Convex Factorization Machine for Toxicogenomics Prediction*

Makoto Yamada
RIKEN AIP, JST PRESTO

Wenzhao Lian
Vicarious

Amit Goyal
Yahooo Research

Jianhui Chen
Microsoft

Kishan Wimalawarne
Kyoto University

Suleiman A Khan
University of Helsinki

Samuel Kaski
Aalto University

Hiroshi Mamitsuka
Kyoto University, Aalto University

Yi Chang
Huawei Research America

ABSTRACT

We introduce the *convex factorization machine* (CFM), which is a convex variant of the widely used Factorization Machines (FMs). Specifically, we employ a linear+quadratic model and regularize the linear term with the ℓ_2-regularizer and the quadratic term with the *trace norm* regularizer. Then, we formulate the CFM optimization as a semidefinite programming problem and propose an efficient optimization procedure with Hazan's algorithm. A key advantage of CFM over existing FMs is that it can find a globally optimal solution, while FMs may get a poor locally optimal solution since the objective function of FMs is non-convex. In addition, the proposed algorithm is simple yet effective and can be implemented easily. Finally, CFM is a general factorization method and can also be used for other factorization problems, including multi-view matrix factorization and tensor completion problems, in various domains including toxicogenomics and bioinformatics. Through synthetic and traditionally used movielens datasets, we first show that the proposed CFM achieves results competitive to FMs. We then show in a toxicogenomics prediction task that CFM predicts the toxic outcomes of a collection of drugs better than a state-of-the-art tensor factorization method.

CCS CONCEPTS

•Applied computing → Bioinformatics; •Computing methodologies → Factorization methods;

KEYWORDS

Factorization Machines, Convex, Toxicogenomics Prediction

ACM Reference format:
Makoto Yamada, Wenzhao Lian, Amit Goyal, Jianhui Chen, Kishan Wimalawarne, Suleiman A Khan, Samuel Kaski, Hiroshi Mamitsuka, and Yi Chang. 2017. Convex Factorization Machine for Toxicogenomics Prediction. In *Proceedings of KDD'17, August 13–17, 2017, Halifax, NS, Canada.,* , 11 pages.

*Produces the permission block, and copyright information

KDD'17, August 13–17, 2017, Halifax, NS, Canada.

978-1-4503-4887-4/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3097983.3098103

DOI: http://dx.doi.org/10.1145/3097983.3098103

1 INTRODUCTION

In recommendation tasks including movie recommendation and news article recommendation, the data are represented in a matrix form, $\boldsymbol{A} \in \mathbb{R}^{|U|\times|I|}$, where $|U|$ is the number of users and $|I|$ is the number of items, respectively. Matrix factorization (MF), which imputes missing entries of a matrix with a *low-rank* constraint, is widely used in recommendation systems for news recommendation, protein-protein interaction prediction, transfer learning, social media user modeling, multi-view learning, and modeling text document collections, among others [12, 14, 24, 27, 33, 48, 50–52, 54].

Moreover, in increasingly complex recommendation and prediction tasks, data sets tend to be represented as multi-view matrices or tensors with more than two modes [21, 22]. Such methodologies have been used in various prediction tasks such as toxicogenomics, bioinformatics, brain activity, chemometrics and temporal sales prediction [15, 21, 25]. Toxicogenomics is a recent and promising application domain, where the task is to predict the toxicological response of the drugs by learning the associations with genomic profiles of cells.

Recently, a general framework of MF called the *factorization machines* (FMs) has been proposed [35–37]. FMs are applied to many regression and classification problems, including the display advertising challenge[1], and they show state-of-the-art performance. The key contribution of FMs is that they reformulate recommendation problems as regression problems, where the input $\boldsymbol{x}$ is a feature vector that indicates the k-th user and the k'-th item, and output y is the rating of the user-item pair:

$$\boldsymbol{x}_i = [\overbrace{0 \cdots 0 \underbrace{1}_{k\text{-th user}} 0 \cdots 0}^{|U|} \overbrace{0 \cdots 0 \underbrace{1}_{k'\text{-th item}} 0 \cdots 0}^{|I|}]^\top \in \mathbb{R}^d,$$
$$y_i = [\boldsymbol{A}]_{k,k'}.$$

Here, $^\top$ is matrix transpose and $d = |U| + |I|$ is the dimensionality of $\boldsymbol{x}$, $[\boldsymbol{A}]_{k,k'}$ is the score of the k-th user and k'-th item, and $|\boldsymbol{A}| = n$ is the number of non-zero elements. The goal of FMs is to find a model that predicts y given an input $\boldsymbol{x}$.

[1] https://www.kaggle.com/c/criteo-display-ad-challenge

The following linear + feature interaction model is employed for FMs:

$$f(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{G}) = w_0 + \boldsymbol{w}_0^\top \boldsymbol{x} + \sum_{\ell=1}^{d} \sum_{\ell'=\ell+1}^{d} \boldsymbol{g}_\ell^\top \boldsymbol{g}_{\ell'} x_\ell x_{\ell'},$$

where $w_0 \in \mathbb{R}$, $\boldsymbol{w}_0 \in \mathbb{R}^d$, and $\boldsymbol{G} = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m] \in \mathbb{R}^{d \times m}$ are model parameters ($m \ll d$). Since only the k-th user and k'-th item element of the input vector $\boldsymbol{x}$ are non-zero, the model can also be written as

$$\widehat{\boldsymbol{A}}_{k,k'} = w_0 + [\boldsymbol{w}_0]_k + [\boldsymbol{w}_0]_{|U|+k'} + \boldsymbol{g}_k^\top \boldsymbol{g}_{|U|+k'},$$

which is equivalent to the matrix factorization model with global, user, and item biases. Moreover, since FMs solve the matrix completion problem through regression, it is easy to utilize side information such as user's and article's meta-information by simply concatenating the meta-information to $\boldsymbol{x}$.

For regression problems, the model parameters are estimated by solving the following optimization problem:

$$\min_{w_0, \boldsymbol{w}, \boldsymbol{G}} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i; w_0, \boldsymbol{w}_0, \boldsymbol{G}))^2 + \lambda_1 \|w_0\|_2^2 + \lambda_2 \|\boldsymbol{w}_0\|_2^2 + \lambda_3 \|\boldsymbol{G}\|_F^2,$$

where the λ_1, λ_2, and λ_3 are regularization parameters, and $\|\boldsymbol{G}\|_F$ is the Frobenius norm. In [36], stochastic gradient descent (SGD), alternating least squares (ALS), and Markov Chain Monte Carlo (MCMC) based approaches were proposed. These optimization approaches work well in practice if regularization parameters and the initial solution of parameters are set appropriately. However, since the loss function is non-convex with respect to $\boldsymbol{G}$, it can converge to a poor local optimum (mode). The MCMC-based approach tends to obtain a better solution than ALS and SGD. However, it requires running the sampler long enough to explore different local modes.

In this paper, we propose the *convex factorization machine* (CFM). We employ the linear+quadratic model, Eq. (1) and estimate $\boldsymbol{w}$ and $\boldsymbol{W}$ such that the squared loss between the output $\boldsymbol{y}$ and the model prediction is minimized. More specifically, we regularize the linear parameter $\boldsymbol{w}$ with the ℓ_2-regularizer and the quadratic parameter $\boldsymbol{W}$ with the *trace norm* regularizer. Then, we formulate the CFM optimization problem as a semidefinite programming problem and solve it with Hazan's algorithm [13], which is a Frank-Wolfe algorithm [11, 18]. A key advantage of the proposed method over existing FMs is that CFM can find a globally optimal solution, while FM can get poor locally optimal solutions. Moreover, our proposed framework is a general variant of convex matrix factorization with nuclear norm regularization, and the CFM algorithm is simple and can be implemented easily. Finally, since CFM is a general factorization framework, it can be easily applied to any factorization problems including multi-view factorization [21] and tensor factorization [44]. We demonstrate the effectiveness of the proposed method first through synthetic and benchmark datasets. Then, we show that the proposed method outperforms a state-of-the-art multi-view factorization method on toxicogenomics data.

Contribution: The contributions of this paper are summarized below:

- We formulate the FM problem as a semidefinite programming problem, which is a convex formulation.
- We show that the proposed CFM framework includes the matrix factorization with nuclear norm regularization [19] as a special case.
- We formulate a Tucker-based tensor completion problem [43, 44, 46] as a CFM problem. Thanks to the formulation, we can naturally handle large-scale sparse tensor completion problems. To our knowledge, this is the first work.
- We propose a simple yet efficient optimization procedure for the semidefinite programming problem using Hazan's algorithm [13].
- We applied the proposed CFM to a toxicogenomics prediction task; it outperformed a state-of-the-art method.

2 PROPOSED METHOD

In this section, we propose the *convex factorization machine* (CFM) for regression problems [2].

2.1 Problem Formulation

We suppose that we are given n independent and identically distributed (i.i.d.) paired samples $\{(\boldsymbol{x}_i, y_i) \mid \boldsymbol{x}_i \in \mathcal{X},\ y_i \in \mathcal{Y},\ i = 1, \ldots, n\}$ drawn from a joint distribution with density $p(\boldsymbol{x}, y)$. We denote $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ as the input data and $\boldsymbol{y} = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$ as the output real-valued vector.

The goal of this paper is to find a model that predicts y given an input $\boldsymbol{x}$.

