Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. 176

Block-Iterative Algorithms for Non-Negative Matrix Approximation

Suvrit Sra

23 Sep 2008

MPI für biologische Kybernetik, AGBS

Block-Iterative Algorithms for Non-Negative Matrix Approximation

Suvrit Sra

Abstract. In this report we present new algorithms for non-negative matrix approximation (NMA), commonly known as the NMF problem. Our methods improve upon the well-known methods of Lee & Seung [19] for both the Frobenius norm as well the Kullback-Leibler divergence versions of the problem. For the latter problem, our results are especially interesting because it seems to have witnessed much lesser algorithmic progress as compared to the Frobenius norm NMA problem. Our algorithms are based on a particular **block-iterative** acceleration technique for EM, which preserves the multiplicative nature of the updates and also ensures monotonicity. Furthermore, our algorithms also naturally apply to the Bregman-divergence NMA algorithms of Dhillon and Sra [8]. Experimentally, we show that our algorithms outperform the traditional Lee/Seung approach most of the time.

---

1 Introduction

Non-negative matrix factorization (NMA), which was introduced under the name *positive matrix factorization* by [28], and later popularized as *non-negative matrix factorization* or NMA by [20], is a popular matrix decomposition technique that is now increasingly employed for data analysis–see [5, 26, 36] and references therein. In its most common incarnation, given a non-negative input matrix A, NMA seeks to minimize

$$\mathscr{F}(\mathsf{A};\hat{\mathsf{A}}) = \|\mathsf{A}-\hat{\mathsf{A}}\|_F^2, \tag{1.1}$$

where $\hat{\mathsf{A}} = \mathsf{WH}$ and $\mathsf{W},\mathsf{H} \geq 0$. Several algorithms exist for this Frobenius norm version of NMA and its variations—see for e.g., [3, 17, 19, 23, 36]. The Kullback-Leibler (KL) divergence NMA problem that seeks to minimize

$$\mathscr{F}(\mathsf{A};\hat{\mathsf{A}}) = \sum\nolimits_{ij} a_{ij}\log(a_{ij}/\hat{a}_{ij}) - a_{ij} + \hat{a}_{ij}, \tag{1.2}$$

where $\hat{\mathsf{A}} = \mathsf{WH}$ and $\mathsf{W},\mathsf{H} \geq 0$, has received much lesser attention. Both (1.1) and (1.2) are special cases of the more general Bregman-divergence NMA problem [8]

$$\mathscr{F}(\mathsf{A};\hat{\mathsf{A}}) = \sum\nolimits_{ij} D_\phi(a_{ij};\hat{a}_{ij}), \tag{1.3}$$

where $\hat{\mathsf{A}}$ is as usual and we define the *Bregman-divergence*

$$\mathscr{F}(x;y) = \phi(x) - \phi(y) - \phi'(y)(x-y), \tag{1.4}$$

for a given strictly convex function ϕ[1]. With $\phi(x) = \frac{1}{2}x^2$ one obtains (1.1) from (1.3), and with $\phi(x) = x\log x$ one obtains (1.2). Note that since D_ϕ is asymmetric we can equally consider the NMA problem that seeks to minimize

$$D_\phi(\hat{\mathsf{A}};\mathsf{A}) = \sum\nolimits_{ij} D_\phi(\hat{a}_{ij};a_{ij}). \tag{1.5}$$

In fact [8] also studied this version of the problem.

In this paper, we derive new algorithms for (1.1)–(1.3) and (1.5). Our approach uncovers several important connections to results related to the EM algorithm that not only cast more light on the genesis of the well-known Lee & Seung algorithms, but also open the door to many potential algorithmic

[1] Actually ϕ must satisfy certain other technical conditions [2, 4], but for our purposes strict convexity is enough.

improvements for NMA. More specifically, we develop block-iterative algorithms for NMA by building on techniques used for accelerating the EM algorithm. We follow the same approach as Lee&Seung [19] by extending the vector based EM methods to NMA via alternating descent.

Before we move onto other details, we point out that our methods can be extended to handle the important *regularized* version of the problem, namely,

$$\underset{\mathsf{W},\mathsf{H}\geq 0}{\text{minimize}} \quad \mathscr{F}(\mathsf{A};\mathsf{WH}) + \lambda P(\mathsf{W}) + \mu Q(\mathsf{H}), \tag{1.6}$$

where $\lambda, \mu > 0$, and $P(\mathsf{W})$, $Q(\mathsf{H})$ are appropriate (convex, differentiable) regularization functions. There are, however, some additional technical issues that one must handle before these functions can be incorporated and we will discuss these in §3.3.

1.1 Related Work

Exact NMA, i.e., where $\mathsf{A} = \mathsf{WH}$ can be exactly computed, was studied in linear algebra in the 70s [24, 25]. Later Paatero and his colleagues studied NMA under the name of Positive Matrix Factorization, developing several complicated albeit non-scalable algorithms [28, 29]. In the field of data mining and machine learning, the NMA problem became popular after Lee&Seung described simple alternating descent algorithms for it in [19]. Since then interest in this problem has literally exploded; we refer the reader to the articles [3, 22, 36] for an extensive list of valuable references including generalizations to tensors, multiple-factors or differing constraints on the factors.

Most of the algorithmic progress on NMA has concentrated on (1.1) with the exception of [6, 8, 14]. In this paper we develop new algorithms that apply to not only (1.1) but also to (1.2), (1.3), and (1.5). Additionally, subject to certain convexity assumptions our techniques can also be extended to other loss functions beyond just Bregman divergences. We limit our discussion to Bregman divergences because they form an increasingly important class of distortion measures [1, 8, 38].

Algorithmically, the most popular approach to NMA is based on alternating descent [19, 22, 36], though some researchers have also considered alternating minimization [3, 17] for (1.1). If we trace the origins of the well-known LS algorithms, we may view them as matrix level generalizations of the successful EMML algorithm of [35]. This connection becomes particularly important because the EMML algorithm and its derivatives have been the subject of intense study in the field of medical imaging [31, 32], whereby one could potentially exploit some of the developments for EMML. Indeed, we exploit some ideas from advances made for EMML generalizing them to the NMA problem in a manner analogous to Lee/Seung [19]. Our generalization goes two steps further because we not only handle more general objective functions, but we also permit the incorporation of regularization terms.

2 Background

To avoid clutter, we concentrate on the two factor version of NMA where $\hat{\mathsf{A}} = \mathsf{WH}$, i.e., we aim to compute $\mathsf{W}, \mathsf{H} \geq 0$ such that $\mathscr{F}(\mathsf{A};\hat{\mathsf{A}}) = \mathscr{F}(\mathsf{A};\mathsf{WH})$ is minimized. Owing to the product $\boldsymbol{BC}$ the resulting problem is almost never convex, whereby it is unrealistic to expect to find a globally optimal solution; this intuition is further strengthened by the recent formal proof of the NP-Hardness of exact NMA [37]. Many times, for instance in (1.1), (1.2), and (1.5), the objective function is individually convex in W and H, whereby one can invoke an alternating minimization procedure that fixes W and solves (optimally) for H and vice-versa.

