Interpretable recommendations via overlapping co-clusters

Reinhard Heckel and Michail Vlachos

Abstract—There is an increasing need to provide not only accurate but also interpretable recommendations, in order to enhance transparency and trust in the recommendation process. This is particularly important in a business-to-business setting, where recommendations are generated for experienced sales staff and not directly for the end-client. In this paper, we consider the problem of generating interpretable recommendations based on the purchase history of clients, or more general, based on positive or one-class ratings only. We present an algorithm that generates recommendations by identifying overlapping co-clusters consisting of clients and products. Our algorithm uses matrix factorization techniques to identify co-clusters, and recommends a client-product pair because of its membership in one or more client-product co-clusters. The algorithm exhibits linear complexity in the number of co-clusters and input examples, and can therefore be applied to very large datasets. We show, both on a real client-product dataset from our institution, as well as on publicly available datasets, that the recommendation accuracy of our algorithm is better than or equivalent to standard interpretable and non-interpretable recommendation techniques, such as standard one-class nearest neighbor and matrix factorization techniques. Most importantly, our approach is capable of offering textually and visually interpretable recommendations.

1 INTRODUCTION

In this paper, we consider the problem of providing interpretable client-product recommendations based on the purchase history of clients. The particular application we are interested in is a business-to-business (B2B) recommender system, in which the recipient of the recommendation is a salesperson responsible for the client, and not the client itself. A salesperson is typically responsible for tens or hundreds of clients and to decide whether to pursue a sales opportunity (i.e., recommendation), he or she relies on evaluating the rationale of the recommendation. It is therefore particularly important to produce recommendations based on interpretable models, rather than on models that merely highlight the propensity of interest. Interpretability in general adds trust, confidence [11] and persuasiveness [47] to the recommendation, and in certain scenarios helps highlight tradeoffs and compromises [24] and even incorporates aspects of social endorsement [41]. Therefore, interpretable recommendations are of interest in many application areas.

We base our recommendations on positive ratings only. In a business recommender system, positive ratings correspond to the purchase history of the clients, that is, the sets of products the clients have purchased. In this scenario, negative ratings are typically not available, as not purchasing a product does not reflect a negative rating. The problem of generating recommendations based on positive ratings of users only is known as One-Class Collaborative Filtering (OCCF). The lack of negative feedback makes this problem challenging, as one has to learn the customer preferences from what they like, without getting to see what the customers dislike. This problem also occurs in other important collaborative filtering settings, in particular when only implicit ratings are available. For example, a provider of a news website faces the problem of generating article recommendations based on the readers’ historical article views. In such a setup, the readers do not provide explicit negative feedback. Equivalent OCCF tasks can be formulated from the browsing history of a visitor at an e-commerce website or when viewing videos on a video-sharing platform.

Our approach to providing interpretable recommendations from positive examples is based on the detection of co-clusters of clients that buy sets or bundles of products. The intuition behind this approach is that there exist clusters, groups, or communities of users that are interested in a subset of the items. These groups are called co-clusters because they consist of both users and items with similar patterns. Users can have several interests, and items might satisfy several needs, so users and items may belong to several co-clusters. This means that co-clusters may be overlapping. Our motivation for driving recommendations based on finding overlapping user-item co-clusters is that such an approach offers an interpretable model: Identification of sets of users that are interested in or have bought a set of items not only allows us to infer latent underlying patterns but can also lead to better, textually and visually interpretable recommendations.

Standard state-of-the-art recommendation approaches based on matrix factorization are in general not applicable here, because of a lack of interpretability, as “latent factors obtained with mathematical methods applied to the user-item matrix can be hardly interpreted by humans” [32]. This observation has been made in several works [2], [33], [48].

Co-clustering principles have been considered before in the context of recommender systems [40], [18], but past work considered non-overlapping co-clusters only, where every item/user belongs to only one co-cluster. In fact, in the traditional co-clustering setting, co-clusters are also non-
overlapping [8]. In contrast, in this work we use overlapping co-clusters to generate interpretable recommendations. Allowing co-clusters to overlap is not only crucial for the recommendation performance, but also enables the discovery of interesting and useful buying patterns. In Figure 1, we provide an example of overlapping co-clusters. A black dot describes a product bought by the user in the past. One can visually identify three potential recommendations indicated by white squares inside the co-clusters. The approach we propose identifies overlapping user-item co-clusters and generates recommendations based on a generative model. The algorithm is scalable, on par in terms of recommendation accuracy with state-of-art OCCF methodologies, and at the same time provides interpretable recommendations.

Outline: The paper is organized as follows. We review related work in Section 2, and present our Overlapping Co-Cluster Recommendation algorithm, henceforth referred to as “OCuLaR”, in Section 4. We will first introduce a generative model, based on which we design a computationally efficient algorithm to produce recommendations and to identify the co-clusters. In Section 4.5, we explain the relation of traditional matrix factorization approaches to our algorithm, and in Section 6 we discuss how to add extra information into OCuLaR. Section 7 compares OCuLaR to other state-of-the-art recommendation algorithms on a real-world client-product dataset from our institution and publicly available datasets. We show that our algorithm performs as well as or better than state-of-the-art matrix factorization techniques, with the added benefit of interpretability. Finally, in Section 8, we present and discuss a deployment of our algorithm in a B2B recommender system at our institution.