2.2 Model

We employ the following model:

$$\begin{aligned} f(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{W}) &= w_0 + \boldsymbol{w}_0^\top \boldsymbol{x} + \sum_{\ell=1}^{d} \sum_{\ell'=\ell+1}^{d} [\boldsymbol{W}]_{\ell,\ell'} x_\ell x_{\ell'}, \\ &= \boldsymbol{w}^\top \boldsymbol{z} + \frac{1}{2} \mathrm{tr}(\boldsymbol{W}(\boldsymbol{x}\boldsymbol{x}^\top - \mathrm{diag}(\boldsymbol{x} \circ \boldsymbol{x}))), \end{aligned} \tag{1}$$

where $\boldsymbol{z} = [1\ \boldsymbol{x}^\top]^\top \in \mathbb{R}^{d+1}$, $\boldsymbol{w} = [w_0\ \boldsymbol{w}_0^\top]^\top \in \mathbb{R}^{d+1}$, $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ is a positive semi-definite matrix, $\mathrm{tr}(\boldsymbol{X})$ is the trace operator, $\circ$ is the elementwise product, and $\mathrm{diag}(\boldsymbol{x}) \in \mathbb{R}^{d \times d}$ is the diagonal matrix whose diagonal elements are $\boldsymbol{x}$. The difference between the FMs model and Eq. (1) is that $\boldsymbol{g}_k^\top \boldsymbol{g}_{k'}$ is parametrized as $W_{k,k'}$. Note that we implicitly assume that $\boldsymbol{W}$ is a symmetric matrix in Eq. (1). Moreover, since $\boldsymbol{W}$ is low-rank, the positive semi-definite condition for $\boldsymbol{W}$ is reasonable.

Here, we show that the proposed parametrization makes the optimization convex under arbitrary loss function. The model can equivalently be written as

$$f(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{W}) = [\boldsymbol{w}^\top\ \mathrm{vec}(\boldsymbol{W})^\top] \begin{bmatrix} \boldsymbol{z} \\ \frac{1}{2}\mathrm{vec}(\boldsymbol{x}\boldsymbol{x}^\top - \mathrm{diag}(\boldsymbol{x} \circ \boldsymbol{x})) \end{bmatrix},$$

where $\mathrm{vec}(\boldsymbol{W}) \in \mathbb{R}^{d^2}$ is the vectorization operator. Since the model is a linear model, the optimization problem is jointly convex with respect to both $\boldsymbol{w}$ and $\boldsymbol{W}$ if we employ a loss function such as squared loss and logistic loss.

[2] Code available at http://www.makotoyamada-ml.com/cfm.html

2.3 Optimization problem

We formulate the optimization problem of CFM as a semidefinite programming problem:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{W}} \quad & \|\boldsymbol{y} - \boldsymbol{f}(X;\boldsymbol{w},\boldsymbol{W})\|_2^2 + \lambda_1\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{W} \succeq 0 \text{ and } \|\boldsymbol{W}\|_{\text{tr}} = \eta, \end{aligned} \tag{2}$$

where

$$\boldsymbol{f}(X;\boldsymbol{w},\boldsymbol{W}) = [f(\boldsymbol{x}_1;\boldsymbol{w},\boldsymbol{W}),\ldots,f(\boldsymbol{x}_n;\boldsymbol{w},\boldsymbol{W})]^\top \in \mathbb{R}^n,$$

and $\lambda_1 \geq 0$ and $\eta \geq 0$ are regularization parameters. $\|\boldsymbol{W}\|_{\text{tr}}$ is the trace norm defined as

$$\|\boldsymbol{W}\|_{\text{tr}} = \text{tr}(\sqrt{\boldsymbol{W}^\top\boldsymbol{W}}) = \sum_{i=1}^{d}\sigma_i,$$

where σ_i is the i-th singular value of $\boldsymbol{W}$. The trace norm is also referred to as the *nuclear norm* [9]. Since the singular values are non-negative, the trace norm can be regarded as the ℓ_1 norm on singular values. Thus, by imposing the trace norm, we can make $\boldsymbol{W}$ to be low-rank.

To derive a simple yet effective optimization algorithm, we first eliminate $\boldsymbol{w}$ from the optimization problem Eq.(2) and convert the problem to a convex optimization problem with respect to $\boldsymbol{W}$. Specifically, we take the derivative of the objective function with respect to $\boldsymbol{w}$ and obtain an analytical solution for $\boldsymbol{w}$:

$$\boldsymbol{w}^* = (ZZ^\top + \lambda_1\boldsymbol{I}_{d+1})^{-1}Z(\boldsymbol{y} - \boldsymbol{f}_Q(X;\boldsymbol{W})),$$

where

$$\begin{aligned} \boldsymbol{f}_Q(X;\boldsymbol{W}) &= [f_Q(\boldsymbol{x}_1;\boldsymbol{W}),\ldots,f_Q(\boldsymbol{x}_n;\boldsymbol{W})]^\top \in \mathbb{R}^n, \\ f_Q(\boldsymbol{x};\boldsymbol{W}) &= \frac{1}{2}\text{tr}(\boldsymbol{W}(\boldsymbol{x}\boldsymbol{x}^\top - \text{diag}(\boldsymbol{x}\circ\boldsymbol{x}))), \end{aligned}$$

is the model corresponding to the quadratic term of $f(\boldsymbol{x};\boldsymbol{w},\boldsymbol{W})$ such that $f(\boldsymbol{x};\boldsymbol{w},\boldsymbol{W}) = \boldsymbol{w}^\top\boldsymbol{z} + f_Q(\boldsymbol{x};\boldsymbol{W})$, $Z = [\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n] \in \mathbb{R}^{d+1\times n}$, $\boldsymbol{I}_{d+1} \in \mathbb{R}^{d+1\times d+1}$ is the identity matrix. Note that $\boldsymbol{w}^*$ depends on the unknown parameter $\boldsymbol{W}$.

Plugging $\boldsymbol{w}^*$ back into the objective function of Eq.(2), we can rewrite the objective function as

$$\min_{\boldsymbol{W}} J(\boldsymbol{W}) \text{ s.t. } \boldsymbol{W} \succeq 0 \text{ and } \|\boldsymbol{W}\|_{\text{tr}} = \eta, \tag{3}$$

where

$$J(\boldsymbol{W}) = (\boldsymbol{y} - \boldsymbol{f}_Q(X;\boldsymbol{W}))^\top C(\boldsymbol{y} - \boldsymbol{f}_Q(X;\boldsymbol{W})),$$

$C = R^\top R + \lambda_1 H^\top H$, $R = \boldsymbol{I}_n - Z^\top(ZZ^\top + \lambda_1\boldsymbol{I}_{d+1})^{-1}Z$, and $H = (ZZ^\top + \lambda_1\boldsymbol{I}_{d+1})^{-1}Z$.

Once $\widehat{\boldsymbol{W}}$ is obtained by solving Eq. (3), we can get the estimated linear parameter $\widehat{\boldsymbol{w}}$ as

$$\widehat{\boldsymbol{w}} = (ZZ^\top + \lambda_1\boldsymbol{I}_{d+1})^{-1}Z(\boldsymbol{y} - \boldsymbol{f}_Q(X;\widehat{\boldsymbol{W}})).$$

Relation to Matrix Factorization with Nuclear Norm Regularization: The constraint on $\boldsymbol{W}$ can be written as

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{U}\boldsymbol{U}^\top & \boldsymbol{M} \\ \boldsymbol{M}^\top & \boldsymbol{V}\boldsymbol{V}^\top \end{bmatrix} \succeq 0,\ \text{tr}(\boldsymbol{U}\boldsymbol{U}^\top) + \text{tr}(\boldsymbol{V}\boldsymbol{V}^\top) = \eta,$$

Algorithm 1 CFM with Hazan's Algorithm

Rescale loss function $J_\eta(\boldsymbol{W}) = J(\eta\boldsymbol{W})$.
Initialize $\boldsymbol{W}^{(1)}$, the curvature parameter $C_f = 1$, and the number of iterations T.
for all $t = 0, 1\ldots, T$ **do**
  Compute $\boldsymbol{p}^{(t)} = \text{approxEV}\big(-\nabla J_\eta(\boldsymbol{W}^{(t)}), \frac{C_f}{(t+1)^2}\big)$.
  $\widehat{\alpha}_t := \frac{2}{t+2}$ (or $\widehat{\alpha}_t = \text{argmin}_\alpha\ J_\eta(\boldsymbol{W}^{(t)} + \alpha(\boldsymbol{p}^{(t)}\boldsymbol{p}^{(t)\top} - \boldsymbol{W}^{(t)}))$).
  $\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} + \widehat{\alpha}_t(\boldsymbol{p}^{(t)}\boldsymbol{p}^{(t)\top} - \boldsymbol{W}^{(t)})$.
end for
return $\boldsymbol{W}^{(T)}$.

where $\boldsymbol{U} \in \mathbb{R}^{|U|\times m}$, $\boldsymbol{V} \in \mathbb{R}^{|I|\times m}$ and $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{V}^\top \in \mathbb{R}^{|U|\times|I|}$. Furthermore, for the CFM setting, the k-th user and k'-th item rating is modeled as

$$[\widehat{\boldsymbol{A}}]_{k,k'} = w_0 + [\boldsymbol{w}_0]_k + [\boldsymbol{w}_0]_{|U|+k'} + [\boldsymbol{M}]_{k,k'}.$$

Lemma 1. *[19] For any non-zero matrix $\boldsymbol{M} \in \mathbb{R}^{d\times n}$ and η:*

$$\|\boldsymbol{M}\|_{\text{tr}} \leq \frac{\eta}{2},$$

iff $\exists$ symmetric matrices $\boldsymbol{G} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{H} \in \mathbb{R}^{n\times n}$

