Artificial Intelligence

Assignment 1

Evolutionary Algorithms

Submitted By
Muhammad Anas
BSCS201949

Submitted To
Dr Junaid Akhtar

Table Of Contents

Abstract:

This report focuses on a problem solving technique called evolutionary algorithm. This report discusses the problem of recognition of single face from a variety of templates by evolutionary algorithms. In this report, we will study the Darwin theory of evolution and other biological processes like reproduction and natural selection and how natural processes help us to solve problems and develop applications more efficiently. This report contains the components of evolutionary algorithms and their importance and their connection to the natural biological processes.

Background:

Before solving the problem of face recognition, we discussed many possible solutions for this problem like sequential search and other exhaustive searches. But our goal is to solve the problem by a powerful and optimized algorithm. To achieve this goal we studied the Darwin theory of evolution to understand the behaviour of nature to tackle the problem and solve it. Let's have a look at Darwin's work and his conclusions about the behaviour of nature.

- **Natural Phenomena:**
  - ➔ **Struggle for Existence:**

    First observation of Darwin is overproduction, for example each year thousands of fish eggs are produced each year but few of them are able to survive. Many of them are food for predators.
  - ➔ **Variation:**

    Second observation of Darwin is variation which means that there is variation in class of the same species. For Example, zebras show variation in pattern and color of their stripes. Goldfish have gold scales, orange scales, or brown scales, or a mixture of all three.
  - ➔ **Selection:**

    All organisms compete with the natural requirements and conditions but nature selects only those which are best and reject all other organisms.For Example, giraffes with a long neck will survive and short neck giraffes will reject by nature by the passage of time.

- **Introduction to Darwin Theory:**

Charles Robert Darwin was an English naturalist, geologist and biologist,best known for his contributions to the science of evolution. His proposition that all species of life have descended from common ancestors is now widely accepted and considered a fundamental concept in science.

Darwin travelled around South America for five years to explore the secrets of nature. He observed the classification of animals and birds with respect to time and place. He noted these behaviors and collected the samples (fossils) and studied these samples for several years and compiled a book named “Darwin theory of evolution”. On the basis of Darwin's observations he conclude his theory of evolution in following points :

- Variation of Population
- Inherited Traits
- Offspring Compete
- Survival of fittest

Relation Between Evolutionary Algorithm And Natural Reality(Darwin theory):

Evolutionary algorithms are derived from Darwin theory of evolution. In evolutionary algorithm we do the following steps:

- **Initializing the random population**:

  First of all we generate a random population. Each individual of the population is supposed to be the solution. As in Darwin theory, variation means diversity in nature so in our random population, we have varieties of solutions.

- **Fitness of Population:**

  As we have a number of random individuals but there are few which compete with nature so we have to find the fitness value of each individual by the correlation method.

- **Selection:**

  According to Darwin theory, nature selects only fittest individuals. Similarly in evolutionary algorithms we sort the best individuals and bad individuals.

- **Evolution:**

  According to Darwin theory the selected individuals evolve more than the rejected ones . In evolutionary algorithms, crossover and mutation processes help to evolve the population.

- **Continuity of processes:**

  This process of creating a new generation performs again and again to evolve the population more efficiently to get the best solution. As in Darwin theory, the exponential increase in population is reduced by the process of selection and evolution.

Computational Model:

I solve the template matching problem by evolutionary algorithm which consists of following functions:

- **Image Reading:**

  By using the “from matplotlib.image import imread” , first of all I convert the image into a 2D array.

- **InitializePop:**

  In a given array, by using “np.random.randint” , I generate the random number of population.

```
def initilizePop(Rows,Columns,Size):
    Row_List=np.random.randint(Rows,size=(Size))
    Col_List=np.random.randint(Columns,size=(Size))
```

```
    pop= [(Row_List[i],Col_List[i]) for i in range
(0,len(Row_List))]
    return(pop)
```

- **Fitness Evaluation:**

  Then I find the fitness value of each random population by correlation formula which is :

```
co_relation=np.mean((num-num.mean())*(image_array_2-image_array_2.me
an()))/(num.std()*image_array_2.std())
```

- **Selection:**

  This selection function ranks the initial population with respect to fitness formula means here sorting occurs which will be used in the next step for selection.

```
def Selection(list1,list2):
    Selected_List=[]
    pairs=[]
    pairs = sorted(zip(list1, list2),reverse=True)
    for i in pairs:
        Selected_List.append(i[0])
    return (Selected_List)
```

