ECE449 Lab 5

November 1st, 2023

Fuzzy Logic with GA

- **Genetic Algorithm:** A heuristic search algorithm for optimizing an objective function
- **Genes:** building blocks of our solutions, string these genes together, we get a **chromosome**, which represents a full solution to our problem.
- **Population:** a diverse group of solutions competing to solve our problem.
- **Fitness Function:** objective function which is minimized or maximized

Fuzzy Logic with GA

- **Crossover:** mixing the best parts of two solutions, hoping that their offspring inherits the best attributes of each parent
- **Mutation:** adding a pinch of randomness to our new solutions, ensuring that we explore a wide range of possibilities and avoid getting stuck

Fuzzy Logic with GA

- Same Tipping Problem
- Use GA to optimize membership functions so that the solution are as close as possible to a dataset
- Clean the given dataset: clip rows with values smaller than 0 and greater than 1
- Note that your tip is between 0-30 in this lab. Other values in the dataset are between 0-1, so multiply by your max universe set value if needed.
- `!pip install scikit-fuzzy`
- `!pip install EasyGA`

Fuzzy Logic with GA

- Don't use `.automf(3)` for membership functions instead use:

These are the solutions we are trying to find

```
temperature['poor'] = fuzz.trimf(temperature.universe, [0, 0, 5])
temperature['average'] = fuzz.trimf(temperature.universe, [0, 5, 10])
temperature['good'] = fuzz.trimf(temperature.universe, [5, 5, 10])
```

- Or Any other membership function you want
- You should be able to access membership parameters this way.

Fuzzy Logic with GA

- Define enough number of rules so that at least one rule is activated for every combination of memberships.
- Define your fitness function:
- Should return the error between the tip found by your fuzzy system and the tip in the dataset.
- `error = ∑abs(actual_tip - predicted_tip)`

Fuzzy Logic with GA

- For Instance:

```
def fitness(chromosome):
    food_sim, service_sim, tip_sim = setup_fuzzy_system(chromosome)

    total_error = 0
    for index, row in train_data.iterrows():
        inputs = {
            'temperature': row['temperature'],
            'flavor': row['flavor'],
            'portion_size': row['portion_size'],
            'attentiveness': row['attentiveness'],
            'friendliness': row['friendliness'],
            'speed': row['speed']
        }
        actual_tip = row['tip']
        predicted_tip = execute_fuzzy_inference(food_sim, service_sim, tip_sim, inputs)
        error = abs(actual_tip - predicted_tip)
        total_error += error
    return total_error
```

Change based on your needs and implementation.

Fuzzy Logic with GA

- Define your GA:
- `ga = EasyGA.GA()`
- Define how the chromosomes should be generated.
- You can assert any constraints on the membership function parameters here.
- Remember that your membership functions should cover the whole universe set they are defined on, and should be within boundaries set by universe set.

```
ga.gene_impl = lambda: generate_chromosome()
```

Fuzzy Logic with GA

```
ga.chromosome_length =
ga.population_size = 200 (whatever you like)
ga.target_fitness_type = 'min'
ga.generation_goal = Number of iteration to run
ga.fitness_function_impl = fitness (your fitness function)

ga.evolve()
ga.print_best_chromosome()
```

Final model you get should outperform unoptimized model.