

Notes on Low-rank Matrix Factorization

Yuan Lu, Jie Yang*
{joyce.yuan.lu,yangjiera}@gmail.com.

* *Faculty of EEMCS,
Delft University of Technology,
Mekelweg 4, 2628 CD Delft, the Netherlands.*

Dedicated to Xiao Baobao and Tu Daye.

1 Introduction

Low-rank matrix factorization (**MF**) is an important technique in data science. The key idea of **MF** is that there exists latent structures in the data, by uncovering which we could obtain a compressed representation of the data. By factorizing an original matrix to low-rank matrices, **MF** provides a unified method for dimension reduction, clustering, and matrix completion.

MF has several nice properties: 1) it uncovers latent structures in the data, while addressing the data sparseness problem [11]; 2) it has an elegant probabilistic interpretation [15]; 3) it can be easily extended with domain specific prior knowledge (e.g., homophily in linked data [19]), thus suitable for various real-world problems; 4) many optimization methods such as (stochastic) gradient-based methods can be applied to find a good solution.

In this article we review several important variants of **MF**, including:

- Basic **MF**,
- Non-negative **MF**,
- Orthogonal non-negative **MF**.

As can be seen from their names, non-negative **MF** and orthogonal non-negative **MF** are variants of basic **MF** with non-negativity and/or orthogonality constraints. Such constraints are useful in specific scenarios. In the first part of this article, we introduce, for each of these models, the application scenarios, the distinctive properties, and the optimizing method. Note that for the optimizing method, we mainly use the alternative algorithm, as similar to [4, 19]. We will derive the updating rules, and prove the correctness and convergence. For reference, matrix operation and optimization can be referred to [2] and [1] respectively.

By properly adapting **MF**, we can go beyond the problem of clustering and matrix completion. In the second part of this article, we will extend **MF** to sparse matrix completion, enhance matrix completion using various regularization methods, and make use of **MF** for (semi-)supervised learning by introducing latent space reinforcement and transformation. We will see that **MF** is not only a useful model but also as a flexible framework that is applicable for various prediction problems.

2 Theory

This section introduces the theory in low-rank matrix factorization. As introduced before, we will go through the following three **MF** variations: basic **MF**, non-negative **MF**, orthogonal non-negative **MF**.

2.1 Basic MF

We start with the basic MF model, formulated as

$$\min_{\mathbf{U}, \mathbf{V}} \|\mathbf{X} - \mathbf{UV}^T\| + \mathcal{L}(\mathbf{U}, \mathbf{V}), \quad (1)$$

where $\mathbf{X} \in \mathbb{R}^{m \times n}$ is the data matrix to be approximated, and $\mathbf{U} \in \mathbb{R}^{m \times k}$, $\mathbf{V} \in \mathbb{R}^{n \times k}$ are two low-dimensional matrices ($k \ll \min(m, n)$). $\mathcal{L}(\mathbf{U}, \mathbf{V})$ is a regularization part to avoid overfitting. Regularization is usually necessary in prediction for bias-variance trade-off [9].

2.1.1 Gradient Descent Optimization

We instantiate Eq. 1 as follows

$$\min_{\mathbf{U}, \mathbf{V}} \mathcal{O} = \|\mathbf{X} - \mathbf{UV}^T\|_F^2 + \alpha \|\mathbf{U}\|_F^2 + \beta \|\mathbf{V}\|_F^2. \quad (2)$$

The reason of using Frobenius Norm is that it has a Gaussian noise interpretation, and that the objective function can be easily transformed to a matrix trace version:

$$\min_{\mathbf{U}, \mathbf{V}} \mathcal{O} = Tr(\mathbf{X}^T \mathbf{X} + \mathbf{VU}^T \mathbf{UV}^T - 2\mathbf{X}^T \mathbf{UV}^T) + \alpha Tr(\mathbf{U}^T \mathbf{U}) + \beta Tr(\mathbf{V}^T \mathbf{V}). \quad (3)$$

Here the matrix calculation rule $\|\mathbf{A}\|_F = \sqrt{Tr(\mathbf{A}^T \mathbf{A})}$ is used in the transformation. Note that trace has many good properties such as $Tr(\mathbf{A}) = Tr(\mathbf{A}^T)$ and $Tr(\mathbf{AB}) = Tr(\mathbf{BA})$, which will be used in the following derivations.

According to trace derivatives $\frac{\partial Tr(\mathbf{AB})}{\partial \mathbf{A}} = \mathbf{B}^T$ and the following rules:

$$\begin{aligned} \frac{\partial Tr(\mathbf{A}^T \mathbf{AB})}{\partial \mathbf{A}} &= \mathbf{A}(\mathbf{B}^T + \mathbf{B}), \\ \frac{\partial Tr(\mathbf{AA}^T \mathbf{B})}{\partial \mathbf{A}} &= (\mathbf{B}^T + \mathbf{B})\mathbf{A} \end{aligned} \quad (4)$$

(see more in [2]), we have the following derivatives for \mathbf{U} and \mathbf{V} ,

$$\begin{aligned} \frac{\partial \mathcal{O}}{\partial \mathbf{U}} &= \frac{\partial Tr(\mathbf{VU}^T \mathbf{UV}^T - 2\mathbf{X}^T \mathbf{UV}^T) + \alpha Tr(\mathbf{U}^T \mathbf{U})}{\partial \mathbf{U}} \\ &= \frac{\partial Tr(\mathbf{U}^T \mathbf{UV}^T \mathbf{V} - 2\mathbf{UV}^T \mathbf{X}^T) + \alpha Tr(\mathbf{U}^T \mathbf{U})}{\partial \mathbf{U}} \\ &= 2(\mathbf{UV}^T \mathbf{V} - \mathbf{XV} + \alpha \mathbf{U}), \\ \frac{\partial \mathcal{O}}{\partial \mathbf{V}} &= \frac{\partial Tr(\mathbf{VU}^T \mathbf{UV}^T - 2\mathbf{X}^T \mathbf{UV}^T) + \beta Tr(\mathbf{V}^T \mathbf{V})}{\partial \mathbf{V}} \\ &= \frac{\partial Tr(\mathbf{V}^T \mathbf{VU}^T \mathbf{U} - 2\mathbf{V}^T \mathbf{X}^T \mathbf{U}) + \beta Tr(\mathbf{V}^T \mathbf{V})}{\partial \mathbf{V}} \\ &= 2(\mathbf{VU}^T \mathbf{U} - \mathbf{X}^T \mathbf{U} + \beta \mathbf{V}). \end{aligned} \quad (5)$$

Using these two derivatives, we can alternatively update \mathbf{U} and \mathbf{V} in each iteration of gradient descent algorithm.

Note that the derivation can also be performed elementarily for each entry in matrix \mathbf{U} , \mathbf{V} – this is, in fact, the original definition of matrix calculus. Such element-wise derivation is especially useful in stochastic optimization. We will touch this in a brief discussion of different algorithm schemes next.

