A Probabilistic Approach to Robust Matrix Factorization

Naiyan Wang[1], Tiansheng Yao[2,*], Jingdong Wang[3], and Dit-Yan Yeung[1]

[1] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China
[2] Computer Science Department, University of California, Los Angeles
[3] Microsoft Research Asia, No. 5 Danling Street, Haidian District, Beijing 100080, China
winsty@gmail.com, tsyao@cs.ucla.edu, jingdw@microsoft.com, dyyeung@cse.ust.hk

Abstract. Matrix factorization underlies a large variety of computer vision applications. It is a particularly challenging problem for large-scale applications and when there exist outliers and missing data. In this paper, we propose a novel probabilistic model called Probabilistic Robust Matrix Factorization (PRMF) to solve this problem. In particular, PRMF is formulated with a Laplace error and a Gaussian prior which correspond to an ℓ_1 loss and an ℓ_2 regularizer, respectively. For model learning, we devise a parallelizable expectation-maximization (EM) algorithm which can potentially be applied to large-scale applications. We also propose an online extension of the algorithm for sequential data to offer further scalability. Experiments conducted on both synthetic data and some practical computer vision applications show that PRMF is comparable to other state-of-the-art robust matrix factorization methods in terms of accuracy and outperforms them particularly for large data matrices.

1 Introduction

Matrix factorization (a.k.a. matrix decomposition) is a fundamental topic in linear algebra and numerical analysis. It also underlies many applications in computer vision and pattern recognition, e.g., structure from motion (SfM) [1] and non-rigid 3D reconstruction [2], which can naturally be formulated as low-rank matrix factorization problems. In a typical low-rank matrix factorization problem, we seek to approximate some given data matrix by the product of two or more smaller matrices such that the difference between the matrix and its factorized form is minimized with respect to some optimality criterion suitable for the problem at hand. Due to data sparsity in many applications, this optimization problem is often cast under a regularization framework by introducing an appropriate regularizer to prevent overfitting from occurring.

One common optimality criterion is the squared error or squared loss which is also known as the ℓ_2 loss. Singular value decomposition (SVD) is a conventional method often used for solving the low-rank matrix factorization problem by minimizing the squared error criterion. In many real-world applications, however, it is not unusual to find noise, outliers and even missing entries in the data matrices. Under such circumstances, it has been shown that the ℓ_2 loss lacks robustness. To overcome this problem,

[*] The work was done when Tiansheng Yao was an undergraduate student in Zhejiang University.

some robust matrix factorization methods have been developed under the framework of minimizing the ℓ_2 loss, e.g., [3].

An alternative optimality criterion is the ℓ_1 loss which corresponds to the least absolute deviations technique. One promising property of this criterion is its robustness against outliers in the data. However, the non-smooth nature of the ℓ_1 loss function poses challenges to many optimization methods. Moreover, most of the existing factorization methods [4,5] that minimize the ℓ_1 loss are based on linear programming techniques which can incur high computational demand, making them unattractive for handling large-scale data sets.

Inspired by previous work on probabilistic matrix factorization based on the ℓ_2 loss [6], we propose in this paper a probabilistic method for robust matrix factorization based on the ℓ_1 loss and ℓ_2 regularizer, which bear duality with the Laplace error and Gaussian prior, respectively. For model learning, we devise an efficient expectation-maximization (EM) algorithm by exploiting a hierarchical representation of the Laplace distribution as a scaled mixture of Gaussians. We also present an online extension of the learning algorithm.

It is worth noting that our model is closely related to the Robust Principal Component Analysis (RPCA) model [7,8]. Their relationship can be revealed by exploiting the equivalence between the nuclear norm and the ℓ_2-norm. As such, our method can also be applied to the many computer vision applications to which RPCA has been applied.

The contributions of this paper are summarized below:

1. We propose a Probabilistic Robust Matrix Factorization (PRMF) model based on the ℓ_1 loss for robust low-rank matrix factorization in the presence of missing data and outliers. In addition, we prove that PRMF is equivalent to RPCA under mild conditions.
2. We devise a parallelizable EM algorithm for model learning which can potentially be applied to large-scale applications. Moreover, we propose an online extension of the algorithm for sequential data to offer further scalability.

2 Related Work

Matrix factorization under situations in which the data matrix is corrupted by outliers is a computational problem that often arises in many computer vision applications. It is well aware that the ℓ_2 loss is far from satisfactory due to its high sensitivity to outliers. Recent years have witnessed various attempts to seek more robustness alternatives. Among these, the ℓ_1 loss has drawn arguably the most attention. Ke and Kanade [4] developed a robust matrix factorization method with a weighted ℓ_1 loss and used linear programming in each iteration of the optimization problem. Eriksson and Hengel [5] extended the Wiberg algorithm originally developed for matrix factorization based on the ℓ_2 loss to matrix factorization based on the ℓ_1 loss. They also used linear programming in each iteration. As these methods require solving multiple linear programming problems, one for each iteration, they have high computational cost and hence cannot cope with large-scale data sets. Moreover, since these methods are not formulated under the regularization framework, overfitting of the data is inevitable.

As a parallel thread of development in recent years, probabilistic formulations of matrix factorization have been proposed. Representative methods taking this approach include Probabilistic Matrix Factorization (PMF) [6] and Bayesian Probabilistic Matrix Factorization (BPMF) [9]. PMF is defined based on the Gaussian error and Gaussian priors, making it equivalent to using the ℓ_2 loss, while BPMF tries to improve PMF by seeking a fully Bayesian treatment. Although both methods have achieved great successes, they are still sensitive to outliers. Some attempts have been made recently to address the robustness issue. For instance, Lakshminarayana *et al.* proposed the Robust Bayesian Matrix Factorization (RBMF) model [10] for collaborative filtering applications.

