Razvoj inteligentnog računalnog igrača

Kopačević, Mateo

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1695

RAZVOJ INTELIGENTNOG RAČUNALNOG IGRAČA

Mateo Kopačević

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1695

RAZVOJ INTELIGENTNOG RAČUNALNOG IGRAČA

Mateo Kopačević

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1695

Pristupnik:	**Mateo Kopačević (0036533644)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	izv. prof. dr. sc. Marko Čupić

Zadatak: **Razvoj inteligentnog računalnog igrača**

Opis zadatka:

Računalne igre ljudima su oduvijek bile interesantne. Međutim, za mnoge igre teško je razviti računalnog igrača koji u igri ostvaruje jednako ili čak i bolje ponašanje u odnosu na ljudskog igrača. U okviru ovog rada potrebno je odabrati jednu računalnu igru te za nju razviti inteligentnog računalnog igrača. Potrebno je opisati igru, način razvoja igrača te komentirati dobivene rezultate.

Rok za predaju rada: 14. lipnja 2024.

Želim izraziti iskrenu zahvalnost svima koji su mi pomogli u realizaciji ovog rada. Prije svega, zahvaljujem svom mentoru Marku Čupiću na njegovoj neizmjernoj podršci, strpljenju i vodstvu kroz cijeli proces istraživanja i pisanja. Njegovo znanje i stručnost bili su od ključne važnosti za razvoj i završetak ovog rada.

Posebnu zahvalu upućujem svojoj obitelji i prijateljima, koji su me neprestano motivirali i pružali emocionalnu podršku. Njihovo razumijevanje i ohrabrenje bili su izvor snage tijekom svih izazova s kojima sam se suočavao.

Također bih želio zahvaliti svojim kolegama i suradnicima na korisnim diskusijama, savjetima i pomoći tijekom istraživanja. Njihov doprinos bio je neprocjenjiv i pomogao mi je proširiti vidike i pristupe problemu.

Svima vama, hvala od srca

Sadržaj

1. Uvod

Razvoj inteligentnih računalnih igrača predstavlja jedno od najizazovnijih područja u suvremenom računalnom znanstvu. Ova disciplina spaja elemente umjetne inteligencije, strojnog učenja, algoritama optimizacije i teorije igara kako bi stvorila autonomne agente sposobne za rješavanje kompleksnih problema u različitim igrama. Jedna od igara koja se ističe svojom izuzetnom težinom i izazovnim mehanikama je jednostavnija verzija popularne igre "Gradius", nazvana "Cubius". Ova igra, koja testira reflekse, preciznost i strateško razmišljanje igrača, predstavlja idealan poligon za istraživanje i razvoj naprednih algoritama za upravljanje računalnim igračima.

U ovom radu, fokusirat ćemo se na proces dizajniranja i implementacije inteligentnog računalnog igrača za "Cubius". Cilj je istražiti različite pristupe i metode koje omogućuju računalnom agentu da uspješno savlada nivoe igre koji su poznati po svojoj izuzetnoj težini. Analizirat ćemo primjenu genetskog optimizacijskog algoritma na učenje višeslojne potpuno povezane neuronske mreže kako bi pronašli zadovoljavajući rezultat s obzirom na kompleksnost stanja igre.

Također, istražit ćemo kako različiti faktori, pout brzine i položaja neprijateljskih objekata , utječu na performanse inteligentnog igrača. Na temelju tih analiza, pružit ćemo uvid u ključne izazove i ograničenja s kojima se susrećemo pri razvoju autonomnih agenata za igre visoke složenosti.

Ovaj rad ima za cilj ne samo unaprijediti razumijevanje postojećih tehnika u području inteligentnih računalnih igrača, već i doprinijeti daljnjem razvoju inovativnih rješenja koja mogu biti primijenjena i u drugim igrama i kontekstima gdje je potrebna visoka razina autonomije i inteligencije računalnih agenata.

2. Glavni dio

2.1. Uvod u igru "Cubius"

"Cubius" je igra kreirana po uzoru na popularnu "Shoot 'em up" igru iz 1980-ih godina poznatu kao "Gradius" koju je proizvela poznata i cijenjena japanska tvrtka Konami. "Cubius" se od svog uzora razlikuje u očitim vizualnim opaskama što su igrač i neprijatelji "kocke", no budući je igra dvodimenzionalnog karaktera prikazani su kao kvadrati.

Igrač kontrolira mali plavi kvadrat koji mora proći kroz niz neprijateljskih objekata kako bi što duže preživio i time akumulirao što veći rezultat. Glavni izazovi uključuju precizno manevriranje kroz uske prolaze, izbjegavanje brzih i nepredvidivih neprijateljskih objekata te savladavanje kompleksnih situacija koje zahtijevaju strateško planiranje. Svaka pogreška rezultira završetkom igre te vraćanjem na početak, što dodatno pojačava težinu igre.

Igra je zabavnog karaktera te je namijenjena jednako svim uzrastima i pojedincima različitih sposobnosti. Igra predstavlja dobar izazov za one koji žele ići korak dalje i testirati svoje vještine i usavršavati reflekse.

2.1.1. Pravila igre

Pravila igre su jednostavna ali stvaraju kompleksne i teško savladive situacije za igrača koje testiraju igračeve reflekse i brzinu donošenja odluka.

