ONLINE NONNEGATIVE MATRIX FACTORIZATION BASED ON KERNEL MACHINES

Fei Zhu, Paul Honeine

Institut Charles Delaunay (CNRS), Université de Technologie de Troyes, France

ABSTRACT

Nonnegative matrix factorization (NMF) has been increasingly investigated for data analysis and dimension-reduction. To tackle large-scale data, several online techniques for NMF have been introduced recently. So far, the online NMF has been limited to the linear model. This paper develops an online version of the nonlinear kernel-based NMF, where the decomposition is performed in the feature space. Taking the advantage of the stochastic gradient descent and the mini-batch scheme, the proposed method has a fixed, tractable complexity independent of the increasing samples number. We derive the multiplicative update rules of the general form, and describe in detail the case of the Gaussian kernel. The effectiveness of the proposed method is validated on unmixing hyperspectral images, compared with the state-of-the-art online NMF methods.

***Index Terms*—** Nonnegative matrix factorization, online learning, kernel machines, hyperspectral unmixing

1. INTRODUCTION

Nonnegative matrix factorization (NMF) consists in approximating a given nonnegative matrix by the product of two low-rank ones [1], the left low-rank matrix is often called basis matrix while the right one is the encoding one. Due to the ability to extract parts-based features for the nonnegative input data, the NMF provides a framework suitable to a host of applications. In particular, applied to the hyperspectral unmixing problem, the NMF jointly estimates the "pure" spectra, namely endmembers (given in the basis matrix) and their fractional abundances at each pixel (given in the encoding matrix).

Most studies concentrate on the linear NMF model, where the objective function is defined by the Frobenius norm in an Euclidean space, called *input space*. In this case, one seeks to minimize the difference between the input matrix and the product of the estimated ones. This linear model is often improved by auxiliary regularization terms. Recently, a few kernel-based NMF have been proposed to extent the linear NMF model to the nonlinear scope. By exploiting the framework offered by the kernel machines, these methods map the data with some nonlinear function from the input space to a *feature space*, and perform the existing linear techniques on the transformed data. Unfortunately, the curse of the pre-image [2], inherent from kernel machines, remains the bottleneck in most kernel-based NMF, *e.g.*, [3, 4]. That is, the obtained bases lie in the feature space, and thus one cannot represent both low-rank matrices in the input space. Recently proposed in [5], the so-called KNMF overcomes this difficultly by minimizing an objective function, which is defined in the feature space, directly in the input space. It is this formulation that is investigated throughout this paper.

To handle large-scale and streaming dynamic data, a couple of NMF methods have been extended from batch mode to online mode. For instance, online methods in [6–9] all deal with the conventional linear NMF model. The projective online NMF (PONMF) [10, 11] maintains the virtue of its batch counterpart in guaranteeing a sparse, parts-based representation. In [12], the authors consider the online version of the NMF with Itakura-Saito divergence. Online NMF with volume constraint is discussed in [13]. Roughly, due to the continuously increasing computational complexity, the naive idea of conducting sequentially batch NMF (or its variants) is far from efficient in online setting. To alleviate this computational overhead, the incremental online NMF (IONMF) in [6] introduced first the fixity of encoding of the processed samples. Since, this assumption was widely adopted in online NMF algorithms, namely [7, 8, 13], to name a few. On the other hand, some online NMF variants, *e.g.* [7, 8, 10], follow the spirit of the stochastic gradient descent (SGD), a prime complexity-reduction strategy for online learning [14]. Instead of considering all the available samples so far, SGD style methods import merely a single or a small batch of samples at each iteration, thereby reducing the complexity. Similarly, to prohibit processing the whole data, the method presented in [9] factorizes the matrix composed by the previous basis matrix and novel sample instead. To the best of our knowledge, current literatures of online NMF are limited to a linear model, whereas no online method exists for nonlinear kernel-based NMF. By taking advantage of the aforementioned stochastic gradient and mini-batch modes, this paper extents the batch KNMF to an online mode by keeping a tractable computational complexity. Moreover, we provide multiplicative update rules of the general form, and describe in detail the case of the Gaussian kernel.