- **CrossOver:**

  For the evolution process, I used the crossover method which takes two parent points and generates two new points by swapping the half part of the input parent with each other.

```
p=np.random.randint(1,len(a)-1)
        for i in range(p,len(a)):
            a[i],b[i]=b[i],a[i]
        # print(list2)
        # print(list1)

        a,b = ''.join(a),''.join(b)
```

```
            # return a ,b
        New_Gen.append((int(a[0:10],2),int(a[10:],2)))
        New_Gen.append((int(b[0:10],2),int(b[10:],2)))
```

- **Mutation:**

  For the evolution process, I used the concept of mutation which actually takes one parent point from an evolved population and then changes one bit of this to zero or one, which creates the new generation.

```
for i in range(0,len(List_Gen_1)):
        a=list(List_Gen_1[i])

        p=np.random.randint(1,len(a)-1)
        One='1'
        Zero='0'
        for j in range(0,len(a)):
            if (j==p):
                if (a[j]==One):
                    a[j]=Zero
                    a=''.join(a)
                    Mutated_Gen.append((int(a[0:10],2)))

                else:
                    a[j]=One
                    a=''.join(a)
                    Mutated_Gen.append((int(a[0:10],2)))
    Mutated_Gen=[(Mutated_Gen[i],Mutated_Gen[i+1]) for i in
range(0,len(Mutated_Gen),2)]
    return Mutated_Gen
```

- **Creating and Evolving Generations:**

  To get the desired value or template I repeat this actions again and again upto 5000 times, meaning maximum 5000 upto generations.

```
# for i in range(5000):
#     for j in range(0,len(fitness_2)):
```

```
#         if (fitness_2[j]>0.85):

#             print("Babe Ki Boothi recognized at" , i,"th" ,
"generation")
#             temp = j
#             break
#             # return()
#     if temp != True:
#         break


#     Ranked_Gen_2=Selection(Gen_2,fitness_2)

#     mean.append(sum(fitness_2)/len(fitness_2))
#     fitness_2.sort()
#     max.append(fitness_2[0])



#     Gen_1_1=CrossOver(Ranked_Gen_2)

#     Gen_2=Mutation(Gen_1_1)

#     fitness_2=FitnessEvaluation(image_array_1,image_array_2,Gen_2)
# print(mean)

# if temp != True:
#     x=Gen_2[temp][0]
#     y=Gen_2[temp][1]
```

- **Stopping Criteria:**

  Then if I get the desired result in given generations, then by using the "break" function, the process should be stopped. And then show the image having true point.

```
if (fitness_2[j]>0.85):

#             print("Babe Ki Boothi recognized at" , i,"th" ,
"generation")
#             temp = j
#             break
#             # return()
#     if temp != True:
#         break
```

Experiments:

Experiment No 1:

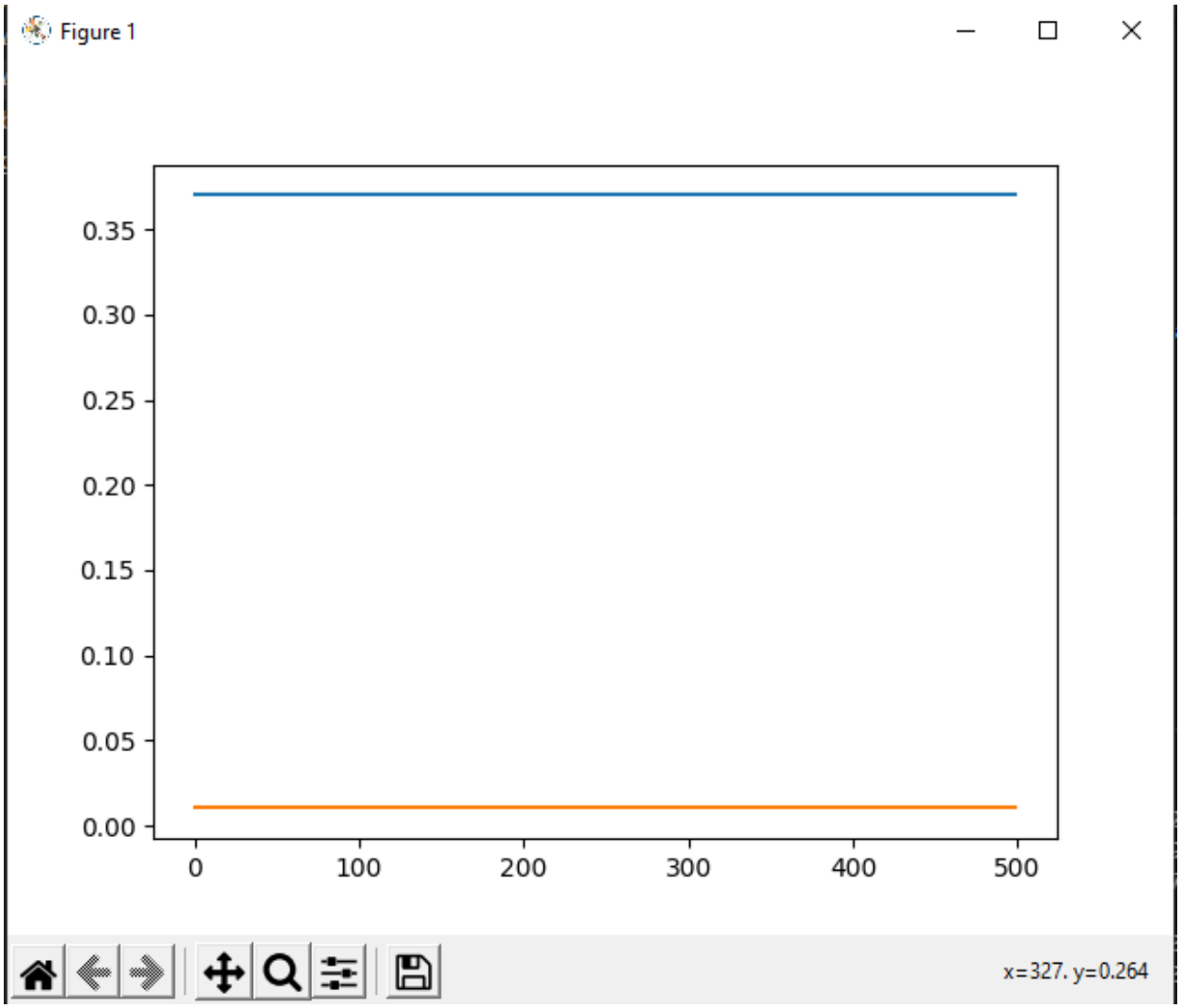
Hypothesis:

While doing programming, I thought that maybe I got the true point on my first randomly created population. So I execute the program several times on the first randomly initialized population. Then I got the following results.

Result:

After trying several times, I do not get the fitness value more than 0.2.

Threshold Value >0.85



Experiment No 2

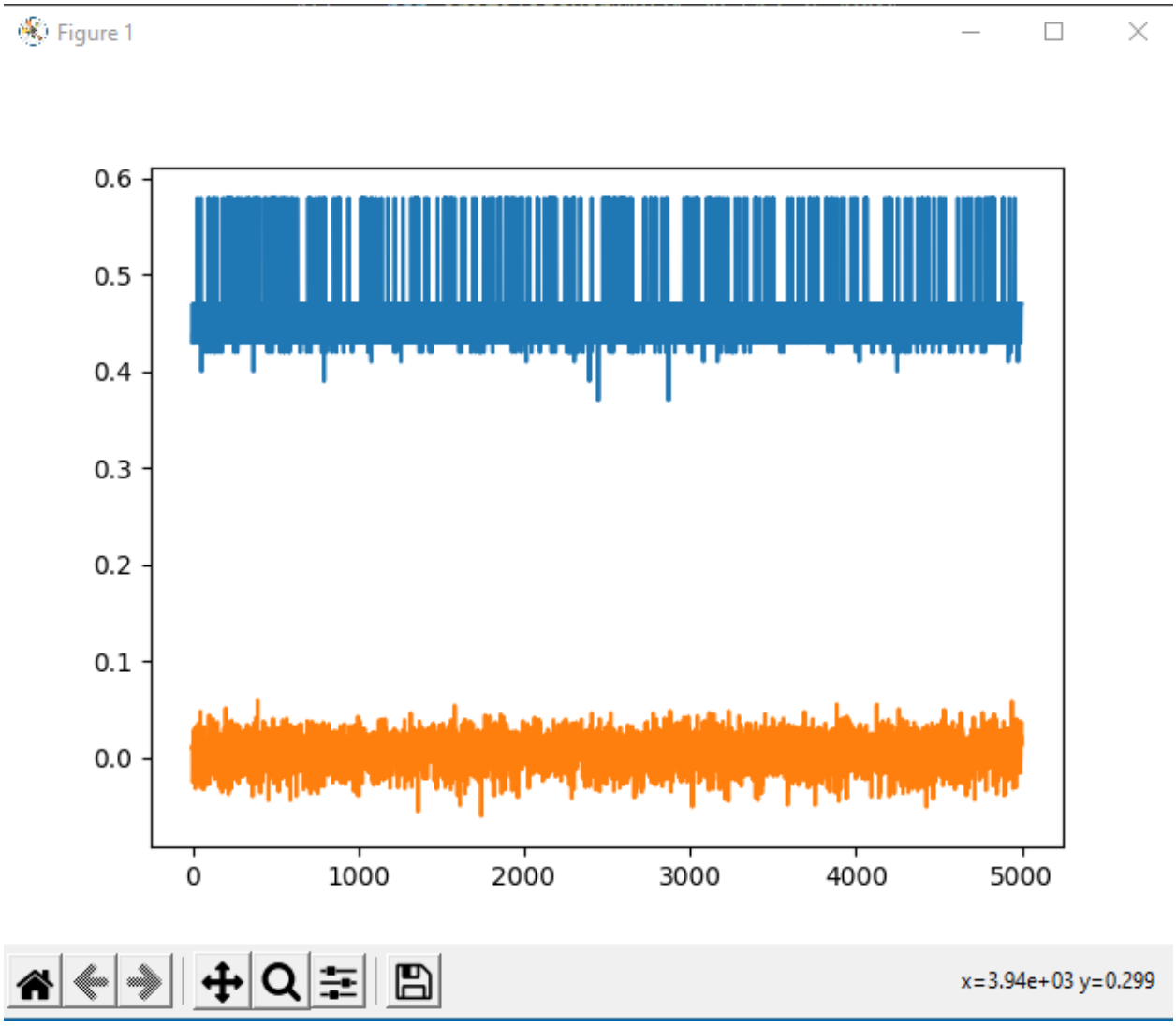
Hypothesis:

While doing the crossover method , I want to get the true value just by passing crossover generation several times without mutation. Then I got the following results.

Result:

I passed the crossover function 10 times to generate the new generation and still don't get the true value.. Which means that only passing crossover function , algorithms cannot be efficient.

True Value >0.85



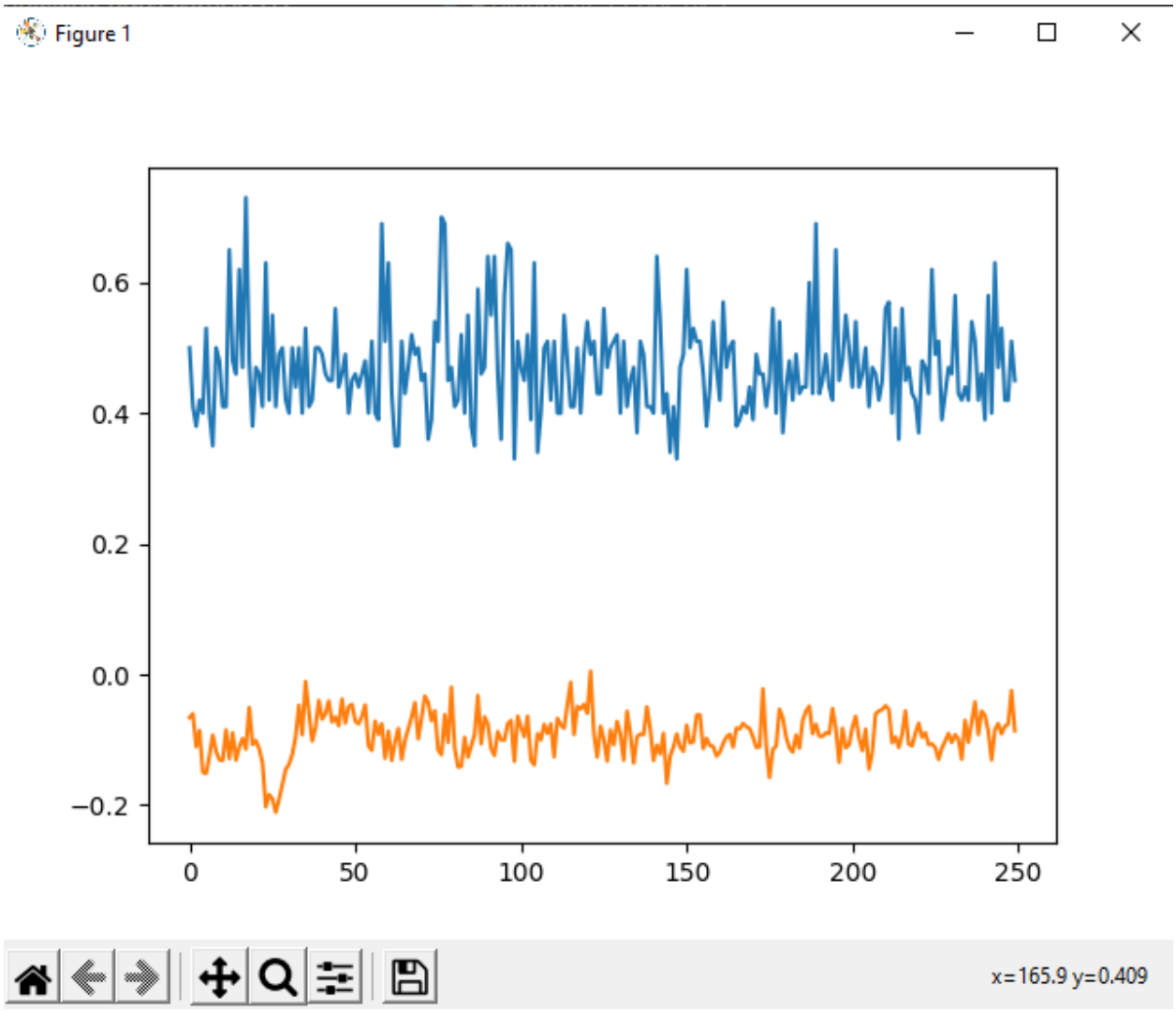
Experiment No 3

Hypothesis:

After getting results from crossover, I used the mutation function to efficient and optimize my algorithm. Then I pass the mutation function 5 times and generate the new generations upto 500.
True Value >0.85

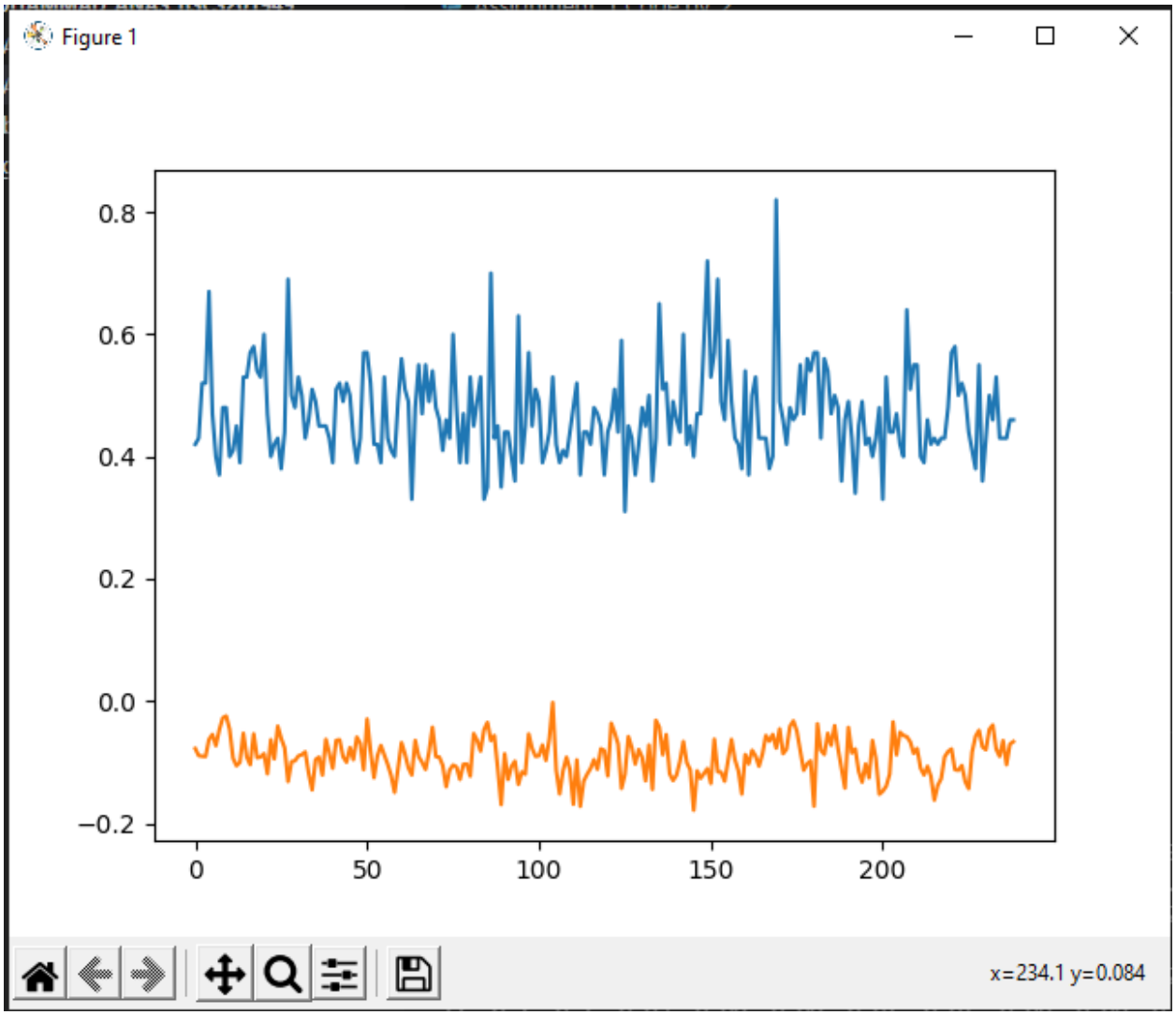
Result:

Luckily, I get the true values at each time and the size of the random population is 100.
At Generation: At 250th Generation



At Generation 239th:

True Value >0.85



Hypothesis No 4:

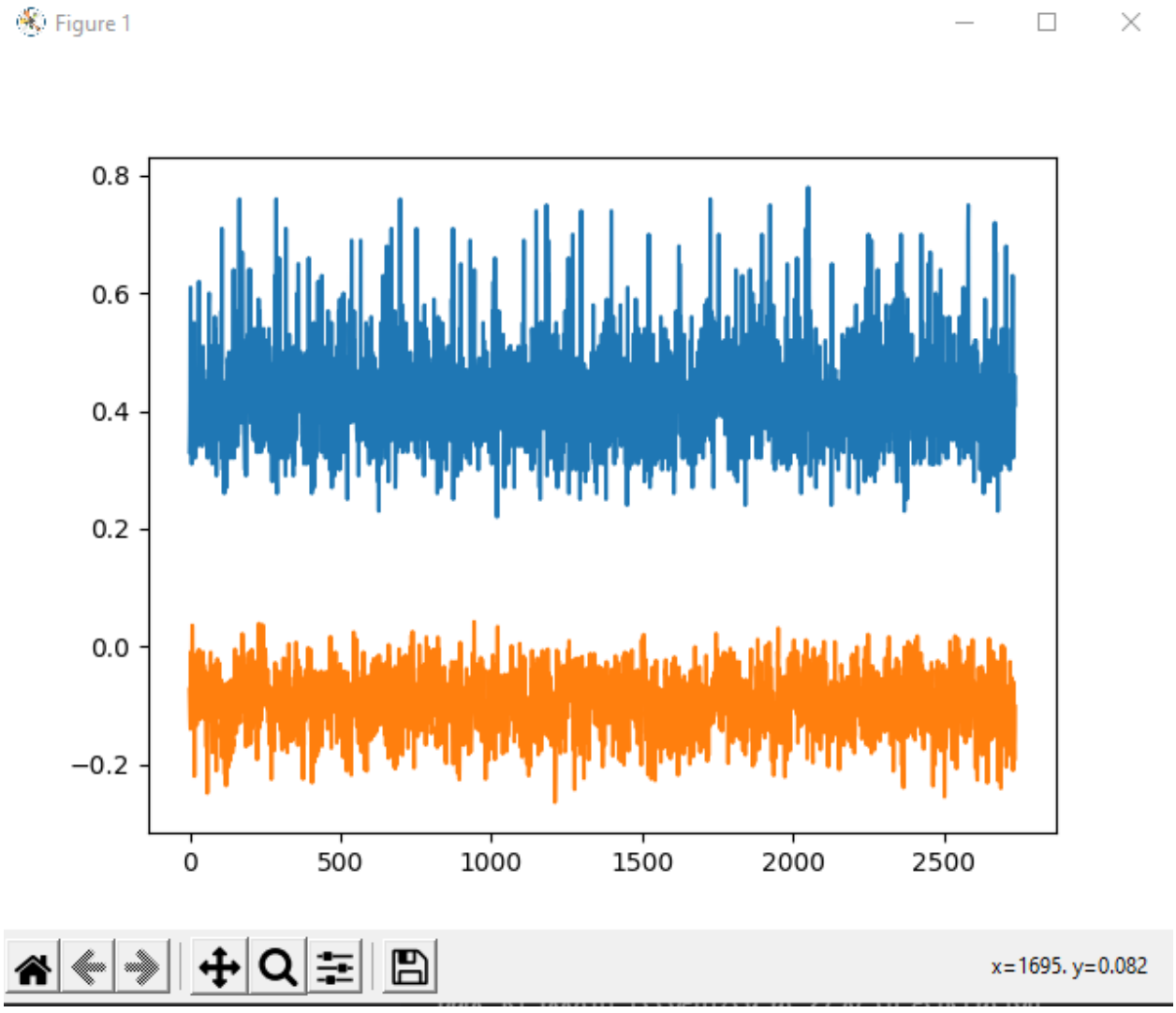
In the same step I reduced the size of the random population to 50. Then I get the following result.