2.1.2 Algorithm Schemes in CF and Others

For collaborative filtering, usually we take one subset of rated entries in \mathbf{X} as training set, and the rest rated entries as validation set. Detailed algorithm can be found in [18]. An important implementation strategy is that, for each rated entry in the training set, we update an entire row of \mathbf{U} and an entire column of \mathbf{V}^T , as the whole row or column is involved in approximating the rated entry. Same updating mechanism could be applied in stochastic algorithm.

In the meanwhile, similarly to stochastic algorithm, this type of updating does not fully utilize the data matrix in each updating iteration. The reason is that, not only an entire row of \mathbf{U} (and a column of \mathbf{V}^T) is involved in a single entry in data matrix \mathbf{X} , but also that a row of \mathbf{U} (and a column of \mathbf{V}^T) influences an entire row (column) of \mathbf{X} . Therefore for faster convergence, we recommend to update the matrix \mathbf{U} and \mathbf{V} by fully using data matrix \mathbf{X} .

As the objective function is non-convex caused by the coupling between \mathbf{U} and \mathbf{V} , we can choose to alternatively update \mathbf{U} and \mathbf{V} in each iteration as in [4, 19]. Detailed algorithm is similar to the one in [19]. Within any of these matrices, updating should be performed simultaneously as in all gradient-based methods. Note that, we still need to choose a small learning rate to ensure that the objective function is monotonically decreasing. Interestingly, the alternative optimization scheme is even more suitable for non-negative **MF** [13, 14, 5, 4], as we will see in the following subsections.

2.2 Non-negative MF

Non-negative **MF** [13] seeks to approximate data matrix \mathbf{X} with low-dimensional matrices \mathbf{U} , \mathbf{V} whose entries are all non-negative, i.e., $\mathbf{U}, \mathbf{V} \geq \mathbf{0}$. The new problem becomes:

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}} \mathcal{O} &= \|\mathbf{X} - \mathbf{UV}^T\|_F^2 + \alpha\|\mathbf{U}\|_F^2 + \beta\|\mathbf{V}\|_F^2 \\ \text{s.t. } &\mathbf{U} \geq \mathbf{0}, \mathbf{V} \geq \mathbf{0}. \end{aligned} \tag{6}$$

Non-negativity constraint is originated from parts-of-whole interpretation [13]. As we can think of, many real-world data are non-negative, such as link strength, favorite strength, etc. Non-negative **MF** may uncover the important parts, which sometimes can not be achieved by non-constrained **MF** [13].

Apart from the advantage of uncovering parts, non-negative **MF** has its own computational advantage: there is a relatively fixed method to find a learning

rate larger than common gradient-based methods. To illustrate this, we will first derive the updating rule for Eq. 6 as an example, then show the general approach for proving the convergence of updating rules derived from the relatively fixed method.

2.2.1 Updating Rule Derivation

The basic idea is using KKT complementary slackness conditions to enforce the non-negativity constraint. Based on this, we can directly obtain updating rules.

The Lagrangian function of Eq. 6 is

$$L = \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 + \alpha\|\mathbf{U}\|_F^2 + \beta\|\mathbf{V}\|_F^2 - Tr(\Lambda_1\mathbf{U}^T) - Tr(\Lambda_2\mathbf{V}^T). \quad (7)$$

We have the following KKT condition,

$$\begin{aligned} \Lambda_1 \circ \mathbf{U} &= \mathbf{0}, \\ \Lambda_2 \circ \mathbf{V} &= \mathbf{0}, \end{aligned} \quad (8)$$

where \circ denotes the Hadamard product. We then have

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{U}} &= \frac{\partial Tr(\mathbf{V}\mathbf{U}^T\mathbf{U}\mathbf{V}^T - 2\mathbf{X}^T\mathbf{U}\mathbf{V}^T) + \alpha Tr(\mathbf{U}^T\mathbf{U}) - Tr(\Lambda_1\mathbf{U}^T)}{\partial \mathbf{U}} \\ &= 2(\mathbf{U}\mathbf{V}^T\mathbf{V} - \mathbf{X}\mathbf{V} + \alpha\mathbf{U}) - \Lambda_1, \\ \frac{\partial L}{\partial \mathbf{V}} &= \frac{\partial Tr(\mathbf{V}\mathbf{U}^T\mathbf{U}\mathbf{V}^T - 2\mathbf{X}^T\mathbf{U}\mathbf{V}^T) + \beta Tr(\mathbf{V}^T\mathbf{V}) - Tr(\Lambda_2\mathbf{V}^T)}{\partial \mathbf{V}} \\ &= 2(\mathbf{V}\mathbf{U}^T\mathbf{U} - \mathbf{X}^T\mathbf{U} + \beta\mathbf{V}) - \Lambda_2. \end{aligned} \quad (9)$$

Let $\frac{\partial L}{\partial \mathbf{U}} = \mathbf{0}$ and $\frac{\partial L}{\partial \mathbf{V}} = \mathbf{0}$ as another KKT condition, we have

$$\begin{aligned} \Lambda_1 &= 2(\mathbf{U}\mathbf{V}^T\mathbf{V} - \mathbf{X}\mathbf{V} + \alpha\mathbf{U}), \\ \Lambda_2 &= 2(\mathbf{V}\mathbf{U}^T\mathbf{U} - \mathbf{X}^T\mathbf{U} + \beta\mathbf{V}). \end{aligned} \quad (10)$$

Now we combine Eq. 8 and Eq. 10, we have

$$\begin{aligned} (\mathbf{U}\mathbf{V}^T\mathbf{V} - \mathbf{X}\mathbf{V} + \alpha\mathbf{U}) \circ \mathbf{U} &= \mathbf{0}, \\ (\mathbf{V}\mathbf{U}^T\mathbf{U} - \mathbf{X}^T\mathbf{U} + \beta\mathbf{V}) \circ \mathbf{V} &= \mathbf{0}. \end{aligned} \quad (11)$$

from which, we have the final updating rules,

$$\begin{aligned} \mathbf{U}(i, j) &\leftarrow \mathbf{U}(i, j) \sqrt{\frac{(\mathbf{X}\mathbf{V})(i, j)}{(\mathbf{U}\mathbf{V}^T\mathbf{V} + \alpha\mathbf{U})(i, j)}}, \\ \mathbf{V}(i, j) &\leftarrow \mathbf{V}(i, j) \sqrt{\frac{(\mathbf{X}^T\mathbf{U})(i, j)}{(\mathbf{V}\mathbf{U}^T\mathbf{U} + \beta\mathbf{V})(i, j)}}. \end{aligned} \quad (12)$$

Detailed algorithm using these rules is similar to the one in [19]. We can see that, instead of manually setting small learning rates Λ 's, Eq. 12 directly offer updating rules that can usually lead to faster convergence.

The correctness of these updating rules is straightforward to find out. Taking \mathbf{U} as an example, from Eq. 12 we have either $\mathbf{U} = \mathbf{0}$ or $\mathbf{U}\mathbf{V}^T\mathbf{V} - \mathbf{X}\mathbf{V} + \alpha\mathbf{U} = \mathbf{0}$, which combined together, exactly equal to Eq. 11. The convergence, however, is somehow more difficult to be proved. We leave this to the next subsection.