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{G} & \boldsymbol{M} \\ \boldsymbol{M}^\top & \boldsymbol{H} \end{bmatrix} \succeq 0,\ \text{tr}(\boldsymbol{G}) + \text{tr}(\boldsymbol{H}) = \eta.$$

Based on Lemma 1, the optimization problem Eq. (2) is equivalent to

$$\min_{\boldsymbol{w},\boldsymbol{M}} \widetilde{J}(\boldsymbol{w},\boldsymbol{M}) + \lambda_1\|\boldsymbol{w}\|_2^2 \quad \text{s.t. } \|\boldsymbol{M}\|_{\text{tr}} \leq \frac{\eta}{2}, \tag{4}$$

where

$$\widetilde{J}(\boldsymbol{w},\boldsymbol{M}) := \sum_{(k,k')\in\Omega}([\boldsymbol{A}]_{k,k'} - w_0 - [\boldsymbol{w}_0]_k - [\boldsymbol{w}_0]_{|U|+k'} - [\boldsymbol{M}]_{k,k'})^2,$$

and Ω is the set of observed values in $\boldsymbol{A}$. If we set $\boldsymbol{w} = \boldsymbol{0}$, the optimization problem is equivalent to the matrix factorization with nuclear norm regularization [19]; CFM includes convex matrix factorization as a special case. Since we would like to have a low-rank matrix $\boldsymbol{M}$ of the user-item matrix $\boldsymbol{A}$ for recommendation, Eq. (2) is a natural formulation for convex FMs. Note that even though CFM resembles the matrix factorization [19]. the MF method cannot incorporate side information, while CFM can deal with side-information by concatenating it to vector $\boldsymbol{x}$. That is, intrinsically, the MF method [19] and CFM are different.

2.4 Hazan's Algorithm

For optimizing $\boldsymbol{W}$, we adopt Hazan's algorithm [13]. It only needs to compute a leading eigenvector of a sparse $d \times d$ matrix in each iteration, and thus it scales well to large problems. Moreover, the proposed CFM update formula is extremely simple, and hence useful for practitioners. The Hazan's algorithm for CFM is summarized in Algorithm 1.

Derivative computation: The objective function $J(\boldsymbol{W})$ can be equivalently written as

$$J(\boldsymbol{W}) = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}(y_i - \boldsymbol{f}_Q(\boldsymbol{x}_i;\boldsymbol{W}))(y_j - \boldsymbol{f}_Q(\boldsymbol{x}_j;\boldsymbol{W})).$$

Then, $\nabla J(\boldsymbol{W}^{(t)})$ is given as

$$\nabla J(\boldsymbol{W}^{(t)}) = \boldsymbol{X}\boldsymbol{D}^{(t)}\boldsymbol{X}^\top - \mathrm{diag}\left((\boldsymbol{X}\circ\boldsymbol{X})\boldsymbol{D}^{(t)}\boldsymbol{1}\right),$$

where

$$\boldsymbol{D}^{(t)} = -\mathrm{diag}\left(\sum_{j=1}^{n} C_{1j}(y_j - f_Q(\boldsymbol{x}_1;\boldsymbol{W})), \ldots, \sum_{j=1}^{n} C_{nj}(y_j - f_Q(\boldsymbol{x}_n;\boldsymbol{W}))\right).$$

Here, we use $\frac{\partial \mathrm{tr}(\boldsymbol{W}\boldsymbol{x}\boldsymbol{x}^\top)}{\partial \boldsymbol{W}} = \boldsymbol{x}\boldsymbol{x}^\top$. Since the derivative is written as $\boldsymbol{X}\boldsymbol{D}^{(t)}\boldsymbol{X}^\top$, the eigenvalue decomposition can be obtained without storing $\nabla J(\boldsymbol{W}^{(t)})$ in memory. Moreover, since the matrix $\boldsymbol{X}$ is a sparse matrix, we can efficiently obtain the leading eigenvector by the Lanczos method. We can use a standard eigenvalue decomposition package to compute the approximate eigenvector by the "approxEV" function. For example in Matlab, we can obtain the approximate eigenvector $\boldsymbol{p}$ by the function $[\boldsymbol{p}, l] = \mathrm{eigs}(-\nabla J(\boldsymbol{W}^{(t)}), 1, 'LA', \mathrm{Options.tol}=\frac{C_f}{(t+1)^2})$, where l is the corresponding eigenvalue.

The proposed CFM optimization requires a matrix inversion (i.e., $O(n^3)$) for computing $\boldsymbol{C}$ in $\boldsymbol{D}^{(t)}$, and it is not feasible if the dimensionality d is large. For example in user-item recommendation task, the total dimensionality of the input can be *the number of users + the number of items*. In such cases, the dimensionality can be 10^6 or more. However, fortunately, the input matrix $\boldsymbol{X}$ is extremely sparse, and we can efficiently compute $\boldsymbol{D}^{(t)}$ by using a conjugate gradient method whose time complexity is $O(n)$.

$\boldsymbol{D}^{(t)}$ can be written as

$$\boldsymbol{D}^{(t)} = \mathrm{diag}(\boldsymbol{C}\bar{\boldsymbol{y}}^{(t)}) = \mathrm{diag}((\boldsymbol{R}^\top\boldsymbol{R} + \lambda_1\boldsymbol{H}^\top\boldsymbol{H})\bar{\boldsymbol{y}}^{(t)}),$$

where $\bar{\boldsymbol{y}}^{(t)} = \boldsymbol{y} - \boldsymbol{f}_Q(\boldsymbol{X};\boldsymbol{W}^{(t)})$. Since the number of samples n tends to be larger than the dimensionality d in factorization machine settings, $\boldsymbol{Z}\boldsymbol{Z}^\top$ becomes full rank. Namely, we can safely make the regularization parameter $\lambda_1 = 0$. In such case, $\boldsymbol{D}^{(t)}$ is given as

$$\boldsymbol{D}^{(t)} = \mathrm{diag}(\bar{\boldsymbol{y}}^{(t)} - \boldsymbol{Z}^\top\widehat{\boldsymbol{w}}^{(t)}),$$

where we use $\boldsymbol{C} = \boldsymbol{R}^\top\boldsymbol{R} = \boldsymbol{I} - \boldsymbol{Z}^\top(\boldsymbol{Z}\boldsymbol{Z}^\top)^{-1}\boldsymbol{Z}$. The $\widehat{\boldsymbol{w}}^{(t)} = (\boldsymbol{Z}\boldsymbol{Z}^\top)^{-1}\boldsymbol{Z}\bar{\boldsymbol{y}}^{(t)}$ is obtained by solving

$$\boldsymbol{Z}^\top\boldsymbol{w} = \bar{\boldsymbol{y}}^{(t)}, \tag{5}$$

where $\boldsymbol{w}^{(t)}$ can be efficiently obtained by a conjugate gradient method with time complexity $O(n)$. Thus, we can compute $\boldsymbol{D}^{(t)}$ without computing the matrix inverse $(\boldsymbol{Z}\boldsymbol{Z}^\top)^{-1}$. To further speed up the conjugate gradient method, we use a preconditioner and the previous solution $\boldsymbol{w}^{(t-1)}$ as the initial solution.

Finally, we compute $\boldsymbol{D}^{(t)}$ as

$$\boldsymbol{D}^{(t)} = \mathrm{diag}(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{X};\widehat{\boldsymbol{w}}^{(t)},\widehat{\boldsymbol{W}}^{(t)})).$$

The diagonal elements of $\boldsymbol{D}^{(t)}$ are the differences between the observed outputs and the model predictions at the t-th iteration. Note that, in our CFM optimization, we eliminate $\boldsymbol{w}$ and only optimize for $\boldsymbol{W}$; however, the $\boldsymbol{w}$ is implicitly estimated in Hazan's algorithm.

Complexity: Iteration t in Algorithm 1 includes computing an approximate leading eigenvector of a sparse matrix with n non-zero elements and an estimation of $\boldsymbol{w}$, which require $O(n)$ computation using Lanczos algorithm and $O(n)$ computation using conjugate gradient descent, respectively. Thus, the entire computational complexity of the proposed method is $O(Tn)$, where T is the total number of iterations in Hazan's algorithm.

Optimal step size estimation: Hazan's algorithm assures $\boldsymbol{W}$ converges to a global optimum with using the step size $\alpha_t = 2/(2+t), t = 0, 1, \ldots, T$ [19]. However, this is in practice slow to converge. Instead, we choose the α_k that maximally decreases the objective function $J(\boldsymbol{W})$. The optimal α_k can be obtained by solving the following equation:

$$\begin{aligned}\widehat{\alpha}_t &= \operatorname*{argmin}_{\alpha} \; J(\boldsymbol{W}^{(t+1)}) \\ &= \operatorname*{argmin}_{\alpha} \left\| \boldsymbol{R}\left(\boldsymbol{y} - \boldsymbol{f}_Q(\boldsymbol{X};(1-\alpha)\boldsymbol{W}^{(t)} + \alpha\boldsymbol{u}^{(t)}\boldsymbol{u}^{(t)\top})\right)\right\|_2^2.\end{aligned}$$

Taking the derivative with respect to α and solving the problem for α, we have

$$\widehat{\alpha}_t = \frac{(\boldsymbol{y} - \boldsymbol{f}_Q(\boldsymbol{X};\boldsymbol{W}^{(t)}))^\top\boldsymbol{R}(\boldsymbol{f}_Q(\boldsymbol{X};\boldsymbol{p}^{(t)}\boldsymbol{p}^{(t)\top} - \boldsymbol{W}^{(t)}))}{\|\boldsymbol{R}(\boldsymbol{f}_Q(\boldsymbol{X};\boldsymbol{p}^{(t)}\boldsymbol{p}^{(t)\top} - \boldsymbol{W}^{(t)})\|_2^2}. \tag{6}$$

The computation of α_t involves the matrix inversion of $\boldsymbol{R}$. However, by using the same technique as in the derivative computation, we can efficiently compute α_t.

