2016 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 13–16, 2016, SALERNO, ITALY

# MINI-BATCH STOCHASTIC APPROACHES FOR ACCELERATED MULTIPLICATIVE UPDATES IN NONNEGATIVE MATRIX FACTORISATION WITH BETA-DIVERGENCE

*Romain Serizel, Slim Essid, Gaël Richard*

LTCI, CNRS, Télécom ParisTech, Université Paris - Saclay, 75013, Paris, France

## ABSTRACT

Nonnegative matrix factorisation (NMF) with $\beta$-divergence is a popular method to decompose real world data. In this paper we propose mini-batch stochastic algorithms to perform NMF efficiently on large data matrices. Besides the stochastic aspect, the mini-batch approach allows exploiting intensive computing devices such as general purpose graphical processing units to decrease the processing time and in some cases outperform coordinate descent approach.

***Index Terms***— Nonnegative matrix factorisation, GPGPU, multiplicative rules, online learning

## 1. INTRODUCTION

Nonnegative matrix factorisation (NMF) is a popular data decomposition method [1]. NMF indeed allows obtaining meaningfull decompositions that are well suited to the underlying structure of the processed data. Over the past years, owing to these assets NMF has been used in a wide variety of applications ranging from text analysis [1], to electroencephalogram decomposition [2, 3], image processing [4], blind source separation [5, 6], music transcription [7] machine listening [8] or speech processing [9, 10].

When considering a large amount of data, applying NMF can quickly become computationally demanding. Therefore, over the years, a substantial amount of work has been dedicated to the design of fast and low complexity optimisation algorithms for NMF. These methods include the approaches based on coordinate descent (CD) [11, 12] and the fast hierarchical alternating least squares method (FastHALS) [11] where each coordinate is optimised sequentially. Other examples include the distributed approaches [13] that could allow one to take advantage of distributed computation architectures and more recently online approaches [4, 14] inspired the stochastic gradient approach [15] where the decomposition is updated on a subset of the data drawn randomly. Most of these approaches focus on the minimisation of a least square criterion that assumes a Gaussian noise model.

When considering real world signals, it is often the case that the noise model is not Gaussian but can be a Poisson distribution or a Gamma distribution in which case the generalised Kullback-Leibler (KL) divergence [16] and the Itakura-Saito (IS) divergence [17] can be more appropriate, respectively.

CD-based approaches have been extended to the KL problem [11, 12] but to our best knowledge the IS problem was never considered. The multiplicative update rules (MU) first introduced for the Euclidean distance [1] and later extended to the KL divergence [18] can be generalised to the $\beta$-divergence that encompasses the Euclidean distance, the KL divergence and the IS divergence [19, 20]. Therefore, MU rules became increasingly popular when dealing with real world signals and in particular audio signals. However, MU rules can be computationally demanding and are generally outperformed by gradient descent based approaches when considering cost functions based on the Euclidean distance.

In this paper, we propose mini-batch methods for accelerated MU rules. These approaches are inspired by the work based on online learning for the Euclidean distance [4, 14] and on the stochastic average gradient approach (SAG) [21]. Besides, these approaches can be run on devices with constraint memory allowing for exploitation of general purpose graphical processing units (GPGPU). When dealing with matrices operations (which is essentially the case in MU rules), GPGPU significantly decrease the processing time further compared to central processing units (CPU) and can outperform CD based approaches even with Euclidean distance cost.

This paper is organised as follows. The notations and the general NMF problem are introduced in Section 2. CD for Euclidean distance and KL divergence are briefly described in Section 3. In Section 4 we propose mini-batch approaches for MU rules in the general case of the $\beta$-divergence and consideration about GPGPU implementation are exposed in Section 5. We present the experiments results and discussion in Section 6 and conclusions are exposed in Section 7.

## 2. PROBLEM STATEMENT AND NOTATIONS

Consider the (nonnegative) matrix $\mathbf{V} \in \mathbb{R}_+^{F\times N}$. Without loss of generality we consider here that $\mathbf{V}$ is the time-feature space representation of a time series where $N$ the number of frames and $F$ is the number of features per frames. The goal of NMF [1] is to find a factorisation for $\mathbf{V}$ of the form:

$$\mathbf{V} \approx \mathbf{WH} \tag{1}$$

where $\mathbf{W} \in \mathbb{R}_+^{F\times K}$, $\mathbf{H} \in \mathbb{R}_+^{K\times N}$ and $K$ is the number of components in the decomposition. The NMF model estimation is usually considered as solving the following optimisation problem:

$$\min_{\mathbf{WH}} D(\mathbf{V}|\mathbf{WH}) \quad \text{s.t.} \quad \mathbf{W} \geq 0,\ \mathbf{H} \geq 0 \tag{2}$$

where $D$ is a separable divergence such as:

$$D(\mathbf{V}|\mathbf{WH}) = \sum_{f=1}^{F}\sum_{n=1}^{N} d([\mathbf{V}]_{fn}|[\mathbf{WH}]_{fn}), \tag{3}$$

with $[.]_{fn}$ is the element on the $n^{th}$ column and the $f^{th}$ line of a matrix and $d$ is a scalar cost function. Similarly let $[.]_{:n}$ and $[.]_{f:}$ be the $n^{th}$ column and the $f^{th}$ row of a matrix. A common choice for the cost function is the $\beta$-divergence [19]. Popular cost functions such as

This work was partly funded by the European Union under the FP7-LASIE project (grant 607480).

