

Efficient algorithms for regularized Poisson Non-negative Matrix Factorization

Nathanaël Perraudin¹, Adrien Teutrie², Cécile Hébert³, and Guillaume Obozinski¹

¹*Swiss Data Science Center, EPFL and ETH Zürich, Andreasstrasse 5, Zürich, 8050, Zürich, Switzerland*

²*Unité Matériaux et Transformations, UMR-CNRS 8207, Université de Lille, Cité scientifique, Bâtiment C6, Villeneuve d'Ascq, 59655, Nord, France*

³*Electron Spectrometry and Microscopy Laboratory, Institute of Physics, EPFL, Bâtiment PH, Station 3, Lausanne, 1015, Vaud, Switzerland*

Abstract

We consider the problem of regularized Poisson Non-negative Matrix Factorization (NMF) problem, encompassing various regularization terms such as Lipschitz and relatively smooth functions, alongside linear constraints. This problem holds significant relevance in numerous Machine Learning applications, particularly within the domain of physical linear unmixing problems. A notable challenge arises from the main loss term in the Poisson NMF problem being a KL divergence, which is non-Lipschitz, rendering traditional gradient descent-based approaches inefficient. In this contribution, we explore the utilization of Block Successive Upper Minimization (BSUM) to overcome this challenge. We build appropriate majorizing function for Lipschitz and relatively smooth functions, and show how to introduce linear constraints into the problem. This results in the development of two novel algorithms for regularized Poisson NMF. We conduct numerical simulations to showcase the effectiveness of our approach.

Disclaimer This document is a technical report and has not undergone peer review. The findings and conclusions presented herein are solely based on the authors' research and analysis. We apologize for any potential errors or shortcomings in the content.

1 Introduction

The problem of factorizing a matrix $\mathbf{Y} \approx \mathbf{WH}$ as Non Negative components $\mathbf{W} \geq 0, \mathbf{H} \geq 0$ is central in many Machine Learning (ML) applications [47, 10, 4]. The motivation for performing such a factorization is that $\mathbf{Y}$ is often associated with a probability distribution density of the form $P_{\mathbf{Y}}(\mathbf{W}, \mathbf{H}) = \tilde{P}_{\mathbf{Y}}(\mathbf{WH})$. Typically, the optimal decomposition is found by minimizing the negative log-likelihood of that distribution:

$$\underset{\mathbf{W} \geq 0, \mathbf{H} \geq 0}{\text{minimize}} \quad - \sum_{i,j} \log(P_{\mathbf{Y}}(\mathbf{W}, \mathbf{H})) \quad (1)$$

If $\mathbf{Y}$ is assumed to be perturbed with Normal noise, we obtain a Gaussian distribution, i.e. $P_{\mathbf{Y}}(\mathbf{W}, \mathbf{H}) \propto e^{-\|\mathbf{WH} - \mathbf{Y}\|^2}$, and we end up with the classic non-negative matrix factorization (NMF) problem [29, 30], where the quadratic function $\|\mathbf{WH} - \mathbf{Y}\|^2$ is minimized. For other distribution families, and in particular exponential families, a log term often appears. In particular, the Poisson negative log-likelihood model [29] leads to a loss of the form

$$\mathcal{L}_{\mathbf{Y}}(\mathbf{W}, \mathbf{H}) := -\langle \mathbf{Y}, \log(\mathbf{WH}) \rangle + \langle \mathbf{1}, \mathbf{WH} \rangle \propto - \sum_{i,j} \log(P_{\mathbf{Y}}(\mathbf{W}, \mathbf{H})), \quad (2)$$

where the inner product over matrices is the Frobenius inner product defined as $\langle \mathbf{A}, \mathbf{B} \rangle = \sum_{i,j} a_{ij} b_{ij}$.

Regularized Poisson Non Negative Matrix Factorisation In many problems (see, for example, [51, 50, 21, 52]), additional prior information about the matrices $\mathbf{W}, \mathbf{H}$ is known. For example, it might be known that the columns of $\mathbf{H}$ are smooth, or that the rows of $\mathbf{W}$ are sparse. One might also be interested in normalizing to unity the columns of $\mathbf{H}$ or the rows of $\mathbf{W}$ because they might quantify physical quantities for which normalization is necessary. For example, in the analysis of hyperspectral imaging data, the images $\mathbf{H}$ are assumed to be smooth and the components $\mathbf{W}$ are summing to the unity [51, 52]. Typically, this information can be encoded via an extra regularization term $R(\mathbf{W}, \mathbf{H})$ and/or additional constraints $\mathbf{W} \in \mathcal{C}_1, \mathbf{H} \in \mathcal{C}_2$. This leads to the general optimization problem we solve in this contribution:

$$\begin{aligned} & \underset{\mathbf{W}, \mathbf{H}}{\text{minimize}} \quad \mathcal{L}_{\mathbf{Y}}(\mathbf{W}, \mathbf{H}) + R_{\mathbf{W}}(\mathbf{W}) + R_{\mathbf{H}}(\mathbf{H}), \\ & \text{subject to} \quad \mathbf{W} \geq \epsilon, \mathbf{H} \geq \epsilon, \mathbf{e}_H^\top \mathbf{H} = \mathbf{1} \text{ or } \mathbf{W} \mathbf{e}_W = \mathbf{1}^\top \end{aligned} \quad (3)$$

where $\epsilon > 0$. The different colors emphasize the changes compared to the traditional problem of [29]. First, in violet, we slightly simplify the problem by imposing strict non-negativity. While, this is not strictly necessary¹, this assumption significantly simplify our analysis. We believe that handling $\epsilon = 0$ is an unnecessary complication, as the results with ϵ close to machine precision will be practically identical. Second, in light blue, we consider the case where the constraints are linear, specifically $\mathbf{e}_H^\top \mathbf{H} = \mathbf{1}$ or $\mathbf{W} \mathbf{e}_W = \mathbf{1}^\top$. In general, it is only meaningful to use one of the constraints, as it will fix the ratio between $\mathbf{W}$ and $\mathbf{H}$. We note that this includes the simplex constraint when $\mathbf{e} = \mathbf{1}$. Third, in brown, we consider regularizations $R_{\mathbf{W}}(\mathbf{W}) + R_{\mathbf{H}}(\mathbf{H})$ of the form:

$$r(\mathbf{x}) = s_L(\mathbf{x}) + s_R(\mathbf{x}) + \sum_{j=1}^n s_C(x_j), \quad (4)$$

where $\mathbf{x}$ is the vector of a row of $\mathbf{W}$ or a column of $\mathbf{H}$, i.e $R_{\mathbf{W}}(\mathbf{W}) = \sum_i r_H(\mathbf{w}_i)$ and $R_{\mathbf{H}}(\mathbf{H}) = \sum_j r_W(\mathbf{h}_j)$.

1. The term s_L is assumed to be σ_L gradient Lipschitz, i.e., there exists a σ_L such that

$$\|\nabla s_L(\mathbf{x}) - \nabla s_L(\mathbf{y})\|_2 \leq \sigma_L \|\mathbf{x} - \mathbf{y}\|_2 \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathcal{C}. \quad (5)$$

Alternatively, this condition could be rewritten as

$$s_L(\mathbf{y}) \leq s_L(\mathbf{x}) + \langle \nabla s_L(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\sigma_L}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathcal{C}.$$

In this contribution, we will consider in particular $s_L(\mathbf{x}) = \mathbf{x}^\top \Delta \mathbf{x}$, where Δ is the Laplacian operator, favoring smoothness in the columns of $\mathbf{H}$. In this case $\sigma_L = 2\lambda_{\max}(\Delta)$.

2. The term $s_R(\mathbf{x})$ is assumed to be σ_R relatively smooth with respect to $\kappa(\mathbf{x}) = -\mathbf{1}^\top \log(\mathbf{x})$. Relative smoothness is a generalization of Lipschitz smoothness, and is defined as follows [35, Definition 1.1]:

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \sigma_R \mathcal{B}_\kappa(\mathbf{y}, \mathbf{x}) \quad (6)$$

where $\mathcal{B}_\kappa$ is the Bregman divergence [3] associated with κ [35, equation 7]:

$$\mathcal{B}_\kappa(\mathbf{y}, \mathbf{x}) := \kappa(\mathbf{y}) - \kappa(\mathbf{x}) - \langle \nabla \kappa(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathcal{C}.$$

While Lipschitz functions can be upper bounded with quadratic functions, we observe from (6) that relative smoothness allows us to upper bound a function with the Bregman divergence of a function κ . This allows us to use a much wider range of functions to regularize our problem, and in particular non-gradient Lipschitz ones. As an example, let us consider $\kappa(\mathbf{x}) = -\mathbf{1}^\top \log(\mathbf{x})$, with its Bregman divergence:

$$\mathcal{B}_\kappa(\mathbf{y}, \mathbf{x}) = \sum_i \left(\frac{y_i}{x_i} - \log \left(\frac{y_i}{x_i} \right) - 1 \right)$$

One can observe that the objective function (2) is relatively smooth with respect to $\kappa(\mathbf{x}) = -\mathbf{1}^\top \log(\mathbf{x})$. This term could also be used to introduce soft constraints such as a log-barrier: $s_R(\mathbf{x}) = -\mathbf{1}^\top \log(\mathbf{x} - \epsilon \mathbf{1})$ for $\mathbf{x} > \epsilon$ and $+\infty$ otherwise.

¹The case $\epsilon = 0$ could be handled with an approach similar to [33].

3. Eventually, $s_C(x)$ is a smooth point-wise concave function (i.e. $-s_C$ is convex). A typical example of this regularisation could be $s_C(x) = \log(x + \alpha^{-1})$ which favors sparsity in the vector $\mathbf{x}$ without penalizing large values too heavily. Its slope starts at α for $x = 0$ and tends to 0 for $x \rightarrow \infty$.

Our approach can handle regularizations of the form $R(\mathbf{W}, \mathbf{H})$. However, for simplicity of notation, we restrict ourselves to separable regularizations.

The fidelity term $\mathcal{L}_Y(\mathbf{W}, \mathbf{H})$ is convex in $\mathbf{H}$ and in $\mathbf{W}$, but jointly non convex. We note that depending on the regularization and the constraints, multiple equivalent scaled solutions (stationary points) could exist, i.e., $\mathbf{W}\mathbf{H} = \mathbf{W}'\mathbf{H}'$ for $\mathbf{W}' = \alpha^{-1}\mathbf{W}$ and $\mathbf{H}' = \alpha\mathbf{H}$. However, this is generally not the case with additional constraints or regularizations.