Robust matrix factorization is also related to some recent advances in Robust Principal Component Analysis (RPCA) [8,7]. The underlying assumption of RPCA is that the observed data matrix has an additive structure with a low-rank component and a separate sparse component. From a numerical optimization perspective, an efficient augmented Lagrangian method was applied to solve the optimization problem for RPCA. Subsequent work [11,12] led to even more remarkable speedup. An online version of RPCA was also proposed very recently [13]. Besides, RPCA has also been studied from a probabilistic perspective. A fully Bayesian treatment was proposed in [14] by modeling the sparseness via the Bernoulli distribution, but inference in the model is very slow due to its high model complexity.

3 Background

3.1 Notations

In this paper, $\mathbf{I}_m$ denotes an $m\times m$ identity matrix. For matrices $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{m\times n}$ and $\mathbf{B} = [b_{ij}] \in \mathbb{R}^{p\times q}$, $\mathbf{A}\odot\mathbf{B} = [a_{ij}b_{ij}]_{m\times n}$ denotes the Hadamard product of $\mathbf{A}$ and $\mathbf{B}$ when $m = p$ and $n = q$. We refer to several matrix norms in this paper with $\|\mathbf{A}\|$ denoting an arbitrary matrix norm of $\mathbf{A}$. More specific matrix norms include the ℓ_1 norm $\|\mathbf{A}\|_1 = \sum_{ij} |a_{ij}|$, ℓ_2 norm (or Frobenius norm) $\|\mathbf{A}\|_2 = (\sum_{ij} a_{ij}^2)^{1/2}$, and nuclear norm (or trace norm) $\|\mathbf{A}\|_*$. As for vectors, we assume that all are column vectors.

3.2 Matrix Factorization

Given a data matrix $\mathbf{Y} = [y_{ij}] \in \mathbb{R}^{m\times n}$ possibly with some values missing, matrix factorization can be formulated as the following optimization problem:

$$\min_{\mathbf{U}\in\mathbb{R}^{m\times r},\ \mathbf{V}\in\mathbb{R}^{n\times r}} \|\mathbf{W}\odot(\mathbf{Y}-\mathbf{U}\mathbf{V}')\|_a^a, \tag{1}$$

where $\mathbf{W} = [w_{ij}]$ is an $m\times n$ binary matrix to cater for the missing values in $\mathbf{Y}$, with $w_{ij} = 1$ if y_{ij} is available and $w_{ij} = 0$ if y_{ij} is missing. When a is either 1 or 2, it corresponds to using the ℓ_1 or ℓ_2 norm, respectively, as loss function.

Since $\mathbf{U}\mathbf{A}^{-1}\mathbf{A}\mathbf{V}' = \mathbf{U}\mathbf{V}'$ holds for any $r\times r$ nonsingular matrix $\mathbf{A}$, the problem in Equation (1) is usually cast under a regularization framework to make the solution identifiable and avoid overfitting. One typical choice is to impose penalty on both $\mathbf{U}$ and $\mathbf{V}$ to give the following regularized matrix factorization problem:

$$\min_{\mathbf{U},\,\mathbf{V}} \; \|\mathbf{W} \odot (\mathbf{Y} - \mathbf{U}\mathbf{V}')\|_a^a + \frac{\lambda_u}{2}\|\mathbf{U}\|_2^2 + \frac{\lambda_v}{2}\|\mathbf{V}\|_2^2, \tag{2}$$

where $\lambda_u, \lambda_v > 0$ are regularization parameters.

4 Probabilistic Robust Matrix Factorization

In this section, we present the methodology of our paper. We first give a probabilistic interpretation of robust matrix factorization in Section 4.1 and then state its relationship with RPCA in Section 4.2. For computational advantages, we reformulate the model in Section 4.3 by representing the Laplace error in the form of a scaled mixture of Gaussians. Based on the new model formulation, we devise an EM algorithm in Section 4.4 for model learning. We further propose an online extension of the algorithm in Section 4.5 to deal with sequential data.

4.1 Probabilistic Interpretation

From a Bayesian perspective, the problem in Equation (2) corresponds to a *maximum a posteriori* (MAP) estimation problem. While the loss function is the negative log-likelihood, the regularization terms correspond to the negative log-priors. For example, Salakhutdinov and Mnih [6] gave an example based on the Gaussian distribution which corresponds to the ℓ_2 loss and ℓ_2 regularizer.

To achieve robustness, however, our focus is on matrix factorization under the ℓ_1 loss, i.e., $a = 1$ in Equation (2). In particular, we consider the following probabilistic model:

$$\mathbf{Y} = \mathbf{U}\mathbf{V}' + \mathbf{E}, \tag{3}$$

$$\mathbf{u}_{ij} \mid \lambda_u \sim N(\mathbf{u}_{ij}|\mathbf{0},\; \lambda_u^{-1}), \tag{4}$$

$$\mathbf{v}_{ij} \mid \lambda_v \sim N(\mathbf{v}_{ij}|\mathbf{0},\; \lambda_v^{-1}) \tag{5}$$

where $\mathbf{E} = [e_{ij}]$ is an $m \times n$ error matrix. Each element e_{ij} is sampled independently from the Laplace distribution $L(e_{ij}|0, \lambda)$, implying that

$$p(\mathbf{E}|\lambda) = \left(\frac{\lambda}{2}\right)^{mn} \exp\{-\lambda\|\mathbf{E}\|_1\}. \tag{6}$$

By treating $\mathbf{U}$ and $\mathbf{V}$ as model parameters and λ_u, λ_v and λ as hyperparameters with fixed values, we use MAP estimation to find $\mathbf{U}$ and $\mathbf{V}$. From Bayes' rule, we have

$$p(\mathbf{U}, \mathbf{V}|\mathbf{Y}, \lambda, \lambda_u, \lambda_v) \propto p(\mathbf{Y}|\mathbf{U}, \mathbf{V}, \lambda)\, p(\mathbf{U}|\lambda_u)\, p(\mathbf{V}|\lambda_v). \tag{7}$$

