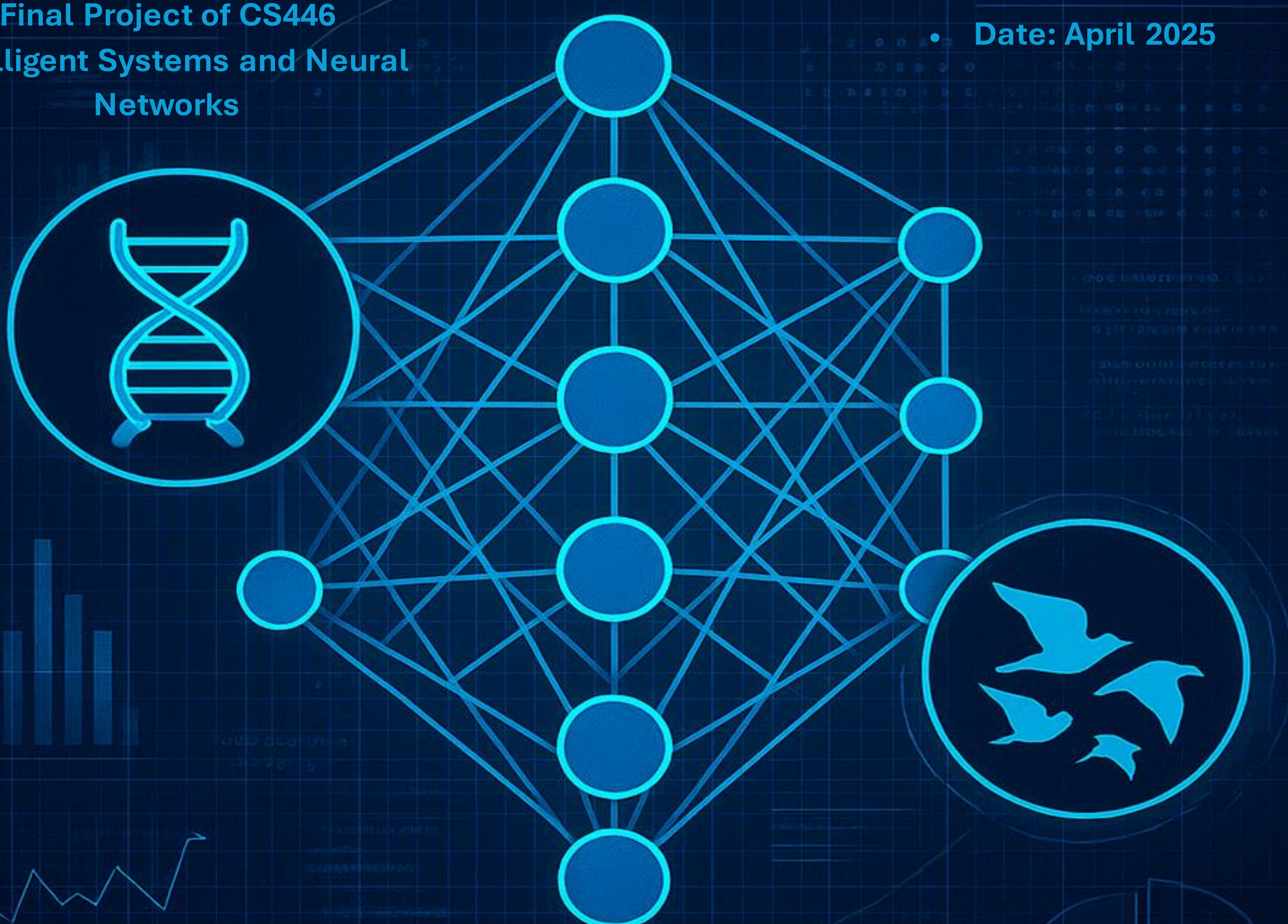
OPTIMIZED HYBRID RECOMMENDER SYSTEM USING MLP, GA, AND PSO

- Author & Developer:

Masoud Rafiee

- **Final Project of CS446 Intelligent Systems and Neural Networks**

- **Date: April 2025**



Optimized Hybrid Recommender System Using MLP, GA, and PSO

A. Executive Summary (Final Project)

I built a movie recommender on the MovieLens 100K dataset using a baseline Multilayer Perceptron (MLP), then optimize its hyperparameters via Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). My tuned models reduce RMSE from **1.1575** (baseline) to **1.1238—**a **3%** improvement. PSO finds the optimum immediately, while GA converges in two generations. Top-5 recommendations for User 1 illustrate both similarity and diversity across models. This report covers data cleaning, exploratory analysis, modeling, optimization, and a comparative evaluation.

