Mach Learn (2014) 97:205–226
DOI 10.1007/s10994-014-5463-y

SAGA: sparse and geometry-aware non-negative matrix factorization through non-linear local embedding

Nicolas Courty · Xing Gong · Jimmy Vandel · Thomas Burger

Received: 19 January 2014 / Accepted: 30 June 2014 / Published online: 13 August 2014

Abstract This paper presents a new non-negative matrix factorization technique which (1) allows the decomposition of the original data on multiple latent factors accounting for the geometrical structure of the manifold embedding the data; (2) provides an optimal representation with a controllable level of sparsity; (3) has an overall linear complexity allowing handling in tractable time large and high dimensional datasets. It operates by coding the data with respect to local neighbors with non-linear weights. This locality is obtained as a consequence of the simultaneous sparsity and convexity constraints. Our method is demonstrated over several experiments, including a feature extraction and classification task, where it achieves better performances than the state-of-the-art factorization methods, with a shorter computational time.

Keywords Non-negative matrix factorization · Manifold sampling · Kernel methods · Sparse projections · Simplex methods · Convexity constraints

Editors: Toon Calders, Rosa Meo, Floriana Esposito, and Eyke Hüllermeier.

N. Courty (✉)
IRISA, Université de Bretagne Sud, Vannes, France
e-mail: ncourty@irisa.fr

X. Gong
Costel, Université de Rennes 2, Rennes, France

X. Gong
NLPR, Institute of Automation, Chinese Academy of Science, Beijing,
People's Republic of China
e-mail: xgong.nlpr@gmail.com

J. Vandel · T. Burger
CEA (iRSTV/BGE), INSERM (U1038), CNRS (FR3425),
Université Grenoble-Alpes, Grenoble, France
e-mail: Jimmy.vandel@cea.fr

T. Burger
e-mail: Thomas.burger@cea.fr

1 Introduction

1.1 Context

Non-negative matrix factorization (or NMF for short) has long been studied and used as a powerful data analysis tool providing a basis for numerous processing, such as dimensionality reduction, clustering, denoising, unmixing, etc. This article is concerned with a variant of NMF, where one tries to decompose a given matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]_{(\mathbf{x}_i \in \mathbb{R}^p)}$ formed by a set of n samples described with p variables ($\in \mathbb{R}^p$) as a product of two matrices $\mathbf{F} \in \mathbb{R}^{p \times \ell}$ and $\mathbf{G} \in \mathbb{R}^{n \times \ell}$ such that $||\mathbf{X} - \mathbf{F}\mathbf{G}^\top||^2$ is minimized, under the constraints that the elements of $\mathbf{G}$ are positive or nil, and that its rows sum to one. In other words, $\mathbf{G}$ serves as a convex surrogate description of $\mathbf{X}$ in the reduced embedding formed by ℓ prototypes (the columns of $\mathbf{F}$).

This convexity property on $\mathbf{G}$ is desirable in a lot of applications for its relations with the physics of the underlying observations: in this acceptation, the datapoints are usually observations of a mixing process where the components of the mixture are not known. Hyperspectral images are a good example, as each pixel describes a spectrum that can be defined as a combination of pure materials spectra (trees, concrete, water, etc.); the combination occurring because of the captor spatial resolution and different scattering effects (Esser et al. 2012; Bioucas-Dias et al. 2012). Other examples of applications are found in archetypal analysis (Mørup and Hansen 2012), or biology (Kersting et al. 2012).

1.2 Notations

Bold capital letters, such as $\mathbf{X}$ or $\mathbf{M}$ refer to matrices. The transposition operator is denoted $\cdot^\top$ (so that $\mathbf{M}^\top$ refers to the transpose of $\mathbf{M}$). The set of $m \times n$ real matrices (respectively real matrices with nonnegative entries) is denoted $\mathbb{R}^{m \times n}$ (respectively $\mathbb{R}_+^{m \times n}$). $\mathbf{I}_n$ is the $n \times n$ identity matrix, and $||\cdot||$ refers to the Frobenius norm. For any matrix $\mathbf{M}$, $\mathbf{M}_{\bullet i}$ (respectively $\mathbf{M}_{i\bullet}$) corresponds to the ith column (respectively ith row) of $\mathbf{M}$. However, in the particular case of $\mathbf{X}$, each column $\mathbf{X}_{\bullet i}$ corresponds to the datum $\mathbf{x}_i$, so that this latter more intuitive notation is preferred. $\Delta^n(\mathbf{M})$ is a $(n-1)$-simplicial polytope formed by n columns of $\mathbf{M}$. The unit $(n-1)$-simplex $\Delta^n(\mathbf{I}_n)$ is simply referred to as Δ^n.

1.3 Related work

Despite appealing properties, NMF presents a number of difficulties: First, while real data are often embedded on complex manifolds, the seminal convex NMF formulation prohibits a large number of latent factors that would explain the data in a geometry preserving way; Second, the level of sparsity in the final embedding (the matrix $\mathbf{G}$) is rarely controllable, and far from data analyst expectations; finally, the computational time is prohibitive when it comes to handling very large size matrices. Even if to date, various state-of-the-art methods already tackle one of or two of these issues, no method addresses these three issues all together.

The convexity constraint we consider on $\mathbf{G}$ differs from that of what is classically referred to as Convex NMF (Ding et al. 2010), i.e. NMF where the columns of $\mathbf{F}$ are convex combinations of columns of $\mathbf{X}$. However, our problem is also not unheard of in the literature, as, beyond its easiness of interpretation, it provides a very appealing computational framework:

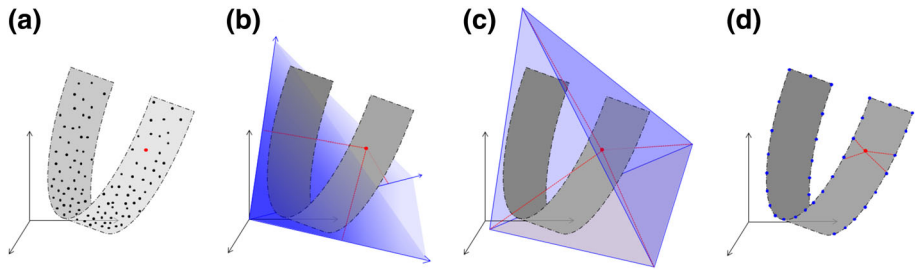

Fig. 1 **a** A dataset **X** embedded in a U-shaped manifold; **b** the separability assumption assumes the dataset is encompassed in a cone spanned by the columns of **F**; a datum is described in a column of **G** through its non-negative coordinates in the cone; **c** Convex hull: the dataset is embedded in a simplex, and each datum is described with barycentric coordinates; **d** Manifold hull: one uses a great enough number of reference points to precisely characterize the manifold, while forcing the sparsity to achieve some local coding. Note that, in order to make these imaged representations clearer, we did not represent **F** as a CSS of **X**

if **G** encodes convex combinations, the ℓ columns of **F** are expected to characterize a hull of **X** in $\mathbb{R}^p$, as data points lying within are perfectly reconstructed. Based on this idea, it is possible to separate the NMF problem into two steps, namely the computation of **F** (in order to determine, or to best approximate the hull of the dataset) and of **G** (which corresponds to the projection of the dataset onto the region bound by the hull). From a computational point of view, this is really interesting, as it is now possible to decompose the NMF (which amounts to finding two matrices with close relationship) into two simple problems, each focused on a particular matrix: first one computes **F** regardless **G**, and second, one computes **G** by projection of the columns of **X** onto **F**. With such an approach, it is possible to expect a lower complexity than that of elder methods based on singular value decomposition (with a $o(n^3)$ complexity), or than that of demanding procedures that iteratively alternates between the minimization of the two matrices, such as in Ding et al. (2010).

Initially, **F** was first related to the *convex hull* of **X** (Cutler and Breiman 1994): Notably, numerous works in the remote sensing and hyperspectral imaging community (see Bioucas-Dias et al. 2012 as well as its references) have pointed out that **F** should be chosen so that $\Delta^\ell(\mathbf{F})$ best encompasses **X**, while other investigations focused on how to find this simplicial convex hull (Wang et al. 2010; Thurau et al. 2010; Çivril and Magdon-Ismail 2009). More recently, a series of works (Arora et al. 2012; Kumar et al. 2013; Gillis and Vavasis 2013; Recht et al. 2012) focused on the *conical hull* of **X**: Here, the idea is to find **F** so that it spans a cone encompassing the dataset. The reason of the recent focus on the conical hull is that it is the geometric translation of an important assumption in NMF, the *separability assumptions* proposed by Donoho and Stodden (2003). This separability assumption reads that (1) the residue $\mathbf{X} - \mathbf{F}\mathbf{G}^\top$ is nil, (2) columns of **F** are collinear to columns of **X**. This second condition regarding the collinearity of the columns of **F** and **X** is interesting beyond the separability assumption: whatever the type of hull defined by **F**, it is possible to assume that the components of the mixture belong to the data, and that any datum is a convex combination of a restricted number of particular selected datapoints (those forming **F**). This approach drastically reduces the complexity of finding **F** as this latter is simply defined by a *column subset selection* (CSS) of **X**.

Figure 1 illustrates a dataset lying on a non-linear manifold resulting from some hidden factors that could be of interest, as well as its conical and convex hulls. It clearly appears that the geometries of these hulls are not adapted to that of the dataset. So far, there has been little interest in trying to characterize the boundary of the manifold dataset in spite of

its non-convexity; in the sequel, we shall address this boundary with the shorter and imaged name *manifold hull*, that is illustrated on Fig. 1d. Naturally, precisely characterizing such a manifold hull would require to increase ℓ, the number of datapoints involved in the CSS; which stresses to a larger extend the need for an adapted control of the sparsity level that is already sought for in numerous applications: Each point should be described as a convex combination of a restricted number λ of prototype points among the ℓ of the CSS.