Thus,

$$\log p(\mathbf{U}, \mathbf{V}|\mathbf{Y}, \lambda, \lambda_u, \lambda_v) = -\lambda\|\mathbf{Y}-\mathbf{U}\mathbf{V}'\|_1 - \frac{\lambda_u}{2}\|\mathbf{U}\|_2^2 - \frac{\lambda_v}{2}\|\mathbf{V}\|_2^2 + C, \tag{8}$$

where C is a constant term independent of $\mathbf{U}$ and $\mathbf{V}$. Obviously, the problem of maximizing $\log p(\mathbf{U}, \mathbf{V}|\mathbf{Y}, \lambda, \lambda_u, \lambda_v)$ w.r.t. $\mathbf{U}$ and $\mathbf{V}$ is equivalent to the following minimization problem:

$$\min_{\mathbf{U},\mathbf{V}} \|\mathbf{Y}-\mathbf{U}\mathbf{V}'\|_1 + \frac{\lambda'_u}{2}\|\mathbf{U}\|_2^2 + \frac{\lambda'_v}{2}\|\mathbf{V}\|_2^2, \tag{9}$$

where $\lambda'_u = \lambda_u/\lambda$ and $\lambda'_v = \lambda_v/\lambda$. Note that we have omitted the binary matrix $\mathbf{W}$ in the problem formulation because our method can incorporate $\mathbf{W}$ into the model easily, with details to be presented in Section 4.4. Based on the analysis above, it can be seen readily that conventional robust matrix factorization based on the ℓ_1 loss can be derived from our probabilistic formulation.

4.2 Relationship with RPCA

Besides giving the robust matrix factorization problem formulation in Equation (9) a probabilistic interpretation, our model is also closely related to RPCA which is formulated as follows,

$$\min_{\mathbf{X}} \quad \|\mathbf{Y} - \mathbf{X}\|_1 + \lambda_r\|\mathbf{X}\|_*, \tag{10}$$

where λ_r is a fixed model parameter which gives a relative weighting of the ℓ_1 loss and the nuclear norm.

The connection between Equation (9) and Equation (10) follows from an important property of the nuclear norm, in that the nuclear norm can be cast in terms of the ℓ_2 norms of two factor matrices. A stronger condition in [15] is posed on the rank of the decomposition as follows.

Lemma 1. *For any matrix $\mathbf{Z} \in \mathbb{R}^{m\times n}$, the following holds:*

$$\|\mathbf{Z}\|_* = \min_{\mathbf{U},\mathbf{V},\mathbf{Z}=\mathbf{U}\mathbf{V}'} \frac{1}{2}(\|\mathbf{U}\|_2^2 + \|\mathbf{V}\|_2^2).$$

If $\operatorname{rank}(\mathbf{Z}) = k \le \min\{m, n\}$*, then the minimum above is attained at a factor decomposition* $\mathbf{Z} = \mathbf{U}_{m\times k}\mathbf{V}'_{n\times k}$.

Using Lemma 1, we can immediately get the following result:

Theorem 1. *Suppose $\mathbf{X}_r$ is a solution for Equation (10) with* $\operatorname{rank}(\mathbf{X}_r) = k$*, then for any solution $\mathbf{U}_r$, $\mathbf{V}_r$ to Equation (9) with $\lambda'_u = \lambda'_v = \lambda_r$ and $r = k$, $\mathbf{U}_r\mathbf{V}'_r$ is also a solution to Equation (10). This implies that the solution space of Equation (10) is contained in that of Equation (9).*

Proof. If we know that $\mathbf{X}_r$ is a solution for Equation (10), it is also a solution to

$$\min_{\mathbf{X},rank(\mathbf{X})=k} \|\mathbf{Y} - \mathbf{X}\|_1 + \lambda_r\|\mathbf{X}\|_*. \tag{11}$$

Since for any $\mathbf{X}$ with $rank(\mathbf{X}) = k$, we can find $\mathbf{U}_{m\times k}$ and $\mathbf{V}_{n\times k}$ satisfying $\mathbf{X} = \mathbf{U}\mathbf{V}'$. Then we have

$$\begin{aligned}&\min_{\mathbf{U},\mathbf{V}} \|\mathbf{Y}-\mathbf{U}\mathbf{V}'\|_1 + \frac{\lambda_r}{2}(\|\mathbf{U}\|_2^2 + \|\mathbf{V}\|_2^2)\\ &= \min_{\mathbf{U},\mathbf{V}} \|\mathbf{Y}-\mathbf{U}\mathbf{V}'\|_1 + \lambda_r\|\mathbf{U}\mathbf{V}'\|_*\\ &= \min_{\mathbf{X},rank(\mathbf{X})=k} \|\mathbf{Y}-\mathbf{X}\|_1 + \lambda_r\|\mathbf{X}\|_*.\end{aligned}$$

The equivalence of criteria in Equation (9) and Equation (10) completes our proof.

Then for the solutions to Equation (9), $\mathbf{U}_{m\times k}\mathbf{V}'_{n\times k}$ gives the solution to Equation (10). Moreover, it is easy to show that the SVD of $\mathbf{X}_r$ for the solution to Equation (10) gives one such solution to Equation (9). It is also worth noting that although our algorithm will produce different estimations of $\mathbf{U}$ nad $\mathbf{V}$ under different initializations, the estimation of $\mathbf{U}\mathbf{V}'$ is stable as guaranteed by Theorem 1 and the convexity of Equation (10).

4.3 Model Reformulation

While the model formulation given in Section 4.1 is rather straightforward, solving the optimization problem directly would be computationally challenging due to the non-smooth nature of the Laplace distribution. To address this computational issue, we reformulate the model by exploiting a two-level hierarchical representation of the Laplace distribution.

We first note that a random variable z follows a Laplace distribution $L(z|u,\alpha^2)$ if its probability density function (pdf) is given by

$$p(z|u,\alpha^2) = \frac{\alpha^2}{2}\exp(-\alpha^2|z-u|).$$

There exists an important property [16] that the Laplace distribution can be equivalently expressed as a scaled mixture of Gaussians, i.e.

