Raimon Fabregat
raimon.fa@gmail.com
at ENS de Lyon

1 Non-negative matrix factorization

1.1 NMF

Non-negative matrix factorization is a problem of dimensionality reduction and source separation of data that consists on approximating a non-negative matrix $\mathbf{V} \in \mathbb{R}^{D\times N}$ as the product of two other non-negative matrices, which in the literature are usually called $\mathbf{W} \in \mathbb{R}^{D\times K}$ and $\mathbf{H} \in \mathbb{R}^{K\times N}$. The parameter K needs to be defined a priori, and is usually set such that $DK + DN << FN$. Intuitively, the problem consists on trying to express a set of N samples $\mathbf{v}_i \in \mathbb{R}^D$, the columns of $\mathbf{V}$, as a linear combinations of K non-negative "meta" vectors $\mathbf{w}_i \in \mathbb{R}^D$, the columns of $\mathbf{W}$, where the coefficients of the linear combination are in the columns of $\mathbf{H}$. That is

$$\mathbf{V} \approx \mathbf{WH} \quad \text{and} \quad \mathbf{v}_i \approx \sum_{j=1}^{K} \mathbf{w}_j \mathbf{H}_{ji} \tag{1}$$

The non-negativity of the decomposition is useful in the cases where non-negativity is inherent in the original data, such as in image processing and document classification, where negative values are not meaningful. In this cases the factorization decomposes the elements into arch-types that are directly interpretable: a usual application in image processing is to use the algorithm to find K sub-images or patterns (the columns of $\mathbf{W}$) a linear combination of which can approximately reconstruct a collection of N images (the columns of $\mathbf{V}$).

Finding this decomposition can be done by solving the following optimization problem

$$\begin{aligned} &\underset{\mathbf{W}\in\mathbb{R}^{D\times K},\mathbf{H}\in\mathbb{R}^{K\times N}}{\text{argmin}} \quad d(\mathbf{V}|\mathbf{WH}) \\ &\text{subject to } 0 \preceq \mathbf{W}, 0 \preceq \mathbf{H} \end{aligned} \tag{2}$$

where $d(\mathbf{V}|\mathbf{WH})$ is an objective function that is minimum and 0 when $\mathbf{V} = \mathbf{WH}$.

In applications where the objective is to recover the matrices $\mathbf{W}$ and $\mathbf{H}$ that a matrix $\mathbf{V}$ is supposed to be made of, different objective functions have been proposed to asses the factorization depending on the type of noise that $\mathbf{V}$ is supposed to be contaminated with. The most popular cost functions used for the NMF are the ones belonging to the β-divergence family, which include the Euclidean distance ($\beta = 2$), the Kullback-Leibler divergence ($\beta = 1$), and the Itakura-Saito divergence ($\beta = 0$). As noted by Févotte and Cemgil (2009) [1], the values $\beta = 2, 1, 0$, are suited for the cases of Gaussian additive, Poisson, and multiplicative gamma noise. In our case, in the absence of knowledge about the nature of the noise, we have used the case $\beta = 2$, which is abusively referred as eucliden distance, and implies to use $\|\mathbf{V} - \mathbf{WH}\|_F^2$ as objective function, where $\|\mathbf{X}\|_F^2 := Tr(\mathbf{XX}^*)$ (it is nothing more than the sum of all the squared elements of $\mathbf{X}$).
The first issue to notice about this problem is that NMF algorithms suffer from rotational indeterminacy, as pointed out in [1], which means that the best solution may not be unique, as one can see:

$$\mathbf{WH} = (\mathbf{WR})(\mathbf{R}^{-1}\mathbf{H}) = \mathbf{W}'\mathbf{H}'$$

The matrices $\mathbf{W}'$ and $\mathbf{H}'$ are restricted to be non-negative though, which significantly reduce the number of possible solutions. The case where $\mathbf{R}$ and $\mathbf{R}^{-1}$ are non-negative is not a concern, as it is easy to prove that in that case $\mathbf{R}$ is restricted to be a generalized permutation matrix, so that the only thing that $\mathbf{R}$ would change would be the order and scale of the columns of $\mathbf{W}$ and the rows of $\mathbf{H}$ (which is basically the same result). The only problem appears if there exist a combination $\mathbf{R}$ and $\mathbf{R}^{-1}$ without sign restrictions such that $\mathbf{W}'$ and $\mathbf{H}'$ are still non-negative. This possibility is usually ignored as in general the chances of such a matrix to exist are small, and not a concern in face of the bigger issues that the problem has for not being convex, issues that we discuss in the following paragraph, but can be addressed by adding extra constraints to the objective function as referred in [2] that would make the degeneration of minimizers disappear.

