Stream to Success

—

An Exploratory Approach to Personalized Recommendation Systems

Meet cineSYNC, the outdated music library hoping to make a comeback. To compete against streaming platform giants like Netflix and HBO Max, we'll explore collaborative filtering recommendation systems to elevate cineSYNC's connection to user preferences.

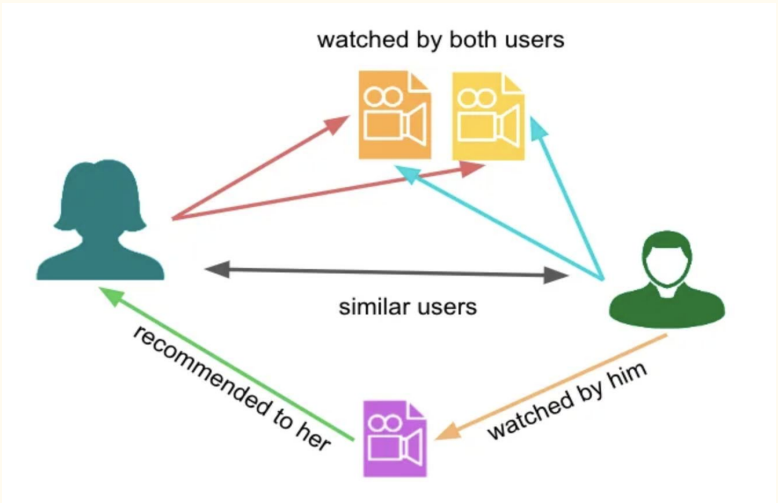
The Dataset

- ratings.csv
- movies.csv

Filtered (21k) version of the 100k MovieLens Dataset | GroupLens Research Lab | University of Minnesota

First, let's talk about the dataset. We are using a filtered version of the small MovieLens dataset from the GroupLens Research Lab out of the University of Minnesota. In modeling, we used the ratings.csv and movies.csv files to obtain user ratings and movie titles.

Collaborative Filtering



To explain what collaborative filtering means in recommendation systems, consider this example: if a person A likes items 1, 2, 3 and B likes items 2,3,4 then there is an overlap in interest. We can guess that A would like item 4 and B would like item 1*. By leveraging the wisdom of the crowd, so to speak, CF systems adapt dynamically to changing user behavior. We will be exploring 2 categories of CF systems, 4 total algorithms, evaluating the predictive power of each model and comparing results to see which performed best. Our target variable, or the variable our models will be making predictions on, will be user 'ratings'.

*Source: https://medium.com/@patelneha1495/recommendation-system-in-python-using-als-algorithm-and-apache-spark-27aca08eaab3

How we're evaluating our algorithms:

RMSE: Root Mean Squared Error

- Illustrates the difference between *predicted* ratings and *actual* ratings.
- The more each score approaches 0 (aka, *no* difference between predicted and actual ratings), the better.
- We are looking for a *low* RMSE score.

To evaluate all algorithms, we will use Root Mean Squared Error across the board. RMSE illustrates the difference between the model's predicted ratings and actual historical user ratings, showing a marginal error of prediction. A score of 0 would indicate there is no error at all, and the model is 100% accurate. This is highly unattainable, so we are looking for a score approaching zero. The lower the RMSE score, the better the model's performance.