1. **Igrač**:
    - Igrač je prikazan kao mali plavi kvadrat.
    - Igrač se može kretati u jednom od četiri pravca: gore, dolje, lijevo, desno.
    - Kombinacijom dvaju pravaca istovremeno postiže se dijagonalno kretanje
    - Igrač ne može izaći izvan granica ekrana, što znači da svaki pokret koji bi igrača odveo izvan ekrana neće biti izveden.
2. **Neprijatelji**:
    - Neprijatelji su prikazani kao crveni kvadrati veličine koja je slučajno odabrana između veličine igrača i dvostruke veličine igrača
    - Neprijatelji se pojavljuju s desne strane ekrana i kreću se prema lijevoj strani konstantnom brzinom.
    - Svakom iteracijom, igra provjerava postoji li 15% šansa da se stvori novi neprijatelj.
    - Ako neprijatelj dosegne lijevu stranu ekrana, on nestaje s ekrana.
    - S obzirom na rastuću brzinu igre, neprijatelji postaju sve teži za izbjegavanje kako vrijeme prolazi.

3. **Novčići**:
    - Novčići su posebni objekti koji se povremeno pojavljuju na ekranu.
    - Ako na ekranu nema novčića, igra će svakih 10 sekundi pokušati stvoriti novi novčić.
    - Novčići se stvaraju na nasumičnom mjestu unutar granica ekrana.
    - Novčić nakon stvaranja ostaje na istom mjestu sve dok ga igrač ne pokupi
    - Nakon što igrač pokupi novčić, on nestaje s ekrana
4. **Konfiguracija igre**:
    - Igra započinje stvaranjem igrača na fiksnom mjestu, u sredini ekrana (gledajući y koordinatu), te nekoliko piksela od lijevog ruba ekrana
    - Igračeva brzina ostaje ista tijekom trajanja igre
    - Jedna iteracija igre ("frame") traje 15 ms, time je ostvareno fiksnih 66 - 67 iteracija po sekundi ("frames per second")
    - Svakih 30 sekundi, brzina igre se povećava za 1, čineći neprijatelje bržim
    - Povećanje brzine znači da igra postaje sve teža kako vrijeme prolazi, zahtijevajući od igrača brže reakcije i preciznije pokrete.
5. **Bodovanje i cilj**:
    - Igrač započinje igru s nulom bodova.
    - Svakih pola sekunde rezultat se automatski povećava za 1 bod, reflektirajući vrijeme preživljavanja.
    - Pokupiti novčić donosi dodatnih 50 bodova.
    - Cilj igre je preživjeti što je dulje moguće, izbjegavajući neprijatelje i skupljajući novčiće kako bi se postigao što veći rezultat.

2.1.2. Grafički dizajn igre

Glavni izbornik

Pokretanjem programa otvara se glavni izbornik u kojemu odabiremo željenu akciju. U igri je korišten font "Press start" preuzet s interneta [1] za prikaz teksta. Navedeni font prisutan je u svim grafičkim dijelovima igre.

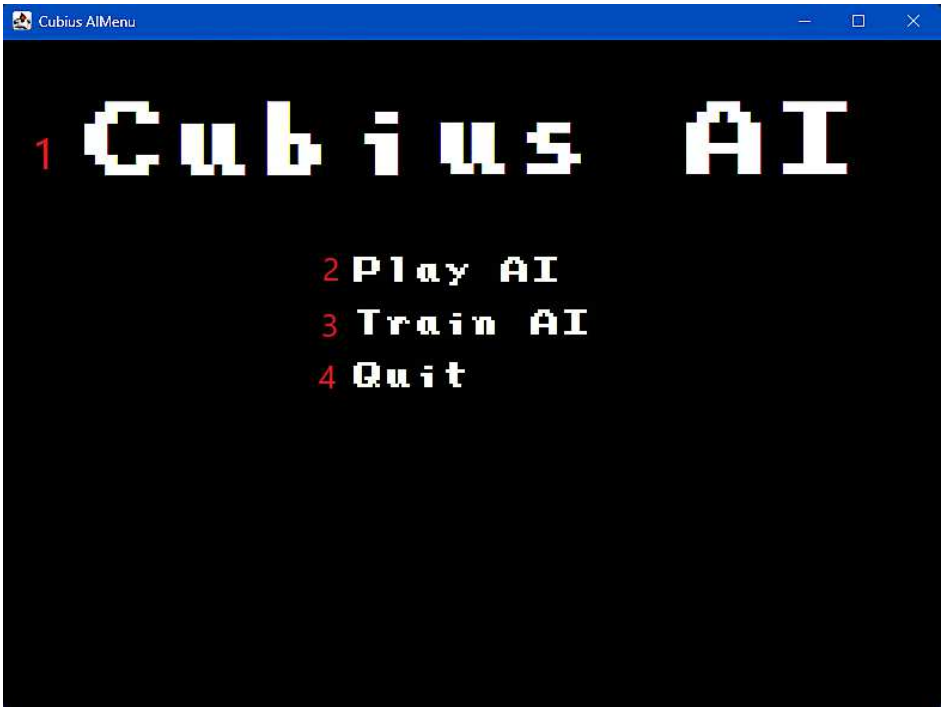
Slika 2.1. Glavni izbornik igre Cubius

1. Naslov same igre Cubius
2. Gumb koji započinje igru koju kontrolira osoba/korisnik
3. Gumb koji pokreće izbornik za upotrebu umjetnog inteligentnog računalnog igrača
4. Gumb koji zatvara prozor te izlazi iz igre
5. Dio ekrana u kojemu se beskonačno odvija animacija u kojoj plavi kvadrat izbjegava stupove crvenih kvadrata (nije predstava same igre)
6. Pozadina je čisto crna a tekst bijeli

[1] Font `PressStart` je licenciran pod `1001Fonts general font usage terms`. Više informacija na `https://www.1001fonts.com/press-start-font.html`.