A bigger drawback, as we just mentioned, is the fact that this problem is not convex. This means that even if a unique minimizer exists, algorithms can only be expected to find a local minimum, which can be different depending on the values that the algorithm is initialized with. This problem is common among all non-convex optimization problems, and may or may not be relevant depending on the nature of the data and the practical application for which the algorithm is used. In some cases, as in compression of data, finding the optimal solution may not be crucial as long as the minimizer found is good enough. There are several ways to address this problem: the algorithm can be initialized with a “smart” setting, if you have a priori ideas of the form of the solutions, or using the result of another approximation such as the Singular Value Decomposition, as done in [3] and [4]. Extra constraints can also be added hoping that they will lead the algorithm towards the desired result, but in the absence of a better solution, and if the execution time of the algorithm is not too long, it is usually ran several times with random initializations and the solution with the lowest objective function is taken.

The NMF problem has had a lot of attention in the past decades, since a simple and fast algorithm was proposed by Lee and Seung [5] in 1999 to solve it. Many other algorithms to solve the NMF have been developed since then, aiming to get even faster and more accurate solutions. Also many variations of the original problem have been introduced in order to add different characteristics to the obtained solutions. This is done by adding extra constraints or extra cost functions to the formulation of the optimization problem. The problem with the NMF literature is that many of the algorithms presented are not build rigorously and don’t follow a solid theoretical background, usually lacking the properties that algorithms for non-convex optimization problems are supposed to have, for example not proving that they converge. For this reason, we have followed the work by Bolte *et al.* in [6], which introduces a solid framework for non-convex and non-smooth multivariate optimization problems called Proximal Alternating Linearized Minimization (PALM).

1.2 PALM

The following is a short description of the framework and its benefits. This method is aimed to solve problems of the following form:

$$\underset{x,y}{\text{minimize}}\ \Psi(x,y) := f(x) + g(y) + H(x,y) \tag{3}$$

where the functions f and g are extended valued (i.e., allowing the inclusion of constraints) and H is a smooth function. The method is not restricted to two variables though, and the results and the properties of the method hold true for any finite number of them.

The standard approach to solve the problem is via the alternating minimization scheme, which consists on generating a sequence $\left\{(x^k, y^k)\right\}_{k\in\mathbb{N}}$ from a given initial point (x^0, y^0) via the steps

$$\begin{aligned} x^{k+1} &\in \operatorname{argmin}_x \Psi(x, y^k) \\ y^{k+1} &\in \operatorname{argmin}_y \Psi(x^k, y) \end{aligned} \tag{4}$$

which can be proven to converge to a critical point of Ψ, but it is only suited for the case where all the functions in the problem 3 are convex and smooth. If that is not the case, each step requires the exact solution of a not trivial non-convex and non-smooth problem. It has in addition two other practical problems: it accumulates computational errors at each step, and it needs to be stopped at each iteration arbitrarily before passing to the next (except in the case where each step has a closed form solution). PALM was developed as a framework to include non-convex and non-smooth settings. It is based on the Proximal Forward-Backward (PFB) algorithm for minimization problems of one variable with non-convex and non-smooth objective functions and constraints. PFB relies on the assumption that the function to be minimized satisfies the Kurdyka-Łojasiewicz property, which allows to derive the convergence of the bounded trajectories of the steepest descent equation to critical points [6]. Basically, when a function f satisfies the KL property in a point a it means that there exists some $\theta \in [\frac{1}{2}, 1)$ such that the function $|f - f(a)|^\theta \|\nabla f\|^{-1}$ remains bounded around a ([7]), and a *KL function* is a function that satisfies the KL property at every point. The class of functions satisfying such property is very large, and covers a considerable amount of non-convex/non-smooth functions arising in many fundamental applications, including of course all the proper and smooth functions. See [7] for a more detailed description of the property and the functions satisfying them.

