# MUSIC RECOMMENDATIONS

Using the Dot Product to Make Predictions towards Calculating Music Recommendations

*Hirsch, Jacqueline Erin*

*MTH520: Numerical Linear Algebra | University of Miami*

## Contents

## Introduction

Music recommendation algorithms are an important tool for helping users discover new songs and artists that they might like, based on their previous listening history and preferences. In recent years, there has been a growing interest in developing algorithms that can accurately predict the ratings and preferences of users and generate personalized recommendations that are tailored to each individual user.

One approach to generating these recommendations is to use the dot product, which is a mathematical operation that allows for the comparison of two vectors and can be used to determine the similarity between them. In the context of music recommendation, the dot product can be used to calculate a recommended score for each user, based on their previous listening history and preferences, and the characteristics of the songs that are being recommended.

In this paper, we present a music recommendation algorithm that uses the dot product to calculate a recommended score for each user and generate personalized recommendations. We describe the details of our algorithm and evaluate its performance using a small dataset that I created. Our results show that our algorithm can generate recommendations.

## Definition of Dot Product

The dot product, also known as the scalar product or inner product, is a mathematical operation that takes two vectors and returns a scalar value. It is defined as the product of the magnitudes of the two vectors and the cosine of the angle between them.

More formally, the dot product of two vectors **a** and **b** is defined as:

$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| * |\mathbf{b}| * \cos(\theta)$

where $|\mathbf{a}|$ and $|\mathbf{b}|$ are the magnitudes of the vectors **a** and **b**, and $\theta$ is the angle between them.

The dot product has the following properties:

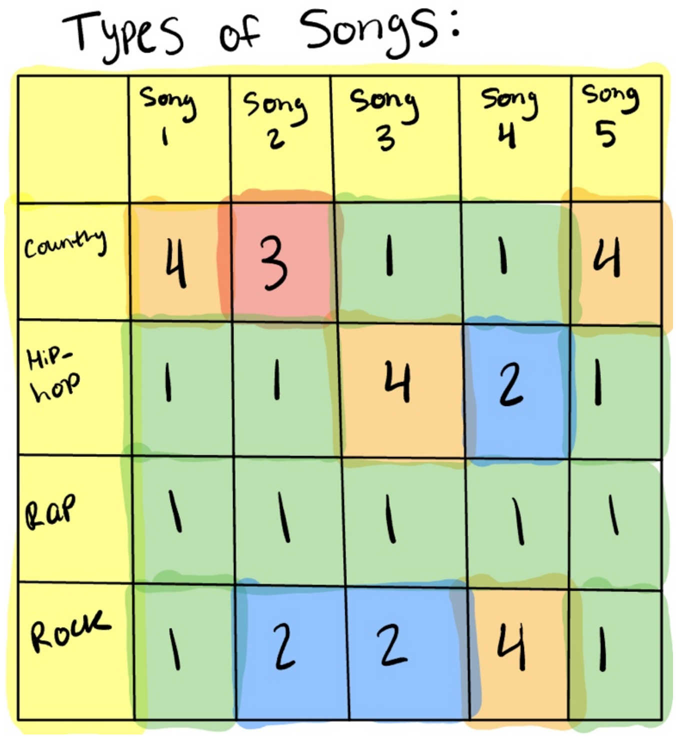
1. It is commutative, meaning that $\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}$ for any vectors **a** and **b**.
2. It is distributive over vector addition, meaning that $\mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}$ for any vectors **a**, **b**, and **c**.
3. It is associative over scalar multiplication, meaning that $\mathbf{a} \cdot (\mathbf{kb}) = (\mathbf{ka}) \cdot \mathbf{b}$ for any vector **a** and scalars **k** and **b**.

The dot product is commonly used in physics and engineering to calculate the projection of one vector onto another, as well as to determine the angle between two vectors. It can also be used in more abstract settings, such as making music suggestions, as described above.

In the context of making music suggestions, the dot product can be used to calculate the similarity between different songs or artists. In this paper, I will focus on using the dot product to find similarity of songs depending on the how well it fits into different genres of music.

