Movie Recommendation System on MovieLens Dataset

New York University Center of Data Science Spring 2022 DS-GA 1004 Big Data Final Project

Adeet Patel
adeet@nyu.edu

Alexandre Vives
av2926@nyu.edu

Ilias Arvanitakis
ia2248@nyu.edu

1 Abstract

The purpose of this project is to create and evaluate a collaborative filtering recommender system with the Alternative Least Squares (ALS) model in Pyspark using the MovieLens dataset containing 27M movie ratings. Additionally, we are going to use a smaller version that contains 100k ratings before scaling up to the main one. An initial non-personalized popularity baseline model was created as a reference to evaluate how impactful personalization in recommender systems is. Then, our team completed two extensions. The first extension contains a single-machine comparison between LightFM and ALS regarding their runtime and accuracy. The second extension consists of doing qualitative error analysis through a brief investigation of the trends and genres of the users with the lowest prediction and training a UMAP model on the item latent factors produces by the ALS model.

2 Recommender System

2.1 Data Overview

The data used for this project was the popular MovieLens dataset, which contains 27M ratings from 280,000 users across 58,000 movies. The features assigned to each rating are the user ID, the movie ID, the rating (ranging from 0.5 to 5 stars) and a timestamp. Additionally, and for convenience purposes, a smaller version of this dataset was also used containing 100K ratings from 600 users for 9,000 movies.

2.2 Data Preprocessing

Before any model implementation, the data was split into training, validation, and testing samples. Instead of the common random split, our team took into consideration how the model selects items to recommend. The end goal is for our model to individually select 100 movies to recommend to each user by using part of its ratings to find other users with similar ratings (and thus similar taste in movies).

First, we split the users along with their ratings into three parts. The first part will contain 60% of the users (training set) and the remaining two have 20% each (validation and testing set). The model will be trained solely on the training set, but in order to make recommendations for the users in the validation and test sets, our model needs some ratings from those users to be able to find similar users on the training set, therefore we still need some of their ratings to be in the training set. Consequently, 30% of each user's ratings on the validation and testing sets were moved into the training set. Eventually, the training set includes all the ratings for 60% of the users and part of the ratings for the test and validation users.

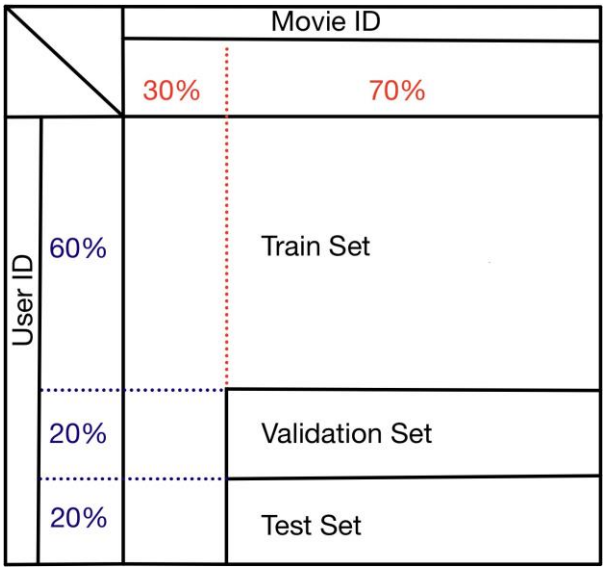


Figure 1: *Train-test split visualization*

Furthermore, the indexes of user and movie IDs were converted from string to integer as per the format required by the ALS model.

2.2 Model Implementation

2.2.1 Evaluation Metrics

Two metrics were chosen to evaluate the models: Precision at k, which calculates the number of relevant items present on the top-k recommendations of the model and Mean Average Precision (MAP), which calculates the number of relevant items present on the list of recommended items but taking into account its order (penalty for highly relevant items is higher).