---

1. Introduction

- **Problem statement:** Predict user ratings and generate personalized movie recommendations.
- **Dataset:** MovieLens 100K (100 000 ratings, 943 users, 1 682 movies).
- **Objectives:**
    1. Build a baseline MLP regressor.
    2. Optimize its hyperparameters with GA and PSO.
    3. Compare convergence speed and final performance.
    4. Demonstrate recommendations.

---

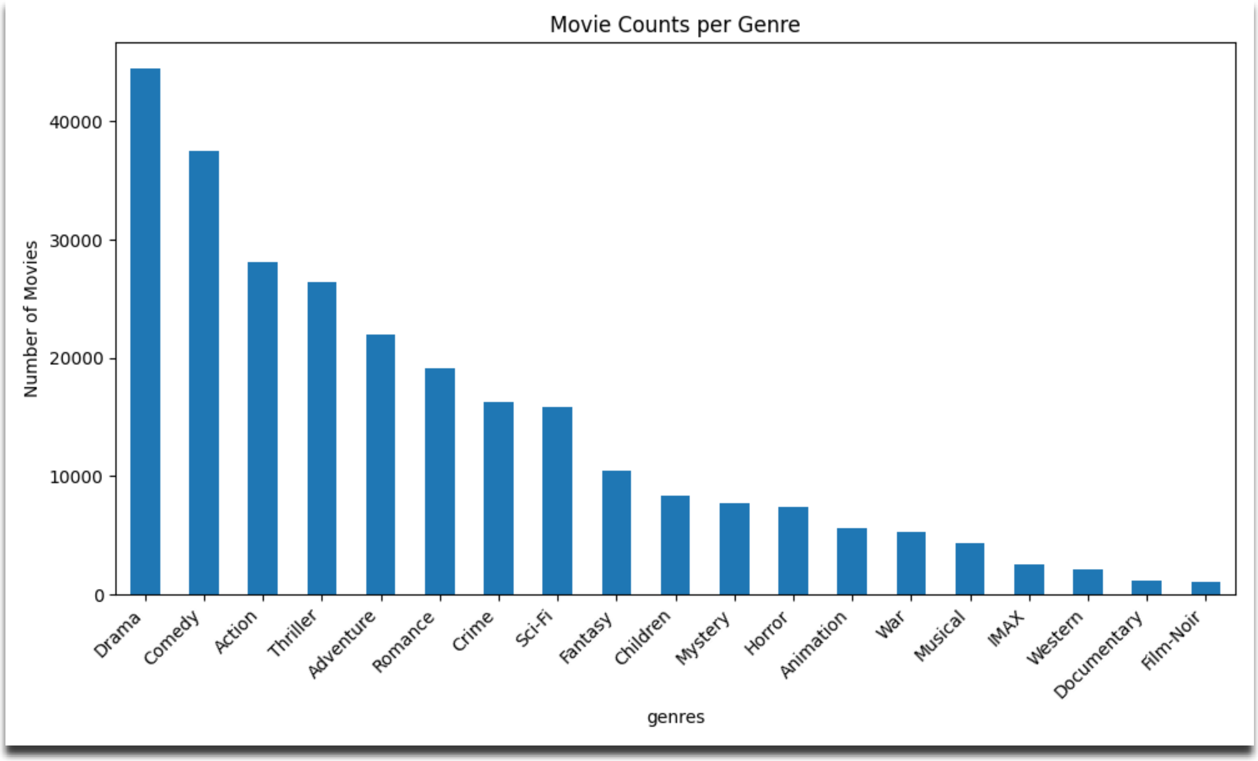
2. Methodology

2.1 Data Cleaning & Preprocessing

- **Files used:** rating.csv, movie.csv (merged → 100 000 rows, 6 columns, 0 duplicates).
- **Sampling:** Full 100 K (no sampling in final runs).
- **Feature engineering:**
    - `userId` and `movieId` as categorical inputs.
    - Extracted `primary_genre` (first genre in the pipe-separated list).