## Goal

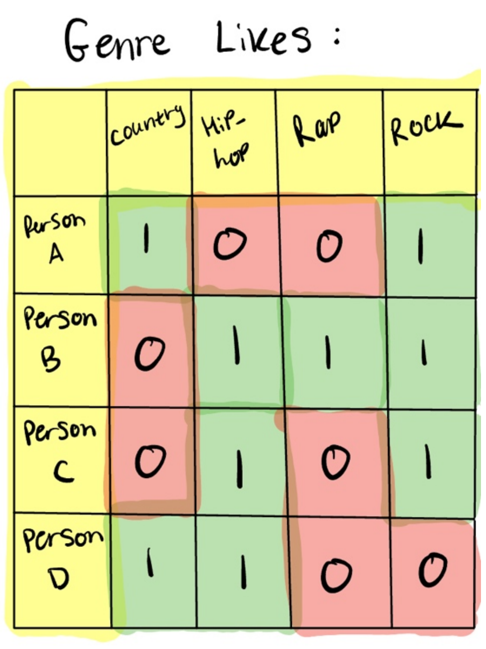
The goal of this project was to show how the dot product can be used to give music recommendations to users. To achieve this, I used two matrices. The first matrix represented how strongly a song aligned with a genre. In this matrix you will be able to see that a song can be aligned with more than one genre but there will be a genre that a song will align to the most. Below is an example of the Types of Songs Matrix.



Types of Songs:

| | Song 1 | Song 2 | Song 3 | Song 4 | Song 5 |
|---|---|---|---|---|---|
| Country | 4 | 3 | 1 | 1 | 4 |
| Hip-hop | 1 | 1 | 4 | 2 | 1 |
| Rap | 1 | 1 | 1 | 1 | 1 |
| Rock | 1 | 2 | 2 | 4 | 1 |

In the Types of Songs Matrix, a one means that the song is not at all aligned to that genre whereas a four represents that a song very aligned to that genre. For example, song two is a mix of country and rock but the song mostly aligns to being a country song. The next matrix represents whether users like a genre or if they do not like a genre. In this matrix, you will be able to see what genres a user likes, and a user can like more than one genre. Below is an example of the Genre Likes Matrix.



Genre Likes :

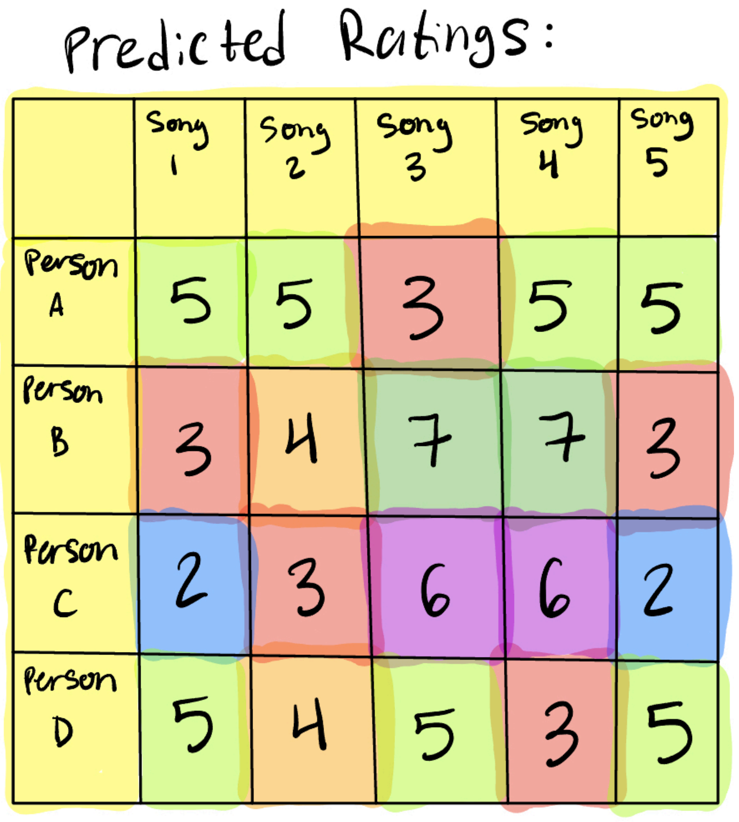
| | country | Hip-hop | Rap | Rock |
|---|---|---|---|---|
| Person A | 1 | 0 | 0 | 1 |
| Person B | 0 | 1 | 1 | 1 |
| Person C | 0 | 1 | 0 | 1 |
| Person D | 1 | 1 | 0 | 0 |

In the Genre Likes Matrix a zero entry means that a user does not like that genre whereas a one entry means that a user does like that matrix. For example, in the above matrix you can see that person A does like country music and rock music, but they do not like to listen to hip-hop and rap music.

