Proceedings of the 15th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2015

Experiments with the NNMFPACK library: influence of β parameter in the NNMF approximation error

P. San Juan Sebastián[1], A.M. Vidal[1], V.M. García-Mollá[1], F.J. Martínez-Zaldívar[2], J. Ranilla[3], P. Alonso[4], M. Alonso-González[3] and R. Cortina[3]

[1] *Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Spain*

[2] *Departamento de Comunicaciones, Universitat Politècnica de València, Spain*

[3] *Departamento de Informática, Universidad de Oviedo, Gijón, Spain*

[4] *Departamento de Matemáticas, Universidad de Oviedo, Spain*

emails: p.sanjuan@upv.es, avidal@dsic.upv.es, vmgarcia@dsic.upv.es, fjmartin@dcom.upv.es, ranilla@uniovi.es, palonso@uniovi.es, monica300876@gmail.com, raquel@uniovi.es

Abstract

NnmfPack is a numerical library designed to compute efficiently the Non-Negative Matrix Factorization (NNMF). It has been conceived for shared memory heterogeneous parallel systems, and it supports, from its conception, both conventional multi-core processors and many-core coprocessors. NnmfPack offers different algorithms which allow to handle different metrics options such as β-divergence or Frobenius norm. In real applications the choice of the β-parameter presents a problem for the users of the NNMF that must decide which value to use to obtain the best approximation. In this paper, the influence of the parameter β in the NNMF approximation error for different problems is empirically evaluated. Different datasets have been used in order to analyze and evaluate the dependency between the quality of the approximation provided by NnmfPack and the value of the β-parameter used.

Key words: NNMF, NnmfPack*, β-divergence, experimental results*

  ISBN: 978-84-617-2230-3

1 Introduction

The Non-Negative Matrix Factorization (NNMF) has become a very important tool in fields such as document clustering, data mining, machine learning, data analysis, image analysis, audio source separation or bioinformatics [1, 2, 3, 4, 5]. NNMF consists on approximating a matrix $A \in \mathbb{R}^{m\times n}$ by the product of two matrices W and H, with some conditions: all elements of the matrix A are non-negative, and $W \in \mathbb{R}^{m\times k}$ and $H \in \mathbb{R}^{k\times n}$ with $k \leq \min(m, n)$ are two lower rank matrices with non-negative elements too, such that $A \approx WH$. The problem can be addressed as the computation of two matrices W_0, H_0 such that

$$\|W_0H_0 - A\|_F = \min_{W,H\geq 0} \|WH - A\|_F. \tag{1}$$

Other norms can be used instead of the Frobenius norm (see, for instance, [6], where the NNMF is also defined in terms of the Kullback-Leibler divergence). Many algorithms have been proposed for NNMF calculation (see [4] and [6, 7, 8, 9]).

The β-divergence was introduced by Eguchi and Minami, see [13], as an error measure. It can be defined as (see, *e.g.* [14])

$$d_\beta(x|y) := \begin{cases} \frac{1}{\beta(\beta-1)}(x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}), & \text{if } \beta \in \mathbb{R}\setminus\{0,1\}, \\ x(\log x - \log y) + (y - x), & \text{if } \beta = 1 \\ \frac{x}{y} - \log\frac{x}{y} - 1, & \text{if } \beta = 0. \end{cases} \tag{2}$$

The previous cost function is defined for all real numbers, but values of β between 0 and 2 are usually considered on practical applications. Mathematically, this divergence is equal to other distances and divergences for some particular values of β: the Frobenius norm ($\beta = 2$), the Kullback-Leibler divergence ($\beta = 1$) and the Itakura-Saito divergence ($\beta = 0$).

Taking into account [14] and using the gradient criterion, it is possible to obtain the following rules to update the matrices H and W:

$$H \leftarrow H \cdot \frac{W^T((WH)^{\cdot\beta-2}\cdot A)}{W^T(WH)^{\cdot\beta-1}}, \quad W \leftarrow W \cdot \frac{((WH)^{\cdot\beta-2}\cdot A)H^T}{(WH)^{\cdot\beta-1}H^T}, \tag{3}$$

where $X^{\cdot n}$ denotes the matrix with entries $([X]_{ij})^n$ and the division is taken entrywise.

