Large-Scale Collaborative Filtering for Movie Recommendations

Peeyush Dyavarashetty

I. Model Overview

The model will predict user ratings (1–5 stars) for unseen movies, enabling personalized recommendations. I propose building a collaborative filtering recommendation system using Apache Spark MLlib to predict user ratings for movies. This model will leverage distributed computing to handle large-scale datasets efficiently, demonstrating Spark's capabilities in production-grade machine learning. [1]

II. Dataset

The MovieLens 32M dataset [2] contains 32,000,204 ratings and 2,000,072 tags applied to 87,585 movies by 200,948 users, collected between January 1995 and October 2023.

Rating scale: 0.5 - 5.0 starts with 0.5 increments. Enables regression-based prediction tasks.

User activity $\geq$ 20 ratings/user. Reduces cold-start bias in validation.

Sparsity 99.9%. Challenges matrix completion algorithms.

The dataset contains the following files.

1) ratings.csv: Primary source for user-item interaction modeling. (columns: userId, movieId, rating, timestamp)

2) movies.csv: Genre metadata (19 categories) enables hybrid filtering. (columns: movieId, title, genres)

3) tags.csv: Optional for content-aware recommendations. (columns: userId, movieId, tag, timestamp)

4) links.csv: Cross-references to IMDb/TMDb for enrichment (columns: movieId, imdbId, tmdbId)

III. Algorithms

A. Baselines

- Most popular movies
- Cosine similarity between users
- User-based collaborative filtering (KNN)
- Matrix factorization with stochastic gradient descent [3]

B. Primary

- Alternating Least Squares (ALS) [4] with implicit feedback (Spark MLlib [5])
- ALS with Neural Network - Matrix Factorization [6]

IV. Hypothesis and Value

Hypothesis: Matrix factorization via ALS will outperform memory-based methods (e.g., KNN) in accuracy and scalability on large datasets due to Spark's distributed optimization. [6]

Why it matters:

- Solves cold-start problems for streaming platforms with sparse interaction data
- Demonstrates Spark's ability to handle iterative ML workloads at scale
- Provides framework for extending to real-time recommendations via Spark Streaming

V. Technical Approach

- Data Processing
- Training
- Evaluation
  - Metrics: RMSE, Precision@10, Coverage (% catalog recommended) [7]
  - Validation: Temporal split (train on pre-2020 data, test on 2020+ data)

VI. Risks and Mitigation

- Computational overhead: Use Spark's persist() for iterative workloads
- Cold-start problem: Hybrid approach with genre-based popularity fallback
- Model interpretability: SHAP values via spark-tfocs library

VII. Deliverables

- Trained recommendation model (Saved as Spark ML Pipeline)
- Technical report comparing ALS vs. baseline performance
- 5-minute demo video showing real-time predictions on sample data

References

[1] M. D. Ekstrand, J. T. Riedl, J. A. Konstan *et al.*, "Collaborative filtering recommender systems," *Foundations and Trends® in Human–Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011.

[2] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM TIIS*, 2015.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[4] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Algorithmic Aspects in Information and Management: 4th International Conference, AAIM 2008, Shanghai, China, June 23-25, 2008. Proceedings 4*. Springer, 2008, pp. 337–348.

[5] X. Meng, J. Bradley, B. Yavuz *et al.*, "Mllib: Machine learning in apache spark," in *JMLR*, vol. 17, no. 34, 2016, pp. 1–7.

[6] V. Ivanova and A. Petrov, "Hybrid neural-matrix factorization for recommender systems," *IJISAE*, vol. 11, pp. 45–59, 2023.

[7] S. Rendle, W. Krichene, and L. Zhang, "Neural collaborative filtering vs. matrix factorization revisited," in *RecSys*, 2020, pp. 240–248.