Fast Collaborative Filtering from Implicit Feedback with Provable Guarantees

Sayantan Dasgupta
Independent Researcher

Abstract

Building recommendation algorithm is one of the most challenging tasks in Machine Learning. Although most of the recommendation systems are built on explicit feedback available from the users in terms of rating or text, a majority of the applications do not receive such feedback. Here we consider the recommendation task where the only available data is the records of user-item interaction over web applications over time, in terms of subscription or purchase of items; this is known as implicit feedback recommendation. There is usually a massive amount of such user-item interaction available for any web applications. Algorithms like PLSI or Matrix Factorization runs several iterations through the dataset and may prove very expensive for large datasets. Here we propose a recommendation algorithm based on Method of Moment, which involves factorization of second and third order moments of the dataset. Our algorithm can be proven to be globally convergent using PAC learning theory. Further, we show how to extract the parameters using only three passes through the entire dataset. This results in a highly scalable algorithm that scales up to million of users even on a machine with a single-core processor and 8 GB RAM and produces competitive performance in comparison with existing algorithms.

Keywords: Computational Learning; Probably Approximately Correct (PAC); Collaborative Filtering; Implicit Feedback; Moment Factorization; Personalization

1. Introduction

Recommendation Systems came into the spotlight through the Netflix One-Million challenge. Most of the early recommendation systems were built using features extracted from the content of the items. These are known as content-based recommendation systems, and they typically fail to capture the user opinion. Collaborative filtering was introduced to mine user feedbacks to overcome the limitation of content-based filtering. Collaborative filtering mostly relies on the availability of user feedback, either in the form of numeric rating, or text, or even through binary 'like' or 'unlike' tags. However, not all applications receive such explicit feedback from users.

Most of the web-based applications receive a significant amount of user traffic. The users interact with different items in the web applications, although they may not always rate the items. The web usage data containing user-item interactions can effectively be mined to build recommendation systems. Also, in applications where a user provides rating or feedback, such as Netflix, he/she rates only a small subset of movies watched. A user may simply avoid rating some of his favourite movies due to the lack of time, and there is no way to know about his interest in those movies except web-usage data of implicit...
Fast Collaborative Filtering

<table>
<thead>
<tr>
<th>User</th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>Item 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>✔</td>
<td>✔</td>
<td>?</td>
<td>✔</td>
<td>?</td>
</tr>
<tr>
<td>User 2</td>
<td>?</td>
<td>✔</td>
<td>?</td>
<td>?</td>
<td>✔</td>
</tr>
<tr>
<td>User 3</td>
<td>?</td>
<td>?</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>User 4</td>
<td>✔</td>
<td>?</td>
<td>✔</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Figure 1: Implicit User-Item Interaction

user-item interaction. Also, the amount of web usage data for such applications is far larger than the amount of rating data available from users, and mining these data can provide an improvement on recommendations drawn only from user ratings. Please note that binary 'like' or 'dislike' tags provided by users are a form of explicit feedback, such as the case of (Wang and Blei, 2011). We do not attempt to build a recommendation algorithm based on user tags here. An appropriate visualization of our recommendation problem in the line of (Wang and Blei, 2011) is shown in Figure 1.

The most common algorithm used by practitioners to build recommendation systems based on implicit feedback is Probabilistic Latent Semantic Indexing (PLSI), such as in personalized ranking of search results (Lin et al., 2005) or personalized news recommendation (Das et al., 2007). However, PLSI trains using EM algorithm that suffers from local maxima problem. Therefore, these recommendation systems more often or so do not give optimal performance. Recent literature on recommendation systems includes different algorithms for implicit feedback dataset, although most of them are tested on datasets of limited size. (Hu et al., 2008) adapts the well-known matrix factorization algorithm for implicit feedback datasets through a weighted matrix factorization (WRMF). The algorithm scans through the entire dataset during every iteration until convergence, and it may prove computationally very expensive for a large volume of user logs stored across multiple nodes in a distributed ecosystem. Bayesian Personalized Ranking (BPR) (Rendle et al., 2009) uses a stochastic approach to sample negative items for each user, and reduces the computation time significantly. There are other algorithms in the literature, which are extensions of these matrix factorization methods. GBPR (Pan and Chen, 2013) builds on BPR and incorporates group preference into it. LorSLIM (Cheng et al., 2014) uses a low rank sparse linear method for implicit feedback datasets. AdaBPR (Liu et al., 2015) introduces a boosting technique to improve on BPR loss. These algorithms are found to outperform other methods such as similarity or neighbourhood based methods.

There have been recent developments in non-iterative learning algorithm based on Method of Moments (MoM) (Anandkumar et al., 2014), also referred to as Spectral Methods in the literature. Unlike traditional clustering algorithms that try to maximize likelihood or minimize cost through iterative steps, MoM attempts to learn the parameters through factorization of higher order moments of the data. It is a non-iterative algorithm and offers much better scalability than iterative counterparts, especially for large datasets. Here we use Method of Moments on the same generative latent variable used by PLSI (Hofmann, 2004), and show how to extract the parameters through factorization of moments of the data. We demonstrate the derivation of our algorithm in next section, prove its conver-
gence bounds, and then compare the performance of our algorithm with PLSI and matrix factorization on real-life datasets.

2. Latent Variable Model

Our method retains the same latent variable structure from PLSI (Hofmann, 2004). However, instead of using EM algorithm, we extract the parameters by factorizing second and third order moments of the dataset.

2.1. Generative Model

Let us assume that there are $U$ users and $D$ items, and the latent variable $h$ can assume $K$ states. For any user $u \in \{u_1, u_2 \ldots u_U\}$, if $n_u$ is the number of items associated with $n_u$, then we first choose a latent state of $h \in \{1, 2 \ldots K\}$ from the discrete distribution $P[h|u]$, then we choose an item $y \in \{y_1, y_2 \ldots y_D\}$ from the discrete distribution $P[y|h]$, and repeat it for $n_u$ times. The final sample $x \in \mathbb{R}^D$ for user $u$ contains counts of different items, which sums to $n_u$.

