Exploiting sparsity to build efficient kernel based collaborative filtering for top-N item recommendation

Exploiting sparsity to build efficient kernel based collaborative filtering for top-N item recommendation

Mirko Polato

University of Padova - Department of Mathematics
Via Trieste, 63, 35121 Padova - Italy

Fabio Aiolli

University of Padova - Department of Mathematics
Via Trieste, 63, 35121 Padova - Italy

Abstract

The increasing availability of implicit feedback datasets has raised the interest in developing effective collaborative filtering techniques able to deal asymmetrically with unambiguous positive feedback and ambiguous negative feedback.

In this paper, we propose a principled kernel-based collaborative filtering method for top-N item recommendation with implicit feedback. We present an efficient implementation using the linear kernel, and we show how to generalize it to kernels of the dot product family preserving the efficiency.

We also investigate on the elements which influence the sparsity of a standard cosine kernel. This analysis shows that the sparsity of the kernel strongly depends on the properties of the dataset, in particular on the long tail distribution. We compare our method with state-of-the-art algorithms achieving good results both in terms of efficiency and effectiveness.

Keywords: Top-N recommendation, Kernel, Collaborative Filtering, Large Scale

1. Introduction

Collaborative filtering (CF) techniques can make recommendation to a user exploiting information provided by similar users. The typical CF setting consists of a set $\mathcal{U}$ of n users, a set $\mathcal{I}$ of m items, and the so-called rating matrix

$\mathbf{R} = \{r_{ui}\} \in \mathbb{R}^{n \times m}$. Ratings represent, numerically, the interactions between users and items. These interactions can be of two different types: explicit feedbacks and implicit feedbacks. The former are interactions performed by users in a direct way, such as, by giving a one-to-five star rate, thumbs-up versus thumbs-down, and so on. The latter, instead, are actions performed by users without the awareness of giving some kind of feedback to the system, e.g., clicks, views, elapsed time on a web page and so on.

The explicit setting have got most of the attention from the CF research community and a lot of rating prediction algorithms have been proposed. More recently however, the focus is constantly shifting towards the implicit feedback scenario since users are not always willing to give their opinion explicitly. Furthermore, implicit interactions are easier to collect: for an already existing system, it is sufficient to store the operations done by a user in a session without changing the front-end and consequently there are no additional burden for the user.

One problem arising in implicit feedback is that typically the feedback is asymmetric, that is, while there can be evidence that a certain user interacted and hence showed some interest towards a product (unambiguous feedback), the opposite is not true, that is the missing evidence of interaction does not imply that a given user dislikes that item.

In this paper we focus on the implicit feedback scenario, and so we assume binary ratings, $r_{ui} \in \{0, 1\}$, where $r_{ui} = 1$ means that user u interacted with the item i (unambiguous feedback) and $r_{ui} = 0$ means there is no evidence that user u interacted with the item i (ambiguous feedback).

Specifically, the main contribution of this paper is a CF framework for top-N item recommendation based on the seminal work described in [1]. Starting from the original convex optimization problem, we propose an efficient variation of that formulation which makes the algorithm able to manage very large scale datasets, by preserving the effectiveness of the original algorithm. This new formulation is then extended to be used with general kernels thus augmenting the expressiveness of the representation in collaborative filtering.

It is well known that kernels are not suited for large datasets since they have a prohibitive computational complexity. Datasets used in CF domains are usually large: rating matrices **R** have tens of thousands of rows and columns and so the application of kernels seems to be difficult. Nevertheless, we can observe that the complexity of the computation of a kernel strictly depends on its sparsity and hence being able to control the sparsity makes the application of kernel methods to very large datasets feasible. Unfortunately, kernels are usually dense unless we are working on very large and sparse feature spaces and this is in contrast with what we would like to have.

Exploiting a well known result from harmonic theory [2], we are able to demonstrate how using *dot product kernels*, a generalization of normalized homogeneous polynomial kernels, kernels as sparse as the standard cosine kernel can be obtained without changing the solution of the problem.

However, it should be noted that there is no guarantees about the sparsity of a cosine kernel because it is closely related to the distribution of the non-zero values in the matrix **R**. CF datasets are known to be very sparse, and also they are usually extracted from e-services which makes them subject to the long tail phenomenon that is often associated with the power law distribution.

Even if not every long tail is a power law [3], this tailed distribution over the item ratings in **R** poses a strong bias on the density of the resulting kernel. In fact, assuming the long tail, in **R** there will be few dense columns and many sparse ones, which intuitively leads to a rather sparse kernel. On the other side, a long tailed distribution over the user ratings can lead to dense kernels matrices since different items tend to be rated by mostly the same users. We provide a deep analysis of this phenomenon showing theoretically and empirically which are the conditions for which a dataset is likely to produce a sparse dot product kernel. It is worth to notice that these results are not only applicable to CF contexts, but they apply on every other context where the data distribution is long tailed.

Finally, in the experimental section we compare our framework with two state-of-the-art methods in top-N recommendation. The empirical work shows how our method can achieve good results in terms of AUC (Area Under the ROC Curve) and also in terms of efficiency.

Summarizing, the contribution of this paper is 3-fold:

1. Starting from the seminal work described in [1] we propose an optimized CF framework for top-N recommendation. Specifically, our proposed method results far more efficient of the original enabling the application of the method to large and very large datasets (e.g. in the order of 1 million of users/items and 50 million of rates) while preserving the state-of-the-art effectiveness of the original algorithm.
2. The formulation depicted above is generalized to be used with non-linear kernels as dot-product kernels, that is kernel functions in the form $k(\mathbf{x}, \mathbf{y}) = f(\mathbf{x} \cdot \mathbf{y})$ where f is a non-linear function admitting a Maclaurin expansion with non-negative coefficients. A sparsification method for dot-product kernels is also provided making the sparsity of any dot-product kernel equal to the sparsity of the linear or cosine kernel.
3. A theoretical and empirical analysis concerning the sparsity of the standard cosine kernel is proposed. This analysis shows that the sparsity of the kernel produced depends on the properties of the user activity and item popularity long-tails. In particular, long-tails observed on item popularity improve the sparsity of the kernels while long-tails observed on user activity is detrimental for the sparsity of the kernels.

This paper is an extended version of a preliminary paper presented at ESANN 2016 [4]. In particular, preliminary work about the contribution 1 and 2 above was already presented in that work while contribution 3 is novel. Moreover, the extensive experimental work of this paper was not present in the preliminary version.

The rest of the paper is organized as follows. In Section 2 we will introduce the notation used throughout the paper. Section 3 presents related works on top-N recommendation for implicit feedback. Sections 4 and 5 describes our framework with a particular focus on the applicability of our proposed kernel method. Finally, Sections 6 and 7 show the experimental results and which directions our research can follow in the future.

2. Notation

In this section we provide some useful notation used throughout the paper. Recommender algorithms are thought to give suggestions to users about items. We call the set of users as $\mathcal{U}$ such that $|\mathcal{U}| = n$, the set of items as $\mathcal{I}$ such that $|\mathcal{I}| = m$ and the set of ratings $\mathcal{R} = \{(u, i)\}$.

We refer to the binary rating matrix with $\mathbf{R} = \{r_{ui}\} \in \mathbb{R}^{n \times m}$, where users are on the rows and items on the columns. We add a subscription to both user and item sets to indicate, respectively, the set of items rated by a user u ($\mathcal{I}_u$) and the set of users who rated the item i ($\mathcal{U}_i$).

3. Related works

Top-N recommendation finds application in many different domains such as TV and movies [5], books [1], music [6, 7], social media [8] and so on.

Top-N recommendation methods can be divided into two macro categories.

The first is the neighbourhood-based CF algorithms [9], in which the recommendation for a target user is made by using the ratings of the most similar users. This category comprises the so called memory-based methods that do not need the construction of a model, but they directly use the data inside the rating matrix.

