SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 171

Primjena genetičkog programiranja za simboličku regresiju

Leon Novački

Zagreb, srpanj 2021.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Zahvaljujem prijateljima s kojima sam proveo prekrasan dio života.

SADRŽAJ

1. Uvod

Simbolička regresija je problem pronalaska algebarskog izraza koji što bolje i jednostavnije opisuje nepoznatu funkciju. Važnost simboličke regresije je u mogućnosti interpretiranja dobivenog rješenja. Tradicionalno se problem simboličke regresije rješava genetskim programiranjem. Ovaj rad opisuje postupak genetskog programiranja, korištene strukture podataka i operacije.

Uz rad priložena je implementacija genetskog programiranja pisana u programskom jeziku Python, te dobiveni rezultati i uspješnost korištenja programa na odabranim primjerima.

2. Genetsko programiranje

Genetsko programiranje je podvrsta genetskog algoritma. Izvodi se evolucija jedinki kako bi se postigla što bolja dobrota, i to postupcima selekcije, križanja i mutacije. Specifičnost genetskog programiranja je to što su jedinke nad kojima algoritam radi stabla. Stablima se mogu prikazivati programi za izvođenje, i kao i u našem slučaju, algebarske izraze čije se vrijednosti izračunavaju.

2.1. Generacijski genetski algoritam

Generacijski genetski algoritam traži nova rješenja problemu kroz generacije. Jedna generacija je populacija rješenja u kojoj se svaka jedinka vrednuje po dobroti rješenja. Jedinke sljedeće generacije stvaramo križanjem roditelja prethodne generacije. Stvorene jedinke se mutiraju i postaju dio sljedeće generacije. Veličina populacije je uvijek jednaka, a algoritam traje sve dok ne nađemo nama prihvatljivo rješenje ili do određenog isteka trajanja algoritma.

Algorithm 1 Generacijski genetski algoritam

Parametri: pop_size **– veličina populacije,** num_gen **– broj generacija**
$P \leftarrow$ inicijaliziramo početnu populaciju
for num_gen **do**
  $P' \leftarrow$ prazna pomoćna populacija
  while $P' < P$ **do**
    $p_1, p_2 \leftarrow$ biramo roditelje selekcijom iz P
    $d \leftarrow$ stvaramo dijete križanjem p_1 i p_2 i mutiramo ga
    $P' \leftarrow$ dodajemo d u pomoćnu populaciju
  end while
  $P \leftarrow$ P' pomoćna populacija postaje glavna
end for

2.1.1. Elitizam

Elitizam je strategija u genetskom algoritmu i genetskom programiranju koja jednu ili više najboljih jedinki iz prethodne generacije nepromijenjene prenosi u sljedeću generaciju. Koristi se kako ne bi izgubili najbolje jedinke neke generacije te da najbolja jedinka generacije ne bude lošija od najbolje jedinke neke prethodne generacije.

2.2. Stabla - jedinke genetskog programiranja

U genetskom programiranju jedinke su stabla. Prednost stabla je fleksibilnost koju pružaju. Stabla mogu imati različite dubine, mogu se granati na proizvoljan broj grana i lako se rekurzivno računaju. Čvorovi stabla koji se koriste ovise o problemu, stoga je potrebno odabrati dobar skup čvorova stabla koji je dovoljan za rješenje problema.

Stabla mogu biti proizvoljne dubine, ali velika stabla je teško interpretirati, a upravo to nam treba u simboličkoj regresiji. Također, velika stabla zahtjevaju više vremena za evaluiranje i više memorije za čuvanje. Potrebno je izabrati maksimalnu dubinu stabla koja neće previše usporiti rad algoritma, ali će biti dovoljna za rješavanje problema. Zbog računalne opreme na kojoj se izvodi genetsko programiranje, maksimalna dubina stabla u ovom radu će biti 7, osim za jedan test koji je izveden na maksimalnoj dubini od 9.

2.2.1. Funkcijski skup

Funkcijski skup je skup čvorova koji čine sve unutarnje čvorove stabla. U ovom radu to su čvorovi koji čine operacije $+$, $-$, $*$, $/$. Funkcijski skup se ovisno o problemu može proširiti sa drugim čvorovima. Ako gledajući podatke predviđamo kružno gibanje, možemo dodati funkcije sin i cos u funkcijski skup. Pokazano je da genetskim programiranjem se mogu otkriti trigonometrijski identiteti kao [$\cos 2\text{x} = 1 - 2\sin^2 x$] (Koza et al., 1992)

Kod izbora funkcijskog skupa treba imati na umu domenu pojedinih funkcija. Dijeljenje nulom je ilegalno i srušiti će program pa se potrebno zaštiti od takvih situacija. Zaštićeno dijeljenje (*engl. protected divison*) je operacija koja prvo provjerava legalnost operacije. Ako je dijelitelj jednak nuli, operacija će vratiti 1, nevezano o vrijednosti dijeljenika.

Na izbor funkcijskog skupa utječu i drugi hiperparametri algoritma. Korištenje eksponencijalne funkcije u stablima dubine 10 mogu dovesti do prelijeva pri računanju pa nisu korišteni u ovom radu.

