SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTA

ZAVRŠNI RAD br. 6388

Rješavanje klasifikacijskih problema uz pomoć kartezijskog genetskog programiranja

Ivica Duspara

Zagreb, lipanj 2019.

Zahvaljujem se mentoru, prof. dr. sc. Domagoju Jakoboviću, za ideje, savjete i pomoć pri izradi ovog završnog rada.

Također se zahvaljujem svojim roditeljima, baki i djedu na svom vremenu, strpljenju i podršci tijekom školovanja.

Sadržaj

1. Uvod

Iz ideje da se omogući računalima da uče i razmišljaju na sličan način kao i ljudi razvilo se područje umjetne inteligencije, a iz toga i grana strojnog učenja. Jedan od problema koje strojno učenje rješava je klasifikacija - svrstavanje primjeraka u klase temeljem značajki tog primjerka.

U radu će se opisati vrste strojnog učenja, problemi koji se javljaju u strojnom učenju te klasifikacija. Nakon toga se opisuje genetsko programiranje, kartezijsko genetsko programiranje te mehanizmi odabira, križanja i mutacije. Nakon toga dan je opis implementacije, parametara te podržanih operatora. Konačno prikazuju se rezultati te usporedba uspjeha s drugim metodama genetskog programiranja.[5] [6]

2. Strojno učenje

U ovom poglavlju bit će dan kratki pregled strojnog učenja i problema koje strojno učenje rješava. Nakon toga ćemo objasniti vrste strojnog učenja, pogledati škole strojnog učenja te konačno o izbjegavanju problema prenaučenosti.

2.1 Definicija

Strojno učenje *(engl. machine learning)* je grana računarske znanosti koja proučava algoritme i statističke modele koje računalo koristi za rješavanje specifičnih problema bez da je eksplicitno programirano za rješavanje tog problema, već se oslanja na uzorke i zaključivanje. To omogućuje računalnom programu da temeljem viđenih podataka može predvidjeti svojstva novih, još neviđenih podataka. Sa stajališta logike, računalni program zapravo radi indukciju.

Definicija 2.1 *Kažemo da računalni program uči iz iskustva E obzirom na zadatke T i mjerom sposobnosti P ako njegov učinak u izvođenju zadataka T mjeren mjerom P povećava iskustvo E*

Zamislimo da imamo zadatak napisati program koji pretvara ljudski govor u tekst. Različiti ljudi iste riječi izgovaraju drukčije zbog razlika u spolu, godinama i naglascima. Strojno učenje pristupa ovom problemu tako da uzme veliki uzorak izgovorenih riječi te nauči povezati izgovorenu riječ s napisanom riječi.
U prethodnom primjeru:

- iskustvo E: prethodno snimljene riječi
- zadatak T: nauči povezivati izgovorene riječi s napisanim riječima
- mjera sposobnosti P: postotak točno određenih riječi

2.2 Vrste strojnog učenja

Najčešća podjela vrsta strojnog učenja jest na tri velike cjeline:

1. nadzirano učenje *(engl. supervised learning)*
2. nenadzirano učenje *(engl. unsupervised learning)*
3. podržano učenje *(engl. reinforcement learning)*

2.2.1 Nadzirano učenje

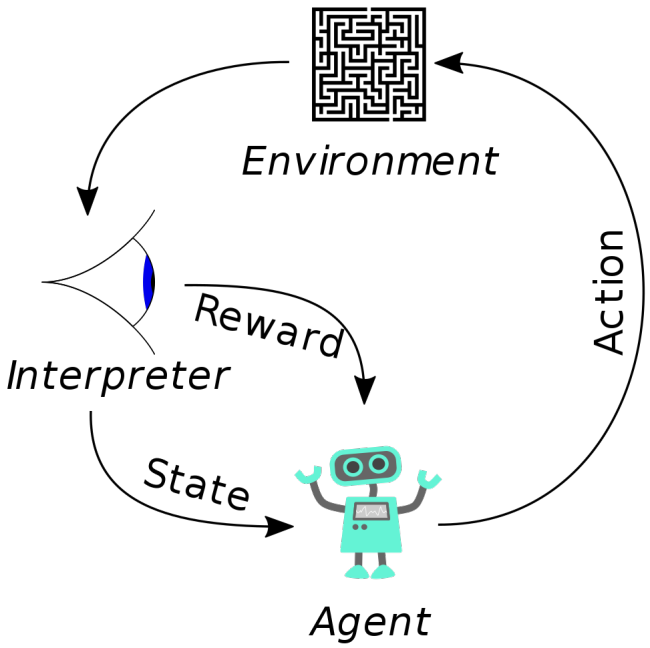
Neka je $\mathbf{x} = (x_1, x_2, ..., x_n)$ skup ulaznih podataka, a $\mathbf{y} = (y_1, y_2, ..., y_n)$ skup odgovarajućih izlaznih podataka. Nadzirano učenje je oblik učenja koje uči na skupu podataka za učenje (koji se sastoji od parova (ulaz, izlaz)=($\mathbf{x}$, $\mathbf{y}$)).
Rezultat učenja je funkcija $f(\mathbf{x})$ koja je u stanju točno odrediti odgovarajuće izlaze za nove primjere ulaznih podataka.

2.2.2 Nenadzirano učenje

Nenadzirano učenje je oblik učenja koji uči na skupu podataka za učenje koji se sastoji samo od primjera ulaznih podataka (bez odgovarajućih izlaznih podataka, odnosno nad podatcima koji nisu klasificirani ili kategorizirani na neki način). Grupiranjem *(engl. clustering)*, otkrivanjem novih/stršećih vrijednosti *(engl. outlier/novelty detection)*, smanjenjem dimenzionalnosti *(engl. dimensionality reduction)* model bi trebao moći neke nove podatke grupirati ispravno.

2.2.3 Podržano učenje

Podržano učenje je oblik učenja koji se znatnije razlikuje od nadziranog učenja i nenadziranog učenja. Podržano učenje ne traži parove (ulaz, izlaz) niti kategorizirane ili klasificirane podatke na ulazu također sub-optimalne akcije ne moraju biti eksplicitno popravljene.
Način na koji podržano učenje radi je da se računalni program *(engl. agent)* nalazi u nekom okolišu *(engl. environment)*. Svakom akcijom koju računalni program donese, okoliš interpretira tu akciju te daje povratnu informaciju *(engl. feedback)* koja može biti pozitivna(nagrada), negativna(kazna) ili neutralna. Temeljem te povratne informacije računalni program se uči ponašati u toj okolini te nastoji maksimizirati vrijednost koju dobiva od svojih akcija.



Slika 1. Prikaz ciklusa potpornog učenja

2.3 Škole strojnog učenja

Podjela strojnog učenja na pet "škola" ili "plemena" u velikoj mjeri je popularizirano knjigom *"The Master Algorithm", (Domingos, 2015.)*. Svaka škola je različita po svojim temeljima i algoritmima koje koristi za učenje.

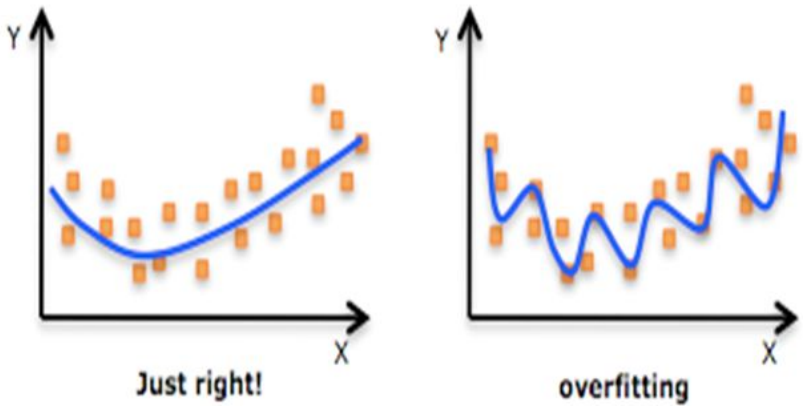
Škola	**Temelji**	**Glavni algoritmi**
Simbolists	logika, filozofija	Indukcija
Connectionists	neuroznanost	backpropagation
Evolutionaries	evolucijska biologija	genetsko programiranje
Bayesians	statistika	probabilističko zaključivanje
Analogizers	psihologija	jezgreni strojevi

Tablica 1. Pet "škola" strojnog učenja

Problemi klasifikacije biti će rješavani nadziranim učenjem i genetičkim programiranjem.

2.4 Izbjegavanje problema prenaučenosti

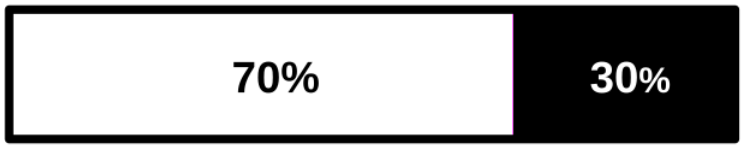
Model nadziranog učenja najprije treba učiti na skupu podataka za učenje. Cilj tog učenja je da model nakon učenja bude u stanju točno odrediti izlaz podataka koje nikada nije vidio. Vrlo složen model se može dobro prilagoditi podatcima i davati odlična predviđanja. Presložen model previše će se prilagoditi podatcima na kojima je treniran, a davat će loša predviđanja na novim podatcima. Česti uzroci prenaučenosti su dodavanje previše neobveznih parametara, učenja nad rijetkim podatcima koje uzrokuje prilagodbu modela na vrlo specifične značajke



Slika 2. Prikaz problema prenaučenosti

Jedna od metoda za izbjegavanje problema prenaučenosti je unakrsna provjera *(engl. cross-validation)*. Budući da neviđeni primjeri nisu dostupni skup podataka za učenje podjelimo na dva podskupa: skup za učenje *(engl. training*

set) i skup za ispitivanje *(engl. test set)*. Za vrijeme učenja model radi samo sa skupom za učenje. Nakon toga predviđanje radimo na skupu za ispitivanje te ondje računamo točnost. Točnost modela upućuje na modelovu sposobnost generalizacije.



Slika 3. Tipična podjela skupa za učenje: 70%-30%

Neki modeli omogućavaju da ugađamo složenosti modela promjenom određenih parametara. Kako bismo odredili optimalnu složenost modela trebamo dodatan skup pri podjeli skupa podataka za učenje, skup za provjeru *(engl. validation set)*. Modele različitih složenosti treniramo na skupu za učenje. Nakon toga svaki ispitujemo na skupu za provjeru i odabiremo optimalni koji testiramo na skupu za ispitivanje.

