Versatile Sparse Matrix Factorization and Its Applications in High-Dimensional Biological Data Analysis

Yifeng Li and Alioune Ngom

School of Computer Science, University of Windsor,
401 Sunset Avenue, Windsor, Ontario, N9B 3P4, Canada
{li11112c,angom}@uwindsor.ca

Abstract. Non-negative matrix factorization and sparse representation models have been successfully applied in high-throughput biological data analysis. In this paper, we propose our *versatile sparse matrix factorization* (VSMF) model for biological data mining. We show that many well-known sparse models are specific cases of VSMF. Through tuning parameters, sparsity, smoothness, and non-negativity can be easily controlled in VSMF. Our computational experiments corroborate the advantages of VSMF.

Keywords: versatile sparse matrix factorization, non-negative matrix factorization, sparse representation,feature extraction, feature selection, biological processes identification.

1 Introduction

Non-negative matrix factorization (NMF) [10] and the wider concept – *sparse representation* (SR) [6] are sparse matrix factorization models that decompose a matrix into a basis matrix and coefficient matrix. They have been applied in many fields of bioinformatics including clustering [3] and biclustering [4], sample prediction [15], biological process identification [9], and transcriptional regulatory network inference [16]. Many variants of NMF and SR have been invented for various situations. Semi-NMF is proposed in [5] for data of mixed signs. Sparse NMF is introduced to guarantee sparse results [8]. We propose kernel NMF in [13] to deal with nonlinearity in microarray data. Kernel NMF also works for relational data. Negative values are allowed in the coefficient matrix of l_1-regularized sparse representation (l_1-SR) models [15]. However, the following challenges have not been well addressed. First, an unified model is very necessary for these variants from both theoretical and practical perspectives. Second, sparsity is usually constrained on the coefficient matrix and the sparsity of basis matrix is not guaranteed in most sparse models. Third, l_1-norm is the most popular way to induce sparsity. However, it does not guarantee that a group of correlated variables can be selected or discarded simultaneously.

In this paper, in order to address these challenges, we propose our *versatile sparse matrix factorization* (VSMF) model. The contributions of this study includes

A. Ngom et al. (Eds.): PRIB 2013, LNBI 7986, pp. 91–101, 2013.
© Springer-Verlag Berlin Heidelberg 2013

1. With its six parameters, VSMF can easily control sparsity, smoothness, and non-negativity on both basis matrix and coefficient matrix. VSMF is thus a generic model. The standard NMF, semi-NMF, sparse NMF, kernel NMF and l_1-SR models are specific cases of VSMF.
2. We devise multiplicative update rules and active-set algorithms for the optimization of VSMF. Analytical solutions, which are useful for kernelization, are also discussed.
3. We demonstrate the usefulness of VSMF in bioinformatics.

The rest of this paper is organized as follows: We first summarize the variants of NMF or SR models in Section 2. Next, we present our VSMF model and its optimization in Section 3. After that, several biological applications of VSMF are demonstrated in Section 4. Finally, we draw our conclusions and mention future works.

2 Related Work

Hereafter, we use the following notations. The training set is denoted by $[\boldsymbol{d}_1, \cdots, \boldsymbol{d}_n] = \boldsymbol{D} \in \mathbb{R}^{m \times n}$, where m and n are the numbers of features and samples respectively. The basis matrix is represented as $[\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k] = \boldsymbol{A} \in \mathbb{R}^{m \times k}$, where $k < \min\{m, n\}$ is the number of basis vectors (or factors). The coefficient matrix is denoted by $[\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n] = \boldsymbol{Y} \in \mathbb{R}^{k \times n}$. Given $\boldsymbol{D}$, the task of sparse matrix factorization is to find $\boldsymbol{A}$ and $\boldsymbol{Y}$ such that $\boldsymbol{D} \approx \boldsymbol{AY}$, where at least one factors among $\boldsymbol{A}$ and $\boldsymbol{Y}$ should be sparse.

For the convenience of discussion, we summarize the existing sparse matrix factorization models in Table 1. It is impossible to enumerate all existing works in this direction, therefore all models mentioned in this table are the most representative ones. The training data $\boldsymbol{D}$ must be non-negative for the standard NMF and sparse NMF. For sparse NMF, α and λ are two non-negative parameters. For kernel NMF and l_1-SR, $\phi(\cdot)$ is a function that maps the training samples into a high-dimensional feature space. $\phi(\boldsymbol{D})$ is the training samples in this feature space. $\boldsymbol{A}_\phi$ is the basis matrix in this feature space.