This work was supported by the French ANR, grand HYPANEMA: ANR-12BS03-0033.

2. NMF AND KNMF

This section briefly reviews the NMF, with its conventional linear model, and the recent kernel-based variant (KNMF).

Given a nonnegative data matrix $\boldsymbol{X} = [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \cdots \ \boldsymbol{x}_T] \in \Re^{L\times T}$, wherein $\boldsymbol{x}_t \in \Re^L$ represents the t-th sample data, linear NMF aims to factorize it into two low-rank nonnegative matrices. Namely $\boldsymbol{X} \approx \boldsymbol{E}\boldsymbol{A}$, under the nonnegativity constraint on all the entries of $\boldsymbol{E} \in \Re^{L\times N}$ and $\boldsymbol{A} \in \Re^{N\times T}$. Let $\boldsymbol{E} = [\boldsymbol{e}_1 \ \boldsymbol{e}_2 \ \cdots \ \boldsymbol{e}_N]$. An equivalent vector-wise model is given by $\boldsymbol{x}_t \approx \sum_{n=1}^N a_{nt}\,\boldsymbol{e}_n$, for $t = 1,\ldots,T$. The latter model can be interpreted as to represent each sample data $\boldsymbol{x}_t$ as a linear combination of vectors $\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_N$, called basis vectors with scalars a_{nt} being the entries of the encoding matrix $\boldsymbol{A}$. Denote $\mathcal{X}$ as the input space spanned by all the vectors $\boldsymbol{x}_t$, as well as the vectors $\boldsymbol{e}_n$. The optimization problem of NMF consists in minimizing the following objective function in $\mathcal{X}$

$$J(\boldsymbol{E},\boldsymbol{A}) = \frac{1}{2}\sum_{t=1}^T \|\boldsymbol{x}_t - \sum_{n=1}^N a_{nt}\,\boldsymbol{e}_n\|^2,$$

with the nonnegative constraints on the matrices $\boldsymbol{E}$ and $\boldsymbol{A}$.

In [5], a kernel-based NMF (KNMF) is proposed by considering the following matrix factorization model

$$\boldsymbol{X}^\Phi \approx \boldsymbol{E}^\Phi \boldsymbol{A},$$

where $\boldsymbol{X}^\Phi = [\Phi(\boldsymbol{x}_1) \ \ \Phi(\boldsymbol{x}_2) \ \cdots \ \Phi(\boldsymbol{x}_T)]$ and $\boldsymbol{E}^\Phi = [\Phi(\boldsymbol{e}_1) \ \Phi(\boldsymbol{e}_2) \ \cdots \ \Phi(\boldsymbol{e}_N)]$, or equivalently in its vector-wise form,

$$\Phi(\boldsymbol{x}_t) \approx \sum_{n=1}^N a_{nt}\,\Phi(\boldsymbol{e}_n), \tag{1}$$

for all $t = 1,\ldots,T$. Here, $\Phi(\cdot)$ is a nonlinear function mapping the columns of the matrix $\boldsymbol{X}$, as well as the columns of the matrix $\boldsymbol{E}$, from the input space $\mathcal{X}$ to some feature space $\mathcal{H}$. Its associated norm is denoted $\|\cdot\|_{\mathcal{H}}$, and the corresponding inner product in the feature space is of the form $\langle\Phi(\boldsymbol{x}_t),\Phi(\boldsymbol{x}_{t'})\rangle_{\mathcal{H}}$, which can be evaluated using the so-called kernel function $\kappa(\boldsymbol{x}_t,\boldsymbol{x}_{t'})$ in kernel machines. Considering the vector-wise form model (1), the objective function is written as

$$J(\boldsymbol{E},\boldsymbol{A}) = \frac{1}{2}\sum_{t=1}^T \Big\|\Phi(\boldsymbol{x}_t) - \sum_{n=1}^N a_{nt}\,\Phi(\boldsymbol{e}_n)\Big\|_{\mathcal{H}}^2, \tag{2}$$

where the nonnegativity constraint is imposed on all the entries of $\boldsymbol{e}_n$ and $\boldsymbol{A}$.