The generative process is as follows.

For every user $u \in \{u_1 \ldots u_N\}$, repeat for $n_u$ times:

$$
\begin{align*}
  h &\sim \text{Discrete}(P[h|u]) \\
  y &\sim \text{Discrete}(P[y|h])
\end{align*}
$$

(1)

Let us denote the probability of the latent variable $h$ assuming the state $k \in 1 \ldots K$ as,

$$
\pi_k = P[h = k]
$$

(2)

Let us define $\bar{\mu}_k \in \mathbb{R}^D$ as the probability vector of all the items conditional to the latent state $k \in 1 \ldots K$, i.e.

$$
\bar{\mu}_k = P[y|h = k]
$$

(3)

Let the matrix $O \in \mathbb{R}^{D \times K}$ denote the conditional probabilities for the items, i.e. $O_{i,k} = P[y_i|h = k]$. Then $O = [\bar{\mu}_1 | \bar{\mu}_2 | \ldots | \bar{\mu}_K]$. We assume that the matrix $O$ is of full rank, and the columns of $O$ are fully identifiable. The aim of our algorithm is to estimate the matrix $O$ as well as the vector $\pi$, and then derive the user personalization parameters $P[h = k|u]$ from them.

Following the generative model in equation 1, we can define the probability of individual item as,

$$
P[y_j] = \sum_{k=1}^{K} P[y_j|h]P[h = k] = \sum_{k=1}^{K} \mu_{kj} \pi_k, \forall j = 1, 2, \ldots D
$$

Therefore, the average probability of the items across the data can be defined as,

$$
M_1 = P[y_1, y_2, \ldots y_D]^\top = \sum_{k=1}^{K} \pi_k [\mu_{k1}, \mu_{k2} \ldots \mu_{kD}]^\top = \sum_{k=1}^{K} \pi_k \bar{\mu}_k
$$

(4)
Now, we try to formulate the matrix of the joint probability distribution of the items. Let us assume that we choose two items $w_1$ and $w_2$ from the list of any user. Please note that, $P[w_1 = y_j]$ or $P[w_2 = y_j]$ is same as $P[y_j]$ for any item $y_j$. Also, $P[w_1 = y_i, w_2 = y_j]$ is same as $P[y_i, y_j]$ for any two items $y_i$ and $y_j$. Therefore,

$$P[y_i, y_j] = P[w_1 = y_i, w_2 = y_j]$$

$$= \sum_{k=1}^{K} P[w_1 = y_i, w_2 = y_j|h = k]P[h = k]$$

$$= \sum_{k=1}^{K} P[w_1 = y_i|h = k]P[w_2 = y_j|h = k]P[h = k]$$

$$= \sum_{k=1}^{K} P[y_i|h = k]P[y_j|h = k]P[h = k]$$

$$= \sum_{k=1}^{K} \mu_{k_i} \mu_{k_j} \pi_k \quad \forall i, j \in \{1, 2 \ldots D\}$$

Defining $M_2$ as the pairwise probability matrix, with $M_{2ij} = P[y_i, y_j]$, we can express it as,

$$M_2 = \sum_{k=1}^{K} \pi_k \mu_k \mu_k^\top = \sum_{k=1}^{K} \pi_k \mu_k \otimes \mu_k$$  \hfill (5)

Similarly, the tensor $M_3$ defined as the third order probability moment, with $M_{3ijk} = P[y_i, y_j, y_l]$, can be represented as,

$$M_3 = \sum_{k=1}^{K} \pi_k \mu_k \otimes \mu_k \otimes \mu_k$$  \hfill (6)

### 2.2. Parameter Extraction

The first step of parameter extraction is to whiten the matrix $M_2$, where we try to find a low-rank matrix $W$ such that $W^\top M_2 W = I$. This is a method similar to the whitening in ICA, with the covariance matrix replaced with the co-occurrence probability matrix in our case.

The whitening is usually done through eigenvalue decomposition of $M_2$. If the $K$ maximum eigenvalues of $M_2$ are $\{\nu_k\}_{k=1}^{K}$, and the corresponding eigenvectors are $\{\omega_k\}_{k=1}^{K}$, then the whitening matrix of rank $K$ is computed as $W = \Omega \Sigma^{-1/2}$, where $\Omega = [\omega_1 | \omega_2 | \ldots | \omega_K]$, $\& \Sigma = \text{diag} (\nu_1, \ldots, \nu_K)$.

Upon whitening $M_2$ takes the form

$$W^\top M_2 W = W^\top \left( \sum_{k=1}^{K} \pi_k \mu_k \mu_k^\top \right) W = \sum_{k=1}^{K} \left( \sqrt{\pi_k} W^\top \mu_k \right) \left( \sqrt{\pi_k} W^\top \mu_k \right)^\top = \sum_{k=1}^{K} \mu_k \mu_k^\top = I$$  \hfill (7)
Hence $\tilde{\mu}_k = \sqrt{\pi_k}W^T\mu_k$ are orthonormal vectors. Multiplying $M_3$ along all three dimensions by $W$, we get

$$\tilde{M}_3 = M_3(W, W, W) = \sum_{k=1}^{K} \pi_k(W^T\tilde{\mu}_k) \otimes (W^T\tilde{\mu}_k) \otimes (W^T\tilde{\mu}_k) = \sum_{k=1}^{K} \frac{1}{\sqrt{\pi_k}} \tilde{\mu}_k \otimes \tilde{\mu}_k \otimes \tilde{\mu}_k \quad (8)$$