2.2.2 Proof of Convergence

We prove the convergence of the updating rules in Eq. 12 with the standard auxiliary function approach, which is proposed in [14] and extended in [5, 4]. Our proof is mainly based on [5, 4], although the objective function Eq. 6 is slightly different.

An **auxiliary function** $G(\mathbf{U}, \mathbf{U}^t)$ of function $L(\mathbf{U})$ is a function that satisfies

$$G(\mathbf{U}, \mathbf{U}) = L(\mathbf{U}), \quad G(\mathbf{U}, \mathbf{U}^t) \geq L(\mathbf{U}). \quad (13)$$

Then, if we take \mathbf{U}^{t+1} such that

$$\mathbf{U}^{t+1} = \arg \min_{\mathbf{U}} G(\mathbf{U}, \mathbf{U}^t), \quad (14)$$

we have

$$L(\mathbf{U}^{t+1}) \leq G(\mathbf{U}^{t+1}, \mathbf{U}^t) \leq G(\mathbf{U}^t, \mathbf{U}^t) = L(\mathbf{U}^t). \quad (15)$$

This proves that $L(\mathbf{U})$ is monotonically decreasing.

Turn back to our problem, we need to take two steps using auxiliary function to prove the convergence of updating rules: 1) find an appropriate auxiliary function, and 2) find the global minima of the auxiliary function. As a remark, the auxiliary function approach in principle is similar to Expectation-Maximization approach that is widely used in statistical inference. Now let us complete the proof by taking the above two steps.

Step 1 - Finding an appropriate auxiliary function needs to take advantage of two inequalities,

$$z \geq 1 + \log z, \quad \forall z > 0, \quad (16)$$

$$\sum_{i=1}^m \sum_{j=1}^k \frac{(\mathbf{A}\mathbf{S}'\mathbf{B})(i, j)\mathbf{S}(i, j)^2}{\mathbf{S}'(i, j)} \geq \text{Tr}(\mathbf{S}'^T \mathbf{A}\mathbf{S}\mathbf{B}),$$

$$\forall \mathbf{A} \in \mathbb{R}_+^{m \times m}, \mathbf{B} \in \mathbb{R}_+^{k \times k}, \mathbf{S}' \in \mathbb{R}_+^{m \times k}, \mathbf{S} \in \mathbb{R}_+^{m \times k}. \quad (17)$$

The proof for Eq. 17 can be found in [5] (Proposition 6).

After removing irrelevant terms, the objective function Eq. 6 in terms of \mathbf{U} can be written as

$$\begin{aligned} & \text{Tr}(\mathbf{V}\mathbf{U}^T\mathbf{U}\mathbf{V}^T - 2\mathbf{X}^T\mathbf{U}\mathbf{V}^T) + \alpha\text{Tr}(\mathbf{U}^T\mathbf{U}) \\ & = \text{Tr}(\mathbf{U}^T\mathbf{U}\mathbf{V}^T\mathbf{V} - 2\mathbf{U}^T\mathbf{X}\mathbf{V}) + \alpha\text{Tr}(\mathbf{U}^T\mathbf{U}) \end{aligned} \quad (18)$$

We now propose an auxiliary function

$$G(\mathbf{U}, \mathbf{U}^t) = -2 \sum_{i,j} (\mathbf{XV})(i,j) \mathbf{U}^t(i,j) \left(1 + \log \frac{\mathbf{U}(i,j)}{\mathbf{U}^t(i,j)}\right) + \sum_{i,j} \frac{(\mathbf{U}^t \mathbf{V}^T \mathbf{V})(i,j) \mathbf{U}(i,j)^2}{\mathbf{U}^t(i,j)} + \alpha \sum_{i,j} \frac{\mathbf{U}^t(i,j) \mathbf{U}(i,j)^2}{\mathbf{U}^t(i,j)}. \quad (19)$$

Combining the two inequalities Eq. 16, 17, it is straightforward to see that Eq. 19 is a legal auxiliary function for Eq. 18, i.e., the two conditions in Eq. 13 are satisfied. Now we proceed to find \mathbf{U}^{t+1} that satisfies condition Eq. 14.

Step 2 - Finding \mathbf{U}^{t+1} can be achieved by obtaining the *global minima* of Eq. 19. First, we have

$$\frac{\partial G(\mathbf{U}, \mathbf{U}^t)}{\partial \mathbf{U}(i,j)} = -2(\mathbf{XV})(i,j) \frac{\mathbf{U}^t(i,j)}{\mathbf{U}(i,j)} + 2 \frac{(\mathbf{U}^t \mathbf{V}^T \mathbf{V})(i,j) \mathbf{U}(i,j)}{\mathbf{U}^t(i,j)} + 2\alpha \mathbf{U}(i,j). \quad (20)$$

Let $\frac{\partial G(\mathbf{U}, \mathbf{U}^t)}{\partial \mathbf{U}(i,j)} = 0$ we have

$$(\mathbf{XV})(i,j) \frac{\mathbf{U}^t(i,j)}{\mathbf{U}^{t+1}(i,j)} = \left(\frac{(\mathbf{U}^t \mathbf{V}^T \mathbf{V})(i,j)}{\mathbf{U}^t(i,j)} + \alpha \right) \mathbf{U}^{t+1}(i,j), \quad (21)$$

from which we directly have

$$\mathbf{U}^{t+1}(i,j) = \mathbf{U}^t(i,j) \sqrt{\frac{(\mathbf{XV})(i,j)}{(\mathbf{U}^t \mathbf{V}^T \mathbf{V} + \alpha \mathbf{U}^t)(i,j)}}, \quad (22)$$

which is exactly the updating rule for \mathbf{U} in Eq. 12. Similar result can be obtained for \mathbf{V} .

General observation If we go over the entire derivation process, by comparing Eq. 22 and Eq. 11, we can observe that the only thing that matters for the final updating rules is the signs of the terms in Eq. 11.

2.3 Orthogonal Non-negative MF

Orthogonality is another important constraint to **MF**. First of all, we formulate the problem as

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}} \mathcal{O} &= \|\mathbf{X} - \mathbf{UV}^T\|_F^2 \\ s.t. \quad \mathbf{U}, \mathbf{V} &\geq \mathbf{0}, \mathbf{U}^T \mathbf{U} = \mathbf{I}, \mathbf{V}^T \mathbf{V} = \mathbf{I}. \end{aligned} \quad (23)$$

Note that here we do not add regularization due to the orthogonality constraint.

It is proved in [3, 5] ([5] gives more mature proof) that this problem is equivalent to K-means clustering: \mathbf{V}' is an indication matrix with $\mathbf{V}'(i, j) = 0$ if \mathbf{x}_i belongs to the j^{th} ($1 \leq j \leq k$) cluster. Here $\mathbf{V} = \mathbf{V}'(\mathbf{V}'^T \mathbf{V}')^{-1/2}$, i.e., \mathbf{V} is a normalized version of \mathbf{V}' : \mathbf{V}' is a constant scaling of corresponding row of \mathbf{V} , and $\|\mathbf{V}(:, j)\|_2^2 = 1$.

