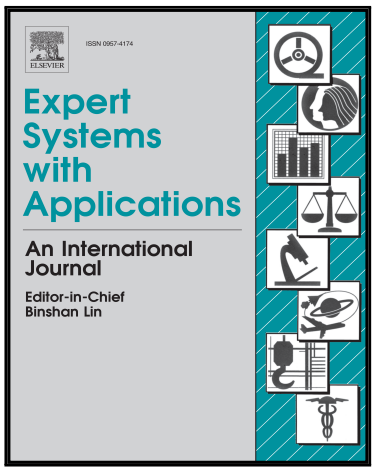


Multi-Modal Matrix Factorization with Side Information for Recommending Massive Open Online Courses

Panagiotis Symeonidis, Dimitrios Malakoudis

PII: S0957-4174(18)30631-6
DOI: https://doi.org/10.1016/j.eswa.2018.09.053
Reference: ESWA 12241

To appear in: *Expert Systems With Applications*

Received date: 17 June 2018
Revised date: 26 September 2018
Accepted date: 27 September 2018

Please cite this article as: Panagiotis Symeonidis, Dimitrios Malakoudis, Multi-Modal Matrix Factorization with Side Information for Recommending Massive Open Online Courses, *Expert Systems With Applications* (2018), doi: https://doi.org/10.1016/j.eswa.2018.09.053

Highlights

- We recommend courses to users of Massive open online courses (MOOCs).
- We perform Multi-Modal Matrix Factorization with Side Information.
- We outperform the classic MF algorithm and other novel algorithms.

Multi-Modal Matrix Factorization with Side Information for Recommending Massive Open Online Courses

Panagiotis Symeonidis[1] and Dimitrios Malakoudis[2]

[1]Free University of Bolzano, Faculty of Computer Science, 39100, Italy
psymeonidis@unibz.it
[2] Aristotle University, Department of Informatics, Thessaloniki 54124, Greece
dmalakoudis@gmail.com

Abstract. Massive open online courses (MOOCs) have recently gained a huge users' attention on the Web. They are considered as a highly promising form of teaching from leading universities such as Stanford and Berkeley. However, users confront the problem of choosing among thousands of offered MOOCs. In such a scenario of severe "information overload", recommender systems can be very useful to recommend the right course to a user, since they base their operation on past user's log history. For example, Coursera recommends courses to users so that, they can acquire those skills, that are expected from their ideal job. These user's preferences are not expected to be independent from others choices, as users follow trends of similar behaviour. In this paper, we propose, xSVD++, where the "x" means that it is a multi-dimensional Matrix Factorization (MMF) model combined with Collaborative Filtering (CF) algorithm, which exploits information from external resources (i.e., users' skills, courses' characteristics, etc.) to predict course trends and to perform rating predictions according to them. Our experimental results indicate that xSVD++ is superior over classic and non-negative matrix factorization algorithms and the state-of-the-art CMF, and SVD++ algorithms.

1 Introduction

Massive Open Online Courses (MOOCs) platforms offer thousands of different courses. Recommending someone the right course for acquiring the skills that are expected from his future ideal job is an important task. For example, the learning outcomes of a course can describe to what extent a person holds a particular qualification/skill. Based on the courses' learning outcomes, we can match the needs (in terms of competence, skills and knowledge) of the labour market with those provided by MOOCs platforms.

In last years, an instance of SVD [5] has been developed, known as UV-decomposition. We can apply UV-decomposition over a user-course rating matrix R to reduce its dimensions and remove noise from data. To do this, we preserve a small number of k latent features (i.e., dimensions) with the objective to reveal

the mainstream users' preferences. For example, in Figure 1, we plot users and courses, assuming that k has been tuned to 2.

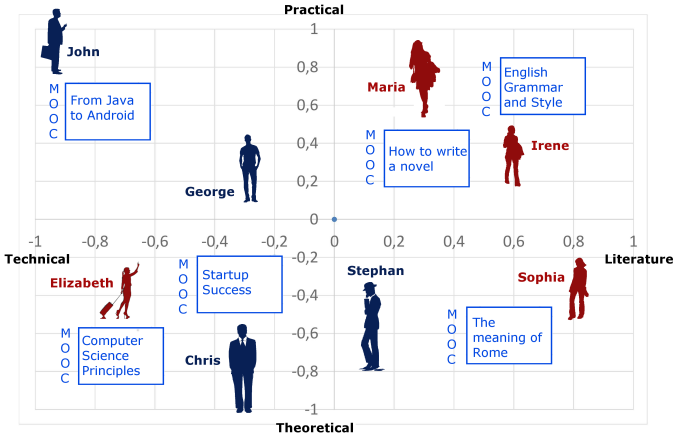


Fig. 1. Users and Courses in the 2-D space.

As shown in Figure 1, courses/users that are placed in close distance, are the most suitable/similar to each other.As shown, women prefer literature courses, whereas men choose the technical ones. Specifically, the course "English Grammar and Style" can be recommended to Maria and Irene, whereas "From Java to Android" course is more suitable to John. Please notice that matrix decomposition has also revealed a second separation, which takes place among people's preference, towards practical and theoretical types of courses.

In this paper, we predict users' ratings over courses based on multi-modal matrix factorization (MMF), by extending the well-known SVD++ algorithm, which exploits only implicit information from the same resource. In contrast, we exploit information from several external resources/matrices, which makes SVD++ to become just a simplified special case of our algorithm and can be easily derived from it.

To do this, we combine information of users' preferences on courses with the courses' content features. Firstly, we exploit information from the users' preferences on courses. That is, we update a user-course matrix R, each time a user selects or rates a course. Secondly, we exploit information extracted from the course characteristics and the skills that a student possesses after completing a course. In particular, we extract the title, the syllabus and the learning outcomes of a course as a bag-of-words describing its content. These course attributes have been chosen because we want to find what are the skills that students acquire after completing a course. This is important because our method recommends courses to users so that, they can acquire those skills for getting their dream job. Thus, we need to simultaneously discover the appropriate skills that someone

needs for getting a job and the skills that someone acquires after completing a course.

Based on these features, we build a course-skill matrix CS, which holds the total number of appearances of a skill inside a course's title, syllabus and its learning outcomes. Thirdly, we exploit user-skill matrix US, where its rows refer to the different users and the columns refer to the different skills that they possess. In summary, we apply multi-modal matrix factorization on the user-course matrix R, fusing also information from the course-skill matrix CS and user-skill matrix US.

The rest of this paper is organized as follows. Section 2 summarizes the related work, whereas the proposed approach is described in Section 3. Section 4 describes our algorithm in pseudocode form for the rating prediction task (i.e., matrix completion task). Section 5 describes the generation of the recommendation list of courses. Experimental results are given in Section 6. Section 7 discusses possible extensions and challenges of our method. Finally, Section 8 concludes this paper.