Whatever the type of hull (convex or conical), separating $\mathbf{F}$ and $\mathbf{G}$ computations has drastically reduced the overall NMF complexity of the state-of-the-art methods: Thurau et al. (2010, 2012) compute $\mathbf{F}$ in linear time and few works (Gillis and Vavasis 2013; Recht et al. 2012) even reach overall linear-complex NMF. However, among them, none allows characterizing the manifold hull, and none allows controlling the solution sparsity. Even if to date, more computationally efficient methods than adding the classical L^1 penalty (Hoyer 2004; Esser et al. 2012) have been developed, such as Kim and Park (2007), Gillis (2012), none reaches a linear complexity.

Even if the characterization of the manifold hull has never been addressed so far, it is well-known that the kernel trick is an efficient way to provide a manifold preserving description of a dataset. In fact, several pre-existing works applied the kernel trick to NMF (Zhang et al. 2006; Buciu et al. 2008; Cai et al. 2011). However, the convexity constraint on $\mathbf{G}$ (and consequently the notion of manifold hull) does not appear. Moreover, neither sparsity nor any linear complexity is achieved.

1.4 Proposal

In this article, we present SAGA (*Sparse And Geometry-Aware*), a new NMF method avoiding the aforementioned limits. It operates in a Reproducing Kernel Hilbert Space (RKHS) (Schölkopf and Smola 2002), where $\mathbf{F}$ is defined by a CSS. Then, according to an expected sparsity level, $\mathbf{G}$, the best dataset projection onto the simplex formed by $\mathbf{F}$ is computed. The advantages of our method are:

First, kernelization is interesting to several extents: (i) it makes the algorithm compliant with dataset where only relationships among objects are available; (ii) its regularization property improves robustness to noise; (iii) it allows using more latent factors than the dimensions of the input data, providing an insightful tool to consider the geometric structure of the data manifold. As both the number of latent factors and the sparsity level are increasing, it appears that the locality of the support of $\mathbf{G}$ is increasing, turning the factorization problem in a *non-linear local embedding* of the original data. As shown in some recent works (Yu et al. 2009; Guillemot and Turkan 2012), this kind of embedding is powerful to describe the non-linear structure of the data manifold, and as such serve as a very good feature extraction framework.

Second, the CSS is defined thanks to a manifold subsampling method (Shroff et al. 2011), which reaches a linear complexity with respect to n, the size of the dataset.

Third, the computation of $\mathbf{G}$ corresponds to a sparse RKHS simplex projection, which is a non-linear optimization problem. Based on recent advances on sparse projected gradient methods, the projection is solved with an algorithm of linear complexity (with respect to n), while naturally embedding sparsity mechanism control.

Forth and finally, since both computations of $\mathbf{F}$ and $\mathbf{G}$ are linear, the overall complexity of SAGA is linear. This makes this algorithm perfectly suitable for very big data.

To the best of our knowledge, no state-of-the-art method simultaneously compiles all these advantages.

1.5 Contributions

SAGA has been developed on the top of several state-of-the-art algorithms, thanks to several technical extensions that are listed here. The computation of matrix **F** is largely inspired by the simplex volume maximization (SiVM) approach of Thurau et al. (2012). However, the method itself has been extended in several manners:

1. It is slightly generalized in order to operate in the Hilbert space reproducing the Gaussian kernel, rather than in $\mathbb{R}^p$.
2. Most importantly, while SiVM proposes an approximate optimization (for several simplifications are made to reach a linear complexity), we propose another solution to the simplex volume maximization which performs an exact optimization.

In spite of these two extensions, our method remains of linear complexity, so that finally, the computation of **F** with SiVM is both a particular case and an approximation of the one produced by SAGA. Then, the computation of matrix **G** is not inspired by any other NMF technique, but is based on a recent sparse projected gradient strategy (Kyrillidis et al. 2013):

3. This latter is adapted to the projection over the CSS in the RKHS, and its linear complexity is kept.
4. We provide with theoretical bounds on the convergence of the projector (linked to the kernel bandwidth, the minimum pairwise distance between the CSS and the sparsity level).

1.6 Outline

We solve our NMF problem by separating the computations of the CSS matrix and of the projection matrix. This is why, Sects. 2 and 3 focus on the computations of **F** and **G** respectively. More specifically, the structure of Sect. 2 is the following: First a small introductive paragraph recalls the basics of the kernel trick, as well as why it is interesting to operate in a RKHS to fulfill our objectives. Then, Sect. 2.1 investigates the consequences of working in such RKHS; They lead us to a particular strategy, which is to characterize the manifold hull of the dataset in $\mathbb{R}^p$ via a simplicial convex hull in the RKHS; and to define this latter with a manifold sampling strategy. At this point, computational constraints direct us toward SiVM-like procedures rather than toward more resource-demanding ones. Section 2.2 jointly presents SiVM, such as defined in the literature (Thurau et al. 2012), as well as the kernel generalization we propose, while Sect. 2.3 describes our modifications to reach exact maximal simplex volume. In a similar way, Sect. 3 is divided into two parts: the first one (Sect. 3.1) explains how projecting the image in the RKHS of any datum $\mathbf{x}_i$ onto the image of the CSS in the RKHS, along with sparsity constraints; Sect. 3.2 provides with a proof that the projector defined in Sect. 3.1 converges. As it is established in the literature (Garg and Khandekar 2009) that the Restricted Isometry Property (RIP, Candes 2008) implies the convergence, we prove that our projector respect the RIP for some particular tuning of its parameters. At this stage, most of the required mathematics is established. Then, in Sect. 4, one summarizes the entire procedure made of the concatenation of the two matrices computation through an easy to implement algorithm, completed by some theoretical assessments of the linear complexity of the algorithm. Finally, Sect. 5 is devoted to experimental validations. In the first series of experiments, one focuses on toy examples, in order to illustrate the behavior of the manifold hull. Then, follow several experiments on simulated datasets, in order to compare the computational efficiency and the computational precision of SAGA with respect to the

state-of-the-art. Finally, we consider real datasets through challenging image classification tasks.

2 Geometry aware CSS procedure

Despite living in $\mathbb{R}^p$, $\mathbf{X}$ may span a nonlinear manifold of intrinsic dimensionality lower than p. A major problem is thus to extend to this nonlinear manifold the classical statistics that are used to work in a vector space. To do so, an essential element is to replace the Euclidean distances by geodesic distances. Depending on the manifold, the associated metric may be difficult to formally define. However, it is possible to characterize it through the time-scale of the well-studied heat diffusion process, the kernel formulation of which is well approximated by the Gaussian kernel (Lafferty and Lebanon 2005; Lafon and Lee 2006): The geometry of the manifold in which $\mathbf{X}$ lies is captured by $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2\right)$ with variance σ^2. Let us call $\phi(\cdot)$ the implicit feature map from $\mathbb{R}^p$ onto $\mathcal{H}$, the RKHS associated to $k(\cdot,\cdot)$. We use the shorthand notation $\Phi = \phi(\mathbf{X}) = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)]_{(\phi(\mathbf{x}_i)\in\mathcal{H})}$. Then, following the notations of Ding et al. (2010), the SAGA solution amounts to finding the indicator matrix $\mathbf{W}^{(\ell)} \in \{0,1\}^{n\times\ell}$ defining the CSS[1] and the projection matrix $\mathbf{G}$ where $\phi(\mathbf{F}) = \phi(\mathbf{X}\mathbf{W}^{(\ell)}) = \Phi\mathbf{W}^{(\ell)}$, such that $||\Phi - \Phi\mathbf{W}^{(\ell)}\mathbf{G}^\top||^2$ is minimized under convexity constraints.

2.1 CSS as a manifold sampling procedure

Let us start by a basic remark,

Remark 1 In the input space, we must have $\ell \leq n$ and $\ell \leq p$. In the Gaussian RKHS, one still has $\ell \leq n$, however, $\ell > p$ becomes possible, for each sample spans its own dimension.

which leads to the following property:

Property 1 (Non-Separability) Separability assumption does not hold for NMF in the Gaussian RKHS.

Proof According to Remark 1, any datum not included in $\mathbf{X}\mathbf{W}$ cannot be expressed as a linear combination of elements of $\phi(\mathbf{X}\mathbf{W})$. □

Thus, one should not consider conical hull in the RKHS. However, Remark 1 leads to:

Corollary 1 *In the Gaussian RKHS, it is possible to use more than p points in the CSS. The latter forms a non-simplicial polytope in $\mathbb{R}^p$ while their image in the Gaussian RKHS is a simplex.*

In other words, the manifold hull of $\mathbf{X}$ can be characterized through the simplicial convex hull $\Delta^\ell(\phi(\mathbf{F}))$. Then, it follows that:

Corollary 2 *No sample lies in $\Delta^\ell(\phi(\mathbf{F}))$ and all the samples will be projected on hyperfaces of $\Delta^\ell(\phi(\mathbf{F}))$, leading to approximate reconstructions. Yet, such an approximation comes with the appealing property of sparsity, discussed later in the article.*

[1] We write $\mathbf{W}$ instead of $\mathbf{W}^{(\ell)}$ when ℓ does not matter, or is implicit regarding the context.

At this point, finding $\Delta^\ell(\phi(\mathbf{F}))$ amounts to finding $\mathbf{W}$, which turns out to subsample the boundary of the data manifold. Most of the methods from the literature address it with the objective of maximizing the representation of the dataset, while here, we are interested in its boundary, which makes the sampling completely different: For instance, a kernel k-means sampling, although very efficient to subsample a given manifold (Lafon and Lee 2006), leads to Convex NMF of Ding et al. (2010), the aim of which is completely different of ours.

However, our problem is not completely unheard of: In Shroff et al. (2011), the authors consider maximizing the diversity of the selected samples using a Karcher variance[2] maximization criterion. Alternative formulation exists, where one seeks for the maximum volume parallelepiped in the data matrix (Çivril and Magdon-Ismail 2009). Interestingly enough, whatever the interpretation (diversity or volume criterion), the corresponding computation can reduce to recursive QR decompositions of the data matrix (Gillis and Vavasis 2013; Shroff et al. 2011). However, in the RKHS, since no explicit coordinates are available, those methods cannot be transposed. The recent proposal of Courty and Burger (2013) regarding a kernel rank revealing Cholesky technique is also of interest, unfortunately, it fails to scale up to big data, because it implies a computationally demanding decomposition at each selection step.