Izbornik za upotrebu umjetnog inteligentnog računalnog igrača

Odabirom drugog gumba na glavnom izborniku pokreće se izbornik za upotrebu umjetnog inteligentnog računalnog igrača.

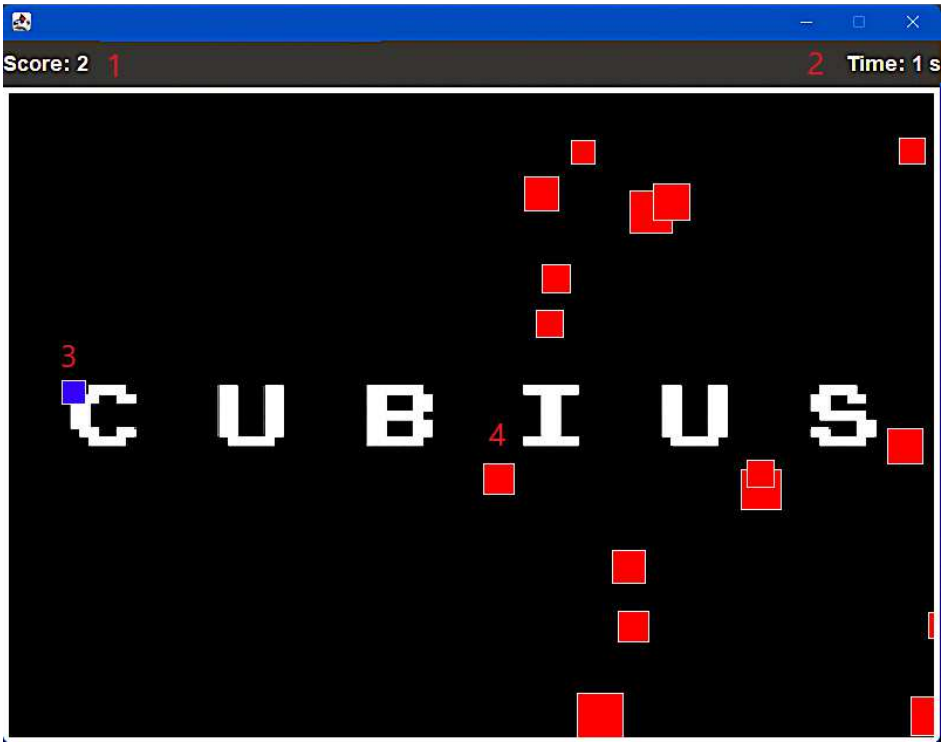


Slika 2.2. Izbornik za upotrebu umjetnog inteligentnog računalnog igrača

1. Naslov izbornika što je samo naslov igre uz dodatak AI
2. Gumb koji započinje igru koju kontrolira umjetni inteligentni računalni igrač
3. Gumb koji započinje treniranje umjetnog inteligentnog računalnog igrača
4. Gumb koji zatvara prozor te izlazi iz igre
5. Pozadina je čisto crna a tekst bijeli

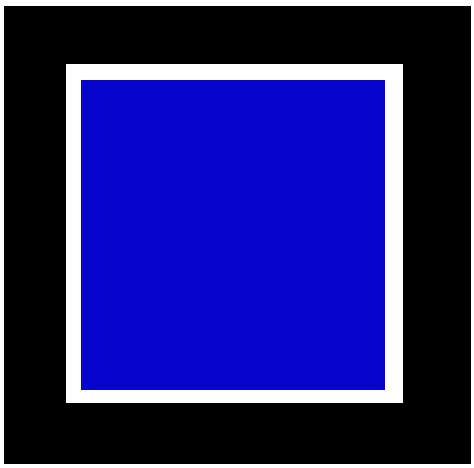
Igra Cubius

Odabirom prvog gumba na glavnom izborniku ili prvog gumba na izborniku za upotrebu umjetnog inteligentnog računalnog igrača pokreće se prozor u kojemu se odvija igranje same igre (korisnik ili umjetni igrač). Ekran je podijeljen u 2 dijela: *gornji dio* u kojemu se prikazuju rezultat na lijevoj strani te proteklo vrijeme u sekundama na desnoj strani i *donji dio* u kojemu se prikazuje igra.



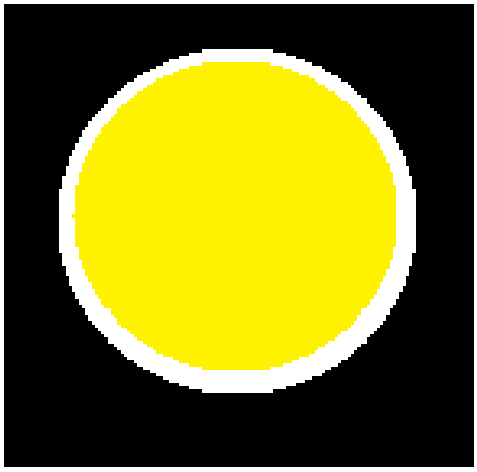
Slika 2.3. Igra Cubius

1. Trenutno ostvareni rezultat, ažurira se svakih pola sekunde ili kada igrač pokupi novčić
2. Trenutno proteklo vrijeme, ažurira se svake sekunde
3. Igrač, započinje igru uz lijevi rub ekrana na sredini
4. Neprijatelj, započinje igru na samom desnom rubu ekrana, nestaje dolaskom na lijevi rub ekrana
5. Pozadina je čisto crna uz tanki bijeli obrub oko čitavog dijela za prikaz igre te velikim naslovom igre u sredini



Slika 2.4. Igrac

Slika 2.5. Neprijatelj



Slika 2.6. Novčić

Slika 2.7. Entiteti u igri

2.2. Umjetna Inteligencija

Umjetna inteligencija (UI) predstavlja područje računalne znanosti koje se bavi razvojem sustava koji imaju sposobnost obavljanja zadataka koji inače zahtijevaju ljudsku inteligenciju.

