Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Dokumentacija za projektni zadatak

Studenti: Đurić Anastasija, SW48/2018

Petković Nenad, SW37/2018

Predmet: Nelinearno programiranje i evolutivni algoritmi

Broj projektnog zadatka: 12

Tema projektnog zadatka: PSO algoritam, “black-box” optimizacija

SADRŽAJ

[**OPIS PROBLEMA** 3](#_Toc29402540)

[**STRUKTURA PROGRAMA I IMPLEMENTACIJA** 3](#_Toc29402541)

[**UVOD** 3](#_Toc29402542)

[**PARTICLE** 3](#_Toc29402543)

[**PSO** 6](#_Toc29402544)

[**REZULTATI ALGORITMA** 9](#_Toc29402545)

# **OPIS PROBLEMA**

Koristeći programski jezik Python, potrebno je implementirati PSO (Particle Swarm Optimization) algoritam, i iskoristiti ga za pronalaženje optimalnog vektora sinaptičkih težina za datu veštačku neuronsku mrežu. Potrebno je pronaći vektor sinaptičkih težina za koji funkcija date neuronske mreže, ***optimality\_criterion***, ima minimalnu vrednost, kao i tu minimalnu vrednost koja predstavlja srednju apsolutnu grešku.

# **STRUKTURA PROGRAMA I IMPLEMENTACIJA**

## **UVOD**

Rešenje projektnog zadatka se sastoji iz dva modula, ***particle*** i ***pso***, koji su u nastavku detaljno opisani.

## **PARTICLE**

Fajl ***particle.py***sadrži implementaciju klase ***Particle***koja ima sledeće atribute:

* ***current\_position*** - vektor koji predstavlja tekuću poziciju posmatrane čestice (potencijalno rešenje). Inicijalizuje se uz pomoć biblioteke ***numpy***, njenog modula ***random*** i funkcije ***uniform*** (*numpy.random.uniform(-5, 5, 60)*), pri čemu se dobija vektor od 60 nasumično generisanih realnih vrednosti između -5 i 5.
* ***velocity*** - vektor koji predstavlja tekuću brzinu kretanja posmatrane čestice. Inicijalizuje se analogno kao i ***current\_position***, s tim što su nasumično generisane vrednosti realni brojevi između -1 i 1 (*numpy.random.uniform(-1, 1, 60*)).
* ***personal\_best*** - vektor koji predstavlja najbolju dostignutu poziciju, do odgovarajućeg trenutka (iteracije), posmatrane čestice. Prilikom inicijalizacije dodeljuje mu se vrednost atributa ***current\_position*** (na samom početku to je jedina pozicija posmatrane čestice pa je samim tim i njena najbolja). Neophodno je napraviti takozvanu “duboku” kopiju pomenutog atributa (*self.current\_position.copy()*).
* ***fitness*** - vrednost funkcije ***optimality\_criterion*** za navedeni vektor ***current\_position***. Prilikom inicijalizacije dodeljuje mu se povratna vrednost funkcije ***optimality\_criterion***, koja se poziva sa atributom ***current\_position*** kao argumentom.
* ***fitness\_best*** - vrednost funkcije ***optimality\_criterion*** za navedeni vektor ***personal\_best***. Inicijalno, ovom atributu se dodeljuje vrednost atributa ***fitness***, jer je na početku trenutna pozicija ujedno i najbolja, pa je samim tim i vrednost funkcije ***optimality\_criterion*** ista.

Prilikom kreiranja objekata klase ***Particle****,* poziva se, u njoj definisani konstruktor, pri čemu gorenavedeni atributi dobijaju svoje inicijalne vrednosti.

