PCML Project 2 - Recommender System for Netflix Prize Dataset

Manana Lordkipanidze, Aidasadat Mousavifar, Frederike Dümbgen
EPFL, Switzerland

I. Introduction

In the age of online platforms for entertainment, shopping and knowledge transfer, the automatic characterization of the available items and the involved users has become a crucial factor of success. It is used to increase the ease of use (bypass exhaustive searching of the database by suggesting appropriate pages), to maximize the effect of the platform (by placing appropriate recommendations, a user might buy/watch/read more than he originally intended) and to increase profit by intelligent ad placement and user screening for personalized marketing. The tools created for such purposes are commonly called "recommender systems". In some cases, the recommendations can be done using additionally provided information on users and items, such as demographic measures like age and gender or official popularity measures and item classifications. This content-based approach tends to be too simplistic and cumbersome for large databases and adds a dependeny on third-party data of unknown accuracy. Methods grouped under the term "Collaborative filtering" overcome these difficulties by predicting behaviour based on past feedback. Feedback can either be of implicit form (how many times did a user click on an item, how long did he watch a TV series, etc.) or of explicit form, such as rankings or feedback form answers. [1] When users and items are grouped by their similarity and recommendations are done based on similar users (user-oriented models) or similar items (item-oriented models) respectively, one talks about neighborhood models. If, on the other hand, the items and users are characterized by a set of K features whose signification is a priori unknown, one talks bout latent factor models The goal of this project is to apply collaborative filtering to the Netflix dataset, which consists of explicit feedback in terms of ratings of $D = 1000$ users and $N = 10000$ items. The main focus is on latent factor models, but neighborhood models can be taken into account to improve predictions.

II. Models and Methods

A. Preprocessing

1) Data analysis: The data provided is of the form

$$\mathbf{X} = (x_{nd}), \tag{1}$$

where x_{nd} corresponds to the rating of user d for movie n. the ratings are discrete and go from 1 to 5. Since every user only rates a very small subset of movies, the matrix is sparse. The data has the following characteristics.

- The trainig set provided has 991561 non-zero entries, which corresponds to a fraction of $f_{train} \approx 0.1$ with respect to the number of unknowns($N \times$D).
- Each user in the dataset has rated at least 8 movies, the maximum ratings per user is 4590. On average, a user rates roughly 1177 movies.
- Each movie in the dataset is rated by at least 3 users, and the maximum of ratings per movie is 522. On average, each move is rated by around 118 users.

2) Bias correction: Since users and movies can vary a lot in terms of their average ratings, i.e. some users might give generally give more positive ratings than others and some movies might be more negative than others, there might be some implicit bias in the data provided. This biais was removed as follows by subtracting a correcting term from each element,

$$\widetilde{x}_{nd} = x_{nd} - \mu_{nd} \tag{2}$$

$$\mu_{nd} = \begin{cases} \mu_n = \frac{1}{|R(n)|} \sum_{d \in R(n)} x_{nd}, & \text{for item bias only} \\ \mu_d = \frac{1}{|R(d)|} \sum_{n \in R(d)} x_{nd}, & \text{for user bias only} \\ \mu = \frac{1}{|R(n,d)|} \sum_{n,d \in R(n,d)} x_{nd}, & \text{for global bias only} \\ \mu_n + \mu_d - \mu & \text{for combined biais} \end{cases}, \tag{3}$$

where the last term corresponds to what is used in [2]. The entries of the residual matrix, $\widetilde{x}_{nd}$ are then factorized as explained in II-B and the bias is added again for the final predictions. 1

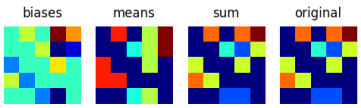

Figure 1. A simple example for bias correction. The bias matrix is obtained by substracting the mean of the corresponding user. The matrix composed of these means and the biases sum up to the original training matrix as expected.

As expected, the combined bias with underlying assumption that each rating can be composed of a contribution by the user and one by the item, is the most accurate model 2

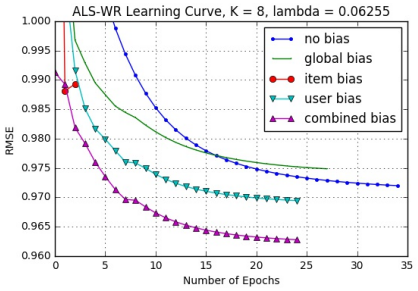

Figure 2. Performance of ALS algorithm using different variants of bias correction. TODO: what are the parameters used for this?

B. Machine learning methods

The goal is to factorize the given ratings matrix using two low-rank matrices,

$$\mathbf{X} = \mathbf{W}\mathbf{Z}^T, \qquad \text{with} \quad \mathbf{W} \in \mathbb{R}^{N\times K}, \mathbf{Z} \in \mathbb{R}^{D\times K}, \tag{4}$$

where K is the number of latent features, and $\mathbf{Z}$ and $\mathbf{W}$ are in the following referred to as the user and feature matrix respectively.