Upon canonical decomposition of $\tilde{M}_3$, if the eigenvalues and eigenvectors are $\{\lambda_k\}_{k=1}^{K}$ and $\{v_k\}_{k=1}^{K}$ respectively, then $\lambda_k = 1/\sqrt{\pi_k}$, i.e., $\pi_k = \lambda_k^{-2}$, and,

$$v_k = \tilde{\mu}_k = \sqrt{\pi_k}W^T\tilde{\mu}_k = \frac{1}{\lambda_k}W^T\tilde{\mu}_k \quad (9)$$

The $\tilde{\mu}_k$s can be recovered as $\tilde{\mu}_k = \lambda_kW^Tv_k$, where $W^\dagger$ is the pseudo-inverse of $W^T$, i.e., $W^\dagger = W(W^TW)^{-1}$. Since we normalize the columns of $O$ as $O_{yk} = \frac{O_{yk}}{\sum_{y}O_{yk}}$, it is sufficient to compute $\tilde{\mu}_k = W^\dagger u_k$, since $\lambda_k$ will be cancelled during normalization. The matrix $O$ can be constructed as $O = [\tilde{\mu}_1|\tilde{\mu}_2|\cdots|\tilde{\mu}_K]$. 

### 2.3. User Personalization

Once we have $O$ and $\pi$, the probability of a user $u \in \{u_1, u_2 \ldots u_N\}$ given $h$ can be expressed as,

$$P[u|h = k] = \prod_{y \in Y_u} P[y|h = k] \quad (10)$$

where $Y_u$ is the list of items selected by the user $u$ in the training set.

Then the user personalization probabilities $P[h = k|u]$ can be estimated using Bayes Rule.

$$P[h = k|u] = \frac{P[h = k] \prod_{y \in Y_u} P[y|h = k]}{\sum_{k=1}^{K} P[h = k] \prod_{y \in Y_u} P[y|h = k]} = \frac{\pi_k \prod_{y \in Y_u} O_{yk}}{\sum_{k=1}^{K} \pi_k \prod_{y \in Y_u} O_{yk}} \quad (11)$$

Finally, we compute the probability of a user $\tilde{u}$ selecting an item $\tilde{y}$ by the following equation, and recommend the items with the highest probability for the user $\tilde{u}$.

$$P[\tilde{y}|\tilde{u}] = \sum_{k=1}^{K} P[\tilde{y}|h = k] P[h = k|\tilde{u}] = \frac{\sum_{k=1}^{K} \pi_k O_{\tilde{y}k} \prod_{y \in Y_u} O_{yk}}{\sum_{k=1}^{K} \pi_k \prod_{y \in Y_u} O_{yk}} \quad (12)$$

Please note that although we use the same latent variable model as PLSI (Hofmann, 2004), our model parameters are only $O$ and $\pi$. Therefore our number of effective parameters is only $(D - 1)K + (K - 1)$, unlike the case of PLSI that uses $(D - 1)K + N(K - 1)$ parameters. The personalization parameters are not model parameters in our case since we derive them from $O$ and $\pi$. 

210
3. Implementation Detail

We create an estimation of the sparse moments $M_2$ by counting the pairwise occurrence of the items across the selections made by all the users in the dataset, and normalizing by the total number of occurrence in each case. This can be achieved in one pass through the dataset using frameworks like Hadoop. Alternatively, if $X \in \mathbb{R}^{N \times D}$ is the binary sparse matrix representing the data, then the pairwise occurrence matrix can be estimated by $X^\top X$, whose sum of all elements is,

$$
\sum_{y} \sum_{y} X^\top X = \sum_{y} \sum_{y} \sum_{i=1}^{N} x_i^\top x_i = \sum_{i=1}^{N} \sum_{y} \sum_{y} x_i^\top x_i = \sum_{i=1}^{N} \text{nnz}(x_i)^2
$$

where $x_i$ is the row of $X$ corresponding to the $i$th user, and $\text{nnz}(x_i)$ is the number of non-zero elements in $x_i$, i.e., the number of items associated with $i$th user. Therefore, $M_2$ can be estimated as,

$$
\hat{M}_2 = \frac{1}{\sum_{i=1}^{N} \text{nnz}(x_i)^2} X^\top X
$$

Similarly, the triple-wise occurrence tensor can be estimated as $X \otimes X \otimes X$, and the sum of all of the elements of the tensor is $\sum_{y} \sum_{y} \sum_{y} X \otimes X \otimes X = \sum_{i=1}^{N} \text{nnz}(x_i)^3$. Therefore, $M_3$ can be estimated as,

$$
\hat{M}_3 = \frac{1}{\sum_{i=1}^{N} \text{nnz}(x_i)^3} X \otimes X \otimes X
$$

The dimensions of $\hat{M}_2$ and $\hat{M}_3$ are $D^2$ and $D^3$ respectively, but in practice, these quantities are extremely sparse. Also, we can estimate $\hat{M}_3$ without estimating $M_3$. Since $\tilde{\hat{M}}_3 = \hat{M}_3(W, W, W)$, it can be estimated as,

$$
\tilde{\hat{M}}_3 = \frac{1}{\sum_{i=1}^{N} \text{nnz}(x_i)^3} XW \otimes XW \otimes XW
$$

$\tilde{\hat{M}}_3$ has a dimension of $K^3$, and can be conveniently stored in the memory ($K \ll D$). Estimating $\tilde{\hat{M}}_3$ takes a second pass through the entire dataset. The entire algorithm is outlined as Algorithm 1. We used the Tensor Toolbox (Bader et al., 2015) for tensor decomposition. Once the matrix $O$ and $\pi_k$ are extracted, it requires one more pass through the entire dataset to compute the user probabilities ($P[h|u]$), resulting in a total of three passes for the extraction of all parameters. Although it is possible to make predictions using only $O$ and $\pi$, it is advisable to compute $P[h|u]$ beforehand to avoid computation cost during prediction step.