Update $\boldsymbol{W}$: When the input dimension d is large, storing the feature-feature interaction matrix $\boldsymbol{W}$ is not possible. To avoid the memory problem, we update $\boldsymbol{W}^{(t)}$ as

$$\begin{aligned}\boldsymbol{W}^{(t+1)} &= \boldsymbol{P}^{(t+1)}\mathrm{diag}(\boldsymbol{\lambda}^{(t+1)})\boldsymbol{P}^{(t+1)\top}, \\ \boldsymbol{P}^{(t+1)} &= [\boldsymbol{P}^{(t)}\ \boldsymbol{p}^{(t)}] \in \mathbb{R}^{d\times(t+1)}, \\ \lambda_k^{(t+1)} &= \begin{cases} (1-\widehat{\alpha}_t)\lambda^{(t)} & (k < t) \\ \widehat{\alpha}_t & (k = t+1) \end{cases},\end{aligned}$$

where $\boldsymbol{\lambda}^{(t+1)} \in \mathbb{R}^{t+1}$. Thus, we only need to store $\boldsymbol{P}^{(t+1)} \in \mathbb{R}^{d\times(t+1)}$ and $\boldsymbol{\lambda}^{(t+1)}$ at the $(t+1)$-th iteration. In practice, Hazan's algorithm converges with $t = 100$ (see experiment section), so the required memory for Hazan's algorithm is reasonable.

Prediction: Let us define $\boldsymbol{U} = \boldsymbol{P}\mathrm{diag}(\boldsymbol{\lambda})^{1/2} \in \mathbb{R}^{d\times t}$ such that $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{U}^\top$. Then, we can efficiently compute the output as

$$f(\boldsymbol{x};\boldsymbol{w},\boldsymbol{W}) = \boldsymbol{w}^\top\boldsymbol{z} + \frac{1}{2}\left(\|\boldsymbol{U}^\top\boldsymbol{x}\|_2^2 - (\boldsymbol{x}\circ\boldsymbol{x})^\top(\boldsymbol{U}\circ\boldsymbol{U})\boldsymbol{1}\right).$$

The time complexities of the terms are $O(d)$, $O(d(t+1))$, and $O(d)$, respectively.

2.5 Tensor completion with CFM

In this section, we formulate a Tucker-based tensor completion problem [43, 44] as a CFM problem.

Let us denote the input 3-way tensor as $\mathcal{Y} \in \mathbb{R}^{n_1\times n_2\times n_3}$, where n_1, n_2, and n_3 are the number of samples in each mode. In this paper,

we consider the following regularization-based learning model:

$$\min_{\{\mathcal{M}^{(m)}\}_{m=0}^{3}} \sum_{(i,j,k)\in\Omega} \left([\mathcal{Y}]_{i,j,k} - [\mathcal{M}^{(0)}]_{i,j,k} - \sum_{m=1}^{3} [\mathcal{M}^{(m)}]_{i,j,k} \right)^2 + \lambda \sum_{m=1}^{3} \|\boldsymbol{M}_{(m)}^{(m)}\|_{\mathrm{tr}}, \tag{7}$$

where $\mathcal{M}^{(0)} \in \mathbb{R}^{n_1\times n_2\times n_3}$ is the bias tensor, $\mathcal{M}^{(m)} \in \mathbb{R}^{n_1\times n_2\times n_3}$ is the m-th mode tensor, $\lambda \geq 0$ is the regularization parameter, $\boldsymbol{M}_{(m)}^{(m)}$ is the unfolded matrix with respect to the m-th mode, $\boldsymbol{M}_{(1)}^{(1)} \in \mathbb{R}^{n_1\times n_2 n_3}$, $\boldsymbol{M}_{(2)}^{(2)} \in \mathbb{R}^{n_2\times n_1 n_3}$, and $\boldsymbol{M}_{(3)}^{(3)} \in \mathbb{R}^{n_3\times n_1 n_2}$. The final goal here is to learn $\mathcal{M}$ from $\mathcal{Y}$ by minimizing $J(\mathcal{M})$. Now, we reformulate Eq. (7) by CFM.

Let us define the pooled matrix:

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{G}_1 & \boldsymbol{M}_{(1)}^{(1)} & & & & & \\ \boldsymbol{M}_{(1)}^{(1)\top} & \boldsymbol{H}_1 & & & & & \\ & & \boldsymbol{G}_2 & \boldsymbol{M}_{(2)}^{(2)} & & & \\ & & \boldsymbol{M}_{(2)}^{(2)\top} & \boldsymbol{H}_2 & & & \\ & & & & \boldsymbol{G}_3 & \boldsymbol{M}_{(3)}^{(3)} \\ & & & & \boldsymbol{M}_{(3)}^{(3)\top} & \boldsymbol{H}_3 \end{bmatrix} \in \mathbb{R}^{d\times d},$$

where $\boldsymbol{W} \geq 0$, $d = \sum_{m=1}^{3} n_m + \sum_{m=1}^{3}\sum_{m'=m+1}^{3} n_m n_{m'}$. Note that the off-diagonal matrices are not important for deriving optimization algorithm, and thus, we omit them here. Moreover, since the matrix $\boldsymbol{W}$ is a positive semi-definite matrix, we can decompose it as

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{U}_1 \\ \boldsymbol{V}_1 \\ \vdots \\ \boldsymbol{U}_3 \\ \boldsymbol{V}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1^\top & \boldsymbol{V}_1^\top & \dots & \boldsymbol{U}_3^\top & \boldsymbol{V}_3^\top \end{bmatrix}.$$

Lemma 2. *[8] For a 3-way tensor case, we have:*

$$[\mathcal{M}^{(1)}]_{i,j,k} = [\boldsymbol{M}_{(1)}^{(1)}]_{i,n_2(k-1)+j},$$
$$[\mathcal{M}^{(2)}]_{i,j,k} = [\boldsymbol{M}_{(2)}^{(2)}]_{j,n_3(i-1)+k},$$
$$[\mathcal{M}^{(3)}]_{i,j,k} = [\boldsymbol{M}_{(3)}^{(3)}]_{k,n_1(j-1)+i}.$$

Then, we can rewrite $\sum_{m=1}^{3}[\mathcal{M}^{(m)}]_{i,j,k}$ as

$$\sum_{m=1}^{3}[\mathcal{M}^{(m)}]_{i,j,k} = \frac{1}{2}\mathrm{tr}\left(\boldsymbol{W}(\boldsymbol{x}_{i,j,k}\boldsymbol{x}_{i,j,k}^\top - \mathrm{diag}(\boldsymbol{x}_{i,j,k}\circ\boldsymbol{x}_{i,j,k}))\right),$$
$$\boldsymbol{x}_{i,j,k}^{(1)} = [\overbrace{0\cdots0\underbrace{1}_{i}0\cdots0}^{n_1}\overbrace{0\cdots0\underbrace{1}_{n_2(k-1)+j}0\cdots0}^{n_2 n_3}]^\top,$$
$$\boldsymbol{x}_{i,j,k}^{(2)} = [\overbrace{0\cdots0\underbrace{1}_{j}0\cdots0}^{n_2}\overbrace{0\cdots0\underbrace{1}_{n_3(i-1)+k}0\cdots0}^{n_1 n_3}]^\top,$$
$$\boldsymbol{x}_{i,j,k}^{(3)} = [\overbrace{0\cdots0\underbrace{1}_{k}0\cdots0}^{n_3}\overbrace{0\cdots0\underbrace{1}_{n_1(j-1)+i}0\cdots0}^{n_1 n_2}]^\top,$$
$$\boldsymbol{x}_{i,j,k} = [\boldsymbol{x}_{i,j,k}^{(1)\top}\ \boldsymbol{x}_{i,j,k}^{(2)\top}\ \boldsymbol{x}_{i,j,k}^{(3)\top}]^\top \in \mathbb{R}^d.$$

where $\boldsymbol{x}_{i,j,k}^{(1)} \in \mathbb{R}^{n_1+n_2 n_3}$, $\boldsymbol{x}_{i,j,k}^{(2)} \in \mathbb{R}^{n_2+n_1 n_3}$, and $\boldsymbol{x}_{i,j,k}^{(3)} \in \mathbb{R}^{n_3+n_1 n_2}$.

For the bias tensor $\mathcal{M}^{(0)}$, we parametrize it as

$$[\mathcal{M}^{(0)}]_{i,j,k} = \boldsymbol{w}^\top\boldsymbol{x}_{i,j,k} = \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{0}_{n_2 n_3} \\ \boldsymbol{w}_2 \\ \boldsymbol{0}_{n_1 n_3} \\ \boldsymbol{w}_3 \\ \boldsymbol{0}_{n_1 n_2} \end{bmatrix}^\top \boldsymbol{x}_{i,j,k}, \tag{8}$$

where $\boldsymbol{w} \in \mathbb{R}^d$, $\boldsymbol{w}_1 \in \mathbb{R}^{n_1}$, $\boldsymbol{w}_2 \in \mathbb{R}^{n_2}$, and $\boldsymbol{w}_3 \in \mathbb{R}^{n_3}$. Note we use this parameterization, since the number of dimension d can be much bigger than the number of non-zero elements n and it is hard to solve Eq. (5).

