Journal of Machine Learning Research 17 (2016) 1-24 Submitted 7/14; Revised 5/15; Published 3/16

LLORMA: Local Low-Rank Matrix Approximation

Joonseok Lee JOONSEOK2010@GMAIL.COM
Seungyeon Kim SEUNGYEONK@GOOGLE.COM
Google Research, Mountain View, CA USA
Guy Lebanon GLEBANON@GMAIL.COM
LinkedIn, Mountain View, CA USA
Yoram Singer SINGER@GOOGLE.COM
Samy Bengio BENGIO@GOOGLE.COM
Google Research, Mountain View, CA USA

Editor: Jeff Bilmes

Abstract

Matrix approximation is a common tool in recommendation systems, text mining, and computer vision. A prevalent assumption in constructing matrix approximations is that the partially observed matrix is low-rank. In this paper, we propose, analyze, and experiment with two procedures, one parallel and the other global, for constructing *local* matrix approximations. The two approaches approximate the observed matrix as a weighted sum of low-rank matrices. These matrices are limited to a local region of the observed matrix. We analyze the accuracy of the proposed local low-rank modeling. Our experiments show improvements in prediction accuracy over classical approaches for recommendation tasks.

Keywords: Matrix approximation, non-parametric methods, kernel smoothing, collaborative Filtering, recommender systems.

1. Introduction

Matrix approximation and completion are prevalent tasks in machine learning. Given few observed matrix entries $\{M_{a_1,b_1}, \ldots, M_{a_m,b_m}\}$, matrix completion constructs a matrix $\hat{M}$ that approximates M at its unobserved entries. Matrix approximation is used heavily in recommendation systems, text processing, computer vision, and bioinformatics. In recommendation systems, for example, the matrix M corresponds to ratings of items (columns) by users (rows). Matrix approximation in this case corresponds to predicting the ratings of all users on all items based on a few observed ratings. In many cases, matrix approximation leads to state-of-the-art models that are used in industrial settings.

In general, the problem of completing a matrix M based on a few observed entries is ill-posed. There are uncountably infinite number of matrices that perfectly agree with the observed entries of M. Therefore, without additional assumptions, selecting or constructing the completion matrix $\hat{M}$ is under-specified and thus ill-defined. A popular assumption is that M is a low-rank matrix, which suggests that it is reasonable to assume that the completed matrix $\hat{M}$ has low-rank. More formally, we approximate a matrix $M \in \mathbb{R}^{n_1 \times n_2}$ by a rank r matrix $\hat{M} = UV^\top$, where $U \in \mathbb{R}^{n_1 \times r}$, $V \in \mathbb{R}^{n_2 \times r}$, and $r \ll \min(n_1, n_2)$. In

many real datasets, the low-rank assumption is realistic. Further, low-rank approximations often yield matrices that generalize well to the unobserved entries.

In this paper, we extend low-rank matrix approximation in a way that significantly relaxes the low-rank assumption. Instead of assuming that M can be globally approximated by a low-rank matrix, we assume that M behaves as a low-rank matrix in the vicinity of certain row-column combinations. We therefore construct several low-rank approximations of M, each being accurate in a particular region of the matrix. We express our estimator as a smoothed convex combination of low-rank matrices each of which approximates M in a local region.

The local low-rank assumption can also be motivated as follows. In numerous settings there are a few key latent factors which determine whether a user would like an item or not. In the movie domain such factors may include the main actress, director, released year, genre, and more. However, the number of latent variable is typically limited to no more than twenty. These factors are tacitly learned and automatically constructed as we do not assume that information such as genre or actors are available. For example, if the rank is 5 the ratings given to an item by a user is the inner product of a vector of length 5 which describes the user preferences with a vector of length 5 that describes the item characteristics (these two vectors are the rows of U, V). The same assumption holds in the local low-rank case with the exception that the linear basis underlying the latent factors may change across different groups of users and different types of items.

We use techniques from non-parametric kernel smoothing to achieve two goals. The first goal is to formally develop a notion of local low-rank approximation, and the second is the aggregation of several local models into unified matrix approximation. Standard low-rank matrix approximation techniques achieve consistency in the limit of large data (convergence to the data generating process) assuming that M is low-rank. Our local method achieves consistency without the low-rank assumption. Instead, we require that sufficient number of samples is available in increasingly small neighborhoods. Our analysis mirrors the theory of non-parametric kernel smoothing that was primarily developed for continuous spaces. We also adapt and generalize well-known compressed sensing results to our setting. Our experiments show that local low-rank modeling is significantly more accurate than global low-rank modeling in the context of recommendation systems.

The rest of the paper is organized as follows. We introduce notations and briefly review low-rank matrix approximation in Section 2. In Section 3 we describe our proposed methods. Section 4 provides formal analysis of the proposed approach. Sections 5 and 6 describe in details the two low-rank approximation algorithms. These sections are followed by experimental evaluations described in Sections 7. We then discuss our contribution in the context of related work in Section 8 and conclude with a summary in Section 9.

2. Background: Low-rank matrix approximation

We describe in this section two standard approaches for low-rank matrix approximation (LRMA). We start by establishing the notation used throughout the paper. We denote matrices using upper case letters. The original (partially observed) matrix is denoted by $M \in \mathbb{R}^{n_1 \times n_2}$. A low-rank approximation of M is denoted by $\hat{M} = UV^\top$, where $U \in \mathbb{R}^{n_1 \times r}$, $V \in \mathbb{R}^{n_2 \times r}$, and $r \ll \min(n_1, n_2)$. The set of integers $\{1, \ldots, n\}$ is abbreviated as $[n]$. The

Notation	Explanation
n_1	Number of users.
n_2	Number of items.
m	Number of available ratings.
r	Rank of approximation matrix.
M	Rating matrix, $M \in \mathbb{R}^{n_1 \times n_2}$
U	"Users" profile matrix, $U \in \mathbb{R}^{n_1 \times r}$
V	"Items" profile matrix, $V \in \mathbb{R}^{n_2 \times r}$
Ω	Observed entries of M.
$\mathcal{P}_\Omega(M)$	Projection operator onto observed entries of Ω.
$\hat{\mathcal{T}}(a,b)$	Local approximation of M centered at (a,b).
$\hat{\hat{\mathcal{T}}}(a,b)$	Global approximation of M centered at (a,b).
$A \odot B$	Hadamard product of matrices A and B.
$\|X\|_F$	Frobenius norm of matrix X.
$\|X\|_*$	Nuclear (trace) norm of matrix X.
$\|X\|_\infty$	Sup-norm of matrix X.
$[n]$	Set of natural numbers $\{1, \ldots, n\}$.

Table 1: Summary of Notations and Matrix Operators.

set of indices of the observed entries of M is denoted by $\Omega \stackrel{\text{def}}{=} \{(a_1, b_1), \ldots, (a_m, b_m)\} \subseteq [n_1] \times [n_2]$. The training set is therefore $\{M_{a,b} : (a,b) \in \Omega\}$. Mappings from matrix indices to a matrix space are denoted in calligraphic letters, e.g. $\mathcal{T}$, and are operators of the form $\mathcal{T} : [n_1] \times [n_2] \to \mathbb{R}^{n_1 \times n_2}$. We denote the entry (i,j) of the matrix $\mathcal{T}(a,b)$ as $\mathcal{T}_{i,j}(a,b)$. A projection $\mathcal{P}_{\mathtt{A}}$ with respect to a set of matrix indices $\mathtt{A}$ is the function $\mathcal{P}_{\mathtt{A}} : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times n_2}$ defined by

$$[\mathcal{P}_\Omega(M)]_{a,b} \stackrel{\text{def}}{=} \begin{cases} M_{a,b} & (a,b) \in \Omega \\ 0 & \text{otherwise.} \end{cases}$$

We denote by $\odot$ the entry-wise product (also known as the Hadamard or Schur products) of two matrices $[A \odot B]_{i,j} = A_{i,j} B_{i,j}$. We use in this paper three matrix norms:

Frobenius norm $\|X\|_F \stackrel{\text{def}}{=} \sqrt{\sum_i \sum_j X_{i,j}^2}$

Sup-norm $\|X\|_\infty \stackrel{\text{def}}{=} \sup_{i,j} |X_{i,j}|$

Nuclear (trace) norm $\|X\|_* \stackrel{\text{def}}{=} \sum_{i=1}^{r} \sigma_i(X)$

For the nuclear norm $\sigma_i(X)$ is the i'th singular value of X where for symmetric matrices $\|X\|_* = \text{trace}(X)$. Table 1 summarizes notations and matrix operators used throughout the paper.

We describe below two popular approaches for constructing a low-rank approximation $\hat{M}$ of M. The first one, incomplete SVD, is based on minimizing the Frobenius norm of $\mathcal{P}_{\mathtt{A}}(M - \hat{M})$, while the second one is based on minimizing the nuclear norm of a matrix satisfying constraints constructed from the training set.

