Online Kernel Nonnegative Matrix Factorization

Fei Zhu, Paul Honeine

Online Kernel Nonnegative Matrix Factorization

Fei Zhu[a], Paul Honeine[b]

[a] *Université de Technologie de Troyes, Institut Charles Delaunay, France.*
[b] *Normandie Université, Université de Rouen, LITIS Lab, Rouen, France.*

Abstract

Nonnegative matrix factorization (NMF) has become a prominent signal processing and data analysis technique. To address streaming data, online methods for NMF have been introduced recently, mainly restricted to the linear model. In this paper, we propose a framework for online nonlinear NMF, where the factorization is conducted in a kernel-induced feature space. By exploring recent advances in the stochastic gradient descent and the mini-batch strategies, the proposed algorithms have a controlled computational complexity. We derive several general update rules, in additive and multiplicative strategies, and detail the case of the Gaussian kernel. The performance of the proposed framework is validated on unmixing synthetic and real hyperspectral images, comparing to state-of-the-art techniques.

Keywords: Nonnegative matrix factorization, online learning, kernel machines, hyperspectral unmixing

1. Introduction

Nonnegative matrix factorization (NMF) aims to approximate a given nonnegative matrix by the product of two low-rank ones [1], and has been a prime dimensionality reduction technique for signal and image processing. Due to its ability to extract the nonnegative and parts-based features for the nonnegative input data, the NMF provides a framework appropriate to many real-world applications. Of particular interest is the hyperspectral unmixing problem. A hyperspectral image is an assemblage of scenes obtained across a certain wavelength range, namely the reflectance spectrum is available at each pixel. Assuming that each spectrum is a mixture of to-be-determined pure spectra, termed endmembers, the unmixing problem estimates these endmembers and their fractional abundances at each pixel. The estimated

endmembers and abundances should be nonnegative for physical interpretation, and a sum-to-one constraint is often imposed to the abundance values at each pixel. The NMF estimates jointly the endmembers (recorded in the *basis* matrix) and their fractional abundances (recorded in the *encoding* matrix). Without loss of generality, we follow this terminology in this paper.

In the linear NMF, each column of the input matrix can be viewed as a linear combination of the basis vectors, with weights given by the encoding matrix. This model has been extensively investigated, with the cost function often augmented by including regularization terms, such as sparsity on the encoding matrix [2] and smoothness penalty on the basis matrix [3]. Taking advantage of the framework provided by kernel machines, a few attempts have been recently made for nonlinear NMF. The trick is to map the data, with some nonlinear transformation, to a higher dimensional *feature space*, where the factorization occurs. Most kernel-based NMF, such as in [4, 5], suffer from the pre-image problem which prevents them from extracting the basis matrix [6]. We have recently proposed in [7, 8, 9, 10] the so-called KNMF which bypasses this problem by optimizing directly in the input space.

To tackle large-scale and streaming dynamic data, a couple of online NMF methods have been proposed. Most of them considered the conventional linear NMF model, as investigated in [11, 12, 13, 14]. In an online setting, the computational complexity is the main concern to address. To prohibit processing the whole data, early work presented in [14] factorized the matrix composed by the previous basis matrix and novel samples. The incremental online NMF (IONMF) in [11] proposed to fix the encoding for the past samples, thereby alleviating the computational overhead. Since that published work, this assumption has been widely applied in online NMF algorithms [12, 13, 15]. Moreover, as the online NMF possesses a separable cost function with respect to samples, the authors of [16] applied the stochastic gradient descent (SGD) strategy, which is a crucial complexity-reduction approach for online learning [17]. This technique consists in substituting the real gradient with the noisy stochastic one, since the latter involves merely a single or a small batch of samples. In [13], the recent robust stochastic approximation technique was exploited for the linear model, where the SGD was improved with a smartly chosen step-size and an average step on the results.

It is noteworthy that the online NMF is closely related to the online dictionary learning and sparse coding, where the basic idea is to adaptably learn a dictionary from data, and to represent each sample as a sparse combination of the dictionary elements. The linear model considers an Euclidean least-

squares loss function, with an ℓ_1-norm regularizer on the coding term in order to promote its sparsity [18]. Extensions has been recently proposed, such as a robust factorization with the Huber loss function as presented in [19], and a nonlinear variation defined on a Riemannian manifold in [20].

To the best of our knowledge, there is no online algorithm for kernel-based NMF. In this paper[1], we propose an online KNMF (OKNMF) framework that explores recent advances from stochastic optimization. By investigating the SGD, mini-batch and averaged SGD (ASGD) strategies, we derive additive and multiplicative update rules to estimate the basis matrix (endmembers) and the encoding vectors (abundances). We provide extensive experiments on synthetic and real hyperspectral images to demonstrate the relevance of the proposed algorithms. The remainder of this paper is organized as follows. Section 2 presents the NMF and its kernel-based variant KNMF. In Section 3, we revisit the latter for online learning and provide appropriate algorithms. Several extensions are presented in Section 4. Experimental results are reported on synthetic and real hyperspectral images in Section 5.

2. A Primer on NMF and KNMF

Given a nonnegative matrix $\boldsymbol{X} = [\boldsymbol{x}_1 \;\; \boldsymbol{x}_2 \;\; \cdots \;\; \boldsymbol{x}_T]$ with $\boldsymbol{x}_t \in \mathcal{X} \subseteq \Re^L$ being the t-th sample (*i.e.*, spectrum), the linear NMF consists in factorizing it into the product of two low-rank nonnegative matrices, namely $\boldsymbol{X} \approx \boldsymbol{E}\boldsymbol{A}$. Let $\boldsymbol{E} = [\boldsymbol{e}_1 \;\; \boldsymbol{e}_2 \;\; \cdots \;\; \boldsymbol{e}_N]$ be the basis matrix. An equivalent formulation is

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

for $t = 1, 2, \ldots, T$. This model represents each sample $\boldsymbol{x}_t$ as a linear combination of the basis vectors $\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_N$ with weights a_{nt} being the entries of the encoding matrix $\boldsymbol{A}$. The conventional NMF minimizes the cost function

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

[1]This paper is an extended version of [21], and provides more relevant algorithms (ASGD, sparse coding, ...), deeper theoretical analysis (convergence, complexity, ...) and extensive experimental results.

with the nonnegativity on the entries of $\boldsymbol{E}$ and $\boldsymbol{A}$. NMF algorithms iteratively alternate the optimization over the two unknown matrices [1, 22].

The kernel-based NMF (KNMF) considers the factorization $\boldsymbol{X}^{\Phi} \approx \boldsymbol{E}^{\Phi}\boldsymbol{A}$, where $\boldsymbol{X}^{\Phi} = [\Phi(\boldsymbol{x}_1) \cdots \Phi(\boldsymbol{x}_T)]$ and $\boldsymbol{E}^{\Phi} = [\Phi(\boldsymbol{e}_1) \cdots \Phi(\boldsymbol{e}_N)]$, or equivalently

$$\Phi(\boldsymbol{x}_t) \approx \sum_{n=1}^{N} a_{nt}\, \Phi(\boldsymbol{e}_n), \tag{2}$$

for all t. Here, $\Phi(\cdot)$ is a nonlinear function mapping any element of the input space $\mathcal{X}$ to some feature space $\mathcal{H}$. The induced norm is denoted $\|\cdot\|_{\mathcal{H}}$ and the corresponding inner product is evaluated using a kernel function $\kappa(\cdot,\cdot)$, such as the Gaussian kernel. The corresponding cost function is

$$\mathcal{J}(\boldsymbol{E},\boldsymbol{A}) = \frac{1}{2}\sum_{t=1}^{T} \left\| \Phi(\boldsymbol{x}_t) - \sum_{n=1}^{N} a_{nt}\, \Phi(\boldsymbol{e}_n) \right\|_{\mathcal{H}}^2, \tag{3}$$

where the nonnegativity constraint is imposed on all $\boldsymbol{e}_n$ and $\boldsymbol{A}$. To get the solution, a two-block coordinate descent strategy is investigated in [9, 10].