Lemma 3. *For the matrices $\boldsymbol{M}_{(1)}^{(1)} \in \mathbb{R}^{n_1\times n_2 n_3}, \boldsymbol{M}_{(2)}^{(2)} \in \mathbb{R}^{n_2\times n_1 n_3}, \boldsymbol{M}_{(3)}^{(3)} \in \mathbb{R}^{n_3\times n_1 n_2}$ and η:*

$$\sum_{m=1}^{3}\|\boldsymbol{M}_{(m)}^{(m)}\|_{\mathrm{tr}} \leq \frac{\eta}{2}$$

iff $\boldsymbol{W} \geq 0$ and $\sum_{m=1}^{3}\mathrm{tr}(\boldsymbol{G}_m) + \mathrm{tr}(\boldsymbol{H}_m) = \eta$.
Proof: See Appendix.

Based on the Lemma 3, we can rewrite the optimization problem as

$$\min_{\boldsymbol{w},\boldsymbol{W}} \sum_{(i,j,k)\in\Omega}\Big([\mathcal{Y}]_{i,j,k} - \boldsymbol{w}^\top\boldsymbol{x}_{i,j,k} - \mathrm{tr}\left(\boldsymbol{W}(\boldsymbol{x}_{i,j,k}\boldsymbol{x}_{i,j,k}^\top - \mathrm{diag}(\boldsymbol{x}_{i,j,k}\circ\boldsymbol{x}_{i,j,k}))\right)\Big)^2$$
$$\text{s.t } \boldsymbol{W} \geq 0, \mathrm{tr}(\boldsymbol{W}) = \eta.$$

Since this is a CFM problem, we can efficiently solve it with Hazan's algorithm.

3 RELATED WORK

The same problem setting as in our work has been addressed recently [3] in a lined work (technical report [53]). The key difference between the proposed method and [3] is that our approach is based on a single convex optimization problem for the interaction term $\boldsymbol{W}$. The approach [3] uses a block-coordinate descent (BCD) algorithm for optimization, optimizing the linear and quadratic terms alternatively. That is, they alternatingly solve the following two update equations until convergence:

$$\widehat{\boldsymbol{w}}^{(t+1)} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \ \|\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{X}; \boldsymbol{w}, \boldsymbol{W}^{(t)})\|_2^2 + \lambda_1 \|\boldsymbol{w}\|_2^2,$$

$$\widehat{\boldsymbol{W}}^{(t+1)} = \underset{\boldsymbol{W}}{\operatorname{argmin}} \ \|\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{X}; \widehat{\boldsymbol{w}}^{(t+1)}, \boldsymbol{W})\|_2^2 + \lambda_2 \|\boldsymbol{W}\|_{\mathrm{tr}},$$

while our proposed approach is simply given as

$$\widehat{\boldsymbol{W}} = \underset{\boldsymbol{W} \succeq 0}{\operatorname{argmin}} \ \|\boldsymbol{R}(\boldsymbol{y} - \boldsymbol{f}_Q(\boldsymbol{X}; \boldsymbol{W}))\|_2^2, \text{s.t.} \ \|\boldsymbol{W}\|_{\mathrm{tr}} = \eta.$$

Hence, the BCD algorithm needs to iterate the sub-problem for $\boldsymbol{W}$ until convergence for obtaining the globally optimal solution.

If an $O(n)$ algorithm is analyzed for the trace norm minimization in BCD, then the entire complexity is $O(T'Tn)$ where T' and T are the number of BCD iterations and the number of the iterations in sub-problem. On the other hand, our algorithm's complexity is $O(Tn)$. Another difference is that our optimization approach includes the matrix factorization with nuclear norm regularization as a special case, while it is unclear whether the same holds for the formulation [3]. Finally, our CFM approach is very easy to implement; the core part of the proposed algorithm can be written within 20 lines in Matlab. Note also that the BCD based approach is more general than our CFM framework; it can be used for other loss functions such as logistic loss and it does not require the positive definiteness condition for $\boldsymbol{W}$.

The convex variant of matrix factorization has been widely studied in the machine learning community [2, 6, 10, 20, 28, 42, 43, 45]. The key idea of the convex approach is to use the trace norm regularizer, and the optimization problem is given as

$$\widehat{\boldsymbol{M}} = \underset{\boldsymbol{M}}{\operatorname{argmin}} \ \|\mathcal{P}_\Omega(\boldsymbol{A}) - \mathcal{P}_\Omega(\boldsymbol{M})\|_F^2 + \lambda \|\boldsymbol{M}\|_{\mathrm{tr}}, \tag{9}$$

where Ω is the set of observed values in $\boldsymbol{A}$, $[\mathcal{P}_\Omega(\boldsymbol{A})]_{i,j} = [\boldsymbol{A}]_{i,j}$ if $i, j \in \Omega$ and 0 otherwise, and $\|\cdot\|_F$ is the Frobenius norm. Since Eq.(9) and Eq.(4) are equivalent when $\boldsymbol{w} = \boldsymbol{0}$, the convex matrix factorization can be regarded as a special case of CFM.

To optimize Eq. (9), the singular value thresholding (SVT) method has been proposed [5, 30], where SVT converges faster in $O(\frac{1}{\sqrt{\epsilon}})$ (ϵ is an approximate error). However, the SVT approach requires solving the full eigenvalue decomposition, which is computationally expensive for large datasets. To deal with large data, Frank-Wolfe based approaches have been proposed including Hazan's algorithm [19], corrective refitting [38], and active subspace selection [16]. However, these approaches cannot incorporate user and item bias. Furthermore, it is not straightforward to incorporate side information to deal with *cold start* problems (i.e., recommending an item to a user who has no click information).

To handle *cold start* problems, collective matrix factorization (collective MF) has been proposed [40]. The key idea of collective MF is to incorporate side information into matrix factorization. More specifically, we prepare a user × user meta matrix (e.g., gender, age, etc.) and an item × item meta matrix (item category, item title, etc) in addition to a user-item matrix. Then, we factorize all the matrices together. A convex variant of CMF called convex collective matrix factorization (CCMF) has been proposed [4]. CCMF employs the convex collective norm, which is a generalization of the trace norm to several matrices. Recently, Hazan's algorithm was introduced to CCMF [12]. More importantly, it has been theoretically justified that CCMF can give better performance in cold start settings. Since FMs can incorporate side information, FMs and CCMF are closely related. Actually, CFM can utilize side information and can learn the user and item bias term together; it can be regarded as a generalized variant of CCMF.

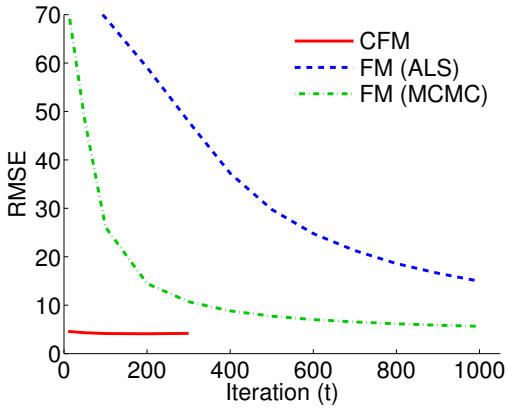


Figure 1: Convergence of the methods: test RMSE of the synthetic experiment. CFM is the proposed convex factorization machine, FM (ALS) is the factorization machine with ALS optimization, and FM (MCMC) is the factorization machine with MCMC optimization. The proposed CFM gets the lowest RMSE values with a small number of iterations, while FMs need many iterations to obtain reasonable performance.

4 EXPERIMENTS

In this section, we first illustrate the proposed CFM using simple synthetic data, and then, we validate CFM with Movielens data (single matrix), which is a standard recommendation dataset. Finally, we apply the proposed algorithm for toxicogenomics prediction task (two-view tensors).

We compare CFM with ridge regression, FM (SGD), FM (MCMC) and FM (ALS), where FM (MCMC) is a state-of-the-art FM optimization method. The ridge regression corresponds to the factorization machine with only the linear term $f(\boldsymbol{x}) = w_0 + \boldsymbol{w}_0^\top \boldsymbol{x}$, which is also a strong baseline. To estimate FM models, we use the publicly available libFM package[3]. For all experiments, the number of latent dimensions of FMs is set to 20, which performs well in practice. For FM (ALS), we experimentally set the regularization parameters as $\lambda_1 = 0$ and $\lambda_2 = 0.01$. The initial matrices $\boldsymbol{W}$ (for CFM) and $\boldsymbol{G}$ (for FMs) are randomly set (this is the default setting of the libFM

[3] http://www.libfm.org

package). For CFM, we implemented the algorithm with Matlab. We experimentally set $C_f = 1$, which works for our experiments. For all experiments, we use a server with 16 core 1.6GHz CPU and 24G memory.

When evaluating the performance of CFM and FMs, we use the root mean squared error (RMSE):

$$\sqrt{\frac{1}{n_{\text{test}}}\sum_{i=1}^{n_{\text{test}}}(y_i^* - \widehat{y}_i)^2},$$

where y^* and $\widehat{y}$ are the true and estimated target values, respectively.

4.1 Synthetic Experiment

First, we illustrate how the proposed CFM behaves using a synthetic dataset.

In this experiment, we randomly generate input vectors $\boldsymbol{x} \in \mathbb{R}^{100}$ as $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and output values as

$$y = \widetilde{w}_0 + \widetilde{\boldsymbol{w}}^\top \boldsymbol{x} + \sum_{\ell=1}^{d}\sum_{\ell'=\ell+1}^{d}[\widetilde{\boldsymbol{W}}]_{\ell,\ell'}x_\ell x_{\ell'},$$

where $\widetilde{w}_0 \sim \mathcal{N}(0,1)$, $\widetilde{\boldsymbol{w}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, $\widetilde{\boldsymbol{W}}_{\ell,\ell'} \sim$ Uniform([0 1]).