The number of elements in $M_2$ is $O\left(\sum_{i=1}^{N} \text{nnz}(x_i)^2\right)$, with the worst case occurring when no two users has any item in common, and all the elements in $X^\top X$ is one. The complexity of extracting $K$ largest eigenvalue of $M_2$ during the whitening step is $O\left(K \left(\sum_{i=1}^{N} \text{nnz}(x_i)^2\right)\right)$. The complexity of Equation 15 is $\Theta(NK^3)$. The tensor factorization step has a complexity of $O\left(K^4 \log \left(1/\epsilon\right)\right)$ to extract all $K$ eigenvalues of $\tilde{\hat{M}}_3$ up to
Algorithm 1 Method of Moments for Parameter Extraction

**Input:** Sparse Data $X \in \mathbb{R}^{N \times D}$ and $K \in \mathbb{Z}^+$

**Output:** $P[y|h]$ and $P[h|u]$

1. Estimate $\hat{M}_2 = (X^T X) / \sum_{i=1}^{N} \text{nnz}(x_i)^2$ (pass # 1)

2. Compute $K$ maximum eigenvalues of $\hat{M}_2$ as $\{\nu_k\}_{k=1}^{K}$, and corresponding eigenvectors as $\{\omega_k\}_{k=1}^{K}$. Define $\Omega = [\omega_1 | \omega_2 | \ldots | \omega_K]$, and $\Sigma = \text{diag}(\nu_1, \nu_2, \ldots, \nu_K)$

3. Estimate the whitening matrix $\hat{W} = \Omega \Sigma^{-1/2}$ so that $\hat{W}^T \hat{M}_2 \hat{W} = I_{K \times K}$

4. Estimate $\hat{M}_3 = (X \hat{W} \otimes X \hat{W} \otimes X \hat{W}) / \sum_{i=1}^{N} \text{nnz}(x_i)^3$ (pass # 2)

5. Compute eigenvalues $\{\lambda_k\}_{k=1}^{K}$ and eigenvectors $\{v_k\}_{k=1}^{K}$ of $\hat{M}_3$

6. Estimate the columns of $O$ as $\hat{\mu}_k = \hat{W}^T v_k$, where $\hat{W}^T = \hat{W} (\hat{W}^T \hat{W})^{-1}$, and $\hat{\pi}_k = \lambda_k^{-2}$, $\forall k \in 1, 2, \ldots, K$

7. Assign $\hat{O} = [\hat{\mu}_1 | \hat{\mu}_2 | \ldots | \hat{\mu}_K] \& \hat{\pi} = [\hat{\pi}_1, \hat{\pi}_2, \ldots, \hat{\pi}_K]^T$

8. Estimate $P[y|h = k] = \frac{O_{yk}}{\sum_y O_{yk}}$, $\forall k \in 1 \ldots K, y \in y_1 \ldots y_D$

9. Estimate $P[h = k|u] = \frac{\hat{\pi}_k \prod_{y \in Y_u} O_{yk}}{\sum_{k=1}^{K} \hat{\pi}_k \prod_{y \in Y_u} O_{yk}}$, $\forall k \in 1 \ldots K, u \in u_1 \ldots u_N$ (pass # 3)

An accuracy of $\epsilon$. These three steps contribute the most to the computational burden of the algorithm. The complexity of the overall algorithm is,

$$O\left(K\left(\sum_{i=1}^{N} \text{nnz}(x_i)^2 \right) + NK^3 + K^4 \log \left(1/\epsilon\right)\right)$$

3.1. Convergence Bound

**Theorem 1** Let us assume that we draw $N$ i.i.d samples $x_1, x_2 \ldots x_N$ corresponding to $N$ users using the generative process in Equation 1. Let us define $\varepsilon = \left(1 + \sqrt{\frac{\log(1/\delta)}{2}}\right)$ for some $\delta \in (0, 1)$. Then, if the number of users $N \geq \max(n_1, n_2, n_3)$, where

- $n_1 = c_2 \left(\log K + \log \frac{K}{c_1} \cdot \sqrt{\frac{\mu_{\text{max}}}{\mu_{\text{min}}}}\right)$
- $n_2 = \Omega \left(\frac{\varepsilon}{d_2 \sigma_K(M_2)}\right)^2$
- $n_3 = \Omega \left(K^2 \left(\frac{10}{d_3 \sigma_K(M_2)} + \frac{2\sqrt{2}}{d_3 \sigma_K(M_2)^{3/2}}\right)^2 \varepsilon^2\right)$

212
for some constants $c_1$ and $c_2$, and we run Algorithm 1 on the data, then the following bounds on the estimated parameters hold with probability at least $1 - \delta$,

$$
\|\mu_k - \hat{\mu}_k\| \leq \left( \frac{160\sqrt{\sigma_1(M_2)}}{d_{2s}\sigma_K(M_2)^{5/2}} + \frac{32\sqrt{2\sigma_1(M_2)}}{d_{3s}\sigma_K(M_2)^{3/2}} + \frac{4\sqrt{\sigma_1(M_2)}}{d_{2s}\sigma_K(M_2)} \right) \frac{\varepsilon}{\sqrt{N}}
$$

and,

$$
|\pi_k - \hat{\pi}_k| \leq \left( \frac{200}{\sigma_K(M_2)^{5/2}} + \frac{40\sqrt{2}}{\sigma_K(M_2)^{3/2}} \right) \frac{\varepsilon}{d_{3s}\sqrt{N}}
$$

where $\sigma_1(M_2) \ldots \sigma_K(M_2)$ are the $K$ largest eigenvalues of the pairwise probability matrix $M_2$, $d_{2s} = \frac{1}{K} \sum_{i=1}^{N} \text{nnz}(x_i)^2$ and $d_{3s} = \frac{1}{K} \sum_{i=1}^{N} \text{nnz}(x_i)^3$, with $\text{nnz}(x_i)$ representing the non-zero elements in the $i$th sample.

The proof is in the appendix.

4. Experimental Results

We show the implementation of our model on three publicly available datasets so that the results can be reproduced whenever necessary. The datasets contain records of user-item interactions over a period, and truly represents implicit feedback systems. We do not convert any dataset with user ratings into implicit feedback dataset, as it may not be an accurate representation of implicit feedback scenario.