2.2.2. Terminalni skup

Terminalni skup čine čvorovi koji su listove rješenja. U simboličkoj regresiji potrebne su konstante, čvorovi koji vraćaju fiksnu vrijednost pohranjenu u čvor, i varijable koje možemo isčitati iz podataka nad kojima radimo. U ovom radu to su čvorovi konstanti i čvorovi varijable označeni sa x_i.

2.2.3. Izgradnja stabla

Kako bi smo mogli stvoriti početnu populaciju, potrebne su nam metode nasumične izgradnje stabla. Dvije često korištene metode za stvaranje stabla su *Full* i *Grow*. Nasumično izgrađena stabala također možemo koristiti kao roditelje u križanju za unos novog genetskog materijala u populaciju.

Full

Full metoda gradi stablo zadane dubine *d* nasumično birajuči čvorove stabla iz skupa funkcija za sve čvorove do $d-1$ dubine. Zatim, lišće stabla se gradi nasumično birajući čvorove iz skupa terminala. Sva stabla građena *full* metodom su potpuna i sama *full* metoda nam ne daju raznolikost u obliku stabla. Zbog toga se koristi zajedno sa *Grow* metodom.

Grow

Grow metoda gradi stablo maksimalne dubine *d* birajuči čvorove stabla iz skupa funkcija i iz skupa terminala. Ako se stablo izgradi do dubine $d-1$, tada se biraju čvorovi samo iz skupa terminala. Ovisno o funkcijskom i terminalnom skupu, nesrazmjer njihovih veličina može narušiti raznolikost oblika kakvu smo htjeli dobiti. U takvim slučajevima možemo promijeniti vjerojatnosti odabira pojedinih čvorova kako bi stvorili poželjnija stabla.

2.3. Inicijalizacija populacije

Pri započinjanju genetskog programiranja potrebno je početi od neke populacije. Jedinke u početnoj populaciji se najčešće generiraju nasumično. Za generiranje stabla korištene su metode *grow* i *full*.

Ramped half-and-half je metoda opisana u (Koza et al., 1992) koja koristi *grow* i *full* metode kako bi se inicijalizirala populacija stabla raznolikih dubina i oblika.

Algorithm 2 Ramped half-and-half

Ulaz: F – skup funkcija, T – skup terminala, d – maksimalna dubina, n – broj stabla po dubini

Izlaz: P – populacija

$P \leftarrow$ stvaramo praznu populaciju

$i \leftarrow 2$

while $i <= d$ **do**

  for $n/2$ **do**

    $P \leftarrow$ **Grow(**i**)**

    $P \leftarrow$ **Full(**i**)**

  end for

  $i \leftarrow i + 1$

end while

Iako je stvaranje nasumične početne populacije standard, moguće je u početnu populaciju ubaciti umjetno stvorena stabla. Svrha toga je iskorištavanje već poznatog znanja ili ubrzavanje pronalaska rješenja, i kao takvo se koristilo u igranju igara (Ferrer i Martin, 1995), financijskom modeliranju (Chen i Yeh, 1997) i kopresiji slika (Nordin i Banzhaf, 1996).

3. Križanje

Križanjem (*engl. crossover)* stvaramo nove programe od dvaju roditeljskih programa. U kontekstu simboličke regresije, stvaramo novo stablo kombinirajući čvorove roditeljskih stabla.

Svrha križanja u genetskom programiranju je pretraživanje prostora rješenja u bližoj okolici programa roditelja.

Standardan operator križanja u genetskom programiranju stabla jest zamjena podstabla dvaju roditelja. Nasumično se odabere podstablo prvog i drugog roditelja, i na mjesto podstabla prvog roditelja se umetne odabrano podstablo drugog roditelja, i obrnuto, na mjesto podstabla drugog roditelja se umetne odabrano podstablo prvog roditelja. Time dobivamo dva nova stabla.

Različite veličine izabranih podstabla dvaju roditelja mogu dovesti do povećanja veličine i napuhivanje stabla.

Operator križanja koji to izbjegava radi standardno križanje, ali ako dobiveni potomak ima veću od najveće dozvoljene dubine, operator vraća jednog nasumično odabranog roditelja (Koza et al., 1992).

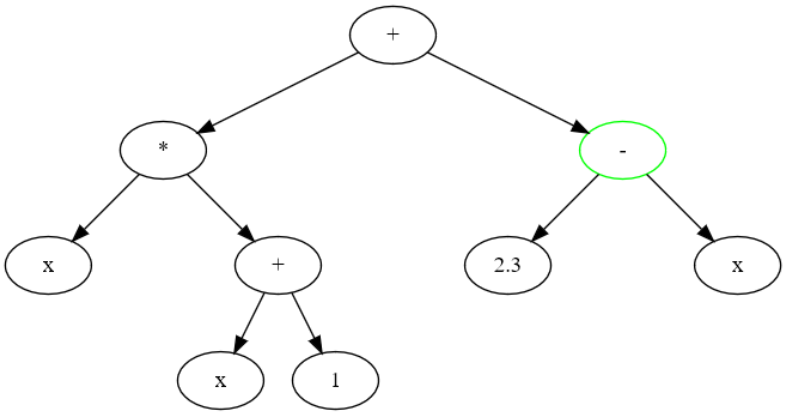
U genetskom programiranju često se za križanje osigurava da su čvorovi križanja u 90% slučajeva unutarnji čvorovi, a 10% listovi. Time se osigurava da dobivena stabla ne budu preslična roditeljima.