Table 1. The Existing NMF and SR Models

NMF/SR	Equations
Standard NMF [10]	$\min_{\boldsymbol{A},\boldsymbol{Y}} \frac{1}{2}\|\boldsymbol{D} - \boldsymbol{AY}\|_F^2$ s.t. $\boldsymbol{A}, \boldsymbol{Y} \geq 0$
Semi-NMF [5]	$\min_{\boldsymbol{A},\boldsymbol{Y}} \frac{1}{2}\|\boldsymbol{D} - \boldsymbol{AY}\|_F^2$ s.t. $\boldsymbol{Y} \geq 0$
Sparse NMF [8]	$\min_{\boldsymbol{A},\boldsymbol{Y}} \frac{1}{2}\|\boldsymbol{D} - \boldsymbol{AY}\|_F^2 + \frac{\alpha}{2}\sum_{i=1}^{k}\|\boldsymbol{a}_i\|_2^2 + \frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{y}_i\|_1^2$ s.t. $\boldsymbol{A}, \boldsymbol{Y} \geq 0$
Kernel NMF [13, 15]	$\min_{\boldsymbol{A}_\phi,\boldsymbol{Y}} \frac{1}{2}\|\phi(\boldsymbol{D}) - \boldsymbol{A}_\phi\boldsymbol{Y}\|_F^2 + \frac{\alpha}{2}\sum_{i=1}^{k}\|\phi(\boldsymbol{a}_i)\|_2^2 + \frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{y}_i\|_1$ s.t. $\boldsymbol{Y} \geq 0$
l_1-SR [15]	$\min_{\boldsymbol{A}_\phi,\boldsymbol{Y}} \frac{1}{2}\|\phi(\boldsymbol{D}) - \boldsymbol{A}_\phi\boldsymbol{Y}\|_F^2 + \frac{\alpha}{2}\sum_{i=1}^{k}\|\phi(\boldsymbol{a}_i)\|_2^2 + \frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{y}_i\|_1$

3 Method

In this section, we first present our versatile sparse matrix factorization model. Then, we give optimization algorithms for this model.

3.1 The Versatile Sparse Matrix Factorization Model

Our *versatile sparse matrix factorization* (VSMF) model can be expressed in the following equation:

$$\begin{aligned}\min_{\boldsymbol{A},\boldsymbol{Y}} f(\boldsymbol{A},\boldsymbol{Y}) &= \frac{1}{2}\|\boldsymbol{D}-\boldsymbol{A}\boldsymbol{Y}\|_F^2 + \sum_{i=1}^{k}\left(\frac{\alpha_2}{2}\|\boldsymbol{a}_i\|_2^2 + \alpha_1\|\boldsymbol{a}_i\|_1\right) \\ &+ \sum_{i=1}^{n}\left(\frac{\lambda_2}{2}\|\boldsymbol{y}_i\|_2^2 + \lambda_1\|\boldsymbol{y}_i\|_1\right) \\ \text{s.t.} &\begin{cases}\text{if } t_1 = 1 & \boldsymbol{A} \geq 0 \\ \text{if } t_2 = 1 & \boldsymbol{Y} \geq 0\end{cases},\end{aligned} \tag{1}$$

where parameter $\alpha_1 \geq 0$ controls the sparsity of the basis vectors, parameter $\alpha_2 \geq 0$ controls the smoothness and scale of the basis vectors, parameter $\lambda_1 \geq 0$ controls the sparsity of the coefficient vectors, parameter $\lambda_2 \geq 0$ controls the smoothness of the coefficient vectors, parameters t_1 and t_2 are boolean variables (0: false, 1: true) that indicate if non-negativity should be enforced on $\boldsymbol{A}$ and $\boldsymbol{Y}$, respectively.

One advantage of VSMF is that both l_1 and l_2-norms can be used on both basis matrix and coefficient matrix. In VSMF, l_1-norms are used to induce sparse basis vectors and coefficient vectors. However, the drawback of l_1-norm is that correlated variables may not be simultaneously non-zero in the induced sparse result. This is because l_1-norm is able to produce sparse but non-smooth result. It is known that l_2-norm is able to obtain smooth but not sparse result. Combining both norms has been proven that correlated variables can be selected or removed simultaneously [18]. In addition to the smoothness of l_2-norm, another benefit of l_2-norm is that the scale of each vector can be restricted. This can avoid the scale interchange between the basis matrix and coefficient matrix. Another advantage of VSMF is that the non-negativity constraint can be switched off/on for either basis matrix or coefficient matrix. If the training data are non-negative, it is usually necessary that the basis matrix should be non-negative as well. In some situations, non-negativity is also needed on the coefficient matrix for better performance and better interpretation.

It can be easily seen that the standard NMF, semi-NMF, and sparse-NMFs are special cases of VSMF. If $\alpha_1 = \alpha_2 = \lambda_1 = \lambda_2 = 0$ and $t_1 = t_2 = 1$, VSMF is reduced to the standard NMF proposed in [10]. If $\alpha_1 = \alpha_2 = \lambda_1 = \lambda_2 = 0$ and $t_1 = 0$ and $t_2 = 1$, then VSMF becomes semi-NMF proposed in [5]. If $\alpha_1 = \lambda_2 = 0$, $\alpha_2, \lambda_1 \neq 0$, and $t_1 = t_2 = 1$, then VSMF is equivalent to the sparse-NMF proposed in [8]. When α_1 is set to zero, VSMF can be kernelized [15].