The different attributes of datasets are described in Table 1. We use $K = 100$ for all the models in our experiments. For the standard form of PLSI, we run EM algorithm until $L^t - L^{t-1} < .001 \times |L^{t-1}|$, where $L^t$ is the log-likelihood at iteration $t$, resulting in around 25 – 30 iterations for each dataset. We use the implementation of WRMF and BPR from MyMediaLite library \(^1\) developed by the authors of (Rendle et al., 2009). We found that the rest of the algorithms, such as (Kang et al., 2016), (Pan and Chen, 2013) or (Cheng et al., 2014) lacks scalability to train on large datasets, at least in their current implementation provided by the authors. We could not find an implementation of AdaBPR(Liu et al., 2015) from the authors. The article uses much smaller datasets, e.g. the authors select only 27,216 users and 9,994 songs from the Million Song Dataset. We used WRMF and BPR for the benchmarking purpose since most of the relevant literature on recommendation systems considered these two algorithms as the state-of-art.

For every dataset, we compute the Precision@$\tau$, Recall@$\tau$, and Mean Average Precision (MAP@$\tau$) for $\tau \in \{5, 10, 20, 40, 60, 80, 100, 200, 300, 400, 500\}$. The Precision-Recall curves as well as MAP@$\tau$ is shown in Figure 2, and the computation time in Table 2. We carried out our experiments on Unix Platform on a single machine with a single-core 2.4GHz processor and 8GB memory, and did not use multi-threading or any other performance enhancement technique. \(^2\).

---

\(^1\) http://www.mymedialite.net/
\(^2\) The code is available at https://github.com/Sayan21/RecSys
Table 1: Descriptions of the Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th># of Users</th>
<th># of Items</th>
<th># of tuples (training)</th>
<th>Sparsity (training)</th>
<th># of tuples (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ta-Feng</td>
<td>Online Grocery</td>
<td>24,304</td>
<td>21,533</td>
<td>417,246</td>
<td>$5.44 \times 10^4$</td>
<td>274,479</td>
</tr>
<tr>
<td>Million Song</td>
<td>Music Subscription</td>
<td>110,000</td>
<td>163,206</td>
<td>1,450,933</td>
<td>$8.08 \times 10^6$</td>
<td>1,368,430</td>
</tr>
<tr>
<td>Yandex</td>
<td>Search Engine logs</td>
<td>1M</td>
<td>718,675</td>
<td>5,669,541</td>
<td>$7.89 \times 10^6$</td>
<td>3,516,216</td>
</tr>
</tbody>
</table>

Figure 2: Precision-Recall curves (1st row) and Mean Average Precision or MAP (2nd row) of different methods on the three datasets

4.1. Ta-Feng Dataset

Ta-Feng dataset consists of online grocery purchase records for the months of January, February, November and December in 2001. We combine the records of January and November resulting in a training set consisting of around 24,000 users and 21,000 products, and around 470,000 sales records. The records of February and December are combined to form the test set. BPR achieves the highest MAP of all, but MoM produces the best Precision-Recall curve, taking similar time as BPR.

Table 2: Computation Time (sec)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>WRMF</th>
<th>BPR</th>
<th>PLSI</th>
<th>MoM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ta-Feng</td>
<td>2200</td>
<td>108</td>
<td>1081</td>
<td>120</td>
</tr>
<tr>
<td>Million Song</td>
<td>14262</td>
<td>510</td>
<td>3036</td>
<td>600</td>
</tr>
<tr>
<td>Yandex</td>
<td>-</td>
<td>2100</td>
<td>15300</td>
<td>2512</td>
</tr>
</tbody>
</table>
4.2. Million Song Dataset

Million Song dataset contains the logs of 1 million users listening to 385,000 song tracks with 48 million observations. Here, we use a subset of the data consisting of 100,000 users and around 165,000 song tracks with around 1.45 million observations released in Kaggle. MoM performs the best regarding MAP and Precision-Recall, except for higher values of $\tau$ when WRMF catches up.

4.3. Yandex Search Log Dataset

Yandex dataset contains the search logs of 27 days for 5.7 million users and 70.3 million URLs. We selected 718,675 URLs, each of which had at least five clicks since it is not possible to personalize URLs with very few clicks. We randomly selected 1M users who clicked one of those 718,675 URLs. We used the data of first 14 days as the training set, and the last 13 days as the test set. WRMF did not finish even after running for a day. MoM outperformed BPR and PLSI while taking similar time as BPR.

5. Conclusion

Here we propose a collaborative filtering algorithm for implicit feedback based on the second and third order moment factorization of the data. Existing methods like PLSI suffers from local maxima problem. Although Matrix factorizations operate on a convex loss, it is far from trivial to reach the global minima of the loss function through gradient descent alternately on user and item features. The Method of Moments, on the other hand, comes with guaranteed convergence bound. The only drawback of Method of Moments is that it will not work when there are only a few users available such that $N < \Theta(K^2)$. However, modern recommendation systems usually operate on a large number of users, and this is far from a possibility.

We demonstrate the competitive performance of Method of Moments through experiments on three real-world datasets, chosen from different domains. BPR performs better in MAP for Ta-Feng datasets. However, as the size and the sparsity of the datasets increase, the performance of BPR gets worse. The Method of Moments performs the best for Million Song and Yandex datasets while taking similar time as BPR. PLSI or Matrix Factorization (WRMF) clearly lacks the scalability that MoM offers, neither do they produce any better result. Further, MoM depends only on various linear algebraic operations, and it is embarrassingly parallel to implement on any parallel platforms. This makes MoM a very suitable choice for large-scale datasets.

