Artificial Intelligence Systems Course

Outfit Synthesis

Using Genetic algorithm

2nd Semester 1440-1441 H

Project Detailed Description:

1) Problem Definition:

You are invited to an event but not sure what to wear, which happens 99% of the times. You stared at your wardrobe for a while, then decided to go shopping! You went to your favorite store and found the stock in the table below.

CATEGORY	PIECE	COLOR	DRESS CODE	PRICE
TOP	t-shirt	Dark, bright	casual, sportswear	0.0 SAR
	blouse	bright	Business, evening	200.0 SAR
	bodysuit	dark	casual, evening	150.0 SAR
	sleeveless	dark	casual	110.0 SAR
	tank	bright	casual, sportswear	70.0 SAR
	sweater	dark	casual, business	200.0 SAR
	vest	dark	business	300.0 SAR
	blazer	dark	business	430.0 SAR
	jacket	bright	casual	0.0 SAR
	hoodie	bright, dark	sportswear	230.0 SAR
	cardigan	bright	casual	300.0 SR
BOTTOM	jeans	dark	casual	150.0 SAR
	knee length pant	bright	casual	220.0 SAR
	ankle length pant	dark	business	0.0 SAR
	high waist pants	bright	business	150.0 SAR
	legging	dark	casual	100.00 SAR
	sweatpants	bright	casual	100.0 SAR
	wide leg pants	dark, bright	business, evening	500.0 SAR
	maxi skirt	bright	evening	500 .0SAR
	midi skirt	dark	business	0.0 SAR
	short skirt	bright	casual	400.0 SAR
SHOES	sandals	dark	casual, evening	120.0 SAR
	sneakers	bright	sportswear, casual	300.0 SAR
	high heel	dark	evening	0.0 SAR
	mid heel	bright	casual, business	400.0 SAR
	low heel	dark	business	150.0 SAR
	flat	bright	casual	0.0 SAR
	boots	dark	casual	500.0 SAR
NECK	necklace	dark	business, evening	150.00 SAR
	choker	bright	casual, evening	0.0 SAR
	scarf	bright	casual	250.0 SAR
	tie	dark	business	100.0 SAR
	bow tie	dark	business	100.0 SAR
HANDBAG	backpack	bright	sportswear	100.0 SAR
	purse	bright	business	600.0 SAR
	clutch	dark	evening	500.0 SAR
	belt bag	dark	casual	300.0 SAR
	cross bag	dark	Business	0.0 SAR

2) Project Goal:

The goal is to optimize the final look, which consists of pieces from all categories (top, bottom, shoes, neck, and purse) given a dress code, color pallet, and budget. Since there is a limited budget, you are not sure which piece you should buy and which you should get from your own wardrobe. In the table above, pieces with 0.0S AR prices indicate wardrobe pieces.

3) Project Solution:

Implement a Genetic algorithm to find the best outfit. The code works as follows; the user enters a dress code (options include casual, sportswear, business, and evening), color pallet (dark or bright), and budget=[SAR 0.0 – SAR∞]. the GA will then run to find the best pieces from each category that maximize the fitness function and output the items. The fitness function includes three parameters: dress code, color pallet, and budget. The dress code and budget are equally important, and the color pallet is half important. The fitness function can be designed as weighted sum and importance of factors can be encoded as weights w_i in the fitness function, where $\Sigma w_i = 1$. And the fitness function value must be in the range [0,1].

The solution implemented in 8 phases which are:

1- Choose a solution representation.
2- Initiate a population.
3- Using Fitness function.
4- Using a Selection method.
5- Crossover.
6- Mutate.
7- Using a Replace method.
8- Terminate.

In the next pages, the 8 phases will be described in detail.

