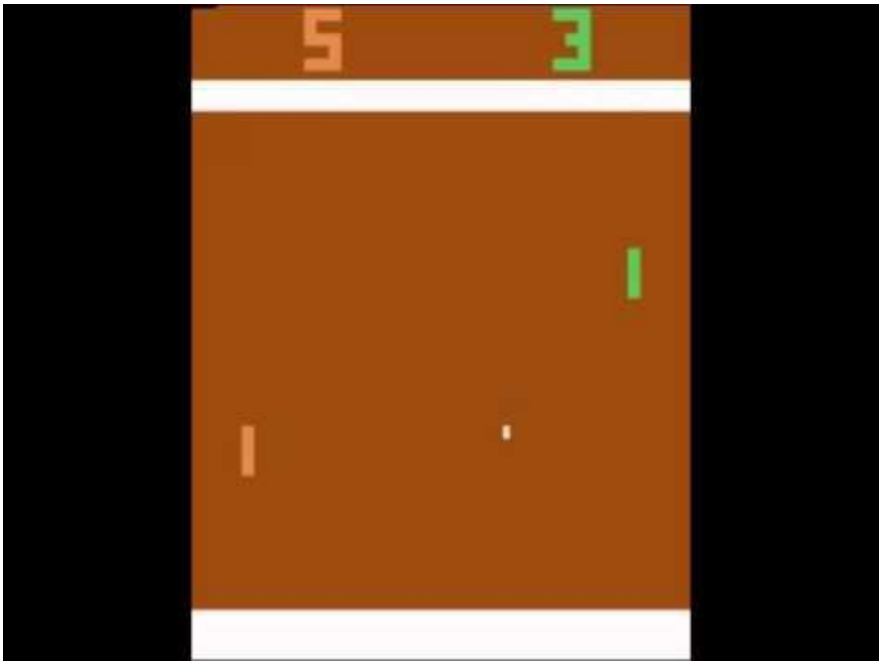
Sadržaj

Uvod

U ovom projektu biće obrađena jedna od najaktuelnijih tema iz oblasti učenja sa podobučavanjem (*reinforcement learning*). Standardni Markovljevi modeli poslužili su kao odlična podloga za razvoj ovog tipa mašinskog učenja jer je omogućeno da se mnogi realni problemi predstave matematičkim modelima. Agent je potpuno opisan samo trenutnim stanjem, ne i prošlim. Ono što je pak njihov nedostatak je izričito zahtevanje poznavanja verovatnoća prelaska među stanjima, kao i sva moguća stanja. Jasno je da ovo nije ostvarivo u realnom životu gde može postojati neograničen broj stanja. Zato se krenulo sa traženjem drugih metoda koje bi se primenile i kada je agent u potpuno nepoznatom okruženju. Za ovim postoji potreba jer je zapravo ovo skoro uvek slučaj. Bilo da je to igranje šaha od strane računara, automatsko navođenje dronova ili roboti koji uče da izvode kompleksne zadatke, agent je gotovo svaki put u nepoznatom okruženju bez ikakvog predznanja od njemu. Metod koji će biti opisan u ovom projektu se naziva *policy gradients*. Ideja je da se pokaže njegova primena na jednom prostom primeru Atari igrice, gde će agent pokušati da nauči pravila igre (Pong) i da pobedi protivnika. Agent ne zna unapred pravila i jedino što dobija od okruženja su nagrade nakon odigranih poteza. Za ovu priliku koristiće se simulirano okruženje *gym* koje je razvijeno od strane *OpenAI* istraživačkog centra.

Okruženje

Najpre je potrebno objasniti okruženje kako bi se razumelo o kakvom problemu se radi. Paket *gym* nudi simulirana okruženja za veliki broj igrica. Za potrebe ovog projekta odabrana je igra Pong, koja ima za cilj da pošalje lopticu na suprotnu stranu tako da je protivnik ne vrati. Ovaj paket ima svoj API i kao ulaz može dobiti jedino akciju koju će agent odraditi, dok kao izlaz daje nagradu, sledeće stanje i informaciju da li je igra završena. Igrica je prikazana na slici 1.

