



Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Dokumentacija za projektni zadatak

Januar 2021.

Studenti:

Ilija Kalinić, SW65/2019

Zoran Bukorac, SW40/2019

Matija Zarić, SW24/2019

Predmet:

Nelinearno Programiranje i Evolutivni Algoritmi

Broj projektnog zadatka:

br. 12

Tema projektnog zadatka:

PSO algoritam, "black-box" optimizacija

1 Opis problema

Zadata je funkcija $f(\underline{w})$ koja je definisana preko veštačke neuronske mreže, pa je možemo smatrati nepoznatom. Poznato je da je \underline{w} vektor dimenzije 60, dakle:

$$\underline{w} = (w_1, w_2, w_3, \dots, w_{58}, w_{59}, w_{60}), \quad (1)$$

i da je vrednost kriterijuma realan, jednodimenzionalan broj.

Funkcija $f(\underline{w})$ je nepoznata u analitičkom smislu (ne postoji matematička formula koja opisuje tu funkciju), ali možemo evaluirati njenu vrednost za proizvoljnu vrednost argumenata \underline{w} . U priloženoj datoteci *ann_criterion.py* se nalazi kod za veštačku neuronsku mrežu odnosno kod za izračunavanje vrednosti našeg kriterijuma optimalnosti.

Potrebno je optimizovati ulazne vrednosti koristeći PSO (Particle Swarm Optimization) algoritam kako bismo minimizovali vrednost zadate nepoznate funkcije, tj. kriterijuma optimalnosti $f(\underline{w})$.



Figure 1: Skica "Black-boxa"¹

2 Uvod

2.1 Kratak opis Particle Swarm Optimization algoritma

Particle Swarm Optimization algoritam je genetički algoritam koji donekle simulira socijalna ponašanja životinja (najviše ptica) u velikim grupama. Glavni cilj algoritma PSO je da optimizuje zadan prostor pretrage koristeći "roj" čestica u tom prostoru. Roj čestica se može još i nazvati populacijom. Svaku česticu karakteriše njen položaj u prostoru, koji je ujedno i jedno potencialno rešenje problema. Inicijalne pozicije svih čestica u prostoru se određuju nasumično. Čestice određuju poziciju u sledećoj iteraciji na osnovu trenutne vrednosti kriterijuma optimalnosti (inercioni faktor), sopstvene najoptimalnije vrednosti

¹Autor: Krauss - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=45608217>

(kognitivni faktor), najoptimalnije vrednosti u celoj populaciji (socijalni faktor). Uticaj sopstvene najoptimalnije vrednosti treba da bude veći u početnim iteracijama (čestica sama traži optimum na osnovu svojih "najboljih iskustava"), a kako teče optimizacija da slabi, a uticaj najoptimalnije vrednosti u celoj populaciji treba da bude obrnut, manji u početku a veći na kraju optimizacije (čestica se posle pridružuje ostatku populacije u pronalaženju optimuma).

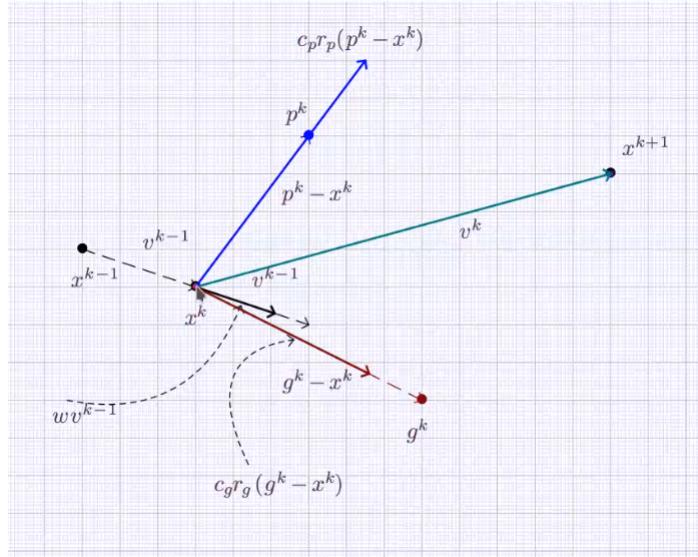


Figure 2: Skica kretanja čestice u postupku optimizacije

3 Implementacija

Uzor za implementaciju PSO algoritma je bilo predavanje profesora Rapaić na kursu Nelinearno Programiranje i Evolutivni Algoritmi.

Implementacija je odrđena u programskom jeziku **Python**, uz dve ugrađene biblioteke *math*, *random*, *time* i dodatnom bibliotekom za numeričke operacije *numpy*.