We use 900 samples for training and 100 samples for testing. We run the experiments 5 times with randomly selecting training and test samples and report the average RMSE scores. Figure 1 shows the test RMSE for CFM and FMs. As can be seen, the proposed CFM gets the lowest RMSE values with a small number of iterations, while FMs need many iterations to obtain reasonable performance.

4.2 Recommendation Experiments

Next, we evaluate our proposed method on the Movielens 100K, 1M, 10M, and 20M datasets [31]. In these experiments, we randomly split the observations into 75% for training and 25% for testing. We run the recommendation experiments on three random splits, which is the same experimental setting as in [3], and report the average RMSE score.

For CFM, the regularization parameter η is experimentally set to 2000 (for 100K), 4000 (for 1M), 20000 (for 10M), and 40000 (for 20M), respectively. For FMs, the rank is set to 20, which gives overall good performance. To investigate the effect of the initialization parameter, we initialize FM (MCMC) with two parameters stdev = 0.05 and stdev = 0.1, which are the standard deviation of the random variable for initializing $\boldsymbol{G}$. We also report the RMSE of the CFM method of [3] for reference. Note that it may be possible to improve results by extensive hyperparameter searches, but it will naturally add to computation time.

Figure 2 shows the training and test RMSE with the CFM (optimal step size) and the CFM ($\alpha_t = \frac{2}{2+t}$) for the Movielens datasets. For both methods, the RMSE of training and test is converging with a small number of iterations. Overall, the optimal step size based approach converges faster than the one based on $\alpha_t = \frac{2}{2+t}$. Figure 3 shows the RMSE over computational time (seconds). For large datasets, the CFM achieves reasonable performance in less than an hour. In Table 1, we show the RMSE comparison of the proposed CFM with FMs. As we expected, CFM compares favorably with FM (SGD) and FM (ALS), since FM (SGD) and FM (ALS) can be easily trapped at poor locally optimal solutions. Moreover, our CFM method compares favorably with also the CFM (BCD) [3]. On the other hand, FM (MCMC) can obtain better performance than CFMs (both our formulation and [3]) for these datasets if we set an appropriate initialization parameter. This is because MCMC tends to avoid poor locally optimal solution if we run the sampler long enough. That is, since the objective function of FMs is non-convex and it has more flexibility than the convex formulation, it can converge to a better solution than CFM if we initialize FMs well.

4.3 Prediction in Toxicogenomics

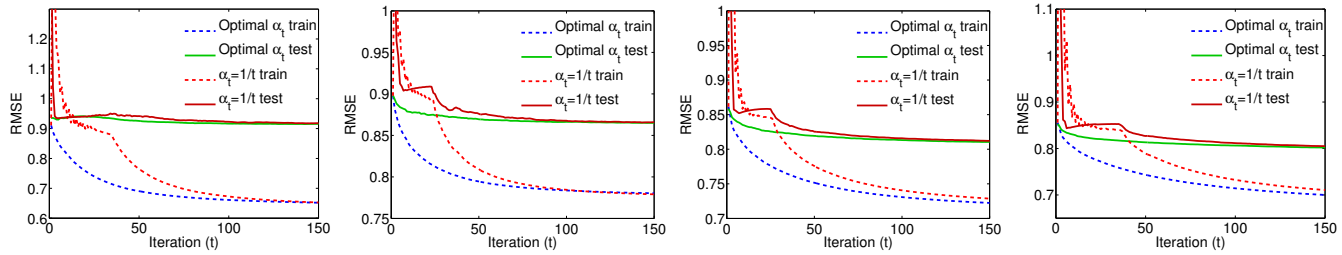
Next, we evaluated our proposed method on a toxicogenomics dataset [21]. The dataset contains three sets of matrices representing gene expression and toxicity responses of a set of drugs. The first set *Gene Expression*, represents the differential expression of 1106 genes in three different cancer types, to a collection of 78 drugs (i.e., $\boldsymbol{A}_l^{(1)} \in \mathbb{R}^{1106\times 78}, l = 1,2,3$). The second set, *Toxicity*, contains three dose-dependent toxicity profiles of the same 78 drugs over the three cancers (i.e., $\boldsymbol{A}_l^{(2)} \in \mathbb{R}^{3\times 78}, l = 1,2,3$). The gene expression data of the three cancers (Blood, Breast and Prostate) come from the Connectivity Map [26] and were pre-processed to obtain differential expression of treatment vs control. As a result, the expression scores represent positive or negative regulation with respect to the untreated level. The toxicity screening data, from the NCI-60 database [39], summarizes the toxicity of drug treatments in three variables GI50, LC50, and TGI, representing the 50% growth inhibition, 50% lethal concentration, and total growth inhibition levels. The data were conformed to represent dose-dependent toxicity profiles for the doses used in the corresponding gene expression dataset.

Various approaches have been used to study toxicogenomics and the impact of drugs on cells, including analysis of drug side effects using similarity based approach [41], kernel methods for predicting drug-targets [17], and factor analysis for modeling dependencies between drug structures and their gene expression responses [23]. These approaches, while interesting in their own right, do not model the dependencies between toxicity profiles and the corresponding post-treatment gene expression.

Predicting both gene and toxicity matrices: We compared our proposed method with existing state-of-the-art methods. In this experiment, we randomly split the observations into 50% for training (129, 466 elements) and 50% for testing (129, 465 elements), which is the exactly same datasets used in [21]. We run the prediction experiments on 100 random splits [21], and report the average *relative MSE* score, which is defined as

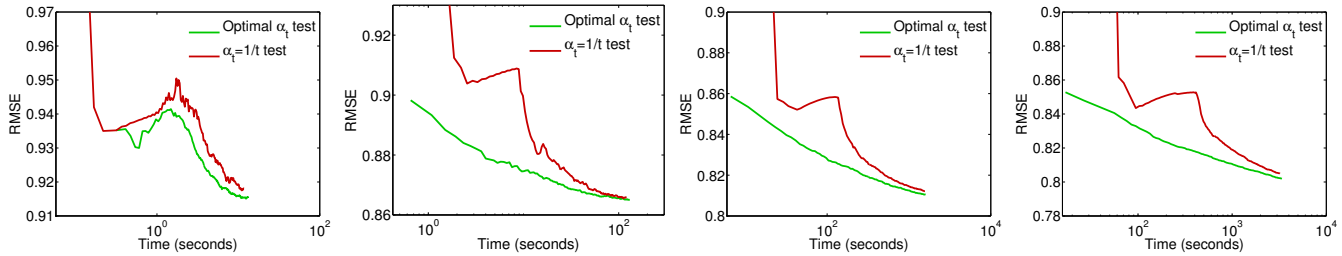
$$\frac{1}{V}\cdot\sum_{v=1}^{V}\frac{\|\boldsymbol{y}^{*,v} - \widehat{\boldsymbol{y}}^v\|_2^2}{\|\boldsymbol{y}^{*,v} - \bar{y}^{*,v}\boldsymbol{1}\|_2^2},$$

where $\boldsymbol{y}^{*,v}$ is the target score vector, $\widehat{\boldsymbol{y}}^v$ is the estimated score vector, and $\bar{y}^{*,v}$ is the mean of elements in $\boldsymbol{y}^{*,v}$, V is the number of views. In this experiment, the number of views is $V = 2$. Since the number of elements in view 1 and view 2 are different, the *relative* MSE score is more suitable than the root MSE score. We compared our proposed method with ARDCP [32], CP [7], Group Factor Analysis (GFA) [49], and Bayesian Multi-view Tensor Factorization



(a) Movielens 100K data. (b) Movielens 1M data. (c) Movielens 10M data. (d) Movielens 20M data.

Figure 2: RMSE over iteration (t). $\alpha_t = \frac{1}{t}$ train and $\alpha_t = \frac{1}{t}$ test are training and test RMSE with using $\frac{2}{2+t}$ stepsize. Optimal α_t train and α_t test are training and test RMSE with using the optimal stepsize Eq. (6). Overall, the optimal step size based approach converges faster than the one based on $\alpha_t = \frac{2}{2+t}$.



(a) Movielens 100K data. (b) Movielens 1M data. (c) Movielens 10M data. (d) Movielens 20M data.

Figure 3: RMSE over time (seconds). $\alpha_t = \frac{1}{t}$ test is the test RMSE with using $\frac{2}{2+t}$ stepsize. Optimal α_t test is the test RMSE with using optimal stepsize. Overall, the optimal step size based approach converges faster than the one based on $\alpha_t = \frac{2}{2+t}$.

Table 1: Test RMSE of CFM, CFM (BCD), FMs, and ridge regression for Movielens data sets. The proposed CFM compares favorably with CFM (BCD), FM_{SGD}, FM_{ALS}, and $\text{FM}^{(0.05)}_{\text{MCMC}}$. FM (MCMC) can obtain better performance than CFMs (both our formulation and [3]) for these datasets if we set an appropriate initialization parameter.

Dataset	CFM	CFM (BCD)	FM_{SGD}	FM_{ALS}	$\text{FM}^{(0.05)}_{\text{MCMC}}$	$\text{FM}^{(0.1)}_{\text{MCMC}}$	Ridge
100K	0.915	0.93	1.078	1.242	0.905	0.901	0.936
1M	0.866	0.85	0.943	0.981	0.877	0.846	0.899
10M	0.810	0.82	0.827	0.873	0.831	0.778	0.855
20M	0.802	n/a	0.821	0.852	0.803	0.768	0.850

(BMTF) [21]. BMTF is a state-of-the-art multi-view factorization method.