However, at this stage two difficulties arise. First, it could be that the objective function is *not* even individually convex in W or H, for e.g., many times in (1.3). Second, performing an exact minimization at each alternating step can be too expensive for solving the overall problem (even when the objective function is individually convex), especially when the matrices become large. Therefore, regardless of the convexity of the objective function $\mathscr{F}(\mathsf{A};\mathsf{WH})$ we would like to have a procedure that performs alternating descent (one could try to directly descend, but alternating descent is simpler). The general algorithmic scheme is summarized in Figure 1.

In this paper we derive algorithms following the general outline of Figure 1. The Lee & Seung algorithms follow this outline too, and so do almost all other NMA algorithms. Assuming that our distortion function $\mathscr{F}$ is separable, i.e.,

$$\mathscr{F}(\mathsf{A};\mathsf{WH}) = \sum\nolimits_j F(\boldsymbol{a}_j;\mathsf{W}\boldsymbol{h}_j),$$

1. Initialize W^0 and/or H^0; set $t \leftarrow 0$.
2. Fix W^t and find H^{t+1} such that

$$\mathscr{F}(\mathsf{W}^t, \mathsf{H}^{t+1}) \leq \mathscr{F}(\mathsf{W}^t, \mathsf{H}^t),$$

3. Fix H^{t+1} and find W^{t+1} such that

$$\mathscr{F}(\mathsf{W}^{t+1}, \mathsf{H}^{t+1}) \leq \mathscr{F}(\mathsf{W}^t, \mathsf{H}^{t+1}),$$

4. $t \leftarrow t+1$: repeat steps 2 & 3 to convergence

Figure 1: Alternating Descent Scheme

where F denotes the column-wise distortion, then we can just focus on solving the core problem

$$\min_{\boldsymbol{h} \geq 0} \; F(\boldsymbol{a}; \mathsf{W}\boldsymbol{h}), \tag{2.1}$$

for an arbitrary column $\boldsymbol{h}$ of matrix H, and corresponding column $\boldsymbol{a}$ of A (by working with rows of W and A we can treat the alternating step over W similarly; we skip details for brevity).

If we solve (2.1) "exactly" over all columns $\boldsymbol{h}$, then the resulting solution H^{t+1} clearly satisfies Step 3 of the above scheme (similarly for W^{t+1}). For the Frobenius norm NMA problem, such an approach is followed by [17], who derived an alternating non-negatively constrained least-squares algorithm (where Steps 2 and 3 become non-negative least squares steps).

However, solving (2.1) exactly can be computationally inefficient for general objective functions, especially because one has to alternate several times. Therefore, one could just "descend" on (2.1) so that descent required by Step 3 of the above algorithmic scheme is still satisfied. The Lee & Seung algorithms or the methods discussed in [8, 22] all produce descent and do not solve the subproblems exactly. Therefore, we can name the overall strategy *alternating descent*. When $\mathscr{F}$ is not convex even on fixing W (or H) the descent strategy remains still applicable, whereas exact minimization can fail due to non-convexity.

The sub-problem (2.1) itself is very important in numerous contexts and applications, for e.g., with $\mathscr{F} = \mathsf{KL}$, (2.1) is equivalent to maximizing Poisson Likelihood, which is crucial to applications in tomography [31, 32, 35]. In the NMA setting one can perform just one (or a few) iterations of EM or some other *guaranteed* descent method and still achieve progress. Indeed, the Lee & Seung algorithms follow this very descent idea, alternating it over W and H while performing just *one* iteration of the EMML [35] or ISRA algorithm [7] to ensure descent. This genesis of NMA algorithms provides us with valuable insight that we exploit below to develop new algorithms.

2.1 Auxiliary functions

The typical approach to obtaining descent algorithms for (2.1) is to first construct an *auxiliary function* $G(\boldsymbol{h}; \tilde{\boldsymbol{h}})$ that satisfies

1. $G(\boldsymbol{h}; \boldsymbol{h}) = F(\boldsymbol{a}; \mathsf{W}\boldsymbol{h})$ for all $\boldsymbol{h} \geq 0$
2. $G(\boldsymbol{h}; \tilde{\boldsymbol{h}}) \geq F(\boldsymbol{a}; \mathsf{W}\boldsymbol{h})$ for all $\boldsymbol{h}$, $\tilde{\boldsymbol{h}} \geq 0$.

The function G is chosen so that it *decouples* the variables $\boldsymbol{h}$ that occur as a linear combination $\mathsf{W}\boldsymbol{h}$. If F is a convex function of $\boldsymbol{h}$, then it is easy to construct (see [8, 30] for details) an auxiliary function G that is easier to optimize. For example, the famous EM algorithm is essentially built around exploiting the convexity of the $-\log x$ function for obtaining an auxiliary function in the E-Step and then optimizing it in the M-step. We remark that one need not always explicitly construct such auxiliary functions. For example, in [8] the authors derive procedures for (1.3) that are not based on auxiliary functions.

Monotonicity. The minimization (3.1) combined with the definition of auxiliary functions guarantees monotonic descent owing to the easily verified *sandwiching* inequalities:

$$\begin{aligned} F(\boldsymbol{a}; \mathsf{W}\boldsymbol{h}^{t+1}) &= G(\boldsymbol{h}^{t+1}, \boldsymbol{h}^{t+1}) \leq G(\boldsymbol{h}^{t+1}, \boldsymbol{h}^t) \\ &\leq G(\boldsymbol{h}^t, \boldsymbol{h}^t) = F(\boldsymbol{a}; \mathsf{W}\boldsymbol{h}^t). \end{aligned}$$

Constructing auxiliary functions. There can be several ways of choosing auxiliary functions. The authors of [8] used *convexity* of F

$$\begin{aligned} F(a_i; \sum\nolimits_j w_{ij} h_j) &\leq \sum\nolimits_j \lambda_j^i F(a_i; w_{ij} h_j / \lambda_j^i) \\ &= G(\boldsymbol{h}, \tilde{\boldsymbol{h}}), \text{ where } \lambda_j^i \geq 0, \sum\nolimits_j \lambda_j^i = 1, \end{aligned}$$

for e.g., with $\lambda_j^i = (w_{ij}\tilde{h}_j)/\sum_l w_{il}\tilde{h}_l$, to obtain auxiliary functions for NMA. This method leads to algorithms with *multiplicative updates*. Several other ways of constructing auxiliary functions are possible; we refer the reader to [18, 21] for details.

2.2 Block-iterative algorithms

We now present some background about some block-iterative algorithms, specifically as a means to understand how we can obtain new algorithms for NMA. Experience with EM has shown that methods based on auxiliary functions usually converge rather slowly. To overcome this hurdle a vast amount of effort has been invested in trying to speed up EM. One of the simplest (and quite successful) ideas here is to divide W into blocks and build an auxiliary function for each block. With this procedure one can still make progress over each block without increasing computational costs exorbitantly. In fact for minimizing the KL-divergence such a *block-iterative* method was indeed popularized under the name ordered subsets EM or OSEM by [16], though the idea of doing such a block-iteration itself can be traced further back [11]. In this paper we extend the block-iterative idea in three main directions:

1. Block-iteration for NMA—originally block-iteration was devised for vector problems; we extend it via alternating descent to matrices(§§ 3.1, 3.2)
2. Extension to general objective functions such as Bregman-divergences (Section 3)
3. Extension to NMA problems *without* using the concept of auxiliary functions (Section 3.2).

3 Algorithms and Analysis

In this Section we derive new algorithms for solving (1.3) and (1.5). We illustrate our methods by essentially showing descent procedures for solving the associated versions of the subproblem (2.1).