U ovom radu vjerojatnost odabira čvora je proporcionalna $x^{d_{max}-d}$ gdje je x broj između $[1, 2]$, a d odgovara dubini čvora. Za većinu problema u ovom radu vrijednost x je postavljena na 2. U potpunom binarnom stablu takav odabir čvorova je ekvivalentan nasumičnom odabiru dubine čvora, te onda nasumičnim odabiru čvora te dubine. Time osiguravamo da listovi nisu prečesto birani, ali i da unutarnji čvorovi blizu korijena budu češće birani naspram unutarnjih čvorova koji se nalaze tik do listova.

Kod problema gdje koristimo unarne sin i cos čvorove, vrijednost x je postavljena na 1.75 kako bi kompenzirali za manjak čvorova koji takva stabla imaju na većim dubinama.

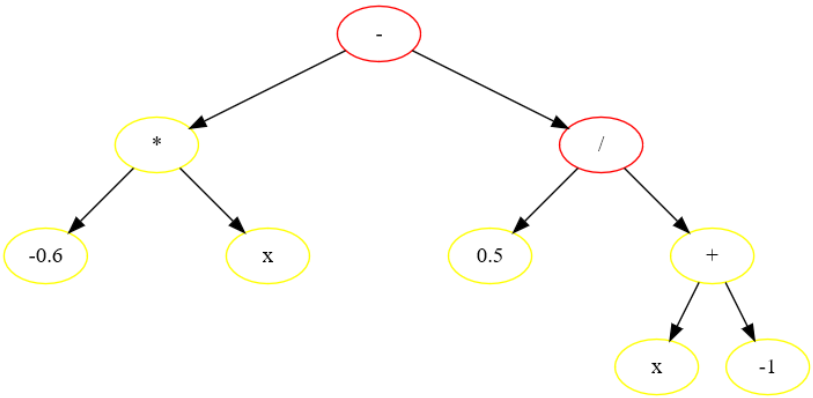
Nakon što je odabran čvor prvog roditelja, biramo čvor drugog roditelja iz skupa

čvorova čije umetanje ne bi povećalo dubinu stabla veću od dopuštene. Vjerojatnost odabira pojedinog čvora iz skupa se određuje na isti način kao i kod prvog roditelja.



Slika 3.1: Prvi roditelj

Slika 3.1 prikazuje stablo prvog roditelja dubine 3 i zelenom je označen čvor dubine 1 koji je odabran za križanje.



Slika 3.2: Drugi roditelj

Slika 3.2 prikazuje stablo drugog roditelja. Uz maksimalnu dubinu stabla 3, crveno su označeni čvorovi koji su ilegalni za umetanje u prvog roditelja, a žuto su označeni čvorovi koji mogu biti odabrani za križanje.

Postoje i drugi operatori križanja koji sprječavaju napuhivanje kao *size fair* križanje (Langdon, 2000).

4. Selekcija

Pri stvaranju potomaka za sljedeću generaciju, potrebno je odabrati roditelje koje ćemo križati. Operator selekcije određuje kako ćemo izabrati roditelje iz populacije trenutne generacije, i kroz njega želimo usmjeriti pretragu prema okolicama rješenja koja su se pokazala dobrima.

Time je očito da će se operator selekcija temeljiti na dobrotama rješenja u populaciji. Pri selekciji temeljnoj na vrijednosti dobrote, možemo naići na problem skale. Ako su pojedine

Odaberemo li prekritičan operator, populacija će prebrzo konvergirati prema najboljem rješenju u populaciji i izgubiti varijancu stabala. To povećava vjerojatnost da će naš genetski algoritam zapeti u lokalnom optimumu jer će se veliki postotak pretrage temeljiti na tom rješenju. To nije veliki problem u genetskom programiranju jer križanjem dvaju istih stabla nećemo dobiti isto stablo kao što se dobiva ista jedinka u genetskom algoritmu nad bitovima ili neuronskim mrežama. Međutim, time i dalje sužavamo prostor pretrage i takvo ponašanje treba uzeti u obzir.

4.1. Vrste selekcija

4.1.1. Proporcionalne selekcije

Proporcionalna selekcija (engl. *Roulette-wheel selection*) je operator selekcije kojemu je vjerojatnost da odabere jedinku proporcionalna dobroti te jedinke. Vjerojatnost odabira se može opisati formulom:

$$p = \frac{fit}{\sum fit}$$

Za minimizacijske probleme potrebno je koristiti funkciju koja će enkapsulirati funkciju dobrote kako proporcionalna seleckija ne bi postigla suprotno od željene svrhe. Jedna od takvih funkcija je oduzimanje dobrote od maksimalne dobrote.

$$g(x) = fit_{max} - fit(x)$$

Za to je potrebno znati maksimalnu vrijednost dobrote što nije uvijek lako, pa se može koristiti najbolja dobrota u trenutnoj populaciji.