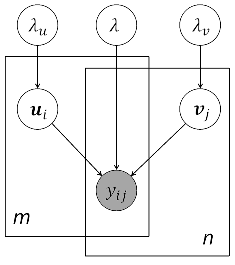
$$L(z|u,\alpha^2) = \int_0^\infty N(z|u,\tau)\,\mathrm{Expon}(\tau|\alpha^2)\,d\tau,$$

where $\mathrm{Expon}(\nu|\alpha^2)$ denotes an exponential distribution with pdf

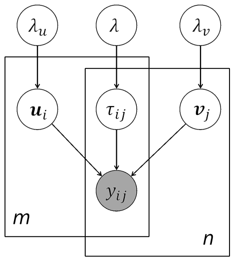
$$p(\nu|\alpha^2) = \frac{\alpha^2}{2}\exp\Big(-\frac{\alpha^2\nu}{2}\Big).$$

To incorporate this hierarchical view of the Laplace distribution, we introduce a matrix $\mathbf{T} = [\tau_{ij}] \in \mathbb{R}^{m\times n}$, where each element τ_{ij} is a latent variable with exponential prior for the corresponding y_{ij}. The latent variables introduced into the model relate the ℓ_1 loss to the (scaled) ℓ_2 loss and hence render a closed-form solution possible. Let $\mathbf{u}_i$ be the ith row of $\mathbf{U}$ and $\mathbf{v}_j$ be the jth row of $\mathbf{V}$. The PRMF model is reformulated as follows:

$$\begin{aligned}\mathbf{y}_{ij} \mid \mathbf{U},\mathbf{V},\mathbf{T} &\sim N(y_{ij}|\mathbf{u}_i'\mathbf{v}_j, \tau_{ij}),\\ \mathbf{u}_{ij} \mid \lambda_u &\sim N(\mathbf{u}_{ij}|\mathbf{0},\ \lambda_u^{-1}),\\ \mathbf{v}_{ij} \mid \lambda_v &\sim N(\mathbf{v}_{ij}|\mathbf{0},\ \lambda_v^{-1}),\\ \tau_{ij} \mid \lambda &\sim \mathrm{Expon}(\tau_{ij}|\lambda/2).\end{aligned} \tag{12}$$



(a) PMF [6]



(b) PRMF (proposed)

Fig. 1. PMF model vs. PRMF model. *(a) PMF model with Gaussian distribution on y_{ij}; (b) PRMF model with Laplace distribution on y_{ij} represented as a scaled mixture of Gaussians via the latent variable τ_{ij}.*

To facilitate the comparison of PRMF with PMF, Figures 1(a) and 1(b) show the graphical models of PMF and PRMF, respectively, using the plate convention.

4.4 EM Algorithm

We devise an EM algorithm for the hierarchical model presented above. In particular, we regard $\mathbf{T}$ as the missing data and $\theta = \{\mathbf{U}, \mathbf{V}\}$ as the parameters to be estimated while the hyperparameters $\lambda_u, \lambda_v, \lambda$ are fixed. To decouple the dependency between $\mathbf{U}$ and $\mathbf{V}$, we resort to the conditional EM (CEM) algorithm [17]. Specifically, each CEM iteration consists of two EM steps, namely, updating $\mathbf{V}$ while fixing $\mathbf{U}$ and updating $\mathbf{U}$ while fixing $\mathbf{V}$.

Let us consider updating $\mathbf{V}$ while assuming $\mathbf{U}$ to be known and fixed. The E-step computes the expectation of the complete-data log-posterior w.r.t. the missing data $\mathbf{T}$, given the current estimates of the parameters $\hat{\theta} = \{\hat{\mathbf{U}}, \hat{\mathbf{V}}\}$, i.e., it computes the so-called Q-function with the following definition:

$$Q(\mathbf{V}|\hat{\theta}) = E_{\mathbf{T}}[\log p(\mathbf{V}|\hat{\mathbf{U}}, \mathbf{Y}, \mathbf{T})|\mathbf{Y}, \hat{\theta}]. \tag{13}$$

In Equation (13) and the following, we make the dependency on the fixed hyperparameters λ_u, λ_v and λ implicit for notational simplicity.

To compute the Q-function, we first apply Bayes' rule. Then we take log on both sides and ignore the terms that do not depend on $\mathbf{V}$. The complete-data log-posterior $\log p(\mathbf{V}|\mathbf{Y}, \mathbf{T})$ can be expressed as

$$\begin{aligned}&\log p(\mathbf{Y}|\mathbf{V}, \hat{\mathbf{U}}, \mathbf{T}) + \log p(\mathbf{V})\\ &= -\frac{1}{2}\sum_i^m \sum_j^n \{\tau_{ij}^{-1}(y_{ij} - \hat{\mathbf{u}}_i'\mathbf{v}_j)^2\} - \frac{\lambda_v}{2}\sum_j^n \mathbf{v}_j'\mathbf{v}_j + C,\end{aligned}$$

where C does not depend on $\mathbf{U}$ and $\mathbf{V}$. It suffices to compute $E[\tau_{ij}^{-1}|\mathbf{Y}, \hat{\theta}]$ in the E-step.

With $p(\mathbf{T}|\mathbf{Y}, \hat{\mathbf{U}}, \hat{\mathbf{V}}) \propto p(\mathbf{Y}|\hat{\mathbf{U}}, \hat{\mathbf{V}}, \mathbf{T})p(\mathbf{T})$ and each element τ_{ij} of $\mathbf{T}$ follows an exponential prior, we find that τ_{ij}^{-1} follows an *inverse Gaussian distribution*. Thus, the posterior expectation is given by

$$E[\tau_{ij}^{-1}|\mathbf{Y}, \hat{\mathbf{U}}, \hat{\mathbf{V}}] = \frac{\sqrt{\lambda}}{|r_{ij}|} \triangleq \langle \tau_{ij}^{-1} \rangle, \tag{14}$$

with $r_{ij} = y_{ij} - (\mathbf{UV}')_{ij}$. A complete derivation is given in the supplemental material.