3.1 Methods based on auxiliary functions

Formally, without loss of generality let the rows (indices) of W be partitioned into p disjoint blocks (or subsets) $S_1, \ldots, S_p$ such that $\cup_i S_i = \{1, \ldots, m\}$, where $\boldsymbol{a} \in \mathbb{R}_+^m$. We process the data by going through these subsets one by one, indexing the current subset of interest by s. Let t denote the index for one complete pass through all the subsets. Since we assumed separability of $\mathscr{F}$ over both rows and columns, we can write

$$F(\boldsymbol{a}; \mathsf{W}\boldsymbol{h}) = \sum_{s=1}^{p} \sum_{i \in S_s} F(a_i; [\mathsf{W}\boldsymbol{h}]_i).$$

Assuming F is convex, we can easily generate auxiliary functions for each block of rows S_s separately. For simplicity, we use the same form of auxiliary functions for each block, though this is not a necessity. Let $G_s(\boldsymbol{h}, \tilde{\boldsymbol{h}})$ denote the auxiliary function for block s, then we have the updates

$$\boldsymbol{h}^{(k,s+1)} = \operatorname*{argmin}_{\boldsymbol{h} \geq 0} G_s(\boldsymbol{h}, \boldsymbol{h}^{(k,s)}), \tag{3.1}$$

where $\boldsymbol{h}^{(1,1)} = \boldsymbol{h}^t$, and $\boldsymbol{h}^{t+1} = \boldsymbol{h}^{(k+\delta,p+1)}$, i.e., the value obtained after cycling through all the blocks δ times. It is easy to see that going through all the blocks essentially involves the same number of operations as going through all the rows at the same time (which is tantamount to using $p = 1$). The Lee & Seung-type NMA algorithms go through all the rows *just once* ($\delta = 1$) and return the resulting $\boldsymbol{h}^{t+1}$. We use the index k in (3.1) to highlight the fact that we permit our algorithm to perform a few iterations ($\delta \in \{1, 4\}$) of (3.1) before obtaining $\boldsymbol{h}^{t+1}$. In practice, this choice can lead to better objective function values.

We are now ready to look at important specific instances of the block-iterative scheme.

Example 1 (Block-KLNMA)**.** *The standard auxiliary function based update for the KLNMA problem (1.2) is [19]*

$$h_j^{t+1} = h_j^t \frac{1}{\sum_i w_{ij}} \sum_i \frac{a_i w_{ij}}{[\mathsf{W}\boldsymbol{h}^t]_i}.$$

Using an appropriate auxiliary function we can replace this with a block-iterative update, where for each block $s = 1, \ldots, p$, we have

$$h_j^{(k,s+1)} = h_j^{(k,s)} \frac{1}{\sum_{i \in S_s} w_{ij}} \sum_{i \in S_s} \frac{a_i w_{ij}}{[\mathsf{W}\boldsymbol{h}^{(k,s)}]_i}. \tag{3.2}$$

We set $\boldsymbol{h}^{(1,1)} = \boldsymbol{h}^t$, $\boldsymbol{h}^{(k+1,1)} = \boldsymbol{h}^{(k+1,p+1)}$, and $\boldsymbol{h}^{t+1} = \boldsymbol{h}^{(k+\delta,p+1)}$ where $\delta \in \{1, 4\}$. Similarly we can perform blocked updates for W by partitioning the columns of the matrix H into subsets. This alternation between W and H, as well as performing just a few iterations of (3.2) distinguish our approach from the vector version of the block-iterative updates of [16]. **Note:** An immediate consequence of our BI-KLNMA algorithm is a related derivation of a blocked algorithm for Probabilistic Latent Semantic Indexing (PLSI) [15] based on the connections between PLSI and KLNMA [10, 12].

Lemma 1 (Monotonicity)**.** *To establish monotonicity of the algorithm based on* (3.2) *it suffices to show that*

$$\sum_{i \in S_s} \left(a_i - [\mathsf{W}\boldsymbol{h}^{(k,s+1)}]_i\right) \log \frac{[\mathsf{W}\boldsymbol{h}^{(k,s+1)}]_i}{[\mathsf{W}\boldsymbol{h}^{(k,s)}]_i} \geq 0. \tag{3.3}$$

Proof. Inequality (3.3) measures the decrease in the objective function restricted to subset S_s, and adding over all subsets we obtain the net decrease, hence overall monotonicity. This inequality follows easily by two applications of the log-sum inequality in combination with the update (3.2), and is omitted for brevity.

Example 2 (Block-LSNMA)**.** *The standard update for the Frobenius norm NMA problem (1.1) is [19]*

$$h_j^{t+1} = h_j^t \frac{\sum_i w_{ij} a_i}{\sum_i w_{ij} [\mathsf{W}\boldsymbol{h}^t]_i}.$$

The corresponding block-iterative version is

$$h_j^{(k,s+1)} = h_j^{(k,s)} \frac{\sum_{i \in S_s} w_{ij} a_i}{\sum_{i \in S_s} w_{ij} [\mathsf{W}\boldsymbol{h}^{(k,s)}]_i}, \tag{3.4}$$

where we set $\boldsymbol{h}^{(1,1)}$ and $\boldsymbol{h}^{t+1}$ as in Example 1.

Monotonicity of (3.4) follows by an easy generalization of proof of unblocked version [19] combined with adding across all subsets.

Example 3 (Block-BregNMA)**.** *The Bregman-divergence NMA algorithms of [8] have the following update*

$$\phi'(h_j^{t+1}) = \phi'(h_j^t) \frac{\sum_i w_{ij} \phi'(a_i)}{\sum_i w_{ij} \phi'([\mathsf{W}\boldsymbol{h}^t]_i)}.$$

The corresponding block-iterative version is

$$\phi'(h_j^{(k,s+1)}) = \phi'(h_j^{(k,s)}) \frac{\sum_{i \in S_s} w_{ij} \phi'(a_i)}{\sum_{i \in S_s} w_{ij} \phi'([\mathsf{W}\boldsymbol{h}^{(k,s)}]_i)}. \tag{3.5}$$

Note that these updates are of the form $\phi'(h_j^{t+1} = \phi'(h_j^t)\eta_j$, and one must compute the inverse of the function ϕ'. Fortunately, this can usually be done in closed form.

3.2 Non-auxiliary function methods

The authors of [8] presented a heuristic approach for minimizing (1.3), which was shown to empirically decrease the objective function monotonically. In our description of block-iterative algorithms above, the concept of auxiliary functions was used only for establishing monotonicity guarantees, not for applying the block-iterative scheme itself. Therefore, we easily obtain block-iterative versions of the algorithms of [8].

Example 4 (Block-BregNMA2)**.** *An update for* (1.5) *is [8]*

$$h_j^{t+1} = h_j^t \frac{\sum_i w_{ij}\phi''([\mathsf{W}\boldsymbol{h}^t]_i)a_i}{\sum_i w_{ij}\phi''([\mathsf{W}\boldsymbol{h}^t]_i)[\mathsf{W}\boldsymbol{h}^t]_i},$$

for which the corresponding block-iterative version is

$$h_j^{(k,s+1)} = h_j^{(k,s)} \frac{\sum_{i\in S_s} w_{ij}\phi''([\mathsf{W}\boldsymbol{h}^{(k,s)}]_i)a_i}{\sum_{i\in S_s} w_{ij}\phi''([\mathsf{W}\boldsymbol{h}^{(k,s)}]_i)[\mathsf{W}\boldsymbol{h}^{(k,s)}]_i}, \tag{3.6}$$

where we set $\boldsymbol{h}^{(1,1)}$ *and* $\boldsymbol{h}^{t+1}$ *as in Example 1.*

Pseudo-code: The overall generic Block-iterative NMA algorithm that uses the above examples as its subproblems is summarized below as Algorithm 1. The algorithm provides only the pseudo-code—in an actual implementation one has to be a little more careful to ensure a more efficient use of memory and computational resources.