As in the conventional NMF and its variants, although the estimation of both unknown matrices jointly is a nonconvex problem, its subproblem with one matrix fixed is convex. Well-known NMF algorithms, including gradient descent method and multiplicative update rules, basically alternate the optimization over two unknown matrices by keeping the other one fixed [1].

3. ONLINE KNMF

We extent the aforementioned KNMF from batch to an online version. In the online setting, the samples arrive successively; the method is expected to produce a sequence of factorizations based on all the samples received so far. An intuitive idea is to iteratively conduct the batch KNMF. Unfortunately, as the samples number continuously grows, this method suffers an increasing, untractable computational complexity. By investigating the stochastic gradient, we propose an online KNMF (KONMF) with fixed computational complexity.

3.1. Problem formulation

From (2), the objective function corresponding to the first k samples is rewritten as

$$J_k(\widetilde{\boldsymbol{E}},\widetilde{\boldsymbol{A}}) = \frac{1}{2}\sum_{t=1}^k \Big\|\Phi(\boldsymbol{x}_t) - \sum_{n=1}^N \widetilde{a_{nt}}\,\Phi(\widetilde{\boldsymbol{e}_n})\Big\|_{\mathcal{H}}^2,$$

where $\widetilde{\boldsymbol{E}}$ and $\widetilde{\boldsymbol{A}}$ denote respectively the basis matrix and the encoding matrix for current k samples. We adopt the following assumption, initially proposed in [6] and employed in most online NMF methods, *e.g.*, [7,8,15]: from k to $k+1$, the encoding vectors for the first k samples remain unchanged, *i.e.*, $\boldsymbol{a}_t = \widetilde{\boldsymbol{a}_t}$, for $t = 1,\ldots,k$.

As the new sample $\boldsymbol{x}_{k+1}$ is available, one needs to estimate the new basis matrix $\boldsymbol{E}$, by updating $\widetilde{\boldsymbol{E}}$, and the novel sample's encoding vector $\boldsymbol{a}_{k+1}$, to be appended to $\widetilde{\boldsymbol{A}}$. The above objective function is modified to

$$\begin{aligned} J_{k+1}(\boldsymbol{E},\boldsymbol{A}) &= \frac{1}{2}\sum_{t=1}^{k+1} \Big\|\Phi(\boldsymbol{x}_t) - \sum_{n=1}^N a_{nt}\,\Phi(\boldsymbol{e}_n)\Big\|_{\mathcal{H}}^2 \\ &= \frac{1}{2}\sum_{t=1}^{k} \Big\|\Phi(\boldsymbol{x}_t) - \sum_{n=1}^N \widetilde{a_{nt}}\,\Phi(\boldsymbol{e}_n)\Big\|_{\mathcal{H}}^2 \\ &\quad + \frac{1}{2}\Big\|\Phi(\boldsymbol{x}_{k+1}) - \sum_{n=1}^N a_{n(k+1)}\,\Phi(\boldsymbol{e}_n)\Big\|_{\mathcal{H}}^2. \end{aligned}$$

It is noteworthy that the above objective function is expressed as a sum of sub-loss functions over data samples. By expanding this expression and removing the constant term $\frac{1}{2}\sum_{t=1}^{k+1}\kappa(\boldsymbol{x}_t,\boldsymbol{x}_t)$, the optimization problem becomes

$$\min_{\boldsymbol{a}_{k+1},\boldsymbol{E}} \sum_{t=1}^{k+1} l(\boldsymbol{x}_t,\boldsymbol{a}_t,\boldsymbol{E}), \tag{3}$$

where $l(\boldsymbol{x}_t,\boldsymbol{a}_t,\boldsymbol{E})$, the sub-loss function over sample $\boldsymbol{x}_t$, takes the form

$$\frac{1}{2}\sum_{n=1}^N\sum_{m=1}^N a_{nt}a_{mt}\kappa(\boldsymbol{e}_n,\boldsymbol{e}_m) - \sum_{n=1}^N a_{nt}\kappa(\boldsymbol{e}_n,\boldsymbol{x}_t).$$

In the following, we adopt a simple alternating technique over the unknown basis matrix $\boldsymbol{E}$ and encoding vector $\boldsymbol{a}_{k+1}$ to minimize the objective function (3).