2.3.1 3-factor MF vs. 2-factor MF

We call Eq. 23 1-sided 2-factor orthogonal non-negative **MF**, as only one factorized matrix needs to be orthogonal, and there are in total two factorized matrices. It is recommended that, to simultaneously cluster rows and columns in \mathbf{X} , we need 3-factor bi-orthogonal non-negative **MF**, i.e., both \mathbf{U} and \mathbf{V} being orthogonal:

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{H}, \mathbf{V}} \mathcal{O} &= \|\mathbf{X} - \mathbf{U}\mathbf{H}\mathbf{V}^T\|_F^2 \\ \text{s.t. } \mathbf{U}, \mathbf{H}, \mathbf{V} &\geq \mathbf{0}, \mathbf{U}^T \mathbf{U} = \mathbf{I}, \mathbf{V}^T \mathbf{V} = \mathbf{I}. \end{aligned} \quad (24)$$

It is proved that, compared to 3-factor bi-orthogonal non-negative **MF**, 2-factor bi-orthogonal non-negative **MF** is too restrictive, and will lead to poor approximation [5].

3-factor bi-orthogonal non-negative **MF** is useful in document-word clustering [5], outperforming K-means (i.e., 1-sided 2-factor orthogonal non-negative **MF**). It has been applied for tasks such as sentiment analysis [10].

2.3.2 Updating Rule Derivation

We now derive updating rules for Eq. 24, as we did before for non-negative **MF**.

The Lagrangian function for Eq. 24 is

$$\begin{aligned} L &= \|\mathbf{X} - \mathbf{U}\mathbf{H}\mathbf{V}^T\|_F^2 - Tr(\Lambda_U \mathbf{U}^T) - Tr(\Lambda_H \mathbf{H}^T) - Tr(\Lambda_V \mathbf{V}^T) \\ &\quad + Tr(\Gamma_U (\mathbf{U}^T \mathbf{U} - \mathbf{I})) + Tr(\Gamma_V (\mathbf{V}^T \mathbf{V} - \mathbf{I})) \end{aligned} \quad (25)$$

We then compute the updating rules for \mathbf{H} , \mathbf{U} , \mathbf{V} sequentially.

Computation of \mathbf{H}

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{H}} &= \frac{\partial Tr(\mathbf{V}\mathbf{H}^T \mathbf{U}^T \mathbf{U}\mathbf{H}\mathbf{V}^T - 2\mathbf{X}\mathbf{V}\mathbf{H}^T \mathbf{U}^T) - Tr(\Lambda_H \mathbf{H}^T)}{\partial \mathbf{H}} \\ &= 2\mathbf{U}^T \mathbf{U}\mathbf{H}\mathbf{V}^T \mathbf{V} - 2\mathbf{U}^T \mathbf{X}\mathbf{V} - \Lambda_H, \end{aligned} \quad (26)$$

We have the following KKT conditions,

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{H}} &= \mathbf{0} \\ \Lambda_H \circ \mathbf{H} &= \mathbf{0}. \end{aligned} \quad (27)$$

Combining the above three equations, we have

$$(\mathbf{U}^T \mathbf{U} \mathbf{H} \mathbf{V}^T \mathbf{V} - \mathbf{U}^T \mathbf{X} \mathbf{V}) \circ \mathbf{H} = \mathbf{0}. \quad (28)$$

Therefore we have the following updating rule for \mathbf{H} ,

$$\mathbf{H}(i, j) \leftarrow \mathbf{H}(i, j) \sqrt{\frac{(\mathbf{U}^T \mathbf{X} \mathbf{V})(i, j)}{(\mathbf{U}^T \mathbf{U} \mathbf{H} \mathbf{V}^T \mathbf{V})(i, j)}}. \quad (29)$$

Note that $\mathbf{U}^T \mathbf{U} \neq \mathbf{I}$ during the optimizing process.

Computation of \mathbf{U}, \mathbf{V}

Due to the orthogonality constraint, obtaining the updating rules for \mathbf{U}, \mathbf{V} needs to eliminate both Λ and Γ in the final updating rules. This will need the following equality,

$$\mathbf{U}^T \Lambda_U = \mathbf{0} \Leftrightarrow \Lambda_U \circ \mathbf{U} = \mathbf{0} \quad (30)$$

The latter will automatically be satisfied according to KKT conditions as we will see below.

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{U}} &= \frac{\partial \text{Tr}(\mathbf{V} \mathbf{H}^T \mathbf{U}^T \mathbf{U} \mathbf{H} \mathbf{V}^T - 2 \mathbf{X} \mathbf{V} \mathbf{H}^T \mathbf{U}^T) - \text{Tr}(\Lambda_U \mathbf{U}^T) + \text{Tr}(\Gamma_U (\mathbf{U}^T \mathbf{U} - \mathbf{I}))}{\partial \mathbf{U}} \\ &= 2 \mathbf{U} \mathbf{H} \mathbf{V}^T \mathbf{V} \mathbf{H}^T - 2 \mathbf{X} \mathbf{V} \mathbf{H}^T - \Lambda_U + 2 \mathbf{U} \Gamma_U, \end{aligned} \quad (31)$$

We have the following KKT conditions,

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{U}} &= \mathbf{0} \\ \Lambda_U \circ \mathbf{U} &= \mathbf{0}. \end{aligned} \quad (32)$$

Combining the above three equations we have

$$(\mathbf{U} \mathbf{H} \mathbf{V}^T \mathbf{V} \mathbf{H}^T - \mathbf{X} \mathbf{V} \mathbf{H}^T + \mathbf{U} \Gamma_U) \circ \mathbf{U} = \mathbf{0} \quad (33)$$

and

$$\Gamma_U = \mathbf{U}^T \mathbf{X} \mathbf{V} \mathbf{H}^T - \mathbf{H} \mathbf{V}^T \mathbf{V} \mathbf{H}^T. \quad (34)$$

Note that here we can have $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ as we only want an expression for Γ_U . Further note that for Λ we have the constraint $\Lambda > \mathbf{0}$ (according to KKT condition) while for Γ we do not have such constraint. Therefore we need to split Γ into two parts,

$$\begin{aligned} \Gamma_U &= \Gamma_U^+ - \Gamma_U^- \\ \Gamma_U^+ &= (|\Gamma_U| + \Gamma_U)/2 \\ \Gamma_U^- &= (|\Gamma_U| - \Gamma_U)/2. \end{aligned} \quad (35)$$

Using this division we rewrite Eq. 33, we then have

$$(\mathbf{U}\mathbf{H}\mathbf{V}^T\mathbf{V}\mathbf{H}^T - \mathbf{X}\mathbf{V}\mathbf{H}^T + \mathbf{U}\Gamma_U^+ - \mathbf{U}\Gamma_U^-) \circ \Lambda_U = \mathbf{0}. \quad (36)$$

Therefore the final updating rule for \mathbf{U} is

$$\mathbf{U}(i, j) \leftarrow \mathbf{U}(i, j) \sqrt{\frac{(\mathbf{X}\mathbf{V}\mathbf{H}^T + \mathbf{U}\Gamma_U^-)(i, j)}{(\mathbf{U}\mathbf{H}\mathbf{V}^T\mathbf{V}\mathbf{H}^T + \mathbf{U}\Gamma_U^+)(i, j)}}. \quad (37)$$

where Γ_U^+ and Γ_U^- is defined in Eq. 34 and 35.