Finally, among all the methods available in the literature, if one discards those (1) which do not sample the boundary, (2) which cannot be conducted in a RKHS, (3) which do not have a linear complexity, we are aware of a single remaining method: the Simplex Volume Maximization (SiVM) (Thurau et al. 2012).

2.2 Original simplex volume maximization

We begin with a review of the original formulation of Thurau et al. (2012) and its direct transposition to the kernel framework. SiVM tries to maximize $\mathrm{Vol}(\Delta^\ell(\mathbf{F}))$, the volume of the simplex spanned by $\Delta^\ell(\mathbf{F})$, which reads:

$$\mathrm{Vol}(\Delta^\ell(\mathbf{F})) = \sqrt{\frac{-1^\ell \cdot \mathrm{cmd}(\mathbf{F})}{2^{\ell-1}(\ell-1)!}}, \text{ with } \mathrm{cmd}(\mathbf{F}) = \det \begin{pmatrix} 0 & 1 & 1 & 1 & \dots & 1 \\ 1 & 0 & d^2_{1,2} & d^2_{1,3} & \dots & d^2_{1,\ell} \\ 1 & d^2_{2,1} & 0 & d^2_{2,3} & \dots & d^2_{2,\ell} \\ 1 & d^2_{3,1} & d^2_{3,2} & 0 & \dots & d^2_{3,\ell} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & d^2_{\ell,1} & d^2_{\ell,2} & d^2_{\ell,3} & \dots & 0 \end{pmatrix}. \quad (1)$$

$\mathrm{cmd}(\mathbf{F})$ is the Cayley–Menger determinant of $\mathbf{F}$ (i.e. the determinant of the matrix accounting for the pairwise distances in $\mathbf{F}$) and $d^2_{i,j}$ is the square Euclidean distance between elements $\mathbf{x}_i$ and $\mathbf{x}_j$. This obviously transposes to the RKHS: The volume of $\Delta^\ell(\Phi\mathbf{W})$ can be expressed by replacing the Cayley–Menger determinant by that of a matrix accounting for pairwise distances in $\mathcal{H}$:

$$\mathrm{Vol}(\Delta^\ell(\Phi\mathbf{W})) = \sqrt{\frac{-1^\ell}{2^{\ell-1}(\ell-1)!}\det(\mathbf{A})} \quad (2)$$

with $\mathbf{A}$ similar to the matrix of Eq. (1) except that $\forall i, j \leq \ell$, $d^2_{i,j}$ is replaced by:

$$\mathbf{A}_{(i+1)(j+1)} = ||\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)||^2 = k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{x}_j, \mathbf{x}_j) - 2k(\mathbf{x}_i, \mathbf{x}_j)$$

The comparison of the volume of the simplices spanned by all the possible CSS is computationally prohibitive. Thus, an approximate search in linear time of the best simplex is

[2] The Karcher variance is a variance accounting for Riemannian distances over the manifold, rather than Euclidean ones.

proposed in Thurau et al. (2012). This is possible thanks to two tricks: The first one is a strong result (Theorem 1 of Thurau et al. 2012), which states that if one adds an element to the simplex, it can only make the reconstruction error smaller or equal. Thus, a simple greedy algorithm can be used to compute $\mathbf{W}$: Starting with the best 1-simplex (i.e. $\mathbf{W}$ has 2 columns), one adds the best third sample (according to the maximization of Eq. 1), then the fourth, and so on until ℓ samples are selected. Practically, this procedure can be transposed to the RKHS: At iteration $p-1$ one selects the element $\phi(\mathbf{x}_i)$ of Φ such that

$$i = \arg\max_q \operatorname{Vol}\left(\Delta^p(\Phi\mathbf{W}) \cup \phi(\mathbf{x}_q)\right). \tag{3}$$

However, this procedure still requires the computation of several Cayley–Menger determinants (each of them being computationally intensive).

At this point shows up the second trick: If one makes the simplifying assumption that the distances between the elements of the CSS defined in the previous iteration are constant and noted a, and if $\alpha_{j,q} = d_{j,q}^2/2$, then, Eq. 3 amounts to finding $\phi(\mathbf{x}_i)$ such that

$$i = \arg\max_q \left[\sum_{k=1}^{p} \alpha_{kq} \cdot \left(a^2 + 2\sum_{j=k+1}^{p} \alpha_{jq}\right) - (p-1) \cdot \sum_{k=1}^{p} \alpha_{kq}^2\right], \tag{4}$$

where $d_{j,q}$ refers to the distance between a point of the CSS $\mathbf{x}_j$ and a point out of the CSS $\mathbf{x}_q$ which is considered for adjunction to the CSS (thus, in the RKHS, one has $\alpha_{j,q} = 1 - k(\mathbf{x}_j, \mathbf{x}_q)$). Finally, the computation of Eq. 4 is sped up by considering that $d_{i,q}^2 \approx d_{i,q}$ and $a^2 \approx a$, leading to

$$i = \arg\max_q \left[\sum_{k=1}^{p} d_{kq} \cdot \left(a + \sum_{j=k+1}^{p} d_{jq}\right) - \frac{p-1}{2} \cdot \sum_{k=1}^{p} d_{kq}^2\right]. \tag{5}$$

Remark 2 Naturally, the magnitude to which the constant distance assumption is violated strongly depends on the dataset. As a consequence, the instantiation of Eq. 3 into 5 may lead to some approximations as already noted in Gillis and Vavasis (2013), where Gillis and Vavasis showed that SiVM can underperform on ill-conditionned datasets.

Despite this remark, the general principle advocated in Eq. 3 remains unalterably valid. This is why, we rely on it to propose an alternative to SiVM, which provides exact volume computation, with a similar linear complexity.

2.3 Exact simplex volume maximization

Let us consider a simplex Δ^p of dimensionality $p-1$ spanned by a subset $\mathbf{XW}^{(p)} = \{\mathbf{x}_1, \ldots, \mathbf{x}_p\}$ of p points of $\mathbf{X}$. If we add a $(p+1)$th point $\mathbf{x}_i \in \mathbf{X} \backslash \mathbf{XW}^{(p)}$ to the simplex, the new volume is given by:

$$\operatorname{Vol}(\Delta^{p+1}) = \frac{\operatorname{Vol}(\Delta^p) \times \operatorname{dist}(\mathbf{x}_i, \mathbf{XW}^{(p)})}{p} \tag{6}$$

where $\operatorname{dist}(\mathbf{x}_i, \mathbf{XW}^{(p)})$ is the distance between $\mathbf{x}_i$ and its projection onto the subspace spanned by $\mathbf{XW}^{(p)}$. According to Courty et al. (2011), in $\mathcal{H}$, this distance reads:

$$\operatorname{dist}(\phi(\mathbf{x}_i), \Phi\mathbf{W}^{(p)}) = 1 - \left(\mathbf{k}_{\mathbf{x}_i}^\top \cdot \mathbf{K}_p^{-1} \cdot \mathbf{k}_{\mathbf{x}_i}\right) \tag{7}$$

where $\mathbf{K}_p^{-1}$ is the inverse of the kernel matrix of the elements of the CSS, i.e. $\mathbf{K}_p^{-1} = (\mathbf{W}^{(p)\top}\Phi^\top\Phi\mathbf{W}^{(p)})^{-1}$, and where $\mathbf{k}_{\mathbf{x}_i}$ is a vector of length p such that $\mathbf{k}_{\mathbf{x}_i} = [k(\mathbf{x}_j, \mathbf{x}_i)]_{\mathbf{x}_j \in \mathbf{X}\mathbf{W}^{(p)}}$. Then, it is possible to use a greedy procedure similar to that of SiVM, where Eq. 3 translates into:

$$\begin{aligned} i &= \arg\max_q \frac{\mathrm{Vol}(\Delta^p(\Phi\mathbf{W}^{(p)})) \times \mathrm{dist}(\phi(\mathbf{x}_q), \Phi\mathbf{W}^{(p)})}{p} \\ &= \arg\min_q \left[\mathbf{k}_{\mathbf{x}_q}^\top \cdot \mathbf{K}_p^{-1} \cdot \mathbf{k}_{\mathbf{x}_q}\right]. \end{aligned} \tag{8}$$

Remark 3 This procedure tends to add datum that most changes the geometry of the manifold spanned by the CSS. As such, it acts as the spectral sampling procedure proposed in Öztireli et al. (2010) for sampling a 3D mesh in a computer graphics context.

Even if $\mathbf{K}_p^{-1}$ is computed a single time at each iteration (it does not depend on $\mathbf{x}_q$), a matrix inversion remains a resource demanding operation. Moreover, if ℓ elements are to be selected, then ℓ inversions of matrices of increasing size are expected. Fortunately, it is possible to bypass this inversion, by iteratively constructing $\mathbf{K}_p^{-1}$ on the basis of the Schur complement (Boyd and Vandenberghe 2004). Once i, the index of the best $(p+1)$th point to add to the CSS is defined (Eq. 8), one computes

$$\begin{aligned} \mathbf{K}_{p+1}^{-1} &= \begin{bmatrix} \mathbf{K}_p & \mathbf{k}_{\mathbf{x}_i} \\ \mathbf{k}_{\mathbf{x}_i}^\top & 1 \end{bmatrix}^{-1} = \mathcal{K} \cdot \begin{bmatrix} \left(1 - \mathbf{k}_{\mathbf{x}_i}^\top \cdot \mathbf{K}_p^{-1} \cdot \mathbf{k}_{\mathbf{x}_i}\right)^{-1} & \mathbf{0}_p^\top \\ \mathbf{0}_p & \mathbf{K}_p^{-1} \end{bmatrix} \cdot \mathcal{K}^\top \\ &\text{with } \mathcal{K} = \begin{bmatrix} -\mathbf{K}_p^{-1} \cdot \mathbf{k}_{\mathbf{x}_i} & \mathbf{I}_p \\ 1 & \mathbf{0}_p^\top \end{bmatrix} \end{aligned} \tag{9}$$

where $\mathbf{I}_p$ is the identity matrix of size p and $\mathbf{0}_p$ is a vector of p zeros. The computation works as long as $\mathbf{k}_{\mathbf{x}_i}^\top \cdot \mathbf{K}_p^{-1} \cdot \mathbf{k}_{\mathbf{x}_i}$ differs from 1, which is always true in the Gaussian RKHS as long as the data points are separated.