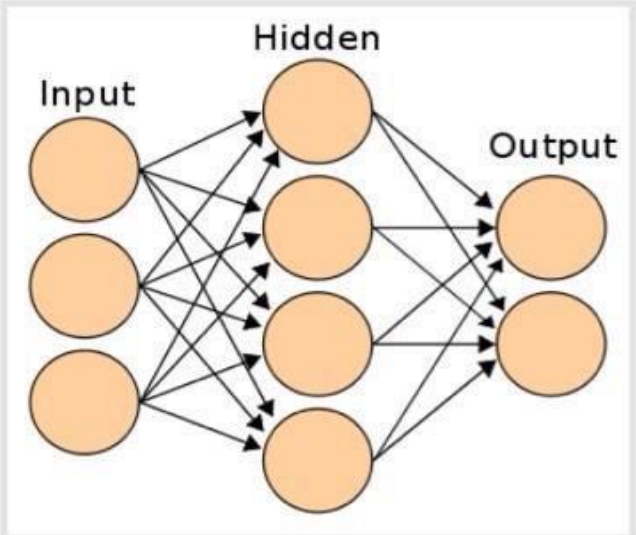


Slika 1: Pong igrica

Igrica se simulira tako što se nakon svake akcije prikazuje frejm (frame). Da bi se dobio vizuelan efekat kretanja loptice i ploča, potrebno je staviti određenu pauzu (100ms) između poteza. Frame je slika dimenzija 210x160x3. Pobednik igrice je ona ploča koja prva sakupi 21 poen. Poen se dobija svaki put kad loptica prođe protivnika. Ukoliko dobije poen naš agent dobija +1, u suprotnom -1. Kada se ne desi poen (lopta putuje sa jedne na drugu stranu) rezultat je 0. Postoje 3 akcije koje je moguće odigrati (gore, dole, u mestu) ali ćemo radi jednostavnosti za ovu priliku koristiti samo akcije gore i dole. To znači da će se naš agent u svakom trenutku pomerati, čak i kada nema potrebe. Ovakve i slične pretpostavke će biti korišćene na još par mesta u projektu radi jednostavni i one nemaju gotovo nikakav uticaj na igru.

Policy gradients metod

Ideja koja stoji iza *policy gradients* je aproksimativno učenje optimalnog zakona upravljanja π. Određivanje optimalnog π je kod poznatih Markovljevih procesa odlučivanja rađeno sa poznavanjem verovatnoća prelaska između stanja. Sa diskretizacijom i velikim brojem stanja ovo jednostavno nije izvodljivo. Zbog toga ćemo definisati *policy network* (mrežu zakona upravljanja) koja je zapravo neuralna mreža koja vrši upravljanje agenta. Mreža će na osnovu ulaza (frejm iz igrice) sračunati verovatnoće za raspoložive akcije. Na slici 2 je prikazana simbolična šema ove mreže.



Slika 2: Dvoslojni perceptron neuralna mreža

Ulaz u mrežu će biti jednodimenzioni niz koji se dobija kada se frejm dimenzija 210x160x3 najpre kropuje na veličinu 80x80x3, zatim se konvertuje iz RGB u Gray i na kraju se matrica se predstavi kao niz 6400x1. Takođe, bilo bi dobro kada bi ulaz u mrežu sadržao bar dva frejma kako bi se detektovao pokret. U projektu je posmatrana razlika između dva susedna frejma. Ovaj niz se dovodi na skriveni sloj od 200 neurona, a odatle na izlaz (output) koji se sastoji od dva neurona gde se svaki od neurona odnosi na jednu moguću akciju. Ovde realno može postojati i samo jedan neuron koji bi davao verovatnoću za akciju GORE, jer su akcije (GORE, DOLE) međusobno isključive ($P_{GORE} = 1 - P_{DOLE}$). Jednačine koje opisuju ovu mrežu su:

$$h_{m\times 200} = ReLU\left(x_{m\times 6400} * W_{1_{6400\times 200}}\right)$$

$$output_{m\times 2} = softmax\left(h_{m\times 200} * W_{2_{200x2}}\right)$$

Softmax funkcija na kraju računa verovatnoću za svaku akciju tako da je $P_{GORE} + P_{DOLE} = 1$.

Ovde je bitno razumeti ponašanje mreže. Kada se dobije ulaz u vidu razlike frejmova, on se propušta kroz mrežu i dobiju se verovatnoće za akcije. Akcija se bira metodom ruleta, tj. svaka akcija ima verovatnoću izbora shodno verovatnoći na izlazu mreže. Pretpostavimo da se izabere akcija GORE. Okruženje na ovakvo ponašanje može da vrati nagradu 0, tj. nije se dogodio nikakav poen (loptica je u međuprostoru između ploča). Nakon toga od okruženja se dobija novi frejm i ponavlja se postupak kroz mrežu. Ovaj proces može da se ponavlja više puta sve dok se ne dogodi poen, odnosno nagrada različita od 0. Kada se posle više ponavljanja

poen dogodi, ne može se tačno reći koja akcija je dovela do poena. Ovo je veoma bitno, jer će upravo *policy gradients* dati rešenje na ovaj problem.

