Kernelized Probabilistic Matrix Factorization: Exploiting Graphs and Side Information

Tinghui Zhou* Hanhuai Shan* Arindam Banerjee* Guillermo Sapiro†

Abstract

We propose a new matrix completion algorithm—Kernelized Probabilistic Matrix Factorization (KPMF), which effectively incorporates external side information into the matrix factorization process. Unlike Probabilistic Matrix Factorization (PMF) [14], which assumes an independent latent vector for each row (and each column) with Gaussian priors, KMPF works with latent vectors spanning all rows (and columns) with Gaussian Process (GP) priors. Hence, KPMF explicitly captures the underlying (nonlinear) covariance structures across rows and columns. This crucial difference greatly boosts the performance of KPMF when appropriate side information, e.g., users' social network in recommender systems, is incorporated. Furthermore, GP priors allow the KPMF model to fill in a row that is entirely missing in the original matrix based on the side information alone, which is not feasible for standard PMF formulation. In our paper, we mainly work on the matrix completion problem with a graph among the rows and/or columns as side information, but the proposed framework can be easily used with other types of side information as well. Finally, we demonstrate the efficacy of KPMF through two different applications: 1) recommender systems and 2) image restoration.

1 Introduction

The problem of missing value prediction, and particularly matrix completion, has been addressed in many research areas, including recommender systems [11, 16], geostatistics [20], and image restoration [3]. In such problems, we are typically given an $N \times M$ data matrix R with a number of missing entries, and the goal is to fill in the missing entries properly such that they are coherent with the existing data, where the existing data may include the observed entries in the data matrix as well as the side information depending on the specific problem domain.

Among the existing matrix completion techniques, factorization based algorithms have achieved great success and popularity [1, 9, 14, 15, 18, 21]. In these algorithms, each row, as well as each column, of the matrix has a latent vector, obtained from factorizing the partially observed matrix. The prediction of each missing entry is thus the inner product of latent vectors of the corresponding row and the corresponding column. However, such techniques often suffer from the data sparsity problem in real-world scenarios. For instance, according to [16], the density of non-missing ratings in most commercial recommender systems is less than 1%. It is thus very difficult to do missing value prediction based on such small amount of data. On the other hand, in addition to the data matrix, other sources of information, e.g., users' social network in recommender systems, are sometimes readily available, and could provide key information about the underlying model, while many of the existing factorization techniques simply ignore such side information, or intrinsically, are not capable of exploiting it (see Section 2 for more on related work)

To overcome such limitations, we propose the Kernelized Probabilistic Matrix Factorization (KPMF) model, which incorporates the side information through kernel matrices over rows and over columns. KPMF models a matrix as the product of two latent matrices, which are sampled from two different zero-mean Gaussian processes (GP). The covariance functions of the GPs are derived from the side information, and encode the covariance structure across rows and across columns respectively. In this paper, we focus on deriving covariance functions from undirected graphs (e.g., users' social network). However, our general framework can incorporate other types of side information as well. For instance, when the side information is in the form of feature vectors [1], we may use the RBF kernel [17] as the covariance function.

Although KPMF seems highly related to the Probabilistic Matrix Factorization (PMF) [14] and its generalized counterpart–Bayesian PMF (BPMF) [15], the key difference that makes KPMF a more powerful model is that while PMF/BPMF assumes an independent latent vector for *each* row, KPMF works with latent vectors spanning *all* rows. Therefore, unlike PMF/BPMF,

*Department of Computer Science and Engineering, University of Minnesota, Twin Cities

†Department of Electrical and Computer Engineering, University of Minnesota, Twin Cities

KPMF is able to explicitly capture the covariances across the rows. Moreover, if an entire row of the data matrix is missing, PMF/BPMF fails to make prediction for that row. In contrast, being a nonparametric model based on a covariance function, KPMF can still make predictions based on the row covariances alone. Similarly, the above discussion holds for columns as well.

We demonstrate KPMF through two applications: 1) recommender systems and 2) image restoration. For recommender systems, the side information is users' social network, and for image restoration, the side information is derived from the *spatial smoothness* assumption–pixel variation in a small neighborhood tends to be small and correlated. Our experiments show that KPMF consistently outperforms state-of-the-art collaborative filtering algorithms, and produce promising results for image restoration.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 gives a brief overview of the background, and in particular, the PMF and BPMF models. Section 4 presents the KPMF model and two methods, gradient descent and stochastic gradient descent, for learning it. We present the experimental results for recommender systems in Section 5 and for image restoration in Section 6, and conclude in Section 7.

2 Related Work

Factorization-based algorithms are powerful techniques for matrix completion. In particular, a probabilistic framework for matrix factorization, namely Probabilistic Matrix Factorization (PMF), was recently proposed in [14], and generalized to a full Bayesian model in [15] (see Section 3.2 for more discussion on PMF and Bayesian PMF). Additionally, Lawrence and Urtasun [9] developed a non-linear extension to PMF using Gaussian process latent variable models. However, one major limitation of the above methods is the lack of ability to incorporate side information into the factorization process.

Now we review some existing matrix completion algorithms that incorporate side information in the framework. The approach proposed in [18] generalizes PMF to a parametric framework and uses topic models for incorporating side information. Similarly, Wang and Blei [21] combined traditional collaborative filtering and probabilistic topic modeling for recommending scientific articles, and Agarwal and Chen [1] developed a matrix factorization method for recommender systems using LDA priors to regularize the model based on item meta-data and user features. Moreover, Ma *et al.* [11] proposed to perform probabilistic matrix factorization on users' social network and the rating matrix jointly, so that the resulting latent matrices depend on both input sources, and Agovic *et al.* [2] proposed the Probabilistic Matrix Addition (PMA) model, where the generative process of the data matrix is modeled as the additive combination of latent matrices drawn from two Gaussian processes: one for the rows, and the other for the columns. The main difference between PMA and KPMF, as we shall see, is that in PMA, the Gaussian processes capturing the covariance structure for rows and columns are combined additively in the generative model, while in KPMF, the Gaussian processes are priors for the row and column latent matrices, and the data matrix is generated from the product of the two latent matrices.

