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 Kashi
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 $\ell_2$-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

1 INTRODUCTION
In recommendation tasks including movie recommendation and news article recommendation, the data are represented in a matrix form, $A \in \mathbb{R}^{[U] \times [I]}$, 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 challenge1, 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 $x$ 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:

$$x_i = [0 \cdots 0 1 0 \cdots 0 0 \cdots 0 1 0 \cdots 0]^\top \in \mathbb{R}^d,$$

$$y_i = [A]_{k,k'}.$$

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

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

$$f(x; w, G) = w_0 + w_0^\top x + \sum_{t=1}^d \sum_{t'=t+1}^d g_t^\top g_{t'} x_t x_{t'},$$

where $w_0 \in \mathbb{R}$, $w_0 \in \mathbb{R}^d$, and $G = [g_1, \ldots, g_m] \in \mathbb{R}^{d \times m}$ are model parameters ($m \ll d$). Since only the $k$-th user and $k'$-th item

---

1https://www.kaggle.com/c/criteo-display-ad-challenge
We employ the linear+quadratic model, Eq. (1) and estimate with the matrix factorization method on toxicogenomics data. Including multi-view factorization [21] and tensor factorization [11, 18]. A key advantage of the proposed method over existing FMs is that the proposed method outperforms a state-of-the-art multi-view factorization [21], 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 \( x \).

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

\[
\min_{w_0, W} \sum_{i=1}^{n} (y_i - f(x_i; w_0, W))^2 + \lambda_1 \|w_0\|_2^2 + \lambda_2 \|W\|_{2,1}^2 + \lambda_3 \|G\|_F^2,
\]

where the \( \lambda_1, \lambda_2, \) and \( \lambda_3 \) are regularization parameters, and \( \|G\|_F \) 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 \( W \), 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 \( w \) and \( W \) such that the squared loss between the output \( y \) and the model prediction is minimized. More specifically, we regularize the linear parameter \( w \) with the \( \ell_2 \)-regularizer and the quadratic parameter \( W \) 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.

#### 2.1 Problem Formulation

We suppose that we are given \( n \) independent and identically distributed (i.i.d.) paired samples \( \{(x_i, y_i)\} \) drawn from a joint distribution with density \( p(x, y) \). We denote \( X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n} \) as the input data and \( y = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n \) as the output real-valued vector.

The goal of this paper is to find a model that predicts \( y \) given an input \( x \).

#### 2.2 Model

We employ the following model:

\[
f(x; w, W) = w_0 + w_\top x + \sum_{f=1}^{d} \sum_{i=1}^{d} \sum_{j=1}^{d} [W]_{f, f, i, j} x_i x_j,
\]

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

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

\[
f(x; w, W) = [w_\top \ \text{vec}(W)^\top] \begin{bmatrix} 1 & \text{vec}(x x^\top)^\top \\ \text{vec}(x x^\top)^\top & 2 \text{vec}(x x^\top - \text{diag}(x \circ x)) \end{bmatrix} Z,
\]

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

#### 2.3 Optimization problem

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

\[
\min_{w, W} \|y - f(X; w, W)\|_2^2 + \lambda_1 \|w\|_2^2 \quad \text{s.t.} \quad W \succeq 0 \text{ and } \|W\|_1 = \eta.
\]

where \( f(X; w, W) = [f(x_1; w, W), \ldots, f(x_n; w, W)]^\top \in \mathbb{R}^n \), \[13\] Code available at http://www.makotoyamada-ml.com/cfm.html
Specifically, we take the derivative of the objective function $W$ to be low-rank.

where $\sigma_i$ is the $i$-th singular value of $W$. 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 $\ell_1$ norm on singular values. Thus, by imposing the trace norm, we can make $W$ to be low-rank.

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

$$w^* = (ZZ^T + \lambda_1 I_{d+1})^{-1}Z(y - f_Q(X; W)),$$

where

$$f_Q(X; W) = \left[f_Q(x_1; W), \ldots, f_Q(x_n; W)\right]^T \in \mathbb{R}^n,$$

and $f_Q(x; W) = (x - \alpha)^T C (x - \alpha)$. The model corresponding to the quadratic term of the problem to a convex optimization problem with respect to $W$. We obtain the unknown parameter $W$ is modeled as $\hat{W}$.