Slika 4. Tipična podjela skupa za učenje: 40%-30%-30%

2.4.1 Klasifikacija

Klasifikacija je primjer raspoznavanja uzorka koji rješava problem svrstavanja slučajeva u neku od postojećih kategorija. Slučajeve klasificiramo prema njihovim značajkama koje mogu biti numeričke (broj neke riječi u mailu, broj kišnih dana u mjesecu, krvni tlak) i kategoričke (krvna grupa, boja kose). Kategoričke značajke kodiramo u numeričke. Klasifikator koji svrstava slučajeve u dvije klase nazivamo binarni klasifikator *(engl. binary classificator)*, a klasifikatore koje svrstavaju u više klasa nazivamo višeklasnim klasifikatorima *(engl. multiclass classificator)*. Ovisno o prirodi problema, različiti klasifikatori imaju različit uspijeh. Nema klasifikatora koji je univerzalno najbolji.

3. Genetsko programiranje

U ovom poglavlju bit će objašnjeno genetsko programiranje. Objasnit će se kartezijsko genetsko programiranje i problemi za koje je ono pogodno.

3.1 Genetsko programiranje

Genetsko programiranje *(engl. genetic programming(GP))* je tehnika evoluiranja jedinki (programa), koje su početno nesposobne za izvršavanje određenog zadatka primijenjujući mehanizme analogne onima korištenim u prirodi. Genetsko programiranje u srži se može svesti na pretragu optimalnog ili suboptimalnog programa u prostoru svih programa.

Mehanizmi korišteni u evoluciji jedinki su odabir *(engl. selection)*, križanje *(engl. crossover)* i mutacija *(engl. mutation)*. Primjenom mehanizama i funkcije dobrote *(engl. fitness function)* na početnu populaciju jedinki *(engl. initial population)* može se doći do raznolikog broja jedinki i približiti se optimalnom ili suboptimalnom programu.

3.2 Mjera dobrote

U genetskom programiranju, nakon svake iteracije algoritma potrebno je obrisati broj jedinki koje se smatraju najlošijima, odabrati broj jedinki koje se smatraju najboljima te iz njih stvoriti novu generaciju jedinki koje će se dalje evaluirati. Funkcija dobrote treba dobro korelirati s postavljenim ciljem jer u suprotnom će algoritam konvergirati ka neprikladnom rješenju ili će imati poteškoće konvergirati.

Funkcija dobrote također treba biti brzo izračunljiva obzirom da tipični genetski algoritam vrši puno iteracija nad puno jedinki kako bi proizveo prihvatljiv rezultat. U određenim situacijama prihvaća se aproksimacija funkcije dobrote *(engl. fitness approximation)*:

- Izračun funkcije dobrote traje previše drugo
- Postoji šum ili nedostaje informacija za izračun funkcije dobrote

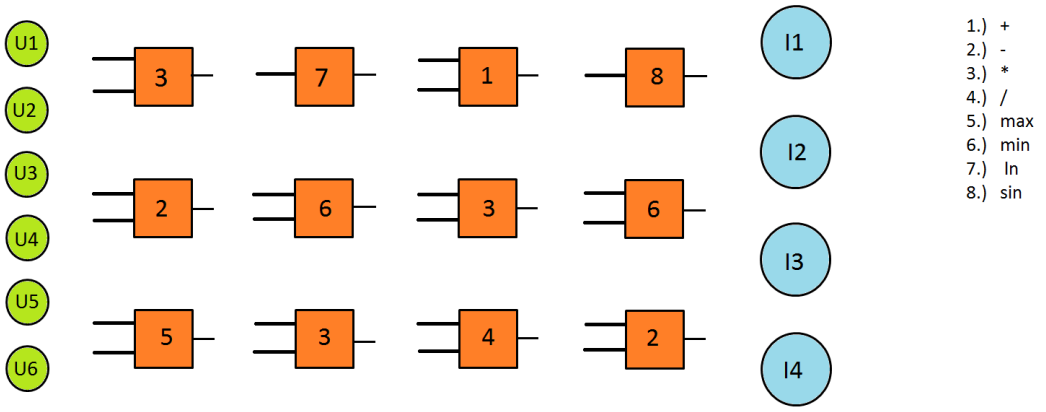
Pogledajmo jedan jednostavan primjer funkcije dobrote. Neka je cilj zadani stih *("There's a feeling I get when I look to the west")*. Generirani stih glasi: *("Theqa'p b viqling I get vonn I okou bn aal hcst")*. Funkcija dobrote se može prikazati kao:

$$\frac{t}{N}$$

gdje je **N** broj slova u točnom stihu, a **t** broj slova koja su točna u generiranom stihu. Ovdje bi to iznosilo $\frac{24}{47} \approx 0.5107$

3.3 Kartezijsko genetsko programiranje

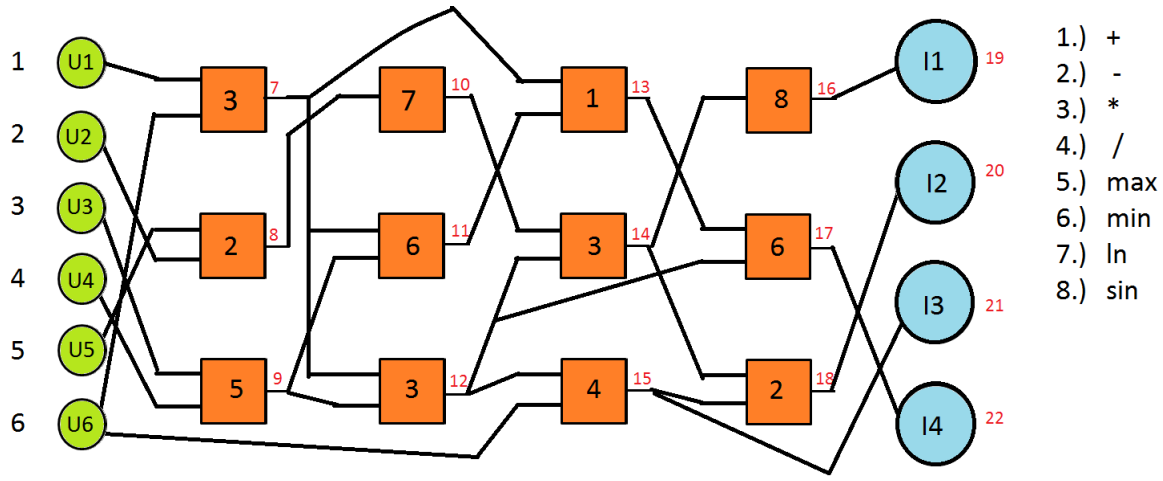
Izraz "kartezijsko genetsko programiranje" prvi put se pojavljuje 1999. godine te je predložen kao oblik genetskog programiranja 2000. godine. Naziva se "kartezijsko" jer predstavlja program koristeći 2D mrežu čvorova. Ta 2D mreža predstavlja usmjereni aciklički graf čiji čvorovi predstavljaju gene. Geni u sebi imaju zapisano koju operaciju obavljaju na svojim ulazima te odakle uzimaju ulaze nad kojim se obavlja zadana operacija. Postoje čvorovi koji predstavljaju ulaze *(engl. inputs)* koji imaju konstantnu brojčanu vrijednost ili brojčanu vrijednost koja je nastala kao izlaz funkcije koja preslikava neku značajku u broj. Konačno postoje čvorovi koji predstavljaju izlaze *(engl. outputs)* - rezultate evaluacije koji nastaju "provlačenjem" ulaza kroz mrežu.



Slika 5. Mreža bez prisutnih veza

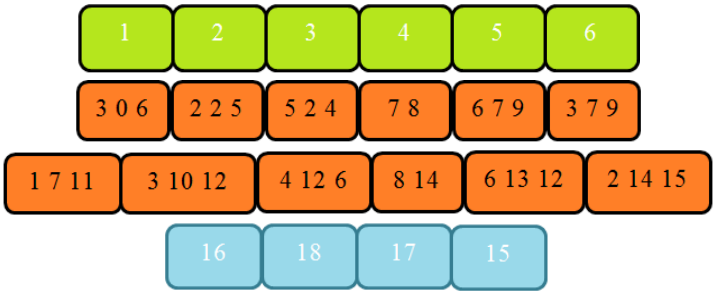
Kao što je prikazano na *slici 5.* svaki gen ima onoliko ulaza koliko uzima operator. Tako će gen koji zbraja na ulazu uzimati 2 operanda, dok gen koji računa sinus će uzimati jedan operand. Nije dozvoljeno spajati gene koji se nalaze u istom stupcu te nije dozvoljeno spajati ulaze gena sa izlazima koji se nalaze u stupcu desnije od stupca tog gena.

Osim broja redaka i stupaca za opis mreže, također se koristi i parametar povezivanja unatrag *(engl. levelsback)*. Povezivanje u nazad govori koliko "daleko" se smiju koristiti izlazi kao ulazi nekog gena. Primjerice, ako je parametar povezivanja u nazad 2, onda geni trećeg stupca za svoje ulaze mogu koristiti izlaze iz drugog stupca i prvog stupca. Gene koji ne sudjeluju (direktno ili indirektno) u izračunu vrijednosti nekog izlaznog čvora nazivamo nekodirajućim *(engl. non-coding)*. U mrežama većih veličina kada se radi dekodiranje veliki broj gena se može ignorirati jer ih nitko ne referencira od ulaza do izlaza.



Slika 6. Primjer povezane mreže

Na *slici 6.* nalazi se mreža dimenzija 3x4, sa 6 ulaza i 4 izlaza. Crvenim znamenkama označeni su izlazi iz gena. Razlog ovomu je računalni zapis mreže. Najčešće 2D mrežu zapisujemo u nizu cijelih brojeva tako da se prvo zapišu ulazi, geni i izlazi. Ulazi se zapisuju tako što im se direktno zapišu vrijednosti, geni se zapisuju kao n-torke čiji je prvi broj naznaka operatora, a ostali članovi su operandi nad kojima se računa. Izlazi su zapisani tako da im brojevi referenciraju neki indeks niza (ulaz ili gen) čije se vrijednosti dovode na izlaz.