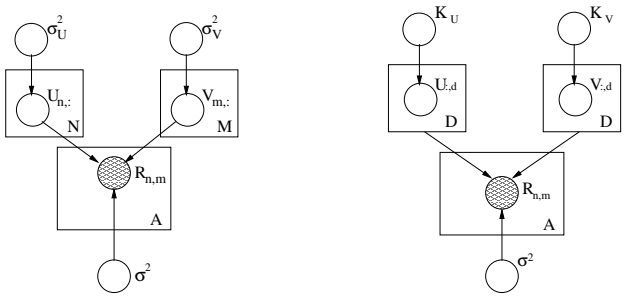
3 Preliminaries

3.1 Notations While the proposed KPMF model is applicable to other matrix completion problems, we focus on recommender systems and develop the notation and exposition accordingly. We define the main notations used in this paper as follows:

R – $N \times M$ data matrix.
$R_{n,:}$ – n^{th} row of R.
$R_{:,m}$ – m^{th} column of R.
N – Number of rows in R.
M – Number of columns in R.
D – Dimension of the latent factors.
U – $N \times D$ latent matrix for rows of R.
V – $M \times D$ latent matrix for columns of R.
$U_{n,:} \in \mathbb{R}^D$ – Latent factors for $R_{n,:}$.
$V_{m,:} \in \mathbb{R}^D$ – Latent factors for $R_{:,m}$.
$U_{:,d} \in \mathbb{R}^N$ – d^{th} latent factor for all rows of R.
$V_{:,d} \in \mathbb{R}^M$ – d^{th} latent factor for all columns of R.
$K_U \in \mathbb{R}^{N \times N}$ – Covariance matrix for rows.
$K_V \in \mathbb{R}^{M \times M}$ – Covariance matrix for columns.
$S_U \in \mathbb{R}^{N \times N}$ – Inverse of K_U.
$S_V \in \mathbb{R}^{M \times M}$ – Inverse of K_V.
$[n]_1^N$ – $n = \{1, 2, \ldots, N\}$.

3.2 PMF and BPMF Consider an $N \times M$ real-valued matrix R with a number of missing entries. The goal of matrix completion is to predict the values of those missing entries. Probabilistic Matrix Factorization (PMF) [14] approaches this problem from the matrix factorization aspect. Assuming two latent matrices: $U_{N \times D}$ and $V_{M \times D}$, with U and V capturing the row and the column features of R respectively, the generative process for PMF is given as follows (also see Figure 1 (a)):

1. For each row n in R, $[n]_1^N$, generate $U_{n,:} \sim \mathcal{N}(\mathbf{0}, \sigma_U^2 I)$, where I denotes the identity matrix.



(a) (b)

Figure 1: (a) The generative process of R in PMF; (b) The generative process of R in KPMF. A is the total number of non-missing entries in the matrix.

2. For each column m in R, $[m]_1^M$, generate $V_{m,:} \sim \mathcal{N}(\mathbf{0}, \sigma_V^2 I)$.
3. For each of the non-missing entries (n, m), generate $R_{n,m} \sim \mathcal{N}(U_{n,:} V_{m,:}^T, \sigma^2)$.

The model has zero-mean spherical Gaussian priors on $U_{n,:}$ and $V_{m,:}$, and each entry $R_{n,m}$ is generated from a univariate Gaussian with the mean determined by the inner product of $U_{n,:}$ and $V_{m,:}$. The log-posterior over the latent matrices U and V is given by:

$$\begin{aligned}(3.1)\quad &\log p(U, V | R, \sigma^2, \sigma_U^2, \sigma_V^2) \\ &= -\frac{1}{2\sigma^2} \sum_{n=1}^{N} \sum_{m=1}^{M} \delta_{n,m} (R_{n,m} - U_{n,:} V_{m,:}^T)^2 \\ &\quad - \frac{1}{2\sigma_U^2} \sum_{n=1}^{N} U_{n,:}^T U_{n,:} - \frac{1}{2\sigma_V^2} \sum_{m=1}^{M} V_{m,:}^T V_{m,:} \\ &\quad - \frac{1}{2} (A \log \sigma^2 + ND \log \sigma_U^2 + MD \log \sigma_V^2) + C,\end{aligned}$$

where $\delta_{n,m}$ is the indicator taking value 1 if $R_{n,m}$ is an observed entry, and 0 otherwise, A is the number of non-missing entries in R, and C is a constant that does not depend on the latent matrices U and V. MAP inference maximizes the log-likelihood with respect to U and V, which could then be used to predict the missing entries in R.

As an extension of PMF, Bayesian PMF (BPMF) [15] introduces a full Bayesian prior for each $U_{n,:}$ and each $V_{m,:}$. $U_{n,:}$ (and similarly for $V_{m,:}$) is then sampled from $\mathcal{N}(\mu_U, \Sigma_U)$, where the hyperparameters $\{\mu_U, \Sigma_U\}$ are further sampled from Gaussian-Wishart priors.

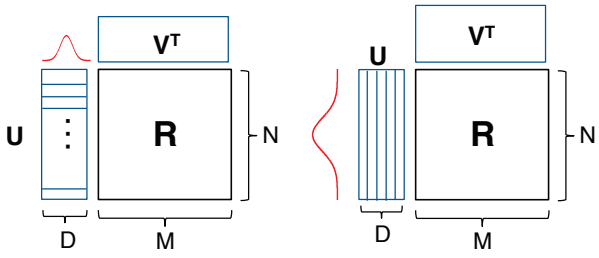
4 KPMF

In this section, we propose a Kernelized Probabilistic Matrix Factorization (KPMF) model, and present both gradient descent and stochastic gradient descent methods for learning the model.

4.1 The Model In KPMF, the prior distribution of each column of the latent matrices, $U_{:,d}$ and $V_{:,d}$, is a zero-mean Gaussian process [13]. Gaussian processes are a generalization of the multivariate Gaussian distribution. While a multivariate Gaussian is determined by a mean vector and a covariance matrix, the Gaussian process $GP(m(x), k(x, x'))$ is determined by a mean function $m(x)$ and a covariance function $k(x, x')$. In our problem, x is an index of matrix rows (or columns). Without loss of generality, let $m(x) = \mathbf{0}$, and $k(x, x')$ denote the corresponding kernel function, which specifies the covariance between any pair of rows (or columns). Also, let $K_U \in \mathbb{R}^{N \times N}$ and $K_V \in \mathbb{R}^{M \times M}$ denote the full covariance matrix for rows of R and columns of R respectively. As we shall see later, using K_U and K_V in the priors forces the latent factorization to capture the underlying covariances among rows and among columns simultaneously.

Assuming K_U and K_V are known,[1] the generative process for KPMF is given as follows (also see Figure 1(b)):

1. Generate $U_{:,d} \sim GP(\mathbf{0}, K_U)$, $[d]_1^D$.
2. Generate $V_{:,d} \sim GP(\mathbf{0}, K_V)$, $[d]_1^D$.
3. For each non-missing entry $R_{n,m}$, generate $R_{n,m} \sim \mathcal{N}(U_{n,:} V_{m,:}^T, \sigma^2)$, where σ is a constant.