3. Online KNMF

In the online setting, the samples arrive successively, at each instant. An intuitive idea is to iteratively conduct the batch KNMF. As the samples number continuously grows, this approach suffers from an intractable computational complexity. We propose in the following a framework for online KNMF (**OKNMF**) with a controlled computational complexity. Additive and multiplicative update rules for the basis matrix (Section 3.2) and the encoding matrix (Section 3.3) are described in the general form, while the Gaussian kernel is detailed in Section 3.4. The computational complexity and the stopping criterion are discussed in Section 3.5. All these developments can be easily extended to other kernel-based NMF formulations, *e.g.*, [4, 5].

3.1. Problem formulation

From (3), the cost function at instant k (*i.e.* of the first k samples) is

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

where $\widetilde{\boldsymbol{E}}$ and $\widetilde{\boldsymbol{A}}$ are respectively the basis and encoding matrices at instant k. We adopt the following assumption, initially proposed in [11] and employed in [12, 13, 16] for online NMF: from k to $k+1$, the encoding vectors for the first k samples remain unchanged, *i.e.*, the matrix $\widetilde{\boldsymbol{A}}$ is only appended.

As a new sample $\boldsymbol{x}_{k+1}$ is available at instant $k+1$, one estimates the matrix $\boldsymbol{E}$ by updating $\widetilde{\boldsymbol{E}}$, and the novel sample's encoding vector $\boldsymbol{a}_{k+1}$ is appended to $\widetilde{\boldsymbol{A}}$, namely $\boldsymbol{A} = [\widetilde{\boldsymbol{A}} \quad \boldsymbol{a}_{k+1}]$. The above cost function becomes

$$\mathcal{J}_{k+1}(\boldsymbol{E}, \boldsymbol{A}) = \frac{1}{2}\sum_{t=1}^{k}\left\|\Phi(\boldsymbol{x}_t) - \sum_{n=1}^{N}\widetilde{a_{nt}}\Phi(\boldsymbol{e}_n)\right\|_{\mathcal{H}}^2 + \frac{1}{2}\left\|\Phi(\boldsymbol{x}_{k+1}) - \sum_{n=1}^{N}a_{n(k+1)}\Phi(\boldsymbol{e}_n)\right\|_{\mathcal{H}}^2. \tag{4}$$

By removing the constant term, the optimization problem becomes

$$\min_{\boldsymbol{a}_{k+1}, \boldsymbol{E}\geq 0}\sum_{t=1}^{k+1}\mathcal{L}(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}), \tag{5}$$

where the sub-loss function $\mathcal{L}(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})$ 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 an alternating strategy to solve the problem (5).

3.2. Basis matrix update

The gradient of the loss function in (5) with respect to the vector $\boldsymbol{e}_n$ is:

$$\nabla_{\boldsymbol{e}_n}\mathcal{J}_{k+1} = \sum_{t=1}^{k+1}\nabla_{\boldsymbol{e}_n}\mathcal{L}(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}), \tag{6}$$

where $\nabla_{\boldsymbol{e}_n}\mathcal{L}(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}) = a_{nt}\big(\sum_{m=1}^{N}a_{mt}\nabla_{\boldsymbol{e}_n}\kappa(\boldsymbol{e}_n, \boldsymbol{e}_m) - \nabla_{\boldsymbol{e}_n}\kappa(\boldsymbol{e}_n, \boldsymbol{x}_t)\big)$. In this expression, $\nabla_{\boldsymbol{e}_n}\kappa(\boldsymbol{e}_n, \cdot)$ denotes the gradient of the kernel with respect to its (first) argument $\boldsymbol{e}_n$, as given in Table 1 for most commonly-used kernels. In the following, we derive additive and multiplicative update rules.

Additive update rules — SGD and ASGD

Consider the projected gradient descent (PGD) (see [22] for an overview of the PGD for batch NMF), then a PGD update for KNMF takes the form

$$\boldsymbol{e}_n = \Big(\boldsymbol{e}_n - \eta_n\sum_{t=1}^{k+1}\nabla_{\boldsymbol{e}_n}\mathcal{L}(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})\Big)_+,$$

Table 1: Commonly-used kernels and their gradients with respect to their first argument.

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	$\tanh(\gamma \boldsymbol{z}^\top \boldsymbol{e}_n + \mathrm{c})$	$\gamma \operatorname{sech}^2(\gamma \boldsymbol{z}^\top \boldsymbol{e}_n + \mathrm{c}) \boldsymbol{z}$

for all n, where η_n is the step-size parameter and $(\cdot)_+$ is the operator that projects to the nonnegative set. This rule cannot be applied in online learning, since it deals with all the k received samples at each instant k.

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

$$\boldsymbol{e}_n = \big(\boldsymbol{e}_n - \eta_n \nabla_{\boldsymbol{e}_n} \mathcal{L}(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})\big)_+. \tag{7}$$

With a drastically simplified procedure, the SGD converges much slower than its batch mode counterpart [17]. A compromise between these two is the mini-batch mode, which aggregates the gradients corresponding to a randomly-chosen subset of samples. Let $\mathcal{I}$ be this subset randomly-chosen at each instant. The mini-batch update rule takes the following form

$$\boldsymbol{e}_n = \Big(\boldsymbol{e}_n - \eta_n \sum_{\boldsymbol{x}_t \in \mathcal{I}} \nabla_{\boldsymbol{e}_n} \mathcal{L}(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})\Big)_+, \tag{8}$$

where the pre-fixed mini-batch size is denoted in the following by $p = \mathrm{card}(\mathcal{I})$.

To accelerate the convergence of the SGD, we further consider an averaged stochastic gradient descent (ASGD) strategy, as initially proposed in [23]. By averaging the results obtained by SGD (with mini-batch) over samples, the ASGD is expressed as $\overline{\boldsymbol{e}_n^j} = \frac{1}{j - j_0} \sum_{j=j_0+1}^{j} \boldsymbol{e}_n^j$, or in the recursive form

$$\overline{\boldsymbol{e}_n^{j+1}} = (1 - \xi_j)\, \overline{\boldsymbol{e}_n^j} + \xi_j\, \boldsymbol{e}_n^{j+1}, \tag{9}$$

where $\xi_j = 1/\max(1, j - j_0)$ is the averaging rate [24, 17]. Theoretical results in [23] show that the ASGD converges as good as the second-order SGD [12], where the latter requires the costly computation of the Hessian.

To appropriately tune the step-size parameters η_n, we revisit recent advances developed in the linear case, where the convexity of the loss function enables robust stochastic approximation [13] and second-order PGD [12], *i.e.*,

the use of the (approximate) inverse of the Hessian as a step-size [16]. However, the kernel-based loss function $\mathcal{L}(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})$ may be nonconvex in terms of $\boldsymbol{e}_n$. Following recent theoretical results [17, 24], we adopt the step-size $\eta_j = \eta_0(1 + \eta_0\lambda j)^{-1}$, equally for all the basis vectors $\boldsymbol{e}_n$. Hence, it starts at a pre-determined value η_0 and diminishes asymptotically as $(\lambda j)^{-1}$, λ being a tunable parameter. According to [17], this form of step-size proves to be effective in SGD algorithms. Moreover, it leads to the best convergence speed when the loss function is convex with λ being the smallest eigenvalue of the Hessian [17, 24]. Regardless of the possible nonconvex loss function, experiments show that this step-size provides excellent results for OKNMF.