These metrics and algorithms were already presented in previous papers [10] and [11], where some results and practical applications can be found. In these works, as well as in other examples present in the NNMF literature, it is noted that there is a relation among data, the value of β parameter and the error obtained when matrix A is approximated by WH.

In real applications the choice of the β parameter presents a problem for the users of the NNMF. They must decide what value to use to obtain the best approximation. Initially,

 ISBN: 978-84-617-2230-3

there is no best β parameter value for the NNMF that ensures the minimal error among different values of β. In practice, the accuracy of the solutions given by the β-divergence algorithms seem to be related to the target problem. Due to this ambiguity we decided to evaluate the influence of the parameter β in the approximation error for different problems.

The easiest way to evaluate this relation may be to use artificial datasets with synthetic matrices, but these experiments may not show the relations among data that probably can influence the behaviour of the errors in the approximation of A by WH. To correctly prove that, we must test with real datasets from practical problems because these data may contain latent variables that affect to the final result of the factorization.

Exploiting that feature, we will try to compute which value of the β parameter is the best to obtain the best approximation for each type of problem. Obviously, different matrices of each type will be used to generalize the analysis for a given subset of problems.

NnmfPack is a parallel library introduced in [11], which allows to compute the NNMF by means of different algorithms, and specifically by means of β-divergence based algorithms which implement the updating rules (3) as introduced in [12]. NnmfPack has parallel implementations of the NNMF, supporting different architectures: multi-core processors, many-core coprocessors and Graphics Processing Units. *dbdiv_cpu* is the NnmfPack routine that concretely implements the expression (3). Throughout this paper we will use this routine as a basic tool to evaluate the relationship between the error in the approximation of A by WH, the value of β and the data types used.

The remainder of the paper is as follows. In Section 2 we explain the experimentation method used in the performed tests, the different error measures used to check the quality of the approximation and the different data matrices used in the tests. In Section 3 we describe the experimental results obtained and expose our thoughts on each dataset. In Section 4 we summarize the results obtained in the experiments. Finally, in Section 5 we expose some conclusions of the paper and we discuss about the future work that could be developed with the observations made during this experiments.

2 Algorithms and data

As said in Section 1, the computational NnmfPack library will be used in the experiments. NnmfPack has an implementation of the multiplicative algorithm (3) called *dbdiv_cpu*. This routine has eight parameters: *dbdiv_cpu(m, n, k, A, W, H, beta, iters)*. The first three are the dimensions of the problem (m, n, k). Then, matrix A is the problem matrix. Matrices W and H, on input, are the initialization matrices W_0 and H_0; on output they are the solution matrices W and H. Last, *beta* is the β parameter and *iters* is the number of iterations.

For the evaluation process, we executed the algorithm several times varying some of the parameters as we describe in the following:

 ISBN: 978-84-617-2230-3

1. β **:** We selected values between 0 and 2 because these are the values usually used in practical applications as said in Section 1. Derived from observations during the tests, we used different β values in some experiments as it will be explained in the proper section.

2. *iters :* We used 100, 200 and 400 iterations. Other NNMF packages (*e.g.* MATLAB) use 100 as their default number of iterations that the algorithm must perform. We decided to increment this number to see the effect on the error measures.

3. k *:* We used $\min(m, n)$ divided by 2, 4 and 8. Each k gives us a problem which will be tackled by varying the other two parameters.

In order to assess correctly the influence of the β parameter, we tested our algorithm over data from different types of problems and some generated data. So we can look deep through the relation of the problem data and the β parameter.

Despite the algorithm tries to minimize the β-divergence error we will use the Frobenius norm of $A - WH$ as another measure of quality of our solutions too. Both error measures are computed as:

$$err_F = \frac{\|A - WH\|_F}{\sqrt{mn}},\ err_\beta = \frac{\sqrt{2D_\beta(A|WH)}}{\sqrt{mn}}. \tag{4}$$

Observe that *dbdiv_cpu* tries to minimize err_β but for $\beta = 2$ we get $err_F = err_\beta$. Obviously the highest decrease in err_F is expected to be reached when $\beta = 2$.