Slika 7. Primjer zapisa mreže sa *slike 6.*

Na *slici 7.* prikazan je zapis mreže *slike 6.* kao niza. Zeleno obojeni članovi su ulazi u mrežu čije su vrijednosti upisane u pravokutnike. Narančasto obojeni članovi su geni. Svaki gen dekodira se na način da se prvo pogleda kojem operatoru odgovara prvi broj. Ostali brojevi predstavljaju izlaze koji se uvrštavaju u operator. Tako će se gen **3 0 6** dekodirati u operaciju množenja(**3**) a ulaz će biti U1(**1**) i U6(**6**). Rezultat množenja iznosi (**6**) i ta vrijednost postaje vrijednost izlaza prvog gena (označeno sa **7**). Pojedini izlazi gena odgovaraju indeksima tih gena u zapisanom nizu. Izlazni čvorovi označeni su plavom bojom, a njihova vrijednost je indeks gena čiju vrijednost prikazuju.

3.4 Mehanizmi evolucije

U ovom potpoglavlju se detaljnije upoznajemo s mehanizmima evolucije i primjerima mehanizama.

3.4.1 Odabir

Odabir *(engl. selekcija)* je mehanizam kojim se biraju jedinke iz populacije koje će sudjelovati u križanju. Algoritmi biraju jedinke prema tomu kakve im je vrijednosti dodjelila funkcija dobrote.

Turnirski odabir *(engl. tournament selection)* je algoritam koji u svakoj iteraciji bira uzorak populacije veličine *(n)* te iz tog uzorka odabire jedinku s najvećom dobrotom. Nakon toga jedinka se može (ne nužno) ukloniti iz populacije te algoritam prelazi na iduću iteraciju. Nakon k-iteracija odabira, prelazi se na križanje jedinki.

Opisani turnirski odabir u svakoj iteraciji uvijek bira najbolju jedinku. Moguća je i varijanta algoritma koji s određenom vjerojatnošću bira najbolju jedinku, a inače izabire nasumičnu jedinku u toj iteraciji. Turnirski odabir je pogodan iz više razloga: lako ga se implementira, radi na paralelnim arhitekturama te se lako bira veličina turnira.

Ruletski odabir *(engl. roulette wheel selection or fitness proportionate selection)* je algoritam koji svakoj jedinci dodjeljuje vjerojatnost izbora temeljenoj na funkciji dobrote te jedinke. Algoritam zatim u svakoj od k iteracija bira neku jedinku. Vjerojatnost izbora jedinke i računa se kao $p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$. Tako u populaciji čije jedinke imaju dobrotu [1 2 3 4 5] peta jedinka ima vjerojatnost odabira $\frac{1}{3}$. Iako kandidati s većom dobrotom imaju veću vjerojatnost izbora, može se dogoditi da će i slabije jedinke biti odabrane. Ovo je dobra strana algoritma jer slabije jedinke mogu imati neke korisne značajke koje će se prenjeti u križanju. Negativna strana ruletskog odabira je stohastički šum zbog kojeg se u praksi često biraju drugi algoritmi.

Stohastičko globalno uzorkovanje *(engl. stohastic universal sampling)* je algoritam koji je nastao kao nadogradnja ruletskog odabira. Naime, ako jedinka u ruletskom odabiru ima vjerojatnost odabira 7% te biramo 100 jedinki, očekujemo da će prosjećno ta jedinka biti odabrana 7 puta. Stohastičko globalno uzorkovanje osigurava ovakav izbor. U ruletskom odabiru takva jedinka može biti odabrana 0 puta, 15 puta, 100 puta. Zamislimo da stohastičko globalno uzorkovanje podjeli "rulet" tako da jedinke zauzimaju onoliki dio kotača kolika je vjerojatnost te jedinke. Zatim se "rulet" zavrti jednom te se nakon toga pomičemo po ruletu u jednakim koracima. Primjerice, ukoliko treba odabrati 15 jedinki tada će se od početne točke nastavljati prolaskom po ruletu za korak $\frac{1}{15} * 360°$. Ovo ne osigurava da će svaka jedinka na ruletu biti odabrana jer će ,ovisno o početnoj točki, biti moguće preskočiti neke jedinke na ruletu.

Odabir skraćivanjem *(engl. truncation selection)* je algoritam koji u svakoj iteraciji odabere jedinke čija dobrota spada u $p\%$ najboljih. Nakon toga stvara novu populaciju veličine $\frac{1}{p}$ križanjem.

3.4.2 Križanje

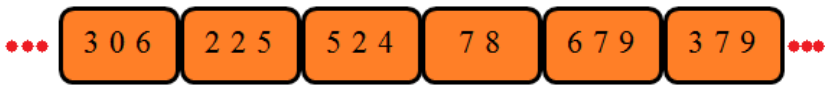
Križanje *(engl. crossover)* je mehanizam kojim se stvara nova jedinka iz dvije roditeljske jedinke. Općenito, nova jedinka će imati gene od oba roditelja, a poredak tih gena ovisi o načinu na kojeg radimo križanje.

Križanje u jednoj točki *(engl. Single point crossover)* odabire nasumičnu točku iz intervala [0, duljina gena]te oba roditelja djeli prema tim točkama. Dijete ovih roditelja će s jedne strane točke imati gene prvog roditelja, a s druge strane točke gene drugog roditelja. Moguće je odabrati više od jedne točke u križanju. Ovisno o broju odabranih točaka algoritam tada nazivamo križanje u k točaka *(engl. k-point crossover)*.

Slika 8. Primjer križanja u točki

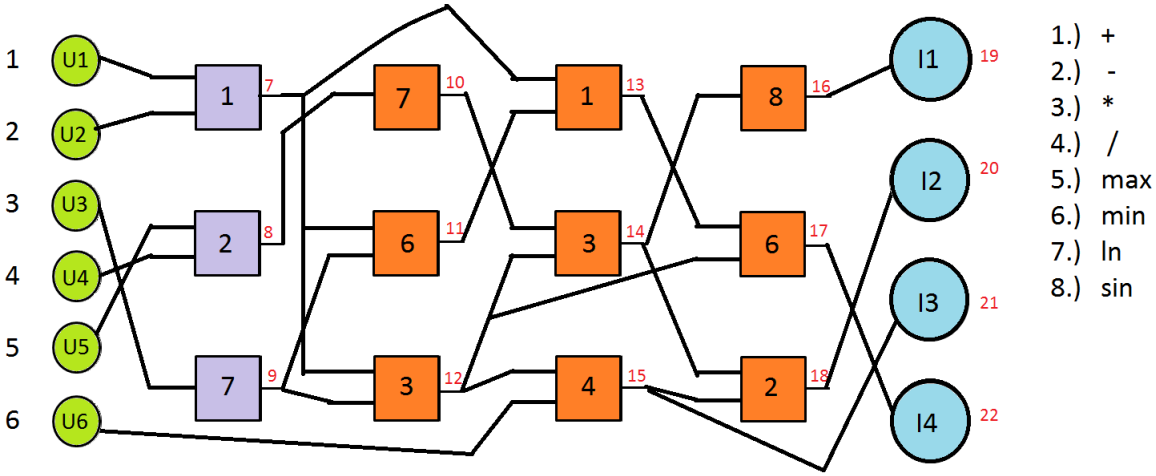
Slika 9. Primjer križanja u dvije točke

Na *slikama 10, 11 i 12* vidimo rezultat križanja mreže sa *slike 6.* s nekom mrežom gdje je točka križanja bila nakon prvog stupca.



Slika 10. Segment gena roditeljske mreže

Slika 11. Segment gena djeteta nakon križanja



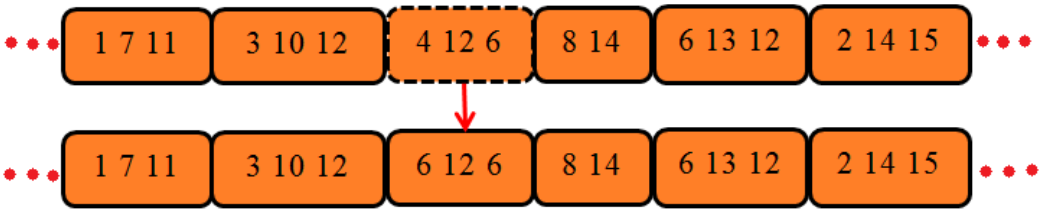
Slika 12. Izgled mreže nakon križanja u jednoj točki

Ujednačeno križanje *(engl. uniform crossover)* je križanje gdje se svaki čvor kojeg ima dijete bira nasumično između jednog od dvaju roditelja. Vjerojatnost izbora čvorova pojedinog roditelja je jednaka. Polu-ujednačeno križanje *(engl. half-uniform crossover)* je križanje gdje se pola čvorova bira nasumično između jednog od dvaju roditelja, a druga polovica gena bude preuzeta u potpunosti od jednog od dvaju roditelja.Nakon što se odradi križanje, često se primjeni mutacija na određeni broj novih jedinki prije nego se te jedinke dodaju u populaciju.

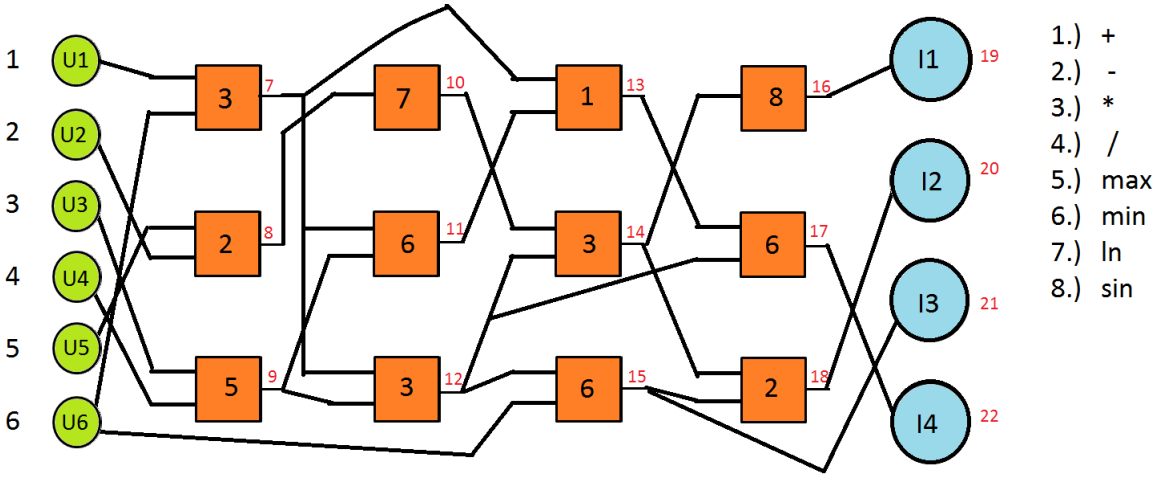
3.4.3 Mutiranje

Mutiranje *(engl. mutation)* je mehanizam kojim se mijenjaju geni pojedinih jedinka radi očuvanja genetičke raznolikosti populacije. Mutacija nasumično mijenja jedan ili više gena neke jedinke. Nakon mutacije rješenje se može znatno izmijeniti. Svrha ovoga je pomoći genetičkom algoritmu da izbjegne lokalne minimume tako što spriječava da geni jedinki postanu previše jednoliki jer takva populacija usporava ili zaustavlja evoluciju. Iz tog razloga se često u iteracijama algoritma osim odabira najboljih jedinki bira i nekoliko nasumičnih jedinki. Ukoliko je vjerojatnost mutacije previsoka, algoritam će se pretvoriti u nasumičnu pretragu.