A1: Incomplete SVD. The incomplete SVD method constructs a low-rank approximation $\hat{M} = UV^\top$ by solving the problem

$$(U, V) = \arg\min_{U,V} \sum_{(a,b)\in\Omega} ([UV^\top]_{a,b} - M_{a,b})^2 \ , \tag{1}$$

or equivalently

$$\hat{M} = \arg\min_{X} \|\mathcal{P}_\Omega(X - M)\|_F \ \text{ s.t. } \operatorname{rank}(X) = r. \tag{2}$$

A2: Nuclear norm minimization. An alternative to (2) that originated from the compressed sensing community (Candès and Tao, 2010) is to minimize the nuclear norm of a matrix subject to constraints constructed from the observed entries:

$$\hat{M} = \arg\min_{X} \|X\|_* \quad \text{s.t.} \quad \|\mathcal{P}_\Omega(X - M)\|_F < \delta \ . \tag{3}$$

Minimizing the nuclear norm $\|X\|_*$ is an effective surrogate for minimizing the rank of X, and solving (3) results in a low-rank matrix $\hat{M} = UV^\top$ that approximates the matrix M. One advantage of *A2* over *A1* is that we do not need to constrain the rank of $\hat{M}$ in advance. Note also that the problem defined by (3), while being convex, may not necessarily scale up easily to large matrices.

3. Local low-rank matrix approximation

In order to facilitate a local low-rank matrix approximation, we need to assume that there exists a metric structure over $[n_1] \times [n_2]$. The distance $d((a, b), (a', b'))$ reflects the similarity between the rows a and a' and columns b and b'. In the case of recommendation systems, for example, $d((a, b), (a', b'))$ expresses the relationship between users a, a' and items b, b'. The distance function may be constructed using the observed ratings $\mathcal{P}_\Omega(M)$ or additional information such as item-item similarity or side information on the users when available. We note that distance between two rows (users) or between two columns (items) is independent of the indices of those rows or columns. As we exchange the order of two rows or columns, the similarity still remains the same. See Section 5 for further details.

In the global matrix factorization setting in Section 2, we assume that the matrix $M \in \mathbb{R}^{n_1\times n_2}$ has a low-rank structure. In the local setting, however, we assume that the model is characterized by multiple low-rank $n_1 \times n_2$ matrices. Specifically, we assume a mapping $\mathcal{T} : [n_1] \times [n_2] \to \mathbb{R}^{n_1\times n_2}$ that associates with each row-column combination $[n_1] \times [n_2]$ a low rank matrix that describes the entries of M in its neighborhood (in particular this applies to the observed entries Ω):

$$\mathcal{T} : [n_1] \times [n_2] \to \mathbb{R}^{n_1\times n_2} \quad \text{where} \quad \mathcal{T}_{a,b}(a, b) = M_{a,b} \ .$$

Without additional assumptions, it is impossible to estimate the mapping $\mathcal{T}$ from a set of $m < n_1 n_2$ observations. We assume, as is often done in non-parametric statistics, that the mapping $\mathcal{T}$ is slowly varying. Since the domain of $\mathcal{T}$ is discrete, the classical definitions of continuity or differentiability are not applicable in our setting. We assume instead that $\mathcal{T}$ is Hölder continuous (see Definition 1 in Section 4).

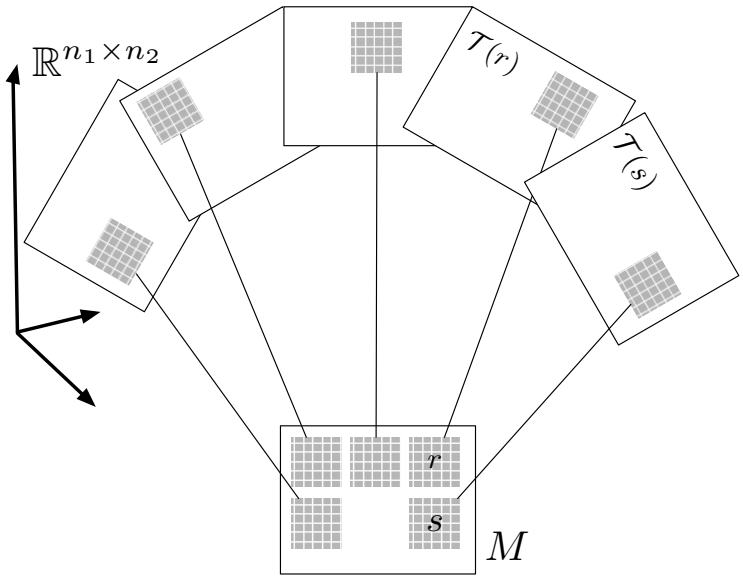

Figure 1: The locally low-rank linear assumption assumes an operator that maps matrix entries to matrices whose image is (a) low-rank, (b) slowly changing, and (c) agrees locally with the original matrix. We emphasize that we draw the figure as if adjacent rows (or columns) are semantically similar just for illustration purpose. In real data, similar users (or items) are usually scattered over the entire space. See text for more details.

Figures 1 shows a graphic illustration of the locally low-rank linear assumption: the operator $\mathcal{T}$ maps matrix entries to matrices whose image is (a) low-rank, (b) slowly changing, and (c) agrees locally with the original matrix. Assumption (b) implies that if $d(s, r)$ is small $\mathcal{T}(s)$ is similar to $\mathcal{T}(r)$, as shown by their spatial closeness in the embedding $\mathbb{R}^{n_1 \times n_2}$. Assumption (c) implies that for all $s \in [n_1] \times [n_2]$, the neighborhood $\{s' : d(s, s') < h\}$ in the original matrix M is approximately described by the corresponding entries of the low-rank matrix $\mathcal{T}(s)$ (shaded regions of M are matched by lines to the corresponding regions in $\mathcal{T}(s)$ that approximate them).

We would like to emphasize that for illustrative purposes, we assume in Figure 1 that there exists a distance function d whose neighborhood structure coincides with the natural order on indices. That is, $s = (a, b)$ is similar to $r = (c, d)$ if $|a - c|$ and $|b - d|$ are small.

Figure 2 shows the relationship between the neighboring entries of the original matrix and the operator image in more detail. The original matrix M (bottom) is described locally by two low-rank matrices $\mathcal{T}(t)$ (near t) and $\mathcal{T}(r)$ (near r). The lines connecting the three matrices identify identical entries: $M_t = \mathcal{T}_t(t)$ and $M_r = \mathcal{T}_r(r)$. The equation at the top right shows a relation tying the three patterned entries. Assuming the distance $d(t, r)$ is small, $\delta = \mathcal{T}_r(t) - \mathcal{T}_r(r) = \mathcal{T}_r(t) - M_r(r)$ is small as well.

Following common approaches in non-parametric statistics, we define a smoothing kernel $K_h(s_1, s_2),\ s_1, s_2 \in [n_1] \times [n_2]$, as a non-negative symmetric unimodal function that is

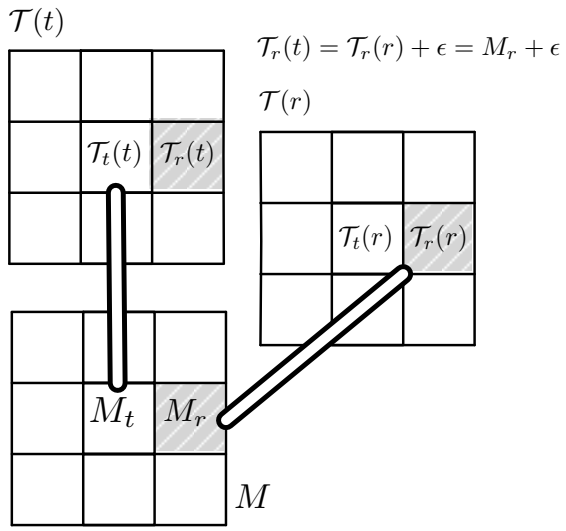

Figure 2: The relationship between the neighboring entries of the original matrix and the operator image. See text for more details.

parameterized by a bandwidth parameter $h > 0$. A large value of h implies that $K_h(s, \cdot)$ has a wide spread, while a small h corresponds to narrow spread of $K_h(s, \cdot)$. Often, it is further assumed that $K_h(x) = K_1(x/h)/h$ and that the kernel integrates to 1: $\int K_h(x)\, dx = 1$. In our case, however, we have a discrete domain rather than a continuous domain. See for instance (Wand and Jones, 1995) for more information on smoothing kernels. Three popular smoothing kernels are the uniform kernel, the triangular kernel, and the Epanechnikov kernel, defined respectively as

$$K_h(s_1, s_2) \propto \mathbf{1}[d(s_1, s_2) < h] \tag{4}$$

$$K_h(s_1, s_2) \propto (1 - h^{-1} d(s_1, s_2))\, \mathbf{1}[d(s_1, s_2) < h] \tag{5}$$

$$K_h(s_1, s_2) \propto (1 - d(s_1, s_2)^2)\, \mathbf{1}[d(s_1, s_2) < h] \ . \tag{6}$$

We denote by $K_h^{(a,b)}$ the matrix whose (i, j)-entry is $K_h((a, b), (i, j))$.

We describe below the local modifications of incomplete SVD (*A1*) and nuclear norm minimization (*A2*) matrix approximations. Both extensions estimate $\mathcal{T}(a, b)$ in the vicinity of $(a, b) \in [n_1] \times [n_2]$ given the samples $\mathcal{P}_\Omega(M)$.

