Comparative Study of Music Recommendation Systems

Ramanan Durairaj, 1219512242

rduraira@asu.edu

Abstract – This project compares music recommendation systems that use the Approximate Nearest Neighbors algorithm (Spotify Annoy) and collaborative filtering which employs Matrix Factorization and Singular Value Decomposition (SVD).

I. INTRODUCTION:

During the last decade, music has shifted from entertainment to service. Many modern streaming services, such as Spotify and Apple Music, allow us to listen to our favorite tunes with only a swipe or a button press. That's all there is to it. Besides maintaining our favorite tracks, such streaming sites can recommend and thereby introducing us to new music. The recommendation algorithm learns from the numerous songs we listen to and uses machine learning to train the model to recommend new music of similar genres. Every platform is unique because it employs various methods for content recommendation; some are well-received, while others are not.

In this project, we compare two types of recommendation algorithms: Approximate Nearest Neighbors and Matrix Factorization-based recommendation algorithms, in order to determine which is more suited to efficiently recommend songs to listeners. We would ideally use elements of crucial relevance, such as recommendation accuracy, as a decision parameter and derive conclusions from it.

Collaborative Filtering:

Collaborative filtering is a technique used by recommender systems. In a larger sense, collaborative filtering is a means of gathering preferences or taste information from a large number of users to create automatic predictions (filtering) about a user's interests (collaborating). The collaborative filtering technique is based on the idea that if person A and person B share the same opinion on a subject, A is more likely than a randomly chosen person to have B's perspective on a separate topic. In a larger sense, collaborative filtering is the process of searching for information or patterns using techniques that require collaboration among multiple actors, viewpoints, data sources, and so on.

Recommendation based on matrix factorization:

In this method, the user-item matrix is decomposed into a product of two lower-dimensional matrices. The creation of latent features occurs when two independent objects are multiplied. In collaborative filtering, matrix factorization is used to find correlations between users and item entities. The techniques of stochastic gradient descent and alternating least squares can be used for Matrix factorization.

Recommendation based on approximate Nearest Neighbors:

Nearest neighbor search(NNS) is the task of finding the point in a set that is closest (or most similar) to a given point. Dissimilarity functions are frequently used to represent similarity: the larger the function values, the less similar the items are. This method is often used to handle problems such as pattern recognition, classification, computational geometry, recommendation systems, plagiarism detection, cluster analysis, chemical similarity identification, and other obstacles. Spotify's Annoy recommendation system, which is a well-known music streaming company, uses this algorithm.

II. DESCRIPTION:

A recommendation system is a type of information filtering system that tries to predict how a user would evaluate or prefer a certain item. In layman's terms, it's an algorithm that recommends products to consumers based on their interests.

Inverse indexing, an information retrieval approach that takes a feature and maps it to those vectors that possess that feature, is used by recommendation systems to accomplish this. It may, for example, take a word at the conclusion of a book and link to all of the chapters that contain that word. Recommendation systems can be divided into three categories: Content-Based

Recommendation Systems, Collaborative Filtering Systems, and Hybrid Systems (Cumulative of the above systems)

The activities of a user are used by collaborative filtering systems to recommend other items. Collaborative filtering systems can be user-based or item-based in general.

- User-based systems: These systems make product recommendations based on the preferences of users who are similar to one another.
- Item-based systems: These systems make recommendations based on the patterns of users who have visited the same item as another user.

i. **Collaborative Filtering with Approximate Nearest Neighbors:**
Similar vectors are found over the entire feature space produced by all of the data points' features, projected onto a high-dimensional space, by recommendation systems. The majority of data points in multi-dimensional space are projected as vectors.
Many metrics, such as Euclidean Distance, dot product, and cosine angle, are used to find "SIMILARITY" between two data points. The approach described above is known as Approximate Nearest Neighbor (ANN) search or Similarity Search.
Among the applications are model-free classification, pattern recognition, collaborative filtering for recommendation, and data compression.

