# A Recommendation System for Spotify

Luigi Barba, Lorenzo Colantoni, Francesco Guglielmino, Jacopo Masini e Gianluca Surico

28/07/2020

# Introduction

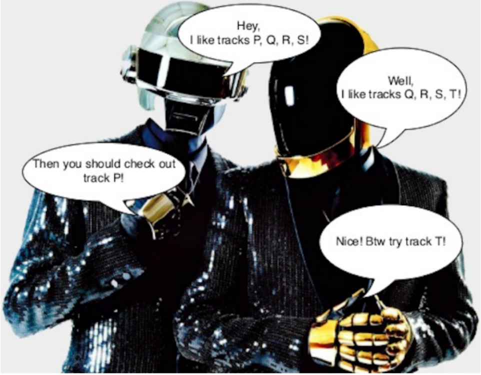
A recommendation system is a content filtering software that creates personalized recommendations specific to the user in order to help him in his choices.

# Introduction

## Approaches

In general, recommendation systems are based on two different strategies:

- Content-based approach
- Collaborative Filtering (CF)



The recommendation systems are based on different types of inputs:

- the easiest to manage is explicit feedback, which includes explicit input from users about their interest in products.

- the easiest to get is implicit feedback, which indirectly reflects the user's opinion.

# Main goal

The goal of our work is to recommend new artists to listen to users taken as input, using collaborative filtering with implicit feedback data. This consists in analyzing the relationship between user and artist, working on implicit signals as artist's track play counts.

# Data collection

We collected data for 12 Spotify accounts, in particular the list of favorite tracks for each user. We have merged the data into an $m \times n$ matrix. The m rows represent the users, the n columns are the unique artists of all users' favorite tracks.

# Model Overview (1) - The input data

We define the matrix $R = (r_{ui}) \in \mathbb{R}^{m \times n}$ as the user-item interaction matrix.
Each entry $r_{ui}$ specifies the interaction of item $i$ by user $u$ and can be thought of as a single observation. Here's an example of $R$:

| | Item 1 | Item 2 | Item 3 |
|---|---|---|---|
| User 1 | 0 | 5 | 4 |
| User 2 | 1 | 2 | 6 |
| User 3 | 0 | 0 | 7 |

**Underlying implicit feedback model hypothesis**: $r_{ui}$ is definitely a measure of appreciation, but it is rather "raw" and needs to be processed further!

# Model Overview (2) - Building a confidence metric

For this reason we are interested in formalizing a measure of confidence in how the user really appreciate the item he interact with:

$$c_{ui} = 1 + \alpha log(1 + r_{ui}) \tag{1}$$

**Remark**: the confidence function is not univocal, but can be chosen at discretion. The better it reflects the nature of the data, the better it improves prediction accuracy.
Multiple confidence functions (along with their parameters) could even be objects of cross-validation!

# Model Overview (3) - Finding latent factor matrices

In Matrix Factorization models, the goal is to find a vector $x_u \in \mathbb{R}^f$ for each user $u$, and a vector $y_i \in \mathbb{R}^f$ for each item $i$ that will factor user preferences.
In other words, preferences are assumed to be the inner products of two vectors known as user-factors and the item-factors, respectively:

$$p_{ui} = x_u^T y_i \tag{2}$$

where $p_{ui}$ is a binary variable:

$$p_{ui} = \begin{cases} 0 & \text{if } r_{ui} = 0 \\ 1 & \text{if } r_{ui} > 0 \end{cases} \tag{3}$$

# Model Overview (4) - The Optimization problem

Based on the what we have seen, we can finally define our Loss Function:

$$\min_{x_*, y_*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda\left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2\right) \tag{4}$$

The first term is the model intrinsic loss, the second one is the penalty assigned to the latent factors in order to prevent overfitting. $\lambda$, along with $\alpha$ (contained in $c_{ui}$), are two parameters to be tuned.

# Model Overview (5) - Additional optimization I

This cost function contains $m \times n$ terms, where $m$ is the number of users and $n$ is the number of items. For typical datasets (even if it is not this project's case) this product can easily reach a few billion. What to do?