Why is this problem challenging? In general, Poisson Non Negative Matrix factorization, i.e. minimizing $\mathcal{L}_Y(\mathbf{W}, \mathbf{H})$ is challenging because it is not gradient Lipschitz² despite being differentiable for $\mathbf{W}, \mathbf{H} > 0$. In practice, this implies that there exists no fixed learning rate that ensures convergence of gradient descent and that line search would have to be used. For that reason, solving the more general problem (3) is a difficult task, and to the best of our knowledge, there are no existing algorithms that can be directly applied to it. Although there exist many algorithms to solve the traditional Poisson NMF problem [16, 30, 29, 24, 12], none of them focuses on the regularized case (see the related work Section 2). Our main contribution is to fill this gap by providing multiple algorithms that minimize (3) for a wide range of regularizations $R(\mathbf{W}, \mathbf{H})$ described by (4) and some linear constraints.

Our approach A natural approach to minimize (3) is to optimize for each variable $\mathbf{W}, \mathbf{H}$ at a time. For example, Block Coordinate Descent (BCD) can be expressed as

$$\mathbf{W}^{t+1} \leftarrow \underset{\mathbf{W} \in \mathcal{C}_W}{\text{minimize}} \quad \mathcal{L}(\mathbf{W}, \mathbf{H}^t), \quad (7)$$

$$\mathbf{H}^{t+1} \leftarrow \underset{\mathbf{H} \in \mathcal{C}_H}{\text{minimize}} \quad \mathcal{L}(\mathbf{W}^{t+1}, \mathbf{H}). \quad (8)$$

This type of iterative scheme ensures that the loss does not increase between iterations and has been successfully used for the L2 case. However, in the Poisson case, there is no closed-form solution for problems (7) and (8), making this approach generally computationally expensive.

Fortunately, in practice, one does not need to find the global minima of (7) and (8) at each iteration. Instead, using a Block Successive Minimization (BSUM) algorithm [41], it is sufficient to minimize approximations of $\mathcal{L}$ which are locally tight upper bounds of $\mathcal{L}$. To use the BSUM efficiently, these approximation functions need to have three properties: (1) to satisfy the hypotheses of the BSUM Theorem [41, Theorem 2], (2) to be as tight as possible, and (3) to be easy to optimize, i.e. to lead to a closed-form solution for each subproblem.

Our contributions can be summarized as follows. We show how regularized Poisson NMF can be efficiently solved using BSUM. We derive tight upper bounds for multiple regularizers and compare our approach with traditional algorithms. We also propose a simple way to introduce linear constraints into the problem and suggest using line search to build even tighter upper bounds. Finally, we propose multiple algorithms for regularized Poisson NMF and conduct numerical simulations to demonstrate the effectiveness of our approach.

Outline of this contribution In Section 2, we provide a review of the literature. In Section 3, we clarify the notation and provide the necessary definitions for the BSUM Theorem [41, Theorem 2] that will be used for the convergence of our algorithm. In Section 4, we develop convenient approximations of the objective and regularization functions leading to sub-problems with closed-form solutions. In Section 4.3, we explore how to modify the optimization scheme to introduce generalized simplex constraints. In Section 5, we present our algorithms. Section 6 provides numerical applications of our algorithm, and Section 7 concludes this contribution.

²The function $\mathcal{L}_Y$ is, according to the definition, gradient Lipschitz because of the constraint $\mathbf{W}, \mathbf{H} \geq \epsilon$. However, in practice, the constant ϵ is chosen to be so small that its actual Lipschitz constant is too large to be useful.

2 Related work

Applications of Poisson Distribution likelihood Maximization The maximization of likelihood in a Poisson distribution finds relevance in various applications, prompting the resolution of the problem outlined in Equation (3).

Many such applications arise in the domain of physical constrained linear unmixing problems [21]. Some noteworthy instances encompass: 1. Scanning transmission electron microscopy (STEM) [52, 45, 5, 19], 2. Hyperspectral Raman and optical imaging [22, 54, 11], 3. Tensor SVD applied to denoise atomic-resolution 4D scanning transmission electron microscopy [59], and 4. Non-local Poisson PCA denoising [44, 57]. It is noteworthy that many of these applications predominantly employ the L2 case, which offers a comparatively simpler solution. As a result, data is often renormalized to convert Poisson distributions into Gaussian distributions [28]. Nevertheless, the efficacy of these applications could be significantly enhanced by the development of algorithms tailored explicitly for the Poisson case.

Furthermore, Problem (3) also surfaces in hyperspectral image denoising, where noise is assumed to follow a Poisson distribution [60, 58]. In the domain of text mining, the Poisson distribution assumption is frequently utilized for modeling word occurrences based on latent variables such as categories, leading to the problem formulation depicted by (3) [17, 38]. Additionally, within the context of recommender systems, several matrix factorization problems can be reformulated into the structure of (3) [46, 13, 27].

Other Optimization Approaches The literature offers a limited number of optimization methods suitable for addressing the problem presented by Equation (3). This constraint arises from the requirement of many optimization techniques to have a continuously differentiable gradient Lipschitz function. Examples of such techniques include gradient descent [40], perturbed gradient descent [20], nonlinear conjugate gradient method [9], various proximal point minimization algorithms [26], and second-order methods like the Newton-CG algorithms [42, 43].

One potentially attractive direction is the utilization of Proximal Alternating (Linearized) Minimization (PALM) [2] or Proximal Alternating Minimization (PAM) [1]. These algorithms are designed to solve problems of the form:

$$\underset{\mathbf{W}, \mathbf{H}}{\text{minimize}} \quad R_W(\mathbf{W}) + R_H(\mathbf{H}) + \mathcal{L}(\mathbf{W}, \mathbf{H})$$

PALM employs a Gauss-Seidel iteration scheme, consisting of the following sub-problems:

$$\begin{aligned} \mathbf{W}^{k+1} &= \arg \min_{\mathbf{W}} R_W(\mathbf{W}) + \mathcal{L}(\mathbf{W}, \mathbf{H}^k) + c_W \|\mathbf{W} - \mathbf{W}^k\|_2^2 \\ \mathbf{H}^{k+1} &= \arg \min_{\mathbf{H}} R_H(\mathbf{H}) + \mathcal{L}(\mathbf{W}^{k+1}, \mathbf{H}) + c_H \|\mathbf{H} - \mathbf{H}^k\|_2^2 \end{aligned}$$

Unfortunately, PALM requires the objective function $\mathcal{L}$ to possess a Lipschitz gradient, which is not the case in our scenario. Additionally, the Gauss-Seidel iterations generally lack a closed-form solution, resulting in a slow algorithm with sub-iterations.

To overcome the non-Lipschitz gradient issue, Bregman gradient descent (B-GD) [31, Algorithm 1.1] can be considered. This type of algorithm has been extended to alternating minimization [31, Algorithm 1.3 and 1.4]. Such an approach can be adapted to our case, as the objective function (3) exhibits relative smoothness for most regularization scenarios (see (6)). However, this optimization scheme involves non-tight majorization functions, leading to slow convergence, as discussed in Section 5.2.

Given the presence of two blocks of variables, Block Coordinate Descent (BCD) algorithms [53] naturally emerge as a potential solution. In fact, previous work [24] demonstrates that many existing approaches can be viewed as Block Coordinate Descent (BCD) problems. However, a primary challenge with BCD lies in its propensity to necessitate full minimization of the sub-problems, which proves to be challenging in the Poisson case. In the L2 case, the subproblems often have closed-form solutions. To address this concern, we explore BSUM algorithms [41] in this study, a generalization of BCD that avoids the requirement for full minimization of the subproblems, instead using upper bounds for the objective function.

Non-Negative Matrix Factorization Non-Negative Matrix Factorization (NMF) algorithms have been extensively studied, and for a comprehensive review, one can refer to [12]. Initially formulated as Positive Matrix Factorization for the Gaussian (L2-NMF) case by [39], NMF has seen numerous algorithmic developments. In the L2 case, popular approaches include the Alternating Nonnegative Least Squares (ANLS) framework [23, 25, 34] and the Hierarchical Alternating Least Squares (HALS) method [7, 8]. As for the Poisson case, known as KL NMF, the first algorithm using Multiplicative Updates (MU) was proposed by [30], with later demonstrations of its convergence provided in [29] for both Poisson and Gaussian cases. A more rigorous convergence analysis is presented in [33].

Considering the specific problem of Poisson KL-NMF, there have been a few notable contributions. For instance, [48] employed the Alternating Direction Method of Multipliers (ADMM) with the variable change $X = WH$. The Primal Dual algorithm, based on the framework from Chambolle-Pock, was explored by [56]. Moreover, [16] conducted a comparative study of various optimization algorithms for KL NMF, including MU [29], ADMM [48], Primal Dual [56], and Cyclic Coordinate Descent Method [18]. They also introduced three new algorithms for KL-NMF, namely Block Mirror Descent Method, A Scalar Newton-Type Algorithm, and A Hybrid SN-MU Algorithm.

In the introduction, we mentioned that there are few contributions that address the problem of regularized NMF, with many of them focusing on the L2 case. [24] demonstrated how many existing works can be cast as Block Coordinate Descent (BCD) problems, allowing the derivation of MU update rules for different regularizers, such as L1 for sparsity. However, their work is limited to L2 NMF. Xu et al. [55] proposed a general optimization scheme for block multiconvex optimization using block coordinate descent, which can accommodate regularization on each block. Although applied to L2-NMF, such an approach may lead to algorithms with sub-iterations. In the context of L2-NMF with sparsity constraints, [50] presented an approach to address this scenario. Additionally, [49] introduced a framework for handling L2-NMF with Lipschitz regularizers, akin to our term s_L . Other forms of regularization have also been explored, such as graph-based [6] or simplex constraint [17].

Regarding the specific problem of regularized KL loss, [15] provided a notable contribution. However, their work focused solely on the subproblems of the NMF problem, rather than addressing the NMF problem itself. Notably, one could potentially employ a similar approach to solve the subproblems of (3) using BCD. Nonetheless, this would result in a less efficient algorithm with sub-iterations.

3 Preliminaries

3.1 Notation

We reserve capital letters for matrices and vectors, e.g., $\mathbf{A}, \mathbf{a}$. We use $\mathbf{a}_i$ to refer to the i -th numbered vector, and a_i to denote the i -th element of vector $\mathbf{a}$. The j -th element of vector $\mathbf{a}_i$ or the element at the i -th row and j -th column of matrix $\mathbf{A}$ is denoted as a_{ij} .

$\mathbf{A} \geq 0$ and $\mathbf{a} \geq 0$ indicate that all entries of matrix $\mathbf{A}$ or vector $\mathbf{a}$ are greater than or equal to 0, i.e., $a_{ij} \geq 0$ for all i, j . $\mathbf{A}^\top, \mathbf{a}^\top$ represent the transpose of $\mathbf{A}$ and $\mathbf{a}$, respectively. We use $\mathbf{A}^t$ to denote $\mathbf{A}$ at step t . $\mathbf{A}^\top$ denotes the transpose of $\mathbf{A}^t$. $\circ$ and $\oslash$ denote elementwise multiplication (also known as the Hadamard product) and division for matrices, respectively. For example, $[\mathbf{A} \circ \mathbf{B}]_{ij} = a_{ij}b_{ij}$ and $[\mathbf{A} \oslash \mathbf{B}]_{ij} = \frac{a_{ij}}{b_{ij}}$.