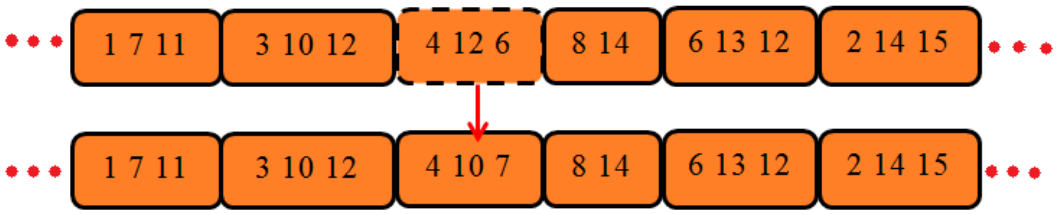
U kartezijskom genetskom programiranju ukoliko se mutira čvor koji predstavlja gen, moguće je mutirati ulaz tog čvora ili samu operaciju čvora.



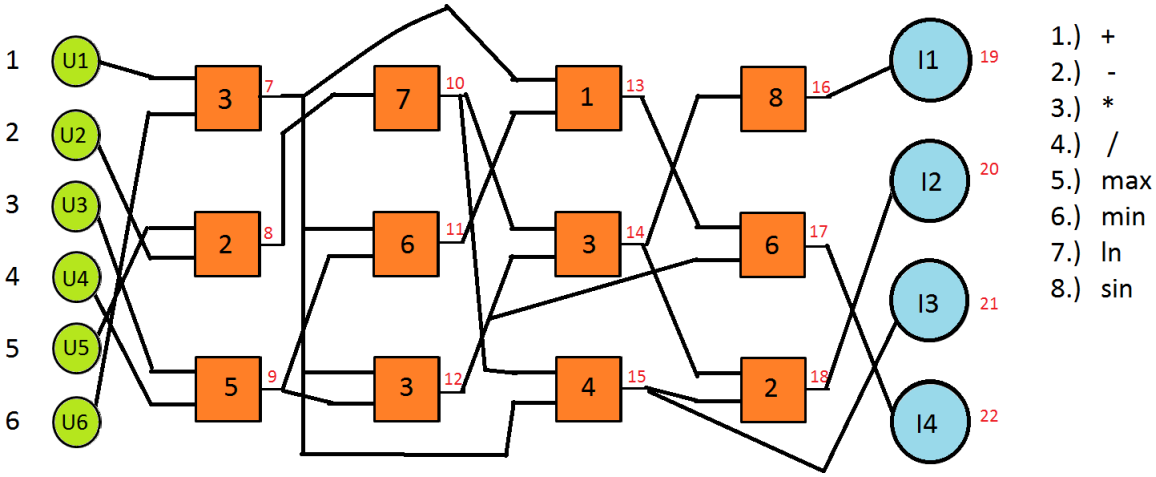
Slika 13. Prikaz mutacije operatora gena



Slika 14. Promjena mreže sa *slike 6.* nakon mutiranja operatora



Slika 15. Prikaz mutacije ulaza gena



Slika 16. Promjena mreže sa *slike 6.* nakon mutiranja ulaza

3.5 Primjena genetskog programiranja

Česta je uporaba genetskog programiranja (tako i kartezijskog genetskog programiranja) za optimizaciju nekih problema. Kako je izračun točnog rješenja često vremenski i prostorno zahtjevna operacija, genetsko programiranje često pronalazi dobro rješenje ako takvo postoji, ali nema jamstva da će se pronaći optimalno rješenje. Moguće je koristiti i genetsko programiranje za "okršaj" s problemima za koje ljudi nisu sigurni kako ih riješiti. Sve što je potrebno da ljudi budu u stanju prepoznati dobro rješenje kada im se prikaže. John R. Koza navodi 76 primjera gdje je genetsko programiranje proizvelo rezultate koji su u rangu s rezultatima koje je proizveo čovjek. Kartezijsko genetsko programiranje posebno je pogodno za višeklasnu klasifikaciju obzirom na višestruke izlaze.

4. Programsko ostvarenje

U ovom poglavlju bit će pojašnjen ECF, radno okruženje u kojem je implementiran klasifikator, način rada klasifikatora, mjera dobrote koje klasifikator koristi, operatori koje čvorovi koriste te potrebni parametri.

4.1 ECF

ECF *Evolutionary Computation Framework* je programski okvir napisan u C++ koji se koristi za primjenu evolucijsko računanja. Za korištenje ECF-a potrebno je definirati iduće komponente:

- *Algorithm* - Definira kakav se algoritam koristi za evoluciju jedinki. Algoritam može uključivati križanje i mutacije. Ukoliko nije definiran algoritam koristit će se pretpostavljeni algoritam *(SteadyState Tournament)*.
- *Genotype* - Definira genotip jedinki koje će algoritam evoluirati. Svaka jedinka u populaciji može imati jedan ili više genotipa. Genotip mora biti definiran. Svaki genotip navodi mehanizme križanja i mutacije koje može koristiti te strukturu podataka u kojima čuva značajke - potrebno za izračun funkcije dobrote.
- Evolutionary system - state - Algoritmi i genotip postoje u nekom stanju. Stanje se brine za veličinu populacije, uvjet zaustavljanja

4.2 Izračun mjera dobrote

Mjere dobrote koje su implementirane nastale su iz matrice zbunjenosti *(engl. confusion matrix)*. Matrica zbunjenosti je kvadratna matrica čiji broj stupaca i redaka odgovara broju klasa skupa podataka kojeg klasificiramo. Stupci matrice predstavljaju pravu klasu primjerka *(engl. actual class)*, a retci predstavljaju predviđenu klasu primjerka *(engl. predicted class)*.

	Natrij	Paladij	Aluminij
Natrij	14	4	1
Paladij	2	25	2
Aluminij	6	1	7

Slika 17. Prikaz matrice zbunjenosti s 3 klase

U primjeru sa *(engl. slike 17)* vidimo rezultat klasifikacije 62 uzorka metala, 22 natrija, 30 paladija, 10 aluminija. Iz matrice zbunjenosti za svaku klasu očitavamo 4 vrijednosti:

- TP_j *(engl. true positive)* - Pogodak, predviđena klasa primjerka odgovara stvarnoj klasi primjerka. Vrijednost j-tog elementa glavne dijagonale
- FP_j *(engl. false positive)* - Svrstavanje pogrešnog primjerka u klasu (sve ono što nije klasa j, a svrstano je u klasu j). Vrijednosti se nalaze u j-tom retku izuzev elementa glavne dijagonale

- FN_j *(engl. false negative)* - Primjerci klase koji su pogrešno svrstani (sve ono što je klasa j, a nije svrstano u klasu j). Vrijednosti se nalaze u j-tom stupcu izuzev elementa glavne dijagonale
- TN_j *(engl. true negative)* - Točno neprihvaćanje primjerka u klasu j. Vrijednosti se računaju kao $N - TP_j - FP_j - FN_j$

Primjerice, vrijednosti za natrij su: $TP_{Na} = 14$, $FP_{Na} = 5$, $FN_{Na} = 8$, $TN_{Na} = 62 - 14 - 5 - 8 = 35$. Koristeći prethodno opisane vrijednosti, izvodimo iduće mjere dobrote:

- Točnost *(engl. accuracy)* je omjer između svih točnih predviđanja te svih predviđanja.
$$\frac{TP + TN}{TP + TN + FP + FN}$$
- Pogreška *(engl. error)* je omjer između svih pogrešnih predviđanja te svih predviđanja.
$$\frac{FP + FN}{TP + TN + FP + FN}$$
- Preciznost *(engl. precision)* je omjer između pravih pozitiva te svih pozitivnih predviđanja.
$$\frac{TP}{TP + FP}$$
- Osjetljivost *(engl. sensitivity)* je omjer između pravih pozitiva te sume pravih pozitiva i lažnih negativa.
$$\frac{TP}{TP + FN}$$
- Specifičnost *(engl. specificity)* je omjer između pravih negativa te sume pravih negativa i lažnih pozitiva.
$$\frac{TN}{TN + FP}$$
- F1 uspjeh *(engl. F1 score)* je harmonijska sredina preciznosti i osjetljivosti.
$$\frac{2 * TP}{2 * TP + FP + FN}$$

Makro-prosjek - izračun mjere za svaku klasu, pa uprosječivanje kroz klase

$$M = \frac{1}{N} * \sum_{j=1}^{N} M_j$$

Odaberemo li kao mjeru dobrote F1 uspjeh, te primjenimo to na matrici zbunjenosti sa *slike 17* tada dobivamo vrijednosti $F1_{Na} = \frac{28}{41} \approx 0.683$, $F1_{Pd} = \frac{50}{59} \approx 0.847$, $F1_{Al} = \frac{7}{12} \approx 0.583$.