The M-step then updates the parameter estimates in each row of $\mathbf{V}$ by maximizing the Q-function in Equation (13) w.r.t. $\mathbf{v}_j$. By setting the partial derivative of the Q-function w.r.t. $\mathbf{v}_j$ to zero, we get a closed-form update formula as:

$$\mathbf{v}_j = (\hat{\mathbf{U}}'\Omega_j\hat{\mathbf{U}} + \lambda_v \mathbf{I}_r)^{-1}\hat{\mathbf{U}}'\Omega_j \mathbf{y}_{\cdot j}, \tag{15}$$

where $\Omega_j = \text{diag}(\langle \tau_{1j}^{-1} \rangle, \ldots, \langle \tau_{mj}^{-1} \rangle)$ and $\mathbf{y}_{\cdot j}$ is the jth column of $\mathbf{Y}$.

In the next EM step which updates $\mathbf{U}$ while fixing $\mathbf{V}$ at the value obtained above, a similar update formula can be obtained:

$$\mathbf{u}_i = (\hat{\mathbf{V}}'\Lambda_i\hat{\mathbf{V}} + \lambda_u \mathbf{I}_r)^{-1}\hat{\mathbf{V}}'\Lambda_i \mathbf{y}_{i\cdot}, \tag{16}$$

where $\Lambda_i \triangleq \text{diag}(\langle \tau_{i1}^{-1} \rangle, \ldots, \langle \tau_{in}^{-1} \rangle)$ and $\mathbf{y}_{i\cdot}$ is the ith row of $\mathbf{Y}$. We note that our algorithm can incorporate missing data easily. This can be done simply by setting the corresponding $\langle \tau_{ij} \rangle$ to 0, meaning that it is a "complete outlier" and hence should not be included in the calculation. The entire CEM algorithm is summarized in Algorithm 1.

We further notice that when we update $\mathbf{U}$ and $\mathbf{V}$ in the algorithm row by row, the computation is highly parallelizable. Specifically, all the inner loops in each M-step are independent of each other and hence can be dispatched to different servers in a cluster because there is no data conflict. The results obtained by different servers are then combined together. This could be a very favorable property to exploit when dealing with massive data sets.

Algorithm 1. CEM algorithm for PRMF

```
1:  Initialize $\mathbf{U}$ and $\mathbf{V}$ randomly.
2:  while not convergent do
3:      while $\mathbf{V}$ not convergent do
4:          E-Step: $\langle \tau_{ij}^{-1} \rangle = \frac{\sqrt{\lambda}}{|r_{ij}|}$
5:          M-Step:
6:          for each $i$th row $\mathbf{v}_i$ of $\mathbf{V}$:
7:              $\Omega_i \triangleq \text{diag}(\langle \tau_{1i}^{-1} \rangle, \ldots, \langle \tau_{mi}^{-1} \rangle)$
8:              $\mathbf{v}_i = (\mathbf{U}'\Omega_i\mathbf{U} + \lambda_v\mathbf{I}_r)^{-1}\mathbf{U}'\Omega_i\mathbf{y}_{\cdot i}$
9:      end while
10:     while $\mathbf{U}$ not convergent do
11:         E-Step: $\langle \tau_{ij}^{-1} \rangle = \frac{\sqrt{\lambda}}{|r_{ij}|}$
12:         M-Step:
13:         for each $j$th row $\mathbf{u}_j$ of $\mathbf{U}$:
14:             $\Lambda_j \triangleq \text{diag}(\langle \tau_{j1}^{-1} \rangle, \ldots, \langle \tau_{jn}^{-1} \rangle)$
15:             $\mathbf{u}_j = (\mathbf{V}'\Lambda_j\mathbf{V} + \lambda_u\mathbf{I}_r)^{-1}\mathbf{V}'\Lambda_j\mathbf{y}_{j\cdot}$
16:     end while
17: end while
```

4.5 Online Extension

In some common computer vision applications of RPCA such as background modeling and face shadow removal, usually the data arrive sequentially. However, all of the state-of-the-art RPCA algorithms fail to capture this sequential nature. As a consequence, when new data arrive, recomputation based on all the old and new data has to be performed. Such a naïve approach is clearly inefficient and memory demanding.

We note that the EM algorithm discussed above can easily be adapted to the online setting to address this issue. In what follows, we discuss the computational steps involved when the $(t+1)$th data point arrives.

Online E-step: This involves calculating the posterior expectation given the $(t+1)$th datum. Specifically, we compute Ω_{t+1}, with $\langle \tau_{t+1,j}^{-1} \rangle = \frac{\sqrt{\lambda}}{|\mathbf{y}_{t+1} - \mathbf{v}_{t+1}\mathbf{U}|_j}$ for $j = 1, 2, \ldots m$.

Online M-step: We specify the update rules for $\mathbf{U}$ and $\mathbf{V}$.

1. **Update V:** We note that Alg. 1 has already updated $\mathbf{V}$ row by row and hence we do not need to modify it specifically. In particular, we have $\mathbf{v}_{t+1} = (\mathbf{U}'\Omega_{t+1}\mathbf{U} + \lambda_v \mathbf{I}_r)^{-1}\mathbf{U}'\Omega_{t+1}\mathbf{y}_{\cdot t+1}$ with Ω_{t+1} obtained in the *online E-step*.
2. **Update U:** Let $\hat{\mathbf{V}}' = [\tilde{\mathbf{V}}', \mathbf{v}]$, where $\tilde{\mathbf{V}} \in \mathbb{R}^{t \times r}$ is from the t previous steps and $\mathbf{v}$ is the $(t+1)$th row just computed, and the ith row of $\mathbf{Y}$ is $\mathbf{y}_{i\cdot} = [\tilde{\mathbf{y}}_i, y_i]$. Furthermore, let $\Lambda_i = \begin{pmatrix} \tilde{\Lambda}_i & 0 \\ 0 & \sigma_i \end{pmatrix}$ be a block-diagonal matrix with $\sigma_i = \langle \tau_{i,t+1}^{-1} \rangle$ and $\tilde{\Lambda}_i \in \mathbb{R}^{t \times t}$, and Equation (16) can be written explicitly with the old and new statistics separated, i.e.

