SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 493

PRIMJENA EVOLUCIJSKIH ALGORITAMA ZA RAZVOJ ARHITEKTURE NEURONSKIH MREŽA

Dina Petrak

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 493

PRIMJENA EVOLUCIJSKIH ALGORITAMA ZA RAZVOJ ARHITEKTURE NEURONSKIH MREŽA

Dina Petrak

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

ZAVRŠNI ZADATAK br. 493

Pristupnica:	**Dina Petrak (0036523849)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Marko Đurasević

Zadatak: **Primjena evolucijskih algoritama za razvoj arhitekture neuronskih mreža**

Opis zadatka:

Proučiti umjetne neuronske metode i različite evolucijske algoritme. Proučiti primjenu evolucijskih algoritama na razvoj arhitektura neuronskih mreža. Razviti programski okvir koji omogućuje razvoj arhitekture neuronske mreže korištenjem odabranog evolucijskog algoritma. Rad algoritma ispitati na skupu regresijskih problema, ocijeniti rad predloženog postupka u usporedbi s fiksnim arhitekturama te predložiti moguća poboljšanja predložene metode. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Rok za predaju rada: 10. lipnja 2022.

Zahvaljujem mentoru doc.dr.sc Marku Đuraseviću na strpljenju, korisnim savjetima i pomoći pri izradi ovog završnog rada.

Hvala prijateljima koji su uvijek bili uz mene i bez kojih preddiplomski studij ne bi prošao tako zabavno.

Veliko hvala mami, tati i sestri Anji koji su me podržavali u svakom trenutku.

Posebno se zahvaljujem bratićima Davidu i Karlu na velikoj pomoći koju su mi pružili.

SADRŽAJ

1. Uvod

Podatak je činjenica predočena u formaliziranom obliku. Danas se zbog velikog napretka u tehnologiji, pojam podatka često povezuje s računalima. Računala su ta koja primaju, obrađuju te prenose iznimno velike količine podataka.

No, odmaknemo li se od apstraktne definicije podatka koju poistovjećujemo s nizovima bitova, možemo vidjeti da su podatci svuda oko nas. Oni se svakodnevno obrađuju u mozgovima živih bića. Mozak prima podatke iz okoline, obrađuje ih i koristi za izvršavanje različitih funkcija. On ima sposobnosti rješavanja kompleksnih zadataka, učenja, pamćenja i generalizacije.

Uzevši to u obzir, razumijemo želju za digitalnim modeliranjem i ostvarivanjem prirodne inteligencije. Umjetna neuronska mreža je struktura koja imitira ljudski mozak s ciljem ostvarivanja sposobnosti učenja.

Naravno, sposobnost učenja ovisi o različitim parametrima. Postepeno poboljšavanje arhitekture i promjene težina veza neurona u konačnici će generirati bolju neuronsku mrežu. Inspiraciju za rješenje problema postepene optimizacije neuronskih mreža ponovno pronalazimo u prirodnom svijetu, točnije u evoluciji.

Kroz milijune godina organizmi se razvijaju i poboljšavaju kao bi mogli preživjeti u promjenjivoj i potencijalno opasnoj okolini. Svaki organizam u nekoj je vrsti natjecanja s drugim organizmom. Najbolji organizmi opstat će te će se njihove karakteristike prenijeti njihovim potomcima. Pozitivne prilagodbe će se tako očuvati, dok će one štetne postepeno nestajati. Naravno, tokom takvog procesa mogu nastupiti nasumične promjene u organizmima koje se nazivaju mutacije. Ako one koriste organizmu, njegove šanse za preživljavanje su veće.

Evolucijsko računarstvo koristi elemente evolucije, poput selekcije najpogodnijih jedinki, reprodukcije, nasljeđivanja i mutacije. Ti se koncepti koriste za postepeno mijenjanje arhitekture i težina neuronskih mreža, kombiniranje karakteristika neuronskih mreža s visokom učinkovitosti te eliminiranje onih lošijih.

U nastavku rada detaljnije će se opisati osnovni principi genetskih algoritama, neuronskih mreža te neuroevolucije te će se dati uvid u njihovu implementaciju. Osim toga, prikazat će se njihova uporaba za rješavanje problema regresije.

2. Genetski algoritam

Evolucija je u suštini proces pretraživanja prostora stanja. U tom slučaju, stanje je skup određenih karakteristika organizma koje mu pomažu u preživljavanju. Evolucijski algoritam primjenjuje tehnike inspirirane evolucijom te pronalazi optimalno rješenje zadanog problema.

Utjecaj evolucije na sam evolucijski algoritam primjećuje se u procesu selekcije, operatorima reprodukcije ali i u reprezentaciji podataka u obliku kromosoma. Evolucijski algoritmi obuhvaćaju vrlo široko područje te se dijele na

- Genetske algoritme
- Evolucijsko programiranje
- Evolucijsku strategiju
- Diferencijalnu evoluciju

Genetski algoritmi modeliraju evoluciju prirodnog genetskog sustava. U tom slučaju, evolucijska promjena posljedica je isključivo varijacije u genima. Karakteristike jedinki, u tim su algoritmima, predstavljene skupovima gena. Različiti operatori inspirirani evolucijom primjenjuju se na jedinke.

Jedinka u populaciji predstavlja jedno moguće rješenje zadanog problema. Za svaku jedinku populacije računa se vrijednost koja predstavlja njezinu kvalitetu. Ona određuje koliku šansu taj član ima za reprodukciju, a time i prenošenje svog genetičkog materijala. Uzevši u obzir izračunatu vrijednost, vrši se proces selekcije a potom i reprodukcije. Time se omogućava da štetne karakteristike postepeno iščezavaju dok se one pozitivne prenose na novonastale jedinke. Naprednije novonastale jedinke postaju dio populacije. Taj proces se ponavlja sve dok nije ispunjen uvjet zaustavljanja.

2.1. Reprezentacija podataka

U prirodi, organizmi su određeni karakteristikama koje utječu na njihovu sposobnost preživljavanja. Te karakteristike pohranjene su u kromosomu organizma. Svaki kromosom sadrži velik broj gena koji direktno utječu na svojstva organizma a time i na šanse preživljavanje.

Takva prirodna struktura pokušava se primijeniti i u genetskom algoritmu. Tada, svaka jedinka predstavlja moguće rješenje zadanog problema a njena svojstva pohranjena su u genomu. Ta svojstva, odnosno geni, postepeno se optimiziraju kroz generacije.

Kvaliteta rješenja genetskog algoritma uvelike ovisi o načinu prikazivanja genoma. Naravno, genom može biti bilo kakva struktura podataka koja može sadržavati varijable. Većina genetskih algoritama koristi vektore određenih tipova podataka, poput binarnih vektora ili vektora s realnim brojevima. Takve strukture podataka omogućavaju efikasno pohranjivanje gena, a time utječu na povećanje učinkovitosti i smanjenje kompleksnosti samog algoritma.

2.2. Populacija

Genetski algoritmi su algoritmi pretraživanja temeljeni na populacijama. Svaka populacija zapravo je skup mogućih rješenja. Prvi korak u implementiranju genetskog algoritma je upravo stvaranje inicijalne populacije generiranjem nasumičnih vrijednosti koje će reprezentirati gene.

Vrlo je bitno da je generirana inicijalna populacije zaista dobra reprezentacija cijelog prostora pretraživanja. Ako populacija ne pokriva cijeli prostor pretraživanja postoji šansa da se potencijalna rješenja preskoče u procesu pretraživanja

Osim toga pozornost se treba staviti i na veličinu populacije. Populacije s velikim brojem jedinki sadrže veliku raznolikost, a time i bolju šansu za stvaranjem naprednijih jedinki. No, s većim brojem jedinki raste i kompleksnost provođenja operacija u algoritmu pa se iz tog razloga prevelike populacije izbjegavaju. S druge strane, manje populacije vjerojatno neće omogućiti pretraživanje cijelog prostora stanja što znači da će potencijalno optimalno rješenje biti preskočeno.

2.3. Funkcija dobrote

Kao što je već spomenuto, u teoriji evolucije, jedinke koje imaju bolje razvijene prilagodbe imaju veće šanse za preživljavanje. Jednako tako, genetski algoritmi boljim genomima daju veće šanse za križanje i dalje prenošenje vlastitih gena. Naravno, postavlja se pitanje kako odrediti koje su jedinke zapravo bolje od drugih. Tu ulogu, u genetskom algoritmu, preuzima funkcija dobrote, koja svakom genomu dodjeljuje brojčanu vrijednost koja opisuje efektivnost jedinke pri rješavanju zadanog zadatka.

Učinkovitost funkcije dobrote uvelike utječe na kvalitetu algoritma te se smatra jednom od najvećih poteškoća prilikom implementacije za određeni zadatak. Naime, funkcija dobrote treba što bolje odražavati problem koji se rješava. Vrlo je važno da se prilikom izračuna vrijednosti funkcije dobrote izbace i zanemare sve nepotrebne i suvišne informacije, tako da se težište stavi samo na one informacije koje zaista utječu na sposobnost jedinke da riješi zadani problem.

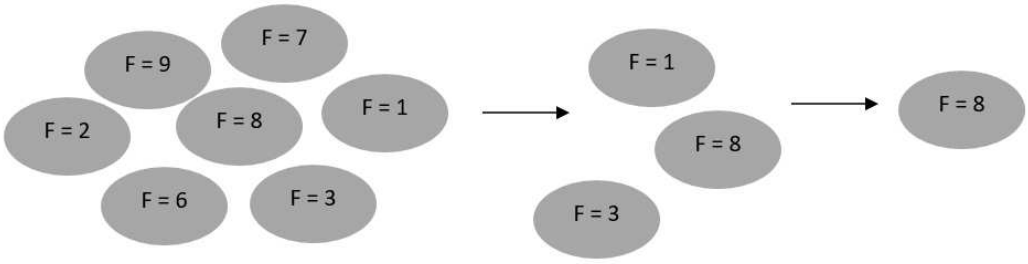
Vrijednost funkcije dobrote krucijalna je u procesu selekcije.

2.4. Selekcija

Selekcija je jedan od najbitnijih operatora genetskog algoritma. Izravna poveznica koncepta selekcije s evolucijom očituje se u aspektu preživljavanja najpogodnijih jedinki. Iz trenutne populacije se, uz korištenje vrijednosti funkcije dobrote, odabiru najbolji genomi koji će biti korišteni u procesu reprodukcije. Time će geni najpogodnijih jedinki biti preneseni u novu generaciju. Takvim postupkom, osigurava se da je svaka iduća generacija naprednija od prethodne.

Selekcija se može vršiti na više načina te je iz tog razvijen velik broj različitih algoritama selekcije. Neki od njih su nasumična selekcija, proporcionalna selekcija, selekcija rangiranjem te boltzmanova selekcija. Posebno zanimljiva je turnirska selekcija.

Turnirska selekcija nasumično odabire određen broj jedinki iz populacije te ih uspoređuje uzimajući u obzir vrijednost funkcije dobrote. Jedinka s najboljim iznosom dobrote smatra se pobjednikom te se koristi za reprodukciju. Bitno je napomenuti kako je ključ kvalitete turnirske selekcije dobro izabrana količina jedinki koja se natječe. Ako je taj broj premalen, turnirska selekcija prelazi u nasumičnu selekciju. S druge strane, ako je taj broj prevelik, pobjednik će uvijek biti najbolja jedinka što će vrlo brzo uzrokovati smanjenje raznolikosti u populaciji. Turniska selekcija vizualo je prikazana na slici 2.1.



Slika 2.1: Turnirska selekcija

2.5. Reprodukcija

Reprodukcija je proces u kojemu se izabrani genomi križaju i na taj način proizvode nove jedinke. U tom procesu može doći promjene nad genetskim materijalom novonastalih jedinki.

2.5.1. Križanje

U koraku selekcije izabrane su jedinke za koje se smatra da su vrlo efikasne prilikom rješavanja zadanog problema. Te jedinke nazivaju se roditeljima. Njihov genetski materijal prenijet će se na novonastalu jedinku, dijete, operatorom križanja. Uzevši u obzir da se roditelji smatraju dobrim jedinkama, njihova djeca trebala bi biti jednako dobra ili bolja. Na taj se način, kvaliteta jedinki kroz generacije poboljšava.