Konačna F1 ocjena je aritmetička sredina prethodno navedenih ocjena:

$$F1 = \frac{F1_{Na} + F1_{Pd} + F1_{Al}}{3} \approx 0.704$$

4.3 Izvedba klasifikatora

U ovom radu implementiran je jedan klasifikator - Višeklasni CGP, čija implementacija odgovara mrežama kakvim smo prethodno opisali kartezijsko genetsko programiranje. Parametre klasifikatora definiramo kroz konfiguracijsku datoteku:

```
<ECF>
  <Algorithm>
      <SteadyStateTournament>
          <Entry key="tsize">3</Entry>
      </SteadyStateTournament>
  </Algorithm>
  <Genotype>
      <Cartesian>
          <Entry key="numvariables">4</Entry>
          <Entry key="numoutputs">3</Entry>
          <Entry key="numrows">15</Entry>
          <Entry key="numcols">15</Entry>
          <Entry key="levelsback">4</Entry>
          <Entry key="functionset">+ / * -  step max min</Entry>
      </Cartesian>
  </Genotype>
  <Registry>
      <Entry key="population.size">30</Entry>
      <Entry key="term.fitnessval">1</Entry>
      <Entry key="term.eval">3000</Entry>
      <Entry key="mutation.indprob">0.3</Entry>
      <Entry key="log.level">3</Entry>
      <Entry key="log.frequency">1</Entry>
      <Entry key="log.filename">log.txt</Entry>
      <Entry key="training.infile">./GlassTraining.csv</Entry>
      <Entry key="testing.infile">./GlassTest.csv</Entry>
      <Entry key="use.intervals">no</Entry>
      <Entry key="intervals.infile">./intervals.txt</Entry>
      <Entry key="measure">F1</Entry>
      <Entry key="hasID">0</Entry>
      <Entry key="evaluate.during.training">0</Entry>
  </Registry>
</ECF>
```

Prikaz konfiguracijske datoteke

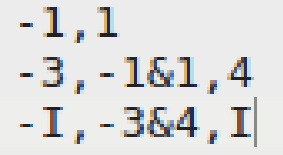
U *<Cartesian>* opisuje se mreža. Broj ulaza *numvariables* u mrežu odgovara broju značajki. Broj izlaza *numoutputs* iz mreže odgovara broju klasa . Broj redaka i stupaca mreže zadaju se u *numrows* i *numcols*. Povezivanje unatrag zadaje se u *levelsback*. Funkcijski skup navodi koji operatori mogu biti korišteni u genima.

<Registry> opisuje dodatne parametre za rad programa. Veličina populacije u svakoj generaciji zadaje se u *population.size*. Uvjeti zaustavljanja su *term.fitnessval* vrijednost mjere dobrote za koju se zaustavlja program ili broj evaluacija *term.eval*.

Vjerojatnost mutacije jedinke zadaje se kao *mutation.indprob*. Putanja do skupa podataka za učenje zadana je u *training.infile*, putanja do skupa podataka za ispitivanje zadana je u *testing.infile*.

Klasifikator ima dva načina određivanja klase izlaza. Prvi i pretpostavljeni način rada je da će svaki izlaz iz mreže predstavljati jednu klasu. Kada algoritam odradi evaluaciju onaj izlaz koji ima najveću brojčanu vrijednost će biti odabran kao klasa trenutnog ulaznog primjera. Primjerice ukoliko mreža jedinke na izlazima ima vrijednosti 0.567, 0.77, 1.21, 0.02, taj ulazni primjer bit će svrstan u 3. klasu.

Ako se parametar "use.intervals" postavi na vrijednost "1 ili yes" klasifikator će preći u drugi način rada. Ovaj način rada zahtijeva da mreža ima samo jedan izlaz. Nakon što algoritam odradi evaluaciju radi provjeru u koji interval pripada dobivena vrijednost te ovisno o pripadnosti to se uzima kao klasa ulaznog primjera.
Definicija intervala klasa zadaje se u parametru *intervals.infile*.



Slika 18. Prikaz definicije intervala klasa

U prvu klasu će biti svrstane ulazni primjeri čiji maksimalni izlaz ima vrijednost između -1 i 1. U drugu klasu će biti svrstane jedinke čiji maksimalni izlaz ima vrijednost između -3 i -1 ili 1 i 4. U treću klasu će biti svrstane jedinke čiji maksimalni izlaz ima vrijednost između -beskonačno i -3 ili 4 i beskonačno. Primjerice neka vrijednost izlaza bude *1.21*. Klasifikator će taj primjer svrstati u drugu klasu.

Ukoliko se parametar "use.intervals" postavi na vrijednost "1 ili yes", ali se ne navede "lokacija definicije klasa" tada će se koristiti pretpostavljeni intervali(granice intervala su potencije broja dva u oba smjera brojevnog pravca). Prva klasa će imati interval između -1 i 1, druga klasa između -2 i -1 ili 1 i 2, treća klasa između -4 i -2 ili 2 i 4 itd.

Funkcija mjere dobrote zadaje se u parametru *measure*. Pretpostavljena mjera dobrote je točnost *(engl. accuracy)*. Parametar *hasID* govori imaju li primjerci u skupovima podataka za učenje i ispitivanje redni broj u zapisu. Pretpostavljena vrijednost ovog parametra je da podatci nemaju rednii broj u zapisu "0 ili no". Ako se parametar *evaluate.during.training* postavi na "1 ili yes", tada će jedinke biti ispitivane na skupu za ispitivanje za vrijeme evolucije. Pretpostavljena vrijednost ovog parametra je "0 ili ne".

4.4 Funkcije čvorova, operatori mutacije i križanja

Implementirano je 11 funkcija koje čvorovi mogu obavljati. Možemo ih podjeliti na funkcije koje uzimaju 2 operanda i funkcije koje uzimaju 1 operand. To su:

- Aritmetičke operacije(+, -, *, /), max, min

- Sinus, kosinus, korijen, logaritam, pos

Max funkcija vraća veću od dvije vrijednosti na ulazu. Min funkcija vraća manju od dvije vrijednosti na ulazu. Korijen vraća kvadratni korijen apsolutne vrijednosti ulaza. Pos funkcija vraća ulaznu vrijednost ukoliko je veća od 0. Inače vraća 0.

Operatori križanja koji se koriste u implementiranom genotipu su križanje u točki, križanje u 2 točke, ujednačeno križanje i polu-ujednačeno križanje. Operatori mutacije koji se koriste su mutacije veze i mutacije funkcije.

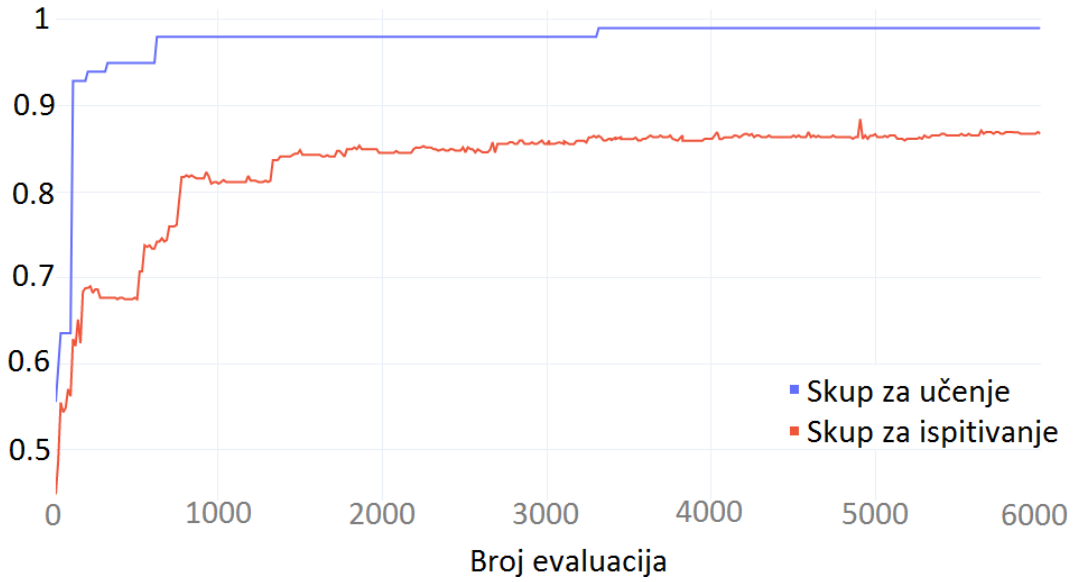
5. Primjena programa i rezultati mjerenja

U ovom poglavlju biti će prikazani rezultati mjerenja nad skupovima podataka. Skupovi podataka će se dijeliti u omjeru (70% - skup za učenje, 30% - skup za ispitivanje). Prvi skup kojeg klasificiramo su porodice cvijeta iris(perunika) *(engl. iris flower dataset)*. Drugi skup kojeg klasificiramo su stakla na osnovu svojstava i kemijskog sastava. Svaki primjer će koristiti F1 uspjeh kao mjeru dobrote.

5.1 Skup cvijeća iris

Skup podataka se sastoji 150 primjeraka cvijeća. Svaki primjerak cvijeta ima 4 značajke(mjerene u centimetrima) koje ga opisuju: duljina peteljke *(engl. sepal length)*, širina peteljke *(engl. sepal width)*, duljina latice *(engl. petal length)* i širina latice *(engl. petal width)*. Porodice cvijeta iris su: Iris-setos, Iris-versicolor i Iris-virginica. Tijekom učenja jedinki, svaka će biti ispitana na skupu za ispitivanje. Nakon određivanja uvjeta zaustavljanja evaluacije, namještati će se pojedini parametri.

