Computational Intelligence Laboratory Project: Collaborative Filtering

Dina Zverinski*, Jan Wilken Dörrie† and Álvaro Marco Añó‡
Group: TheNonstandardDeviations
Department of Computer Science, ETH Zurich, Switzerland
*Email: *zdina@student.ethz.ch, †dojan@student.ethz.ch, ‡malvaro@student.ethz.ch*

Abstract—

I. Introduction

II. Models and Methods

In this section we describe what pre-existing algorithms we based our work on and how we used blending in the end to combine the predictions of several independent approaches. Furthermore we explain how we optimized our grade by finding a good quality / CPU time trade-off.

A. Preliminaries

Throughout this section we make frequent use of the following notation. We denote by $\mathcal{K}$ the set of all existing ratings, i.e. $\mathcal{K} = \{(u,i) \mid \text{user } u \text{ rated item } i\}$. Additionally we define M and N to be the number of all users and items, respectively. In addition we define the set of ratings for user's and item's, $R(u) = \{i \mid (u,i) \in \mathcal{K}\}$ and $R(i) = \{u \mid (u,i) \in \mathcal{K}\}$. Furthermore we define the true rating of an item i by user u as r_{ui} and denote this rating's approximation by $\hat{r}_{ui}$. Moreover we make frequent use of Greek letters, where $\mu = |\mathcal{K}|^{-1} \sum_{(u,i)\in\mathcal{K}} r_{ui}$ is the global mean of all issued ratings, λ denotes regularization constants and γ is the learning rate in an Stochastic Gradient Descent algorithm.

B. Data Imputation and Baseline Estimators

Given that standard Collaborative Filtering algorithms such as K-Means and Singular Value Decomposition (SVD) require dense instead of sparse matrices different strategies exist to fill in the missing values. The most simple one is to replace all missing values with a constant zero value. Accuracy can be improved by considering other constants such as the global mean μ of all existing entries. More advanced techniques try to approximate a given missing value for a user u and item i by taking a combination the global mean and the specific user and item means b_u and b_i [1]–[4]:

$$b_{ui} = \mu + b_u + b_i. \tag{1}$$

b_u and b_i can be directly computed from the data, however it is advisable to introduce regularize terms to overcome over-fitting when only very few ratings are available. This leads to the following equations, where $R(u)$ and $R(i)$ denote the sets of all ratings by user u and item i respectively [1]–[4]:

$$b_i = \frac{\sum_{u\in R(i)}(r_{ui} - \mu)}{\lambda_i + |R(i)|} \qquad b_u = \frac{\sum_{i\in R(u)}(r_{ui} - \mu - b_i)}{\lambda_u + |R(u)|} \tag{2}$$

λ_i and λ_u are regularize parameters that should be found via cross-validation. Finally it is also possible to estimate b_u and b_i by solving the regularized least squares problem [1]–[4]

$$\min_{b_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda\left(\sum_u b_u^2 + \sum_i b_i^2\right). \tag{3}$$

Possible solution approaches include Alternating Least Squares (ALS) or Stochastic Gradient Descent (SGD). The baseline estimators are able to capture a lot of signal present in the data so that it is advisable to make this preprocessing step part of every more involved algorithm. For example, the RMSE score for a simple SVD solution improved drastically when missing data was imputed with optimized baseline estimators (see Section III).

C. Factorized Models

Motivated by the progress a simple SVD approach could make we investigated more sophisticated approaches.

1) Regularized SVD: Shortly after the Netflix Challenge started in 2006 Simon Funk proposed the idea of a "regularized SVD" [5]. In contrast to an ordinary Singular Value Decomposition this approach does not rely on the imputation of missing values, but only considers actual present ratings for training. The algorithm tries to find two matrices $\mathbf{P} \in \mathbb{R}^{M\times K}$ and $\mathbf{Q} \in \mathbb{R}^{N\times K}$ that accurately represent user-item interactions. Both users and items get transformed to the same latent factor space of a fixed dimension K where their dot product is taken to measure their compatibility. A rating r_{ui} is then approximated by $\hat{r}_{ui} = b_{ui} + \mathbf{q}_i^T \mathbf{p}_u$. In order to avoid over-fitting to the data a regularize term is added that penalizes large magnitudes of $\mathbf{p}_u$ and $\mathbf{q}_i$. The associated least squares problem is the following:

$$\min_{\mathbf{q}_*,\mathbf{p}_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - b_{ui} + \mathbf{q}_i^T\mathbf{p}_u)^2 + \lambda(\|\mathbf{q}_i\| + \|\mathbf{p}_u\|) \tag{4}$$

This problem again can be solved using either ALS or SGD. Here we assume that the biases have been estimated in a preprocessing step, however it is also possible to learn them at the same time as $\mathbf{P}$ and $\mathbf{Q}$.