The PFB minimizes the sum of a smooth function h with a non-smooth one σ of one variable, and consists of a gradient step on the smooth part followed by by a proximal step on the non-smooth part:

$$x^{k+1} \in \operatorname{argmin}_{x\in\mathbb{R}^d} \left\{ < x - x^k, \nabla h(x^k) > + \frac{t}{2}\|x - x^k\|_2^2 + \sigma(x) \right\}, \ (t > 0),$$

or using the proximal map notation,

$$x^{k+1} \in \operatorname{prox}_t^\sigma (x^k - \frac{1}{t}\nabla h(x^k)) \tag{5}$$

PALM combines the alternating minimization scheme with the PFB, applying PFB steps alternating between variables. What follows is the pseudo-code of the algorithm as presented in [6]:

```
PALM: Proximal Alternating Linearized Minimization

1. Initialization: start with any $(x^0, y^0)$
2. For each $k = 0, 1, ...$ generate a sequence $\{(x^k, y^k)\}_k$ as follows:
    2.1. Take $\gamma_1 > 1$, set $c_k = \gamma_1 L(y^k)$ and compute
            $x^{k+1} \in \operatorname{prox}_{c_k}^f (x^k - \frac{1}{c_k}\nabla_x H(x^k, y^k))$
    2.2. Take $\gamma_2 > 1$, set $d_k = \gamma_2 L(x^{k+1})$ and compute
            $y^{k+1} \in \operatorname{prox}_{d_k}^g (y^k - \frac{1}{d_k}\nabla_y H(x^{k+1}, y^k))$
```

where $L(x^k)$ is the Lipschitz constant of $\nabla_x H(x^k, y^k)$, and ensures convergence. The values γ_1 and γ_2 are only restricted to be bigger than one. We have used $\gamma_1 = \gamma_2 = 1.1$, but other values can be assigned to them, affecting only the speed of convergence.

This algorithm is proven to converge by Bolte in [6], and its benefits are that it gives a solid model that can be applied to a wide variety of problems and its variants, for example to the NMF problem that concerns us.

1.3 PALM-NMF

As Bolte already point out in [6], the original version of the NMF can be easily solved using PALM, for instance taking

$$\begin{aligned}\Psi(\mathbf{W},\mathbf{H}) =& \|\mathbf{V}-\mathbf{WH}\|_F^2 + \delta_{\mathbf{W}\succeq 0} + \delta_{\mathbf{H}\succeq 0}\\ =& H(\mathbf{W},\mathbf{H}) + f(\mathbf{W}) + g(\mathbf{H})\end{aligned}$$

where $\delta_{\mathbf{X}\succeq 0}$ is the indicator function of the set $\{\mathbf{X}$, such that $\mathbf{X}\succeq 0\}$, defined as

$$\delta_{\mathbf{X}\succeq 0} = \begin{cases} 0 \text{ if } \mathbf{X}\succeq 0 \\ \infty \text{ else} \end{cases}$$

The variables x and y have been substituted by $\mathbf{W}$ and $\mathbf{H}$ (not to be confused with the function H). We have used this notation trying to follow at the same time the one usually used in the NMF literature and the one used in the article by Bolte [6] about PALM.

The PALM ingredients are easily derived:

- $H(\mathbf{W},\mathbf{H}) = \|\mathbf{V}-\mathbf{WH}\|_F^2 = Tr((\mathbf{V}-\mathbf{WH})(\mathbf{V}-\mathbf{WH})^T)$
  - $\nabla_{\mathbf{W}} H(\mathbf{W},\mathbf{H}) = 2\mathbf{WHH}^T - 2\mathbf{VH}^T$
  - $\nabla_{\mathbf{H}} H(\mathbf{W},\mathbf{H}) = 2\mathbf{W}^T\mathbf{WH} - 2\mathbf{W}^T\mathbf{V}$
  - $L(\mathbf{W}) = 2\|\mathbf{HH}^T\|_F$ and $L(\mathbf{H}) = 2\|\mathbf{W}^T\mathbf{W}\|_F$
- $\text{prox}_t^{\delta_{\mathbf{H}\succeq 0}}(\mathbf{H}) = \text{argmin}_{\mathbf{X}}\ \delta_{\mathbf{H}\succeq 0} + \frac{t}{2}\|\mathbf{X}-\mathbf{H}\|_F^2 = P_+(\mathbf{H}) = max\,\{0,\mathbf{H}\}$ which is nothing more than the projection of H to the $\mathbb{R}_+^{K\times N}$ space, or in even simpler words, setting all the negative values of $\mathbf{H}$ to zero. The same applies for $\mathbf{W}$.