Sparse matrix factorization is a low-rank approximation problem. The number of ranks, that is k, is crucial for good performance of an analysis. Selecting k is still an open problem in both statistical inference and machine learning. We propose an adaptive rank selection method for VSMF. We base our idea on the sparsity of columns of $\boldsymbol{A}$ and $\boldsymbol{Y}$. We first set k to a relatively large integer. During the optimization of VSMF, if a column of $\boldsymbol{A}$ or a row of $\boldsymbol{Y}$ is null due to the sparsity controlled by the corresponding parameters, then both of the column of $\boldsymbol{A}$ and the row of $\boldsymbol{Y}$ corresponding to this null

factor are removed. Therefore k is reduced. When the optimization terminates, we can obtain the correct k corresponding to the current sparsity controlling parameters.

3.2 Optimization

Like most of NMF and SR models, the optimization of VSMF is non-convex. The most popular scheme to optimize these models are the block-coordinate descent method [2]. The basic idea of this scheme is in the following. $\boldsymbol{A}$ and $\boldsymbol{Y}$ are updated iteratively and alternatingly. In each iteration, $\boldsymbol{A}$ is updated while keeping $\boldsymbol{Y}$ fixed; then $\boldsymbol{A}$ is fixed and $\boldsymbol{Y}$ is updated. Based on this scheme, we devise the multiplicative update rules and active-set algorithms for VSMF. These two algorithms are given below.

Multiplicative Update Rules. If both $\boldsymbol{A}$ and $\boldsymbol{Y}$ are non-negative, we can equivalently rewrite $f(\boldsymbol{A}, \boldsymbol{Y})$ in Equation (1) to

$$\frac{1}{2}\|\boldsymbol{D} - \boldsymbol{A}\boldsymbol{Y}\|_F^2 + \frac{\alpha_2}{2}\mathrm{tr}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}) + \alpha_1\mathrm{tr}(\boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{A}) + \frac{\lambda_2}{2}\mathrm{tr}(\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}) + \lambda_1\mathrm{tr}(\boldsymbol{E}_2^{\mathrm{T}}\boldsymbol{Y}), \quad (2)$$

where $\boldsymbol{E}_1 \in \{1\}^{m\times k}$, and $\boldsymbol{E}_2 \in \{1\}^{k\times n}$. Fixing $\boldsymbol{A}$, updating $\boldsymbol{Y}$ can hence be expressed as

$$\begin{aligned} &\min_{\boldsymbol{Y}} f(\boldsymbol{Y}) = \frac{1}{2}\|\boldsymbol{D} - \boldsymbol{A}\boldsymbol{Y}\|_F^2 + \frac{\lambda_2}{2}\mathrm{tr}(\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}) + \lambda_1\mathrm{tr}(\boldsymbol{E}_2^{\mathrm{T}}\boldsymbol{Y}) \\ &\text{s.t. } \boldsymbol{Y} \geq 0. \end{aligned} \quad (3)$$

Similarly, Fixing $\boldsymbol{Y}$, updating $\boldsymbol{A}$ can be expressed as

$$\begin{aligned} &\min_{\boldsymbol{A}} f(\boldsymbol{A}) = \frac{1}{2}\|\boldsymbol{D} - \boldsymbol{A}\boldsymbol{Y}\|_F^2 + \frac{\alpha_2}{2}\mathrm{tr}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}) + \alpha_1\mathrm{tr}(\boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{A}) \\ &\text{s.t. } \boldsymbol{A} \geq 0. \end{aligned} \quad (4)$$

We design the following multiplicative update rules for VSMF model in the case of $t_1 = t_2 = 1$:

$$\begin{cases} \boldsymbol{A} = \boldsymbol{A} * \frac{\boldsymbol{D}\boldsymbol{Y}^{\mathrm{T}}}{\boldsymbol{A}\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} + \alpha_2\boldsymbol{A} + \alpha_1} \\ \boldsymbol{Y} = \boldsymbol{Y} * \frac{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{D}}{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Y} + \lambda_2\boldsymbol{Y} + \lambda_1} \end{cases}, \quad (5)$$

where $\boldsymbol{A} * \boldsymbol{B}$ and $\frac{\boldsymbol{A}}{\boldsymbol{B}}$ are element-wise multiplication and division between matrix $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively. This algorithm is a gradient-descent based method. Both rules are derived in the following.