Local-A1: Incomplete SVD

$$\hat{\mathcal{T}}(a, b) = \arg\min_X \|K_h^{(a,b)} \odot \mathcal{P}_\Omega(X - M)\|_F \quad \text{s.t.} \quad \text{rank}(X) = r \ . \tag{7}$$

Local-A2: Nuclear norm minimization

$$\hat{\mathcal{T}}(a, b) = \arg\min_X \|X\|_* \quad \text{s.t.} \quad \|K_h^{(a,b)} \odot \mathcal{P}_\Omega(X - M)\|_F < \delta \ . \tag{8}$$

The two optimization problems above describe how to estimate $\hat{\mathcal{T}}(a, b)$ for a particular choice of $(a, b) \in [n_1] \times [n_2]$. Conceptually, this technique can be applied at each test entry (a, b), resulting in the matrix approximation $\hat{M} \approx M$ where

$$\hat{M}_{a,b} = \hat{\mathcal{T}}_{a,b}(a, b), \quad (a, b) \in [n_1] \times [n_2] \ .$$

However, such a construction would require solving an optimization problem for each matrix entry (a, b) and is thus computationally prohibitive. Instead, we describe in the next subsection how to use a set of q local models $\hat{\mathcal{T}}(s_1), \ldots, \hat{\mathcal{T}}(s_q)$, $s_1, \ldots, s_q \in [n_1] \times [n_2]$ to obtain a computationally efficient estimate $\hat{\hat{\mathcal{T}}}(s)$ for all $s \in [n_1] \times [n_2]$.

3.1 Global Approximation

The problem of recovering a mapping $\mathcal{T}$ from q values without imposing a strong parametric form is known as non-parametric regression. We propose using a variation of locally constant kernel regression (Wand and Jones, 1995), also known as Nadaraya-Watson regression

$$\hat{\hat{\mathcal{T}}}(s) = \sum_{i=1}^{q} \frac{K_h(s_i, s)}{\sum_{j=1}^{q} K_h(s_j, s)} \hat{\mathcal{T}}(s_i) \ . \tag{9}$$

Equation (9) is simply a weighted average of $\hat{\mathcal{T}}(s_1), \ldots, \hat{\mathcal{T}}(s_q)$, where the weights ensure that values of $\hat{\mathcal{T}}$ at indices close to s contribute more than those further away from s. Note that both the left-hand side and the right-hand side of (9) denote matrices. The denominator in (9) ensures that the weights sum to one.

In contrast to $\hat{\mathcal{T}}$, the estimate $\hat{\hat{\mathcal{T}}}$ can be computed for all $s \in [n_1] \times [n_2]$ efficiently since computing $\hat{\hat{\mathcal{T}}}(s)$ simply requires evaluating and averaging $\hat{\mathcal{T}}(s_i)$, $i = 1, \ldots, q$. The resulting matrix approximation is $\hat{\hat{M}}_{a,b} = \hat{\hat{\mathcal{T}}}_{a,b}(a, b)$ and $(a, b) \in [n_1] \times [n_2]$.

The accuracy of $\hat{\hat{\mathcal{T}}}$ as an estimator of $\hat{\mathcal{T}}$ improves with the number of local models q and the degree of continuity of $\hat{\mathcal{T}}$. The accuracy of $\hat{\hat{\mathcal{T}}}$ as an estimator of $\mathcal{T}$ is limited by the quality of the local estimators $\hat{\mathcal{T}}(s_1), \ldots, \hat{\mathcal{T}}(s_q)$. However, assuming that $\hat{\mathcal{T}}(s_1), \ldots, \hat{\mathcal{T}}(s_q)$ are accurate in the neighborhoods of $s_1, \ldots, s_q$, and q is sufficiently large, the estimation error $\hat{\hat{\mathcal{T}}}_{a,b}(a, b) - \mathcal{T}_{a,b}(a, b)$ is likely to be small as we analyze in the next section. We term the resulting approach LLORMA standing for Local LOw Rank Matrix Approximation.

4. Estimation accuracy

In this section we analyze the estimation accuracy of LLORMA. Our analysis consists of two parts. In the first we analyze the large deviation of $\hat{\mathcal{T}}$ from $\mathcal{T}$. Then, based on this analysis, we derive a deviation bound on the global approximation $\hat{\hat{\mathcal{T}}}$. Our analysis technique is based on the seminal paper of Candès and Tao (2010). The goal of this section is to underscore the characteristics of estimation error in terms of parameters such as the train set size, matrix dimensions, and kernel bandwidth.

4.1 Analysis of $\hat{\mathcal{T}} - \mathcal{T}$

Candès and Tao (2010) established that it is possible to estimate an $n_1 \times n_2$ matrix M of rank r if the number of observations $m \geq C\mu r n \log^6 n$, where $n = \min(n_1, n_2)$, C is a

constant, and μ is the strong incoherence property parameter described in Candès and Tao (2010). This bound is tight in the sense that it is close to the information theoretic limit of $\Omega(r\, n \log n)$. As in Candès and Tao (2010), we assume that the observed entries are sampled at random without replacement, avoiding trivial situations in which a row or a column is unsampled.

The aforementioned result is not applicable in our case since the matrix M is not necessarily of low-rank. Concretely, when $r = O(n)$ the bound above degenerates into a sample complexity of $O(n^2 \log n)$ which is clearly larger than the number of entries in the matrix M. We develop below a variation on the results in Candès and Tao (2010) and Candès and Plan (2010) that applies to the *local-A2* compressed-sensing estimator $\hat{\mathcal{T}}$.

Definition 1. Let X be a metric space. A function $f : X \to \mathbb{R}^{n_1 \times n_2}$ is *Hölder continuous* with parameters $\alpha, \beta > 0$ if

$$\forall x, x' \in X : \ \|f(x) - f(x')\|_F \ \leq \ \alpha\, d^{\beta}(x, x') \ . \tag{10}$$

In our analysis we make the following assumptions: (i) $\mathcal{T}$ is Hölder continuous, (ii) $\mathcal{T}(s)$ is a rank r matrix that satisfies the strong incoherence property, and (iii) the kernel K_h is a uniform kernel based on a product distance function. The Hölder continuity assumption on $\mathcal{T}$ can be replaced by the following weaker condition without affecting the results

$$\|K_h^s \odot (\mathcal{T}(s) - \mathcal{T}(s'))\|_F \ \leq \ \alpha d^{\beta}(s, s'). \tag{11}$$

We denote by $B_h(s)$ the neighborhood of indices near s, $B_h(s) \stackrel{\text{def}}{=} \{s' \in [n_1] \times [n_2] : d(s, s') < h\}$ and we use $n_1(h, s)$ and $n_2(h, s)$ to denote the number of unique row and column indices, respectively, in $B_h(s)$. Finally, we denote $\gamma = \min(n_1(h, s), n_2(h, s))$.

The proposition below provides a bound on the average squared-error within a neighborhood of s

$$\mathcal{E}(\hat{\mathcal{T}})(s, h) = \sqrt{\frac{1}{|B_h(s)|} \sum_{s' \in B_h(s)} \left(\hat{\mathcal{T}}_{s'}(s) - \mathcal{T}_{s'}(s)\right)^2} \ .$$

Proposition 1. *If $|\Omega \cap B_h(s)| \geq C\mu^2 \gamma r \log^6 \gamma$, then with probability of at least $1 - \delta$,*

$$\mathcal{E}(\hat{\mathcal{T}})(s, h) \leq \frac{\alpha h^{\beta}}{\sqrt{|B_h(s)|}} \left(4\sqrt{\frac{\gamma(2+p)}{p}} + 2\right) \ ,$$

where $\gamma = \sqrt[3]{1/\delta}$ and $p = |\Omega \cap B_h(s)|/|B_h(s)|$.

Proof Assumptions (i) and (iii) above imply that if $K_h(s, s') > 0$ then

$$\|K_h^s \odot (\mathcal{T}(s) - \mathcal{T}(s'))\|_{\infty} < \alpha h^{\beta} \ .$$

We can thus assume that if $d(s, s') < h$, an observation $M_{s'} = \mathcal{T}_{s'}(s')$ is equal to $\mathcal{T}_{s'}(s) + Z$ where Z is a random variable whose absolute value is bounded by αh^{β}. This means that we can use observations $M_{s'} = \mathcal{T}_{s'}(s')$ for estimating the local model $\mathcal{T}(s)$ as long as we admit a noisy measurement process.

Since K is a uniform kernel based on a product distance by assumption (iii), the set $B_h(s)$ is a Cartesian product set. We view this product set as a matrix of dimensions $n_1(h,s) \times n_2(h,s)$ that we approximate. (Note that $n_1(h,s)$ and $n_2(h,s)$ are monotonically increasing with h, and as $h \to \infty$, $n_1(h,s) = n_1$, $n_2(h,s) = n_2$.) The number of observed entries in this matrix approximation problem is $|\Omega \cap B_h(s)|$.