References


Appendix A. Vector Norms

Let the true pairwise probability matrix and the third order probability moment be $M_2 = p(y, y)$ and $M_3 = p(y, y, y)$, where $y$ stands for the items. Let us assume that we select $N$ i.i.d. samples $x_1, \ldots, x_N$ from the population, and the estimates of pairwise matrix and third order moment are $\hat{M}_2 = \hat{p}(y, y)$ and $\hat{M}_3 = \hat{p}(y, y, y)$. Let $\varepsilon_{M_2} = ||M_2 - \hat{M}_2||_2$. We use the second order operator norm of the matrices here. Let us assume $\varepsilon_{M_2} \leq \sigma_K(M_2)/2$, where $\sigma_K$ is the $K$th largest eigenvalue of $M_2$. We will derive the conditions which satisfies this later.

If $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_K)$ are the top-$K$ eigenvalues of $M_2$, and $U$ are the corresponding eigenvectors, then the whitening matrix $W = U\Sigma^{-1/2}$. Also, $W^\top M_{2K} W = I_{K \times K}$, where $M_{2K}$ is the $K$ rank approximation of $M_2$. Then,

$$||W||_2 = \sqrt{\max \text{eig}(W^\top W)} = \sqrt{\max \text{eig}(\Sigma^{-1})} = \frac{1}{\sqrt{\sigma_K(M_2)}}$$

Similarly, if $W^\top = W(W^\top W)^{-1}$, then $W^\top = W\Sigma = U\Sigma^{1/2}$. Therefore,

$$||W^\top||_2 = \sqrt{\max \text{eig}(\Sigma)} = \sqrt{\sigma_1(M_2)}$$  \hspace{1cm} (16)

Let $\hat{W}$ be the whitening matrix for $\hat{M}_2$, i.e., $\hat{W}^\top \hat{M}_2 \hat{W} = I_{K \times K}$. Then by Weyl's inequality, $\sigma_K(M_2) - \sigma_K(\hat{M}_2) \leq ||M_2 - \hat{M}_2||, \forall k = 1, 2, \ldots, K$.

Therefore,

$$||\hat{W}||_2^2 = \frac{1}{\sigma_K(M_2)} \leq \frac{1}{\sigma_K(M_2) - ||M_2 - \hat{M}_2||} \leq \frac{2}{\sigma_K(M_2)}$$  \hspace{1cm} (17)

Also, by Weyl’s Theorem,

$$||\hat{W}^\top||_2^2 = \sigma_1(\hat{M}_2) \leq \sigma_1(M_2) + \varepsilon_{M_2} \leq 1.5\sigma_1(M_2) \implies ||\hat{W}^\top||_2 \leq \sqrt{1.5\sigma_1(M_2)} \leq 1.5\sqrt{\sigma_1(M_2)}$$  \hspace{1cm} (18)

Let $D$ be the eigenvectors of $\hat{W} M_2 \hat{W}$, and $A$ be the corresponding eigenvalues. Then we can write, $\hat{W} M_2 \hat{W} = ADA^\top$. Then $W = WAD^{-1/2}A^\top$ whitens $M_2$, i.e., $W^\top M_2 W = I$. Therefore,
\[ \|I - D\|_2 = \|I - ADA^{\top}\|_2 = \|I - \hat{W}M_2\hat{W}\|_2 = \|\hat{W}M_2\hat{W} - \hat{W}M_2\hat{W}\|_2 \leq \|\hat{W}\|_2^2\|M_2 - M_2\| \leq \frac{2}{\sigma_K(M_2)} \varepsilon_{M_2} \]  

(19)

\[ \varepsilon_W = \|W - WAD^{1/2}A^{\top}\|_2 = \|W\|_2\|I - AD^{1/2}A^{\top}\|_2 = \|W\|_2\|I - D^{1/2}\|_2 \leq \|W\|_2\|I - D^{1/2}\|_2\|I + D^{1/2}\|_2 = \|W\|_2\|I - D\|_2 \leq \frac{2}{\sigma_K(M_2)^{3/2}} \varepsilon_{M_2} \]  

(20)

\[ \varepsilon_{W^\dagger} = \|W^\dagger - \hat{W}^\dagger\|_2 = \|\hat{W}^\dagger AD^{1/2}A^{\top} - \hat{W}^\dagger\|_2 = \|\hat{W}^\dagger\|_2\|I - AD^{1/2}A^{\top}\|_2 \leq \|\hat{W}^\dagger\|_2\|I - D\|_2 \leq \frac{2\sqrt{\sigma_1(M_2)}}{\sigma_K(M_2)} \varepsilon_{M_2} \]  

(21)

**Appendix B. Tensor Norm**

Let us define the second order operator norm of a tensor \(T \in \mathbb{R}^{D \times D \times D}\) as,

\[\|T\|_2 = \sup_v \{\|T(v, v, v)\| : v \in \mathbb{R}^D \land \|v\| = 1\}\]  

(22)

**Lemma 2** For a tensor \(T \in \mathbb{R}^{D \times D \times D}\), \(\|T\|_2 \leq \|T\|_F\), where \(\|T\|_F\) is the Frobenius norm defined as,

\[\|T\|_F = \sqrt{\sum_{i,j,k} (T_{i,j,k})^2}\]

**Proof** For any real matrix \(A\), \(\|A\|_2 \leq \|A\|_F\). Let us unfold the tensor \(T\) as the collection of \(D\) matrices, as, \(T = \{T_1, T_2, \ldots, T_D\}\). Then,

\[T(v, v, v) = v^\dagger [T_1v|T_2v|\ldots|T_Kv]v = \langle v^\dagger T_1v, v^\dagger T_2v, \ldots, v^\dagger T_Kv, v \rangle\]  

(23)
Therefore,
\[ ||T||_2 = \sup_v \{|T(v, v, v) : v \in \mathbb{R}^D & ||v|| = 1\} \]
\[ = \sup_v \{|[v^\top T_1 v, v^\top T_2 v, \ldots, v^\top T_K v], v) : v \in \mathbb{R}^D & ||v|| = 1\} \]