Despite, in general, these methods suffer from low accuracy, in 2013 the winner of the remarkable challenge organized by Kaggle, the Million Songs Dataset challenge [10], was an extension of the well known item-based nearest-neighbors (NN) algorithm [11]. This extension [6] (here called MSDW) introduced an asymmetric similarity measure, called asymmetric cosine. In a classic item-based CF method, the scoring function for a user-item pair (u, i) is computed by a weighted sum over the items liked by u in the past, that is:

$$\hat{r}_{ui} = \sum_{j \in \mathcal{I}} w_{ij} r_{uj} = \sum_{j \in \mathcal{I}_u} w_{ij},$$

where w_{ij} expresses the similarity between item i and item j.

One of the main contribution, presented in [6], is the asymmetric cosine (asymC) similarity. The intuition behind asymC comes from the observation that the cosine similarity can be expressed, over a pair of items (a, b), as the square root of the product of the reciprocal conditional probabilities. Let $\mathbf{a}, \mathbf{b} \in \{0, 1\}^n$ be respectively the binary vector representations of items a and b. The idea of asymmetric cosine similarity is to give different weights to the conditional probabilities, that is

$$S_\alpha(a, b) = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\|^{2\alpha} \|\mathbf{b}\|^{2(1-\alpha)}} = P(a|b)^\alpha P(b|a)^{1-\alpha},$$

with $0 \leq \alpha \leq 1$. In case of binary rates this asymmetric similarity can be computed as in the following. Let $\mathcal{U}_i$ represents the set of users who rated the item i, then the asymC between item i and item j is defined by:

$$w_{ij} = S_\alpha(i, j) = \frac{|\mathcal{U}_i \cap \mathcal{U}_j|}{|\mathcal{U}_i|^\alpha |\mathcal{U}_j|^{1-\alpha}}.$$

Besides its outstanding performance in terms of mAP@500, the MSD winning solution is also easily scalable to very large datasets. However, one drawback of this solution is that it is not theoretically well founded.

The second category is the model-based CF techniques, which construct a model of the information contained in the rating matrix $\mathbf{R}$. In the last two decades many different model-based approaches have been proposed. A particular attention has been devoted to latent factor models which try to factorize the rating matrix into two low-rank matrices, $\mathbf{R} = \mathbf{WX}$, which represent user-factors ($\mathbf{W}$) and item-factors ($\mathbf{X}$). These factors are “meta-features” that define the user tastes and how much of these features represent an item. Usually these methods are referred to as matrix factorization methods. The prediction for a user-item pair is simply done by a dot product of the corresponding row and column in the factorized matrices.

One of the most used matrix factorization approach for implicit feedback is presented in [5] (WRMF: Weighted Regularized Matrix Factorization). In this work Hu et al. propose an adaptation of the classic SVD (Singular Value Decomposition) method in which they minimize the square-loss using two regularization terms in order to avoid overfitting. Their optimization criterion is defined as:

$$\sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} c_{ui} (\mathbf{w}_u^\top \mathbf{x}_i - 1)^2 + \lambda \|\mathbf{W}\|^2 + \lambda \|\mathbf{X}\|^2,$$

where c_{ui} are a-priori weights for each pair (u, i) such that positive feedbacks have higher weights. Actually, this method uses information about the rating values (not binary) to give more importance to user-item interactions with high rating values, in fact, c_{ui} is calculated by: $c_{ui} = 1 + \alpha r_{ui}$. In their experiments the best performances has been achieved with $\alpha = 40$, but with binary rating matrices this parameter losses a lot of its importance. As we will see in the experimental section, changing the value of α does not change the performance of the algorithm. A similar approach has been used by Wang et al. [12] where they proposed a framework for broadcast email prioritization based on a novel active learning method.

In [13] the top-N recommendation problem has been formulated as a ranking problem. Rendle et al. proposed a Bayesian Personalized Ranking (BPR) criterion that is the maximum posterior estimator derived from a Bayesian analysis. They also provide a learning method based on stochastic gradient descent with bootstrap sampling. They finally show how to adopt this criterion for kNN (BPRkNN) and MF methods (BPRMF).

In [14] Ning et al. presented SLIM (Sparse LInear Method) which generates top-N recommendation by aggregating from user rating profiles. SLIM learns an aggregation coefficient matrix by solving a regularized optimization problem.

More recently, a new principled algorithm for CF, which explicitly optimizes the AUC, has been proposed with very nice performances on the MovieLens dataset [1]. Since this algorithm represents the seminal work for our framework, we will discuss about it in details in the next section.

A more direct approach in order to build a good ranking over the items is the so called *learning to rank*[15]. Learning to rank methods exploit supervised machine learning to solve ranking problems. These techniques can be divided into three categories: pointwise approaches [8, 16, 17] in which for each query-document (i.e., user-item) pair a score is predicted and then used to build the ranking; pairwise approaches [18, 19] face the ranking problem as a binary classification one (positive document versus negative ones) in which they try to minimize the number of inversions in the ranking; listwise approaches [20, 21] try to directly optimize one of the ranking evaluation measures. The big challenge here is the fact that most of the measures are not continuous function w.r.t. the model's parameters and for this reason approximations have to be used.

4. CF-OMD framework

4.1. CF-OMD

In this section we present the seminal CF algorithm, called CF-OMD (Optimization of the Margin Distribution) [1], for top-N recommendation inspired by preference learning [22][23], and designed to explicitly maximize the AUC (Area Under the ROC Curve).

Consider the matrix $\mathbf{W} \in \mathbb{R}^{n \times k}$ be the embeddings of users in a latent factor space and $\mathbf{X} \in \mathbb{R}^{k \times m}$ be the embeddings of items in the space. Given a user, a ranking over items can be induced by the factorization $\hat{\mathbf{R}} = \mathbf{W}\mathbf{X}$, where $\hat{r}_{ui} = \mathbf{w}_u^\top \mathbf{x}_i$ with the constraint $\|\mathbf{w}_u\| = \|\mathbf{x}_i\| = 1$. The model parameters (i.e., $\mathbf{W}, \mathbf{X}$) are generally computed minimizing a regularized loss function over the ground truth matrix $\mathbf{R}$. Let now fix the item representation as $\mathbf{x}_i = \mathbf{r}_i / \|\mathbf{r}_i\|$, and let $\rho(i \prec_u j) = (\hat{r}_{ui} - \hat{r}_{uj})/2 = \mathbf{w}_u^\top (\mathbf{x}_i - \mathbf{x}_j)/2$ be the margin for an item pair (i, j) for user u. Let also define the probability distribution over the positive and negative items for u,

$$\mathbf{A}_u = \{\boldsymbol{\alpha}_u \in \mathbb{R}_+^m | \sum_{i \in \mathcal{I}_u} \alpha_{ui} = 1, \sum_{i \notin \mathcal{I}_u} \alpha_{ui} = 1\}.$$

In [1] it is proposed an approach to maximize the minimum margin inspired by preference learning where the ranking task is posed as a two-player zero-sum game. Let P_{max} and P_{min} be the players: on each round of the game, P_{min} picks a preference $i \prec_u j$ and, simultaneously, P_{max} picks an hypothesis $\mathbf{w}_u$ with the aim of maximizing the margin $\rho(i \prec_u j)$. The value of the game, i.e., the expected margin, is computed by:

$$\mathbb{E}_{\alpha}[\rho] = \frac{1}{2}\mathbf{w}_u^{\top}\mathbf{X}\mathbf{Y}_u\boldsymbol{\alpha}_u,$$

where $\mathbf{Y}_u$ is a diagonal matrix, $\mathbf{Y}_u = diag(\mathbf{y}_u)$, such that $y_{ui} = 1$ if $i \in \mathcal{I}_u$, -1 otherwise. It can be demonstrated that the $\mathbf{w}_u^*$ maximizing the expected margin is equal to $\mathbf{w}_u^* = \mathbf{X}\mathbf{Y}_u\boldsymbol{\alpha}_u$ normalized. Finally, the best strategy for P_{min} can be expressed as a convex quadratic optimization problem:

$$\boldsymbol{\alpha}_u^* = \underset{\boldsymbol{\alpha}_u \in \mathbf{A}_u}{\operatorname{argmin}} \; \boldsymbol{\alpha}_u^{\top}\left(\mathbf{Y}_u\mathbf{X}^{\top}\mathbf{X}\mathbf{Y}_u + \boldsymbol{\Lambda}\right)\boldsymbol{\alpha}_u, \tag{1}$$

in which $\boldsymbol{\Lambda}$ is a diagonal matrix such that $\boldsymbol{\Lambda}_{ii} = \lambda_p$ if $i \in \mathcal{I}_u$, otherwise $\boldsymbol{\Lambda}_{ii} = \lambda_n$, where λ_p and λ_n are regularization parameters ($\lambda_p, \lambda_n \geq 0$).