Our team believes that MAP should be much more meaningful due to the typically low consumption of movies (compared to songs, for instance) and therefore making it more important to recommend the highest rated ones.

2.2.2 Popularity Baseline Model

The Popularity Baseline Model consists of selecting the highest rated movies across the training set and recommending them to every single user. This model served as a baseline for ALS in terms of the accuracy achieved by a non-personalized model (recommends a single list of movies to every user) compared to a personalized one (recommends a different list of movies to each user), which is useful to identify how much accuracy is due to the personalized recommendation.

The baseline model achieved a precision at k of 2.26% and a MAP of 0.127% for the small dataset and a precision at k of 0.156% and MAP of 0.0084% for the big dataset. Both measures are considerably higher for the small dataset, which is reasonable since we only have 9,000 movies and thus there is a higher chance of making correct recommendations just because we have a small pool of available choices. The large dataset has 58,000 movies and as a result there is a lower chance of success when suggesting the same movie to everyone. Therefore, there is the need to use a more sophisticated approach that considers the preferences of each user.

2.2.3 Alternating Least Squares (ALS)

Alternating Least Squares is a matrix factorization algorithm that aims to factorize a matrix into a product of two matrices. Specifically, the objective is to factorize the utility matrix into the product of a user and an item matrix, which initially contain random values named latent factors and get iteratively updated to approximate the utility matrix.

2.2.4 Hyper-parameter tuning

During the optimization process of the ALS hyper-parameters, our team focused on rank (columns of latent factors assigned to both the user and item matrices) and the regularization parameter.

The strategy implemented consisted of testing on the small dataset a broad range of both the rank and the regularization parameter in order to first cover the largest range possible and then zoom into the better performing values on subsequent runs.

The following graphs show the changing accuracy, represented by both precision at k and Mean Average Precision (MAP) while keeping the regularization parameter constant at 0.01 and changing the rank.

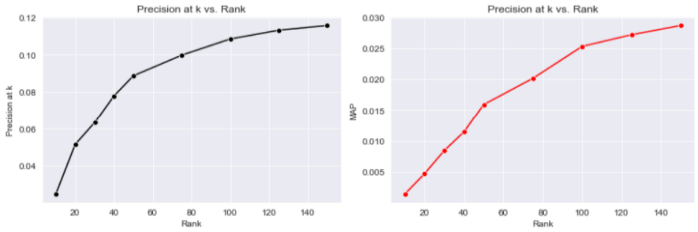


Figure 2: *Precision at K and MAP vs. Rank*

Notice a steady increase in accuracy along with the rank increase due to the fact that increasing the rank means increasing the number of latent factors and therefore the complexity capacity of the model. Eventually, on ranks above 110, it plateaus, which combined with the

increasingly long runtimes convinced us to select a rank of 110.

Similarly, the following graphs show how the accuracy changes while varying the regularization parameter while keeping the rank constant at 110.

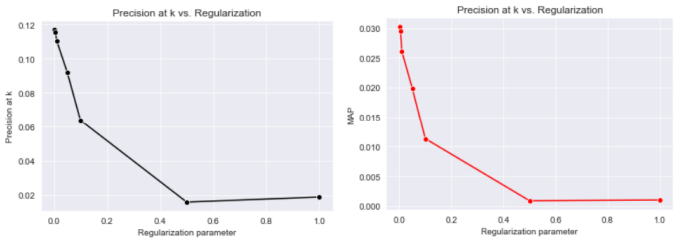


Figure 3: *Precision at K and MAP vs. Regularization*

Eventually, the ALS model with optimized hyperparameters yields a precision at k of 11.77% and a MAP of 3.02%.

Hyperparameter	Optimal Choice	Tuning Range
RegParam	0.01	0.001, 0.01, 0.1
Rank	110	50 to 115

Figure 4: *Hyper-parameter ranges for ALS model*

We then used these hyperparameters to train the ALS model on the full dataset and evaluate it. We achieved a precision at k of 1.183% and a MAP of 0.376%.