2 RELATED WORK

Collaborative Filtering (CF): Early approaches to recommender systems performed either user-based or item-based collaborative filtering. User-based techniques infer preferences of a given user based on the preferences of similar or like-minded users by, e.g., recommending products that the nearest neighbors of a user have bought in the past. Similarly, item-based techniques exploit item-to-item similarities to generate recommendations. Item- and user-based techniques yield a reasoning of the sort “similar users have also bought”, but are often outperformed by latent factor models, such as matrix factorization approaches [19]. Matrix factorization techniques in their traditional form predict ratings or preferences well, but the latent features make it difficult to explain a recommendation [2], [33], [32]. Recently, [48] investigates a method for explainable factorization by extracting explicit factors (sentiment, keywords, etc) from user reviews; such an approach is applicable only in the presence of additional textual information for each recommendation. Such data, however, are rarely available in a B2B setting. Generic methodologies for communicating explanations to the user have been studied in [38], [11], [12].

Conceptually, the algorithm proposed in this work can be regarded as a combination of item-based and user-based approaches. Specifically, we discover similarities that span across both the user and the item space by identifying user-item co-clusters. By identifying user-item co-clusters via matrix factorization techniques, we obtain an estimate of the probability that an unknown example is positive, which automatically results in ranked recommendations.

OCCF models are used to predict preferences from implicit feedback. This is a typical recommendation scenario when browsing, buying, or viewing history is available, or in general in setups where the user does not provide any explicit negative feedback. Recommendation techniques based on positive ratings try to learn by considering the implicit feedback either as absolute preferences [27], [16], or as relative preferences [31], [28]. We compare with techniques from both categories in Section 7, and state the formal relation of our approach to matrix factorization techniques in Section 4.5.

Co-clustering: The majority of the literature on co-clustering considers the detection of non-overlapping co-clusters [10], [29]. Notable exceptions are the approach in [3], which introduces a generative model for overlapping co-clusters along with a generative alternating minimization algorithm for fitting the corresponding model to given data, and the approach in [36], which considers the problem of simultaneously clustering documents and terms. That latter approach is based on a generative model and estimation of the corresponding model parameters [36]. In both [3], [36], the focus is on discovering co-clusters, whereas here we are interested in explaining the recommendations produced by our algorithm with co-clusters. Although co-clustering approaches were used before in the collaborative filtering setting, the majority of those papers [13], [18], [40] is restricted to non-overlapping co-clusters. An exception is [44], which explores the multiclass co-clustering problem in the context of CF.

Community detection: Related to co-clustering is community detection. To see this, observe that the positive examples are the edges in a bipartite graph of users and items (see Figure 1). Then, the co-clusters of users and items correspond to user-item communities in the bipartite graph. With the advent of social networking tools, interest in detecting communities of users that share the same interests has increased. Traditionally, community detection was formulated as a graph-partitioning problem (e.g., mincut); therefore the corresponding approaches in general do not tackle the case of overlapping communities. One of the
best-known community detection algorithms is based on the notion of modularity [25]. Specifically, the modularity algorithm by Girvan & Newman [14] is one of the most widely-used community detection algorithms and is used in many software packages (gephi, Mathematica, etc). It has the advantage that it can automatically discover the number of communities; however it does not support discovery of overlapping communities. Other work on detecting non-overlapping communities includes [17], [9].

Recently, interest in identification of overlapping communities has grown [26], [1], [21], [45], [30]. Related to our work is the BIGCLAM algorithm proposed by Yang and Leskovec [45]; main differences are that in our approach we consider a particular bi-partite graph and use regularization, which turns out to be crucial for recommendation performance. In general, off-the-shelf community detection methodologies are not directly applicable to the one class collaborative filtering problem, as they yield an assignment of users/items to communities, but not a ranked list of recommendations.

3 Formal problem statement

We are given a matrix $R$ where the rows correspond e.g., to users or clients and the columns to items or products. If the $(u,i)$th element of $R$ takes on the value $r_{ui} = 1$ this indicates that user $u$ has purchased item $i$ in the past or, more generally, that user $u$ is interested in item $i$. We assume that all values $r_{ui}$ that are not positive ($r_{ui} = 0$) are unknown ($r_{ui} = 0$) in the sense that user $u$ might be interested in $i$ or not. Our goal is to identify those items a user $u$ is likely to be interested in. Put differently, we want to find the positives among the unknowns, given only positive examples.

4 OCuLaR algorithm

In this section, we present our Overlapping co-CLuster Recommendation algorithm, or for short “OCuLaR”. We assume an underlying model whose parameters are factors associated with the users and items. Those factors are learned, such that the fitted model explains well the given positive examples $r_{ui} = 1$. The specific choice of our model has several advantages, e.g., it allows us to design an efficient algorithm to learn the factors. Moreover, the factors encode co-cluster membership and affiliation strength, and are interpretable in that sense.

4.1 Generative model

We start with the generative model underlying our recommendation approach. The generative model is similar to the BIGCLAM model [45] used for community detection. It formalizes the following intuition: There exist clusters, groups, or communities of users that are interested in a subset of the items. As users can have several interests, and items might satisfy several needs, each user and item can belong to several co-clusters consisting of users and items. However, a co-cluster must contain at least one user and one item, and can therefore not consist of users or items alone.

Suppose there are $K$ co-clusters ($K$ can be determined from the data, e.g., by cross-validation, as discussed later). Affiliation of a user $u$ and item $i$ with a co-cluster is modeled by the $K$-dimensional co-cluster affiliation vectors $f_u$ and $f_i$, respectively. The entries of $f_u$, $f_i$ are constrained to be non-negative, and $[f_u]_c = 0$ signifies that user $u$ does not belong to co-cluster $c$. Here, $[f]_c$ denotes the $c$-th entry of $f$. The absolute value of $[f_u]_c$ corresponds to the affiliation strength of $u$ with co-cluster $c$; the larger it is, the stronger the affiliation.