Although this algorithm has shown state-of-the-art results in terms of AUC, it is not suitable to deal with large datasets. In fact, let assume that each optimization problem can be solved by an algorithm with a complexity quadratic on the number of parameters. Then the global complexity would be $O(n_{ts}m^2)$, where n_{ts} is the number of users in the test set, and for the MSD it would be $O(10^{19})$.

4.2. Efficient CF-OMD

The main issue of CF-OMD, in terms of efficiency, is the number of parameters which is equal to the cardinality of the item set. Analyzing the results reported in [1], we noticed that high values of λ_n did not particularly affect the results, because it tends to flatten the contribution of the ambiguous negative feedbacks toward the average, mitigating the relevance of noisy information.

In CF contexts the data sparsity is particularly high, this means, on average, that the number of ambiguous negative feedbacks is orders of magnitude greater than the number of positive feedbacks. Formally, given a user u, let $m_u^+ = |\mathcal{I}_u|$ and $m_u^- = |\mathcal{I} \setminus \mathcal{I}_u|$ then $m = m_u^- + m_u^+$, where $m_u^+ \ll m_u^-$, and generally $O(m) = O(m_u^-)$.

On the basis of this observation, we can simplify the optimization problem (1), by fixing $\lambda_n = +\infty$, which means that $\forall i \notin \mathcal{I}_u, \alpha_{ui} = 1/m_u^-$:

$$\boldsymbol{\alpha}_u^* = \underset{\boldsymbol{\alpha}_u}{\operatorname{argmin}} \; \|\boldsymbol{\alpha}_{u^+}^\top \mathbf{X}_{u^+} - \boldsymbol{\mu}_u^-\|^2 + \lambda_p \|\boldsymbol{\alpha}_{u^+}\|^2 \tag{2}$$

$$= \underset{\boldsymbol{\alpha}_u}{\operatorname{argmin}} \; \|\boldsymbol{\alpha}_{u^+}^\top \mathbf{X}_{u^+}\|^2 - \|\boldsymbol{\mu}_u^-\|^2 - 2\boldsymbol{\alpha}_{u^+}^\top \mathbf{X}_{u^+}^\top \boldsymbol{\mu}_u^- + \lambda_p \|\boldsymbol{\alpha}_{u^+}\|^2 \tag{3}$$

$$= \underset{\boldsymbol{\alpha}_{u^+} \in \mathbf{A}_u}{\operatorname{argmin}} \; \boldsymbol{\alpha}_{u^+}^\top \mathbf{X}_{u^+}^\top \mathbf{X}_{u^+} \boldsymbol{\alpha}_{u^+} + \lambda_p \|\boldsymbol{\alpha}_{u^+}\|^2 - 2\boldsymbol{\alpha}_{u^+}^\top \mathbf{X}_{u^+}^\top \boldsymbol{\mu}_u^-, \tag{4}$$

where

$$\boldsymbol{\mu}_u^- = \frac{1}{m_u^-} \sum_{i \notin \mathcal{I}_u} \mathbf{x}_i$$

is the centroid of the convex hull spanned by the negative items and $\boldsymbol{\alpha}_{u^+}$ are the probabilities associated with the positive items, $\mathbf{X}_{u^+}$ is the sub-matrix of $\mathbf{X}$ containing only the columns corresponding to the positive items. The number of parameters in (4) is m_u^+ and hence the complexity from $O(n_{ts} m^2)$ drops to $O(n_{ts} \overline{m}_u^{+2})$, where $\overline{m}_u^+ = \mathbb{E}[|\mathcal{I}_u|]$ is the expected cardinality of the positive item set. In MSD $\overline{m}_u^+ \approx 47.46$ which leads to a complexity $O(10^8)$.

4.2.1. Implementation trick

Notwithstanding the huge improvement in terms of complexity, a naïve implementation would have an additional cost due to the calculation of $\boldsymbol{\mu}_u^-$. For all users in the test set the cost would be $O(n_{ts} n \overline{m}_u^-)$, where $\overline{m}_u^- = \mathbb{E}[|\mathcal{I} \setminus \mathcal{I}_u|]$, and it can be approximated with $O(n_{ts} n m)$.

To overcome this bottleneck, we propose an efficient incremental way of calculating $\boldsymbol{\mu}_u^-$. Consider the mean over all items

$$\boldsymbol{\mu} = \frac{1}{m} \sum_{i \in \mathcal{I}} \mathbf{x}_i,$$

then, for a given user u, we can express

$$\boldsymbol{\mu}_u^- = \frac{1}{m_u^-} \left(m \cdot \boldsymbol{\mu} - \sum_{i \in \mathcal{I}_u} \mathbf{x}_i \right).$$

From a computational point of view, it is sufficient to compute the sum $\sum_{i\in\mathcal{I}} \mathbf{x}_i$ once (i.e., $m \cdot \boldsymbol{\mu}$) and then, for every $\boldsymbol{\mu}_u^-$, subtract the sum of the positive items. Using this simple trick, the overall complexity drops to $O(nm) + O(n_{ts}^2 \overline{m}_u^+)$.

In the experimental section we successfully applied this algorithm to the MSD achieving competitive results against the state-of-the-art method but with higher efficiency.

5. Kernelized CF-OMD

The method proposed in Section 4.2, can be seen as a particular case of a kernel method. In fact, $\mathbf{X}_{u^+}^\top \mathbf{X}_{u^+}$ is a kernel matrix, let call it $\mathbf{K}_{u^+}$ with the corresponding (linear) kernel function $K : \mathbb{R}^{m_u^+} \times \mathbb{R}^{m_u^+} \to \mathbb{R}$. Given K we can reformulate (4) as:

$$\boldsymbol{\alpha}_{u^+}^* = \underset{\boldsymbol{\alpha}_{u^+} \in \mathbf{A}_u}{\operatorname{argmin}} \;\; \boldsymbol{\alpha}_{u^+}^\top \mathbf{K}_{u^+} \boldsymbol{\alpha}_{u^+} + \lambda_p \|\boldsymbol{\alpha}_{u^+}\|^2 - 2\boldsymbol{\alpha}_{u^+}^\top \mathbf{q}_u, \tag{5}$$

where elements of the vector $\mathbf{q}_u \in \mathbb{R}^{m_u^+}$ are defined as

$$q_{ui} = \frac{1}{m_u^-} \sum_{j \notin \mathcal{I}_u} K(\mathbf{x}_i, \mathbf{x}_j).$$

Actually, inside the optimization problem (5) we can plug any kernel function. Throughout the paper we will refer to this method as CF-KOMD. Generally speaking, the application of kernel methods on a huge dataset have an intractable computational complexity. Without any shrewdness the proposed method would not be applicable because of the computational cost of the kernel matrix and $\mathbf{q}_u$.

