Sparkling Love - A recommendation engine for dating

Felipe Chamma, Felipe Ferreira, Ryan Speed

March 7th, 2016

Abstract

Given the rise of dating related apps - such as Tinder, OkCupid, Hitch - the algorithms for recommending new users have got a lot more complex. In a market with such a low entry barrier, suggesting users that are more likely to yield matches seems like an important competitive edge.

In that context, this project analyzed data from a dating website to try to come up with the best possible recommendation system. To accomplish that we have used cutting edge machine learning applications, such as collaborative filtering - using matrix decomposition for better performance.

It's important to have in mind that differently from a regular recommendation system for items (such as books, music or movies), dating recommendation algorithms should take into account the mutual affinity likelihood between users being suggested to each other - which makes the problem a bit more complex.

Also, we discovered that the gender affects a lot the pattern for rating other people - it was easier to predict men's ratings for women than the other way around. Therefore, we had to use different models for the matrix decomposition based on the gender we were trying to predict for.

Overview

Data set description

We used data from LibimSeTi, a well known czech dating website. There are 2 datasets: one with users' gender info (user id, gender) and the other is a rating matrix from pairs of users (user 1, user 2, rating).

These files contain 17,359,346 anonymous ratings of 168,791 profiles made by 135,359 users as dumped on April 4, 2006. The data is available from this link.

Here is a quick summary for these data sets:

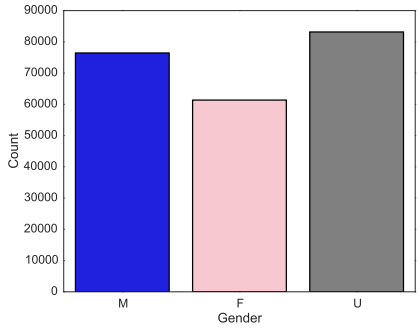


Figure 1: User count by gender

As we can see from the plot above, users are not required to inform their gender. And since we don't want to risk recommending an undesired gender to the users, we will simply remove those unknowns from our data. Below we can see the distribution of user ratings after removing unknowns:

Rating Distribution by Gender

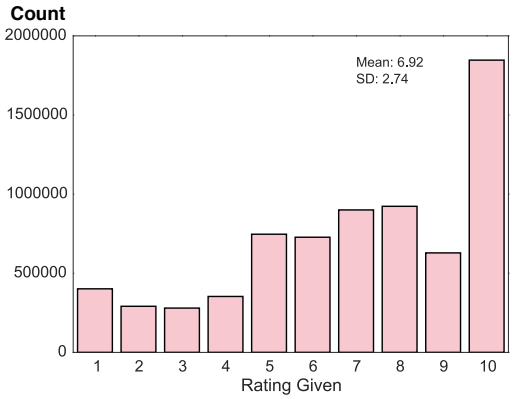


Figure 2: Female to Male

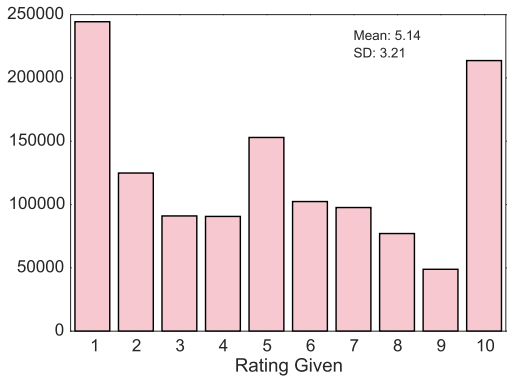


Figure 3: Female to Female

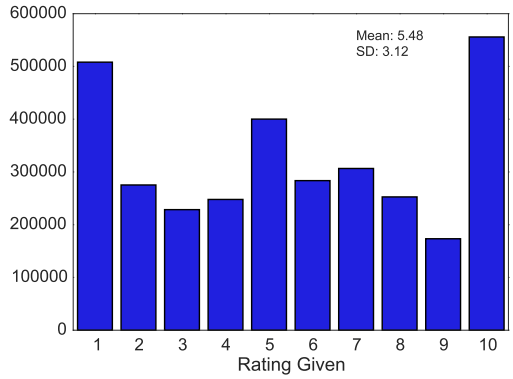


Figure 4: Male to Female

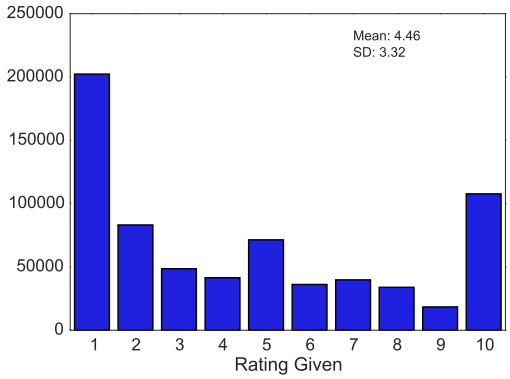


Figure 5: Male to Male

We notice that the distribution is slightly different depending on the gender of the user doing the rating, and the user being rated. We will have to account for this when interpreting the predicted user-to-user ratings.

Methodology

To accomplish our task, the first step is to predict how a user will rate another user, based on similar users' ratings. To achieve this, we will use gradient descent to help factorize a $users \times users$ utility matrix, which is computationally faster than a pivoting algorithm with $O(n^2)$ complexity.

In the second and last step, we will apply our recommendation algorithm that will suggest new users based on the predicted rating, in which we will also take into account the mutual predicted ratings and each user's average rating.

All data will be loaded, cleaned and analyzed using Spark clusters. All code will be written in Python.