Once $w^*$ is computed, we can back into the objective function $J(w, M)$ to recover $\hat{W}$ as

$$\hat{W} = (ZZ^T + \lambda_1 I_{d+1})^{-1}Z(y - f_Q(X; \hat{W})),$$

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

$$W = \begin{bmatrix} UU^T & M \\ M^T & VV^T \end{bmatrix} \geq 0, \ tr(UU^T) + tr(VV^T) = \eta,$$

where $U \in \mathbb{R}^{U \times m}$, $V \in \mathbb{R}^{V \times m}$ and $M = UV^T \in \mathbb{R}^{U \times V}$. Furthermore, for the CFM setting, the $k$-th user and $k'$-th item rating is modeled as

$$[\tilde{A}]_{k, k'} = w_0 + [w_0]_k + [w_0]_{k'} + [M]_{k, k'}.$$

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

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

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

$$W = \begin{bmatrix} G & M \\ M^T & H \end{bmatrix} \geq 0, \ tr(G) + tr(H) = \eta.$$

Based on Lemma 1, the optimization problem $J(w, M) = \min_{w, M} J(w, M) + \lambda_1 ||w||_2^2$ s.t. $||M||_{\text{tr}} \leq \frac{\eta}{2}$, is equivalent to

$$\tilde{J}(w, M, \lambda_1 ||w||_2^2) \leq \frac{\eta}{2},$$

where $\tilde{J}(w, M) = \sum_{\{k, k'\} \in \Omega} ([A]_{k, k'} - w_0 - [w_0]_k - [w_0]_{k'} + [M]_{k, k'})^2$, and $\Omega$ is the set of observed values in $A$. If we set $w = 0$, the optimization problem is equivalent to the matrix factorization with nuclear norm regularized [19]; CFM includes convex matrix factorization as a special case. Since we would like to have a low-rank matrix $M$ of the user-item matrix $A$ 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 $x$. That is, intrinsically, the MF method [19] and CFM are different.

2.4 Hazan’s Algorithm

For optimizing $W$, 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.

**Algorithm 1 CFM with Hazan’s Algorithm**

Rescale loss function $J_q(W) = f(\eta W)$.

Initialize $W(0)$, the curvature parameter $C_f = 1$, and the number of iterations $T$.

for all $t = 0, 1, \ldots, T$ do

Compute $\hat{p}^{(t)} = \text{approxEV}\{-\nabla J_q(W(t)), C_f/(t+1)\}.

Compute $\hat{\alpha}_t := \frac{\lambda}{t+2}$ (or $\hat{\alpha}_t = \text{argmin}_\alpha J_q(W(t) + \alpha(\hat{p}^{(t)}p^{(t)\top} - W(t)))$).

$$W(t+1) = W(t) + \hat{\alpha}_t(\hat{p}^{(t)}p^{(t)\top} - W(t)).$$

end for

return $W(T)$.
without storing $\nabla J(W^{(t)})$ in memory. Moreover, since the matrix $X$ is a sparse matrix, we can efficiently obtain the leading eigenvector by the `approxEV` function. For example in Matlab, we can obtain the approximate eigenvector $p$ by the function $[p, l] = \text{eigs}(\nabla J(W^{(t)}))$, where $l$ is the corresponding eigenvalue.

The proposed CFM optimization requires a matrix inversion (i.e., $O(n^3)$) for computing $C$ in $D(t)$, 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 $X$ is extremely sparse, and we can efficiently compute $D(t)$ by using a conjugate gradient method whose time complexity is $O(n)$.