Applying Theorem 7 in Candès and Plan (2010) to the matrix completion problem described above, we get that if $|\Omega \cap B_h(s)| \geq C\mu^2 \gamma r \log^6 \gamma$, then with probability greater than $1 - \gamma^{-3}$,

$$\|K_h^s \odot (\mathcal{T}(s) - \hat{\mathcal{T}}(s))\|_F \leq \alpha h^\beta \left(4\sqrt{\frac{\gamma(2+p)}{p}} + 2 \right),$$

where $p = \frac{|\Omega \cap B_h(s)|}{|B_h(s)|}$ is the density of observed samples. Dividing by $\sqrt{|B_h(s)|}$ concludes the proof. ∎

When the observed samples are uniformly spread over the matrix, we get $p = m/(n_1 n_2)$, so

$$\begin{aligned} 4\sqrt{\gamma\frac{2+p}{p}} + 2 &= 4\sqrt{\gamma\frac{2 + m/(n_1n_2)}{m/(n_1n_2)}} + 2 \\ &= 4\sqrt{\frac{\gamma(2n_1n_2 + m)}{m}} + 2. \end{aligned}$$

Multiplying $\alpha h^\beta / \sqrt{|B_h(s)|}$ yields Corollary 1.

Corollary 1. Assume that the conditions of Proposition 1 hold and in addition the observed samples are spread uniformly with respect to d. Then, the following inequality holds

$$\mathcal{E}(\hat{\mathcal{T}})(s,h) \leq \frac{4\alpha h^\beta}{\sqrt{|B_h(s)|}} \sqrt{\frac{\gamma(2n_1n_2 \,+\, m)}{m}} + \frac{2\alpha h^\beta}{\sqrt{|B_h(s)|}}.$$

If in addition the matrix M is squared ($n_1 = n_2 = n$) and the distribution of distances d is uniform, then $n_1(h,s) = n_2(h,s) = n/h$, $|B_h(s)| = (n/h)^2$, and $\gamma = n/h$. In this case, the bound on $\mathcal{E}(\hat{\mathcal{T}})(s,h)$ becomes

$$4\alpha h^{\beta+1/2} \sqrt{\frac{2n}{m} + \frac{1}{n}} + \frac{2\alpha h^{\beta+1}}{n}. \tag{12}$$

In the case of a square matrix with uniformly spread samples, it is instructive to view n, m, h as monotonically increasing sequences, indexed by $k \in \mathbb{N}$ and assume that $\lim_{k\to\infty} n_{[k]} = \lim_{k\to\infty} m_{[k]} = \infty$. In other words, we consider the limit of matrices of increasing sizes with an increasing number of samples. In the case of uniformly distributed distances, the bound (12) will converge to zero if

$$\lim_{k\to\infty} \frac{h_{[k]}^{\beta+1}}{n_{[k]}} = \lim_{k\to\infty} \frac{h_{[k]}^{2\beta+1}}{n_{[k]}} = \lim_{k\to\infty} \frac{h_{[k]}^{2\beta+1} n_{[k]}}{m_{[k]}} = 0.$$

4.2 Analysis of $\hat{\hat{\mathcal{T}}} - \mathcal{T}$

We start by showing that $\hat{\mathcal{T}}$ is Hölder continuous with high probability, and then proceed to analyze the estimation error of $\hat{\hat{\mathcal{T}}}$.

Proposition 2. *If $d(s,s') < h$ and Proposition 1 holds at s, s', then with probability at least $1-\delta$,*

$$\|K_h^s \odot (\hat{\mathcal{T}}(s) - \hat{\mathcal{T}}(s'))\|_F \leq \alpha h^\beta \left(8\sqrt{\frac{\gamma(2+p)}{p}} + 5 \right).$$

where $\gamma = \sqrt[3]{2/\delta}$.

Proof Using the triangle inequality for $\|\cdot\|_F$,

$$\begin{aligned} \|K_h^s \odot (\hat{\mathcal{T}}(s) - \hat{\mathcal{T}}(s'))\|_F &\leq \|K_h^s \odot (\hat{\mathcal{T}}(s) - \mathcal{T}(s))\|_F \\ &+ \|K_h^s \odot (\hat{\mathcal{T}}(s') - \mathcal{T}(s'))\|_F \\ &+ \|K_h^s \odot (\mathcal{T}(s) - \mathcal{T}(s'))\|_F. \end{aligned}$$

We apply the bound from Proposition 1 to the first two terms and use the assumption that $\mathcal{T}$ is Hölder continuous to bound the third term. The adjustment to the confidence level $2\gamma^{-3}$ is obtained using the union bound. ■

Proposition 3. *Assume that Proposition 1 holds. Then, with probability of at least $1-\delta$,*

$$\mathcal{E}(\hat{\hat{\mathcal{T}}})(s,h) \leq \frac{\alpha h^\beta}{\sqrt{|B_h(s)|}} \left(12\sqrt{\frac{\gamma(2+p)}{p}} + 7 \right) .$$

where $\gamma = \sqrt[3]{(2|\Omega \cap B_h(s)| + 1)/\delta}$.

Proof Using the triangle inequality we get

$$\begin{aligned} &\|K_h^s \odot (\hat{\hat{\mathcal{T}}}(s) - \mathcal{T}(s))\|_F \leq \\ &\quad \|K_h^s \odot (\hat{\mathcal{T}}(s) - \mathcal{T}(s))\|_F + \|K_h^s \odot (\hat{\hat{\mathcal{T}}}(s) - \hat{\mathcal{T}}(s))\|_F. \end{aligned} \tag{13}$$

We bound the first term using Proposition 1. Since $\hat{\hat{\mathcal{T}}}(s)$ is a weighted average of $\hat{\mathcal{T}}(s_i)$, $i = 1, \ldots, q$ with $s_i \in B_h(s)$, the second term is bounded by

$$\begin{aligned} &\|K_h^s \odot (\hat{\hat{\mathcal{T}}}(s) - \hat{\mathcal{T}}(s))\|_F \\ &\quad = \left\| K_h^s \odot \left(\sum_i \frac{w_i}{\sum_j w_j} \hat{\mathcal{T}}(s_i) - \hat{\mathcal{T}}(s) \right) \right\|_F \\ &\quad = \left\| K_h^s \odot \sum_i \frac{w_i}{\sum_j w_j} (\hat{\mathcal{T}}(s_i) - \hat{\mathcal{T}}(s)) \right\|_F \\ &\quad \leq \sum_i \left\| \frac{w_i}{\sum_j w_j} K_h^s \odot (\hat{\mathcal{T}}(s_i) - \hat{\mathcal{T}}(s)) \right\|_F \\ &\quad \leq \sum_i \frac{w_i}{\sum_j w_j} \|K_h^s \odot (\hat{\mathcal{T}}(s_i) - \hat{\mathcal{T}}(s))\|_F \ . \end{aligned}$$

There are $|\Omega \cap B_h(s)|$ summands in the above term. We bound each of them using Proposition 2. Together with the bound (13) this gives the desired result (after dividing by $\sqrt{|B_h(s)|}$). The adjustment to the confidence level $(2|\Omega \cap B_h(s)| + 1)\gamma^{-3}$ is obtained using the union bound. ■

5. The Parallel LLORMA Algorithm

In the previous sections, we assumed a general kernel function $K_h(s_1, s_2)$, where $s_1, s_2 \in [n_1] \times [n_2]$. This kernel function may be defined in several ways. For simplicity, we assume a product form $K_h((a, b), (c, d)) = K_{h_1}(a, c)K'_{h_2}(b, d)$ where K and K' are kernels on the spaces $[n_1]$ and $[n_2]$, respectively. We used the Epanechnikov kernel (6) for both K, K' as it achieves the lowest integrated squared error (Wand and Jones, 1995), but other choices are possible as well.

The distance d in (6) can be defined using additional information from an outside source describing row (user) similarity or column (item) similarity. If there is no such information available (as is the case in our experiments), d can be computed solely based on the partially observed matrix M. In that case, we may use any distance measure between two row vectors (for K) or two column vectors (for K'). Empirically, we found that standard distance measures such as the 2-norm or cosine similarity do not perform well when M is sparse.

We therefore instead factorize M using standard incomplete SVD (1) $M \approx UV^\top$. Then, we proceed to compute d based on the distances between the rows of factor matrices U (and V). Concretely, we used arc-cosine between users a and c (and items b and d):

$$d(a, c) = \arccos\left(\frac{\langle U_a, U_c\rangle}{\|U_a\| \cdot \|U_c\|}\right) \quad d(b, d) = \arccos\left(\frac{\langle V_a, V_d\rangle}{\|V_b\| \cdot \|V_d\|}\right) \tag{14}$$

where U_i, V_i are the ith row of the matrix U and V. We tried numerous other distances and similarity scores such as the Euclidean distance and cosine similarity. The arc-cosine score empirically performed better than the other scores we experimented with.

Besides of the distance metric, the anchor points $(s_1, \ldots, s_q)$ that define $\hat{\hat{\mathcal{T}}}$ also play a significant role. There are several ways of choosing the anchor points. We randomly choose among training data points unless stated otherwise. Detailed discussion is on Section 7.3.