3.2. Basis matrix update

Keep always in mind that $\boldsymbol{a}_t = \widetilde{\boldsymbol{a}}_t$ holds for $t = 1, \ldots, k$. The gradient of (3) with respect to the vector $\boldsymbol{e}_n$ is:

$$\nabla_{\boldsymbol{e}_n} J_{k+1} = \sum_{t=1}^{k+1} \nabla_{\boldsymbol{e}_n} l(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}), \tag{4}$$

where

$$\nabla_{\boldsymbol{e}_n} l(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}) = a_{nt}\Big(\sum_{m=1}^{N} a_{mt}\, \nabla_{\boldsymbol{e}_n} \kappa(\boldsymbol{e}_n, \boldsymbol{e}_m) - \nabla_{\boldsymbol{e}_n} \kappa(\boldsymbol{e}_n, \boldsymbol{x}_t)\Big).$$

This expression is explicit, since $\nabla_{\boldsymbol{e}_n} \kappa(\boldsymbol{e}_n, \cdot)$, which is the gradient of the kernel with respect to its argument $\boldsymbol{e}_n$, can be determined for most valid kernels. We list in Table 1 the cases with some commonly-used kernels.

A batch gradient descent update rule takes the form

$$\boldsymbol{e}_n = \boldsymbol{e}_n - \eta_n \sum_{t=1}^{k+1} \nabla_{\boldsymbol{e}_n} l(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}), \tag{5}$$

for $n = 1, \ldots, N$, where the step-size parameter η_n may depend on n. Unfortunately, this rule cannot be considered in online learning, since it deals with all the $k+1$ received samples up to each iteration and has a computational cost proportional to the number of samples.

A stochastic gradient descent (SGD) update alleviates this computational burden, by approximating the above gradient based on a single, randomly chosen, $\boldsymbol{x}_t$ at each iteration, and is of the form

$$\boldsymbol{e}_n = \boldsymbol{e}_n - \eta_n \nabla_{\boldsymbol{e}_n} l(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}), \tag{6}$$

for $n = 1, 2, ..., N$. Despite a drastically simplified procedure, the SGD asymptotically converges much slower than its batch mode counterpart [14]. A compromise between these two is the mini-batch mode, which aggregates the gradients corresponding to a randomly picked set of samples. For notational simplicity, denote $\mathcal{I}$ as the sample set therein contain the randomly picked samples employed for updating at each iteration. The mini-batch mode takes the following form

$$\boldsymbol{e}_n = \boldsymbol{e}_n - \eta_n \sum_{\boldsymbol{x}_t \in \mathcal{I}} \nabla_{\boldsymbol{e}_n} l(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}), \tag{7}$$

for $n = 1, 2, ..., N$, where the mini-batch size, pre-fixed, is denoted in the following by p with p=card$(\mathcal{I})$.

In the above gradient descent update rules from (5) to (7), the step-size parameters η_n should be appropriately set. Moreover, a rectification function $a = \max(a, 0)$ should follow after each update in order to guarantee the nonnegativity. To overcome these difficulties, we present below the multiplicative update rules, which are originated by Lee and Seung [1] and have been the baseline for most existing NMF variants. Compared with additive gradient descent update rules, the resulting methods lead to nonnegative factorization with neither the projection to the nonnegative constraint set, nor the pain of choosing the step-size parameter. To this end, we split the gradient corresponding to some sample $\boldsymbol{x}_t$ as the subtraction of two nonnegative terms, *i.e.*,

$$\nabla_{\boldsymbol{e}_n} l(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}) = \nabla_{\boldsymbol{e}_n} l^+(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}) - \nabla_{\boldsymbol{e}_n} l^-(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}). \tag{8}$$