As mentioned in the introduction, the matrix to be factorized is generally denoted as $\mathbf{Y} \approx \mathbf{WH}$, where $\mathbf{W}$ and $\mathbf{H}$ are its factors. We will use $\mathbf{w}, \mathbf{h}$ as the vectorized versions of $\mathbf{W}, \mathbf{H}$, while $\mathbf{w}_i, \mathbf{h}_i$ will denote the i -th column of $\mathbf{W}, \mathbf{H}$. $x, \mathbf{x}$ are general variables that can replace either $\mathbf{w}$ or $\mathbf{h}$. We use $\mathcal{L}(\mathbf{W}, \mathbf{H})$ as the general loss function, and $f(\mathbf{x})$ is broadly used to denote a multivariate scalar function.

Finally, we commonly employ calligraphic notation for variable domains. Let $\mathcal{X}$ serve as a generic domain for the variable x . In practical terms, it is frequently defined by the $\geq \epsilon$ constraint, specifically as $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^m | \mathbf{x} \geq \epsilon\}$. Additionally, we utilize $\mathcal{C}_w$ and $\mathcal{C}_h$ to represent the domains of $\mathbf{w}$ and $\mathbf{h}$, respectively.

3.2 Definitions

In this contribution, we consider the loss function $\mathcal{L}(\mathbf{w}, \mathbf{h})$ with two blocks of variables: $\mathbf{w} \in \mathcal{C}_w$ and $\mathbf{h} \in \mathcal{C}_h$, where $\mathcal{C}_w \subset \mathbb{R}^{m_w}$ and $\mathcal{C}_h \subset \mathbb{R}^{m_h}$ are both non-empty convex sets. Here, $\mathbf{w}$ and $\mathbf{h}$

correspond to the vectorized versions of the two matrices $\mathbf{W}$ and $\mathbf{H}$, respectively. Therefore, $\mathcal{C}_w$ and $\mathcal{C}_h$ often correspond to the sets $\mathbf{w} \geq \epsilon$ and $\mathbf{h} \geq \epsilon$ with $\epsilon > 0$. Let us use $\mathbf{z} = [\mathbf{w}, \mathbf{h}]$ to denote all the variables. We have $\mathbf{z} \in \mathcal{C} = \mathcal{C}_w \times \mathcal{C}_h \subset \mathbb{R}^m = \mathbb{R}^{m_w} \times \mathbb{R}^{m_h}$, where the total dimension of the problem is $m = m_w + m_h$.

Definition 1 (Directional derivative). *Let $\mathcal{L} : \mathcal{C} \rightarrow \mathbb{R}$ be a scalar function, where $\mathcal{C} \subset \mathbb{R}^m$ is a convex set. The directional derivative of $\mathcal{L}$ at point $\mathbf{x}$ in the direction $\mathbf{d}$ is defined by*

$$\mathcal{L}'(\mathbf{z}; \mathbf{d}) := \lim_{\lambda \downarrow 0} \frac{\mathcal{L}(\mathbf{z} + \lambda \mathbf{d}) - \mathcal{L}(\mathbf{z})}{\lambda}$$

Note that when $\mathcal{L}$ is differentiable, $\mathcal{L}'(\mathbf{z}; \mathbf{d}) = \nabla \mathcal{L}(\mathbf{z}) \mathbf{d}^\top$ since $\mathbf{d}$ and $\mathbf{z}$ are row vectors. In this contribution, almost all functions are differentiable on the domain of interest.

Definition 2 (Coordinatewise Minimum). *The point $\mathbf{z} = [\mathbf{w}, \mathbf{h}] \in \mathcal{C}$ is a coordinatewise minimum of a function $\mathcal{L}$ if*

$$\begin{aligned} \mathcal{L}(\mathbf{w} + \mathbf{d}_w, \mathbf{h}) &\geq \mathcal{L}(\mathbf{w}, \mathbf{h}) \quad \forall \mathbf{d}_w \in \mathbb{R}^{m_w} \quad \text{with} \quad \mathbf{w} + \mathbf{d}_w \in \mathcal{C}_w \\ \mathcal{L}(\mathbf{w}, \mathbf{h} + \mathbf{d}_h) &\geq \mathcal{L}(\mathbf{w}, \mathbf{h}) \quad \forall \mathbf{d}_h \in \mathbb{R}^{m_h} \quad \text{with} \quad \mathbf{h} + \mathbf{d}_h \in \mathcal{C}_h \end{aligned}$$

A coordinatewise minimum is a natural termination point for an alternating minimization algorithm. However, it is important to note that a coordinatewise minimum is not equivalent to a local minimum, as it does not guarantee minimality in all directions. Figure 1 (left) provides a counterexample illustrating this.

Another significant concept is the notion of a stationary point, where the gradient is non-negative in all directions.

Definition 3 (Stationary Points of a function). *Let $\mathcal{L} : \mathcal{C} \rightarrow \mathbb{R}$ be a scalar function, where $\mathcal{C} \subset \mathbb{R}^m$ is a convex set. A point $\mathbf{x}$ is a stationary point of $\mathcal{L}$ if*

$$\mathcal{L}'(\mathbf{z}; \mathbf{d}) \geq 0 \quad \forall \mathbf{d} | \mathbf{z} + \mathbf{d} \in \mathcal{C}$$

We emphasize that a stationary point is not equivalent to a *strict* local minimum as there might be directions where the directional derivative equals 0. For example, in the simple function $f([\mathbf{w}, \mathbf{h}]) = (w\mathbf{h} - 2)^2$, the point $[\mathbf{w}, \mathbf{h}] = [\sqrt{2}, \sqrt{2}]$ has a zero derivative in the direction $[1, -1]$, which corresponds to rescaling the solution as $[\alpha, 1/\alpha]$. Even worse, a stationary point is not necessarily a local minimum, even if it is a coordinatewise minimum, as shown in Figure 1 (a). Here, in the diagonal directions, the directional derivative equals 0 but the function is concave in this direction.

When it comes to the Poisson loss function, there is, at least, a continuous set of local minimas corresponding to rescaling the solution. This is illustrated in Figure 1 (b). Note that the introduction of regularization or constraints can lead to strict local minima, as shown in Figure 1 (c).

In this contribution, we prove convergence to a coordinatewise minimum that is also a stationary point. To accomplish this, we will consider a class of functions that are regular at their coordinatewise minima.

Definition 4 (Regularity of a function at a point). *The function $\mathcal{L} : \mathcal{C} \rightarrow \mathbb{R}$ is said to be regular at the point $\mathbf{z} \in \mathcal{C}$ if $\mathcal{L}'(\mathbf{z}; \mathbf{d}) \geq 0$ for all $\mathbf{d} = [\mathbf{d}_w, \mathbf{d}_h] \in \mathbb{R}^m$ such that $\mathcal{L}'(\mathbf{z}; [\mathbf{d}_w, \mathbf{0}]) \geq 0$ and $\mathcal{L}'(\mathbf{z}; [\mathbf{0}, \mathbf{d}_h]) \geq 0$.*

Lemma 1. *Continuously differentiable functions are regular at their coordinatewise minima.*

Proof is provided in Appendix A.1. Lemma 1 plays a crucial role, as it ensures that the coordinatewise minimum we converge to in Theorem 1 is also a stationary point. In this work, we assume the regularizer to be continuously differentiable on the domain.

3.3 Approximation functions

In order to facilitate optimization algorithms, it is beneficial to work with approximation functions that majorize or approximate the objective function at a given point. One commonly used class of approximation functions is known as *first-order majorization* functions. These functions provide a convenient framework for constructing surrogates and facilitating optimization. We adopt the definition of first-order majorization functions from [41].

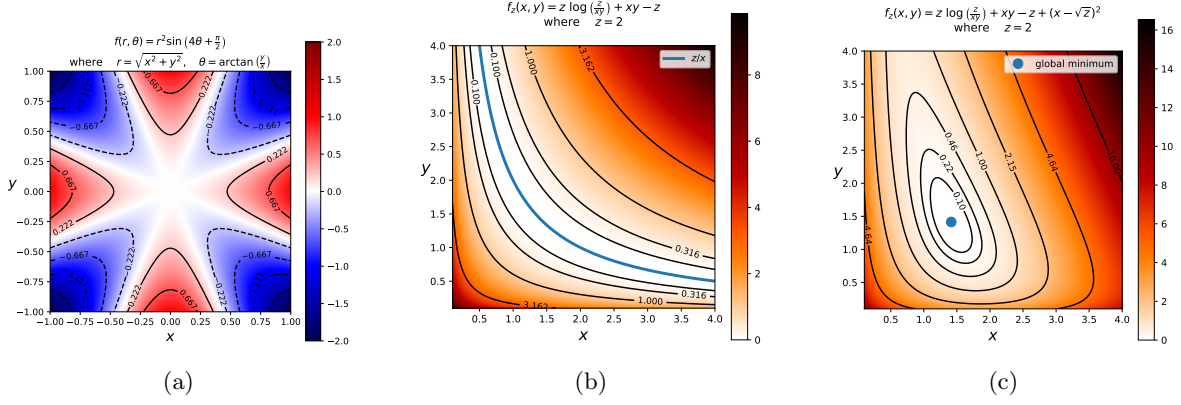


Figure 1: (a) A function where the point $(0,0)$ is a coordinatewise minimum along the x and y directions but is not a local minimum. (b) The blue line represents the set of all global minima for the one-dimensional case of (2), where all coordinatewise minima are also stationary points. (c) We show the effect of adding a quadratic regularization term to (2), ensuring the uniqueness of the global minimum.

Definition 5. [41, Assumption 1] A function $g(\mathbf{x}, \mathbf{x}^t)$ is said to be a first-order majorization of f at the point $\mathbf{x}^t$ if it satisfies the following properties:

- A.1 $g(\mathbf{x}, \mathbf{x}^t) \geq f(\mathbf{x}) \quad \forall \mathbf{x}, \mathbf{x}^t \in \mathcal{X},$
- A.2 $g(\mathbf{x}^t, \mathbf{x}^t) = f(\mathbf{x}^t) \quad \forall \mathbf{x}^t \in \mathcal{X},$
- A.3 $g'(\mathbf{x}, \mathbf{x}^t; \mathbf{d})|_{\mathbf{x}=\mathbf{x}^t} = f'(\mathbf{x}^t; \mathbf{d}) \quad \forall \mathbf{d} \text{ such that } \mathbf{x}^t + \mathbf{d} \in \mathcal{X},$
- A.4 $g(\mathbf{x}, \mathbf{x}^t)$ is continuous in $(\mathbf{x}, \mathbf{x}^t)$.