Inteligencija u raznim područjima raznih prirodnih i društvenih znanosti poprima različita značenja, no u kontekstu računalne znanosti inteligencija označava ponašanje slično čovjeku, mogućnost zaključivanja i donošenja odluka u odnosu na određene informacije koje koristi za izvršavanje tih odluka.

Cilj umjetne inteligencije jest naučiti računalo, jer jedna umjetna inteligencija nije ništa drugo nego računalni program, ponašati se kao čovjek i obavljati određene funkcije koje samo čovjek može ali sve to odrađivati puno brže i u puno većoj količini od čovjeka.

Ova tehnologija temelji se na algoritmima i modelima koji omogućuju računalima da uče iz podataka, prepoznaju obrasce, donose odluke te rješavaju probleme. Glavni grane umjetne inteligencije uključuju strojno učenje, duboko učenje, obradu prirodnog jezika i računalni vid.

UI ima široku primjenu u mnogim industrijama, uključujući zdravstvo, financije, automobilsku industriju i robotiku. Unatoč mnogim prednostima, postoji i zabrinutost zbog etičkih pitanja, privatnosti i mogućnosti zamjene ljudskih poslova automatizacijom.

2.2.1. Strojno Učenje

Strojno učenje predstavlja modeliranje i programiranje računala da na određene načine pronalazi rješenja za dane probleme, poboljšavajući i optimizirajući ta rješenja kroz proces učenja. Učenje se odvija na podatkovnom skupu primjera ili prethodnog iskustva računala. Nad modelom se definiraju parametri koji se onda učenjem optimiraju i na poslijetku dovode do boljeg i boljeg rješenja.

Cilj strojnog učenja jest stvoriti računalo koje može predviđati svojstva na novim primjerima koji se nisu koristili pri učenju modela, odnosno mora moći generalizirati. Problemi koje rješavamo strojnim učenjem jesu vremenska i memorijska složenost koji bi se javili primitivnim načinima rješavanja određenih problema. Stoga se rješenje svodi na pretvorbu stvarnih situacija u numeričke predstave nad kojima se obavljaju određene matematičke operacije što uvelike ubrzava proces potrage za rješenjem.

Postoje 3 vrste strojnog učenja:

- **Nadzirano učenje** - učenje nad primjerima gdje su poznate i ulazne informacije i očekvani rezultati, ako je rezultat nebrojčana vrijednost govorimo o klasifikaciji, a ako jest brojčana govorimo o regresiji
- **Nenadzirano učenje** - učenje gdje nam nisu unaprijed poznati očekivani nego je potrebno naći pravilnosti i uzorke u podacima
- **Podržano učenje** - učenje gdje istražujemo optimalne strategije na osnovu pokušaja s nagradama (kaznama)

Pri strojnom učenju može doći do *prenaučenosti*. Prenaučenost se javlja kada se model prilagodio viđenim podacima te je stoga samo dobar za te podatke a na još neviđenim podacima daje jako loše rezultate. Kako bi izbjegli prenaučenost vodimo se politikom da model umjetne inteligencije treba učiti na 40% ukupnih podataka što čini skup za učenje, a ostatak podijeliti na dva dijela: skup za provjeru (30%) i skup za testiranje (30%). Skup za provjeru upotrebljavamo za provjeru da model nismo prenaučili, a skup za testiranje je konačni skup na kojem evaluiramo koliko dobro funkcionira naš model.

Slika 2.8. Graf strojnog učenja [1]

2.2.2. Neuronska Mreža

Postoje mnogi modeli umjetne inteligencije no jedan od najzanimljivijih te model koji se koristi u ovom radu jest umjetna neuronska mreža.

Neuronske mreže su model inspiriran biološkim neuronskim sustavima, posebno ljudskim mozgom. One se sastoje od čvorova (neurona) povezanih međusobno putem veza (sinapsi). Svaki čvor prima ulazne signale, obrađuje ih pomoću aktivacijske funkcije, te šalje izlazne signale dalje kroz mrežu. [2]

Svaka veza između ulaza i neurona ili 2 neurona ili neurona i izlaza poprima određenu težinu koja je jedan decimalni broj. Težine su parametri modela umjetne inteligencije koje nastojimo prilagoditi i optimizirati tijekom strojnog učenja.

