Matematički fakultet

Računarska inteligencija

NEAT algoritam za igranje igrice

NeuroEvolution of Augmenting Topologies

Filip Jovanović 98/2015
Stefan Kapunac 157/2015
Nenad Ajvaz 316/2015

Maj 2019.

Sažetak

U uvodu ćemo ukratko predstaviti NEAT algoritam, a u nastavku se nalazi opis problema koji ovaj projekat treba da reši. U poglavlju 3 će biti predstavljena struktura projekta kao i kratko objašnjenje implementacije, a u poglavlju 4 prikaz koliku ulogu podešavanje parametara ima u uspešnom radu neuronske mreže.

Sadržaj

1 Uvod

NEAT je tehnika treniranja neuronske mreže genetskim algoritmom. Razvili su ga Kenet Stenli i Risto Mikulainen[1] 2002. godine, na Univerzitetu u Teksasu, SAD [**?**]. Razlikuje se od ostalih neuronskih mreža po tome što, pored težina može da izmeni i topologiju same mreže. Na početku je topologija minimalna (ulazni čvorovi povezani sa izlaznim), ali će se mutacijom kreirati novi čvorovi, kao i nove grane između njih. Ovo je jedna od ključnih osobina NEAT algoritma, jer nalazi minimalnu neuronsku mrežu koja rešava problem, ali ako to problem zahteva, mreža može postati kompleksna.
Testovima je pokazano da NEAT nadmašuje i najbolje mreže fiksne topologije kod zahtevnih zadataka sa pojačanim učenjem (*eng. reinforcment learning*).

Ova tehnika, kao i većina neuronskih mreža, se može koristiti u raznim oblastima. Budući da se NEAT oslanja na neuro-evoluciju, njegovi tvorci smatraju da se algoritam veoma dobro snalazi u neprekidnom i visokodimenzionom prostoru stanja. Iz tog razloga, ovaj algoritam je primenjiv i na probleme učenja igranja video igara, što ovaj projekat obuhvata.
Pored zvanične dokumentacije za ovaj algoritam, koristili smo se i literaturom o veštačkoj inteligenciji u kreiranju video igara [**?**].

2 Opis problema

U prvom delu poglavlja će biti predstavljena sama igra koju mreža treba da savlada, dok ćemo u drugom delu poglavlja objasniti osnovne koncepte NEAT algoritma i njegovu primenu na igrici.

2.1 Video igra - MouseRun

Prvi korak projekta bio je osmisliti adekvatnu video igru, koja bi bila dovoljno interesantna za čoveka, ali i posmatranje kako računar pokušava da je savlada. Kako bismo mogli da lako menjamo bilo koji njen segment u cilju poboljšanja parametara za algoritam, igru smo napravili od nule. Kontroliše se miš čiji je cilj da pobegne od mačke koja ga juri, a to može da uradi samo ako održava energiju jedući sireve, pritom izbegavajući mišolovke i bare koje ga usporavaju. Prikaz igrice na slici 1.

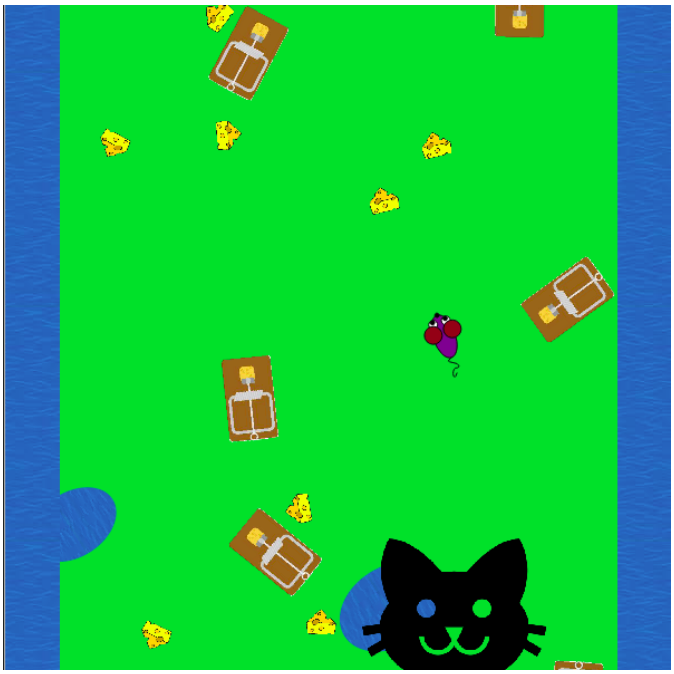
2.2 NEAT algoritam

Zbog složenije prirode algoritma, neophodan je malo širi uvod, pa će se u nastavku ukratko opisati neki njegovi aspekti, sa osvrtom na neuronske mreže fiksnih topologija i prednosti u odnosu na njih.