U implementaciji se koriste 3 datoteke koji sadrže potrebne klase i kod za algoritam (*options.py*, *particle.py* i *pso.py*) , priložena datoteka koja sadrži kriterijum optimalnosti (*ann_criterion.py* i skripta za pokretanje programa (*main.py*).

3.1 Implementacija pomoćnih struktura

Implementirane su klase:

- **MyOptions** - sadrži atribute za sve opcije i parametre koji su potrebni za izvršavanje optimizacionog algoritma.
- **Particle** - sadrži atribute u kojima se čuvaju vrednosti tačaka u u svim iteracijama, brzine čestice u svim iteracijama, tačka u kojoj je čestica bila najoptimalnija, kao i vrednost kriterijuma optimalnosti u toj tački i broj dimenzija prostora.
Dodatno postoje i funkcije za evaluiraće trenutne tacke u kojoj se nalazi čestica, i pomoćne funkcije koje ažuriraju ostale atribute.
- **PSO** - sadrži glavnu petlju za izvršavanje optimizacionog algoritma, a u atributima čuva vrednosti najbolje tačke i vrednosti kriterijuma optimalnosti u toj tački, populaciju (skup svih čestica) u listi.

3.2 Glavni deo programa

Glavni fajl koji povezuje sve strukture i funkcije je skripta *main.py*. U njoj se inicijalizuje objekat *Options* koji sadrži potrebne parametre za socijalni, kognitivni i inercioni faktor, zatim korisnik određuje broj iteracija i broj populacije (upisom tih vrednosti u sam fajl), pa se skripta može pokrenuti.

Na kraju izvršenja optimizacije se na konzoli ispiše optimalna tačka i njena vrednost, kao i okvirno vreme za koliko se izvršila optimizacija upotrebom modula *time*.

3.3 Klasa *Options*

Veoma je značajna klasa *Options* iz razloga što se u njoj nalaze početne i kranje vrednosti inercionog, kognitivnog i socijalnog faktora kretanja svake čestice. Profesor Rapaić je na svom predavanju o PSO algoritmu objasnio da se od nastanka algoritma eksperimentalnim putem doslo do optimalnih podešavanja za pocetne i krajnje vrednosti ovih parametra.

4 Zaključak

Puštanjem skripte da izvrši optimizaciju nekoliko puta, svaki put uz podešavanje broja iteracija i brojnosti populacije smo došli do najoptimalnijeg zabeleženog rešenja:

$$\underline{w} = (-1.705, -1.594, -8.866, -1.112, -0.347, -1.810, \\ -0.996, -1.206, 0.432, -1.427, 0.015, 0.856, \\ 2.411, -0.674, 0.318, 1.397, -0.698, 0.965, \\ 0, 7.238, -1.523, 0.028, -0.660, 0.996, \\ 1.197, -0.887, 0.036, -1.417, -0.735, 1.044, \\ -1.075, 1.648, 2.054, -0.028, -0.207, -4.721, \\ -6.096, -1.138, 0.329, -3.026, -0.700, 1.835, \\ 2.744, 0.012, 0.360, -0.873, -0.946, 0.437, \\ -2.413, -1.176, -0.404, 0.510, -1.886, 3.665, \\ -0.772, -1.806, 1.316, -0.803, -1.501, 1.126) \\ f(\underline{w}) = 0.011$$

Gde je **broj iteracija** bio **500**,
a **brojnost populacije** **300**

Uspešni rezultati optimizacije se nalaze u priloženim datotekama *optimization_first_run.txt*, *optimization_second_run.txt*, *optimization_third_run.txt*, *optimization_fourth_run.txt* i *optimization_first_run.txt*.

Kod najvećeg broja rezultata optimizacije, dobili smo slične optimume sa razlikama u redu 10^{-3} .

Pored preciznosti optimizacije, dodatan problem pravi i vreme za koje će se izvršiti optimizacija. Na ovo utiče broj iteracija i brojnost populacije. Ustanovljeno je da u slučaju optimizacije zadatog kriterijuma optimalnosti, mnogo više utiče **brojnost populacije** nego broj iteracija u brzini izvršavanja optimizacije, prepostavljamo zbog toga što je dimenzionalnost kriterijuma optimalnosti 60.

```
class Options(object):
    def __init__(self):
        self.cp_init = 2.5
        self.cp_final = 0.5
        self.cg_init = 0.5
        self.cg_final = 2.5
        self.w_init = 0.9
        self.w_final = 0.4
```

Figure 3: Isečak koda iz fajla options.py
cp - kognitivni faktor, cg - socijalni faktor, w - inercioni faktor