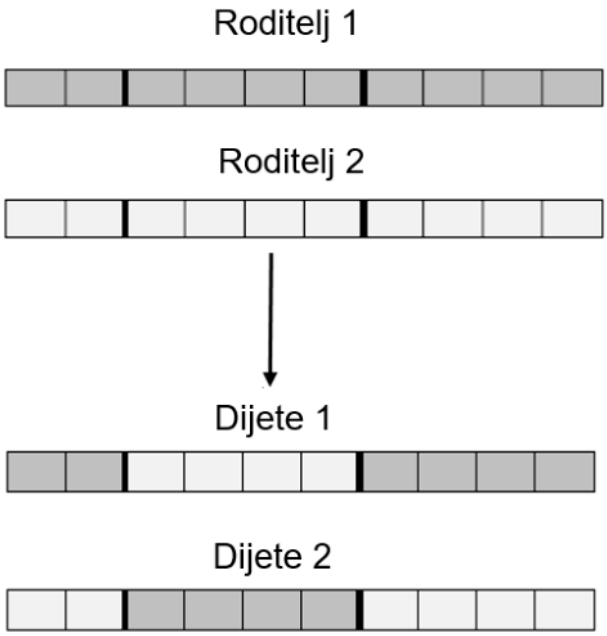
Kao što već spomenuto, jedinke su u genetskom algoritmu prikazane vektorom određenog tipa podataka. Iz tog razloga križanje je u suštini manipulacija vektora roditeljskih jedinki. Geni u roditeljskim vektorima se kombiniraju na različite načine te se tako stvaraju njihova djeca. Sama implementacija operatora križanja uvelike ovisi o korištenom prikazu genoma te ju je najlakše razmatrati za binarne vektore.

Kod binarnog prikaza, odabiru se jedna ili više točaka presjeka te bitovi između tih točaka mijenjaju. Na slici 2.2 prikazano je križanje s dvije točke presjeka.

2.5.2. Mutacija

U procesu reprodukcije može doći i do nasumične promjene gena djece. Cilj mutacije je uvesti novi genetski materijal u već postojeće jedinke. Na taj način povećava se raznolikost jedinki.

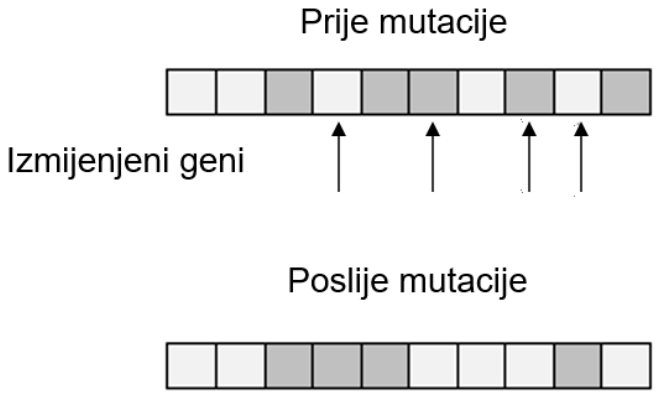
Jednako kao i križanje, operator mutacije temelji se na izmjeni vektora jedinki te ovisi o samom načinu prikaza. Svaki gen u vektoru ima određenu vjerojatnost da će



Slika 2.2: Križanje

biti izmijenjen. Vrlo je bitno da vjerojatnost takve promjene nije prevelika. Naime, u tom se slučaju riskira velika izmjena neke potencijalno dobre jedinke.

Za pojednostavljene operatora mutacije, ponovno se koristi prikaz binarnim vektorom. Geni koji se mutiraju mogu biti izabrani na različite načine. Na slici 2.3 prikazana je mutacija četiri nasumično odabrana gena u genomu.



Slika 2.3: Mutacija

3. Neuronske mreže

U današnjem svijetu, sustavi koji funkcioniraju uz pomoć umjetne inteligencije postaju sve značajniji. No, pokušaji da se modelira neka vrsta inteligencije postojali su i prije nastanka prvih računala. Od konstrukcije automata Turčina, preko ideja o postojanju robota, do prvih računala koja su uspješno izvodila zadatke do tada rezervirane samo za ljude, želja za konstruiranjem inteligentnog stroja prisutna je već stoljećima.

No, postavlja se pitanje što je to zapravo inteligencija. Definicija inteligencije još je uvijek predmet rasprave pa tako suglasnost oko definicije inteligencije i dalje ne postoji. Često se s pojmom inteligencije povezuju sposobnosti poput učenja, pamćenja, generaliziranja, planiranja, postupanja suvislo s danim informacijama te apstraktnog rasuđivanja. Pokušaji projektiranja tih karakteristika doveli su do razvoja novog područja koji se danas naziva umjetna inteligencija.

Modeliranje prirodne inteligencije omogućilo je nevjerojatne napretke u računarstvu. Takvi algoritmi i strukture stvorili su uvjete za rješavanje vrlo kompleksnih zadataka poput predviđanja trendova na temelju već prikupljenih podataka, dokazivanja teorema, raspoznavanja uzoraka, razumijevanje i obrade prirodnih jezika.

Ljuski mozak smatra se vrhuncem inteligencije poznate ljudima. Čak i današnja računala nemaju mogućnost izvođenja toliko kompleksnih zadataka. Razvoj biologije i drugih znanosti omogućio je uvid u samu strukturu mozga te time potaknuo istraživanje i modeliranje umjetnih neuronskih sustava, odnosno umjetnih neuronskih mreža.

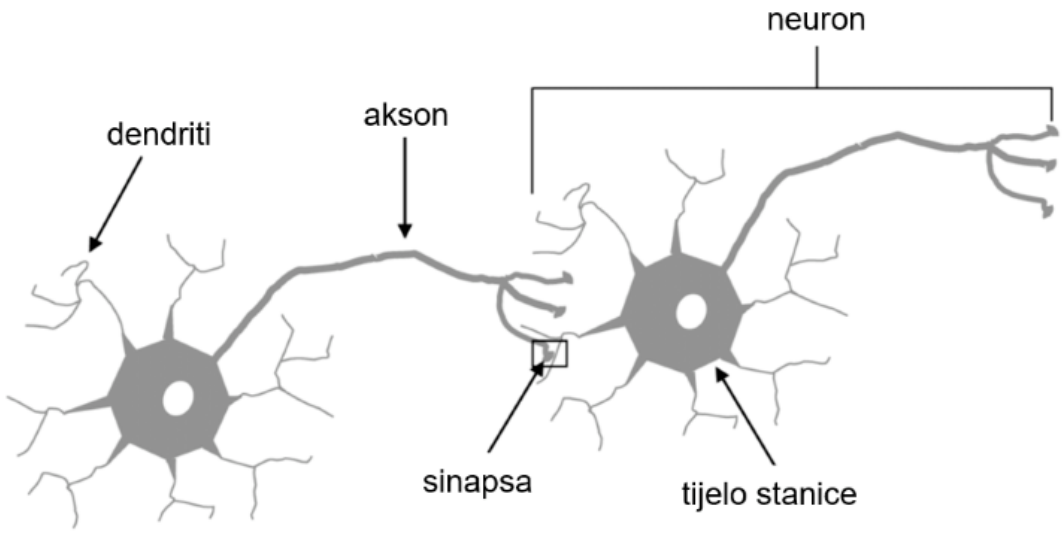
Ljudski mozak je kompleksno, nelinearno i paralelno računalo. Pretpostavlja se da ljudski mozak ima oko 86 milijardi neurona koji su zajedno povezani u informacijsku mrežu mozga. Biološki neuron je osnovna jedinica živčanog sustava te ima iznimno veliku važnost u građi i radu mozga. Iz tog razloga, građa i struktura biološkog neurona, inspiracija su za modeliranje umjetnog neurona koji je krucijalan element svih umjetnih neuronskih mreža.

3.1. Neuron

Biološki neuron je temeljna strukturno-funkcijska jedinica živčanog sustava. Njegova uloga je primanje, obrađivanje i odašiljanje podataka. Svaki neuron je, kao što je prikazano na slici 3.1, građeni od tijela stanice, dendrita i aksona.

Neuroni su jako dobro međusobno povezani u informacijsku mrežu mozga. Veza između neurona nastaje između završetka aksona jednog neurona i dendrita nekog drugog neurona. Ta veza naziva se sinapsom.

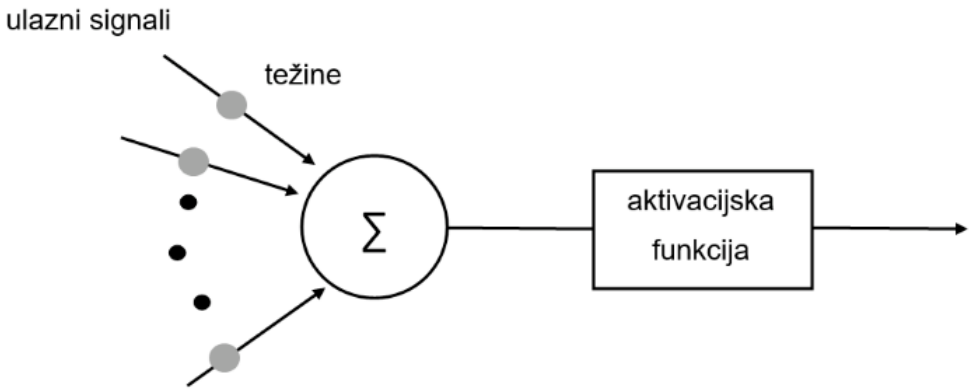
Informacije predstavljene električnim potencijalom putuju od dendrita, preko tijela stanice do aksona gdje se sinapsom prenose na dendrit drugog neurona. Dendriti pojačavaju ili prigušuju impulse koji se sumiraju u jezgri tijela. Signal putuje do aksona stanice jedino ako ukupni napon pređe određeni prag. Tada se kaže da neuron „pali“.



Slika 3.1: Biološki neuron[13]

Umjetni neuron modelira funkcionalnosti i građu biološkog neurona. Svaki umjetni neuron prima impulse iz ulaza ili od drugih neurona. Impulsi su opisani numeričkim vrijednostima te se pri ulasku u neuron množe s težinskim faktorima koji pojačavaju ili prigušuju impulse. Pomnoženi impulsi sumiraju se u neuronu analogno potencijalima u tijelu biološke stanice. Kada umjetni neuron „pali“ , on šalje svoj izlazni signal svim neuronima s kojima je povezan. Struktura umjetnog neurona prikazana je slikom 3.2.

Slanje izlaznog signala kontrolirano je aktivacijskom funkcijom. Ta funkcija zapravo definira način na koji se ponderirana suma transformira u izlaz.



Slika 3.2: Umjetni neuron

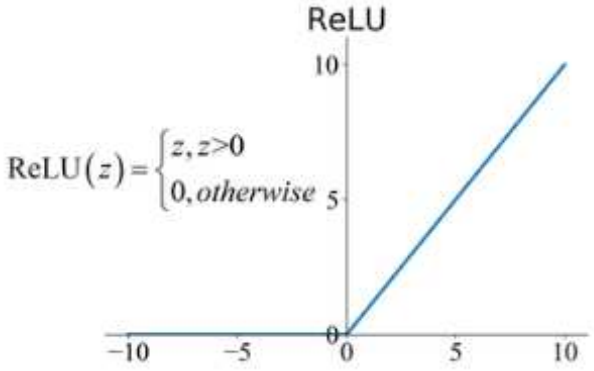
3.2. Aktivacijska funkcija

Aktivacijske funkcije izuzetno su važan dio implementacije neuronskih mreža te imaju velik utjecaj na izvođenje neuronskih mreža. Neuronska mreža sastoji se od ulaznog sloja, skrivenih slojeva te izlaznog sloja. Izbor aktivacijskih funkcija u skrivenom sloju neuronskih mreža utjecat će na kvalitetu učenja same neuronske mreže, dok će izbor aktivacijske funkcije na izlaznom sloju utjecati na vrste predikcija koje neuronska mreža može raditi.