Algorithm 1 describes the learning algorithm for estimating the local models at the anchor points $\hat{\mathcal{T}}(s_i)$, with $i = 1, \ldots, q$. In line 14, we solve a weighted (by K_{h_1} and K_{h_2}) SVD problem with L_2 regularization. This minimization problem can be computed with gradient-based methods. After these models are estimated, they are combined using (9) to create the estimate $\hat{\hat{\mathcal{T}}}(s)$ for all $s \in [n_1] \times [n_2]$.

Algorithm 1 can actually run faster than vanilla SVD since (a) the q loops may be computed in parallel, and (b) the rank of the our local models can be significantly lower than the rank of global SVD for similar performance (see Section 7). Also, as the kernel K_h has limited support, (c) $K_h(s, s')$ will have few non-zero entries. The weighted SVD problem at line 14 should be sparser than the global SVD, which should result in an additional speedup.

Algorithm 1 The Parallel LLORMA Algorithm

input: $M \in \mathbb{R}^{n_1 \times n_2}$ whose entries are defined over Ω
parameters: kernel function $K(\cdot)$ of widths h_1 and h_2
rank r and number of local models q
regularization values λ_U, λ_V
for all $t = 1, \ldots, q$ **parallel do**
  select (a_t, b_t) at random from Ω
  for all $i = 1, \ldots, n_1$ **do**
    construct entry i: $[K^{a_t}]_i := K_{h_1}(a_t, i)$
  end for
  for all $j = 1, = \ldots, n_2$ **do**
    construct entry j: $[K^{b_t}]_j := K_{h_2}(b_t, j)$
  end for
  set $(U^{(t)}, V^{(t)})$ to be the minimizer of:

$$\sum_{(i,j)\in\Omega} [K^{(a_t)}]_i \, [K^{(b_t)}]_j \left([U\, V^\top]_{i,j} - M_{i,j} \right)^2 + \lambda_U \sum_{i,k} U_{i,k}^2 + \lambda_V \sum_{j,k} V_{j,k}^2$$

end for
output: $\left\{ a_t,\, b_t,\, U^{(t)}, V^{(t)} \right\}_{t=1}^{q}$

6. Global LLORMA

Recall that the parallel LLORMA algorithm from Section 5 constructs q local models based on q different anchor points. It then combines the models via kernel regression to produce $\hat{M}$ using (9) which yields the following approximation,

$$\hat{M}_{u,i} = \sum_{t=1}^{q} \frac{K(u_t, u)K(i_t, i)}{\sum_s K(u_s, u)K(i_s, i)} [U^{(t)} {V^{(t)}}^\top]_{u,i}. \tag{15}$$

That is, the algorithm learns each local model independently based on different subsets of the matrix (with some potential overlap). Alternatively, we can directly optimize a joint loss using all local models while bearing in mind the form of the final model as given by (15). In this section we describe an algorithmic alternative that minimizes the following loss with respect to $\{U^{(t)}, V^{(t)}\}_{t=1}^{q}$,

$$\sum_{(u,i)\in\Omega} (\hat{M}_{u,i} - M_{u,i})^2 = \sum_{(u,i)\in\Omega} \left(\sum_{t=1}^{q} \frac{K(u_t, u)K(i_t, i)}{\sum_s K(u_s, u)K(i_s, i)} [(U^{(t)} {V^{(t)}}^\top]_{u,i} - M_{u,i} \right)^2 . \tag{16}$$

This optimization problem can be solved using gradient-based methods, as we use for the global incomplete SVD. Hence, we solve jointly multiple SVD problems with different weights $(K(u_t, u)K(i_t, i))$ multiplying the original matrix. Since we can decompose M into weighted sums,

$$M_{u,i} = \frac{\sum_t K(u_t, u)K(i_t, i)}{\sum_s K(u_s, u)K(i_s, i)} M_{u,i} \ ,$$

the objective function given by (16) can be rewritten as follows,

$$\sum_{(u,i)\in\Omega}\left(\sum_{t=1}^{q}\frac{K(u_t,u)K(i_t,i)}{\sum_s K(u_s,u)K(i_s,i)}[(U^{(t)}V^{(t)\top}]_{u,i} - \sum_{t=1}^{q}\frac{K(u_t,u)K(i_t,i)}{\sum_s K(u_s,u)K(i_s,i)}M_{u,i}\right)^2$$
$$= \sum_{(u,i)\in\Omega}\left(\sum_{t=1}^{q}\frac{K(u_t,u)K(i_t,i)}{\sum_s K(u_s,u)K(i_s,i)}\left([(U^{(t)}V^{(t)\top}]_{u,i} - M_{u,i}\right)\right)^2 . \quad (17)$$

Let us now examine the difference between the above objective with the one tacitly employed by parallel LLORMA algorithm, which amounts to,

$$\sum_{(u,i)\in\Omega}\sum_{t=1}^{q}K(u_t,u)K(i_t,i)\left([(U^{(t)}V^{(t)\top}]_{u,i} - M_{u,i}\right)^2 . \quad (18)$$

By construction, both objectives are minimized with respect to $\{U^{(t)}, V^{(t)}\}_{t=1}^{q}$ using the squared deviation from M. The difference between the two models is that (17) has a square that encompasses a sum over anchor points. When expanded the term includes the individual squared terms that appear in (18) as well as additional interaction terms. Namely, parallel LLORMA minimizes the sum of square deviations while global LLORMA minimizes the square deviation of sums. In Algorithm 2 we provide the pseudocode of global LLORMA.

A priori we should expect the global version of LLORMA to result in more accurate estimates of M than parallel LLORMA described in Section 5. However, since the objective can no longer be decoupled, the run time global LLORMA is likely to be longer than its parallel counterpart. We provide experimental results which compare the two versions in terms of performance and running time in Section 7.2.

7. Experiments

We conducted several experiments with recommendation data. In Section 7.1, we compare LLORMA to SVD and other state-of-the-art techniques. We also examine in the section dependency of LLORMA on the rank r, the number of anchor points q, and the training set size. In Section 7.2, we compare the parallel and global versions of LLORMA. Section 7.3 introduces several anchor point selection schemes and compare them experimentally.

We used four popular recommendation systems datasets. The MovieLens[1] dataset is one of the most popular datasets in the literature. We used all versions of MovieLens dataset, namely: 100K ($1K \times 2K$ with 10^5 observations), 1M ($6K \times 4K$ with 10^6 observations), and 10M ($70K \times 10K$ with 10^7 observations). We also tested LLORMA on the Netflix dataset which is of size $480K \times 18K$ with 10^8 observations and the Bookcrossing dataset ($100K \times 300K$ with 10^6 observations). These two datasets are much larger than the MovieLens dataset. We also report results on the Yelp dataset ($40K \times 10K$ with 10^5 observations), which is a recent dataset that is part of the ACM RecSys 2013 challenge[2]. The Bookcrossing

1. http://www.grouplens.org/
2. http://recsys.acm.org/recsys13/recsys-2013-challenge-workshop/

Algorithm 2 The Global LLORMA Algorithm

input: $M \in \mathbb{R}^{n_1 \times n_2}$ whose entries are defined over Ω
parameters: kernel function $K(\cdot)$ of widths h_1 and h_2
rank r and number of local models q
regularization values λ_U, λ_V
for all $t = 1, \ldots, q$ **do**
  select (a_t, b_t) at random from Ω
  for all $i = 1, \ldots, n_1$ **do**
    construct entry i: $[K^{a_t}]_i := K_{h_1}(a_t, i)$
  end for
  for all $j = 1, = \ldots, n_2$ **do**
    construct entry j: $[K^{b_t}]_j := K_{h_2}(b_t, j)$
  end for
end for
minimize with respect to $\{(U^{(t)}, V^{(t)})\}_{t=1}^{q}$:

$$\sum_{(i,j)\in\Omega} \left(\sum_{t=1}^{q} \frac{[K^{(a_t)}]_i \, [K^{(b_t)}]_j \, [U^{(t)} {V^{(t)}}^\top]_{i,j}}{\sum_s [K^{(a_s)}]_i \, [K^{(b_s)}]_j} - M_{i,j} \right)^2 + \lambda_U \sum_{i,k} \left[U_{i,k}^{(t)} \right]^2 + \lambda_V \sum_{j,k} \left[V_{j,k}^{(t)} \right]^2$$

Output: $\left\{ a_t,\, b_t,\, U^{(t)}, V^{(t)} \right\}_{t=1}^{q}$

and Yelp datasets reflect a recent trend of very high sparsity, often exhibited in real-world recommendation systems.

Unless stated otherwise, we randomly divided the available data into training and test sets such that the ratio of training set size to test set size was 9:1. We created five random partitions and report the average performance over the five partitions. We used a default rating of $(max + min)/2$ for test users or items which lack any rating, where max and min indicate respectively the maximum and minimum possible rating in the dataset.

In our experiments, we used the Epanechnikov kernel with $h_1 = h_2 = 0.8$, a fixed step-size for gradient descent of $\mu = 0.01$, and a 2-norm regularization value of $\lambda_U = \lambda_V = 0.001$. These values were selected using cross-validation. We set and did not attempt to optimize the parameters $T = 100$ (maximum number of iterations), $\epsilon = 0.0001$ (gradient descent convergence threshold), and $q = 50$ (number of anchor points). We selected anchor points by sampling uniformly users and items from the training points without replacement. We examine more complex anchor point selection scheme in Section 7.3.