An important observation is that the complexity is strictly connected with the sparsity of the kernel matrix which is, unfortunately, commonly dense. However, we can leverage on a well known result from harmonic theory [2] to keep the kernel as sparse as possible without changing the solution of CF-KOMD.

Theorem 1. *A function $f : \mathbb{R} \to \mathbb{R}$ defines a positive definite kernel k : $\mathbf{B}(0,1) \times \mathbf{B}(0,1)$ as $k : (\mathbf{x}, \mathbf{y}) \mapsto f(\mathbf{x} \cdot \mathbf{y})$ iff f is an analytic function admitting a Maclaurin expansion with non-negative coefficients, $f(x) = \sum_{s=0}^{\infty} a_s x^s, a_s \geq 0$.*

As emphasized in [2, 24], many kernels used in practice [25] satisfy the abovementioned condition. These kernels are called dot product kernels because they are defined as a function of the dot product of the input vectors. Table 1 gives some example of these kind of kernels.

Kernel	**Definition**	**Coefficients** a_s
Linear	$\mathbf{x} \cdot \mathbf{y}$	$1, s = 1$
Polynomial	$(\mathbf{x} \cdot \mathbf{y} + c)^d$	$\binom{d}{n} c^{(d-s)}, \forall s \in [0, d]$
RBF	$e^{-\gamma \lVert \mathbf{x} - \mathbf{y} \rVert^2}$	$e^{-2\gamma \frac{(2\gamma)^{2s}}{s!}}, \forall s$
Tanimoto	$\frac{\mathbf{x} \cdot \mathbf{y}}{\lVert \mathbf{x} \rVert^2 + \lVert \mathbf{y} \rVert^2 - \mathbf{x} \cdot \mathbf{y}}$	$2^{-s}, \forall s > 0$

Table 1: Some examples of dot product kernels.

We can observe that the kernel matrices induced by these kernels are, in general, dense due to the zero degree term (i.e., $s = 0$) which is a constant added to all the entries. Adding a constant to a whole matrix means a space translation and we can demonstrate that this operation does not affect the margin in CF-KOMD (this is valid also in the generic formulation of CF-OMD).

Proof. Let $\mathbf{K} = \mathbf{K}_0 + \hat{\mathbf{K}}$ be a dot product kernel matrix where $\mathbf{K}_0$ is the constant matrix induced by the 0 degree term of the MacLaurin expansion (i.e. $s = 0$). Let also $\mathbf{q}_u$ be consequently defined as:

$$
\begin{aligned}
q_{ui} &= \frac{1}{m_u^-} \sum_{j \notin \mathcal{I}_u} \Big(K_0(\mathbf{x}_i, \mathbf{x}_j) + \hat{K}(\mathbf{x}_i, \mathbf{x}_j) \Big) && (6) \\
&= \frac{1}{m_u^-} \left[m_u^- \cdot k_0 + \sum_{j \notin \mathcal{I}_u} \hat{K}(\mathbf{x}_i, \mathbf{x}_j) \right] && (7) \\
&= k_0 + \hat{q}_{ui} \Rightarrow \mathbf{q}_u = \mathbf{k}_0 + \hat{\mathbf{q}}_u. && (8)
\end{aligned}
$$

We can rewrite the optimization problem (5) as (we omit the u^+ subscription for brevity):

$$\boldsymbol{\alpha}^* = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbf{A}_u} \; \boldsymbol{\alpha}^\top(\mathbf{K}_0 + \hat{\mathbf{K}})\boldsymbol{\alpha} + \lambda_p\|\boldsymbol{\alpha}\|^2 - 2\boldsymbol{\alpha}^\top(\mathbf{k}_0 + \hat{\mathbf{q}}_u) \tag{9}$$

$$= \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbf{A}_u} \; \boldsymbol{\alpha}^\top\mathbf{K}_0\boldsymbol{\alpha} + \boldsymbol{\alpha}^\top\hat{\mathbf{K}}\boldsymbol{\alpha} + \lambda_p\|\boldsymbol{\alpha}\|^2 - 2\boldsymbol{\alpha}^\top\mathbf{k}_0 - 2\boldsymbol{\alpha}^\top\hat{\mathbf{q}}_u \tag{10}$$

where both $\boldsymbol{\alpha}^\top\mathbf{K}_0\boldsymbol{\alpha}$ and $-2\boldsymbol{\alpha}^\top\mathbf{k}_0$ are constant values independent from $\boldsymbol{\alpha}$:

$$\boldsymbol{\alpha}^\top\mathbf{K}_0\boldsymbol{\alpha} = k_0\sum_{i\in\mathcal{I}_u}\sum_{j\in\mathcal{I}_u}\alpha_i\alpha_j = k_0\sum_{i\in\mathcal{I}_u}\alpha_i\sum_{j\in\mathcal{I}_u}\alpha_j = k_0;$$

$$-2\boldsymbol{\alpha}^\top\mathbf{k}_0 = -2\sum_{i\in\mathcal{I}_u}k_0\alpha_i = -2k_0\sum_{i\in\mathcal{I}_u}\alpha_i = -2k_0;$$

and hence the solution of the optimization problem does not depend on $\mathbf{K}_0$:

$$\boldsymbol{\alpha}^* = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbf{A}_u} \; \boldsymbol{\alpha}^\top\hat{\mathbf{K}}\boldsymbol{\alpha} + \lambda_p\|\boldsymbol{\alpha}\|^2 - 2\boldsymbol{\alpha}^\top\hat{\mathbf{q}}_u.$$

and it is the same as in (5). □

For this reason we can "sparsify" these kernels (we will call theme RDP Kernels: Reduced Dot Product Kernels) by removing the zero degree factor obtaining kernel matrices whose sparsities depend only on the distribution of the input data since they are defined as linear combination of powers of dot products. This also implies that the sparsity of a RDP kernel is exactly the same as in the simple linear kernel (i.e., $\mathbf{K} = \mathbf{X}^\top\mathbf{X}$).

5.1. Sparsity and long tail distribution

As mentioned in Section 4.2, CF datasets are, in most of the cases, very sparse and in general the distribution of the ratings has a long tail form from the items perspective [26]. This means that a small set of items, the most popular ones, receive great part of the whole set of ratings.

What we need to understand are the conditions under which a RDP kernel remains sufficiently sparse. To study this phenomenon we use the linear kernel, but results also apply for every other RDP kernel as well because they contain exactly the same zero entries as the linear one (which is in fact a special case of

RDP kernel).

Let $\mathbf{K} = \mathbf{X}^\top\mathbf{X}$ ($\mathbf{X} \in \mathbb{R}^{n\times m}$) be a kernel matrix and let $\mathbb{P}(K_{ij} \neq 0)$ be the probability that the entry K_{ij} is not zero. Given an a-priori probability distribution over the ratings, and assuming the independence of the ratings, we can estimate the probability of having a value different from zero in the kernel matrix with:

$$\begin{aligned}
\mathbb{P}(K_{ij} \neq 0) &= 1 - \mathbb{P}(K_{ij} = 0) &(11)\\
&= 1 - \prod_h \mathbb{P}(x_{ih} \cdot x_{jh} = 0) &(12)\\
&= 1 - \prod_h (1 - \mathbb{P}(x_{ih} \cdot x_{jh} \neq 0)) &(13)\\
&= 1 - \prod_h (1 - \mathbb{P}(x_{ih} \neq 0)\mathbb{P}(x_{jh} \neq 0)) &(14)\\
&= 1 - (1 - \mathbb{P}(x_{ih} \neq 0) \cdot \mathbb{P}(x_{jh} \neq 0))^n &(15)
\end{aligned}$$

where $\mathbb{P}(x_{ih} \neq 0)$ and $\mathbb{P}(x_{jh} \neq 0)$ are the probability of having a non zero entry in the rating matrix. It is worth to notice that this probability, in the uniform case, is actually the density of the matrix. However, it does not take into account the fact that all the elements in the diagonal of the kernel are for sure non zero. With this consideration in mind we can define an estimate of the kernel density $d(\mathbf{K})$, with $\mathbf{K} \in \mathbb{R}^{m\times m}$, as follows:

$$d(\mathbf{K}) = \frac{1}{m^2}\left[m + (m^2 - m)\mathbb{P}(K_{ij} \neq 0)\right].$$

Intuitively, we can argue that anytime both $\mathbf{x}_i$ and $\mathbf{x}_j$ are popular items, i.e., $\mathbb{P}(x_{ih} \neq 0)$ and $\mathbb{P}(x_{jh} \neq 0)$ are close to 1, then $\mathbb{P}(K_{ij} \neq 0)$ tends to be high and hence $\mathbf{K}$ is likely to be dense. On the contrary, when one of the two vectors represents an unpopular item, then the probability $\mathbb{P}(K_{ij} \neq 0)$ is likely close to zero. Since we are assuming a long tail distribution over the items, most of the kernel entries result from a dot product of two unpopular items with very few users that rated them and so the kernel tends to be sparse.