(a) PMF and BPMF (b) KPMF

Figure 2: (a) U is sampled in a "row-wise" manner in PMF and BPMF; (b) U is sample in a "column-wise" manner in KPMF.

The likelihood over the observed entries in the target matrix R given latent matrices U and V is

$$(4.2)\quad p(R | U, V, \sigma^2) = \prod_{n=1}^{N} \prod_{m=1}^{M} [\mathcal{N}(R_{n,m} | U_{n,:} V_{m,:}^T, \sigma^2)]^{\delta_{n,m}},$$

[1] The choice of K_U and K_V depends on the specific problem domain, and will be addressed later with particular examples.

with the priors over U and V given by

$$p(U|K_U) = \prod_{d=1}^{D} GP(U_{:,d}|0, K_U), \tag{4.3}$$

$$p(V|K_V) = \prod_{d=1}^{D} GP(V_{:,d}|0, K_V). \tag{4.4}$$

For simplicity, we denote K_U^{-1} by S_U, and K_V^{-1} by S_V. The log-posterior over U and V is hence given by

$$\begin{aligned} &\log p(U, V|R, \sigma^2, K_U, K_V) \\ =& -\frac{1}{2\sigma^2}\sum_{n=1}^{N}\sum_{m=1}^{M}\delta_{n,m}(R_{n,m} - U_{n,:}V_{m,:}^T)^2 \\ &-\frac{1}{2}\sum_{d=1}^{D}U_{:,d}^T S_U U_{:,d} - \frac{1}{2}\sum_{d=1}^{D}V_{:,d}^T S_V V_{:,d} \\ &- A\log\sigma^2 - \frac{D}{2}(\log|K_U| + \log|K_V|) + C, \end{aligned} \tag{4.5}$$

where A is the total number of non-missing entries in R, $|K|$ is the determinant of K, and C is a constant term not depending on the latent matrices U and V.

4.2 KPMF Versus PMF/BPMF We illustrate the difference between KPMF and PMF/BPMF in Figure 2. In PMF/BPMF, U is sampled in a "row-wise" manner (Figure 2(a)), i.e., $U_{n,:}$ is sampled for each row in R. $\{U_{n,:}, [n]_1^N\}$ are hence conditionally independent given the prior. As a result, correlations among rows are not captured in the model. In contrast, in KPMF, U is sampled in a "column-wise" manner (Figure 2(b)), i.e., for each of the D latent factors, $U_{:,d} \in \mathbb{R}^N$ is sampled for all rows of R. In particular, $U_{:,d}$ is sampled from a Gaussian process whose covariance K_U captures the row correlations. In this way, during training, the latent factors of each row ($U_{n,:}$) are correlated with those of all the other rows ($U_{n',:}$ for $n' \neq n$) through K_U. Roughly speaking, if two rows share some similarity according to the side information, the corresponding latent factors would also be similar after training, which is intuitively what we want to achieve given the side information. Similarly, the discussion above also applies to V and the columns of R. The difference between KPMF and BPMF is subtle but they are entirely different models, and cannot be viewed as special cases of each other. PMF is a simple special case of both models, with neither correlations across the rows, nor correlations across latent factors captured.

The row and column independencies in PMF and BPMF significantly undermine the power of the model, since strong correlations among rows and/or among columns are often present in real scenarios. For instance, in recommender systems, users' decisions on item ratings (represented by rows) are very likely to be influenced by other users who have social connections (friends, families, etc.) with them. PMF and BPMF fail to capture such correlational dependencies. As a result, the proposed KPMF performs considerably better than PMF and BPMF (see Section 5).

4.3 Gradient Descent for KPMF We perform a MAP estimate to learn the latent matrices U and V, which maximize the log-posterior in (4.5), and is equivalent to minimizing the following objective function:

$$\begin{aligned} E =& \frac{1}{2\sigma^2}\sum_{n=1}^{N}\sum_{m=1}^{M}\delta_{n,m}(R_{n,m} - U_{n,:}V_{m,:}^T)^2 \\ &+ \frac{1}{2}\sum_{d=1}^{D}U_{:,d}^T S_U U_{:,d} + \frac{1}{2}\sum_{d=1}^{D}V_{:,d}^T S_V V_{:,d}\ . \end{aligned} \tag{4.6}$$

Minimization of E can be done through gradient descent. In particular, the gradients are given by

$$\begin{aligned} \frac{\partial E}{\partial U_{n,d}} =& -\frac{1}{\sigma^2}\sum_{m=1}^{M}\delta_{n,m}(R_{n,m} - U_{n,:}V_{m,:}^T)V_{d,m} \\ &+ \boldsymbol{e}_{(n)}^T S_U U_{:,d}, \end{aligned} \tag{4.7}$$

$$\begin{aligned} \frac{\partial E}{\partial V_{m,d}} =& -\frac{1}{\sigma^2}\sum_{n=1}^{N}\delta_{n,m}(R_{n,m} - U_{n,:}V_{m,:}^T)U_{d,n} \\ &+ \boldsymbol{e}_{(m)}^T S_V V_{:,d}\ , \end{aligned} \tag{4.8}$$

where $\boldsymbol{e}_{(n)}$ denotes an N-dimensional unit vector with the n^{th} component being one and others being zero. The update equations for U and V are

$$U_{n,d}^{(t+1)} = U_{n,d}^{(t)} - \eta\frac{\partial E}{\partial U_{n,d}}, \tag{4.9}$$

$$V_{m,d}^{(t+1)} = V_{m,d}^{(t)} - \eta\frac{\partial E}{\partial V_{m,d}}\ , \tag{4.10}$$

where η is the learning rate. The algorithm updates U and V following (4.9) and (4.10) alternatively until convergence. It should be noted that since K_U and K_V remain fixed throughout all iterations, S_U and S_V need to be computed only once at initialization.

Now Suppose an entire row or column of R is missing. While PMF and BPMF fail to address such problem, KPMF still works if appropriate side information is given. In this case, the update equations in (4.9) and

(4.10) become

$$U_{n,d}^{(t+1)} = U_{n,d}^{(t)} - \eta \boldsymbol{e}_{(n)}^T S_U U_{:,d}$$

$$= U_{n,d}^{(t)} - \eta \sum_{n'=1}^{N} S_U(n,n') U_{n',d} \tag{4.11}$$

$$V_{m,d}^{(t+1)} = V_{m,d}^{(t)} - \eta \boldsymbol{e}_{(m)}^T S_V V_{:,d}$$

$$= V_{m,d}^{(t)} - \eta \sum_{m'=1}^{M} S_V(m,m') V_{m',d} \ . \tag{4.12}$$

In this case, update of the corresponding $U_{n,:}$ is based on the weighted average of the current U over all rows, including the rows that are entirely missing and the rows that are not, and the weights $S_U(n,n')$ reflect the correlation between the current row n and the rest. The same holds for V and the columns.