3 Sparse projections onto the RKHS simplex

In this section, we focus on the computation of $\mathbf{G}$. We give the formulation of our sparse RKHS simplex projector, and then we discuss its convergence. We notably show some analytical bounds required for the convergence of the method.

3.1 Projection on the RKHS simplex

We search for the projection of any point $\phi(\mathbf{x}_i)$ onto the simplex $\Delta^\ell(\Phi\mathbf{W})$, i.e. the point of the simplex which minimizes the Euclidean distance to $\phi(\mathbf{x}_i)$. It amounts to solving the n independent problems of computing the rows of $\mathbf{G}$:

$$\mathbf{G}_{i\bullet} = \arg\min_{\mathbf{G}_{i\bullet}} ||\phi(\mathbf{x}_i) - \Phi\mathbf{W}\mathbf{G}_{i\bullet}^\top||^2 \text{ s. t. } \sum_j \mathbf{G}_{ij} = 1,\ \mathbf{G}_{ij} \geq 0, \forall j \tag{10}$$

The constraint $\sum_j \mathbf{G}_{ij} = 1,\ \mathbf{G}_{ij} \geq 0, \forall j$ is equivalent to have $\mathbf{G}_{i\bullet}$ in the unit standard simplex Δ^ℓ, ($\mathbf{G}$ encodes the barycentric coordinates of $\mathbf{X}$ in $\Delta^\ell(\Phi\mathbf{W})$). At this point, it is possible to force the sparsity of the projection to $\lambda < \ell$, without any extra computational

cost: We only need to replace the previous constraint by $\mathbf{G}_{i\bullet} \in \Delta^{\lambda}$. Thus, Eq. (10) reads:

$$\mathbf{G}_{i\bullet} = \arg\min_{\mathbf{G}_{i\bullet}} ||\phi(\mathbf{x}_i) - \Phi\mathbf{W}\mathbf{G}_{i\bullet}^{\top}||^2 \text{ s. t. } \mathbf{G}_{i\bullet} \in \Delta^{\lambda} \tag{11}$$

Instead of considering quadratic programming, such as in Thurau et al. (2012) or Kumar et al. (2013), we follow some recent work on the projection on the unit standard simplex (Kyrillidis et al. 2013), and we propose to use a simple projected gradient descent algorithm to solve Eq. (11), which amounts to iterating through different possible solutions of

$$\mathbf{G}_{i\bullet}^{t+1} = \mathcal{P}_{\lambda}\left(\mathbf{G}_{i\bullet}^{t} - \varepsilon_t \nabla(||\phi(\mathbf{x}_i) - \Phi\mathbf{W}{\mathbf{G}_{i\bullet}^{t}}^{\top}||^2)\right) \tag{12}$$

t being the iteration index, ε_t a (possibly varying) step size, $\nabla(\cdot)$ the gradient operator and $\mathcal{P}_{\lambda}(\cdot)$ the projector onto Δ^{λ}. This kind of projected gradient descent method has recently shown its computational efficiency and is also endowed with theoretical convergence guarantees (Garg and Khandekar 2009). The gradient reads (we omit the iteration index t for clarity):

$$\begin{aligned}\nabla(||\phi(\mathbf{x}_i) - \Phi\mathbf{W}\mathbf{G}_{i\bullet}^{\top}||^2) &= \nabla((\phi(\mathbf{x}_i) - \Phi\mathbf{W}\mathbf{G}_{i\bullet}^{\top})^{\top}(\phi(\mathbf{x}_i) - \Phi\mathbf{W}\mathbf{G}_{i\bullet}^{\top})) \\ &= \nabla(\mathbf{G}_{i\bullet}\mathbf{K}_{\ell}\mathbf{G}_{i\bullet}^{\top} - 2\mathbf{k}_{\mathbf{x}_i}\mathbf{G}_{i\bullet}^{\top} + k(\mathbf{x}_i, \mathbf{x}_i)), \\ &= 2(\mathbf{G}_{i\bullet}^{\top}\mathbf{K}_{\ell} - \mathbf{k}_{\mathbf{x}_i}).\end{aligned} \tag{13}$$

As for $\mathcal{P}_{\lambda}(\cdot)$, we rely on the Greedy Selector and Simplex Projector (GSSP) algorithm of Kyrillidis et al. (2013) which can be summarized as a two-step procedure: firstly the coordinates of the vector are sorted by magnitude, and then the λ greatest values are projected on the unit simplex Δ^{λ} (while the other vector entries are set to zero).

Remark 4 Since the GSSP procedure projects $\mathbf{G}_{i\bullet}$ on the subspace spanned by the ℓ columns of $\mathbf{F}$, the coordinates of $\mathbf{G}_{i\bullet}$ embeds the projection over each of the ℓ selected elements in the feature space, so that the sparsity support is chosen in the closest elements in the feature space.

The Gaussian kernel is monotonically decreasing according to the neighboring distance; This implies that for each datum, the sparsity support is made of the closest CSS elements in the input space. It follows that:

Property 2 (Non-linear local coding) The sparse RKHS simplex projector describes any element $\mathbf{x}_i$ of $\mathbf{X}$ with $\mathbf{G}_{i\bullet}$, which interprets as its non-linear (because of the kernel non-linearity) barycentric coordinates according to λ prototype points. These prototype points are found in the closest elements of the CSS, thus providing a non-linear local coding for $\mathbf{x}_i$.

3.2 Convergence of the projector

Finally, we establish the convergence of that projector, which ensures (Kyrillidis et al. 2013) that the final vector $\mathbf{G}_{i\bullet}$ is the best λ-sparse solution. To do so, we rely on Garg and Khandekar (2009), which states that for Eq. 11 to be minimized via the projected gradient approach, $\Phi\mathbf{W}$ has to satisfy λ-restricted isometry property (or λ-RIP for short), with $\delta_{\lambda} \in [0, 1[$. This latter reads:

Definition 1 The linear operator $\Phi\mathbf{W}$ respects the λ-*restricted isometry property* (Candes 2008) with constant δ_{λ} if

$$(1 - \delta_{\lambda})||\mathbf{x}||_2^2 \leq ||\Phi\mathbf{W}\mathbf{x}||_2^2 \leq (1 + \delta_{\lambda})||\mathbf{x}||_2^2 \tag{14}$$

where $||\cdot||_2$ refers to the L^2 norm, and for every λ-sparse vector $\mathbf{x}$.

Equivalently, the constant δ_λ can also be defined as:

$$\delta_\lambda := \max_{\substack{\mathcal{L}\subseteq\{1,\ldots,\ell\},\\ |\mathcal{L}|=\lambda}} ||(\Phi\mathbf{W})_{\mathcal{L}}^{\top}(\Phi\mathbf{W})_{\mathcal{L}} - \mathbf{I}_\lambda||_2 \tag{15}$$

where $(\Phi\mathbf{W})_{\mathcal{L}}$ denotes a subset matrix of $\Phi\mathbf{W}$, with λ columns corresponding to a subset $\mathcal{L}$ of cardinality λ picked up among the ℓ indices of the CSS. Thus, $(\Phi\mathbf{W})_{\mathcal{L}}^{\top}(\Phi\mathbf{W})_{\mathcal{L}}$ is simply the related Gram matrix, noted $\mathbf{K}_\lambda$. δ_λ is defined according to the subset providing a maximum among all the possible combinations of those columns.

As a matter of fact, such convergence holds for particular values of λ and σ, such as stated by the following proposition:

Proposition 1 *If* $\lambda > 2$, *and if* $\sigma < \frac{d_{\min}}{\sqrt{2\ln(\lambda-1)}}$, *then,* $\Phi\mathbf{W}$ *satisfies the* λ*-RIP with constant* $\delta_\lambda \in [0, 1[$. *If* $\lambda \leq 2$, *there is no particular bound to* σ.

Proof Let us first note that since we are working in the Gaussian RKHS, the columns of $\Phi\mathbf{W}$ have unit norms. The diagonal entries of $\mathbf{K}_\lambda$ are 1, and $\forall i, j \in \mathcal{L}$ the off-diagonal element (i, j) is $k(\mathbf{x}_i, \mathbf{x}_j)$.

Let ν_i be an eigenvalue of a matrix $\mathbf{A}$. By the Gershgorin circle theorem, we know that:

$$|\nu_i - \mathbf{A}(i,i)| < \sum_{j\leq\lambda, i\neq j} |\mathbf{A}(i,j)| \tag{16}$$

Thus, for $\mathbf{A} = \mathbf{K}_\lambda - \mathbf{I}_\lambda$, we obtain $|\nu_i| < \sum_{j\leq\lambda, i\neq j} k(\mathbf{x}_i, \mathbf{x}_j)$. Let μ be the greatest dot product between the elements of $(\Phi\mathbf{W})_{\mathcal{L}}$, i.e. $\mu := \max k(\mathbf{x}_i, \mathbf{x}_j), \forall i, j \in \mathcal{L}, i \neq j$. We can write:

$$\sum_{j\leq\lambda, i\neq j} k(\mathbf{x}_i, \mathbf{x}_j) \leq (\lambda - 1)\mu \tag{17}$$

and we have a bound for every eigenvalue ν_i and every subset $\mathcal{L}$ of cardinality λ. Thus, $\Phi\mathbf{W}$ follows the λ-RIP with $\delta_\lambda = (\lambda - 1)\mu$ (Bandeira et al. 2012). Let $d^2_{\min}$ be the minimum squared distance between two distinct elements of $\mathbf{XW}$, i.e. $d^2_{\min} = \min_{i,j,i\neq j} ||\mathbf{x}_i - \mathbf{x}_j||^2$. As $\mu = \exp\left(-d^2_{\min}/2\sigma^2\right)$, one has:

$$\delta_\lambda < 1 \Leftrightarrow (\lambda - 1)\exp\left(\frac{-d^2_{\min}}{2\sigma^2}\right) < 1 \Leftrightarrow \sigma < \frac{d_{\min}}{\sqrt{2\ln(\lambda - 1)}}, \lambda > 2 \tag{18}$$

□

This allows deriving conditions on σ for different convergence or approximation guarantees, such as $\delta_{2\lambda} < 1/3$ for convergence (Garg and Khandekar 2009) (Theorem 2.1). This point is discussed in Kyrillidis et al. (2013) (Sect. 2). We note that similar developments using the Gershgorin circle theorem have been used for the deterministic creation of projection matrices in the domain of compressed sensing (Bandeira et al. 2012) (Sect. 2.1).