978-1-5090-0746-2/16/$31.00 

the Euclidean distance, the generalised KL divergence [16] and the IS divergence [17] are all particular cases of the $\beta$-divergence (obtained for $\beta = 2$, 1 and 0, respectively). The use of the $\beta$-divergence for NMF has been studied extensively in Févotte et al. [20]. In most cases the NMF problem is solved using a two-block coordinate descent approach. Each of the factors $\mathbf{W}$ and $\mathbf{H}$ is optimised alternatively. The sub-problem in one factor can then be considered as a nonnegative least square problem (NNLS) [22]. Two different approaches are considered here to solve these NNLS problems: the coordinate descent approach [11, 12] and the MU rules [1].

## 3. NMF WITH COORDINATE DESCENT

The main idea in the coordinate descent (CD) method is to update one coordinate at a time until convergence. Recently a method called FastHALS was proposed to solve the NMF with Euclidean norm with CD [11] and an extension to KL was later proposed in Hsieh et al. [12]. Both these methods are described below and are used as baseline for performance evaluation.

### 3.1. Euclidean distance

The goal in CD for NMF is to solve alternatively the set of one variable sub-problems deriving from (3). For $(k, n) \in [\![1, K]\!] \times [\![1, N]\!]$ solve (4). Then for $(f, k) \in [\![1, F]\!] \times [\![1, K]\!]$ solve (5) (see also Algorithm 1).

$$\min_{[\mathbf{H}]_{kn}} \frac{1}{2} \sum_f ([\mathbf{V}]_{fn} - [\mathbf{WH}]_{fn} - [\mathbf{W}]_{fk}[\mathbf{H}]_{kn})^2 \quad \text{s.t.} \quad [\mathbf{H}]_{kn} \geq 0 \tag{4}$$

$$\min_{[\mathbf{W}]_{fk}} \frac{1}{2} \sum_n ([\mathbf{V}]_{fn} - [\mathbf{WH}]_{fn} - [\mathbf{W}]_{fk}[\mathbf{H}]_{kn})^2 \quad \text{s.t.} \quad [\mathbf{W}]_{fk} \geq 0 \tag{5}$$

A full pass through the data is referred to as an epoch to avoid confusion with local iteration (on the coordinate of the matrices or on the mini-batch in the next section). The number of epochs performed by the algorithm is controlled by a parameter. Each one variable sub-problem can be solved exactly. The solutions of these problems lead to the following update rules [11]:

$$[\mathbf{H}]_{kn} \leftarrow \left[ [\mathbf{H}]_{kn} - \frac{[\mathbf{W}^T\mathbf{WH} - \mathbf{W}^T\mathbf{X}]_{kn}}{[\mathbf{W}^T\mathbf{W}]_{kk}} \right]_+ \tag{6}$$

$$[\mathbf{W}]_{fk} \leftarrow \left[ [\mathbf{W}]_{fk} - \frac{[\mathbf{WHH}^T - \mathbf{XH}^T]_{fk}}{[\mathbf{HH}^T]_{kk}} \right]_+, \tag{7}$$

where $[.]_+$ is the half-wave rectifying operator $[.]_+ = \max(., 0)$.

### 3.2. Generalised Kullback-Leibler divergence

A similar approach can be applied to NMF with KL divergence except that now the sets of one variable sub-problems do not have a closed form solution [11, 12]. However, the cost functions are twice differentiable, therefore, the sub-problems can be solved with Newton method which lead to the following update rules [12]:

$$[\mathbf{H}]_{kn} \leftarrow \left[ [\mathbf{H}]_{kn} - \frac{\sum_f [\mathbf{W}]_{fk}(1 - \frac{[\mathbf{X}]_{fn}}{[\mathbf{WH}]_{kn}})}{\sum_f \frac{[\mathbf{X}]_{fn}[\mathbf{W}]_{fk}}{[\mathbf{WH}]_{kn}^2}} \right]_+ \tag{8}$$

$$[\mathbf{W}]_{fk} \leftarrow \left[ [\mathbf{W}]_{fk} - \frac{\sum_n [\mathbf{H}]_{kn}(1 - \frac{[\mathbf{X}]_{fn}}{[\mathbf{WH}]_{kn}})}{\sum_n \frac{[\mathbf{X}]_{fn}[\mathbf{H}]_{kn}}{[\mathbf{WH}]_{kn}^2}} \right]_+. \tag{9}$$

