Kernel based collaborative filtering for very large scale top-N item recommendation

Mirko Polato and 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 how to generalize it to other kernels preserving efficiency. We compare our method with the state-of-the-art algorithm on the Million Songs Dataset achieving an execution about 5 time faster, while having comparable effectiveness.

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 $\mathbf{R} = \{r_{ui}\} \in \mathbb{R}^{n \times m}$. In this paper we focus on implicit feedbacectk, and so we assume binary ratings, $r_{ui} \in \{0, 1\}$, where $r_{ui} = 1$ means that user u interacted with item i (unambiguous feedback) and $r_{ui} = 0$ means there is not evidence that user u interacted with item i (ambiguous feedback). Unlike traditional CF algorithms for explicit feedback, where one wants to accurately predict ratings for each unseen user-item pair, the goal in the implicit feedback domain is to generate a top-N ranking of items. Top-N recommendation with implicit feedback was the subject of one recent remarkable challenge organized by Kaggle, the Million Songs Dataset challenge [1], that was defined on a very large dataset with roughly 1.1M users and 380K items (i.e., songs) for a total of about 50M ratings. The winning solution described in [2] (here called MSDW) is an extension of the well known item-based nearest-neighbors (NN) algorithm [3] that uses an asymmetric similarity measure, called asymmetric cosine. 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. More recently, a new principled algorithm for CF (CF-OMD) which explicitly optimizes the AUC has been proposed with very nice performances on the MovieLens dataset [4]. Unfortunately, this last algorithm cannot be promptly applied to large datasets as it requires the optimization of n quadratic problems each one defined on m variables.

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

Here, we propose a variant of the CF-OMD algorithm that makes it applicable to very large datasets achieving an execution time about 5 times faster than the MSDW algorithm on the MSD dataset. Secondly, we present strategies that allow the same algorithm to be applied with quite general kernels without loss in efficiency.

2 CF-OMD (Optimization of the Margin Distribution)

In this section we present a CF algorithm, called CF-OMD [4], for top-N recommendation inspired by preference learning, and designed to explicitly maximize the AUC (Area Under the ROC Curve) . Consider the normalized rating matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$, with columns $\mathbf{x}_i = \mathbf{r}_i / \|\mathbf{r}_i\|$ and let $\mathcal{I}_u$ be the set of items rated by the user u. Let also define the probability distribution over the positive and negative items for u as $\mathbf{\Gamma}_u = \{\boldsymbol{\alpha}_u \in \mathbb{R}_+^m | \sum_{i \in \mathcal{I}_u} \alpha_{ui} = 1, \sum_{i \notin \mathcal{I}_u} \alpha_{ui} = 1\}$. Then, for each test user, the following convex optimization problem has to be solved:

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

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, and $\mathbf{\Lambda}$ is a diagonal matrix such that $\mathbf{\Lambda}_{ii} = \lambda_p$ if $i \in \mathcal{I}_u$, otherwise $\mathbf{\Lambda}_{ii} = \lambda_n$, where λ_p and λ_n are regularization parameters. These parameters balance the contribution of the unambiguous ratings (λ_p) and the ambiguous ones (λ_n). Once solved the optimization problem the scores of the user u is calculated by $\hat{\mathbf{r}}_u = \mathbf{X}^\top \mathbf{X} \mathbf{Y}_u \boldsymbol{\alpha}_u^*$, and the recommendation is made accordingly.

Although this algorithm has shown state-of-the-art results in terms of AUC, it is not suitable to deal with large dataset. 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 $O(10^{19})$.

3 Efficient CF-OMD

Analyzing the results reported in [4], the authors 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 $\forall i \notin \mathcal{I}_u, \alpha_{ui} = 1/m_u^-$:

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

where $\boldsymbol{\alpha}_{u^+}$ 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 and $\boldsymbol{\mu}_u^- = (\sum_{i \notin \mathcal{I}_u} \mathbf{x}_i)/m_u^-$ is the centroid of the convex hull spanned by the negative items. The number of parameters in (2) is m_u^+ and hence the complexity from $O(n_{ts} m^2)$ is dropped to $O(n_{ts} \overline{m}_u^{+\,2})$, where $\overline{m}_u^+ = \mathbb{E}[|\mathcal{I}_u|]$. In MSD $\overline{m}_u^+ \approx 47.46$ which leads to a complexity $O(10^8)$.

3.1 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.