Setting the step-size parameter as

$$\eta_n = \frac{\boldsymbol{e}_n}{\sum_{\boldsymbol{x}_t \in \mathcal{I}} \nabla_{\boldsymbol{e}_n} l^+(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})}$$

yields the following multiplicative update rule of the general form

$$\boldsymbol{e}_n = \boldsymbol{e}_n \otimes \frac{\sum_{\boldsymbol{x}_t \in \mathcal{I}} \nabla_{\boldsymbol{e}_n} l^-(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})}{\sum_{\boldsymbol{x}_t \in \mathcal{I}} \nabla_{\boldsymbol{e}_n} l^+(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})}, \tag{9}$$

where the multiplication $\otimes$ and the division are componentwise. Analogous to the aforementioned additive cases, three multiplicative update rules can be distinguished depending on the number p of samples investigated at each iteration:

- If $p = k+1$, all the samples are proceed and (9) is reduced to the multiplicative update rule for batch KNMF;
- If $p = 1$, the multiplicative update rule (9) corresponds to the stochastic gradient case (6);
- If $1 < p < k+1$, then (9) has the mini-batch gradient case (7) as its additive counterpart, with the mini-batch size equals to p.

Table 1: Commonly-used kernels and their gradients with respect to $\boldsymbol{e}_n$.

Kernel	$\kappa(\boldsymbol{e}_n, \boldsymbol{z})$	$\nabla_{\boldsymbol{e}_n} \kappa(\boldsymbol{e}_n, \boldsymbol{z})$
Linear	$\boldsymbol{z}^\top \boldsymbol{e}_n$	$\boldsymbol{z}$
Polynomial	$(\boldsymbol{z}^\top \boldsymbol{e}_n + \mathrm{c})^d$	$d\,(\boldsymbol{z}^\top \boldsymbol{e}_n + \mathrm{c})^{(d-1)} \boldsymbol{z}$
Gaussian	$\exp(\frac{-1}{2\sigma^2}\|\boldsymbol{e}_n - \boldsymbol{z}\|^2)$	$-\frac{1}{\sigma^2}\kappa(\boldsymbol{e}_n, \boldsymbol{z})(\boldsymbol{e}_n - \boldsymbol{z})$
Sigmoid	$\tanh(\gamma \boldsymbol{z}^\top \boldsymbol{e}_n + \mathrm{c})$	$\gamma \mathrm{sech}^2(\gamma \boldsymbol{z}^\top \boldsymbol{e}_n + \mathrm{c})\boldsymbol{z}$

3.3. Encoding vector update

To estimate the encoding vector $\boldsymbol{a}_{k+1}$ for the newly available $\boldsymbol{x}_{k+1}$, we determine the partial derivative of J_{k+1} with respect to $a_{n(k+1)}$, namely

$$\nabla_{a_{n(k+1)}} J_{k+1} = -\kappa(\boldsymbol{e}_n, \boldsymbol{x}_{k+1}) + \sum_{m=1}^{N} a_{m(k+1)}\, \kappa(\boldsymbol{e}_n, \boldsymbol{e}_m),$$

for $n = 1, 2, ..., N$. Applying the gradient descent scheme, a simple additive update rule is constructed as

$$a_{n(k+1)} = a_{n(k+1)} - \eta_n\, \nabla_{a_{n(k+1)}} J_{k+1}, \tag{10}$$

for $n = 1, 2, ..., N$, where the step-size parameters η_n can be set differently depending on n. Additionally, a rectification

function $a_{nt} = \max(a_{nt}, 0)$ is necessary after each iteration, in order to guarantee the nonnegativity of the entries in $\boldsymbol{a}_{k+1}$. Replacing the step-size parameters η_n in (10) by

$$\eta_n = \frac{1}{\sum_{m=1}^{N} a_{m(k+1)}\, \kappa(\boldsymbol{e}_n, \boldsymbol{e}_m)},$$

the multiplicative update rule for $a_{n(k+1)}$ can be expressed as

$$a_{n(k+1)} = a_{n(k+1)} \times \frac{\kappa(\boldsymbol{e}_n, \boldsymbol{x}_{k+1})}{\sum_{m=1}^{N} a_{m(k+1)}\, \kappa(\boldsymbol{e}_n, \boldsymbol{e}_m)}, \quad (11)$$

for $n = 1, 2, ..., N$.