**Algorithm 1** CD for the Euclidean norm (FastHALS)

**Require:** $\mathbf{V} \in \mathbb{R}_+^{F \times N}$, max_epoch
Initialise $\mathbf{H}$, $\mathbf{W}$ with nonnegative random coefficients
**for** $ep = 0$; $ep <$ max_epoch **do**
  Compute $\mathbf{W}^T\mathbf{W}$ and $\mathbf{W}^T\mathbf{X}$
  **for** $n = 1$; $j \leq N$ **do**
    **for** $k = 1$; $r \leq K$ **do**
      Update $[\mathbf{H}]_{kn}$ with (6)
    **end for**
  **end for**
  Compute $\mathbf{HH}^T$ and $\mathbf{XH}^T$
  **for** $f = 1$; $j \leq F$ **do**
    **for** $k = 1$; $r \leq K$ **do**
      Update $[\mathbf{W}]_{fk}$ with (7)
    **end for**
  **end for**
**end for**
**return** $\mathbf{H}$, $\mathbf{W}$

**Algorithm 2** CD for the KL divergence

**Require:** $\mathbf{V} \in \mathbb{R}_+^{F \times N}$, max_epoch
Initialise $\mathbf{H}$, $\mathbf{W}$ with nonnegative random coefficients
**for** $ep = 0$; $ep <$ max_epoch **do**
  Compute $\mathbf{W}^T\mathbf{W}$ and
  **for** $n = 1$; $j \leq N$ **do**
    **for** $k = 1$; $r \leq K$ **do**
      **repeat**
        Update $[\mathbf{H}]_{kn}$ with (8)
        $[\mathbf{WH}]_{:n} \leftarrow \mathbf{W}[\mathbf{H}]_{:n}$
      **until** Convergence
    **end for**
  **end for**
  Compute $\mathbf{HH}^T$
  **for** $f = 1$; $j \leq F$ **do**
    **for** $k = 1$; $r \leq K$ **do**
      **repeat**
        Update $[\mathbf{W}]_{fk}$ with (9)
        $[\mathbf{WH}]_{f:} \leftarrow [\mathbf{W}]_{f:}\mathbf{H}$
      **until** Convergence
    **end for**
  **end for**
**end for**
**return** $\mathbf{H}$, $\mathbf{W}$

## 4. BETA-NMF WITH MULTIPLICATIVE UPDATES

The MU rules introduced in Lee et al. [1] for the Euclidean distance were obtained using the heuristic which consists in expressing the gradient of the cost function (2) as the difference between a positive contribution and a negative contribution. The MU rules then have the form of a quotient of the negative contribution by the positive contribution. These update rules were later generalised to the KL divergence [18] and the $\beta$-divergence [20] leading to the following expressions:

$$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{W}^T \left[ (\mathbf{WH})^{\beta-2} \odot \mathbf{V} \right]}{\mathbf{W}^T (\mathbf{WH})^{\beta-1}} \tag{10}$$

$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{\left[(\mathbf{WH})^{\beta-2} \odot \mathbf{V}\right] \mathbf{H}^T}{(\mathbf{WH})^{\beta-1}\mathbf{H}^T}; \tag{11}$$

where $\odot$ is the element-wise product (Hadamard product) and division and power are element-wise. The matrices $\mathbf{H}$ and $\mathbf{W}$ are then updated according to the alternating update scheme described in Algorithm 3.

**Algorithm 3** Basic alternating scheme for MU rules

**Require:** $\mathbf{V} \in \mathbb{R}_+^{F \times N}$, $\beta$, max_iter
Initialise $\mathbf{H}$, $\mathbf{W}$ with nonnegative random coefficients
**for** $it = 0$; $it <$ max_iter **do**
  Update $\mathbf{H}$ with (10)
  Update $\mathbf{W}$ with (11)
**end for**
**return** $\mathbf{H}$, $\mathbf{W}$

### 4.1. Cyclic mini-batch updates

The standard update scheme (described in Algorithm 3) requires the complete matrix $\mathbf{V}$. When considering time series with a large number of data points (or time series that are expending in time) running this algorithm can become prohibitive. Capitalising on the separability of the divergence (3) it is possible to perform NMF on mini-batch of data to reduce the computational burden or to allow for parallel computations [13].

When considering time series as defined above, each column ($[\mathbf{V}]_{:n}$) of the matrix $\mathbf{V}$ contains all the features for a specific time frame and each row ($[\mathbf{V}]_{f:}$) represents a particular feature along time. The number of rows is then a parameter of the low-level representation and only the number of columns can increase while increasing the amount of data. Therefore, in contrast to the approach proposed in Şimşekli et al. [13] we decide to decompose the matrix $\mathbf{V}$ in $B$ mini-batches of time frames that contain all the features for the given frame (see also Figure 1).

