Fuzzy Clustering using PSO (Particle Swarm Optimization)

Enhanced Clustering with Entropy based Smarter Adaptive Reduction (E-SAPR) Algorithm

- **Group Members Name:** Sanjiv Mishra , Agniva Mishra, Barsha Singh, Bhawna Jain, Hitendra Kumar.

- **Under Guidance:** Kalyan Kumar Das.
- **Course:** B.Tech in Computer Science & Engineering.
- **University Name:** Techno India University.
- **Date:** 15.05.2025

Introduction to Particle Swarm Optimization (PSO)

- Particle Swarm Optimization (PSO) is an algorithm inspired by how birds flock together to find food. In PSO, we have a group of particles, where each particle represents a possible solution to a problem. These particles "fly" through the solution space, adjusting their positions based on their own experience (the best position they've found, called personal best or pBest) and the experience of the whole group (the best overall position, called global best or gBest). Each particle has a velocity that helps it move towards better solutions, influenced by its current direction, its pBest, and the gBest. The particles update their positions over several iterations, trying to find the best possible solution. By sharing information and learning from one another, the group eventually converges on the optimal or near-optimal solution. PSO is simple, efficient, and widely used for solving optimization problems because of its collective intelligence approach, allowing particles to explore and exploit the search space effectively.

Fuzzy Clustering

- What is Fuzzy Clustering?
- Fuzzy clustering is a technique where each data point can belong to multiple clusters with varying degrees of membership, unlike traditional clustering methods like k-means that assign points to only one cluster. In fuzzy clustering, each point has a membership value for each cluster, and these values sum to 1. The algorithm, often implemented as Fuzzy C-Means (FCM), works by iteratively updating the cluster centroids and the membership values based on the distances between data points and centroids. It continues until the centroids and membership values stabilize, making it useful for situations where data points overlap between clusters or have uncertain boundaries. This method is applied in areas like image processing, pattern recognition, and bioinformatics.

Motivation Of Clustering

- **What is Clustering?**
  Clustering is the process of grouping data points into clusters based on similarity.
  Used in diverse fields: bioinformatics, image segmentation, customer segmentation, etc.
- **Why is Clustering Important?**
  Identifies patterns and structures in large datasets.
  Enables better decision-making by uncovering meaningful groups.
  Addresses challenges of ambiguous, overlapping, and high-dimensional data.

Problem Statement

- Standard PSO might struggle with population size or premature convergence in the case of complex and large datasets.
- **The goal of this project is to enhance PSO by introducing:**

E-SAPR Algorithm:
Enhance PSO with entropy-based diversity control and adaptive population reduction:

- Dynamically reduce swarm size based on **entropy** values to maintain diversity.
- Refines centroids, improving clustering quality and computational efficiency.

How Normal PSO Works

- In Normal PSO, each particle starts with a random position and speed within certain limits. It tries to find the best position by evaluating its fitness using a specific function. The particle remembers its best position so far (personal best), and the whole group tracks the best position found (global best). The particle updates its speed and position based on its personal best and the global best, moving closer to better solutions. This process is repeated over several generations, with each particle adjusting its position to find the best possible answer.

How Modified PSO Works

- **What is E-SAPR?**
  Enhances PSO by introducing entropy-based diversity control and adaptive population reductio
  Addresses premature convergence and computational inefficiency.
- **Modifications Introduced:**
- **Entropy Calculation:**
  Measures swarm diversity: $H = -\sum p_i \log p_i.$
- **Adaptive Swarm Reduction:**
  Dynamically reduces swarm size when entropy drops below a threshold.
- **Stagnation Prevention:**
  Introduces mutation in velocities to avoid local minima.

Modified PSO with Fuzzy Clustering (How It Works for Better Results)

Modified PSO with Fuzzy Clustering: How It Works for Better Results

- **Entropy-Based Adaptive Population**:
  The modified PSO reduces or maintains the number of particles dynamically based on *entropy*, which measures the diversity of solutions. High diversity = more exploration; low diversity = fewer, focused particles.
- **Improved Search Efficiency**:
  By shrinking the population intelligently, it avoids wasting resources on poor solutions and focuses more on promising areas — leading to faster and more accurate convergence.