Klasa ***Particle***sadrži metodu ***update\_position*** koja kao parametre prima:

1. ***options*** - python rečnik koji sadrži ključeve “*num\_of\_iters*”, “*num\_of\_particles*”, “*w*”, “*cp*” i “*cg*” čije vrednosti su parametri algoritma i oni će kasnije biti detaljno opisani
2. ***global\_best*** - vektor koji predstavlja globalno najbolju dostignutu poziciju, do odgovarajućeg trenutka (iteracije), roja kao celine. To je, u stvari, najbolja pozicija koju je neka od čestica iz roja dostigla do posmatranog trenutka (iteracije)

Pozivom metode ***update\_position*** za odgovarajuću česticu, računa se nova vrednost atributa ***velocity***, uz pomoć vrednosti pomenutih parametara algoritma i atributa ***current\_position***, ***personal\_best*** i ***velocity***, koji su grupisani u tri komponente u zavisnosi od svoje uloge, na sledeći način:

self.velocity = options['w'] \* self.velocity +

options['cp'] \* rand.random() \* (self.personal\_best -self.current\_position) +

options['cg'] \* rand.random() \* (global\_best - self.current\_position)

* inercijalna komponenta - promena brzine
* kognitivna komponenta - pomeranje u pravcu lične najbolje dostignute pozicije
* socijalna komponenta - pomeranje u pravcu globalne najbolje pozicije
* poziv *rand.random()* vraća slučajnu realnu vrednost između 0 i 1

Nova tekuća pozicija predstavlja pomeranje čestice iz trenutne tačke, i računa se uz pomoć novodobijene brzine na sledeći način:

self.current\_position = self.current\_position + self.velocity

Nakon toga, vrši se provera svih vrednosti vektora tekuće pozicije, pri čemu se one koje su manje od -5 zamene sa -5, a one koje su veće od 5 zamene sa 5, da bi vrednosti vektora ostale u intervalu od -5 do 5.

Na samom kraju, računa se vrednost atributa ***fitness*** za novu vrednost tekuće pozicije, koja se nakon toga poredi sa vrednošću atributa ***fitness\_best***. U slučaju da je vrednost atributa ***fitness*** manja od vrednosti atributa ***fitness\_best***, čestica je dostigla novu najbolju tekuću poziciju, pa se atributu ***personal\_best*** dodeljuje vrednost atributa ***current\_position***, a atribut ***fitness\_best*** dobija vrednost atributa ***fitness***.

## **PSO**

U fajlu ***pso.py*** nalaze se funkcije:

* ***get\_options***

Poziva se iz ***main***-a.

***parametri:*** ova funkcija nema ulaznih parametara

***povratna vrednost:*** python rečnik sa podrazumevanim vrednostima parametara algoritma

opts = {

'num\_of\_particles': 30,

'num\_of\_iters': 30,

'w': 0.9,

'cp': 2.5,

'cg': 0.5

}

* ***create\_swarm*** - funkcija koja služi za kreiranje roja čestica. Poziva se iz funkcije ***pso***.

***parametri***: ***size*** (broj čestica roja koji se kreira, prosleđuje se vrednost ključa “*num\_of\_iters*” iz rečnika ***options***)

***povratna vrednost***: pyton lista sa zadatim brojem čestica (čestice su objekti klase ***Particle***), koja predstavja roj

* ***pso*** - ključna funkcija modula. U njoj se kreira roj. Čestice kroz zadati broj iteracija menjaju svoju poziciju u potrazi za optimumom (za svaku česticu se poziva funkcija ***update\_position***). Nakon svake iteracije ažuriraju se vrednosti parametara algoritma (pozivom funkcije ***update\_parameters***) i globalnog najboljeg rešenja (pozivom funkcije ***update\_global\_best***, ako je neka od čestica tokom te iteracije dostigla bolju poziciju od trenutne najbolje globalne). Poziva se iz ***main***-a.

***parametri:*** ***options*** (python rečnik sa parametrima algoritma)

***povratna vrednost***: vektor koji predstavlja najbolje globalno rešenje nakon poslednje iteracije i vrednost funkcije ***optimality\_criterion*** za taj vektor

* ***update\_parameters*** - funkcija koja služi za ažuriranje parametara algoritma. Poziva se iz funkcije ***pso***.

options['w'] = options['w'] - 0.5 / (num\_of\_iters + 1)

options['cp'] = options['cp'] - 2 / (num\_of\_iters + 1)

options['cg'] = options['cg'] + 2 / (num\_of\_iters + 1)