4.4 Stochastic Gradient Descent for KPMF As stochastic gradient descent (SGD) usually converges much faster than gradient descent, we also derive the SGD update equations for KPMF below.

The objective function in (4.6) could be rewritten as

$$E = \frac{1}{\sigma^2} \sum_{n=1}^{N} \sum_{m=1}^{M} \delta_{n,m} (R_{n,m} - U_{n,:} V_{m,:}^T)^2$$

$$+ \mathrm{Tr}(U^T S_U U) + \mathrm{Tr}(V^T S_V V), \tag{4.13}$$

where $\mathrm{Tr}(X)$ denotes the trace of matrix X. Moreover,

$$\mathrm{Tr}(U^T S_U U) = \mathrm{Tr}(U U^T S_U)$$

$$= \mathrm{Tr} \Bigg\{ \begin{pmatrix} U_{1,:}^T U_{1,:} & U_{1,:}^T U_{2,:} & \dots & U_{1,:}^T U_{N,:} \\ U_{2,:}^T U_{1,:} & U_{2,:}^T U_{2,:} & \dots & U_{2,:}^T U_{N,:} \\ \vdots & \vdots & \ddots & \vdots \\ U_{N,:}^T U_{1,:} & U_{N,:}^T U_{2,:} & \dots & U_{N,:}^T U_{N,:} \end{pmatrix}$$

$$\begin{pmatrix} S_U(1,1) & S_U(1,2) & \dots & S_U(1,N) \\ S_U(2,1) & S_U(2,2) & \dots & S_U(2,N) \\ \vdots & \vdots & \ddots & \vdots \\ S_U(N,1) & S_U(N,2) & \dots & S_U(N,N) \end{pmatrix} \Bigg\}$$

$$= \sum_{n=1}^{N} \sum_{n'=1}^{N} S_U(n,n') U_{n,:}^T U_{n',:},$$

and similarly,

$$\mathrm{Tr}(V^T S_V V) = \sum_{m=1}^{M} \sum_{m'=1}^{M} S_V(m,m') V_{m,:}^T V_{m,:}.$$

Therefore, (4.13) becomes

$$E = \sum_{n=1}^{N} \sum_{m=1}^{M} \delta_{n,m} \Big[\frac{1}{\sigma^2} (R_{n,m} - U_{n,:} V_{m,:}^T)^2$$

$$+ \frac{1}{\tilde{M}_n} U_{n,:}^T \sum_{n'=1}^{N} S_U(n,n') U_{n',:}$$

$$+ \frac{1}{\tilde{N}_m} V_{m,:}^T \sum_{m'=1}^{M} S_V(m,m') V_{m',:} \Big] = \sum_{n=1}^{N} \sum_{m=1}^{M} \delta_{n,m} E_{n,m} \ ,$$

where $\tilde{M}_n$ is the number of non-missing entries in row n and $\tilde{N}_m$ is the number of non-missing entries in column m. Finally, for each non-missing entry (n,m), taking gradient of $E_{n,m}$ with respect to $U_{n,:}$ and $V_{m,:}$ gives:

$$\frac{\partial E_{n,m}}{\partial U_{n,:}} = -\frac{2}{\sigma^2} (R_{n,m} - U_{n,:}^T V_{m,:}) V_{m,:}$$

$$+ \frac{1}{\tilde{M}_n} \left[\sum_{n'=1}^{N} S_U(n,n') U_{n',:} + S_U(n,n) U_{n,:} \right],$$

$$\frac{\partial E_{n,m}}{\partial V_{m,:}} = -\frac{2}{\sigma^2} \delta_{n,m} (R_{n,m} - U_{n,:}^T V_{m,:}) U_{n,:}$$

$$+ \frac{1}{\tilde{N}_m} \left[\sum_{m'=1}^{M} S_V(m,m') V_{m',:} + S_V(m,m) V_{m,:} \right] \ .$$

4.5 Learning KPMF with Diagonal Kernels If both K_U and K_V are diagonal matrices, meaning that both the rows and the columns are drawn i.i.d., KPMF reduces to PMF. If only one of the kernels is diagonal, it turns out that the corresponding latent matrix can be marginalized, yielding a MAP inference problem on one latent matrix. To illustrate this, suppose K_V is diagonal ($K_V = \sigma_V^2 I$). Following the derivation in [9], the likelihood for each column in R is given by

$$p(R_{\mathbf{n}_{<m>},m} | U, V) = \mathcal{N}(R_{\mathbf{n}_{<m>},m} | (U_{\mathbf{n}_{<m>},:} V_{m,:}^T), \sigma^2 I) \ ,$$

where $\mathbf{n}_{<m>}$ denotes the row indices of non-missing entries in the m^{th} column, so if there are $\tilde{N}$ non-missing entries in the column m, $R_{\mathbf{n}_{<m>},m}$ is a $\tilde{N}$-dimensional vector, and $U_{\mathbf{n}_{<m>},:}$ is a $\tilde{N} \times D$ matrix with each row corresponding to a non-missing entry in column m.

Since $p(V_{m,:} | \sigma_V^2) = \mathcal{N}(V_{m,:} | 0, \sigma_V^2 I)$, we can marginalize over V, obtaining

$$p(R|U) = \prod_{m=1}^{M} p(R_{\mathbf{n}_{<m>},m} | U)$$

$$= \prod_{m=1}^{M} \int_{V_{m,:}} \mathcal{N}(R_{\mathbf{n}_{<m>},m} | (U_{\mathbf{n}_{<m>},:} V_{m,:}^T), \sigma^2 I)$$

$$\times \mathcal{N}(V_{m,:} | 0, \sigma_V^2 I) dV_{m,:}$$

$$= \prod_{m=1}^{M} \mathcal{N}(0, \ \sigma_V^2 U_{\mathbf{n}_{<m>},:} U_{\mathbf{n}_{<m>},:}^T + \sigma^2 I)$$

The objective function then becomes:

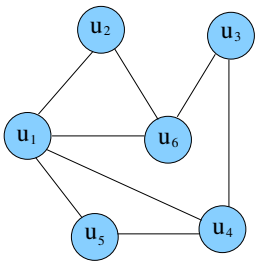
$$E = \sum_{m=1}^{M} \left(R^T_{\mathbf{n}_{<m>},m} C^{-1} R_{\mathbf{n}_{<m>},m} + \log |C| \right) + \sum_{d=1}^{D} U^T_{:,d} S_U U_{:,d} \ , \tag{4.14}$$

where $C = \sigma_V^2 U_{\mathbf{n}_{<m>},:} U^T_{\mathbf{n}_{<m>},:} + \sigma^2 I$. Moreover, since V is no longer a term in the objective function, the gradient descent could be performed solely on U. However, in this case, updating U at each iteration involves the inversion of C, which becomes computationally prohibitive when N is large. Hence we do not use this formulation in our experiments.