Pokazano je da proporcionalna selekcija daje veliku varijancu u broju djece nastale na temelju određene jedinke (Engelbrecht, 2007). Povremeno se događa da najbolja jedinka u generaciji uopće ne sudjeluje u križanju i time gubimo najbolje rješenje. Takav problem se može izbjeći korištenjem elitizma, a predloženi su i druge selekcijske metode koje nemaju takvu manu kao turnirska selekcija i stohastička univerzalna selekcija (Grefenstette, 2013).

Relativne razlike u dobroti jedinka mogu uzrokovati velike ili gotovo nikakve razlike u vjerojatnosti selekcije, ovisno o skali na kojoj se dobrote nalaze. Promotrimo razdiobu vjerojatnosti odabira nad skupom jedinki s dobrotama 1, 2 i 3 i razdiobu vjerojatnosti odabira nad skupom jedinki s dobrotama 101, 102 i 103. U prvom skupu vjerojatnost odabira najbolje jedinke je 50%. U drugom skupu vjerojatnost odabira najbolje jedinke je 33.6%. Taj nedostatak proporcionalne selekcije naziva se *problemom skale*.

4.1.2. Turnirska selekcija

Turnirska selekcija je ona u kojoj se nasumično odabere *n* jedinki iz populacije, te se nakon toga bira najbolja jedinka unutar tog turnira na temelju dobrote. Mijenjanje parametra *n* je lako i omogućuje nam da biramo koliko snažan selekcijski pritisak vršimo. Za velike vrijednost *n* selekcijski pritisak je snažan i kroz generacije naše najbolje rješenje će brzo konvergirati. Ako izaberemo mali *n*, pritisak se smanjuje i umanjujemo vjerojatnost da algoritam konvergira u lokalni optimum. U praksi se koriste turnirske selekcije manjih veličina, primjerice 2 ili 3 (Čupić, 2016).

Algorithm 3 Turnirska selekcija

Ulaz: P – populacija, n – veličina turnira
Izlaz: T – izabrano stablo
P' – stvaramo praznu pomoćnu populaciju
for n **do**
    $P' \leftarrow$ nasumično odabrano stablo iz P
end for
sortiramo P' po dobroti
vraćamo najbolje stablo T iz P'

Za dobivanje dvaju roditelja potrebnih da se izvede operacija križanja, selekcijski postupak se izvodi dva puta, svaki put za jednog roditelja. Takva selekcija je korištena u implementaciji GP-a u ovom radu.

4.1.3. Selekcije po rangu

Za razliku od selekcija na temelju vrijednosti dobrote, mogu se koristiti i selekcije koje se temelje na rangu jedinki unutar populacije, sortirane po dobroti. Time izbjegavamo problem skale kakvu ima proporcionalna selekcija.

Vjerojatnost odabira jedinki možemo sami definirati, pa možemo napraviti selekciju linearnim rangiranjem, selekciju eksponencijalnim rangiranjem ili neku drugu.

4.1.4. Ostale selekcije

Postoje i razne druge selekcije koje neće biti opisane u ovom radu, kao Boltzmannova selekcija i (λ, μ) -selekcija.

5. Mutacija

Svrha operatora mutacije je da unese novi genetski materijal u populaciju. Mutacija omogućava da sva moguća rješenja ostanu u prostoru pretraživanja. Korištenje samo selekcije i križanja može dovesti do gubitka genetskog materijala, čime potencijalno gubimo dobra rješenja.

Mutacija se odvija pri stvaranju novih stabla za sljedeću generaciju, odmah nakon križanja. Svaki čvor nastalog stabla ima vjerojatnost p da bude mutiran.

S obzirom da je cilj križanja usmjeravanje pretrage u obećavajućem smjeru, vjerojatnost p ne smije biti prevelika kako ne bismo izgubili svojstva radi kojih smo križali dva roditelja. U ovom radu korištene vjerojatnosti mutacije su 0.05 i 0.1.

5.1. Vrste mutacija

5.1.1. Izmjena funkcijskog čvora

Algorithm 4 Izmjena funkcijskog čvora

Ulaz: F – skup funkcija, T – stablo koje mutiramo, p – vjerojatnost mutacije
for čvor x in T **do**
  if random() < p **then**
    $x \leftarrow$ nasumično odabran čvor iz F
  end if
end for

5.1.2. Križanje stabla s nasumično generiranim stablom

Generiramo novo stablo koristeći *full* ili *grow* metodu. Izvodimo križanje izabranog stabla s generiranim stablom. Ovakva mutacija je više destruktivna jer radimo veću promjenu nad originalnim stablom nego drugi operatori mutiranja. Destruktivne muta-

cije su korisne za iskakanje iz lokalnog optimuma i unošenje novog genetskog materijala u populaciju. Nije preporučeno koristiti samo ovakvu mutaciju jer velike promjene mogu otežati ili čak onemogućiti finu pretragu za pronalazak optimuma.