2.1 Test matrices

In this section the matrices used in the experiments are described.

1. **Random matrices:** The problem matrix A was generated randomly and so were the initialization matrices W_0 and H_0. The matrices were generated using a uniform distribution. To avoid side effects coming from the random number generation, some tests were performed with normal distribution generation too. Motivated by the results of the other experiments the tests were repeated with several ranges of the elements of the matrices:

   (a) *[0-1]:* The basic approach was to generate random matrices with entries between 0 and 1.

   (b) *[0-255]:* Grey scale images give rise to matrices with entries between 0 and 255. In this case random matrices in the same range of the images (0-255) were tested.

   (c) *[x-y]:* The results of the experiments with the previous two types showed that the range of the matrix values is related with the β parameter influence. Some more tests were executed with different ranges of values to prove this relation between β and the range of the matrix values.

 ISBN: 978-84-617-2230-3

2. **Synthetic matrices:** In this experiment two random matrices W and H with a fixed k were used to create the matrix A. The matrix was created as: $A = WH + \epsilon$, being ϵ a little constant error. The initial matrices W_0 and H_0 were generated randomly as in the experiments with type 1 matrices.

3. **Images:** This experiment was developed using grey-scale images as data matrices. For each image processed, a set of initial matrices (W_0, H_0) with certain k were created. Each set of 3 matrices (A, W_0, H_0) gives us a complete problem to test the β parameter and the number of iterations.

   (a) *[0-255]:* The original images were equivalent to matrices valued between 0 and 255.

   (b) *[0-1]:* Using the information of the random matrices experiment, some tests were done scaling the matrices to the range 0-1.

4. **Audio matrices:** The last experiment was to decompose matrices in the frequency domain [15] extracted from music tracks. These matrices were a great candidate to obtain information of the relation between the data of the matrix and the best β value because the matrices are extracted from a real world problem. The initial matrices W_0 and H_0 were generated following a uniform 0-1 distribution and a folded normal distribution in different tests.

   (a) *without scaling:* The original audio matrices in the spectrum domain elements had different ranges of values. The test were performed with 5 audio matrices called from this point forward: audio matrix 1-5.

   (b) *scaled:* The audio matrices of (a) were scaled down to values between 0 and 1.

3 Experimental results

Several tests with different datasets were done. In this section we will expose the results of those tests. Some especial test were motivated by the results of them on other datasets and will be explained in its corresponding subsections.

All the experiments were done in the same machine with the following technical specifications:

1. **Hardware:**

   (a) **CPU:** Intel(R) Xeon(R) CPU E5-2697 v2 @ 2.70GHz

   (b) **CPU physical cores:** 24

   (c) **RAM:** 128 GB

 ISBN: 978-84-617-2230-3

2. **Software:**
   (a) **Compilers:** gcc, icc.
   (b) **Parallelism libraries:** OpenMP, Intel OpenMP
   (c) **Generic Mathematical Libraries:** Intel MKL, Atlas (BLAS, LAPACK)
   (d) **Specific libraries:** NnmfPack v2.0

The NnmfPack library works over several architectures and can be installed with different software configurations. The NnmfPack has parallel implementations of the evaluated algorithm for GPU and man-ycore (Intel MIC), but in this paper we are not focusing on the execution speed so we did our test only on CPU. The software configuration used in this installation of the library was: icc as compiler and MKL as mathematical library.