$D(t)$ can be written as

$$D(t) = \text{diag}(Cg^{(t)} - Z^T \hat{W}^{(t)}),$$

where $g^{(t)} = y - f_0(X; W^{(t)})$. Since the number of samples $n$ tends to be larger than the dimensionality $d$ in factorization machine settings, $ZZ^T$ becomes full rank. Namely, we can safely make the regularization parameter $\lambda_1 = 0$. In such case, $D(t)$ is given as

$$D(t) = \text{diag}(\hat{y}^{(t)} - Z^T \hat{w}^{(t)}),$$

where we use $C = R^T R = I - Z^T (ZZ^T)^{-1} Z$. The $\hat{w}^{(t)} = (ZZ^T)^{-1} Z\hat{y}^{(t)}$ is obtained by solving

$$Z^T w = \hat{y}^{(t)},$$

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

Finally, we compute $D(t)$ as

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

The diagonal elements of $D(t)$ are the differences between the observed outputs and the model predictions at the $t$-th iteration. Note that, in our CFM optimization, we eliminate $w$ and only optimize $W$; however, the $w$ 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 $w$, 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 $O(nT)$, where $T$ is the total number of iterations in Hazan’s algorithm.

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

$$\hat{\alpha}_t = \arg\min_{\alpha} \frac{J(W^{(t+1)})}{J(W^{(t)})}$$

$$= \arg\min_{\alpha} \frac{\|R_y f_0(X; (1-\alpha)W^{(t)} + \alpha u^{(t)} u^{(t)\top})\|_2^2}{\|R_y f_0(X; p^{(t)} p^{(t)\top} - W^{(t)})\|_2^2}.$$  

Taking the derivative with respect to $\alpha$ and solving the problem for $\alpha$, we have

$$\hat{\alpha}_t = 1 + \frac{\|R_y f_0(X, W^{(t)})\|_2}{\|R_y f_0(X; p^{(t)} p^{(t)\top} - W^{(t)})\|_2}.$$  

The computation of $\hat{\alpha}_t$ involves the matrix inversion of $R_y$. However, by using the same technique as in the derivative computation, we can efficiently compute $\hat{\alpha}_t$.

**Update $W$:** When the input dimension $d$ is large, storing the feature-feature interaction matrix $W$ is not possible. To avoid the memory problem, we update $W$ as

$$W^{(t+1)} = p^{(t+1)} \text{diag}(\lambda^{(t+1)}) p^{(t+1)\top},$$

$$p^{(t+1)} = [p^t, p^t],$$

$$\lambda^{(t+1)} = 1 - \lambda^{(t)} \hat{\alpha}_t,$$

where $\lambda^{(t+1)} \in \mathbb{R}^{d+1}$. Thus, we only need to store $p^{(t+1)} \in \mathbb{R}^{d(1+t)}$ and $\lambda^{(t+1)}$ 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 $U = P \text{diag}(\lambda)^{1/2} \in \mathbb{R}^{d \times T}$ such that $W = U U^\top$. Then, we can efficiently compute the output as

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

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 $y \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, 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:

$$\min_{M(m)_{i,j,k}} \sum_{(i,j,k) \in \Omega} \left| y_{i,j,k} - M^{(0)}_{i,j,k} - \sum_{m=1}^3 M^{(m)}_{i,j,k} \right|^2 + \lambda \sum_{m=1}^3 \|M^{(m)}\|_{tr},$$

where $M^{(0)} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is the bias tensor, $M^{(m)} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is the $m$-th mode tensor, $\lambda \geq 0$ is the regularization parameter, $M_{i,j,k}$ is the unfolded matrix with respect to the $m$-th mode, $M^{(1)}_{i,j,k} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $M^{(2)}_{i,j,k} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, and $M^{(3)}_{i,j,k} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. The final goal here is to learn $M$ from $y$ by minimizing $J(M)$. Now, we reformulate Eq. (7) by CFM.
where \( W \geq 0, d = \sum_{m=1}^{3} n_m + \sum_{m=1}^{3} \sum_{m'=1}^{3} n_m n_m'. \) Note that the off-diagonal matrices are not important for deriving optimization algorithm, and thus, we omit them here. Moreover, since the matrix \( W \) is a positive semi-definite matrix, we can decompose it as