5.1.3. Mijenjanje poretka djece u funkcijskom čvoru

U odabranom unutarnjem čvoru zamijenimo mjesta lijevog i desnog dijeteta. Ovakva mutacija ne donosi veliku promjenu u populaciju, stoga sama po sebi nije dovoljna.

5.1.4. Izmjena terminalnog čvora

Stvaramo novi terminalni čvor birajući nasumično iz skupa terminalnih čvorova. Umećemo novi terminalni čvor na mjesto starog.

5.1.5. Gaussovsko mutiranje terminalnog čvora

Odabere se list koji ima neku konstantu vrijednost. Generira se nasumično odabrana vrijednost in $\mathcal{N}(0, \sigma^2)$. Dobivena vrijednost se pribroji konstanti koju list sadrži. Srednja vrijednost je stavljena na 0 kako bismo mogli dobiti i pozitivne i negativne promjene. Normalna distribucija je pogodna za pronalazak pojedinih konstanti u izrazu i kao takva se često koristi.

5.1.6. Orezivanje stabla

Odabire se unutarnji čvor u stablu i zamijeni se nasumično odabranim terminalnim čvorom. Služi za smanjenje veličine stabla i pogodna je za stabla koja ima prazne grane.

Prazne grane su one koje ne mijenjaju vrijednost stabla kada bi se izbacile, na primjer podstablo čiji je korijen množenje, a jedno od djeteta je list koji predstavlja konstantu 0.

5.2. Odabir mutacija

Moguće je kombinirati različite mutacije u genetskom programiranju. Svaki operator mutacija ima svoju svrhu i svakom operatoru možemo dati svoju vjerojatnost izvođenja. Pri odabira mutacija koje se koriste treba uzeti u obzir i kakve operatore selekcije i križanja se koriste.

U ovom radu koristi se izmjena funkcijskog čvora i izmjena terminalnog čvora. Mutacija se odvija nakon križanja i može se dogoditi na svakom čvoru stabla. Zbog odabira selekcije i križanja na stablima, populacija bi trebala sačuvati genetsku raznolikost i nisu potrebne destruktivne mutacije kao križanje stabla sa nasumično generiranim stablom.

Dinamičko mijenjanje mutacija

Mutacija je instrument kojime osiguravamo genetsku raznolikost i proširujemo prostor pretraživanja za rješenjem. Gubitkom genetske raznolikosti možemo zapeti u lokalnom optimumu i stagnirati kroz puno generacija. Radi sprječavanja takvih pojava stvorene su strategije koje mijenjaju hiperparametre mutacija kada genetski algoritam prestane napredovati ili kada je narušena genetska raznolikost populacije. Primjeri takvih promjena su povećanje standarne devijacije kod mutacija nad konstantama, povećanje učestalosti mutacije, i uključivanje destruktivnijih mutacija. Takve strategije nisu korištene u ovom radu.

6. Rezultati

U ovom radu koristimo genetsko programiranje za pronalazak simboličkog izraza koji najbolje opisuju dane podatke. Podatci nad kojima učimo su generirani jednoliko birajući 1000 točaka na intervalu domene funkcije. Podatci za testiranje rješenja su generirani na isti način. Radi testiranja uspješnosti programa, za svaku funkciju je program izvršen 30 puta.

6.1. Rezultati za funkciju $2x^4 - 0.5x^2 + 1$

Funkcija s jednom varijablom x domene $[-5, 5]$. Kroz nekoliko pokretanja smo prilagodili hiperparametre algoritma i sada prikazujemo dobivene rezultate algoritma kroz 300 generacija, veličine populacije 100, vjerojatnosti mutacije 0.1. Maksimalna dubina stabla je 7, a početna populacija generirana je *ramped half-and-half* metodom od dubine 2 do dubine 7. Procjenjujemo kvalitetu rješenja funkcijom kazne koja je srednja kvadratna pogreška (MSE) 1000 točaka iz skupa za testiranje.

Od 30 pokretanja 3 puta je dobiveno egzaktno rješenje s MSE 0 koji se razlikuju samo po veličini stabla, 17 rješenja imaju MSE ispod 1, 5 rješenja imaju MSE između 1 i 10, Ostalih 5 rješenja imaju MSE veći od 10, od kojih najveći ima MSE 10157. U tablici 6.1 prikazujemo aritmetičku sredinu MSE, standardnu devijaciju MSE, te min i max za 20 najuspješnih rješenja.

Tablica 6.1: Statistike MSE za funkciju $2x^4 - 0.5x^2 + 1$

$\overline{MSE}$	σ	min	max
0.21539	0.2957	0	0.96468

Interpretabilnost genetskog programiranja je važna, stoga u tablici 6.2 prikazujemo dobivene simboličke izraze nakon sređivanja i veličine stabla nekoliko najboljih rješenja i medijan.