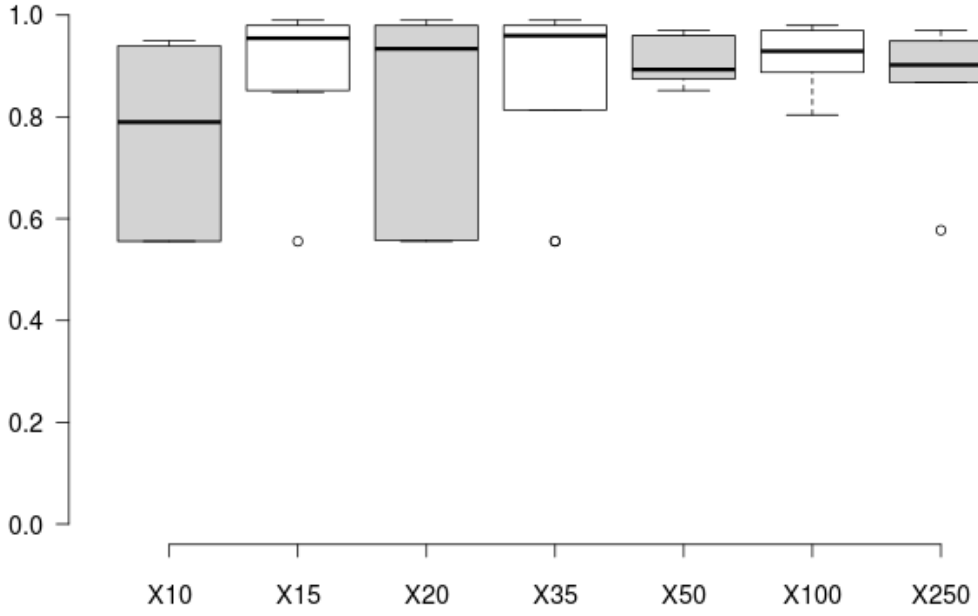
Parametar	**Početna vrijednost**
Veličina populacije	15
Veličina mreže	10 x 10
Levelsback	3
Vjerojatnost mutacije	0.3
Korištene mutacije	Mutacija veze i mutacija operatora
Korištena križanja	Križanje u jednoj točki, ujednačeno i polu-ujednačeno
Korištene funkcije	+, -, *, /, step, max, min,



Slika 19. Kretanje prosjeka najboljih jedinki u 10 pokretanja

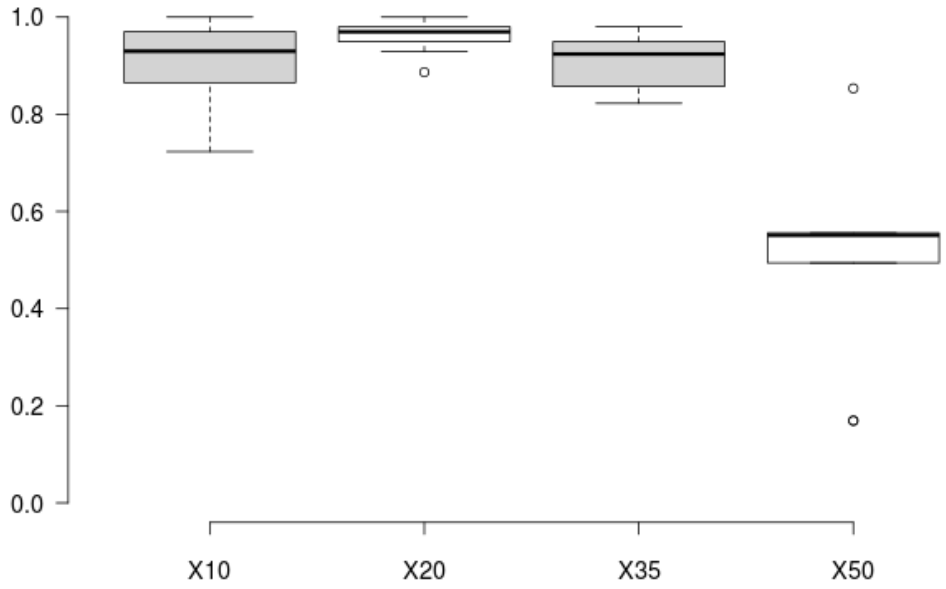
Na *slici 19.* može se vidjeti da na skupu za učenje dolazi do stagnacije nakon 630 evaluacija. Uvjet kraja evaluiranja postaviti će se na 1800 evaluacija. Uspjeh na skupu za ispitivanje prati skup za učenje.

Prvi parametar koji se namješta je veličina populacije. Ispitati će se uspjesi nad veličinama od 10, 15, 20, 35, 50, 100 i 250 jedinki. Prevelikm brojem populacije, događa se manje evaluacija na svakoj jedinki (time manje mutacija i križanja) pa se utvrđuje da je bolje imati manju populaciju. Odabire se 35 za veličinu populacije.

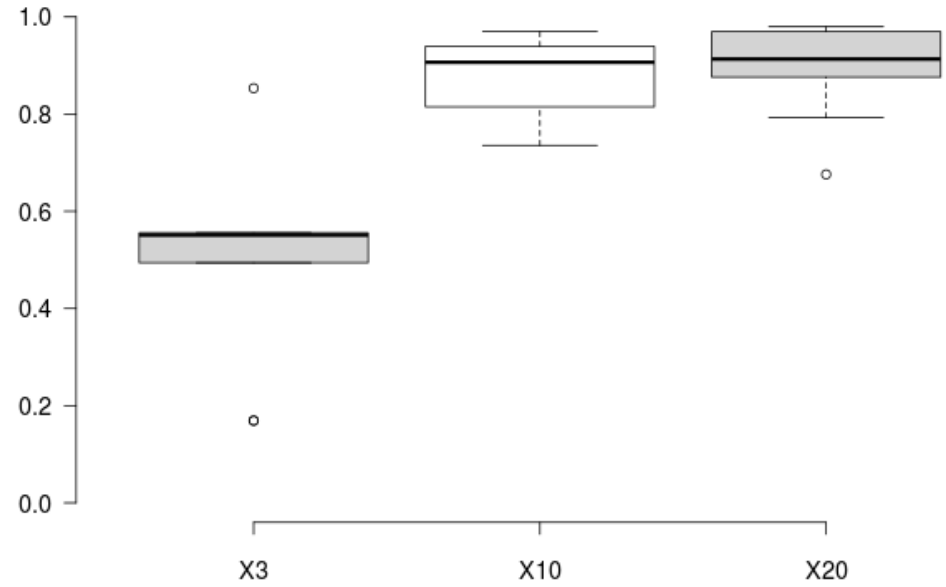


Slika 20. Utjecaj veličine populacije

Drugi parametar koji se namješta je dimenzija mreže. Ispitati će se uspjesi nad dimenzijama 10x10, 20x20, 35x35, 50x50. Zanimljivo je vidjeti kako popraviti mrežu velikih dimenzija - povećanjem *levelsbacka* što se vidi na *slici 22.*. Ipak evaluacija velike mreže traje vremenski duže te će zbog toga biti odabrana mreža dimenzija 20x20.



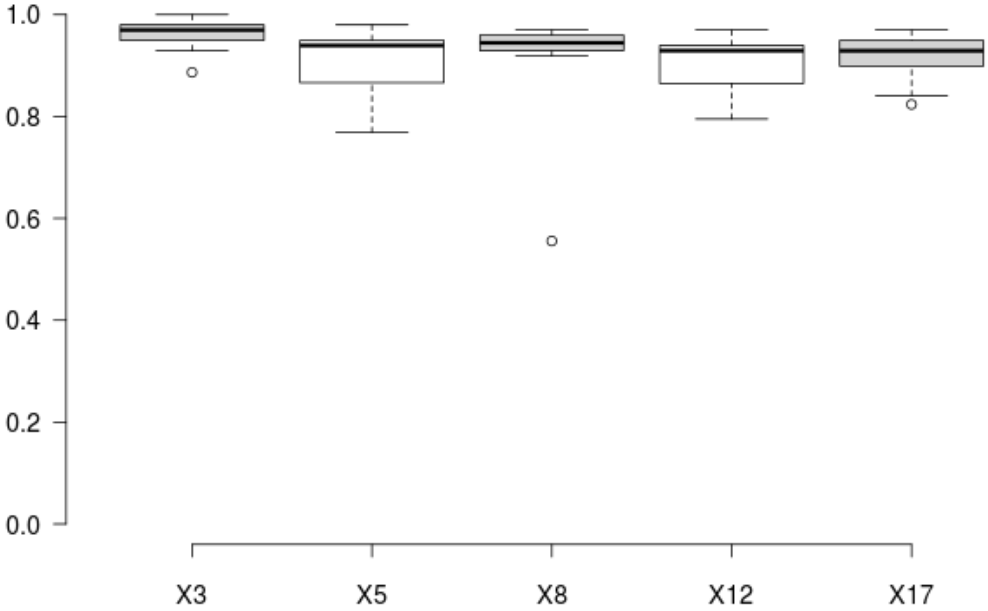
Slika 21. Utjecaj dimenzije mreže



Slika 22. Popravljanje mreže velikih dimenzija promjenom levelsbacka

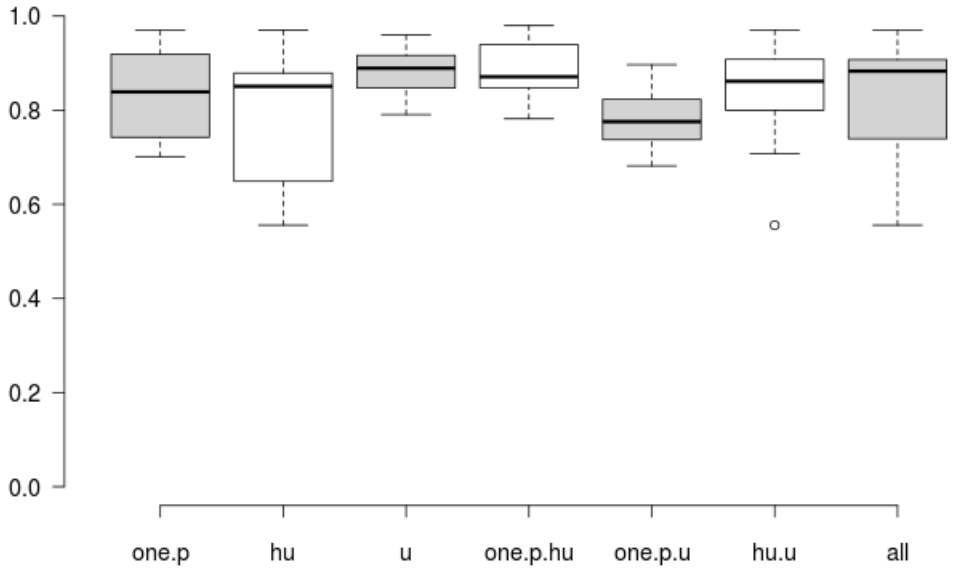
Pogledajmo utjecaj levelsbacka na mrežu manjih dimenzija. Ispitati će se levelsback vrijednosti 3, 5, 8, 12, 17 na mreži 20x20. Pokazuje se da levelsback nema

značajan utjecaj na uspjeh mreže.