Positive examples are explained by the co-clusters as follows. If user $u$ and item $i$ both lie in co-cluster $c$, then this co-cluster generates a positive example with probability

$$1 - e^{-[f_u]_c [f_i]_c}.$$ 

Assuming that each co-cluster $c = 1, ..., K$, generates a positive example independently, it follows that

$$1 - P[r_{ui} = 1] = \prod_c e^{-[f_u]_c [f_i]_c} = e^{-(f_u, f_i)},$$

where $(f, g) = \sum_c [f]_c [g]_c$ denotes the inner product in $\mathbb{R}^K$. Thus

$$P[r_{ui} = 1] = 1 - e^{-(f_u, f_i)}.$$  

(1)

Note that so far the model cannot explain a positive example by means other than co-cluster affiliation. Although we could incorporate user, item, and overall bias by supposing that the probability of an example being positive is given by

$$P[r_{ui} = 1] = 1 - e^{-(f_u, f_i) - b_u - b_i - b},$$

we found that fitting the corresponding model does not increase the recommendation performance for the datasets considered in Section 7, and therefore will not discuss it further.

4.2 Fitting the model parameters

Given a matrix $R$, we fit the model parameters by finding the most likely factors $f_u, f_i$ to the matrix $R$ by maximizing the likelihood (recall that we assume positive examples to be generated independently across co-clusters and across items and users in co-clusters):

$$L = \prod_{(u,i)} (1 - e^{-(f_u, f_i)}) \prod_{(u,i) : r_{ui} = 0} e^{-(f_u, f_i)}.$$ 

Maximizing the likelihood is equivalent to minimizing the negative log-likelihood:

$$-\log L = - \sum_{(u,i) : r_{ui} = 1} \log (1 - e^{-(f_u, f_i)}) + \sum_{(u,i) : r_{ui} = 0} (f_u, f_i).$$

(2)

To prevent overfitting, we add an $\ell_2$ penalty, which results in the following optimization problem:

$$\text{minimize } Q \text{ subject to } [f_u]_c, [f_i]_c \geq 0, \text{ for all } c,$$

(3)

where

$$Q = -\log L + \lambda \sum_i \|f_i\|_2^2 + \lambda \sum_u \|f_u\|_2^2$$

(4)

and $\lambda \geq 0$ is a regularization parameter. As will we discuss in more detail in Section 4.5, this optimization problem can be viewed as a variant of non-negative matrix factorization (NMF), specifically NMF with a certain cost function.
A common approach to solve an NMF problem is alternating least squares, which iterates between fixing $f_u$, and minimizing with respect to $f_i$, and fixing $f_i$ and minimizing with respect to $f_u$, until convergence. This strategy is known as cyclic block coordinate descent or the non-linear Gauss-Seidel method. Whereas $Q$ is non-convex in $f_i,f_u$, $Q$ is convex in $f_i$ (with $f_u$ fixed) and convex in $f_u$ (with $f_i$ fixed). Therefore, a solution to the subproblems of minimizing $Q$ with fixed $f_i$ and minimizing $Q$ with fixed $f_u$ can be found, e.g., via gradient descent or Newton’s method.

As this optimization problem is non-convex, one cannot in general guarantee convergence to a global minimum; however convergence to a stationary point can be ensured. Specifically, provided that $\lambda > 0$, $Q$ is strongly convex in $f_i$ (with $f_u$ fixed) and in $f_u$ (with $f_i$ fixed). Thus, the subproblems have unique solutions, and therefore, if we solve each subproblem exactly, convergence to a stationary point is ensured by [4, Prop. 2.7.1].

However, as noted in the context of matrix factorization [15], solving the subproblems exactly may slow down convergence. Specifically, when $f_u,f_i$ are far from a stationary point, it is intuitive that there is little reason to allocate computational resources to solve the subproblems exactly. It is therefore often more efficient to solve the subproblem only approximately in each iteration [5], [15].

For the above reasons, we will only approximately solve each subproblem by using a single step of projected gradient descent with backtracking line search, and iteratively update $f_i$ and $f_u$ by single projected gradient descent steps until convergence. In Section 4.4, we provide further implementation details. Convergence is declared if $Q$ stops decreasing. This results in a very efficient algorithm that is essentially linear in the number of positive examples $\{(u,i) : r_{ui} = 1\}$, and the number of co-clusters $K$. Our simulations have shown that performing only one gradient descent step significantly speeds up the algorithm.

**Choice of $K$:** Recall that the number of co-clusters $K$ is a model parameter. $K$ can be determined from the data via cross-validation. Specifically, to determine a suitable $K$, we train a model on a subset of the given data for different choices of $K$, and select the $K$ for which the corresponding model performs best on the test set. The model can thereby be tuned for an appropriate metric. In our experiments, we measure the recommendation performance in terms of the recall-at-M items [23], which is a standard performance metric in recommender system settings.