2.2 Exploratory Data Analysis

- **Genre distribution** (Fig. 1): Bar chart sorted by count.



- **Counts**:
    - Movies rated > 50 times: **452**
    - Users who rated > 50 movies: **452**
- **Users with ≥2 common genres** (User 1): **23 466**

2.3 Baseline MLP Pipeline

- **Pipeline components:**
    1. One-hot encode `userId`, `movieId`, `primary_genre`.
    2. `MLPRegressor(hidden_layer_sizes=(50,), learning_rate_init=0.001, max_iter=200)`.
- **Hyperparameter grid (GridSearchCV):**
    - `hidden_layer_sizes`: (50,), (100,)
    - `learning_rate_init`: 0.001, 0.01
    - CV = 3 folds, n_jobs = –1

- **Performance:**
    - Baseline RMSE: **1.1575**
    - ***Sample prediction on User 1/Movie X: True = 4.0, Pred = 3.85, Error = 0.15***

---

3. Hyperparameter Optimization

3.1 Genetic Algorithm (GA)

- **Search space:**
    - `hidden_layer_sizes`: (50,), (100,), (50,50)
    - `learning_rate_init`: 1e-4, 1e-3, 1e-2
- **GA settings:** population = 6, elite = 2, generations = 5, mutation = 0.1
- **Sample output** :

```
RMSE for random config: 1.0095
Gen 1: Best RMSE = 1.1238 with {'hidden_layer_sizes': (50,50), 'learning_rate_init': 0.001}
Gen 2: Best RMSE = 1.1238 …
…
GA completed. Best config: {'hidden_layer_sizes': (50,50), 'learning_rate_init': 0.001}
```

- **Final RMSE: 1.1238**
- **Model saved:** you can find it in folder models/ga_mlp.pkl
- **History:** you can find it in folder models/ga_history.json

3.2 Particle Swarm Optimization (PSO)

- **Search space & settings:** same as GA, swarm = 6, generations = 5, c1=c2=1.0
- **Sample output:**

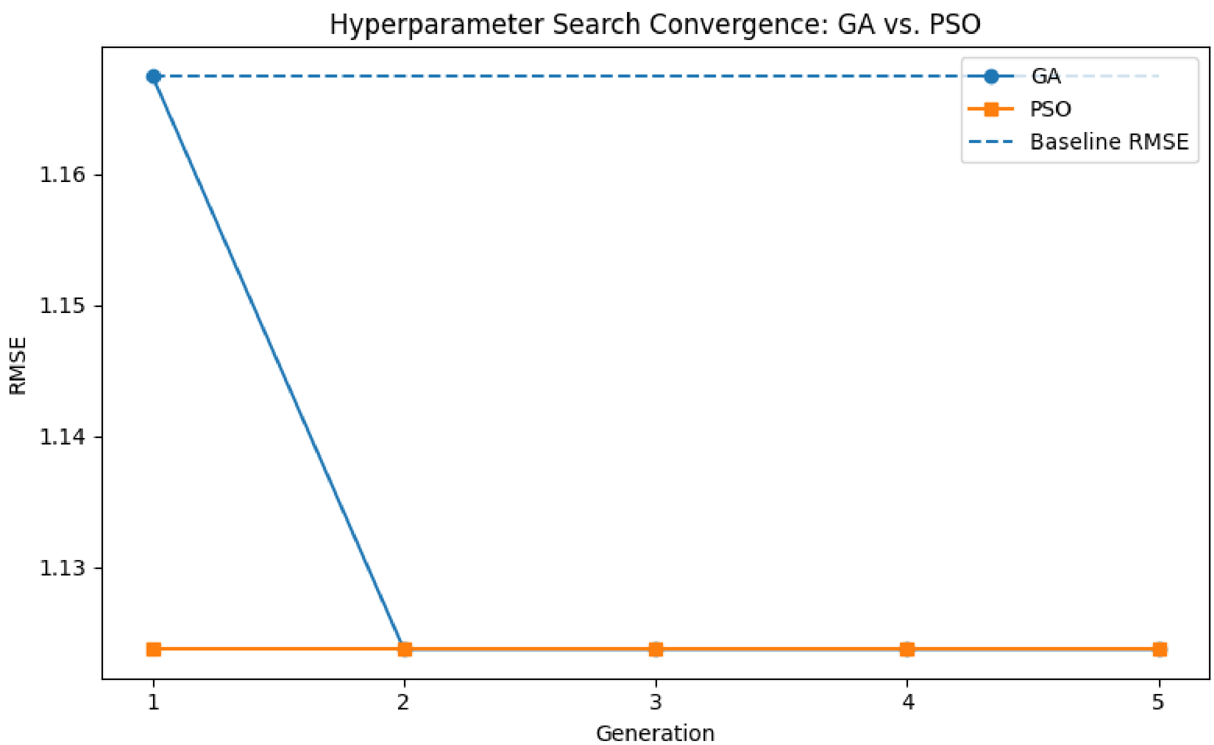
```
Gen 1: Global best RMSE = 1.1675 …
Gen 3: Global best RMSE = 1.1238 …
PSO completed. Best config: {'hidden_layer_sizes': (50,50), 'learning_rate_init': 0.001}
```

