# Multi-Persona Music Recommendations[¶](#Multi-Persona-Music-Recommendations)

### Matrix Factorization solved by an Augmented Alternating Least Squares[¶](#Matrix-Factorization-solved-by-an-augme)

## Team Member:[¶](#Team-Member:)

Christopher Oyer

## Problem Statement[¶](#Problem-Statement)

I wanted to investigate matrix factorization for content prediction, focusing on diversity of an individual's taste. I was especially interested in representing a user's taste with multiple 'personas'.

Standard matrix factorization is a decomposition of a ratings matrix, **X**∈ℝm×n as a user matrix **U**∈ℝm×k and a feature matrix **V**∈ℝn×k  
such that **X**≡UV⊺

I changed the User matrix to instead to be tensor (in the looser sense of a matrix of more than 2 dimensions **U**∈ℝm×k×p

X ≡ *argmax*P UPV⊺

That is, the use's predicted rating whichever of their personas maximizes the predicted rating, when multiplied by the V matrix.

## Data Source[¶](#Data-Source)

I used a pre-existing dataset, the <http://millionsongdataset.com/tasteprofile/>.

I aggregated from song counts up to artist counts using an allied dataset provided by the same organization. Artist recommendations make the data denser, and processing to artist counts will reduced the matrix size was faster and easier to fit in memory.

The user rating was implicit from the play count. I initially tried using an offset, so that a small number of plays indicated a negative rating as compared to zero listens, but this made the data too sparse in the positive ratings.

I also removed users who only listened to a single artist.

## Methodology[¶](#Methodology)

After preprocessing the data, I rearranged it into a user x artist rating matrix, and took a log(x+eta) transform. This was beneficial to make the ratings closer to a gaussian distribution, which aligns with the

A standard train/test/validation split could be problematic because the holdout set would face a 'cold-start' problem: if an artist or user is only in the holdout, and is then introduced, there is no data on them to enable a prediction. The ideal use case for the model is for a user to get a new recommendation, based on the model's knowledge of their taste (user x latents) and existing artists' location in the latent space.

Thus, I randomly selected 15% of the user/artist pairs and masked them for the training, and then used these as the test set.

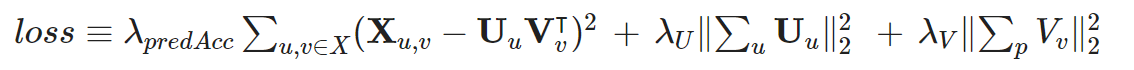
Next, the U and V matrices were initialized with random gaussian noise.

I used matrix factorization, implementing in numpy. There were several reasons for this:

* I was especially interested in understanding the algorithmic implications of different loss functions
* the multi-persona approach is not available in typical packages

To solve for U and V, I chose to use Alternating Least Squares. This is a way to solve that iteratively fixes **U** analytically find the best **V**Tto minimize the loss function, then fixes **V**T, and analytically finds the best **U.** It converges quickly, because unlike gradient descent, it jumps directly to the best of the **U** / **T** given the other two matrices.

The loss function that is minimized is



I began with creating a standard Alternating Least Squares python module. This was able to very quickly converge.

I then implemented ALS with for multiple personas. The step wherein the **V**T is optimized was very similar to the standard implementation. The only change was taking the U tensor and flattening all of the user personas for all users into a single dimension, as though they were independent users. The other step, optimizing the **B** matrix, was more difficult. I computed for each 2-d slice along the P dimension of **U** (each slice having 1 persona per user). From these, I determined the persona that was associated with the maximum predicted rating for the artist for each user, and collapsed to only the maxium latents vector for that user

I also considered a deep learning implementation of this model.

## Evaluation and Final Results[¶](#Evaluation-and-Final-Results)

This was disappointing. While trying numerous combinations of number of latents, number of personas, and different regularization weights, there was almost no difference in the converged test set prediction loss, except that less complex models were superior. Generally, the best values were found in the first three iterations, and often after the first iteration. This was surprising as it indicated that the regularization terms were of little benefit or even harmful to the test error, (even when the contribution to the total loss was almost entirely from the prediction loss, not the regularization terms).

And example of this effect:

Shape

Description automatically generated

However, the values were much better than the baseline value (from the randomized starting values), so I believe the actual code was true to the specified model.

Results for variations on the number of latent variables and the number of personas:

Shape

Description automatically generated Shape

Description automatically generated

More disappointing was that the prediction loss was worsened by the addition of the persona compared to the baseline model. Generally, the best results came from the simplest models, with only a few latents and with a single person dimension, i.e, for baseline model.

A picture containing text

Description automatically generated

Additional research on this could prove more fruitful. Future directions could include increasing the dataset size to support more model complexity, or using more complicated loss function on the predictions to support predictions which are higher in terms of squared distance from the original data but lower in some other distance metric.

## References

Dataset requested citation:

@INPROCEEDINGS{Bertin-Mahieux2011,

author = {Thierry Bertin-Mahieux and Daniel P.W. Ellis and Brian Whitman and Paul Lamere},

title = {The Million Song Dataset},

booktitle = {{Proceedings of the 12th International Conference on Music Information

Retrieval ({ISMIR} 2011)}},

year = {2011},

owner = {thierry},

timestamp = {2010.03.07}

}

Papers:

* <https://dl.acm.org/doi/pdf/10.1145/2507157.2507209>
* <https://cseweb.ucsd.edu/~jmcauley/pdfs/cikm15.pdf>

Code for project available at