Informacije od ulaza do izlaza neuronske mreže prolaze put mnogih matematičkih operacija koje kulminiraju skupom linearnih kombinacija svih neurona prema određenom izlazu. Ovisno o konfiguraciji neuronske mreže, podaci prolaze kroz aktivacijsku funkciju (funkcija prijelaza) koja na određeni način dobivene podatke transformira u druge podatke. Neke od aktivacijskih funkcija su:

- Izravan prijenos bez promjene

$$f(\text{net}) = \text{net}$$

- Funkcija skoka

$$f(\text{net}) = \begin{cases} 0 & \text{za net} < 0 \\ 1 & \text{inače} \end{cases}$$

- Sigmoidalna funkcija

$$f(\text{net}) = \frac{1}{1 + e^{-\text{net}}}$$

Neuronske mreže su posebno uspješne u zadacima kao što su prepoznavanje uzoraka, klasifikacija podataka, obrada prirodnog jezika i analiza slika. Duboke neuronske mreže, poznate i kao duboko učenje, su vrsta neuronskih mreža s više slojeva, koje omogućuju obradu složenijih podataka. Ove mreže uče putem algoritama kao što su gradijentni spust, koji prilagođava težine kako bi minimizirao pogrešku mreže. [3]

Primjena neuronskih mreža sve više raste u područjima poput medicinske dijagnostike, autonomnih vozila, preporuka proizvoda i personalizirane medicine. Unatoč uspjesima, izazovi uključuju interpretaciju modela, potrebu za velikim skupovima podataka i računalnu snagu potrebnu za treniranje kompleksnih modela.

2.2.3. Primjena u igri

Pri implementaciji strojnog učenja korištenjem neuronske mreže za igru Cubius potrebno je prethodno prikladno definirati konfiguraciju neuronske mreže ali još važnije odrediti način na koji se podaci uzimaju iz igre. Potrebno je definirati koliko informacija i koliko često uzeti.

Budući je igra interaktivna te zahtijeva resurse operacijskog sustava koje programski jezik "Java" preuzima u posebnoj dretvi (EDT - Event Dispatch Thread), korištenje stvarne igre za učenje bilo bi kontraintuitivno jer smo prethodno rekli da je cilj riješiti se vremenske i memorijske složenosti, a upravo bi te probleme imali učenjem u stvarnoj okolini igre. Stoga ćemo učenje provoditi unutar vjerodostojne simulacije koja će u sebi sadržavati ključne informacije stvarne igre.

U simulaciji ćemo oponašati iteracije koje se događaju "automatski" prilikom interaktivnog igranja, uzimat ćemo unos korisnika odnosno rezultat koji će nam neuronska mreža dati, ručno ažurirati stanje igre pomicanjem igrača i neprijatelje itd. Tako ćemo efektivno trenirati neuronsku mrežu na igri ali u kontroliranim uvjetima i puno, puno brže.

Iz simulacije ćemo za potrebe učenja preuzeti poziciju igrača, poziciju novčića te pozicije određenog broja neprijatelja te udaljenosti od njih do igrača. To će biti naš ulaz u neuronsku mrežu.

2.2.4. Arhitektura neuronske mreže

Koristili smo neuronsku mrežu s 2 skrivena sloja, svaki po 5 neurona. Ulaz nam čini :

- 2 elementa od igrača: **x** i **y** koordinata pozicije
- 2 elementa od novčića: **x** i **y** koordinata pozicije
- 3 elementa of svakog neprijatelja kojeg smo uzeli u obzir: **x** i **y** koordinata pozicije te udaljenost od igrača

Ukupno imamo (*veličina ulaza* = 2 + 2 + 3 * *broj neprijatelja*) elemenata koje koristimo kao ulaz.

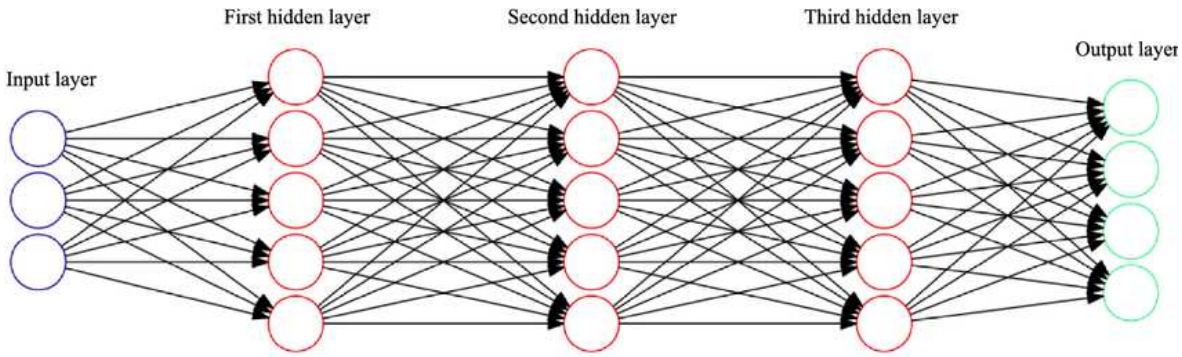
Način na koji pohranjujemo ulaz, težine i izlaz jest unutar zasebnih matrica. Pohrana podataka unutar matrica je standardna metoda manipulacije podacima prilikom strojnog učenja umjetne neuronske mreže jer računala posjeduju grafičke kartice koje odlično obavljaju operacije s matricama posebno množenje matrica.

- Ulaz se pohranjuje u matricu dimenzija (*veličina ulaza*), 1) tzv. stupčani vektor.
- Težine se spremaju u matrice na idući način:
  - Svaki sloj će imati svoju matricu dimenzija (*broj neurona u trenutnom sloju*, *broj neurona u prošlom sloju*), s tim da je u prvom sloju broj neurona prošlog sloja zapravo veličina ulaza.
  - Osim toga pohranjujemo i dodatne težine zvane pristranost koju svaki neuron ima za sebe neovisno o vezama s drugim neuronima. Pristranost pohranjujemo u matricu dimenzija (*broj neurona trenutnog sloja*, 1)
- Izlaz se pohranjuje u matricu dimenzija (*broj neurona u krajnjem skrivenom sloju*, 1)

Primijetimo da pri tome moramo paziti o kompatibilnosti dimenzija prilikom množenja matrica, no neuronska mreža je model koji je tako ostvaren da će matrice uvijek biti kompatibilne ako sve točno radimo.