### 4.3 Generating interpretable recommendations

Suppose we want to recommend $M$ items to each user. After having fitted the model parameters, we recommend item $i$ to user $u$ if $r_{ui}$ is among the $M$ largest values $P[r_{ui} = 1]$, where $i'$ is over all items that user $u$ did not purchase, i.e., over all $i'$ with $r_{ui'} = 0$. The probability $P[r_{ui} = 1]$ is large if the user-item pair $(u,i)$ is in one or more user-item co-clusters. Thus, along with a recommendation, we can output the corresponding user-item co-clusters that cause $P[r_{ui'} = 1]$ or, equivalently, $(f_u,f_i) = \sum_{c} [f_u]_c [f_i]_c$ to be large. The user-item co-cluster $c$ is determined as the subset of users and items for which $[f_u]_c$ and $[f_i]_c$, respectively, are large.

In the B2B setting that we consider it is also important to explicitly mention who are the clients that have purchased a similar bundle of products. Contrary to the B2C setting, where, for privacy reasons, one only mentions what similar clients purchase, in a B2B setting this is not a concern. The salesperson, who is the recipient of the recommendation, can use this information (explicit names of similar clients) to understand better the types of clients/companies that typically require such a solution. Our approach directly provides this information, because each co-cluster consists of specific clients (users) and products, and does not merely describe an average behavior.

**Example:** Here we provide a cogent example about the interpretability of the recommendations by OCuLaR. We will use the user-item array from Figure 1. Let us consider making one recommendation to User 6, where Users 0–11 correspond to the rows, and Items 0–11 to the columns of the matrix. The probabilities of the fitted model for each user-item pair are depicted in Figure 2. The probability estimate $P[r_{ui} = 1] = 1 - e^{-\langle r_{ui}, f_i \rangle}$, for $u = 6$ is maximized among the unknown examples ($r_{ui} = 0$) for Item $i = 4$, and is given by 0.83. Therefore, OCuLaR recommends Item $i = 4$ to User $u = 6$. The corresponding factors are $f_i = [1.39, 0.73, 0.82]$ and $f_u = [0, 1.05, 1.25]$, which means that Item $i = 4$ is in all three co-clusters, while User $u = 6$ is in co-clusters 2 and 3 only. The probability estimate $1 - e^{-\langle r_{ui}, f_i \rangle}$ for $u = 6, i = 4$ is large because both User 6 and Item 4 are in co-clusters 2 and 3. To justify recommending Item 4 to User 6, we can therefore give the following, automatic interpretation to the user of the recommender system:

*Item 4 is recommended to Client 6 with confidence 0.83 because:*

- **Client 6 has purchased Items 1-3. Clients with similar purchase history (e.g., Clients 4-5) also bought Item 4.**
- **Moreover, Client 6 has purchased Items 5-9. Clients with similar purchase history (e.g., Clients 7-9) also bought Item 4.**
Naturally, once the co-clusters have been discovered, additional information (derived from the co-clusters) can be attached in the rationale presented. In Section 6, we also discuss how additional information can be incorporated directly into the OCUaLar model. Finally, in the experimental Section 7, we discuss the recommendation rationale for an industrial deployment of OCUaLar (see Figure 6).

4.4 Implementation and complexity

Here we examine in more detail the proposed gradient descent approach we use to solve the subproblems and the complexity of the overall optimization algorithm. It is sufficient to discuss minimization of $Q$ with respect to $f_i$, as minimization with respect to $f_u$ is equivalent. The following approach for minimizing $Q$ was also used in similar form in [45], [22]. We start by noting that, because of

$$Q = \sum_i \left( -\sum_{u: r_{ui}=1} \log(1 - e^{-\langle f_u, f_i \rangle}) + \sum_{u: r_{ui}=0} \langle f_u, f_i \rangle \right) + \lambda \sum_u \|f_u\|_2^2 + \lambda \sum_i \|f_i\|_2^2,$$

we can minimize $Q$ for each $f_i$ individually. The part of $Q$ depending on $f_i$ is given by

$$Q(f_i) = -\sum_{u: r_{ui}=1} \log(1 - e^{-\langle f_u, f_i \rangle}) + \sum_{u: r_{ui}=0} \langle f_u, f_i \rangle + \lambda \|f_i\|_2^2.$$  (5)

As mentioned, we update the parameter $f_i$ by performing a projected gradient descent step. The projected gradient descent algorithm [4, Sec. 2.3] is initialized with a feasible initial factor $f_i^0$ and updates the current solution $f_i^k$ to $f_i^{k+1}$ according to

$$f_i^{k+1} = (f_i^k - \alpha_k \nabla Q(f_i^k))_+,$$

where $(f)_+$ projects $f$ on its positive part, $\max(0, \langle f \rangle)$, and the gradient is given by

$$\nabla Q(f_i) = -\sum_{u: r_{ui}=1} f_u e^{-\langle f_u, f_i \rangle} + \sum_{u: r_{ui}=0} f_u + 2\lambda f_i.$$  (6)