4. CASE WITH THE GAUSSIAN KERNEL

The multiplicative update rules with a given kernel (belonging to but not restricted to Table 1) can be derived, by appropriately replacing the expressions $\kappa(\boldsymbol{e}_n, \boldsymbol{z})$ and $\nabla_{\boldsymbol{e}_n}\kappa(\boldsymbol{e}_n, \boldsymbol{z})$ in (4), and splitting the gradient as in (8). It is noteworthy that the trivial case with the linear kernel corresponds to the linear NMF in batch mode, and to the IONMF [6] in online mode.

Without losing generality, we detail below the derivation of the multiplicative update rules for the Gaussian kernel. In this case, the matrix factorization is performed in the feature space induced by the Gaussian kernel. The Gaussian kernel is defined by $\kappa(\boldsymbol{e}_n, \boldsymbol{z}) = \exp(\frac{-1}{2\sigma^2}\|\boldsymbol{e}_n - \boldsymbol{z}\|^2)$, where σ is the tunable bandwidth parameter. Its gradient with respect to $\boldsymbol{e}_n$ is $\nabla_{\boldsymbol{e}_n}\kappa(\boldsymbol{e}_n, \boldsymbol{z}) = -\frac{1}{\sigma^2}\kappa(\boldsymbol{e}_n, \boldsymbol{z})(\boldsymbol{e}_n - \boldsymbol{z})$, for any $\boldsymbol{z} \in \mathcal{X}$. Splitting the gradient of the loss function $\nabla_{\boldsymbol{e}_n} l(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})$, as given in (8), yields the following two nonnegative terms:

$$\begin{cases} \nabla_{\boldsymbol{e}_n} l^+ = \frac{a_{nt}}{\sigma^2}(\kappa(\boldsymbol{e}_n, \boldsymbol{x}_t)\boldsymbol{e}_n + \sum_{m=1}^{N} a_{mt}\kappa(\boldsymbol{e}_n, \boldsymbol{e}_m)\boldsymbol{e}_m); \\ \nabla_{\boldsymbol{e}_n} l^- = \frac{a_{nt}}{\sigma^2}(\kappa(\boldsymbol{e}_n, \boldsymbol{x}_t)\boldsymbol{x}_t + \sum_{m=1}^{N} a_{mt}\kappa(\boldsymbol{e}_n, \boldsymbol{e}_m)\boldsymbol{e}_n). \end{cases}$$

Setting the step-size parameter as

$$\eta_n = \frac{\sigma^2 \boldsymbol{e}_n}{\sum\limits_{\boldsymbol{x}_t \in \mathcal{I}} a_{nt}\Big(\kappa(\boldsymbol{e}_n, \boldsymbol{x}_t)\boldsymbol{e}_n + \sum\limits_{m=1}^{N} a_{mt}\kappa(\boldsymbol{e}_n, \boldsymbol{e}_m)\boldsymbol{e}_m\Big)}$$

leads to the multiplicative update rule for $\boldsymbol{e}_n$

$$\boldsymbol{e}_n = \boldsymbol{e}_n \otimes \frac{\sum\limits_{\boldsymbol{x}_t \in \mathcal{I}} a_{nt}\Big(\boldsymbol{x}_t\, \kappa(\boldsymbol{e}_n, \boldsymbol{x}_t) + \sum\limits_{m=1}^{N} a_{mt}\, \boldsymbol{e}_n\, \kappa(\boldsymbol{e}_n, \boldsymbol{e}_m)\Big)}{\sum\limits_{\boldsymbol{x}_t \in \mathcal{I}} a_{nt}\Big(\boldsymbol{e}_n\, \kappa(\boldsymbol{e}_n, \boldsymbol{x}_t) + \sum\limits_{m=1}^{N} a_{mt}\, \boldsymbol{e}_m\, \kappa(\boldsymbol{e}_n, \boldsymbol{e}_m)\Big)}, \quad (12)$$

where the multiplication $\otimes$ and the division are componentwise. For the encoding vector update, the multiplicative updating rule remains unchanged.

