Assignment-7: Genetic Algorithm

Munem Shahriar 2021251042
Tausif Mushtaque 2031020642
Email addresses: munem.shahriar07@nothsouth.edu
tausif.mushtaque@northsouth.edu

December 8, 2024

I. Introduction

Genetic algorithms (GAs) are optimization and search techniques inspired by the principles of natural selection and genetics. These algorithms are widely used to solve problems where the search space is vast and complex. GAs operate by encoding potential solutions as chromosomes and iteratively evolving these solutions using operators like selection, crossover, and mutation.

II. Methods

A. Dataset

We utilized the UCI repository to fetch the `Car Evaluation` dataset. The dataset includes various features and target labels, representing car acceptability based on input criteria.

Below is a snippet of the code used to load and inspect the dataset:

```
from ucimlrepo import fetch_ucirepo

# Fetch dataset
car_evaluation = fetch_ucirepo(id=19)

# Data (as pandas DataFrames)
X = car_evaluation.data.features
y = car_evaluation.data.targets

# Inspect metadata
print(car_evaluation.metadata)
```

B. Preprocessing

The data was examined for missing values and processed accordingly. An example of checking null values:

```
# Check for missing values
X.isnull()
y.isnull()
```

C. Genetic Algorithm Implementation

The genetic algorithm operates in the following steps:

1. **Initial Population Generation**: A population of P individuals is generated. Each individual is represented as a vector of hyperparameters:

$$\mathbf{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\},$$

where x_{ij} corresponds to the j-th hyperparameter of the i-th individual.

2. **Fitness Evaluation**: The fitness of each individual is determined using an evaluation function $f(\mathbf{x}_i)$, such as the accuracy of a decision tree model:

$$f(\mathbf{x}_i) = \text{Accuracy}(\mathbf{x}_i, \text{train data}).$$

3. **Selection**: Individuals are selected based on their fitness scores $f(\mathbf{x}_i)$ using a probability proportional to their fitness:

$$P(\mathbf{x}_i) = \frac{f(\mathbf{x}_i)}{\sum_{j=1}^{P} f(\mathbf{x}_j)}.$$

4. **Crossover**: Two selected parents produce an offspring $\mathbf{x}_{\text{child}}$ by combining their genes:

$$x_{\text{child},j} = \begin{cases} x_{\text{parent1},j} & \text{with probability } 0.5, \\ x_{\text{parent2},j} & \text{with probability } 0.5. \end{cases}$$

5. **Mutation**: Random mutations are introduced with a small probability μ, altering the value of some hyperparameters:

$$x_{\text{mutated},j} = \begin{cases} x_j + \Delta & \text{with probability } \mu, \\ x_j & \text{otherwise.} \end{cases}$$

Below is the Python implementation of these steps:

```
# Generate initial population
def generate_population(size):
    population = []
    for _ in range(size):
        individual = {
            "max_depth": random.randint(*hyperparameter_ranges["max_depth"]),
            "min_samples_split": random.randint(*hyperparameter_ranges["min_samples_split"]),
            "min_samples_leaf": random.randint(*hyperparameter_ranges["min_samples_leaf"]),
        }
        population.append(individual)
    return population

# Fitness function
def fitness(individual):
    clf = DecisionTreeClassifier(
        max_depth=individual["max_depth"],
        min_samples_split=individual["min_samples_split"],
        min_samples_leaf=individual["min_samples_leaf"],
    )
    clf.fit(Xtrain_set, Ytrain_set)
    accuracy = clf.score(Xval_set, Yval_set)
    return accuracy

# Crossover
def crossover(parent1, parent2):
    child = {}
    for key in parent1.keys():
        child[key] = random.choice([parent1[key], parent2[key]])
    return child

# Mutation
def mutate(individual, mutation_rate=0.1):
    for key in individual.keys():
        if random.random() < mutation_rate:
            individual[key] = random.randint(*hyperparameter_ranges[key])
    return individual

# Selection
def select(population, fitnesses):
    selected = random.choices(population, weights=fitnesses, k=2)
    return selected
```

D. Matrix Representation

The features matrix (X) can be represented as follows (example for a subset of data):

$$X = \begin{bmatrix} 2 & 1 & 3 & 4 \\ 1 & 2 & 3 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

III. Conclusion

The genetic algorithm, combined with preprocessing steps like handling missing data, provides an efficient method for analyzing datasets like the `Car Evaluation`. Through this approach, we can better understand feature significance and optimize performance metrics. Future work could focus on enhancing the selection and mutation operators to achieve improved results in larger datasets.