For Equation (3), the first-order update rule of $\boldsymbol{Y}$ should be generally

$$\boldsymbol{Y} = \boldsymbol{Y} - \boldsymbol{\eta}_2 * \frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{Y}}. \quad (6)$$

where matrix $\boldsymbol{\eta}_2$ is step. We take the derivative of $f(\boldsymbol{Y})$, in Equation (3), with respect to $\boldsymbol{Y}$:

$$\frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{Y}} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Y} - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{D} + \lambda_2\boldsymbol{Y} + \lambda_1\boldsymbol{E}_2. \quad (7)$$

And we let the step $\boldsymbol{\eta}_2$ to be

$$\boldsymbol{\eta}_2 = \frac{\boldsymbol{Y}}{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Y} + \lambda_2\boldsymbol{Y} + \lambda_1\boldsymbol{E}_2}. \tag{8}$$

Substituting Equations (7) and (8) into Equation (6), we have

$$\boldsymbol{Y} = \boldsymbol{Y} * \frac{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{D}}{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Y} + \lambda_2\boldsymbol{Y} + \lambda_1\boldsymbol{E}_2}. \tag{9}$$

Similarly, for Equation (4), the first-order update rule of $\boldsymbol{A}$ should be generally

$$\boldsymbol{A} = \boldsymbol{A} - \boldsymbol{\eta}_1 * \frac{\partial f(\boldsymbol{A})}{\partial \boldsymbol{A}}. \tag{10}$$

We take the derivative of $f(\boldsymbol{A})$, in Equation (4), with respect to $\boldsymbol{A}$:

$$\frac{\partial f(\boldsymbol{A})}{\partial \boldsymbol{A}} = \boldsymbol{A}\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} - \boldsymbol{D}^{\mathrm{T}}\boldsymbol{Y} + \alpha_2\boldsymbol{A} + \alpha_1\boldsymbol{E}_1. \tag{11}$$

And we let the step to be

$$\boldsymbol{\eta}_1 = \frac{\boldsymbol{A}}{\boldsymbol{A}\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} + \alpha_2\boldsymbol{A} + \alpha_1\boldsymbol{E}_1}. \tag{12}$$

Substituting Equations (11) and (12) into Equation (10), we have

$$\boldsymbol{A} = \boldsymbol{A} * \frac{\boldsymbol{D}\boldsymbol{Y}^{\mathrm{T}}}{\boldsymbol{A}\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} + \alpha_2\boldsymbol{A} + \alpha_1}. \tag{13}$$

If we let $\alpha_1 = \alpha_2 = \lambda_1 = \lambda_2 = 0$, then the update rules in Equations (5) becomes the update rules of the standard NMF [11]. We can find that enforcing sparsity and smoothness on both basis matrix and coefficient matrix does not increase the time-complexity.

Active-Set Quadratic Programming. The multiplicative update rules above only works under the condition that both $\boldsymbol{A}$ and $\boldsymbol{Y}$ are non-negative. We devise active-set algorithms which allow us to relax the non-negativity constraints. We now show that when $t_1($ or $t_2) = 1$, $\boldsymbol{A}$ (or $\boldsymbol{Y}$) can be updated by our active-set *non-negative quadratic programming* (NNQP) algorithm; when $t_1($ or $t_2) = 0$, $\boldsymbol{A}$ (or $\boldsymbol{Y}$) can be updated by our active-set l_1*-regularized QP* (1_1QP) algorithm.

If $t_2 = 1$, the objective in Equation (3) can be rewritten as:

$$\begin{aligned} f(\boldsymbol{Y}) &= \mathrm{tr}(\frac{1}{2}\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Y} + \frac{1}{2}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{D} - \boldsymbol{D}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Y} + \frac{\lambda_2}{2}\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y} + \lambda_1\boldsymbol{E}_2^{\mathrm{T}}\boldsymbol{Y}) \\ &= \mathrm{tr}(\frac{1}{2}\boldsymbol{Y}^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda_2\boldsymbol{I})\boldsymbol{Y} + (\lambda_1\boldsymbol{E}_2^{\mathrm{T}} - \boldsymbol{D}^{\mathrm{T}}\boldsymbol{A})\boldsymbol{Y} + \frac{1}{2}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{D}) \\ &= \sum_{i=1}^{n} \frac{1}{2}\boldsymbol{y}_i^{\mathrm{T}}\boldsymbol{H}_2\boldsymbol{y}_i + \boldsymbol{g}_{(2)i}^{\mathrm{T}}\boldsymbol{y}_i + \frac{1}{2}\boldsymbol{d}_i^{\mathrm{T}}\boldsymbol{d}_i, \end{aligned} \tag{14}$$

where $\boldsymbol{H}_2 = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda_2\boldsymbol{I}$, and $\boldsymbol{g}_{(2)i} = \lambda_1 - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{d}_i$ and $\boldsymbol{G}_{(2)} = \lambda_1 - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{D}$. Therefore, we can see that updating non-negative $\boldsymbol{Y}$ is multiple NNPQ problem. We proposed a parallel active-set algorithm for NNQP in [15]. This algorithm can be used to solve the problem in Equation (14).