Multiplicative update (MU)

We present next the multiplicative update rules, by revisiting the original work in [1] for batch NMF. As opposed to the additive update rules, the resulting algorithms neither require the rectification to impose the nonnegativity, nor have the pain of choosing the step-size parameter. To this end, we split the gradient in (6) as the subtraction of two positive terms, namely

$$\nabla_{\boldsymbol{e}_n}\mathcal{L}(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}) = \mathcal{G}^+(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}) - \mathcal{G}^-(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}). \tag{10}$$

By setting the step-size to $\eta_n = \boldsymbol{e}_n / \sum_{\boldsymbol{x}_t \in \mathcal{I}} \mathcal{G}^+(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})$, this yields the following multiplicative update rule of the general form

$$\boldsymbol{e}_n = \boldsymbol{e}_n \otimes \frac{\sum_{\boldsymbol{x}_t \in \mathcal{I}} \mathcal{G}^-(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})}{\sum_{\boldsymbol{x}_t \in \mathcal{I}} \mathcal{G}^+(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})}, \tag{11}$$

where the multiplication $\otimes$ and the division are component-wise. Analogous to the additive cases, three multiplicative update rules are proposed, depending on the pre-fixed number p of samples investigated at each instant: (i) If $p = k + 1$, all the samples are processed and (11) boils down to the multiplicative update rule of the batch KNMF; (ii) If $p = 1$, then (11) uses a single randomly chosen sample, as with the stochastic gradient descent (7); (iii) If $1 < p < k + 1$, then (11) operates with a mini-batch update of size equal to p, as its additive counterpart (8).

3.3. Encoding vector update

To estimate the encoding vector $\boldsymbol{a}_{k+1}$ for the newly available sample $\boldsymbol{x}_{k+1}$, the basis matrix $\boldsymbol{E}$ is fixed, as well as the previously estimated encoding

vectors $\boldsymbol{a}_t$, for $t = 1, 2, \ldots, k$. The optimization problem becomes

$$\min_{\boldsymbol{a}_{k+1} \geq \boldsymbol{0}} \frac{1}{2} \Big\| \Phi(\boldsymbol{x}_{k+1}) - \sum_{n=1}^{N} a_{n(k+1)} \, \Phi(\boldsymbol{e}_n) \Big\|_{\mathcal{H}}^2. \tag{12}$$

The kernel-based model (2) is linear-in-the-parameters a_{nt}. As a consequence, we can investigate well-known algorithmssuch as the active set method implemented in [12, 13] and the multiplicative update routine of NMF [11, 15].

The derivative of $\mathcal{J}_{k+1}(\boldsymbol{E}, \boldsymbol{A})$ in (4) with respect to $a_{n(k+1)}$ is

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

Applying the gradient descent scheme yields the update rule

$$a_{n(k+1)} = a_{n(k+1)} - \eta'_n \, \nabla_{a_{n(k+1)}} \mathcal{J}_{k+1},$$

for all n, where η'_n denotes the step-size parameter. Additionally, a rectification is necessary at each iteration in order to guarantee the nonnegativity. By replacing the step-size parameter η'_n with $\eta'_n = 1/\sum_{m=1}^{N} a_{m(k+1)} \kappa(\boldsymbol{e}_n, \boldsymbol{e}_m)$, the multiplicative update rule for $a_{n(k+1)}$ can be expressed as

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

for all n. If the sum-to-one constraint is required, the resulting encoding vector can be divided at each iteration by its ℓ_1-norm, namely $\sum_{m=1}^{N} a_{m(k+1)}$.

3.4. Case of the Gaussian kernel

The update rules for any given kernel (including but not limited to the ones in Table 1) can be derived, by appropriately replacing the expressions of $\kappa(\boldsymbol{e}_n, \boldsymbol{z})$ and $\nabla_{\boldsymbol{e}_n} \kappa(\boldsymbol{e}_n, \boldsymbol{z})$ in (6) (SGD/ASGD), and splitting the gradient as in (10) (MU). The trivial case with the linear kernel corresponds to the linear NMF in batch mode, and to IONMF [11] in online mode.

We detail below the derivation of the update rules for the Gaussian kernel (see Table 1). For the encoding vector update, the update rule remains unchanged. For the basis matrix, the mini-batch SGD update (8) becomes

$$\boldsymbol{e}_n = \Big(\boldsymbol{e}_n - \frac{\eta}{\sigma^2} \sum_{\boldsymbol{x}_t \in \mathcal{I}} a_{nt} \big(\kappa(\boldsymbol{e}_n, \boldsymbol{x}_t)(\boldsymbol{e}_n - \boldsymbol{x}_t) - \sum_{m=1}^{N} a_{mt} \, \kappa(\boldsymbol{e}_n, \boldsymbol{e}_m)(\boldsymbol{e}_n - \boldsymbol{e}_m) \big) \Big)_+.$$

The corresponding ASGD update is given by sequently addressing the above output of SGD with (9). By splitting the gradient of the loss function $\nabla_{\boldsymbol{e}_n}\mathcal{L}(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E})$, as given in (10), we get the following two nonnegative terms:

$$\begin{cases} \mathcal{G}^+(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}) = \dfrac{a_{nt}}{\sigma^2}\Big(\kappa(\boldsymbol{e}_n, \boldsymbol{x}_t)\boldsymbol{e}_n + \displaystyle\sum_{m=1}^{N} a_{mt}\kappa(\boldsymbol{e}_n, \boldsymbol{e}_m)\boldsymbol{e}_m\Big); \\ \mathcal{G}^-(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}) = \dfrac{a_{nt}}{\sigma^2}\Big(\kappa(\boldsymbol{e}_n, \boldsymbol{x}_t)\boldsymbol{x}_t + \displaystyle\sum_{m=1}^{N} a_{mt}\kappa(\boldsymbol{e}_n, \boldsymbol{e}_m)\boldsymbol{e}_n\Big). \end{cases}$$

Setting the step-size as aforementioned leads to the multiplicative update

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

where the multiplication $\otimes$ and the division are component-wise.

3.5. Complexity and stopping criterion

We first analyse the computational complexity and memory usage for the OKNMF with SGD/ASGD/MU algorithms. Assume $N \ll \min(L, k)$.

The time complexity for updating each basis vector $\boldsymbol{e}_n$ is $\mathcal{O}(pNL)$ per iteration, which holds for commonly-used kernels due to a roughly equal time complexity of L for computing κ. Thus, the total time complexity for the basis matrix is $\mathcal{O}(ps_1N^2L)$, where s_1 is the number of iterations. In the online setting, the sequential batch update has an increasing value $\mathcal{O}(ks_1N^2L)$, k being the current number of samples, and therefore is unrealistic for streaming data. The mini-batch is attractive for its fixed complexity $\mathcal{O}(ps_1N^2L)$ and for its performance as demonstrated with the experiments conducted in Section 5. The time complexity for the encoding vector update is $\mathcal{O}(s_2NL)$, s_2 being the number of iterations. The total time complexity for the encoding matrix update remains unchanged, since the matrix is only appended.

The complexity in terms of memory usage is $\mathcal{O}(Lk + Nk)$ at instant k, by keeping in memory all the proceeded data and their encoding vectors. To alleviate this storage burden, we retain in memory only the last q samples with their encoding vectors. Termed "buffering strategy" in [13], this scheme can reduce the memory complexity to the fixed value $\mathcal{O}(Lq + Nq)$.

Concerning the stopping criterion, a two-fold criterion is applied for both basis matrix and encoding vector updates: either the decrease in objective function between two successive iterations is small enough (10^{-4} in experiments), or the preset maximum number of iterations is reached (*e.g.,* 100).

4. Extensions of OKNMF

This section presents several extensions of OKNMF, with the most known regularizations. Other extensions can be conveniently incorporated in the OKNMF framework, by revisiting the work on the batch mode in [10].

*4.1. Sparse coding (**sOKNMF**)*

Sparsity, introduced to NMF in [2] by penalizing the ℓ_1-norm of the encoding vector, allows to represent each sample $\boldsymbol{x}_t$ with only few basis vectors. Under the nonnegativity constraint, $\|\boldsymbol{a}_t\|_{\ell_1} = \sum_{n=1}^{N} a_{nt}$. Adding such regularization in the initial cost function brings no effect to the basis matrix update, since it is independent of $\boldsymbol{e}_n$. For the encoding vector given with the optimization problem (12), this leads to the sparsity-promoting version

$$\min_{\boldsymbol{a}_{k+1}\geq \boldsymbol{0}} \frac{1}{2}\Big\|\Phi(\boldsymbol{x}_{k+1}) - \sum_{n=1}^{N} a_{n(k+1)}\,\Phi(\boldsymbol{e}_n)\Big\|_{\mathcal{H}}^2 + \beta \sum_{n=1}^{N} a_{n(k+1)},$$

where β controls the tradeoff between the factorization accuracy and the level of sparsity. By taking the derivative with respect to $a_{n(k+1)}$ and appropriately choosing the step-size, the multiplicative update rule of $\boldsymbol{a}_{k+1}$ becomes

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

The influence of sparse coding on the regularization of the update is clear.