To complete my goal, I will be performing the dot product on these two matrices in order to predict what songs users would like in order to give music recommendations by using the single matrix that the dot product algorithm returns.

## Approach

The approach I took to make music recommendations for users was performing the dot product on the Types of Songs Matrix and the Genre Likes Matrix. From doing the dot product on these two matrices we can get “rankings” or in other words how likely a user would like that song. These rankings are put into one big matrix and from this matrix we can see the predictions of how likely a user is to like a song which we can then use to make recommendations. The higher the ranking score, the more likely the algorithm is to recommend that song to the user. Below is an example of the matrix that returns ratings to be used to make music suggestions to users.



Predicted Ratings:

| | Song 1 | Song 2 | Song 3 | Song 4 | Song 5 |
|---|---|---|---|---|---|
| Person A | 5 | 5 | 3 | 5 | 5 |
| Person B | 3 | 4 | 7 | 7 | 3 |
| Person C | 2 | 3 | 6 | 6 | 2 |
| Person D | 5 | 4 | 5 | 3 | 5 |

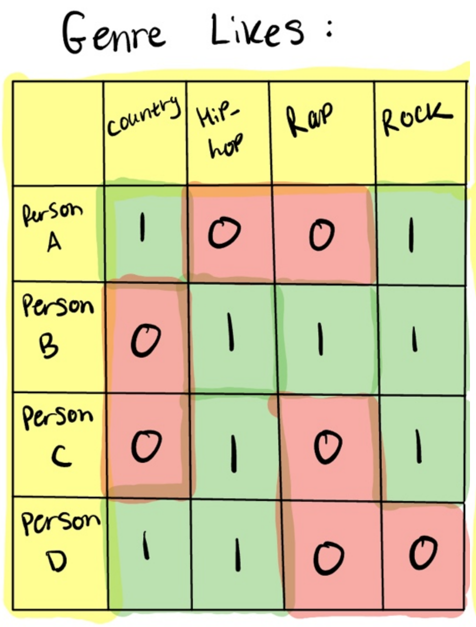
The Predicted Rating Matrix above is from taking the dot product of the Types of Songs Matrix and Genre Likes Matrix that was shown in the goal section of this paper. From this matrix, we can what songs the users are predicted to like and how strongly they are to like these songs.

We can use the rating to make music suggestions from the Predicted Ratings Matrix:

- Person A would be recommended songs one, two, four, and five with a strong rating score of five. Song three may also be suggested but after the other songs because song three did not get as high of a ranking score.
- Person B would strongly be recommended songs three and four with a high ranking score of seven. The other songs did not get high ranking scores so those songs would not be as likely to be suggested to the user to listen to.
- Person C is similar to person B where songs three and four had high rating scores, but the other songs did not. Person C would be suggested to listen to songs three and four with higher priority.
- Person D would be suggested to listen to songs one, three, and five due to those songs having the highest ranking score. Song two may also be suggested as it is right behind those songs with a ranking score of four.