4 Complete SAGA procedure

We give in Alg. 1 the complete SAGA matrix factorization procedure.[3] We then discuss its computational complexity.

[3] The MATLAB source code is available on http://people.irisa.fr/Nicolas.Courty/SAGA.

4.1 Algorithm

In addition to the data matrix $\mathbf{X}$, the user needs to tune the following parameters: σ the Gaussian kernel bandwidth, λ the expected sparsity level, and ℓ the number of prototypes in the CSS (with $\lambda \leq \ell$). Additional parameters for the gradient descent can be tuned to optimize the computation load, yet it is not mandatory. At first, one computes the CSS, then the matrix of projections. Regarding the CSS, the procedure is initialized (Lines 1–4) as in Thurau et al. (2012): First, one randomly selects a datum. Then, one finds the most distant datum to that first one. Finally, the most distant datum to this second datum is selected, and is considered as the first element of the CSS (see Lines 2 and 3 in Alg. 1). After initialization, the iterative increment of the simplex dimensionality is implemented in the loop from Line 5–8. Regarding the projection, each datum is processed separately thanks to the loop from Line 9–16. Within this loop, another loop deals with the gradient descent up to a stopping criterion (loop from Line 11–16).

Algorithm 1: The SAGA Matrix Factorization algorithm

input : $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]_{(\mathbf{x}_i \in \mathbb{R}^p)}$ the data matrix
σ the Gaussian kernel bandwidth
λ the expected sparsity level
ℓ the number of prototypes
$E = \{\epsilon, (\varepsilon_t)_{(t \in \mathbb{N})}\}$ additional parameters for the gradient descent
output: $\mathbf{W}$ the indicator matrix
$\mathbf{G}$ the reduced sparse convex embedding of $\mathbf{X}$

1 //* **First step**: Column Subset Selection as a manifold subsampling
2 $\mathbf{W}^{(i=1)} \leftarrow 0$ //*
3 [h]$W^{(i)} \in \{0, 1\}^{n \times i}$
4 $t \leftarrow \arg\min_q \left[k(\mathbf{x}_q, \mathbf{x}_{\text{rand}[1,n]})\right]$
5 $e \leftarrow \arg\min_q \left[k(\mathbf{x}_q, \mathbf{x}_t)\right]$ //*
6 [h]$\mathbf{x}_e$ is the first element
7 $\mathbf{W}_{et}^{(i=1)} \leftarrow 1$
8 **for** $i \leftarrow 2$ ***to*** ℓ **do**
9 $e \leftarrow \arg\min_q \left[\mathbf{k}_{\mathbf{x}_q}^\top \cdot \mathbf{K}_{i-1}^{-1} \cdot \mathbf{k}_{\mathbf{x}_q}\right]$
10 $\mathbf{W}_{\bullet i}^{(i)} \leftarrow 0$; $\mathbf{W}_{ei}^{(i)} \leftarrow 1$
11 compute $\mathbf{K}_i^{-1}$ from $\mathbf{K}_{i-1}^{-1}$ using Eq. 9
12 //* **Second step**: Sparse Projection over the defined simplex
13 **for** $\mathbf{x}_i \in \mathbf{X}$ **do**
14 $\mathbf{G}_{i\bullet}^{(k=0)} \leftarrow [1/\ell, \ldots, 1/\ell]$
15 **repeat**
16 $\mathbf{G}_{i\bullet}^{(k+1)} \leftarrow \mathbf{G}_{i\bullet}^{(k)} - \varepsilon_t ({\mathbf{G}_{i\bullet}^{(k)}}^\top \mathbf{K}_\ell - \mathbf{k}_{\mathbf{x}_i})$
17 find the indices of the λ greatest values of $\mathbf{G}_{i\bullet}^{(k+1)}$
18 project the corresponding elements of $\mathbf{G}_{i\bullet}^{(k+1)}$ onto Δ^λ (Maculan and Paula 1989)
19 set the other elements to 0
20 **until** $||\mathbf{G}_{i\bullet}^{(k+1)} - \mathbf{G}_{i\bullet}^{(k)}||^2 < \epsilon$;

4.2 Computational complexity

Property 3 (Computational complexity for **W**–noted as **First step** in Alg. 1) The selection of the CSS defining **W** based on the exact and incremental simplex volume maximization has complexity of $o(n\ell^3)$, i.e. it has a linear complexity with respect to n.

Proof The entire CSS construction is based on Eqs. 8 and 9. At each step of the selection, the procedure amounts to a linear scanning procedure where the volume increment is computed for each element in the dataset (size n). For one element $\mathbf{x}_i$, with $i \in [1, n]$, this requires to compute the associated bilinear form $\mathbf{k}_{\mathbf{x}_q}^\top \cdot \mathbf{K}_{i-1}^{-1} \cdot \mathbf{k}_{\mathbf{x}_q}$ $\mathbf{k}_{\mathbf{x}_q}$ vector. with an assymptotical computational complexity of $o(p^2)$. As p varies 1 to ℓ, one ends up with a cubical complexity term regarding ℓ. □

We note that thanks to Eq. 9, almost all the values (except the one corresponding to the kernel evaluation with the last chosen element) have already been computed in the previous iteration and do not need to be computed again. This makes the overall CSS computation that efficient. However, for exceptionally large data matrices, it is possible to improve it with the randomized approximate search of Thurau et al. (2010), that can be directly adapted. Finally, let us remark that, in spite of being linear in terms of n, the number of elements in the dataset, the procedure is not linear with respect to ℓ. However, as ℓ is classically several order of magnitude smaller than n, so that it is seldom important. This is why, in a similar way, SiVM is also not linear with respect to ℓ.

The complexity of the computation of the projection matrix **G** is by construction linear with respect to n as in the algorithm, it is decomposed into a succession of n independent projections. However, the complexity of the projection with respect to ℓ is linear:

Property 4 (Computational complexity for **G**—noted as **Second step** in Alg. 1) Each projection has a linear complexity regarding λ.

Proof Each projection is computed through a gradient descent, each iteration of which has a complexity dominated by that of the GSSP (Eq. 12). The latter requires getting the λ greatest values of $\mathbf{G}_{i\bullet}^t$. To do so, the GSSP classically relies on sorting the elements of $\mathbf{G}_{i\bullet}^t$ (a vector of size ℓ), with a $o(\ell \log(\ell))$ complexity. However, it is possible to be more efficient by achieving this task in $o(\lambda)$ thanks to the median-finding algorithm (Maculan and Paula 1989). Finally, if q iterations are needed in the descent, then the total complexity of one projection is $o(q\lambda)$. □

Let us note here that q typically depends on the choice of the magnitude of the gradient step ε_t, which can be efficiently set following the results of Garg and Khandekar (2009). In practice, only a few tens of iterations are necessary and if one has $n >> q\lambda$, the influence of the number of iterations is immaterial.

Finally, as both the definition of the CSS and the projection have a linear complexity with respect to the dataset size, the overall SAGA procedure also has. This allows factorizing very large matrices in tractable time.

5 Experiments and results

In this section, we first observe the behavior of SAGA on some toy datasets. Our goal is to verify the behavior of our algorithm with respect to the theoretical properties given in

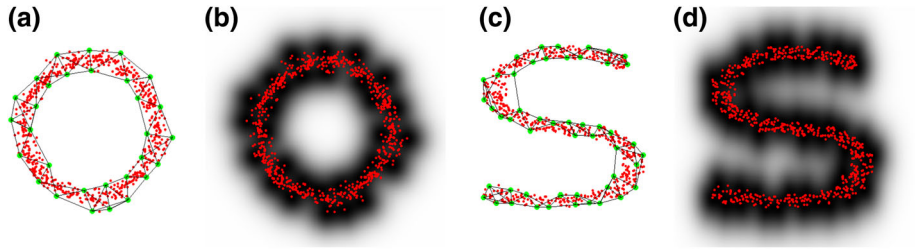

Fig. 2 A ring and an S-shape datasets. **a–c** It is possible to have $\ell > p$ elements in the CSS forming a non-simplicial polytope (here illustrated by a 4-NN graph) which approximates well the contour of the shape. **b–d** Each pixel of the image is projected in the RKHS onto the 29-simplex (**b**) or 49-simplex (**d**). Each pixel is colored according to its reconstruction error (*black* for low values, *white* for high ones)

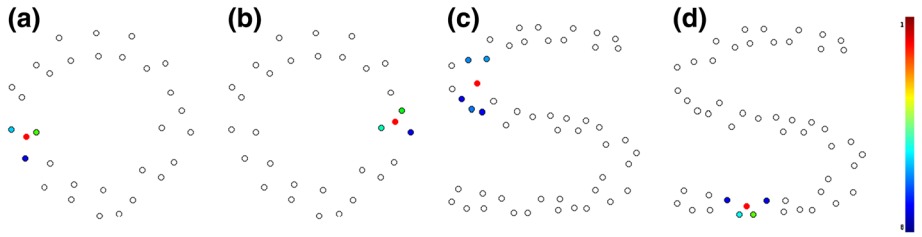

Fig. 3 **a–b** The 29-simplex on the ring shape dataset: the *red point* is projected in the RKHS as a 3-sparse vector. Each *vertex color* accounts for the corresponding convex coordinates (from *blue*= 0 to *red*= 1). **c–d** The S dataset with 5-sparse vectors (in (**d**), only 4 projections are non-nil)

the previous section, with a special focus on the nature of the subsampling occurring the RKHS. Then, the volume maximization strategy is discussed, as well as its impact on the reconstruction errors over a toy and a real dataset. We finally discuss the potential use of SAGA in a feature extraction context for classification purpose. The performances of our method are then compared to a selection of state-of-the-art methods performing NMF with characteristics shared by our method (sparsity, kernels, convexity, etc.).