Pre toga, važno je razumeti kako nadgledano (*supervised*) učenje funkcioniše jer će se pomenuti problem rešavati analogno tome. Kod nadgledanog učenja se za svaki ulaz unapred zna korektan izlaz. Ovo je značajno kad treba da se greška propagira unazad kroz mrežu. To bi u našem slučaju značilo da kad se frejm (zapravo razlika frejmova) propusti kroz mrežu odmah zna koja akcija treba da se odabere. Pretpostavimo da je to akcija GORE. To znači da izlaz za tu akciju dobija vrednost 1 i ta vrednost se propagira unazad kroz mrežu (*backpropagation*) kako bi se sračunalo $\nabla_W \log p(y = GORE|x)$. Ovaj gradijent nam kaže kako treba da se promene parametri u mreži tako da mreža ima veću verovatnoću da predvidi akciju GORE kada ona zaista i treba da se dogodi. Na primer, ako neki parametar ima vrednost gradijenta -1.65 znači da ako se vrednost parametra uveća za 0.01, $\log p(y = GORE|x)$ će se na izlazu umanjiti za 1.65 * 0.01. Ako se sada ažurira vrednost tog parametra, mreža će bolje predviđati GORE kada vidi sličan ulaz u budućnosti.

Kod učenja sa podobučavanjem u trenutku kad odaberemo akciju mi ne znamo da li je ona korektna i stoga nemamo pravu labelu. Ukoliko za labelu koju mi odaberemo kažemo da je tačna i izvršimo propagaciju unazad kao da je to ispravno, može doći do veoma loše istrenirane mreže i pogrešnih rezultata. Stoga je bitno da se mreža trenira za labelu koja je tačna. Na sreću postoji način kako može da se dobije takav podatak. Primetimo da je moguće sačekati neko vreme(određeni broj akcija) dok se ne dogodi poen. Do tog trenutka mreža vrši samo propagaciju unapred i akcija se bira stohastički. Kada se dogodi poen dobija se nagrada +1 ili -1. Sada možemo ovu nagradu uzeti za izlazni gradijent svih akcija koje su se izvršile do tog trenutka. Ukoliko je nagrada negativna, to će smanjiti verovatnoću akcija koje su se izvršile ukoliko se pojavi slična situacija u budućnosti. Ukoliko je pak pozitivna, to će ohrabriti mrežu i povećati verovatnoću tih akcija. U ovom projektu nagrada je skalirana stepenom faktora γ (slabljenje, *discount*) tako da u koraku t nagrada iznosi

$$R_t = \sum_k^{\infty} \gamma^k r_{t+k}$$

Funkcija gubitka (*loss*) koja se optimizuje je $J = \sum_i R_i \log P(y_i \mid x_i)$.

Cilj *policy gradients* metoda je da maksimizuje očekivanu nagradu. To očekivanje matematički zapisano glasi $E_{x\sim p(x;\theta)}[f(x)]$ gde je $f(x)$ funkcija nagrade, $p(x)$ mreža upravljanja i θ parametri mreže. Ako se sada sračuna njegov gradijent

$$\nabla_\theta E_x[f(x)] = \nabla_\theta \sum_x p(x)f(x) =$$

$$= \sum_x \nabla_\theta p(x)f(x) = \sum_x p(x)\frac{\nabla_\theta p(x)}{p(x)} f(x) = \sum_x p(x)\nabla_\theta \log p(x) f(x) = E_x[f(x)\nabla_\theta \log p(x)]$$

Dakle, ukoliko se teži da se maksimizuje očekivana nagrada treba ići u smeru gradijenta tog očekivanja, a to je zapravo gradijent funkcije gubitka J.

Implementacija

Ceo kod je pisan u Python programskom jeziku. Korišćen je pomenuti paket *gym* kojim se simulira okruženje igrice. Primer koda za korišćenje ovog paketa je prikazan na slici 3.

```
import gym
env = gym.make('CartPole-v0')
for i_episode in range(20):
    observation = env.reset()
    for t in range(100):
        env.render()
        print(observation)
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        if done:
            print("Episode finished after {} timesteps".format(t+1))
            break
```

