SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Seminar

NEUROEVOLUCIJA POVEĆAVAJUĆIH TOPOLOGIJA

Josip Kelava

Zagreb, svibanj 2022.

Sadržaj

Uvod

S napretkom tehnologija suočavamo se sa sve težim i kompleksnijim problemima te ih rješavamo kompleksnijim algoritmima. Dosta često, međutim, problemi budu gotovo nerješivi nekim logičkim algoritmom; na primjer, razlikovati sliku mačke ili psa, ili što bi autonomno vozilo trebalo raditi u nekom trenu. Na takve probleme primjenjujemo umjetnu inteligenciju – što je skupni naziv za niz deterministčkih algoritama i heuristika. U ovom seminaru ćemo razmatrati neuronske mreže, treniranje strukture i parametara mreže genetskim algoritmom.

Neuroevolucija

Neuronske mreže iznimno su korisne jer mogu rješavati iznimno korisne jer uz relativno jednostavan algoritam niza matematičkih funkcija mogu riješiti iznimno kompleksne probleme, kao što je prepoznavanje objekata na slici[1], igranje igara[2], autonomna vožnja[3] i mnoge druge. Tradicionalno se koristi fiksna struktura mreže, odabrana sukladno problemu koji rješava, te se kasnije nekim algoritmom (propagacija diferencijala greške unazad, genetski algoritam) modificiraju parametri dok se ne dobije mreža koja dobro rješava zadatak.

Veličina korištene neuronske mreže ovisi u zadatku koji želimo riješiti. Veće mreže mogu definirati kompleksnije funkcije i riješiti kompleksnije probleme, međutim to dolazi uz cijenu performansi. Prvo, sama mreža je veća, pa je njeno korištenje nešto sporije, što može biti bitno u sustavima koji sustavima koji rješavaju problem u realnom vremenu, ili moraju brzo odraditi veliki broj iteracija. Nadalje, točnost izlaza neuronske mreže konvergirati će sporije, što znači da će biti potreban veći broj iteracija. Konačno, da bi se izbjegla prenaučenost, za veću mrežu će biti potreban veći broj primjera ulaza i izlaza – što je potrebno nabaviti, i dodatno će usporiti izvedbu jedne iteracije.

Kvalitetu strukture mreže teško je procijeniti prije nego što pri treniranju parametara funkcija greške konvergira / počne se primjećivati prenaučenost – povratna informacija se dobiva sporo, a nakon većine promjena se proces treniranja resetira.

Ideja NEAT-a

Neuroevolucija povećavajućih topologija (NEAT) eliminira problem „pogađanja“ strukture mreže – umjesto fiksne strukture mreže, krećemo s praznom mrežom, gdje će struktura biti izgrađena postepeno kroz genetski algoritam. Uklanjanjem fiksne strukture prostor pretrage rješenja znatno je proširen povećavajući mogućnost pronalaska dobrog rješenja. Korištenje genetskog algoritma donosi niz prednosti: genotipovi teže tome da prate gradijentni skup, ali zbog nasumičnosti mutaciju su značajno manje ograničeni; većina implementacija genetskog algoritma sadrži neki mehanizam koji nagrađuje raznolikost gena, čime se postiže da algoritam prođe kroz puno veći dio prostora pretrage rješenja. Postoji i više načina za rješavanje problema lokalnog

optimuma: u nedostatku povoljnih mutacija moguća je pojava isprva štetne mutacije genotipa koja će u nekom trenu prijeći na gradijent do drugog optimuma, gradijentni spust se događa u široj fronti umjesto u jednoj točki pa je ovisno o obliku funkcije dobrote moguće da će dio populacije zaobići lokalni optimum i nastaviti dalje. Križanjem (kombiniranjem genotipova), koje podržava paralelni pronalazak različitih dijelova rješenja, dobiva se bolje rješenje i moguć je značajni pomak na plohi funkcije greške, moguće prelazeći prema boljem optimumu.