1- Solution representation:

```
population = [["Top", "t-shirt", "dark", "casual", 0.0],
              ["Top", "t-shirt", "bright", "casual", 0.0],
              ["Top", "t-shirt", "dark", "sportswear", 0.0],
              ["Top", "t-shirt", "bright", "sportswear", 0.0],
              ["Top", "blouse", "bright", "Business", 200.0],
              ["Top", "blouse", "bright", "evening", 200.0],
              ["Top", "bodysuit", "dark", "evening", 150.0],
              ["Top", "bodysuit", "dark", "casual", 150.0],
              ["Top", "sleeveless", "dark", "casual", 110.0],
              ["Top", "tank", "bright", "casual", 70.0],
              ["Top", "tank", "bright", "sportswear", 70.0],
              ["Top", "sweater", "dark", "casual", 200.0],
              ["Top", "sweater", "dark", "Business", 200.0],
              ["Top", "vest", "dark", "Business", 300.0],
              ["Top", "blazer", "dark", "Business", 430.0],
              ["Top", "jacket", "bright", "casual", 0.0],
              ["Top", "hoodie", "bright", "sportswear", 230.0],
              ["Top", "hoodie", "dark", "sportswear", 230.0],
              ["Top", "cardigan", "bright", "casual", 300.0],

              ["Bottom", "jeans", "dark", "casual", 150.0],
              ["Bottom", "knee length pant", "bright", "casual", 220.0],
              ["Bottom", "ankle length pant", "dark", "business", 0.0],
              ["Bottom", "high waist pants", "bright", "business", 150.0],
              ["Bottom", "legging", "dark", "casual", 100.0],
              ["Bottom", "sweatpants", "bright", "casual", 100.0],
              ["Bottom", "wide leg pants ", "dark", "business", 500.0],
              ["Bottom", "wide leg pants ", "bright", "business", 500.0],
              ["Bottom", "wide leg pants ", "bright", "evening ", 500.0],
              ["Bottom", "wide leg pants ", "dark", "evening ", 500.0],
              ["Bottom", "maxi skirt", "bright", "evening ", 500.0],
              ["Bottom", "midi skirt", "dark", "business", 0.0],
```

```
              ["Shoes", "sandals", "dark", "casual", 120.0],
              ["Shoes", "sandals", "dark", "evening", 120.0],
              ["Shoes", "sneakers", "bright", "sportswear", 300.0],
              ["Shoes", "sneakers", "bright", "casual", 300.0],
              ["Shoes", "high heel", "dark", "evening", 0.0],
              ["Shoes", "mid heel", "bright", "casual", 400.0],
              ["Shoes", "mid heel", "bright", "business", 400.0],
              ["Shoes", "low heel", "dark", "business", 150.0],
              ["Shoes", "flat", "bright", "casual", 0.0],
              ["Shoes", "boots", "dark", "casual", 500.0],

              ["Neck", "necklace", "dark", "evening", 150.0],
              ["Neck", "necklace", "dark", "business", 150.0],
              ["Neck", "choker", "bright", "casual", 0.0],
              ["Neck", "choker", "bright", "evening", 0.0],
              ["Neck", "scarf", "bright", "casual", 250.0],
              ["Neck", "tie", "dark", "business", 100.0],
              ["Neck", "bow tie", "dark", "business", 100.0],

              ["Handbag", "backpack", "dark", "sportswear", 100.0],
              ["Handbag", "purse", "dark", "business", 600.0],
              ["Handbag", "clutch", "bright", "evening", 500.0],
              ["Handbag", "belt bag", "bright", "casual", 300.0],
              ["Handbag", "cross bag", "bright", "business", 0.0]]
```

The population represented in an array of arrays. Each array has Category, Piece, Color, Dress Code, Price. So, the solution will be representing as String.

2- Initial population:

```
init_population = [] # Initial population
for i in range(0, generation_size):
    init_population.append([population[randrange(0, 19)], population[randrange(19, 32)]
                        , population[randrange(32, 42)], population[randrange(42, 49)]
                        , population[randrange(49, 54)]])
return init_population
```

The first population initiated randomly by using Randrange() function in Python. As we see in the solution representation above, the first 18 arrays take the Top category, and from 18-32 take the bottom category, ...etc.

So init_population after deciding the generation size n, the array will have n outfits, each outfit contain the five categories (top, bottom, shoe, neck, handbag).

3- Fitness function:

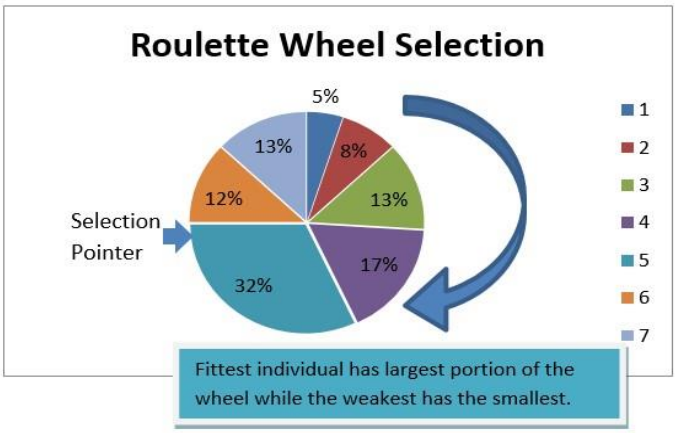
```
def fitness_function(outfit):
    fitness_value = 0 # wighted sum
    cost = 0.0
    for i in range(0, 5):
        if outfit[i][2] == color:
            fitness_value = fitness_value + 0.2
        if outfit[i][3] == dress_code:
            fitness_value = fitness_value + 0.4
        cost += outfit[i][4]
    fitness_value = fitness_value / 5
    if cost <= user_money :
        fitness_value = fitness_value + 0.4
    arr = [fitness_value, outfit]
    return arr
```

The Fitness function gives each outfit a score to be able to compete with other outfits, the higher the score is, the more likely to be chosen.
The criteria's that determined the range of the score are: Dress code, Color, Budget. The dress code and budget are equally important, and the color is half important. So, we chose numbers, and the sum of these numbers = 1 as follow:
Dress code = 0.4
Budget = 0.4
Color = 0.2
Then, the process will be conditions using if statements.