4.6 Prediction Learning based on gradient descent or SGD gives us the estimate of the latent matrices $\hat{U}$ and $\hat{V}$. For any missing entry $R_{n,m}$, the maximum-likelihood estimation is the inner product of the corresponding latent vectors, i.e., $\hat{R}_{n,m} = \hat{U}_{n,:} \hat{V}^T_{m,:}$.

5 Experiments on Recommender Systems

In this section, we evaluate the KPMF model for item recommendation with known user relations. In particular, we are given a user-item rating matrix with missing entries as well as a social network graph among users (see Figure 3). The goal is to predict the missing entries in the rating matrix by exploiting both the observed ratings and the underlying rating constraints derived from the social network. We run experiments on two publicly available datasets, i.e. Flixster [7] and Epinion [12], and compare the prediction results of KPMF with several other algorithms.



(a) Social network graph

$$\begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{array} \begin{array}{c} \begin{array}{cccc} i_1 & i_2 & i_3 & i_4 \end{array} \\ \begin{pmatrix} 3 & 5 & ? & 5 \\ ? & 1 & ? & 4 \\ 3 & ? & 4 & 1 \\ ? & ? & 5 & 5 \\ 5 & ? & ? & 2 \\ ? & 4 & 2 & ? \end{pmatrix} \end{array}$$

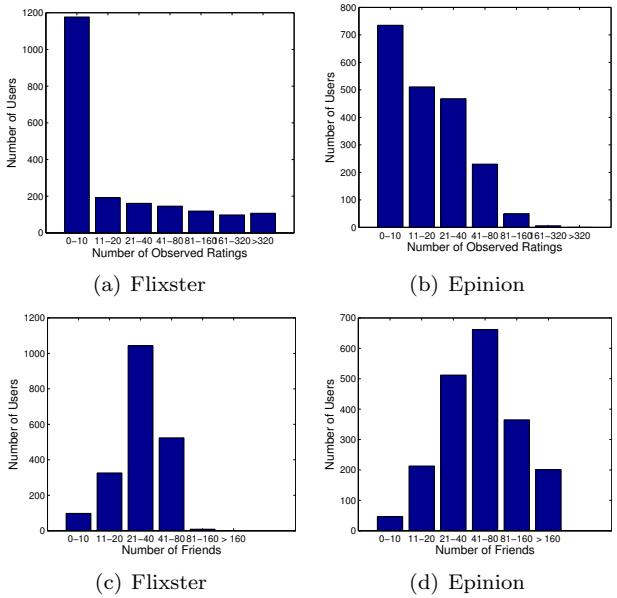
(b) Rating matrix

Figure 3: Example input data: (a) social network among 6 users; (b) observed rating matrix for the 6 users on 4 items.

5.1 Datasets Flixster[2] is a social movie website, where users can rate movies and make friends at the same time. The social graph in Flixster is undirected, and the rating values are 10 discrete numbers ranging from 0.5 to 5 in steps of 0.5.

	Flixster	Epinion
# Users	2000	2000
# Items	3000	3000
# Ratings	173,172	60,485
# Relations	32,548	74,575
Rating Density	2.89%	1.00%

Table 1: Statistics of the datasets used.



(a) Flixster (b) Epinion

(c) Flixster (d) Epinion

Figure 4: (a) and (b): Histograms of users' rating frequencies. (c) and (d): Histograms of the number of friends for each user.

Epinion[3] is a customer review website where users share opinions on various types of items such as electronic products, companies, and movies, through writing reviews or giving ratings. Each user also maintains a list of people he/she trusts, which forms a social network with trust relationships. Unlike Flixster, social network in Epinion is an directed graph, but for simplicity, we convert the directed edges to be undirected ones by keeping only one edge between two users if they are connected in either way originally. The rating values in Epinion are discrete values ranging from 1 to 5.

For each dataset, we sampled a subset with 2,000 users and 3,000 items. For the purpose of testing our hypothesis—whether the social network could help in making ratings prediction—the 2,000 users selected are users with most friends in the social network, while the 3,000 items selected are the most frequently rated overall. The statistics of the datasets are given in

[2] www.flixster.com

[3] www.epinions.com

Table 1. Figure 4 shows the histograms for the number of past ratings and number of friends each user has.

Unless otherwise specified, from all the ratings, we randomly hold out 10% ratings as the validation set and another 10% as the test set. The rest 80% of ratings, along with the whole social network, are used for creating the training sets. To better evaluate the effect of social network, we have 4 training sets that are increasingly sparse, i.e., we use not only all the 80% but also 60%, 40%, and 20% of the ratings to create 4 different training sets (note that the social network remains the same). For each observed rating r, we normalize it to $[0.2, 1]$ using r/r_{max}, where r_{max} is the maximum possible value for the ratings.

5.2 Graph Kernels To construct kernel matrices suitable to our problem, we consider the users' social network as an undirected, unweighted graph G with nodes and edges representing users and their connections. Elements in the adjacency matrix of G are determined by $A_{i,j} = 1$ if there's an edge between user i and j, and 0 otherwise. The Laplacian matrix [4] of G is defined as $L = D - A$, where the degree matrix D is a diagonal matrix with diagonal entries $d_i = \sum_{j=1}^{N} A_{i,j} (i = 1, ..., N)$.

Graph kernels provide a way of capturing the intricate structure among nodes in a graph (If instead we are given features or attributes of the users, we could replace graph kernels with polynomial kernels, RBF kernels, etc. [17]). In our case, a graph kernel defines a similarity measure for users' taste on certain items. Generally speaking, users tend to have similar taste with their friends and families, and thus their ratings for the same items would also be correlated. Graph kernels could capture such effects in the social network, and the resulted kernel matrix would provide key information about users' rating patterns.

In this work, we examine three different graph kernels and refer readers to [8] for more available choices.

Diffusion kernel: The Diffusion kernel proposed in [8] is derived from the idea of matrix exponential, and has a nice interpretation on the diffusion process of substance such as heat. In particular, if we let some substance be injected at node i and flow along the edges of the graph, $K_D(i, j)$ can be regarded as the amount of the substance accumulated at node j in the steady state. The diffusion kernel intuitively captures the global structure among nodes in the graph, and it can be computed as follows:

$$K_D = \lim_{n \to \infty} \left(1 - \frac{\beta L}{n}\right)^n = e^{-\beta L}, \tag{5.15}$$

where β is the bandwidth parameter that determines the extent of diffusion ($\beta = 0$ means no diffusion).