4 Kernelized CF-OMD

The method proposed in Section 3, 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 $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Given K we can reformulate (2) as:

$$\boldsymbol{\alpha}_{u^+}^* = \underset{\boldsymbol{\alpha}_{u^+} \in \boldsymbol{\Gamma}_u}{\operatorname{argmin}} \quad \boldsymbol{\alpha}_{u^+}^\top \mathbf{K}_{u^+} \boldsymbol{\alpha}_{u^+} + \lambda_p \|\boldsymbol{\alpha}_{u^+}\|^2 - 2 \boldsymbol{\alpha}_{u^+}^\top \mathbf{q}_u, \tag{3}$$

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

Actually, inside the optimization problem (3) we can plug any kernel function. We will refer to this method as CF-KOMD. Generally speaking, the application of kernel methods on 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 an important result to keep the kernel as sparse as possible without changing the solution of CF-KOMD. In [5] Karnick et al. observed that: if a function $f : \mathbb{R} \to \mathbb{R}$ admits a Maclaurin expansion with only nonnegative coefficients i.e., $\sum_{n=0}^{\infty} a_n x^n, a_n \geq 0$, then it defines a positive definite kernel as $K : (\mathbf{x}, \mathbf{y}) \mapsto f(\langle \mathbf{x}, \mathbf{y} \rangle)$. As emphasized in [5], many kernels used in practice [6] satisfy the above-mentioned condition.

Consider the application of this result on the polynomial kernel $K_p : (\mathbf{x}, \mathbf{y}) \mapsto (\langle \mathbf{x}, \mathbf{y} \rangle + c)^d$ where $c \in \mathbb{R}$ and $d \in \mathbb{N}$. K_p can be defined as:

$$K_p(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{d} \binom{d}{i} c^{(d-i)} \langle \mathbf{x}, \mathbf{y} \rangle^i. \tag{4}$$

When the polynomial is not homogeneous (i.e., $c \neq 0$) the kernel matrix induced by K_p is dense due to the zero degree term (i.e., c^d) which is added to all entries. Since adding a constant to a whole matrix means a space translation, it can be demonstrated that this operation does not affect the margin in CF-KOMD. For this reason we can "sparsify" the kernel by removing the factor c^d obtaining a kernel matrix whose sparsity depends on the distribution of the input data.

Let $\mathbf{K} = \mathbf{X}^\top \mathbf{X}$ be a kernel matrix and let $\mathbb{P}(K_{ij} \neq 0)$ be the probability that the entry $\mathbf{K}_{ij}$ is not zero. Given the a-priori probabilities $\mathbb{P}(x_{ih} \neq 0)$ and $\mathbb{P}(x_{jh} \neq 0)$, we can say that $\mathbb{P}(K_{ij} \neq 0) = 1 - (1 - \mathbb{P}(x_{ih} \neq 0) \cdot \mathbb{P}(x_{jh} \neq 0))^n$. 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 high, then $\mathbb{P}(K_{ij} \neq 0)$ tends to be high as well. On the contrary, when one of the two vectors represents an unpopular item then the probability $\mathbb{P}(K_{ij} \neq 0)$ goes to zero. In CF contexts this situation is pushed towards the limit since the popularity distribution generally follows a power low, and this often guarantees the sparsity of the resulting kernel .

Using the "sparsified" kernel, 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 $\mathbb{E}[K(\mathbf{x}_i, \mathbf{x})]$. Formally, consider, without any loss of generality, a normalized kernel function K and let the approximation of $\mathbf{q}_u$ be $\hat{\mathbf{q}}$ s.t. $\hat{q}_i = \frac{1}{m} \sum_{j \in \mathcal{I}} K(\mathbf{x}_i, \mathbf{x}_j)$. At each component of $\hat{\mathbf{q}}$, the approximation error is bounded by $\frac{2m_u^+}{m}$ (see Appendix A), which is linear on the sparsity of the dataset.

5 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 used two datasets: MSD, described in Section 1, and Movielens, which consists of 3850 users and 2273 items for a total of 315K ratings. Methods have been compared using the mAP [2] and AUC measures. All methods have been implemented in Python[1][2]. In this section we will refer to the Efficient CF-OMD with ECF-OMD and to the Kernelized CF-OMD with CF-K.

5.1 Movielens dataset

The Movielens dataset has been randomly divided into a training set of roughly 250K ratings and a test set of 60K ratings. Since this dataset contains ratings in

[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/

the form of a 5 stars preference, we had to convert them into binary ones where all values greater than 0 are treated as 1. This test aims to show the accuracy and the computational performance of the proposed methods on a medium size dataset. Table 1 summarizes the results.

	MSDW (α)		ECF-OMD (λ_p)			CF-K (λ_p)
	0.15	0.5	0.01	0.1	1	0.1
mAP@100	0.10369	0.11262	0.13172	0.13224	0.13429	**0.13505**
AUC	0.87542	0.87162	0.89610	0.89619	0.89554	**0.89665**

Table 1: Ranking accuracy on Movielens dataset using AUC and mAP@100.