\[
W = [U_1 \quad V_1 \quad \ldots \quad U_3 \quad V_3],
\]

\[
W = \left[ \begin{array}{c}
U_1 \\
V_1 \\
\vdots \\
U_3 \\
V_3
\end{array} \right],
\]

where \( W = \sum_{m=1}^{3} [M(m)]_{i,j,k} \) as

\[
\sum_{m=1}^{3} [M(m)]_{i,j,k} = \frac{1}{2} \text{tr} \left( W (x_{i,j,k}^T x_{i,j,k} - \text{diag}(x_{i,j,k} \circ x_{i,j,k})) \right),
\]

\[
x_{i,j,k}^{(1)} = \begin{bmatrix}
0 & \ldots & 0 & 1 & \ldots & 0 & \ldots & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0
\end{bmatrix}^T,
\]

\[
x_{i,j,k}^{(2)} = \begin{bmatrix}
0 & \ldots & 0 & 1 & \ldots & 0 & \ldots & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0
\end{bmatrix}^T,
\]

\[
x_{i,j,k}^{(3)} = \begin{bmatrix}
0 & \ldots & 0 & 1 & \ldots & 0 & \ldots & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 & 0 & \ldots & 0
\end{bmatrix}^T,
\]

\[
x_{i,j,k} = [x_{i,j,k}^{(1)} \quad x_{i,j,k}^{(2)} \quad x_{i,j,k}^{(3)}]^T \in \mathbb{R}^d.
\]

For the bias tensor \( M(0) \), we parametrize it as

\[
[M(0)]_{i,j,k} = w^T x_{i,j,k}, \quad (8)
\]

where \( w \in \mathbb{R}^d, w_1 \in \mathbb{R}^{n_1}, w_2 \in \mathbb{R}^{n_2}, \) and \( w_3 \in \mathbb{R}^{n_3}. \) 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 \( M^{(1)}_{(1)} \in \mathbb{R}^{n_1 \times n_2 n_3}, M^{(2)}_{(2)} \in \mathbb{R}^{n_2 \times n_1 n_3}, M^{(3)}_{(3)} \in \mathbb{R}^{n_3 \times n_1 n_2} \) and \( \eta \):

\[
\sum_{m=1}^{3} \|M(m)\|_{\text{tr}} \geq \frac{\eta}{2}
\]

iff \( W \succeq 0 \) and \( \sum_{m=1}^{3} \text{tr}(G_m) + \text{tr}(H_m) = \eta. \)

**Proof:** See Appendix.

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

\[
\min_{w,W} \sum_{(i,j,k) \in \Omega} \left( [Y]_{i,j,k} - w^T x_{i,j,k} \right)^2
\]

\[
-\text{tr} \left( W (x_{i,j,k} x_{i,j,k}^T - \text{diag}(x_{i,j,k} \circ x_{i,j,k})) \right)
\]

s.t. \( W \succeq 0, \text{tr}(W) = \eta. \)

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 linear 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 \( W \). 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:

\[
\begin{align*}
\tilde{w}^{(t+1)} & = \arg\min_w \|y - f(X; w, W^{(t)})\|_2^2 + \lambda_1 \|w\|_2^2, \\
\tilde{W}^{(t+1)} & = \arg\min_W \|y - f(X; \tilde{w}^{(t+1)}, W)\|_2^2 + \lambda_2 \|W\|_F,
\end{align*}
\]

while our proposed approach is simply given as

\[
\tilde{W} = \arg\min_W \|R(y - fQ(X; W))\|_2^2, \text{s.t. } \|W\|_F = \eta.
\]

Hence, the BCD algorithm needs to iterate the sub-problem for \( W \) 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 \( W \).

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

$$\hat{M} = \arg\min_M \|P_{Ω}(A) - P_{Ω}(M)\|_F + \lambda \|M\|_{\Omega},$$

(9)

where $Ω$ is the set of observed values in $A$, $[P_{Ω}(A)]_{i,j} = [A]_{i,j}$ if $i, j \in Ω$ and 0 otherwise, and $\| \cdot \|_F$ is the Frobenius norm. Since Eq.(9) and Eq.(4) are equivalent when $w = 0$, 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 $O(\frac{1}{\sqrt{\epsilon}})$ ($\epsilon$ 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.