# Model Overview (6) - Additional optimization II

**Solution**: by exploiting the fact that when either the user-factors or the item-factors are fixed the cost function becomes quadratic (so its global minimum can be readily computed), we could consider an alternating-least-squares optimization process, where we alternate between re-computing user-factors and item-factors, and each step is guaranteed to lower the value of the cost function.

- Step 1: Compute Y
- Step 2: Fix Y from Step 1 and compute X
- Step 3: Fix X from Step 2 and compute Y
- And so on, until it stabilizes

# Model Overview (7) - Additional optimization III

This model's version of the alternating-least-squares optimization is peculiar since it requires to take into account the confidence levels we defined before. In fact, $X$ and $Y$ are computed as follows:

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u) \tag{5}$$

$$y_i = (X^T C^i X + \lambda I)^{-1} X^T C^i p(i) \tag{6}$$

where $C^u$ is a the diagonal $n \times n$ matrix with $C^u_{ii} = c_{ui}$, and $p(u)$ is a vector $\in \mathbb{R}^n$ that contains all the preferences by $u$ (the $p_{ui}$ values)

# The implementation

Our idea is to split the dataset into training and test set and then artificially split the test set into history and future data in order to quantify the performance of the model.

In R this will be done using the createDataPartition of the library caret.

# The design matrix

The design matrix $X$ is a user-item matrix in which the $r_{ui}$ element is the number of saved songs made by artist $i$ saved by user $u$.

The feedback is implicit (no explicit rating).

In R, this will be saved in the 'sparseMatrix' format mainly because of the sparsity of $X$.

# The Tuning

The tuning of the parameters will be done using a 4-fold cross-validation scheme.

There are 3 tuning parameters: the number of latent factors, the $\alpha$ confidence parameter and the regularization parameter $\lambda$.

The fitting of the model is made possible by using the *model$fit_transform* function of the rsparse package after setting up the model using $model = WRMF\$new()$.

# The Evaluation Metric

The validation scheme is based on a precision measure called $MAP@k$: it is the average over the test set of the average precision for each user.

$$AP@k = \sum_{k=1}^{n} \frac{P(k) * rel(k)}{I(u)} \tag{7}$$

In R, this is done using the $ap\_k$ function of the rsparse package, which returns the average precision of the predictions for each user in the test set: the $MAP@k$ will be the mean of this vector

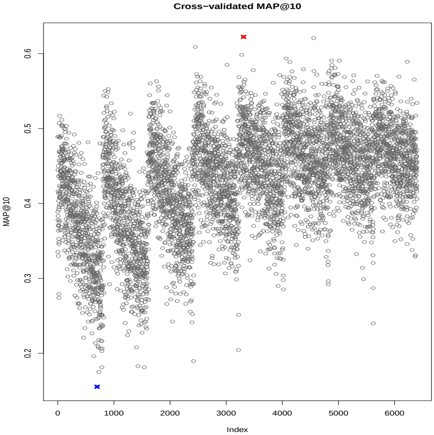
# The recommendations

After tuning the parameters, the recommended items will be displayed after using the *model$prediction* function, which - on a mathematical level - just performs another round of the ALS algorithm.

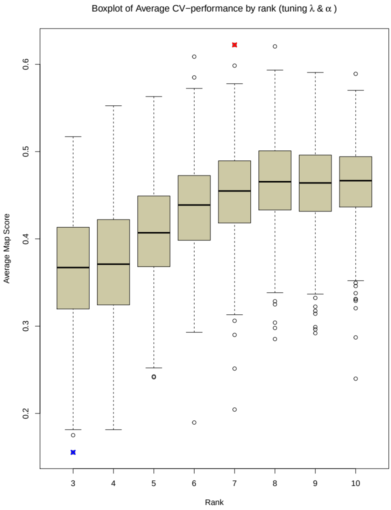
# Cross Validation and Parameters Tuning

