SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Treniranje neuronske mreže genetskim algoritmom: lansiranje rakete

David Emanuel Lukšić

Voditelj: Domagoj Jakobović

Zagreb, ožujak, 2016.

**Sadržaj**

[1. Uvod 1](#_Toc451375196)

[1.1 Programska izvedba 1](#_Toc451375197)

[2. Fizikalni model rakete 2](#_Toc451375198)

[2.1 Ulazni parametri 2](#_Toc451375199)

[2.2 Fizikalni parametri 2](#_Toc451375200)

[2.3 Sile 2](#_Toc451375201)

[3. Dizajn neuronske mreže 3](#_Toc451375202)

[3.1 Model mreže 3](#_Toc451375203)

[3.2 Ulazi mreže 3](#_Toc451375204)

[4. Genetski algoritam 4](#_Toc451375205)

[4.1 Kreiranje potomstva i mutacije 4](#_Toc451375206)

[4.2 Algoritam simulacije 6](#_Toc451375207)

[4.3 Parametri genetskog algoritma 6](#_Toc451375208)

[5. Zaključak 8](#_Toc451375209)

[6. Literatura 9](#_Toc451375210)

# Uvod

Algoritmi strojnog učenja počeli su se razvijati 1959. godine kada je Arthur Samuel definirao strojno učenje kao granu znanosti koja proučava mogućnost računala da uče bez eksplicitnih uputa. Od tad su se razvili mnogi algoritmi učenja, među kojima su neuronske mreže i genetski algoritmi, koje ćemo koristiti za naše lansiranje simulirane rakete. Sama simulacija rakete jednostavan je model krutog tijela na kojeg djeluju sile. Cilj lansiranja je dobiti maksimalnu udaljenost od lansirnog mjesta (na desno, pozitivno na x osi) uz ograničeno gorivo. Nakon što „naučimo računalo“ da igra ovu igricu, moći ćemo se i sami natjecati s računalom. Pretpostavka je da će računalo naučiti igrati igricu tako dobro da ga čovjek ne može pobijediti. Očekujemo da će računalo pronaći optimalnu strategiju sagorijevanja goriva / skretanja.

Ovakav sustav mogao bi se koristiti npr. prilikom balansiranja igre, na način da dizajner raketa može u vrlo kratkom vremenu ispitati do kuda najdalje može raketa doletiti za zadanu konfiguraciju, bez puno truda (eliminira potrebu za test igračima). Ukoliko raketa predaleko leti, može smanjiti zalihu goriva, ili pak neki drugi parametar ovisno o tipu rakete. To je vrlo korisna informacija i slične tehnike se koriste u današnjoj industriji video igara.

## Programska izvedba

Ovaj projekt kodiran je u c#-u koristeći Monogame kao engine za simuliranje i crtanje igrice (simulacije). Sastoji se od pet projekata od kojih je jedan zadužen za odrađivanje genetskog algoritma (ConsoleEvolver), a jedan za grafički prikaz simulacije zadnje generacije (RocketSimulator). Oni komuniciraju putem datoteka na način da ConsoleEvolver svakih 1000 generacija zapiše čitavu generaciju u datoteku, koju onda RocketSimulator pročita i simulira korisniku, tako da se korisnik može natjecati sa simuliranim raketama. Osim zadnje generacije, ConsoleEvolver konstantno ispisuje log prosječne i maksimalne dobrote svake 50-te generacije u „.csv” datoteku koju kasnije možemo koristiti u Matlabu za crtanje grafova.

# Fizikalni model rakete

## Ulazni parametri

Raketa je modelirana kao jednostavno dvodimenzionalno kruto tijelo. Njome se upravlja pomoću dva parametra: pozicija zakrilca i potisak. Pozicija (kut) zakrilca je realni broj iz intervala [-1, 1] što se tokom simulacije skalira u kut ovisno o maksimalnom kutu kojeg svaka raketa ima definiranog. Potisak je boolean varijabla koja kaže treba li raketa u nekom trenutku sagorijevati gorivo ili ne. Svaka raketa ima ograničenu količinu goriva.