7.1 Performance of Parallel LLORMA

Table 2 lists the performance of LLORMA with 50 anchor points, SVD, and two recent state-of-the-art methods based on results published in (Mackey et al., 2011). For a fixed rank r, LLORMA always outperforms SVD. Both LLORMA and SVD perform better as r increases. Both SVD and LLORMA exhibit diminishing returns as the rank increases. LLORMA with a modest rank $r = 5$ outperforms SVD of any rank. We can see that LLORMA also outperforms the Accelerated Proximal Gradient (APG) and Divide-and-Conquer Matrix Factorization (DFC) algorithms. For a reference, the Root Mean Square

Method	MovieLens 1M		MovieLens 10M		Netflix		Yelp		Bookcrossing	
APG	–		0.8005		0.8433		–		–	
DFC-NYS	–		0.8085		0.8486		–		–	
DFC-PROJ	–		0.7944		0.8411		–		–	
Rank	SVD	LLORMA	SVD	LLORMA	SVD	LLORMA	SVD	LLORMA	SVD	LLORMA
Rank-1	0.9201	0.9135	0.8723	0.8650	0.9388	0.9295	1.4988	1.4490	3.3747	3.1683
Rank-3	0.8838	0.8670	0.8348	0.8189	0.8928	0.8792	1.4824	1.3133	3.3679	3.0315
Rank-5	0.8737	0.8537	0.8255	0.8049	0.8836	0.8604	1.4775	1.2358	3.3583	2.9482
Rank-7	0.8678	0.8463	0.8234	0.7950	0.8788	0.8541	1.4736	1.1905	3.3488	2.8828
Rank-10	0.8650	0.8396	0.8219	0.7889	0.8765	0.8444	1.4708	1.1526	3.3283	2.8130
Rank-15	0.8652	0.8370	0.8225	0.7830	0.8758	0.8365	1.4685	1.1317	3.3098	2.7573
Rank-20	0.8647	0.8333	0.8220	0.7815	0.8742	0.8337				

Table 2: RMSE achieved by different algorithms on five datasets: MovieLens 1M, MovieLens 10M, Netflix, Bookcrossing, and Yelp. Results for APG (Toh and Yun, 2010) and DFC were taken from (Mackey et al., 2011).

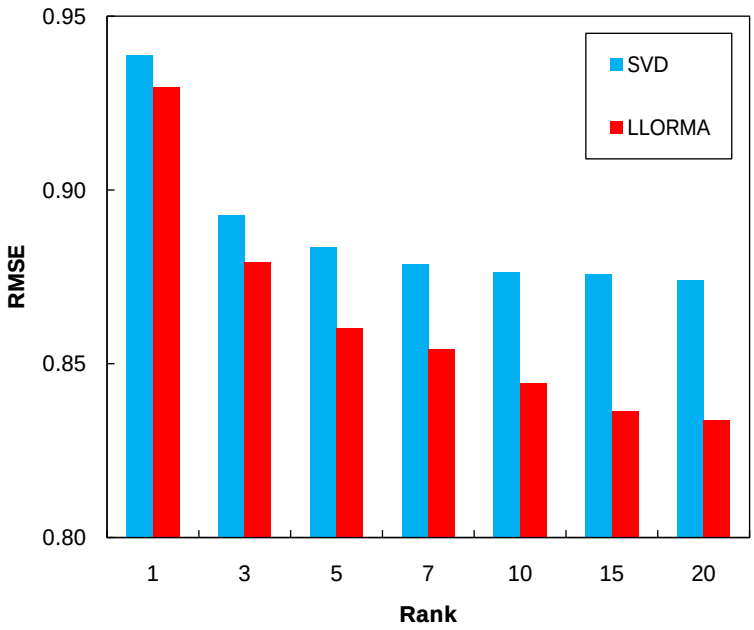

Figure 3: RMSE of LLORMA and SVD as a function of the rank on the Netflix dataset.

Error (RMSE) we achieved (0.8337) is a better score than the goal of Netflix competition (0.8567).[3]

As we can see from Figure 3, the improvement in the performance of SVD is rather minor beyond a rank of 7 while LLORMA's improvement is still evident until a rank of about 20. Both approaches cease to make substantial improvements beyond the aforementioned ranks and exhibit diminishing returns for high ranks. As discussed in earlier sections, the diminishing returns of SVD for high ranks can be interpreted in two ways: (i) The

3. We provide this number merely as reference since we measured our performance on a test set different from original Netflix test set, which is no longer available. Our result are based on a random sub-sampling of the Netflix training data as described above. See also the discussion in (Mackey et al., 2011).

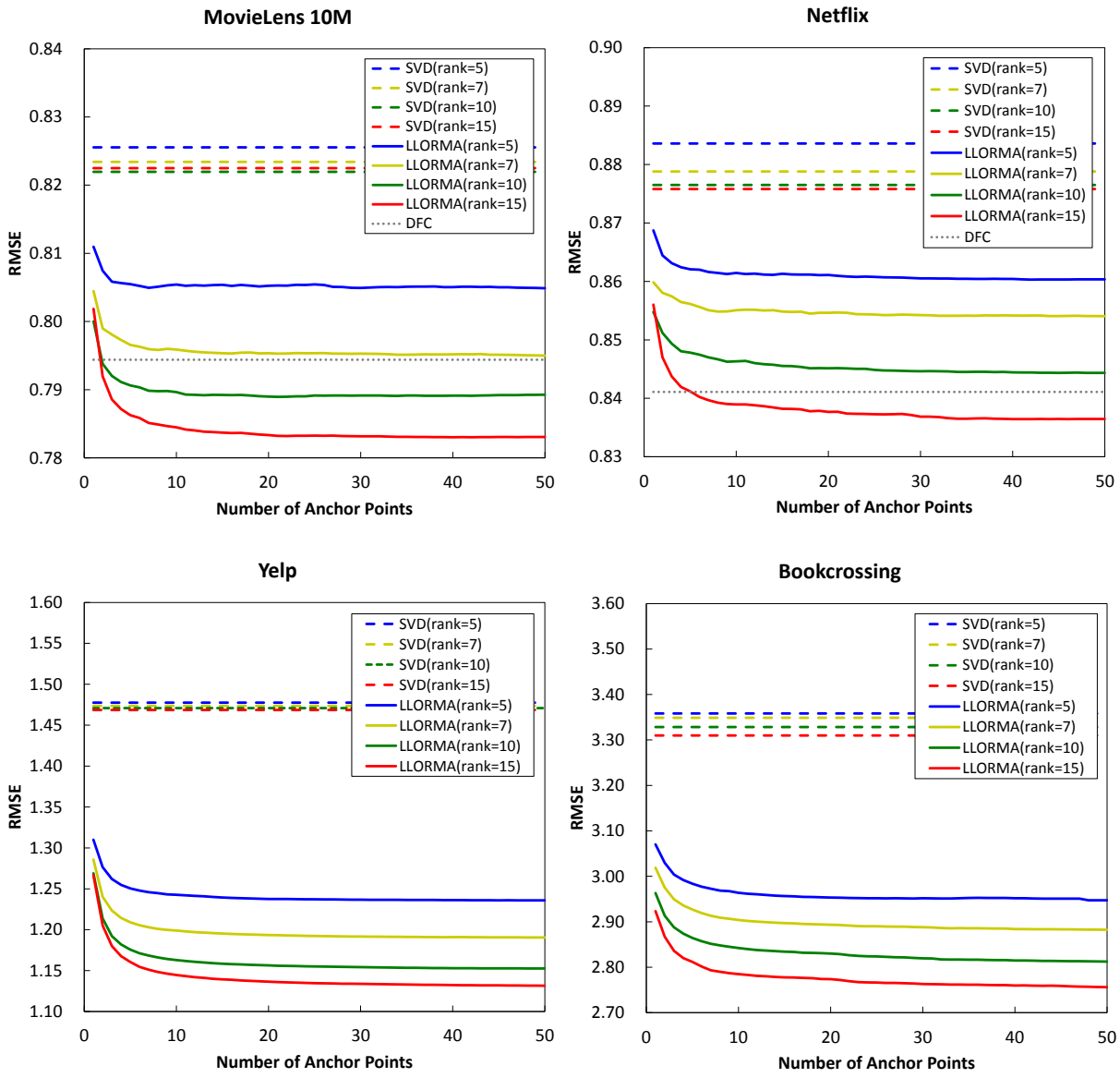

Figure 4: RMSE of LLORMA, SVD, and two baselines on MovieLens 10M (top-left), Netflix (top-right), Yelp (bottom-left), and Bookcrossing (bottom-right) datasets. The results for LLORMA are depicted by thick solid lines, while for SVD with dotted lines. Models of the same rank are have identical colors.