Aktivacijske funkcije definiraju način na koji se ponderirana suma transformira u izlaz neurona. U skrivenim slojevima obično se koriste nelinearne aktivacijske funkcije koje uvode element nelinearnosti u dizajnu mreže. Osim toga, time se omogućava neuronskoj mreži da uči kompleksnije funkcije od neke neuronske mreže koja koristi linearne aktivacijske funkcije. Neke od aktivacijskih funkcija koje se koriste za skriveni sloj su:

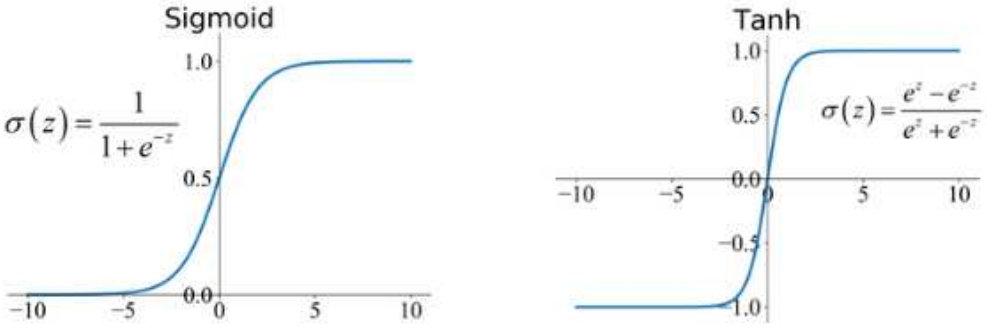
- ReLu funkcija
- Sigmoidna funkcija
- Hiperbolička tangencijalna funkcija

ReLu funkcija, poznata i kao funkcija ispravljene linearne jedinice, je funkcija koja za sve negativne vrijednosti daje nulu, a na sve pozitivne vrijednosti djeluje kao linearna funkcija. ReLu je zbog svoje efikasnosti jedna od najčešće korištenih funkcija za skriveni sloj. Iako je naizgled vrlo jednostavna, pokazala se vrlo djelotvornom upravo zbog svoje nelinearnosti. ReLu funkcija prikazana je slikom 3.3.



Slika 3.3: ReLu funkcija[8]

Sigmoidna i hiperbolička tangencijalna funkcija također su nelinearne. Razlika između tih dviju funkcija je u tome što sigmoidna funkcija daje izlaze između -1 i 1, a hiperbolička funkcija između 0 i 1. Sigmoidna i hiperbolička tangencijalna funkcija prikazane su slikom 3.4.



Slika 3.4: Sigmoidna i hiperbolična funkcija[8]

Izlazni sloj je dio neuronske mreže koji daje izlaznu vrijednost. U tom sloju najčešće se koriste:

- Linearna funkcija
- Sigmoidna funkcija

Za probleme regresije, najčešće se koristi linearna aktivacijska funkcija dok se za probleme klasifikacije češće koristi sigmoidna funkcija.

3.3. Princip učenja mreže

Kao što je već spominjano, neuronska mreža je skup velikog broja umjetnih neurona koji su međusobno povezani i interaktivni. Mreža se sastoji od ulaznog sloja, jednog ili više skrivenog sloja te izlaznog sloja. Umjetni neuroni jednog sloja povezani su s umjetnim neuronima narednog sloja. U nekim slučajevima, moguće su i povratne veze.

Danas postoji velik broj vrsta neuronskih mreža te ih je teško sustavno kvalificirati. Jedna od postojećih podjela je na *feedforward* te *feedback* mreže. Bez obzira na vrstu, oblik ili veličinu, sve neuronske mreže karakterizira sposobnost učenja.

Rad neuronske mreže započinje fazom učenja nakon koje slijedi faza testiranja. Prije samog učenja potrebno je odrediti skup ulaznih podatke te skup odgovarajućih izlaza. Prikupljene podatke potrebno je podijeliti na dva uzorka koji će se koristiti za samo učenje ali i testiranje naučene mreže.

Učenje neuronskih mreža nije ništa drugo doli postupak postepenog i iterativnog mijenjanja težinskih faktora. Neuronska mreža će, uzevši u obzir stvarne izlaze za dane ulazne podatke, postepeno prilagođavati težinske faktore dok određeni kriterij nije zadovoljen.

3.3.1. Gradijentni spust

Danas, zahvaljujući razvoju tehnologije postoje mnoga pravila i metode učenja mreža. Jedna od najkorištenijih strategija optimizacije je korištenje gradijentnog spusta. Ta metoda postavila je temelje za strojno i duboko učenje. Gradijentni spust iterativni je algoritam koji se koristi za pronalazak minimuma određene diferencijabilne funkcije. Taj se princip u učenju koristi za minimiziranje funkcije troška.

Kao što je već spominjano, prije samog treniranja mreže, potrebno je odrediti podatke koji se uče, odnosno ulaze i očekivane izlaze. Funkcija koja računa pogrešku između predviđenih i stvarnih izlaza, naziva se funkcija troška. Postoje različiti načini računanja funkcije troška te oni uvelike ovise o samom problemu koji se rješava neuronskom mrežom. Nakon izračuna funkcije troška, računa se gradijent.

Gradijent je jedan od osnovnih koncepata u vektorskoj analizi te nije ništa drugo doli viševarijabilna generalizacija derivacije. Ako se promatraju funkcije s jednom varijablom, govori se o derivacijama, dok za funkcije s više varijabli tu ulogu preuzima gradijent. Jednako kao i derivacija, gradijent određuje nagib tangente grafa funkcije. Time se zapravo određuje za koliko se izlaz funkcije promijeni ako se malo promjeni

ulaz.

Računanje gradijenta omogućava određivanje smjera greške. U neuronskim mrežama ta se vrijednost koristi kako bi se znalo u kojem smjeru treba mijenjati vrijednosti te ih propagirati po neuronskoj mreži algoritmom propagacije unazad.

Računanje gradijenta samo je jedna od mnogih metoda koji se koriste za optimiziranje težina neuronskih mreža. Posebno zanimljiva je tehnika koja koristi genetski algoritam za poboljšavanje težinskih faktora. Osim toga, genetski algoritmi mogu se koristiti i za određivanje same arhitekture neuronske mreže. Takav pristup optimizaciji neuronskih mreža naziva se neuroevolucijom.

4. Genetski algoritam za razvoj neuronskih mreža

Neuroevolucija je tehnika strojnog učenja koja koristi evolucijske algoritme u optimiziranju različitih parametara neuronskih mreža. U ovom poglavlju, naglasak će biti stavljen na postepeno prilagođavanje težinskih faktora neuronskih mreža, dok će se u idućim poglavljima govoriti o mogućnosti oblikovanja same arhitekture neuronske mreže.

Za razliku od ostalih metoda učenja, primjerice gradijentnog spusta, o kojem se govorilo u prijašnjem poglavlju, metoda neuroevolucije je općenita. Omogućava učenje bez eksplicitno zadanih ciljeva i bez korištenja diferencijabilnih aktivacijskih funkcija. Najveću popularnost metodi neuroevolucije daje upravo činjenica da ne koristi gradijent. Time se izbjegava potencijalno zaglavljivanje u lokalnom minimumu, što prilikom korištenja metode gradijentog pada predstavlja veliki problem.

Iako su genetski algoritmi dobili na popularnosti kada su u pitanju neuronske mreže, vrlo ih se često kritizira zbog sporog učenja i velike računske zahtjevnosti.

Jasno je da učinkovitost neuroevolucije ima svoje prednosti i mane, no sam koncept genetskih algoritama u ulozi optimiranja neuronskih mreža je nedvojbeno zanimljiv.

4.1. Programsko ostvarenje neuroevolucije

Genetski algoritmi stvaraju više rješenja za određeni problem te ih principima evolucije unaprjeđuju kroz generacije. Svaka jedinka populacije, zapravo je potencijalno rješenje i sadrži parametre koji mogu pomoći optimizirati rezultat.

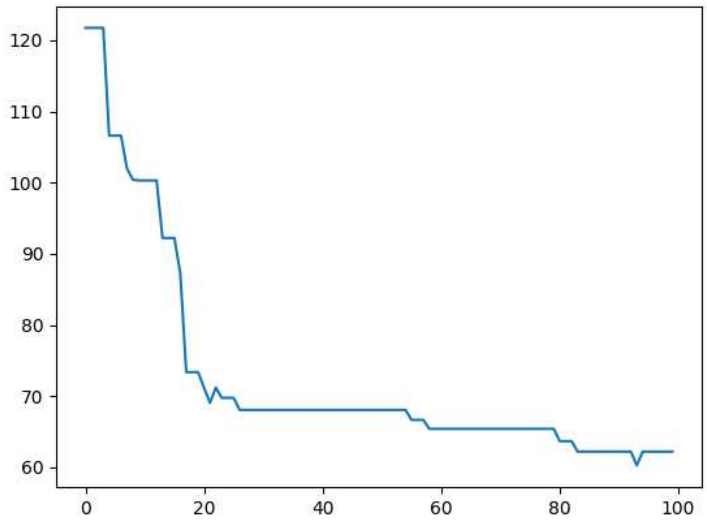
U slučaju optimiziranja neuronske mreže, jedno rješenje, odnosno genom unutar populacije sadržavat će sve težine unutar neuronske mreže. Primjerice, ako neuronska mreža ima tri sloja, ulazni s dva neurona, skriveni s deset neurona te izlazni jednim neuronom, svaki genom imat će trideset gena, odnosno težina.

Kao što je već spominjano, implementacija genetskog algoritma započinje s odabi-

rom reprezentacije genoma. Zbog jednostavnosti se u prijašnjim poglavljima koristila reprezentacija binarnim vektorom. Prikaz podataka uvelike ovisi o problemu koji se rješava. U slučaju neuronskih mreža, geni su zapravo težinski faktori, odnosno realni brojevi pa inicijalizacija populacije nije ništa drugo nego generiranje vektora nasumičnih realnih brojeva.

Tako prikazani podatci postepeno se evoluiraju genetskim algoritmom. Kvaliteta jedinki određuje se funkcijom dobrote koja je implementirana kao razlika predikcija neuronske mreže i stvarnog izlaza. Potom se vrši turnirska selekcija koja vraća četiri roditelja. Roditelji se križaju kombiniranjem težinskih faktora. Nasumično izabrani geni njihove djece bit će promijenjeni na način da se težinskim vrijednostima u tim genima oduzme ili pribroji mala nasumična vrijednost. Nakog toga, potrebno je odabrati novu populacija koja sadrži nove ali i stare jedinke. Postupak se ponavlja sto generacija.

Takav proces trebao bi optimizirati težine i time postepeno poboljšavati sposobnost neuronske mreže da radi predikcije. Smanjenje funkcije dobrote, odnosno greške predikcije, kroz generacije prikazano je slikom 4.1.



Slika 4.1: Smanjenje greške kroz generacije

Konkretna slika odnosi se na problem predikcije temperature koristeći ostale spojeve u zraku. U danjim poglavljima detaljnije će se govoriti o primjeni neuroevolucije za probleme regresije nad različitim skupovima podataka te će se dati uvid u rezultate koje neuroevolucija može postići.

5. Genetski algoritam za razvoj arhitekture neuronskih mreža

Iako se do sada pričalo o korištenju genetskog algoritma u svrhu optimiziranja težinskih faktora neuronskih mreža, genetski algoritam može se upotrijebiti i za poboljšanje mnogih drugih parametara, kao što je i sama topologija neuronske mreže. Na vrlo apstraktnoj razini, ideja je vrlo jednostavna. Umjesto određivanja fiksne strukture neuronske mreže, dopušta se genetskom algoritmu da odredi što pogodniju arhitekturu, broj neurona te broj slojeva.

Jedan od algoritama koji se koristi za generiranje i optimiziranje arhitektura naziva se NEAT, odnosno *NeuroEvolution of Augmenting Topologies*. Prije NEAT algoritma nije postojalo puno uspješnih algoritama. Sve ideje imale su podosta problema koji su se morali riješiti prije nego što bi odgovarajući algoritmi mogli raditi. NEAT je uspješno riješio te probleme te je uspio iskoristiti prednosti evoluiranja topologija neuronskih mreža.

5.1. NEAT

NEAT ili *NeuroEvolution of Augmenting Topologie* je genetski algoritam za generiranje i evoluiranje umjetnih neuronskih mreža. Osmislio ga je i razvio Ken Stanley, 2002. godine. Taj algoritam mijenja težinske faktore i samu strukturu neuronske mreže, istovremeno pazeći na kvalitetu rješenja i zadržavanje različitosti unutar populacija. Temelji se na tri ključne tehnike: označavanju i praćenju gena, korištenju vrsta za određivanje sličnih topologija te postepenoj optimizaciji jednostavne početne strukture.

