Recommender system based on ASVD

Maria Zharova & Merkulov Daniil
Optimization Project. MIPT
May 2022

Introduction

Matrix factorization methods based on SVD decomposition are widely used in the implementation of recommender systems. One of the most effective at the moment is ASVD, it gained fame after the Netflix prize competition in 2009. This project implements a recommender system, based on this approach, that was tested on data from the one movie site [1].

The main idea of ASVD

Recall that in the standard singular value decomposition, the prediction is made according to the following rule:

$$r_{ui} = b_{iu} + p_u^T q_i,$$

here each user u is associated with a vector of factors $p_u \in R_f$, and each element i is associated with a vector of factors of elements $q_i \in R_f$.

In ASVD, the previous user factor p_i was replaced by the sum that takes into account implicit information [2]:

$$r_{ui} = b_{iu} + q_u^T(|R(u)|^{\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj})x_j + |N(u)|^{\frac{1}{2}} \sum_{j \in R(u)} y_j)$$

- here each element of i is associated with three-factor vectors $q_i,\ x_i,\ y_i \in R_f$, i.e. instead of explicitly parameterizing users, we represent them through elements that they might prefer. $R(u)$ and $N(u)$ are the user's ratings and implicit reviews, respectively.

Advantages of the ASVD model

Let us explain the advantages of the modified formula for ASVD [2]:

- The main advantage is the inclusion of implicit relationships that provide an additional indication of user preferences. To give the necessary significance to explicit and implicit reviews, N and R are preceded by the necessary coefficients in the model algorithm; in particular, by setting relative values;
- **complexity reduction** due to the use of fewer parameters (user characteristics p_u are replaced by object features $q_i,\ x_i,\ y_i$);
- the ability to process new users without retraining the model and estimating new parameters due to the lack of parameterization - thus, with the addition of new data, an updated result can be obtained faster;
- predictions are well explained because are a direct function of the answers of past users - such a structure allows you to determine which of the past actions of the user most influence the calculated forecast.

Algorithm

Let us consider in more detail, using the example of pseudocode, the work of the ASVD algorithm:

Part 1. Realization of ASVD [3].

$\tilde{R} = \mu + b_u + b_i + P \cdot Q$
#μ, b_u, b_i are global effects: mean rating, user bias, item bias
without global effects $\tilde{r}_{ui} = \sum_k p_{uk} q_{ik}$, where
$\tilde{r}_{ui}$ - estimated rating for user u on item i
p_{uk} - how much user u likes feature k
q_{ik} - how much feature k is important in item i.

For new users we will calculate the predictions using the following:
$P = R \cdot Z$
i.e. $p_{uk} = \sum_j r_{uj} z_{jk} = r_u z_k$, where
r_u means how much user u likes the items
z_k means how much the feature k is present in the items.

Thus, the final formula for predictions is:
$\tilde{R} = \mu + b_u + b_i + R \cdot Z \cdot Q$
i.e. $\tilde{r}_{ui} = \mu + b_u + b_i + \sum_k \sum_j r_{uj} z_{jk} q_{ki}$, where
r_{uj} is recorded user preference for item j
z_{jk} is a latent feature preference factor
q_{ki} means how much latent feature k is present in item i

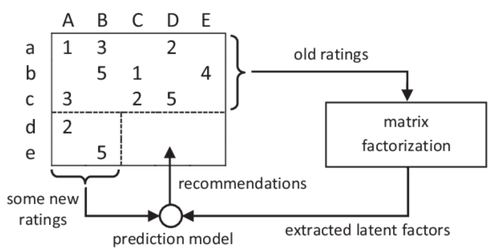
Part 2. Find the optimal parameters.

As usual, we learn the values of involved parameters by minimizing the regularized squared error function associated with

$$\min_{q_*, z_*, r_*, b_*} \left(\sum_{u,i} (r_{ui} - \tilde{r_{ui}})^2 + \lambda (b_u^2 + b_i^2 + ||q_i||^2 + \sum_j ||z_j||^2 + \sum_j ||r_j||) \right)$$

We employ a simple gradient descent scheme to solve the system. On the Netflix data, we used 30 iterations, with step size of 0.002 and $\lambda = 0.04$.

Here is the simplest visualization of the work recommender system using matrix factorization [4]:



Data

We have 3 datasets: users, movies, ratings. The first contains information about the users of the site, indexing goes by user_id; similarly, in the second dataset there is information about the films, each is also assigned a movie_id; the latter contains the rating history (including the user's id and movie id).

Thus, we merge these three files at first. Id-features are the basis for matrices $R,\ Z$ and Q; other features are represented in "global effects" $\mu,\ b_u$ and b_i.

Results & Conclusion

In the course of the work, a product recommendation system was implemented in Python, based on the ASVD algorithm (using the Python library TensorFlow). The numerical results were obtained for the dataset with movie ratings:

$precision \approx 0.59$

$recall \approx 0.33$

Also, according to this dataset, a recommendation model was built using Polara [5], its quality metric is precision = 0.56, recall = 0.29.

All results can be viewed on my GitHub [6].

This project was the final one in the Optimization course, which gave me an advantage in the exam. It is supposed to publish this work in the scientific journal "Proceedings of the MIPT", make a presentation at the MIPT scientific conference and use the algorithm in further work, for example, in Kaggle competitions.

Acknowledgements

This material is based upon work supported by the article "Factorization meets the neighborhood: a multifaceted collaborative filtering model" and movie data.

References

[1] https://movielens.org

[2] Yehuda Koren, Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model, 2008.

[3] EIT Digital, Politecnico di Milano, Paolo Cremonesi "Advanced Recommender Systems" (coursera).

[4] Li Pu, Boi Faltings "Understanding and improving relational matrix factorization in recommender systems", 2013.

[5] https://github.com/evfro/polara

[6] https://github.com/MariaZharova/HM-recommender-system