2 Related work

Singular Value Decomposition [5] is a well-known factorization technique that factorizes a matrix into three matrices. An instance of SVD, known as UV decomposition, searches for two matrices (U and V), whose their multiplication gives an approximation of the original matrix R. That is, if we have a matrix R with n rows and m columns, the SVD decomposition consists of three matrices (U, S and V), where S is the matrix that keeps the singular values of R. In the UV decomposition [21], there is a matrix $\mathcal{V}$, which is the product of S and $V^\top$. In other words, S is left-blended into matrix $V^\top$ and produces matrix $\mathcal{V}$. Thus, UV decomposition consists of two matrices, one U with n rows and k columns and one $\mathcal{V}$ with m rows and k columns, such that $U\mathcal{V}^\top$ produces R with the blank entries filled and a small deflection of the initial values.

$$R \approx U\mathcal{V}^\top = \hat{R} \tag{1}$$

The prediction of a user' s rating on a course can be calculated by the dot product of the two vectors, which correspond to u_i and v_j, as seen in Equation 2.

$$\hat{r}_{ij} = u_i v_j^\top = \sum_{k=1}^{K} u_{ik} v_{kj} \tag{2}$$

The next step of the method is to find a way to obtain U and V. One way to solve the problem is to initialise the two matrices with some random values and compute each time how "different" their product is compared with R. Then, we will iteratively try to minimise this difference. Such a method is called *Gradient Descent*, aiming to find a local minimum of the difference.

The difference actually is the square error between the real rating and the predicted one and can be calculated using Equation 3 for each user-course pair:

$$e_{ij}^2 = \sum_{i,j \in R} (r_{ij} - \hat{r}_{ij})^2 = \sum_{i,j \in R} (r_{ij} - \sum_{k=1}^{K} u_{ik} v_{kj})^2 \quad (3)$$

Based on Equation 3 and the definition of the Frobenius norm, we can formulate the objective function as follows:

$$G = \|R - \hat{R}\|_F^2 = \|R - UV\|_F^2 \quad (4)$$

A common extension to the basic UV-decomposition algorithm is to introduce regularization to avoid overfitting. The *overfitting* problem occurs when a model begins to "memorize" training data rather than "learning" to generalize from the trends of data. In our objective function, to avoid overfitting we add a parameter λ and modify the squared error as follows:

$$G = min \sum_{i,j \in R} (r_{ij} - u_i v_j^T)^2 + \lambda(\|U\|^2 + \|V\|^2) \quad (5)$$

The new parameter λ is used to control the magnitudes of the user-latent feature and course-latent feature vectors. So, the challenge is to minimize the error of the differences among the real and the predicted rating values of items. Please recall that there is no closed form solution for minimising function G, and we can only use a numerical method, such as *Gradient Descent* or *Alternating Least Squares*, to solve it. There have been proposed several ways to compute matrices U and V. For example, Lee and Seung [10] proposed the definition of a cost function (i.e., $\|R - UV\|^2$), which can be minimised either by using multiplicative or additive update rules.

Another widely known method in dimensionality reduction and data analysis is *Non-negative Matrix Factorization* (NMF) [11]. The NMF algorithm factorises a matrix A in two matrices U and V, with the property that both matrices have no negative elements. Please notice that in order to prevent that the values of the matrices U and V become negative, after the application of each update rule of the gradient descent method, we set any derived negative value of matrices U and V to be equal to 0. This non-negativity makes the resulting matrices more suitable for clustering of objects.

A significant improvement on the prediction accuracy of classic Matrix Factorisation (MF) algorithm may be obtained through the incorporation of implicit feedback into the MF model [16, 8, 9]. For example, the user-course rating matrix does not only tell us the rating values, but also which courses users rate, regardless of how they rated these courses. Therefore, the prediction rule of Equation 2 can be extended with a complementary constraint. That is, each user u can be connected with a set of items $N(u)$, for which s/he expressed an implicit preference, as shown in the extended prediction rule of Equation 6.

$$\hat{r}_{ij} = \mu + bu_i + bv_j + \sum_{k=1}^{K} (u_{ik} + \frac{\sum_{i \in N(u)} y_{ki}}{\sqrt{|N(u)|}}) v_{kj} \quad (6)$$

Now, a user u is modeled as $\boxed{u_{ik} + \frac{\sum_{x \in N(u)} y_{ki}}{\sqrt{|N(u)|}}}$, where y_{ki} describes if a user has showed an implicit preference for an item i in $N(u)$. Moreover, μ is the mean value of all ratings, whereas bu_j and bv_j parameters indicate the observed deviation from the average of user u and item v, respectively. This prediction model is known as SVD++. It has been experimentally proven that SVD++ is more accurate compared to classical matrix factorization [9]. SVD++ incorporates into the prediction rule a set of predictors that learn latent factors by exploiting the implicit information of user preferences. For example, a user that has purchased many books by the same author probably likes the author.

Extensions of classic MF algorithm and other methods [1, 15] have been applied for course recommendations. [3] provided top-N course recommendation based on MF, by incorporating into their models additional student and course academic features (e.g., student major, course topic, etc.) and by building multi-granularity student and course groups accordingly. In the MOOC domain, to reduce the high students' drop-out rates, [23] provided recommendations of useful forum threads to students based on their blog history inside a MOOC discussion forum. Moreover, [22] proposed a matrix factorization method that considers also constraints (e.g., students should not be over-burdened with too many questions, etc.) for the task of providing question recommendations in discussion forums that concern a MOOC.

In contrast to the aforementioned works of [9] and [22, 23], our proposed method also incorporates external additional information into the user and course profiles. This additional information is taken from two external resources of the user-skill and the course-skill matrices, as will be shown in Section 3. In particular, we differ from [9] and from [23] implicit feedback, by the fact that we enrich the user and course profile with explicit information (i.e. exploiting the user-skill and course-skill matrices). In contrast, they both try to gain knowledge by re-processing the same data to capture implicit feedback. Since our models exploits autonomous external resources of information is more effective in terms of rating prediction.

There are also works that incorporate side information in matrix factorization. [20] proposed the collective matrix factorization (CMF), which simultaneously factorizes several matrices, sharing parameters among factors when an entity participates in multiple relations. For example, they proposed to simultaneously factorize user-item and item-feature matrices, where the entity of items creates a common latent space, since it exists in both matrices. [25] introduced a personalized recommendation algorithm for LBSNs, which performs Personalized Collaborative Location and Activity Filtering (PCLAF). Unlike CLAF [26], PCLAF treats each user differently and uses a coupled tensor and matrix factorization to provide personalized recommendations.

[13] proposed a set of Sparse LInear Methods (SSLIM), which involve an optimization process to learn a sparse aggregation coefficient matrix based on both user-item purchases matrix and item side information. Collective SLIM (cSLIM) was also proposed by Ning and Karypis [14] and uses also side information to

provide item recommendations. However, SLIM captures only relations between items that have been co-rated/co-purchased by at least one user (i.e., can not capture transitive relations). To overcome this, [7] proposed the Factored Item Similarity Model (FISM), which combines the idea of SLIM with the idea of Regularised SVD [9]. FISM learns the item-item similarity matrix as a product of two latent factor matrices.

Another well-known model is Bayesian Personalized Ranking (BPR) [18], which formulates the item recommendation problem not as a classification problem, but as a ranking problem using pairs of positive items (in the train set) and negative items (not in the train set) as pairwise input. BPR optimises a simple ranking loss such as AUC (the area under the ROC-curve) and uses matrix factorization as the ranking function, that can be optimized directly using a stochastic gradient algorithm. BPR computes the pair-wise ranking loss of the objective function (not the element-wise square loss like the aforementioned methods do), aiming to optimise the ranking quality (not the rating prediction quality). In addition, [17] proposed Factorization machines (FM) as a generic model that allows to describe a wide variety of data by feature engineering. FMs combine the flexibility of feature engineering with the advantages of factorization. Recently, [6] for obtaining robust personalized ranking results, associate content information of entities (i.e., users and items) with implicit feedbacks to develop a Pairwise Ranking Factorization Machines (PRFM) [6], which alleviates the cold start problem and enhances the performance of personalized ranking by incorporating BPR learning [18] with Factorization Machines [17].