3.1 Random matrices

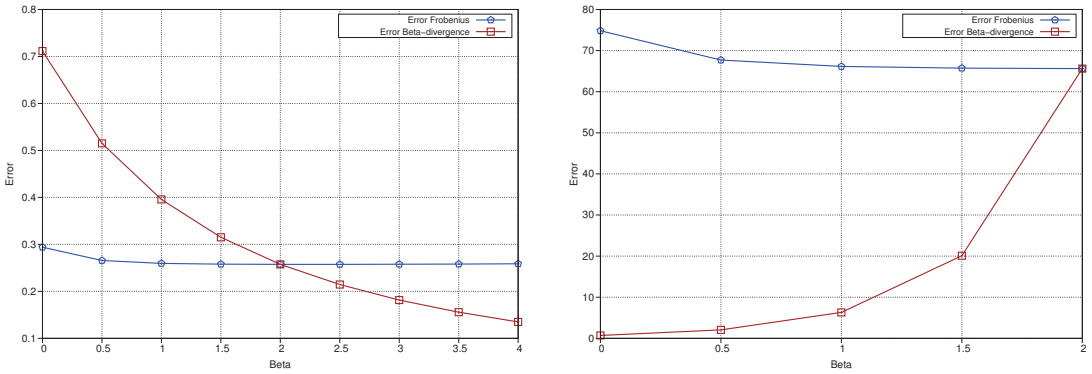
Tests proved that there was no difference between both types of random number generated matrices (normal distribution or uniform distribution), when the *dbdiv_cpu* algorithm is applied. The experiments with the matrices of type *1.(a)* [0-1] showed that the Frobenius error and the β-divergence error always decrease when the number of iterations are increased as expected. With regard to variation of β parameter, both error measures have their maximum value in $\beta = 0$ and decrease continuously as β increases up to $\beta = 2$. In the case of $\beta > 2$ a difference between both error measures appears. The Frobenius error has its minimum in $\beta = 2$. On the other hand, err_β decreases while β increases. In the multiplicative updates (3) the β is an exponent and this causes an overflow when big values of β (*e.g.* $\beta > 128$) are reached. Therefore the best β-divergence error that can be achieved is the one with the biggest β executable without overflowing. This behaviour is represented in Figure 1(a).

In the matrix type *1.(b)* [0-255] the Frobenius error keeps the same behaviour, with its best value in $\beta = 2$. Nevertheless the β-divergence error has its minimum value in $\beta = 0$ and increases with the increment of β. This behaviour is represented in Figure 1(b). Note the different value of the Frobenius error, which is 20 times lower in the [0-1] case than in the [0-255] case.

Finally for the matrix type *1.(c)* the behaviour was the same as for the type *1.(b)* tests. That is, Frobenius error having its minimum in $\beta = 2$ and β-divergence having it in $\beta = 0$. Furthermore, the errors were bigger for bigger values of the upper limit of the range (y) and for wider range ($y - x$).

3.2 Synthetic matrices

The goal of this experiment was to check the accuracy of the NNMF when we know that an exact solution exists for a given k. The error added to the matrix creation is to simulate



(a) Random 0-1 matrices

(b) Random 0-255 matrices

Figure 1: Evolution of Frobenius error and beta-divergence error

some noise in the data matrix.

The trends of both error measures were the same as in the case of the random matrices, in the [0-1] case as well as in the [x-y] case and both errors increase as the noise introduced in the generation process (ϵ) does. If $\epsilon = 0$ the algorithm returned the exact matrices for all β values.

3.3 Practical applications

3.3.1 Images

For the type *3.(a)* [0-255] matrices, the same behaviour is observed for both measures: With more iterations the error decreases. The lowest error value is obtained always with $\beta = 0$, increasing the error while β is increased (see 1(b)).

For the type *3.(b)* [0-1] matrices, both error measures decrease when the number of iterations increase. The err_β starts from the same value than in the matrices of type *3.(a)* but now decrease while β is increased. In addition, the Frobenius error has a different behaviour, keeping its lower value in $\beta = 0$ and increasing it as β grows. Despite maintaining its tendency, the scaling decreases notoriously the Frobenius error for all β values. Table 1 shows an example of these results.