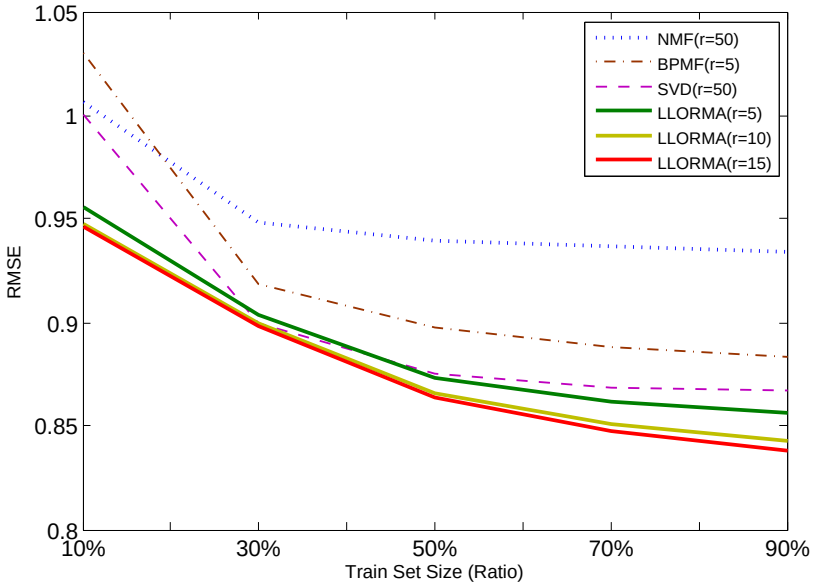

Figure 5: RMSE of SVD, LLORMA, NMF, and BPMF methods as a function of training set size for the MovieLens 1M dataset.

global low-rank assumption is correct and the SVD approximates the rating matrix almost optimally. (ii) The global low-rank assumption is incorrect and the diminishing returns are due to various deficiencies such as overfitting or reaching an inferior local minimum. Since LLORMA improves beyond SVD's best performance, it deems that the second hypothesis is more plausible. The fact that LLORMA's performance asymptotes at higher ranks than SVD may indicate the first hypothesis is to a large extent correct at a local region yet not globally.

Figure 4 compares the RMSE of LLORMA, SVD, and the DFC method of Mackey et al. (2011). We plot the RMSE of LLORMA as a function of the number of anchor points. As in the case of Table 2, both LLORMA and SVD improve as r increases. Here again LLORMA with local rank of at least 5 outperforms SVD of any rank. Moreover, LLORMA outperforms SVD even with only a few anchor points. Figure 5 shows the RMSE of LLORMA as a function of the training set size, and compares it with global SVD of 50. We also plot results for a few other methods that have shown to achieve very good approximation accuracy: non-negative matrix factorization (NMF) with rank 50 (Lee and Seung, 2001) and Bayesian probabilistic matrix factorization (BPMF) with rank 5 (Salakhutdinov and Mnih, 2008b). The test set size was fixed to 10% of the MovieLens 1M and the RMSE was averaged over five random train-test splits. The graph shows that all methods improve as the training set size increases while LLORMA consistently outperforms SVD and the other baselines.

To recap, the experimental results presented thus far indicate that LLORMA outperforms SVD and other state-of-the-art methods even when using relatively lower-rank ap-

proximation. Moreover, LLORMA is capable of achieving good accuracy with rather small number of anchor points and seems to perform well across a variety of training set sizes.

7.2 Comparison of Global and Parallel LLORMA

We proposed two implementations of LLORMA in this paper: a decoupled parallel approach (Section 5) and a global approximation version (Section 6). In earlier sections, we conjectured that the global version is likely to be more accurate in terms of RMSE as it directly optimizes the objective function. In terms of computational efficiency, however, we naturally expected the parallel version to be faster as it can take advantage of multicore and distributed computing architectures. In this subsection, we experimentally verify the two conjectures.

We compared the two versions of LLORMA on MovieLens 100K dataset. We tested local rank values in $\{1, 3, 5, 7, 10\}$. The experiment was conducted on a quad-core machine with 4 threads while suspending any other process. For both versions, we constructed 50 local models and repeated the experiment 5 times with different anchor points. Table 3 reports the average test RMSE and average elapsed time for training. As conjectured, global LLORMA results in more accurate estimations on unseen data than parallel LLORMA. However, the performance gap between the two approaches reduces as the rank increases. The parallel version LLORMA runs about 3 times faster than global LLORMA indicating that a fairly high utilization of the multicore architecture.

Method	Global LLORMA		Parallel LLORMA	
Rank	Test RMSE	Time	Test RMSE	Time
1	0.9072	6:04	0.9426	1:09
3	0.9020	10:27	0.9117	3:20
5	0.8990	14:40	0.9041	5:26
7	0.8986	19:43	0.9010	7:50
10	0.8975	28:59	0.8985	11:49

Table 3: RMSE and training time for Global and Parallel LLORMA on MovieLens 100K.

7.3 Anchor Points Selection

The method for selecting anchor points in LLORMA is important as it may affect the prediction time as well as generalization performance. If the row and column indices of the test data are provided in advance, we may choose anchor points from the test set distribution. However, in most applications the test set is not known apriori, and in fact is likely to increase and change in time. We therefore confined ourselves to three sampling methods and one clustering methods based on low-rank representation of the data. In the rest of the section we use q to denote the number of anchor points. The following are anchor point selection methods we tried.

Complete: Sample anchor points uniformly from the entire set of indices $[n_1] \times [n_2]$.

Trainset: Sample anchor points uniformly from the observed entries, Ω.

Coverage: Select anchor points such that no entry in $[n_1] \times [n_2]$ is too distant from the rest of the anchor points.

k-means: Run k-means clustering on the entries in Ω each represented using the induced d-dimensional space obtained by SVD with $k = q$.

The first two sampling methods are straightforward. The third method, termed "Coverage" was implemented by sampling aq with $a > 1$ user-item pairs and then adding an anchor point whose minimum distance to existing anchor points is the largest. The fourth selection methods that we tested is based on clustering the observed entries. It is based on the observation that an anchor point need not be an entry in the observation matrix but may rather consist of a combination of matrix entries. Recall that we weigh each local model based on user and item similarity. As explained in Section 5, each a user (item) is represented as a row of a low-rank matrices U (respectively, V) attained by the global SVD, namely, $M \approx UV^\top$. Denoting the intrinsic dimension of U and V by d, each user and item can be represented as a d-dimensional vector. Instead of each row of U and V, we can generalize the space of anchor points to any point in this d-dimensional vector space. A good set of anchor points may be the q cluster centers of the d-dimensional representations of the entries of M which is computed using the k-means algorithm.

Table 4 provides performance results of *Global* LLORMA using different anchor point selection methods. In the experiments we used the MovieLens 100K datasets with 5 random train-test partitions. The results we report are based on averages over the 5 random splits. As one may anticipate, the different selection schemes perform similarly when q is large. For small values of q (10 or 20), we would like to underscore the following observations. The first two methods (Complete and Trainset) perform similarly. The clustering-based anchor point construction generally performs slightly better than the three other methods. Somewhat surprisingly, the Coverage method performs the worst for small values of q. Nonetheless, when q is sufficiently large all methods achieve about the same results.

	10 Local models				20 Local Models			
Rank	Complete	Trainset	Coverage	k-means	Complete	Trainset	Coverage	k-means
1	0.9276	0.9296	0.9360	0.9230	0.9195	0.9206	0.9235	0.9166
3	0.9245	0.9237	0.9327	0.9208	0.9164	0.9152	0.9189	0.9134
5	0.9240	0.9221	0.9277	0.9190	0.9125	0.9128	0.9154	0.9111
7	0.9231	0.9251	0.9279	0.9202	0.9140	0.9129	0.9163	0.9103
10	0.9242	0.9271	0.9301	0.9236	0.9129	0.9125	0.9141	0.9112

Table 4: RMSE for various anchor point selection schemes on MovieLens 100K datset.

7.4 Comparison to Ensemble of Independent Models

The algebraic form of the parallel estimator $\hat{\hat{T}}$ as given by (9) is a linear combination of local models, each of which focuses on a subset of user-item pairs. This algebraic form is reminiscent of ensemble methods (Jacobs et al., 1991) such as Bagging (Breiman, 1996), where the final model is a linear combination of simple models, each weighed by a predefined coefficient. Ensemble methods such as boosting and Bagging have been shown to be effective tools for combining models primarily for classification tasks. In this section we examine the

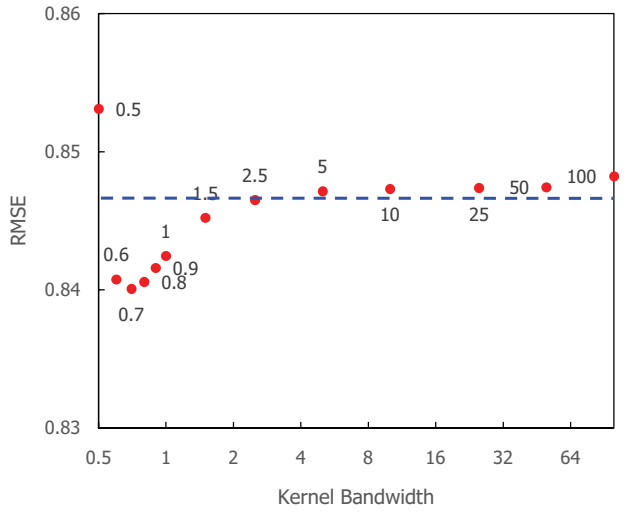

Figure 6: Performance of LLORMA as a function of the kernel width. The vertical axis indicates the approximations' root mean squared error (RMSE) and the horizontal axis the kernel width ($h_1 = h_2$) in log scale. The dotted blue line shows the average RMSE of Bagging for reference.

connection between LLORMA and an ensemble method based on Bagging for low rank matrix approximation.