Tradicionalno se arhitektura neuronske mreže odabirala prije same implementacije, dok su se težinski faktori mijenjali i optimizirali u procesu učenja. Takvim se postupkom stvara situacija u kojoj je potrebno puno različitih pokušaja manualnog određivanja topologije, kako bi se odredilo koja je najpogodnija.

NEAT proces optimiziranja započinje generiranjem vrlo jednostavne strukture neuronske mreže te kao i svaki genetski algoritam, tehnikama selekcije, križanja i mutacije, postepeno unaprjeđuje jedinke koje su u ovom slučaju neuronske mreže. Naravno, svaka implementacija genetskog algoritma započinje određivanjem reprezentacije podataka.

5.2. Reprezentacija podataka u NEAT algoritmu

U biologiji se često spominju pojmovi genotip i fenotip. Genotip je genetska reprezentacija određene jedinke, dok je fenotip skup svih njezinih vidljivih karakteristika i osobina. Fenotip rezultira iz genotipa, odnosno genetskog koda. Svi genetski algoritmi pa tako i NEAT, pronalaze inspiraciju u prirodi za mnoge metode ali i za samu reprezentaciju podataka. Time se zapravo postavlja pitanje kako na najefikasniji način prikazati genotipe jedinki, uzevši u obzir da takva reprezentacija mora biti pogodna za evolucijske procese selekcije, mutacije i križanja.

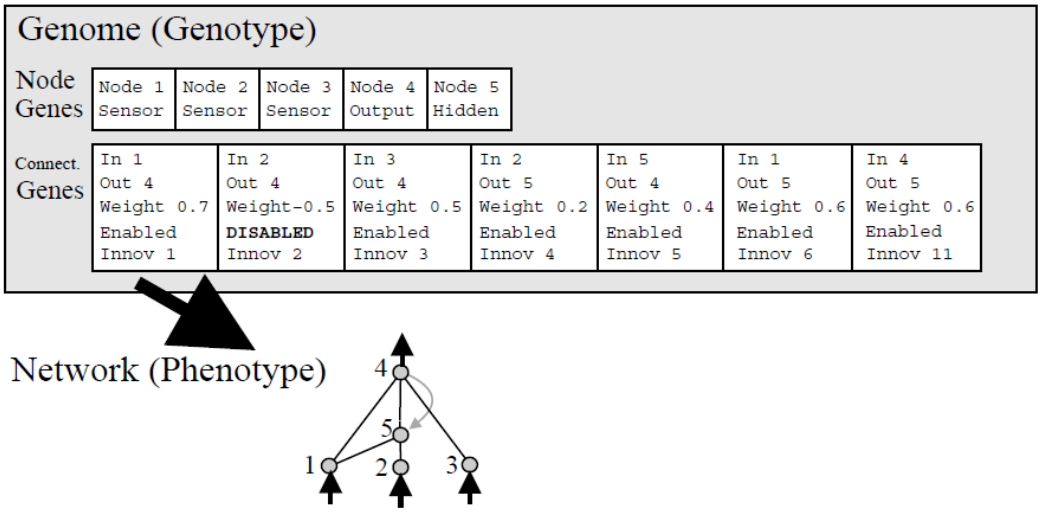
NEAT algoritam koristi direktno kodiranje. Takva reprezentacija eksplicitno specificira sve o određenoj jedinki. U slučaju neuronskih mreža to znači da će svaki gen biti direktno povezan s jednim neuronom, vezom ili nekim drugim svojstvom mreže. Direktnu vrstu prikaza koristili smo i do sada, no reprezentacija koja se koristi za NEAT je ipak malo kompliciranija.

Za reprezentaciju genoma u NEAT algoritmu koriste se dvije liste gena. Jedna se primjenjuje za pohranu gena koji određuju neurone, dok se druga koristi za pohranu konekcija između njih. Primjer takve reprezentacije prikazan je slikom 5.1. Bitno je napomenuti da se ulazni i izlazni neuroni ne mijenjaju u procesu evoluiranja.

Geni za konekciju sadrže informaciju o tom koja dva gena povezuju, iznos težine te veze i informaciju o tome jeli veza *enabled* ili *disabled*. Osim toga, i neuroni i konekcije sadrže inovacijski broj.

Inovacijski broj koristi se kako bi se omogućilo križanje. Pretpostavimo slučaj u kojem jedan genom sadrži neurone A, B i C te je reprezentiran listom koja redom sadrži gene A, B i C. Kada bi se taj genom križao s nekim drugim, funkcionalno identičnim genomom, koji je reprezentiran listom koja redom sadrži gene C, B, A nastao bi veliki problem. Naime, dijete koje bi nastalo takvim križanjem imalo bi genom oblika A, B, A ili C, B, C. Time bi se izgubila jedna trećina informacije koju su ti genomi inače nosili. NEAT rješava taj problem koristeći inovacijski broj.

Inovacijski broj je neka vrsta oznake koja se pridjeljuje određenom genu. Prilikom stvaranja novog gena, provjerava se postoji li već taj gen. Ako postoji, dodjeljuje



Slika 5.1: Reprezentacija genoma[16]

mu se postojeći inovacijski broj. U suprotnom, stvara se novi inovacijski broj koji će označavati novonastali gen.

5.3. Selekcija u NEAT algoritmu

NEAT algoritam koristi jako zanimljive metode u procesu selekcije. U prošlim poglavljima pričalo se o tome da je selekcija postupak odabiranja roditelja. Roditelji bi trebali biti što bolje jedinke kako bi njihova djeca mogla naslijediti pozitivne karakteristike. Takvim se postupkom omogućava postepeno poboljšavanje populacije.

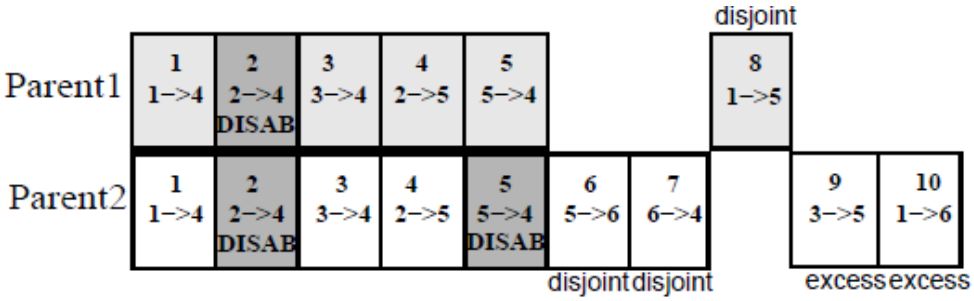
Princip rada NEAT algoritma je u suštini jednak svim genetskim algoritmima te se temelji na uvođenju inovacija u strukture jedinki. No, iako je cilj inovacija optimiziranje struktura, često se dogodi da novonastala promjena u početku zapravo ima negativan utjecaj. Primjerice, dodavanjem nove konekcije, sposobnost jedinke da riješi zadani zadatak bit će smanjena jer se težina novonastale veze još nije stigla optimizirati. Nažalost, zbog početnog smanjenja funkcionalnosti, može se dogoditi da se ta jedinka proglasi lošom u procesu selekcije. Kako bi se svim jedinkama omogućilo da ne budu zapostavljene zbog stvaranja promjena i nedovoljnog vremena za optimizaciju, NEAT algoritam koristi vrste.

Vrste su osnovne jedinice biološke raznovrsnosti. Uobičajena definicija određuje vrstu kao grupu prirodnih populacija rasplodno izoliranih od drugih grupa. Rasplodna izoliranost znači da se jedinke iz odvojenih grupa ne mogu križati.

Vrste se u NEAT algoritmu koriste kao svojevrsna zaštita jedinki. Strukture ne-

uronskih mreža ne natječu se s cijelom populacijom, već samo s jedinkama unutar vlastite vrste. Tim se postupkom promjene u topologijama štite na način da im se dopušta da se optimiziraju. Dakle, ideja je da se cijela populacija podijeli u vrste koje se određuju po razini sličnosti struktura.

Razina različitosti struktura naziva se i distancom između dvije jedinke. Naravno, što je distanca veća, to su jedinke različitije. Geni jedinki koji nemaju isti inovacijski broj, odnosno nisu jednaki mogu biti odvojeni geni ili geni viška. Odvojeni geni, odnosno *disjoint genes* nalaze se unutar intervala inovacijskog broja druge jedinke, dok se viška geni, odnosno *excess genes*, nalaze izvan tog intervala. Razlika između disjoint i excess gena prikazana je slikom 5.2.



Slika 5.2: Reprezentacija genoma[16]

Distanca dviju jedinki računa se kao linearna kombinacija različitih gena i razlika u težini. Izračunata vrijednost se potom koristi za određivanje vrste kojoj pripada određena jedinka. Ako je distanca između odabrane jedinke i predstavnika vrste manja od određenog praga, ta će jedinka pripadati upravo toj vrsti.

5.4. Reprodukcija u NEAT algoritmu

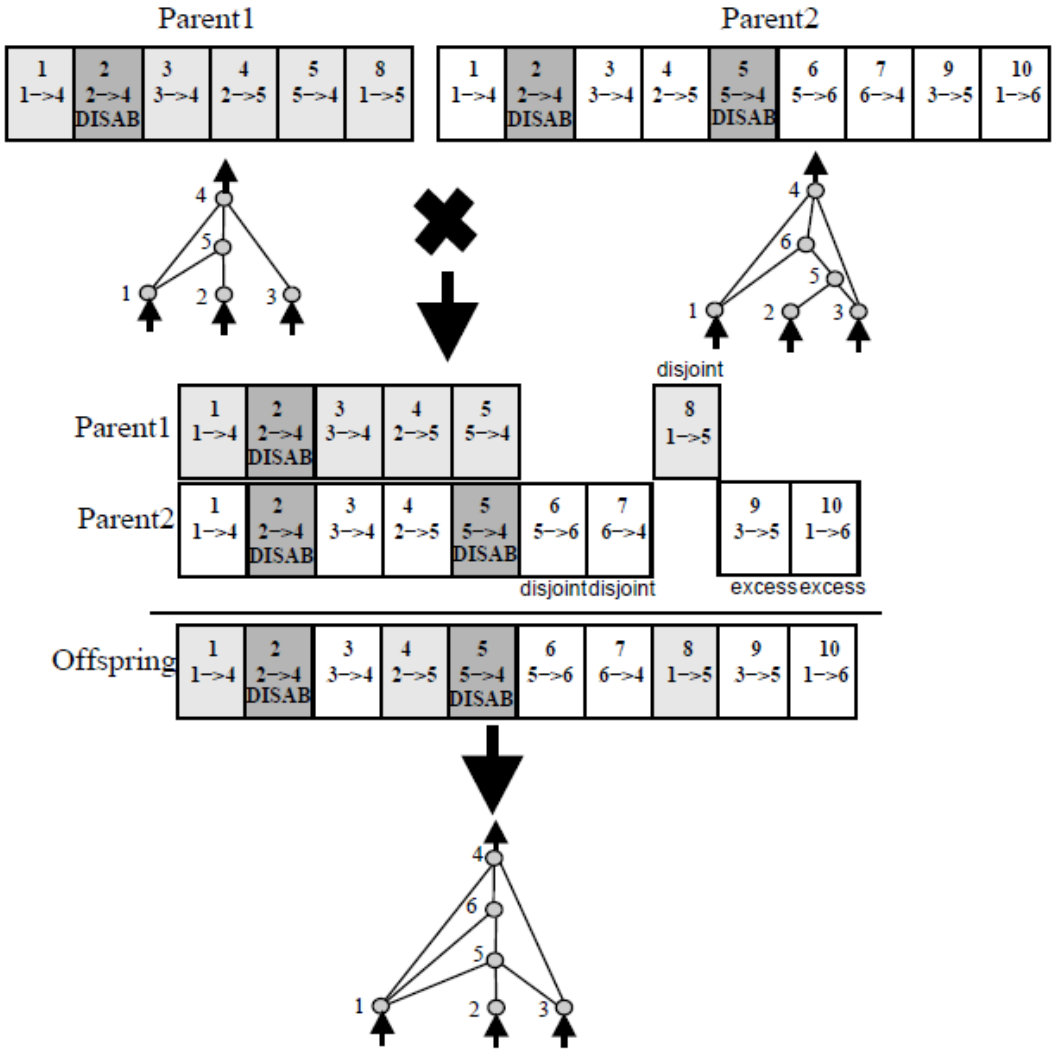
Baš kao i svaki genetski algoritam, NEAT algoritam implementira reprodukciju između selektiranih jedinki. Cilj reprodukcije je stvoriti nove jedinke koje nasljeđuju svojstva svojih roditelja.