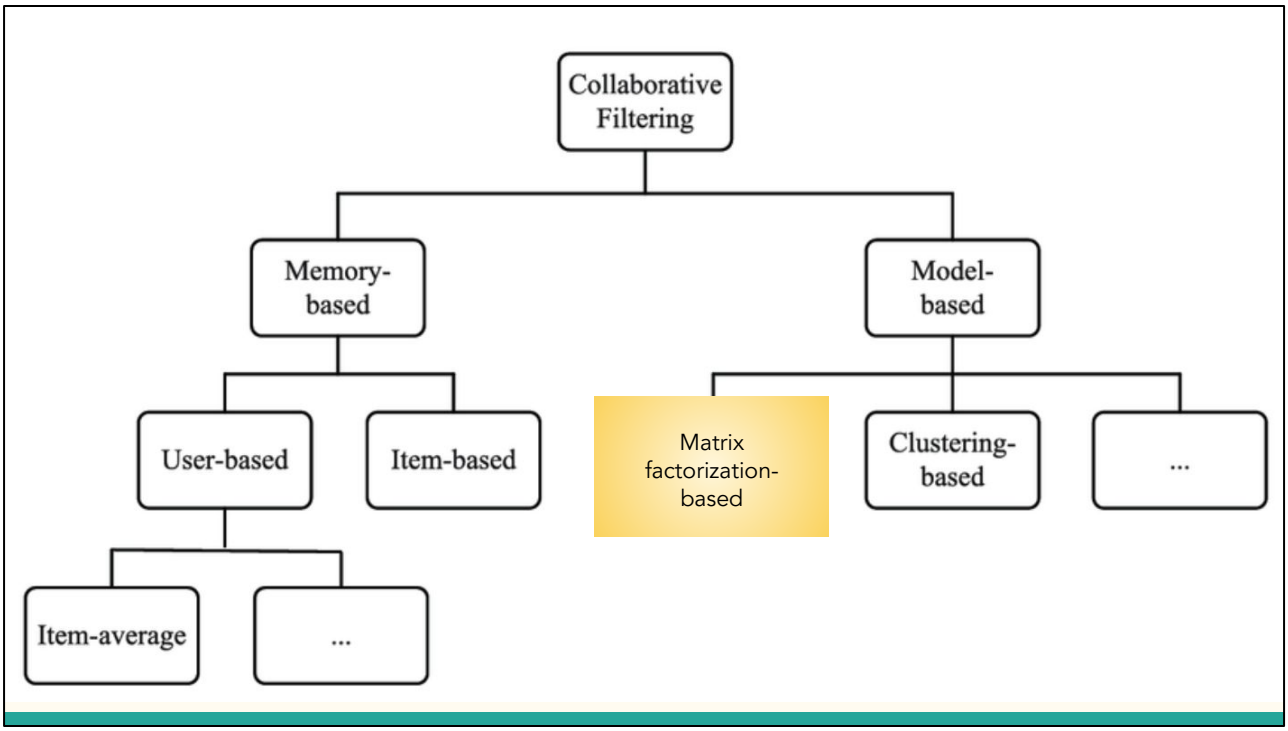
$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ are predicted values

$y_1, y_2, \ldots, y_n$ are observed values

n is the number of observations

The math behind RMSE: RMSE equals the square root of the average squared difference between actual and predicted values for a set of data points.

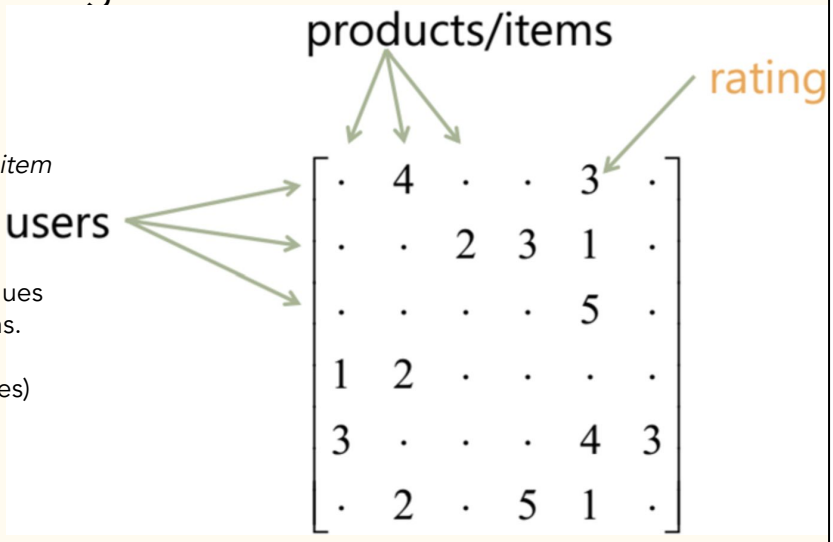


Like I said, we will be discussing 2 categories of CF systems. Let's first talk briefly about the Matrix Factorization algorithms we tested.