If $t_2 = 0$, the objective of updating $\boldsymbol{Y}$ can be reformulated as:

$$\begin{aligned} f(\boldsymbol{Y}) &= \mathrm{tr}(\frac{1}{2}\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Y} + \frac{1}{2}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{D} - \boldsymbol{D}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Y} + \frac{\lambda_2}{2}\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y}) + \lambda_1\|\boldsymbol{Y}\|_1 \\ &= \mathrm{tr}(\frac{1}{2}\boldsymbol{Y}^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda_2\boldsymbol{I})\boldsymbol{Y} + (-\boldsymbol{D}^{\mathrm{T}}\boldsymbol{A})\boldsymbol{Y} + \frac{1}{2}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{D}) + \lambda_1\|\boldsymbol{Y}\|_1 \\ &= \sum_{i=1}^{n} \frac{1}{2}\boldsymbol{y}_i^{\mathrm{T}}\boldsymbol{H}_2\boldsymbol{y}_i + \boldsymbol{g}_{(2)i}^{\mathrm{T}}\boldsymbol{y}_i + \lambda_1\|\boldsymbol{y}_i\|_1 + \frac{1}{2}\boldsymbol{d}_i^{\mathrm{T}}\boldsymbol{d}_i, \end{aligned} \tag{15}$$

where $\boldsymbol{H}_2 = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda_2\boldsymbol{I}$, and $\boldsymbol{g}_{(2)i} = -\boldsymbol{A}^{\mathrm{T}}\boldsymbol{d}_i$ and $\boldsymbol{G}_{(2)} = -\boldsymbol{A}^{\mathrm{T}}\boldsymbol{D}$. This is a l_1QP problem which can be solved by our active-set l_1QP algorithm proposed in [15].

Similarly, if $t_1 = 1$, $f(\boldsymbol{A})$ in Equation (3) can be expressed as

$$\begin{aligned} f(\boldsymbol{A}) &= \mathrm{tr}(\frac{1}{2}\boldsymbol{A}\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}} + \frac{1}{2}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{D} - \boldsymbol{D}\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}} + \frac{\alpha_2}{2}\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} + \alpha_1\boldsymbol{E}_1^{\mathrm{T}}\boldsymbol{A}) \\ &= \mathrm{tr}(\frac{1}{2}\boldsymbol{A}(\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} + \alpha_2\boldsymbol{I})\boldsymbol{A}^{\mathrm{T}} + (\alpha_1\boldsymbol{E}_1^{\mathrm{T}} - \boldsymbol{D}\boldsymbol{Y}^{\mathrm{T}})\boldsymbol{A}^{\mathrm{T}} + \frac{1}{2}\boldsymbol{D}\boldsymbol{D}^{\mathrm{T}}) \\ &= \sum_{i=1}^{m} \frac{1}{2}\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{H}_{\boldsymbol{1}}\boldsymbol{w}_i + \boldsymbol{g}_{(1)i}^{\mathrm{T}}\boldsymbol{w}_i + \frac{1}{2}\boldsymbol{D}_{i,:}(\boldsymbol{D}^{\mathrm{T}})_{:,i}, \end{aligned} \tag{16}$$

where $\boldsymbol{W} = \boldsymbol{A}^{\mathrm{T}}$, $\boldsymbol{H}_{\boldsymbol{1}} = \boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} + \alpha_2\boldsymbol{I}$, $\boldsymbol{g}_{(1)i} = \alpha_1 - \boldsymbol{Y}(\boldsymbol{D}^{\mathrm{T}})_{:,i}$ and $\boldsymbol{G}_{(1)} = \alpha_1 - \boldsymbol{Y}\boldsymbol{D}^{\mathrm{T}}$. Again, it can be seen that this problem is also a NNQP problem.

If $t_1 = 0$, the objective of updating $\boldsymbol{A}$ can be written as

$$\begin{aligned} f(\boldsymbol{A}) &= \mathrm{tr}(\frac{1}{2}\boldsymbol{A}\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}} + \frac{1}{2}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{D} - \boldsymbol{D}\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}} + \frac{\alpha_2}{2}\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}) + \alpha_1\|\boldsymbol{A}\|_1 \\ &= \mathrm{tr}(\frac{1}{2}\boldsymbol{A}(\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} + \alpha_2\boldsymbol{I})\boldsymbol{A}^{\mathrm{T}} + (-\boldsymbol{D}\boldsymbol{Y}^{\mathrm{T}})\boldsymbol{A}^{\mathrm{T}} + \frac{1}{2}\boldsymbol{D}\boldsymbol{D}^{\mathrm{T}}) + \alpha_1\|\boldsymbol{A}^{\mathrm{T}}\|_1 \\ &= \sum_{i=1}^{m} \frac{1}{2}\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{H}_{\boldsymbol{1}}\boldsymbol{w}_i + \boldsymbol{g}_{(1)i}^{\mathrm{T}}\boldsymbol{w}_i + \alpha_1\|\boldsymbol{w}_i\|_1 + \frac{1}{2}\boldsymbol{D}_{i,:}(\boldsymbol{D}^{\mathrm{T}})_{:,i}, \end{aligned} \tag{17}$$

where $\boldsymbol{W} = \boldsymbol{A}^{\mathrm{T}}$, $\boldsymbol{H}_{\boldsymbol{1}} = \boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} + \alpha_2\boldsymbol{I}$, $\boldsymbol{g}_{(1)i} = -\boldsymbol{Y}(\boldsymbol{D}^{\mathrm{T}})_{:,i}$ and $\boldsymbol{G}_{(1)} = -\boldsymbol{Y}\boldsymbol{D}^{\mathrm{T}}$. This is also a l_1QP problem that can be solved by our active-set l_1QP algorithm [15].