It is worth noting that for continuously differentiable functions, the third statement can be equivalently expressed as $\nabla_{\mathbf{x}} g(\mathbf{x}^t, \mathbf{x}^t) = \nabla f(\mathbf{x}^t)$. Although the definition of first-order majorization functions resembles the concept of surrogate functions introduced in [37, Definition 2.2], the additional requirement for a surrogate function is that $g(\mathbf{x}, \mathbf{x}^t) - f(\mathbf{x})$ is L gradient Lipschitz as defined in (5). Importantly, all majorization functions defined in the following Section 4.1 satisfy this condition and can thus serve as majorization functions.

Conveniently, majorization functions can be built term by term, leveraging their additivity property. This property allows us to combine multiple majorization functions to obtain a new majorization function.

Lemma 2. First-order majorization functions are additive. If $g_1(\mathbf{x}, \mathbf{x}^t)$ and $g_2(\mathbf{x}, \mathbf{x}^t)$ majorize $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ at $\mathbf{x}^t$, respectively, then $g_1(\mathbf{x}, \mathbf{x}^t) + g_2(\mathbf{x}, \mathbf{x}^t)$ majorizes $f(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x})$ at $\mathbf{x}^t$.

Proof. The additivity property preserves each property of (5). $\square$

Lemma 2 provides a valuable tool for constructing majorization functions by combining simpler majorization functions. Additionally, when proving that a function is majorizing, it is often unnecessary to explicitly demonstrate the equality of partial derivatives or gradients at $\mathbf{x}^t$. Instead, in the case of differentiable functions, it is typically sufficient to establish the first two properties (A.1 and A.2). According to [41, Proposition 1], properties A.3 and A.4 follow as a consequence. Intuitively, one can observe that the continuity of the gradient ensures that the majorization function g shares the tangent spaces with f at the point $\mathbf{x}^t$.

3.4 Two Blocks Successive Minimization (TBSUM)

The TBSUM algorithm is designed to solve the following problem:

$$\begin{aligned}
 & \underset{\mathbf{h}, \mathbf{w}}{\text{minimize}} && \mathcal{L}(\mathbf{w}, \mathbf{h}) \\
 & \text{such that} && \mathbf{h} \in \mathcal{C}_h, \mathbf{w} \in \mathcal{C}_w
 \end{aligned} \tag{9}$$

It relies on two first-order majorizing functions: $g_w(\mathbf{w}, \mathbf{w}^t, \mathbf{h}^t)$ and $g_h(\mathbf{h}, \mathbf{h}^t, \mathbf{w}^t)$, which majorize $\mathcal{L}(\mathbf{w}, \mathbf{h})$ at $(\mathbf{w}^t, \mathbf{h}^t)$ for all $\mathbf{w}^t \in \mathcal{C}_w$ and $\mathbf{h}^t \in \mathcal{C}_h$. The construction of these functions will be presented in Section 4. The TBSUM algorithm, outlined in Algorithm 1, alternates between minimizing g_w and g_h . It is assumed that the subproblem solutions are unique. Theorem 1 establishes the convergence of the TBSUM algorithm, which is a variant of the algorithm presented in [41, Theorem 2a] adapted for solving the specific problem at hand.

Algorithm 1 TBSUM: Two-Block Successive Minimization Algorithm

Require: Initialize the variables to a feasible point $\mathbf{w}^0 \in \mathcal{C}_w$, $\mathbf{h}^0 \in \mathcal{C}_h$, and set $t = 0$

- 1: **repeat**
 - 2: $\mathbf{w}^{t+1} \leftarrow \arg \min_{\mathbf{w} \in \mathcal{C}_w} g_w(\mathbf{w}, \mathbf{w}^t, \mathbf{h}^t)$
 - 3: $\mathbf{h}^{t+1} \leftarrow \arg \min_{\mathbf{h} \in \mathcal{C}_h} g_h(\mathbf{h}, \mathbf{h}^t, \mathbf{w}^{t+1})$
 - 4: $t \leftarrow t + 1$
 - 5: **until** some convergence criterion is met
-

Theorem 1 (Convergence of TBSUM Algorithm 1). *Given two quasi-convex first order majorizing functions $g_h(\mathbf{w}, \mathbf{w}^t, \mathbf{h}^t)$ and $g_w(\mathbf{h}, \mathbf{h}^t, \mathbf{w}^{t+1})$ of $\mathcal{L}(\mathbf{w}, \mathbf{h})$ at $(\mathbf{w}^t, \mathbf{h}^t)$, $\forall \mathbf{w}^t, \mathbf{h}^t \in \mathcal{C}_w \times \mathcal{C}_h$. Furthermore assuming that the two subproblems in the TBSUM Algorithm 1 have unique solutions for any points $\mathbf{w}^t \in \mathcal{C}_w$, $\mathbf{h}^t \in \mathcal{C}_h$. Then, every limit point $\mathbf{z} = [\mathbf{w}, \mathbf{h}]$ of the iterates generated by the TBSUM Algorithm 1 is a coordinatewise minimum of (9). In addition, if $\mathcal{L}$ is regular at any point $\mathbf{z} \in \mathcal{C}$, then $\mathbf{z}$ is a stationary point of (9).*

4 Subproblem minimization

In this section, we focus on constructing the appropriate majorization functions $g_h(\mathbf{w}, \mathbf{w}^t, \mathbf{h}^t)$ and $g_w(\mathbf{h}, \mathbf{h}^t, \mathbf{w}^{t+1})$ for our problem (3). Since we consider the same type of regularization for $\mathbf{w}$ and $\mathbf{h}$, both subfunctions have the same form.

Practically, the loss function can be rewritten as

$$\mathcal{L}_Y(\mathbf{W}, \mathbf{H}) + R(\mathbf{W}, \mathbf{H}) = \sum_j f_w(\mathbf{w}_j) = \sum_i f_h(\mathbf{h}_i)$$

where $\mathbf{w}_j$ and $\mathbf{h}_i$ are the i^{th} row and j^{th} column of $\mathbf{W}$ and $\mathbf{H}$. The functions f_w and f_h have the form:

$$f(\mathbf{x}) = - \sum_{i=1}^m (b_i \log(\mathbf{a}_i^\top \mathbf{x}) + \mathbf{a}_i^\top \mathbf{x}) + s_L(\mathbf{x}) + s_R(\mathbf{x}) + \sum_{j=1}^n s_C(x_j). \quad (10)$$

Therefore, in this section, our objective is to find majorization functions for (10). Once this is done, we will provide closed-form solutions for steps 1 and 2 of Algorithm 1. It is worth noting that each term of (10) can be handled separately using the additivity property of majorizing functions (Lemma 2).

4.1 Majorizing functions

The following four lemmas provide majorizing functions for the different term of our objective function. Proofs are provided in Appendix A.2.

In order to develop an efficient algorithm, our objective is to identify majorizing functions that result in sub-problems with closed-form tractable solutions. Often, this can be accomplished under two conditions: 1. all the majorizing functions are of the same form, and, 2. the majorization function is separable with respect to the variables $\mathbf{x}$, i.e., $g(\mathbf{x}) = \sum_i g_i(\mathbf{x}_i)$. Within the scope of this contribution, we consider two forms of majorizing functions: quadratic $g(x) = a + bx + cx^2$ and logarithmic $g(x) = a + bx - c \log(x)$.

First, we propose a majorization scheme for the logarithmic term $\log(\mathbf{a}_i^\top \mathbf{x})$ in the objective function (10). We utilize a widely used majorization technique based on the concavity of the logarithm function. This technique has been employed in the original work by Lee and Seung [29] as well as in many EM (Expectation-Maximization) schemes.

Lemma 3 (Log majorization). *Assuming $\mathbf{a} \circ \mathbf{x} > 0$, for $\mathbf{x} \in \mathcal{C}$, let us define $q_j = \frac{a_j x_j^t}{\sum_k a_k x_k^t}$ for $\mathbf{x}^t \in \mathcal{C}$, then $g(\mathbf{x}, \mathbf{x}^t) = -\sum_j q_j \log\left(\frac{a_j x_j}{q_j}\right)$ is a first order majorizing function of $f(\mathbf{x}) = -\log(\mathbf{a}^\top \mathbf{x}) = -\log\left(\sum_j a_j x_j\right)$.*

We now proceed to majorize the different terms of the regularisation function $s_L(\mathbf{x})$, $s_R(\mathbf{x})$, and $s_C(x_j)$. We can majorize any Lipschitz function using the following lemma.

Lemma 4 (Lipschitz-majorization). *Given $s_L(\mathbf{x})$ a gradient Lipschitz function with constant σ_L over the domain $\mathbf{x} \in \mathcal{C}$. The functions*

$$g_1(\mathbf{x}, \mathbf{x}^t) = s_L(\mathbf{x}^t) + (\mathbf{x} - \mathbf{x}^t)^\top \nabla s_L(\mathbf{x}^t) + \sigma_L \|\mathbf{x} - \mathbf{x}^t\|_2^2 \quad (11)$$

and

$$g_2(\mathbf{x}, \mathbf{x}^t) = s_L(\mathbf{x}^t) + (\mathbf{x} - \mathbf{x}^t)^\top \nabla s_L(\mathbf{x}^t) + 2\sigma_L \left(\max_j x_j^t\right) \left(\sum_j x_j^t \log\left(\frac{x_j^t}{x_j}\right) - x_j^t + x_j\right) \quad (12)$$

are first order majorizing functions at $\mathbf{x}^t \in \mathcal{C}$.

We note that (11) (quadratic majorisation) is tighter than (12) (logarithmic majorisation). However, the looser majorisation function is needed to obtain a close form solution for the MU (see Section 4).

Next, the term that is relatively smooth can be majorized using the following lemma.

Lemma 5 (Relative smoothness majorization). *Assuming $s_R(\mathbf{x})$ a σ_R relatively smooth function with respect to $\kappa(\mathbf{x}) = -\mathbf{1}^\top \log(\mathbf{x})$ for $\mathbf{x} \in \mathcal{C} \subset \mathbb{R}_+^n$. Then the function*

$$g(\mathbf{x}, \mathbf{x}^t) = s_R(\mathbf{x}^t) + \langle \nabla s_R(\mathbf{x}^t), \mathbf{x} - \mathbf{x}^t \rangle + \sigma_R \sum_i \left(\frac{x_i}{x_i^t} - \log\left(\frac{x_i}{x_i^t}\right) - 1 \right) \quad (13)$$

is a first order majorizing function of $s_R(\mathbf{x})$ for $\mathbf{x}^t \in \mathcal{C}$.

Lemma 6 (Concave majorisation). *Given $s(x)$ a concave function defined on $x \in \mathcal{C} \subset \mathbb{R}$, it's linear approximation at the point x^t*

$$g(x, x^t) = s(x^t) + \frac{\partial s(x^t)}{\partial x} (x - x^t) \quad (14)$$

is a first order majorization function for $x^t \in \mathcal{C}$.