If we go over the same process again for \mathbf{V} , we have the following updating rules, Therefore the final updating rule for \mathbf{U} is

$$\mathbf{V}(i, j) \leftarrow \mathbf{V}(i, j) \sqrt{\frac{(\mathbf{X}^T\mathbf{U}\mathbf{H} + \mathbf{V}\Gamma_V^-)(i, j)}{(\mathbf{V}\mathbf{H}^T\mathbf{U}^T\mathbf{U}\mathbf{H} + \mathbf{V}\Gamma_V^+)(i, j)}}. \quad (38)$$

where Γ_V^+ , Γ_V^- are defined similarly as in Eq. 35 (replace \mathbf{U} with \mathbf{V}), and Γ_V is defined as

$$\Gamma_V = \mathbf{V}^T\mathbf{X}^T\mathbf{U}\mathbf{H} + \mathbf{H}^T\mathbf{U}^T\mathbf{U}\mathbf{H}. \quad (39)$$

Choice of 2/3-factor MF How do we choose between 2-factor or 3-factor **MF** in real-world applications? A general principle is that: if we only need to place regularizations on one latent matrix, i.e. either \mathbf{U} or \mathbf{V} , then we can use 2-factor **MF**; if both \mathbf{U} and \mathbf{V} are to be regularized, either explicitly or implicitly, 3-factor **MF** might be a better choice.

3 Adapataions and Applications

MF has been used for a wide range of applications in social computing, including collaborative filtering (CF), link prediction (LP), sentiment analysis, etc. It can not only provide as a single model for matrix completeion or clutering, but also as a framework for solving almost all categories of prediction problems.

In this part we will extend **MF** to highly sparse cases. For the cases in which we have additional data, e.g. link data between users (in CF, or additional links in LP) or description data of users and items, we can incorporate different regularization techniques to enhance the matrix completion performance. Moreover, by properly manipulating latent factors derived from **MF**, we can adapt **MF** to (un-/semi-)supervised learning.

3.1 Sparse Matrix Completion

Here we address the problem of using **MF** for collborative filtering, link prediction and clustering. We start with a basic assumption, which makes the

previously introduced models unsuitable. This basic assumption is: high portion of the data is missing, i.e. data matrix is incomplete. Such assumption is very common in real-world cases [12].

The problem is solved by modeling directly the observed data. Eq. 1 is modified as follows:

$$\min_{\mathbf{U}, \mathbf{V}} \mathcal{O} = \|\mathbf{O} \circ (\mathbf{X} - \mathbf{UV}^T)\|_F^2 + \alpha \|\mathbf{U}\|_F^2 + \beta \|\mathbf{V}\|_F^2, \quad (40)$$

in which \mathbf{O} poses constraints on only these observed data entries, i.e. $\mathbf{O}(i, j) = 1$ if entry (i, j) is observed, and $\mathbf{O}(i, j) = 0$ otherwise.

In this case, the objective function is transformed as follows:

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}} \mathcal{O} = & Tr((\mathbf{O}^T \circ \mathbf{X}^T)(\mathbf{O} \circ \mathbf{X}) + (\mathbf{O}^T \circ \mathbf{VU}^T)(\mathbf{O} \circ \mathbf{UV}^T) \\ & - 2(\mathbf{O}^T \circ \mathbf{X}^T)(\mathbf{O} \circ \mathbf{UV}^T)) + \alpha Tr(\mathbf{U}^T \mathbf{U}) + \beta Tr(\mathbf{V}^T \mathbf{V}). \end{aligned} \quad (41)$$

And the gradients become:

$$\begin{aligned} \frac{\partial \mathcal{O}}{\partial \mathbf{U}} &= \frac{\partial Tr((\mathbf{O}^T \circ \mathbf{VU}^T)(\mathbf{O} \circ \mathbf{UV}^T) - 2(\mathbf{O}^T \circ \mathbf{X}^T)(\mathbf{O} \circ \mathbf{UV}^T)) + \alpha Tr(\mathbf{U}^T \mathbf{U})}{\partial \mathbf{U}} \\ &= \frac{\partial Tr(\mathbf{U}^T(\mathbf{O} \circ \mathbf{O} \circ \mathbf{UV}^T)\mathbf{V} - 2(\mathbf{O}^T \circ \mathbf{O}^T \circ \mathbf{X}^T)\mathbf{UV}^T) + \alpha Tr(\mathbf{U}^T \mathbf{U})}{\partial \mathbf{U}} \\ &= 2((\mathbf{O} \circ \mathbf{O} \circ \mathbf{UV}^T)\mathbf{V} - (\mathbf{O} \circ \mathbf{O} \circ \mathbf{X})\mathbf{V} + \alpha \mathbf{U}), \\ \frac{\partial \mathcal{O}}{\partial \mathbf{V}} &= 2((\mathbf{O}^T \circ \mathbf{O}^T \circ \mathbf{VU}^T)\mathbf{U} - (\mathbf{O}^T \circ \mathbf{O}^T \circ \mathbf{X}^T)\mathbf{U} + \beta \mathbf{V}). \end{aligned} \quad (42)$$

In the derivation above we use the following rule of Hadamard product:

$$Tr((\mathbf{O}^T \circ \mathbf{A}^T)(\mathbf{O} \circ \mathbf{A})) = Tr(\mathbf{A}^T(\mathbf{O} \circ \mathbf{O} \circ \mathbf{A})). \quad (43)$$

The updating rules for non-negative **MF** and orthogonal non-negative **MF** is straightforward: the methods of getting Λ, Γ are exactly the same as what we did in Theory Section. For updating rules of non-negative **MF** and orthogonal non-negative **MF**, the reader can refer to [7] and [8], respectively.

3.1.1 Calculating Memory Occupation

Note that the updating rules above are again purely matrix-wise – this is to be consistent with the style of this article. In matrix completion, however, sometimes the size of the data matrix is bigger than memory size, making stochastic gradient descent algorithm more suitable than the matrix-wise method.

The question here is, how do we calculate the size of a matrix to see if it fits to memory. Here is a easy way to make such a calculation. Assume we have a $10K \times 10K$ matrix, with each entry allocated a 32bit float (e.g. float32 in python), then the memory allocation for the whole matrix can be roughly calculated as

$$(10^4 \times 10^4 \times 4) / 10^6 = 400M.$$

So for a computer with 4G memory, we can fit a matrix $100K \times 10K$ matrix into memory. For a computer with 32G memory, we can fit a matrix of size $100K \times 80K$ ($10 \times 8 \times 400M = 32G$).