3 Extension 1: LightFM vs. ALS single-machine comparison

The purpose of this extension is to compare the accuracy and runtime difference between the ALS model and the LightFM model on a single machine, with a variable training set size.

LightFM is a hybrid matrix factorization algorithm, which means that it uses both collaborative and content-based filtering. Even though it is able to use both implicit and explicit feedback, our data restricts us to use the implicit feedback approach, for which lightFM offers two methods: Bayesian Personalized Ranking (BRP) and Weighted Approximate-Rank Pairwise (WARP).

Before comparing both models, in order to have a fair comparison, hyper-parameter tuning was performed on the following LightFM hyper-parameters and ranges:

Hyperparameter	Optimal Choice	Tuning Range
Number of Components	32	5, 25,32,40,50,75,100
Learning Schedule	adadelta	adagrad, adadelta
Loss	Bpr	Bpr, Warp
Learning Rate	0.5	0.001, 0.01, 0.1, 0.25, 0.5, 1
Item Alpha	1e-08	1e-5, 1e-8, 1e-10
User Alpha	1e-12	1e-10, 1e-12, 1e-14
Max Sampled	5	5, 10

Figure 5: *Hyper-parameter ranges for LightFM model*

Once the best hyper-parameters for LightFM are identified, both models are compared in terms of both their precision at k and runtime.

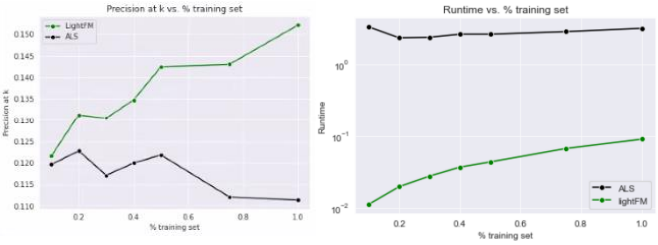


Figure 6: *ALS model vs. LigthFM model performance*

Figure 6 shows the performance of both models on the validation set after being trained in increasing training set sizes. It is clear that LightFM with a Bayesian Personalized Ranking (BRP) loss performs better than the ALS model and it runs in a shorter amount of time. With lightFM being used for a single machine implementation, we decided to only use the small dataset for this purpose. The big dataset being excessively large, the single machine implementation was not possible and as a result we decided to limit our approach. However, we can clearly see the superior performance of the lightFM model compared to the ALS.

4 Extension 2: Qualitative error analysis

4.1 Investigating the trends of the lowest-scoring predictions

Once the worst predictions made by the model were identified (highest squared error between the ground truth rating and the model's predicted rating), our team discovered that most of the worst predictions were associated to movies released before the year 2000. Specifically, 30 out of the 50 predictions with the highest square loss were associated to movies with a release date before the year 2000 and the remaining 20 movies have a release date after the year 2000. Additionally, the genres of Comedy, Action and Drama underperformed considerably compared to all other genre, as they were predominant among the lowest scored predictions.

4.2 Visualizing the item latent factors via UMAP

UMAP is a powerful dimensionality reduction tool used to understand and visualize high-dimensional data. At its core, it constructs a high dimensional graph of the data and then designs a low-dimensional graph to be as structurally similar as possible.