Introducing the previous results to the PALM scheme directly gives the full minimization algorithm. The stopping criteria can either be the relative size of the step being small enough or a fixed number of iterations.

1.4 Modified PALM-NMF

This framework allows also to easily add extra constraints to the objective function to get solutions with different characteristics. For our purposes, we developed a modified version of the algorithm with two extra constraints that we thought that could be useful in the analysis of our data, which are:

1. **Sparse patterns** A usual desired attribute in decompositions like NMF is that the patterns in which the signals are decomposed are as sparse as possible. In our case, it means promoting sparsity in the subgraphs in which the networks are decomposed. This can be done adding a term proportional to the L_1 norm of $\mathbf{W}$ to the function to minimize. In [8] it is shown how adding the L_1 in this way mimics the L_0 norm (i.e. the number of elements in the variable that are non-zero) and promotes sparsity.

2. **Smooth activation coefficients** In cases similar as ours, where two adjacent columns of $\mathbf{V}$ are adjacent in time, it may be desirable to require that their representation doesn't change too much from one to the other, in case it is known that the evolution of the signals cannot be too fast. This can be done using the Tikhonov regularization, which means adding an extra term proportional to $\|\mathbf{H\Gamma}\|_F^2$ to the objective function, where Γ is the following matrix:

$$\Gamma = \begin{bmatrix} 1 & 0 & \dots & 0 \\ -1 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 1 \\ 0 & \dots & 0 & -1 \end{bmatrix}$$

Basically this extra term "encourage" the algorithm to minimize the difference between the coefficients of two adjacent elements.

With these constraints we need to add also extra terms to control that the values inside the matrices $\mathbf{W}$ and $\mathbf{H}$ don't grow to big. This is necessary as, for example when using the sparsity variant, the algorithm could make the L_1 norm of $\mathbf{W}$ arbitrarily small just scaling down all the values in the matrix $\mathbf{W}$ and scaling up $\mathbf{H}$ by the same factor; on the other side, with the smoothness constraint, the algorithm could arbitrarily reduce $\|\mathbf{W\Gamma}\|_F^2$ scaling down $\mathbf{H}$ and scaling up $\mathbf{W}$. Controlling that the matrices don't grow too large can easily be done just adding terms proportional to $\|\mathbf{W}\|_F^2$ and $\|\mathbf{H}\|_F^2$.

So the function to optimize becomes

$$\Psi(\mathbf{W},\mathbf{H}) = \|\mathbf{V}-\mathbf{WH}\|_F^2 + \eta\|\mathbf{H\Gamma}\|_F^2 + \delta_{\mathbf{W}\succeq 0} + \lambda\|\mathbf{W}\|_1 + \delta_{\mathbf{H}\succeq 0} + \beta_{\mathbf{W}}\|\mathbf{W}\|_F^2 + \beta_{\mathbf{H}}\|\mathbf{H}\|_F^2 \tag{6}$$

The Tikhonov regularization is a smooth function and can be included in the gradient step. On the other side, the L_1 norm is not smooth, but can be included in the function f and in the proximal operator. We have used $\beta_{\mathbf{H}} = \beta_{\mathbf{W}} = 0.1$, but any positive value can be assigned to them. The changes for the PALM steps are

- $H(\mathbf{W},\mathbf{H}) = \|\mathbf{V}-\mathbf{WH}\|_F^2 + \eta\|\mathbf{H\Gamma}\|_F^2 + \beta_{\mathbf{W}}\|\mathbf{W}\|_F^2 + \beta_{\mathbf{H}}\|\mathbf{H}\|_F^2$
  - $\nabla_{\mathbf{H}} H(\mathbf{W},\mathbf{H}) = 2\mathbf{W}^T\mathbf{WH} - 2\mathbf{W}^T\mathbf{V} + 2\eta\mathbf{H\Gamma\Gamma}^T + 2\beta_{\mathbf{H}}\mathbf{H}$
  - $\nabla_{\mathbf{W}} H(\mathbf{W},\mathbf{H}) = 2\mathbf{WHH}^T - 2\mathbf{VH}^T + 2\beta_{\mathbf{W}}\mathbf{W}$
  - $L(\mathbf{W}) = 2\|\mathbf{HH}^T\|_F + 2\beta_{\mathbf{H}}$ and $L(\mathbf{H}) = 2\|\mathbf{W}^T\mathbf{W}+\|_F + 2\beta_{\mathbf{W}} + 2\eta\|\mathbf{\Gamma\Gamma}^T\|_F$.
- Now $f(\cdot) = \delta_{\cdot\succeq 0} + \lambda\|\cdot\|_1$, and it can easily be shown that $\text{prox}_t^f(\cdot) = max\left\{0, \cdot - \frac{2\lambda}{t}\right\}$