3.2 Enhanced Matrix Completion

We looked at **MF** with different *constraints*, e.g. non-negativity and orthogality, and one type of *regularization* which prevents the entries in low-rank matrices being too large. This subsection considers other kinds of *regularization* when external data source becomes available, i.e. goes beyond the data matrix **X**. Usually this is the real-world case, since most social media data contains rich data sources.

In this subsection we consider two types of regularization with corresponding additional data:

1. **self-regularization** when we have additional linked data between users (in CF, or additional link type in LP);
2. **2-sided regularization** when we have description data of users and items.

We further point to two publications [19] and [8], to demonstrate the above two types of regularization, respectively.

3.2.1 Enhancing Matrix Completion with Self-regularization

By self-regularization, we refer to the regularization of rows in low-rank matrix **U** or **V**. Assume now we are dealing with a LP problem, in which we would like to predict if a user trust another – trust relation are common in review sites like Epinions. Usually there exist another type of links between users, i.e. social relation. Can we use social relation to boost the performance of trust relation prediction? This is exactly the research question proposed in [19].

It turns out the answer is yes – as expected, users with social relation tend to share similar preferences. The basic idea to incorporate this into trust prediction is by adding the regularization term Eq. 44 into the general **MF** framework. In Eq. 44, ξ is the entries in the additional link matrix **Z** and **D** is the diagonal matrix with $\mathbf{D}(i, i) = \sum_{j=1}^m \mathcal{Z}_{j=1}^m(j, i)$, thus \mathcal{L} is the Laplacian matrix of **D**. It is interesting that, using trace operator, the regularization Eq. 44 become such simple.

Social relation is common in social computing, the similarity in people with social relation has a specific name in social theory - ‘homophily’, making this type of regularization applicable to a lot of social computing scenarios. If we generalize a bit, we may assume that many linked objects, not necessarily web users, have similarities, in terms of their entries of data matrix **X** that we would like to predict. For instance, while predicting the sentiment of articles, we may assume that articles authored by the same users tend to express similar sentiment, e.g. political reviewers expressing negative sentiment in their news

reviewing articles. We will see that this type of regularization is used in a sentiment analysis paper [10], which we will analyze later.

$$\begin{aligned}
& \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \xi(i, j) \|\mathbf{U}(i, :) - \mathbf{U}(j, :)\|_2^2 \\
&= \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \sum_{d=1}^k \xi(i, j) (\mathbf{U}(i, d) - \mathbf{U}(j, d))^2 \\
&= \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \sum_{d=1}^k \xi(i, j) (\mathbf{U}^2(i, d) - 2\mathbf{U}(i, d)\mathbf{U}(j, d) + \mathbf{U}^2(j, d)) \tag{44} \\
&= \sum_{i=1}^m \sum_{j=1}^m \sum_{d=1}^k \xi(i, j) \mathbf{U}^2(i, d) - \sum_{i=1}^m \sum_{j=1}^m \sum_{d=1}^k \xi(i, j) \mathbf{U}(i, d)\mathbf{U}(j, d) \\
&= \sum_{d=1}^k \mathbf{U}^T(:, d)(\mathbf{D} - \mathcal{Z})\mathbf{U}(:, d) \\
&= \text{Tr}(\mathbf{U}^T \mathcal{L} \mathbf{U})
\end{aligned}$$

Regularization and Sparseness More regularization sometimes can conquer the data sparsity problem, to some extent. On the other hand, modelling the error only on observed data entries, as what \mathbf{O} does in previous subsection, could be also very effective.

3.2.2 Enhancing Matrix Completion with 2-sided regularization

Here we consider placing regularization on both \mathbf{U} and \mathbf{V} together, which we call 2-sided regularization.

Before we start, we review orthogonal non-negative MF a bit. Orthogonality constraint in orthogonal non-negative MF is similar to a 2-sided regularization:

$$\text{Tr}(\Gamma_U^T (\mathbf{U}^T \mathbf{U} - \mathbf{I})), \text{Tr}(\Gamma_V^T (\mathbf{V}^T \mathbf{V} - \mathbf{I}))$$

are two equality constraints over low-rank matrices. Such equality needs to be strictly satisfied. **Regularization, differing from constraints, however can be viewed as a soft type of constraints:** it only needs to be satisfied to some extent, while constraints need to be strictly satisfied. This is the reason why we consider non-negativity and orthogonality constraints, while call homophily regularization.

Now let us turn our attention back to 2-sided regularization, basing the example from [8], which considers POI recommendation in location-based social network (LBSN). The first data we have is a check-in data \mathbf{X} that encodes the interaction between users and POI's. We are further given some description data \mathbf{A} of user interest, and \mathbf{B} of POI property, both in the form of word vectors.

Question here is, how do we make use of \mathbf{A} and \mathbf{B} to enhance the matrix completion problem for interacting matrix \mathbf{X} ?

Since we are coping with 2-sided regularization, we use 3-factor MF:

$$\min_{\mathbf{U}, \mathbf{H}, \mathbf{V}} \mathcal{O} = \|\mathbf{X} - \mathbf{U}\mathbf{H}\mathbf{V}^T\|_F^2 - Tr(\Lambda_U \mathbf{U}^T) - Tr(\Lambda_H \mathbf{H}^T) + R's. \quad (45)$$

The only thing here is, how to add the 2-sided regularization terms R 's, as we did for orthogonality constraints.

To utilize \mathbf{A} and \mathbf{B} , we assume that there are some connections between them, such that they can be used to regularize \mathbf{U} and \mathbf{V} . In the context of LBSN, we may assume that \mathbf{A} and \mathbf{B} have similar vocabulary, in which the words have similar latent space. Therefore we can approximate \mathbf{A} and \mathbf{B} with 2-factor MF:

$$\mathbf{A} \approx \mathbf{U}\mathbf{G}^T, \mathbf{B} \approx \mathbf{V}\mathbf{G}^{*T} \quad (46)$$

with connection

$$\|\mathbf{G} - \mathbf{G}^*\|_1 \approx 0. \quad (47)$$

Eq. 47 is important since it really connect \mathbf{U} with \mathbf{V} , forming a 2-sided regularization. The final objective function now becomes:

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{H}, \mathbf{V}} \mathcal{O} = & \|\mathbf{X} - \mathbf{U}\mathbf{H}\mathbf{V}^T\|_F^2 - Tr(\Lambda_U \mathbf{U}^T) - Tr(\Lambda_H \mathbf{H}^T) \\ & + \lambda_A \|\mathbf{A} - \mathbf{U}\mathbf{G}^T\|_F^2 + \lambda_B \|\mathbf{B} - \mathbf{V}\mathbf{G}^{*T}\|_F^2 + \delta \|\mathbf{G} - \mathbf{G}^*\|_1 \\ & + \alpha (\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2 + \|\mathbf{H}\|_F^2 + \|\mathbf{G}\|_F^2). \end{aligned} \quad (48)$$

The last line is to regularize in approximating \mathbf{A}, \mathbf{B} ; note that since here we use regularization, instead of constraints as in non-negative orthogonal MF, we can add regularization to $\mathbf{U}, \mathbf{V}, \mathbf{H}$.