We tested MSDW fixing the locality parameter [2] $q = 1$ and varying the asymmetric cosine weight α. For ECF-OMD we tried different λ_p but its effect is minimal on the final ranking, and for this reason we fixed it during the CF-K experiment. In this experiment we used the polynomial kernel of degree 2 with $c = 1$. Results show that both proposed methods have higher AUC and mAP@100 with a slightly better performance for CF-K. With this dataset all methods terminate in few seconds.

5.2 MSD

We used MSD as described in the Kaggle challenge[3]: 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 the best performing one on the Movielens dataset.

Results are presented in Table 2. 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-K 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 2: Ranking accuracy on MSD using AUC and mAP@500.

The computational costs on this dataset are reported in Figure 1.

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 1 have a constant overhead due to read operations. Results show that ECF-OMD and CF-K 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

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

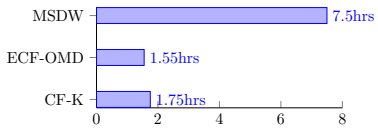

Fig. 1: Average computational time in hours for 1K users.

very close to ECF-OMD, and this highlights the positive effects of the complexity optimization presented in this paper.

A Appendix

A.1 Optimization problem simplification

Let $\boldsymbol{\mu}_u^-$ defined as in Sec.3 and let $\mathbf{X}_{u^+}$, $\mathbf{X}_{u^-}$ be the sub-matrices of $\mathbf{X}$ containing only the columns corresponding, respectively, to the positive and negative items for u. Then, by fixing $\lambda_n = +\infty$, we can simplify (1) as:

$$\begin{aligned}\boldsymbol{\alpha}_u^* &= \underset{\boldsymbol{\alpha}_u}{\operatorname{argmin}} \ \|\boldsymbol{\alpha}_{u^+}^\top \mathbf{X}_{u^+} - \boldsymbol{\mu}_u^-\|^2 + \lambda_p \|\boldsymbol{\alpha}_{u^+}\|^2 \\ &= \underset{\boldsymbol{\alpha}_u}{\operatorname{argmin}} \ \|\boldsymbol{\alpha}_{u^+}^\top \mathbf{X}_{u^+}\|^2 - \|\boldsymbol{\mu}_u^-\|^2 - 2\boldsymbol{\alpha}_{u^+}^\top \mathbf{X}_{u^+}^\top \boldsymbol{\mu}_u^- + \lambda_p \|\boldsymbol{\alpha}_{u^+}\|^2 = (2)\end{aligned}$$

A.2 Approximation error

Let $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, then:

$$\begin{aligned}|\hat{q}_i - q_{ui}| &= \left| \frac{1}{m} \sum_{j \in \mathcal{I}} K_{ij} - \frac{1}{m_u^-} \sum_{j \notin \mathcal{I}_u} K_{ij} \right| = \left| \frac{1}{m} \left[\sum_{j \in \mathcal{I}_u} K_{ij} + \sum_{j \notin \mathcal{I}_u} K_{ij} \right] - \frac{1}{m_u^-} \sum_{j \notin \mathcal{I}_u} K_{ij} \right| \\ &= \left| \frac{1}{m} \sum_{j \in \mathcal{I}_u} K_{ij} - \frac{m - m_u^-}{m \cdot m_u^-} \sum_{j \notin \mathcal{I}_u} K_{ij} \right| \leq \left| \frac{1}{m} \sum_{j \in \mathcal{I}_u} K_{ij} \right| + \left| \frac{m - m_u^-}{m \cdot m_u^-} \sum_{j \notin \mathcal{I}_u} K_{ij} \right| \\ &\leq \left| \frac{m_u^+}{m} \right| + \left| \frac{m - m_u^-}{m \cdot m_u^-} m_u^- \right| \leq \frac{m_u^+ + m - m_u^-}{m} = \frac{2m_u^+}{m}.\end{aligned}$$

References

[1] Brian McFee, Thierry Bertin-Mahieux, Daniel P.W. Ellis, and Gert R.G. Lanckriet. The million song dataset challenge. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 909–916, New York, NY, USA, 2012. ACM.

[2] Fabio Aiolli. Efficient top-N recommendation for very large scale binary rated datasets. In *ACM Recommender Systems Conference*, pages 273–280, Hong Kong, China, 2013.

[3] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[4] Fabio Aiolli. Convex AUC optimization for top-N recommendation with implicit feedback. In *ACM Recommender Systems Conference*, pages 293–296, New York, USA, 2014.

[5] Purushottam Kar and Harish Karnick. Random feature maps for dot product kernels. In Neil D. Lawrence and Mark A. Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12)*, volume 22, pages 583–591, 2012.

[6] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.