Neuro-evolucija

Kod tradicionalnih pristupa neuro-evolucije, topologija mreže se bira pre eksperimenta. Obično postoji jedan skriveni sloj čiji su čvorovi povezani sa svim ulazima i izlazima. Prvo se optimizuju težine grana - traže se mreže sa najboljim učinkom, a zatim se one reprodukuju kao čin evolucije, u cilju određivanja funkcionalnosti te mreže. Mana ovakvih fiksnih

[1] Kenneth O. Stanley, Risto Miikkulainen



Slika 1: MouseRun

topologija je što nisu samo težine grana bitne za funkcionalnost, već i struktura, koja je u ovom slučaju nepromenljiva. Takođe, zahteva i ručno određivanje optimalne topologije (broj skrivenih čvorova) za specifičan problem metodom pokušaja i greške.

Izazovi u implementaciji

Prednost NEAT algoritma je u tome što prvo minimizuje strukturu mreže kroz evoluciju, kako bi smanjio dimenzionalnost prostora pretrage težina grana, što dovodi do značajnog ubrzanja kod učenja. Ovo sa sobom nosi nekoliko izazova:

1. kako odabrati pogodnu genetsku reprezentaciju za ukrštanje
2. kako očuvati topologije kojima je potrebno nekoliko generacija za optimizaciju
3. kako minimizovati topologiju *kroz evoluciju* bez posebnih testiranja kompleksnosti

Rešenja problema su ukratko objašnjena u nastavku.

1. Odabir reprezentacije

Mreže se mogu predstaviti direktnim i indirektnim enkodiranjem. Direktno enkodiranje tačno navodi čvorove i grane u fenotipu[2], ono je jednostavnije i koriste ga većina mreža.
Indirektno kodiranje je kompaktnije - navode se samo pravila za konstruisanje fenotipa, čvorovi i grane se mogu izvesti iz tih pravila. NEAT koristi

[2] Fenotip je skup osobina organizma koje su nastale zajedničkim delovanjem genotipova i uslova sredine u kojoj se organizam razvija.

indirektno kodiranje, jer njihova fleksibilnost može da usmeri pretragu na nepredvidive načine.

2. Očuvanje specijacijom

Kada se u mrežu doda nova grana, veoma retko će odmah dovesti do poboljšanja performansi. Celokupan fitnes će opasti, jer težina nove grane nije optimizovana. Potrebno je da prođe nekoliko generacija da bi se ta grana poboljšala, ali i u međuvremenu sačuvati da mreža sa takvom nestabilnom strukturom ne bude odbačena.
Rešenje toga leži u specijaciji, a ona predstavlja grupisanje organizama u vrste koji imaju slične karakteristike. Kod algoritma NEAT, to se radi preko funkcije *explicit fitness sharing*. Ideja je da se populacija podeli na „vrste“ i spreči da neka od vrsti postane dominantna u populaciji, jer ima visok stepen prilagođenosti. To se radi tako što se vrednosti fitnesa normalizuju veličinom trenutne populacije i njihove celokupne sposobnosti.

3. Očuvanje minimalnosti

Kod određenih vrsta mreža, najlakši način da se očuva minimalnost je smanjivanje fitnesa kad mreža postane velika, ali teško je pogoditi parametar za takvo penalizovanje, jer se mora znati koliko je izračunavanje kompleksno. NEAT ne mora da ima nikakav eksplicitan mehanizam za to, jer uvek kreće od minimalne topologije. Tako je pretraga malih dimenzionalnosti, i raste po potrebi. Ovo se odnosi i na sve usputne međumreže.

3 Implementacija NEAT algoritma

U nastavku će ukratko biti opisani najbitniji delovi klasa programa, a zatim princip rada važnijih delova algoritma.

3.1 Struktura programa

- **Genome** - reprezentacija jedne instance neuronske mreže. Sadrži listu čvorova i grana mreže, broj ulaza, izlaza, slojeva i koeficijent fitnesa kao merilo prilagođenosti. Sadrži funkciju koja prolazi kroz mrežu i na osnovu njene uspešnosti daje različite komande miševima. Genom može i da se ukršta i mutira.
- **NodeGene** - predstavlja čvor mreže. Svaki čvor pamti identifikator, u kom se sloju nalazi, sa kim je povezan i izlaznu vrednost koju propagira kroz mrežu nakon aktivacije.
- **ConnectionGene** - predstavlja granu mreže. Čuva ulazni i izlazni čvor, težinu i inovativni broj[3].
- **Species** - predstavlja vrste unutar generacije. U njoj se nalaze slične instance klase *Genome*, u kojoj se čuva najbolja i prosečna prilagođenost, najbolja jedinka, kao i indikatori da li je vrsta zaostala. Postoje funkcije za utvrđivanje pripadnosti vrsti, kreiranja nove jedinke, ali i uništavanje vrste ako stagnira.
- **Game** - predstavlja instancu jedne partije igre. Pošto jedna partija sadrži nekoliko desetina miševa, pored svih njih pamti i identifikator najboljeg.