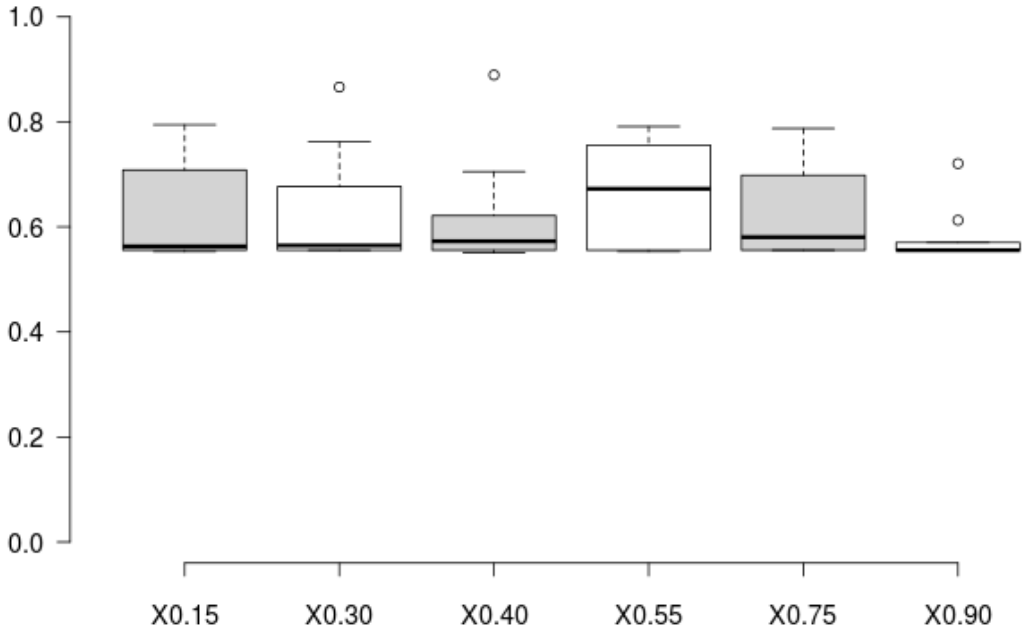
Slika 23. Utjecaj levelsbacka na mrežu dimenzija 20x20

U nastavku promatramo utjecaje križanja i mutacija na uspjeh evaluatora. *Slika 24.* prikazuje rezultate gdje su omogućeni različiti oblici križanja, a mutacija je onemogućena. Jednoliko križanje(stupac u) ostvaruje najbolji uspjeh od pojedinačnih križanja što je očekivano obzirom da su geni stvorenog djeteta u ovom križanju najraznolikiji. U svakoj varijanti pojedine jedinke dolaze do najboljeg uspjeha od 95% - 97%.

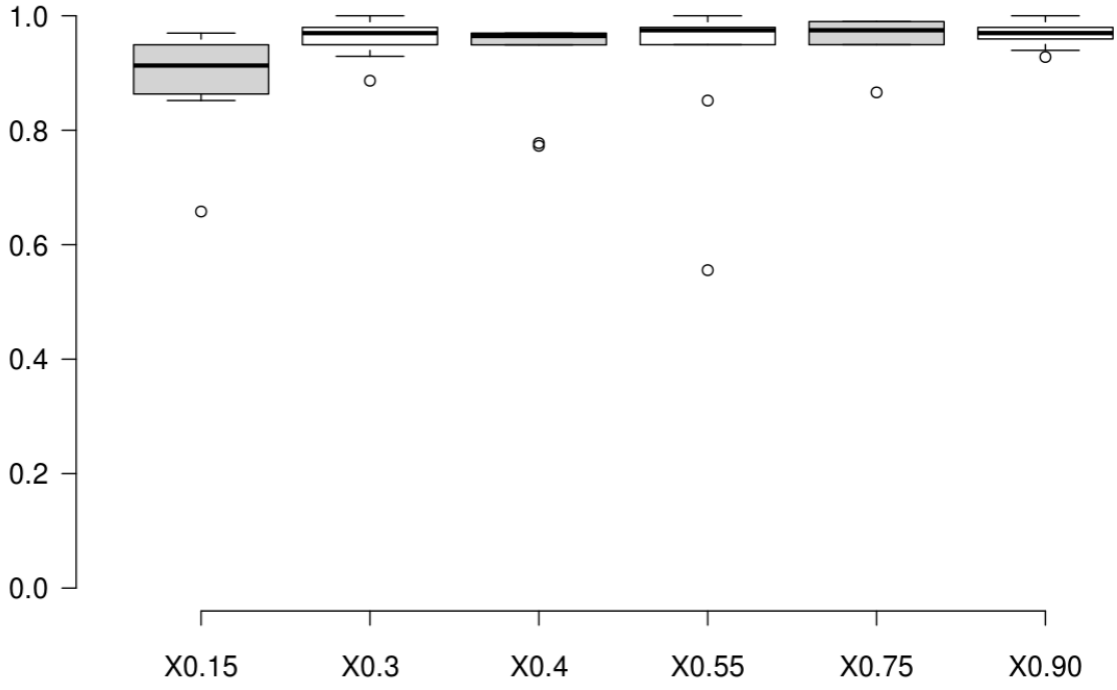


Slika 24. Utjecaj križanja na uspjeh jedinki

Onemogućavanjem križanja i omogućavanjem mutacije operatora i veze (uz različite vjerojatnosti) dobivaju se rezultati na *slici 25.*. Uspjeh ovakvih jedinki je znatno slabiji od jedinki gdje je omogućeno križanje. Kao što je prethodno navedeno, mutacija je mehanizam koji pridonosi raznolikosti populacije te izbjegavanju lokalnog optimuma, ali niije dovoljna za napredak jedinki prema cilju.



Slika 25. Utjecaj mutacije na uspjeh jedinki



Slika 26. Utjecaj mutacije uz omogućena križanja

Konačno omogućavanjem svih križanja i mutacija uz različite vjerojatnosti mutacije dobivamo rezultate sa *slike 26.*. Uspjeh ovakvih rezultata bolji je od obje prethodne varijante. Time se odabire parametar vjerojatnosti mutacije od *0.55* te se uključuju svi oblici mutacije i križanja u evaluator. Konačni izbor parametara dan je u idućoj tablici:

Parametar	**Konačna vrijednost**
Veličina populacije	35
Veličina mreže	20 x 20
Levelsback	3
Vjerojatnost mutacije	0.55
Korištene mutacije	Mutacija veze i mutacija operatora
Korištena križanja	Križanje u jednoj točki, ujednačeno i polu-ujednačeno
Korištene funkcije	+, -, *, /, step, max, min,

5.2 Klasifikacija stakla

Skup se sastoji od 215 primjeraka. Svaki primjerak stakla opisuje se s 9 značajki i pripada jednoj od 6 klasa. Ovaj skup podataka za razliku od prethodnog nema jednolik rasproed primjeraka po kategorijama zbog čega možemo oečkivati da će klasifikatoru biti teže odrediti točnu klasu. Značajke koje određuju lom stakla su:

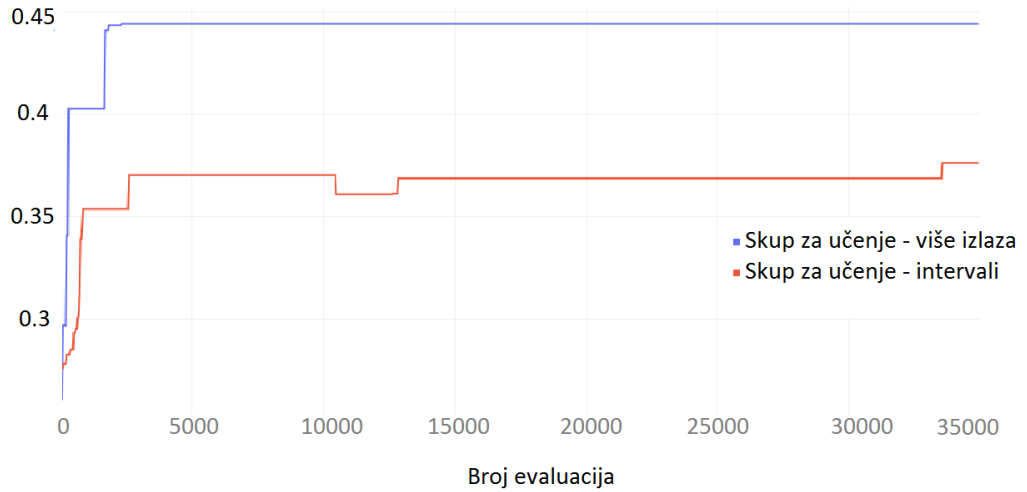
- Lom svjetla
- Natrij
- Magnezij
- Aluminij
- Silicij
- Kalij
- Kalcij
- Barij
- Željezo

Moguće klase su:

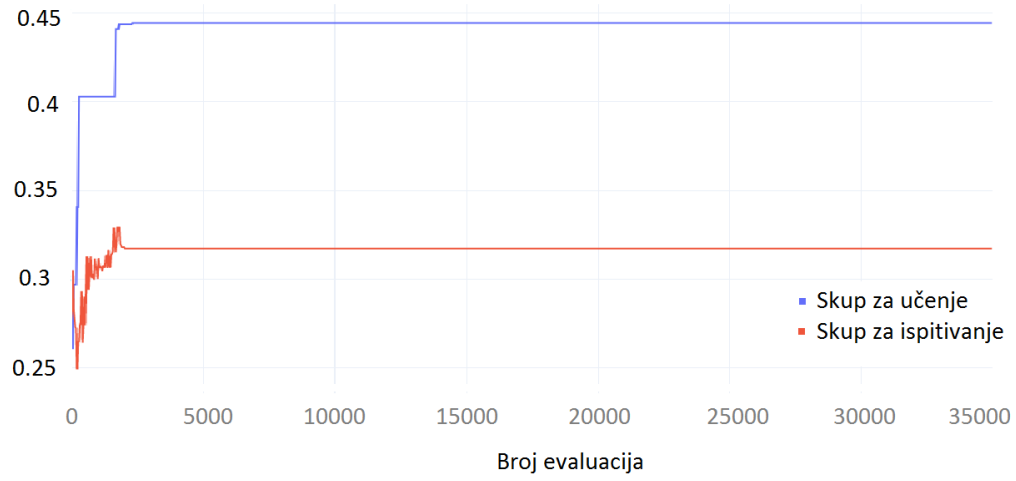
- Obrađenoo staklo prozora zgrada
- Neobrađeno staklo prozora zgrada
- Obrađeno staklo prozora vozila
- Staklo posuđa
- Stolno posuđe

- Staklo automobilskih farova

Slika 27. prikazuje učenje koristeći više izlaza te učenje koristeći binarne intervale i jedan izlaz. Obzirom da učenje koristeći više izlaza daje bolje rezultate, *slika 28.* prikazuje usporedbu uspjeha skupa za učenje i skupa za ispitivanje. Odrađeno je 35000 evaluacija $\approx$ 100 generacija.



