IIAI30017

Artificial Intelligence

HW2: Genetic Algorithm

Instructor: YuanFu Yang

yfyangd@nycu.edu.tw

國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

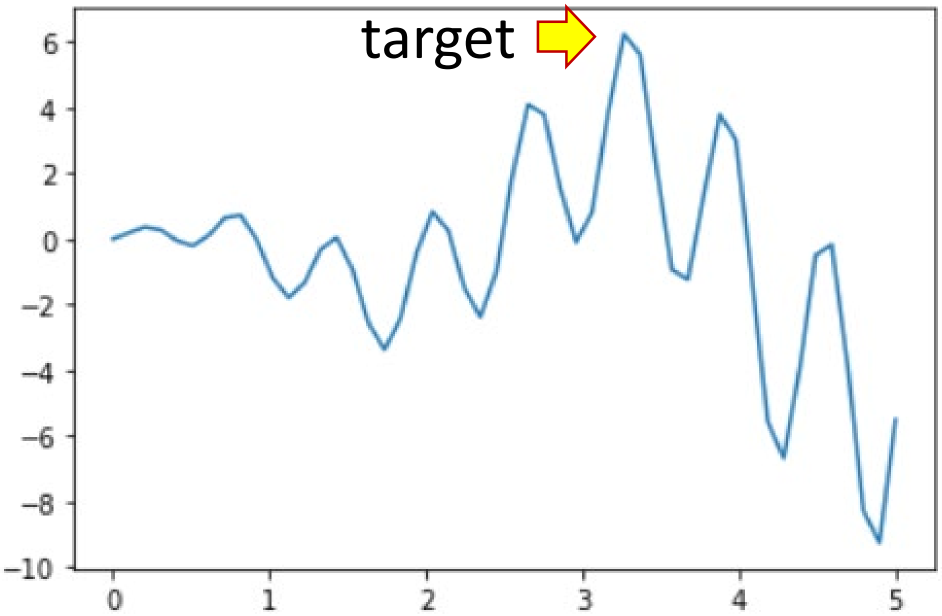
Genetic Algorithm

- Homework due: 10/22
- Late submissions will incur a penalty of one point for each day overdue.
- The assignment allows a maximum extension of 3 days (it will not be accepted if submitted later than 3 days).
- Submit files: code and report (10 questions), and submit them in both **.ipynb** and **PDF** file formats respectively.
- This assignment can be carried out using Colab or completed on your PC.

Genetic Algorithm

- The assignment is to design a Genetic Algorithm to solve a mathematical maximization problem. The target function is as follows:

$$F(x) = \sin(10x) \cdot x + \cos(2x) \cdot x$$



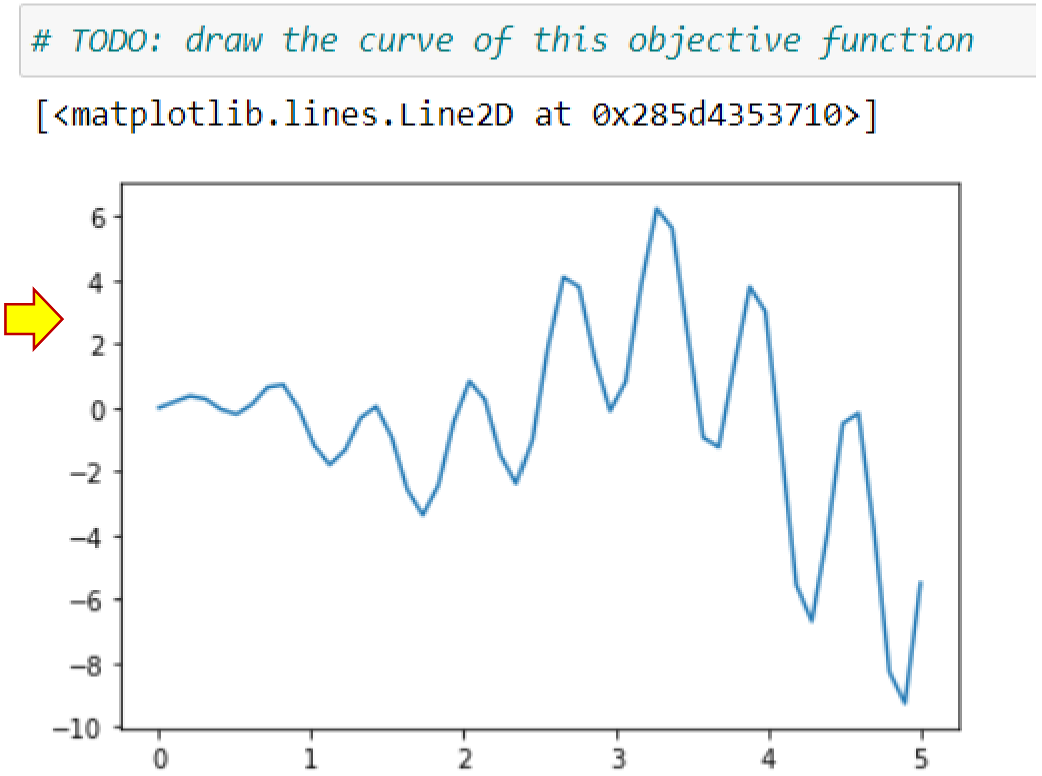
- Follow the instructions in the "Genetic Algorithm.jpynb" file, complete the 10 problems, and write the related code.

Genetic Algorithm

- HW3.1 Target Function

From x=0~5, draw the curve of this objective function

Expect Result



```
# TODO: draw the curve of this objective function
```

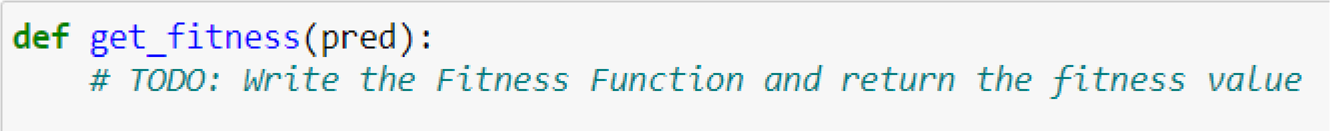
```
[<matplotlib.lines.Line2D at 0x285d4353710>]
```

Genetic Algorithm

- HW3.2 Design Fitness Function

Based on the following formula, design a Fitness Function.
You can also design other fitness functions.

$$fitness(pred) = \mathrm{pred} + 10^{-3} - \min(\mathrm{pred})$$



```
def get_fitness(pred):
    # TODO: Write the Fitness Function and return the fitness value
```

Genetic Algorithm

- HW3.3 Representation

Design a function translateDNA(pop) to convert binary-encoded DNA (gene sequences) into a real number, used in genetic algorithms to map binary genes to a value within a specified range.

Input:

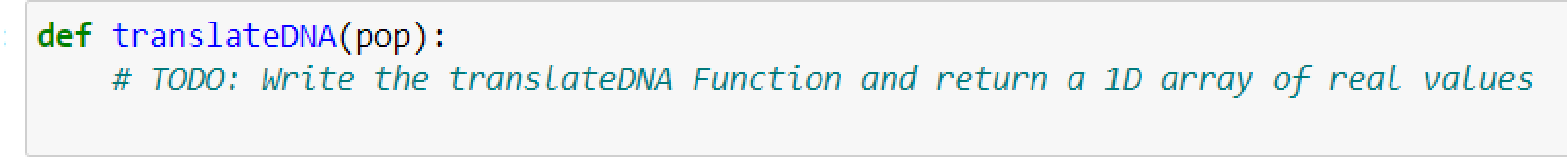
- pop: A 2D array where each row represents an individual's DNA, encoded as a binary sequence.