$$\mathbf{u}_i = (\tilde{\mathbf{V}}'\tilde{\Lambda}_i\tilde{\mathbf{V}} + \sigma_i \mathbf{v}\mathbf{v}' + \lambda_v \mathbf{I}_r)^{-1}(\tilde{\mathbf{V}}'\tilde{\Lambda}_i\tilde{\mathbf{y}}_i + \sigma_i y_i \mathbf{v}). \tag{17}$$

   To further enhance the efficiency of the online algorithm, we want to get rid of the inverse operation by the Sherman-Morrison identity for rank-one matrix inversion:

$$(\mathbf{A} + vv')^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}vv'\mathbf{A}^{-1}}{1 + v'\mathbf{A}^{-1}v}. \tag{18}$$

   Thus we can have an online updating mechanism for $\mathbf{u}_i$ with the previous additive statistics defined as

$$\tilde{\mathbf{A}}_i \triangleq (\tilde{\mathbf{V}}'\tilde{\Lambda}_i\tilde{\mathbf{V}} + \lambda_v \mathbf{I}_r)^{-1}, \qquad \tilde{\mathbf{B}}_i \triangleq \tilde{\mathbf{V}}'\tilde{\Lambda}_i\tilde{\mathbf{y}}_i. \tag{19}$$

   The update rules for the $(t+1)$th data point can be obtained readily:

$$\mathbf{A}_i = \tilde{\mathbf{A}}_i - \frac{\sigma_i \tilde{\mathbf{A}}_i \mathbf{v}\mathbf{v}'\tilde{\mathbf{A}}_i}{1 + \sigma_i \mathbf{v}'\tilde{\mathbf{A}}_i\mathbf{v}}, \qquad \mathbf{B}_i = \tilde{\mathbf{B}}_i + \sigma_i y_i \mathbf{v}, \qquad \mathbf{u}_i = \mathbf{A}_i\mathbf{B}_i. \tag{20}$$

In our experiments, we first warm-start with a small batch of data points for $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$, and then update them incrementally as each new data point arrives. Besides, in practice, we may also need to scale down the weight of the "past data" $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ by a factor ρ to make the estimation fit the latest data better. Although we have not conducted theoretical analysis to derive the convergence bounds, our empirical findings show that the online algorithm does converge and has competitive performance in terms of accuracy. The online EM algorithm is summarized in Algorithm 2.

Algorithm 2. Online EM algorithm for PRMF

```
1: Initialize $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ using Algorithm 1 for a small batch of data points.
2: for the $(t+1)$th data point, $t = 0, 1, \dots D$ do
3:     while not convergent do
4:         $\langle \tau_{j,t+1}^{-1} \rangle = \frac{\sqrt{\lambda}}{|\mathbf{y}_{t+1} - \mathbf{v}_{t+1}\mathbf{U}|_j}$ for $j = 1, 2, \dots m$.
5:         $\Omega_{t+1} = \mathrm{diag}(\langle \tau_{1,t+1}^{-1} \rangle, \dots, \langle \tau_{m,t+1}^{-1} \rangle)$.
6:         $\mathbf{v} = (\mathbf{U}'\Omega_{t+1}\mathbf{U} + \lambda_v \mathbf{I}_r)^{-1}\mathbf{U}'\Omega_{t+1}\mathbf{y}_{\cdot t+1}$
7:         for each $\mathbf{u}_i$, $i = 1, 2, \dots, m$ do
8:             $\sigma_i = \langle \tau_{i,t+1}^{-1} \rangle$
9:             $\tilde{\mathbf{A}}_i = \tilde{\mathbf{A}}_i / \rho$, $\tilde{\mathbf{B}}_i = \tilde{\mathbf{B}}_i \times \rho$
10:            $\mathbf{A}_i = \tilde{\mathbf{A}}_i - \frac{\sigma_i \tilde{\mathbf{A}}_i \mathbf{v}\mathbf{v}' \tilde{\mathbf{A}}_i}{1 + \sigma_i \mathbf{v}' \tilde{\mathbf{A}}_i \mathbf{v}'}$
11:            $\mathbf{B}_i = \tilde{\mathbf{B}}_i + \sigma_i y_i \mathbf{v}$
12:            $\mathbf{u}_i = \mathbf{A}_i \mathbf{B}_i$
13:        end for
14:    end while
15:    $\mathbf{V}' = [\mathbf{V}', \mathbf{v}]$
16:    $\tilde{\mathbf{A}}_i = \mathbf{A}_i$ and $\tilde{\mathbf{B}}_i = \mathbf{B}_i$.
17: end for
```

5 Experiments

Now that we have presented our model in detail, we turn to its experimental validation by conducting experiments using both synthetic and real data sets. We compare PRMF with several state-of-the-art matrix factorization methods, which include PMF [6], RPCA [7], GoDec [12], and Bayesian Robust PCA (BRPCA) [14]. The MATLAB implementations of all these methods can be found in their authors' websites. The desktop computer used to run our experiments has a 64-bit Intel Core i7-2600 processor and 8GB RAM.

5.1 Analysis on Synthetic Data

In this experiment, we assess the performance quantitatively on synthetic data sets of various sizes. The parameters of all the compared algorithms have been carefully tuned. To generate the data, each synthetic data point consists of two parts. The low-rank part is the product of an $m \times r$ matrix and an $r \times n$ matrix, with each element of the matrices generated i.i.d. from a Gaussian distribution. For the outlier part, we randomly choose 10% of the elements and add noise randomly drawn from a uniform distribution defined on the range $[-50, 50]$. We do not explicitly introduce missing entries since our method can treat them as a special type of outliers. To mimic more realistic settings with additional random noise, we further add to each entry of the matrix a small noise term drawn from $\mathcal{N}(0, 0.001)$. The performance comparison is tabulated in Table 1 using performance measures for both efficiency and accuracy. In particular, the relative error is computed with respect to the ground truth, which is defined as $\frac{\|\mathbf{M}-\mathbf{N}\|_2}{\|\mathbf{M}\|_2}$, where $\mathbf{N}$ is the recovered low-rank matrix and $\mathbf{M}$ is the ground-truth matrix.