Using Cauchy-Schwarz inequality,
\[ ||T||_2 \leq \sup_v \{||[v^\top T_1 v, v^\top T_2 v, \ldots, v^\top T_K v]| |v|| : v \in \mathbb{R}^D & ||v|| = 1\} \]
\[ = \sup_v \{||[T_1||_F, ||T_2||_F, \ldots, ||T_D||_F]| \}
\[ \leq \sqrt{(||T_1||_F^2 + ||T_2||_F^2 + \ldots + ||T_D||_F^2)} \]
\[ = ||T||_F \tag{24} \]

**Lemma 3** (Robust Power Method from (Anandkumar et al., 2014)) If \( \hat{T} = T + E \in \mathbb{R}^{K \times K \times K} \), where \( T \) is an symmetric tensor with orthogonal decomposition \( T = \sum_{k=1}^{K} \lambda_k u_k \otimes u_k \otimes u_k \) with each \( \lambda_k > 0 \), and \( E \) has operator norm \( ||E||_2 \leq \epsilon \). Let \( \lambda_{\min} = \min_{k=1}^{K} \lambda_k \) and \( \lambda_{\max} = \max_{k=1}^{K} \lambda_k \). Let there exist constants \( c_1, c_2 \) such that \( \epsilon \leq c_1 \cdot (\lambda_{\min}/K) \), and \( N \geq c_2 (\log K + \log \log (\lambda_{\max}/\epsilon)) \). Then if Algorithm 1 in (Anandkumar et al., 2014) is called for \( K \) times, with \( L = \text{poly}(K) \log(1/\eta) \) restarts each time for some \( \eta \in (0, 1) \), then with probability at least 1 - \( \eta \), there exists a permutation \( \pi \) on \([K]\), such that,
\[ ||u_{\pi(k)} - \hat{u}_k|| \leq \frac{8 \epsilon}{\lambda_{\pi(k)}} \quad |\lambda_k - \lambda_{\pi(k)}| \leq 5\epsilon \quad \forall k \in [K] \tag{25} \]

Since \( \epsilon \leq c_1 \cdot (\lambda_{\min}/K) \) and \( \lambda_k = \frac{1}{\sqrt{\pi_k}} \), \( \forall k \in [K] \), we need
\[ N \geq c_2 \left( \log K + \log \log \left( \frac{K \lambda_{\max}}{c_1 \lambda_{\min}} \right) \right) = c_2 \left( \log K + \log \log \left( \frac{K}{c_1 \sqrt{\lambda_{\min} \lambda_{\max}}} \right) \right) \tag{26} \]

This contributes in the first lower bound \((n_1)\) of \( N \) in Theorem 1.

**Appendix C. Tail Inequality**

**Lemma 4** If we draw \( N \) i.i.d. samples \( x_1, x_2 \ldots x_N \) through the generative process in Equation 1 corresponding to \( N \) users, and the vectors probability mass function of the items \( y \) estimated from these \( N \) samples are \( \hat{p}(y) \) whereas the true p.m.f is \( p(y) \) with \( y \in \{y_1, y_2 \ldots y_D\} \), then with probability at least 1 - \( \delta \) with \( \delta \in (0, 1) \),
\[ ||\hat{p}(y) - p(y)||_F \leq \frac{2}{d_{1s}\sqrt{N}} \left( 1 + \sqrt{\frac{\log(1/\delta)}{2}} \right) \]  
(27)

\[ ||\hat{p}(y, y) - p(y, y)||_F \leq \frac{2}{d_{2s}\sqrt{N}} \left( 1 + \sqrt{\frac{\log(1/\delta)}{2}} \right) \]  
(28)

\[ ||\hat{p}(y, y, y) - p(y, y, y)||_F \leq \frac{2}{d_{3s}\sqrt{N}} \left( 1 + \sqrt{\frac{\log(1/\delta)}{2}} \right) \]  
(29)

where, \( d_{1s} = \frac{1}{N} \sum_{i=1}^{N} \text{nnz}(x_i) \), \( d_{2s} = \frac{1}{N} \sum_{i=1}^{N} \text{nnz}(x_i)^2 \), \( d_{3s} = \frac{1}{N} \sum_{i=1}^{N} \text{nnz}(x_i)^3 \), and \( \text{nnz}(x_i) \) is the non-zero entries in row \( x_i \) of the data \( X \) as described in section 3.

The proof is included in the supplementary material.

Appendix D. Bounds on the Parameters

Assigning \( \varepsilon = \left( 1 + \sqrt{\frac{\log(1/\delta)}{2}} \right) \) in the inequalities of Lemma 4, we get

\[ \varepsilon_{M_2} = ||\hat{p}(y, y) - p(y, y)||_2 \leq ||\hat{p}(y, y) - p(y, y)||_F \leq \frac{2\varepsilon}{d_{2s}\sqrt{N}} \],

and

\[ \varepsilon_{M_3} = ||M_3 - \hat{M}_3||_2 = ||\hat{p}(y, y, y) - p(y, y, y)||_2 \leq ||\hat{p}(y, y, y) - p(y, y, y)||_F \leq \frac{2\varepsilon}{d_{3s}\sqrt{N}} \]

since operator norm is smaller than Frobenius norm for both matrices and tensors.

Therefore, to satisfy \( \varepsilon_{M_2} \leq \sigma_K(M_2)/2 \), we need \( N \geq \Omega \left( \frac{\varepsilon}{\sigma_2\sigma_K(M_2)} \right) \). This contributes in the second lower bound \( n_2 \) of \( N \) in Theorem 1.