4.2. Smoothness on the basis vectors

The smoothness regularization is of great interest when dealing with ill-posed problems, namely in hyperspectral unmixing since the extracted bases should be less "spiky" in order to be relevant endmembers. In the following, we examine several regularizations to promote the smoothness.

A natural regularization is the ℓ_2-norm penalty [3], which leads to

$$\mathcal{J}_{k+1}^{\text{2-norm}}(\boldsymbol{E}, \boldsymbol{A}) = \sum_{t=1}^{k+1} \mathcal{L}(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}) + \frac{\delta}{2} \sum_{n=1}^{N} \|\boldsymbol{e}_n\|^2.$$

Its gradient with respect to any $\boldsymbol{e}_n$ can be expressed with ease, as well as the additive and multiplicative update rules. Another way to promote smooth

solutions for NMF is proposed in [25] by penalizing variations between successive values, by minimizing $\sum_{l=2}^{L-1} |e_{ln} - e_{(l-1)n}|$ for all n. In this case

$$\mathcal{J}_{k+1}^{\text{fluct}}(\boldsymbol{E}, \boldsymbol{A}) = \sum_{t=1}^{k+1} \mathcal{L}(\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{E}) + \frac{\gamma}{2} \sum_{n=1}^{N} \sum_{l=2}^{L-1} |e_{ln} - e_{(l-1)n}|.$$

The corresponding update rules can be obtained by following the derivations given in Section 3. Other smoothness regularizations can be considered [10].

5. Experiments on Synthetic and Real Hyperspectral Images

This section first presents the relevance of the SGD/ASGD/MU algorithms within the proposed OKNMF framework on several classes of synthetic data, before investigating three well-known real hyperspectral images.

5.1. State-of-the-art methods for online NMF

The online NMF (**ONMF**) [14] has a reduced computational complexity thanks to the full-rank decomposition theorem. However, it yields inferior performance as demonstrated in many works, such as in [13] and in this paper. The incremental online NMF (**IONMF**) [11] uses an incremental subspace learning to tackle the increasing complexity in the online setting. In [12], an optimization problem similar to IONMF is solved with additive update rules. Therein, the update with first-order PGD is closely related to the work in [16], where online factorization with sparse coding is discussed. By replacing the step-size in the PGD with the approximated inverse Hessian, the second-order PGD (**HONMF**) is advocated in [12]. The robust stochastic approximation (**RSA**) [13] benefits from the recent progress in choosing the step-size and averaging over the results, with a convergence rate of $\mathcal{O}(1/\sqrt{k})$ guaranteed for the basis update. The projective online NMF (**PONMF**) yields orthogonal and sparse basis vectors, with the factorization $\boldsymbol{X} = \boldsymbol{E}\boldsymbol{E}^\top\boldsymbol{X}$ [26]. The volume-constrained online NMF in [15] minimizes $\log|\det(\boldsymbol{E})|$, under the assumption of a square basis matrix.

5.2. Evaluation metrics

The reconstruction error is $\text{RE} = \sqrt{\frac{1}{TL}\sum_t \|\boldsymbol{x}_t - \sum_{n=1}^{N} a_{nt}\boldsymbol{e}_n\|^2}$. From [7, 8, 10], the reconstruction error in the feature space is defined by $\text{RE}^\Phi = \sqrt{\frac{1}{TL}\sum_t \left\|\Phi(\boldsymbol{x}_t) - \sum_{n=1}^{N} a_{nt}\Phi(\boldsymbol{e}_n)\right\|_{\mathcal{H}}^2}$. When ground-truth is available with

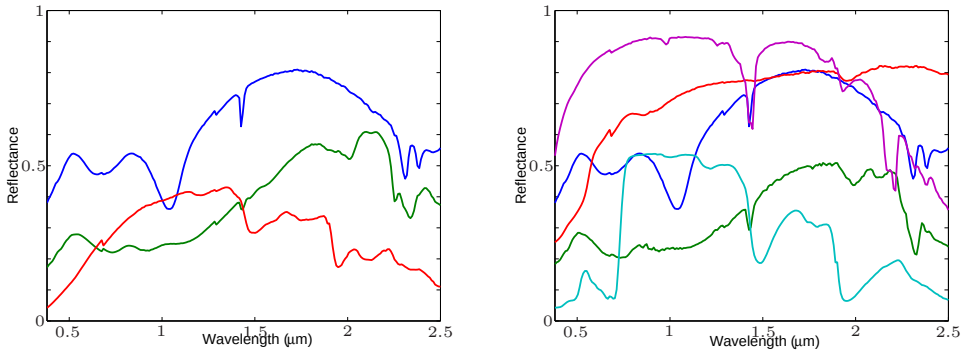


Figure 1: The spectra used for generating the synthetic images, with $N = 3$ endmembers for the GBM (left) and $N = 6$ for the PNMM (right).

Table 2: Parameter Settings for the Synthetic Images

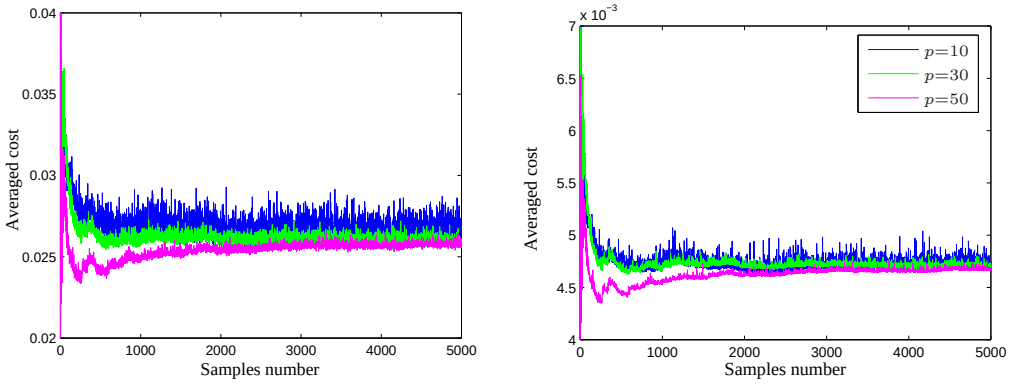
		GBM				PNMM			
		σ	p	η_0	λ	σ	p	η_0	λ
OKNMF	SGD	5.5	30	0.25	2^{-8}	6.5	30	0.25	2^{-8}
	ASGD			2	2^{-9}			1	2^{-9}
	MU			-	-			-	-

the real endmembers $\underline{\boldsymbol{e}_n}$ and abundances $\underline{\boldsymbol{a}_t}$, the unmixing quality can be further evaluated with two other metrics. The averaged spectral angle distance between endmembers [27] is $\mathrm{SAD} = \frac{1}{N}\sum_{n=1}^{N} \arccos \frac{\boldsymbol{e}_n^\top \underline{\boldsymbol{e}_n}}{\|\boldsymbol{e}_n\|\|\underline{\boldsymbol{e}_n}\|}$, and the root mean square error of abundances [28, 29] is $\mathrm{RMSE} = \sqrt{\frac{1}{NT}\sum_t \|\boldsymbol{a}_t - \underline{\boldsymbol{a}_t}\|^2}$.