Dobivanje rezultata neuronskom mrežom naziva se unaprijedni prolaz kroz neuronsku mrežu. Unaprijedni prolaz ostvarujemo sljedećim redoslijedom:

1. Prolazimo sloj po sloj
2. U trenutnom sloju množimo matricu težina s matricom ulaza te dodajemo matricu pristranosti
3. Dobiveni rezultat prolazi kroz aktivacijsku funkciju
4. Rezultat koji smo dobili u prošlom koraku koristimo kao ulaz za sljedeći sloj
5. Ponavljamo korake 2. - 4. do zadnjeg sloja
6. Nakon zadnjeg sloja dobiveni rezultat je izlaz naše neuronske mreže



Slika 2.9. Primjer unaprijedne potpuno povezane neuronske mreže

2.3. Genetski Evolucijski Algoritam

2.3.1. Osnove genetskog algoritma

Genetski algoritmi inspirirani su prirodnim evolucijskim procesima. Oni koriste populaciju mogućih rješenja koja evoluira tijekom vremena putem selekcije, križanja (crossover) i mutacije. Svako rješenje (individua) ocjenjuje se na temelju funkcije prilagodbe (fitness function) koja mjeri koliko je rješenje dobro. [4]

Osnovni koncepti genetskog evolucijskog algoritma su: populacija, jedinka, dobrota (fitness), selekcija, križanje, mutacija i zamjena.

Populacija

Genetski algoritam započinje s inicijalnom populacijom jedinki. Svaka jedinka predstavlja moguće rješenje problema koje se optimizira.

Reprezentacija jedinki

Svaka jedinka u populaciji je predstavljena kao niz genetskih varijabli koje se nazivaju geni. Ova reprezentacija može biti binarna, cjelobrojna ili čak kombinacija različitih tipova.

Funkcija dobrote (fitness funkcija)

Svaka jedinka u populaciji evaluira se koristeći funkciju cilja koja ocjenjuje kvalitetu rješenja. Cilj je maksimizirati ili minimizirati ovu funkciju kako bi se pronašlo optimalno rješenje.

Selekcija

Odabir jedinki za reprodukciju temelji se na njihovoj prikladnosti (fitness). Jedinke s većom prikladnošću imaju veću vjerojatnost odabira za reprodukciju.

Križanje (crossover)

Ovaj genetski operator simulira reprodukciju. Križanjem dvije roditeljske jedinke stvaraju se potomci (djeca) koji nasljeđuju genetske karakteristike svojih roditelja.

Mutacija

Mutacija je genetski operator koji slučajno mijenja jedan ili više gena jedinke. Ovaj korak omogućava uvođenje novih genetskih varijacija koje nisu bile prisutne u roditeljima.

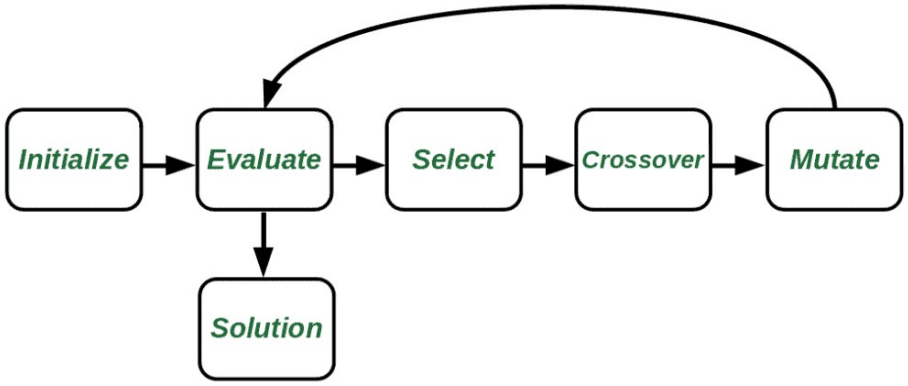
Zamjena (replacement)

Nakon generiranja potomaka, oni zamjenjuju postojeće jedinke u populaciji prema određenom kriteriju zamjene (npr. elitizam ili generacijska zamjena).

Proces evolucije

1. **Inicijalizacija populacije:** - Generira se početna populacija slučajnih strategija.
2. **Evaluacija:** - Svaka strategija se testira u igri i ocjenjuje prema funkciji prilagodbe.
3. **Selekcija:** - Najbolje strategije biraju se za reprodukciju.
4. **Križanje i mutacija:** - Odabrane strategije križaju se kako bi se stvorila nova generacija strategija, dok se manji dio strategija mutira kako bi se uvele varijacije.
5. Proces selekcije, križanja i mutacije ponavlja se kroz više generacija dok se ne postigne zadovoljavajuća razina performansi igrača.