Factorization vs. Regularization We remark here that the idea of co-factoring two matrices (\mathbf{X}, \mathbf{A}) with shared factors (\mathbf{U}) originates from collective matrix factorization [17], which has many applications in CF [16]. A interesting comparative study between collective factorization and self-regularization can be found in [20].

3.3 From Clustering to (Un-/Semi-)supervised Learning

Although different types of extra data sources can be used in enhanced MF, the purpose so far to remains be matrix completion. This subsection, however, considers other types of machine learning problems, i.e. (un-/semi-)supervised learning. The essential assumption of using MF for (un-/semi-)supervised learning is that the latent row(column) is or can be predictable for some dependent variables.

To make use of the predictability, we need mechanisms to connect the latent vectors to responses. Following are the two mechanisms:

1. **enforcement** directly enforce the latent space to be the response space;
2. **transformation** transform the latent space to response space. This is similar as what people do in machine learning.

We point to publications [10] and [6] for the demonstration of the above two methods, respectively.

3.3.1 Enforcing Latent Factor to be Response

In previous regularizations, we do not force the latent space to be interpretable space. For instance, in the 2-sided regularization, we do not specify the meaning of \mathbf{U} that is used in both \mathbf{X} and \mathbf{A} factorization. However, (un-/semi-)supervised learning requires the latent space to be interpretable. The method, still, is regularization.

[10] deals with the problem of sentiment analysis, for which the authors use 3-factor non-negative orthogonal **MF**. The input is a post-word matrix \mathbf{X} . In addition, we are given emotion indication in some of the posts. “The key idea of modeling post-level emotion indication is to make the sentiment polarity of a post as close as possible to the emotion indication of the post.”, formulated as

$$\mathbf{G}^u \|\mathbf{U} - \mathbf{U}_0\|_F^2,$$

in which $\mathbf{U} \in \mathbb{R}^{m \times 2}$ is the post-sentiment matrix, i.e. $\mathbf{U}(i, :) = (1, 0)$ representing that the i th post has a positive sentiment, and $\mathbf{U}_0 \in \mathbb{R}^{m \times 2}$ is the post-emotion indication matrix, i.e. $\mathbf{U}_0(i, :) = (1, 0)$ meaning the i th post contains positive emotion indication. Similar regularization is applied to \mathbf{V} as well.

Such an idea is quite simple, however it explicitly poses a notable question: is it computationally feasible that we strictly enforce the \mathbf{U}, \mathbf{V} to any pre-defined space, i.e. sentiment space in this case. Based on Proposition 1 in [5], we know that the answer is no. However, as we see in this sentiment analysis work [10], **regularization is always possible!**

In fact, the enforcement regularization that we see in this work is the most constrained regularization: it is 2-sided regularization for both \mathbf{U}, \mathbf{V} , and it is enforcement without any transformation coefficients. We will see next how to regularize for supervised learning by transformation.

3.3.2 Transforming Latent Factor to Response

As we pointed out, the essential idea of supervised learning is to transform the latent variables to some response variable. To see this, we study an example that exploit matrix factorization to boost (sparse) regression.

Here we solve the following optimization problem:

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V} \geq \mathbf{0}} \quad & \|\mathbf{X} - \mathbf{UV}^T\|_F^2 + \lambda \|\mathbf{O} \odot (\mathbf{UW}^T - \mathbf{Y})\|_F^2 \\ & + \lambda_X (\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2) + \lambda_Y \|\mathbf{W}\|_1. \end{aligned} \tag{49}$$

Optimizing the objective function accomplishes two goals simultaneously: 1) learning the latent factors; and, 2) predicting the dependent variables based on the learnt latent factors. As the learning of \mathbf{U} is guided by the prediction of \mathbf{Y} (proved later), the learned latent factors can be more predictive in the regression. Note that the parameter λ controls the relative importance between matrix factorization and regression – a larger λ indicates that the regression should dominate.

\mathbf{O} is a mask vector with the first n_{train} – the size of training set – entries equal to 1, and the other n_{test} – the size of test set – entries equal to 0. Correspondingly, \mathbf{X} contains both the training data in the first n_{train} rows and the test data, in the remaining n_{test} rows. \mathbf{Y} is also composed of two parts, the first n_{train} entries being the complexity values of the training tasks; the other entries can be any values, as they are not involved in model learning, which is controlled by the 0's in \mathbf{O} .

The Lagrangian function of the objective function is

$$L = \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 + \lambda\|\mathbf{O} \odot (\mathbf{U}\mathbf{W}^T - \mathbf{Y})\|_F^2 + \lambda_X(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2) + \lambda_Y\|\mathbf{W}\|_1 - Tr(\Lambda_U\mathbf{U}^T) - Tr(\Lambda_V\mathbf{V}^T). \quad (50)$$

The derivative of \mathbf{U} is:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{U}} &= \frac{\partial Tr(\mathbf{V}\mathbf{U}^T\mathbf{U}\mathbf{V}^T - 2\mathbf{X}^T\mathbf{U}\mathbf{V}^T)}{\partial \mathbf{U}} \\ &+ \frac{\partial \lambda Tr((\mathbf{O}^T \odot \mathbf{W}\mathbf{U}^T)(\mathbf{O} \odot \mathbf{U}\mathbf{W}^T) - 2(\mathbf{O}^T \odot \mathbf{Y}^T)(\mathbf{O} \odot \mathbf{U}\mathbf{W}^T))}{\partial \mathbf{U}} \\ &+ \frac{\partial \lambda_X Tr(\mathbf{U}^T\mathbf{U}) - Tr(\Lambda_U\mathbf{U}^T)}{\partial \mathbf{U}} \\ &= 2(\mathbf{U}\mathbf{V}^T\mathbf{V} - \mathbf{X}\mathbf{V} + \lambda(\mathbf{O} \odot (\mathbf{U}\mathbf{W}^T))\mathbf{W} - \lambda(\mathbf{O} \odot \mathbf{Y})\mathbf{W} + \lambda_X\mathbf{U}) \\ &- \Lambda_U. \end{aligned} \quad (51)$$

The derivative of \mathbf{V} is:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{V}} &= \frac{\partial Tr(\mathbf{V}\mathbf{U}^T\mathbf{U}\mathbf{V}^T - 2\mathbf{X}^T\mathbf{U}\mathbf{V}^T) + \lambda_X Tr(\mathbf{V}^T\mathbf{V}) - Tr(\Lambda_V\mathbf{V}^T)}{\partial \mathbf{V}} \\ &= 2(\mathbf{V}\mathbf{U}^T\mathbf{U} - \mathbf{X}^T\mathbf{U} + \lambda_X\mathbf{V}) - \Lambda_V. \end{aligned} \quad (52)$$

Note that for \mathbf{W} the problem becomes a classic Lasso problem, we can update it use standard algorithm such as LARS.