5.1 Experiments on toy datasets

The SAGA paradigm is first illustrated on toy datasets (Figs. 2, 3 and 4). In the first examples we consider simulated datasets (ring and S shaped respectively) made of 600 points in $\mathbb{R}^2$: In the input space, the points of the CSS form a non-simplicial polytope which intuitively fits with the manifold hull idea (both inner and outer contours are displayed for the ring). If one projects points onto the corresponding CSS ($\ell = 30$ for the ring, and $\ell = 50$ for the S), the magnitude of the reconstruction error fits with the non-linear geometry of the simplex in the input space (Fig. 2). The sparsity and the locality of the reconstruction are displayed on Fig. 3: It appears that for each point, the number of non-null components is smaller than or equal to $\lambda = 3$ (ring) or $\lambda = 5$ (S). Moreover, these non-nil components are all located in the very close neighborhood of the projected point, as expected for a non-linear local embedding. Figure 4a shows the evolution of the reconstruction error and the locality of the samples used for the reconstruction. This last term is measured as the radius of the minimum volume enclosing ball (computed as a smallest enclosing ball problem Gärtner 1999). As expected, this mean radius is decreasing as more samples are taken from the dataset.

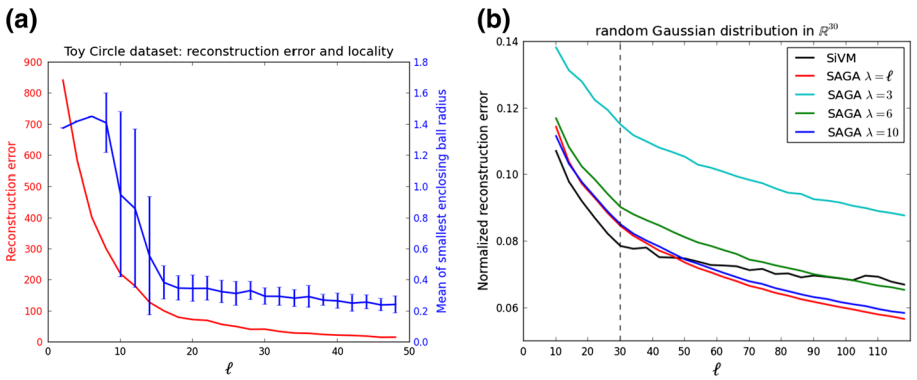

Fig. 4 **a** Reconstruction error (in *red*) and mean radius of smallest enclosing ball (*blue*) for the ring dataset. **b** Normalized reconstruction error

Next, the reconstruction error of SAGA is evaluated with respect to the chosen sparsity level and compared to the original version of SiVM (Thurau et al. 2010), yet in the Gaussian RKHS, i.e. a simplex volume maximization of Eq. 5, followed by a projection based on quadratic programming. We draw 2, 000 points in $\mathbb{R}^{30}$ according to a Gaussian distribution with covariance matrix $\sigma = 0.5\mathbf{I}$. We measure for SiVM and SAGA the normalized reconstruction error $||\Phi - \Phi\mathbf{W}\mathbf{G}^\top||^2/||\Phi||^2$. We remark here that this formula is correct for SiVM if it is implemented through a kernel form with the linear kernel. Results are displayed in Fig. 4b. When $\ell \leq p = 30$, SAGA (with or without sparsity constraint) as well as SiVM directly operates as a dimensionality reduction method: As the SiVM curve is below that of SAGA $\lambda = \ell$, SiVM appears as more reliable, which makes sense, as the Gaussian distribution provides a rather convex dataset. However, even on such a dataset, if the reconstruction error is considered along with sparsity, it appears that, whatever the value λ^* chosen for parameter λ, it is always possible to find a value ℓ^* for ℓ, such that the reconstruction error with SAGA tuned with $(\ell = \ell^*, \lambda = \lambda^*)$ is smaller than with SiVM tuned with $(\ell = \lambda^*)$. This illustrates well that it is possible to reduce the reconstruction error while constraining the solution sparsity. If one considers the case where $\ell \geq p = 30$, SiVM fails in producing reconstruction error which decreases when ℓ increases: From Eq. (1), the addition of a $p+1$ vertex leads to a simplex of null volume, which is impossible to maximize, and turning SiVM into a random projection method. Thus, the reconstruction error becomes greater than with SAGA, the latter enhancing the reconstruction quality, despite strong sparsity constraints.

5.2 Comparison on subsampling strategies

The quality of the overall matrix factorization procedure relies on two elements: the ability of the CSS to correctly define the manifold hull, and the precision of the projection. A particular focus is given here on the first one, as various sampling strategies are compared and discussed.

The goal of our first comparison is to confirm that, among the methods based on volume maximization, ours is the most accurate. To this end, we compare three volume maximization methods (operating in the RKHS, to fit the objectives of this work): The first one is very method from Thurau et al. (2012), reported through Eq. 5, including the constant distance assumption as well as the replacement of squared distances by simple distances. As this point is only supported by computational considerations, we also consider the method summarized by Eq. 4, where the constant distance assumption still holds, but where the squared distances

Table 1 Mean ratio and variance (in %, over 30 runs) of maximum simplex volumes found by both approximations from Eqs. 4 and 5, as well as SAGA over the exact CMD approach

	Uniform	Ill-conditioned	COIL-20
SiVM-approx-(4)	98.93 (0.07)	59.18 (1.6)	97.37 (0.12)
SiVM-approx-(5)	96.03 (0.21)	55.85 (1.5)	96.28 (0.13)
SAGA	**100.13** (**2** $\times$ **10**$^{-3}$)	**100.04** (**6** $\times$ **10**$^{-3}$)	**100.00** (**0.00**)

For each dataset, the most accurate result is in bold font

are kept. These two methods are referred to as SiVM-approx-(4) and SiVM-approx-(5). Naturally, the third one is that of SAGA, based on Eqs. 8 and 9. As a reference, we consider the exact volume computation based on the Cayley–Menger determinant (CMD), and we compare the ratios of the volumes derived by the three methods over the reference one. Due to the computation cost of CMD, we have restrict the size of the CSS to $\ell = 8$.

Three types of datasets are used for this comparison. First datasets are composed of 2,000 points in $\mathbb{R}^{30}$ according to a uniform distribution. The second type of datasets is used to test the robustness of the methods to ill-conditioned data; so, 2,000 elements in $\mathbb{R}^{50}$ are generated and their singular values are transformed such as described in Gillis and Vavasis (2013). Finally, our last dataset is obtained through a random selection of 160 images from the real dataset COIL-20 (Columbia University Image Library). COIL-20 contains 128 $\times$ 128 gray images of 20 objects at different view angles, for a total number of sample of 1,440 (Nene et al. 1996). We use 30 datasets of each type and we compute the mean ratio (and variance) of the volumes, as described above (see Table 1).

Some conclusions can be drawn from Table 1. First, the lower ratios of SiVM-approx-(4) and -(5) for ill-conditioned datasets confirm the conclusion of Gillis and Vavasis (2013) regarding the constant distance approximation, while SAGA is not bothered. Second, it is possible to notice the slight decrement of the performances due to the supplemental approximation in SiVM-approx-(5) where the squared distances are not considered anymore for computational reasons. Also, for all datasets, SAGA finds the most similar simplex volumes to the reference ones, as the ratios are the closest to 1. Oddly enough, the ratios are sometimes even slightly greater than 1, due to the numerical imprecisions of the determinant computations which may induce different choices for the simplex vertices. In the meantime, while volume differences between SAGA and CMD are low, their respective computational performances are quite different. As an example, with COIL dataset, each CMD computation requires $\approx$ 10 s. whereas SAGA takes $\approx$ 0.01 s in an unoptimized implementation. As a conclusion, SAGA provides a volume as large as what of CMD approach, yet in much less time. However, the computational accuracy of SAGA is fully investigated in the next subsection.

Now that it is established that SAGA provides a better hull for the manifold than classical methods based on a kernelization of the simplex volume maximization, let us compare it to other manifold sampling strategies. As it is not meaningful to use the simplex volume as a criterion, we consider the normalized reconstruction errors, such as with toy datasets. We can also consider larger CSS values, ranging from 10 to more than 100, as the computation of the CMD is not an issue anymore. In the comparison, we keep SiVM-approx-(5), as it corresponds to a kernelized version of the original paper (Thurau et al. 2012). We also consider random sampling (the NMF reducing to a random projection algorithm), kernel k-means and kernel rank-revealing Cholesky of Courty and Burger (2013), as a surrogate for the rank-revealing QR method of Shroff et al. (2011) which cannot be kernelized.