Kombinacija neuronskih mreža i genetskih algoritama omogućava razvijanje sofisticiranog inteligentnog igrača sposoban za savladavanje raznih izazova. Neuronske mreže omogućuju adaptivno i učenje iz iskustva, dok genetski algoritmi pružaju moćan mehanizam za istraživanje i optimizaciju strategija kroz evoluciju.. [5]



Slika 2.10. Genetski Evolucijski Algoritam

2.3.2. Primjena u igri

Genetski evolucijski algoritam započinje kreacijom početne populacije od 10 jedinki. Svaka jedinka predstavlja jednog umjetnog inteligentnog igrača odnosno jednu neuronsku mrežu kojoj su početno sve težine nasumičnim odabirom postavljene na decimalnu vrijednost iz uniformne distribucije slučajnih brojeva.

Započinjemo iteraciju po svakoj jedinci iz populacije koju evaluiramo na 10 različitih simulacija igre. Nakon svake simulacije dohvaćamo rezultat koji je neuronska mreža ostvarila te izračunamo funkciju dobrote koja uzima u obzir vrijeme preživljavanja, ostvareni rezultat te ukupan broj pokupljenih novčića. Nakon izvršenih 10 simulacija uzimamo prosjek svih dobrota te je to krajnja dobrota koju dalje koristimo.

Nakon što smo evaluirali svaku jedinku i dobili dobrote za svaku jedinku, najbolju jedinku prenosimo u iduću generaciju populacije. Time ostvarujemo elitizam.
Zatim provodimo proces selekcije, križanja i mutacije jediniki. Selekciju vršimo tzv. proporcionalnom selekcijom (*proportional selection wheel*). Jedinke s većom dobrotom imaju veću vjerojatnost biti izabrane.
Selektiramo 2 jedinke iz trenutne populacije (roditelji) nad kojima provodimo križanje tako što za svaku vrijednost djetetovih težina u 50% slučajeva uzimamo vrijednost prvog roditelja a u drugih 50% uzimamo vrijednos drugog roditelja.

Jednom kad smo dobili novu jedinku (dijete) križajući 2 roditelja, nad djetetom provodimo mutaciju. Mutacija se izvodi tako da s vjerojatnošću od 5% svaku vrijednost težina inkrementiramo za slučajnu vrijednost iz Gaussove distribucije slučajnih brojeva.
Novodobivena jedinka se dodaje u novu generaciju populacije. Stvaranje novih jedinik procesom selekcije, križanja i mutacije ponavljamo sve dok nova populacija ne dosegne veličinu prethodne.
Korake iteracije ponavljamo do željene dobrote ili do maksimalne omogućene iteracije nakon čega u memoriju pohranjujemo najbolju jedinku koju kasnije možemo evaluirati na stvarnoj interaktivnoj igri.

3. Rezultati i rasprava

3.1. Rezultati

3.1.1. Performanse neuronskih mreža

Nakon treniranja neuronske mreže korištenjem genetskog evolucijskog algoritma učenja, analizirali smo performanse mreže u igri "Cubius".

Prosječno vrijeme preživljanja po generaciji populacije povećavalo se vrlo sporo, ali s obzirom na kompleksnost i nasumični karakter igre rezultati su zadovoljavajući i ukazuju na uspješnost upotrebe neuronske mreže u kombinaciji s genetskim evolucijskim algoritmom.

Prosječno vrijeme preživljavanja jedinke iznosi 30 sekundi što se dužim treniranjem može povećati ali bez kompleksnijih modela dovesti će do prenaučenosti, samim time do neupotrebljivog modela.

Genetski evolucijski algoritam porast funkcije dobrote smanjuje u prosjeku nakon 5000 iteracija, iako je i do tad porast bio spor ali opet znatno brži nego nakon navedenog broja iteracija.

3.2. Rasprava

3.2.1. Usporedba neuronskih mreža i genetskog algoritma

Neuronske mreže pokazale su se učinkovitijima u brzom učenju i adaptaciji na izazove u igri "Cubius". Njihova sposobnost kontinuiranog učenja iz iskustva omogućila je brže postizanje visokih performansi.
S druge strane, genetski algoritmi, iako nešto sporiji u postizanju optimalnih rješenja, pokazali su se robusnima u istraživanju velikog prostora mogućnosti i optimizaciji strategija.

3.2.2. Prednosti kombiniranog pristupa

Kombinacija neuronskih mreža i genetskog algoritma pokazala se kao uspješna metoda pri učenju računalnog inteligentnog igrača. Neuronske mreže omogućile su brzo učenje i prilagodbu, dok su genetski algoritmi dodatno poboljšali strategije kroz evoluciju. Ovaj sinergijski efekt rezultirao je značajno boljim performansama u odnosu na pojedinačne pristupe.

3.2.3. Utjecaj različitih faktora na performanse

Detaljna analiza pokazala je da složenost situacije, brzina i količina neprijateljski objekata značajno utječu na performanse inteligentnog igrača. Lakše situacije s manje prepreka rezultirali su većom uspješnošću, dok su kompleksnije situacije zahtijevali više iteracija za optimizaciju.

3.2.4. Ograničenja i izazovi

Glavna ograničenja uključivala su računalne resurse potrebne za treniranje neuronskih mreža i evoluciju genetskih algoritama. Također, kompleksnost nivoa igre ponekad je rezultirala lokalnim minimumima u procesu učenja, što je zahtijevalo dodatne metode za prevladavanje tih izazova. Veliki broj pokušaja i pogrešaka pri prilagodbi neuronske mreže su ključni za pronalazak optimalnog rješenja.

Kako konfigurirati neuronsku mrežu, koliko neprijatelja uzeti u obzir, koliko veliku populaciju, kroz koliko iteracija provoditi treniranje, kako izbjeći prenaučenost, kako pronaći bijeg iz lokalnog minimuma ili maksimuma su problemi koji su se izuzetno očitovali tijekom izrade ovog rada. Razne kombinacije parametara dovodile su do raznih rezultata te je jako važno dobro oblikovati model prije početka treniranja jer svaka sitnica može imati velike posljedice.