2) SVD++: Based on the work of Simon Funk, several improvements to regularized SVD were suggested, that also consider the item-item relations present in the data. Theses methods include "NSVD" [6] and "Asymmetric-SVD" [1] which both aim to model the user vector $\mathbf{p}_u$ based on the items that given user rated. For example in NSVD $\mathbf{p}_u$ is modeled via the sum $\left(\sum_{j \in R(u)} \mathbf{x}_j\right) / \sqrt{|R(u)|}$ where $\mathbf{X} \in \mathbb{R}^{N \times K}$ is a item dependent factor matrix learned from the data. The term $|R(u)|^{-1/2}$ serves as a user dependent normalization constant, which stabilizes the sum's variance across the range of observed values of $|R(u)|$ [4].

Building on top of these ideas two very similar models were developed, that again include an explicit $\mathbf{p}_u$ vector for every user. These models are dubbed "MF-NSVD1" [7] and "SVD++" [1], [4] respectively. In SVD++ in addition to the user vector $\mathbf{p}_u$ the sum over all rated items is kept, which leads to the following approximation formula:

$$\hat{r}_{ui} = b_{ui} + \mathbf{q}_i^T \left(\mathbf{p}_u + |R(u)|^{-1/2} \sum_{j \in R(u)} \mathbf{x}_j \right) \tag{5}$$

Observing that the modified user term is completely independent of the current item leads to a very quick learning and prediction algorithm that has a linear time complexity with regard to the input size. Similar to previous models we solved the regularized least squares problem using SGD.

D. Neighborhood Models

Historically neighborhood models enjoyed a huge popularity in Collaborative Filtering applications. These models include user-user and item-item based models. In user-user models one tries to find similar users to a given user u and use their existing ratings to recommend new items to u. Conversely, in item-item based models one tries to find similar items to the items a given user already rated, and recommend those. Usually the number of distinct items are less than the number of distinct users which results in item-item models being more efficient, because it is faster to compute similarities between all items than it is to compute them between all users. In addition, item-item models tend to provide a better user experience, because they offer an explanation for their recommendations and users are usually more familiar with their previous rated items than other users that are supposedly similar to them.

1) A factorized User-User Model: Given the fact that we already integrated an item-item model into the regularized SVD to obtain SVD++ we shift our focus here to user-user models. One possible approach to implement such a model is given with the following formula [3], [4]:

$$\hat{r}_{ui} = b_{ui} + |R(i)|^{-1/2} \sum_{v \in R(i)} (r_{vj} - b_{vj}) w_{uv} \tag{6}$$

Here $\mathbf{W} \in \mathbb{R}^{M \times M}$ is the similarity matrix between users. These similarities are weighted by the difference between the real rating and the baseline estimate. This is motivated by the fact that we want to be able to adjust our actual estimate when either the difference to the estimate or the weight between users is large. If either one is close to zero, it means that either our baseline estimate is already very good or the current users are not similar, so that we do not want to alter our current estimate [3], [4].

The current model has time and space complexity both quadratic in the number of users, which is prohibitively expensive when the number of users is large. Assuming that the weights can be expressed as $w_{uv} = \mathbf{p}_u^T \mathbf{z}_v$ for suitable choices of $\mathbf{P}, \mathbf{Z} \in \mathbb{R}^{M \times K}$ remedies this effect. Using this assumption, the rating estimate can be expressed as the following [3], [4]:

$$\hat{r}_{ui} = b_{ui} + |R(i)|^{-1/2} \mathbf{p}_u^T \left(\sum_{v \in R(i)} (r_{vj} - b_{vj}) \mathbf{z}_v \right) \tag{7}$$

The term containing the sum does not depend on the current user u and thus can be precomputed, leading to linear time and space complexity. Once more the associated least squares problem can be solved with the help of SGD, while regularizing by penalizing large magnitudes of the vectors involved.

