Netflix Prize
Challenge Report

Mattia Bonfanti (5002273)
Jonathan Garack (4889045)
Kaggle ID: Group4

Introduction

The Netflix Prize was a competition for the best algorithm to predict user ratings for films using only previous user ratings. The competition was held between 2006 and 2009 with the final prize of $1,000,000 awarded to the algorithm with the best improvement over Netflix's own "Cinematch" algorithm. The winning algorithm presented a Test RMSE of 0.8567 with an improvement of over 10% compared to "Cinematch". [1]

A smaller version of this challenge has been proposed by the course of "CSE2525 Data Mining" at TU Delft. The goal is to explore different solutions to provide a recommendation system that predicts a user's rating using the stars method (1 to 5). The following datasets are provided:

- Users: information about 6,040 users indicating their gender, age (if known) and profession;
- Movies: information about 3,706 movies indicating their year of release and title;
- Ratings: information about 910,190 ratings that users gave to movies;
- Predictions: the 90,019 predictions that our algorithm had to compute.

Our group started with the implementation of two basic algorithms, collaborative filtering (user-user based) and latent factors. Starting with the computation of the identity matrix and basic similarity matrix, we then proceeded to expand and refine our algorithms. We used techniques that we studied in class as well as ones that we independently researched with the goal to achieve a better score with each solution.

Below we have included more details about the methodologies we used and the results we achieved as well as reflection about further improvements.

Methodology

1. Collaborative filtering

The implementation of Collaborative filtering started with a basic **user-user based similarity**. For a user X, we have found a set N of other users whose ratings are similar to X's ratings. We then used the ratings of users in N to estimate the ratings of user X.

The similarities between users have been calculated using the **Pearson's correlation coefficient** as distance metrics. This way we were able to base the similarity scores on the actual ratings that users gave to the same movies.

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

The computed similarity matrix is a 6,040x6,040 matrix representing the similarities between users. The closer the similarity score is to 1, the more similar the users are. **For each user x, we used a set of his/her nearest neighbors (tested with 1, 10 and 100) to estimate the ratings he/she would give to a movie i**. The formula we used is the following:

$$r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$$

The next step was to implement an **item-item based similarity**. As before, we used **Pearson's correlation coefficient** as distance metrics (formula is the same as listed above with x and y as movies instead of users). The computed similarity matrix is a 3,706x3,706 matrix representing the similarities between movies. **For each movie i, we used a set of its nearest neighbors (tested with 1, 10 and 100) to estimate the ratings it would receive from a user x**. The formula we used is the following:

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

This solution was then improved by including global and local effects. **For each movie i, we used a set of its nearest neighbors (with each rating balanced by their local baseline) to estimate the ratings it would receive from a user x. We then balanced the estimate with its global baseline.** The formula we used is the following:

$$\hat{r}_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

The baselines are calculated as follows:

$b_{xi} = \mu + b_x + b_i$, where μ is the overall mean rating, bx is the rating deviation of user x and bi is the rating deviation of movie i;

$b_{xj} = \mu + b_x + b_j$, where μ is the overall mean rating, bx is the rating deviation of user x and bi is the rating deviation of movie j;

We then reused this formula to implement a version of the algorithm with expanded baselines. Alongside the deviation of the user and the movie, we also included the deviations related to the movie year, the user age, gender and profession.

The last addition we made to our item-item based collaborative filtering algorithm was based on weights interpolation. The weights matrix is a 3,706x3,706 matrix representing the relationship between movies. We can use this parameter instead of the similarities. The computation of the weights is performed through the gradient descent, in which we minimize the SSE on training data made of the best predictions so far.

$$SSE = \sum_{x,i} \left(\left[b_{xi} + \sum_{j \in N(i;x)} w_{ij} (r_{xj} - b_{xj}) \right] - r_{xi} \right)^2$$

$$\nabla_w SSE = \left[\frac{\partial SSE(w)}{\partial w_{ij}} \right] = 2 \sum_{x,i} \left(\left[b_{xi} + \sum_{k \in N(i;x)} w_{ik} (r_{xk} - b_{xk}) \right] - r_{xi} \right) (r_{xj} - b_{xj})$$

The ratings estimation formula becomes the following:

$$\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij} (r_{xj} - b_{xj})$$

2. Latent factors

The first approach for the latent factor decomposition was to implement a **simple UV decomposition**. The UV decomposition algorithm factorizes the utility matrix into two smaller matrices U and V such that, UV = R, where R is the utility matrix (*m* movies by *u* users), U is a matrix (*m* by *k*) and V is a matrix (*k* by *u*).

$$U \times V = R' \simeq R$$

To create the matrices we started with arbitrarily chosen U and V and trained them by minimizing the **error function** given by:

$$\Sigma_{r \neq 0}(r_{xi} - u_i v_x)^2$$

To train the model we used a special gradient descent algorithm called **stochastic gradient descent (SGD)**, which instead of multiplying both matrices and finding the error on each iteration, considers one rating at a time. It is therefore more time efficient.