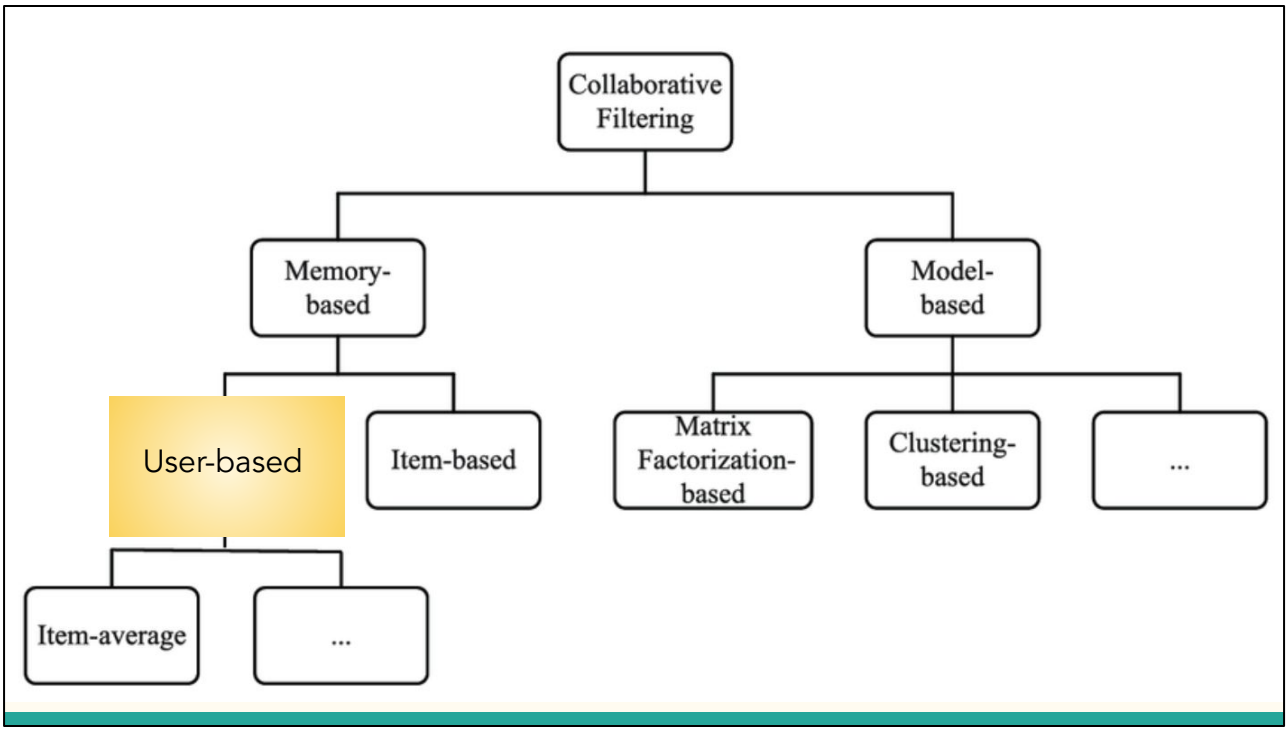
Matrix Factorization Algorithms

A process of decomposing a *user-item matrix*
into latent (underlying) factors, capturing hidden patterns and enabling prediction of missing values for personalized recommendations.

- ALS (Alternating Least Squares)
- SVD (Singular Value Decomposition)



A bit about Matrix Factorization: this is a process that focuses on imputing or calculating missing values within a user-item matrix, like the one you see above. Different processes of MF decompose and reassemble the matrix (this is factorization) in different ways to predict a user's rating for a movie title. We tested 2 MF algorithms: ALS (Alternating Least Squares) and SVD (Singular Value Decomposition), and evaluated them based on their RMSE scores, which we will present and analyze shortly.



I want to focus now on the second category of CF: memory based systems, specifically the user-based systems.

Memory-based
(e.g., *k*-nearest neighbors)

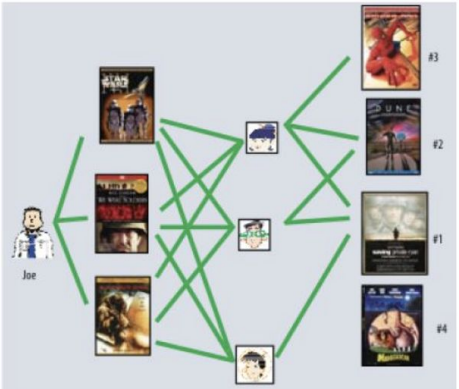


Figure 1. The user-oriented neighborhood method. Joe likes the three movies on the left. To make a prediction for him, the system finds similar users who also liked those movies, and then determines which other movies they liked. In this case, all three liked *Saving Private Ryan*, so that is the first recommendation. Two of them liked *Dune*, so that is next, and so on.