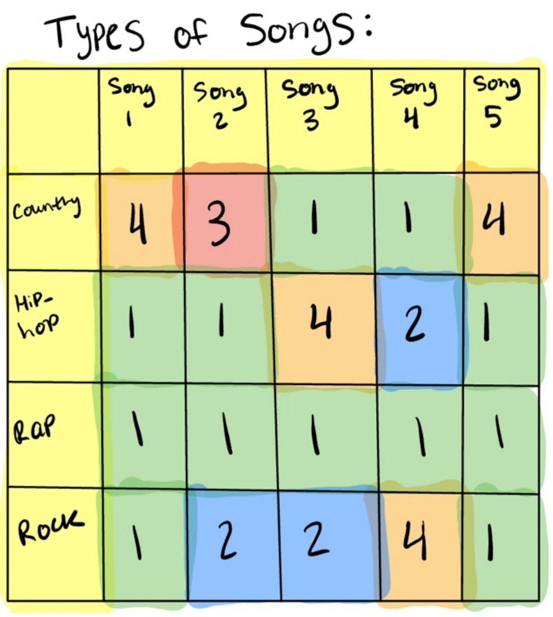
## Analysis

I am happy with the Predicted Ratings Matrix presented in the previous section, the approach. The Predicted Ratings Matrix presents data that is accurate to what genres the users like to listen to.



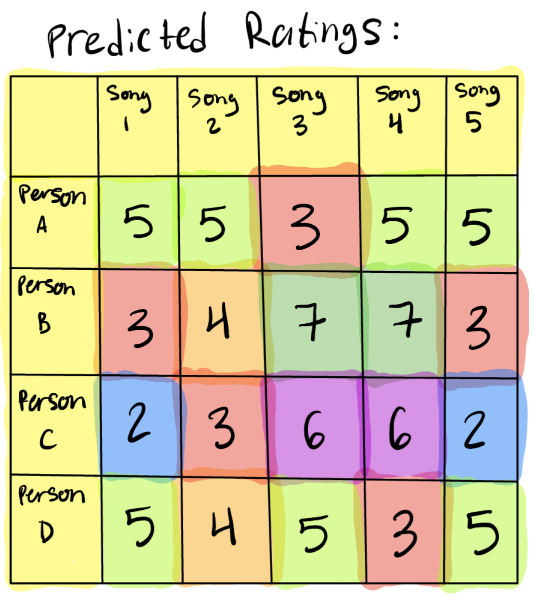
Genre Likes:

| | Country | Hip-hop | Rap | Rock |
|---|---|---|---|---|
| Person A | 1 | 0 | 0 | 1 |
| Person B | 0 | 1 | 1 | 1 |
| Person C | 0 | 1 | 0 | 1 |
| Person D | 1 | 1 | 0 | 0 |



Types of Songs:

| | Song 1 | Song 2 | Song 3 | Song 4 | Song 5 |
|---|---|---|---|---|---|
| Country | 4 | 3 | 1 | 1 | 4 |
| Hip-hop | 1 | 1 | 4 | 2 | 1 |
| Rap | 1 | 1 | 1 | 1 | 1 |
| Rock | 1 | 2 | 2 | 4 | 1 |



Predicted Ratings:

| | Song 1 | Song 2 | Song 3 | Song 4 | Song 5 |
|---|---|---|---|---|---|
| Person A | 5 | 5 | 3 | 5 | 5 |
| Person B | 3 | 4 | 7 | 7 | 3 |
| Person C | 2 | 3 | 6 | 6 | 2 |
| Person D | 5 | 4 | 5 | 3 | 5 |

From the Genre Likes Matrix we can see that Person A likes listening to country music and rock music. From the Types of Songs Matrix, we can see that the songs that are most aligned to country and rock are songs one, two, four, and five. From the Predicted Rankings Matrix, we can see that those are the songs with the highest ranking score which means those are the best songs to suggest to that user since those songs align with the genre the user likes to listen to.

We can make these music suggestions with the other users to show that taking the dot product of the Genre Likes Matrix and the Types of Songs Matrix works:

- Person A:
    - Genre Likes: Country and Rock
    - Types of Songs: Songs 1, 2, 4, 5
    - Predicted Ratings: Songs 1, 2, 4, 5
- Person B:
    - Genre Likes: Hip-hop, Rap, and Rock
    - Types of Songs: Songs 3 and 4
    - Predicted Ratings: Songs 3 and 4
        - Important to note that none of the songs in the Types of Songs Matrix aligned to the Rap genre.
- Person C:
    - Genre Likes: Hip-hop and Rock
    - Types of Songs: Songs 3 and 4
    - Predicted Ratings: Songs 3 and 4
        - Persons C and D are similar.
- Person D:
    - Genre Likes: Country and Hip-hop
    - Types of Songs: Songs 1, 3, 5

- Predicted Ratings: Songs 1, 3, 5

From this analysis of the dot product you can see that the dot product finds similarities between the Genre Likes and the Types of Songs Matrices which explains why there is a pattern between the Types of Songs that are the genres the user listens to are the ones being the highest in suggestion in the Predicted Ratings Matrix.