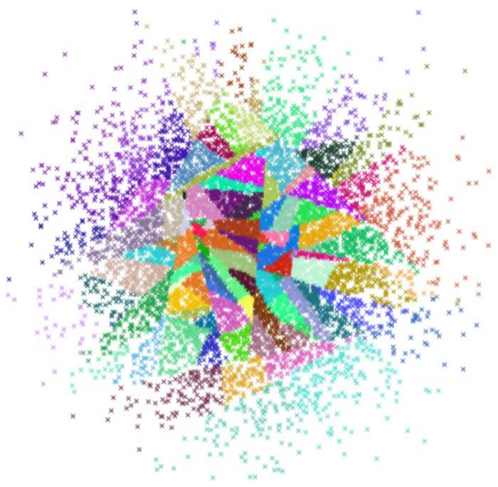


Figure 1 shows within ANNOY, how it separates hyperplanes and points

The model's main features:

- k-Nearest Neighbors (kNN) is an optimization problem that involves finding the points in a given set that are closest to a given point.
- KNN becomes unfeasible in most modern-day applications with large datasets and high dimensionality since it requires an exhaustive search.
- The Curse of Dimensionality - As the number of data points and dimensions increases, all of them become equidistant from each other.
- In exchange for a huge improvement in speed, we'll have to forgo some precision.

Model Information:

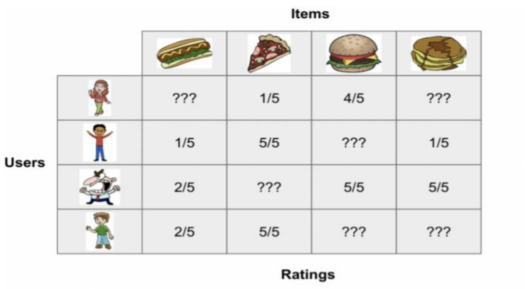
To generate trees in Approximate Nearest Neighbors, the dataset is partitioned into subgroups. Each tree is created by randomly selecting two sites and dividing the space in half using their hyperplanes. Split the space recursively into subspaces until the number of points connected to a node is small enough. The tree is traversed to obtain a set of candidate points for searching for the newly formed index; the query's closest point is returned. As the number of binary trees created increases, the output accuracy improves.

ii. **Collaborative Filtering with Matrix Factorization:**

Matrix factorization is a well-known method in the field of recommender systems. When dealing with recommendation systems, matrix factorization approaches are intensively researched. A group of m entities maps to a group of n other entities in this case. A common example is user-generated music suggestions. The goal is to calculate the distance and proximity between a piece of music and a user as if they were two distinct things. To give a representation that can be compared and matched, both music and users must be translated to vectors of the same dimension.

Matrix factorization is a popular and effective technique for developing a recommendation system. Rather than looking at every item a user has ever rated, matrix factorization tries to figure out how to quantitatively characterise both users and items in a lower-dimensional space so that these characterizations can be used to predict user behaviour and preferences for a known set of possible items. Matrix factorization has gained popularity in recent years because to its precision and scalability.

Each row in the matrix represents a distinct user, each column represents a different item, and each entry at location ij in the matrix represents user i's rating for item j.


Items
Users
Ratings

	???	1/5	4/5	???
	1/5	5/5	???	1/5
	2/5	???	5/5	5/5
	2/5	5/5	???	???

Figure shows recommendation matrix for User-item

As a result, we can deconstruct a matrix by determining which two matrices can be multiplied to form the original matrix. But, in a recommendation system, how does matrix factorization work?

When these two matrices are formed, they each have their own set of information about users, items, and their relationships.

One of the matrices will store information about individuals, while the other will store information about things. Each row of the user (left) matrix is a k-dimensional vector that quantifies a single individual, while each column of the item (right) matrix is a k-dimensional vector that characterizes a single thing. The latent dimensionality (or embedding size) of these vectors, k, is a hyperparameter in the matrix factorization model that must be tuned – The model can capture more complicated relationships and store more data with a greater latent dimensionality, but it also increases the danger of overfitting.

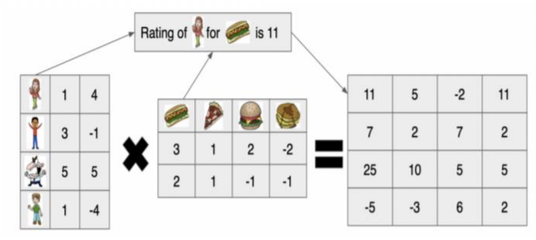


Figure 3 shows Matrix Factorization with User - Item matrices

User I's rating for item j is the inner product of user i's vector from the left matrix and item j's vector from the right matrix. This may appear strange at first, but it makes a lot more sense when you understand how the matrices are multiplied to form the user-item matrix. As a result, if the vector embeddings for each user and item are trained to store useful, characteristic information, the inner product can accurately predict a user-item rating based on an item's characteristics and a user's preferences for those characteristics, all of which are stored in the embedding vector's values.