For each batch $b$, the update of the activations $\mathbf{H}_b$ corresponding to $\mathbf{V}_b$ can be obtained independently from all the other batches with the standard MU rules (10). The update of the bases $\mathbf{W}$ on the other hand requires the whole matrix $\mathbf{V}$ to be processed. The positive contribution of the gradient (thereafter denoted $\nabla^+\mathbf{W}$) and the negative contribution of the gradient (thereafter denoted $\nabla^-\mathbf{W}$) are accumulated along the mini-batches. $\mathbf{W}$ is updated once per epoch with the standard MU rule (11) as described in Algorithm 1. Note that this is theoretically similar to the standard full-gradient (FG) MU rules.

**Algorithm 4** Cyclic mini-batch for MU rules

**Require:** $\mathbf{V} \in \mathbb{R}_+^{F \times N}$, $\beta$, max_epoch
Initialise $\mathbf{H}$, $\mathbf{W}$ with nonnegative random coefficients
**for** $ep = 0$; $ep <$ max_epoch **do**
  Initialise $\nabla^-\mathbf{W} = 0$ and $\nabla^+\mathbf{W} = 0$
  **for** $b = 1$; $b < B$ **do**
    Update $\mathbf{H}_b$ with (10)
    $\nabla^-\mathbf{W}$ += $\nabla^-\mathbf{W}_b$
    $\nabla^+\mathbf{W}$ += $\nabla^+\mathbf{W}_b$
  **end for**
  $\mathbf{W} \leftarrow \frac{\nabla^-\mathbf{W}}{\nabla^+\mathbf{W}}$
**end for**
**return** $\mathbf{H}$, $\mathbf{W}$

### 4.2. Stochastic mini-batch updates

When aiming at minimizing an Euclidean distance, it has been shown that drawing samples randomly can improve greatly the convergence speed in dictionary learning and by extension in NMF [4, 14]. We propose to apply a similar idea to MU rules in order to take advantage of the wide variety of divergences covered by the $\beta$-divergence. Instead of selecting the mini-batch sequentially on the original data $\mathbf{V}$ as in Section 4.1, we propose to draw mini-batches randomly on a shuffled version of $\mathbf{V}$. The mini-batch update of $\mathbf{H}$ still needs one full pass through the data so the mini-batches are drawn from $\mathbf{b}_{rnd}$, a random permutation of $[\![1, B]\!]$. On the other hand, the shuffling of $\mathbf{V}$ is then used to ensure that there is enough data variability in each mini-batch and a single mini-batch can be used to update $\mathbf{W}$.

Two different strategies can be considered to update $\mathbf{W}$. The first option is to update $\mathbf{W}$ for each mini-batch as described in Algorithm 5. This approach is denoted asymmetric SG mini-batch MU rules (ASG-MU) as $\mathbf{H}$ and $\mathbf{W}$ are updated asymmetrically (the full $\mathbf{H}$ is updated once per epoch while $\mathbf{W}$ is updated for each mini-batch). The second option is to update $\mathbf{W}$ only once per epoch on a randomly selected mini-batch. In practice as mini-batches are drawn from a random permutation of $[\![1, B]\!]$ updating $\mathbf{W}$ on the last mini-batch selected is equivalent to select a random mini-batch as described in Algorithm 6. This approach will be referred to as greedy SG mini-batch MU rules (GSG-MU).

**Algorithm 5** Asymmetric SG mini-batch MU rules (ASG-MU)

**Require:** $\mathbf{V} \in \mathbb{R}_+^{F \times N}$, $\beta$, max_epoch
Initialise $\mathbf{H}$, $\mathbf{W}$ with nonnegative random coefficients
Shuffle $\mathbf{V}$
**for** $ep = 0$; $ep <$ max_epoch **do**
  $\mathbf{b}_{rnd} \leftarrow$ permutation of $[\![1, B]\!]$
  **for** $b \in \mathbf{b}_{rnd}$ **do**
    Update $\mathbf{H}_b$ with (10)
    Update $\mathbf{W}$ with (11) (with $\mathbf{H}$ replaced by $\mathbf{H}_b$)
  **end for**
**end for**
**return** $\mathbf{H}$, $\mathbf{W}$

**Algorithm 6** Greedy SG mini-batch MU rules (GSG-MU)

**Require:** $\mathbf{V} \in \mathbb{R}_+^{F \times N}$, $\beta$, max_epoch
Initialise $\mathbf{H}$, $\mathbf{W}$ with nonnegative random coefficients
Shuffle $\mathbf{V}$
**for** $ep = 0$; $ep <$ max_epoch **do**
  $\mathbf{b}_{rnd} \leftarrow$ permutation of $[\![1, B]\!]$
  **for** $b \in \mathbf{b}_{rnd}$ **do**
    Update $\mathbf{H}_b$ with (10)
  **end for**
  Update $\mathbf{W}$ with (11) (with $\mathbf{H}$ replaced by $\mathbf{H}_{[\mathbf{b}_{rnd}]_B}$)
**end for**
**return** $\mathbf{H}$, $\mathbf{W}$

### 4.3. Stochastic average gradient mini-batch updates

SAG [21] is a method recently introduced for optimizing cost functions that are a sum of convex functions (which is the case here). SAG provides an intermediate between FG-like methods (as used in Section 4.1) and SG-like methods (as used in Section 4.2). SAG then allows to obtain similar convergence rate than FG methods with a complexity comparable to SG methods.