The step size $\alpha_k$ is selected using a backtracking line search, also referred to as the Armijo rule along the projection arc [4]. Specifically, $\alpha_k = \beta^k t_k$, where $t_k$ is the smallest positive integer such that

$$Q(f_i^{k+1}) - Q(f_i^k) \leq \sigma \langle \nabla Q(f_i^k), f_i^{k+1} - f_i^k \rangle$$

where $\sigma, \beta \in (0, 1)$ are user-set constants. As the computation of both $\nabla Q(f_i)$ and $Q(f_i)$ requires $\sum_{u: r_{ui}=0} f_u$, and typically, the number of items for which $r_{ui}=1$ is small relative to the total number of items, we precompute $\sum_u f_u$ before updating all $f_i$, and then compute $\sum_{u: r_{ui}=0} f_u$ via

$$\sum_{u: r_{ui}=0} f_u = \sum_u f_u - \sum_{u: r_{ui}=1} f_u.$$  

Using the precomputed $\sum_{u: r_{ui}=0} f_u$, a gradient descent step of updating $f_i$ requires $O(|u: r_{ui}=1| |K|)$, and the number of items of which $r_{ui}=1$ is small relative to the total number of items, we precompute $\sum_u f_u$ before updating all $f_i$, and then compute $\sum_{u: r_{ui}=0} f_u$ via

$$\sum_{u: r_{ui}=0} f_u = \sum_u f_u - \sum_{u: r_{ui}=1} f_u.$$  

4.5 Relation to Matrix Factorization

Latent factor models are a prevalent approach to recommend systems. Among them, matrix factorization approaches are particularly popular, because of their good performance properties. In this section, we discuss the connection of our approach to standard matrix factorization techniques. Matrix factorization models in their most basic form represent users and items by latent vectors $f_u, f_i$ in a low-dimensional space $\mathbb{R}^K$, and form a rating according to

$$r_{ui} = \langle f_u, f_i \rangle.$$  

A common approach to fit the latent factors based on a given set of examples $\{r_{ui}\}$ is to minimize

$$\min \sum_{u,i} \ell(r_{ui}, \langle f_u, f_i \rangle) + \lambda \|f_u\|_2^2 + \lambda \|f_i\|_2^2,$$  (7)

where $\ell$ is a loss function and $\|f_u\|_2^2, \|f_i\|_2^2$ are regularization terms to prevent overfitting. A common choice for non-binary ratings is the quadratic loss function $\ell(r_{ui}, \langle f_u, f_i \rangle) = (r_{ui} - \langle f_u, f_i \rangle)^2$. For the one-class collaborative filtering problem, quadratic loss is not directly applicable, as the question remains on how to deal with the unknowns, for which $r_{ui} = 0$. Performing summation in (7) only over the positive examples ($r_{ui} = 1$) is not sensible, as it results in a trivial solution of $f_u = f_i, \forall u$ and all $i$ to (7). A different approach is to treat the unknowns as negative ratings. However, this might cause recommendations as some of the unknown examples are actually positive ratings. To resolve this issue, it has been proposed in [27] to give different weights to the error terms corresponding to positive and unknown ratings in the objective function, specifically to use the cost function

$$\ell(r_{ui}, \langle f_u, f_i \rangle) = \begin{cases} \frac{(r_{ui} - \langle f_u, f_i \rangle)^2}{b(r_{ui} - \langle f_u, f_i \rangle)^2} & \text{if } r_{ui} = 1, \\ \frac{(r_{ui} - \langle f_u, f_i \rangle)^2}{b(r_{ui} - \langle f_u, f_i \rangle)^2} & \text{if } r_{ui} = 0, \end{cases}$$  (8)

where $b < 1$ is a weight assigned to unknown ratings. A common approach to solve the corresponding optimization problem is weighted alternating least squares (wALS) [27].

Our approach is equivalent to choosing the loss function

$$\ell(r_{ui}, \langle f_u, f_i \rangle) = -\log(r_{ui} - \langle f_u, f_i \rangle).$$  (9)

This results in a large penalty if $\langle f_u, f_i \rangle$ is small for a positive example $r_{ui} = 1$, and a moderate penalty if $\langle f_u, f_i \rangle$ is large for an unknown example $r_{ui} = 0$. Our approach is therefore similar in spirit to that of giving different weights to positive ($r_{ui} = 1$) and unknown ($r_{ui} = 0$) examples.

Interpretability: Matrix factorization approaches, such as the wALS algorithm (with loss function (8)), yield good empirical performance, as we will see in the experiments. Their main disadvantage for our scenario is that the latent space is typically not easy to interpret. This drawback of MF techniques is attested in various studies [20], [2], [33], [39]. This statement also applies to standard non-negative matrix factorization (NMF) techniques, where the factors are constrained to be non-negative. Our approach also uses factorization principles, however, we confine the factors to explicitly model user and item participation, which is key for not compromising the model interpretability.
5 RELATIVE OCuLaR ALGORITHM (R-OCuLaR)

As we saw in the related work section, OCCF problems can be viewed as learning of either absolute or relative preferences of users. The OCuLaR algorithm belongs to the first category because it tries to learn the absolute ratings of the users. Specifically, as we saw, OCuLaR can be viewed as a non-negative matrix factorization approach with a particular loss function (i.e., (9)), with the loss function assigning a large penalty to positive examples that are not well explained by the model (i.e., $\langle f_u, f_i \rangle$ small for $r_{ui} = 1$), and only a moderate penalty to unknown examples that are not well explained by the model (i.e., $\langle f_u, f_i \rangle$ large for $r_{ui} = 0$).

In this section, we examine how OCuLaR could be adapted to treat the positive examples as relative preferences. The notion of relative preferences in OCCF was first explored by Rendle et al. [31], who proposed to predict the relative preferences (i.e., $r_{ui}$) such that $r_{ui} = 1$ means user $u$ prefers item $i$ over item $j$, rather than the absolute rankings. The underlying idea for this construction is that there is an associated latent personalized ranking $\langle u, i, j \rangle$ for each user $u$, where $i >_u j$ signifies that user $u$ prefers item $i$ over item $j$. To this end, in a first step, one can construct a “training” item ranking set for each user from the set of positive examples, denoted by $S = \{(u, i, j): r_{ui} = 1\}$. The underlying assumption for this construction is that, if $r_{ui} = 1$, then user $u$ prefers item $i$ over item $j$.