Modified PSO with Fuzzy Clustering (How It Works for Better Results)

- **Better Cluster Center Initialization**:
  The best position found by modified PSO helps initialize the fuzzy c-means clustering process closer to optimal cluster centers, reducing the chances of bad local optima.
- **Fuzzy Membership Integration**:
  After PSO optimization, fuzzy c-means assigns soft cluster memberships — giving flexibility in assigning data points to multiple clusters, improving clustering quality.
- **Quality Metrics Improved**:
  - **Lower XB Index** → Indicates compact and well-separated clusters.
  - **Lower DB Index** → Reflects better inter-cluster separation and intra-cluster cohesion.

Code Implementation - Original PSO

Key Components :

1. **Particle Initialization**: Random positions and velocities.
2. **Fitness Evaluation**: Each particle evaluates the fitness of its position using a **fitness function** (e.g., the objective function for clustering).
3. **Update Equations**:
    1. Position and velocity updates are based on the cognitive and social components.

Features

•**Number of Particles**: 30
•**Generations**: 500
•**Fitness Function**: A function that evaluates the clustering quality.
•**Equation for Position Update**:
vi(t+1)=w·vi(t)+c1·rand1·(pbesti−xi(t))+c2·rand2·(gbest−xi(t))
xi(t+1)=xi(t)+vi(t+1)x_i(t+1) = x_i(t) + v_i(t+1)xi(t+1)=xi(t)+vi(t+1)
•Where → vi(t) is the velocity of particle iii,
pbesti is the best solution of particle iii,
gbest is the best solution found by the entire swarm.

Code Implementation - Modified PSO

- Key Changes:
- **Entropy Calculation:**
  The Shannon entropy of particle positions is computed to quantify swarm diversity. Lower entropy indicates reduced variability, signaling convergence.
- **Adaptive Population Reduction:**
  Based on entropy values, the swarm size is dynamically adjusted. Redundant particles with minimal contributions to diversity are removed while ensuring coverage of the search space.
- **Convergence Criterion:**
  The algorithm terminates when the maximum number of iterations is reached, or no significant fitness improvement occurs.

Performance Comparison (Original vs. Modified PSO)

Original PSO

- **Execution Time**: 0.25 seconds
- **Fitness Value Comparison**: 1.954
- **Results**: The modified PSO shows better fitness values and faster convergence, highlighting the benefits of the introduced modifications. The fixed swarm size increases computational overhead, making it less efficient for larger datasets.

Modified PSO

- **Execution Time**: 0.19 seconds
- **Fitness Value Comparison**: 1.999
- **Results**: Display the convergence curves for both algorithms to show Entropy-based diversity control that prevents premature convergence, while adaptive population reduction optimizes computational efficiency.

Velocity Update Logic Normal PSO

- # Velocity update rule for normal PSO
- new_velocity = (
-     inertia * particle.velocity +
-     cognitive * (particle.best_position - particle.position) +
-     social * (global_best_position - particle.position)
-)
- particle.velocity = new_velocity
- particle.position += new_velocity

Velocity Update Logic Modified PSO

- # Velocity update rule in E-SAPR PSO
- new_velocity = (
-     inertia * particle.velocity +
-     cognitive * (particle.best_position - particle.position) +
-     social * (self.global_best_position - particle.position)
-)
- particle.velocity = new_velocity
- particle.position += new_velocity

Adaptive Population Reduction in Modified PSO

- entropy = compute_entropy(np.array([p.position[0] for p in self.population]))
- new_size = max(self.min_size, int(self.initial_size * (entropy / np.log2(self.initial_size))))

- # Retain only the best-performing particles
- self.population.sort(key=lambda p: p.fitness)
- self.population = self.population[:new_size]

Entropy Calculation for Diversity Monitoring

```
def compute_entropy(particles):
    if len(particles) == 0:
        return 0  # No entropy if no particles exist

    min_val, max_val = np.min(particles), np.max(particles)
    if min_val == max_val:
        return 0  # No diversity if all particles are the same

    norm_particles = (particles - min_val) / (max_val - min_val + 1e-9)  # Normalize
    histogram, _ = np.histogram(norm_particles, bins=10, density=True)
    histogram = histogram / np.sum(histogram)  # Convert to probability distribution

    entropy = -np.sum(histogram * np.log2(histogram + 1e-9))  # Shannon entropy
    return entropy
```

Thank you