Algorithm 1: Block-iterative NMA

Input: $\mathsf{A} \in \mathbb{R}_+^{m\times n}$: input matrix, K: rank of approximation
Output: $\mathsf{W} \in \mathbb{R}_+^{m\times K}$, and $\mathsf{H} \in \mathbb{R}_+^{K\times n}$ such that $\mathscr{F}(\mathsf{A};\mathsf{WH})$ is minimized.
{Initialization} ;
W^0: Initialize randomly or otherwise
$t \leftarrow 0$
inner_lim $\in \{1, 4\}$
repeat
  {Compute H^{t+1}};
  foreach *column* h^t *of* H^t **do**
    $h^{(1,1)} = h^t$;
    for $k = 1$ *to* innner_lim **do**
      for $s = 1$ *to* p **do**
        Compute $\boldsymbol{\eta}^{(k,s)}$ using Examples 1-5 depending on $\mathscr{F}$ and W^t;
        $h^{(k,s+1)} = h^{(k,s)} \odot \boldsymbol{\eta}^{(k,s)}$
      end
      $h^{(k+1,1)} = h^{(k,p+1)}$
    end
    $h^{t+1} = h^{(k,p+1)}$
  end
  {Compute W^{t+1}};
  {Method similar to that for H^{t+1} above};
  {Depends on $\mathscr{F}$ and H^{t+1}};
  $t \leftarrow t + 1$
until *Stopping criteria are met* ;

3.3 Handling Regularization

Assuming that the regularization function Q in (1.6) is separable, we have the following regularized subproblem

$$\min_{\boldsymbol{h}\geq 0} \; F(\boldsymbol{a};\mathsf{W}\boldsymbol{h}) + \mu Q(\boldsymbol{h}), \tag{3.7}$$

where $\mu > 0$ is the regularization parameter. A particularly simple method to tackle (3.7) is the so-called one-step late (OSL) heuristic [13] that was rediscovered in the context of NMA by [8]. Here, while deriving

updates such as (3.2)–(3.6) one replaces $\nabla[Q(\boldsymbol{h})]_j$ by $\nabla[Q(\boldsymbol{h}^t)]_j$ (i.e., the derivative at the previous step). For example, for (3.2) this leads to the update

$$h_j^{(k,s+1)} = \frac{h_j^{(k,s)}}{\sum_{i \in S_s} w_{ij} + \mu[\nabla Q(\boldsymbol{h}^{(k,s)})]_j} \sum_{i \in S_s} \frac{a_i w_{ij}}{[\mathsf{W}\boldsymbol{h}^{(k,s)}]_i}.$$

However, despite its simplicity this update can be invalid because it does not assure monotonicity like its unregularized counterpart (3.2). In general for Problem (3.7) if F is as in Section 3.1 and Q is the form $Q(\boldsymbol{h}) = \sum_j q([\boldsymbol{P}\boldsymbol{h}]_j)$, where q is also convex, then we can construct an auxiliary function $G + \bar{Q}$ as in §2.1 and use the update

$$\boldsymbol{h}^{(k,s+1)} = \operatorname*{argmin}_{\boldsymbol{h} \geq 0} G_s(\boldsymbol{h}, \boldsymbol{h}^{(k,s)}) + \mu \bar{Q}(\boldsymbol{h}; \boldsymbol{h}^{(k,s)}). \tag{3.8}$$

Further note that whenever (3.8) is not solvable in closed form, we can perform a few steps of Newton's method to obtain an approximate solution. The auxiliary functions G and $\bar{Q}$ are selected to decouple the variables $\boldsymbol{h}$, whereby invoking Newton's method or some other non-linear equation solver is feasible for (3.8). However, Newton's method or other typical convex optimization methods could become impractical for the original problem (3.7) due to the non-negativity constraints and subproblem sizes.

4 Experiments

We now show experiments that demonstrate how block-iteration helps to improve upon the standard algorithmic techniques. The methods that we compare are:

1. LSNMA–Frobenius norm algorithm of [19] against BI-LSNMA, our block-iterative version based on (3.4)
2. KLNMA–KL-Divergence algorithm of [19] against BI-KLNMA, our block-iterative version based on (3.2).
3. BREGNMA of [8] against BI-BREGNMA, our block-iterative version based on (3.6)

LSNMA, KLNMA, and BREGNMA were obtained from *http://www.cs.utexas.edu/ suvrit/work/progs/nnma.m*. Our algorithms were also implemented in MATLAB. To ensure fairness for LSNMA and KLNMA, we edited the BREGNMA source-code to make it more efficient. Our experiments show that within the same running time, the block-iterative versions of the NMA codes obtain better objective function values than the non-blocked versions.

For comparing the different algorithms we ran an extensive suite of experiments (reported in Figures 2–7 and Table 1) across a range of ranks (of matrices W and H) and degree of noise in the dataset. The basic methodology was: 1) for each given value of the rank K of the NMA, we generated random matrices W_0 and H_0 and formed the dataset A; 2) we added uniform random or Poisson noise $\boldsymbol{N}$ to A so that (after scaling) $\|(\mathsf{A} + \boldsymbol{N}) - \mathsf{A}\|_{\mathrm{F}} / \|\mathsf{A}\|_{\mathrm{F}}$ corresponds to a given level of distortion (2%, 5%, 10%, or 20%); 3) ran all algorithms until the relative change in iterates fell below 10^{-3} or a running time threshold was exceeded.

In all the experiments reported we started each algorithm with the *same* random initialization. The number of blocks used by our algorithms was roughly chosen based on the average number of rows falling in each subset. For e.g., when $m = 1000$, then usually between 2–10 subsets sufficed, and for $m = 3000$ selecting between 20–50 subsets led to better results. The iterative steps (3.2), (3.4), or (3.6) were run between 1 to 4 times.

4.1 Least-squares NMA

Our first results are on the basic LSNMA algorithm [19] that we compare to BI-LSNMA. Figure 2 compares the objective function values on a 1000×1000 matrix for varying ranks of decomposition across a range of distortion levels. We can see that BI-LSNMA performs better than the basic Lee & Seung algorithm, and these differences become more significant as the rank of the approximation is increased. Note that we ran both algorithms for the *same* amount of time, whereby even when our algorithm is only slightly better than the Lee& Seung algorithm, we do not lose anything.

