HOMEWORK 4 FOR MODERN OPTIMIZATION METHODS

R09946006 | 何青儒 | HO, Ching-Ru | Oct. 12, 2020

1. Programming of Particle Swarm Optimization.

- (a) Notation
    - Integer:
        - `N`: Swarm Size, fixed.
        - `D`: Dimension Size, fixed.
        - `T`: Maximum Iteration times, fixed.
        - `t`: Iteration counter, from 0 to `T` times.
    - Float:
        - `c1`, `c2`: Relative importance of the global and local.
        - `r1`, `r2`: Random float between 0 and 1.
        - `x_min`: Lower bound of X.
        - `x_max`: Upper bound of X. Noticed that `x_max > x_min`
        - `pg`: Global Best ever.
        - `w`: Inertia parameter.
    - List:
        - `X = [X_{j,d,t}]`: Particle location list, with particle j in t-th iteration, size = `[N][D][T]`
        - `V = [V_{j,d,t}]`: Particle velocity list, with particle j in t-th iteration, size = `[N][D][T]`
        - `Pb = [Pb_{j}]`: particle j's historical best position during iteration, size = `[N]`
    - Function:
        - `f(・)`: Objective function, needed to maximize.
        - `g(・)`: Inertia update function (if needed)
- (b) Psuedo-code of PSO (following the formula on lecture note 04-1, P.14)

```
// We are going to find the maximun value of objective function
// INPUT AND PARAMENTERS
N <- int // Swarm size
D <- int // Dimension size
T <- int // Maximum Iteration times
t <- 0 // Iteration counter
c1 <- float
c2 <- float
x_min <- float // Lower bound of X
x_max <- float // Upper bound of X
pg <- float // A value can let objective function be small
w <- float
X <- 3D empty list // Size = [N][D][T]
V <- 3D empty list // Size = [N][D][T]
Pb <- 1D empty list // Size = [T]
```

```
// DEFINE FUNCTIONS
FUNCTION f() // Objective function
FUNCTION g() // Inertia update function

// INITIALIZATION
FOR j from 1 to N DO:
    FOR d from 1 to D DO:
        // Initial every particles' location between upper bound and lower bound randomly
        x_{j,d,0} <- RANDOM(between x_min and x_max)
        // Initial every particles' velocity with 0
        v_{j,d,0} <- 0
    END FOR
    // Iniitail particle's historical best position
    Pb_{j} <- x_{j}
    // If find a value larger than global best found before, update global position
    IF f(Pb_{j}) > f(pg) THEN
        pg <- Pb_{j}
    END IF
END FOR
t <- t + 1

// ITERATION AND TRY TO FIND GLOBAL BEST
WHILE t <= T DO:
    FOR j from 1 to N DO:
        // For any particle, if find a value larger than historical best itself found before,
update the particle's historical best
        IF f(x_{j}) > f(Pb_{j}) THEN:
            Pb_{j} <- x_{j}
        END IF
        // For any particle, if find a value larger than global best found before, update the
global best
        IF f(Pb_{j}) > f(gb) THEN:
            gb <- Pb_{j}
        END IF
    END FOR

    // Update particle's velocity and position
    FOR j from 1 to S DO:
        FOR d from 1 to D RO:
            // Generate r1 and r2 ramdomly
            r1 = RANDOM(between 0 and 1)
            r2 = RANDOM(between 0 and 1)
            V_{j,d,t} <- w_{t}*V_{j,d,t-1} + c1*r1*(Pb_{j,d}-X_{j,d,t-1}) + c2*r2*(pg-
X_{j,d,t-1}]
            X_{j,d,t} <- X_{j,d,t} + V_{j,d,t}
        END FOR
    END FOR

    // Update inertia w, if needed
    w <- g(w)
    // Update iteraiton times
    t <- t + 1
END WHILE

// OUTPUT
```

```
PRINT(gb) // Find the global best location which can let objective function be maximized.
PRINT(f(gb)) // The maximal value of objective function
```

- (c) See `hw4ass1.py`. Noticed that this program isn't completely because we don't know the objective function and other parameters.