E. Blending of Results

In order to be able to make use of all predictors to obtain our final rating estimate we applied binned Ridge Regression. Ridge Regression is a generalized linear model that includes the weight vector's L_2 norm in the cost function to penalize large magnitudes. Given a data matrix $\mathbf{X}$ and a target vector $\mathbf{y}$ it is described in the following way:

$$\min_{\boldsymbol{\beta}} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2 + \lambda \|\boldsymbol{\beta}\|^2 \tag{8}$$

The solution has form $\boldsymbol{\beta}^* = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$ and can be found very efficiently. The regularize parameter λ has to found via cross-validation. For our purposes the columns of $\mathbf{X}$ contain the estimated ratings for which the true ratings are available in $\mathbf{y}$. With fixing a column of $\mathbf{X}$ to only consist of ones we are introducing a constant offset. In addition we partitioned the user ratings into a fixed number of equal sized bins, where we grouped users with a similar number of issued ratings together. This allows us to optimize the weights for every bin independently, which is motivated by the fact that the algorithms perform differently depending on the number of ratings issued by a given user. The usage of Binned Ridge Regression was motivated by the "The BigChaos Solution to the Netflix Grand Prize" [8].

III. Results

A. Dataset

The dataset which was used throughout the development of our algorithm was made available on the project course webpage [9]. It contains the ratings of $10\ 000$ users on 1000 movies on a scale between 1 and 5. Naturally, not every user rated every movie, in fact only $1\ 388\ 107$ of the $10\ 000\ 000$ possible ratings were present leading to a data sparsity of 86%.

B. Evaluation of Results

In order to train and test the developed algorithm the present ratings were randomly split in two disjoint sets of equal size. The first set then formed the training set, i.e. data that was trained on, and the second set was used for testing the algorithm. The chosen metric is "Root Mean Squared Error" (RMSE) which is defined in the following way [1], [3], [4]:

$$\sqrt{\sum_{(u,i)\in \mathrm{TestSet}} \frac{(r_{ui}-\hat{r}_{ui})^2}{|\mathrm{TestSet}|}} \tag{9}$$

Here r_{ui} denotes the real rating while $\hat{r}_{ui}$ is the approximation through the algorithm. With ratings restricted to the interval $[1,5]$ valid RMSE values range between 0 and 4, with lower scores being better. In addition, the raw CPU time was evaluated for the project grade. This favors solutions that are efficient with regard to time.

IV. Discussion

V. Summary

References

[1] Y. Koren, "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08, Las Vegas, Nevada, USA: ACM, 2008, pp. 426–434, ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1401944.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009, ISSN: 0018-9162. DOI: 10.1109/MC.2009.263.

[3] Y. Koren, "Factor in the Neighbors: Scalable and Accurate Collaborative Filtering," *ACM Trans. Knowl. Discov. Data*, vol. 4, no. 1, pp. 1–24, Jan. 2010, ISSN: 1556-4681. DOI: 10.1145/1644873.1644874.

[4] Y. Koren and R. Bell, "Advances in Collaborative Filtering," English, in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds., Springer US, 2011, pp. 145–186, ISBN: 978-0-387-85819-7. DOI: 10.1007/978-0-387-85820-3_5.

[5] S. Funk. (Dec. 2006). Netflix Update: Try this at Home, [Online]. Available: http://sifter.org/~simon/journal/20061211.html (visited on 06/16/2015).

[6] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proceedings of KDD Cup Workshop at SIGKDD'07*, San Jose, CA, USA, 2007, pp. 39–42.

[7] G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk, "A unified approach of factor models and neighbor based methods for large recommender systems," in *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, IEEE, Aug. 2008. DOI: 10.1109/icadiwt.2008.4664342.

[8] A. Töscher, M. Jahrer, and R. M. Bell, "The BigChaos Solution to the Netflix Grand Prize," *Netflix prize documentation*, 2009.

[9] Computational Intelligence Lab. (2015). Collaborative Filtering, [Online]. Available: http://cil.inf.ethz.ch/applications/collaborative_filtering (visited on 06/16/2015).