However, up to now, we are assuming a uniform distribution over the users and in real datasets this is often not the case. Any other probability distribution would highly affect our estimation $d(\mathbf{K})$. This is because, if we assume a long tail distribution over the users, the probability of having at least one user in common between two items would be generally high, since the ratings for an item are likely to be concentrated to the (few) most active users. Considering that a mathematical proof of this intuition is quite complicated, in the next section we provide an empirical analysis on real CF datasets.

5.1.1. Empirical analysis of CF datasets

In order to validate our thesis, we empirically analyze a set of famous CF datasets comparing the theoretical sparsity with uniform ratings distribution with the sparsity of the linear kernel. We expect that the long tail distribution of the ratings will tend to lower the likelihood of having a non zero value in the kernel matrix, with respect to the $d(\mathbf{K})$ estimate.

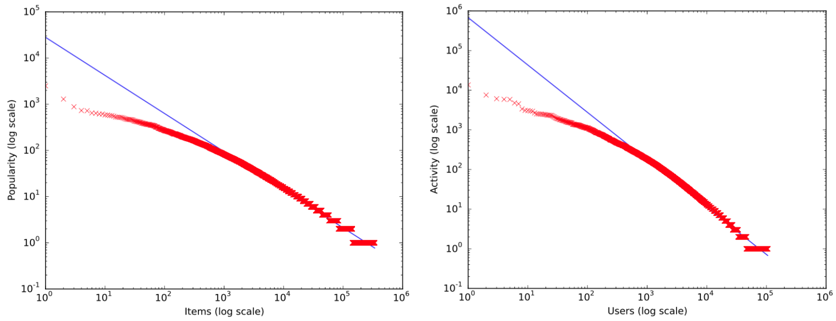
The empirical analysis has been performed as follows: for each dataset, we build the corresponding rating matrix $\mathbf{R}$, we calculate the expected density using (11) by fixing $\mathbb{P}(x_{ih} \neq 0)$ equals to the density of $\mathbf{R}$; we calculate the linear kernel $\mathbf{K} = \mathbf{R}^\top\mathbf{R}$ and finally we compare the sparsity of the kernel with $d(\mathbf{K})$. Table 2 summarizes the results.

Dataset	**Density R**	**Density K**	$d(\mathbf{K})$
Delicious	0.08%	0.128%	0.13%
LastFM	0.28%	0.85%	1.48%
Book-Crossing	0.003%	0.687%	0.011%
Movielens 10M	1.34%	85.56%	99.99%
Netflix	0.99%	98.03%	99.99%
Ciao	0.025%	2.51%	0.12%
FilmTrust	1.13%	11.11%	17.74%

Table 2: Analysis of the sparsity of the linear kernel.

Although our intuition seems to work with most of the datasets, we can notice that with the *Ciao* and *Book Crossing* datasets the kernels are more dense than the estimation $d(\mathbf{K})$.

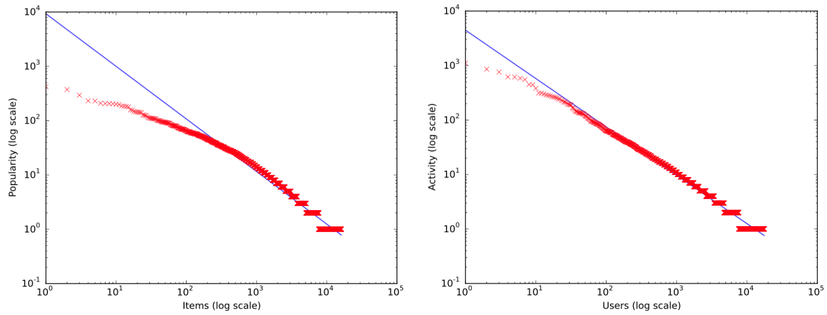
The reason why these datasets behave differently is clearly depicted in the plots 1 - 7. Every pair of plots show the distribution of the item popularity and the user activity. The plots have a loglog scale and the blue line represents the best fitting power low function. The fitting has been made using the least square method.



(a) Item Popularity

(b) User Activity

Figure 1: Book Crossing. The plots are in loglog scale.



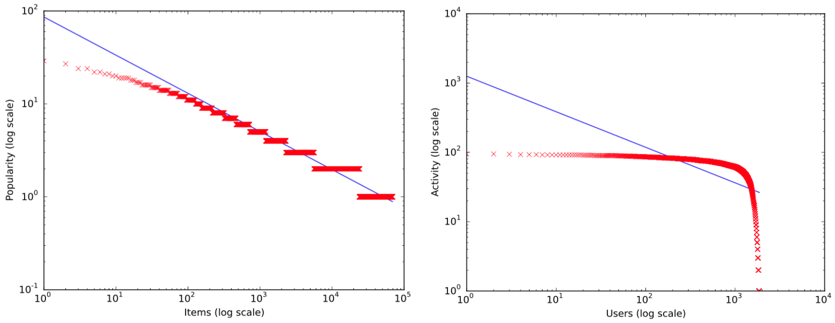
(a) Item Popularity

(b) User Activity

Figure 2: Ciao. The plots are in loglog scale.

From the plots we can observe that:

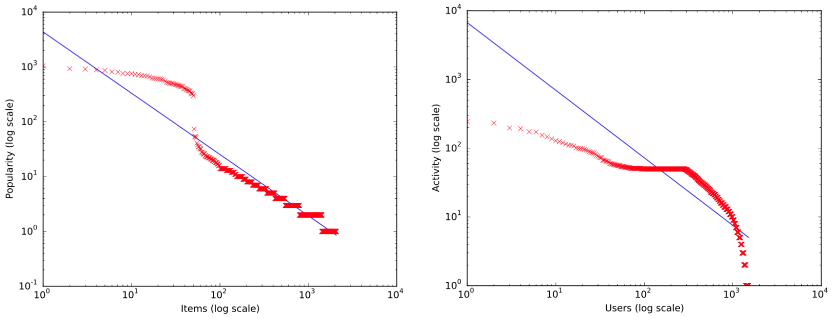
- none of the item distributions follows exactly a power low, especially in the head of the distribution;



(a) Item Popularity

(b) User Activity

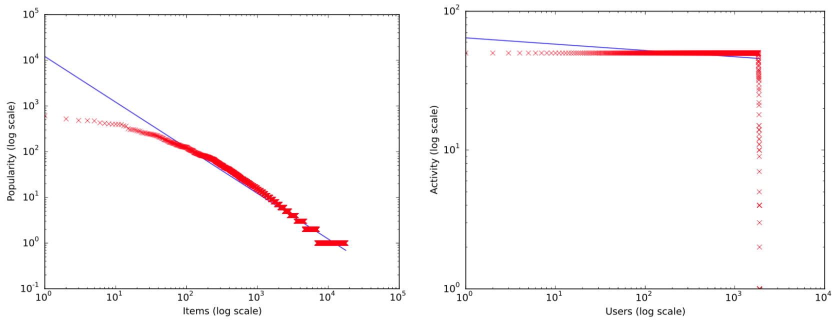
Figure 3: Delicious. The plots are in loglog scale.