Slika 27. Usporedba učenja u različitim načinima rada



Slika 28. Uspjeh na skupu za učenje i ispitivanje (više izlaza)

5.3 Ispitivanje kvalitete auta

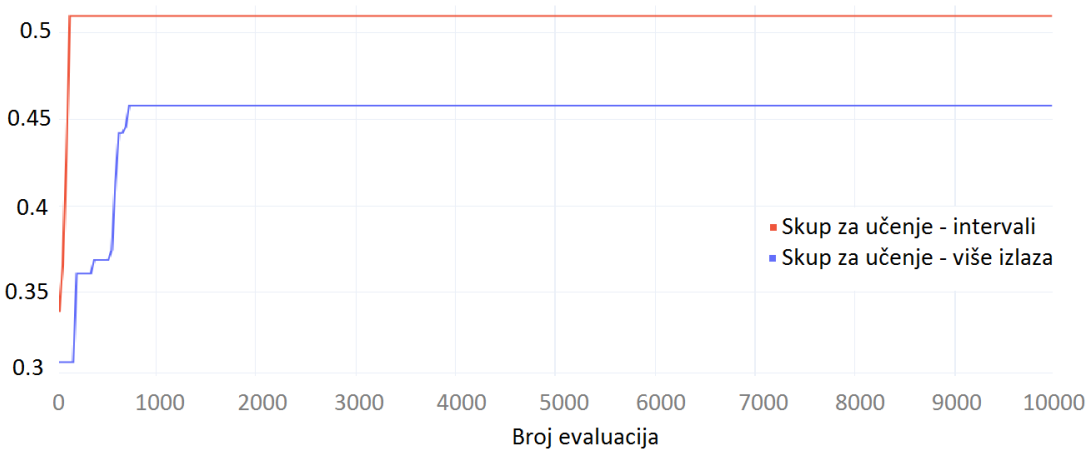
Skup se sastoji od 1728 primjeraka. Svaki primjerak auta opisuje se sa 6 značajki i pripada jednoj od 4 klasa. Značajke koje određuju kvalitetu auta:

- Cijena
- Cijena održavanja
- Broj vrata
- Broj ljudi
- Veličina pretinca za prtljagu
- Sigurnost automobila

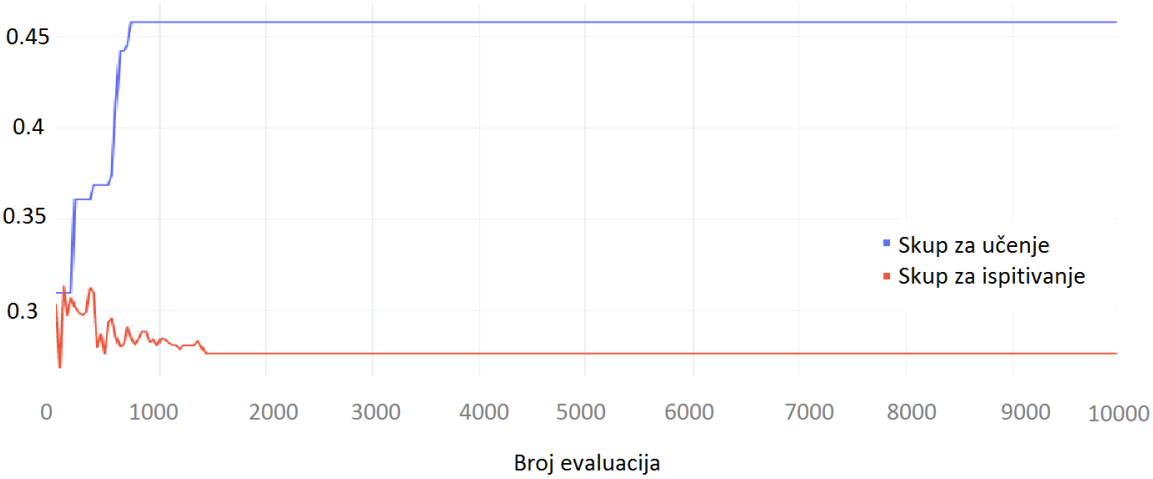
Moguće klase su:

- Neprihvatljiv
- Prihvatljiv
- Dobar
- Vrlo dobar

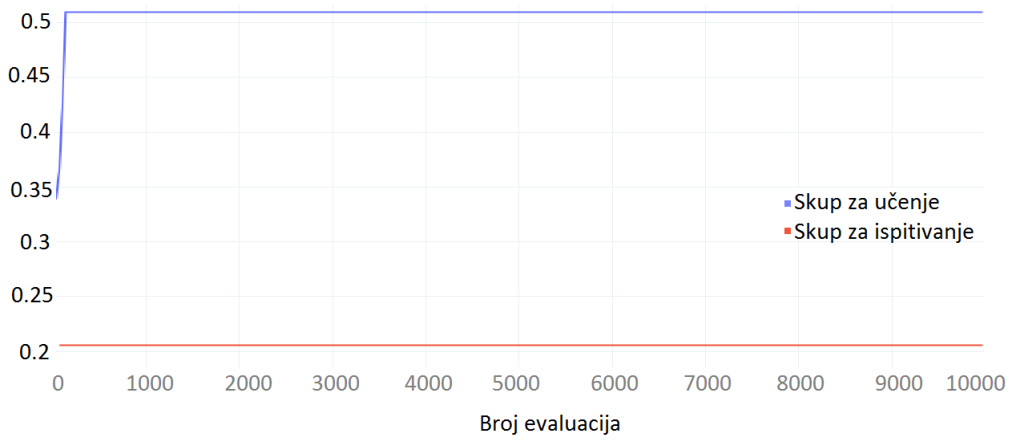
Slika 29. prikazuje učenje koristeći više izlaza te učenje koristeći binarne intervale i jedan izlaz. Za razliku od prethodnog skupa *(5.2)* ovdje je način rada koristeći binarne intervale uspješniji te je postigao staganciju vrlo rano - nakon svega 140 evaluacija = 3 generacije. Međutim, *slika 30.* i *slika 31.* pokazuju da je na skupu za ispitvanje bolji uspjeh ostvario način rada s više izlaza. Odrađeno je 10000 evaluacija $\approx$ 286 generacija.



Slika 29. Kretanje prosjeka najboljih jedinki kroz 10 pokretanja na skupu za učenje



Slika 30. Uspjeh na skupu za učenje i skupu za ispitivanje(više izlaza)



Slika 31. Uspjeh na skupu za ispitivanje koristeći binarne intervale

5.4 Usporedba dobivenih rezultata s drugim genetskim algoritmima

Dana tablica uspoređuje uspjehe mjera dobrote nad skupovima za ispitivanje klasifikatora ostvarenog kartezijskim genetskim programiranjem sa klasifikatorima koji koriste stabla odluke [6] te klasifikatorima koji koriste jedno ili više regresijskih stabala [5]. Svaki od klasifikatora u tablici za mjeru dobrote koristio je F1 uspjeh.

Skup podataka	Broj generacija	CGP	Stabla odluke	Regresijsko stablo	Više regresijskih stabala
Iris flower	100	84%	99%	100%	100%
Valjanost auta	100	≈27%	≈34-36%	≈56%-57%	≈55%-56%
Staklo	100	≈31%	/	≈60%	≈60%

Slika 32. Tablica usporedbi uspjeha klasifikatora

6. Zaključak

U radu je opisano kartezijsko genetsko programiranje i problem klasifikacije. Pokazana je prednost korištenja mutacija i križanja (zajedno) u evoluciji klasifikatora. Klasifikator pokazuje vrlo dobre rezultate na skupovima podataka čiji su primjeri jednoliko raspoređeni među klasama te imaju manje značajki, dok su rezultati za skupove s više značajki prihvatljivi.

Literatura

1. Alpaydin, M. Introduction to Machine Learning (second edition). MIT Press, Cambridge, MA, USA, 2010. ISBN 978-0-262-01243-0 stranice 1-14

2. Miller, J.F. Cartesian Genetic Programming in a nutshell. Ten-slide introduction to CGP

3. Ecf programski okvir. ECF

4. Bašić, B.D., Šnajder, J. Prikaznice iz predmeta umjetna inteligencija: Strojno učenje, Zagreb 2017. Strojno učenje

5. Vlašić, I. Primjena genetskog programiranja na problem klasifikacije podataka, Zagreb 2016. Vlašić, I. - Završni rad

6. Kolobara, V. Rješavanje klasifikacijskih problema evolucijom stabla odluke, Zagreb 2016. Kolobara, V. - Završni rad

Rješavanje klasifikacijskih problema uz pomoć kartezijskog genetskog programiranja

Sažetak

Stojno učenje je grana računarske znanosti posebno pogodna za rješavanje probelma koje je teško eksplicitno programirati. Jedan od problema kojeg možemo rješavati strojnim učenjem jest klasifikacija - svrstavanje pojedinih stvari u određene kategorije temeljem njihovih značajki. Za rješavanje problema klasifikacije primjenjuje se kartezijsko genetsko programiranje. Implementacija je odrađena unutar radnog okvira ECF te su prikazani rezultati klasifikacije nad različitim podatcima. Klasifikator je davao bolje rezultate za probleme s manje katergorija i jednolikijim rasporedom jedinki po kategorijama, dok je za probleme s više kategorija i nejednolikim rasporedom davao prihvatljive rezultate.

Ključne riječi: Strojno učenje, klasifikacija, kartezijsko genetsko programiranje

Solving classification problems using cartesian genetic programming

Abstract

Machine learning is a branch of computer science which is very applicable for solving problems that can not be explicitly programmed. One of problems which can be solved using machine learning is classification - classifing an example in specific category based on example's features. For solving classification problems Cartesian Genetic Programming is used. Implementation is done using framework ECF and results of classification on different data is shown. Classificator produced better results for problems which have less categories and uniform distribution of examples over categories while it gave acceptable performance for problems with more categories and not uniform distribution of examples.

Keywords: Machine learning, classification, cartesian genetic programming