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 $f(x) = w_0 + \mathbf{w}^\top \mathbf{x}$, which is also a strong baseline. To estimate FM models, we use the publicly available libFM package.

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):

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

where $y^*$ and $\hat{y}$ 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 $x \in \mathbb{R}^{100}$ as $x \sim \mathcal{N}(0, I)$, and output values as

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

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

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 \( \eta \) 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 stddev = 0.05 and stdev = 0.1, which are the standard deviation of the random variable for initializing \( G \). 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 \((\alpha_t = \frac{2}{i^2})\) 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 \( \alpha_t = \frac{2}{i^2} \). 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., \( A^{(1)}_l \in \mathbb{R}^{1106 \times 78}, l = 1, 2, 3 \)). The second set, Toxicity, contains three dose-dependent toxicity profiles of the same 78 drugs over the three cancers (i.e., \( A^{(2)}_l \in \mathbb{R}^{3 \times 78}, l = 1, 2, 3 \)). 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.
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 FM_{MCMC}. FM (MCMC) can obtain better performance than CFMs (both our formulation and [3]) for these datasets if we set an appropriate initialization parameter.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CFM</th>
<th>CFM (BCD)</th>
<th>FM_{SGD}</th>
<th>FM_{ALS}</th>
<th>FM_{MCMC}^{(0.05)}</th>
<th>FM_{MCMC}^{(0.1)}</th>
<th>Ridge</th>
</tr>
</thead>
<tbody>
<tr>
<td>100K</td>
<td>0.915</td>
<td>0.93</td>
<td>1.078</td>
<td>1.242</td>
<td>0.905</td>
<td>0.901</td>
<td>0.936</td>
</tr>
<tr>
<td>1M</td>
<td>0.866</td>
<td>0.85</td>
<td>0.943</td>
<td>0.981</td>
<td>0.877</td>
<td>0.846</td>
<td>0.899</td>
</tr>
<tr>
<td>10M</td>
<td>0.810</td>
<td>0.82</td>
<td>0.827</td>
<td>0.873</td>
<td>0.831</td>
<td>0.778</td>
<td>0.855</td>
</tr>
<tr>
<td>20M</td>
<td>0.802</td>
<td>n/a</td>
<td>0.821</td>
<td>0.852</td>
<td>0.803</td>
<td>0.768</td>
<td>0.850</td>
</tr>
</tbody>
</table>

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

$$
\frac{1}{V} \sum_{v=1}^{V} \left\| \mathbf{y}_v^{x,v} - \mathbf{\hat{y}}_v^{x,v} \right\|_2^2,
$$

where $\mathbf{y}_v^{x,v}$ is the target score vector, $\mathbf{\hat{y}}_v^{x,v}$ is the estimated score vector, and $V$ 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 (BMTF) [21]. BMTF is a state-of-the-art multi-view factorization method.

For CFM, we first concatenate all view matrices as

$$
\mathbf{A} = \begin{bmatrix} A_1^{(1)} & A_2^{(1)} \\ A_1^{(2)} & A_2^{(2)} \end{bmatrix} \in \mathbb{R}^{332 \times 78},
$$

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

$$
\mathbf{x} = \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{1} & \cdots & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{1} & \cdots & \mathbf{0} \end{bmatrix}^\top, \\
\mathbf{y} = [\mathbf{A}]_{i,j},
$$

where $\mathbf{x} \in \mathbb{R}^{3407}$.

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 use 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:

- **Mean of $m$-nearest neighbor similarities ($x_{\text{mean}}$):** We first find the $m$-nearest neighbors of the $i$-th drug target, 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 ($x_{\text{std}}$):** Similarly to the mean feature, we first found $m$-nearest neighbor similarities and then computed their standard deviation.