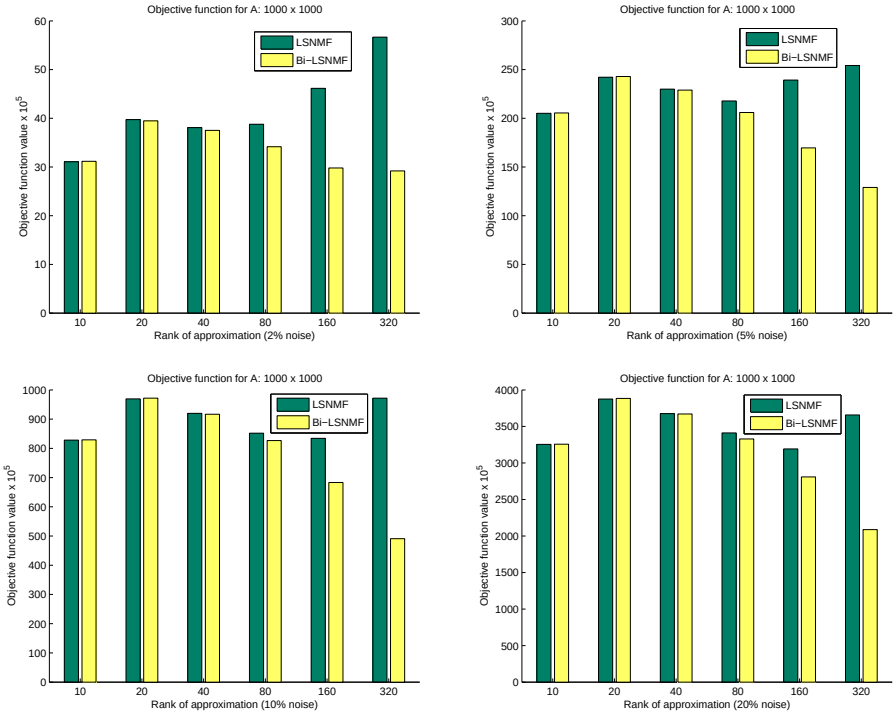

Figure 2: Comparisons of LSNMA to BI-LSNMA for a matrix of size 1000×1000 with varying noise-levels and rank of approximation. Both algorithms were run for ≈ 60 seconds each.

Next, in Figure 3 we show similar experiments, but this time on 3000×3000 matrices. As before both the rank and noise level in the data are varied across a range. We notice that once again, our BI-LSNMA algorithm outperforms the LSNMA algorithm, given the same amount of running time. If we run the algorithms longer, these differences remain almost the same, suggesting that our method is in fact reaching better local optima. This difference in optima is also related to the fact that we also perform the inner iteration (3.4) more than once.

4.2 KL-Divergence NMA

In Figure 4 proceeding row-wise, the figures show objective function values for different noise levels in the data as the rank of approximation is varied for a 1000×1000 input matrix. BI-KLNMA consistently outperforms KLNMA, and this improvement becomes substantial as the rank of approximation is increased. For example, in Figure 4 for rank 320 decompositions, BI-KLNMA yields objective function values up to almost 3 times smaller than the Lee/Seung KLNMA algorithm, in the same amount of running time.

Figure 5 shows the result of similar experiments on input matrices of size 3000×3000. Here too, our method consistently outperforms the KLNMA method. The absolute difference in objective function values is not as dramatic as in Figure 4, but nevertheless significant enough given that both algorithms were run for equally long.

4.3 Bregman-divergence NMA

We now illustrate our algorithm for Bregman-Divergence NMA by showing results for $D_\phi(\mathsf{A}; \mathsf{WH})$ corresponding to $\phi = -\log x$. With this choice of ϕ, the updates (3.6) simplify to become

$$h_j^{(k,s+1)} = h_j^{(k,s)} \frac{\sum_{i \in S_s} w_{ij} [\mathsf{W}\boldsymbol{h}^{(k,s)}]_i^{-2} a_i}{\sum_{i \in S_s} w_{ij} [\mathsf{W}\boldsymbol{h}^{(k,s)}]_i^{-1}}.$$

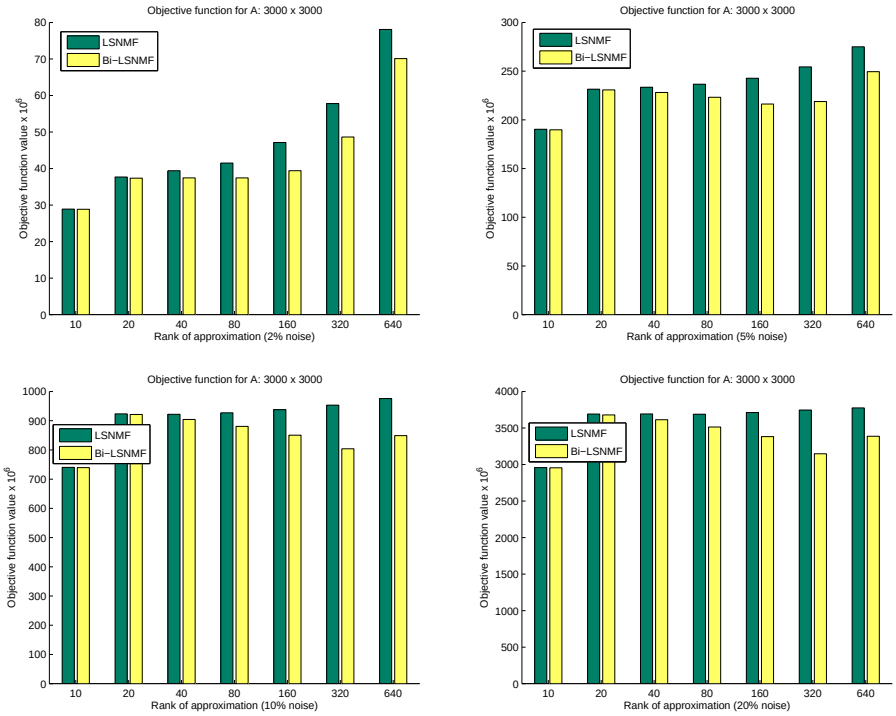

Figure 3: Comparisons of LSNMA to BI-LSNMA for a matrix of size 3000×3000 with varying noise-levels and rank of approximation. Both algorithms were run for ≈ 110 seconds.

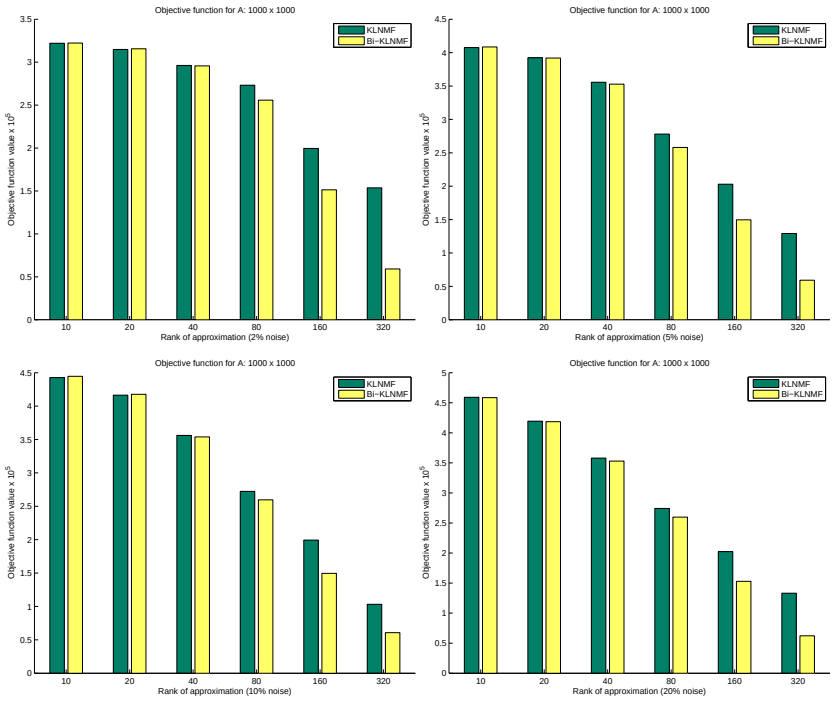

Figure 4: Comparisons of KLNMA to BI-KLNMA for a matrix of size 1000×1000 with varying noise-levels and rank of approximation. Both algorithms were run for ≈ 200 seconds.