Note that this experiments and the corresponding results have been carried out by iterating up to 400 iterations. This is the reason why it seems to have a different behaviour compared to the random and synthetic matrices experiments. That experiments showed that the Frobenius error always had its minimum value for $\beta = 2$, contrary to the results of that images test which gives us a minimum value for $\beta = 0$. But, increasing the number of iterations in several tests showed that the minimum Frobenius error value tends to happen

 ISBN: 978-84-617-2230-3

Table 1: Image value range comparison 0-255 vs 0-1 with 100 iterations

Case	β **value**				
	0	0,5	1	1,5	2
err_β 0-255	0.142010	0.465982	1.576253	5.424420	19.030323
err_β 0-1	0.142010	0.116609	0.098708	0.085005	0.0746287
err_F 0-255	15.296724	15.663154	16.539595	17.644217	19.030323
err_F 0-1	0.059987	0.061424	0.064861	0.069193	0.0746287

for $\beta = 2$. In the image whose data is in the Table 1 we achieve the point where the minimum Frobenius error matches $\beta = 2$ with 13200 iterations. This effect will be detailed in Section 3.3.2. Once achieved that point, the matrix types *3.(a)* and *3.(b)* behave like the matrix types *1.(b)* and *1.(a)*. This behaviour is represented in Figures 1(b) and 1(a).

3.3.2 Audio processing matrices

For the matrix type *4.(a)* as the number of iteration increases both error measures decrease as expected. The β parameter shows the influence of the relation between the data into the matrices giving us two different behaviours: the Frobenius error has its minimum for $\beta = 1.5$ for all audio matrices and the β-divergence error has its minimum in a different value depending on the matrix as shown in Table 2. The Frobenius error being minimal in $\beta = 1.5$ implies that each problem has a different optimum value for β. Furthermore, having β-divergence minimum values for different β values implies that each matrix can behave differently for the β-divergence error measure.All these measures were carried out with up to 400 iterations.

Table 2: Value of β parameter for the minimum β-divergence error

audio matrix #	1	2	3	4	5
β	0.5	0	0	0.5	0.5

For the audio matrices type *4.(b)* [0-1] both error measures decrease notoriously and behave in the same way as in the type *3.(b)* [0-1] (see Figure 1(a)). The Frobenius error decreases when β increases keeping its behaviour and the β-divergence error begins decreasing with the increment of β instead of increasing.

To ensure the Frobenius error will reach the best value in $\beta = 2$, as said in subsection 3.3.1, we tested all audio matrices (*4.(a)* and *4.(b)*) increasing the number of iterations. The Figure 2 shows the evolution of the Frobenius error for each β value. They always reach the point were the Frobenius error is smaller for $\beta = 2$ than for $\beta = 1.5$. But there is

no fixed number of iterations where that will occur (*e.g.* audio matrix 1 arrives to the point around 1200 iterations, audio matrix 2 around 6200 and audio matrix 3 around 3300).

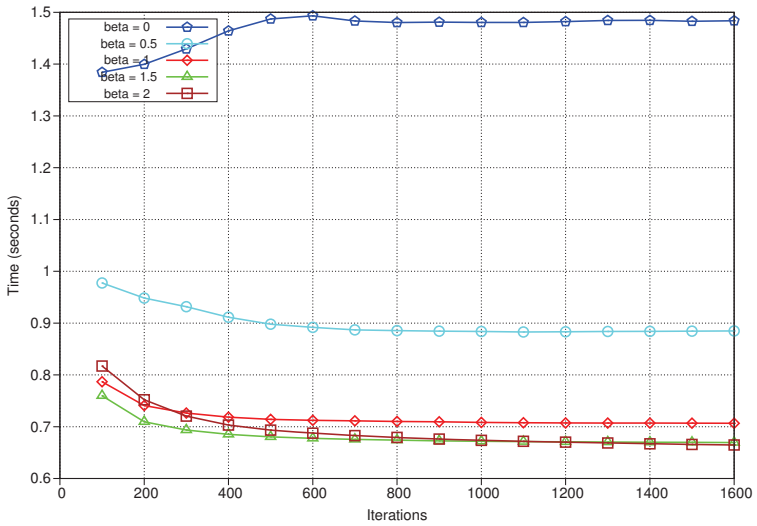


Figure 2: Evolution of the Frobenius error when the iteration number is increased in audio matrix 1.