2. Optimization of a Function via Particle Swarm Optimization.

- (a) Show detailed calculations for 4 iterations.
  - Iteration $i = 0$ (initial):
    - Initial locations: $x_1 = -2, x_2 = 0, x_3 = 1, x_4 = 3$
    - Objective Function: $f(x_1) = -53, f(x_2) = -5, f(x_3) = 19, f(x_4) = -53$
    - Initial velocities: $v_1 = 0, v_2 = 0, v_3 = 0, v_4 = 0$
    - Upload iteration number as $i = 1$
  - Iteration $i = 1$
    - Find local best locations: $P_1 = -2, P_2 = 0, P_3 = 1, P_4 = 3$
    - Find global best location: $P_G = 1, \because f(P_G) = 19$
    - Given $c_1 = c_2 = w = 1$, and set $r_1 = 0.4$ and $r_2 = 0.6$ randomly.
    - Find the velocities: $v_1 = 1.8, v = 2 = 0.6, v_3 = 0, v_4 = -1.2$
    - Update new locations:
      $x_1 = -2 + 1.8 = -0.2, x_2 = 0 + 0.6 = 0.6, x_3 = 1 + 0 = 1, x_4 = 3 - 1.2 = 1.8$
    - Evaluate the objective function:
      $f(x_1) = -9.03968, f(x_2) = 8.00224, f(x_3) = 19, f(x_4) = 41.26432$
    - Update iteration number as $i = 2$
  - Iteration $i = 2$
    - Find local best locations: $P_1 = -0.2, P_2 = 0.6, P_3 = 1, P_4 = 1.8$
    - Find global best locations: $P_G = 1.8, \because f(P_G) = 41.26432$
    - Set $r_1 = 0.2$ and $r_2 = 0.4$ randomly.
    - Find the velocities: $v_1 = 2.6, v_2 = 1.08, v_3 = 0.32, v_4 = -1.2$
    - Update new locations: $x_1 = 2.4, x_2 = 1.68, x_3 = 1.32, x_4 = 0.6$
    - Evaluate the objective function:
      $P(x_1) = 32.49378, f(x_2) = 38.92538, f(x_3) = 28.89238, f(x_4) = 8.00224$
    - Update iteration number as $i = 3$
  - Iteration $i = 3$
    - Find local best locations: $P_1 = 2.4, P_2 = 1.68, P_3 = 1.32, P_4 = 1.8$
    - Find global best locations: $P_G = 1.8, \because f(P_G) = 41.26432$
    - Set $r_1 = 0.4$ and $r_2 = 0.3$ randomly.
    - Find the velocities: $v_1 = 2.42, v_2 = 1.116, v_3 = 0.464, v_4 = -0.72$
    - Update new locations: $x_1 = 4.82, x_2 = 2.796, x_3 = 1.784, x_4 = -0.12$
    - Evaluate the objective function:
      $f(x_1) = -1950.26721, f(x_2) = -10.66760, f(x_3) = 40.99862, f(x_4) = -7.40862$
    - Update iteration number as $i = 4$
  - Iteration $i = 4$
    - Find local best locations: $P_1 = 2.4, P_2 = 1.68, P_3 = 1.784, P_4 = 1.8$
    - Find global best locations: $P_G = 1.8, \because f(P_G) = 41.26432$
    - Set $r_1 = 0.6$ and $r_2 = 0.8$ randomly.
    - Find the velocities: $v_1 = -1.448, v_2 = -0.3504, v_3 = 0.4768, v_4 = 1.776$
    - Update new locations: $x_1 = 3.372, x_2 = 2.4456, x_3 = 2.2608, x_4 = 1.656$
    - Evaluate the objective function:
      $f(x_1) = -181.80647, f(x_2) = 29.56347, f(x_3) = 38.93084, f(x_4) = 38.37274$
    - Update iteration number as $i = 5$ (omitted).
- (b) Find the optimal solution using your program in Question 1

- See `hw4ass2.ipynb` files.
- Terminal output:

```
Iteration times:  0  | Best of objective function:  [19.]
Iteration times:  1  | Best of objective function:  31.672610325881855
Iteration times:  2  | Best of objective function:  41.67564413277344
Iteration times:  3  | Best of objective function:  42.879238998917
Iteration times:  4  | Best of objective function:  42.879238998917
Iteration times:  5  | Best of objective function:  42.879238998917
Iteration times:  6  | Best of objective function:  42.99968366190463
Iteration times:  7  | Best of objective function:  42.99968366190463
Iteration times:  8  | Best of objective function:  42.99968366190463
Iteration times:  9  | Best of objective function:  42.99968366190463
Iteration times:  10  | Best of objective function:  42.99968366190463
Iteration times:  11  | Best of objective function:  42.99968366190463
Iteration times:  12  | Best of objective function:  42.99968366190463
Iteration times:  13  | Best of objective function:  42.99968366190463
Iteration times:  14  | Best of objective function:  42.99968366190463
Iteration times:  15  | Best of objective function:  42.99968366190463
Iteration times:  16  | Best of objective function:  42.99968366190463
Iteration times:  17  | Best of objective function:  42.99968366190463
Iteration times:  18  | Best of objective function:  42.99968366190463
Iteration times:  19  | Best of objective function:  42.99968366190463
Iteration times:  20  | Best of objective function:  42.99968366190463
```

3. Firefly Algorithm.

- (a) pseudocodes
  - Concept:
    1. Every firefly in swarm will attract others.
    2. Attraction and light intensity are in direct proportion. The darker firefly will be tend to move to the lighter one. However, the light intensity will decrease while the distance is increasing.
    3. If there have no lightest firefly, every firefly in swarm will move randomly.

```
T <- Max Iteration
N <- Fireflies size
L[] <- list of light intensity, size = [N]

// DEFINE FUNCTION
FUNCTION f() // Objective function

// INITIALIZATION

// ITERATION AND TRY TO FIND GLOBAL BEST
WHILE t <= T DO :
    FOR i from 1 to N DO:
        FOR j from 1 to i DO: //(c)
            IF L[i] > L[j] THEN:
                //move fireflies i towards fireflies j
            END IF
        //evaluate new solution and update light intensity
        END FOR //(c)
    ENF FOR
    //rank the fireflies and find the current best
    t <- t + 1
END WHILE

// OUTPUT
```

- (b) In Fireflies algorithm, particles (fireflies) move depend on its light intensity (some value which is proportional to value of objective function) and other particles'. However, in PSO the particle is compare to its best location in historical data.
- (c) Yes, if we remove the loop in line 13 of psuedo code, i.e., `FOR j from 1 to i DO: //(c)`, and replace `L[j]` as global best `gb`, then firefly algorithm essentially becomes the standard PSO.