4.2 Subproblem updates

Now that we have defined majorizing functions for each term of (10), we can apply the additivity property of Lemma 2 to obtain a general majorizing function for $f(\mathbf{x})$:

$$\begin{aligned} g(\mathbf{x}, \mathbf{x}^t) = & -\sum_{i=1}^m \left(b_i \sum_j q_{ij} \log\left(\frac{a_{ij} x_j}{q_{ij}}\right) + \mathbf{a}_i^\top \mathbf{x} \right) \\ & + s_L(\mathbf{x}^t) + (\mathbf{x} - \mathbf{x}^t)^\top \nabla s_L(\mathbf{x}^t) + 2\sigma_L \left(\max_j x_j^t\right) \left(\sum_j x_j^t \log\left(\frac{x_j^t}{x_j}\right) - x_j^t + x_j\right) \\ & + s_R(\mathbf{x}^t) + \langle \nabla s_R(\mathbf{x}^t), \mathbf{x} - \mathbf{x}^t \rangle + \sigma_R \sum_j \left(\frac{x_j}{x_j^t} - \log\left(\frac{x_j}{x_j^t}\right) - 1 \right) \\ & + \sum_{j=1}^n s_C(x_j^t) + \frac{\partial s_C(x_j^t)}{\partial x_j} (x_j - x_j^t) \end{aligned} \quad (15)$$

where $q_{ij} = \frac{a_{ij}x_j^t}{\sum_k a_{ik}x_k^t}$. We use the colors green, blue, orange, and purple to denote and keep track of the dependencies of the different terms in (10). Finding the local optimum of the majorizing function will provide us with an update for Algorithm 1.

Proposition 1 (Generalized MU for (10)). *Assuming $\mathbf{x}^t, \mathbf{x}, \mathbf{b}, \mathbf{A} > 0$, the first-order majorizing function defined in (15) is strictly convex, and its global minimum $\mathbf{x}^{t+1}$ is given by*

$$x_j^{t+1} = x_j^t \frac{\alpha_j^t}{\beta_j^t} \quad (16)$$

where

$$\alpha_j^t = \sum_i b_i \frac{a_{ij}}{\sum_k a_{ik}x_k^t} + 2 \left(\max_i x_i^t \right) \sigma_L + \frac{\sigma_R}{x_j^t} \quad \text{and} \quad (17)$$

$$\beta_j^t = \sum_i a_{ij} + \nabla_{x_j} s_L(\mathbf{x}^t) + 2 \left(\max_i x_i^t \right) \sigma_L + \nabla_{x_j} s_R(\mathbf{x}^t) + \frac{\sigma_R}{x_j^t} + \frac{\partial s_C(x_j^t)}{\partial x}. \quad (18)$$

The proof is provided in Appendix A.3.

Generalization of the traditional MU Rule We note that (16) serves as a generalization of the original Multiplicative Update (MU) rule presented in [29]. Removing the regularization terms (blue, orange, and purple) results in precisely the MU rule as outlined in [29].

Connection with (Block) Mirror Descent [16, Algorithm 1] Another interesting observation is that the majorization of the relatively smooth term is done similarly to a Bregman proximal method algorithm [14]. Since the objective function $-\sum_{i=1}^m (b_i \log(\mathbf{a}_i^\top \mathbf{x}) + \mathbf{a}_i^\top \mathbf{x})$ is relatively smooth, one could drop all terms except for s_R and optimize using Block Bregman Proximal Gradient (BBPG) [50]. This would result in an algorithm very similar to Block Mirror Descent (BMD), which has recently been proposed for solving Poisson NMF [16]. Nevertheless, we advice against this this solution as discussed further in Section 5.2.

Alternative majorizing function and Quadratic Update (QU) As shown experimentally in Section 6 and illustrated in Figure 2, having a majorization function as tight as possible leads to faster convergence of the algorithm. In (1), we deliberately choose to use a looser majorizing function for the term s_L in order to recover an algorithm with multiplicative update that generalizes the original approach from [29]. However, instead of using (12), one can also use (11) when constructing the majorizing function:

$$\begin{aligned} g(\mathbf{x}, \mathbf{x}^t) = & - \sum_{i=1}^m \left(b_i \sum_j q_{ij} \log \left(\frac{a_{ij}x_j}{q_{ij}} \right) + \mathbf{a}_i^\top \mathbf{x} \right) \\ & + s_L(\mathbf{x}^t) + (\mathbf{x} - \mathbf{x}^t)^\top \nabla s_L(\mathbf{x}^t) + \sigma_L \|\mathbf{x} - \mathbf{x}^t\|_2^2 \\ & + s_R(\mathbf{x}^t) + \langle \nabla s_R(\mathbf{x}^t), \mathbf{x} - \mathbf{x}^t \rangle + \sigma_R \sum_j \left(\frac{x_j}{x_j^t} - \log \left(\frac{x_j}{x_j^t} \right) - 1 \right) \\ & + \sum_{j=1}^n s_C(x_j^t) + \frac{\partial s_C(x_j^t)}{\partial x_j} (x_j - x_j^t) \end{aligned} \quad (19)$$

which is also a strictly convex function.

Proposition 2 (QU for (10)). *Assuming $\mathbf{x}^t, \mathbf{x}, \mathbf{b}, \mathbf{A} > 0$, the first-order majorizing function defined in Equation (19) is strictly convex, and its global minimum $\mathbf{x}^{t+1}$ is given by*

$$x_j^{t+1} = \frac{-\beta_j^t + \sqrt{(\beta_j^t)^2 + 4\alpha\zeta_j^t}}{2\alpha} \quad (20)$$

where

$$\alpha = 2\sigma_L \quad \beta_j^t = \sum_i a_{ij} + \nabla_{x_j} s_L(\mathbf{x}^t) - 2\sigma_L x_j^t + \nabla_{x_j} s_R(\mathbf{x}^t) + \frac{\sigma_R}{x_j^t} + \frac{\partial s_C(x_j^t)}{\partial x} \quad \zeta_j^t = \sum_i b_i \frac{a_{ij} x_j^t}{\sum_k a_{ik} x_k^t} + \sigma_R. \quad (21)$$

The proof is provided in Appendix A.3. Both of these propositions lead to the update rule for our MU and QU algorithms detailed in Section 5. We note also that, with the appropriate assumptions, the update rule 16 and 20 will preserve positivity of the variable $\mathbf{x}$. However, since our desire is also to handle extra constraint, we develop in the next section rigorous approach.

4.3 Generalized simplex constraint

We need to handle two constraints: 1. the linear constraint $\mathbf{x} \geq \epsilon$, and, 2. the scale constraint $\mathbf{e}^\top \mathbf{x} = 1$, where $\mathbf{e} \geq 0$. While the first one is used to keep the variable non-negative, typically with a strictly positive small ϵ , the second one can set the scale of one of the variables ($\mathbf{W}$ or $\mathbf{H}$) in the factorization problem. Furthermore, in the case $\mathbf{e} = \mathbf{1}$, the simplex constraint is recovered. It turns out that the update rules of (16) and (20) can simply be updated to handle this constraint. The actual optimization problem we want to solve becomes:

$$\underset{\mathbf{x} \geq \epsilon}{\text{minimize}} \quad f(\mathbf{x}) \quad \text{such that} \quad \mathbf{e}^\top \mathbf{x} = 1.$$

where f is given in (10).

To solve this problem, we used the KKT approach, i.e, we find points that satisfy the KKT (Karush-Kuhn-Tucker) conditions:

$$\begin{aligned} 1. \text{ Stationarity} & \quad \nabla_{\mathbf{x}} L(\dot{\mathbf{x}}, \nu, \boldsymbol{\mu}) = \mathbf{0}, \\ 2. \text{ Primal feasibility} & \quad \begin{cases} \mathbf{e}^\top \dot{\mathbf{x}} - 1 = 0, \\ \dot{\mathbf{x}} \geq \epsilon \end{cases}, \\ 3. \text{ Dual feasibility} & \quad \boldsymbol{\mu} \geq \mathbf{0}, \\ 4. \text{ Complementary slackness} & \quad \boldsymbol{\mu}^\top (-\mathbf{x} + \epsilon \mathbf{1}) = 0, \end{aligned}$$

where the Lagrangian is defined as:

$$L(\mathbf{x}, \nu, \boldsymbol{\mu}) = f(\mathbf{x}) + \nu (\mathbf{e}^\top \mathbf{x} - 1) + \boldsymbol{\mu}^\top (-\mathbf{x} + \epsilon \mathbf{1}).$$

We follow the same method as developed in Section 4.2, except that we majorize the Lagrangian $L(\mathbf{x}, \nu, \boldsymbol{\mu})$. The resulting first-order majorizing function is given by:

$$g'(\mathbf{x}, \mathbf{x}^t, \nu, \boldsymbol{\mu}) = g(\mathbf{x}, \mathbf{x}^t) + \nu (\mathbf{e}^\top \mathbf{x} - 1) + \boldsymbol{\mu}^\top (-\mathbf{x} + \epsilon \mathbf{1})$$

where $g(\mathbf{x}, \mathbf{x}^t)$ is given in (15) or (19). We repeat the development of Section 4.2 (and the proofs of Appendix A.3). We end up with an update that is very similar to (16) or (20). In the MU case, we end up with:

$$x_j^{t+1} = x_j^t \frac{\alpha_j}{\beta_j + \nu e_j - \mu_j},$$

where the only final difference consists of two terms in cyan and violet (α_j and β_j remain identical). For the QU, we stick to the same update rule (20), where only β_j^t is modified:

$$\beta_j^{t'} = \beta_j^t + \nu e_j - \mu_j.$$

This update rule ensures the first of the KKT conditions (stationarity). We now find $\nu, \boldsymbol{\mu}$ such that the second KKT condition holds (primal feasibility). It turns out that $\boldsymbol{\mu}$ does not need to be computed explicitly. In the MU case, μ_j is selected to be large enough such that

$$x_j^{t+1} = \max \left(x_j^t \frac{\alpha_j}{\beta_j + \nu e_j}, \epsilon \right) = \frac{x_j^t \alpha_j}{\min \left(\frac{x_j^t \alpha_j}{\epsilon}, \beta_j + \nu e_j \right)}. \quad (22)$$

In the QU case, we obtain

$$\begin{aligned} x_j^{t+1} &= \max \left(\frac{-(\beta_j^t + \nu e_j) + \sqrt{(\beta_j^t + \nu e_j)^2 + 4\alpha\zeta_j^t}}{2\alpha}, \epsilon \right) \\ &= \frac{-\min \left(\frac{\zeta_j^t}{\epsilon} - \epsilon\alpha, \beta_j^t + \nu e_j \right) + \sqrt{\left(\min \left(\frac{\zeta_j^t}{\epsilon} - \epsilon\alpha, \beta_j^t + \nu e_j \right) \right)^2 + 4\alpha\zeta_j^t}}{2\alpha} \end{aligned} \quad (23)$$

Note that dual feasibility and complementary slackness could be verified, but we leave them out for simplicity. We then need to find the value of ν such that $\mathbf{e}^\top \mathbf{x} = 1$, which is equivalent to searching for

$$h_1(\nu) = \sum_j e_j \frac{x_j^t \alpha_j}{\min \left(\frac{x_j^t \alpha_j}{\epsilon}, \beta_j^t + \nu e_j \right)} - 1 = 0 \quad (24)$$

Similarly, for the quadratic update of (20), we search for ν that satisfies

$$h_2(\nu) = \sum_j e_j \frac{-\min \left(\frac{\zeta_j^t}{\epsilon} - \epsilon\alpha, \beta_j^t + \nu e_j \right) + \sqrt{\left(\min \left(\frac{\zeta_j^t}{\epsilon} - \epsilon\alpha, \beta_j^t + \nu e_j \right) \right)^2 + 4\alpha\zeta_j^t}}{2\alpha} - 1 = 0. \quad (25)$$

There is no closed-form solution for ν ; however, the value can be found using a simple dichotomy search. Bounds for starting the dichotomy are computed in Appendix B.