Tablica 6.2: Dobiveni izrazi, njihove greške, i veličine stabla

rang	simbolički izraz	MSE	veličina
1	$2x^4 - 0.5x^2 + 1$	0	33
2	$2x^4 - 0.5x^2 + 1$	0	39
3	$2x^4 - 0.5x^2 + 1$	0	45
4	$2x^4 - 0.4547x^2 + 1$	0.009	33
15	$2x^4 - 0.5x^2 + 0.17372x + 1$	0.252	39

Sva 3 najbolja rješenja za $2x^4 - 0.5x^2 + 1$ nisu koristili niti jedan list koji sadrži konstantu vrijednost. $2x$ je pronađen pomoću $(x + x)/x$, $0.5x$ na sličan način, a 1 je pronađen dijeljenjem varijable x samom sa sobom.

6.2. Rezultati za funkciju $x_1^2 + 1.4x_1x_2 - 0.7x_2 + 2$

Korištenjem dviju varijabli prostor pretrage se znatno povećava, pa je i teže dobiti uspješno rješenje. U ovoj pretrazi x_1 i x_2 su domene $[-5, 5]$, maksimalno trajanje je 200 generacija, veličina populacije je 100, a vjerojatnost mutacije je 0.1. Maksimalna dubina stabla je 7, a početna populacija je inicijalizirana *ramped half-and-half* metodom.

Pojedina rješenja ponovno vrednujemo srednjim kvadratom pogreške. Od 30 pokretanja njih 8 ima MSE < 0.1, njih 13 ima 1 < MSE < 0.1, a preostalih 9 ima 1 < MSE < 6.

Tablica 6.3: Statistike MSE 8 najuspješnijih rješenja

μ	σ	min	max
0.0540	0.0300	0.0204	0.0982

U tablici 6.4 prikazujemo 3 najbolja rješenja i medijan 30 pokretanja programa.

Tablica 6.4: Dobiveni izrazi, njihove greške i veličine stabla

rang	simbolički izraz	MSE	veličina
1	$x_1^2 + 1.3826x_1x_2 - 0.7075x_2 + 2.001$	0.0204	43
2	$0.3815x_2/x_1 + 0.31525x_2 + x_1 + 2.001$	0.0252	35
3	$x_1^2 + 1.4161x_1x_2 - 0.6834x_2 + 1.873$	0.0365	31
15	$x_1^2 + 1.4543x_1x_2 - 0.7271x_2 + 2.102$	0.2406	31

6.3. Rezultati za funkciju $x_1x_2^2/x_3 + x_1x_2/x_3 + x_1$

Domena x_1 i x_2 je i dalje $[-5, 5]$, a za x_3 je $[-1, 1]$. Izvodi se 300 generacija, vjerojatnost mutacije je 0.1, veličina populacije je 100. Maksimalna duljina stabla je 6, inicijalizacija početne populacije je *ramped half-and-half* metodom. Trajanje 30 izvođenja iznosi 1 sat i 57 minuta.

17 od 30 rezultata ovog testa imaju MSE manji od 10^{-22}, dok svih ostalih 13 ima MSE veći od 10^5, a najveći od njih ima MSE 13512813296. Tako velika razlika između tih rješenja je prenaučenost. Od tih 13 loših rezultata, njih 6 ima MSE manji od 1 na skupu za učenje, a veći od 1000 na skupu za testiranje. Štoviše, prosječna razlika reda veličine funkcije kazne dobivena na skupu za učenje i skupu za testiranje iznosi 3.6 na svih 30 rješenja. Sama prenaučenost se događala i na najboljim rješenjima, gdje je simbolički izraz isti, ali MSE se razlikuje i do 4 reda veličine. Takve razlike su prikazane u tablici 6.5, na temelju koje se može usporediti dobivene MSE vrijednosti za najbolje, Q1, Q2, Q3 i najlošije rješenje.

Tablica 6.5: 5 rješenja za funkciju $x_1x_2^2/x_3 + x_1x_2/x_3 + x_1$

rang	MSE_{test}	MSE_{train}	simbolički izraz	veličina
1	$1.7 * 10^{-27}$	$3.8 * 10^{-29}$	$x_1x_2^2/x_3 + x_1x_2/x_3 + x_1$	19
8	$1.5 * 10^{-26}$	$6.8 * 10^{-26}$	$x_2x_2^2/x_3 + x_1x_2/x_3 + x_1$	19
15	$2.2 * 10^{-25}$	$7.8 * 10^{-25}$	$x_1x_2^2/x_3 + x_1x_2/x_3 + x_1$	15
22	943331	0.32	$x_1x_2^2/x_3 + x_1x_2 + x1$	17
30	$1.3 * 10^{10}$	1.9	$2x_2 * (x_1x_2 + x_1 + 1) + x_1$	27

Ovaj eksperiment je dobar pokazatelj korisnosti terminiranja pretrage ako je pronađeno dovoljno dobro rješenje. Razlika reda veličine MSE postoji u 15 od 17 najboljih rješenja, što znači da je genetsko programiranje optimiziralo redoslijed operacija koje Python treba napraviti kako bi imao što manju *floating-point* grešku pri računanju na skupu za učenje.

Za sprječavanje prenaučenosti potrebno je koristiti više točaka na skupu za učenje jer povećanjem broja varijabli se povećava domena funkcije koju tražimo.