For CFM, we first concatenate all view matrices as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1^{(1)} & \boldsymbol{A}_1^{(2)} \\ \boldsymbol{A}_2^{(1)} & \boldsymbol{A}_2^{(2)} \\ \boldsymbol{A}_3^{(1)} & \boldsymbol{A}_3^{(2)} \end{bmatrix} \in \mathbb{R}^{3327 \times 78},$$

and use this matrix for learning. The regularization parameter η is experimentally set to 1000. To deal with multi-view data, we form the input and output of CFM as

$$\boldsymbol{x} = [0 \cdots 0 \overbrace{\underbrace{1}_{i\text{-th gene}} 0 \cdots 0}^{3327} \overbrace{0 \cdots 0 \underbrace{1}_{j\text{-th drug}} 0 \cdots 0}^{78} \overbrace{\underbrace{1}_{\text{1st view}} 0}^{2}]^\top,$$

$$y = [\boldsymbol{A}]_{i,j},$$

where $\boldsymbol{x} \in \mathbb{R}^{3407}$

Table 2 shows the average relative MSE of the methods. As can be seen, the proposed method outperforms the state-of-the-art methods.

Predicting toxicity matrices using Gene expression data: We further evaluated the proposed CFM on the toxicity prediction task. For this experiment, we randomly split the observations of the toxicity matrices into 50% for training (341 elements) and 50% for testing (341 elements). Then, we used the gene expression matrices $\boldsymbol{A}_1^{(1)}, \boldsymbol{A}_2^{(1)}, \boldsymbol{A}_3^{(1)}$ as side information for predicting the toxicity matrices. More specifically, we designed two types of features from the gene expression data:

- **Mean of m-nearest neighbor similarities (x_{mean}):** We first find the m-nearest neighbors of the i-th drug target,

Table 2: Test relative MSE of both gene expression and toxicogenomics data. We compared our proposed method with ARDCP [32], CP [7], Group Factor Analysis (GFA) [49], and Bayesian Multi-view Tensor Factorization (BMTF) [21]. In the multi-view algorithms, we used $A^{(1)}$ and $A^{(2)}$ for factorization, while we used only $A^{(2)}$ for the single-view algorithms. The proposed CFM outperforms the state-of-the-art methods.

	Multi-view				Single-view		
	CFM	BMTF	GFA	ARDCP	CP	ARDCP	CP
Mean	**0.4037**	0.4811	0.5223	0.8919	5.3713	0.6438	5.0699
StdError	**0.0163**	0.0061	0.0041	0.0027	0.0310	0.0047	0.0282

where the Gaussian kernel is used for similarity computation. Then, we average the similarity of $1, \ldots, m$-th nearest neighbors.

- **Standard deviation of m-nearest neighbor similarities ($x_{\mathbf{std}}$):** Similarly to the mean feature, we first found m-nearest neighbor similarities and then computed their standard deviation.

Then, we used these features as

$$\boldsymbol{x} = [\overbrace{0 \cdots 0 \underbrace{1}_{i\text{-th drug}} 0 \cdots 0}^{78} \overbrace{0 \underbrace{1}_{k\text{-th sensit.}} 0}^{3} \overbrace{0 \underbrace{1}_{l\text{-th cancer}} 0}^{3} \overbrace{x_{\text{mean}}\ x_{\text{std}}}^{2}]^\top,$$

$$y = [\boldsymbol{A}_l^{(2)}]_{i,k},$$

where $\boldsymbol{x} \in \mathbb{R}^{86}$.

We run the prediction experiments on 100 random splits, and report the average RMSE score (Table 3). 'CFM' is 'CFM without any additional features. It is clear that the performance of CFM improves by simply adding manually designed features. Thus, we can improve the prediction performance of CFM by designing new features, and it is useful for various prediction tasks in biology data.

5 CONCLUSION

We proposed the *convex factorization machine* (CFM), which is a convex variant of factorization machines (FMs). Specifically, we formulated the CFM optimization problem as a semidefinite program (SDP) and solved it with Hazan's algorithm. A key advantage of the proposed method over FMs is that CFM can find a globally optimal solution, while FMs can get poor locally optimal solutions since they are non-convex approaches. The derived algorithm is simple and can be easily implemented. The key difference between the proposed method and another CFM approach [3] is that our approach is based on a single convex optimization problem for the interaction term $\boldsymbol{W}$, while the approach [3] uses a block-coordinate descent (BCD) algorithm for optimization, optimizing the linear and quadratic terms alternatively. Moreover, we applied CFM to toxicogenomics prediction task. Finally, we also showed the connections between CFM and convex matrix factorization methods and CFM and convex tensor completion methods which were not addressed in [3]. Through synthetic and real-world experiments, we showed that the proposed CFM achieves results competitive with state-of-the-art methods. Moreover, for a toxicogenomics prediction task, CFM outperformed a state-of-the-art multi-view tensor factorization method.

There are several interesting future opportunities. First, since the proposed CFM is formulated as a Frank-Wolfe algorithm, we can easily accelerate CFM by using recent results such as [29, 47]. Moreover, employing other optimization techniques such as stochastic subgradient descent [1, 34] for solving trace norm regularization would be an interesting direction. Second, we showed the connection of Tucker decomposition and factorization machines. That is, it may be possible to solve other types of tensor decomposition with factorization machine framework. Finally, applying the proposed algorithm to other biology related tasks is also an interesting direction of future work.

ACKNOWLEDGMENT

MY was supported by the JST PRESTO program JPMJPR165A and partly supported by MEXT KAKENHI 16K16114. SK was supported by the Academy of Finland (Finnish Center of Excellence in Computational Inference Research COIN and grants 292334, 294238, 295503). SAK was supported by Academy of Finland (296516). HM was supported by FiDiPro, Tekes KAKENHI, and KAKENHI 16H02868.

APPENDIX

Proof of Lemma 3: This is a variation of the Lemma1 of [19]. From the characterization:

$$\sum_{m=1}^{3} \|\boldsymbol{M}_{(m)}^{(m)}\|_{\text{tr}} = \min_{\{\boldsymbol{U}_m \boldsymbol{V}_m^\top = \boldsymbol{M}_{(m)}^{(m)}\}_{m=1}^3} \quad \frac{1}{2} \sum_{m=1}^{3} (\|\boldsymbol{U}_m\|_F + \|\boldsymbol{V}_m\|_F)$$

we have that $\exists\, \boldsymbol{U}_m, \boldsymbol{V}_m, \boldsymbol{U}_m \boldsymbol{V}_m^\top = \boldsymbol{M}_{(m)}^{(m)}, m = 1, 2, 3$ s.t.

$$\begin{aligned} 2 \sum_{m=1}^{3} \|\boldsymbol{M}_{(m)}^{(m)}\|_{\text{tr}} &= \sum_{m=1}^{3} \|\boldsymbol{U}_m\|_F^2 + \|\boldsymbol{V}_m\|_F^2 \\ &= \sum_{m=1}^{3} \text{tr}(\boldsymbol{U}_m \boldsymbol{U}_m^\top) + \text{tr}(\boldsymbol{V}_m \boldsymbol{V}_m^\top) \leq \eta. \end{aligned}$$

That is, we have

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{U}_1 \boldsymbol{U}_1^\top & \boldsymbol{M}_{(1)}^{(1)} & & & & & \\ \boldsymbol{M}_{(1)}^{(1)\top} & \boldsymbol{V}_1 \boldsymbol{V}_1^\top & & & & & \\ & & \boldsymbol{U}_2 \boldsymbol{U}_2^\top & \boldsymbol{M}_{(2)}^{(2)} & & & \\ & & \boldsymbol{M}_{(2)}^{(2)\top} & \boldsymbol{V}_2 \boldsymbol{V}_2^\top & & & \\ & & & & \boldsymbol{U}_3 \boldsymbol{U}_3^\top & \boldsymbol{M}_{(3)}^{(3)} & \\ & & & & \boldsymbol{M}_{(3)}^{(3)\top} & \boldsymbol{V}_3 \boldsymbol{V}_3^\top & \end{bmatrix},$$

Table 3: Test relative MSE on toxicogenomics prediction task. In this task, we used the gene expression matrices $A_1^{(1)}, A_2^{(1)}, A_3^{(1)}$ as side information for predicting the toxicity matrices. More specifically, we designed two types of features from the gene expression data, where m is the tuning parameter for the features. We can improve the prediction performance of CFM by designing new features, and it is useful for various prediction tasks in biology data.

	CFM	CFM (+mean/std features)			CFM (+mean feature)		
		$m = 5$	$m = 10$	$m = 15$	$m = 5$	$m = 10$	$m = 15$
Mean	0.5624	**0.5199**	0.5207	0.5215	0.5269	0.5234	0.5231
StdError	0.0501	0.0464	0.0451	0.0450	0.0466	0.0454	0.0450

where $\mathrm{tr}(\boldsymbol{W}) \leq \eta$ and $\boldsymbol{W} \succeq 0$. If $s = \mathrm{tr}(\boldsymbol{W}) < \eta$, we can add $(t-s)\boldsymbol{e}_1\boldsymbol{e}_1^\top$ to $\boldsymbol{U}_1\boldsymbol{U}_1^\top$, and we have $\mathrm{tr}(\boldsymbol{W}) = \eta$.