5.3. Performance of OKNMF on synthetic data

The performance is evaluated by unmixing two series of synthetic hyperspectral images, each composed of 50 000 pixels. The endmembers used for data generation are from the USGS digital spectral library used in [30]. These spectra, with $L = 224$ bands, are shown in Figure 1. Five images are generated using the generalized bilinear model (GBM) [28], defined by

$$\boldsymbol{x}_t = \sum_{n=1}^{N} a_{nt}\, \boldsymbol{e}_n + \sum_{n=1}^{N-1} \sum_{m=n+1}^{N} \gamma_{nm}\, a_{nt}\, a_{mt}\, (\boldsymbol{e}_n \otimes \boldsymbol{e}_m) + \boldsymbol{n},$$



(a) RE (b) RE^{Φ}

Figure 2: Influence of the value of the mini-batch size p on the reconstruction errors in the input and feature spaces, using the MU algorithm, with $p = 10$, 30, or 50.

where $\gamma_{nm} \in [0, 1]$ is generated from the uniform distribution. Another five images are defined with a postnonlinear mixing model (PNMM) [31], namely

$$\boldsymbol{x}_t = \sum_{n=1}^{N} a_{nt}\, \boldsymbol{e}_n + b \left(\sum_{n=1}^{N} a_{nt}\, \boldsymbol{e}_n \right) \otimes \left(\sum_{n=1}^{N} a_{nt}\, \boldsymbol{e}_n \right) + \boldsymbol{n},$$

where b is uniformly generated within $[-0.3, 0.3]$ according to [31]. For each image, the abundance values a_{nt} are uniformly generated within $[0, 1]$ under the sum-to-one constraint, and $\boldsymbol{n}$ is a Gaussian noise with SNR = 30 dB.

Experiments are conducted with the following settings. The bandwidth of the Gaussian kernel is determined with the batch KNMF [10]. The mini-batch size is chosen with $p = \min\{\lceil \frac{k}{10} \rceil, m\}$, which allows to balance between the computational cost and the smoothness of the convergence. It influence is shown with the MU (other algorithms depend also on the step-size parameter). As illustrated in Figure 2, a moderate value is 30. While small values of p cause fluctuating convergence, an over-sized $p = 50$ is computational expensive without significant improvement due to the redundancy within the data. Last, concerning the algorithms SGD/ASGD with additive updates, the optimal values of η_0 and λ are determined with a 10-fold cross-validation on 1000 random pixels, using the candidate values $\eta_0 = \{2^{-3}, 2^{-2}, \ldots, 2^{1}\}$ and $\lambda = \{2^{-15}, 2^{-14}, \ldots, 2^{1}\}$. Table 2 summarizes the parameter settings.

For the sake of fair comparison, the basis matrix is identically initialized for all algorithms, using the endmembers estimated by the NMF algorithm

Table 3: Unmixing Performance for the Synthetic Images ($\times 10^{-2}$)

		GBM				PNMM			
		SAD	RMSE	RE	RE^{Φ}	SAD	RMSE	RE	RE^{Φ}
	ONMF	100.39±2.75	48.87±0.02	66.56±28.51	6.68±0.00	104.66±18.3	48.82±0.04	79.52±65.8	6.68±0.00
	IONMF	③ 10.06±0.94	16.41±1.83	1.81±0.01	6.68±0.00	19.52±7.37	③ 15.06±2.29	③ 1.71±0.14	6.68±0.00
	HONMF	20.62±4.07	④ 14.47±4.97	② **1.57**±0.21	5.58±1.76	10.40±1.67	19.16±4.30	① **1.40**±0.01	4.68±0.85
	RSA	32.95±8.98	34.93±5.38	① **1.56**±0.19	635.38±0.04	33.96±8.63	35.56±5.11	② **1.54**±0.20	631.92±0.04
	PONMF	71.53±0.03	① **12.58**±0.01	③ 1.81±0.00	71.34±0.01	71.54±0.04	① **12.56**±0.01	1.79±0.00	70.46±0.01
OKNMF	SGD	12.48±2.42	17.17±2.77	2.51±0.19	③ 0.51±0.03	① **8.44**±3.46	19.70±2.71	2.65±0.13	② **0.45**±0.01
OKNMF	ASGD	① **9.19**±0.57	② **14.43**±1.30	2.25±0.01	① **0.47**±0.00	② **8.93**±1.70	② **15.01**±2.95	2.40±0.11	① **0.42**±0.01
OKNMF	MU	② **10.00**±1.48	17.08±4.89	2.38±0.13	② **0.49**±0.02	③ 9.42±3.18	18.72±3.85	2.60±0.08	③ 0.45±0.02

on a small subset of samples. With five Monte-Carlo simulations, the aforementioned metrics are given in Table 3. OKNMF provides jointly the best endmembers/abundances in terms of SAD/RMSE. The only competitive algorithm seems to be PONMF when dealing with the RMSE of abundances. However, the estimated endmembers are the worst with PONMF, up to eightfold compared to the proposed ASGD algorithm. The reconstruction errors allow to measure the relevance of the jointly estimated endmembers and abundances. The most accurate reconstruction is achieved by all the proposed algorithms (SGD/ASGD/MU), with the reconstruction error in the feature space RE^{Φ}. It is noticeable that these errors with RE^{Φ} are at least threefold lower than the ones obtained by all state-of-the-art algorithms with RE. This means that the Gaussian-based model (2) provides the most suitable factorization for the studied images, outperforming the linear one (1).

5.4. Performance of the sparse coding OKNMF on synthetic data

To study the relevance of the sparsity-promoting sOKNMF algorithm, we generate four sets (each at a given sparsity level) of synthetic data, each set consisting of five images of 50×50 pixels using the PNMM model and endmembers shown in Figure 1. To define the sparsity level, a proportion (15%, 30%, 45% and 60%) of the encoding matrices are nullified, by ensuring at least one non-zero entry existing for each column. For sOKNMF, the parameter β should be tuned according to the sparsity of the unknown abundance matrix. According to [2, 32], a rough estimator from the input spectra is

$$s = \frac{1}{\sqrt{L}} \sum_{l=1}^{L} \frac{\sqrt{T} - \|\boldsymbol{X}^l\|_1 / \|\boldsymbol{X}^l\|_2}{\sqrt{T} - 1},$$

where $\boldsymbol{X}^l$ is the l-th row of $\boldsymbol{X}$. The candidate value set of β is $\{0, 0.01s, 0.1s, s\}$, with $s = 0$ corresponding to the non-sparse OKNMF. In all experiments, the

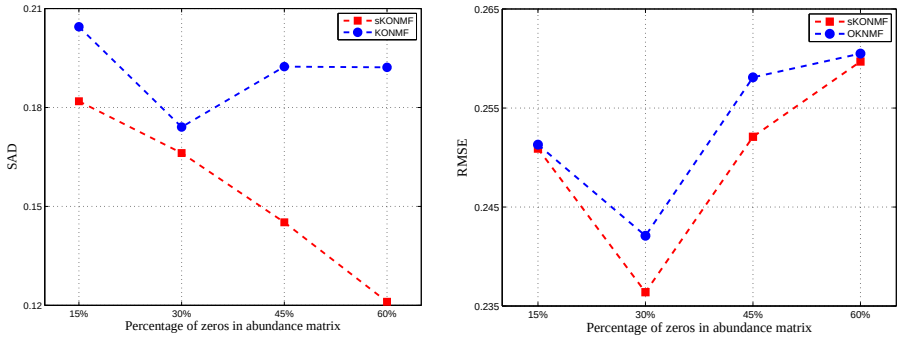


Figure 3: The averaged spectral angle distance (SAD) and the averaged root mean square error (RMSE) versus the sparsity level in the abundances, using OKNMF and sOKNMF.

(a) Moffett (b) Samson (c) Urban

Figure 4: The real hyperspectral images

ASGD is applied with $\eta_0 = 2$, $\lambda = 2 \times 10^{-8}$, and $\sigma = 6.0$. With ten Monte-Carlo simulations, the averaged SAD and RMSE are given in Figure 3. Both SAD and RMSE are enhanced by the sparsity-promoting sOKNMF.

5.5. Real hyperspectral images

In the following, we describe three real hyperspectral images, and provide experimental results in Section 5.6.