Model-based
(e.g., matrix factorization)

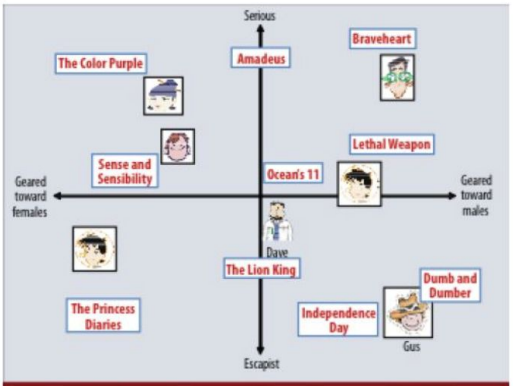
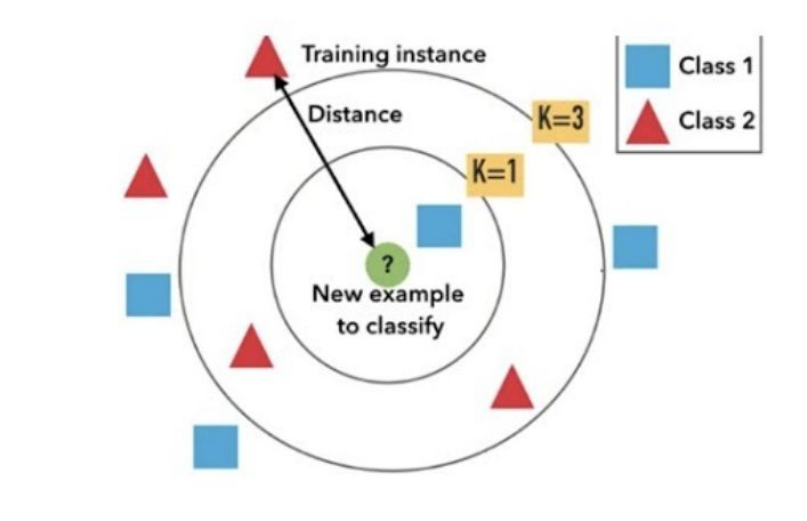


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Instead of relying on the user item matrix, as in MF, Memory-based systems directly use the observed user-item interactions to calculate similarities between users or items. For user-based CF, the algorithm identifies a set of users with similar preferences to the target user

Here is an example: User Joe to the left likes these three movies. To make a prediction for him, the system finds 3 users who also liked those movies, and then determines which other movies they liked. If all three of like a particular movie, say, Finding Nemo, then this would be the first recommendation. If 2 of them liked, say, Avatar, then that would be Joe's second ranked recommendation, and so on.

KNNWithMeans Algorithm



To visualize a knn with means algorithm, I'll present here a data space where a point represents a specific user. When we say "k-nearest neighbors", we are referring to the specific number (represented by k) of users that are most similar to the user in question. The algorithm computes the average ratings of these nearest user "neighbors" to predict ratings for unrated titles for this user.

Our Findings

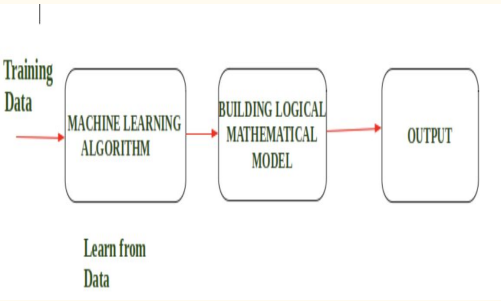
On average, the tuned KNN model's predictions deviate from *actual* ratings by around 0.32 stars. This is an improvement upon the ALS, SVD and first KNN model scores.