4- Selection method (Roulette wheel):

```
def roulette_wheel(popu):# (selection method)
    total_sum = 0
    for outfit in popu:
        total_sum = total_sum+ outfit[0]
    r = random.uniform(0, total_sum)
    partial_sum = 0
    for outfit in popu:
        partial_sum = partial_sum + outfit[0]
        if partial_sum >= r:
            return outfit[1]
```

The basic idea of the selection method is to choose two parents. Roulette wheel is one of the types of the selection methods, it works like a wheel where the outfit that has higher score most likely will be selected as parent.



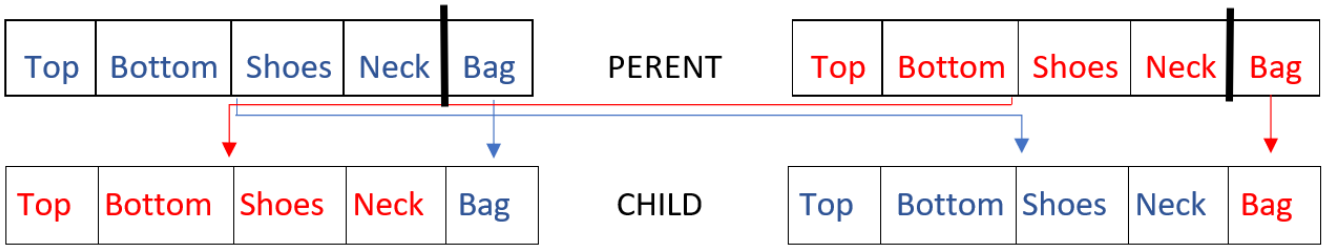
Where the selection pointer is random number from 0 to the sum of the fitness values of all outfits in the population.

5- Crossover:

Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes [1].

```
def crossover(p1, p2):
    k = randrange(0, 5)
    for i in range(0,k):
        temp=p1[i]
        p1[i]=p2[i]
        p2[i]=temp
    return [p1,p2]
```

For example, when k=3.

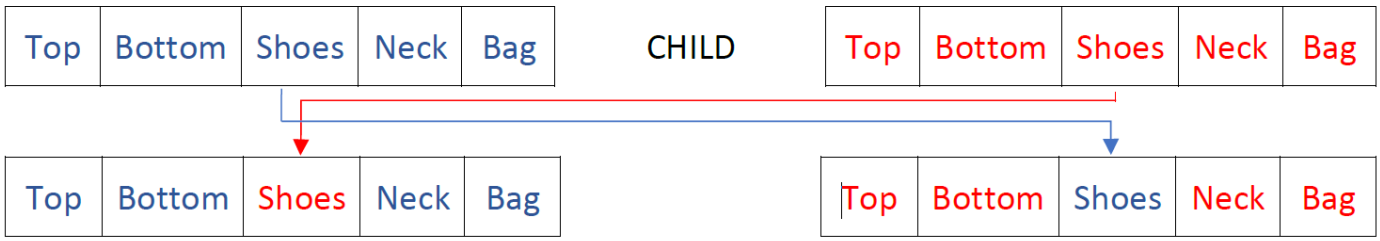


6- Mutation:

In certain new offspring formed, some of their genes can be subjected to a mutation with a low random probability. This implies that some of the bits in the bit string can be flipped.

```
def mutation(i, j):
    k = randrange(0, 5)
    i1=i[k]
    j1=j[k]
    i[k]=j1
    j[k]=i1
    return [i, j]
```

For example, when k=2.



7- Replacement:

```
def replacement(par,new):
    length = len(population)
    for i in range(0, length):
        if population[i] == par:
            population.remove(par)
            population.append(new)
            break
```

In this stage the winning individuals in the population are replaced with the parents to introduce the next generation. So, we will check the parents in the population and remove them, then the winning outfits will be added.

8- Termination condition.

```
termination_c = 200 # termination condition
for i in range(0, termination_c):
```

The code will terminate after generating 200 generation.

4) Programming language

PYTHON.

5) Tools:

Software tools: PyCharm IDE.

6) References:

[1] V. Mallawaarachchi, "towards Data science," 8 Jul 2017. [Online]. Available: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3.