Univerzitet u Novom Sadu
Fakultet Tehničkih nauka

dokumentacija PROJEKTNOG ZADATKA

Studenti:	Jelena Miletić SW45-2018 , Dina Petrov SW52-2018
Predmet:	Nelinearno programiranje i evolutivni algoritmi
Broj zadatka:	11
Tema zadatka:	Genetski algoritam, "black-box optimizacija" neuronska mreža

opis problema

U projektu rešavamo problem optimizacije date neuronske mreže. Kao izlaz neuronske mreže dobijamo njenu grešku i pokušavamo da optimizujemo izlaz pomoću genetskog algoritma tj. da što više smanjimo vrednost (da bude što bliža nuli). Neuronskoj mreži prosleđujemo vektor od 60 vrednosti pomoću kojih ona dobija težine na svakoj od grana unutar nje. Ovakvom optimizacijom tražimo minimum funkcije.

uvod

GENETSKI ALGORITAM

Genetski algoritam predstavlja optimizacioni metod koji dolazi do rešenja oponašajući osnovne mehanizme evolucije iz prirode kao što su selekcija, ukrštanje, mutacija..
Svaka iteracija genetskog algoritma odgovara jednoj 'generaciji jedinki', odnosno trenutnoj populaciji. Svaka jedinka neke generacije predstavlja potencijalno rešenje našeg problema. One imaju niz osobina (karakteristika), kao i svoj stepen prilagodjenosti (engl. *fitness*). U našim numeričkim metodama on odgovara vrednosti funkcije, odnosno vrednosti kriterijuma optimalnosti f(x). Ove vrednosti nam pomažu da procenimo koliko nam je korisna neka jedinka kao rešenje problema. Pošto tražimo minimum funkcije, bolje je prilagođena jedinka čija je vrednost kriterijuma optimalnosti *manja*.
U toku rada genetskog algoritma dolazi do smenjivanja generacija jedinki sa ciljem povećavanja njihove prilagođenosti. Samim tim dolazi i do poboljšanja vrednosti kriterijuma optimalnosti.

Algoritam ukratko:

1. Inicijalno se generiše skup jedinki koji predstavljaju početnu populaciju
2. Za svaku jedinku iz početne populacije računa se njena prilagođenost
3. Vrši se proces *selekcije* nad početnom populacijom - biraju se one jedinke koje će ući u proces reprodukcije, odnosno *ukrštanja.* Jedinka se može više puta odabrati za proces ukrštanja. Veće šanse za odabir pri selekciji imaju one jedinke koje su bolje prilagođene.
4. Nad dve selektovane jedinke (roditelji) vrši se *ukrštanje* - dolazi do kombinacije osobina roditelja i stvaraju se nove jedinke - *potomci*.
5. Nad nekim, slučajno odabranim potomcima vrši se *mutacija* sa veoma malom verovatnoćom. Mutacija obuhvata izmenu nekih osobina (gena) potomka. Mutacije su neophodne jer sprečavaju prevremenu konvergenciju i zaglavljivanje u lokalnom ekstremu.

6. Kada se procesom reprodukcije stvori dovoljan broj potomaka, oni se ubacuju u populaciju i zamenjuju sve ili samo neke jedinke prethodne, roditeljske populacije.
7. Stvaranjem nove populacije se završava jedna iteracija (generacija) algoritma, a zatim se ciklično ponavlja proces dok se ne dostigne predodređen broj generacija.

- Struktura programa

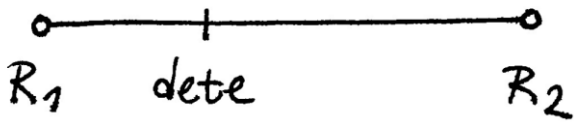
  Program sadrži klasu *Individual* koja predstavlja jednu jedinku u populaciji. Ona sadrži vektor svih svojih karakteristika *W = [w1, w2, w3, ..., w59, w60]* i vrednost svoje prilagođenosti koju dobijamo od izlaza iz neuronske mreže.
  Neke od vrednosti algoritma koje su postavljen na početku su:

```
population_size = 200
characteristics_num = 60
min_range = -3
max_range = 3
mutation_chance = 0.1
mutation_genes_num = 3
generation_max = 50
```

characteristics_num - broj vrednosti u vektoru svake jednike
min_range i *max_range* - minimalna i maksimalna vrednost težina u vektoru karakteristika jedinki
mutation_genes_num - broj karakteristika jedinke koje menjamo u slučaju mutacije