Case $\epsilon = 0$ Most of our reasoning relies on the fact that $x_i > 0$ and, therefore, on the domain constraint $\epsilon > 0$. We have found that setting ϵ to a small non-zero value works well in practice. However, our approach can likely be generalized to the case where $\epsilon = 0$, following the approach of [33, Section 4], which studies the unregularized Poisson NMF case.

5 Algorithms for Poisson matrix factorisation

Equipped with the update rules developed in the previous Sections 4 and 4.3, we are ready to tackle the general problem of this contribution³ which consists of minimizing (2):

$$\begin{aligned} \dot{\mathbf{W}}, \dot{\mathbf{H}} &= \arg \min_{\mathbf{W}, \mathbf{H}} -\langle \mathbf{Y}, \log(\mathbf{WH}) \rangle + \langle \mathbf{1}, \mathbf{WH} \rangle + R_W(\mathbf{W}) + R_H(\mathbf{H}) \\ &\text{such that } \mathbf{W} \geq \epsilon, \mathbf{H} \geq \epsilon, \mathbf{e}^\top \mathbf{H} = \mathbf{1} \end{aligned} \quad (26)$$

We first observe that with respect to each variable $\mathbf{W}, \mathbf{H}$, the problem is separable by column/row. For example, given $\mathbf{W}$, finding the optimal $\mathbf{H}$ can be done for each column:

$$\dot{\mathbf{h}}_i = \arg \min_{\mathbf{h}_i} -\langle \mathbf{y}_i, \log(\mathbf{W}\mathbf{h}_i) \rangle + \langle \mathbf{1}, \mathbf{W}\mathbf{h}_i \rangle + r_H(\mathbf{h}_i) \quad \text{such that } \mathbf{h}_i \geq \epsilon, \mathbf{e}^\top \mathbf{h}_i = 1$$

We therefore apply the TBSUM Algorithm 1, where all lines of $\mathbf{W}$ and all columns of $\mathbf{H}$ are updated independently, and obtain the two Algorithms 2 and 3. We note here that the function $\max(\cdot, \epsilon)$ ensures that the solutions are $\geq \epsilon$ (non-negativity).

³Here we show the problem with the linear constraint on $\mathbf{H}$, however, by symmetry, a similar algorithm can be developed with the constraint on $\mathbf{W}$.

Algorithm 2 MU Algorithm for Regularized Poisson NMF

```
1: Initialize the variables  $\mathbf{W}^0 \geq \epsilon$ ,  $\mathbf{H} \geq \epsilon$ ,  $t = 0$  such that  $\mathbf{e}^\top \mathbf{H} = \mathbf{1}$ .
2: while some convergence criterion is met do
3:   for each line  $\mathbf{w}_i^{t\top}$  of  $\mathbf{W}^t$  do
4:     Compute  $\alpha_i^{t\top}$  and  $\beta_i^{t\top}$  using (17) and (18).
5:     Update using the MU rule (22):  $w_{ij}^{t+1} \leftarrow \max\left(w_{ij}^t \frac{\alpha_{ij}^t}{\beta_{ij}^t}, \epsilon\right)$ .
6:   end for
7:   for each column  $\mathbf{h}_i^t$  of  $\mathbf{H}^t$  do
8:     Compute  $\alpha_i^t$  and  $\beta_i^t$  using (17) and (18).
9:     Find the dual variable  $\nu_i$  by dichotomy of the function (24) (set  $\nu = 0$  if no constraint is present).
10:    Update using the MU (22):  $h_{ij}^{t+1} \leftarrow \max\left(h_{ij}^t \frac{\alpha_{ij}^t}{\beta_{ij}^t + \nu_i e_j}, \epsilon\right)$ .
11:  end for
12:   $t \leftarrow t + 1$ .
13: end while
```

Algorithm 3 QU Algorithm for Regularized Poisson NMF

```
1: Initialize the variables  $\mathbf{W}^0 \geq \epsilon$ ,  $\mathbf{H} \geq \epsilon$ ,  $t = 0$  such that  $\mathbf{e}^\top \mathbf{H} = \mathbf{1}$ .
2: while some convergence criterion is met do
3:   for each line  $\mathbf{w}_i^{t\top}$  of  $\mathbf{W}^t$  do
4:     Compute  $\alpha^t$ ,  $\beta_i^{t\top}$ , and  $\gamma_i^{t\top}$  using (21).
5:     Update using the QU rule (23):  $w_{ij}^{t+1} \leftarrow \max\left(\frac{-\beta_{ij}^t + \sqrt{(\beta_{ij}^t)^2 + 4\alpha^t \zeta_{ij}^t}}{2\alpha^t}, \epsilon\right)$ .
6:   end for
7:   for each column  $\mathbf{h}_i^t$  of  $\mathbf{H}^t$  do
8:     Compute  $\alpha^t$ ,  $\beta_i^t$ , and  $\gamma_i^t$  using (21).
9:     Find the dual variable  $\nu_i$  by dichotomy of the function (25) (set  $\nu = 0$  if no constraint is present).
10:    Update using the QU rule (23):  $h_{ij}^{t+1} \leftarrow \max\left(\frac{-(\beta_{ij}^t + \nu_i e_j) + \sqrt{(\beta_{ij}^t + \nu_i e_j)^2 + 4\alpha^t \zeta_{ij}^t}}{2\alpha^t}, \epsilon\right)$ .
11:  end for
12:   $t \leftarrow t + 1$ .
13: end while
```

Convergence The two update rules in steps 5 and 10 correspond to minimizing first-order strictly convex majorization functions. As a result, we can apply Theorem 1 *to guarantee convergence towards a coordinate-wise minimum*. It is important to note that *this coordinate-wise minimum is also a stationary point*, given that the objective function remains regular for any point $\mathbf{W} \in \mathcal{C}_W$, $\mathbf{H} \in \mathcal{C}_H$.

5.1 Algorithm complexity

Let's examine the complexity of both Algorithms 2 and 3 when considering $\mathbf{W} \in \mathbb{R}^{n \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times m}$. In each iteration, the following complexities are observed:

- (a) Step 4 has a complexity of $\mathcal{O}(nmk)$.
- (b) Step 5 has a complexity of $\mathcal{O}(nk)$.
- (c) Step 8 has a complexity of $\mathcal{O}(nmk)$.
- (d) Step 9 has a complexity of $\mathcal{O}(c_d km)$, where c_d denotes the number of iterations performed by the dichotomy.
- (e) Step 10 has a complexity of $\mathcal{O}(mk)$.

Thus, the overall complexity per iteration can be expressed as $\mathcal{O}(nmk) + \mathcal{O}(c_d km) = \mathcal{O}((n + c_d)mk)$. This indicates that the computational complexity per iteration is linear with respect to the problem size, i.e., nm , multiplied by the number of components, i.e., k .

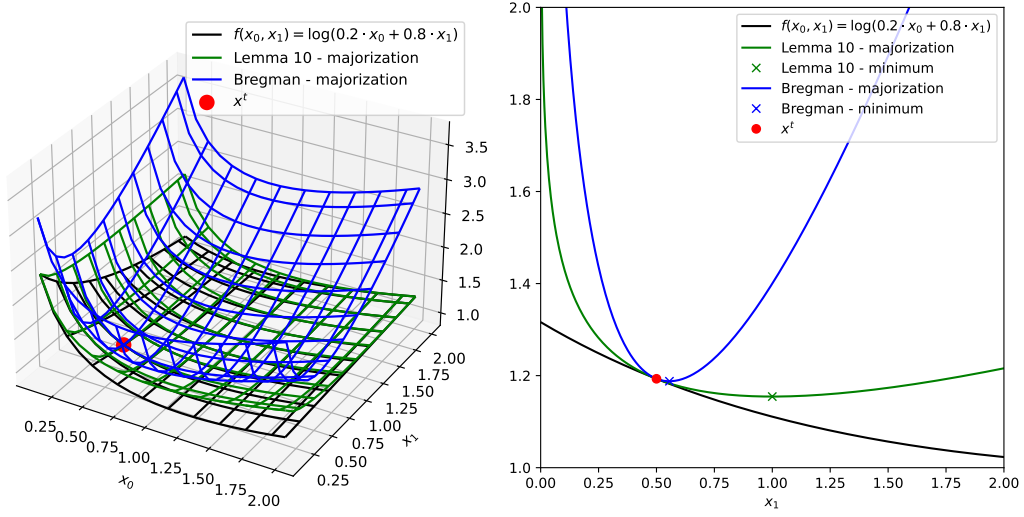


Figure 2: Two first-order majorizing functions for the function $f(x_0, x_1) = \log(0.2x_0 + 0.8x_1)$ are considered. We observe that Lemma (3) provides a tighter majorization than Lemma (5) for f , resulting in a more optimal update $\mathbf{x}^{t+1}$.

Impact of the dichotomy When n is small, the computational cost of the dichotomy in step 4 becomes dominant. Nevertheless, in general, for larger values of n , the impact of the dichotomy becomes negligible.

5.2 Tight Majorizing Functions

While we do not make any theoretical contributions concerning the speed of convergence of the algorithm, we want to emphasize the natural fact that *tighter majorizing functions lead to faster convergence*. Therefore, when evaluating an algorithm, we believe that the analysis of the underlying majorizing function is as insightful as the experimental evaluation. As an example, we could have used Block Mirror Descent to solve (3), as was done in [16, Algorithm 1]. This algorithm uses a Bregman Difference to create a majorization function for the subproblem. However, this would result in a much looser majorization function, which partly explains the slow convergence of this algorithm observed in [16]. This difference between majorization functions is exemplified in Figure 2.