Feature engineering

A classical recommendation system looks to recommend items to a user, with the only concern that this particular user will enjoy it. In the domain of dating however we are looking to recommend users to other users in hopes of sparking mutual interest. In this section we design a feature that will incorporate the predicted ratings of users towards each other.

We believe that users whom show a preference towards each other relative to the general population, and are also on the same attractiveness tier (as perceived by rating score), will have higher probability of being compatible. In practice we could create several metrics to represent users' match potential, then perform A/B testing to see what metric sparks the most interest.

We will define a standard of measurement *love_score* in an attempt to recommend the best possible match. The *love_score* between user i and j is as follows:

$$love\_score\,(U_i, U_j) = \lambda_1\big(10 - |Att(U_i) - Att(U_j)|\big) + \lambda_2 Pref(U_i, U_j) + \lambda_3 Pref(U_j, U_i)$$

Where the first term captures the similarity amongst two user's attractiveness tier. The second term represents how we predict user i will prefer user j. The third term represents how we predict user j will prefer user i. The respective weight to each feature can be controlled with the λ parameters.

We define:

$\overline{U_{iR}}$: Average rating user i receives
$\overline{U_{iG}}$: Average rating user i gives
$\overline{MF}$: Average rating of all females, rated by males

$$Att(U_i) = \begin{cases} \overline{U_{iR}} - \overline{MF} & if\ U_i\ is\ male\ looking\ for\ female \\ \overline{U_{iR}} - \overline{FM} & if\ U_i\ is\ female\ looking\ for\ male \\ \overline{U_{iR}} - \overline{MM} & if\ U_i\ is\ male\ looking\ for\ female \\ \overline{U_{iR}} - \overline{FF} & if\ U_i\ is\ female\ looking\ for\ female \end{cases}$$

*measure of relative attractiveness as inferred by rating.

$$Pref(U_i, U_j) = \hat{f}(U_i, U_j) - U_{iG}$$

*measure of how much user i prefers user j, relative to the population.

Where:

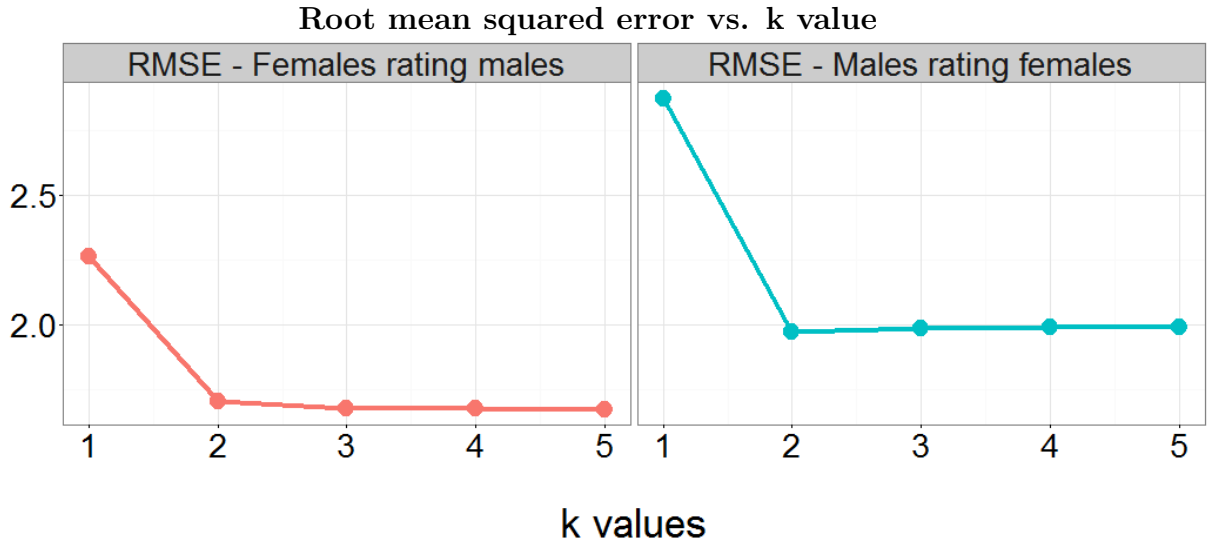
$\hat{f}(U_i, U_j)$: predicted rating of user j by user i

Experimental results

Predicting ratings

The evaluation of the rating prediction was assessed by changing the parameter k in the matrix decomposition step. This parameter reflects the number of dimensions of the decomposed matrices - higher k values mean more complex models. However, k values that are too high may result in over fitting and might increase the error in the test set.

It turned out the ideal k value for decomposing the matrices was different for males and females, as we can see in the chart below. To yield the best possible RMSE, we used $k = 2$ for males rating females and $k = 5$ for females rating males. We decided not to pursue a model for same-sex ratings, as the training set for that was too small.



Recommending Users

Using our model for predicting ratings, we were able to get a complete matrix with all the ratings for every possible male-female and female-male interactions. From that, we could calculate the love_score metric for each pair of users.

We can then get the top 10 users using the love_score metric to make our recommendations. The information for the top 10 recommended users can be retrieved via our Flask app (which we designed using python, postgreSQL and 2 AWS Instances). Just access this link using a valid user id after the last forward slash.

Unfortunately, we have no formal metric for assessing the effectiveness of the recommendation algorithm, as we can not estimate how users would have enjoyed our suggestions.

Key lessons

These are the key takeaways from this project, considering a learning perspective:

- Spark can be a powerful tool, if used correctly. Failing to set the spark context configurations and the level of parallelism will make it slower than pandas.
- Collaborative filtering and recommendation demand maximum algorithm efficiency. Using a similarity matrix to predict the ratings could possibly result in a lower error, but the order complexity would be more than $O(n^2)$. This was not feasible considering our data set and our hardware, so we settled for matrix decomposition.