3 Our proposed method

In this Section, we extend the classic MF objective function (Equation 5), which is shown in Section 2, by fusing into it additional information, that comes from the course-skill and user-skill matrices, respectively. Then, we generate the top-N recommendation list of courses. Conclusively, the provided recommendations consider the existence of user rating trends, as the similarities are computed in the reduced k-dimensional space, where the k dimensions correspond to trends of users.

In the following, to ease the discussion, we will use a running example illustrated in Figure 2, where U_{1-4} are users, C_{1-6} are courses and S_{1-4} are skills. In particular, the null (not rated) cells of user-course matrix R (Figure 2a) are presented with dash. Moreover, in Figure 2b, the elements of course-skill matrix CS show how many times a skill keyword is included in course's title, syllabus and its learning outcomes. Please notice that the user-skill matrix US of Figure 2c is not a matrix which is derived transitively from the combination of the user-course matrix R with the course-skill matrix CS. In contrast, it should be presumed as an autonomous resource of information, which is explicitly and externally are given by users.

	C_1	C_2	C_3	C_4	C_5	C_6
U_1	-	1	-	-	2	-
U_2	-	1	-	3	-	-
U_3	-	-	-	-	-	2
U_4	3	-	2	-	-	-

(a)

	S_1	S_2	S_3	S_4
C_1	2	1	1	0
C_2	1	2	1	0
C_3	0	1	4	0
C_4	2	1	0	2
C_5	3	1	1	0
C_6	0	0	0	4

(b)

	S_1	S_2	S_3	S_4
U_1	1	1	0	0
U_2	0	1	1	0
U_3	0	0	0	1
U_4	1	0	0	1

(c)

Fig. 2. (a) User-Course matrix R (b) Course-Skill matrix CS (c) User-Skill matrix US

3.1 Fusing additional information in the objective function

In this Section, we fuse additional information in the objective function (Equation 5), to improve its rating prediction accuracy. To do this, we will incorporate into our objective function, new parameters (i.e. mean value of ratings, users' biases in expressing their preference, etc.) and information extracted from the course-skill and user-skill matrices.

Firstly, we insert into the objective function the mean value μ of all ratings and the parameters bu and bc, which indicate the observed deviations (biases) of user u and course c, respectively, from the average. Biases are used to capture the fact that some users tend to rate higher than other, and some courses to get higher ratings than others. Thus, the new predicted rating $\hat{r}_{ij}$ of a user i on a course j is shown in Equation 7.

$$\hat{r}_{ij} = \mu + bu_i + bc_j + \sum_{k=1}^{K} u_{ik} v_{kj} \tag{7}$$

Furthermore, in order to avoid overfitting we upgrade the objective function, which we present in Equation 8. Variables λ_1 and λ_2 are used to penalize big values of bu_i, bc_j, u_{ik} and v_{kj}. In addition, u_{ik} is the value of the k^{th} latent feature for user i, whereas v_{kj} is the value of the k^{th} latent feature for course j, after the convergence of the factorization.

$$G = min \sum_{i,j} \left(r_{ij} - \mu - bu_i - bc_j - \sum_{k=1}^{K} u_{ik} v_{kj} \right)^2 + \lambda_1 (bu_i^2 + bc_j^2) + \lambda_2 \left(\sum_{k=1}^{K} u_{ik}^2 + v_{kj}^2 \right) \tag{8}$$

Our next insertion into the objective function concerns the course profile. The new prediction rule consists of two terms as we illustrate on Equation 9.

$$\hat{r}_{ij} = \sum_{k=1}^{K} u_{ik} (v_{kj} + \frac{\sum_{s=1}^{S} cs_{js} cs_{ks}^*}{\sum_{s=1}^{S} cs_{js}}) \tag{9}$$

The first term is v_{kj}, whereas the second one is the fraction $\boxed{\frac{\sum_{s=1}^{S} cs_{js} cs_{ks}^*}{\sum_{s=1}^{S} cs_{js}}}$ which includes information from matrices CS and CS^* of Figures 2b and 3, respectively, and adds the effect of how well a skill s describes course j. Please notice that Figure 3 shows a 2-dimensional (i.e., $k = 2$) latent space for the skill dimension. The course-skill matrix CS of Figure 2b holds the total number of appearances of a skill inside a course's title, syllabus and its learning outcomes. Moreover, CS^* is a skill – latent feature matrix and each of its elements cs_{ks}^* expresses how well a skill s characterises a course and describes its content. If a course j includes skill s for cs_{js} times inside its title, syllabus and learning outcomes, then element cs_{ks}^* is included cs_{js} times in the calculation of the objective function in Equation 10. We expect that with this addition, course profile will be more representative as it includes also the skill impact.

	S_1	S_2	S_3	S_4
k_1	cs_{11}^*	cs_{12}^*	cs_{13}^*	cs_{14}^*
k_2	cs_{21}^*	cs_{22}^*	cs_{23}^*	cs_{24}^*

Fig. 3. Skill latent feature matrix CS^*

We present the upgraded objective function in Equation 10. Please note that variable λ_3 is used to avoid overfitting caused by big values of CS^* matrix.

$$G = min \sum_{i,j} \left(r_{ij} - \sum_{k=1}^{K} u_{ik} (v_{kj} + \frac{\sum_{s=1}^{S} cs_{js} cs_{ks}^*}{\sum_{s=1}^{S} cs_{js}}) \right)^2 + \lambda_2 \sum_{k=1}^{K} \left(u_{ik}^2 + v_{kj}^2 \right) + \lambda_3 \sum_{k=1}^{K} \sum_{s=1}^{S} cs_{js} (cs_{ks}^*)^2 \tag{10}$$

In the next step, we reform the prediction rule, which is now based only on the user profile, and it is the sum of two terms, as shown in Equation 11.

$$\hat{r}_{ij} = \sum_{k=1}^{K}(u_{ik} + \frac{\sum_{s=1}^{S} us_{is}us^*_{sk}}{\sum_{s=1}^{S} us_{is}})v_{kj} \tag{11}$$

Following the same way as with the construction of the course profile, the first term of the user profile is u_{ik}, and it is complemented with the fraction $\boxed{\frac{\sum_{s=1}^{S} us_{is}us^*_{sk}}{\sum_{s=1}^{S} us_{is}}}$ which gathers information from matrices US and US^* of Figures 2c and 4 respectively and shows the influence of skill s in user's i profile modeling. Please notice that Figure 4 shows a 2-dimensional latent space for the skill dimension (i.e. k=2).

