Incremental SVD++ for recommender systems

Phuong Anh Hoang
Advanced Program in Computer Science
University of Science, VNU-HCMC
hpanh@apcs.vn

Dung Anh Cao
Advanced Program in Computer Science
University of Science, VNU-HCMC
cdanh@apcs.vn

***Abstract*—Recommender systems were developed to help users to deal with information. These systems has become an important part of e-commerce. With the tremendous growth of e-commerce, scalability is one of biggest challenges for recommender systems. To addressing the scalability problem, we propose *Incremental SVD++* method. *Incremental SVD++* is based on a state-of-the-art recommendation algorithm SVD++. To our knowledge, the work reported is the first to extend SVD++ with incremental updates. Our method improves performance of classic SVD++, while maintaining the recommendation quality**

***Keywords*—recommender systems, incremental, SVD++**

I. Introduction

The explosive growth of the World-wide-web and the emergence of e-commerce gives users many benefits. However, information overload and the abundance of choice became a big challenge. Recommender systems have been used for by various e-commerce websites to help users to deal with this challenge. These systems attempt to predict information/items that a user may be interested in.

One of the earliest and most successful recommender technologies is collaborative filtering (CF). CF techniques relies only on user past behaviour (previous transactions or product ratings) allows uncovering complex and unexpected patterns that would be difficult or impossible to profile using known data attributes. In order to generate recommendations, CF systems need to compare basically different objects: items against users. There are two main approaches to help such a comparison, which make the two main parts of CF: the neighbourhood approach and latent factor models. Neighborhood methods are focused on computing the relationships between users or items. Latent factor models, sush as Singular Value Decomposition (SVD) approach which transforming both users and items to the same latent factor that are comparable to each other.

The tremendous growth of customers and products in recent years poses two key challenges for recommender systems: quality and scalability. In some ways there are conflicts in these two challenges. If algorithm spends less time for searching a neighbours, it will be more scalable and worse its quality.

To address these challenges, many researchers suggest that SVD-based approach may be a solution in some case. SVD-based approach produced results that were better than a traditional CF algorithm most of the time when applied to a Movie data set [1]. However, the matrix factorization step of SVD-based approach is computationally very expensive because it takes a lot of memory and time to factorize a matrix. This limitations make SVD-based approach less suitable for large scale deployment in e-commerce system.

In the other hand, a new approach of SVD was presented by Simon Funk in the context of the Netflix Prize [2]. This method used an approximate way to compute the low-rank approximation of the matrix by minimizing the squared error loss.

Since then, many other SVD-based models have been created. One of them is SVD++ model, which is desribed by Yehuda Koren , the winner of Netflix Prize in his paper. SVD++ model relies on both explicit feedback and implicit feedback [3]. Implicit feedback is an especially valuable information source for users who do not provide much explicit feedback. It indirectly reflect opinion through observing user behavior. Hence, SVD++ uses implicit feedback as a secondary source of information. This model improves prediction accuracy to some extent.

In this paper, we show how to extend SVD++ with incremental updates, through the *Incremental SVD++ model*. Our method improves performance of classic SVD++, while maintaining the recommendation quality. The pre-computed model is updated incrementally at the time of rating activity and recommended items are modified based on newest data.

The rest of the paper is organized as follows. The next section gives a brief overview of the SVD-based latent factor models. Section 3 outlines our incremental SVD++ algorithm. Section 4 presents our experimental procedure, results and discussion. The final section provides some concluding remarks and future research directions.

II. Related work

Latent factor models approach collaborative filtering with the holistic goal to uncover latent features that explain observed ratings. Most of the Matrix Factorization models are based on the latent factor model [4]. Matrix Factorization approach is found to be most accurate approach to reduce the problem from high levels of sparsity in recommender system database. These methods have become popular recently by combining good scalability with predictive accuracy. They offers much flexibility for modeling various real-life applications.