Analytical Solutions and Kernelization. If $t_2 = 0$ and $\lambda_1 = 0$, from $\frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{Y}} = 0$, $\boldsymbol{Y}$ can be updated analytically:

$$\boldsymbol{Y} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \lambda_2\boldsymbol{I})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{D} = \boldsymbol{A}^{\ddagger}\boldsymbol{D}. \tag{18}$$

From the previous section, we can see that only $\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}$ and $\boldsymbol{Y}\boldsymbol{D}^{\mathrm{T}}$ are required to update $\boldsymbol{A}$. According to Equation (18), $\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}$ and $\boldsymbol{Y}\boldsymbol{D}^{\mathrm{T}}$ can be expressed as

$$\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} = \boldsymbol{A}^{\ddagger}\boldsymbol{D}\boldsymbol{D}^{\mathrm{T}}(\boldsymbol{A}^{\ddagger})^{\mathrm{T}}. \tag{19}$$

$$\boldsymbol{Y}\boldsymbol{D}^{\mathrm{T}} = \boldsymbol{A}^{\ddagger}\boldsymbol{D}\boldsymbol{D}^{\mathrm{T}}. \tag{20}$$

We can see that in this situation, updating $\boldsymbol{A}$ only requires the previous value of $\boldsymbol{A}$ and the inner products of rows of $\boldsymbol{D}$.

Similarly, if $t_1 = 0$ and $\alpha_1 = 0$, $\boldsymbol{A}$ can be updated analytically:

$$\boldsymbol{A} = \boldsymbol{D}\boldsymbol{Y}^{\ddagger}, \tag{21}$$

where $\boldsymbol{Y}^{\ddagger} = \boldsymbol{Y}^{\mathrm{T}}(\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}} + \alpha_2\boldsymbol{I})^{-1}$. From the previous section, we know that updating $\boldsymbol{Y}$ only requires the inner products $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ and $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{D}$. They can be updated by the following equations:

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = (\boldsymbol{Y}^{\ddagger})^{\mathrm{T}}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{Y}^{\ddagger}. \tag{22}$$

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{D} = (\boldsymbol{Y}^{\ddagger})^{\mathrm{T}}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{D}. \tag{23}$$

Due to the analytical solution of $\boldsymbol{A}$, updating $\boldsymbol{Y}$ only requires the previous value of $\boldsymbol{Y}$ and the inner products of columns of $\boldsymbol{D}$.

These analytical solutions have two advantages. First, the corresponding matrix can be easily updated without resorting to any numerical solver. Second, we can see that only inner products are needed to update $\boldsymbol{Y}$ (or $\boldsymbol{A}$), when $\boldsymbol{A}$ (or $\boldsymbol{Y}$) can be analytically obtained. Using this property, we can obtain the kernel version of VSMF, which are described in the following. In sparse representation, at least one matrix among $\boldsymbol{A}$ and $\boldsymbol{Y}$ must be sparse. That is the analytical solutions in Equations (18) and (21) can not be used simultaneously. In practice, if each column of the training data $\boldsymbol{D}$ is the object to be mapped in high-dimensional feature space, we can analytically update $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ (or the corresponding kernel version $k(\boldsymbol{A}, \boldsymbol{A}) = (\phi(\boldsymbol{A}))^{\mathrm{T}}\phi(\boldsymbol{A})$ where $k(\cdot, \cdot)$ is a kernel function corresponding to $\phi(\cdot)$) and $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{D}$ (or $k(\boldsymbol{A}, \boldsymbol{D}) = (\phi(\boldsymbol{A}))^{\mathrm{T}}\phi(\boldsymbol{D})$), and then update $\boldsymbol{Y}$ via a numerical solver described in the previous section. This leads to the kernel sparse representation proposed in [15]. Alternatively, if each row of $\boldsymbol{D}$ is the object to be mapped in high-dimensional feature space, $\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}$ and $\boldsymbol{Y}\boldsymbol{D}^{\mathrm{T}}$ should be updated analytically, then $\boldsymbol{A}$ is updated by a solver given in the previous section. This leads to an alternative kernel sparse representation model.

4 Computational Experiment

Sparse matrix factorization has a wide ranges of applications in biological data analysis. Technically speaking, these applications are based on clustering, biclustering, feature extraction, classification, and feature selection. In this paper, we give three examples to show that promising performance can be obtained by VSMF for feature extraction, feature selection, and biology process identification. For other applications of NMF, please refer to [14].