(a) Item Popularity

(b) User Activity

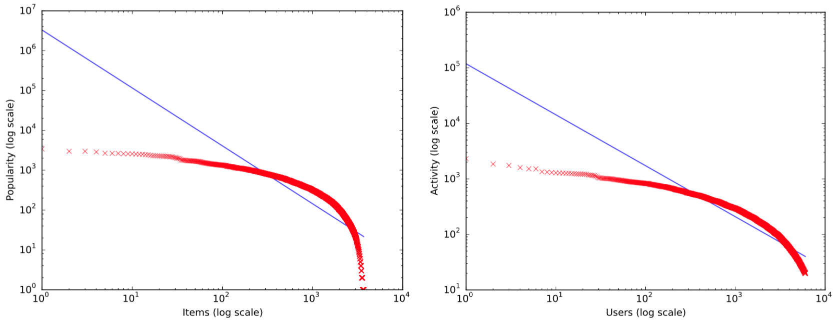
Figure 4: Film Trust. The plots are in loglog scale.



(a) Item Popularity

(b) User Activity

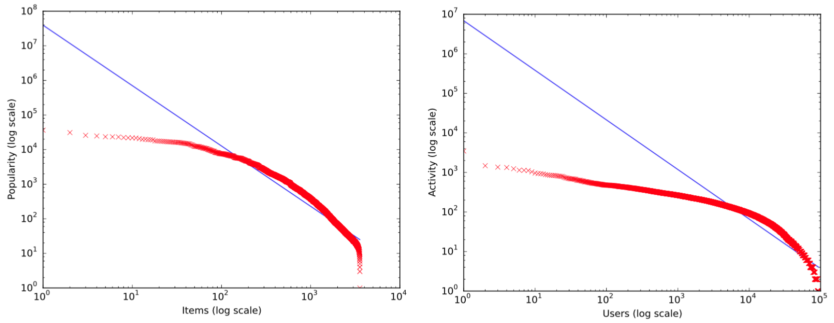
Figure 5: LastFM. The plots are in loglog scale.



(a) Item Popularity

(b) User Activity

Figure 6: Movielens. The plots are in loglog scale.



(a) Item Popularity

(b) User Activity

Figure 7: Netflix. The plots are in loglog scale.

- datasets with a very dense kernel tend to have shorter tails: the right part of the plot exceed the power low line;
- generally items are long tailed while users tend to be more uniform;
- users distributions are in general not well fitted by a power law, with the exception of *Ciao* and *Book Crossing* datasets.

The observations listed above point out that *Ciao* and *Book Crossing* are the only two datasets with a well defined long tail distribution over both users and items. This confirm our intuition about the likelihood of having a denser kernel with both long tailed distributions.

In conclusion, the long tail distribution over the items keeps the RDP kernels sparse while a long tail distribution over the users increase the density.

5.2. Approximation of $\mathbf{q}_u$

Using the RDP kernels, we can further optimize the complexity by providing a good approximation of $\mathbf{q}_u$ that can be computed only once, instead of n_{ts} times. The idea consists in replacing every q_{ui} with an estimate of $\mathbb{E}[K(\mathbf{x}_i, \mathbf{x})]$ which is the expected value of the kernel between the item i and every other items. Formally, consider, without any loss of generality, a normalized kernel function K and let the approximation of $\mathbf{q}_u$ be $\tilde{\mathbf{q}}$ such that:

$$\tilde{q}_i = \frac{1}{m} \sum_{j \in \mathcal{I}} K(\mathbf{x}_i, \mathbf{x}_j).$$

At each component of $\hat{\mathbf{q}}$, the approximation error is bounded by $\frac{2m_u^+}{m}$, which is linear on the sparsity of the dataset.

Proof.

$$|\hat{q}_i - q_{ui}| = \left| \frac{1}{m} \sum_{j \in \mathcal{I}} K(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{m_u^-} \sum_{j \notin \mathcal{I}_u} K(\mathbf{x}_i, \mathbf{x}_j) \right| \tag{16}$$

$$= \left| \frac{1}{m} \left[\sum_{j \in \mathcal{I}_u} K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{j \notin \mathcal{I}_u} K(\mathbf{x}_i, \mathbf{x}_j) \right] - \frac{1}{m_u^-} \sum_{j \notin \mathcal{I}_u} K(\mathbf{x}_i, \mathbf{x}_j) \right| \tag{17}$$

$$= \left| \frac{1}{m} \sum_{j \in \mathcal{I}_u} K(\mathbf{x}_i, \mathbf{x}_j) - \frac{m - m_u^-}{m \cdot m_u^-} \sum_{j \notin \mathcal{I}_u} K(\mathbf{x}_i, \mathbf{x}_j) \right| \tag{18}$$

$$\leq \left| \frac{1}{m} \sum_{j \in \mathcal{I}_u} K(\mathbf{x}_i, \mathbf{x}_j) \right| + \left| \frac{m - m_u^-}{m \cdot m_u^-} \sum_{j \notin \mathcal{I}_u} K(\mathbf{x}_i, \mathbf{x}_j) \right| \tag{19}$$

$$\leq \left| \frac{m_u^+}{m} \right| + \left| \frac{m - m_u^-}{m \cdot m_u^-} m_u^- \right| \leq \frac{m_u^+ + m - m_u^-}{m} = 2\frac{m_u^+}{m}. \tag{20}$$

□

6. Experiments and Results

Experiments have been performed comparing the proposed methods against the state-of-the-art method on MSD (MSDW) with respect to the ranking quality and computational performance.

We also compared our framework, in terms of AUC, against other state-of-the-art methods on top-N recommendation with implicit feedback, namely WRMF and BPR.

Our framework and MSDW are both implemented in Python[1][2][3], while for WRMF and BPR we used the java implementation provided by the open source LibRec library [4].

6.1. Datasets

In this section we introduce the datasets used in the experiments. Table 3 shows a brief description of the datasets.

Dataset	**Item type**	$\vert\mathcal{U}\vert$	$\vert\mathcal{I}\vert$	$\vert\mathcal{R}\vert$
MovieLens	Movies	6040	3706	1M
MSD	Music	1.2M	380K	48M
Ciao	General	17615	16121	72664
Netflix	Movie	93705	3561	3.3M
FilmTrust	Movie	1508	2071	35496

Table 3: Brief description of the used datasets.

The MSD dataset is used only to demonstrate the applicability of our kernel method to huge datasets. All the other datasets are used to compare our framework with the state-of-the-art method in top-N recommendation with implicit feedback.

6.2. Experimental setting

Experiments have been performed 5 times for each dataset. Datasets have been pre-processed as described in the following:

[1] We used CVXOPT package to solve the optimization problem

[2] The MSDW implementation is available at http://www.math.unipd.it/ aiolli/CODE/MSD/

[3] The framework implementation is available at https://github.com/makgyver/pyros

[4] http://www.librec.net/

1. we split randomly the users in 5 sets of the same dimension;
2. for each user in a set we further split its ratings in two halves;
3. at each round test, we use all the ratings in 4 sets of users plus the first half of ratings of the remaining set as training set, and the rest as test set.

This setting avoids situations of cold start for users, because in the training phase we have at least a rating for every user. We also force users with less than 5 ratings to be in the training set. The results reported below are the averages (with its standard deviations) over the 5 folds.