A. A Basic Matrix Factorization Model

Matrix factorization models transform both users and items into the same latent feature space of dimensionality f. The prediction is done by taking an inner product in that space:

$$\hat{r}_{ui} = q_i^T p_u \tag{1}$$

where each item i is associated with a vector $q_i \in R^f$, and each user u is associated with a vector $p_u \in R^f$. For a given item i, the elements of qi measure the extent to which the item possesses those factors; for a given user u, the elements of pu measure the extent of interest the user has in items that are high on the corresponding factors. Therefore, their dot product $q_i^T p_u$ denotes the overall interest of the user in characteristics of the item. We learn the values of the factor vectors (p_u and q_i) by minimizing the regularized squared error function associated with (1)

$$min_{q*,p*} \sum_{(u,\, i)\in K} \left(r_{ui} - \; q_i^T p_u\right)^2 + \; \lambda \left(\; \|q_i\|^2 + \; \|p_u\|^2 \right) \tag{2}$$

Here, is the set of the (u, i) pairs for which rui is known the training set. A simple gradient descent technique was applied successfully to solving (2). We loop through all known ratings in K. For each given rating $r_u i$, a prediction $\hat{r}_{ui}$ is made, and the associated prediction error $e_{ui} \overset{\text{def}}{=} \; r_{ui} - \hat{r}_{ui}$ is computed. For a given training case $r_u i$, we modify the parameters by moving in the opposite direction of the gradient, yielding:
$q_i \leftarrow q_i + \; \gamma\left(e_{ui}p_u - \lambda_2 q_i\right)$
$p_u \leftarrow p_u + \; \gamma\left(e_{ui}q_i - \lambda_2 p_u\right)$
where γ is the learning rate.

B. SVD with Bias Terms (Bias-SVD)

We add biases to the regularized SVD model, one parameter b_u for each user and one b_i for each item. The parameters b_u and b_i indicate the observed deviations of user u and item i from the average. For example, the average rating over all movies is 3.7 stars. Furthermore, *Titanic* is better than an average movie, so it receives 0.5 stars above the average. On the other hand, *Joe* is a critical user, who tends to rate 0.3 stars lower than the average. Thus, 0.5 and -0.3 reflects the deviations of *Titanic* and *Joe* from the global average 0.37. Therefore, in this model, a rating is predicted by the rule:

$$\hat{r}_{ui} = \; \mu + \; b_u + \; b_i + q_i^T p_u \tag{3}$$

To learn the model parameters, we should also minimize the regularized squared error:

$$min_{q*,p*,\; b*} \sum_{(u,\, i)\in K} \left(r_{ui} - \; \mu + \; b_u + \; b_i + \; q_i^T \; p_u\right)^2 + \lambda \left(b_u{}^2 + \; b_i{}^2 + \; \|q_i\|^2 + \; \|p_u\|^2\right) \tag{4}$$

Similarly, we use the gradient descent method to get the update rule for each parameter:
$b_u \leftarrow b_u + \; \gamma\left(e_{ui} - \; \lambda_1 b_u\right)$
$b_i \leftarrow b_i + \; \gamma\left(e_{ui} - \; \lambda_1 b_i\right)$
$p_u \leftarrow p_u + \; \gamma\left(e_{ui}q_i - \lambda_2 p_u\right)$
$q_i \leftarrow q_i + \; \gamma\left(e_{ui}p_u - \lambda_2 q_i\right)$
where $e_{ui} \overset{\text{def}}{=} \; r_{ui} - \hat{r}_{ui}$ and γ is the learning rate.

C. SVD with Implicit Feedback (SVD++)

III. Incremental SVD++

IV. Experiments

V. Conclusion

References

[1] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. (2000). ”Application of Dimensionality Reduction in Recommender SystemA Case Study”, WEBKDD2000.

[2] S. Funk, ”Netflix Update: Try This At Home”, http://sifter.org/simon/journal/20061211.html, 2006.

[3] Y. Koren, Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model, *Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[4] Yehuda Koren, Matrix Factorization Techniques for Recommender Systems, Published by the IEEE Computer Society, IEEE 0018-9162/09, pp. 42- 49, IEEE, August 2009.