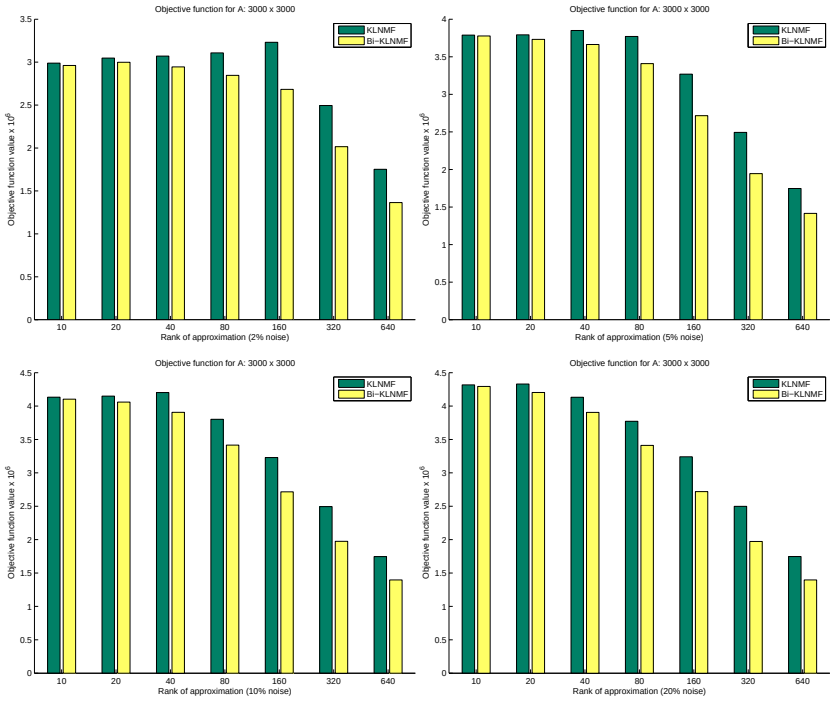

Figure 5: Comparisons of KLNMA to BI-KLNMA for a matrix of size 3000×3000 with varying noise-levels and rank of approximation. Both algorithms run for ≈ 200 seconds.

We also specialized the generic Bregman-NMA code of [8] to take advantage of similar simplifications, whereby the resulting update was

$$h_j^{t+1} = h_j^t \frac{\sum_i w_{ij} [\mathsf{W}\boldsymbol{h}^t]_i^{-2} a_i}{\sum_i w_{ij} [\mathsf{W}\boldsymbol{h}^t]_i^{-1}}.$$

These simplifications ensure fairness because they greatly speed up the implementation of [8].

Figure 6 shows experiments on input data matrices of size 1000×1000 with varying ranks and noise levels for the associated NMA. In the figure, for lower rank approximations, both the blocked and unblocked algorithms perform almost the same. The differences for higher rank approximations are overshadowed due to the bad scaling of the y-axis. To highlight the differences better, we summarize the objective function values for the rank ≥ 80 approximations in Table 1, which show that BI-BREGNMA actually yields objective function values twice as good as BREGNMA for these cases.

2%	5%	10%	20%
6.75 / **5.65**	23.49 / **20.80**	49.89 / **46.20**	88.84 / **84.42**
2.37 / **1.50**	9.12 / **6.36**	23.03 / **17.17**	48.73 / **39.27**
0.82 / **0.40**	3.11 / **1.53**	9.22 / **4.62**	24.11 / **13.09**

Table 1: Objective function values ($\times 10^3$) for ranks 80, 160, and 320 (row-wise) and noise-levels (column-wise) corresponding to Figure 6. The numbers in bold indicate the results of BI-BREGNMA. For rank-320 factorizations our BI-BREGNMA method yields values approximately twice as good as BREGNMA.

Figure 7 shows results for a 3000×3000 input matrix as before. Here we once again observe the results to be consistently better than BREGNMA.

4.4 Effect of Objective function

In addition to showing numerical results above comparing our algorithms to the unblocked NMA algorithms, we show results on an interesting example to highlight some of the potential variations that

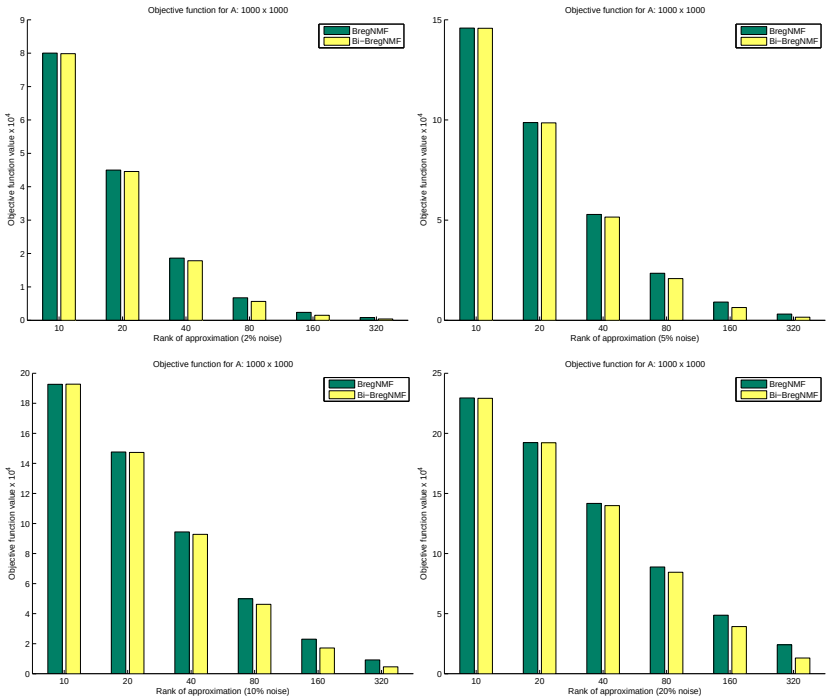

Figure 6: Comparisons of BREGNMA to BI-BREGNMA for a matrix of size 1000×1000 with varying noise-levels and ranks. Both algorithms were run for ≈ 60 seconds.