Dodatna fleksibilnosti strukture mreže jest da će konačna mreža generalno biti manja nego mreža unaprijed određene strukture. Kad NEAT generira korisnu neuronsku vezu, genetski algoritam će ju proširiti po većini, ili dijelu populacije. S druge strane, da bismo osigurali da sve potrebne veze postoje u unaprijed određenoj strukturi, vjerojatno će biti uključen i veliki broj nepotrebnih veza.

Implementacija NEAT-a

NEAT je u srži genetski algoritam. Generiranje nasumičnih početnih topologija smanjuje šanse za dobiti minimalnu topologiju. Početna populacija se stoga sastoji od nepovezanih mreža, ili povezanih mreža bez skrivenih neurona. Kroz evoluciju dodaju se nove veze i skriveni neuroni te mijenjaju težine veza. Svaka nova veza zapisuje se u DNA jedinke, ali i u jednu dijeljenju DNA koja zapisuje sve gene. Time se umanjuje mogućnost paralelizacije, ali dobivamo mogućnost svaku vezu definirati jedinstvenim identitetom (bitno za križanje jedinki). U svakoj iteraciji, jedinke se ocjenjuju, dio ih se eliminira, te se zatim generira nova populacija. Nove jedinke nastaju križanjem dviju mreža iz prethodne generacije, praćene mutacijom nove mreže. Pošto je svakoj neuronskoj mreži dodijeljen jedinstveni broj koji ju identificira, moguće je lako odrediti koje se veze poklapaju u obje mreže, a koje ne. Novonastala mreža nasumično preuzima podudarajuće gene (težine veza prisutnih u obje križane mreže), te ne podudarajuće gene bolje ocijenjene jedinke.

Kako bi se spriječilo da jedna mreža (genotip) prevlada cijelom populacijom, te da se da dodatna prilika novim kompleksnijim mrežama da si podese težine, NEAT koristi podjelu jedinki na vrste te eksplicitno dijeljenje dobrota. Za svaki par jedinki možemo definirati njihovu udaljenost:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \bar{W}$$

N – broj gena veće jedinke (roditelj s više aktivnih veza)

E – broj gena većeg roditelja koji su van raspona gena manjeg roditelja

D –broj gena oba roditelja koji se ne poklapaju unutar raspona manjeg roditelja

W – prosječna razlika u vrijednostima poklapajućih gena (težina veza)

Dobrota svake jedinke dijeli se brojem jedinki koje su dovoljno slične razmatranoj jedinki – dvije jedinke smatramo dovoljno sličnima ako je njihova udaljenost manja od odabrane udaljenosti δ_i .

Ista provjera sličnosti koristi se za podjelu populacije u vrste – za svaku vrstu se definira predstavnik, te se nova jedinka redom uspoređuje s predstavnicima svih vrsta. Nova jedinka pridjeljuje se prvoj vrsti čiji joj je predstavnik dovoljno sličan. Ako nije pridijeljena nijednoj postojećoj vrsti, nova jedinka postaje predstavnikom nove vrste.

Ovakva podjela na vrste može se koristiti za više stvari:

- Može zamijeniti dijeljenje dobrote, gdje se sad dobrota dijeli s brojem jedinki u vrsti (brži izračun).
- Može se osigurati elitizam unutar vrste.
- Može se zabraniti razmnožavanje vrstama koje određeni broj generacija nisu napredovale; takve vrste vjerojatno nemaju mogućnost razvoja korisne strukture mreže, te su već razvile veći broj beskorisnih veza.

Dodavanje skrivenog neurona radi se tako da se veza težine w_i razlomi na dvije veze težina redom 1 i w_i. Ovakvom mutacijom se dobrota jedinke ne mijenja direktno zbog mutacije, što bi trebalo jedinki dati priliku da mutira i dodatnu vezu s tim novim čvorom.

Rezultati

U sklopu seminara implementiran je NEAT genetski algoritam u programskom jeziku C++ te je zatim testiran na dva problema.