Result:

By reducing the size of population to 50, I get the true value only one time.

At 2736th Generation:

True Value >0.85



Hypothesis No 4:

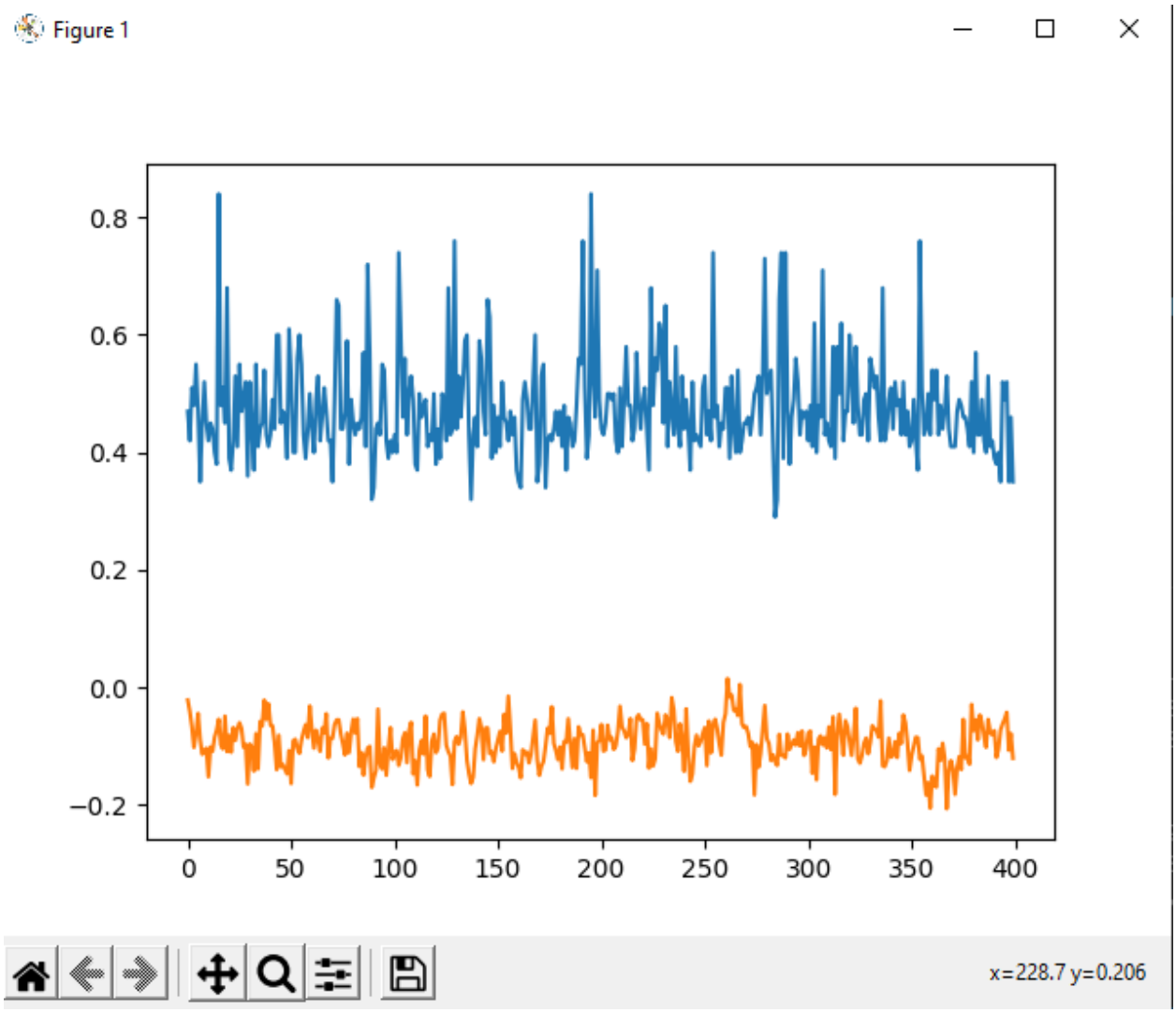
If i do not execute the crossover step and directly execute the mutation step on ranked pop.