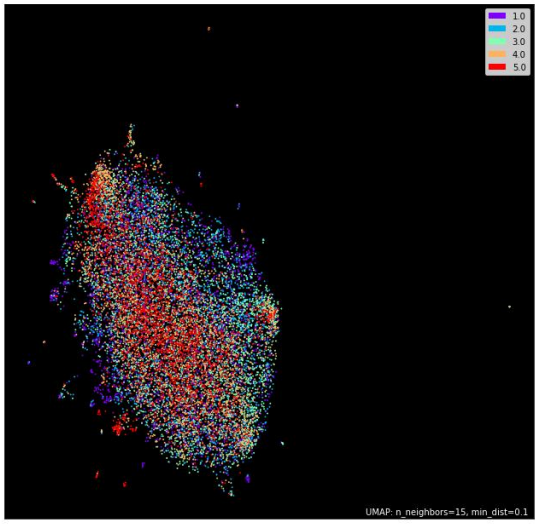


Figure 7: *UMAP model visualization*

The figure above depicts the UMAP of the dimensionality reduced item latent factors with its default hyperparameters: *n_neighbors* = 15, *min_dist* = 1 and *spread* = 1. Each color represents a rank assigned to each movie based on the value and number of reviews received. The ranking for each movie was obtained by using the following formula:

Where S is the sum of all ratings and N is the number of ratings.

$$\frac{a}{a+b} - 1.65\sqrt{\frac{ab}{(a+b)^2(a+b+1)}} \qquad a = 1+S \qquad b = 1+N-S$$

Figure 8: *Ranking equation used for popularity*

Hyperparameter tuning is very important in UMAP and it depends on the data and the project's goals. Our team considered the three most commonly used hyper-parameters: *n_neighbors*, *min_dist* and *spread*, which have the function of controlling the balance between local and global structure in the final projection.

The most important hyper-parameter is n_neighbors, which represents the number of approximate nearest neighbors used to construct the initial high dimensional graph. It sways the balance between the local and global structure - low values will increase the focus on the local structure while larger values will pivot importance to the overall structure.

The second hyper-parameter we will tune is *min_dist*, which represents the minimum distance between points in the low dimensional space. Mainly, this parameter controls how tightly UMAP brings the cluster points together leading to denser clusters.

Lastly, the *spread* hyper-parameter is used in conjunction with *min_dist*, as it represents the effective scale of the data points and thus controlling the size of the clusters.

Hyperparameter	Optimal Choice	Tuning Range
N-Neighbors	1000	15, 20, 50, 200, 1000
Min-dist	0.1	0.1, 0.25, 0.5, 0.75, 1
Spread	0.1	0.1, 0.15, 0.25, 0.5, 1

Figure 9: *Hyper-parameter ranges for UMAP model*

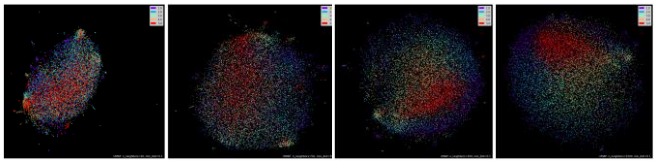


Figure 10: *UMAP visualization varying n_neighbors*

The figure above shows the changes in the UMAP plots when increasingly varying the *n_neighbors* parameter while keeping all other hyper-parameters at their default values. Notice how the clusters, while keeping its internal structure, are separated from each other.

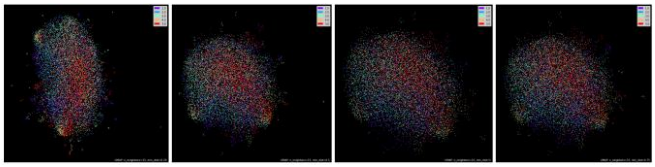


Figure 11: *UMAP visualization varying min_dist*

Similarly, in Figure 11 we keep all hyper-parameters constant except *min_dist*. Even though the clusters are not well defined, most probably due to the low value for *n_neighbors*, it is evident that there is an increase in the variance within each cluster, expanding the space each occupies accordingly.