5.4.1. Križanje

Križanje jedinki je u NEAT algoritmu malo kompleksnije zbog problema koji se naziva *competing conventions*, odnosno konkurentne konvencije. O samom problemu govori se više u poglavlju o reprezentaciji podataka. Naime, problem nastaje kada se zbog

loše reprezentacije jedinki, prilikom križanja, izgube određene informacije. Taj se problem rješava inovacijskim brojem.

Konekcijski geni roditelja izabranih za križanje se, na početku križanja, poredaju i poravnaju po inovacijskom broju. Ako geni imaju isti inovacijski broj znači da se radi o jednakim genima te se u tom slučaju nasumično odabire roditelj čiji će gen biti naslijeđen. Geni viška, odnosno *excess genes*, bit će naslijeđeni samo ako dolaze od pogodnijeg roditelja. Slika 5.3 prikazuje proces križanja jedinki u NEAT algoritmu. U slučaju koji je prikazan slikom, *Parent2* smatra se pogodnijom jedinkom.

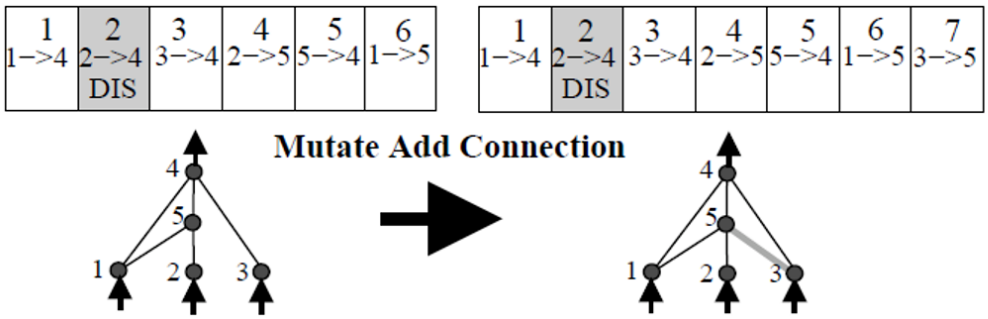


Slika 5.3: Prikaz križanja[16]

5.4.2. Mutacija

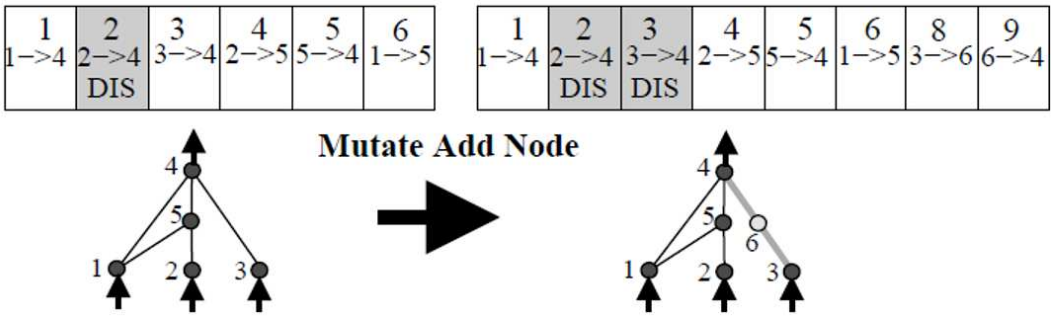
Uloga mutacija u genetskim algoritmima je unijeti inovaciju u genome jedinki. U NEAT algoritmu mutacija može promijeniti već postojeću konekciju ili može uvesti novu strukturu.

Mutacija može dodati novu konekciju između dva nasumično odabrana gena. Ako veza između ta dva gena već postoji, biraju se novi geni. U suprotnom se stvara nova konekcija kojoj se pridodaje nasumična težina. Mutacija dodavanjem konekcija prikazana je slikom 5.4.



Slika 5.4: Prikaz mutacije dodavanjem nove konekcije[16]

Osim dodavanja konekcije, postoji mutacija dodavanjem neurona. U tom se slučaju novi neuron postavlja između dva već postojeća neurona, a prijašnja konekcija se onesposobljava. Prijašnji početni i dodani novi čvor povezuju se konekcijom s težinskim faktorom 1, dok se novi čvor i prijašnji krajnji čvor povezuju konekcijom s težinskim faktorom ekvivalentnim onom u prijašnjoj vezi. Takva mutacija prikazana je slikom 5.5.



Slika 5.5: Prikaz mutacije dodavanjem novog neurona[16]

Osim promjena u samoj strukturi neuronskih mreža, NEAT algoritam implementira i postepeno optimiziranje težinskih faktora postojećih konekcija.

6. Usporedba neuroevolucije kod fiksne i varijabilne arhitektura

Genetski algoritmi se, kao što je spominjano u prijašnjim poglavljima, mogu koristiti za mnoge optimizacijske probleme. Jedan od njih je i poboljšavanje kvalitete neuronskih mreža. U ovom je radu, genetski algoritam prvo korišten za optimizaciju težina neuronske mreže fiksne strukture te je zatim primijenjen i na samu arhitekturu neuronskih mreža.

U ovom poglavlju se, na temelju četiri primjera, uspoređuje rad neuronske mreže fiksne strukture s onom varijabilne strukture. U sva četiri primjera radi se o problemu regresije, dakle za zadane ulazne podatke pokušava se odrediti izlaz. Primjeri su zapravo funkcije različitih složenosti.

Prva i najjednostavnija funkcija naziva se *easy* funkcija te glasi:

$$5x$$

Zatim se koristila *medium* funkcija s jednom varijablom više i elementom kvadriranja:

$$5x + 0.5y^2$$

Osim toga koristila se još složenija funkcija, *hard* funkcija s još jednom dodanom varijablom i elementom međudjelovanja dviju varijabli :

$$0.1xy + 0.04y^2 + 0.2z$$

Na kraju se koristio skup podataka koji sadrži razinu određenih spojeva u zraku te izmjerenu temperaturu. Cilj je iz danih podataka razine spojeva odrediti temperaturu zraka. U tom slučaju, uopće nije sigurno da postoji određena zavisnost ulaznih i izlaznih podataka.

Naravno skupovi podataka podijeljeni su na dio za učenje i dio za testiranje. Skup za učenje sadrži 70% ukupnih podataka dok skup za testiranje sadrži 30% ukupnih podataka. Za stvarne funkcije *easy*, *medium* te *hard* cijeli skup podataka sadrži 3000

različitih ulaznih podataka od kojih 2100 pripada skupu za učenje, a 900 skupu za testiranje. Funkcija zraka je malo kompleksnija te sadrži 9358 korisnih ulaza od kojih se 6550 nalazi u skupu za treniranje, a 2808 u skupu za testiranje.

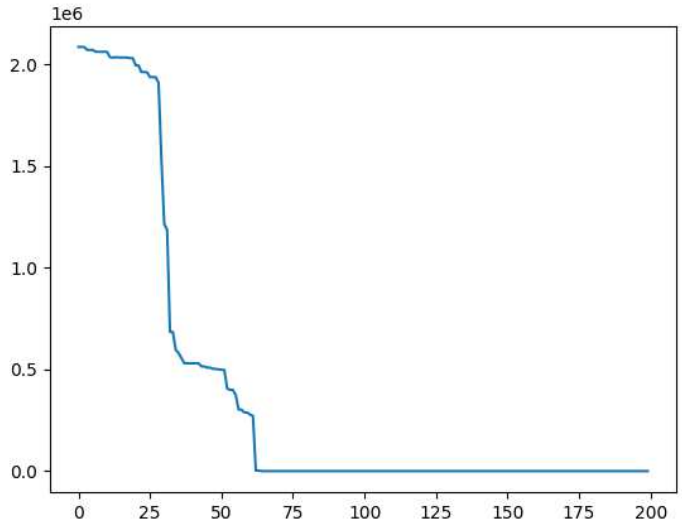
Prilikom učenja, kao mjera dobrote korištena je mjera MSE odnosno prosječna kvadrirana greška. Iz tog razloga, poželjno je da se funkcija dobrote, koja predstavlja razliku između predikcije i stvarnog izlaza, smanjuje. Inicijalna populacija, naravno nije prošla proces evolucije, odnosno optimizacije te bi ona trebala imati najveće odstupanje prilikom davanja predikcija. Proces evolucije trebao bi optimizirati populaciju kroz niz generacija. Ta optimizacija jasno je vidljiva u padu funkcije dobrote.

Na slikama u nastavku prikazan je pad funkcije dobrote kroz generacije. Na apscisi su istaknute generacije u procesu evolucije dok je na ordinati prikazana funkcija dobrote odnosno izračunat MSE za predikcije populacije u točno toj generaciji.

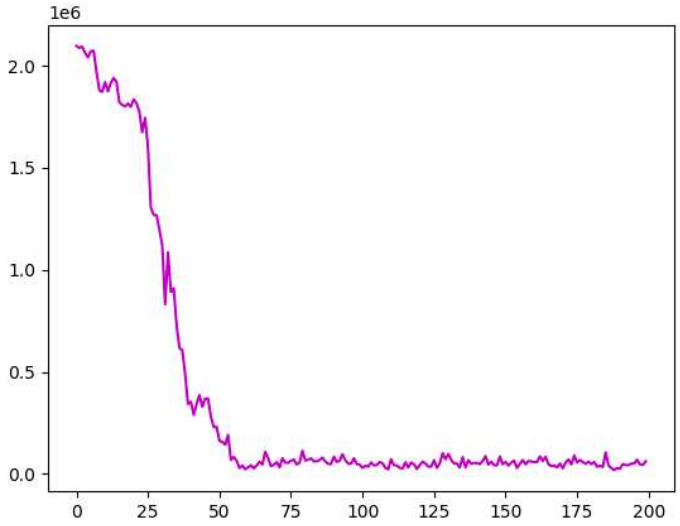
U nastavku je prvo prikazana funkcija dobrote u procesu učenja za primjer *easy* funkcije. Slika 6.1 odnosi se na neuronsku mrežu fiksne arhitekture, dok je na slici 6.2 prikazana funkcija dobrote neuronske mreže varijabilne arhitekture.

Bitno je napomenuti kako je neuronska mreža fiksne strukture vrlo jednostavna te sadrži samo jedan skriveni sloj.

U oba slučaja primjećuje se drastični pad funkcije dobrote što je pozitivno. Isto tako, za obje funkcije genetski je algoritam evoluirao dvjesto generacija. Naime to je nepotrebno, uzevši u obzir da se funkcija dobrote nije smanjila nakon sedamdeset i pet generacija.