From Table 1, we can see that except for PMF which is not a robust method, all other methods have satisfactory performance in terms of recovery accuracy. However, BRPCA has very high computational requirement and hence is significantly slower than the other methods. For the 1000×1000 data matrix, it has taken about 1.5 hours to complete. We do not have results for the larger matrices because the program does not yet terminate even after running for five hours. We note that PRMF, RPCA and GoDec are very comparable in terms of both efficiency and accuracy. While RPCA is efficient for smaller matrices, PRMF is always the most efficient for matrices of size 1000×1000 or larger, showing a larger gap as the size and rank increase. Moreover, while tuning the parameters for all algorithms to achieve the best results, we notice that the recovery accuracy of PRMF is least sensitive to the parameters. This is a favorable property from a practitioner's point of view as less time will be needed on parameter tuning.

Table 1. Performance comparison of PRMF, RPCA, GoDec, BRPCA and the non-robust PMF. Time is measured in seconds and the error measure has an implicit factor of 10^{-4}.

Data matrix		PRMF		RPCA		GoDec		BRPCA		PMF	
Size	Rank	Time	Error	Time	Error	Time	Error	Time	Error	Time	Error
100×100	3	0.08	6.70	0.06	1.59	0.09	1.57	15.81	1.47	1.23	2.02E4
200×200	5	0.21	4.12	0.27	1.06	0.25	1.08	50.04	1.09	4.83	1.12E4
500×500	10	1.25	2.47	0.72	0.49	1.82	0.66	737.75	0.65	36.38	6.64E3
1000×1000	15	3.47	0.52	6.31	0.49	8.98	0.47	5310.4	18.50	142.21	4.29E3
2000×2000	20	15.40	0.34	42.32	0.34	70.27	0.32	NA	NA	760.55	2.96E3
5000×5000	25	176.17	0.21	793.96	0.23	234.08	0.21	NA	NA	6268.60	2.04E3

We further examine the time requirements of PRMF more closely by varying the matrix rank and size. For the experiment of varying size, we fix one dimension to 500 and then vary the other. The computation time is measured using the same convergence criterion to facilitate comparison. Fig. 3 and Fig. 4 show the results. We can see that the computation time needed generally increases linearly as the rank or size increases. This further demonstrates that PRMF may be a good candidate for large-scale applications.

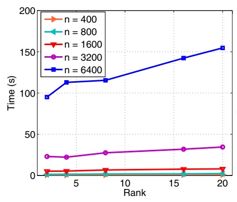


Fig. 3. Time vs. Rank

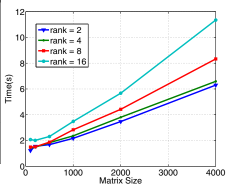


Fig. 4. Time vs. Size

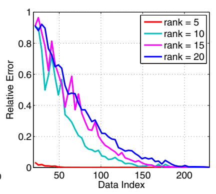


Fig. 5. Error vs. Data Index(Vary rank)

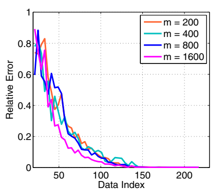
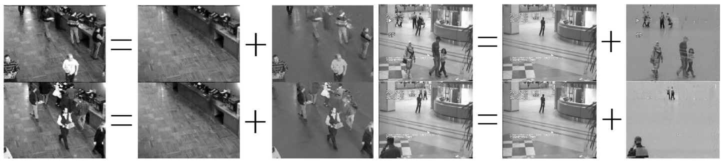


Fig. 6. Error vs. Data Index(Vary size)

To study the convergence behavior of the online extension of the learning algorithm, we monitor the trajectory of relative errors as the online algorithm is applied to the synthetic data observed sequentially. As above, both the matrix rank and size are varied. We warm-start the algorithm with a small batch of 20 data points. The results are shown in Fig. 5 and Fig. 6. We note that convergence to a very low error level is often achieved after observing a reasonably small number of data points. For the very low rank case with rank equal to 5, warm-start initialization alone is sufficient to reach a very low error level. When the rank is higher, convergence generally takes longer but the number of additional data points that need to be observed is still quite small.

5.2 Background Modeling

We now consider a real computer vision application which performs background modeling in surveillance video sequences. The goal of this application is to separate the static background from the dynamic foreground which may include moving objects and variation in illumination. Since the background is relatively stationary, it is reasonable to assume that its representation involves only a small number of latent factors. Therefore the background is modeled by a low-rank matrix. On the other hand, the foreground contains outliers which are relatively sparse. We sample 200 frames from each of two popular video sequences [1] as test data. The result is depicted in Fig. 7. We can see that PRMF preforms quite well on this task by separating the background and foreground successfully. In addition, we also compare in Table 5.2 the time needed for the four robust algorithms.



(a) Result of *Bootstrap* sequence (b) Result of *Hall* sequence

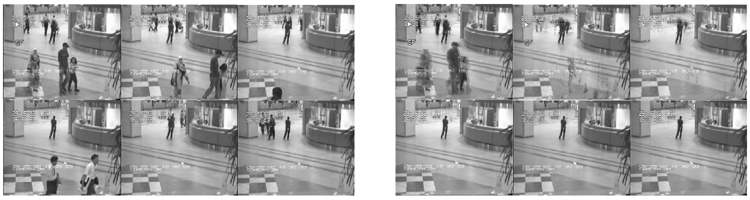
Fig. 7. Background modeling results. The leftmost image shows the original frame, the middle one shows the background (low-rank), and the rightmost one shows the foreground (outlier).