Find more about these methods in [3]

1) Bias Stochastic Gradient Descent: The stochastic gradient descent (SGD) algorithm used in this report was provided by the *Surprise* library [1] by Nicolas Hug. It is based on the algorithm provided by Simon Funk [2], which uses bias correction as described in [2]. Without this bias correction, it reduces to the algorithm given in [4].

As shown in II-A2, we expect better predictions when removing the user and item biases, so we choose to use this correction and we choose the other hyperparameters using cross-validation.

2) Alternating Least Squares:

C. Blending

It has been shown that if one disposes of several well performing methods, a clever combination of the methods can improve the accuracy of predictions. A combination can overcome problems when the used algorithms are prone to converge to local optima, or full convergence is too computationally expensive. [**?**] A combination of multiple algorithms can be particularly beneficial when some are shown to perform better in certain regions of the features space than others, giving each method a higher weight in the regions where it performs best.

The splitting of the kaggle dataset is done as suggested in [**?**]: each method is trained on a fixed subset of 95% of the dataset (training set), while the remaining 5% are split up equally, and one half is used to compare the different methods (probe set) and the other half is used to create a valuable prediction of the performance of the final, blended method (test set). This splitting ensures at least 30'000 non-zero entries in the smallest probe and test sets, which was considered enough for the evaluation, and removes only a small proportion of the valuable training data.

The process of linear blending is shown in Figure 3. The blending model is obtained by minimizing the mean squared prediction error on the probe set as suggested in [**?**]. If we call $\mathbf{p}_i$ the vector of N_P ordered predictions of method i ($i = 1 \dots N_M$) on the probe set, and $\mathbf{r}$ the true values (in the same order), then the weight vector $\mathbf{x}$ is obtained by solving

$$\mathbf{P} = [\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_{N_M}] \in \mathbb{R}^{N_P \times N_M}, \tag{5}$$

$$\mathbf{x} = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{r}, \qquad \text{with} \quad \mathbf{x}, \mathbf{r} \in \mathbb{R}^{N_M}. \tag{6}$$

The best prediction is obtained by applying these weights to the predictions of the methods on the kaggle set of size N_T, $\mathbf{q}_i$, yielding the optimum ratings vector,

$$\mathbf{Q} = [\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_{N_M}] \in \mathbb{R}^{N_T \times N_M}, \tag{7}$$

$$\hat{\mathbf{q}}_i = \mathbf{Q}\mathbf{x} \in \mathbb{R}^{N_T}. \tag{8}$$

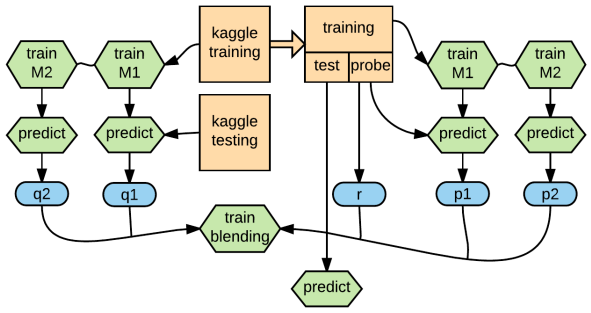

Figure 3. Visualization of blending process used to combine the methods (**orange**: datasets, **green**: operations, **blue**: results). The models are trained on a subset of the kaggle training set and predictions tested on the probe set to evaluate their performance (p_i vs. r). The predictions (q_i) for the kaggle test set are optimally weighted using linear regression, and the obtained blending method is tested on the test to predict its performance.

D. Evaluation

1) TODO: write about why kaggle results tend to be different than lcoal data for SGD, but not for ALS.
2) **Cross valdiation** A 10-fold cross validation is implemented for the whole training dataset to fix the hyperparameters.

[1] https://github.com/NicolasHug/Surprise
[2] http://sifter.org/ simon/journal/20061211.html

3) **Performance prediction** The training data was split into two sets of ratio 1:2, emulating the actual data ratio between Kaggle's training and test set. Doing the matrix factorization with this training/test set pair enables us to predict if we can expect a better performance on the kaggle dataset. TODO: what is this ratio for this dataset?

REFERENCES

[1] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative Filtering for Implicit Feedback Datasets Yifan," *IEEE International Conference on Data Mining*, pp. 263–272, 2008.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 42–49, 2009.

[3] C. R. Aberger, Y. Koren, R. Bell, and C. Volinsky, "Recommender : An Analysis of Collaborative Filtering Techniques," *Computer*, vol. 42, no. 8, pp. 42–49, 2009.

[4] R. Salakhutdinov and A. Mnih, "Probabilistic Matrix Factorization," in *Neural Information Processing Systems 21*, vol. 25, no. 2005, 2008, pp. 880–887.

[5] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, *Large-Scale Parallel Collaborative Filtering for the Netflix Prize*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–348. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-68880-8{_}32