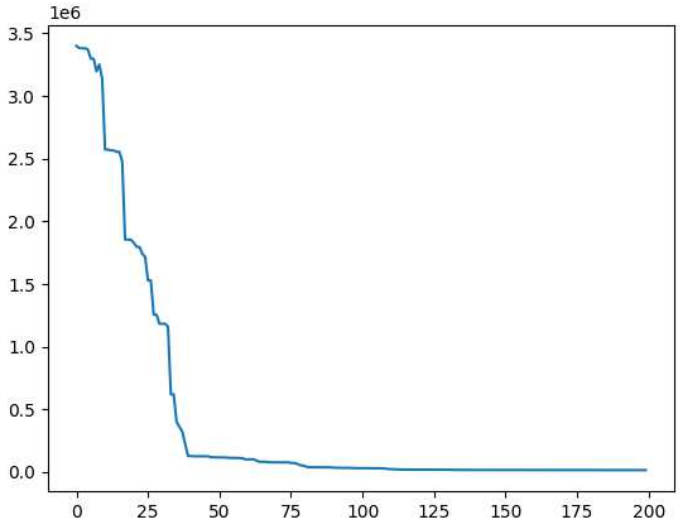


Slika 6.1: Funkcije dobrote dobivene neuronskom mrežom fiksne arhitekture

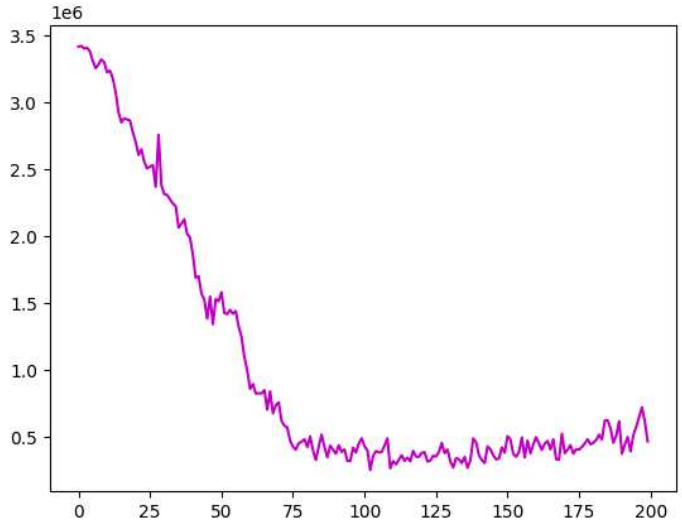


Slika 6.2: Funkcije dobrote dobivene neuronskom mrežom varijabilne arhitekture

Na primjeru *medium* funkcije ponovo se uočava padanje funkcije dobrote za obje neuronske mreže. No, u slučaju korištenja neuronske mreže varijabilne arhitekture, prvo dolazi do naglog pada te se potom funkcija dobrote ne stabilizira potpuno već nastavlja varirati u određenom intervalu. Osim toga, jasno se vidi da funkcija dobrote neuronska mreža fiksne arhitekture na posljetku postiže niže vrijednosti što ukazuje na to da će bolje raditi predikcije. Opisane funkcije dobrote prikazane su slikama 6.3 i 6.4.

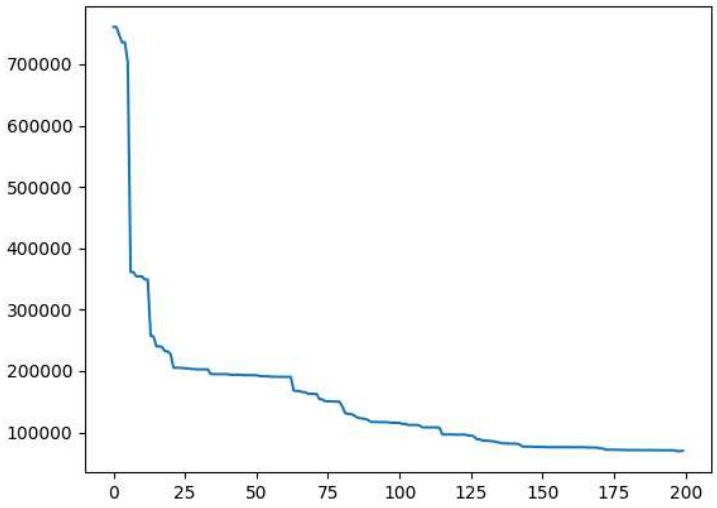


Slika 6.3: Funkcije dobrote dobivene neuronskom mrežom fiksne arhitekture

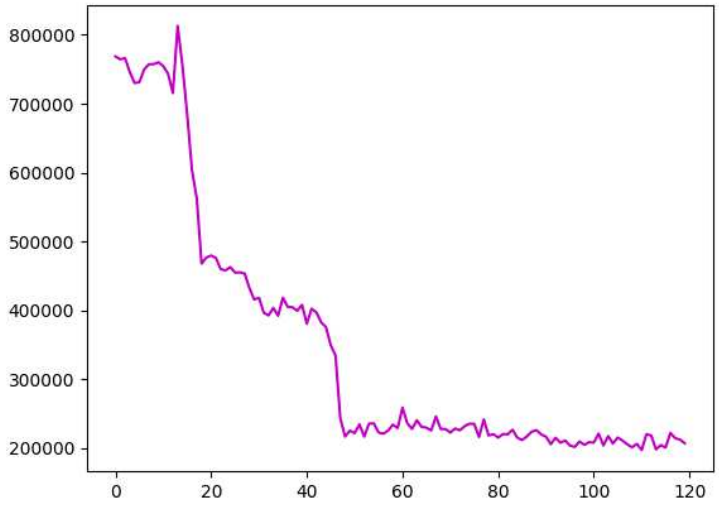


Slika 6.4: Funkcije dobrote dobivene neuronskom mrežom varijabilne arhitekture

Za *hard* funkciju, ponovno se primjećuje pad funkcija dobrote. I u ovom slučaju funkcija postiže niže vrijednosti za neuronsku mrežu fiksne arhitekture. Funkcije dobrote za *hard* funkciju prikazane su na slikama 6.5 i 6.6.



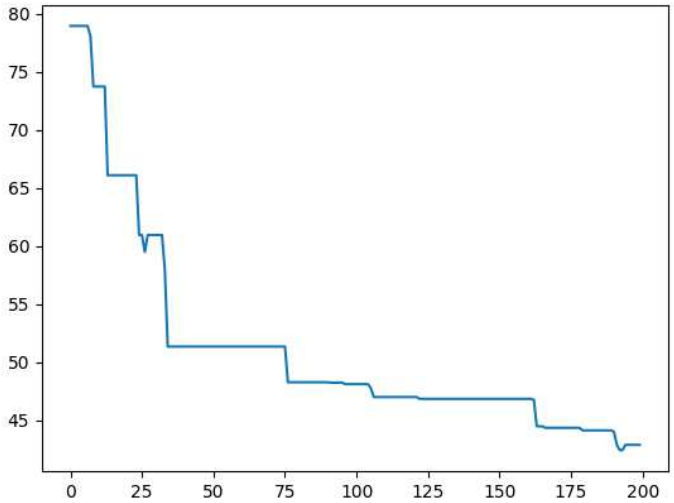
Slika 6.5: Funkcije dobrote dobivene neuronskom mrežom fiksne arhitekture



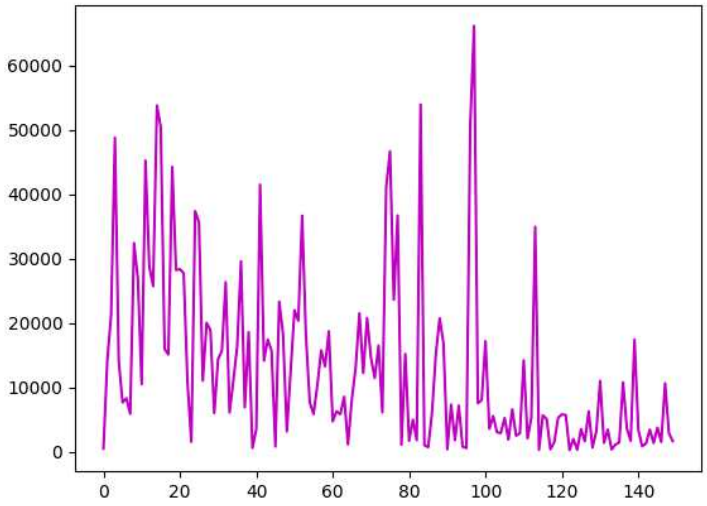
Slika 6.6: Funkcije dobrote dobivene neuronskom mrežom varijabilne arhitekture

Funkcije dobrote dobivene učenjem nad skupom podataka spojeva u zraku, podosta se razlikuju. Korištenjem neuronske mreže fiksne arhitekture ponovno se dobiva padajuća funkcija dobrote dok se za neuronsku mrežu varijabilne arhitekture ne može uočiti jasan padajući trend. To je prikazano slikama 6.7 i 6.8.

Uzevši u obzir vrlo loš pad funkcije dobrote na ovom primjeru te manji i nesigurniji pad na prošlim primjerima, može se pretpostaviti da će neuronska mreža dobivena NEAT algoritmom raditi slabije predikcije.



Slika 6.7: Funkcije dobrote dobivene neuronskom mrežom fiksne arhitekture

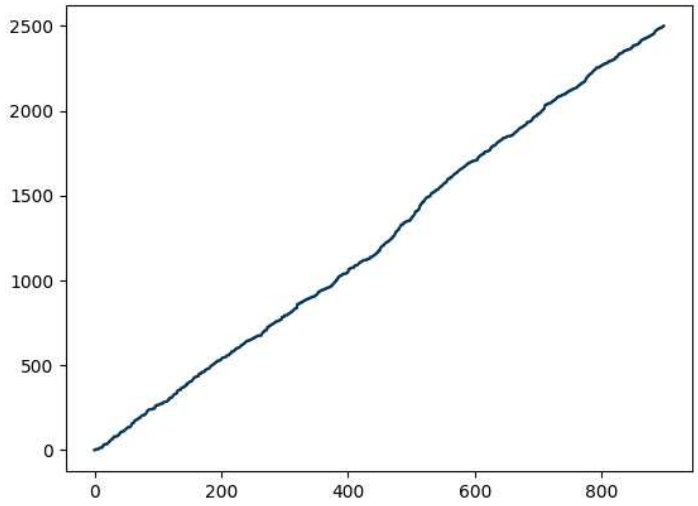


Slika 6.8: Funkcije dobrote dobivene neuronskom mrežom varijabilne arhitekture

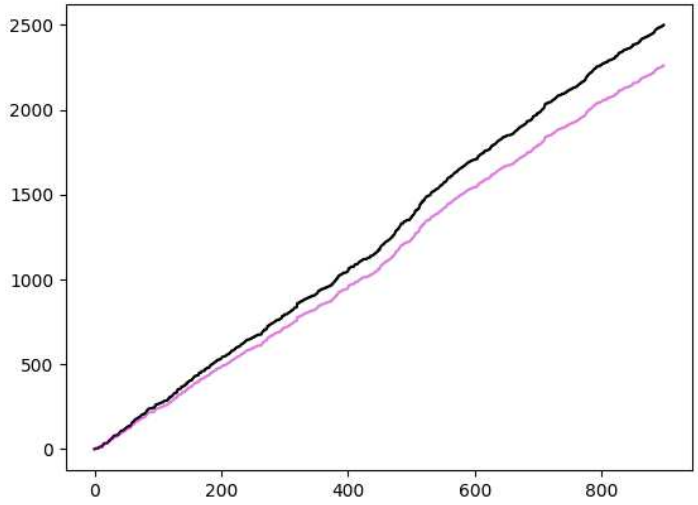
Kao što je već spomenuto, skup podataka podijeljen je na dio za učenje te dio za testiranje. Skup za testiranje predaje se već naučenoj neuronskoj mreži te sadrži nove, neviđene podatke. Neuronska mreža bi, na temelju tih podataka, trebala moći napraviti predikcije. Kvaliteta neuronske mreže, u tom je slučaju, upravo sposobnost da radi predikcije što sličnije stvarnom izlazu.

U nastavku su prikazane predikcije neuronskih mreža za spomenute primjere. Na osi apscisa označeno je o kojoj se predikciji radi dok je na osi ordinata plavom ili ružičastom označena vrijednost predikcije koju daje neuronska mreža fiksne ili varijabilne arhitekture. Osim toga, na grafovima su crnom bojom naznačene i stvarne odnosno očekivane vrijednost za tu funkciju. Iznosi stvarnih vrijednosti i odgovarajućih predikcija su sortirani uzlazno zbog lakšeg prikaza

Promatranjem predikcija za *easy* funkciju, prikazanih slikom 6.9, primjećuje se da neuronska mreža fiksne arhitekture radi gotovo savršene predikcije. Kao što se dalo naslutiti promatranjem funkcija dobrote, neuronska mreža dobivena NEAT algoritmom daje približno dobre predikcije no, one ipak nisu potpuno precizne. Te su predikcije prikazane slikom 6.10.

