Al-Mustaqbal University
College of Sciences
Intelligent Medical System Department

جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم
قسم الانظمة الطبية الذكية

Lecture: (7)

Implement genetic algorithms and fuzzy set approaches for classification

Subject: Clinical Data Mining

Level: Four

Lecturer: Dr. Maytham Nabeel Meqdad

Implement genetic algorithms and fuzzy set approaches for classification.

Classification Using Genetic Algorithms and Fuzzy Sets

1. Classification Using Genetic Algorithms (GA)

Genetic Algorithms (GA) use the concept of natural evolution to find the best possible classification model by optimizing a set of rules or parameters.

1.1 What GA Tries to Optimize in a Classification System

GA can be used in three main areas:

A. Rule Generation GA can generate rules of the form:

IF (Feature1 = High) AND (Feature3 = Medium) THEN Class = C1

B. Feature Selection: Selecting the most important features that significantly affect classification.

C. Parameter Optimization Tuning model parameters to improve performance.

1.2 Steps to Build a Classifier Using GA

Step 1: Solution Representation (Chromosome Encoding) Each chromosome represents a rule or a set of rules.

Example of rule encoding:

[Feature1:0/1] [Feature2:0/1] [Feature3:0/1] → [Class] Where 0 = not used, 1 = used.

Step 2: Performance Evaluation (Fitness Function) The fitness function may include: -

1.Classification accuracy.

2. Number of rules (to simplify the model)

3. Sensitivity

4. Precision

Example

fitness function: **Fitness = 0.7 * Accuracy + 0.3 * (1 / Number of Rules)**

Step 3: GA Operators :

1. Selection: Choose the best rules.

2. Crossover: Combine rules to produce new rules.

3. Mutation: Modify part of a rule (e.g., adjust a feature threshold).

Step 4: Obtain the Final Model After several generations; the GA produces an optimized set of rules for classification.

1.3 Simple Example

Classify patients into: - High-Risk - Low-Risk

Based on: - Age - Blood Pressure - Heart Rate

GA may generate rules such as:

IF Age > 50 AND BloodPressure > 140 THEN Class = High-Risk

2. Classification Using Fuzzy Sets

Fuzzy logic uses degrees of membership instead of crisp values.

Example:

Blood Pressure = 130 → membership 0.6 in "High" Blood Pressure = 130 → membership 0.4 in "Normal"

2.1 Components of a Fuzzy Classification System

A. Fuzzification Convert numeric values into membership degrees.

Example

for "Blood Pressure": Low / Medium / High

Membership functions: Gaussian, Trapezoidal, Triangular

B. Rule Base Fuzzy logical rules:

IF BP is High AND Age is Old THEN Risk is High

C. Inference Engine Applies fuzzy rules using methods such as Mamdani or Sugeno.

D. Defuzzification Converts the fuzzy result into a final class label.

2.2 Why Fuzzy Logic is Suitable for Classification

1. Handles uncertainty and vagueness

2. Works well with medical data

3. Can naturally incorporate expert knowledge

4. Can be combined with GA to create Fuzzy-GA Classifiers.

3. Hybrid System: Fuzzy Genetic Classification (Most Powerful)

Fuzzy Genetic Classification

Idea: Use **Genetic Algorithms** to automatically **generate, tune, and optimize fuzzy rules and membership functions** for classification.

Steps:

1. **Initialization**
   - Start with a random population of fuzzy rules and membership functions.
   - Each individual in the population represents a **set of fuzzy rules**.
2. **Fuzzy Rule Evaluation**
   - Apply the fuzzy rules to the training data.
   - Compute **fitness** based on classification accuracy (how well the rules classify data).
3. **Selection**
   - Select the best-performing individuals for reproduction.
4. **Crossover & Mutation**
   - Combine rules from two individuals (crossover) and randomly alter some parts (mutation) to create new candidates.
   - This explores new fuzzy rule combinations and improves diversity.
5. **Iteration**
   - Repeat evaluation, selection, crossover, and mutation for multiple generations until:
     - High classification accuracy is achieved.
     - Maximum generations are reached.
6. **Defuzzification (Optional)**
   - Convert fuzzy outputs into crisp class labels for final classification.

Advantages of Fuzzy Genetic Classification

- Automatically finds **optimal fuzzy rules** and **membership functions**.
- Handles **uncertainty, vagueness, and overlapping classes**.
- Avoids manual tuning of fuzzy systems.
- Can achieve **very high classification accuracy** even with complex datasets.
-

Disadvantages

- Computationally intensive (especially for large datasets).
- Requires careful design of:
  - Fitness function
  - GA parameters (population size, mutation rate, etc.)
- Interpretation of resulting fuzzy rules may be complex.