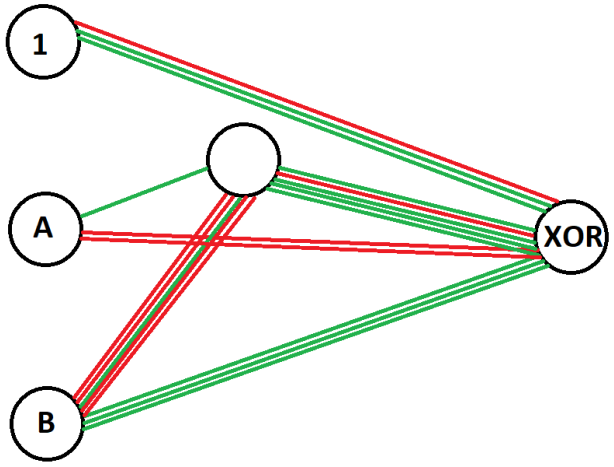
Prvi problem na kojem je NEAT testiran jest XOR – jednostavan problem s nelinearnim rješenjem. XOR problem zahtijeva nelinearnu aktivaciju neurona (korištena ReLU aktivacija. Također, potrebna je nešto kompliciranija struktura mreže: da bi se dobilo potpuno točno rješenje, mreža mora sadržavati neuron i skrivenom sloju, koji ne postoji u početnom genotipu. Zbog toga je XOR odličan test za NEAT, jer je problem s relativno jednostavnim rješenjem, ali istovremeno testira mogućnost algoritma da odredi točne težine veza u mreži i da razvije kompliciraniju mrežu nego onu s kojom je počeo. Kao funkcija greške korištena je kvadrat razlike očekivanog o dobivenog rješenja, a funkcija dobrote je definira kao:

$$fit = 4 - f_{err}(0,0) - f_{err}(0,1) - f_{err}(1,0) - f_{err}(1,1)$$

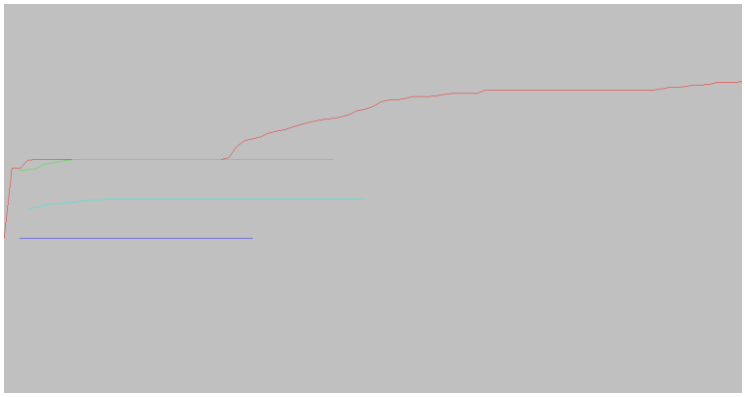
Uz sljedeće postavke (slika 1.) vrlo brzo se pojavila mreža s točnom (dovoljnom, slika 2.) topologijom, a točne težine evoluirale su do 164-te generacije. Izvršavanje programa trajalo je manje od jedne sekunde.

```
Real MaxValue = (Real)2.0f;
float MutationRadius = 0.05f;
int32 MutationResilience = 4;
float NewNodeChance = 0.03f;
float NewEdgeChance = 0.1f;
float RemoveNodeChance = 0;
float RemoveEdgeChance = 0;

int32 SelectionSize = 50;
int32 GenerationSize = 300;

float ExcessGeneFactor = 1.0f;
float DisjointGeneFactor = 1.0f;
float WeightDiffFactor = 0.4f;
float SameSpecieDistance = 0.5f;
float NumberOfGenesGround = 1000.0f;
int32 GenerationsWithNoImprovementKill = 30;
```

Slika 1. NEAT parametri korišteni pri XOR treningu



Slika 2. Mreža iz 193. generacije s maksimalnom dobrotom (4). Zelene linije predstavljaju veze s pozitivnom težinom, dok crvene veze predstavljaju veze s negativnom težinom.



Slika 3. Dobrote najboljih jedinki živih vrsta (XOR)

Vrijedi primijetiti da, iako je dobivena mreža minimalna po broju čvorova i njihovim povezanostima, između povezanih čvorova postoji po od jedne veze. Glavni razlog toga je što je s korištenim parametrima dodavanje nove veze između već povezanih čvorova može ispraviti ukupnu težinu njihove veze. To je, međutim, moguće lagano ispraviti promjenom parametara algoritma ili proširenjem algoritma.