We propose to apply SAG-like methods to update the dictionaries $\mathbf{W}$ in a mini-batch based NMF. Note that, as the full pass through the data is needed to update $\mathbf{H}$ it would not make sense to apply SAG here. The key idea is that for each mini-batch $b$ instead of using the gradient computed locally to update $\mathbf{W}$, we propose to use

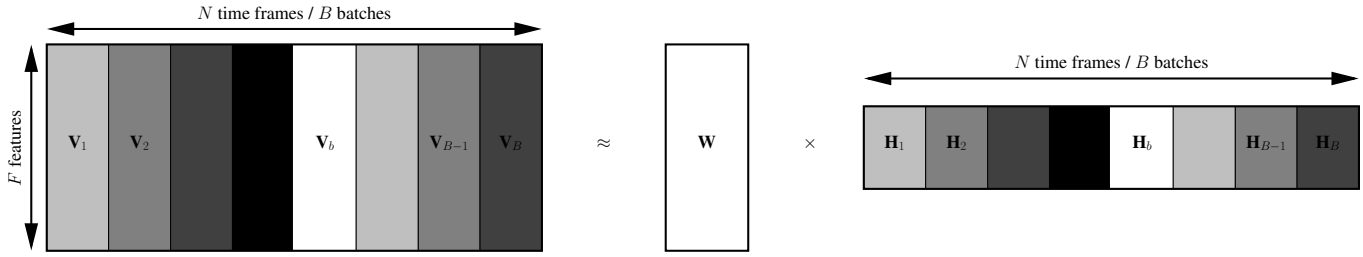



**Fig. 1**. Mini-batch decomposition of the data matrix $\mathbf{V}$

the mini-batch data to update the full gradient negative and positive contributions. In the case of MU rules, in order to prevent the oldest (and possibly outdated) values of the gradient from causing instabilities we use an averaging based on an exponential forgetting factor $\lambda \in [0, 1]$:

$$\nabla^{-}\mathbf{W} \leftarrow (1-\lambda)\nabla^{-}\mathbf{W} + \lambda\nabla^{-}_{\text{new}}\mathbf{W}_b \quad (12)$$

$$\nabla^{+}\mathbf{W} \leftarrow (1-\lambda)\nabla^{+}\mathbf{W} + \lambda\nabla^{+}_{\text{new}}\mathbf{W}_b \quad (13)$$

where $\nabla^{-}_{\text{new}}\mathbf{W}_b$ and $\nabla^{+}_{\text{new}}\mathbf{W}_b$ are the negative and positive contribution to the gradient of $\mathbf{W}$ calculated on the mini-batch $b$, respectively. Note that for $\lambda = 1$ this gradient update formulation is equivalent to the SG approach described above.

Similarly as in Section 4.2, two different strategies can be considered to update $\mathbf{W}$. The first option is to update $\mathbf{W}$ for each mini-batch as described in Algorithm 7. This approach is denoted asymmetric SAG mini-batch MU rules (ASAG-MU). The second option is to update $\mathbf{W}$ only once per epoch on the last mini-batch from the permutation $\mathbf{b}_{rnd}$, as described in Algorithm 8. This approach is termed greedy SAG mini-batch MU rules (GSAG-MU).

**Algorithm 7** Asymmetric SAG mini-batch MU rules (ASAG-MU)

**Require:** $\mathbf{V} \in \mathbb{R}_{+}^{F\times N}$, $\beta$, max_epoch
Initialise $\mathbf{H}$, $\mathbf{W}$ with nonnegative random coefficients
Shuffle $\mathbf{V}$
**for** $ep = 0$; $ep <$ max_epoch **do**
  $\mathbf{b}_{rnd} \leftarrow$ permutation of $[\![1, B]\!]$
  **for** $b \in \mathbf{b}_{rnd}$ **do**
    Update $\mathbf{H}_b$ with (10)
    Udpate $\nabla^{-}\mathbf{W}$ with (12)
    Udpate $\nabla^{+}\mathbf{W}$ with (13)
    $\mathbf{W} \leftarrow \frac{\nabla^{-}\mathbf{W}}{\nabla^{+}\mathbf{W}}$
  **end for**
**end for**
**return** $\mathbf{H}$, $\mathbf{W}$

## 5. GPGPU IMPLEMENTATION FOR ACCELERATED UPDATES

Over the past years, GPGPU become increasingly popular in scientific computing as they allow one to drastically improve execution speed on computationally intensive functions such as manipulation on large matrices. Owing to these assets, GPGPU programming played a central part in the recent success of deep learning and many GPGPU programming packages have emerged among which: Theano [23], Torch7 [24] or tensorflow [25]. During this work we decided to use Theano which is a python library and therefore allows flexible integration within a full scientific programming framework.