In appendix 2 is included a comparison between different NMF methods and some toy examples to see how NMF works and how the extra constraints that we included alter the results. The code of the algorithm PALM-NMF with the constraints as we presented it can be found in the link of the footnote.[1].

2 NMF comparation and examples of applications

2.1 Comparison of PALM with other existing NMF algorithms

In this section we compare the PALM-NMF algorithm with other existing ones applied to the original NMF problem without extra constrains.

[1] https://github.com/raimon-fa/palm-nmf

The original problem is convex for any of the two variables, although not for both of them. The alternating minimization scheme 4 can be therefore applied directly, solving each sub-problem iteratively using any of the existing algorithms for convex optimization, although the practical problems that we mentioned regarding when to stop each iteration and accumulative errors are still present. This method to solve the NMF is usually referred as Alternating Non-Negative Least Squares (ANNLS) and was introduced by Paatero and Tapper [9].

One of the most popular algorithms for NMF that was introduced by D. Lee and H. Seung [5] finds a point that satisfies the Karush-Kuhn-Tucker (KKT) conditions at each iteration, which are first-order necessary conditions for a solution in non-linear programming to be optimal. This lead them to the following multiplicative updates

$$\begin{aligned} \mathbf{W} &\leftarrow \mathbf{W} \circledast [(\mathbf{V}\mathbf{H}^T) \oslash (\mathbf{W}\mathbf{H}\mathbf{H}^T + \epsilon)] \\ \mathbf{H} &\leftarrow \mathbf{H} \circledast [(\mathbf{W}^T\mathbf{V}) \oslash (\mathbf{W}^T\mathbf{W}\mathbf{H} + \epsilon)] \end{aligned} \tag{7}$$

These steps ensure that the algorithm converges to a critical point. It quickly reaches a good approximation, but it has a very slow convergence. To know more about the KKT conditions and its relevance in the field of convex optimization one can refer to the book *Convex Optimization* by Boyd and Vandenberghe [8].

To implement previously mentioned algorithms we used the matlab toolbox *The non-negative matrix factorization toolbox for biological data mining* [10], which includes an implementation of each one of them.

We also compare the PALM-NMF with a more recent algorithm by Févotte [1] which uses a family of algorithms called Majorization-Equalization (ME) which also guarantees that the solution will reach a critical point.

To compare the performance of the algorithms we create a matrix $\mathbf{V} \in \mathbb{R}^{D\times N}$, with $D = 50$ and $N = 1000$, of random numbers uniformly distributed between zero and one and we use them to decompose it into $\mathbf{W}$ and $\mathbf{H}$ with $K = 5$.

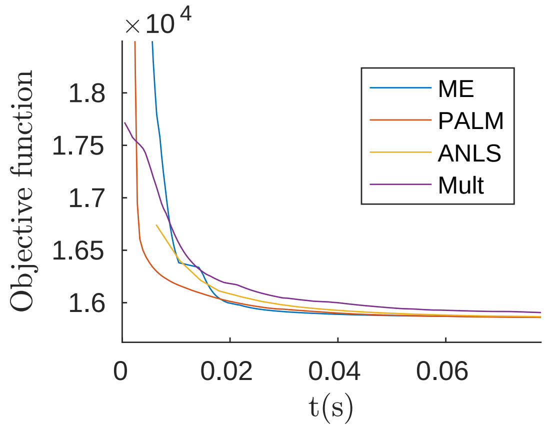
In figure 1 can be found the convergence plots of a run of each of the algorithms mentioned, i.e., the objective function as a function of time for each algorithm. We can see that while the algorithm by D. Lee and H. Seung quickly reaches a good approximation in very few steps, it is the slowest one to converge. The PALM NMF and the algorithm by Févotte ME are the fastest ones to converge, although PALM reaches a good approximation a little bit faster. The ANLS converges almost as fast as the two mentioned before, but has a very slow start. We see that the PALM-NMF algorithm is very competitive compared to other existing ones for the original NMF problem, but its most beneficial characteristic is that it can easily be tuned and adapted to modifications of the original NMF problem.