As shown in [31], assuming that the users act independently of each other and that $>_u$ is independent across $u$, the model likelihood can be maximized by maximizing

$$L(\theta) = \prod_{(u, i, j) \in D_S} P[i >_u j]. \tag{10}$$

To adapt the notion of relative preferences for OCuLaR, suppose that the probability of a user preferring item $i$ over item $j$ is given by

$$P[i >_u j] = (1 - e^{-\langle f_u, f_i \rangle})e^{-\langle f_u, f_j \rangle}.$$

To see the formal relation to the model in Section 4.1, simply note that, with this choice,

$$P[i >_u j] = P[r_{ui} = 1]P[r_{uj} = 0]$$

with $P[r_{ui} = 1]$ as defined in (1). Maximizing (10) is equivalent to minimizing the logarithm of (10), i.e., minimizing

$$L(\theta) = \sum_{(u, i, j) \in D_S} \log P[r_{ui} = 1] + \log P[r_{uj} = 0].$$

where we defined $w_u = \{(u, i, j): r_{ui} = 1\}$. Note that the RHS above differs only in the factors $w_u$ from the negative log-likelihood (2) in Section 4.2. The factors $w_u$ assign a large weight to positive examples of users that have few positive examples associated with them.

We denote the algorithm obtained by substituting the log-likelihood $-\log L$ in Section 4.2 with the RHS above as the relative OCuLaR (R-OCuLaR) algorithm. Its implementation is essentially equivalent to that of the (original) OCuLaR algorithm; in fact, it has exactly the same complexity.

6 ADDING EXTRA INFORMATION

While the focus of this paper is on generating recommendations based on positive examples only, e.g., on the purchase history of clients, we briefly explain the incorporation of additional information in the OCuLaR algorithm in this section. This is important for addressing the cold-start problem, i.e., for making recommendations to users that have no purchase history, or for recommending items that have not been purchased. In addition, it can also improve the recommendation performance for users/items for which positive examples exist. Additional information can be of the form of user and item attributes, such as the affiliation of a user or item with a certain hierarchy or ontology. For example, in the B2B setting, a client-company belongs to a certain industry (automotive, banking, etc.), and in the traditional consumer setting, a movie or a book belongs to one or more genres (biography, fiction, etc.).

We restrict our attention to the incorporation of a single categorical item attribute, denoted by $A_i$, that takes on values in a finite set $\{a_1, \ldots, a_J\}$. The extension to several (distinct) item attributes and to user attributes will turn out to be straightforward. The item attribute is incorporated into the generative model by assuming that the attribute values $A_i$ are generated by the co-cluster affiliations, similarly as the positive examples are generated by the co-cluster affiliations (cf. Section 4.1). This approach is quite common; for example, in the context of article recommendations, the collaborative topic model [42] assumes that article ratings are generated by latent vectors and by topic proportions (i.e., by item attributes). Similarly, binary features can be incorporated into the community model by proposing a model that generates both node attributes and edges in the corresponding graph with communities by regarding the community affiliation vectors as the inputs features to a (binary) logistic model [46].

In the same way, we assume that the $A_i$ are generated by the co-clusters by regarding the co-cluster affiliation vectors
\( f_i \), \( f_i \) as input features to a logistic model. Specifically, the probability of \( A_i \) taking on the value \( a_k \) is given by

\[
P[A_i = a_k | F] = \frac{e^{-w_{ik} - (w_{ik} f_i)}}{1 + \sum_{\ell=1}^{J-1} e^{-w_{ik} - (w_{ik} f_i)}}, \quad k = 1, ..., J - 1,
\]

\[
P[A_i = a_j | F] = \frac{1}{1 + \sum_{\ell=1}^{J-1} e^{-w_{ik} - (w_{ik} f_i)}},
\]

where \( w_{ik}, w_{ik} \in \mathbb{R}^K, \ell = 1, ..., J - 1 \), are the associated weight factors, and \( F \) contains \( f_i, f_i \) as columns. Assuming that attributes are generated independently for each item, the attribute likelihood is given by

\[
\mathcal{L}_A = \prod_i P[A_i | F, W],
\]

where the product is over all items \( i \) that have a feature associated with them, and \( W \) denotes the set of associated weight factors \( w_{ik}, w_{ik} \in \mathbb{R}^K, \ell = 1, ..., J - 1 \). Assuming that the positive examples and the attributes are generated independently based on the community membership, the negative model log likelihood of the overall model is given by

\[
- \log \tilde{\mathcal{L}} = - \log \mathcal{L} - \mu \log \mathcal{L}_A, \quad \text{where } \mu \text{ is a parameter that weights the likelihoods.}
\]

To find the factors \( f_i, f_i \) and the weights of the logistic model \( w_{ik}, w_{ik} \), we solve—analogously as in (3)—the following optimization problem:

\[
\text{minimize } \tilde{Q} \text{ subject to } [f_i], [f_i] \geq 0, \forall c, \quad (11)
\]

where

\[
\tilde{Q} = - \log \mathcal{L} - \mu \log \mathcal{L}_A + \lambda \sum_i ||f_i||_2^2 + \lambda \sum_u ||f_u||_2^2.
\]

Here, \( \lambda \geq 0 \) is a regularization parameter, and minimization is with respect to the factors \( f_i, f_i \) and the weights \( w_{ik}, w_{ik} \).