The latency in data transfers to and from the GPGPU internal memory is a critical aspect in GPGPU programming. Indeed the cost in time of these transfers is high and transfering data too often to and from the GPGPU internal memory can quickly overcome the benefits of GPGPU programming. One consequence of this limitation is that loop-based algorithms have to be treated really carefully as any transfer to or from the GPGPU internal memory inside the loop would result in slow execution.

**Algorithm 8** Greedy SAG mini-batch MU rules (GSAG-MU)

**Require:** $\mathbf{V} \in \mathbb{R}_{+}^{F\times N}$, $\beta$, max_epoch
Initialise $\mathbf{H}$, $\mathbf{W}$ with nonnegative random coefficients
Shuffle $\mathbf{V}$
**for** $ep = 0$; $ep <$ max_epoch **do**
  $\mathbf{b}_{rnd} \leftarrow$ permutation of $[\![1, B]\!]$
  **for** $b \in \mathbf{b}_{rnd}$ **do**
    Update $\mathbf{H}_b$ with (10)
  **end for**
  Udpate $\nabla^{-}\mathbf{W}$ with (12) (with $\mathbf{H}$ replaced by $\mathbf{H}_{[\mathbf{b}_{rnd}]_B}$)
  Udpate $\nabla^{+}\mathbf{W}$ with (13) (with $\mathbf{H}$ replaced by $\mathbf{H}_{[\mathbf{b}_{rnd}]_B}$)
  $\mathbf{W} \leftarrow \frac{\nabla^{-}\mathbf{W}}{\nabla^{+}\mathbf{W}}$
**end for**
**return** $\mathbf{H}$, $\mathbf{W}$

### 5.1. Coordinate descent

The general CD scheme relies on iterations over the coordinate of the factor. There is only limited performance gains in performing the full nested loop section on the GPGPU (except that it would reduce the risk of unecessary memory transfer). The positive impact of GPGPU computing on CD algorithms essentially resides in the dot product to be performed at the beginning of each epoch. For Euclidean distance cost (Algorithm 1), the dot products $\mathbf{W}^T\mathbf{W}$ (line 3), $\mathbf{W}^T\mathbf{X}$ (line 3), $\mathbf{H}\mathbf{H}^T$ (line 9) and $\mathbf{X}\mathbf{H}^T$ (line 9) are computed on the GPGPU, the rest is executed on the CPU side. For KL divergence cost $\mathbf{W}^T\mathbf{W}$ (line 3) and $\mathbf{H}\mathbf{H}^T$ (line 12) are computed on the GPGPU, the rest is executed on the CPU side.

### 5.2. Multiplicative updates

By design, the MU rules essentially rely on matrices operation and therefore appear as a good match for GPGPU programming. The only looping in MU rules is related to iterating on epoch and on mini-batch, all the rest (the core of the optimisation algorithm) can be executed on GPGPU. Therefore, for each variation on the MU rules described above, the portion of code running on the GPGPU is the following: cyclic mini-batch MU (Algorithm 4, lines 5–7 and 9), ASG-MU (Algorithm 5, lines 6–7), GSG-MU (Algorithm 6, lines 6 and 8), ASAG-MU (Algorithm 7, lines 6–9) and GSAG-MU (Algorithm 8, lines 6 and 8–10).

## 6. EXPERIMENTS

### 6.1. Experimental setup and corpus

The NMF algorithms are evaluated on a subset of the ESTER corpus. ESTER is a corpus for automatic speech recognition composed of data recorded on broadcast radio [26]. The subset of ESTER used for evaluation is composed of 6 hours and 11 minutes of training data [10], 132 constant Q transform [27] coefficients are extracted 16ms frames resulting in a $132 \times 1\ 394\ 375$ data matrix $\mathbf{V}$. A larger corpus would not fit in the memory of the GPGPU used in these experiments (see also below) and we would not be able to compare standard batch algorithms to the proposed mini-batch algorithms. Note also that on much smaller datasets, the benefits of using a GPGPU would vanish and CPU may outperform GPGPU.

Two different platforms are used to run the CPU code and the GPU code[1]. The CPU code runs on Intel® Xeon® E5-2687W with 16 cores at 3.10GHz and 64GiB, processing is accelerated with Intel® math kernel library (Intel® MKL) to fully exploit the 16 cores. The GPU code runs on Nvidia® Tesla S2050 with 3GiB internal memory. The host computer is equipped with 4 Intel® Xeon® X5670 with 6 cores at 2.93GHz each (24 core in total) and 192GiB of RAM. The access to the GPGPU platform is mutualised so there is no guarantee to access all the 24 cores at once within a limited waiting time. Therefore it was decided to run the CPU part of the code on a single core without Intel® MKL acceleration to ensure consistent hardware setup from one experiment to another.

