COSC 520 Presentation: Recommender Algorithms

RILEY EATON, University of British Columbia, Canada
AARAV GOSALIA, University of British Columbia, Canada

1 Introduction

Explanation of Recommender Algorithms, the specific algorithms we will cover, and their use cases

2 Algorithmic Background

The evolution of recommender algorithms from simple rule-based systems to deep learning architectures has been driven by data growth, computational advances, and the economic value of personalization. This evolution provides context for examining the privacy, fairness, and ethical challenges of modern recommender systems.

2.1 Early Approaches: Content-Based and Demographic Filtering

Content-based filtering recommends items similar to those a user has previously liked, using item features and user preferences to compute similarity [6]. For example, recommending movies with similar genres or actors. While intuitive and privacy-preserving, these systems suffer from limited diversity and cannot leverage collective user patterns.

Demographic filtering recommends items based on age, gender, or location, assuming users in similar demographic groups share preferences. However, this naive approach often reinforces stereotypes, and its effectiveness is limited in a diverse, modern world. These limitations motivated the development of collaborative filtering.

2.2 Collaborative Filtering: Leveraging Collective Intelligence

Collaborative filtering marked a paradigm shift in recommender systems. Resnick et al. [7] introduced GroupLens, pioneering the idea that users who agreed in the past will likely agree in the future. User-based collaborative filtering identifies users with similar rating patterns and recommends items those users enjoyed. This enables serendipitous discovery and leverages collective intelligence. However, computing similarities between all user pairs becomes computationally prohibitive as the user base grows.

To address this issue of scalability, Linden et al. [5] developed item-to-item collaborative filtering at Amazon. This approach computes item similarities based on user co-interactions. Since items typically grow more slowly than users, and item similarities remain stable over time, they can be pre-computed. Despite these advances, collaborative filtering still faced cold-start problems (inability to recommend for new users or items with no interaction history) and sparse rating matrices (users rate only a tiny fraction of available items, leaving insufficient overlap for similarity computation).

2.3 Matrix Factorization: Uncovering Latent Factors

The Netflix Prize competition (held from 2006-2009) prompted the introduction of many novel matrix factorization techniques. Among the most influential contributions, Koren et al. [4]—whose work was central to the winning solution—decomposed the sparse user-item rating matrix into two lower-dimensional matrices representing users and items in a shared latent factor space. Mathematically, the rating matrix $R \in \mathbb{R}^{m \times n}$ (with m users and n items) is approximated as

Authors' Contact Information: Riley Eaton, ryeaton@student.ubc.ca, University of British Columbia, Kelowna, BC, Canada; Aarav Gosalia, agosalia@student.ubc.ca, University of British Columbia, Kelowna, BC, Canada.

$R \approx U \cdot V^T$, where $U \in \mathbb{R}^{m \times k}$ represents user latent factors, $V \in \mathbb{R}^{n \times k}$ represents item latent factors, and $k \ll \min(m, n)$ is the number of latent dimensions. Each row u_i in U is the latent factor vector for user i, and each row v_j in V is the latent factor vector for item j. The predicted rating for user i and item j is computed as $\hat{r}_{ij} = u_i \cdot v_j^T$, the dot product of their respective latent vectors [4].

This approach handles sparsity, provides dimensionality reduction for computational efficiency, and captures meaningful latent concepts. These latent concepts would otherwise not be discoverable through manual feature engineering, as they emerge automatically from user-item interaction patterns. Techniques like SVD and ALS became the standard methods for learning these factorizations [4]. However, matrix factorization assumes linear relationships and cannot easily incorporate temporal dynamics or rich metadata, motivating later exploration of non-linear approaches.

2.4 Deep Neural Networks: Modern Architectures

Deep learning revolutionized recommender systems by enabling the use of complex, non-linear representations. Covington et al. [2] described YouTube's two-stage architecture: candidate generation (producing user embeddings to retrieve candidates) and ranking (scoring candidates using hundreds of features). Neural networks model non-linear feature interactions, incorporate heterogeneous features (categorical, continuous, sequential), and leverage transfer learning. These capabilities enable models to capture complex user preferences that linear methods miss and combine diverse data types into a single framework, while pre-trained representations improve performance with limited data.

He et al. [3] introduced Neural Collaborative Filtering (NCF), which generalizes matrix factorization by replacing the dot product with a multi-layer network learning arbitrary interaction functions. Despite strong performance, deep recommenders require substantial computational resources, lack interpretability, and amplify privacy concerns by encoding societal biases from training data [9].

2.5 Current Trends and Hybrid Approaches

Contemporary recommender systems employ hybrid approaches combining collaborative filtering, matrix factorization, and deep learning. State-of-the-art techniques include graph neural networks for user-item-context relationships [8], reinforcement learning for long-term engagement [1], and context-aware recommendations. These advances improve recommendation quality but intensify privacy, fairness, and ethical concerns.

3 Analysis

Analysis of the drawbacks and advantages of the algorithm(s). Write a critical overview of whether the algorithm is practical (e.g., in terms of computational cost and time/space complexity), in which scenarios or application it can be used, where does it fail, etc.

4 Team Roles

Statement about the role of each team member. This is worth 20% of the report for some reason, so it should be detailed.

References

[1] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 456–464.

[2] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[3] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[4] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[5] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.

[6] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[7] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. 175–186.

[8] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *Comput. Surveys* 55, 5 (2020), 1–37.

[9] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

Acknowledgments

Anthropic's Claude was used to copy-edit this report. The complete source code and accompanying presentation are available at https://github.com/rileyeaton-ubc/ubc-520-presentation.