5. EXPERIMENTAL RESULTS

This section studies the performance of the proposed method on unmixing two well-known hyperspectral images. We study two sub-images with 50×50 pixels, taken respectively from the well-known Moffett and Urban image. According to the literature [16, 17], $L = 186$ clean bands are of interest for Moffett, and $L = 162$ for Urban, while the endmember number is $N = 3$ for the former, and $N = 4$ for the latter.

To provide a comprehensive comparison, we consider five state-of-the-art online NMF algorithms: online NMF using Hessian matrix (HONMF) [7], incremental online NMF (IONMF) [6], online NMF based on full-rank decomposition theorem (ONMF) [9], projective online NMF (PONMF) [10] and online NMF with robust stochastic approximation (RSA) [8]. The unmixing performance is evaluated with two metrics described in detail in [16]: the reconstruction error in the input space (RE), defined by

$$\text{RE} = \sqrt{\frac{1}{TL}\sum_{t=1}^{T}\|\boldsymbol{x}_t - \sum_{n=1}^{N} a_{nt}\boldsymbol{e}_n\|^2},$$

and the reconstruction error in the feature space (RE^{Φ}), which is defined by

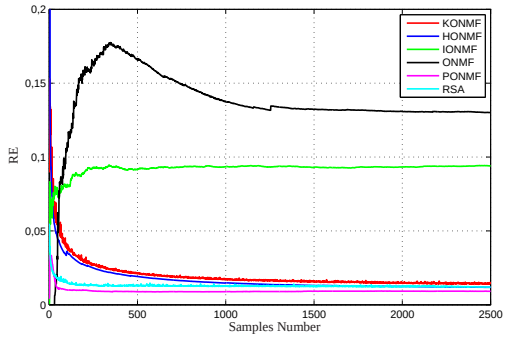
$$\text{RE}^{\Phi} = \sqrt{\frac{1}{TL}\sum_{t=1}^{T}\Big\|\Phi(\boldsymbol{x}_t) - \sum_{n=1}^{N} a_{nt}\Phi(\boldsymbol{e}_n)\Big\|_{\mathcal{H}}^2}.$$

The experiments are conducted using the Gaussian kernel. By conducting a preliminary analysis with the batch KNMF on a small-size dataset, we choose the bandwidth in kernel as $\sigma = 3.3$ for Moffett and $\sigma = 3.0$ for Urban. Regarding the parameter-setting, the mini-batch size is empirically chosen as in [8], with $p = \min\{\lceil \frac{k}{10} \rceil, 30\}$. To ensure a fair comparison, the iteration number is equally set to be $I = 100$ in all the algorithms.

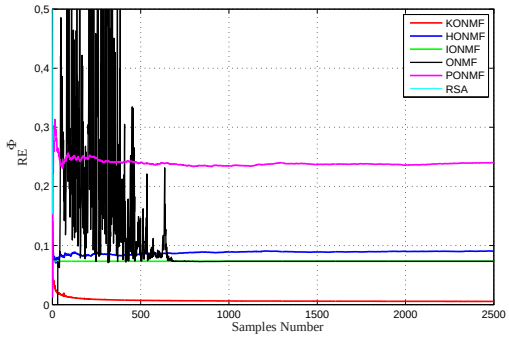
As shown in Figure 1 and Figure 2, the proposed KONMF with the Gaussian kernel outperforms ONMF and INMF in terms of the reconstruction error in the input space, and surpasses all the state-of-the-art methods in terms of the reconstruction error in the feature space.

6. CONCLUSION

This paper presented a novel online nonnegative matrix factorization based on kernel machines. Exploiting SGD and mini-batch strategies, we derived the general form of multiplicative update rules that maintain a tractable computational complexity. The effectiveness of the method was demonstrated for unmixing hyperspectral images. Future works include mini-batch size determination, step-size choice in additive update rules, as well as speedup strategies.