6.4. Rezultati za funkciju $x_2x_4^3 + x_1x_2x_3x_4 + x_1x_3/x_2 - x_1x_2 + x_3$

Proširen je broj korištenih varijabli na 4. x_1 i x_2 su i dalje domene $[-5, 5]$, dok x_3 je domene [-10, 10], a x_4 domene [-1,1]. Broj generacija u jednom izvođenju je 300, vjerojatnost mutacije 0.1, veličina populacije 100, početna inicijalizirana *ramped-half-and-half* metodom. Maksimalna dubina stabla je 7. Trajanje 30 izvođenja su 2 sata i 9 minuta.

Traženje ove funkcije pokazuje granice izvođenja GP-a sa ovakvim hiperparametrima. Od 30 izvođenja samo 5 rješenja ima MSE manji od 2, ostala rješenja nisu korisna.

Tablica 6.6: 5 najboljih rješenja za funkciju $x_2x_4^3 + x_1x_2x_3x_4 + x_1x_3/x_2 - x_1x_2 + x_3$ s maksimalnom dubinom stabla 7

rang	simbolički izraz	MSE	veličina
1	$x_2x_4 + x_1x_2x_3x_4 + x_1x_3/x_2 - x_1x_2 + x_3$	0.599	29
2	$x_1x_2x_3x_4 + x_1x_3/x_2 - x_1x_2 + x_3 - 0.05x_1 - 0.05$	1.069	29
3	$x_1x_2x_3x_4 + x_1x_3/x_2 - x_1x_2 + x_3$	1.107	27
4	$x_1x_2x_3x_4 + x_1x_3/x_2 - x_1x_2 + x_3$	1.157	23
5	$x_1x_2x_3x_4 + x_1x_3/x_2 - x_1x_2 + x_3$	1.267	25

Gledanjem dobivenih simboličkih izraza u Tablici 6.6 vidi se da je GP-u teško pronaći $x_2x_4^3$. Ostali dijelovi funkcije $x_1x_2x_3x_4$, x_1x_3/x_2, x_1x_2, x_3 jednostavniji su za pronaći jer su međusobno sličniji. Kada GP pronađe rješenje koje sadrži $x_1 * x_3$, onda isto to podstablo može ponovno doći do njega kada se križa sa nekim stablom čiji je on bio roditelj ili predak, pa od istog podstabla može sagraditi i x_1x_3/x_2 i $x_1x_2x_3x_4$.

Izraz $x_2x_4^3$ dijeli x_2x_4 sa izrazom $x_1x_2x_3x_4$, i taj dio može izgraditi, kao što je izgrađen u najboljem rješenju Tablice 6.6, ali skok do $x_2x_4^3$ je velik. Postepen prijelaz $(x_2x_4) * x_4$ teško da će preživjeti jer daje lošije rezultate od samog x_2x_4 zbog domene x_4. Na donjoj polovici $[-1, 0]$, kvadriranje mijenja negativni predznak x_4 u pozitivni i stvara grešku. Također, funkcija nigdje ne sadrži x_4^2 kako bi mogla odvojeno izgraditi taj izraz i križanjem ga cijelog dodati na $*(x_2x_4)$.

Ovaj eksperiment je ponovljen ali sa povećanom maksimalnom dubinom stabla 9. Populacija je inicijalizirana *ramped-half-and-half* metodom do dubine 9 i sa smanjenom veličinom populacije na 70. Dobiveni rezultati su slični prethodnom testu opisanom u Tablici 6.6. Također je samo 5 rezultata sa MSE manjim od 2, a ostali su

beskorisni.

Tablica 6.7: 5 najboljih rješenja za funkciju $x_2x_4^3 + x_1x_2x_3x_4 + x_1x_3/x_2 - x_1x_2 + x_3$ s maksimalnom dubinom stabla 9

rang	simbolički izraz	MSE	veličina
1	$x_1x_2x_4 - x_1x_4 + x_1x_2x_3x_4 + x_1x_3/x_2 + -x_1x_2 + x_3$	0.982	31
2	$x_1x_2x_3x_4 + x_1x_3/x_2 - x_1x_2 + x_3$	1.119	27
3	$x_1x_2x_3x_4 + x_1x_3/x_2 - x_1x_2 + x_3$	1.107	19
4	$x_1x_2x_3x_4 + x_1x_3/x_2 - x_1x_2 + x_3$	1.157	17
5	$x_1x_2x_3x_4 + x_1x_3/x_2 - x_1x_2 + x_3$	1.267	33

Broj čvorova u najboljih stablima oba eksperimenta je podjednak, samo što u drugom eksperimentu 1. i 5. rješenje u tablici 6.7 imaju izduženiji oblik naspram ostalih stabla.

Veća dopuštena dubina nije dovela do boljih rješenja, nego samo do vremenskog troška. I uz smanjenje veličine populacije za 30%, vrijeme izvođenja iznosi 2 sata i 27 minuta, što je povećanje od 15.7%.

Male veličine stabala prvog i drugog eksperimenta upućuje na to da korištena implementacija križanja i selekcije može uzrokovati smanjenje veličina stabala u populaciji.