The first image is a 50×50-pixel sub-image from the well-known Moffett image, illustrated in Figure 4(a). This image, acquired with the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) with channels ranging from $0.4\mu m$ to $2.5\mu m$, has $L = 186$ spectral bands after removing noisy and water absorption bands. While this image is relatively small, it has been widely investigated and is known to be composed of vegetation, soil and water. The

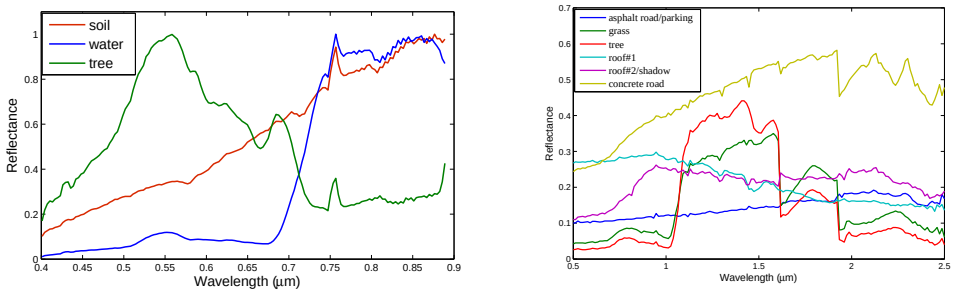


Figure 5: The ground-truth endmembers in Samson (left) and Urban (right) images.

second image was acquired by the Spectroscopic Aerial Mapping System with Onboard Navigation (SAMSON), with 156 channels ranging from $0.4\mu m$ to $0.9\mu m$. We consider an image of 95×95 pixels illustrated in Figure 4(b), as in [33]. Given in Figure 5, the scene is known to be mainly composed of soil, water and tree. The third image is the relatively big Urban image illustrated in Figure 4(c). It is available from the Hyperspectral Digital Imagery Collection Experiment (HYDICE), and contains 307×307 pixels with the wavelength ranging from $0.4\mu m$ to $2.5\mu m$. After removing the noisy bands, $L = 162$ spectral bands are of interest. With up to $N = 6$ endmembers [32, 33], this scene is mainly composed of asphalt road/parking, grass, tree, roof♯1, roof♯2/shadow, and concrete road, as given in Figure 5.

5.6. Performance on real hyperspectral images

The parameter settings given in Table 4 are obtained by performing a similar procedure as with the synthetic data, with ten Monte-Carlo simulations are carried out for the Moffett and Samson images, and three Monte-Carlo simulations for the Urban image.

The learning curves (*i.e.*, averaged cost at each instant) in Figure 6 show the fast convergence of the ASGD, SGD, and MU algorithms on the three images, with ASGD having the most stable decrease. The reconstruction errors (RE and RE^{Φ}) are given in Table 5. With ground-truth information

Table 4: Parameter Settings for the Real Images

		Moffett				Samson				Urban			
		σ	p	η_0	λ	σ	p	η_0	λ	σ	p	η_0	λ
ONMF	SGD	3.3	30	1	2^{-8}	7.0	30	1	2^{-11}	3.0	30	1	2^{-8}
	ASGD			1	2^{-12}			2	2^{-11}			1	2^{-12}
	MU			-	-			-	-			-	-

Table 5: Unmixing Performance for the Moffett, Samson and Urban Images ($\times 10^{-2}$)

		Moffett		Samson		Urban	
		RE	RE^{Φ}	RE	RE^{Φ}	RE	RE^{Φ}
	ONMF	11.94±3.49	7.36±0.01	31.95±11.83	8.01±0.01	231.48±23.23	7.88±0.01
	IONMF	① **0.82**±0.13	7.33±0.00	2.85±0.86	8.01±0.01	1.27±0.32	7.86±0.00
	HONMF	③ 1.03±0.69	9.60±3.69	① **0.87**±0.24	346.24±138.02	② **0.87**±0.22	17.02±4.56
	RSA	1.21±0.01	84.66±0.12	③ 1.36±0.31	235.35±0.52	① **0.69**±0.04	228.64±0.04
	PONMF	② **0.90**±0.02	23.82±0.93	② **0.99**±0.01	29.38±0.21	③ 1.12±0.08	39.02±1.25
OKNMF	SGD	1.53±0.26	① **0.55**±0.08	3.99±0.99	② **0.62**±0.12	3.16±0.15	① **0.95**±0.02
OKNMF	ASGD	1.54±0.21	② **0.57**±0.05	3.45±0.70	① **0.58**±0.01	2.53±0.02	③ 1.02±0.01
OKNMF	MU	2.08±0.96	③ 0.93±0.27	3.91±0.96	③ 0.63±0.12	2.45±0.07	② **1.01**±0.04

Table 6: Averaged spectral angle distance (SAD) for the Samson Image ($\times 10^{-2}$)

		soil	water	tree	MEAN
	ONMF	-	-	-	$\geq$ 97.16
	IONMF	41.87	③ 9.01	45.65	32.18
	HONMF	37.60	① **5.92**	③ 26.01	③ 23.18
	RSA	② **21.42**	10.23	98.22	43.29
	PONMF	122.54	20.27	47.12	63.31
OKNMF	SGD	③ 28.26	12.87	② **23.93**	② **21.68**
OKNMF	ASGD	① **17.71**	② **8.01**	30.33	① **18.68**
OKNMF	MU	48.37	11.71	① **19.02**	26.37

on the Samson and Urban images, the relevance of the estimated endmembers is measured with the SAD, as given in Tables 6 and 7. The nonlinear model in OKNMF leads to the smallest reconstruction error in the feature space. Despite a relatively high RE in the input space, the nonlinear model results in the lowest SAD, namely the extracted endmembers are the closest to the ground-truth, which reveals the underlying nonlinearity in the images.

Figures 7 and 8 illustrate the estimated endmembers and abundance maps for the Moffett and Samson images, and Figure 9 for the Urban image. As observed, the proposed OKNMF is able to recognize regions that are the most consistent with the ground-truth, whereas state-of-the-art techniques can only distinguish partly the regions while resulting in spiky/noisy endmembers and incoherent abundance maps.

6. Conclusion

This paper presented a novel framework for online kernel-based NMF methods, by exploring the stochastic gradient descent and the mini-batch strategies in stochastic optimization. Experimental results for unmixing synthetic and real hyperspectral images demonstrated the effectiveness of the derived algorithms. These algorithms outperformed state-of-the-art methods,

Table 7: Averaged spectral angle distance (SAD) for the Urban Image ($\times 10^{-2}$)

		asphalt	grass	tree	roof♯1	roof♯2	cr. road	MEAN
	ONMF	-	-	-	-	-	-	$\geq$ 133.75
	IONMF	53.45	46.84	41.38	24.47	③ 64.15	56.77	47.84
	HONMF	56.25	58.99	40.64	53.87	② **48.39**	47.93	51.09
	RSA	58.83	② **27.68**	③ 29.90	41.89	86.15	29.27	45.61
	PONMF	94.86	97.55	101.84	97.85	105.83	109.44	101.12
OKNMF	SGD	③ 35.63	42.06	51.27	③ 17.84	68.59	① **2.97**	③ 36.39
OKNMF	ASGD	① **29.02**	① **25.37**	① **6.93**	① **5.48**	71.32	② **3.41**	② **23.58**
OKNMF	MU	② **31.54**	③ 36.47	② **9.59**	② **15.92**	① **31.56**	③ 3.68	① **21.45**

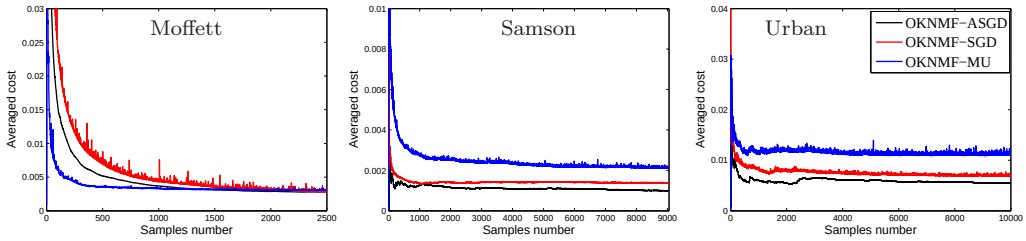


Figure 6: The learning curves (averaged cost per instant), of the ASGD, SGD, and MU algorithms on the Moffett, Samson and Urban (first 10 000 samples) images.

and the estimated bases are the closest to the ground-truth endmembers. Future works include theoretical results on the convergence rate.