This already gave us well trained decomposition, but there was still space for improvement. The next step was to **regularize the SGD** by adding two factors to the error function (the regularization parameters prevent overfitting):

$$\Sigma_{r \neq 0}(r_{xi} - u_i v_x)^2 + \left[\gamma \Sigma \left\| u_i^2 \right\| + \gamma \Sigma \left\| v_x^2 \right\|\right]$$

A significant improvement of the predictions of the LF model occured when we added the **global effect and user biases*** to the model and thus obtained our final model. We introduced the global effect by changing the overall formula to:

$$\mu_{global} + \mu_{user\ ratings} + \mu_{movie\ ratings} + U \times V = R' \simeq R$$

* The baselines were computed as stated in the collaborative filtering section

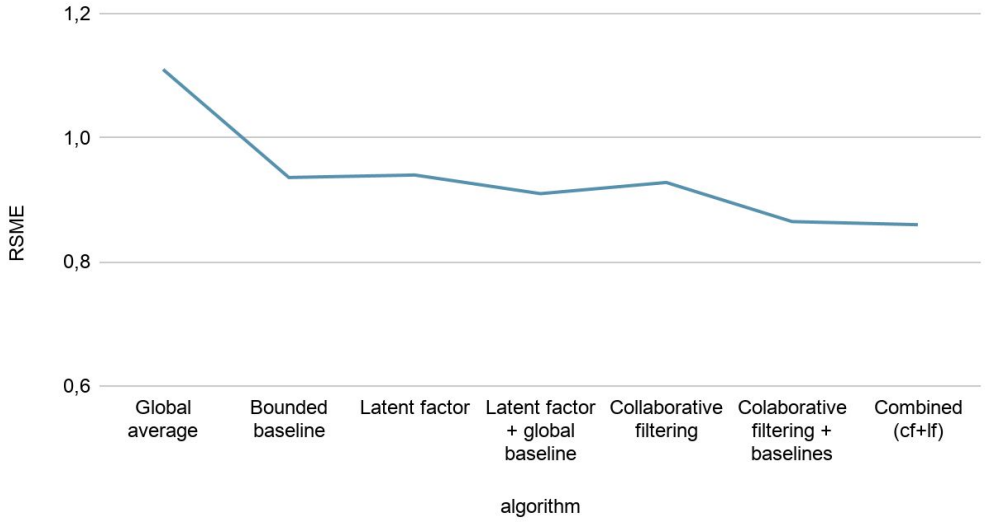
The error that we were trying to minimize changed to:

$$\Sigma_{r \neq 0}(r_{xi} - (u_i v_x \mu_{global} + \mu_{user\ ratings} + \mu_{movie\ ratings}))^2 +$$
$$+ \left[\gamma \Sigma \left\| u_i^2 \right\| + \gamma \Sigma \left\| v_x^2 \right\| + \gamma \Sigma \left\| \mu_{global}{}^2 \right\| + \gamma \Sigma \left\| \mu_{user\ ratings}{}^2 \right\| + \gamma \Sigma \left\| \mu_{movie\ ratings}{}^2 \right\|\right]$$

Results

Figure 1 shows the best results we obtained when we tested our models against the kaggle test data. Just using the **global average** gave us a **RMSE of 1.11** which was **improved to 0,936 by adding the average difference between mean for each user and movie**. The **basic latent factor** gave us a score of **0.940**, but when the **baseline** was added it improved to **0.912**, which was an improvement of 0.024 compared to just the bias model. The most efficient algorithm that we implemented was the **collaborative filtering** which scored **0.928**. After tweaking the algorithm and **adding baselines** the **RMSE** was **0.865**. The final **combination** of the latent factor model and the Collaborative filtering ($0.2*LF_{predictions} + 0.8*CF_{predictions}$) obtained a **RMSE** of **0.862**.

Figure 1: Root mean square error per algorithm implementation



Comparison to known results

Table 1.0 shows our best RSME compared to the known results, keeping in mind that we used a smaller and adjusted dataset compared to the dataset used during the original Netflix Prize.

Algorithm	Our Implementation Results	Known Results
Latent Factor + biases	0.91	0.89
Collaborative filtering + biases	0.865	0.91

table 1.0

Conclusion & Possible Improvements

The implementation of our solutions led us to an overall result that is much better than Netflix's own "Cinematch" algorithm used in 2006. [2] It was interesting to see how we were able to start from basic solutions and then keep adding improvements to achieve better scores.

Alongside our implementation we have also thought about possible future improvements that can lead to even better results.

The item-item collaborative filtering can become more accurate by using additional details about the movies, which can be pulled from IMDB. The user-user based method can be also improved by adding demographic related data for the users. This will allow us to model the global and local baselines even better.

The implementation of the weights based recommendation wasn't as successful as we hoped. The amount of computation needed was very high and we decided to simplify it for efficiency reasons. As a future development for this, we can focus on adjusting the gradient descent and possibly rely on machine learning algorithms to keep improving our predictions based on the previous ones.

Working on this project allowed us to strengthen and deepen our knowledge about Data Mining techniques, which will be very useful for our academic and working careers.

References:

[1] Wikipedia (2020), Netflix Prize, https://en.wikipedia.org/wiki/Netflix_Prize
[2] Wikipedia (2020), Netflix Prize, https://en.wikipedia.org/wiki/Netflix_Prize#Prizes

Note: All formulas are from the CSE2525 Data Mining course slides.