Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Dokumentacija za projektni zadatak

Studenti: Dušan Brkić, SW42/2018

Filip Živanac, SW66/2018

Predmet: Nelinearno programiranje i evolutivni algoritmi

Broj projektnog zadatka: 12

Tema projektnog zadatka: PSO algoritam, "black-box" optimizacija

# Opis problema

Potrebno je optimizovati funkciju koja se sastoji od 60 dimenzija zadatu preko istrenirane neuronske mreže algoritmom PSO (Particle Swarm Optimization).

# Uvod

PSO je zasnovana na imitaciji ponašanja životinjskih skupina, odnosno jedinki u tim skupinama (jata ptica i riba, rojevi insekata itd.). PSO je evolutivna, populaciona tehnika i skup tačaka (potencijalnih rešenja) posmatramo kao čestice, čije promene posmatramo kao pomeranje pozicije usred pretrage.

Svaka jedinka pamti svoju:

* Tekuću poziciju (potencijalno rešenje)
* Najbolju poziciju ikad dostignutu (najbližu rešenju)
* Tekuću brzinu

Roj, kao celina, pamti svoju najbolju poziciju ikada postignutu.

Iterativno, za k, za svaku česticu računa se brzina, nakon čega dolazi do menjanja pozicije:

Gde su:

– brzina, – lična najbolja pozicija, – trenutna pozicija, – globalna najbolja pozicija

– inercioni faktor, na početku je 1, kroz algoritam se smanjuje (kako se približavamo optimumu čestice se kreću sve sporije)

i – promenljivi faktori ubrzanja, smanjuju šanse da se algoritam zaglavi u lokalnom optimumu,

je na početku 2,5 i smanjuje se, dok je = 0,5 i kroz iteracije se povećava

i su nasumični faktor, to su brojevi od 0 do 1 koji služe da razbiju monotonost algoritma

Zaustavljanje zavisi od broja iteracija i razlike pozicija između dve iteracije.

# Implementacija

Implementacija algoritma izvršena je u programskom jeziku Python.

Čestica je implementirana kao objekat klase Particle, koji sadrži sve neophodne podatke o čestici:

class Particle:  
 def \_\_init\_\_(self, lb, ub, dim):  
 self.pozicija = np.random.uniform(low=lb, high=ub, size=dim)  
 self.brzina = np.full(dim, 0, dtype=float)  
 self.y = np.Inf  
 self.najbolji\_X = self.pozicija  
 self.najbolji\_y = self.y

PSO je implementiran kao funkcija:

def particle\_swarm\_optimisation(funkcija, lowerb, upperb, maxiter=20, npart=50, dim=60, tol=10\*\*-15, printData=False):

Početni faktori su u početku su definisani na sledeći način:

w = 1  
c1 = 2.5  
c2 = 0.5

gde su c1 i c2 - i

Njihovo menjanje vrednosti kroz svaku iteraciju je definisano ovako:

w = w \* dmp  
c1 = c1 - 2 / ((maxiter\*(maxiter-i))\*100)  
c2 = c2 + 2 / ((maxiter\*(maxiter-i))\*100)

Gde je dmp konstanta i ima vrednost 0.99, maxiter je broj iteracija zadatih kao kriterijum zaustavljanja, a i je broj trenutne iteracije. Množi se sa 100 kao ublažavajući faktor (jer će se koristiti mali broj iteracija)

Jednačina promene brzine je implementirana na sledeći način:

particles[j].brzina = w \* particles[j].brzina + c1 \* np.random.uniform(size=dim) \* (  
 particles[j].najbolji\_X - particles[j].pozicija) + c2 \* np.random.uniform(size=dim) \* (  
 globalno\_najbolji\_X - particles[j].pozicija)

i – implementirani funkcijom np.random.uniform(size=dim) koja vraća numpy.array popunjen nasumičnim vrednostima od 0 do 1 veličine dim.

Kriterijumi zaustavljanja: Broj iteracija maxiter, tolerancija (razlika izmedju novog I starog globalnog optimuma).