If the matrix is symmetric and positive semi-definite, we can decompose $\boldsymbol{W}$ as

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{U}_1 \\ \boldsymbol{V}_1 \\ \vdots \\ \boldsymbol{U}_3 \\ \boldsymbol{V}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1^\top & \boldsymbol{V}_1^\top & \dots & \boldsymbol{U}_3^\top & \boldsymbol{V}_3^\top \end{bmatrix},$$

such that $\boldsymbol{U}_m\boldsymbol{V}_m^\top = \boldsymbol{M}_{(m)}^{(m)}, m = 1, 2, 3$ and $\eta = \sum_{m=1}^3 \mathrm{tr}(\boldsymbol{U}_m\boldsymbol{U}_m^\top) + \mathrm{tr}(\boldsymbol{V}_m\boldsymbol{V}_m^\top) = \sum_{m=1}^3 \|\boldsymbol{U}_m\|_F^2 + \|\boldsymbol{V}_m\|_F^2$. □

REFERENCES

[1] Haim Avron, Satyen Kale, Shiva Kasiviswanathan, and Vikas Sindhwani. 2012. Efficient and practical stochastic subgradient descent for nuclear norm regularization. In *ICML*.

[2] Francis Bach, Julien Mairal, and Jean Ponce. 2008. Convex sparse matrix factorizations. *arXiv preprint arXiv:0812.1869* (2008).

[3] Mathieu Blondel, Akinori Fujino, and Naonori Ueda. 2015. Convex Factorization Machines. In *ECMLPKDD*.

[4] Guillaume Bouchard, Dawei Yin, and Shengbo Guo. 2013. Convex collective matrix factorization. In *AISTATS*.

[5] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. 2010. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization* 20, 4 (2010), 1956–1982.

[6] Emmanuel J Candès and Benjamin Recht. 2009. Exact matrix completion via convex optimization. *Foundations of Computational mathematics* 9, 6 (2009), 717–772.

[7] J Douglas Carroll and Jih-Jie Chang. 1970. Analysis of individual differences in multidimensional scaling via an N-way generalization of fiEckart-Youngfi decomposition. *Psychometrika* 35, 3 (1970), 283–319.

[8] Joon Hee Choi and S Vishwanathan. 2014. DFacTo: Distributed factorization of tensors. In *NIPS*.

[9] Patrick L Combettes and Valérie R Wajs. 2005. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation* 4, 4 (2005), 1168–1200.

[10] Maryam Fazel, Haitham Hindi, and Stephen P Boyd. 2001. A rank minimization heuristic with application to minimum order system approximation. In *ACC*.

[11] Marguerite Frank and Philip Wolfe. 1956. An algorithm for quadratic programming. *Naval Research Logistics Quarterly* 3, 1-2 (1956), 95–110.

[12] Suriya Gunasekar, Makoto Yamada, Dawei Yin, and Yi Chang. 2015. Consistent Collective Matrix Completion under Joint Low Rank Structure. In *AISTATS*.

[13] Elad Hazan. 2008. Sparse approximate solutions to semidefinite programs. In *LATIN 2008: Theoretical Informatics*.

[14] Liangjie Hong, Aziz S Doumith, and Brian D Davison. 2013. Co-factorization machines: modeling user interests and predicting individual decisions in twitter. In *WSDM*.

[15] Victoria Hore, Ana Viñuela, Alfonso Buil, Julian Knight, Mark I McCarthy, Kerrin Small, and Jonathan Marchini. 2016. Tensor decomposition for multiple-tissue gene expression experiments. *Nature Genetics* 48, 9 (2016), 1094–1100.

[16] Cho-Jui Hsieh and Peder Olsen. 2014. Nuclear norm minimization via active subspace selection. In *ICML*.

[17] Laurent Jacob and Jean-Philippe Vert. 2008. Protein-ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics* 24, 19 (2008), 2149–2156.

[18] Martin Jaggi. 2013. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML*.

[19] Martin Jaggi and Marek Sulovsky. 2010. A simple algorithm for nuclear norm regularized problems. In *ICML*.

[20] Shuiwang Ji and Jieping Ye. 2009. An accelerated gradient method for trace norm minimization. In *ICML*.

[21] Suleiman A Khan and Samuel Kaski. 2014. Bayesian multi-view tensor factorization. In *ECML*.

[22] Suleiman A Khan, Eemeli Leppäaho, and Samuel Kaski. 2016. Bayesian multi-tensor factorization. *Machine Learning* 105, 2 (2016), 233–253.

[23] Suleiman A Khan, Seppo Virtanen, Olli P Kallioniemi, Krister Wennerberg, Antti Poso, and Samuel Kaski. 2014. Identification of structural features in chemicals associated with cancer drug response: a systematic data-driven analysis. *Bioinformatics* 30, 17 (2014), i497–i504.

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[25] Dana Lahat, Tülay Adali, and Christian Jutten. 2015. Multimodal data fusion: an overview of methods, challenges, and prospects. *Proc. IEEE* 103, 9 (2015), 1449–1477.

[26] Justin Lamb et al. 2006. The Connectivity Map: Using Gene-Expression Signatures to Connect Small Molecules, Genes, and Disease. *Science* 313, 5795 (2006), 1929–1935. https://doi.org/10.1126/science.1132939

[27] Wenzhao Lian, Piyush Rai, Esther Salazar, and Lawrence Carin. 2015. Integrating Features and Similarities: Flexible Models for Heterogeneous Multiview Data. In *AAAI*.

[28] Yong-Jin Liu, Defeng Sun, and Kim-Chuan Toh. 2012. An implementable proximal point algorithmic framework for nuclear norm minimization. *Mathematical Pprogramming* 133, 1-2 (2012), 399–436.

[29] Francesco Locatello, Rajiv Khanna, Michael Tschannen, and Martin Jaggi. 2017. A Unified Optimization View on Generalized Matching Pursuit and Frank-Wolfe. *AISTATS* (2017).

[30] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. 2010. Spectral regularization algorithms for learning large incomplete matrices. *JMLR* 11 (2010), 2287–2322.

[31] Bradley N Miller, Istvan Albert, Shyong K Lam, Joseph A Konstan, and John Riedl. 2003. MovieLens unplugged: Experiences with an occasionally connected recommender system. In *IUI*.

[32] Morten Mørup and Lars Kai Hansen. 2009. Automatic relevance determination for multi-way models. *Journal of Chemometrics* 23, 7-8 (2009), 352–363.

[33] Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. 2010. Transfer Learning in Collaborative Filtering for Sparsity Reduction.. In *AAAI*.

[34] Benjamin Recht and Christopher Ré. 2013. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation* 5, 2 (2013), 201–226.

[35] Steffen Rendle. 2010. Factorization machines. In *ICDM*.

[36] Steffen Rendle. 2012. Factorization machines with libFm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 57.

[37] Steffen Rendle. 2013. Scaling factorization machines to relational data. In *VLDB*, Vol. 6. 337–348.

[38] Shai Shalev-Shwartz, Alon Gonen, and Ohad Shamir. 2011. Large-Scale Convex Minimization with a Low-Rank Constraint. In *ICML*.

[39] Robert H Shoemaker. 2006. The NCI60 human tumour cell line anticancer drug screen. *Nature Reviews Cancer* 6, 10 (2006), 813–823.

[40] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *KDD*.

[41] Masataka Takarabe, Masaaki Kotera, Yosuke Nishimura, Susumu Goto, and Yoshihiro Yamanishi. 2012. Drug target prediction using adverse event report systems: a pharmacogenomic approach. *Bioinformatics* 28, 18 (2012), i611–i618.

[42] Kim-Chuan Toh and Sangwoon Yun. 2010. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific Journal of Optimization* 6, 615-640 (2010), 15.

[43] Ryota Tomioka, Kohei Hayashi, and Hisashi Kashima. 2010. Estimation of low-rank tensors via convex optimization. *arXiv preprint arXiv:1010.0789* (2010).

[44] Ryota Tomioka and Taiji Suzuki. 2013. Convex tensor decomposition via structured schatten norm regularization. In *NIPS*.

[45] Ryota Tomioka, Taiji Suzuki, Kohei Hayashi, and Hisashi Kashima. 2011. Statistical performance of convex tensor decomposition. In *NIPS*.

[46] Ledyard R Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31, 3 (1966), 279–311.

[47] Marina Vinyes and Guillaume Obozinski. 2017. Fast column generation for atomic norm regularization. In *AISTATS*.

[48] Seppo Virtanen, Arto Klami, and Samuel Kaski. 2011. Bayesian CCA via group sparsity. In *ICML*.

[49] Seppo Virtanen, Arto Klami, Suleiman A Khan, and Samuel Kaski. 2012. Bayesian Group Factor Analysis. In *AISTATS*.

[50] Mingrui Wu. 2007. Collaborative filtering via ensembles of matrix factorizations. In *KDD*.

[51] Qian Xu, Evan Wei Xiang, and Qiang Yang. 2010. Protein-protein interaction prediction via collective matrix factorization. In *BIBM*.

[52] Wei Xu, Xin Liu, and Yihong Gong. 2003. Document clustering based on non-negative matrix factorization. In *SIGIR*.

[53] Makoto Yamada, Wenzhao Lian, Amit Goyal, Jianhui Chen, Kishan Wimalawarne, Suleiman A Khan, Samuel Kaski, Hiroshi Mamitsuka, and Yi Chang. 2015. Convex Factorization Machine for Regression. *arXiv preprint arXiv:1507.01073* (2015).

[54] Yan Yan, Mingkui Tan, Ivor Tsang, Yi Yang, Chengqi Zhang, and Qinfeng Shi. 2015. Scalable maximum margin matrix factorization by active Riemannian subspace search. In *IJCAI*.