* w - faktor inercije, smanjuje se u svakoj iteraciji preko navedene formule. Njegova inicijalna vrednost je 0.9 (u prvoj iteraciji), a u poslednjoj iteraciji 0.4. Čestica se najbrže kreće dok je “mlada“. Uočeno je da se performanse algoritma (naročito u okolini rešenja) bitno poboljšavaju uvođenjem promenljivog faktora inercije.
* cp - kognitivni faktor, smanjuje se u svakoj iteraciji. Na početku, njegova vrednost je 2.5, a u poslednjoj iteraciji 0.5. Dok je mlađa, čestica se više vodi sopstvenim iskustvom. Promenljiv kognitivni faktor povećava sposobnost algoritma da pretražuje prostor pretrage.
* cg - socijalni faktor, povećava se u svakoj iteraciji. U prvoj iteraciji je 0.5 a u poslednjoj 2.5. Tokom vremena, uticaj grupnog iskustva se povećava. Promenljiv socijalni faktor povećava sposobnost algoritma da pretražuje prostor pretrage.

***parametri***: ***options*** (python rečnik sa parametrima algoritma)

***povratna\_vrednost***: ova funkcija nema povratnu vrednost

* ***update\_global\_best*** - funkcija koja služi za ažuriranje globalne najbolje pozicije. Poziva se iz funkcije ***pso***.

***parametri***:

1. ***swarm*** (python lista sa česticama)
2. ***result*** (python rečnik sa ključevima “*global\_best*” i “*global\_best\_fitness*” čije su vrednosti vektor koji predstavlja globalnu najbolju poziciju i vrednost funkcije ***optimality\_criterion*** za taj vektor)

***povratna vrednost***: ova funkcija nema povratnu vrednost

* ***print\_initial\_info*** - funkcija koja štampa informacije o globalnoj najboljoj poziciji za roj na samom početku (pre prve iteracije). Poziva se iz funkcije ***pso***.

***parametri:***  ***result*** (python rečnik sa ključevima “*global\_best*” i “*global\_best\_fitness*” čije su vrednosti vektor koji predstavlja globalnu najbolju poziciju i vrednost funkcije ***optimality\_criterion*** za taj vektor)

***povratna\_vrednost:*** ova funkcija nema povratnu vrednost

* ***print\_iteration\_info*** - funkcija koja štampa redni broj tekuće iteracije i informacije o globalnoj najboljoj poziciji za roj nakon nje. Poziva se iz funkcije ***pso***.

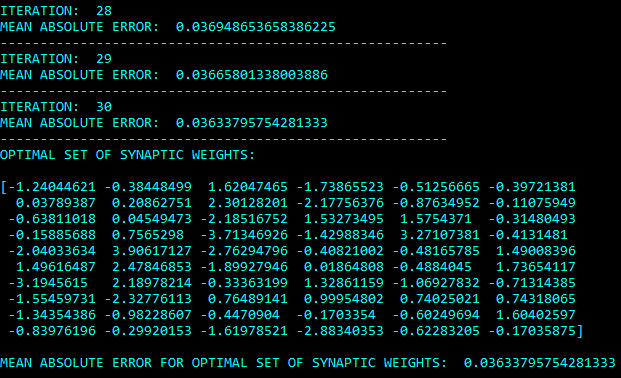
***parametri:***

1. ***iteration*** (redni broj iteracije)
2. ***result*** (python rečnik sa ključevima “*global\_best*” i “*global\_best\_fitness*” čije su vrednosti vektor koji predstavlja globalnu najbolju poziciju i vrednost funkcije ***optimality\_criterion*** za taj vektor)

***povratna vrednost***: ova funkcija nema povratnu vrednost

# **REZULTATI ALGORITMA**

Primer dobijenog rezultata nakon jednog pokretanja fajla ***main.py*** (broj čestica i iteracija je 30, a interval pretrage [-5, 5]) :



Dobijeni rezultati su zadovoljavajući. PSO se pokazao kao dobar algoritam za rešavanje problema ovog tipa.

Isprobavanjem nekoliko različitih intervala pretrage došli smo do zaključka da je, za dati problem, najpogodnije koristiti interval [-5, 5]. Povećanjem broja iteracija (npr. 100), rešenje se neznatno poboljša s obzirom na povećanje vremena izvršavanja programa. Slično, povećanjem broja čestica (npr. 60), rešenje je gotovo isto kao i za 30 čestica, s tim što se vreme izvršavanja znatno poveća.