6.3. Evaluation measures

The rankings' evaluation metric used to compare the performances of the methods is the AUC (Area Under the receiver operating characteristic Curve) defined as in the following:

$$AUC = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}_u| \cdot |\mathcal{I} \setminus \mathcal{I}_u|} \sum_{i \in \mathcal{I}_u} \sum_{j \notin \mathcal{I}_u} \mathbb{I}[\hat{r}_{ui} > \hat{r}_{uj}]$$

where $\mathbb{I} : Bool \to \{0,1\}$ is the indicator function which returns 1 if the predicates is true 0 otherwise. In the experiments which use the MSD dataset we also compared the algorithms using the same metric as in the challenge, that is the Mean Average Precision at 500 (mAP@500), which is the mean over all users of the average precision at N ($N = 500$). Formally is defined as:

$$AP(\pi_u)@N = \frac{1}{\min(|\mathcal{I}_u|, N)} \sum_{k=1}^{N} P(\pi_u)@k \cdot \hat{r}_{u\pi_u(k)},$$

where $\mathcal{I}_u$ is the set of positive associated items with the user u, π_u is the items ranking for user u, such that $\pi_u(k) = i$ means that item i is ranked at position k, and $P(\pi_u)@k$ is the precision at k:

$$P(\pi_u)@k = \frac{1}{k} \sum_{p=1}^{k} \hat{r}_{u\pi_u(p)}.$$

6.4. Results

6.4.1. MSD

We used MSD as described in the Kaggle challenge[5]: the training set is composed by 1M users (plus 10K users as validation set) with all their listening history and for the rest (i.e., 100K users) only the first half of the history is provided, while the other half constitutes the test set. In these experiments we fixed the λ_p parameter to 0.01.

Results are presented in Table 4. In this case MSDW maintains its record performance in terms of mAP@500, while for the AUC all methods have very good results. This underline the fact that both ECF-OMD and CF-KOMD (polynomial kernel with $c = 1$) try to optimize the AUC rather than the mAP.

	MSDW (α, q)	ECF-OMD (λ_p)	CF-K (λ_p)
	$0.15, 3$	0.1	0.1
mAP@500	**0.16881**	0.16391	0.15967
AUC	**0.97342**	0.97034	0.97065

Table 4: Ranking accuracy on MSD using AUC and mAP@500.

The computational costs on this dataset are reported in Figure 8.

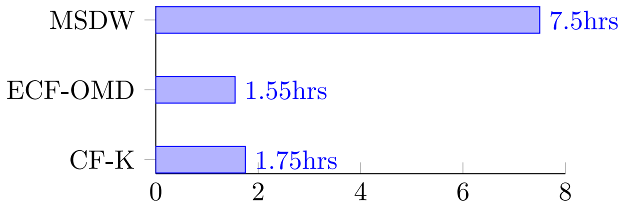


Figure 8: Average computational time in hours for 1K users.

The results are the average computing time over 1K test users. All methods run on a machine with 150Gb of RAM and 2 x Eight-Core Intel(R) Xeon(R) CPU E5-2680 0 @ 2.70GHz. Actually the times in Figure 8 have a constant overhead due to read operations. Results show that ECF-OMD and CF-K

[5] https://www.kaggle.com/c/msdchallenge

(abbreviation for CF-KOMD) are almost 5 time faster than MSDW even though they require more RAM to store the kernel matrix. It is worth to notice that CF-K has a computational time very close to ECF-OMD, and this highlights the positive effects of the complexity optimization presented in this paper.

6.4.2. Other datasets

This section shows the performance for the top-N recommendation task, in terms of AUC, achieved by our framework. We compared our methods with some state-of-the-art techniques. In particular, we used the following settings for each of the competing algorithm:

ECF-OMD : we fixed the regularization parameter $\lambda_p = 0.01$;

CF-K$_P$: it is the CF-KOMD method with the polynomial kernel. We tested different values for $c \in \{0.5, 1, 2, 4\}$ and we fixed $\lambda_p = 0.01$;

CF-K$_T$: it is the CF-KOMD method with the tanimoto kernel. We fixed the regularization parameter $\lambda_p = 0.01$;

MSDW : tests have been performed varying the value of the α parameter in the real range [0,1] with a step of 0.25;

WRMF : we reported only the performance achieved with $\alpha = 1$ since the results obtained with different α were substantially the same. We fixed the maximum number of iteration to 30, the learning rate $\rho = 0.001$, the regularization term $\lambda = 0.001$ and the number of factor $k = 100$;

BPR : we do not have free parameter. We fixed, as in WRMF, the maximum number of iteration to 30, the learning rate $\rho = 0.001$ and the regularization term $\lambda = 0.001$.

Table 5 summarizes the obtained results.

On the *MovieLens* dataset, methods of our framework have significant better performance against all the other, achieving an AUC of 0.896 with CF-KOMD with the polynomial kernel ($c = 4$). CF-KOMD has the higher AUC (0.964)

		MovieLens	**Netflix**	**FilmTrust**	**Ciao**
MSDW	$\alpha = 0.00$	$0.867_{\pm 0.001}$	$0.939_{\pm 0.0002}$	$0.961_{\pm 0.004}$	$\mathbf{0.824_{\pm 0.008}}$
MSDW	$\alpha = 0.25$	$0.870_{\pm 0.001}$	$0.939_{\pm 0.0002}$	$0.960_{\pm 0.005}$	$0.813_{\pm 0.01}$
MSDW	$\alpha = 0.50$	$0.875_{\pm 0.001}$	$0.936_{\pm 0.0006}$	$0.959_{\pm 0.005}$	$0.805_{\pm 0.01}$
MSDW	$\alpha = 0.75$	$0.862_{\pm 0.001}$	$0.912_{\pm 0.0001}$	$0.955_{\pm 0.005}$	$0.793_{\pm 0.008}$
MSDW	$\alpha = 1.00$	$0.458_{\pm 0.007}$	$0.409_{\pm 0.019}$	$0.833_{\pm 0.011}$	$0.784_{\pm 0.009}$
ECF-OMD	-	$0.895_{\pm 0.0003}$	$\mathbf{0.943_{\pm 0.001}}$	$0.961_{\pm 0.005}$	$0.718_{\pm 0.006}$
CF-K_P	$c = 0.50$	$0.893_{\pm 0.0003}$	$0.937_{\pm 0.001}$	$0.961_{\pm 0.006}$	$0.731_{\pm 0.003}$
CF-K_P	$c = 1.00$	$0.893_{\pm 0.0003}$	$0.937_{\pm 0.001}$	$0.960_{\pm 0.006}$	$0.730_{\pm 0.003}$
CF-K_P	$c = 2.00$	$0.894_{\pm 0.0003}$	$0.938_{\pm 0.001}$	$0.959_{\pm 0.006}$	$0.721_{\pm 0.003}$
CF-K_P	$c = 4.00$	$\mathbf{0.896_{\pm 0.0003}}$	$0.938_{\pm 0.001}$	$0.958_{\pm 0.006}$	$0.718_{\pm 0.003}$
CF-K_T	-	$0.895_{\pm 0.0004}$	$0.941_{\pm 0.001}$	$\mathbf{0.964_{\pm 0.005}}$	$0.734_{\pm 0.004}$
WRMF	$\alpha = 1.00$	$0.870_{\pm 0.006}$	$0.804_{\pm 0.002}$	$0.947_{\pm 0.007}$	$0.565_{\pm 0.004}$
BPR	-	$0.854_{\pm 0.005}$	$0.843_{\pm 0.001}$	$0.954_{\pm 0.008}$	$0.549_{\pm 0.004}$

Table 5: AUC results of our framework against state-of-the-art methods.

also in the *FilmTrust* dataset, but this time with the Tanimoto kernel. In this dataset, anyway, the polynomial achieved result comparable with ECF-OMD. Good results are also achieved by the MSDW. On the *Netflix* dataset, the best AUC is 0.943 by ECF-OMD. We got similar result with the Tanimoto kernel. Surprisingly, with the *Ciao* dataset, MSDW obtained very good result while all the other approaches are quite behind,n particular, WRMF and BPR have a very poor performances.