[3] Broj koji predstavlja jedinstvenu globalnu oznaku gena.

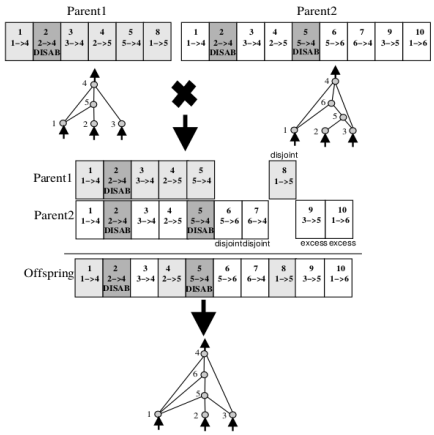
- **Controller** - glavni pokretač celog programa. Sadrži listu genoma kao populaciju, listu vrsti kojoj ti genomi pripadaju kao i broj generacija kroz koje mreža treba da prođe. U ovoj klasi se nalazi najbitnija funkcija *evolve*, koja primenjuje genetskog algoritma na mrežu.

3.2 Bitni koncepti

Mutacija kod NEAT algoritma ne menja samo težine grana, već i topologiju mreže - tako raste od minimalne ka optimalnoj. Topologija mreže može se izmeniti na dva načina:

1. **dodavanje grana** - nova grana se stvara i spaja dva do sad nepovezana čvora, a težina grane se određuje na slučajan način.
2. **dodavanje čvorova** - postojeća grana se deli i onemogućava, zatim se pravi novi čvor na njoj koji je povezan sa ista dva čvora kao i prethodna grana. Nova grana koja vodi u novi čvor će imati težinu 1, grana koja vodi iz novog čvora će zadržati prethodnu težinu.

Kao što je ranije rečeno, dodavanje novih grana ne mora da znači da će postojati negativan uticaj na mrežu, **specijacijom** se ona može očuvati kroz generacije. Ona funkcioniše tako što za jedinku proveravamo koliko ima sličnu strukturu sa ostalim jedinkama u vrsti, uparimo ih po njihovom inovativnom broju i merimo broj čvorova koji su višak (*eng. excess*) i razdvojeni (*eng. disjoint*). Višak su čvorovi koji se ne podudaraju na kraju, a razdvojeni oni koji se ne podudaraju u sredini. Na slici 2 se nalazi primer.



Slika 2: Poklapanje čvorova dve mreže

Nakon računanja takvih čvorova, po sledećoj formuli se određuje da li će jedinka pripasti vrsti:

$$\delta = \frac{c1 \cdot E}{N} + \frac{c2 \cdot D}{N} + c3 \cdot \overline{W},$$

gde je δ rastojanje podudarnosti, E broj višak čvorova, D broj razdvojenih čvorova, $\overline{W}$ prosečno rastojanje težina kod čvorova koji se podudaraju, a N broj grana u većoj jedinci. Koeficijenti $c1, c2, c3$ su parametri koji

balansiraju važnost ova tri faktora. Ako je δ veće od unapred postavljenog praga δ_t, jedinka pripada vrsti.
Ranije smo naveli funkciju *explicit fitness sharing* i njenu ulogu, a sada navodimo kako se ta nova prilagođenost računa za svaki organizam i:

$$f'_i = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))},$$

gde je $sh(\delta(i,j)) = 1$ ako je $\delta > \delta_t$, inače je 0.

Kada je u pitanju **ukrštanje** genoma, sličan je postupak kao kod merenja pripadnosti neke jedinke vrsti. Za dve jedinke koje želimo da ukrstimo se prvo poređaju grane sa istim inovativnim brojem, i one se nasumično prosleđuju u dete, dok se geni koji se ne poklapaju nasleđuju od prilagođenijeg roditelja.

4 Podešavanje parametara i rezultati

Veći problem od same implementacije algoritma bio je izbor odgovarajućih parametara - veličina populacije, funkcija cilja kao i ulazne vrednosti za neuronsku mrežu.