There are two main differences between Bagging and LLORMA. In Bagging, the dataset constructed for training each sub-model is uniformly sampled with replacements. In contrast, LLORMA's sampling is based on a non-uniform distribution respecting locality as defined by the distance metric over user-item pairs. The second difference is that Bagging assigns equal weights to each base-model. In LLORMA, each *local* model is associated with a weight that is proportional to the proximity of the anchor point to the test point. That is, the weights of the local models vary and are determined at inference time.

We conducted two set of experiments comparing LLORMA with Bagging. In the first experiment, we varied the kernel widths gradually increasing the widths to the matrix dimensions, thus ending with a uniform kernel over Ω. We normalized the kernel width so that a value of 1 corresponds to the full dimension. As the width increases, the overlap between local models becomes more and more substantial. In addition, the bias of each local model increases due to the decrease in locality. Analogously, the variance of each model decreases due to the increase in the actual training set size.

Figure 6 shows performance of LLORMA on MovieLens 100K dataset for various kernel widths. The best performance is obtained for kernel width between 0.7 and 0.8. The performance rapidly deteriorates as the width decreases and slowly degrades as the width increases with an optimal width close to the middle of the range.

In our second experiment, we compared LLORMA with Bagging by taking $|\Omega|$ samples with replacements, which in expectation covers two thirds of the observed entries. Table 5 compares the performance of global SVD of rank 10 and 15, LLORMA with 100 local

models, and Bagging with 100 models. Each result is the average of 5 random splits of the dataset. It is apparent from the table that both LLORMA and Bagging outperform global SVD. Further, LLORMA achieves lower RMSE than Bagging. The improvement of LLORMA over Bagging is statistically significant based on a paired t-test with p-values of 0.0022 for MovieLens 100K and 0.0014 for MovieLens 1M. These p-values correspond to a confidence level over 99%. LLORMA also outperforms Bagging with respect to the median average error (MAE).

Dataset	MovieLens 100K		MovieLens 1M	
Method	MAE	RMSE	MAE	RMSE
SVD rank=10	0.7189	0.9108	0.6922	0.8683
SVD rank=15	0.7170	0.9094	0.6913	0.8676
Bagging	0.6985	0.8930	0.6620	0.8481
LLORMA	0.6936	0.8881	0.6577	0.8423

Table 5: Comparison of the median average error (MAE) and root mean squared error (RMSE) for SVD, Bagging, and LLORMA on MovieLens dataset.

8. Related work

Matrix factorization for recommender systems have been the focus of voluminous amount of research especially since the Netflix Prize competition. It is clearly impossible to review all of the existing approaches. We review here a few of the notable approaches. Billsus and Pazzani (1998) initially proposed applying SVD for collaborative filtering problems. Salakhutdinov and Mnih (2008a) presented probabilistic matrix factorization (PMF) and later Salakhutdinov and Mnih (2008b) extended matrix factorization to fully Bayesian approach. Lawrence and Urtasun (2009) proposed a non-linear version of PMF. Rennie and Srebro (2005) proposed a maximum-margin approach. Lee et al. (2012b) conducted a comprehensive experimental study comparing a number of state-of-the-art and traditional recommendation system methods using the PREA toolkit (Lee et al., 2012c). Further algorithmic improvements in matrix completion were demonstrated in Toh and Yun (2010); Keshavan et al. (2010). The work that is perhaps the most similar in to LLORMA is Divide-and-Conquer Matrix Factorization (DFC) (Mackey et al., 2011). DFC also divides the completion problems into a set of smaller matrix factorization problems. Our approach differs DFC in that we use a metric structure on $[n_1] \times [n_2]$ and use overlapping partitions. Another matrix approximation by sub-division based on clustering was reported in Mirbakhsh and Ling (2013) for the task of seeking user and item communities. In addition to monolithic matrix factorization scheme, several ensemble methods have also been proposed. DeCoste (2006) suggested ensembles of maximum margin matrix factorization (MMMF). The Netflix Prize winner (Bell et al., 2007; Koren, 2008) used combination of memory-based and matrix factorization methods. The Netflix Prize runner-up (Sill et al., 2009) devised Feature-Weighted Least Square (FWLS) solver, using a linear ensemble of learners with dynamic weights. Lee et al. (2012a) extended FWLS by introducing automatic stage-wise feature induction. Kumar et al. (2009) and Mackey et al. (2011) applied ensembles to Nys-

trom method and DFC, respectively. Other local learning paradigms were suggested in the context dimensionality such as local principal component analysis (Kambhatla and Leen, 1997) and local linear embedding (LLE) (Roweis and Saul, 2000). A relatively recent paper on matrix completion (Wang et al., 2013) applies low-rank factorization to clusters of points. Last, we would like to point to the formal work on low-rank matrix Completion that is closest to the analysis presented in this paper. Candès and Tao (2010) derived a bound on the performance of low-rank matrix completion. As mentioned in previous section our analysis is based on (Candès and Plan, 2010) who adapted the analysis of Candès and Tao (2010) to noisy settings. Some more remote related results were presented in (Shalev-Shwartz et al., 2011; Foygel and Srebro, 2011; Foygel et al., 2012).

9. Summary

We presented a new approach for low-rank matrix approximation based on the assumption that the matrix is locally low-rank. Our proposed algorithm, called LLORMA, is highly parallelizable and thus scales well with the number of observations and the dimension of the problem. Our experiments indicate that LLORMA outperforms several state-of-the-art methods without a significant computational overhead. We also presented a formal analysis of LLORMA by deriving bounds that generalize standard compressed sensing results and express the dependency of the modeling accuracy on the matrix size, training set size, and locality (kernel bandwidth parameter). Our method is applicable beyond recommendation systems so long as the locality assumption holds and a reasonable metric space can be identified.

Acknowledgments

We would like to thank Le Song for insightful discussions. Part of this work was done while the first and second authors were in Georgia Institute of Technology.

References

R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proc. of the ACM SIGKDD*, 2007.

D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Proc. of the International Conference on Machine Learning*, 1998.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

E.J. Candès and Y. Plan. Matrix completion with noise. *Proc. of the IEEE*, 98(6):925–936, 2010.

E.J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

D. DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proc. of the ICML*, 2006.

R. Foygel and N. Srebro. Concentration-based guarantees for low-rank matrix reconstruction. *ArXiv Report arXiv:1102.3923*, 2011.

R. Foygel, N. Srebro, and R. Salakhutdinov. Matrix reconstruction with the local max norm. *ArXiv Report arXiv:1210.5196*, 2012.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Nandakishore Kambhatla and Todd K Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9(7):1493–1516, 1997.

R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 99:2057–2078, 2010.

Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

S. Kumar, M. Mohri, and A. Talwalkar. Ensemble nystrom method. In *Advances in Neural Information Processing Systems*, 2009.

N. D. Lawrence and R. Urtasun. Non-linear matrix factorization with gaussian processes. In *Proc. of the International Conference on Machine Learning*, 2009.

D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, 2001.

J. Lee, M. Sun, S. Kim, and G. Lebanon. Automatic feature induction for stagewise collaborative filtering. In *Advances in Neural Information Processing Systems*, 2012a.

J. Lee, M. Sun, and G. Lebanon. A comparative study of collaborative filtering algorithms. *ArXiv Report 1205.3193*, 2012b.

J. Lee, M. Sun, and G. Lebanon. Prea: Personalized recommendation algorithms toolkit. *Journal of Machine Learning Research*, 13:2699–2703, 2012c.

L. W. Mackey, A. S. Talwalkar, and M. I. Jordan. Divide-and-conquer matrix factorization. In *Advances in Neural Information Processing Systems*, 2011.

N. Mirbakhsh and C. X. Ling. Clustering-based matrix factorization. *ArXiv Report arXiv:1301.6659*, 2013.

J.D.M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proc. of the International Conference on Machine Learning*, 2005.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, 2008a.

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proc. of the International Conference on Machine Learning*, 2008b.

S. Shalev-Shwartz, A. Gonen, and O. Shamir. Large-scale convex minimization with a low-rank constraint. In *Proc. of the International Conference on Machine Learning*, 2011.

J. Sill, G. Takacs, L. Mackey, and D. Lin. Feature-weighted linear stacking. *Arxiv preprint arXiv:0911.0460*, 2009.

K.C. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific Journal of Optimization*, 6(15):615–640, 2010.

M. P. Wand and M. C. Jones. *Kernel Smoothing*. Chapman and Hall/CRC, 1995.

Yi Wang, Arthur Szlam, and Gilad Lerman. Robust locally linear analysis with applications to image denoising and blind inpainting. *SIAM Journal on Imaging Sciences*, 6(1):526–562, 2013.