## Code Analysis

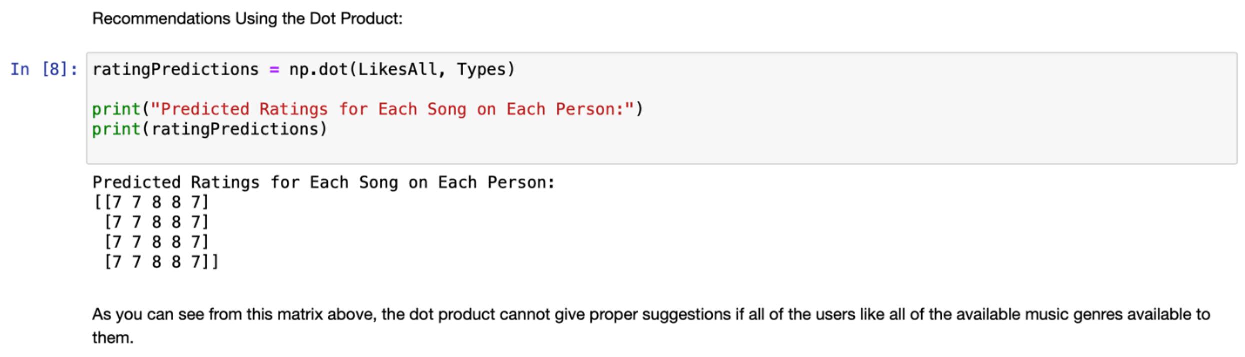
To test for when the dot product works best and for when it fails to work, I made a Jupyter Notebook file to perform the analysis. To do this, I made two cases for when the dot product should work best and two cases for when the dot product fails to work.

**Edge-Cases (dot product fails):**

Case 1: Users like all of the genres.

Since the users like all of the genres, the Genre Likes Matrix will be an all-ones matrix. When continuing with the same Types of Songs Matrix that has been used throughout this paper, we will see when we take the dot product with new the Genre Likes Matrix it will try to suggest all of the music options to all of the users. This not what we want so the dot product algorithm will not work in this case.

Analysis from the Code:



Recommendations Using the Dot Product:

```
In [8]: ratingPredictions = np.dot(LikesAll, Types)

        print("Predicted Ratings for Each Song on Each Person:")
        print(ratingPredictions)
```

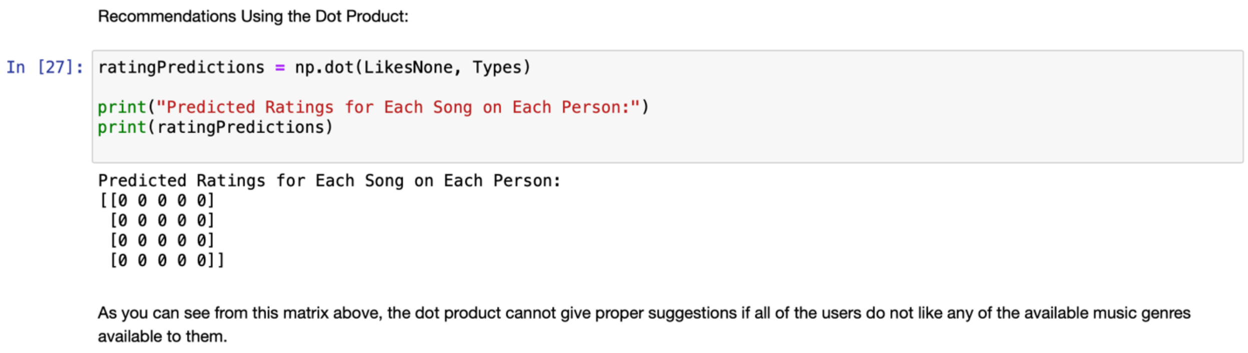
```
Predicted Ratings for Each Song on Each Person:
[[7 7 8 8 7]
 [7 7 8 8 7]
 [7 7 8 8 7]
 [7 7 8 8 7]]
```

As you can see from this matrix above, the dot product cannot give proper suggestions if all of the users like all of the available music genres available to them.