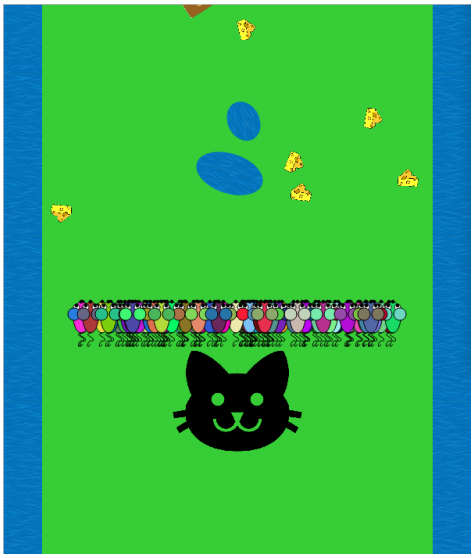
- **Veličina populacije** - Zbog hardverskih ograničenja, velike populacije su pravile dva problema:
  1. Ocenjivanje je bilo sporo - ovaj problem smo rešili podelom populacije na manje grupe od 100 jedinki, pa bi se ocenjivanje vršilo po grupama umesto na celoj populaciji odjednom.
  2. Stvaranje sledeće generacije je bilo sporo - ovaj problem smo mogli da izbegnemo jedino ograničavanjem veličine populacije na 1000-2000.
- **Funkcija cilja** - Početna ideja je bila da ocena za igrača bude vreme koje je uspeo da ostane u životu. Međutim, nastao je problem u tome što se u prvim generacijama najčešće više isplatilo da samo idu napred. Posle mnogo eksperimentisanja, odlučili smo da uvedemo bonuse za svako kretanje, skretanje, izbgavanje prepreka i konzumiranje sira, kao i penale za prolaske kroz bare, stajanje u mestu i kretanje isključivo u jednom smeru.
- **Ulazne vrednosti** - Prva zamisao je bila da svaki igrač može da vidi svoju poziciju, rotaciju, kao i poziciju svih ostalih predmeta na sceni. Glavna mana ovog pristupa je bila to što za veliki broj ulaznih vrednosti, evolucija predugo traje. Da bismo redukovali broj ulaznih vrednosti, dali smo igračima tri senzora - jedan ispred njega i po jedan sa svake strane. Senzor govori igraču da li, i koji objekat se nalazi u njegovoj blizini, kao i tačne koordinate objekta. Na ovaj način, broj ulaznih vrednosti za svaki od senzora je 3, što sa pozicijom i rotacijom igrača čini 12 ulaznih vrednosti. Ovo se pokazalo kao optimalno.

Program je napravljen i testiran na Linux Ubuntu sistemu, u jeziku C++17 uz Qt biblioteke, korišćenjem alata QtCreator. Mašina na kojoj je najduže trajala evolucija, i postigla najbolje rezultate, je desktop računar sa četvorojezgarnim procesorom AMD A8, 2.9GHz radnog takta i 4MB L2 keša.

Sa ovakvom konfiguracijom, nakon celog dana evolucije i nekoliko stotina generacija, miševi su počeli da uspešno izbegavaju neke prepreke, a neki su uspeli i da opstanu iznenađujuće dugo. Neuronske mreže najboljih jedinki su u tom trenutku imale 20-30 neurona raspoređenih u 3-5 slojeva i 20-30 sinapsi.

Zbog vremenskih i hardverskih ograničenja, naš cilj da istreniramo savršenog igrača nije ostvaren, ali evolucija i napredak se jasno mogu videti. Čovek je i dalje verovatno sposoban da postigne bolji rezultat, ali imajući u vidu da je broj neurona i sinapsi čoveka daleko veći od 30, to nije iznenađujuće.

Na slikama ispod se mogu videti stanja igre u početnim fazama.



Slika 3: Generacija 0, serija 1.

Slika 4: Generacija 1, serija 1.

Slika 5: Generacija 1, serija 5.

Slika 6: Generacija 2, serija 3.

5 Zaključak

NEAT je jedan veoma koristan, ali i dosta kompleksan algoritam. Rezultati koje on daje su vidno bolji od mnogih drugih algoritama sa neuronskim mrežama. Smatramo da je mreža prilično dobro savladala igricu, budući da su najuspešniji pojedinci uspeli da nadmaše najbolji rezultat svakog od autora ovog projekta, iako nismo imali hardver ogromnih performansi za učenje mreže. Danas je ova oblast veoma popularna, i sigurno je da će oblast neuro-evolucije nastaviti da se koristi u problemima raznih sfera.

Literatura

[1] R. M. Kenneth O. Stanley, "Evolving neural networks through augmenting topologies," *Commun. ACM*, vol. 10, no. 2, 2002. Online at: http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf.

[2] M. Buckland, *AI Techniques for Game Programming*. 2645 Erie Avenue, Suite 41, Cincinnati, Ohio 45208: Premier Press, 2002.