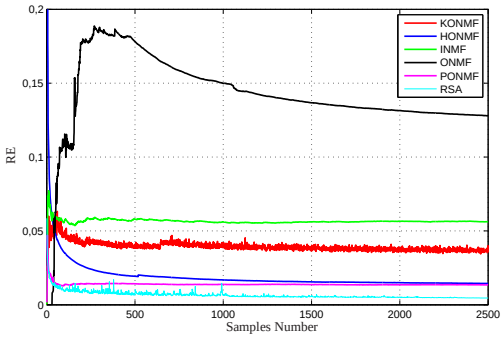


(a) RE

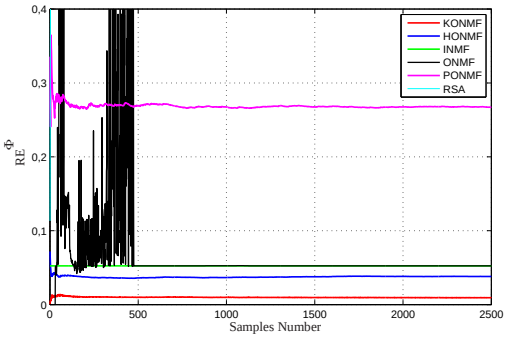


(b) RE^{Φ}

Fig. 1: Evolution of the reconstruction errors in the input and feature spaces on the Moffett image.



(a) RE



(b) RE^{Φ}

Fig. 2: Evolution of the reconstruction errors in the input and feature spaces on the Urban image.

REFERENCES

[1] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization.," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.

[2] P. Honeine and C. Richard, "Preimage problem in kernel-based machine learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 77–88, 2011.

[3] D. Zhang, Z. Zhou, and S. Chen, "Non-negative matrix factorization on kernels," in *Lecture Notes in Computer Science*. 2006, vol. 4099, pp. 404–412, Springer.

[4] C. Ding, T. Li, and M. I. Jordan, "Convex and Semi-Nonnegative Matrix Factorizations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 45–55, Nov. 2010.

[5] F. Zhu, P. Honeine, and M. Kallas, "Kernel non-negative matrix factorization without the pre-image problem," in *IEEE workshop on Machine Learning for Signal Processing*, Reims, France, Sep. 2014.

[6] S. S. Bucak and B. Gunsel, "Incremental subspace learning via non-negative matrix factorization," *Pattern Recognition*, vol. 42, no. 5, pp. 788 – 797, 2009.

[7] F. Wang, C. H. Tan, P. Li, and A. C. König, "Efficient document clustering via online nonnegative matrix factorizations," in *Eleventh SIAM International Conference on Data Mining*. April 2011, Society for Industrial and Applied Mathematics.

[8] N. Y. Guan, D. C. Tao, Z. G. Luo, and B. Yuan, "Online nonnegative matrix factorization with robust stochastic approximation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 7, pp. 1087–1099, July 2012.

[9] B. Cao, D. Shen, J. T. Sun, X. H. Wang, Q. Yang, and Z. Chen, "Detect and track latent factors with online nonnegative matrix factorization.," in *IJCAI*, 2007, vol. 7, pp. 2689–2694.

[10] Z. R. Yang, H. Zhang, and E. Oja, "Online projective nonnegative matrix factorization for large datasets," in *Neural Information Processing*. Springer, 2012, pp. 285–290.

[11] D. Wang and H. C. Lu, "On-line learning parts-based representation via incremental orthogonal projective non-negative matrix factorization," *Signal Processing*, vol. 93, no. 6, pp. 1608–1623, 2013.

[12] A. Lefevre, F. Bach, and C. Fevotte, "Online algorithms for nonnegative matrix factorization with the itakura-saito divergence," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2011 IEEE Workshop on*, Oct 2011, pp. 313–316.

[13] G. X. Zhou, Z. Y. Yang, S. l. Xie, and J.-M. Yang, "Online blind source separation using incremental nonnegative matrix factorization with volume constraint," *IEEE Transactions on Neural Networks*, vol. 22, no. 4, pp. 550–560, April 2011.

[14] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, pp. 421–436. Springer, 2012.

[15] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *The Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[16] F. Zhu, P. Honeine, and M. Kallas, "Kernel nonnegative matrix factorization without the curse of the pre-image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (submitted in 2014) 2015 preprint.

[17] S. Jia and Y. T. Qian, "Spectral and spatial complexity-based hyperspectral unmixing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 12, pp. 3867–3879, Dec. 2007.