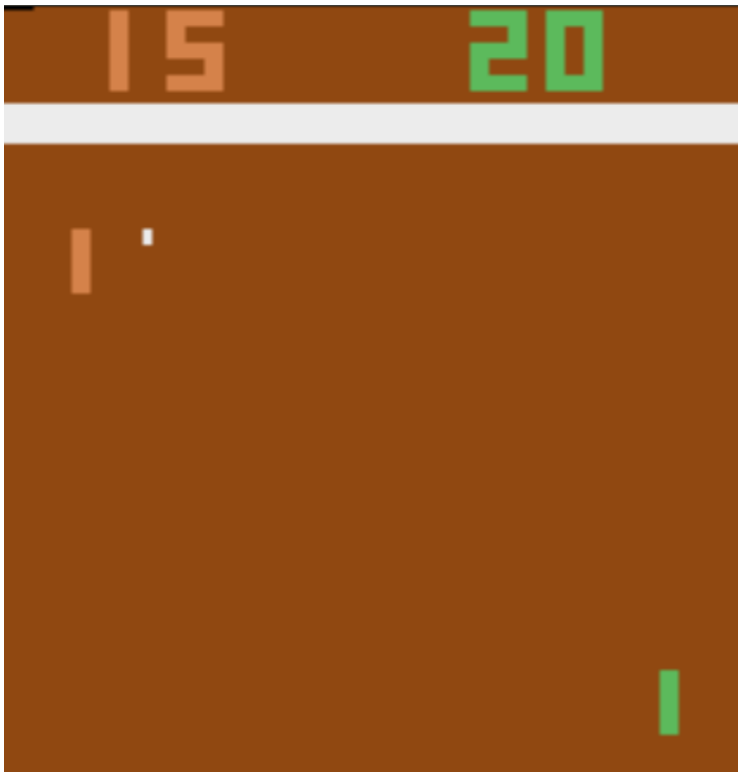
Slika 3: Učitavanje okruženja i slučajno generisanje akcija iz paketa *gym*

Neuralna mreža je realizovana u *TensorFlow* paketu koji podržava *CUDA*-u. Ovo omogućava mnogo brže treniranje mreže nego na *CPU* jer se mnogi procesi paralelizuju.

Mreža je napisana u fajlu *nn_model.py*, dok je interakcija sa okruženjem i prikupljanje nagrada potrebnih za treniranje smešteno u fajlu *ping_pong_policy_gradient.py*. U fajlu *simulate_agent.py* nalazi se kod za pokretanje simulacije istreniranog agenta.

Treniranje

Mreža se trenira nakon određenog broja epizoda koje se smeštaju u *batch*. Epizoda je broj akcija do postizanja prvog poena. Treniranjem se traži minimum funkcije gubitka pomoću *RMSOptimizera* čiji je *learning rate* 1e-3. Ovaj parametar je određen odokativno, bitno je da ne bude mnogo veliki. Tokom treniranja prati se rezultat igre koji je u početku 21 : 0 u korist računara, ali vremenom se razlika smanjuje. Treniranje je trajalo oko 12 sati i nakon toga agent potpuno može da parira računaru. Ukoliko bi se treniranje nastavilo, agent bi bio sve bolji i bolji. U projektu je bilo od interesa bilo pokazati da *policy gradients* funkcioniše uspešno tako da nije bilo potrebe da se agent dalje poboljšava. Takođe, rezultat treniranja se posle izvesnog broja epizoda čuva u *checkpoint* fajlu i ukoliko dođe do prekida treniranja, nakon ponovnog pokretanja, treniranje nastavlja tamo gde je prekinuto. Na slici 4 je prikazan agent nakon treniranja kako pobeđuje računar.



Slika 4: Agent (zelena ploča) pobeđuje računar(narandžasta ploča)

Zaključak

Cilj projekta je bio analiza *policy gradients* metoda i njegova primena na konkretnom problemu. Iako igra Pong nije preterano komplikovana, bilo je veoma izazovno obučiti agenta bez ikakvog poznavanja okruženja i predznanja da nauči da igra. Ovaj projekat je pokazao da ovakav postupak definitivno ima velike mogućnosti i veliki potencijal primene u mnogo složenijim problemima. On još više dobija na značaju ako se zna da neuralna mreža može da simulira zakon upravljanja. To daje dodatan stepen slobode i razvoj u oblasti *deep learning*-a svakako će uticati i na poboljšanja u *reinforcement learning*-u. Policy gradients pokazuju da agent veoma dobro uči samo na osnovu nagrade ne obraćajući pažnju ni na šta ostalo. Iako deluje da agent na kraju razume igru i može da planira, ovo treba uzeti sa rezervom. Agent jednostavno dobija odgovor na svoje ponašanje i ukoliko je odgovor pozitivan on nastavlja sa tim, u suprotnom redukuje takve postupke. Pong igrica je suviše prosta da bi se uvidelo da li agent planira na duže staze jer se poen brzo postiže (već posle par razmena). Pitanje je koliko bi čovek u istoj situaciji mogao brzo da razume igru pre svega zbog ljudske prirode da posmatra neke druge aspekte u okruženju. Takođe, čovek unosi ogromno predznanje, što mu dosta olakšava igru.