4 Experimental analysis

We outline here the main results obtained in our experimental analysis:

1. As the experiments proved, increasing the number of iterations always decreases the error obtained, with independence on the β parameter. This is the expected behaviour because the multiplicative algorithm used is based on a gradient descent minimization. Is the user choice to increase the number of iterations to get a better result. There is a trade-off between computing time and accuracy that the user must evaluate.

2. The experiments also showed that scaling the matrix to the range 0-1 both error measures decrease, independently of what is the chosen β value. Also, when the matrices are scaled to this range, β-divergence error decreases with the increment of β indefinitely as seen in Section 3.1 .

3. The Frobenius error always has its best value in $\beta = 2$ if enough iterations are done. That is completely logical because for $\beta = 2$ the β-divergence is mathematically equivalent to the Frobenius error. So for $\beta = 2$ we are minimizing the Frobenius error.

 ISBN: 978-84-617-2230-3

4. If we only take the iteration number as measure of algorithm progression, it seems not worthwhile to increase the number of iterations so much to achieve the minimum value of the Frobenius error at $\beta = 2$. But if we consider execution time it is worthwhile. The NnmfPack library has been optimized for certain values of β. In the case of $\beta = 2$ the library uses an efficient implementation of the MLSA algorithm (see [10]) which is considerably faster than the generic β-divergence algorithm. In Figure 3 you can see the variation of execution times when iterations are increased. Generic cases ($\beta = 0$, $\beta = 0.5$ and $\beta = 1.5$) have the same computational cost. As can be seen, a big amount of iterations for $\beta = 2$ are still faster than a few iterations for $\beta = 1.5$ achieving a smaller Frobenius error.

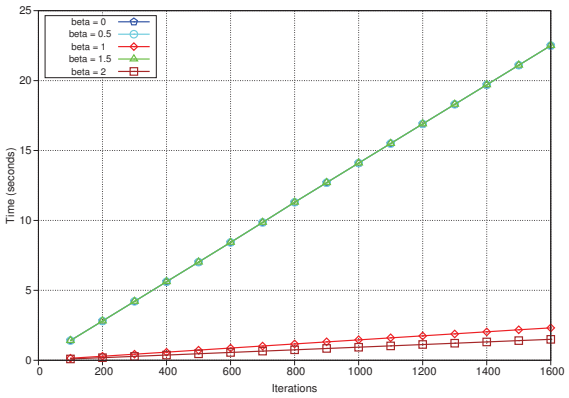


Figure 3: Execution times.

5. We conclude that the NNMF exploits the relation in the data inside the matrix because the error for real applications was lower than the error of random problems of the same characteristics.

5 Conclusions and future work

In conclusion, the experiments showed that the β-divergence algorithm for the NNMF has different behaviours depending on the range of the values of the data matrix, being the data range 0-1 the one with lowest errors.

It is also seen that with enough iterations the lowest Frobenius error is always achieved with $\beta = 2$. However, depending on the data, algorithm with $\beta = 2$ can be slow compared to others with $\beta \neq 2$.

Finally, we must remark that matrices from real applications (image and audio) always reach better error with the same number of iterations than its random equivalent. This

shows us that the NNMF exploits the relation between data as we suspected.

As future work, it will be interesting to test if it is possible to add an automatic scaling to the 0-1 range before the algorithm execution and a scaling to the original range after without changing the solution of the problem. If it is possible it will be an easy way to improve the results of the algorithm.

Another interesting thing will be to explore if a lower β-divergence error its better for real applications despite having a worse Frobenius error.

We well keep going deep the understanding of the relation between the data on the matrices and the several divergences that the β-divergence algorithm offers.

Acknowledgments. This work has been partially supported by Ministerio de Economía y Competitividad from Spain, under the projects TEC2012-38142-C04-01 and TEC2012-38142-C04-04 and by PROMETEO FASE II 2014/003 projects of Generalitat Valenciana. The authors wish to thank Professor Pedro Vera and his research group "Signal Processing in Telecommunications Systems" at the University of Jaén for providing the audio matrices used in this article.