NMF are trained with $K = 100$ components. The Euclidean norm cost and the KL divergence cost and the IS divergence cost are considered. In order to limit the performance variability due to the random initialisation, the processing time and the value of the cost function are averaged over 5 different initialisations and the set of 5 initialisations is the same for all the algorithms in order to ensure consistency of the tests. The forgetting factor for the SAG-based algorithms is set to $\lambda = 2$ for the Euclidean norm cost and and the KL divergence and $\lambda = 1$ for IS divergence. The mini-batch size is $50\ 000$. For clarity, we decided to plot the logarithm of the cost function normalised by the initial cost function:

$$\log\left(\frac{D(\mathbf{X}|\mathbf{WH})}{D(\mathbf{X}|\mathbf{W}_{\text{init}}\mathbf{H}_{\text{init}})}\right) \tag{14}$$

### 6.2. Results

In a first experiment we aim at comparing the performance of the CD and the MU rules for CPU and GPGPU implementations. The MU rules are applied to the three particular cases of the $\beta$-divergence: the Euclidean norm (Figure 2-a), the KL divergence (Figure 2-b) and the IS divergence (Figure 2-c). The CD is applied to the Euclidean norm case and the KL divergence case. To our best knowledge there is no CD for IS divergence and deriving one is out of the scope of this paper. In general, GPGPU are always subsentially faster than CPU implementations. This difference is lower for NMF with Euclidean norm cost which was expected as they need a bit less matrix manipulations. Regarding the NMF with Euclidean norm cost, the CD allows to obtain a lower cost than the MU rules which confirms previous work in the domain. Applying the CD to the KL divergence was too slow (about 2 hours per epoch) compared to the MU rules so the results are not presented here.

In a second experiment we compare the behaviour of the mini-batch approaches to the baselines established earlier (see Figure 3). The greedy algorithms (GSG-MU and GSAG-MU) in general obtain performance comparable with the MU baseline. The dimension of $\mathbf{W}$ is rather small compared to the $\mathbf{H}$ so it seems that there is not much speed to gain by updating $\mathbf{W}$ only once per epoch. The behaviour of SAG-based algorithms is more spurious as can be seen of Figure 3-b and depends largely of the forgetting factor $\lambda$. In the KL divergence case $\lambda = 2$ seems already too large. The asymetric algorithms (ASG-MU and ASAG-MU) decrease the cost function faster than other MU rules and to a lower value. However, in the case of the Euclidean norm cost they are still outperformed by the CD.

[1] Source code is available be at `https://github.com/rserizel/minibatchNMF`

## 7. CONCLUSIONS

In this paper we proposed mini-batch stochastic approaches for the MU rules and compared their performance with CD when available for the divergence considered. The GPGPU was used to take advantage of the particular mini-batch structure and to efficiently deal with the matrices manipulations. In every case, the GPGPU allowed reducing the computation time. When considering the Euclidean norm cost, the proposed ASG-MU and ASAG-MU improved the standard MU performance and bring the MU rules closer to CD. When considering other cost functions (the KL divergence and the IS divergence), the CD is inefficient or simply has not been derived yet and MU is the most credible approach to perform NMF. The proposed approaches then provide significant reduction in computation time and obtain lower cost than standard NMF.

## 8. REFERENCES


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

[2] H. Lee and S. Choi, "Group nonnegative matrix factorization for eeg classification," in *Proc. of AISTATS*, 2009, pp. 320–327.

[3] F. Miwakeichi, E. Martınez-Montes, P. A. Valdés-Sosa, N. Nishiyama, H. Mizuhara, and Y. Yamaguchi, "Decomposing EEG data into space–time–frequency components using parallel factor analysis," *NeuroImage*, vol. 22, no. 3, pp. 1035–1045, 2004.

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

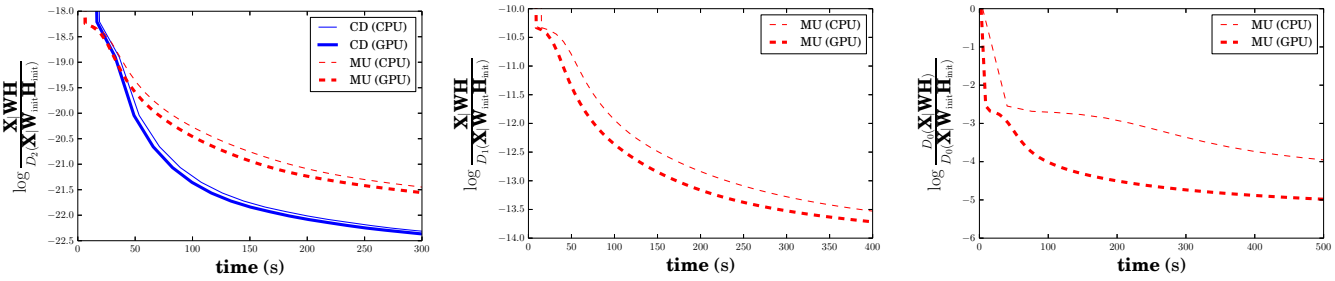
[5] A. Ozerov, E. Vincent, and F. Bimbot, "A general flexible framework for the handling of prior information in audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1118–1133, 2012.