6.5. Rezultati za funkciju $x_1/0.8 * sin(x_2) + cos(x_1/2) + x_1 * x_2$

Ovaj eksperiment prikazuje rezultate GP-a u pronalasku funkcije koja sadrži trigonometrijske funkcije. Skup funkcijskih čvorova je proširen sa sin i cos čvorovima. Domena x_1 je $[-5, 5]$ a domena x_2 je $[0, 2\pi]$. Izvodimo 300 generacija, populacija sadrži 100 jedinki, maksimalna dubina stabla je 7. Vjerojatnost mutacije je 0.1.

Od 30 rješenja njih 16 ima MSE manji od 1. Tablica 6.8 prikazuje aritmetičku sredinu MSE, standardnu devijaciju MSE, medijan MSE, te MSE najboljeg i najlošijeg rješenja.

Najlošiji rezultat je velika stršeća vrijednost te medijan bolje opisuje centralnu tendenciju dobivenih rješenja.

Tablica 6.9 prikazuje 5 najboljih dobivenih rješenja, njihove greške i veličine stabala

Tablica 6.8: Statistike MSE za funkciju $x_1/0.8 * sin(x_2) + cos(x_1/2) + x_1 * x_2$

$\overline{MSE}$	σ	medijan	min	max
6.196	20.78	0.5854	0.0086	114.248

Tablica 6.9: 5 najboljih rješenja za funkciju $x_1/0.8 * sin(x_2) + cos(x_1/2) + x_1 * x_2$

rang	MSE	veličina
1	0.0086	20
2	0.063	22
3	0.195	21
4	0.224	22
5	0.236	20

Sama točnost dobivenih rješenja je dobra, ali interpretabilnost varira od rješenja do rješenja. Najbolje rješenje se lako interpretira i ima formulu $1.294x_1sin(x_2) + cos(0.492997x_1) + x_1x_2$. Simbolički izraz drugog rješenja je $1.403x_1sin(sin(x_2)) + x_1x_2 + cos(x_1 * sin(cos(cos(x_1)))) + 0.166221$. Takav izraz se teško interpretira i unatoč maloj greški koju stvara, takvo rješenje nam nije toliko korisno.

Da bi se poboljšala korisnost dobivenih rješenja, bilo bi potrebno uvesti mehanizam pojednostavljanja stabala u genetskom programiranju koje prelazi opseg ovog rada.

7. Zaključak

Genetsko programiranje je često korišten algoritam za problem simboličke regresije. Ovaj rad objašnjava način rada genetskog programiranja. Također, opisuju se operatori selekcije, križanja i mutacije, te opisuju se često korištene implementacije tih operatora.

Genetsko programiranje u ovom radu implementirano je u programskom jeziku Python. Prikazani su i analizirani rezultati programa pri traženju 5 različitih funkcija. Prikazani su problemi na kojima GP radi dobro i daje kvalitetna rješenja i sa kakvim problemima GP ima poteškoća te zašto te poteškoće otežavaju rad GP-a.

LITERATURA

Shu-Heng Chen i Chia-Hsuan Yeh. Using genetic programming to model volatility in financial time series. U *Genetic Programming 1997: Proceedings of the Second Annual Conference*, stranice 288–306. Morgan Kaufmann, 1997.

Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley Publishing, 2nd izdanju, 2007. ISBN 0470035617.

Gabriel Ferrer i W. Martin. Using genetic programming to evolve board evaluation functions. 11 1995. doi: 10.1109/ICEC.1995.487479.

J.J. Grefenstette. *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*. Taylor & Francis, 2013. ISBN 9781134989737. URL `https://books.google.hr/books?id=MYJ\_AAAAQBAJ`.

J.R. Koza, J.R. Koza, i J.P. Rice. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. A Bradford book. Bradford, 1992. ISBN 9780262111706. URL `https://books.google.hr/books?id=Bhtxo60BV0EC`.

W. Langdon. Size fair and homologous tree crossovers for tree genetic programming. *Genetic Programming and Evolvable Machines*, 1:95–119, 01 2000. doi: 10.1023/A:1010024515191.

Peter Nordin i Wolfgang Banzhaf. Programmatic compression of images and sound. U *Genetic Programming 1996: Proceedings of the First Annual Conference*, stranice 345–350. MIT Press, 1996.

Marko Čupić. *Umjetna inteligencija - Evolucijsko računarstvo*, 2016.

POPIS SLIKA

Primjena genetičkog programiranja za simboličku regresiju

Sažetak

Ovaj rad opisuje problem simboličke regresije. Objašnjava genetsko programiranje i opisuje njegove operatore. Sadrži implementaciju genetskog programiranja u programskom jeziku Python te rezultate pri korištenju njega na sintetskim podatcima.

Ključne riječi: genetsko programiranje, genetski algoritam, simbolička regresija.

Application of genetic programming for symbolic regression

Abstract

This thesis describes the problem of symbolic regression. It contains a description of genetic programming and it's main operations. Contains an implementation in the programming language Python and it's results on synthetic data.

Keywords: genetic programming, genetic algorithm, symbolic regression