Output:

- A 1D array of real values, where the binary DNA has been converted to real numbers and scaled according to X_BOUND[1].

Explanation:

- The function converts the binary DNA representation into a real number by treating the binary sequence as a number in base-2, normalizing it by dividing by the maximum possible value, and scaling it to fit within the specified bounds (X_BOUND[1]).

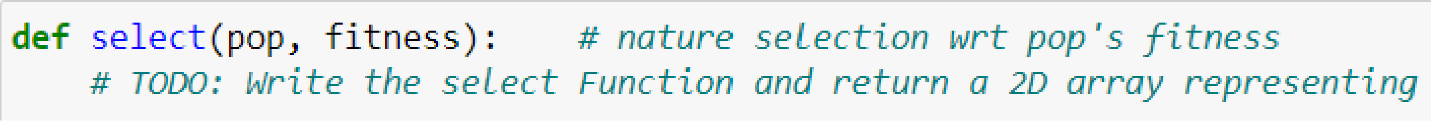


```
def translateDNA(pop):
    # TODO: Write the translateDNA Function and return a 1D array of real values
```

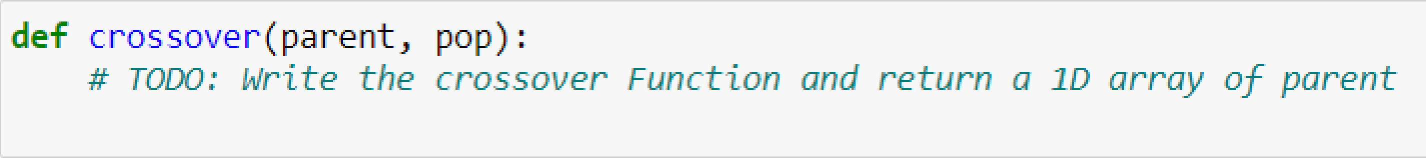
Genetic Algorithm

- HW3.4 Selection/ HW3.5 Crossover / HW3.6 Mutation

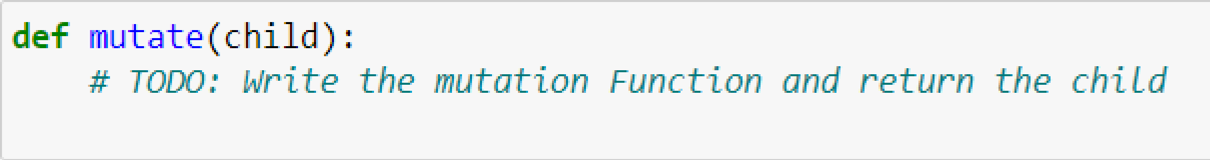
Please follow the instructions in the "Genetic Algorithm.ipynb" file and complete the basic functions in the genetic algorithm: Select(), Crossover(), and Mutate().



```python
def select(pop, fitness):    # nature selection wrt pop's fitness
    # TODO: Write the select Function and return a 2D array representing
```



```python
def crossover(parent, pop):
    # TODO: Write the crossover Function and return a 1D array of parent
```



```python
def mutate(child):
    # TODO: Write the mutation Function and return the child
```

Genetic Algorithm

- ## HW3.7 GA Function

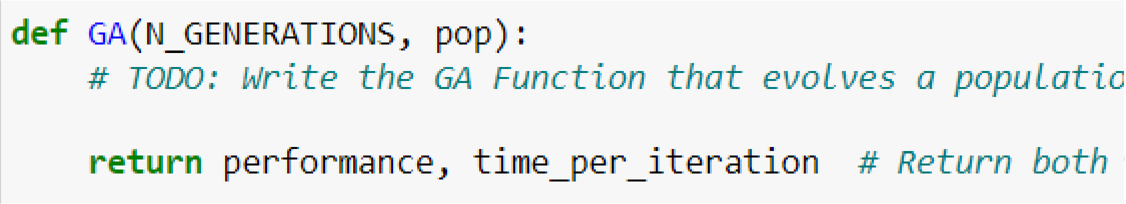
Write a function GA() that implements a basic genetic algorithm (GA) that evolves a population over many generations to optimize the function F() based on the function you wrote above.