Case 2: Users like none of the genres.

Since the users like none of the genres, the Genre Likes Matrix will be an all-zero matrix. When continuing with the same Types of Songs Matrix that has been used throughout this paper, we will see when we take the dot product with new the Genre Likes Matrix it will try to suggest none of the music options to all of the users. This not what we want so the dot product algorithm will not work in this case.

Analysis from the Code:



Recommendations Using the Dot Product:

```
In [27]: ratingPredictions = np.dot(LikesNone, Types)

         print("Predicted Ratings for Each Song on Each Person:")
         print(ratingPredictions)
```

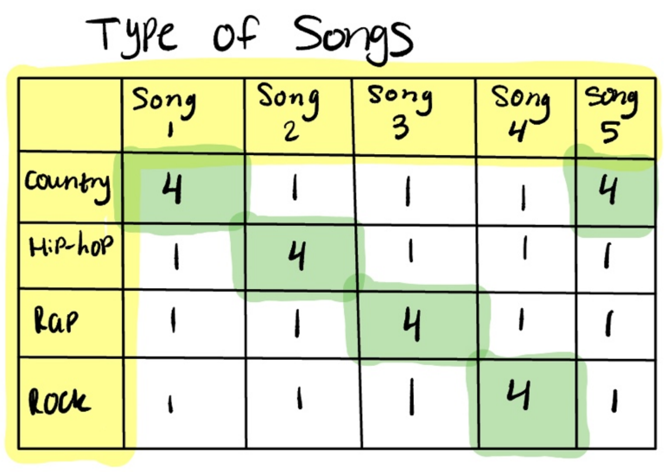
```
Predicted Ratings for Each Song on Each Person:
[[0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]]
```

As you can see from this matrix above, the dot product cannot give proper suggestions if all of the users do not like any of the available music genres available to them.

**Dot Product Works Best:**

Case 1: Each song aligns to one song.

Since each song will only be aligned to one genre, there will be only one “four” entry for each song. The rest of the entries will be ones. We still use the same Genre Likes Matrix that has been used throughout this paper. Below is what the new Types of Songs Matrix will look like.



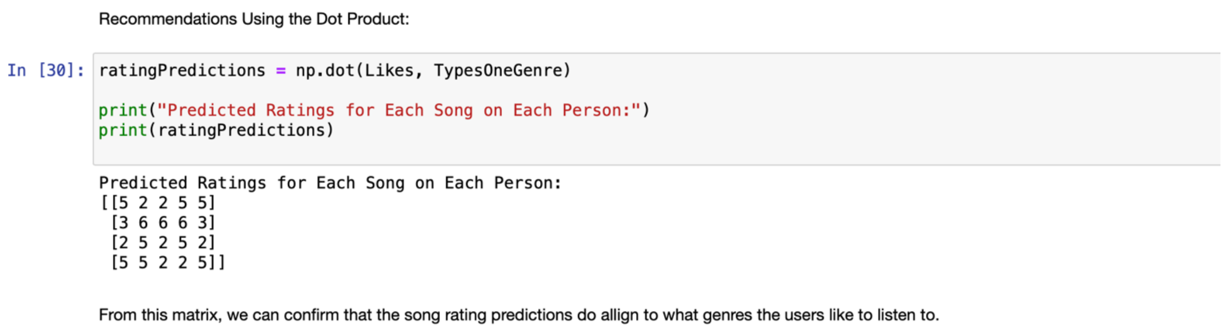
Type of Songs

| | Song 1 | Song 2 | Song 3 | Song 4 | Song 5 |
|---|---|---|---|---|---|
| Country | 4 | 1 | 1 | 1 | 4 |
| Hip-hop | 1 | 4 | 1 | 1 | 1 |
| Rap | 1 | 1 | 4 | 1 | 1 |
| Rock | 1 | 1 | 1 | 4 | 1 |

It is important to keep in mind that since we are using the same Genre Likes Marix we know that:

- Person A: Likes Country and Rock
- Person B: Likes Hip-hop, Rap, and Rock
- Person C: Likes Hip-hop and Rock
- Person D: Likes Country and Hip-hop

Analysis from the Code:



Recommendations Using the Dot Product:

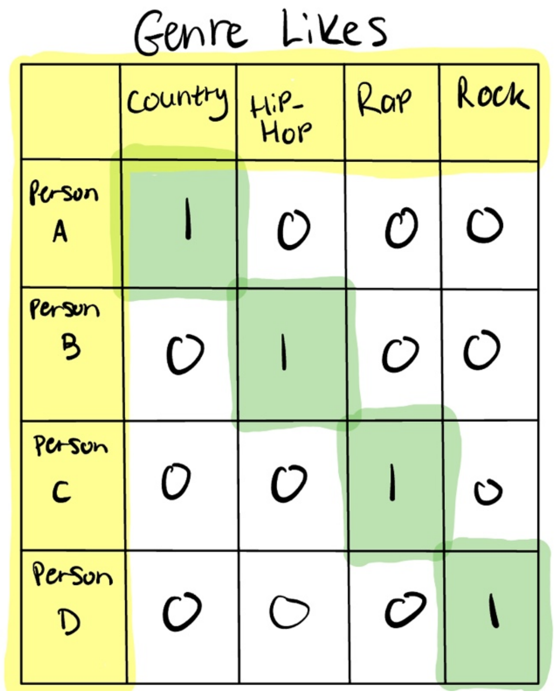
```
In [30]: ratingPredictions = np.dot(Likes, TypesOneGenre)

         print("Predicted Ratings for Each Song on Each Person:")
         print(ratingPredictions)
```

```
Predicted Ratings for Each Song on Each Person:
[[5 2 2 5 5]
 [3 6 6 6 3]
 [2 5 2 5 2]
 [5 5 2 2 5]]
```

From this matrix, we can confirm that the song rating predictions do allign to what genres the users like to listen to.

Case 2: Each person only listens to one genre.

Since in this case each person only likes and listens to one genre there will only be one "one" entry per user in the Genre Likes Matrix. The rest of the entries will be zeroes. We still use the same Types of Songs Matrix that has been used throughout this paper. Below is what the new Genre Likes Matrix will look like.

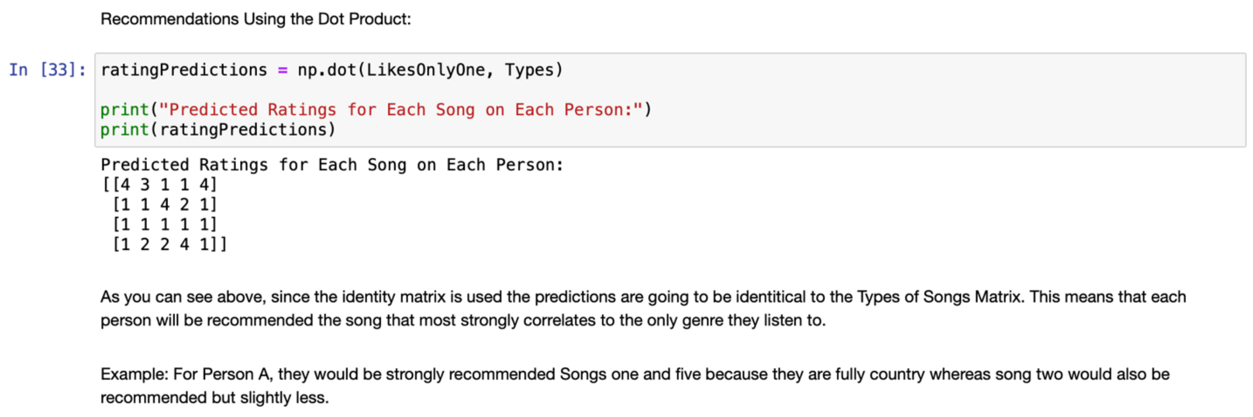


Genre Likes

| | Country | Hip-Hop | Rap | Rock |
|---|---|---|---|---|
| Person A | 1 | 0 | 0 | 0 |
| Person B | 0 | 1 | 0 | 0 |
| Person C | 0 | 0 | 1 | 0 |
| Person D | 0 | 0 | 0 | 1 |

From the new Genre Likes Matrix, you can see that it is the identity matrix. The identity matrix is what makes each user only like one genre of music.