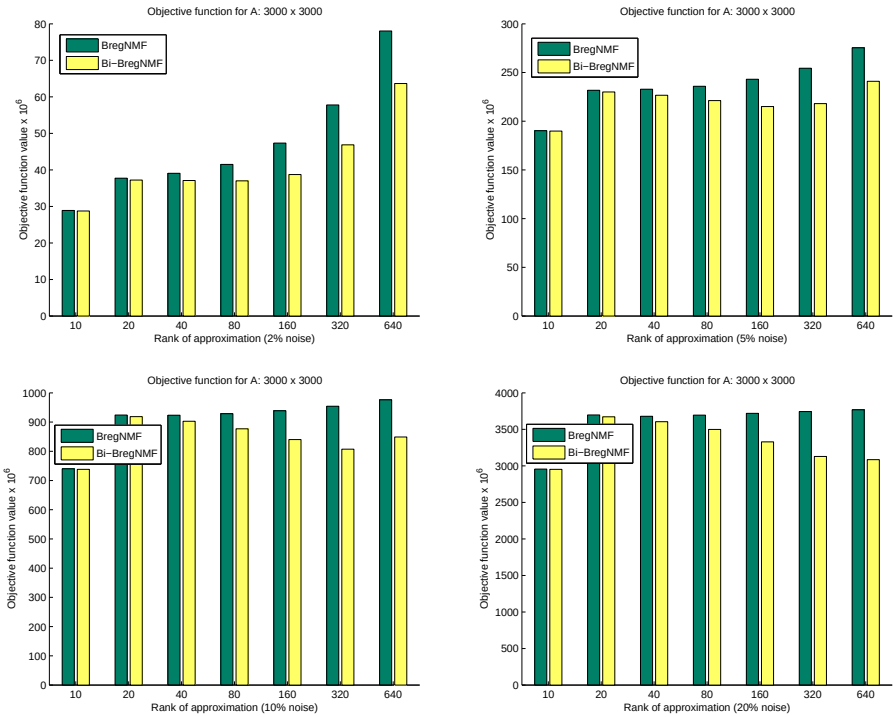

Figure 7: Comparisons of BREGNMA to BI-BREGNMA for a matrix of size 3000×3000 with varying noise-levels and rank of approximation. Both algorithms run for ≈ 130 seconds.

can arise by selecting different objective functions for NMA. The aim of these results is to provide some insight into assessing which objective function might be better for a particular NMA task. In general selecting which particular distortion measure for a given problem is a difficult question to answer, and several parameters must be taken into account. Answering this question completely remains an open problem. Our experiments provide some simple insights for a particular data set.

We consider the World-flags dataset [27] that consists of 258 color images of flags of different countries or international regions in the world. We preprocess this dataset to reduce the size of all the images to be 162×128 while making them grayscale at the same time. We also removed 14 duplicate flags from the dataset. After vectorizing all the images we obtained an input matrix A of size 20736×244 that is dense and non-negative. Some sample flags from this dataset are shown in Figure 8.

Figure 9 shows a sample of the results obtained by computing a rank-10 approximation to the flags matrix A. The results shown are essentially pick 3 columns out of the total 10 from the factor matrix W. The three algorithms compared were Bi-Lsnma, BI-Klnma, and Bi-Bregnma (with $\phi = 0.25x^4$). All algorithms were run for 120 seconds starting from the same random initialization.

We observe that both Bi-Lsnma and Bi-Bregnma yield "basis" images that are smoother than the sharper results of BI-Klnma. This sharpness suggests that the results yielded by BI-Klnma are somehow sparser than those yielded by the other algorithms. In fact, we observed that the vector ℓ_1-norm $\|\mathsf{WH}\|_1$ was the smallest for BI-Klnma. All the results, however, do indicate some of the expected structure in the basis-vectors, e.g., stripes or a blob in the middle, as most flags do have such patterns. Given these results, one can conclude that if the aim is to discover some latent structure in the flags dataset, then it might be better to use BI-Klnma, as opposed to the other objective functions.

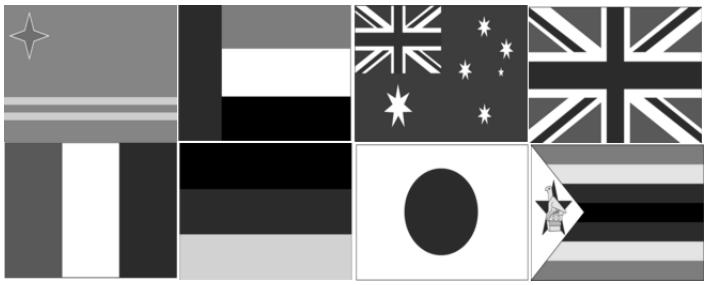

Figure 8: Samples flag images from the world-flags dataset

5 Extensions

The idea of block-iteration for NMA algorithms can be naturally extended to deal with the following generalized versions of the NMA problem.

1. **Multi-factor Problem.** Here we wish to minimize $\mathscr{F}(\mathsf{A}; \mathsf{W}_1\mathsf{W}_2 \ldots \mathsf{W}_T)$, where each $\mathsf{W} \geq 0$. Clearly, by cycling through all the factors and applying a block-iterative method to each subproblem our methods generalize trivially in this case.

2. **Convex and Semi-NMA.** These generalized variations of the NMA problem [9] that drop some non-negativity constraints can also benefit from block-iterative methods. As a first attempt, one can just take the algorithms of [9] and apply the block-iterative techniques to immediately obtain new algorithms.

3. **Non-negative tensor factorization.** NTF is generalization of the NMA idea to tensors, see for e.g., [33, 34, 39], and the block-iterative idea can be easily extended to the EM style algorithms for NTF. An interesting experiment would be to apply NTF to the World-flags data in color instead of grayscale, to see what affect does the color-distribution have on the "basis" flags recovered.

6 Conclusions and Future Work

In this paper we introduced block-iterative algorithms for solving the NMA problem. We showed the generic technique, and applied it to the Frobenius norm, KL-Divergence, and Bregman-divergence NMA

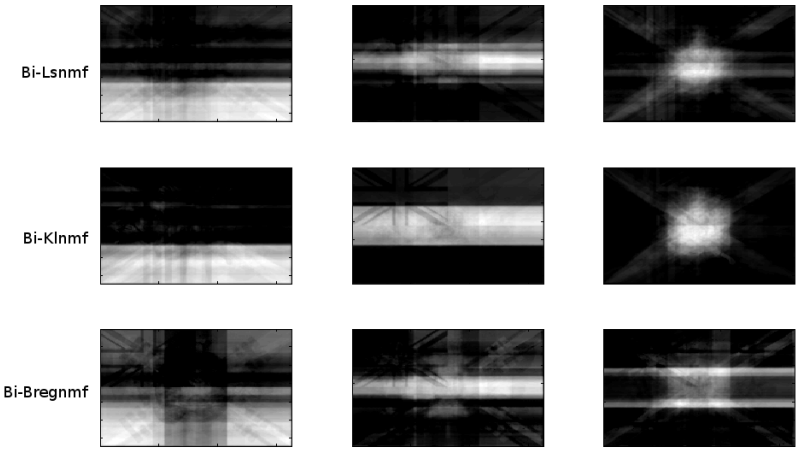

Figure 9: Some of the "basis" flags (columns of matrix W). The top-row shows 3 sample basis-flags obtained via the Bi-Lsnma algorithm, the middle-row shows corresponding flags via BI-Klnma, and the last row corresponds to Bi-Bregnma (run with $\phi = 0.25x^4$). We see that BI-Klnma returns the "sharpest" looking basis vectors; both Bi-Lsnma and Bi-Bregnma return smoother results, as was to be expected. Interestingly, the factorization WH returned by BI-Klnma had the smallest ℓ_1-norm amongst the three algorithms. All three algorithms were run for ≈ 120 seconds.

problems (of which the first two are special cases). Our use of block-iterative methods was motivated by the successful exploitation of EM techniques for NMA by Lee & Seung [19]. We performed extensive experiments to show the benefits obtained by using a block-iterative scheme, mainly showing that in the same running time, the block-iterative algorithms achieve lower objective function values that the traditional approaches. Furthermore, block-iterative methods are not much more complicated to implement.