Commute Time (CT) kernel: As proposed in [6], the Commute Time kernel is closely related to the so-called average commute time (the number of steps a random walker takes to commute between two nodes in a graph), and can be computed using the pseudo-inverse of the Laplacian matrix: $K_{CT} = L^{\dagger}$.

Moreover, since K_{CT} is conditionally positive definite, $\sqrt{K_{CT}(i, j)}$ behaves exactly like an Euclidean distance between nodes in the graph [10]. As a consequence, the nodes can be isometrically embedded in the subspace of $\mathbb{R}^n$ (n is the number of nodes), where the Euclidean distance between the points is $\sqrt{K_{CT}(i, j)}$.

Regularized Laplacian (RL) kernel: Smola and Kondor [19] introduce a way of performing regularization on graphs that penalizes the variation between adjacent nodes. In particular, it turns out that the graph Laplacian could be equally defined as a linear operator on the nodes of the graph, and naturally induces a semi-norm on $\mathbb{R}^n$. This semi-norm quantifies the variation of adjacent nodes, and could be used for designing regularization operators. Furthermore, such regularization operators give rise to a set of graph kernels, and among them is the Regularized Laplacian kernel:

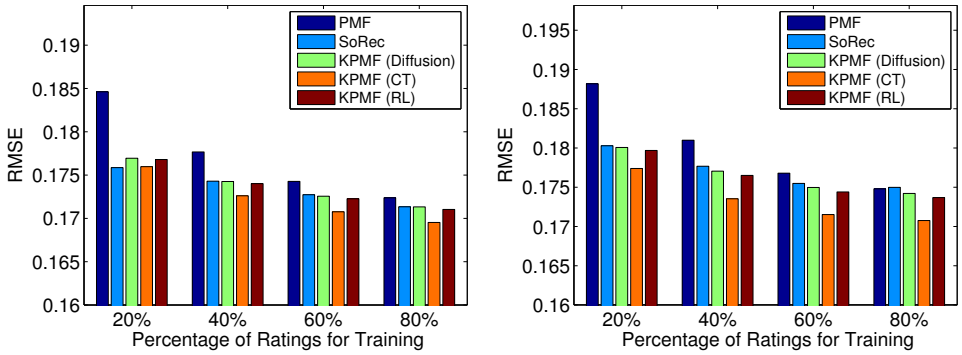
$$K_{RL} = (I + \gamma L)^{-1}, \tag{5.16}$$

where $\gamma > 0$ is a constant.

5.3 Methodology Given the social network, we use the Diffusion kernel, the CT kernel, and the RL kernel described above to generate the covariance matrix K_U for users. The parameter setting is as follows: $\beta = 0.01$ for the Diffusion kernel and $\gamma = 0.1$ for the RL kernel, which are chosen via validation. Since no side information is available for the items in the Flixster or Epinion dataset, K_V is assumed diagonal: $K_V = \sigma_V^2 I$, where I is an $M \times M$ identity matrix.

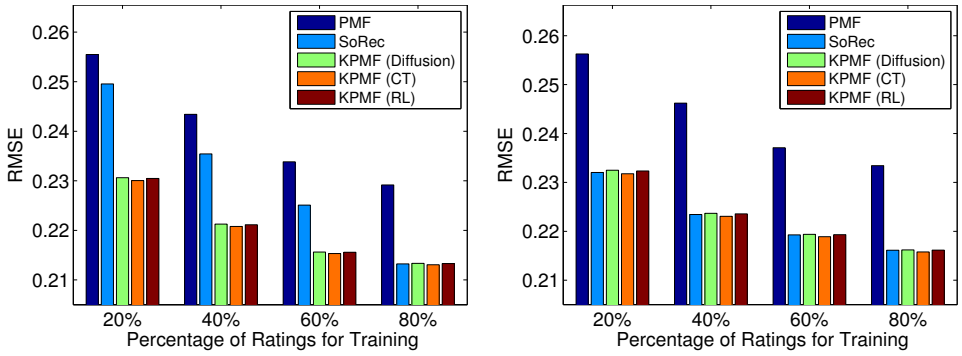
Given the covariance matrix K_U generated from the graph kernels, we perform gradient descent (KPMF with stochastic gradient descent is discussed later) on U and V using the update equations (4.7) and (4.8).[4] At each iteration, we evaluate the Root Mean Square Error (RMSE) on the validation set, and terminate training once the RMSE starts increasing, or the maximum number of iterations allowed is reached. The learned U and V are then used to predict the ratings in the test set. We run the algorithm with different latent vector dimensions, viz, $D = 5$ and $D = 10$.

[4] While gradient descent in KPMF involves the inverse of K_U, we actually don't need to invert the matrix when using the CT kernel or the RL kernel since $K_{CT}^{-1} = L$ and $K_{RL}^{-1} = I + \gamma L$.



(a) Flixster, D=5

(b) Flixster, D=10



(c) Epinion, D=5

(d) Epinion, D=10

Figure 5: RMSE for different algorithms on Flixster and Epinion datasets (best viewed in color). Lower is better.

We compare the performance of KPMF (using gradient descent) with three algorithms: The first one uses only the information from the social network. More specifically, to predict the rating $R_{n,m}$, we take the neighbors of user n from the social network, and average their ratings on item m as the prediction for $R_{n,m}$. We denote this method as social network based algorithm (SNB). If none of the neighbors of user n has rated item m, SNB cannot predict $R_{n,m}$. The second algorithm we compare with is PMF [14], which only uses the information from the rating matrix.[5] The third algorithm is SoRec [11], a state-of-the-art collaborative filtering algorithm that combines information from both the rating matrix and the social network by performing matrix factorization jointly. In addition, we also compare the computational efficiency between KPMF using gradient descent and KPMF using stochastic gradient descent.

Another experiment is prediction for users with no past ratings. We test on 200 users who have most connections in the social network (so that the effect of the social network is most evident). All the past ratings made by these 200 users in the training set are not used for learning. Therefore, the observed rating matrix would contain 200 rows of zeros, and the rest 1,800 rows remain unchanged.