Result:

When the size of the pop is 50, I execute the program 10 times , but it does not give me true value. But when the size of pop increased to 100 then it gave me true value one times.

At 400th Generation:

True Value >0.85



Output of Algorithm Is:

It gives me an immense pleasure that after evolving my random population by crossover and mutation function, upto 90% it gives me the true value at threshold > 0.85.

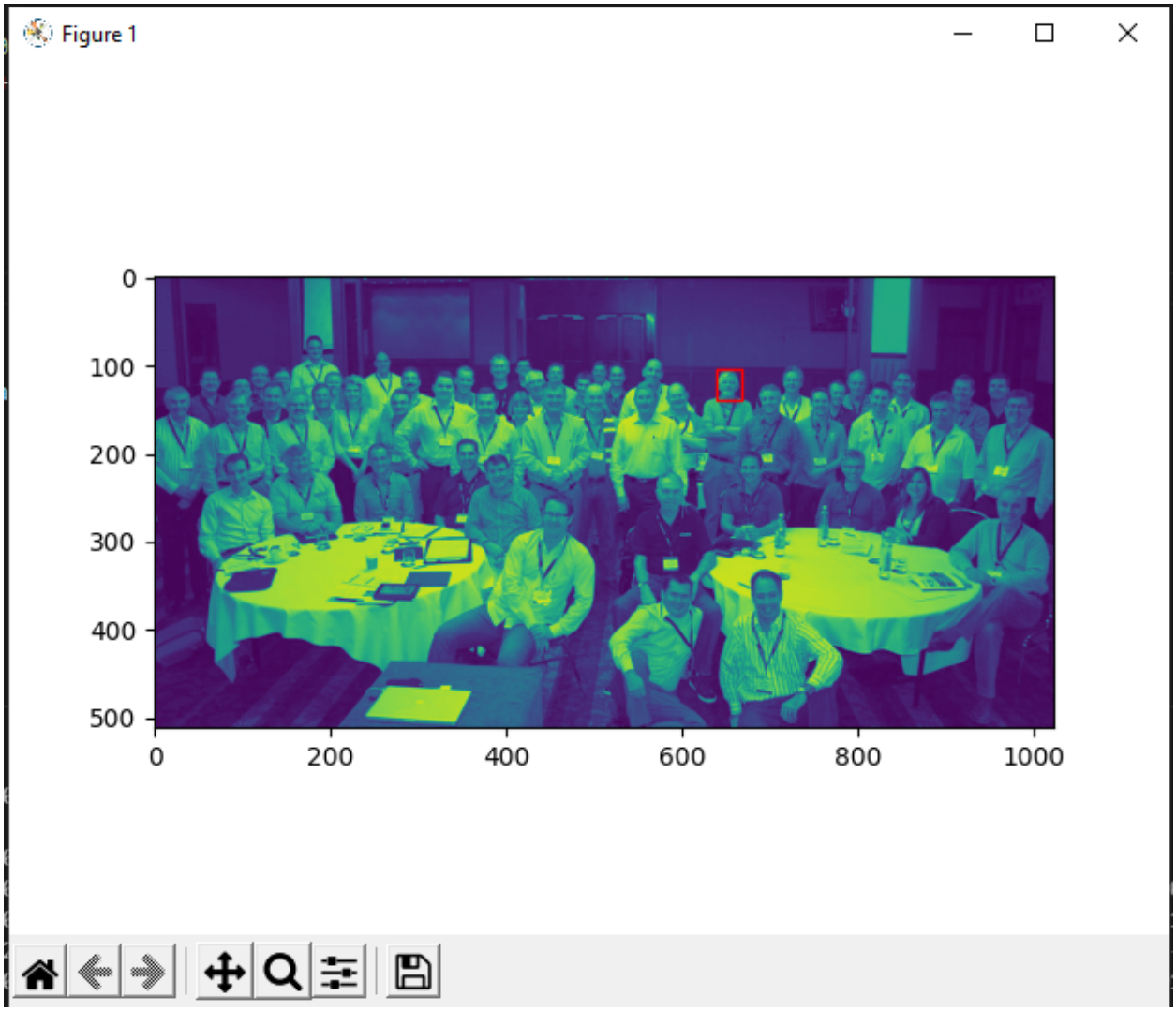


Figure 1
0
100
200
300
400
500
0
200
400
600
800
1000