Table 2. Time comparison of PRMF, RPCA, GoDec and BRPCA in background modeling task

	PRMF	RPCA	GoDec	BRPCA
Bootstrap	2.87	5.21	38.41	1602.60
Hall	3.61	7.91	39.37	2086.81

We note the sequential nature of surveillance video data and hence apply the online PRMF algorithm to investigate its performance qualitatively. Figure 8 shows the recovery result of the *Hall* sequence. In line with our observations on the synthetic data, the recovery result improves as more frames are observed and incorporated into the online algorithm. Visually, the residue on the reconstructed background reduces greatly. In particular, the background modeling result is already quite satisfactory after 100 frames. We also compare our algorithm with a state-of-the-art online robust matrix factorization algorithm GRASTA [13] . We set up this experiment by following the same settings in that paper. Specifically, in the *Hall* sequence, the first 100 frames are cropped using

[1] `http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html`



(a) Original video frames 5, 20, 50, 100, 150 and 200 of the *Hall*

(b) Extracted backgrounds of the corresponding frames

Fig. 8. Background modeling results of the online algorithm

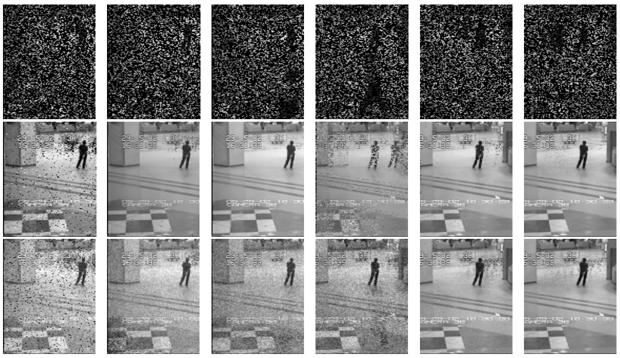


Fig. 9. Results with presenting missing entries and suddenly changed background. The first line shows frames 40, 70, 100, 130, 170, 200 used for training. The second line shows the results obtained by PRMF. The last line shows the results obtained by GRASTA [13]

a window of half of the video width. Starting from frame 101, the window slides to the right by 20 pixels. In addition, to demonstrate its robustness to missing data, we randomly drop 70% of the pixels. The results are illustrated in Fig. 9. Both algorithms can adapt to the background changes successfully, but PRMF can give a much cleaner background than that of GRASTA.

6 Conclusion and Future Work

We have developed a novel probabilistic model in this paper for robust matrix factorization based on the ℓ_1 loss. The model is robust against outliers and missing data. We have devised an efficient conditional EM algorithm for model learning. In addition, we have also devised an online extension of the batch algorithm to handle sequential data encountered in some applications. For experimental validation, we have compared our model with some state-of-the-art robust matrix factorization algorithms on both synthetic data and practical computer vision application. The experimental results are very encouraging. In particular, our model outperforms the other methods in terms of efficiency particularly for large data matrices.

Our current model is based on the empirical Bayes approach. It would be interesting to explore a fully Bayesian model for robust matrix factorization. In such a model, inference may adopt a variational approximation approach or a sampling approach such as a Gibbs sampler. Another possible future research direction is to enhance the model for specific applications. In the background modeling application, for example, we may incorporate the Markov property between video frameworks into the model to lead to further improvement. Such extensions will be pursued in our future work.

Acknowledgment. This research has been supported by General Research Fund 621310 from the Research Grants Council of Hong Kong.

References

1. Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: a factorization method. International Journal of Computer Vision 9(2), 137–154 (1992)
2. Bregler, C., Hertzmann, A., Biermann, H.: Recovering non-rigid 3d shape from image streams. In: CVPR, vol. 2, pp. 690–696 (2000)
3. Aanæs, H., Fisker, R., Astrom, K., Carstensen, J.M.: Robust factorization. IEEE Trans. Pattern Anal. Mach. Intelligence 24(9), 1215–1225 (2002)
4. Ke, Q., Kanade, T.: Robust l_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In: CVPR, pp. 739–746 (2005)
5. Eriksson, A., van den Hengel, A.: Efficient computation of robust low-rank matrix approximations in the presence of missing data using the l1 norm. In: CVPR, pp. 771–778 (2010)
6. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: NIPS, vol. 20 (2008)
7. Lin, Z., Chen, M., Wu, L., Ma, Y.: The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. Arxiv preprint arXiv:1009.5055 (2010)
8. Candès, E.J., Li, X., Ma, Y., Wright, J.: Robust principal component analysis? J. ACM 58(3), 11 (2011)
9. Salakhutdinov, R., Mnih, A.: Bayesian probabilistic matrix factorization using markov chain monte carlo. In: ICML, pp. 880–887 (2008)
10. Lakshminarayanan, B., Bouchard, G., Archambeau, C.: Robust bayesian matrix factorisation. In: AISTATS (2011)
11. Mu, Y., Dong, J., Yuan, X., Yan, S.: Accelerated low-rank visual recovery by random projection. In: CVPR, pp. 2609–2616 (2011)
12. Zhou, T., Tao, D.: Godec: Randomized low-rank & sparse matrix decomposition in noisy case. In: ICML (2011)
13. He, J., Balzano, L., Szlam, A.: Incremental gradient on the grassmannian for online foreground and background separation in subsampled video. In: CVPR (2012)
14. Carin, L., Ding, X., He, L.: Bayesian robust principal component analysis. IEEE Transactions on Image Processing 20(12), 3419–3430 (2011)
15. Mazumder, R., Hastie, T., Tibshirani, R.: Spectral regularization algorithms for learning large incomplete matrices. The Journal of Machine Learning Research 11, 2287–2322 (2010)
16. Lange, K., Sinsheimer, J.S.: Normal/independent distributions and their applications in robust regression. Journal of Computational and Graphical Statistics, 175–198 (1993)
17. Jebara, T., Pentland, A.: Maximum conditional likelihood via bound maximization and the CEM algorithm. In: NIPS, vol. 11 (1999)