- Selekcija jedinki za ukrštanje (roditelji)
  - Za odabir kandidata za ukrštanje koristile smo koncept elitizma Ideja je da roditelji imaju priliku da budu ubačeni u narednu generaciju umesto svoje dece ako su bolje prilagođeni od njih. Postiže se sortiranjem roditelja i potomaka po njihovoj prilagođenosti.
  - Nakon sortiranja, odaberemo dvadeset najprilagođenijih jedinki. Kako ne bismo imali prevremenu konvergenciju algoritma, takođe ubacujemo i nasumično odabrane jedinke koje nisu u najboljih 20 i njima popunjavamo ostatak nove populacije.

- Ukrštanje jedinki
  - Kako su vrednosti karakteristika jedinki realni brojevi, iskoristile smo ideju da će vrednost jedne karakteristike jedinke biti kombinacija vrednosti iste te karakteristike oba roditelja jedinke.



- Uzimamo svaki "gen" (karakteristiku) iz vektora od 60 vrednosti oba roditelja. Na primer, uzimamo prvu vrednost od prvog roditelja i prvu vrednost od drugog roditelja. Prva vrednost u vektoru potomka će tada biti u opsegu ograničenim vrednostima koje imaju roditelji.

```
for i,j in zip(parent1.characteristics, parent2.characteristics):
    if i < j:
        val = np.random.uniform(i, j)
        child.characteristics[counter] = val
    else:
        val = np.random.uniform(j, i)
        child.characteristics[counter] = val
    counter += 1
```

- Mutacija jedinki
  - Definisale smo vrednost promenljive **mutation_chance** koja iznosi 0.1. To znači da svaka jedinka ima 10% šanse da se nad njom obavi mutacija. Takođe smo odredile vrednost promenljive mutation_genes_num = 3 koja govori da prilikom mutacije menjamo tri vrednosti unutar vektora karakteristika jedinke.
  - Korišćenjem *numpy* biblioteke i funkcije *random.uniform()* postigle smo slučajni odabir jedinki koje će se mutirati, osobina unutar vektora karakteristika čije će vrednosti menjati, kao i generisanje novih vrednosti tih karakteristika.

```
if np.random.uniform(0, 1) <= mutation_chance:
    for i in range(mutation_genes_num):
        pos_to_mutate = randint(0, characteristics_num - 1)
        self.characteristics[pos_to_mutate] = np.random.uniform(min_range, max_range)
```

- Odabir parametara algoritma
  - Prilikom testiranja projekta zaključile smo da je za postizanje najoptimalnijeg odnosa vremena izvršavanja i tačnosti rešenja idealan broj generacija 50 (čije dostizanje je i ujedno kriterijum zaustavljanja).
  - Veličina jedne populacije iznosi 200 jedinki.

    - Šansa za mutaciju je 10% (<= 0.1) kako bismo smanjile šansu da algoritam upadne u lokalni minimum funkcije.

- Rezultati algoritma

```
Minimum funkcije je: 0.02615587693161249
Vreme izvrsavanja za 50 generacija je: 195.780699968338
```

    - Razlika u rezultatima kada u populaciji imamo 200 i 300 jedinki je zanemarljivo mala, ali se u slučaju sa 200 jedinki program izvršava značajno brže.

    - Implementacijom koncepta elitizma, vreme izvršavanja programa je osetno duže nego što bi bilo koristeći neki drugi metod selekcije. Glavni razlog jeste postupak sortiranja svake populacije kako bi se izvukle najprilagođenije jedinke, što je dugotrajan proces sa velikim brojem generacija. Uprkos tome, svesno smo odabrale upravo taj metod kako bi naš algoritam bio što 'sličniji' prirodnim procesima evolucije koje oponaša.