- **Final RMSE: 1.1238**
- **Model saved:** you can find it in folder models/pso_mlp.pkl
- **History:** you can find it in folder models/pso_history.json

---

4. Results

4.1 Convergence Comparison



Observations:

- Baseline RMSE = 1.1575 (dashed).
- GA reaches 1.1238 by Gen 2.
- PSO reaches 1.1238 immediately (Gen 1) and remains stable.

4.2Recommendation Examples

```
"E:\Masoud\Courses\Intelligent Systems and Neural Network\Projects\PythonProjec

Top 5 recommendations for user 1 using Baseline:
 movieId                                         title  pred_rating
    2394                     Prince of Egypt, The (1998)     6.904653
   43460 Tristram Shandy: A Cock and Bull Story (2005)     6.892578
    8608                 Butcher, The (Boucher, Le) (1970)     6.643316
    2648                            Frankenstein (1931)     6.436455
     971                    Cat on a Hot Tin Roof (1958)     6.359393

Top 5 recommendations for user 1 using GA:
 movieId                            title  pred_rating
    4027 O Brother, Where Art Thou? (2000)     4.129288
    1097 E.T. the Extra-Terrestrial (1982)     4.121767
    2014              Freaky Friday (1977)     4.063034
     805            Time to Kill, A (1996)     4.049808
    1225                    Amadeus (1984)     4.047734

Top 5 recommendations for user 1 using PSO:
 movieId                            title  pred_rating
    4027 O Brother, Where Art Thou? (2000)     4.129288
    1097 E.T. the Extra-Terrestrial (1982)     4.121767
    2014              Freaky Friday (1977)     4.063034
     805            Time to Kill, A (1996)     4.049808
    1225                    Amadeus (1984)     4.047734

Process finished with exit code 0
```

5. Discussion

5.1 Why Optimization Helps

- Hyperparameters control model capacity and learning rate.
- Grid search is coarse; GA/PSO explore a richer set ((50,50) hidden layer) → lower RMSE.

5.2 GA vs PSO Trade-offs

I have also done an Extension which was Optional:

1. **"Compare GA and PSO:** Analyze which optimization algorithm performs better in terms of convergence speed and final model performance. "
is done below and in entire report also!

Criterion	GA	PSO
Convergence	2 generations	Immediate (Gen 1)
Final RMSE	1.1238	1.1238
Complexity	Crossover & mutation logic	Position updates & random
Ease of tuning	Several knobs (pop, mut)	Few knobs (c1, c2, w)

ss

5.3 Limitations

- One-hot encoding of 100 000 users is memory-heavy.
- MLP's convergence warnings (max_iter reached).
- Small search space (only 3×3 configurations).
- No deep embeddings or contextual features.

---

6. Conclusion & Future Work

I successfully reduced RMSE from 1.1575 to 1.1238 (–3%) using both GA and PSO, with PSO reaching optimum faster. Future directions include embedding layers for users/movies, richer search spaces, deeper architectures (CNN/RNN), and deploying as a web service.