Linesearch By tightening the bounds we used to construct the surrogate function, we can develop a more efficient algorithm. Here, we apply a classic "linesearch" method to the functions s_L . However, the same technique can be trivially applied to s_R as well. First, in (12) or (11), replace the constant σ_L with a parameter γ and initialize it with σ_L . Second, at each iteration, update the parameter γ according to the following rule:

$$\gamma^{t+1} = \begin{cases} v\gamma^t & \text{if } s_L(\mathbf{x}) \geq g(\mathbf{x}, \mathbf{x}^t, \gamma) \\ \frac{1}{\tau}\gamma^t & \text{otherwise.} \end{cases}$$

Here, v and τ are two update rates that determine how fast γ is updated. Choosing values that are too small for these parameters leads to an inefficient linesearch, while selecting values that are too large can result in strong oscillation patterns. Typical values for v and τ range from 1.05 to 1.5. However, it is important to note that when using linesearch, we are not guaranteed to converge, as we might invalidate the assumptions of Theorem 1.

6 Numerical Simulation

Problem In this section, we analyze the speed of convergence of Algorithms 2 and 3 through numerical simulations. As a regularizer for $\mathbf{H}$, we consider the Laplacian regularization $R(\mathbf{H}, \lambda) =$

$\frac{\lambda}{2} \text{tr}(\mathbf{H}^T \Delta \mathbf{H})$, where Δ represents the two-dimensional Laplacian for the k th line of $\mathbf{H} \in \mathbb{R}^{k \times p^2}$ reshaped as k images of size $p \times p$. Since a straightforward approach to minimize $R(\mathbf{H}, \lambda)$ is to reduce the amplitude of $\mathbf{H}$, we add the simplex constraint $\mathbf{1}^T \mathbf{H} = \mathbf{1}$. This leads to the following optimization problem:

$$\dot{\mathbf{W}}, \dot{\mathbf{H}} = \arg \min_{\mathbf{W}, \mathbf{H}} - \langle \mathbf{Y}, \log(\mathbf{W}\mathbf{H}) \rangle + \langle \mathbf{1}, \mathbf{W}\mathbf{H} \rangle + \frac{\lambda}{2} \text{tr}(\mathbf{H}^T \Delta \mathbf{H}) \quad \text{subject to } \mathbf{W} \geq \epsilon, \mathbf{H} \geq \epsilon, \mathbf{1}^T \mathbf{H} = \mathbf{1}$$

This particular problem can be applied in various domains, such as Non-Negative Matrix Factorization for hyperspectral images [36] and remote sensing [32] (See Related Work Section 2 for more references and applications). Our algorithms and regularisations were specifically developed for the *espm* python package [51]. All algorithms and experiments can be found in the *espm* package.

Dataset We construct two datasets consisting of 50 randomly drawn samples. In the first dataset, both matrices $\mathbf{W}$ and $\mathbf{H}$ are randomly generated from a uniform distribution. In the second dataset, each column of $\mathbf{W}$ corresponds to the sum of Gaussian functions that are randomly centered and scaled. The matrix $\mathbf{H}$ represents random smooth images. This second dataset is created using the *espm* package [51], where the toy model is used for $\mathbf{W}$ and $\mathbf{W}$ is generated using the "laplacian" weight type. This choice of dataset is selected because it can benefit from the Laplacian regularization on $\mathbf{H}$.

Once $\mathbf{W}$ and $\mathbf{H}$ are generated, the noiseless matrix $\mathbf{Y}$ is obtained as $\mathbf{Y} = \mathbf{W}\mathbf{H}$. We introduce noise by independently sampling each element $\tilde{\mathbf{Y}}_{ij} \sim \frac{\text{Poisson}(\lambda \mathbf{Y}_{ij})}{\lambda}$, where λ can be regarded as the noise control parameter. For all samples, we set $\mathbf{W} \in \mathbb{R}^{n \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times p^2}$ with $k = 3$, $p = 64$, and n selected from the set 25, 100, 500, 1000. Thus, the images in the dataset have dimensions of 64×64 .

Results We compare the performance of Algorithm 2 (MU), Algorithm 3 (QU), Block Mirror Descent (similar to [16, Algorithm 1]) and the projected gradient algorithm applied to (3). Figure 3 displays the convergence curves for 1000 iterations and $n = 25$. Although the overall complexity of all algorithms is the same, the time per iteration differs due to the different operations performed within each iteration and the time spent on dichotomy to compute the dual variable ν . Therefore, we provide the time in seconds for each algorithm in Figure 4 for various value of n . All results are averaged over 50 repetitions.

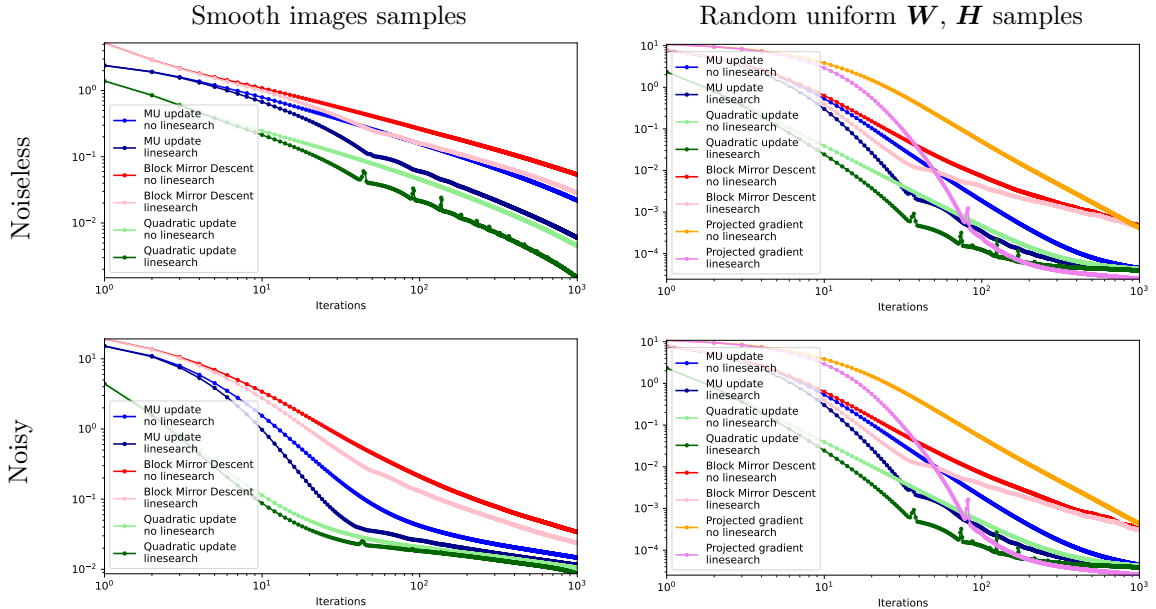


Figure 3: Convergence curves for 1000 iterations. We remove the minimum loss $\mathcal{L}(\dot{\mathbf{W}}, \dot{\mathbf{H}})$

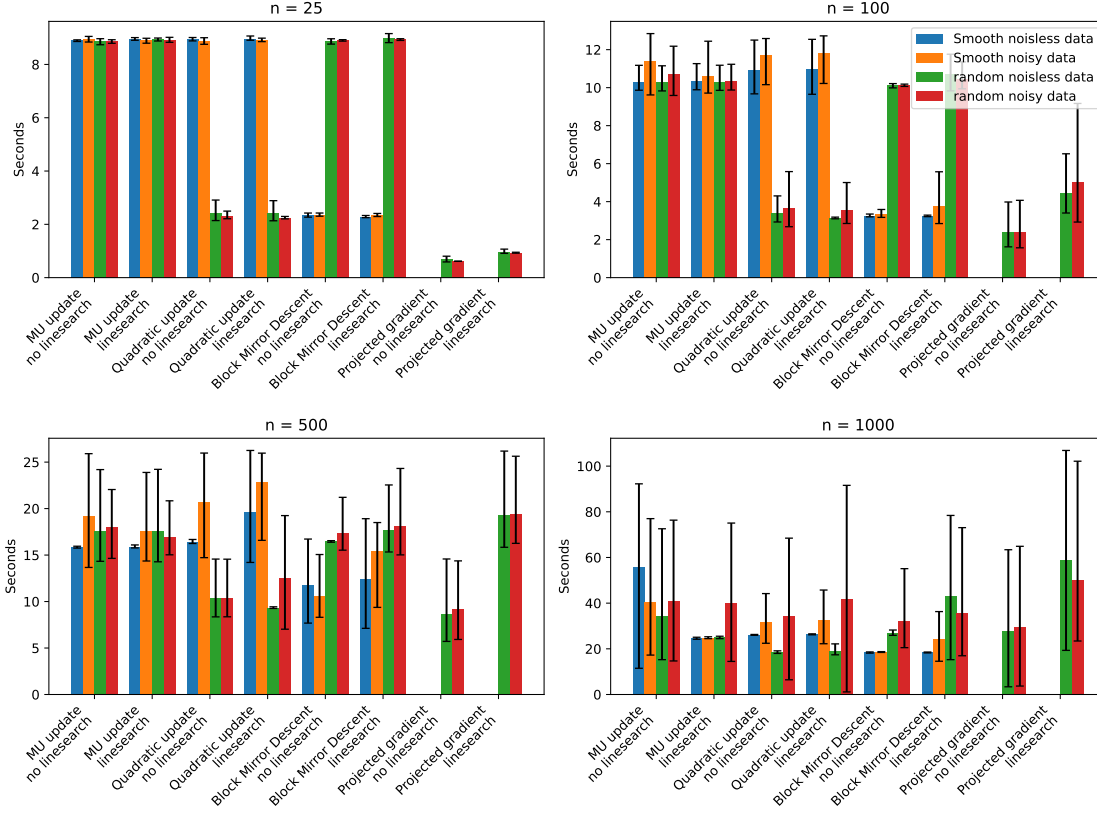


Figure 4: Execution time for 100 iterations for different problem sizes. Here we fix the dimension of $\mathbf{H}$ to 3×64^2 and varies $\mathbf{W}$ from 25×3 to 1000×3 .

Discussion Let’s discuss the results in more detail:

- *Number of iterations:* Figure 3 illustrates the convergence behavior of QU and MU algorithms. It is evident that QU converges faster per iteration compared to MU, which aligns with our expectations due to the tighter majorizing function used in QU. However, it is important to note that the introduction of the linesearch technique, while accelerating convergence, can lead to occasional instability, as indicated by occasional increases in the loss function. This observation supports our earlier discussion in Section 5.2. The challenge with Projected Gradient Descent is that we need to find an initial learning rate that is not too large, as the algorithm can diverge and not too small as the algorithm can be slow. Overall, since the selected learning rate cannot be selected optimally, the algorithm is slower than QU and MU. The Block Mirror Descent algorithm is also slower than QU and MU, which is consistent with the results of [16] and can be explained by the fact that the majorizing function used in the algorithm is looser.