The user-skill matrix US holds information about the skills that users possess. Moreover, US^* is a skill – latent feature matrix and each of its elements us^*_{sk} expresses how well a skill s, describes the user profile. If a user possesses a particular skill s, then us^*_{sk} gets involved in the calculation of the user profile. Therefore, the upgraded objective function is depicted in Equation 12.

	k_1	k_2
S_1	us^*_{11}	us^*_{12}
S_2	us^*_{21}	us^*_{22}
S_3	us^*_{31}	us^*_{32}
S_4	us^*_{41}	us^*_{42}

Fig. 4. Skill latent feature matrix US^*

$$\begin{aligned} G = min \sum_{i,j} \left(r_{ij} - \sum_{k=1}^{K}(u_{ik} + \frac{\sum_{s=1}^{S} us_{is}us^*_{sk}}{\sum_{s=1}^{S} us_{is}})v_{kj} \right)^2 \\ +\lambda_2 \sum_{k=1}^{K} \left(u_{ik}^2 + v_{kj}^2\right) + \lambda_3 \sum_{k=1}^{K}\sum_{s=1}^{S} us_{is}(us^*_{sk})^2 \end{aligned} \tag{12}$$

3.2 Putting all together into the objective function

In this Section, we insert all the additional information to construct the full upgraded prediction rule (Eq. 13) and the objective function (Eq. 14), respectively. As it is presented in Equation 13, we have used all previous insertions, such as the mean value μ of all ratings, the bias vectors bu and bc of users and courses, user-skill and course-skill matrices US and CS and skill latent feature matrices US^* and CS^*.

$$\hat{r}_{ij} = \mu + bu_i + bc_j + \sum_{k=1}^{K}(u_{ik} + \frac{\sum_{s=1}^{S} us_{is}us^*_{sk}}{\sum_{s=1}^{S} us_{is}})(v_{kj} + \frac{\sum_{s=1}^{S} cs_{js}cs^*_{ks}}{\sum_{s=1}^{S} cs_{js}}) \tag{13}$$

Please note that in our proposed final objective function of Equation 14, we have included regularization terms λ_1, λ_2 and λ_3 to avoid overfitting caused by big values of elements bu_i, bc_j, u_{ik}, v_{kj}, us^*_{sk} and cs^*_{ks}.

$$G = min \sum_{i,j} \left(r_{ij} - \mu - bu_i - bc_j - \sum_{k=1}^{K} (u_{ik} + \frac{\sum_{s=1}^{S} us_{is} us^*_{sk}}{\sum_{s=1}^{S} us_{is}})(v_{kj} + \frac{\sum_{s=1}^{S} cs_{js} cs^*_{ks}}{\sum_{s=1}^{S} cs_{js}}) \right)^2 + \lambda_1 (bu_i^2 + bc_j^2) + \lambda_2 \sum_{k=1}^{K} \left(u_{ik}^2 + v_{kj}^2 \right) + \lambda_3 \sum_{k=1}^{K} \sum_{s=1}^{S} us_{is} (us^*_{sk})^2 + cs_{js} (cs^*_{ks})^2 \tag{14}$$

So, to minimize our final objective function G (Equation 14) we have to compute the partial derivatives for bu, bc, U, V, US^* and CS^*. The following partial derivatives of objective function G are used to form the new update rules. So after adding the new constraints, the new update rules are as follows in Equations 15 to 20:

$$bu_i \leftarrow bu_i + \eta_1 (e_{ij} - \lambda_1 bu_i) \tag{15}$$

$$bc_j \leftarrow bc_j + \eta_1 (e_{ij} - \lambda_1 bc_j) \tag{16}$$

$$u_{ik} \leftarrow u_{ik} + \eta_2 \left[e_{ij} (v_{kj} + \frac{\sum_{s=1}^{S} cs_{js} cs^*_{ks}}{\sum_{s=1}^{S} cs_{js}}) - \lambda_2 u_{ik} \right] \tag{17}$$

$$v_{kj} \leftarrow v_{kj} + \eta_2 \left[e_{ij} (u_{ik} + \frac{\sum_{s=1}^{S} us_{is} us^*_{sk}}{\sum_{s=1}^{S} us_{is}}) - \lambda_2 v_{kj} \right] \tag{18}$$

$$\forall s, \quad us_{is} > 0$$
$$us^*_{sk} \leftarrow us^*_{sk} + \eta_2 us_{is} \left[\frac{e_{ij}}{\sum_{d=1}^{S} us_{id}} (v_{kj} + \frac{\sum_{d=1}^{S} cs_{jd} cs^*_{kd}}{\sum_{s=1}^{S} cs_{jd}}) - \lambda_3 us^*_{sk} \right] \tag{19}$$

$$\forall s, \quad cs_{js} > 0$$
$$cs^*_{ks} \leftarrow cs^*_{ks} + \eta_2 cs_{js} \left[\frac{e_{ij}}{\sum_{d=1}^{S} cs_{jd}} (u_{ik} + \frac{\sum_{d=1}^{S} us_{id} us^*_{dk}}{\sum_{s=1}^{S} us_{id}}) - \lambda_3 cs^*_{ks} \right] \tag{20}$$

4 The Proposed Algorithm

In this Section, we will describe with pseudocode the implementation of our algorithm. The input data are the user-course rating matrix R ($R \in \mathbb{R}^{m \times n}$), the user-skill matrix $US \in \mathbb{R}^{m \times s}$, the course-skill matrix $CS \in \mathbb{R}^{n \times s}$, the objective

function G (Equation 14), the learning rates η_1 and η_2, the regularization parameters λ_1, λ_2 and λ_3, the number of total latent-features k, and the number of *steps* of algorithm's predictions. Moreover, we use the adjacency list $uSkill$ that holds user's skills and inserts relative information in arrays $uSkillsIndex$ and $uSkillsSum$, and the same data structures for courses.

As shown in the first line of Algorithm 1, we initialize with random values the two vectors bu and bc and the four matrices U, V, US^* and CS^*. Please notice that one of the dimensions of U, V, US^* and CS^* matrices is K ($k = K$). At line 2, we calculate the mean value of matrix R. Then, we start the repetitive process of prediction. At lines 8-12, we compute for each user and course the $SCUP$ (Skill Contribution to User Profile) and $SCCP$ (Skill Contribution to Course Profile) vectors, respectively. After that, the next step (lines 14 to 17) is to find the prediction error e, which is the difference between real and predicted rating. The prediction error e is then used to update the two vectors and the four matrices: bu and bc are updated at lines 18 and 19, whereas U, V, US^* and CS^* in lines 20 to 32, respectively. The process that has just described above will be repeatedly executed, until objective function G (Equation 14) ceases to improve or until the number of maximum predictions (iterations) that we have set is reached (line 46). In the end, we will have computed the values bu, bc, U, V, US^* and CS^* for getting the minimum value of G. The final step is to use these values to calculate the predicted rating of matrix $\hat{R}$ (line 42). In the next Section, we will describe how we compute the top-N course recommendation list.

5 Generating the Course Recommendation List

In this Section, we adopt a ranking criterion for the generation of the course recommendation list, denoted as the "most frequent item in the neighborhood" (MFIN), which has been shown to be very effective in terms of accuracy [2]. In particular, [2] have demonstrated that the strategy of ranking recommendable items based on their predicted rating value (HPR) to generate top-N recommendations can be considerably improved with the MFIN approach. The reason is that HPR is suitable only for predicting item ratings and not for recommending items, since many items can be predicted with the highest rating (i.e., 5 in a rating scale 1-5). This means that these items cannot be ranked since they have the same predicted rating value and thus, they are recommended with a random rank (by chance).