Then, we use these features as

$$
\mathbf{x} = [0 \cdots 0 \mathbf{1} \cdots \mathbf{0} 3 \cdots 0 \mathbf{1} \cdots \mathbf{0} \mathbf{x}_{\text{mean}} \mathbf{x}_{\text{std}}]^\top,
$$

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

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 $\mathbf{W}$, 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
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 $A^{(1)}$, $A^{(2)}$, and $A^{(3)}$ for factorization, while we used only $A^{(2)}$ for the single-view algorithms. The proposed CFM outperforms the state-of-the-art methods.

<table>
<thead>
<tr>
<th></th>
<th>CFM</th>
<th>BMTF</th>
<th>GFA</th>
<th>ARDCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.4037</td>
<td>0.4811</td>
<td>0.5223</td>
<td>0.8919</td>
</tr>
<tr>
<td>StdError</td>
<td>0.0163</td>
<td>0.0061</td>
<td>0.0041</td>
<td>0.0027</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Single-view</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>5.3713</td>
</tr>
<tr>
<td>StdError</td>
<td>0.0282</td>
</tr>
</tbody>
</table>

Table 3: Test relative MSE on toxicogenomics prediction task. In this task, we used the gene expression matrices $A^{(1)}$, $A^{(2)}$, and $A^{(3)}$ 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.

<table>
<thead>
<tr>
<th></th>
<th>CFM ($\pm$mean/std features)</th>
<th>CFM ($\pm$mean feature)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>m = 5</td>
<td>m = 10</td>
</tr>
<tr>
<td>Mean</td>
<td>0.5624</td>
<td>0.5199</td>
</tr>
<tr>
<td>StdError</td>
<td>0.0501</td>
<td>0.0464</td>
</tr>
</tbody>
</table>

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 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 Lemma 1 of [19]. From the characterization:

\[
\sum_{m=1}^3 \|M^{(m)}\|_F = \min_{\{U_m, V_m, M^{(m)}\}_{m=1}^3} \frac{1}{2} \sum_{m=1}^3 (\|U_m\|_F^2 + \|V_m\|_F^2)
\]

we have that \( \exists U_m, V_m, M^{(m)} = M^{(m)}_{(m)}, m = 1, 2, 3 \) s.t.

\[
2 \sum_{m=1}^3 \|M^{(m)}_{(m)}\|_F = \sum_{m=1}^3 \|U_m\|_F^2 + \|V_m\|_F^2 = \sum_{m=1}^3 \text{tr}(U_m U_m^T) + \text{tr}(V_m V_m^T) \leq \eta.
\]

That is, we have

\[
W = \begin{bmatrix} U_1 U_1^T & M^{(1)}_{(1)} & M^{(1)}_{(2)} & M^{(1)}_{(3)} \\ V_1 V_1^T & M^{(1)}_{(2)} & M^{(2)}_{(2)} & M^{(2)}_{(3)} \\ V_3 V_3^T & \ddots & \ddots & \ddots \\ U_3 U_3^T & V_3 V_3^T & \ddots & \ddots \end{bmatrix},
\]

where \( \text{tr}(W) \leq \eta \) and \( W \geq 0 \). If \( s = \text{tr}(W) < \eta \), we can add \((t-s)e_1^T e_1^T\) to \(U_1 U_1^T\), and we have \(\text{tr}(W) = \eta\).

If the matrix is symmetric and positive semi-definite, we can decompose \(W\) as

\[
W = \begin{bmatrix} U_1^T \\ V_1 \\ \vdots \\ U_3 \\ V_3 \end{bmatrix} \begin{bmatrix} U_1^T & V_1^T & \ldots & U_3^T & V_3^T \end{bmatrix},
\]

such that \(U_m V_m^T = M^{(m)}_{(m)}, m = 1, 2, 3\) and \(\eta = \sum_{m=1}^3 \text{tr}(U_m U_m^T) + \text{tr}(V_m V_m^T) = \sum_{m=1}^3 \|U_m\|_F^2 + \|V_m\|_F^2\). \(\square\)