## Fizikalni parametri

Fizikalni parametri koji opisuju gibanje rakete (osim osnovnih parametara krutog tijela) su sljedeći:

|  |  |
| --- | --- |
| $$v\_{exit}, P$$ | * terminalna brzina, snaga motora
 |
| $$α\_{max}$$ | * maksimalan kut zakrilca
 |
| $$G$$ | * količina goriva
 |
| $$Ω\_{flap}$$ | * koef. otpora zakrilca
 |
| $$Ω\_{body}$$ | * koef. otpora tijela rakete
 |

## Sile

Tijekom leta, na raketu djeluju 4 sile (Slika 1): gravitacija (Fg), otpor zraka na tijelu (Fbody), otpor zraka na zakrilcu (Fflap) i potisak motora (Fengine). Pomoću Fflap i Fengine upravljamo raketom.

|  |
| --- |
|  |
| Slika 1, sile koje djeluju na raketu u letu. |

# Dizajn neuronske mreže

## Model mreže

Neuronske mreže jednostavan su model mozga kojeg nalazimo u živim bićima u prirodi. Sastoji se od mreže međusobno povezanih neurona. Neuronska mreža može imati određen broj ulaznih neurona (eng. *input layer*), izlaznih neurona (eng. *output layer*) i nekoliko skrivenih slojeva neurona (eng. *hidden layer*). Neuroni također imaju definiranu i funkciju aktivacije, kojom transformiraju zbroj svojih ulaza u izlaz. Za potrebe problema lansiranja rakete odabrao sam model 7 – 5 – 2, što znači: 7 ulaznih neurona (ulazni parametri, stanje rakete), 5 neurona u skrivenom sloju i 2 neurona izlaza (ulazni parametri za upravljanje raketom, kut zakrilca i potisak). Neuroni se aktiviraju sigmoidalnom funkcijom (1). Ovaj model ispostavio se kao minimalan uz osobinu da vrlo brzo i kvalitetno uči.

|  |  |  |
| --- | --- | --- |
| (1) | $$S\left(t\right)=\frac{1}{1+ⅇ^{-t}}$$ |  |

## Ulazi mreže

Na uspješnost učenja neuronske mreže uvelike utječe ulaz koji dajemo mreži, zato joj moramo pružiti sve potrebne parametre (ali izbjegavati zalihost) na temelju kojih može zaključiti što napraviti u danom trenutku. Odabranih sedam parametara su sljedeći (standardni instrumenti u letjelicama): brzina propadanja, brzina rotacije, trenutna pozicija na x osi, visina, nagib rakete, preostalo gorivo i brzina vjetra. Brzina vjetra računa se kao projekcija apsolutne brzine na horizontalnu os rakete.

# Genetski algoritam

## Kreiranje potomstva i mutacije

Kako bismo opisali neuronsku mrežu, potrebno je znati njenu topografiju (7-5-2) i sve njene težine. Dakle, naš kromosom je zapravo niz realnih brojeva koji opisuje težine u zadanoj topografiji (u našem slučaju njih 7\*5 + 5\*2 = 45). Prilikom odabira roditelja i kreiranja potomstva koristimo sljedeći algoritam:

|  |
| --- |
|  generiraj početnu populaciju veličine X, slučajne težine; dok (uvjet završetka nije zadovoljen) { slučajno odaberi Y jedinki; // Y < X uzmi najbolju od Y, od nje kreiraj Z potomaka { // Z < X za svaku težinu w[i] u mreži potomka { x = slučajna varijabla od –1 do 1 mutacija = x \* abs(x)^p w[i] = w\_roditelj[i] + mutacija;  } } simuliraj potomke; zamijeni najlošijih Z jedinki s novih Z potomaka; } |
| Algoritam 1 |

Iz algoritma vidimo da je svaka težina u mreži djeteta jednaka težini roditelja, s dodanom mutacijom, kako bi mreža mogla „eksperimentirati“. Mutacije su realizirane kao funkcija slučajne varijable (2), čija vrijednost se dodaje na svaku težinu u novonastaloj mreži. Na taj način dobili smo pomaknutu razdiobu slučajnih vrijednosti gdje je većina oko nule (no skoro nikad točno nula), a poneke vrijednosti imaju veći iznos (ili manji u slučaju da je x negativno). Koliko velike mutacije i koliko često se događaju, kontroliramo varijablom p: Veći p 🡪 manje mutacija 🡪 preciznije, ali sporije traženje (Slika 2).

|  |  |  |
| --- | --- | --- |
| (2) | $$f\left(x\right)=x·\left|x\right|^{p}$$ | $$x\in \left(-1,1\right), p\in \left[0,+\infty \right)$$ |

|  |
| --- |
|  |
| Slika 2, funkcija slučajne varijable korištena za mutacije. |