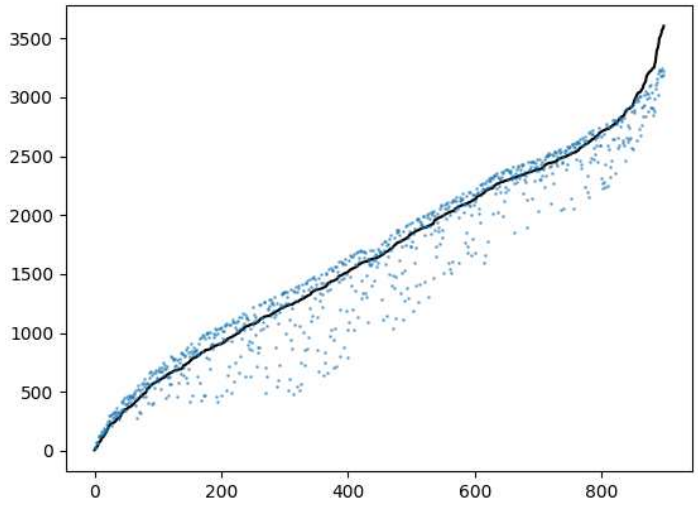


Slika 6.9: Predikcije easy funkcije neuronske mreže fiksne arhitekture

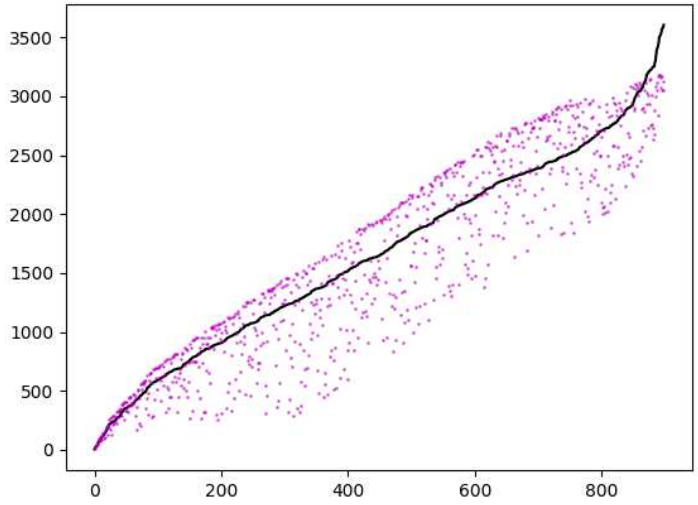


Slika 6.10: Predikcije easy funkcije neuronske mreže varijabilne arhitekture

Predikcije za *medium* funkciju prikazane su slikama 6.11 i 6.12. U ovom slučaju neuronska mreža fiksne arhitekture daje približno dobre predikcije. Naravno, ova neuronska mreža je zbog svoje jednostavne arhitekture ograničena po pitanju složenosti problema koje može uspješno riješiti. Iz tog razloga, očekivano je da će njene predikcije biti lošije za složenije funkcije. Neuronska mreža dobivena NEAT algoritmom već u ovom primjeru radi podosta slabije predikcije.

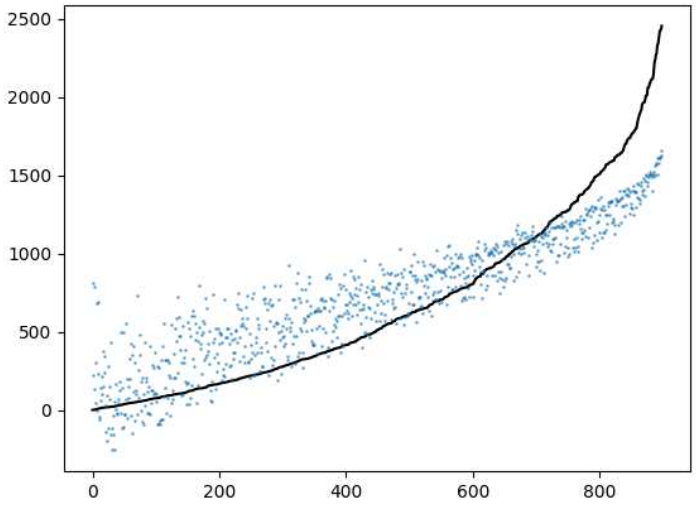


Slika 6.11: Predikcije medium funkcije neuronske mreže fiksne arhitekture

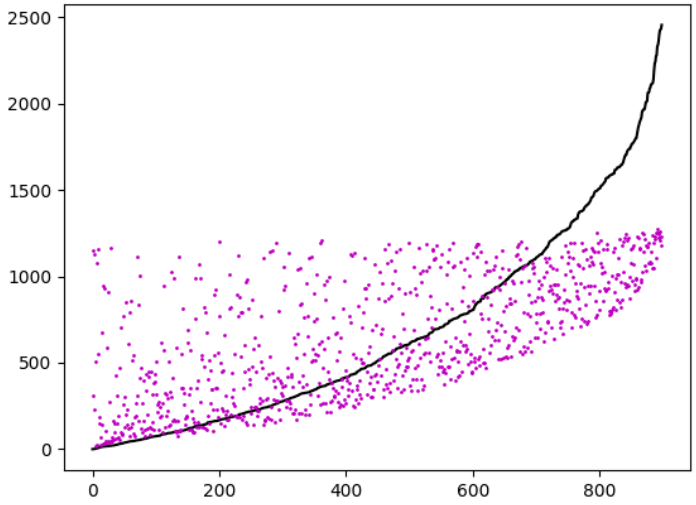


Slika 6.12: Predikcije medium funkcije neuronske mreže varijabilne arhitekture

Promatranjem predikcija za *hard* funkciju, prikazanih slikama 6.13 i 6.14 uočava se slabija sposobnost neuronskih mreža. Naime, obje mreže rade predikcije koje nisu u potpunosti nasumične no i dalje uvelike odstupaju od stvarnih vrijednosti.

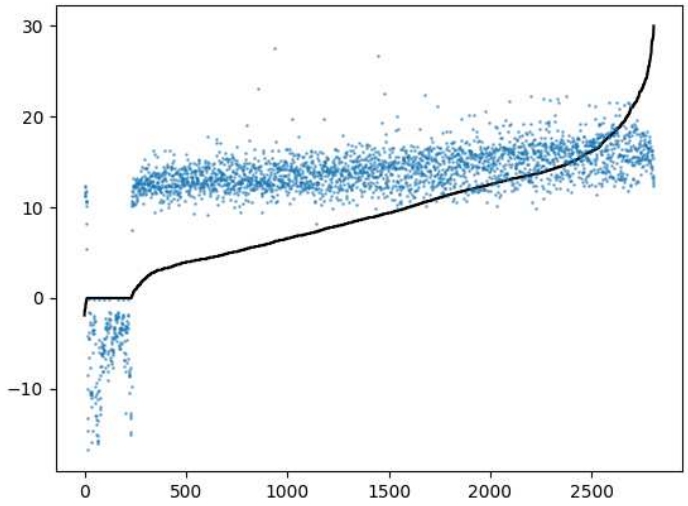


Slika 6.13: Predikcije hard funkcije neuronske mreže fiksne arhitekture

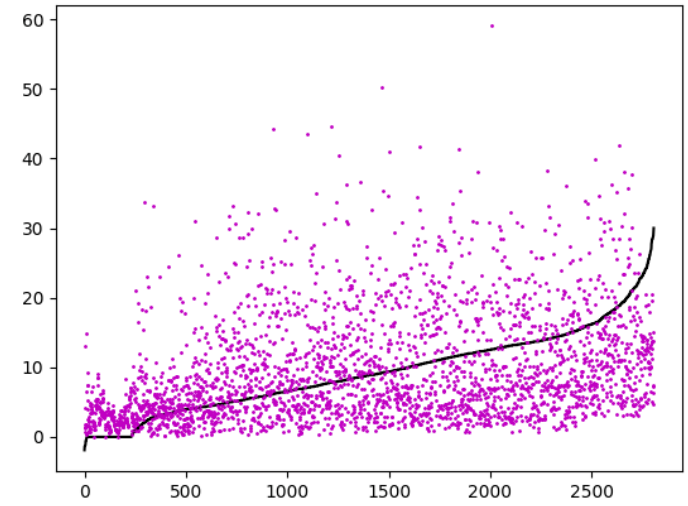


Slika 6.14: Predikcije hard funkcije neuronske mreže varijabilne arhitekture

Predikcije za funkciju zraka prikazane su slikama 6.15 i 6.16. Obje neuronske mreže daju vrlo slabe rezultate. Neuronska mreža dobivena NEAT algoritmom radi gotovo nasumične predikcije što se moglo pretpostaviti iz vrlo slabog napretka funkcije dobrote.



Slika 6.15: Predikcije funkcije zraka neuronske mreže fiksne arhitekture



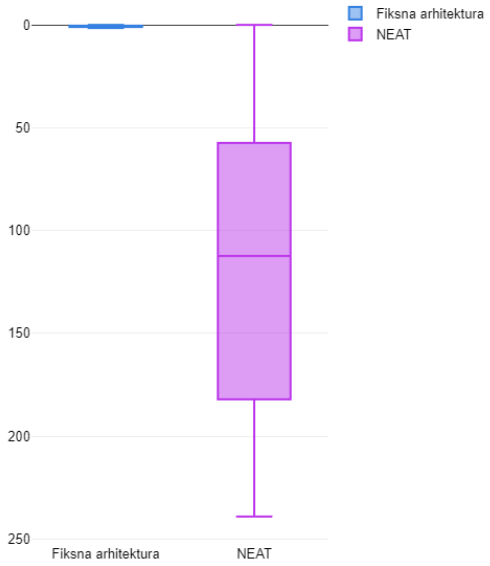
Slika 6.16: Predikcije funkcije zraka neuronske mreže varijabilne arhitekture

Detaljnije vrijednosti predikcija neuronskih mreža prikazane su tablicom 6.1. Za svaku funkciju izračunate su razlike stvarne vrijednosti i predikcije. U tablici je prikazano minimalno, maksimalno te prosječno odstupanje. Bitno je napomenuti kako se prilikom računanja odstupanja za tablični prikaz te boxplotove koristila apsolutna vrijednost razlike. To je drugačije od mjere MSE koja se koristila kao mjera dobrote rješenja. Naime u mjeri MSE odstupanje je kvadrirana razlika dok je u tabličnom i boxplot prikazu odstupanje izraženo kao apsolutna vrijednost razlike.

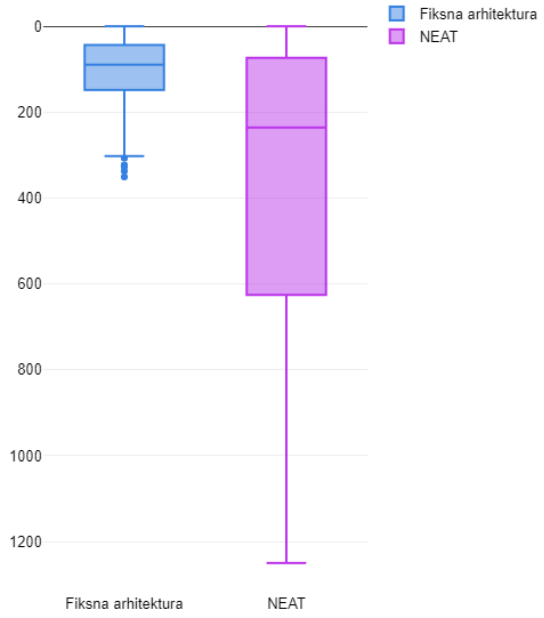
Tablica 6.1: Analiza predikcija

Funkcija	Vrijednost	Fiksna arhitektura	NEAT
easy funkcija	min	0.0016	0
	avg	0.7145	118.8744
	max	1.5215	239.0455
medium funkcija	min	0.0398	0.0149
	avg	100.0205	370.4209
	max	302.5405	1250
hard funkcija	min	0.1500	0.4689
	avg	203.1502	292.0003
	max	596.5387	1005.3471
funkcija zraka	min	0.0007	0.0128
	avg	3.3668	5.8112
	max	11.3418	16.796

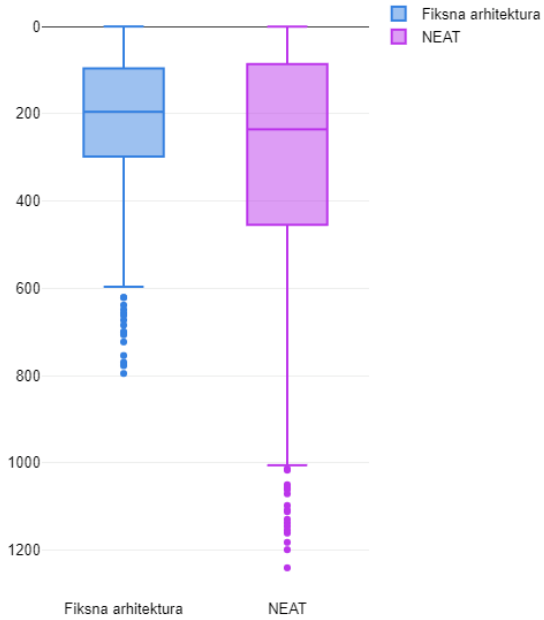
Osim tablicom, vrijednosti odstupanja vizualno su prikazane i *boxplotovima* na slikama 6.17, 6.18, 6.19 i 6.20. Na svakom prikazu dana su dva *boxplota*, jedan za neuronsku mrežu fiksne arhitekture a drugi za neuronsku mrežu dobivenu NEAT algoritmom. Svaki *boxplot* na osi ordinata prikazuje vrijednosti odstupanja predikcija već naučene mreže od stvarnih vrijednosti.