2.2 Additional constraints

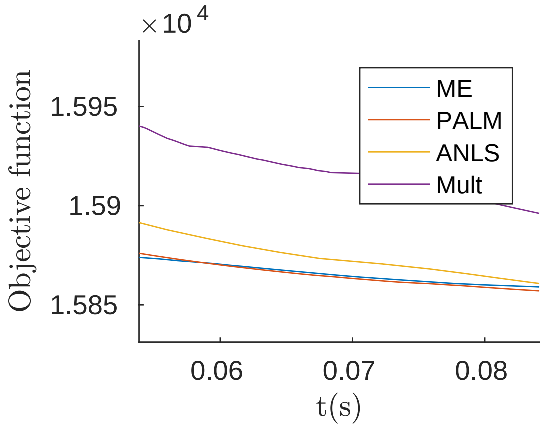
As we mentioned one of the benefits of PALM is that it easily allows to tune the objective function in order to get solutions with certain characteristics. Here we present examples of how the two extra constraints that we added, smoothness in the raws of $\mathbf{H}$ and sparsity in $\mathbf{W}$, can be used.

2.3 Sparsity

To illustrate how the sparsity constraint alters the solution of the NMF we use the algorithm for one of its popular application, the deconstruction of images of faces [11]. To do it we take a set of 2-D pictures of faces, flatten each one of them to a vector, and concatenate them together to form a matrix $\mathbf{V}$ that



(A)



(B)

FIGURE 1: Convergence plots for the different algorithms mentioned to solve the NMF problem. A) General view of the plots. B) Zoom in the convergence zone.

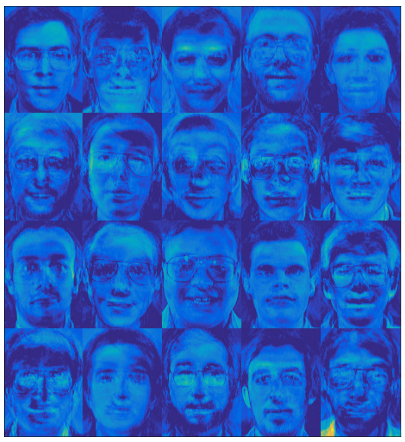
we can use as input for the NMF. In figure 2a are plotted the pictures of the images of faces that we have used for the decomposition, their approximation after applying the NMF and the patterns learned, for the cases with and without sparsity. The decomposition without sparsity in this case is not very interesting, as we have used a high value of K (so the dimensionality is not reduced so much) and the patterns learned are some kind of fusion of faces. In the case with sparsity however, the patterns learned are localized and characterize different parts of the faces, highlighting structures such as the forehead, the hair or the cheekbones, giving an approximation which is almost as good than the one without sparsity, but with patterns that much more "empty".
A problem with this approach, however, is that the sparsity is constraining the whole matrix $\mathbf{W}$, but it does not care where the non-zero values are allocated in the matrix. So if the parameter controlling the sparsity, λ, is set too high, the algorithm may set a whole pattern to zero and give a solution with $K-1$ non sparse patterns.