Model	RMSE Score
ALS	0.57
SVD	0.48
KNN	0.38
KNN2	0.32

So, after evaluating all models, we found that the KNN model performed best, and further tuning of the KNN model's hyperparameters (represented as KNN2) yielded an even lower RMSE score. You can see all the tested models and their corresponding RMSE scores reproduced to the right.

On average, the tuned KNN model's predictions deviate from *actual* ratings by around 0.32 stars. This is an improvement upon both the ALS and SVD models' scores.

KNN Output: User 59



Movie Title	Predicted Rating
Brazil	4.87
The Big Lebowski	4.86
The Shawshank Redemption	4.86
Life is Beautiful	4.86
12 Angry Men	4.84
Shaun of the Dead	4.84
The Usual Suspects	4.83
Eternal Sunshine of the Spotless Mind	4.82
Spirited Away	4.82
Requiem for a Dream	4.79

Here is a sample output of the KNN model at work for user 59.

Recommendations

- Use and refine the tuned KNN2 model
- To address the cold start problem, we suggest a content-based or hybrid approach

Here are some
- Use and refine the tuned KNNWithMeans (KNN2) model.
- Experiment with different values* for parameters like 'k,' 'min_support,' and 'shrinkage' to optimize configuration.

- Advantages:
  - Its interpretability and adaptability to changing user preferences make it well-suited for smaller datasets.
  - Unlike matrix factorization, which excels at handling sparse data and imputing missing values, KNNWithMeans provides transparency in recommendations, facilitating a clearer understanding of the underlying patterns.

- Disadvantages:
  - This model can be computationally expensive, especially as the dataset grows.
  - This model may also struggle with the cold start problem for new users or items with limited interaction history.
  - To address the cold start problem, we suggest a content-based or hybrid approach to provide content based recs for users with limited interaction. By combining content-based recommendations with k-NN with Means, you can provide meaningful suggestions even when there is insufficient interaction data.

Next Steps:

1. Scale up dataset.
2. Explore additional metrics, like NDCG or MAP.
3. Track user feedback.

1. To manage the scope of our project, we utilized a small subset of the available GroupLens data, around 21k data points out of 100k. Moving forward, we recommend expanding usage of the small dataset, or even venturing into a subset of the large available dataset (20M+) to capture a broader spectrum of user preferences.
2. While RMSE and MAE provide valuable insights into model accuracy, it's essential to consider alternative evaluation metrics such as NDCG (Normalized Discounted Cumulative Gain) or MAP (Mean Average Precision). These metrics may offer a more nuanced understanding of model performance, especially in scenarios where the emphasis is on the ranking quality of user recommendations.
3. Enhance the dataset by engineering features that capture user feedback and interaction. Implementing a simple thumbs-up or thumbs-down feedback option for quick responses could be one way to binarize user feedback instantly into a yes or no, true or false, 1 or 0 problem. Additionally, tracking implicit feedback could include logging the amount of time a user spends watching a recommended movie or the number of times they replay a specific scene to give a more nuanced view of the quality of a user's interaction with a movie.

Image Sources

https://medium.com/@ashmi_banerjee/understanding-collaborative-filtering-f1f496c673fd

https://www.researchgate.net/figure/Classification-of-collaborative-filtering-algorithms_fig4_303556519

https://docs.oracle.com/en/cloud/saas/planning-budgeting-cloud/pfusu/img/insights_rmse_formula.jpg

https://www.researchgate.net/publication/344710326/figure/fig1/AS:960349452902410@1605976560779/User-item-matrix-in-a-recommender-system.png

https://www.researchgate.net/profile/Serkan-Ayvaz/publication/338721765/figure/fig2/AS:894097657184256@1590180901710/Alternating-Least-Squares-Method.jpg

https://cdn.analyticsvidhya.com/wp-content/uploads/2019/07/svd.jpg1.jpg

https://abgoswam.files.wordpress.com/2016/06/cf.jpg

https://i.ytimg.com/vi/kccT0FVK6OY/maxresdefault.jpg

https://www.geeksforgeeks.org/design-a-learning-system-in-machine-learning/

Thank you. Let's Connect!

E-mail:
mbirchhn@gmail.com

Github:
www.github.com/madelinebirch

LinkedIn:
https://www.linkedin.com/in/madeline-birch-164000b5/

Questions?