Analysis from the Code:



Recommendations Using the Dot Product:

```
In [33]: ratingPredictions = np.dot(LikesOnlyOne, Types)

         print("Predicted Ratings for Each Song on Each Person:")
         print(ratingPredictions)
```

```
Predicted Ratings for Each Song on Each Person:
[[4 3 1 1 4]
 [1 1 4 2 1]
 [1 1 1 1 1]
 [1 2 2 4 1]]
```

As you can see above, since the identity matrix is used the predictions are going to be identitical to the Types of Songs Matrix. This means that each person will be recommended the song that most strongly correlates to the only genre they listen to.

Example: For Person A, they would be strongly recommended Songs one and five because they are fully country whereas song two would also be recommended but slightly less.

## Conclusion

In my original project proposal, I talked about how I was going to use matrix factorization instead of the dot product. Both approaches can be used to make predictions to make music recommendations. I chose to work with the dot product instead of matrix factorization for my project because I worked with small and simple datasets. In conclusion, the dot product has proven to be more efficient when working with small datasets and was more efficient at comparing users, songs, and genres in the Genre Likes Matrix and the Types of Songs Matrix.

To improve my project for the future, I want to use matrix factorization and use a bigger dataset. It is important though to keep in mind that I will still have to use a low-dimensional dataset (not many features or attributes in the data), otherwise matrix factorization could be difficult to use to find meaningful patterns in the dataset. In order to use matrix factorization, I will also need to make sure that the dataset I am working with is clean and preprocessed to ensure that the data is in a format that can be used for matrix factorization. This could include removing outliers / missing values, scaling the data, or performing other types of data transformation. I will also need to consider using regularization techniques since I will be working with a large dataset. This is important because when data becomes extremely large overfitting becomes an issue. Overfitting is when a data model becomes too complex, and it starts to memorize training data rather than learning general patterns which will take away from giving accurate music recommendations.

## References

Brownlee, Jason. “A Gentle Introduction to Matrix Factorization for Machine Learning.” *MachineLearningMastery.com*, 9 Aug. 2019, https://machinelearningmastery.com/introduction-to-matrix-decompositions-for-machine-learning/.

Donges, Niklas. “Gradient Descent in Machine Learning: A Basic Introduction.” *Built In*, 22 Aug. 2022, https://builtin.com/data-science/gradient-descent.

“How Does Netflix Recommend Movies? Matrix Factorization.” *YouTube*, YouTube, 7 Sept. 2018, https://www.youtube.com/watch?v=ZspR5PZemcs. Accessed 15 Dec. 2022.

Kwiatkowski, Robert. “Gradient Descent Algorithm-a Deep Dive.” *Medium*, Towards Data Science, 13 July 2022, https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21.

“Matrix Factorization.” *Matrix Factorization - an Overview | ScienceDirect Topics*, https://www.sciencedirect.com/topics/computer-science/matrix-factorization.

Mavuduru, Amol. “How to Build an Amazing Music Recommendation System.” *Medium*, https://towardsdatascience.com/how-to-build-an-amazing-music-recommendationsystem-4cce2719a572.

Saluja, Chhavi. “Recommendation Systems Made Simple!” *Medium*, Medium, 13 Feb. 2018, https://medium.com/@chhavi.saluja1401/recommendation-systems-made-simple-b5a79cac8862.

Valkov, Venelin. “Music Artist Recommender System Using Stochastic Gradient Descent: Machine Learning from Scratch...” *Medium*, Towards Data Science, 30 June 2019, https://towardsdatascience.com/music-artist-recommender-system-using-stochasticgradient- descent-machine-learning-from-scratch-5f2f1aae972c.

Wolfe, Cameron. “Building a Music Recommendation Engine with Probabilistic Matrix Factorization in Pytorch.” *Medium*, Towards Data Science, 21 Mar. 2019, https://towardsdatascience.com/building-a-music-recommendation-engine-withprobabilistic- matrix-factorization-in-pytorch-7d2934067d4a.