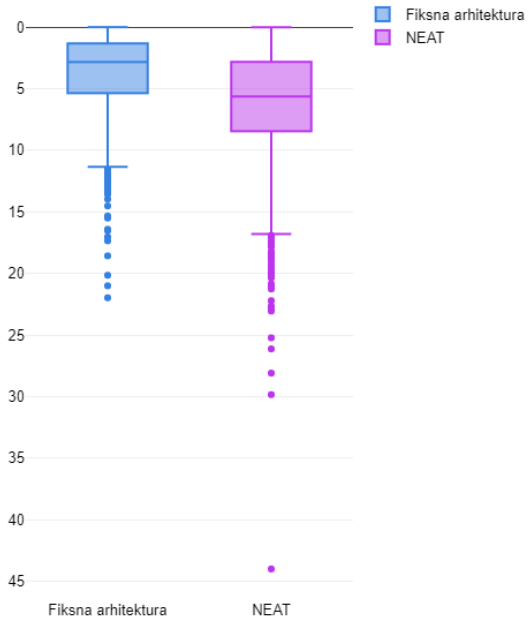
Slika 6.17: Boxplot predikcija easy funkcije



Slika 6.18: Boxplot predikcija medium funkcije



Slika 6.19: Boxplot predikcija hard funkcije



Slika 6.20: Boxplot predikcija funkcije zraka

Na vizualnim prikazima jasno se uočava da je odstupanje rezulta neuronske mreže fiksne strukture i one dobivene NEAT algoritmom manje što je problem složeniji. Za primjer easy funkcije razlika u kvaliteti je velika dok se za hard funkciju ili funkciju zraka ona drastično smanjuje.

Jedan mogući razlog tomu leži upravo u kvaliteti implementacije. Naime neuronska mreža s fiksnom arhitekturom ima ograničenje upravo zbog svoje jednostavnosti. Ona za jednostavnije primjere vrlo dobro radi dok za one složenije postepeno gubi sposobnost davanja preciznih predikcija.

NEAT algoritam bi u teoriji trebao optimizirati i stvoriti složeniju arhitekturu neuronske mreže. Takva bi mreža tada trebala raditi predikcije za složenije probleme. No, bitna razlika između jednostavne fiksne arhitekture i NEAT algoritma leži upravo u težini implementacije. Prilikom ostvarivanja neuronske mreže NEAT algoritmom neupitno je više parametara koji se trebaju ispravno optimizirati te se iz tog razloga stvara veći prostor za pogreške.

Jedan mogući razlog za dobivena rješenja leži upravo u složenosti implementacije. Zbog nesavršenosti u implementaciji NEAT algoritma ta neuronska mreža kontinuirao daje nesavršena rješenja dok jednostavna neuronska mreža očekivano smanjuje preciznost predikcije ovisno o složenosti.

Naravno, velika je vjerojatnost da bi iznimno dobro implementiran NEAT algoritam radio bolje predikcije za složenije primjere.

U rezultatima se uočava jedna zanimljiva pojava prilikom rada neuronskih mreža na funkciji zraka. Naime, promatrajući funkciju dobrote može se uočiti kako smanjenje greške prilikom rada na funkciji zraka nije toliko veliko kao u slučaju rada nad primjerice *medium* funkcijom. Prilikom rada na *medium* funkciji, greška se smanji osamdeset puta koristeći fiksnu arhitekturu te deset puta koristeći varijabilnu arhitekturu. S druge strane, prilikom rada na funkciji zraka, neuronska mreža fiksne arhitekture smanjuje grešku samo dva puta dok ju neuronska mreža dobivena NEAT algoritmom potencijalno ne smanji uopće.

Osim toga, uspoređujući vizualne prikaze predikcija za *medium* funkciju s predikcijama funkcije zraka jasno se uočava razlika u preciznosti. Usporedimo li sliku 6.11 sa slikom 6.15 ili sliku 6.12 sa slikom 6.16 vrlo je jasna razlika u preciznosti predikcija.

No, temeljem brojčanih rezultata, neuronske mreže daju izuzetno niska odstupanja za funkciju zraka. Jedan mogući razlog tome je činjenica da inicijalne populacije neuronskih mreža fiksne i varijabilne arhitekture jednostavno daju bolja rješenja prilikom rada na funkciji zraka. Na slici 6.1 uočava se kako je greška inicijalne populacije za rad na *medium* funkciji 3 500 000, dok je ta greška za rad na funkciji zraka nešto niža od 80. Dakle inicijalne populacije prilikom rada na funkciji zraka, iako su nasumično generirane vrlo dobro daju predikcije.

Osim toga vrlo velik razlog prividne kvalitete neuronskih mreža za rješavanje funkcije zraka leži upravo u zanemarivanju relativnosti greške. Naime prosječno odstupanje prilikom rada s funkcijom zraka iznosi 3.3668 što zaista jest manje od prosječnog odstupanja prilikom rada s *medium* funkcijom koje iznosi 100.0205 . No, bitno je primijetiti kako se stvarni izlazi za funkciju zraka nalaze u intervalu između -2 i 30 dok se stvarni izlazi za *medium* funkciju nalaze u intervalu između 0 i 3750. Ako se prosječne pogreške stave u relativnu perspektivu uzevši u obzir moguće intervale uočava se da je ipak odstupanje funkcije zraka značajnije.

7. Zaključak

U zadnjih se nekoliko godina, zbog napretka u tehnologiji, razvila potreba za brzim i efikasnim načinima obrade podataka. Inspiracija za razne metode koje bi uspješno obrađivale velike količine podataka našla se u prirodi.

Umjetna neuronska mreža je skup međusobno povezanih jednostavnih procesnih elemenata, čija se funkcionalnost temelji na biološkom neuronu. Sama sposobnost mreže da izvršava zadatke sadržana je u snazi, odnosno teži veza između neurona. Težine se, postupkom učenja prilagođavaju i optimiziraju.

Postoje razne metode i postupci učenja mreža, a u ovom radu je obrađena metoda koja koristi genetski algoritam. Genetski algoritam je tehnika optimiziranja koja se temelji na evoluciji te se svodi na iterativno poboljšavanje populacije operatorima selekcije, križanja i mutacije.

U ovom je radu genetski algoritam prvo korišten za optimiziranje težina neuronske mreže čija je jednostavna arhitektura bila određena prije početka učenja. No, uzevši u obzir da kvaliteta mreže podosta ovisi i o arhitekturi neuronske mreže, u drugom se dijelu završnog rada genetski algoritam koristio za optimiziranje same arhitekture nove neuronske mreže.

Dobiveni rezultati prikazani su u radu te su primijećene neke razlike u kvalitetama predikcija dviju neuronskih mreža. Hipotetski bi jednostavna neuronska mreža trebala raditi vrlo dobro za jednostavne primjere, dok bi za složenije primjere njena efikasnost trebala padati. To podupiru i dobiveni rezultati. S druge strane, varijabilna neuronska mreža trebala bi optimizirati arhitekturu te omogućiti rad i na složenijim primjerima. No, rezultati dobiveni u ovom radu ne podupiru takvu pretpostavku. Razlog tomu, vrlo vjerojatno leži prvenstveno u težini implementacije, odnosno u napravljenim propustima.

Kao nastavak ovog rada bilo bi zanimljivo dorađivati implementaciju genetskog algoritma u svrhu optimiziranja neuronske mreže te dobivene rezultate koristiti za neka potencijalno korisna predviđanja.

LITERATURA

[1] Ž. Ujević Andrijić. Umjetne neuronske mreže. 2019.

[2] Bojana Dalbelo Bašić, Jan Šnajder, i MarkoČupić. Umjetne neuronske mreže, 2008. URL https://www.fer.unizg.hr/_download/repository/UmjetneNeuronskeMreze.pdf.

[3] Bojana Dalbelo Bašić i Jan Šnajder. Uvod u umjetnu inteligenciju, 2019. URL https://www.fer.unizg.hr/_download/repository/UI-1-Uvod.pdf.

[4] Jason Brownlee. How to choose an activation function for deep learning, 2021. URL https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/.

[5] Niklas Donges. Gradient descent: An introduction to 1 of machine learning's most popular algorithms, 2021. URL https://builtin.com/data-science/gradient-descent.

[6] Finn Eggers. Ai-neat, 2019. URL https://www.youtube.com/playlist?list=PLgomWLYGNl1fcL0o4exBShNeCC5tc6s9C.

[7] Andries P. Engelbrecht. *Computational Intelligence : An Introduction*. Wiley, 2007.

[8] Junxi Feng. Reconstruction of porous media from extremely limited information using conditional generative adversarial networks. 2019.

[9] Ahmed Gad. Artificial neural networks optimization using genetic algorithm with python, 2019. URL shorturl.at/mpCPW.

[10] Edgar Galvan i Peter Mooneyć. Neuroevolution in deep neural networks:current trends and future challenges. 2020.

[11] Hunter Heidenreich. Neat: An awesome approach to neuroevolution, 2019. URL shorturl.at/afrDV.

[12] Tin Krambeger, Bojan Nožica, Ivica Dodig, i Davor Cafuta. Pregled tehnologija u neuronskim mrežama. 2019.

[13] David Ernest Lynch, Richard Smith, i Bruce Allen Knight. *Learning management: transitioning teachers for national and international change*. Pearson Education, 2007.

[14] Paul Pauls. A primer on the fundamental concepts of neuroevolution, 2020. URL shorturl.at/alxMV.

[15] Victor Sim. Using genetic algorithms to train neural networks, 2020. URL shorturl.at/deguK.

[16] Kenneth O. Stanley i Risto Miikkulainen. Evolving neural networks through augmenting topologiess. 2002.

Primjena evolucijskih algoritama za razvoj arhitekture neuronskih mreža

Sažetak

U današnje vrijeme, neuronske mreže sve su veći predmet interesa. Oponašanjem funkcionalnosti i povezanosti neurona u mozgu, umjetne neuronske mreže mogu učiti te raditi predikcije. Razvijene su mnoge tehnike učenja, a u ovom je radu opisano učenje genetskim algoritmom, još poznato i kao neuroevolucija. Implementirane su dvije neuronske mreže. U prvoj je genetski algoritam korišten samo za optimizaciju težina, dok je u drugoj NEAT algoritam primijenjen za optimizaciju same arhitekture neuronske mreže. Te su neuronske mreže korištene za rješavanje problema regresije. Primjer rada neuronskih mreži prikazan je na četiri funkcije različite težine. Funkcije *easy*, *medium* i *hard* generirane su kao primjer funkcija različitih težina u kojima sigurno postoji ovisnost izlaza o ulazu dok za funkciju zraka ta povezanost nije jasna. Na samome kraju prikazani su dobiveni rezultati.

Ključne riječi: neuroevolucija, NEAT, optimizacija arhitekture, regresija.

Application of evolutionary algorithms for the design of neural network architectures

Abstract

Today, neural networks are an emerging topic of interest. By emulating the functions and connections of neurons in the human brain, artificial neural networks are able to learn and make predictions. While many different techniques can be used in the process of learning, this work focuses on using genetic algorithms. That technique is called neuroevolution. Two neural networks were implemented. The first one had a fixed structure where only the weights were optimized by the genetic algorithm. For the second one, genetic algorithm NEAT was used to improve the architecture. Neural networks were then used to solve regression problems by making predictions of four different functions. Easy, medium and hard functions are an example of functions where the connection between outputs and inputs is certain and clear. That is not the case with the air function. Results are shown in the final part of the work.

Keywords: neuroevolution, NEAT, architecture optimization, regression.