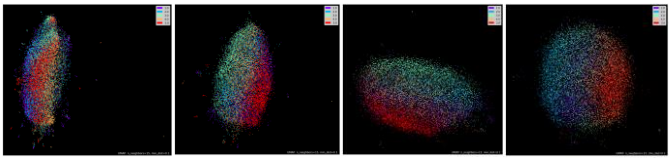
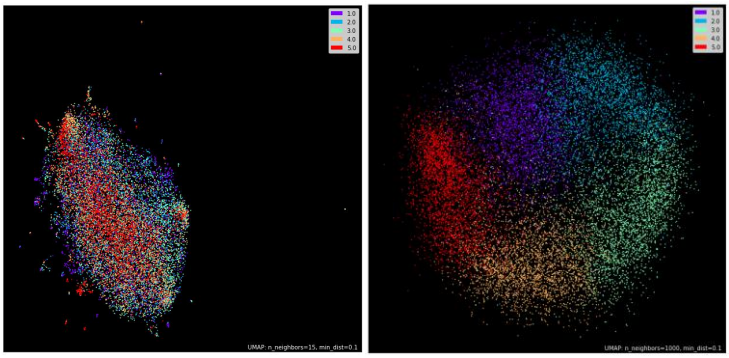


Figure 12: *UMAP visualization varying spread*

Finally, in Figure 12 we keep all hyper-parameters constant except *spread*. It is visible how the spread of the data points increases while keeping the inter-cluster distance closer enough to maintain a similar cluster size.

4.3 UMAP results

The following two figures show the transition from the initial UMAP with default parameters to the hyper-parameter tuned UMAP.



The optimized UMAP plot is clearly able to differentiate movies grouped by the quality of their reviews (on other words, their popularity). Additionally, there is an overlap over clusters which is understandable as the least popular movie from a rank is very similarly equally popular to the most popular movie from the next lower rank.

References

[1] "MovieLens Latest Datasets", 2018, https://grouplens.org/datasets/movielens/latest/

[2] "Welcome to Spark Python API Docs!", https://spark.apache.org/docs/3.0.1/api/python/index.html

[3] "UMAP API Guide", https://umap-learn.readthedocs.io/en/latest/api.html#umap

[4] "LightFm Documentation", https://github.com/lyst/lightfm

[5] "Understanding UMAP, Andy Coenen and Adam Pearce", https://pair-code.github.io/understanding-umap/

[6] "Light FM Recommendation System Explained", Pio Calderon, 2018, https://medium.com/@piocalderon/light-fm-recommendation-system-explained-9c8bb382d9e0

[7] "How Exactly UMAP Works and why exactly it is better than tSNE", Nikolay Oskolkov, 2019, https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668

[8] "Evaluation Metrics for Recommender Systems", Claire Longo, 2018, ttps://towardsdatascience.com/evaluation-metrics-for-recommender-systems-df56c6611093

[9] "Bayesian Statistics for Ranking", https://nbviewer.org/github/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers/blob/master/Chapter4_TheGreatestTheoremNeverTold/Ch4_LawOfLargeNumbers_PyMC3.ipynb#Example:-How-to-order-Reddit-submissions

Appendix A: Contributions

Tasks	Contributions
Train, Test and Validation Split	Adeet Patel, Alexandre Vives
Baseline Model and Evaluation	Alexandre Vives, Ilias Arvanitakis
ALS Training and Tuning	Adeet Patel, Alexandre Vives
ALS Evaluation	Ilias Arvanitakis, Adeet Patel
LIghtFM Training and Tuning	Alexandre Vives, Adeet Patel
LightFM vs ALS Comparison	Ilias Arvanitakis, Alexandre Vives
Investigating Trends for lowest-scoring predictions	Adeet Patel, Ilias Arvanitakis
U-MAP Visualization	Alexandre Vives, Ilias Arvanitakis
Report Writeup	Adeet Patel, Alexandre Vives, Ilias Arvanitakis

Appendix B: File organization

Popularity Baseline Model
model_baseline.py

Recommender System
Train_test_validation_script.py
Model_als.py

Extension 1: ALS vs. LightFM

Model_extension1_lightfm.py

Extension 2: Error Analysis + UMAP visualization

Latent_factors.csv

Model_extension2_trends.py
Model_extension2_umap.py