Based on MFIN, we count the frequency of each course inside the found target user's neighborhood, and recommend the N most frequent ones. That is, our algorithm finds, firstly, the target user's neighbors in the latent k space and then counts presences of courses in the user-course matrix based on the aforementioned neighbors' ratings. For the generation of the top-N course recommendation list, we count the frequency of each "positively" (e.g., > 3 in a rating scale [1-5]) rated course inside the found neighborhood of a target user,

Algorithm 1 xSVD++ Algorithm

Require: 2D sparse user-course rating matrix $R \in \mathbb{R}^{m \times n}$, 2D sparse user-skill matrix $US \in \mathbb{R}^{m \times s}$, 2D sparse course-skill matrix $CS \in \mathbb{R}^{n \times s}$, G objective function (in Equation 14), η_1 and η_2 learning rates, λ_1, λ_2 and λ_3 regularization parameters, K number of total latent-features, *steps* maximum number of algorithm's predictions, adjacency lists $uSkill$ and $cSkill$ and matrices $uSkillsIndex$,$uSkillsSum$, $cSkillsIndex$ and $cSkillsSum$.

Ensure: Complete prediction matrix $\widehat{R} \in \mathbb{R}^{m \times n}$

Initialize bu,bc, U,V,US^*, CS^*;
Calculate $\mu = mean(R)$;
repeat
  for $i = 1 : m$ **do**
    for $j = 1 : n$ **do**
      for $k = 1 : K$ **do**
        for $m = 1 : uSkillsIndex[i]/cSkillsIndex[j]$ **do**
          $SCUP[k] += \ US[i][uSkill[i][m]]US^*[uSkill[i][m]][k]$;
          $SCCP[k] += \ CS[j][cSkill[j][m]]CS^*[k][cSkill[j][m]]$;
        end for
        $SCUP[k] /= uSkillsSum[i]$;
        $SCCP[k] /= cSkillsSum[j]$;
      end for
      $e = R[i][j] - \mu - bu[i] - bc[j]$;
      for $k = 1 : K$ **do**
        $e -= (U[i][k] + SCUP[k])(V[k][j] + SCCP[k])$;
      end for
      $bu[i] += \eta_1(e - \lambda_1 bu[i])$; (**Equation 15**)
      $bc[j] += \eta_1(e - \lambda_1 bc[j])$; (**Equation 16**)
      for $k = 1 : K$ **do**
        $tempU[k] = U[i][k]$;
        $tempV[k] = V[k][j]$;
        $temp = \eta_2 \left[e(V[k][j] + SCCP[k]) - \lambda_2 U[i][k]\right]$; (**Equation 17**)
        $V[k][j] += \eta_2 \left[e(U[i][k] + SCUP[k]) - \lambda_2 V[k][j]\right]$; (**Equation 18**)
        $U[i][k] += temp$;
      end for
      for $m = 1 : uSkillsIndex[i]$ **do**
        for $k = 1 : K$ **do**
          $skill = uSkill[i][m]$;
          $US^*[skill][k] += \eta_2 US[i][skill] \cdot \cdot \left[\frac{e}{uSkillsSum[i]}(tempV[k] + SCCP[k]) - \lambda_3 US^*[skill][k]\right]$; (**Equation 19**)
        end for
      end for
      for $m = 1 : cSkillsIndex[j]$ **do**
        for $k = 1 : K$ **do**
          $skill = cSkill[j][m]$;
          $CS^*[k][skill] += \eta_2 CS[j][skill] \cdot \cdot \left[\frac{e}{cSkillsSum[j]}(tempU[k] + SCUP[k]) - \lambda_3 CS^*[k][skill]\right]$; (**Equation 20**)
        end for
      end for
      $\widehat{r}[i][j] = \mu + bu[i] + bc[j]$;
      for $k = 1 : K$ **do**
        Calculate $SCUP[k], SCCP[k]$;
        $\widehat{r}[i][j] += (U[i][k] + SCUP[k])(V[k][j] + SCCP[k])$;
      end for
    end for
  end for
until G (Equation 14) ceases to improve **OR** maximum algorithms predictions *steps* reached

and recommend the N most frequent ones. This means that "negatively" rated items do not count for the top-N course recommendation list formation.

Related work in Collaborative Filtering [19] has used either Pearson correlation or Cosine similarity to compute similarity among users. In our method, we will use the cosine similarity, because it favors the set of latent features, that are important in both users' profiles. Equation 21 measures the similarity between two users, u and v:

$$sim_{uv} = \frac{\sum_{\forall k} up_{uk} \quad up_{vk}}{\sqrt{\sum_{\forall k} {up_{uk}}^2} \sqrt{\sum_{\forall k} {up_{vk}}^2}} \tag{21}$$

As it is expressed by Equation 21, we apply cosine similarity in the user profile matrix UP. Please notice that the profile of a user i in the latent k space consists of two parts (i.e., matrices U and $SCUP$). Equation 22 depicts how we compute the user profile UP in the latent k space, whereas the $scup_{ik}$ represents the skills's contribution to the user profile:

$$up_{ik} = u_{ik} + scup_{ik} = u_{ik} + \frac{\sum_{s=1}^{S} us_{is} us^*_{sk}}{\sum_{s=1}^{S} us_{is}} \tag{22}$$

6 Experimental Evaluation

In this Section, we will perform experiments to test the prediction accuracy of our proposed method against four other algorithms, the classic MF algorithm (Equation 5), the Non-Negative Matrix Factorization (NMF) [11], the CMF [20] and SVD++ algorithms [8]. We will use two real data sets that have been used as benchmarks in prior work (MERLOT and MACE data sets). We perform all experiments with 5-fold cross validation, with a training-test split percentage, 80%-20%. In particular, we split the ratings of each tested user seperately. Afterwards, we train all algorithms according the data in the training set and check their performance based on the data in the test set. We use the RMSE and precision-recall metrics to test the accuracy performance of all algorithms.

6.1 Data Sets

The first data set derives from the Multimedia Educational Resource for Learning and Online Teaching[1] (MERLOT), a very popular US educational portal. It consists of 5,626 ratings using a scale from 0 to 5. In MERLOT dataset 1,829 users, have rated 2,987 learning objects. Registered users have additionally provided comments on learning objects. Moreover, the learning objects are graded by domain experts using a peer-review process. Peer reviews evaluate three dimensions: quality of the content, usability, and effectiveness as a learning tool. Each aspect is rated on a scale from 0 to 5. Moreover, peer-reviews contain 2626

[1] `http://merlot.org`

comments (concerns and strengths) for each learning object that come along with the three evaluated dimensions. Based on these comments, we build the learning object-characteristics matrix.

The second data set has been extracted from the MACE project [12]. The MACE project provided to students of architectural science, access to learning objects that were stored in different repositories all over Europe. Therefore, MACE enabled learners to search through and find learning objects that are appropriate for their context. This data set contains 117,907 actions made by 628 registered users on 12,369 learning objects, in the time period of October 2009 to October 2012. The 117,907 actions concern 5 different user choices. Firstly, a user can rate a learning object in a scoring scale from 1 to 5 (568 actions). Moreover, a student may have a look at the detail page of a learning object in the MACE portal (50,883 actions). Subsequently, a learner can leave the portal and visit the object's web page (9,061 actions). Finally, a user may insert a tag (56,854 actions) or a competence (523 actions) in the learning object's description. Please notice that for the purpose of our experiments, we assume that a learning object is closely related with these tags and competencies which are assigned to it by users.