4.1 Feature Extraction and Classification

NMF is a successful feature extraction method in bioinformatics [12]. Dimension reduction including feature extraction and feature selection is an important step for classification. We compared the performance of our VSMF (for feature extraction) with NMF on a popular microarray gene expression data – Colon [1]. This data set has 2000 genes (features) and 62 samples. There are two classes in this data set. Each sample is normalized to have unit l_2-norm. We employed 4-fold cross-validation to split the whole data into training and test sets. For each split, features were extracted by NMF or VSMF from the training set. The *nearest neighbor* (NN) classifier was used to predict the class labels of the test set. 4-fold cross-validation was repeated for 20 times. We initialized $k = 8$, thus the actual value of k, after calling VSMF, should be less than or equal to 8. *Radial basis function* (RBF) is used in the kernel VSMF. We set the kernel parameter $\sigma = 2^0$. The mean accuracy, *standard deviation* (STD), computing time, and parameter setting are given in Table 2. The standard NMF obtained a mean accuracy of 0.7645, while the linear VSMF yielded 0.7919. The highest accuracy, 0.7944, is obtained by the kernel VSMF. The kernel VSMF only took 1.3346 seconds, which is faster than the linear VSMF and NMF, because the analytical solution of $\boldsymbol{A}$ can be computed for kernel VSMF. We treat this comparison as a demonstration that tuning the parameters of VSMF may obtain better accuracy than NMF. VSMF can be used for many other types of high-throughput data such as copy number profiles and mass spectrometry data.

Table 2. The Classification Performance of VSMF Compared to The Standard NMF. The time is measure by stopwatch timer (the `tic` and `toc` functions in MATLAB) in seconds.

Method	Accuracy (STD)	Time	Parameters
NN	0.7742(0.0260)	0.0137	
NMF+NN	0.7645(0.0344)	4.3310	
Linear VSMF+NN	0.7919(0.0353)	3.1868	$\alpha_2 = 2^{-3}, \lambda_1 = 2^{-6}, t_1 = t_2 = 1$
Kernel VSMF+NN	0.7944(0.0438)	1.3346	$\alpha_2 = 2^{-3}, \lambda_1 = 2^{-6}, t_1 = t_2 = 1, \sigma = 2^0$

4.2 Feature Selection

VSMF can be applied to feature selection. The basic idea is to make the basis vectors sparse, and then select features that vary dramatically among the basis vectors. In our current study of gene selection, we use the following strategy on the sparse basis matrix $\boldsymbol{A}$. For the i-th row (that is the i-th gene), We denote $\boldsymbol{g}_i = \boldsymbol{A}_{i,:}$. If the maximum value in $\boldsymbol{g}_i$ is greater than $\theta = 10^4$ times of the rest values in $\boldsymbol{g}_i$, then we select this gene, otherwise discard it. We tested this VSMF-based feature selection method on a microarray breast tumor data set which have 13582 genes and 158 samples from five classes [7]. The data were normalized so that each gene has mean zero and STD 1. We used the following parameters of VSMF: $\alpha_1 = 2^4$, $\alpha_2 = 2^0$, $\lambda_1 = 0$, $\lambda_2 = 2^0$, $t_1 = 0$, and $t_2 = 1$. The value of k was initialized by 5. The genes selected were validated by classification performance. We employed 20 runs of 4-fold cross-validation. For

each split of training and test sets, genes were selected using the training set. On the dimension-reduced training set, a linear *support vector machine* (SVM) was learned in order to predict the class labels of the corresponding test set. When using all genes to training SVM, we obtained a mean accuracy of 0.8250 with STD 0.0201. When applying the VSMF-based gene selection, we achieved a mean accuracy of 0.8271 with STD 0.0174. We can see that SVM using our gene selection strategy can obtain similar performance with that of using all genes.

4.3 Biological Process Identification

NMF has been applied on either static gene-sample or time-series microarray data to identify potential biological processes [8, 9, 16, 17]. In our experiment, we run our VSMF on the *Gastrointestinal stromal tumor* (GIST) time-series data to show that VSMF can smooth biological processes compared with the result obtained by the standard NMF. This data set was obtained after the treatment of imatinib mesylate. It has 1336 genes and 9 time points. Each gene time-series is normalized to have unit l_2-norm. The smoothness is controlled by parameter α_2. We set the parameters of VSMF to $\alpha_2 = 2^{-2}$, $\lambda_1 = 2^{-8}$, $\alpha_1 = \lambda_2 = 0$, and $t_1 = t_2 = 1$. The number of factors k was set to 3. The basis vectors of NMF and VSMF are shown at the left and right sides of Fig. 1, respectively. We can see that both of them can reconstruct the falling, rising, and transient patterns identified in [16]. The patterns obtained by VSMF are smoother than those of the standard NMF.