Similarly as before, one can use a cyclic block coordinate descent approach to minimize (11). Specifically, we iteratively update first \( f_i \), then \( f_i \) both via projected gradient descent, and finally the weights \( w_{ik}, w_{ik} \), with gradient descent. We finally note that this variant of OCULaR is very efficient; specifically, the complexity of updating all factors and weights is \( O((|\{(i, u): r_{ui} = 1\}| + K + JK N)) \), where \( N \) is the number of items.

7 EXPERIMENTS

Here, we compare the prediction accuracy of OCULaR to that of other interpretable and non-interpretable one-class CF algorithms. We compare with interpretable user-based and item-based collaborative filtering approaches, and non-interpretable state-of-art one-class recommendation algorithms based on matrix factorization. Both OCULaR and R-OCULaR typically outperform or are on par with the existing one-class recommendation algorithms in terms of recommendation performance. We also discuss the choice of the input parameters for our approach. We demonstrate that OCULaR is scalable and easily parallelizable, and we conclude with an industrial deployment.

7.1 Datasets

We use four datasets. First, we consider a real-world dataset from our institution. It consists of the buying history of 80,000 clients with whom our institution interacts and 3,000 products or services offered by our institution. The clients in this case are not individuals but companies, and the recommender system operates in an B2B setting. We call this dataset B2B-DB. The dataset is highly imbalanced in the sense that the most frequently purchased item was bought by more than 50\% of the users, whereas the most infrequently purchased item was bought only 50 times.

We also consider three public datasets. The first was extracted by [43] from the CiteULike website, which assists users in creating collections of scientific articles. The dataset consists of 5,551 users and 16,980 articles. Each user has a number of articles in their collection, which are considered as positive examples. Based on the positive examples, the goal is to generate new article recommendations. The second dataset is the Movielens 1 million dataset, which consists of 1 million ratings from 6,000 users on 4,000 movies. The final dataset is the Netflix dataset, consisting of about 100 million ratings that 480,189 users gave to 17,770 movies.

In both the Movielens and the Netflix dataset, the users provide ratings between 1 and 5 stars. As we consider a one-class collaborative filtering task, we adopt a convention from many previous works (e.g., [28], [37]) and only consider ratings larger than or equal to 3 as positive examples and ignore all other ratings. Therefore, the task now is equivalent to predicting whether the user will give a rating greater than 3 (i.e., is likely to enjoy the movie).

7.2 Recommendation performance

7.2.1 Evaluation Metrics

We measure performance in terms of recall at \( M \) items (recall@M), and mean average precision at \( M \) items (MAP@M).

In the one-class setting, recall is a more sensible measure than precision, because an example being unknown \( (r_{ui} = 0) \) does not mean that user \( u \) would not rate item \( i \) positively [35]. Given an ordered (by relevance) list of \( M \) recommendations for user \( u \), denoted by \( i_1, ..., i_M \), the recall@M items for user \( u \) is defined as

\[
\text{recall@M}(u) = \frac{|\{i: r_{ui} = 1\} \cap \{i_1, ..., i_M\}|}{|\{i: r_{ui} = 1\}|}.
\]

The overall recall@M is obtained as the average over recall@M(u).

MAP is commonly used in information retrieval for evaluating a ranked or ordered list of items, and is considered a good measure of performance when a short list of the most relevant items is shown to a user. MAP@M items is the mean (over all users) of the average precision at \( M \) items (AP@M), defined as

\[
\text{AP}@M(u) = \sum_{m=1}^{M} \text{Prec}(m) \frac{1_{\{r_{ui_m} = 1\}}}{\min(|\{i: r_{ui} = 1\}|, M)},
\]

where \( |\{i: r_{ui} = 1\}| \) is the number of positive examples corresponding to user \( u \); \( 1_{\{r_{ui_m} = 1\}} \) is equal to 1 if \( r_{ui_m} = 1 \) and zero otherwise, and \( \text{Prec}(m) \) is the precision at a cutoff rate \( m \):

\[
\text{Prec}(m) = \frac{|\{i: r_{ui} = 1\} \cap \{i_1, ..., i_m\}|}{m}.
\]
The results across all datasets show that OCuLaR and R-OCuLaR are either the best or the second-best performing algorithm (together with wALS). In addition, OCuLaR has the advantage of providing interpretable recommendations, an aspect that is compromised when using other OCCF approaches such as wALS or BPR. Moreover, OCuLaR is significantly better than the user- and item-based algorithms, its interpretable competitors. This is not surprising because the user- and item-based approaches both consider only clustering in either the user or the item space, whereas OCuLaR can discover more complex structures in the joint user-item space.

All approaches require the setting of hyperparameters. To make the comparison fair, we try to find the best set of hyperparameters using a grid search approach. We do not consider the variant of OCuLaR which uses extra information for fairness to the baseline techniques that use only the matrix of positive examples. For OCuLaR and R-OCuLaR, we performed a grid search over $K$ and $\lambda$. For the user- and item-based collaborative filtering approaches, we performed a grid search over the number of nearest neighbors. For wALS, we choose the weight in the loss function (8) as $b = 0.01$, the regularization parameter as $\lambda = 0.01$, and performed a grid search over the dimension of the latent vectors. Finally, for BPR we searched over the dimension of the latent factors $K$, and the regularization parameter $\lambda$.

In Table 1, we plot the results obtained with the parameters that yield the best recall@50 for each approach. The Netflix dataset is not included because not all baselines can be run for very large datasets.