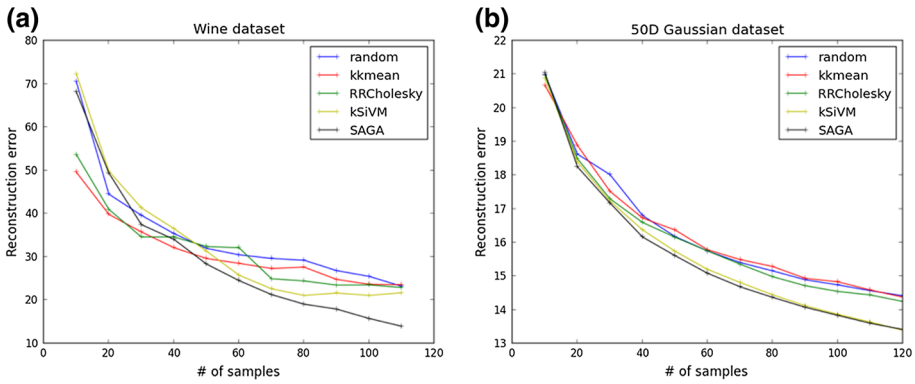

Fig. 5 Comparison of the reconstruction error according to the size of the CSS: **a** Wine dataset; **b** simulated dataset (50 dimensional Gaussian)

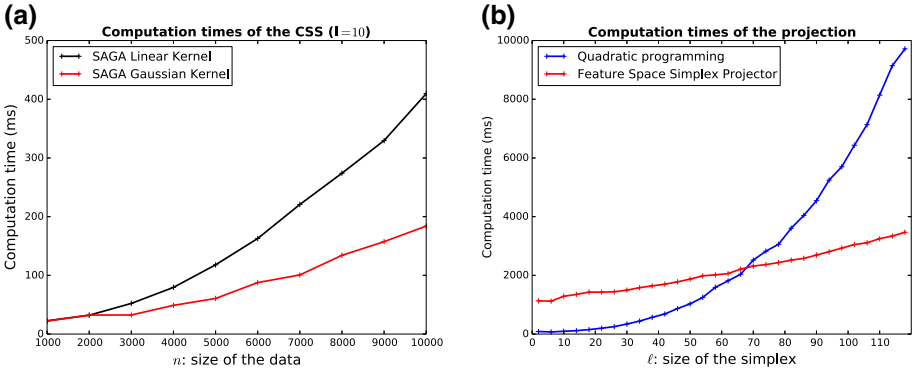

Fig. 6 Comparison of the computational times for the CSS (**a**) and the projection (**b**) for SiVM and SAGA

Two datasets are used, namely the UCI Wine dataset (Bache and Lichman 2013) composed of 178 instances with 13 attributes and a simulated dataset which consist of 2,000 points in $\mathbb{R}^{50}$ according to a Gaussian distribution. Reconstruction error curves are presented in Fig. 5 and demonstrate the best overall performances of SAGA with increasing ℓ values. However this general trend differs according to the considered dataset and the ℓ values. For example if SAGA clearly outperforms SiVM with the Wine dataset, their performances are very close with the Gaussian one. With lowest ℓ values, kernel k-means and kernel rank-revealing Cholesky get better results on the Wine dataset, however, the reconstruction error remains high whatever the strategy.

5.3 Computational complexity

First, we consider the complexity of the CSS computation. A random dataset in $\mathbb{R}^{30}$ of size n, with n ranging in [1,000–10,000] is considered, with $\ell = 10$, for both linear and Gaussian kernel. Each time the experiment is repeated 20 times to stabilize the measure, and the computation times are reported in Fig. 6a. As expected, we observe a near linear trend with both different versions of kernel. The computational differences occurring between the two kernels are mostly due to the evaluation of the kernel: the linear kernel results in a

simple dot product operation, whereas evaluating the Gaussian kernel involves computations of transcendental operators.

If we now turn to the complexity of the projection, it is useless to consider it with respect to n as the projection is dealt datum by datum. However, it is interesting to consider the complexity regarding ℓ. To do so, we use the same experimental setting, yet, n is fixed and ℓ varies from 1 to 120. Fig. 6b clearly highlights the outperformance of our projector compared to the quadratic programming approach used in SiVM.

5.4 Application to classification on real datasets

In order to propose a more application-oriented test of SAGA, we propose to use the result of various NMF methods as feature extraction processes in a classification problem. Experiments are conducted on four publicly available datasets. The previously described COIL-20 dataset, the CMU PIE dataset, the MNIST dataset and the CIFAR-10 dataset: The CMU PIE dataset contains the 32 gray scale facial images of 68 people, any of each having 42 images at different illumination and facial expression conditions (Sim et al. 2002). The MNIST dataset (LeCun et al. 1998) contains the 28×28 gray scale images of handwritten numbers (from 0 to 9). Its total sample number is 70,000. Finally, CIFAR-10 is made of 60,000 32×32 color images in 10 classes, with 6,000 images per class.

Each dataset is partitioned into two sets: one for training (one tenth of the dataset), and the other for testing. On the training dataset $\mathbf{X}_t$, a factorization is conducted and leads to derive $\mathbf{F}_t$ and $\mathbf{G}_t$. $\mathbf{F}_t$ plays the role of a visual coding dictionary, and is used to reconstruct the test set. We note here that $\mathbf{F}_t$ could have been constructed with respect to the entire dataset, in a unsupervised learning manner, but it was not the case. $\mathbf{G}_t$ is used to train a SVM classifier (with Gaussian kernel), with parameters optimized by standard cross-validation. The testing set $\mathbf{X}_s$ is then projected over $\mathbf{F}_t$ which allows deriving $\mathbf{G}_s$, used for testing with the SVM classifier. This process is repeated 10 times for each value of $\ell \in \{10, 20, 30, 40, 50\}$, and for each ℓ/λ ratio $\in \{1, 1.5, 2\}$, in order to stabilize the performances. In this setting, we have compared the result of SAGA to other state-of-the-art algorithms: Sparse NMF (Kim and Park 2007) (with $\ell/\lambda = 2$), Kernel NMF (Li and Ngom 2012), Convex NMF (Ding et al. 2010) and Kernel Convex NMF (Ding et al. 2010) (or KC NMF for short). We have used the same kernel variance for all kernel-based techniques, on the basis of a standard rule of thumb (Luxburg 2007). The mean results over the 10 repetitions are displayed in Fig. 7. It shows that SAGA produces the most accurate results on three datasets out of four. In the case of CIFAR-10 dataset, one notices first the very low performances of all the methods with respect to the state-of-the-art works focusing on classification performances. The reason is that the size of the CSS remains rather low in our experimental setting: the point of this comparison is not to exhibit the highest possible accuracies, but rather to provide a sound experimental setting across various datasets of heterogeneous difficulty. However, this does not explain why SAGA does not compete with Sparse NMF and Kernel NMF on this dataset. A possible explanation stems from the complexity of the manifold hull which cannot be described efficiently by so few elements. In this particular case, SAGA would perform slightly worse than other state-of-the-art methods. Interestingly enough, whatever the dataset, when λ stays the same, the SAGA classification accuracy improves with the increasing number ℓ of simplex vertices. However, ℓ exerts little influence when λ is already large. As discussed in Yu et al. (2009), we can relate the optimal value of ℓ to the intrinsic dimensionality of the manifold where the data live.

Table 2 summarizes the results of Fig. 7 by averaging the performances $\forall \ell \in \{10, 20, 30, 40, 50\}$, (for Sparse NMF and SAGA, with $\ell/\lambda = 2$): The variances which

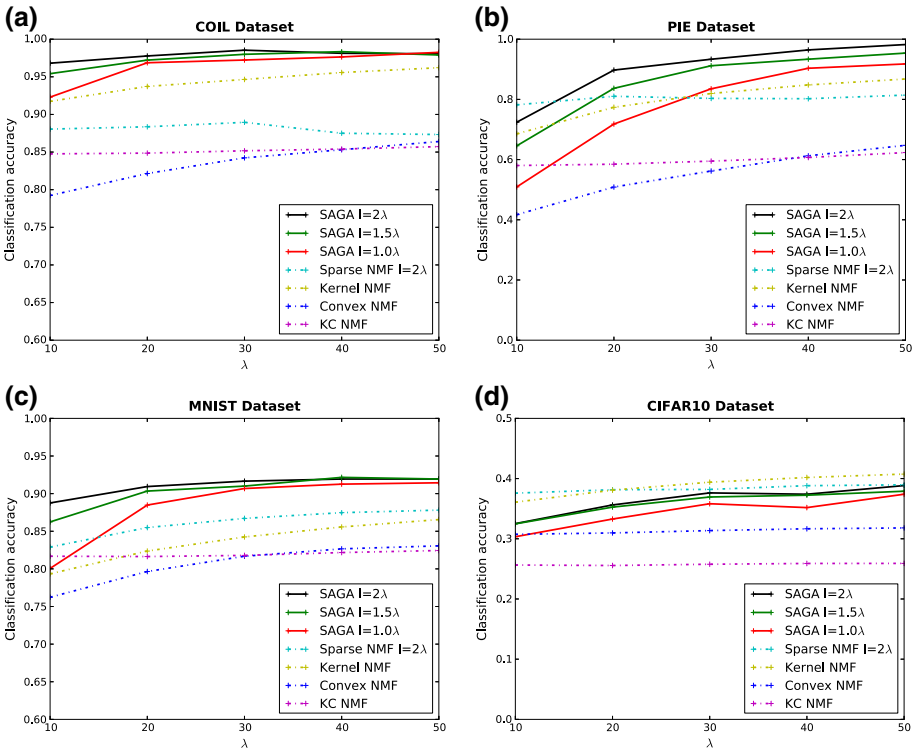

Fig. 7 Feature extraction evaluation on **a** COIL-20, **b** PIE, **c** MNIST and **d** CIFAR-10 datasets at different reduced dimension

Table 2 Mean accuracy and variance NMF-based classification

	COIL-20	PIE	MNIST	CIFAR-10
Sparse NMF	88.04 (0.66)	80.24 (1.25)	86.08 (1.99)	38.34 (0.56)
Kernel NMF	94.38 (1.75)	79.89 (7.25)	83.61 (2.86)	**38.90** (**1.87**)
Convex NMF	83.46 (2.84)	54.95 (9.07)	80.66 (2.80)	31.31 (0.45)
KC NMF	85.18 (0.39)	59.80 (1.75)	81.94 (0.35)	25.76 (0.16)
SK SiVM	96.79 (1.34)	83.73 (9.32)	89.71 (3.78)	35.50 (2.20)
SAGA	**97.23** (**1.58**)	**84.44** (**13.62**)	**89.92** (**3.18**)	35.59 (2.44)
t-statistics	3.2327	1.8872	2.3257	X
Confidence Level	$\geq$99 %	$\geq$95 %	$\geq$97.5 %	X