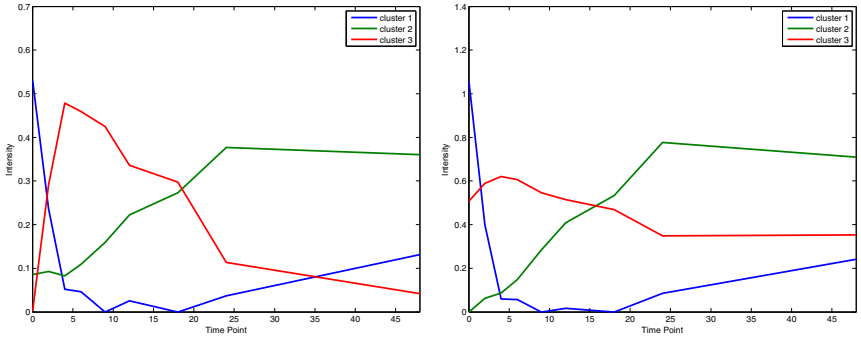

Fig. 1. The Biological processes identified b the standard NMF (*left*) and VSMF (*right*). The result of VSMF is smoother than that of the standard NMF.

5 Conclusions

In this paper, we propose a versatile sparse matrix factorization (VSMF) model for biological data analysis. VSMF is a unified model of many variants of NMF and SR. We give efficient optimization algorithms for VSMF. As shown in our computational demonstrations, many analysis can be conveniently conducted by VSMF for biological

data. The implementation of VSMF can be found in our open-source MATLAB NMF toolbox [14]. The multiplicative-update-rules based VSMF is implemented in function `sparsenmf2rule` and the function `vsmf` includes the implementation of NNQP, l_1QP, and analytical solutions.

We present our on-going work on VSMF in this paper. There remains many interesting challenges in its theoretical and practical aspects. First, there are four key parameters, in the objective of VSMF, which provide flexibility, while rise concerns on the model selection. The two parameters in the constraints can be determined by the signs of a data set. We are working on a guide of parameter selection for VSMF which can be easily tailored for various applications. The value of k is also related with the sparsity, thus we need further investigation on it. Second, increasing the value of α_1 leads to a more sparse basis matrix. This is very helpful for feature selection. We will investigate more effective feature selection method using VSMF. The performance of VSMF for feature selection will be compared statistically with existing approaches. The genes selected will be validated by permutation test and gene set enrichment analysis.

Acknowledgments. This research is supported by IEEE CIS Walter Karplus Summer Research Grant 2010, Ontario Graduate Scholarships 2011-2013, NSERC Grants #RGPIN228117-2011, and several scholarships from The University of Windsor.

References

1. Alon, U.: Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays. PNAS 96(12), 6745–6750 (1999)
2. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, 2nd edn., Belmont, MA (2008)
3. Brunet, J., Tamayo, P., Golub, T., Mesirov, J.: Metagenes and molecular pattern discovery using matrix factorization. PNAS 101(12), 4164–4169 (2004)
4. Carmona-Saez, P., Pascual-Marqui, R.D., Tirado, F., Carazo, J.M., Pascual-Montano, A.: Biclustering of gene expression data by non-smooth non-negative matrix factorization. BMC Bioinformatics 7, 78 (2006)
5. Ding, C., Li, T., Jordan, M.I.: Convex and semi-nonnegative matrix factorizations. TPAMI 32(1), 45–55 (2010)
6. Elad, M.: Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing. Springer, New York (2010)
7. Hu, Z.: The molecular portraits of breast tumors are conserved across microarray platforms. BMC Genomics 7, 96 (2006)
8. Kim, H., Park, H.: Sparse non-negatice matrix factorization via alternating non-negativity-constrained least aquares for microarray data analysis. SIAM J. Matrix Analysis and Applications 23(12), 1495–1502 (2007)
9. Kim, P., Tidor, B.: Subsystem identification through dimensionality reduction of large-scale gene expression data. Genome Research 13, 1706–1718 (2003)
10. Lee, D.D., Seung, S.: Learning the parts of objects by non-negative matrix factorization. Nature 401, 788–791 (1999)
11. Lee, D., Seung, S.: Algorithms for non-negative matrix factorization. In: Advances in Neural Information Processing Systems, pp. 556–562. MIT Press (2001)
12. Li, Y., Ngom, A.: Non-negative matrix and tensor factorization based classification of clinical microarray gene expression data. In: BIBM, pp. 438–443. IEEE Press, Piscataway (2010)

13. Li, Y., Ngom, A.: A new kernel non-negative matrix factorization and its application in microarray data analysis. In: CIBCB, pp. 371–378. IEEE Press, Piscataway (2012)
14. Li, Y., Ngom, A.: The non-negative matrix factorization toolbox for biological data mining. BMC Source Code for Biology and Medicine 8, 10 (2013)
15. Li, Y., Ngom, A.: Sparse representation approaches for the classification of high-dimensional biological data. BMC Systems Biology (in press, 2013)
16. Ochs, M., Fertig, E.: Matrix factorization for transcriptional regulatory network inference. In: CIBCB, pp. 387–396. IEEE Press, Piscataway (2012)
17. Ochs, M., Rink, L., Tarn, C., Mburu, S., Taguchi, T., Eisenberg, B., Godwin, A.: Detection of treatment-induced changes in signaling pathways in sastrointestinal stromal tumors using transcripttomic data. Cancer Res. 69(23), 9125–9132 (2009)
18. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society - Series B: Statistical Methodology 67(2), 301–320 (2005)