References

[1] E. Battenberg, A. Freed and D. Wessel, *Advances in the Parallelization of Music and Audio Applications*, Proceedings of the International Computer Music Conference, New York City/Stony Brook, New York, 2010.

[2] J. Wnag, W. Zhong and J. Zhang, *NNMF-Based Factorization Techniques for High-Accuracy Privacy Protection on Non-negative-valued Datasets*, Proceedings of the Sixth IEEE International Conference on Computing and Processing, Data Mining Workshops ICDM Workshops, 2006, pp. 513-517.

[3] F.J. Rodriguez-Serrano, J.J.Carabias-Orti, P. Vera-Candeas, T.Virtanen and N. Ruiz-Reyes, *Multiple Instrument Mixtures Source Separation Evaluation Using Instrument-Dependent NMF Models*, LNCS 7191, Springer-Verlag, 2012.

[4] M.W. Berry, M. Browne, A. Langville, V. Pauca and R. Plemmons, *Algorithms and applications for approximate nonnegative matrix factorization*, Comput. Statist. Data Anal., vol. 52, pp. 155-173, 2007.

[5] K. Devajaran, *Nonnegative Matrix Factorization: An Analytical and Interpretative Tool in Computational Biology*, PLoS Comput Biol 4(7): e1000029. doi:10.1371/journal.pcbi.1000029, 2008.

 ISBN: 978-84-617-2230-3

[6] D.D. Lee and H.S. Seung, *Algorithms for non-negative matrix factorization*, Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 2001.

[7] J. Kim and H. Park, *Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method*, SIAM J. Matrix Anal. Appl., vol. 30, 2008, pp. 713-730.

[8] N. Guan, D. Tao, Z. Luo and B. Yuan, *NeNMF: An Optimal Gradient Method for Non-negative Matrix Factorization*, IEEE Transactions on Signal Processing, vol. 60(6), pp. 2882-2898, 2012.

[9] A. Cichocki, R. Zdunek and S.I Amari, *Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. Independent Component Analysis and Signal Separation*, LNCS 4666, Springer Springer-Verlag, 2007.

[10] P. Alonso, V.M. García, F.J. Martínez-Zaldívar, A. Salazar, L. Vergara and A.M. Vidal, *Parallel approach to NNMF on multicore architecture*, Journal of Supercomputing, DOI 10.1007/s11227-013-108, 2014.

[11] N. Díaz-Gracia, A. Cocaña-Fernández, M. Alonso-González, F.J. Martínez-Zaldívar, R. Cortina, V.M. García-Mollá, P. Alonso, J. Ranilla, A.M. Vidal, *NNMFPACK: a versatile approach to an NNMF parallel library*, Proceedings of the 2014 International Conference on Computational and Mathematical Methods in Science and Engineering, Cdiz, 2014, pp. 456-465.

[12] N Díaz-Gracia, A Cocaña-Fernández, M Alonso-González, FJ Martínez-Zaldívar, R Cortina, VM García-Mollá, P Alonso, J Ranilla, and AM Vidal. Improving nnmfpack with heterogeneous and efficient kernels for beta-divergence metrics. *The Journal of Supercomputing*, pages 1–11, 2014.

[13] M. Minami and S. Eguchi, *Robust blind source separation by beta-divergence*, Neural Computation, vol. 14, pp. 1859-1886, 2002.

[14] C. Févotte, N. Bertin, and J.-L. Durrieu *Nonnegative matrix factorization with the Itakura-Saito divergence. With application to music analysis*, Neural Computation, vol. 21, pp. 793-830, 2009.

[15] Francisco Jesus Canadas-Quesada, Pedro Vera-Candeas, Nicolas Ruiz-Reyes, Julio Carabias-Orti, and Pablo Cabanas-Molero. Percussive/harmonic sound separation by non-negative matrix factorization with smoothness/sparseness constraints. *EURASIP Journal on Audio, Speech, and Music Processing*, 2014(1):1–17, 2014.

 ISBN: 978-84-617-2230-3