6.2 Tuning of the k latent feature space

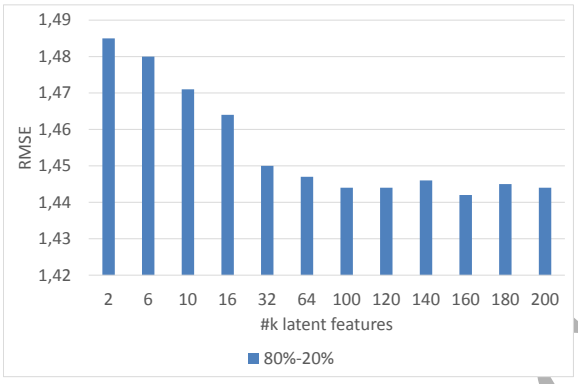
In this Section, we will examine how parameter k affects the effectiveness and performance of our algorithm. Thus, a small number k of latent features means that we keep both (U and V) matrices thin, and thus the required storage space small. To adjust parameter k, we keep the number of algorithm's predictions (iterations) fixed to 2,000. Moreover, we keep the learning rate η very small and equal to 0.001 to avoid missing a minimum.

For the MERLOT data set, as shown in Figure 5(a), as we increase the value of parameter k, RMSE decreases. The best RMSE value is attained when parameter k is equal to 160 latent features, and after this value the RMSE becomes worse.

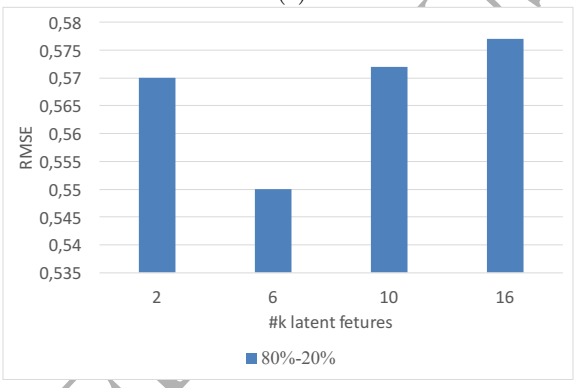
For the MACE data set, the tuning of k is quite different, than the one of the MERLOT data set. As it is illustrated in Figure 5(b), the best RMSE value appears when parameter k is quite small and equal to 6 latent features. This means that the small number of k latent features removes the noise from data in the MACE data set, resulting to lower RMSE values. For the next experiments, we keep fixed parameter k equal to 160 for the MERLOT data set. Moreover, we set the value of parameter k equal to 6 for the MACE data set.

6.3 Tuning of the *steps* Parameter

In this Section, we will test how parameter *steps*, i.e. the maximum number of algorithm's iterations, affects the effectiveness and performance of our algorithm. As shown in line 46 of Algorithm 1, our algorithm stops when a maximum number of *steps* (iterations) is reached. The algorithm's procedure is executed like this: On the first algorithm's iteration, we predict the missing ratings and we



(a)



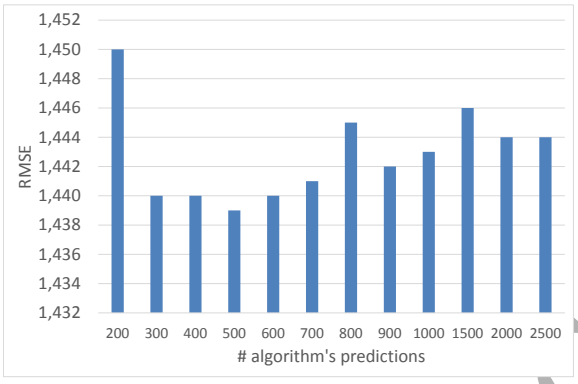
(b)

Fig. 5. RMSE vs. different number of k latent features for the (a) MERLOT and (b) MACE data sets (The lower values are better).

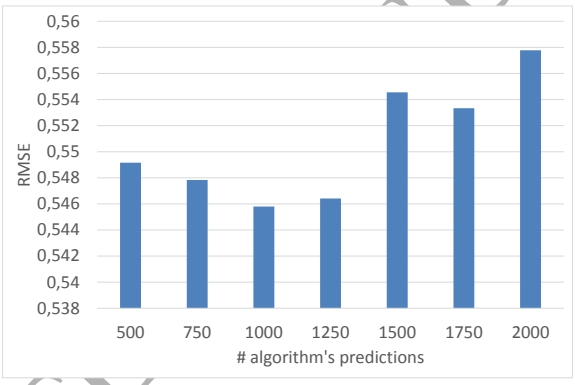
store them in the training set. On the second algorithm's iteration we re-predict the ratings using the information stored in the previous step and so on. For the appropriated tuning of the *steps* parameter, we keep fixed the learning rate η equal to 0.001.

For the MERLOT data set, as it is illustrated in Figure 6(a), the best RMSE reaches a minimum value of 1.439, when the maximum number of algorithm's predictions is equal to 500.

For the MACE data set, as it is depicted in Figure 6(b), the RMSE's minimum value of 0.546 is attained, when the maximum number of algorithm's iterations is equal to 1000. As we can see, the tuning of parameter *steps* resulted to an



(a)



(b)

Fig. 6. RMSE values vs. different number of algorithm's predictions (iterations) for the (a) MERLOT and (b) MACE data sets.

improvement of the RMSE value in both data sets. Henceforth, for the upcoming experiments we keep fixed the adjusted values for parameter *steps*, equal to 500 and 1000, for the MERLOT and the MACE data sets, respectively.

6.4 Tuning of λ Parameter

In this Section, we adjust parameter λ of Equation 5. Please notice that Equation 5 represents the classic MF algorithm, as it is described in Section 2. Param-

eter λ is responsible to control the magnitudes of user-latent and course-latent features of U and V matrices, respectively. As already mentioned, parameter λ is used to overcome the problem of overfitting.

For the MERLOT data set, RMSE reaches the minimum value of 1.435 when λ is equal to 0.1, resulting in a small improvement (0.28%) over the previous calculated value of 1.439, which was illustrated in Figure 6a, after the appropriate tuning of the *steps* parameter.

For the MACE data set, parameter λ does not help in decreasing the RMSE value. In particular, the usage of λ parameter of Equation 5, increases the error from 0.546 (Figure 6b of Section 6.3) to 0.549. This means that the usage of regularisation terms does not always contribute positively in the task of rating prediction.

6.5 Tuning of λ_1 Parameter

Parameter λ_1 of Equation 8 controls the magnitudes of *bc* and *bu* bias vectors for courses and users, respectively.

For the MERLOT data set, we keep fixed the value of parameter λ_2 equal to 0.1, because this value of λ (now λ_2), gave us the best RMSE in Section 6.4. Then, we tune parameter λ_1 from 0.01 to 0.3. The best RMSE value (1.275) is attained when we set λ_1 equal to 0.04.

For the MACE data set, we first have to find out, which value of parameter λ_2 we should start with. Thus, we conduct an experiment using a very small value for parameter λ_1. In fact, we set λ_1 equal to 0.05 and discover that the best value for λ_2 is 0.05. We then tune parameter λ_1 from 0.05 to 0.9, getting the best RMSE value of 0.491, when λ_1 is equal to 0.7. Consequently, we managed to decrease the RMSE value by 10.6%, compared to the previously computed RMSE value of 0.549.

6.6 Fusing the side information

In this Section, we fuse the user-skill and course-skill matrices into our objective function of Equation 14 and adjust parameter λ_3. As shown in Equation 14, parameter λ_3 controls the magnitude of the course-skill and user-skill contribution matrices (i.e., matrices CS^* and US^*).