3.2.5. Potencijal za buduća istraživanja

Ovaj rad otvara nove smjerove za buduća istraživanja. Mogućnosti uključuju primjenu naprednijih tehnika poput dubokog učenja, konvolucijske i rekurentne neuronske mreže, te istraživanje hibridnih modela koji kombiniraju više algoritamskih pristupa. Dodatno, optimizacija računalnih resursa i ubrzanje procesa treniranja ostaju ključni izazovi za daljnji napredak.

4. Zaključak

Razvoj inteligentnog računalnog igrača za igru "Cubius" predstavljao je izazovan i složen zadatak, koji je zahtijevao primjenu naprednih tehnika iz područja umjetne inteligencije, uključujući neuronske mreže i genetske algoritme. Kroz ovaj rad istražili smo i implementirali dva ključna pristupa za stvaranje autonomnog igrača sposobnog za navigaciju kroz teške i izazovne situacije igre.

Primjenom neuronskih mreža, omogućili smo agentu da uči iz iskustva i prilagođava svoje akcije temeljem informacija iz okoline. Proces genetskog algoritma omogućio je modelu da iterativno poboljšava svoje performanse, smanjujući broj pogrešaka i povećavajući uspješnost u postizanju ciljeva. Neuronske mreže su se pokazale kao snažan alat za rješavanje problema visoke složenosti i dinamičkih promjena unutar igre.

Genetski algoritam omogućio je nam istraživanje i optimizaciju različitih strategija kroz proces evolucije. Kroz selekciju, križanje i mutaciju, evoluirali smo populaciju strategija, postupno poboljšavajući performanse igrača. Ovaj pristup pružio je robustan mehanizam za pronalaženje optimalnih rješenja u situacijama gdje su klasični algoritmi suočeni s izazovima pretraživanja velikog prostora mogućnosti.

Kombinacija neuronske mreže i genetskog evolucijskog algoritma omogućila nam je razviti inteligentnog igrača koji je sposoban donositi autonomne odluke u realnom vremenu, prilagođavati se kompleksnim situacijama i postići visoku razinu uspješnosti u igri "Cubius". Naši rezultati pokazuju da integracija neuronskih mreža i genetskih algoritama može značajno unaprijediti sposobnosti umjetnih agenata, čineći ih efikasnijima u rješavanju zahtjevnih problema.

Zaključno, ovaj rad doprinosi boljem razumijevanju i primjeni umjetne inteligencije u području razvoja računalnih igrača. Naša istraživanja i implementacije pružaju temelj za daljnje unapređenje metoda učenja i optimizacije, te otvaraju nove mogućnosti za primjenu ovih tehnika u širem spektru igara i drugih kompleksnih sustava. Kroz buduće radove, dodatna istraživanja mogu se usmjeriti na poboljšanje efikasnosti treniranja, smanjenje računalnih resursa te primjenu novih algoritamskih rješenja koja će dodatno unaprijediti performanse inteligentnih igrača.

Literatura

[1] B. D. Basic i J. Snajder, "10. strojno ucenje", 2019/2020., powerPoint prezentacija, Uvod u Umjetnu Inteligenciju, Sveuciliste u Zagrebu, Fakultet elektrotehnike i racunarstva.

[2] S. Haykin, *Neural Networks and Learning Machines*. Pearson, 2009.

[3] B. D. Basic, J. Snajder, i M. Cupic, "Umjetne neuronske mreze", 2008., skripta, Umjetna inteligencija, Sveuciliste u Zagrebu, Fakultet elektrotehnike i racunarstva.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Playing atari with deep reinforcement learning", *arXiv preprint arXiv:1312.5602*, 2013.

Sažetak

Razvoj inteligentnog računalnog igrača

Mateo Kopačević

Razvoj inteligentnog računalnog igrača za igru "Cubius" predstavlja izazovan zadatak koji zahtijeva napredne tehnike umjetne inteligencije. U ovom radu istraženi su i implementirani neuronske mreže i genetski algoritmi kako bi se stvorio autonomni igrač sposoban za navigaciju kroz složene situacije igre. Neuronske mreže omogućile su igraču učenje iz iskustva i prilagodbu akcija temeljem povratnih informacija, dok su genetski algoritmi omogućili evoluciju i optimizaciju strategija. Kombinacija ovih pristupa rezultirala je inteligentnim igračem s visokom razinom uspješnosti. Ovaj rad doprinosi razumijevanju i primjeni umjetne inteligencije u razvoju računalnih igrača i otvara nove mogućnosti za daljnja istraživanja.

Ključne riječi: Umjetna inteligencija; Neuronske mreže; Evolucijski algoritam

Abstract

Intelligent Computer-based Game Player Development

Mateo Kopačević

The development of an intelligent computer player for "Cubius" is a challenging task requiring advanced artificial intelligence techniques. This paper explores and implements neural networks and genetic algorithms to create an autonomous player capable of navigating through complex game situations. Neural networks enabled the player to learn from experience and adapt actions based on feedback, while genetic algorithms allowed for the evolution and optimization of strategies. The combination of these approaches resulted in a highly successful intelligent player. This work contributes to the understanding and application of artificial intelligence in the development of computer players and opens new possibilities for further research.

Keywords: Artificial intelligence; Neural networks; Evolution algorithm