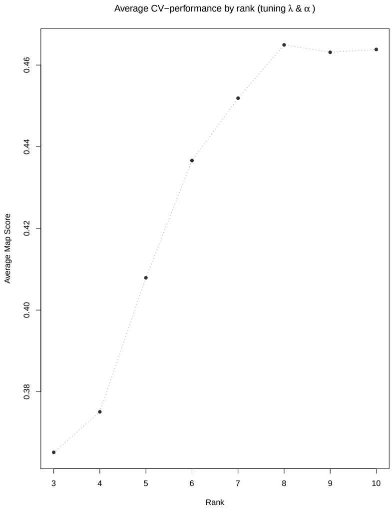
Before starting analyzing the results we got, let's remark briefly an important concept on which our work is based on:

- the Cross Validation we applied isn't a "pure" one based on the usually loss function $\longrightarrow$ it's based on the **MAP@k** metric evaluation that is independent from $\alpha$ parameter



Cross−validated MAP@10
MAP@10
0.2
0.3
0.4
0.5
0.6
0
1000
2000
3000
4000
5000
6000
Index


Boxplot of Average CV−performance by rank (tuning λ & α )
Average Map Score
0.2
0.3
0.4
0.5
0.6
3
4
5
6
7
8
9
10
Rank


Average CV−performance by rank (tuning λ & α )
Average Map Score
0.38
0.40
0.42
0.44
0.46
3
4
5
6
7
8
9
10
Rank

# Model Recommendation

| Lorenzo | | Jacopo | | Gianluca | |
|---|---|---|---|---|---|
| **Score** | **Recommendation** | **Score** | **Recommendation** | **Score** | **Recommendation** |
| 0.8517400 | Arctic Monkeys | 0.5618116 | Bruce Springsteen | 0.7689953 | Coez |
| 0.7965103 | Bruno Mars | 0.5284429 | Simon & Garfunkel | 0.7043944 | Simon & Garfunkel |
| 0.7849014 | Black Eyed Peas | 0.4720040 | Coez | 0.6490270 | J-AX |
| 0.7609935 | Billie Eilish | 0.4713720 | J-AX | 0.6147510 | Bob Dylan |
| 0.7443408 | Sia | 0.4706330 | The Clash | 0.5899966 | Linkin Park |
| 0.7443408 | Michael Jackson | 0.4620708 | U2 | 0.5757663 | U2 |
| 0.7337686 | Mannarino | 0.4590332 | The Beatles | 0.5708236 | Jefferson Airplane |
| 0.7205724 | Eminem | 0.4552083 | Linkin Park | 0.5637806 | The Doors |
| 0.7071880 | Rino Gaetano | 0.4484908 | Ludovico Einaudi | 0.5576449 | The Smashing Pumpkins |
| 0.6844891 | Nitro | 0.4376672 | alt-J | 0.5376591 | Jarabe De Palo |

# Similarity between artists

- recommended artists
- similar artists

Look for similarities between the items of our artists' basin.

After implementing our recommendation system we tried to exploit the information coming from the "latent" item matrix in order to suggest to the users artists similar to the one already recommended.

This was done using the rsparse *model$components*, computing cosine similarities between the columns.

# Similarity between artists

Cosine similarity:

$$sim_{ab} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}} \tag{8}$$

where $a$, $b$ represent the vectors in a latent factor space for two artists.

# Similarity between artists

| Latent factor | | | |
|---|---|---|---|
| The Doors | David Bowie | Jefferson Airplane | Bruce Springsteen |
| 0.20757844 | 0.20052335 | 0.16216339 | 0.167778510 |
| 0.18957669 | 0.17499336 | 0.18517567 | 0.170741923 |
| 0.20070742 | 0.27311633 | 0.10546285 | 0.100863064 |
| -0.17000637 | -0.09837597 | -0.11116437 | -0.088528736 |
| -0.13047577 | 0.14050596 | -0.06114843 | -0.059720767 |
| 0.05400830 | -0.13187250 | -0.13761145 | -0.119644873 |
| -0.01616389 | -0.07034186 | -0.03062981 | -0.003315523 |

# Similarity between artists

| Similar artists | | |
|---|---|---|
| The Doors | The xx | Iron Maiden |
| David Bowie | Beck | Lou Reed |
| Jefferson Airplane | Talking Heads | alt-J |
| Bruce Springsteen | Little Simz | The Rolling Stones |
| Bracket | Fleet Foxes | Phil Collins |
| Lucio Battisti | Massive Attack | Caravan |

# Similarity between artists