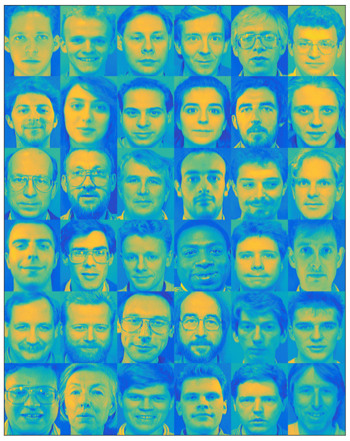
2.4 Smoothness

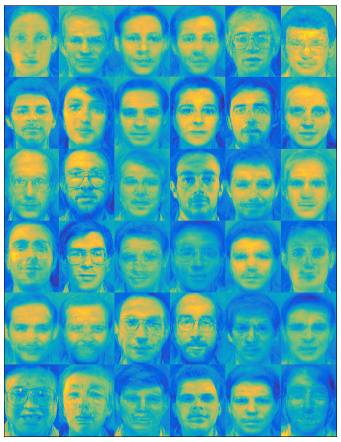
Asking for a smooth evolution of the representation of the signals can be beneficial in many different frameworks. As an example, we consider the case when a priory one knows that the coefficients of the linear combination, the rows of $\mathbf{H}$, are smooth functions. We create a non-negative matrix $\mathbf{V}$ using a random matrix $\mathbf{W}_r \in \mathbb{R}^{100\times 5}$ and a matrix $\mathbf{H}_r \in \mathbb{R}^{5\times 200}$ whose rows ,plotted in figure 3a, are smooth functions. We add also Gaussian noise: $\mathbf{V} = max(\mathbf{W}_r \cdot \mathbf{H}_r + \mathcal{N}(0,\sigma^2), 0)$, setting the negative values that the noise may create to zero to ensure that there are no negative values in $\mathbf{V}$.
Then we use the PALM-NMF with the objective function 6 with a positive η and no sparsity constraints ($\lambda = 0$). The value of the parameter η is arbitrary, the higher the smoother the solutions will be, but there is no way to control "degree" of smoothness of the solution.
In the figure 3 we can see the benefits of our approach. The kernels learned with the smooth NMF resemble much more the original ones, up to some scaling, even though the approximated $\mathbf{V}$ from the PALM-NMF without smoothness constraints was closer to the real $\mathbf{V}$ than the one obtained with smoothness constraints (the Frobenius norm of the difference between the real matrix $\mathbf{V}$ and the approximation $\mathbf{WH}$, $\|\mathbf{V} - \mathbf{WH}\|_F$ was 53.85 for the NMF with smoothness, and 55.38 without smoothness, so the approximation of the original NMF was better).

(A)

(B)


(C)


(D)


(E)

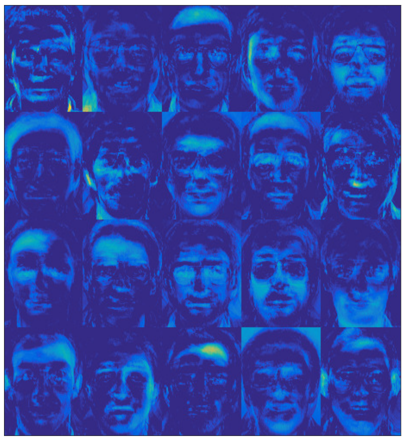
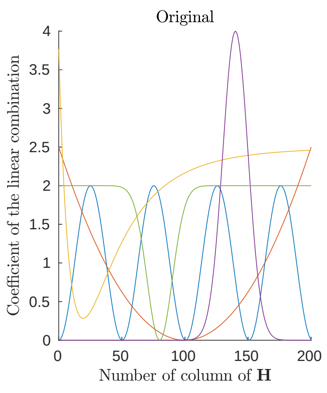
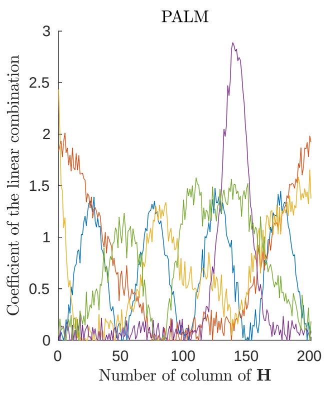

(F)

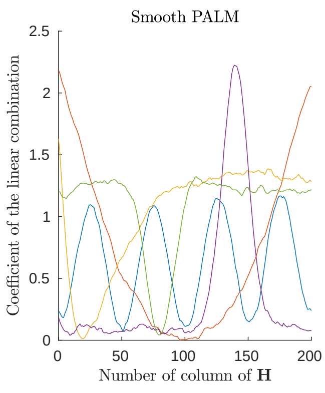
Figure 2: In the figures A and D are plotted the original faces. B and E are the faces that the algorithm approximates as a combination of 20 sub-images. C and F are the sub-images or patterns that the algorithm learned for the cases without sparsity (C) and with sparsity constraints (F). We see that the sparse patterns characterize different parts of a face, and are able to reconstruct the original faces equally well, to the point that it is hard to notice the differences between B and E.