Parametri poziva funkcije:

* funkcija - Adresa optimizacionog problema kome tražimo optimum
* lowerb - Donja granica
* upperb - Gornja granica
* maxiter=20 - Maksimalan broj iteracija algoritma
* npart=50 – Broj čestica
* dim=60 – Broj dimenzija funkcije
* tol=10\*\*-15 – Tolerancija između novog i starog globalnog optimuma
* printData=False – Štampanje rezultata funkcije **po iteracijama**

Funkcija je pozvana sledećim parametrima:

X = particle\_swarm\_optimisation(optimality\_criterion, -10, 10, maxiter=100, npart=240, printData=True)  
y = optimality\_criterion(X)

Broj iteracija: 100, Broj čestica: 240

Skup rešenja izgleda ovako:

w = [ 1.80742666e-01 -1.10372117e+00 5.11975950e-01 -5.33745630e-03

-1.19568959e+00 1.56556570e+00 -7.36228719e-01 3.29077985e+00

3.86315141e+00 1.12269409e+00 -1.00572825e+00 -1.00217930e+00

-1.62849221e-01 2.46388179e+00 -1.94080535e+00 7.16596146e-02

4.63894133e-01 8.39053411e-01 1.59053456e+00 1.75397604e+00

5.74665062e-01 1.10151972e+00 3.43855144e+00 -6.76680105e-01

2.07458173e+00 -8.97498842e-01 -8.25382581e-01 1.86192232e+00

1.38123160e+00 -1.43990652e+00 1.58305875e+00 -8.79471908e-01

-1.21495838e-01 -5.45201410e+00 -1.44938919e+00 -2.57208477e-01

-3.66140047e+00 -3.39660535e+00 -3.26114316e+00 2.39821828e-01

-1.97043917e+00 -2.68725557e-01 -1.33554728e-01 9.51798354e-01

-1.57580122e+00 1.12926942e+00 -2.41457492e+00 2.73238334e+00

1.93922315e+00 9.91006100e-01 2.15106627e-01 -2.97468119e+00

7.11060400e-01 6.37438085e-02 1.18754267e+00 -3.25130349e+00

-1.02275240e+00 -1.49159653e+00 -2.35896927e+00 -2.11579329e+00]

F(w) = 0.026469310080646822

Potrebno vreme: 246.1701693534851 sekundi

Možemo jasno videti po iteracijama konvergiranje algoritma:

0. Globalno 0.2632371990874093

1. Globalno 0.19798161184893498

2. Globalno 0.17470038791334574

3. Globalno 0.1297970237275931

4. Globalno 0.10867053224739259

5. Globalno 0.10867049693180988

6. Globalno 0.08441726713275759

…

96. Globalno 0.026488433283730406

97. Globalno 0.026473322751268197

# Zaključak

PSO algoritam je jednostavno objasniti i implementirati, što ga čini pogodnim rešenjem za traženje globalnog optimuma optimizacionog problema.

Algoritam PSO je u velikoj zavisnosti od definisanja početnih vrednosti parametara w, cp i cg; kao i njihovog menjanja u toku iteracija.

Prilikom izbora parametara cp i cg primećeno je da, ukoliko se menjaju prebrzo, rešenje neće konvergirati. Stoga je bilo problema jer je optimizacioni problem veoma obiman i mora se dostaviti manje iteracija da bi bio efikasan. Iz tog razloga je u jednačine

c1 = c1 - 2 / (maxiter\*(maxiter-i))  
c2 = c2 + 2 / (maxiter\*(maxiter-i))

dodato \* 100 da bi u slučajevima kada je maxiter mali, a to je veoma neophodno, da bi rešenje ipak konvergiralo. U suprotnom, globalni faktor bi prerano postao prejak i tokom promene položaja čestica kroz iteracije desilo bi se da nijedna ne nadmaši globalni faktor svojim ličnim, odnosno, čestice bi se prebrzo sjatile na nasumična mesta u toku konvergencije.

Pošto se u funkciji koristi nasumični faktor, korišćenje kriterijuma zaustavljanja putem tolerancije nije potpuno pouzdano, odnosno velike su šanse da se upadne u lokalni optimum.