Acknowledgment

This work was supported by the French ANR, grant ANR-12BS03-0033.

References

[1] D. D. Lee, H. S. Seung, Learning the parts of objects by non-negative matrix factorization., Nature 401 (6755) (1999) 788–791.

[2] P. O. Hoyer, P. Dayan, Non-negative matrix factorization with sparseness constraints, Journal of Machine Learning Research 5 (2004) 1457–1469.

[3] J. Piper, V. P. Pauca, R. J. Plemmons, M. Giffin, Object characterization from spectral data using nonnegative factorization and information theory, in: Proceedings of AMOS Technical Conference, 2004.

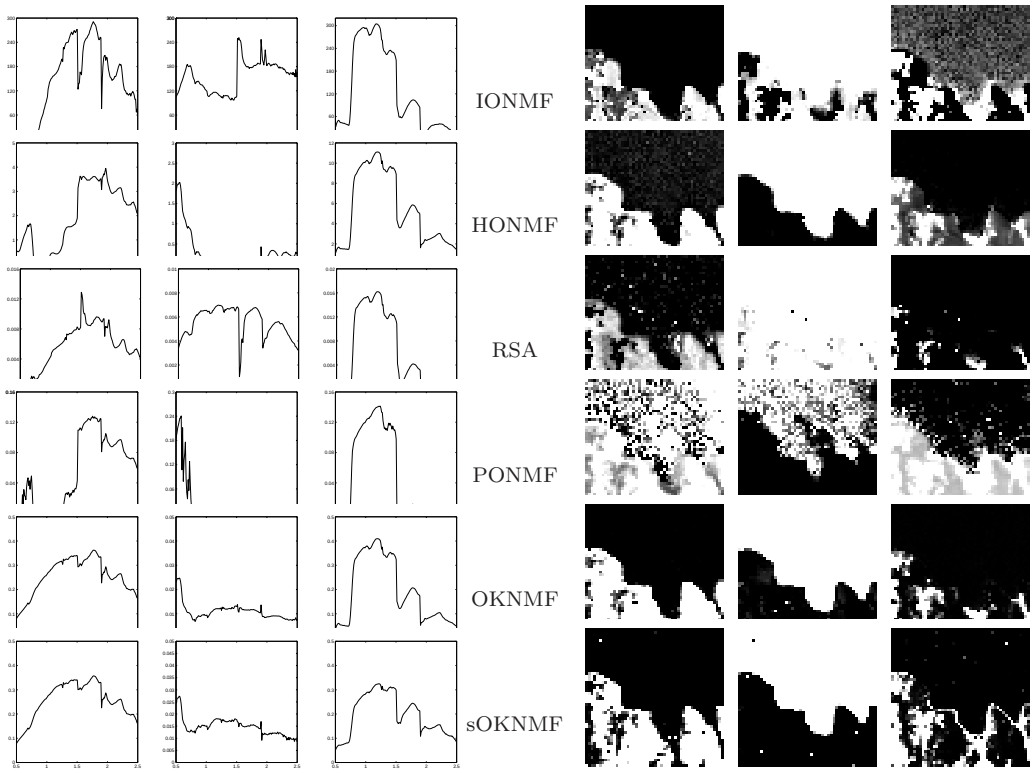


Figure 7: Left to right: estimated endmembers (soil, water, vegetation) and their corresponding abundance maps on the Moffett image. Top to bottom: IONMF, HONMF, RSA, PONMF, OKNMF with ASGD, and sOKNMF with $\beta = 0.1s = 0.41$.

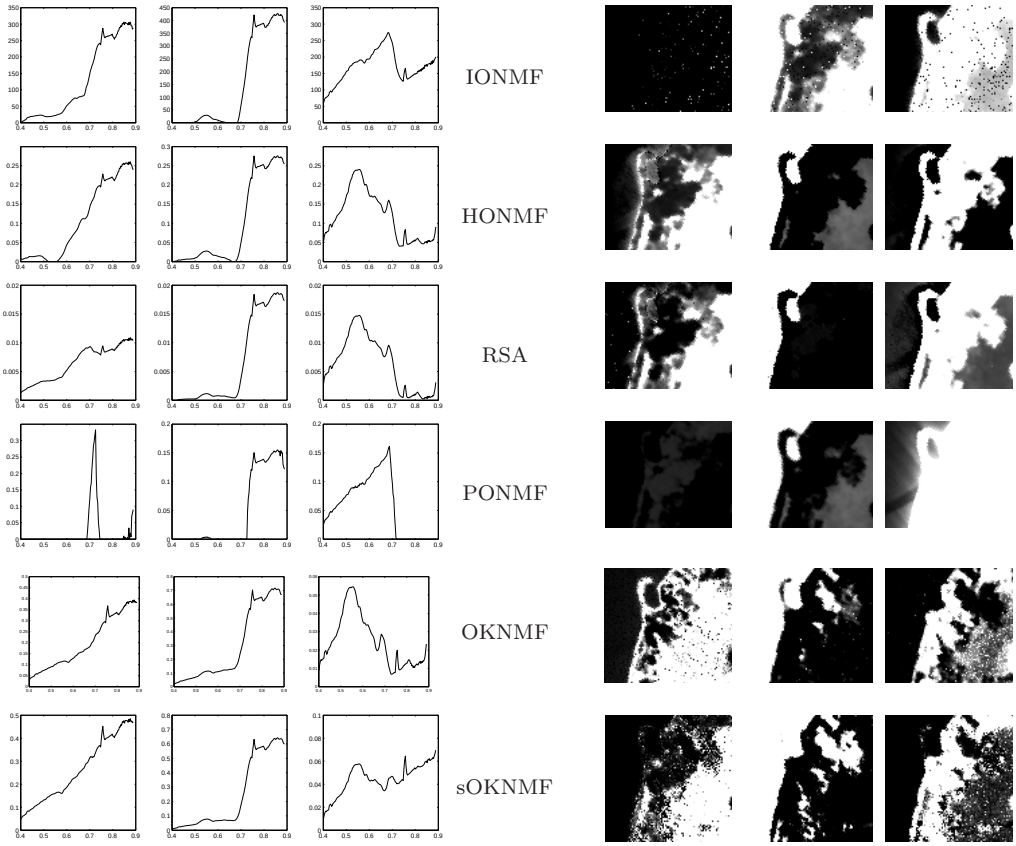


Figure 8: Results on the Samson image. Same legend as Figure 7, with $\beta = 0.1 s = 0.21$.