According to the KKT conditions:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{U}} = 0, \quad \frac{\partial L}{\partial \mathbf{V}} = 0, \\ \Lambda_U \odot \mathbf{U} = 0, \Lambda_V \odot \mathbf{V} = 0. \end{aligned} \quad (53)$$

We have

$$\begin{aligned} (\mathbf{U}\mathbf{V}^T\mathbf{V} - \mathbf{X}\mathbf{V} + \lambda(\mathbf{O} \odot (\mathbf{U}\mathbf{W}^T))\mathbf{W} \\ - \lambda(\mathbf{O} \odot \mathbf{Y})\mathbf{W} + \lambda_X\mathbf{U}) \odot \mathbf{U} = 0, \\ (\mathbf{V}\mathbf{U}^T\mathbf{U} - \mathbf{X}^T\mathbf{U} + \lambda_X\mathbf{V}) \odot \mathbf{V} = 0. \end{aligned} \quad (54)$$

It leads to the following updating rules for \mathbf{U}, \mathbf{V} :

$$\begin{aligned}\mathbf{U}(i, j) &\leftarrow \mathbf{U}(i, j) \sqrt{\frac{(\mathbf{X}\mathbf{V} + \lambda(\mathbf{O} \odot \mathbf{Y})\mathbf{W})(i, j)}{(\mathbf{U}\mathbf{V}^T\mathbf{V} + \lambda(\mathbf{O} \odot (\mathbf{U}\mathbf{W}^T))\mathbf{W} + \lambda_X\mathbf{U})(i, j)}}, \\ \mathbf{V}(i, j) &\leftarrow \mathbf{V}(i, j) \sqrt{\frac{(\mathbf{X}^T\mathbf{U})(i, j)}{(\mathbf{V}\mathbf{U}^T\mathbf{U} + \lambda_X\mathbf{V})(i, j)}}.\end{aligned}\tag{55}$$

3.3.3 A Comprehensive Model

Here we review an application of [6] that integrates the methods of enforcement and transformation. In this application, we would like to model a user's attitude towards some controversial topic, reflected by his opinion, sentiment and retweeting action. We are given a retweeting matrix \mathbf{X} representing users' retweeting action to some tweets, and we would like to predict users' opinion \mathbf{O} and sentiment \mathbf{P} , and the task is to predict these three variables given the user feature \mathbf{F} .

We first introduce how the model is built in [6], then discuss other alternatives. To train such a model, the authors propose the following model

$$\begin{aligned}\min_{\mathbf{W}, \mathbf{V}} \mathcal{O} &= \|\mathbf{X} - (\mathbf{F}\mathbf{W}^T)\mathbf{V}^T\|_F^2 + \lambda_1\|\mathbf{F}\mathbf{W}^T - \mathbf{O}\|_F^2 + \lambda_2\|(\mathbf{F}\mathbf{W}^T)\mathbf{S} - \mathbf{P}\|_F^2 \\ &+ \lambda_3\|\mathbf{W}\|_1 + \alpha\|\mathbf{W}\|_F^2 + \beta\|\mathbf{V}\|_F^2 + \gamma\|\mathbf{S}\|_F^2 - Tr(\Lambda_1\mathbf{U}^T) - Tr(\Lambda_2\mathbf{V}^T),\end{aligned}\tag{56}$$

in which $\lambda_1\|\mathbf{F}\mathbf{W}^T - \mathbf{O}\|_F^2$ and $\lambda_3\|\mathbf{W}\|_1$ models opinion from the user feature by bringing in the classical linear regression. We can see that modeling the sentiment is also straightforward: $\lambda_2\|(\mathbf{F}\mathbf{W}^T)\mathbf{S} - \mathbf{P}\|_F^2$ simply transfers again the user feature with a linear transformation \mathbf{S} . The retweeting matrix \mathbf{X} , similarly, also using $\mathbf{F}\mathbf{W}^T$ as the latent vectors.

To summarize, the model Eq. 56 bases the prediction of retweeting action, opinion and sentiment all on the user features. If we make λ_1 to be infinitely large, meaning that we enforce $\mathbf{F}\mathbf{W}^T = \mathbf{O}$, then in fact, $\mathbf{X} \approx \mathbf{O}\mathbf{V}^T$ and $\mathbf{O}\mathbf{S} \approx \mathbf{P}$. Such choice is based on the assumption that opinion drives both the retweeting action and sentiment.

Model Eq. 56 is an comprehensive model, in the sense that the subtask of matrix completion, clustering and regression are fused together, by basing all prediction on user feature transformation. What if we are not given the user feature information? Instead, we directly model the relation between retweeting action, opinion and sentiment. A straightforward model could be

$$\begin{aligned}\min_{\mathbf{U}, \mathbf{V}} \mathcal{O} &= \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 + \lambda_1\|\mathbf{U} - \mathbf{O}\|_F^2 + \lambda_2\|\mathbf{U}\mathbf{S} - \mathbf{P}\|_F^2 \\ &+ \alpha\|\mathbf{U}\|_F^2 + \beta\|\mathbf{V}\|_F^2 + \gamma\|\mathbf{S}\|_F^2 - Tr(\Lambda_1\mathbf{U}^T) - Tr(\Lambda_2\mathbf{V}^T).\end{aligned}\tag{57}$$

References

- [1] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2009.
- [2] Mike Brookes. The matrix reference manual. *Imperial College London*, 2005.
- [3] Chris Ding, Xiaofeng He, and Horst D Simon. On the equivalence of non-negative matrix factorization and spectral clustering. In *SDM'05*, pages 606–610. SIAM, 2005.
- [4] Chris Ding, Tao Li, and Michael I Jordan. Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding. In *ICDM'08*, pages 183–192. IEEE, 2008.
- [5] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *KDD'06*, pages 126–135. ACM, 2006.
- [6] Huiji Gao, Jalal Mahmud, Jilin Chen, Jeffrey Nichols, and Michelle Zhou. Modeling user attitude toward controversial topics in online social media. In *ICWSM'14*, 2014.
- [7] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Exploring temporal effects for location recommendation on location-based social networks. In *RecSys'13*, pages 93–100. ACM, 2013.
- [8] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Content-aware point of interest recommendation on location-based social networks. In *AAAI'15*, 2015.
- [9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer, 2009.
- [10] Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. Unsupervised sentiment analysis with emotional signals. In *WWW'13*, pages 607–618. ACM, 2013.
- [11] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD'08*, pages 426–434. ACM, 2008.
- [12] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
- [13] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.
- [14] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS'01*, pages 556–562, 2001.

- [15] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *NIPS'07*, pages 1257–1264, 2007.
- [16] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):3, 2014.
- [17] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *KDD'08*, pages 650–658. ACM, 2008.
- [18] Gábor Takács, István Pilászy, Botyán Németh, and Domonkos Tikk. Matrix factorization and neighbor based algorithms for the netflix prize problem. In *RecSys'08*, pages 267–274. ACM, 2008.
- [19] Jiliang Tang, Huiji Gao, Xia Hu, and Huan Liu. Exploiting homophily effect for trust prediction. In *WSDM'13*, pages 53–62. ACM, 2013.
- [20] Quan Yuan, Li Chen, and Shiwan Zhao. Factorization vs. regularization: fusing heterogeneous social relationships in top-n recommendation. In *Recsys'11*, pages 245–252. ACM, 2011.