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

Softver igra sve veću ulogu u modernom svetu i stoga je sve važnije da on bude ispravan. Navešćemo primere katastrofalnih softverskih grešaka, a zatim ćemo prikazati različite metode za izbegavanje i uklanjanje takvih grešaka u različitim fazama razvoja. Posvetićemo pažnju i mogućnostima unapređenja procesa razvoja softvera koje nam donose nove tehnologije.

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 [1]. 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.

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.

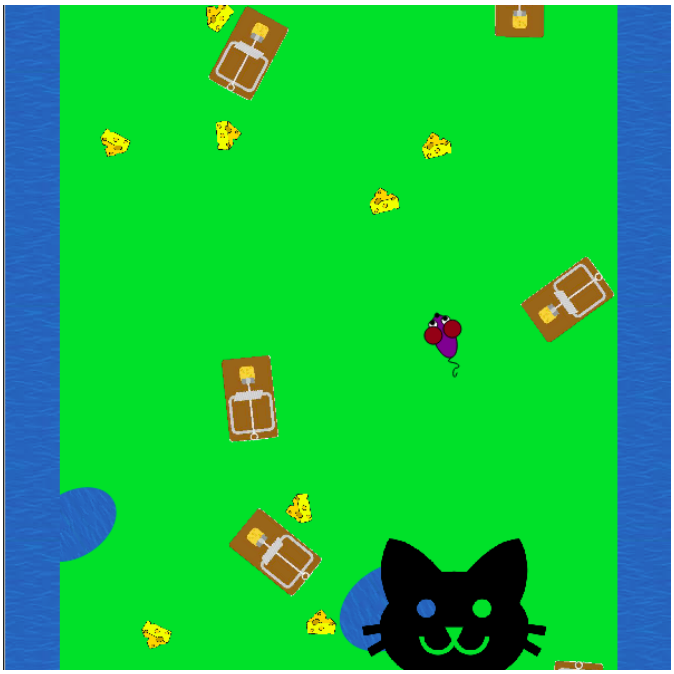
2.2 NEAT algoritam

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 topologija je što nisu samo težine grana bitne za funkcionalnost, već i

[1] Kenneth O. Stanley, Risto Miikkulainen



Slika 1: MouseRun

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 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. Možda ulepšati ovo malo

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

(Možda treba neki bolji uvod u ovo)

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, ali o tome više reči kasnije.
- **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, inovativni broj (TODO: objasniti ovo), kao i funkciju koja mutira težinu grane.

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

- **Species** - predstavlja vrstu 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.
- **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

...

4 Podešavanje parametara i rezultati

TODO

5 Zaključak

TODO

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.