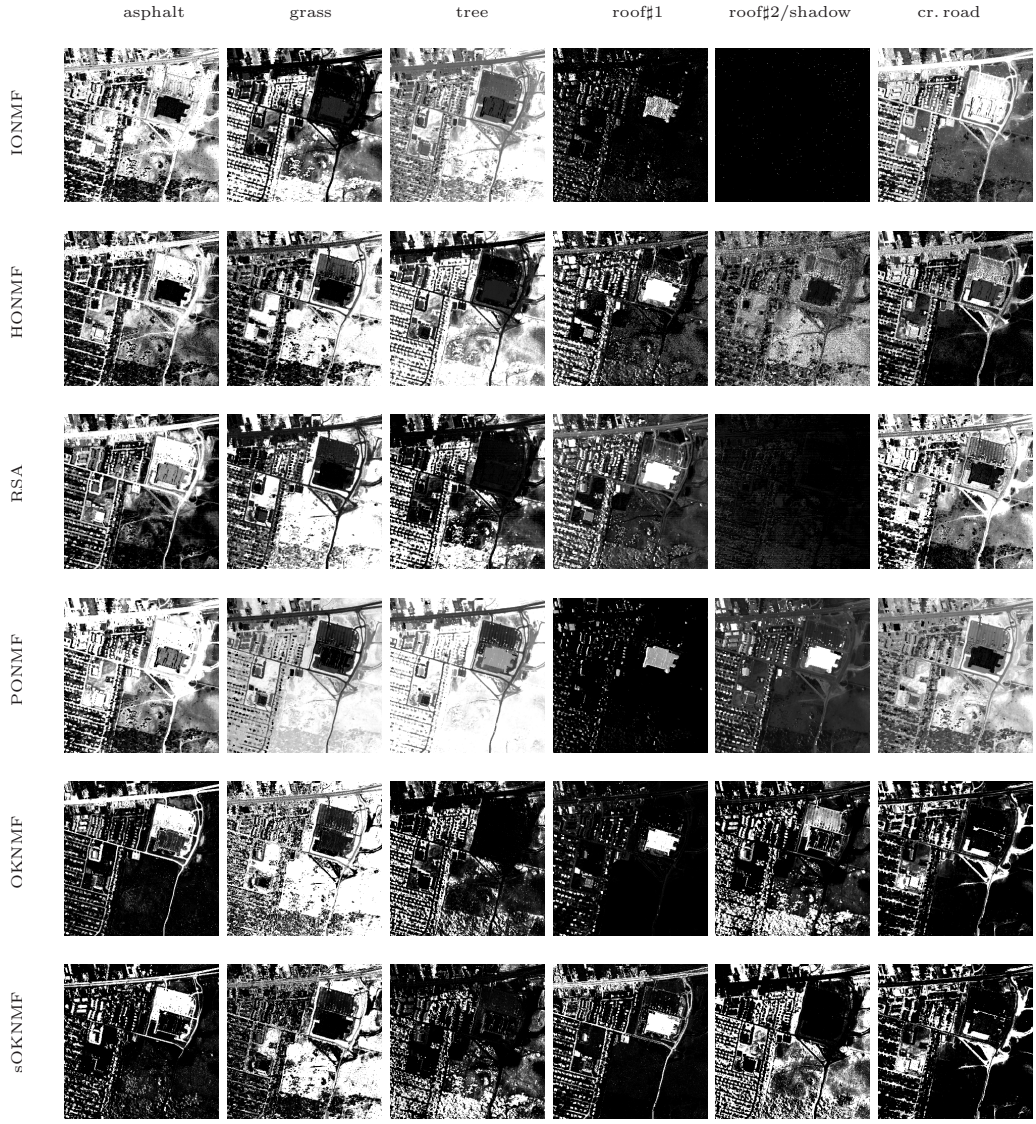


Figure 9: Estimated abundance maps of the Urban image. Left to right: asphalt road/parking, grass, tree, roof♯1, roof♯2/shadow, and concrete road. Top to bottom: IONMF, HONMF, RSA, PONMF, ASGD (OKNMF), and sOKNMF with $\beta = 0.1s = 0.19$.

[4] D. Zhang, Z. Zhou, S. Chen, Non-negative matrix factorization on kernels, in: Lecture Notes in Computer Science, Vol. 4099, Springer, 2006, pp. 404–412.

[5] C. Ding, T. Li, M. I. Jordan, Convex and Semi-Nonnegative Matrix Factorizations, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (1) (2010) 45–55.

[6] P. Honeine, C. Richard, Preimage problem in kernel-based machine learning, IEEE Signal Processing Magazine 28 (2) (2011) 77–88.

[7] F. Zhu, P. Honeine, Bi-objective nonnegative matrix factorization: Linear versus kernel-based models, IEEE Transactions on Geoscience and Remote Sensing (2016) 1–11.

[8] F. Zhu, P. Honeine, Pareto front of bi-objective kernel-based nonnegative matrix factorization, in: Proc. 23th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium, 2015, pp. 1–6.

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

[10] F. Zhu, P. Honeine, M. Kallas, Kernel nonnegative matrix factorization without the curse of the pre-image, arXiv preprint arXiv:1407.4420.

[11] S. S. Bucak, B. Gunsel, Incremental subspace learning via non-negative matrix factorization, Pattern Recognition 42 (5) (2009) 788 – 797.

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

[13] N. Guan, D. Tao, Z. Luo, B. Yuan, Online nonnegative matrix factorization with robust stochastic approximation, IEEE Transactions on Neural Networks and Learning Systems 23 (7) (2012) 1087–1099.

[14] B. Cao, D. Shen, J. Sun, X. Wang, Q. Yang, Z. Chen, Detect and track latent factors with online nonnegative matrix factorization., in: IJCAI, Vol. 7, 2007, pp. 2689–2694.

[15] G. Zhou, Z. Yang, S. Xie, J. Yang, Online blind source separation using incremental nonnegative matrix factorization with volume constraint, IEEE Transactions on Neural Networks 22 (4) (2011) 550–560.

[16] J. Mairal, F. Bach, J. Ponce, G. Sapiro, Online learning for matrix factorization and sparse coding, The Journal of Machine Learning Research 11 (2010) 19–60.

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

[18] J. Mairal, F. Bach, J. Ponce, G. Sapiro, Online dictionary learning for sparse coding, in: Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 689–696.

[19] N. Wang, J. Wang, D. Y. Yeung, Online robust non-negative dictionary learning for visual tracking, in: Computer Vision (ICCV), 2013 IEEE International Conference on, IEEE, 2013, pp. 657–664.

[20] J. Ho, Y. Xie, B. Vemuri, On a nonlinear generalization of sparse coding and dictionary learning, in: Proceedings of The 30th International Conference on Machine Learning, 2013, pp. 1480–1488.

[21] F. Zhu, P. Honeine, Online nonnegative matrix factorization based on kernel machines, in: Proceedings of 23rd European Signal Processing Conference (EUSIPCO), 2015, pp. 2381–2385.

[22] C. Lin, Projected gradient methods for nonnegative matrix factorization, Neural computation 19 (10) (2007) 2756–2779.

[23] B. T. Polyak, A. B. Juditsky, Acceleration of stochastic approximation by averaging, SIAM Journal on Control and Optimization 30 (4) (1992) 838–855.

[24] W. Xu, Towards optimal one pass large scale learning with averaged stochastic gradient descent, arXiv preprint arXiv:1107.2490.

[25] T. Virtanen, Sound source separation using sparse coding with temporal continuity objective, in: Proceedings of International Computer Music Conference, ICMC, Vol. 3, 2003, pp. 231–234.

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

[27] A. Huck, M. Guillaume, J. Blanc-Talon, Minimum dispersion constrained nonnegative matrix factorization to unmix hyperspectral data, IEEE Transactions on Geoscience and Remote Sensing 48 (6) (2010) 2590–2602.

[28] A. Halimi, Y. Altmann, N. Dobigeon, J. Y. Tourneret, Nonlinear unmixing of hyperspectral images using a generalized bilinear model, IEEE Transactions on Geoscience and Remote Sensing 49 (11) (2011) 4153–4162.

[29] N. Yokoya, J. Chanussot, A. Iwasaki, Nonlinear unmixing of hyperspectral data using semi-nonnegative matrix factorization, IEEE Transactions on Geoscience and Remote Sensing 52 (2) (2014) 1430–1437.

[30] J. M. Bioucas-Dias, J. M. P. Nascimento, Hyperspectral subspace identification, IEEE Transactions on Geoscience and Remote Sensing 46 (8) (2008) 2435–2445.

[31] Y. Altmann, A. Halimi, N. Dobigeon, J. Y. Tourneret, Supervised nonlinear spectral unmixing using a postnonlinear mixing model for hyperspectral imagery, IEEE Transactions on Image Processing 21 (6) (2012) 3017–3025.

[32] Y. Qian, S. Jia, J. Zhou, A. Robles-Kelly, Hyperspectral unmixing via sparsity-constrained nonnegative matrix factorization, IEEE Transactions on Geoscience and Remote Sensing 49 (11) (2011) 4282–4297.

[33] F. Zhu, Y. Wang, B. Fan, S. Xiang, G. Meng, C. Pan, Spectral unmixing via data-guided sparsity, IEEE Transactions on Image Processing 23 (12) (2014) 5412–5427.