### 7.2.2 Comparison with baselines

We compare the OCuLaR algorithm with various interpretable and non-interpretable baseline algorithms:

- **User-based** collaborative filtering using a cosine similarity metric [34]. Such an algorithm is interpretable because a recommendation can be justified with a reasoning of the type: “item $i$ is recommended because the similar users $u_1, \ldots, u_k$ also bought item $i^\prime$.”

- **Item-based** collaborative filtering using cosine similarity [7]. This algorithm is also interpretable. It can be accompanied with a recommendation as follows: “item $i$ is recommended because user $u$ bought the similar items $i_1, \ldots, i_k$.”

- **Weighted Alternating Least Squares (wALS)**, a state-of-the-art one-class matrix factorization approach [27]. wALS minimizes (7) with loss function (8) by alternatingly optimizing the user and the item factor via least-squares. It offers good prediction performance but the reasoning is not directly interpretable, as mentioned earlier.

- **Bayesian personalized ranking (BPR)**, a state-of-the-art matrix factorization approach that converts the set of positive examples into a set of relative preferences [31] (see also Section 5). The recommendations provided by BPR are also not directly interpretable. For BPR we used the python/theano implementation from https://github.com/bbcrd/theano-bpr.git.

All approaches require the setting of hyperparameters. To make the comparison fair, we try to find the best set of hyperparameters using a grid search approach. We do not consider the variant of OCuLaR which uses extra information for fairness to the baseline techniques that use only the matrix of positive examples. For OCuLaR and R-OCuLaR, we performed a grid search over $K$ and $\lambda$. For the user- and item-based collaborative filtering approaches, we performed a grid search over the number of nearest neighbors. For wALS, we choose the weight in the loss function (8) as $b = 0.01$, the regularization parameter as $\lambda = 0.01$, and performed a grid search over the dimension of the latent vectors. Finally, for BPR we searched over the dimension of the latent factors $K$, and the regularization parameter $\lambda$.

In Table 1, we plot the results obtained with the parameters that yield the best recall@50 for each approach. The Netflix dataset is not included because not all baselines can be run for very large datasets.
co-clusters is neither too big nor too small, and also that each user or item does not belong to too many co-clusters. The size of the co-clusters can depend on application-specific criteria, such as, for example, that a co-cluster should contain at least 100 users. For the specific example, a value of $K$ in the range between 100-200 would be adequate to ensure good prediction and avoid excessively big, thus not dense, co-clusters.

The textual content of the figure shows recall@50 items, users in co-cluster, items in co-cluster, co-cluster densities, co-clusters an item is in, and co-clusters a user is in. The figure also shows the running time per iteration of the OCuLaR algorithm applied to a fraction of the Netflix dataset as a function of the the number of threads.

### 7.4 Scalability

In this section, we investigate the scalability of the OCuLaR algorithm. We consider the full Netflix dataset, consisting of 100,480,507 ratings from 480,189 users on 17,770 movie titles. As before, we take the ratings $\geq 3$ as positive examples, i.e., $r_{ui} = 1$. As analyzed in Section 4.4, the training time required by the OCuLaR algorithm is essentially linear in the number of positive examples $|\{(u, i) : r_{ui} = 1\}|$, and linear in the number of co-clusters $K$. In Figure 5 we plot the running time per iteration for increasing fractions of the Netflix dataset (i.e., non-zero entries), chosen uniformly from the whole Netflix dataset. We see that the training time is indeed linear in the number of positive examples $|\{(u, i) : r_{ui} = 1\}|$ and linear in the number of co-clusters $K$.

OCuLaR can also easily be parallelized because for an item (user) update cycle, each item (user) factor can be updated independently. Therefore independent threads can update a fraction of the user and item factors. Figure 5 depicts the reduction in running time that can be realized by the corresponding parallel OCuLaR implementation as a function of the number of threads.

### 8 INDUSTRIAL DEPLOYMENT

We used our algorithm in a B2B recommender system of our institution. These recommendations are not offered directly to the clients of our institution, but rather to our sales teams. The salesperson responsible for an account examines the recommendations provided and decides whether to act on the recommendation and approach the client. This decision is based on the reasoning provided and on the salesperson’s own experience through past interaction with the client.

The textual output of the recommendation, which conveys the corresponding rationale is shown in Figure 6. For reasons of anonymity, the names of the companies/clients belonging to each co-cluster have been omitted.

In the example, it can be seen that the service “Custom Cloud” is recommended to Client 1 because the client is affiliated with three co-clusters. Co-cluster 1 shows an affinity of Client 1 with several airlines, and in Co-cluster 3 with telco companies.

During the deployment of the OCuLaR algorithm in our organization, we interacted with many sellers that used the platform. Sellers expressed satisfaction about the reasoned...
aspect of the recommendations. An interesting comment that we received is that the tool could also constitute an educational platform [6] for young sellers, because the detailed reasoning can teach them the currently discovered buying patterns. In an extended version, we plan to provide additional metrics, such as the conversion ratio and the monetary benefits of the industrial deployment.

9 Conclusion
A large amount of work on recommender systems and machine learning has focused primarily on the accuracy of prediction rather than on interpretability. This work explicitly addresses the aspect of interpretability by enabling the detection of overlapping co-clusters that can be easily visualized and transcribed into a textual description. The methodology presented is interpretable and at the same time scalable. More importantly, our approach does not sacrifice accuracy of prediction.

10 Acknowledgements
The research leading to these results has received funding from the European Research Council under the European Union’s Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no. 259569.

References