Singular Value Decomposition:

The matrix factorization approach we employed was Singular Value Decomposition (SVD). The main purpose of SVD is dimensionality reduction, or the reduction of high-dimensional data to low-dimensional data. A matrix is factorized into a sequence of linear approximations using Singular Value Decomposition. These approximations will show the matrix's fundamental structure. SVD can be seen from three different perspectives.

To begin, SVD converts a matrix of clearly connected variables into an uncorrelated matrix, allowing for a better comprehension of the relationship between all data points that might not be obvious in their original configurations. This is useful since song relationships can be confusing at times, or it can imply an unspoken relationship between two songs.

Second, SVD is used to determine the association between distinct items (represented as data points in the matrix) and aligns the data in the product matrix so that the data points with the most variation have the most variance. Finally, once the vectors with the biggest variances have been identified, we may search the original data using fewer dimensions to find the best approximations.

The SVD approach is extremely beneficial for dimensionality reduction. SVD applications include signal processing, Latent Semantic Analysis, Pattern Recognition, Low Range Matrix Approximation, and Weather Prediction, to name a few.

The steps involved in creating the model are as follows:

- Data collection - data was gathered and imported from a dataset; data values and labels were combined to create data points.
- Preprocessing of data - Users that don't utilise the service on a regular basis were removed. Only popular songs, which are heard by everyone, are considered. The parameter: play count was used to filter the results.
- Implementing a Recommendation Engine - Convert a data frame to a sparse matrix, perform the matrix factorization, and find latent features Obtain the matrix that is estimated.

III. FINAL RESULTS

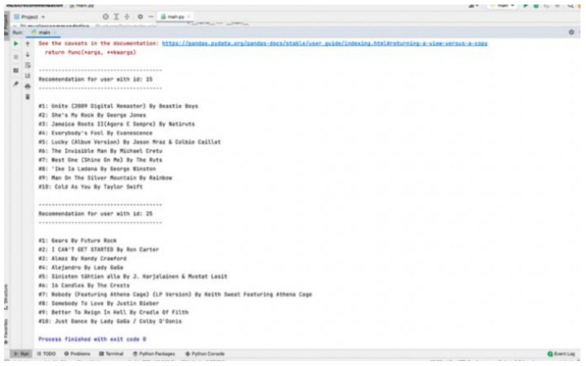
K-Nearest Neighbors:

[With Fashion MNIST data] Highest Accuracy attained: 91.22 percent

[With 10k data points trained], the highest precision was achieved: 97.20 percent.

[With glove-25-angular data] Lowest Accuracy attained: 60.98 percent

Matrix Factorization:



```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return func(*args, **kwargs)

--------------------------------------
Recommendation for user with id: 15
--------------------------------------

#1: Unite (2009 Digital Remaster) By Beastie Boys
#2: She's My Rock By George Jones
#3: Jamaica Roots II(Agora E Sempre) By Natiruts
#4: Everybody's Fool By Evanescence
#5: Lucky (Album Version) By Jason Mraz & Colbie Caillat
#6: The Invisible Man By Michael Cretu
#7: West One (Shine On Me) By The Ruts
#8: 'Ike Ia Ladana By George Winston
#9: Man On The Silver Mountain By Rainbow
#10: Cold As You By Taylor Swift

--------------------------------------
Recommendation for user with id: 25
--------------------------------------

#1: Gears By Future Rock
#2: I CAN'T GET STARTED By Ron Carter
#3: Almaz By Randy Crawford
#4: Alejandro By Lady GaGa
#5: Sinisten tähtien alla By J. Karjalainen & Mustat Lasit
#6: 16 Candles By The Crests
#7: Nobody (Featuring Athena Cage) (LP Version) By Keith Sweat Featuring Athena Cage
#8: Somebody To Love By Justin Bieber
#9: Better To Reign In Hell By Cradle Of Filth
#10: Just Dance By Lady GaGa / Colby O'Donis

Process finished with exit code 0
```

Figure 4 shows the results of the CF with Matrix Factorization model.