We also compared the execution time of the algorithm. All these experiments has been made on a MacBook Pro late 2012 with 16GB of RAM and CPU Intel(R) Core i7 @ 2.70GHz. The results are shown in Figure 9. The first thing we can notice is that the WRMF algorithm is always the most time consuming one. To keep the plot readable, we cut the bars longer than 100 minutes. Actually, in both *Netflix* and *Ciao* datasets WRMF took more than 7 hours

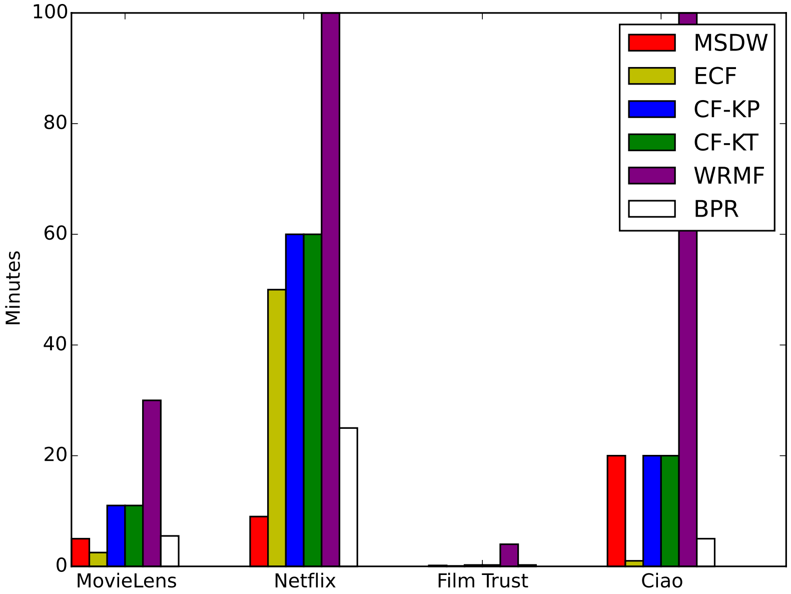


Figure 9: Execution time took by the tested methods.

(i.e., 420 minutes). The two kernel based methods, CF-K$_P$ and CF-K$_T$ took almost the same time and they are a little bit slower than the linear (ECF-OMD) method which is often the fastest one. MSDW is in general quite fast, but it seems to suffer when the number of items increase (e.g., *Ciao* dataset). These results show how our framework have, both in efficacy and efficiency, performances at the state-of-the-art.

7. Conclusions

In this paper we have proposed a collaborative filtering kernel-based method for the top-N recommendation. The method belongs to a more general framework, inspired by preference learning and designed to explicitly maximize the AUC. We have also proposed a strategy for the "sparsification" of the dot product kernels and in which conditions this strategy works.

Our analysis, conducted over CF datasets, have shown the effect of the long tail distribution on the sparsity of the kernel. Since this kind of distribution is very common, our results can apply in many domains other than CF. Finally, the experiments we reported have shown that the proposed kernel-based method achieve good result in terms of AUC and it is also efficient even with large scale datasets.

Acknowledge

This work was supported by the University of Padova under the strategic project BIOINFOGEN.

References

[1] F. Aiolli, Convex AUC optimization for top-N recommendation with implicit feedback, in: ACM Recommender Systems Conference, New York, USA, 2014, pp. 293–296.

[2] P. Kar, H. Karnick, Random feature maps for dot product kernels, in: N. D. Lawrence, M. A. Girolami (Eds.), Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12), Vol. 22, 2012, pp. 583–591.

[3] P. Singer, Not every long tail is power law! (2013).
URL `http://www.philippsinger.info/?p=247`

[4] M. Polato, F. Aiolli, Kernel based collaborative filtering for very large scale top-n item recommendation, in: Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 2016, pp. 11–16.

[5] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: ICDM, 2008, pp. 263–272.

[6] F. Aiolli, Efficient top-N recommendation for very large scale binary rated datasets, in: ACM Recommender Systems Conference, Hong Kong, China, 2013, pp. 273–280.

[7] S. Tan, J. Bu, C. Chen, X. He, Using Rich Social Media Information for Music Recommendation via Hypergraph Model, Springer London, London, 2011, pp. 213–237.

[8] B. Wang, C. Wang, J. Bu, C. Chen, W. V. Zhang, D. Cai, X. He, Whom to mention: Expand the diffusion of tweets by recommendation on micro-blogging systems, in: Proceedings of the 22Nd International Conference on World Wide Web, WWW '13, ACM, New York, NY, USA, 2013, pp. 1331–1340.

[9] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: WWW, 2001, pp. 285–295.

[10] B. McFee, T. Bertin-Mahieux, D. P. Ellis, G. R. Lanckriet, The million song dataset challenge, in: Proceedings of the 21st international conference companion on World Wide Web, WWW '12 Companion, ACM, New York, NY, USA, 2012, pp. 909–916. doi:10.1145/2187980.2188222.
URL http://doi.acm.org/10.1145/2187980.2188222

[11] M. Deshpande, G. Karypis, Item-based top-n recommendation algorithms, ACM Trans. Inf. Syst. 22 (1) (2004) 143–177.

[12] B. Wang, M. Ester, J. Bu, Y. Zhu, Z. Guan, D. Cai, Which to view: Personalized prioritization for broadcast emails, in: Proceedings of the 25th International Conference on World Wide Web, WWW '16, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 2016, pp. 1181–1190.

[13] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: Proceedings of

the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI '09, AUAI Press, Arlington, Virginia, United States, 2009, pp. 452–461.

[14] X. Ning, G. Karypis, SLIM: sparse linear methods for top-n recommender systems, in: 11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011, 2011, pp. 497–506. doi:10.1109/ICDM.2011.134.

[15] C. Zhe, Q. Tao, L. Tie-Yan, T. Ming-Feng, L. Hang, Learning to rank: From pairwise approach to listwise approach, Tech. rep. (April 2007).

[16] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in: Proceedings of the 22Nd International Conference on Machine Learning, ICML '05, ACM, New York, NY, USA, 2005, pp. 89–96.

[17] V. C. Ostuni, T. Di Noia, E. Di Sciascio, R. Mirizzi, Top-n recommendations from implicit feedback leveraging linked open data, in: Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, ACM, New York, NY, USA, 2013, pp. 85–92.

[18] H. Zhong, W. Pan, C. Xu, Z. Yin, Z. Ming, Adaptive pairwise preference learning for collaborative recommendation with implicit feedbacks, in: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM '14, ACM, New York, NY, USA, 2014, pp. 1999–2002.

[19] S. Rendle, C. Freudenthaler, Improving pairwise learning for item recommendation from implicit feedback, in: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, WSDM '14, ACM, New York, NY, USA, 2014, pp. 273–282.

[20] Y. Shi, M. Larson, A. Hanjalic, List-wise learning to rank with matrix factorization for collaborative filtering, in: Proceedings of the Fourth ACM

Conference on Recommender Systems, RecSys '10, ACM, New York, NY, USA, 2010, pp. 269–272.

[21] S. Huang, S. Wang, T.-Y. Liu, J. Ma, Z. Chen, J. Veijalainen, Listwise collaborative filtering, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, ACM, New York, NY, USA, 2015, pp. 343–352.

[22] F. Aiolli, A preference model for structured supervised learning tasks, IEEE International Conference on Data Mining (2005) 557–560.

[23] F. Aiolli, G. Da San Martino, A. Sperduti, A Kernel Method for the Optimization of the Margin Distribution, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 305–314.

[24] M. Donini, F. Aiolli, Learning deep kernels in the space of dot product polynomials, in: Machine Learning, 2017 (in press).

[25] B. Scholkopf, A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, USA, 2001.

[26] C. Anderson, The Long Tail: Why the Future of Business Is Selling Less of More, Hyperion, 2006.