For the MERLOT dataset, we use the course-skill matrix as side information. We conduct the next experiments by keeping fixed the previously computed values for λ_1 (0.04) and λ_2 (0.1). Figure 7 presents two algorithms (only ratings, and ratings combined with only course content). As shown, when we combine ratings with the course content, the best value for λ_3 is 0.2 and the RMSE value is equal to 1.261, which is a 1,1% improvement, compared to the RMSE result, that exploits only the ratings information (i.e. RMSE = 1.275).

For the MACE data set, we conduct the experiments by keeping fixed the previously computed values for λ_1 (0.7) and λ_2 (0.05). Figure 8 shows four different algorithms (only ratings, ratings combined with only course content,

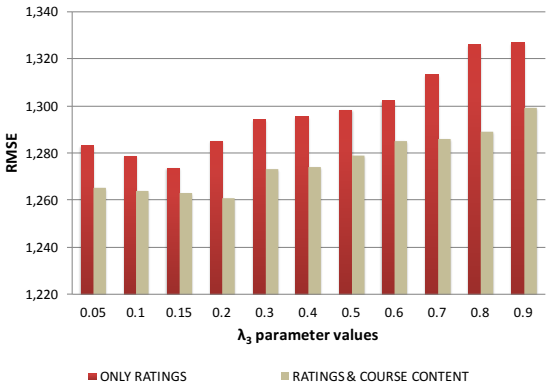


Fig. 7. Test set RMSE vs. different numbers of λ_3 on the MERLOT data set.

ratings combined with only user content, and ratings combined with all side information). As shown, when we exploit ratings combined with only the skills that users possess, the best value for λ_3 is 0.6 and the RMSE value is equal to 0.488, which is a tiny 0.2% improvement, compared to the RMSE result from the ratings combined with only the course-skills information. This result indicates that user-skill data are more significant than course-skill information on the MACE dataset.

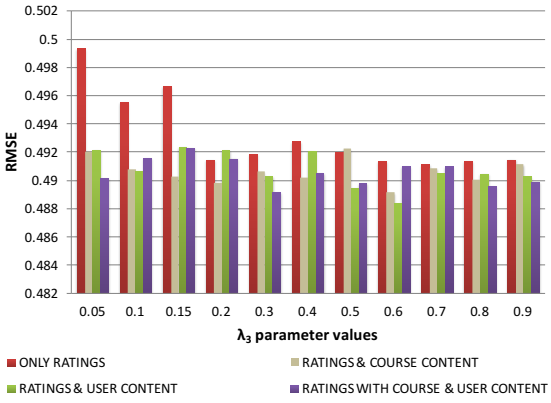


Fig. 8. Test set RMSE vs. different numbers of λ_3 on the MACE data set.

6.7 Comparison with other methods

In this Section, we compare our method, denoted as xSVD++, with CMF [20], SVD++ [8], the Non-Negative Matrix Factorization (NMF) [11], and classic MF (see Equation 5) algorithm on the MERLOT and the MACE data sets.

Firstly, we compare all algorithms in terms of RMSE. Table 1 reports the RMSE values of algorithms on MERLOT and MACE data sets. As shown, xSVD++ and CMF outperform the other four algorithms because they exploit more additional information, whereas SVD++ relies on the implicit side information that comes again from the same user-item rating matrix. Please also notice that the RMSE score is worst in the MERLOT dataset, because this dataset is more sparse and its rating scale is wider ([0-5]) than the one in the MACE data set ([1-5]).

Algorithm	**MERLOT**	**MACE**
xSVD++	1.261	0.488
CMF	1.285	0.501
SVD++	1.306	0.521
NMF	1.374	0.532
MF	1.435	0.546

Table 1. RMSE values for all algorithms on MACE data set. The smaller values are better.

Since the RMSE values in Table 1 are very close, we wanted to validate whether the xSVD++ approach outperforms others with statistical significance. Thus, we performed a statistical test, by running each experiment 30 times and computing the mean of difference μ_d between the results of the RMSE performance of each method and xSVD++. That is, for each pair between xSVD++ and a competitor (i.e., xSVD++ vs. CMF, xSVD++ vs. SVD++, xSVD++ vs. NMF, and xSVD++ vs. MF), we have run paired t-test with the null hypothesis H0($\mu_d = 0$), where μ_d is the mean of the difference between their RMSE values. We found that for all different pairs in both datasets, H0($\mu_d = 0$) is rejected at the 0.01 significance level, which shows that xSVD++ indeed significantly outperforms other methods.

Next, we compare all algorithms in terms of precision and recall. Figures 9(a) and 9(b) visualise the precision versus recall curve for the MERLOT and the MACE datasets, respectively. These experiments present the accuracy performance of the aforementioned algorithms, as we increase the number of top-N recommended items. As the number N of the recommended courses varies starting from the top-1 to top-N, we examine the precision and recall scores. Achieving high recall scores, while precision falls with the minimum decline indicates the robustness of the examined algorithm. As shown in Figure 9(a), for the MERLOT data set, xSVD++ algorithm outperforms all other algorithms in terms of

precision and recall for all different values (1, 2, 3, 4, 5) of top-N recommended courses.

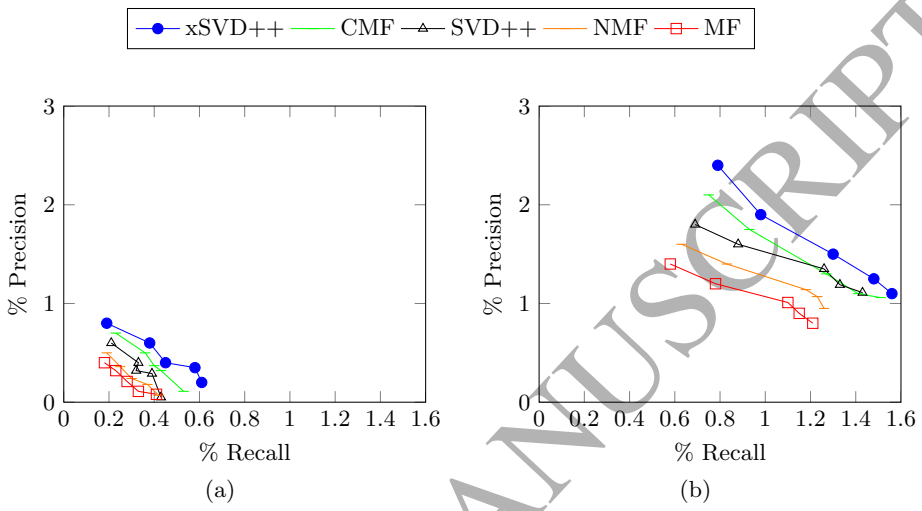


Fig. 9. Comparing xSVD++, CMF, SVD++, NMF, and MF performance in term of Precision and Recall at top-N recommended learning objects on (a) MERLOT dataset and (b) MACE dataset.