From Appendix B in (Chaganty and Liang, 2013),

\[ \varepsilon_{tw} = ||M_3(W, W, W) - \hat{M}_3(\hat{W}, \hat{W}, \hat{W})||_2 \]

\[ \leq ||M_3||_2 \left( ||\hat{W}||^2_2 + ||\hat{W}||_2||W||_2 + ||W||^2_2 \right) \varepsilon_W + ||\hat{W}||_3^3 \varepsilon_{M_3} \]

\[ \leq ||M_3||_2 \frac{2 + \sqrt{2}}{\sigma_K(M_2)} \varepsilon_W + \frac{2\sqrt{2}}{\sigma_K(M_2)^{3/2}} \varepsilon_{M_3} \]

\[ \leq ||M_3||_2 \frac{3 + \sqrt{2}}{\sigma_K(M_2)^{3/2}} \varepsilon_{M_3} + \frac{2\sqrt{2}}{\sigma_K(M_2)^{3/2}} \varepsilon_{M_3} \]

\[ \leq \frac{10||M_3||_2}{\sigma_K(M_2)^{5/2}} \varepsilon_{M_2} + \frac{2\sqrt{2}}{\sigma_K(M_2)^{3/2}} \varepsilon_{M_3} \]

\[ \leq \left( \frac{10}{d_{2s}\sigma_K(M_2)^{5/2}} + \frac{2\sqrt{2}}{d_{3s}\sigma_K(M_2)^{3/2}} \right) \frac{2\varepsilon}{\sqrt{N}} \]  
(30)

Please note that \( ||M_3||_2 \leq ||M_3||_F \leq 1 \), because \( M_3 \) is a tensor with individual elements as probabilities, with the sum of all elements being 1.
From 3, \( \varepsilon \leq c_1 \cdot (\lambda_{\min}/K) \), and we can assign \( \varepsilon \) as the upper bound of \( \varepsilon_{tw} \). To satisfy this, we need
\[
\left( \frac{10}{d_{2s} \sigma_K(M_2)^{5/2}} + \frac{2\sqrt{2}}{d_{3s} \sigma_K(M_2)^{3/2}} \right) \frac{2\varepsilon}{\sqrt{N}} \leq c_1 \frac{\lambda_{\min}}{K}, \text{ or,}
\left( \frac{10}{d_{2s} \sigma_K(M_2)^{5/2}} + \frac{2\sqrt{2}}{d_{3s} \sigma_K(M_2)^{3/2}} \right) \frac{2\varepsilon}{\sqrt{N}} \leq \frac{1}{K \pi_{\max}}.
\]
Since \( \pi_{\max} \leq 1 \), we need \( N \geq \Omega \left( K^2 \left( \frac{10}{d_{2s} \sigma_K(M_2)^{5/2}} + \frac{2\sqrt{2}}{d_{3s} \sigma_K(M_2)^{3/2}} \right)^2 \varepsilon^2 \right) \). This contributes to \( n_3 \) in Theorem 1.

Here, we will derive the final bounds for the reconstruction error for the parameters. Since \( \mu_k = W^t u_k \) (Algorithm 1), with probability at least \( 1 - \delta \),
\[
||\mu_k - \hat{\mu}_k|| = ||W^t u_k - \hat{W}^t \hat{u}_k|| = ||W^t u_k - \hat{W}^t \hat{u}_k|| + ||\hat{W}^t - W^t|| ||u_k|| \\
\leq ||W^t||_2 ||u_k - \hat{u}_k|| + ||W^t - \hat{W}^t|| ||u_k|| \\
\leq ||W^t||_2 \frac{8\varepsilon}{\lambda_k} + \varepsilon||W^t|| \\
\leq 8\sqrt{\sigma_1(M_2)} + \frac{2\sqrt{\sigma_1(M_2)}}{\sigma_K(M_2)} \varepsilon_{M_2}.
\] (31)

Since \( \frac{1}{\lambda_k} = \sqrt{\pi_k} \leq 1 \). Therefore, with probability at least \( 1 - \delta \),
\[
||\mu_k - \hat{\mu}_k|| \leq 8\sqrt{\sigma_1(M_2)} \left( \frac{10}{d_{2s} \sigma_K(M_2)^{5/2}} + \frac{2\sqrt{2}}{d_{3s} \sigma_K(M_2)^{3/2}} \right) \frac{2\varepsilon}{\sqrt{N}} + \frac{2\sqrt{\sigma_1(M_2)}}{\sigma_K(M_2)} \frac{2\varepsilon}{d_{2s} \sqrt{N}} \\
\leq \left( \frac{160\sqrt{\sigma_1(M_2)}}{d_{2s} \sigma_K(M_2)^{5/2}} + \frac{32\sqrt{2}\sigma_1(M_2)}{d_{3s} \sigma_K(M_2)^{3/2}} + \frac{4\sqrt{\sigma_1(M_2)}}{d_{2s} \sigma_K(M_2)} \right) \frac{\varepsilon}{\sqrt{N}}.
\] (32)

\[
|\pi_k - \hat{\pi}_k| = \left| \frac{1}{\lambda_k^2} - \frac{1}{\hat{\lambda}_k^2} \right| = \left| \frac{(\lambda_k + \hat{\lambda}_k)(\lambda_k - \hat{\lambda}_k)}{\lambda_k^2 \hat{\lambda}_k^2} \right| = \left| \sqrt{\pi_k \pi_k} (\sqrt{\pi_k} + \sqrt{\pi_k}) (\lambda_k - \hat{\lambda}_k) \right| \\
\leq 2|\lambda_k - \hat{\lambda}_k| \leq 10\varepsilon.
\] (33)

since \( |\lambda_k - \hat{\lambda}_k| \leq 5\varepsilon \) from Lemma 3. Therefore, with probability at least \( 1 - \delta \), we get
\[
|\pi_k - \hat{\pi}_k| \leq \left( \frac{200}{\sigma_K(M_2)^{5/2}} + \frac{40\sqrt{2}}{\sigma_K(M_2)^{3/2}} \right) \frac{\varepsilon}{d_{3s} \sqrt{N}}
\] (34)

where \( \varepsilon = \left( 1 + \sqrt{\frac{\log(1/\delta)}{2}} \right) \) all along. This completes the proof of Theorem 1.