(A)

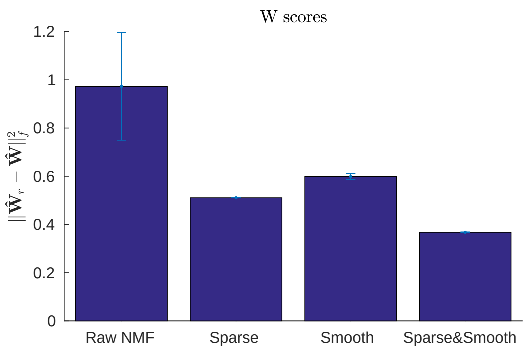


(B)

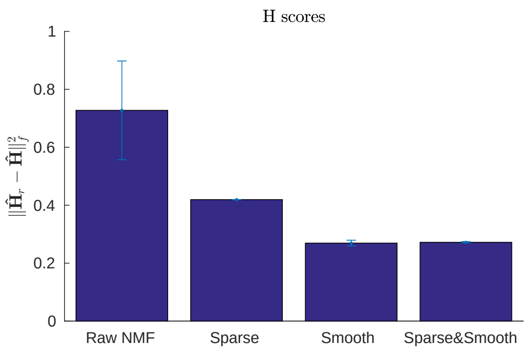


(C)

Figure 3: A) original smooth functions stored in the rows of $\mathbf{H}_r$. B) the learned rows of $\mathbf{H}$ with the original PALM-NMF without extra constraints. C) result using the PALM-NMF with the smoothness constraint.



(A)



(B)

FIGURE 4: Bar plots with the scores of the different variations of the algorithm on the data generated artificially

2.5 Simultaneous smoothness and sparsity

To see that both constraints can be combined, we take again the same toy example as in the smoothness test, building a matrix $\mathbf{V} = |\mathbf{W}_r \cdot \mathbf{H}_r + \mathcal{N}(0, \sigma^2)$ with the rows of $\mathbf{H}_r$ being the smooth functions plotted in figure 3a, but now with $\mathbf{W}_r$ being a sparse matrix, initialized with random values and with 80% of its entries set to zero. Therefore, we want to recover the matrices $\mathbf{W}_r$ and $\mathbf{H}_r$ knowing that the rows of $\mathbf{H}_r$ are smooth and that the matrix $\mathbf{W}_r$ is sparse. To measure the quality of the recovery, we can take the learned matrices $\mathbf{W}$ and $\mathbf{H}$ and compare them to the original ones $\mathbf{W}_r$ and $\mathbf{H}_r$. To do so, we normalize the columns of $\mathbf{W}$ and $\mathbf{W}_r$ and the rows of $\mathbf{H}$ and $\mathbf{H}_r$ and order the columns and rows of $\mathbf{W}$ and $\mathbf{H}$ such that the learned patterns that match the original ones are in the same order in the matrices. In figure 4 is shown the "quality" of the learned normalized and ordered $\hat{\mathbf{W}}$s and $\hat{\mathbf{H}}$s in terms of their distance to the normalized original ones, $\hat{\mathbf{W}}_r$ and $\hat{\mathbf{H}}_r$, using the original NMF without constraints, with the sparsity constraint, with the smoothness constraint and with both of them at the same time. Each variation of the algorithm was ran 15 times with different random initializations. We can see that using both sparsity and smoothness gives the best performance. The constraints not only improve the quality of the recovery but also greatly reduce the variability of the solutions due to the random initializations.

References

[1] Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural computation*, 23(9):2421–2456, 2011.

[2] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation.* John Wiley & Sons, 2009.

[3] Christos Boutsidis and Efstratios Gallopoulos. Svd based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.

[4] M Rezaei, R Boostani, and M Rezaei. An efficient initialization method for nonnegative matrix factorization. *Journal of Applied Sciences*, 11(2):354–359, 2011.

[5] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[6] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.

[7] Jérôme Bolte, Aris Daniilidis, and Adrian Lewis. The łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM Journal on Optimization*, 17(4):1205–1223, 2007.

[8] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[9] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[10] Yifeng Li and Alioune Ngom. The non-negative matrix factorization toolbox for biological data mining. *Source code for biology and medicine*, 8(1):10, 2013.

[11] Stefanos Zafeiriou, Anastasios Tefas, Ioan Buciu, and Ioannis Pitas. Exploiting discriminant information in nonnegative matrix factorization with application to frontal face verification. *IEEE Transactions on Neural Networks*, 17(3):683–695, 2006.