[6] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, 2007.

[7] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *Proc. of WASPAA*. IEEE, 2003, pp. 177–180.

[8] V. Bisot, R. Serizel, S. Essid, and G. Richard, "Acoustic scene classification with matrix factorisation for unsupervised feature learning," in *Proc. of ICASSP*, 2016.

[9] A. Hurmalainen, R. Saeidi, and T. Virtanen, "Similarity induced group sparsity for non-negative matrix factorisation," in *Proc. of ICASSP*, 2015, pp. 4425–4429.

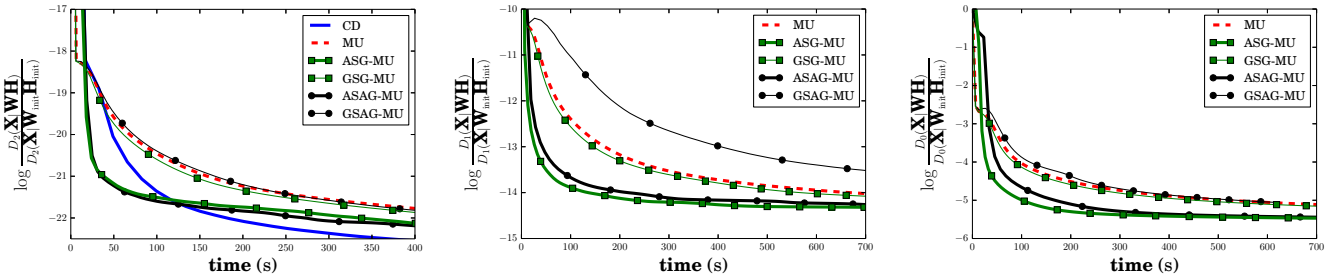



(a) CD and MU rules with Euclidean distance

(b) MU rules with KL divergence

(c) MU rules IS divergence

**Fig. 2**. Cost function evolution for CPU and GPU implementation




(a) Euclidean distance

(b) KL divergence

(c) IS divergence

**Fig. 3**. Comparison of the different mini-batch algorithms (GPU implementation)


[10] R. Serizel, S. Essid, and G. Richard, "Group nonnegative matrix factorisation with speaker and session variability compensation for speaker identification," in *Proc. of ICASSP*, 2016.

[11] A. Cichocki and A.-H. Phan, "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 92, no. 3, pp. 708–721, 2009.

[12] C.-J. Hsieh and I. S. Dhillon, "Fast coordinate descent methods with variable selection for non-negative matrix factorization," in *Proc. of the 17th SIGKDD*, 2011.

[13] U. Şimşekli, H. Koptagel, H. Güldaş, A. Taylan Cemgil, F. Öztoprak, and Ş. İlker Birbil, "Parallel Stochastic Gradient Markov Chain Monte Carlo for Matrix Factorisation Models," *ArXiv e-prints*, June 2015.

[14] Arthur Mensch, Julien Mairal, Bertrand Thirion, and Gaël Varoquaux, "Dictionary Learning for Massive Matrix Factorization," in *Proc. of ICML*, 2016.

[15] L. Bottou, "Online learning and stochastic approximations," *On-line learning in neural networks*, vol. 17, no. 9, pp. 142.

[16] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, pp. 79–86, 1951.

[17] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 23, no. 1, pp. 67–72, 1975.

[18] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. of NIPS*, 2000, pp. 556–562.

[19] A. Cichocki and S.-I. Amari, "Families of alpha-beta-and gamma-divergences: Flexible and robust measures of similarities," *Entropy*, vol. 12, no. 6, pp. 1532–1568, 2010.

[20] C. Févotte and J. Idier, "Algorithms for nonnegative matrix factorization with the $\beta$-divergence," *Neural Computation*, vol. 23, no. 9, pp. 2421–2456, 2011.

[21] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing Finite Sums with the Stochastic Average Gradient," *ArXiv e-prints*, Sept. 2013.

[22] N. Gillis, "The why and how of nonnegative matrix factorization," in *Regularization, Optimization, Kernels, and Support Vector Machines*, M. Signoretto J.A.K. Suykens and A. Argyriou, Eds., Machine Learning and Pattern Recognition Series, pp. 257 – 291. Chapman & Hall/CRC, 2014.

[23] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, et al., "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[24] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *Proc. of BigLearn, NIPS Workshop*, 2011.

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, C. S. Corrado, et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.

[26] G. Gravier, J.-F. Bonastre, E. Geoffrois, S. Galliano, K. Mc Tait, and K. Choukri, "ESTER, une campagne d'évaluation des systemes d'indexation automatique d'émissions radiophoniques en français.," in *Proc. of Journées d'Etude sur la Parole*, 2004.

[27] J. C. Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.