Comparison:

Outcome of Collaborative Filtering with ANN (ANNOY)

fashion-mnist-784-euclidean accuracy: 91.22%

glove-25-angular accuracy: 68.90%

nytimes-16-angular accuracy: 80.86%

limit: 100 precision: 20.98% avg time: 0.000074s

limit: 1000 precision: 56.60% avg time: 0.000276s

limit: 10000 precision: 97.20% avg time: 0.001828s

Outcome of Collaborative Filtering with Matrix Factorization:

Coverage: 58.29%

Average execution time: 18-20s

Our algorithm generates music recommendations that can be utilized to create future recommendations for our consumers.

Following a recommendation, the user must choose and listen to a song, and further recommendations can be made based on their listening habits.

IV. PERSONAL CONTRIBUTION TO THE PROJECT

With the growth of Spotify, Youtube, Amazon, Netflix, and other similar web services over the last few decades, recommender systems have become increasingly important in our lives. Recommender systems are becoming ubiquitous in our daily online trips, from e-commerce (suggest articles to buyers that may be of interest) to online advertising (suggest the suitable contents to users based on their preferences). This encouraged us to choose recommendation systems as our project topic because it would provide us with a wonderful opportunity to work and learn this field.

My key contributions to this study project were conducting research on all algorithms suitable for building a music recommendation system by reviewing journal publications and conference papers. I also conducted considerable study into the parameters that might be used to compare and evaluate the efficiency and effectiveness of the algorithms chosen to build the music recommendation system. I attended weekly meetings/discussions in addition to writing and studying the topic. I went to millionsongdataset.com and found an appropriate dataset that could be used for our project, then I prepped and processed the data to make it ready to use. We split our group into two groups to work on separate algorithms for the system's development. We developed the system in Python as part of a team working on a music recommendation system with Collaborative filtering utilising matrix factorization. The factorization of the matrix was determined using Singular Value Decomposition (SVD).

I have made other improvements to this project, such as changing the python code to only deliver recommendations based on the songs the user listened to in the previous year. Because the dataset lacked the year 2021, I queried the maximum of listening year to obtain the most recent year, then filtered the dataset to remove the data points with the listening year not equal to the most recent year. When the new dataset was used to run the algorithm, the accuracy stayed almost the same, but the running time was cut in half, nearly tenfold.

V. LESSONS LEARNED AND TEAM MEMBERS

1) Lessons Learned and Acquired Skills:

- Gained knowledge in designing and implementing a recommendation system for a music streaming service.
- I studied about several approaches and machine learning algorithms to build this model.
- Acquired expertise in python and python libraries such as Pandas, Numpy, etc.
- To prepare data for use in a model, I learned how to gather, clean, process, and analyse a dataset.
- To work on this project, I also learned to manage my time and collaborate with team members.

2) Team Members

- Ramanan Durairaj
- Dhruv Malhotra
- Sharanya Banerjee
- Tapas Kar
- Pavan Kumar Mannam
- Ruthwik Reddy Ratna

VI. REFERENCES

[1] Wolfe, C., 2019. Building a Music Recommendation Engine with Probabilistic Matrix Factorization in PyTorch. [online] Medium. Available : https://towardsdatascience.com/building-a-musicrecommendation-engine-with-probabilistic-matrix-factorization-inpytorch-7d2934067d4a [Accessed 30 September 2021].

[2] "Collaborative filtering - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Collaborative_filtering. [Accessed: 30 September 2021]

[3] "implicit vs annoy - compare differences and reviews?", Libhunt.com, 2021. [Online]. Available: https://www.libhunt.com/compare-implicit-vs-annoy. [Accessed: 30 September 2021]

[4] V. (2021, December 20). Recommendation Systems Explained - Towards Data Science. Medium. https://towardsdatascience.com/recommendation-systems-explained-a42fc60591ed

[5] Pykes, K. (2021, December 15). 3 Approaches To Building A Recommendation System - Towards Data Science. Medium. https://towardsdatascience.com/3-approaches-to-build-a-recommendation-system-ce6a7a404576

[6] Longo, C. (2018, December 3). Evaluation Metrics for Recommender Systems - Towards Data Science. Medium. https://towardsdatascience.com/evaluation-metrics-for-recommender-systems-df56c6611093