The main measure we use for performance evaluation is Root Mean Square Error (RMSE):

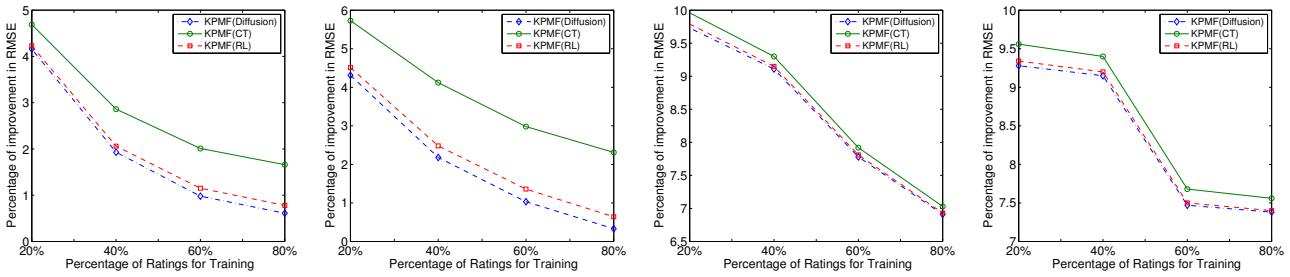
$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(r_i - \hat{r}_i)^2}{n}}\ , \tag{5.17}$$

where r_i denotes the ground-truth rating value, $\hat{r}_i$ denotes its predicted value, and n is the total number of ratings to be predicted.

For SNB, since we cannot do missing value prediction for some entries if none of the corresponding user's neighbors has rated the target item, we also define a measure of *coverage* for reference, which is defined as the percentage of ratings that can be predicted among all the test entries.

5.4 Results The RMSE on Flixster and Epinion for PMF, SoRec, and KPMF (with different kernels) are given in Figure 5. In each plot, we show the results with different number of ratings used for training, ranging from 20% to 80% of the whole dataset. The main

[5] BPMF actually performs worse than PMF in our experiments, which might be due to improper parameter setting in the code published by the authors. Thus we omit reporting BPMF results here.



(a) Flixster, D=5 (b) Flixster, D=10 (c) Epinion, D=5 (d) Epinion, D=10

Figure 6: Performance improvement of KPMF compared to PMF on training sets with different number of observed ratings (best viewed in color). The improvement of KPMF versus PMF decreases as more training data are used. This is because for sparser datasets, PMF would have relatively more difficulty in learning users' preferences from fewer number of past ratings, while KPMF could still take advantage of the known social relations among users and utilize the observed ratings better.

(a) Flixster

Training data used	20%	40%	60%	80%
RMSE	0.289	0.280	0.272	0.267
Coverage	0.31	0.50	0.62	0.70

(b) Epinion

Training data used	20%	40%	60%	80%
RMSE	0.276	0.264	0.259	0.253
Coverage	0.45	0.64	0.73	0.80

Table 2: RMSE and Coverage from SNB on Flixster and Epinion.

observations are as follows:

1. KPMF, as well as SoRec, outperforms PMF on both Flixster and Epinion, regardless of the number of ratings used for training. While KPMF and SoRec use both the social network and the rating matrix for training, PMF uses the rating matrix alone. The performance improvement of KPMF and SoRec over PMF suggests that social network is indeed playing a role in helping predict the ratings. In addition, KPMF also outperforms SoRec for most cases.

2. Figure 6 shows KPMF's percentage of improvement compared to PMF in terms of RMSE. For KPMF, we can see that the performance gain increases as the training data gets sparser. It implies that when the information from the rating matrix is getting weaker, the users' social network is getting more useful for prediction.

3. As shown in Figure 5, among the three graph kernels examined, the CT kernel leads to the lowest RMSE on both Flixster and Epinion. The advantage is more obvious on Flixster than on Epinion.

4. We also give the RMSE of SNB in Table 2 for reference. RMSE for this simple baseline algorithm is much higher than the other algorithms. The coverage is low with a sparse training matrix, but gets higher when the sparsity decreases.

Table 3 shows the results for the experiment of prediction for users with no past ratings. The RMSE are over the selected 200 users who have most connections in the social network, and their past ratings are not utilized during training. To contrast, we only show results on the datasets with 20% and 80% training data. KPMF consistently outperforms Item Average[6] by a large margin if 20% training data are used, but the advantage is not so obvious for the dataset of 80% training data (Note that Item Average actually outperforms KPMF on Epinion for Diffusion and RL kernels). This result again implies that the side information from users' social network is more valuable when the observed rating matrix is sparse, and the sparsity is indeed often encountered in real data [16].

Finally, we compare the computational efficiency between KPMF with stochastic gradient descent (KPMF_{SGD}) and KPMF with gradient descent (KPMF_{GD}). Table 4 shows the RMSE results and running time for the two, where we set $D = 10$ and use $p\% = 20\%$ of ratings for training. Although KPMF_{SGD} has slightly higher RMSE than KPMF_{GD}, it is hundreds of times faster. Similar results are also observed in experiments with other choices of D and $p\%$. Therefore, for large scale datasets in real applications, KPMF_{SGD} would be a better choice.

[6]The algorithm that predicts the missing rating for an item as the average of its observed ratings by other users.

(a) 20% training data used

	Flixster		Epinion	
	D = 5	D = 10	D = 5	D = 10
Item Average	0.2358	0.2358	0.3197	0.3197
KPMF(Diffusion)	0.2183	0.2180	0.2424	0.2436
KPMF(CT)	0.2184	0.2180	**0.2375**	**0.2378**
KPMF(RL)	**0.2182**	**0.2179**	0.2422	0.2433

(b) 80% training data used

	Flixster		Epinion	
	D = 5	D = 10	D = 5	D = 10
Item Average	0.2256	0.2256	0.2206	0.2206
KPMF(Diffusion)	0.2207	0.2209	0.2257	0.2269
KPMF(CT)	0.2209	0.2208	**0.2180**	**0.2180**
KPMF(RL)	**0.2206**	**0.2207**	0.2252	0.2263

Table 3: RMSE on users with no ratings for training.

	Flixster		Epinion	
	KPMF_{GD}	KPMF_{SGD}	KPMF_{GD}	KPMF_{SGD}
RMSE	**0.180**	0.184	**0.232**	0.240
Time (sec)	1353.6	**5.5**	1342.3	**7.8**

Table 4: Comparison of RMSE and running time for KPMF_{GD} and KPMF_{SGD}. KPMF_{SGD} is slightly worse than KPMF_{GD} in terms of RMSE, but significantly faster.

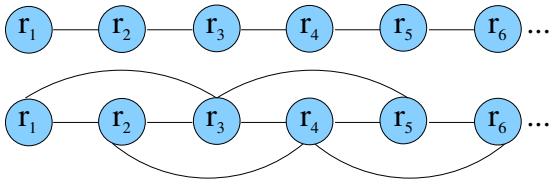


Figure 7: The graph constructed for the rows of the image using the spatial smoothness property. (Top: $\Delta = 1$, Bottom: $\Delta = 2$.)