The same results can be observed in Figure 9(b), for the MACE data set, for all different values (1, 2, 3, 4, 5) of top-N recommended items. In summary, xSVD++ demonstrates the best results in both datasets. xSVD++ outperforms CMF because the second enforces the factorised matrices to share several latent factors, which may affect the optimization procedure. Please also notice that the recommendation accuracy in both data sets is very small. In particular, the average precision of all algorithms is less than 1% in MERLOT and less than 3% in MACE. The reason is that the data sets are very sparse and there are not enough ratings to build a model that recommends learning objects with high accuracy. For example, the average ratings per user in MERLOT is 3.01 ratings and in MACE is 18.77. To further prove this assumption, in another experiment we consider only those users who had expressed more than 6 ratings in MERLOT and more than 35 ratings in MACE (i.e. have twice more ratings than the average). When we considered only these users (i.e. the users who rated many learning objects) in our evaluation, then the average precision of all algorithms was increased to 1.8% in MERLOT and 4.6% in MACE, respectively.

7 Discussion

Our method extends SVD++ to consider also information about the user skills and course features. Thus, when we do not have enough information about the courses that a user likes, then we can use additional information based on the skills that this user has or based on the features of the courses that he has interacted with. Moreover, matrix factorisation brings into surface the main trends of users' behaviour in terms of the courses they like or attend. Thus, when a user is new to the system and there is not enough data available about him, then we can incorporate information from the public trends (e.g., what are the main trends that users follow and recommend to him those trends). However, in case that a user does not follow a specific trend, then we have to find similarities of his preferences only with those users that have similar behaviour. Then, we have to measure the cosine similarity of the target user's preferences vector with those vectors of other users in the reduced dimensional space and recommend those courses to him that this neighbourhood of similar users have liked in the past.

Another extension of our method could be to fuse into our objective function, also information from the social network of users. Thus, we can fuse into the objective function additional information from the user-user friendship network. To do this, we can add a new constraint into the objective function, which takes into account the friendship among users [4, 24]. The information of the friendship network of users is kept by an adjacency matrix, which is square and symmetric. The symmetry of this adjacency matrix is obvious because if a user U_a is friend with user U_b then user U_b is friend with user U_a, which means that friendship is reciprocal. By adding the constraint of friendship into our objective function, we practically influence the prediction of item's rating, since we ensure that the taste of a user is close to that of all his friends. Of course, by exploiting information of the friends of the users does not create privacy issues, since users explicitly accept to share their information with their friends. However, in cases that we would try to use the information of other similar users (not friends) of the target user based on his common implicit interaction with them (i.e., co-commenting on posts, co-rating items, etc.), we may raise privacy issues. Thus, the exploitation of this implicit interactions should be avoided if the users have not accepted to share their account information with others.

8 Conclusions

In this paper, we extend the well-known SVD++ algorithm, for performing multi-modal Matrix Factorization. Thus, SVD++ becomes just a simplified special case of our xSVD++, which exploits information from external resources (i.e., users' skills, courses' characteristics, etc.) to perform rating predictions. The experimental results have showed an improvement over classic and non-negative matrix factorization algorithms and the state-of-the-art CMF, and SVD++ algorithms. In our future work, we will consider the issue of fusing into our objective

function, also information from the business network of users (i.e., Linkedin, Xing, etc.).

Bibliography

[1] Bendakir, N., Aïmeur, E.: Using association rules for course recommendation. In: Proceedings of the AAAI Workshop on Educational Data Mining. vol. 3 (2006)

[2] Coba, L., Symeonidis, P., Zanker, M.: Replicating and improving top-n recommendations in open source packages. In: Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics. ACM (2018)

[3] Elbadrawy, A., Karypis, G.: Domain-aware grade prediction and top-n course recommendation. In: Proceedings of the 10th ACM conference on Recommender systems (2016)

[4] Forsati, R., Mahdavi, M., Shamsfard, M., Sarwat, M.: Matrix factorization with explicit trust and distrust side information for improved social recommendation. ACM Trans. Inf. Syst. 32(4), 17:1–17:38 (2014)

[5] Furnas, G., Deerwester, S., Dumais, S.: Information retrieval using a singular value decomposition model of latent semantic structure. In: Proc. ACM SIGIR Conf. pp. 465–480 (1988)

[6] Guo, W., Wu, S., Wang, L., Tan, T.: Personalized ranking with pairwise factorization machines. Neurocomputing 214(Supplement C), 191–200 (2016)

[7] Kabbur, S., Ning, X., Karypis, G.: Fism: Factored item similarity models for top-n recommender systems. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 659–667. KDD '13, ACM, New York, NY, USA (2013)

[8] Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 426–434. ACM (2008)

[9] Koren, Y., Bell, R.: Advances in Collaborative Filtering. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 145–186. Springer (2011)

[10] Lee, D.D., Seung, H.S.: Learning the parts of objects by nonnegative matrix factorization. Nature 401, 788–791 (1999)

[11] Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: NIPS. pp. 556–562 (2000)

[12] Niemann, K., Wolpers, M.: Real world object based access to architecture learning material–the mace experience. In: Proceedings of the World Conference on Educational Multimedia and Hypermedia (EDMEDIA 2010), Toronto, Canada (June 2010) (2010)

[13] Ning, X., Karypis, G.: Slim: Sparse linear methods for top-n recommender systems. In: Proceedings of the 11th International Conference on Data Mining. pp. 497–506. IEEE (2011)

[14] Ning, X., Karypis, G.: Sparse linear methods with side information for top-n recommendations. In: Proceedings of the sixth ACM conference on Recommender systems. pp. 155–162. ACM (2012)
[15] Parameswaran, A., Venetis, P., Garcia-Molina, H.: Recommendation systems with complex constraints: A course recommendation perspective. ACM Transactions on Information Systems (TOIS) 29(4), 20 (2011)
[16] Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD cup and workshop. pp. 5–8 (2007)
[17] Rendle, S.: Factorization machines. In: Proceedings of the 10th IEEE International Conference on Data Mining. IEEE Computer Society (2010)
[18] Rendle, S., Freudenthaler, C., Gantner, Z., Lars, S.T.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. pp. 452–461. UAI '09 (2009)
[19] Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proc. WWW Conf. pp. 285–295 (2001)
[20] Singh, A.P., Gordon, G.J.: Relational learning via collective matrix factorization. In: Li, Y., Liu, B., Sarawagi, S. (eds.) KDD. pp. 650–658. ACM (2008)
[21] Symeonidis, P., Zioupos, A.: Matrix and Tensor Factorization Techniques for Recommender Systems. SpringerBriefs in Computer Science, Springer (2016)
[22] Yang, D., Adamson, D., Rosé, C.P.: Question recommendation with constraints for massive open online courses. In: Proceedings of the 8th ACM Conference on Recommender systems. pp. 49–56. ACM (2014)
[23] Yang, D., Piergallini, M., Howley, I., Rose, C.: Forum thread recommendation for massive open online courses. In: Proceedings of 7th International Conference on Educational Data Mining (2014)
[24] Yuan, Q., Chen, L., Zhao, S.: Factorization vs. regularization: fusing heterogeneous social relationships in top-n recommendation. In: Proceedings of the fifth ACM conference on Recommender systems. pp. 245–252. ACM (2011)
[25] Zheng, V., Cao, B., Zheng, Y., Xie, X., Q., Y.: Collaborative filtering meets mobile recommendation: A user-centered approach. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI). Atlanta, GA (2010)
[26] Zheng, V., Zheng, Y., Xie, X., Yang, Q.: Collaborative location and activity recommendations with gps history data. In: Proceedings of the 19th International Conference on World Wide Web (WWW). pp. 1029–1038. New York, NY (2010)