For each dataset, the most accurate method is in bold font. This property is assessed thanks to a paired Student's t test between the SAGA and the best other method (apart from SK SiVM which can be seen as a particular case of SAGA)

are not displayed on Fig. 7 for clarity sakes are given here. SAGA variance is sometimes important due to the strong increment of the performances when ℓ increases. To allow for more complete comparisons, we have also added a kernel version of the original SiVM with our projection method, noted Sparse Kernel SiVM (SK SiVM for short), which basically

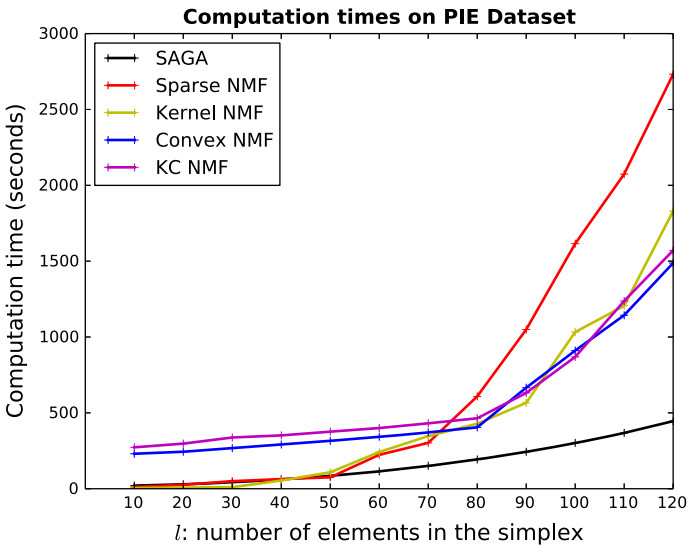

Fig. 8 Computational performances between SAGA and the other considered NMF methods on the whole PIE dataset

amounts to using SAGA, yet with Eq. 5 instead of Eqs. 8 and 9; it appears that it is always less accurate than SAGA.

Finally, on the majority of the datasets, the superiority of SAGA is established. Also, it is the fastest of all, as can be seen in Fig. 8, where the factorization performances are reported for SAGA and the considered state-of-the-art NMF methods for the whole PIE dataset when varying the size of the CSS. This illustrates the computational benefits of our approach.

6 Conclusion

SAGA (Sparse and Geometry Aware) is a new matrix factorization algorithm which has the following properties: (1) it operates in the Gaussian RKHS, which accounts for potential nonlinearity in the dataset geometry; (2) it provides sparse and convex projections onto a reduced embedding spanned by selected typical samples, which facilitates the human interpretation, and leads to a non-linear local representation of the data; (3) it has a complexity linear with the number of data entries, which allows dealing with big data. SAGA relies on both a manifold sampling strategy and a data projection. This latter has been proved to converge under some conditions regarding the Gaussian kernel variance. Finally, SAGA has been tested on toy, simulated and real datasets. The following conclusions can be drawn from the experiments: we observed in accordance with the theory that SAGA encodes the data as convex combinations of neighbor samples; the proposed volume maximization heuristic leads to better subsampling of the original data with respect to the volume of the simplex formed by the CSS; and the performances of its feature extraction have proved to outperform the selected state-of-the-art other NMF methods on classification tasks.

Acknowledgments This work was partially supported by the ANR fundings ANR-10-INBS-08 (ProFI project, “Infrastructures Nationales en Biologie et Santé”, “Investissements d’Avenir”), ANR-13-JS02-0005-01 (Asterix project). and the Prospectom project (Mastodons 2012 CNRS challenge).

References

Arora, S., Ge, R., Kannan, R., & Moitra, A. (2012) Computing a nonnegative matrix factorization-provably. In: *Proceedings of the 44th Symposium on Theory of Computing* (pp. 145–162). ACM.

Bache, K., & Lichman, M. (2013). UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 27 Dec 2013.

Bandeira, A., Fickus, M., Mixon, D., & Wong, P. (2012). The road to deterministic matrices with the restricted isometry property. arXiv preprint arXiv:1202.1234.

Bioucas-Dias, J. M., Plaza, A., Dobigeon, N., Parente, M., Du, Q., Gader, P., et al. (2012). Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *5*(2), 354–379.

Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. New York: Cambridge University Press.

Buciu, I., Nikolaidis, N., & Pitas, I. (2008). Nonnegative matrix factorization in polynomial feature space. *IEEE Transactions on Neural Networks*, *19*(6), 1090–1100.

Cai, D., He, X., Han, J., & Huang, T. (2011). Graph regularized nonnegative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(8), 1548–1560.

Candes, E. (2008). The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, *346*(9–10), 589–592.

Çivril, A., & Magdon-Ismail, M. (2009). On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science*, *410*(47–49), 4801–4811.

Courty, N., & Burger, T. (2013). A kernel view on manifold sub-sampling based on karcher variance optimization. In: *Geometric Science of Information* (pp. 751–758). Berlin: Springer.

Courty, N., Burger, T., & Johann, L. (2011). PerTurbo: A new classification algorithm based on the spectrum perturbations of the laplace-beltrami operator. In: *Proceedings of ECML/PKDD* (vol. 1, pp. 359–374). Berlin: Springer

Cutler, A., & Breiman, L. (1994). Archetypal analysis. *Technometrics*, *36*(4), 338–347.

Ding, C., Li, T., & Jordan, M. (2010). Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(1), 45–55.

Donoho, D. L., & Stodden, V. C. (2003). When does non-negative matrix factorization give a correct decomposition into parts? In: *NIPS*.

Esser, E., Moller, M., Osher, S., Sapiro, G., & Xin, J. (2012). A convex model for nonnegative matrix factorization and dimensionality reduction on physical space. *IEEE Transactions on Image Processing*, *21*(7), 3239–3252.

Garg, R., & Khandekar, R. (2009). Gradient descent with sparsification: An iterative algorithm for sparse recovery with restricted isometry property. In: *ICML* (pp. 337–344).

Gärtner, B. (1999). Fast and robust smallest enclosing balls. In: *Proceedings of the 7th Annual European Symposium on Algorithms, ESA '99* (pp. 325–338).

Gillis, N. (2012). Sparse and unique nonnegative matrix factorization through data preprocessing. *JMLR*, *13*, 3349–3386.

Gillis, N., & Vavasis, S. (2013). Fast and robust recursive algorithms for separable nonnegative matrix factorization. IEEE Transactions on Pattern Analysis and Machine Intelligence. doi:10.1109/TPAMI.2013.226.

Guillemot, C., & Turkan, M. (2012). Neighbor embedding with non-negative matrix factorization for image prediction. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012* (pp. 785–788).

Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *JMLR*, *5*, 1457–1469.

Kersting, K., Xu, Z., Wahabzada, M., Bauckhage, C., Thurau, C., Roemer, C., Ballvora, A., Rascher, U., Leon, J., & Pluemer, L. (2012). Pre-symptomatic prediction of plant drought stress using dirichlet-aggregation regression on hyperspectral images. *AAAI Computational Sustainability and AI Track*.

Kim, H., & Park, H. (2007). Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, *23*(12), 1495–1502.

Kumar, A., Sindhwani, V., & Kambadur, P. (2013). Fast conical hull algorithms for near-separable non-negative matrix factorization. In: *ICML* (pp. 231–239).

Kyrillidis, A., Becker, S., & Cevher, V. (2013). Sparse projections onto the simplex. *JMLR W&CP: Proceedings of The 30th International Conference on Machine Learning (ICML 2013)* (vol.28, 235–243).

Lafferty, J. D., & Lebanon, G. (2005). Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, *6*, 129–163.

Lafon, S., & Lee, A. (2006). Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *PAMI*, *28*(9), 1393–1403.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, Y., & Ngom, A. (2012). A new kernel non-negative matrix factorization and its application in microarray data analysis. In: *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology 2012 (CIBCB)* (pp. 371–378). IEEE.

Maculan, N., & de Paula, G. (1989). A linear-time median-finding algorithm for projecting a vector on the simplex of rn. *Operations Research Letters*, *8*(4), 219–222.

Mørup, M., & Hansen, L. (2012). Archetypal analysis for machine learning and data mining. *Neurocomputing*, *80*, 54–63.

Nene, S., Nayar, S., & Murase, H. (1996). *Columbia object image library (coil-20)*. New York: Department of Computer Science, Columbia University.

Öztireli, C., Alexa, M., & Gross, M. (2010). Spectral sampling of manifolds. In: *SIGGRAPH ASIA*.

Recht, B., Re, C., Tropp, J., & Bittorf, V. (2012). Factoring nonnegative matrices with linear programs. In: *NIPS* (pp. 1223–1231).

Schölkopf, B., & Smola, A. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. Cambridge: The MIT Press.

Shroff, N., Turaga, P., & Chellappa, R. (2011). Manifold precis: An annealing technique for diverse sampling of manifolds. In: *NIPS* (pp. 154–162).

Sim, T., Baker, S., & Bsat, M. (2002). The cmu pose, illumination, and expression (pie) database. In: *Proceedings of Fifth IEEE International Conference on Automatic Face and Gesture Recognition, 2002* (pp. 46–51). IEEE.

Thurau, C., Kersting, K., & Bauckhage, C. (2010). Yes we can: Simplex volume maximization for descriptive web-scale matrix factorization. In: *CIKM* (pp. 1785–1788).

Thurau, C., Kersting, K., Wahabzada, M., & Bauckhage, C. (2012). Descriptive matrix factorization for sustainability adopting the principle of opposites. *Data Mining and Knowledge Discovery*, *24*(2), 325–354.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, *17*(4), 395–416.

Wang, F. Y., Chi, C. Y., Chan, T. H., & Wang, Y. (2010). Nonnegative least-correlated component analysis for separation of dependent sources by volume maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(5), 875–888.

Yu, K., Zhang, T., & Gong, Y. (2009). Nonlinear learning using local coordinate coding. In: *NIPS* (pp. 2223–2231).

Zhang, D., Zhou, Z., & Chen, S. (2006). Non-negative matrix factorization on kernels. In: *PRICAI 2006: Trends in Artificial Intelligence* (pp. 404–412). New York: Springer.