4. Comparison Between Methods

Aspect	GA	Fuzzy	Fuzzy + GA
Handles Uncertainty	□	✔	✔
Model Interpretability	Medium	High	High
Accuracy	High	Medium	Very High
Training Speed	Slow	Medium	Relatively Slow

Examples

1. Fuzzy Classification Example (Python)

```
# Install scikit-fuzzy if not installed
# !pip install scikit-fuzzy

import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# Define input variable: Age
age = ctrl.Antecedent(np.arange(0, 101, 1), 'age')

# Define output variable: Risk
risk = ctrl.Consequent(np.arange(0, 11, 1), 'risk')

# Membership functions for age
age['young'] = fuzz.trimf(age.universe, [0, 0, 30])
age['middle'] = fuzz.trimf(age.universe, [20, 50, 80])
age['old'] = fuzz.trimf(age.universe, [60, 100, 100])

# Membership functions for risk
risk['low'] = fuzz.trimf(risk.universe, [0, 0, 5])
risk['medium'] = fuzz.trimf(risk.universe, [3, 5, 7])
risk['high'] = fuzz.trimf(risk.universe, [5, 10, 10])

# Fuzzy rules
rule1 = ctrl.Rule(age['young'], risk['low'])
rule2 = ctrl.Rule(age['middle'], risk['medium'])
rule3 = ctrl.Rule(age['old'], risk['high'])
```

```
# Control system
risk_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
risk_sim = ctrl.ControlSystemSimulation(risk_ctrl)

# Test input
risk_sim.input['age'] = 45
risk_sim.compute()

print(f"Fuzzy Classification Result: {risk_sim.output['risk']:.2f}")
```

2. Simple Genetic Algorithm Example (Python)

```
# Simple Genetic Algorithm to maximize x^2 in range 0-31
import random

def fitness(x):
    return x**2

def selection(pop):
    return sorted(pop, key=fitness, reverse=True)[:2]

def crossover(p1, p2):
    mask = random.randint(1, 31)
    return (p1 & mask) | (p2 & ~mask)

def mutate(x):
    if random.random() < 0.1:
        bit = 1 << random.randint(0, 4)
        x = x ^ bit
    return x

# Initialize population
population = [random.randint(0, 31) for _ in range(6)]

for generation in range(10):
    selected = selection(population)
    next_gen = []
    for _ in range(3):
        offspring = crossover(selected[0], selected[1])
        offspring = mutate(offspring)
        next_gen.append(offspring)
    population = next_gen
    print(f"Generation {generation}: {population}")

best = max(population, key=fitness)
print(f"Best solution found: x = {best}, fitness = {fitness(best)}")
```

3. Fuzzy Genetic Classification Example (Python)

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import random

# Training data (Age, Risk Level)
data = np.array([
    [15, 2],
    [25, 3],
    [35, 5],
    [45, 6],
    [60, 8],
    [75, 9]
])

# Initialize fuzzy membership parameters: [young_end, middle_start,
middle_end, old_start]
params = [30, 20, 50, 60]

def fuzzy_classification_score(params):
    young_end, middle_start, middle_end, old_start = params

    # Define fuzzy sets
    age = ctrl.Antecedent(np.arange(0, 101, 1), 'age')
    risk = ctrl.Consequent(np.arange(0, 11, 1), 'risk')

    age['young'] = fuzz.trimf(age.universe, [0, 0, young_end])
    age['middle'] = fuzz.trimf(age.universe, [middle_start, middle_end,
middle_end+30])
    age['old'] = fuzz.trimf(age.universe, [old_start, 100, 100])

    risk['low'] = fuzz.trimf(risk.universe, [0, 0, 5])
    risk['medium'] = fuzz.trimf(risk.universe, [3, 5, 7])
    risk['high'] = fuzz.trimf(risk.universe, [5, 10, 10])

    # Fuzzy rules
    rules = [ctrl.Rule(age['young'], risk['low']),
             ctrl.Rule(age['middle'], risk['medium']),
             ctrl.Rule(age['old'], risk['high'])]

    # Control system
    system = ctrl.ControlSystem(rules)
    sim = ctrl.ControlSystemSimulation(system)

    # Compute fitness as inverse of error
    error = 0
    for row in data:
        sim.input['age'] = row[0]
        sim.compute()
        error += abs(sim.output['risk'] - row[1])
    return -error  # GA maximizes fitness
```

```
# Simple GA to optimize parameters
population = [ [random.randint(20,40), random.randint(20,40),
random.randint(40,60), random.randint(50,70)] for _ in range(6) ]

for generation in range(10):
    population.sort(key=fuzzy_classification_score, reverse=True)
    next_gen = population[:2]  # select top 2
    # Crossover
    while len(next_gen) < 6:
        p1, p2 = random.sample(population[:4], 2)
        child = [(a+b)//2 for a,b in zip(p1,p2)]
        # Mutation
        child = [c + random.randint(-2,2) for c in child]
        next_gen.append(child)
    population = next_gen

best_params = population[0]
print(f"Optimized Fuzzy Parameters: {best_params}")
print(f"Best Fitness: {fuzzy_classification_score(best_params)}")
```

References

[1] Data Mining: Concepts and Techniques, Second Edition" Jiawei Han and Micheline Kamber"

[2] .M.H. Dunham, " Data Mining Introductory and Adv anced Topics", Pear son Education Jiawei Han and Micheline Kamber