Osim XOR-a, NEAT je testiran na simulaciji igre gdje je postavljeno N „kolačića“ unutar kvadrata po kojem se jedinka može kretati, a cilj je skupiti što više kolačića u ograničenom vremenu. Kao funkcija dobrote korišten je broj skupljenih kolačića i, da bi se nagrađivale i manje evolucije, od funkcije dobrote oduzeta je udaljenost od najbližeg kolačića u trenutku završetka simulacije. Da bi se izbjegle negativne dobrote (bitno za dijeljenje dobrote), ta udaljenost se prije oduzimanja dijeli s 10, te se još dodaje 0.5 na dobrotu. Time je osigurano da jedinka s većim brojem kolačića ima veću dobrotu nego jedinka s manjim, ali nagrađena je i blizina sljedećem kolačiću te je izbjegnuta negativna dobrota. Svaki put kad je kolačić pokupljen, pojavi se novi na nekom drugom mjestu u kvadratu.

NEAT je uspio naučiti skupljati kolačiće dosta brzo, ali nije uspio razviti kompleksniju strukturu s većim brojem kolačića prisutnim odjednom – međutim s većim brojem kolačića rezultat je bio sličan – nije uspio razvoj kompleksnije topologije koja bi dinamički odlučivala koji kolačić pokupiti, nego bi se mreža uvijek fokusirala na isti.

Zaključak

NEAT se pokazao donekle uspješnim, brzo evoluirajući validna rješenja za oba testirana problema. Međutim, moguće je dobiti i bolje rješenje na drugom problemu. Uočio sam par prilika za poboljšanje algoritma:

- Mogućnost dodavanja većih nasumičnih topologija, s idejom da se može dio topologije potreban za bolje rješenje odjednom. Kako bi se izbjegle nepotrebne veze dodane na taj način, moguće je kažnjavati kompleksnost mreže, te dodati mogućnost brisanja veza.
- Kad se dodaje novi čvor, dodaju se i dvije nove veze, dok se razlomljena veza „briše“; točnije, razlomljena mreža ostaje zapisana kao postojeća, ali ugašena veza. To je bitno za križanje, jer ako imamo jedinku koja je razlomila neku vezu na dvoje i križamo ju s nekom koja nije, ne želimo sadržavati oba puta (direktni i preko skrivenog čvora), nego samo razlomljeni (preko skrivenog čvora). Međutim, ako bi se koristila dulja evolucija (vjerojatno zahtijeva neku mjeru protiv nepotrebnih veza), moglo bi se dodati brisanje veza koje su onemogućene u svim jedinkama, što bi ubrzalo izračun križanja i time evoluciju mreže.

Višestruke veze između dva čvora usporavaju brzinu izvođenja programa i štete mogućnosti mutaciji strukture povezanosti istih čvorova. Dodavanje veze između već povezanih čvorova trebalo bi zabraniti ili zamijeniti dodavanjem nove težine na težinu već postojeće mreže.

Reference

[1] Prepoznavanje objekata na slici, Google Lens: https://lens.google/

[2] Igranje igara, AlphaStar: https://www.deepmind.com/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii

[3] Autonomna vožnja: https://www.bug.hr/transport/autonomna-cestovna-vozila-robote-vozi-polako-20775

-

Literatura

Henry AI Labs, *Neuroevolution of Augmenting Topologies (NEAT)*, https://www.youtube.com/watch?v=b3D8jPmcw-g, 11.5.2022.

Kenneth O. Stanley and Risto Miikkulainen, *Evolving Neural Networks through Augmenting Topologies*, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.28.5457&rep=rep1&type=pdf, 11.5.2022.

Sažetak

NEAT koristi genetski algoritam za izgradnju topologije neuronske mreže te određivanje težina veza u izgrađenoj topologiji. Omogućuje smisleno križanje topologija pridjeljivanjem jedinstvenih identifikatora vezama. Počinjanjem od minimalne topologije mogu se dobiti rješenja s minimalnim brojem nepotrebnih veza.