## Algoritam simulacije

Svaka nova raketa (novi potomak) odrađuje sljedeći algoritam simulacije, čime se određuje kvaliteta svake jedinke genetskog algoritma:

|  |
| --- |
|  dok (vrijeme < 100 ili brzina > 0) { proslijedi parametre neuronskoj mrezi; upravljaj raketom pomocu izlaza neur. mreze { ako neuron[1] > 0.5 { akceleriraj raketu; troši gorivo; } pozicija zakrilca = neuron[2]; } simuliraj fiziku rakete za jednu sliku { simuliraj kruto tijelo rakete; dodaj silu frontalnog otpora; dodaj silu bočnog otpora; dodaj silu otpora na zakrilcu; } vrijeme++; // 1 slika = 1 vrijeme } |
| Algoritam 2 |

Kada se algoritam završi (raketa ima brzinu 0, a prošlo je više od 100 slika), raketa se ocjenjuje prema svojoj horizontalnoj poziciji od lansirnog mjesta. Ta vrijednost uglavnom bude između 0 i 30 000.

## Parametri genetskog algoritma

Odabrati parametre nije tako jednostavan posao. Najbolje je probati s nekoliko vrijednosti dok se ne dobije prihvatljiv omjer opširnog traženja i preciznosti tako da ne zapne u lokalnom minimumu, a opet pronađe optimalno riješenje. U nastavku ćemo razmotriti grafove napretka za različite parametre X, Y i Z (Algoritam 1). Primijetite (Slika 3) da povećanjem populacije dobivamo sporiji, ali kontinuiran napredak, dok se kod manje populacije vrlo brzo usvajaju promjene na bolje, no onda tamo algoritam stoji do sljedeće povoljne mutacije, ponekad vrlo dug vremenski period!

|  |
| --- |
|  |
| Slika 3, grafovi napretka uz razne parametre. |

# Zaključak

Vidjeli smo kako se neuronske mreže u kombinaciji s genetskim algoritmom mogu koristiti za učenje računala da obavlja neki zadatak. U konkretnom primjeru računalo je naučilo lansirati raketu s iznimnom preciznošću, koju ljudski igrači ne mogu postići. Igricu je probalo igrati nekoliko igrača i nitko nije došao ni blizu udaljenosti koju preleti računalo. Zašto bi nam to bilo korisno?

Prilikom izrade (kvalitetnih) video igrica, velik dio vremena troši se na balansiranje sustava. Pogledajmo na primjeru lansiranja rakete. Recimo da korisnik za svakih 10 metara koje preleti dobije određen iznos novca. Kao dizajner igrice, ne želite mu dati toliko novca da može kupiti nadogradnju rakete koja će donijeti puno novca vrlo brzo, no ne želite niti učiniti igru presporom tako da korisnik ne vidi napredak. Kako bi održali igrača zainteresiranog, trebate balans između te dvije krajnosti.

Testirati sve moguće nadogradnje i njihove utjecaje na optimalne strategije zahtijevalo bi puno vremena od strane testera. Umjesto toga, dizajner može u vrlo kratkom vremenu saznati kako promjena nekog parametra utječe na maksimalni domet te rakete te na taj način odrediti koliko će ta nadogradnja koštati (ili obratno, postaviti promjenu parametara rakete prema cijeni). Rezultat ovakvog pristupa je automatiziran i vrlo efikasan način provjere rada sustava same video igrice.

Što je moglo bolje? Mogli smo probati druge verzije genetskih algoritama te provjeriti postoji li neki koji bolje paše zadanom problemu. Simulacije same po sebi nisu previše zahtjevne, stoga bi se mogle simulirati masovno paralelno na grafičkoj kartici, što bi omogućilo veće populacije, stabilnija i iscrpnija pretraživanja. Parametri rakete bi se također mogli automatski korigirati tako da podilaze nekim predefiniranim zahtjevima. Npr. povećaj gorivo toliko da raketa odleti 20 m dalje. Uvesti kašnjenje ulaza neuronske mreže kako bi se realnije simulirao korisnik (oko – ruka koordinacija kod čovjeka kasni oko 100 – 150 ms, što bi u slučaju video igrice, koja se vrti na 60 Hz, bilo oko 6 do 8 slika!).

# Literatura

1. Marko Cupic, Bojana Dalbelo Basic, Marin Golub, Neizrazito, evolucijsko i neuroracunarstvo., 12 Kolovoza 2013.
2. Avni Rexhepi, Adnan Maxhuni, Agni Dika, Analysis of the impact of parameters values on the Genetic Algorithm for TSP, Znanstveni rad, University of Pristina, Kosovo, Siječanj 2013.
3. Monogame, 2013., <https://github.com/mono/MonoGame>
4. Alex J. Champandard, Making Designers Obsolete? Evolution in Game Design, February 2012., <http://aigamedev.com/open/interview/evolution-in-cityconquest/>