6 Experiments on Image Restoration

In this section, we demonstrate the use of KPMF in image restoration to further illustrate the broad potential of this proposed framework and the relevance of incorporating side information. Image restoration is the process of recovering corrupted regions of a target image [3]. Let us denote the $N \times M$ target image by P, and the corrupted region to be recovered by Ω (see black scribbles on the second column of Figure 8). The task is to fill in the pixels inside Ω in a way that is coherent with the known pixels outside Ω, i.e. $P - \Omega$. Now one might notice that this problem is quite similar to the one faced in recommender systems, where the rating matrix R becomes P, ratings become pixel values, and the missing entries become Ω. Therefore, if we consider the rows of the image as users and the columns as items, we could apply the KPMF algorithm to fill in the pixels in Ω just as predicting missing entries in recommender systems. However, since no direct information of correlations among rows or columns of the image is given, the difficulty arises in obtaining proper kernel matrices.

One way to address this is to construct such a graph for images in analogy to the users' social network for recommender systems. To do that, we consider the spatial smoothness in an image (while this will be used here to illustrate the proposed framework, we can consider graphs derived from other attributes as well, e.g., smoothness in feature space, with features derived from local patches or texture-type multiscale analysis [5]). Below we describe how to construct the graph for rows using this property (the graph for columns can be constructed in a similar fashion). First, we assume that each row is similar to its neighboring rows and thus directly connected in the graph. Let r_i $(i = 1, ..., N)$ be the node in the graph that represents the i^{th} row of the image (nodes representing parts of the image rows could be considered as well to further localize the structure). Then there exists an edge between r_i and r_j $(j \neq i)$ if and only if $|i - j| \leq \Delta$, where Δ is a constant that determines the degree of r_i (see Figure 7). The corresponding adjacency matrix of the graph is a *band matrix* with 1's confined to the diagonal band and 0's elsewhere.

Given the graphs for rows and columns, we can obtain their corresponding kernel matrices by applying the graph kernels from Section 5.2. Since each color image is composed of three channels (Red, Green and Blue), the KPMF update equations for learning the latent matrices are applied to each channel independently. Finally, the estimation for Ω from the three channels, along with the known pixels in $P - \Omega$, are combined together to form the restored image. Restoration results using PMF and KPMF on several corrupted images are shown in Figure 8, and the RMSE comparison is given in Table 5 (note that all pixel values are normalized to [0, 1].). Unlike KPMF that is able to utilize the spatial smoothness assumption, PMF can only use the observed pixels in the image for restoration. Thus, as expected, its restoration quality is worse than KPMF.

7 Conclusion

We have presented a new matrix completion algorithm–KPMF, which exploits the underlying covariances among rows and among columns of the data matrix simultaneously for missing value prediction. KPMF introduces Gaussian process priors for latent matrices in the

	KPMF			PMF			
Image	Channel R	Channel G	Channel B	Channel R	Channel G	Channel B	#Masked pixels
1	0.082	0.073	0.066	0.120	0.101	0.094	5426
2	0.160	0.164	0.156	0.173	0.177	0.170	5975
3	0.135	0.133	0.131	0.179	0.171	0.164	31902
4	0.100	0.106	0.124	0.149	0.141	0.173	32407
5	0.103	0.049	0.030	0.143	0.081	0.040	28025
6	0.109	0.091	0.081	0.141	0.109	0.105	23061

Table 5: RMSE comparison between PMF and KPMF on RGB channels of restored images. Smaller is better.

generative model, which forces the learned latent matrices to respect the covariance structure among rows and among columns, enabling the incorporation of side information when learning the model. As demonstrated in the experiments, this characteristic could play a critical role in boosting the model performance, especially when the observed data matrix is sparse.

Another advantage of KPMF over PMF and BPMF is its ability to predict even when an entire row/column of the data matrix is missing as long as appropriate side information is available. In principle, KPMF is applicable to general matrix completion problems, but in this paper we focus on two specific applications: recommender systems and image restoration. In the future, we would like to generalize the current model to handle the case of weighted entries, where different entries are assigned different weights according to some pre-defined criteria.

Acknowledgments

The research was supported by NSF grants IIS-0812183, IIS-0916750, IIS-1029711, and NSF CAREER award IIS-0953274. Guillermo Sapiro is supported by NSF, ONR, ARO, NGA, NSSEFF, and DARPA.

References

[1] Deepak Agarwal and Bee-Chung Chen. flda: Matrix factorization through latent dirichlet allocation. In *WSDM*, 2010.

[2] Amrudin Agovic, Arindam Banerjee, and Snigdhansu Chatterjee. Probabilistic matrix addition. In *ICML*, 2011.

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of ACM SIGGRAPH*, pages 417–424, 2000.

[4] F.R.K. Chung. *Spectral graph theory*. American Mathematical Society, 1997.

[5] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *Proc. ICCV*, pages 1033–1038, 1999.

[6] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[7] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*, 2010.

[8] R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *ICML*, 2001.

[9] N. Lawrence and R. Urtasun. Non-linear matrix factorization with Gaussian processes. In *ICML*, 2009.

[10] L. Lovasz. Random walks on graphs: a survey. *Combinatorics, Paul Erdos is Eighty*, 2:353–397, 1996.

[11] H. Ma, H. Yang, and I. King M. Lyu. Social recommendation using probabilistic matrix factorization. In *CIKM*, 2008.

[12] P. Massa and P. Avesani. Trust metrics in recommender systems. In *Computing with Social Trust*. Springer London, 2009.

[13] Carl Edward Rasmussen. *Gaussian Processes in Machine Learning*. MIT Press, 2006.

[14] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

[15] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *ICML*, 2008.

[16] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 10, pages 285–295, 2001.

[17] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2001.

[18] H. Shan and A. Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *ICDM*, 2010.

[19] A. Smola and R. Kondor. Kernels and regularization on graphs. *The Sixteenth Annual Conference on Learning Theory/The Seventh Workshop on Kernel Machines*, 2003.

[20] H. Wackernagel. *Multivariate Geostatistics: An Introduction With Applications*. Springer-Verlag, 2003.

[21] Chong Wang and David Blei. Collaborative topic modeling for recommending scientific articles. In *SIGKDD*, 2011.

Figure 8: Image restoration results using PMF and KPMF (best viewed in color). From left to right: original images, corrupted images (regions to be restored are in black), images restored using PMF, and images restored using KPMF. For KPMF, Δ equals to 5 when constructing the row and column graphs, and Diffusion kernel with $\beta = 0.5$ is used to obtain the kernel matrices.