We also listed 3 possible easy extensions of our block-iterative NMA algorithms. Implementing and experimenting with these extensions is a part of our ongoing work. On a more theoretical side, an important part of work related to this research is obtaining a better theoretical analysis of convergence of the various algorithms in addition to potential methods of convergence acceleration.

References

[1] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman Divergences. *JMLR*, 6(6):1705–1749, October 2005.

[2] H. H. Bauschke and J. M. Borwein. Legendre functions and the method of random Bregman projections. *Journal of Convex Analysis*, 4:27–67, 1997.

[3] M. Berry, M. Browne, A. Langville, P. Pauca, and R. J. Plemmons. Algorithms and Applications for Approximation Nonnegative Matrix Factorization. *Computational Statistics and Data Analysis*, 2006. Preprint.

[4] Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1997.

[5] M. Chu and R. Plemmons. Nonnegative matrix factorization and applications. *IMAGE, Bulletin of the International Linear Algebra Society*, 34:2–7, 2005.

[6] A. Cichocki, R. Zdunek, and S. Amari. Csiszar's Divergences for Non-Negative Matrix Factorization: Family of New Algorithms. In *6th International Conference on Independent Component Analysis and Blind Signal Separation*, volume Springer LNCS 3889, pages 32–39, Charleston SC, USA, 2006.

[7] M. E. Daube-Witherspoon and G. Muehllehner. An Iterative Image Space Reconstruction Algorthm Suitable for Volume ECT. *IEEE Tran. Medical Imaging*, 5(2):61–66, 1986.

[8] I. S. Dhillon and S. Sra. Generalized Nonnegative Matrix Approximations with Bregman Divergences. In *NIPS 18*, Vancouver, Canada, 2006.

[9] C. Ding, T. Li, and M. I. Jordan. Convex and Semi-Nonnegative Matrix Factorization. Technical Report LBNL TR # 60428, LBNL, 2006.

[10] C. Ding, T. Li, and W. Peng. On the Equivalence Between Nonnegative Matrix Factorization and Probabilistic Latent Semantic Indexing. *Computational Stat. and Data Analysis*, 52, 2008.

[11] J. Fessler. *Image reconstruction: Algorithms and Analysis*. Under Preparation, 2008.

[12] E. Gaussier and C. Goutte. Relation between PLSA and NMF and implications. In *ACM SIGIR conference on Research and development in information retrieval*, pages 601–602, 2005.

[13] P. J. Green. On Use of the EM for Penalized Likelihood Estimation. *J. Roy. Stat. Soc. Series B*, 52 (3):443–452, 1990.

[14] D. Guillamet, J. Vitrià, and B. Schiele. Introducing a weighted nonnegative matrix factorization for image classification. *Pattern Recognition Letters*, 24(14):2447–2454, 2003.

[15] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. ACM SIGIR*. ACM Press, August 1999.

[16] H. M. Hudson and R. S. Larkin. Accelerated image reconstruction using ordered subsets of projection data. *IEEE Tran. Medical Imaging*, 13(4):601–609, 1994.

[17] D. Kim, S. Sra, and I. S. Dhillon. Fast Newton-type methods for the least squares nonnegative matrix approximation problem. In *SIAM Data Mining*, 2007.

[18] K. Lange, D. R. Hunter, and I. Yang. Optimization Transfer using Surrogate Objective Functions. *J. Comput. and Graph. Stat.*, 9(1):1–20, 2000.

[19] D. D. Lee and H. S. Seung. Algorithms for Nonnegative Matrix Factorization. In *Neural Information Processing Systems*, pages 556–562, 2000.

[20] D. D. Lee and H. S. Seung. Learning The Parts of Objects by Nonnegative Matrix Factorization. *Nature*, 401:788–791, 1999.

[21] J. De Leeuw and G. Michailidis. Majorization methods in statistics. *Journal of Computational and Graphical Statistics*, 9:26–31, 2000.

[22] T. Li and C. Ding. The Relationships Among Various Nonnegative Matrix Factorization Methods for Clustering. In *Proc. IEEE Int. Conf. Data Mining*, pages 362–371, 2006.

[23] C. Lin. Projected Gradient Methods for Non-negative Matrix Factorization. Technical Report ISSTECH-95-013, National Taiwan University, 2005.

[24] T. L. Markham. Factorizations of completely positive matrices. *Proceedings of the Cambridge Philosophical Society*, 69:53–58, 1971.

[25] T. L. Markham. Factorizations of nonnegative matrices. *Proceedings of the American Mathematical Society*, 32(1):45–47, March 1972.

[26] MMDS. MMDS 2006. Workshop on Algorithms for Modern Massive Data Sets. URL: http://www.stanford.edu/group/mmds/, June 2006.

[27] S. Nowozin. The world-flags data. Email request to nowozin@tuebingen.mpg.de. Compiled from flag images from CIA Factbook.

[28] P. Paatero and U. Tapper. Positive Matrix Factorization: A Nonnegative Factor Model with Optimal Utilization of Error Estimates of Data Values. *Environmetrics*, 5(111–126), 1994.

[29] P. Paatero, U. Tapper, P. Aalto, and M. Kulmala. Matrix factorization methods for analysing diffusion battery data. *Journal of Aerosol Science*, 22(Supplement 1):S273–S276, 1991.

[30] A. R. De Pierro. A modified Expectation Maximization Algorithm for Penalized Likelihood Estimation in Emission Tomography. *IEEE Tran. Med. Imag.*, 14(1):132–137, 1995.

[31] A. J. Reader and H. Zaidi. Advances in PET Image Reconstruction. *PET Clinics*, 2(2):173–190, April 2007.

[32] R. M. Rewitt and S. Matej. Overview of methods for image reconstruction from projections in emission computed tomography. *Proc. IEEE*, 91(10):1588–1611, October 2003.

[33] A. Shashua and T. Hazan. Non-Negative Tensor Factorization with Applications to Statistics and Computer Vision. In *ICML*, 2005.

[34] A. Shashua, R. Zass, and T. Hazan. Multiway Clustering using Super-symmetric Nonnegative Tensor Factorization. In A. Leonardis, H. Bischof, and A. Prinz, editors, *ECCV*, pages 595–608. Springer, 2006.

[35] L. A. Shepp and Y. Vardi. Maximum likelihood reconstruction for emission tomography. *IEEE Tran. Medical Imaging*, 1:113–122, October 1982.

[36] S. Sra and I. S. Dhillon. Nonnegative Matrix Approximation: Algorithms and Applications. Technical Report TR-06-27, Univ. of Texas at Austin, 2006.

[37] S. Vavasis. On the complexity of nonnegative matrix factorization. arXiv, September 2007. arXiv:0708.4149v2.

[38] M. Warmuth and K. Azoury. Relative Loss Bounds for On-line Density Estimation with the Exponential Family of Distributions. *Machine Learning*, 43(3):211–246, 2001.

[39] M. Welling and M. Weber. Positive tensor factorization. *Pattern Recognition Letters*, 22:1255–1261, 2001.