Input:

- N_GENERATIONS: An integer representing the number of generations (iterations) the genetic algorithm will run.
- pop: A 2D array where each row represents an individual's DNA sequence encoded in binary (the initial population).

Output:

- performance: A list that records the sum of fitness values (F_values.sum()) for each generation. This allows the user to analyze how the overall fitness of the population evolves across generations. Additionally, for each generation, a scatter plot is generated to visually represent the solutions' progression.
- time_per_iteration: A list used to record the computation time for each generation.



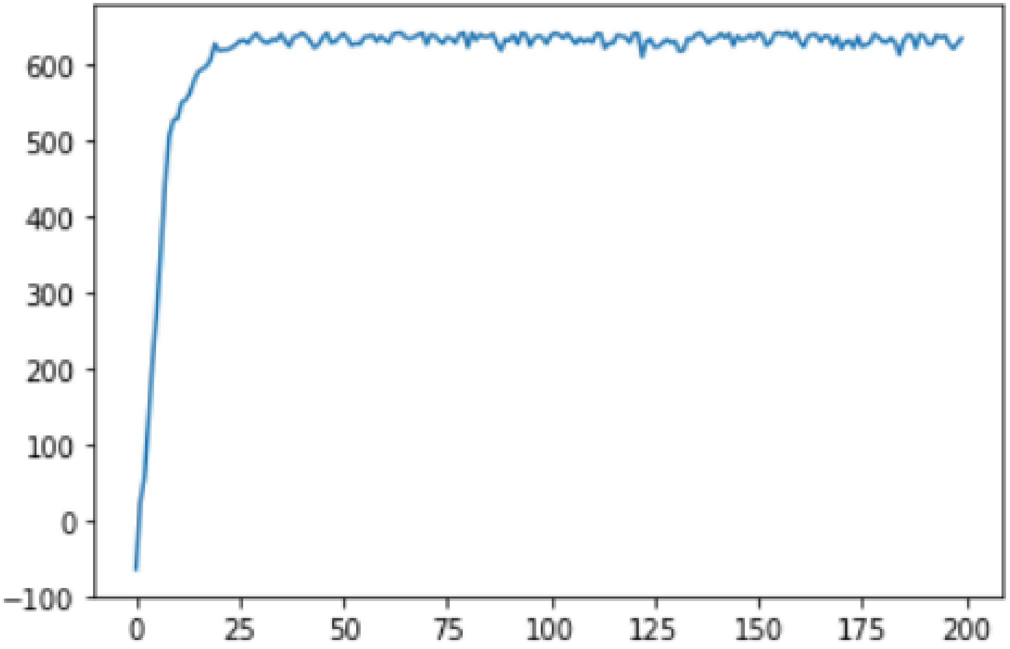
```
def GA(N_GENERATIONS, pop):
    # TODO: Write the GA Function that evolves a populatio

    return performance, time_per_iteration  # Return both
```

Genetic Algorithm

- HW3.8 Performance Visualization

Generates a line plot using matplotlib or other tool to visualize the performance of the genetic algorithm over multiple generations.for each generation.

Expect Result



Genetic Algorithm

- HW3.9 Discuss 1

  Increase the mutation rate, follow the above steps, observe the performance, and provide a discussion.

- HW3.10 Discuss 2

  Based on the time taken for each generation, do you think GA is an efficient algorithm? Please compare it with traditional heuristic algorithms and modern machine learning methods in your explanation.

IIAI30017

Artificial Intelligence

HW2: Genetic Algorithm

Q & A

Instructor: YuanFu Yang

yfyangd@nycu.edu.tw