- *Time per iteration:* Figure 4 presents the total time taken by the algorithms to complete 100 iterations. The results demonstrate that for small values of n , the computation of the p^2 dual variable during the dichotomy process dominates the overall execution time. However, as n increases, the time spent on dichotomy becomes negligible in comparison. These findings align with the complexity per iteration discussed in Section 5.1.

7 Conclusion

This contribution is the first to address the Poisson NMF problem with general regularization terms, such as Lipschitz functions, relatively smooth functions, or those expressed as linear constraints. We introduce two new algorithms and demonstrate their convergence to a coordinate-wise minimum, which is also a stationary point. Emphasizing the impact of the majorizing function choice on convergence speed, we validate our findings through numerical simulations. In essence, we believe that this work

serves as a helpful guide for developing efficient algorithms suited for regularized Poisson NMF problems.

A Proofs

In this Appendix, we provide the different proofs used in the paper.

A.1 Proof of Lemma 1

We note that this lemma and its proof likely exist in the literature, but we were unable to find a reference.

Lemma 1. *Continuously differentiable functions are regular at their coordinatewise minimums.*

Proof. If $\mathcal{L}$ is continuously differentiable, the directional derivative can be written as $\mathcal{L}'(\mathbf{z}; \mathbf{d}) = \mathbf{d}^\top \nabla \mathcal{L}(\mathbf{z})$. At a coordinatewise minimum $\mathbf{z}$, we have by definition:

$$\mathcal{L}'(\mathbf{z}; [\mathbf{d}_w, \mathbf{0}]) = \nabla \mathcal{L}(\mathbf{z}) [\mathbf{d}_w, \mathbf{0}]^\top \geq 0$$

and

$$\mathcal{L}'(\mathbf{z}; [\mathbf{0}, \mathbf{d}_h]) = \nabla \mathcal{L}(\mathbf{z}) [\mathbf{0}, \mathbf{d}_h]^\top \geq 0.$$

Therefore,

$$\begin{aligned} \mathcal{L}'(\mathbf{z}; \mathbf{d}) &= \nabla \mathcal{L}(\mathbf{z}) [\mathbf{d}_w, \mathbf{d}_h]^\top \\ &= \nabla \mathcal{L}(\mathbf{z}) ([\mathbf{d}_w, \mathbf{0}]^\top + [\mathbf{0}, \mathbf{d}_h]^\top) \\ &= \nabla \mathcal{L}(\mathbf{z}) [\mathbf{d}_w, \mathbf{0}]^\top + \nabla \mathcal{L}(\mathbf{z}) [\mathbf{0}, \mathbf{d}_h]^\top \\ &\geq 0. \end{aligned}$$

□

A.2 Proof of majorizing functions

Lemma 3 (Log majorization). *Assuming $\mathbf{a} \circ \mathbf{x} > \mathbf{0}$, for $\mathbf{x} \in \mathcal{C}$, let us define $q_j = \frac{a_j x_j^t}{\sum_k a_k x_k^t}$ for $\mathbf{x}^t \in \mathcal{C}$, then $g(\mathbf{x}, \mathbf{x}^t) = -\sum_j q_j \log\left(\frac{a_j x_j}{q_j}\right)$ is a first order majorizing function of $f(\mathbf{x}) = -\log(\mathbf{a}^\top \mathbf{x}) = -\log\left(\sum_j a_j x_j\right)$.*

Proof. First let us observe that

$$\begin{aligned} g(\mathbf{x}^t, \mathbf{x}^t) &= -\sum_j \frac{a_j x_j^t}{\sum_k a_k x_k^t} \log\left(\frac{a_j x_j^t}{\sum_k a_k x_k^t}\right) \\ &= -\sum_j \frac{a_j x_j^t}{\sum_k a_k x_k^t} \log\left(\sum_k a_k x_k^t\right) \\ &= -\log\left(\sum_k a_k x_k^t\right) = f(\mathbf{x}^t). \end{aligned}$$

Therefore, we have $g(\mathbf{x}, \mathbf{x}^t) \geq f(\mathbf{x})$. The inequality follows from the convexity of the $-\log$ function:

$$-\log\left(\sum_j a_j x_j\right) = -\log\left(\sum_j q_j \frac{a_j x_j}{u_j}\right) \leq -\sum_j q_j \log\left(\frac{a_j x_j}{u_j}\right)$$

where we set $q_j = \frac{a_j x_j^t}{\sum_k a_k x_k^t}$. Finally, by continuity, we obtain the 3rd and 4th properties of majorizing functions. □

We now proceed to majorize $s_L(\mathbf{x})$, $s_R(\mathbf{x})$, and $s_C(x_j)$.

Lemma 4 (Lipschitz-majorization). *Given $s_L(\mathbf{x})$ a gradient Lipschitz function with constant σ_L over the domain $\mathbf{x} \in \mathcal{C}$. The functions*

$$g_1(\mathbf{x}, \mathbf{x}^t) = s_L(\mathbf{x}^t) + (\mathbf{x} - \mathbf{x}^t)^\top \nabla s_L(\mathbf{x}^t) + \sigma_L \|\mathbf{x} - \mathbf{x}^t\|_2^2 \quad (11)$$

and

$$g_2(\mathbf{x}, \mathbf{x}^t) = s_L(\mathbf{x}^t) + (\mathbf{x} - \mathbf{x}^t)^\top \nabla s_L(\mathbf{x}^t) + 2\sigma_L \left(\max_j x_j^t \right) \left(\sum_j x_j^t \log \left(\frac{x_j^t}{x_j} \right) - x_j^t + x_j \right) \quad (12)$$

are first order majorizing functions at $\mathbf{x}^t \in \mathcal{C}$.

Proof. First it can be trivially observed that

$$s_L(\mathbf{x}^t) = g_1(\mathbf{x}^t, \mathbf{x}^t) = g_2(\mathbf{x}^t, \mathbf{x}^t),$$

which satisfies the first property. We then take 1st order Taylor expansion of s_L around $\mathbf{x}^t$ and find

$$s_L(\mathbf{x}) = s_L(\mathbf{x}^t) + (\mathbf{x} - \mathbf{x}^t)^\top \nabla s_L(\mathbf{x}^t) + \mathcal{R}(\mathbf{x}, \mathbf{x}^t)$$

where $\mathcal{R}(\mathbf{x}, \mathbf{x}^t) \leq \sigma_L \|\mathbf{x} - \mathbf{x}^t\|_2^2$ since the function s_L is gradient Lipschitz with constant σ_L . Therefor we have

$$\begin{aligned} s_L(\mathbf{x}) &\leq s_L(\mathbf{x}^t) + (\mathbf{x} - \mathbf{x}^t)^\top \nabla s_L(\mathbf{x}^t) + \sigma_L \|\mathbf{x} - \mathbf{x}^t\|_2^2 = g_1(\mathbf{x}, \mathbf{x}^t) \\ &\leq s_L(\mathbf{x}^t) + (\mathbf{x} - \mathbf{x}^t)^\top \nabla s_L(\mathbf{x}^t) + 2\sigma_L \left(\max_j x_j^t \right) \left(\sum_j x_j^t \log \left(\frac{x_j^t}{x_j} \right) - x_j^t + x_j \right) \\ &= g_2(\mathbf{x}, \mathbf{x}^t), \end{aligned} \quad (27)$$

where (27) will be shown later in this proof. By continuity, we obtain the 3rd and 4th property of majorizing functions. We now need to prove (27) and reformulate it as

$$\|\mathbf{x} - \mathbf{x}^t\|_2^2 \leq 2 \left(\max_j x_j^t \right) \left(\sum_j x_j^t \log \left(\frac{x_j^t}{x_j} \right) - x_j^t + x_j \right) = 2 \left(\max_j x_j^t \right) D_{GKL}(\mathbf{x} \parallel \mathbf{x}^t). \quad (28)$$

For simplicity, let us define the function

$$q(\mathbf{x}) = \sum_j x_j \log(x_j),$$

with the gradient $\nabla_{x_j} q(\mathbf{x}) = \log(x_j) + 1$ and the Hessian

$$\mathbf{H}_{ij}^q(\mathbf{x}) = \frac{\partial q}{\partial x_i \partial x_j}(\mathbf{x}) = \begin{cases} \frac{1}{x_j} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Note that q is a strictly convex function for $\mathbf{x} > 0$. We expand the generalized KL divergence:

$$\begin{aligned} D_{GKL}(\mathbf{x} \parallel \mathbf{x}^t) &= \sum_j x_j^t \log(x_j^t) - \sum_j x_j^t \log(x_j) - \sum_j x_j^t + \sum_j x_j \\ &= \sum_j x_j^t \log(x_j^t) - \sum_j x_j \log(x_j) - \sum_j (\log(x_j) + 1)(x_j^t - x_j) \\ &= q(\mathbf{x}^t) - \left(q(\mathbf{x}) + \nabla q(\mathbf{x})^\top (\mathbf{x}^t - \mathbf{x}) \right) \\ &= \frac{1}{2} (\mathbf{x}^t - \mathbf{x})^\top \mathbf{H}^q(\tilde{\mathbf{x}}) (\mathbf{x}^t - \mathbf{x}), \end{aligned} \quad (29)$$

where $\tilde{\mathbf{x}}$ is selected such that the last equality holds. Since q is a strictly convex function, we know that $\tilde{\mathbf{x}} = \rho \mathbf{x} + (1 - \rho) \mathbf{x}^t$ for some $\rho \in [0, 1]$. Now we bound the Hessian as

$$\mathbf{H}^q(\mathbf{x}) \geq \frac{1}{\max_j x_j} \mathbf{I}$$

and introducing this inequality in (29), we obtain

$$D_{GKL}(\mathbf{x} \parallel \mathbf{x}^t) \geq \frac{1}{2 \max_j x_j} \|\mathbf{x} - \mathbf{x}^t\|_2^2,$$

which is equivalent to (28) and completes the proof. $\square$

Lemma 5 (Relative smoothness majorization). *Assuming $s_R(\mathbf{x})$ a σ_R relatively smooth function with respect to $\kappa(\mathbf{x}) = -\mathbf{1}^\top \log(\mathbf{x})$ for $\mathbf{x} \in \mathcal{C} \subset \mathbb{R}_+^n$. Then the function*

$$g(\mathbf{x}, \mathbf{x}^t) = s_R(\mathbf{x}^t) + \langle \nabla s_R(\mathbf{x}^t), \mathbf{x} - \mathbf{x}^t \rangle + \sigma_R \sum_i^n \left(\frac{x_i}{x_i^t} - \log\left(\frac{x_i}{x_i^t}\right) - 1 \right) \quad (13)$$

is a first order majorizing function of $s_R(\mathbf{x})$ for $\mathbf{x}^t \in \mathcal{C}$.

Proof. The first property $g(\mathbf{x}^t, \mathbf{x}^t) = s_R(\mathbf{x}^t)$ can be trivially verified. Then using by the definition of relatively smooth function

$$s_R(\mathbf{x}) \leq s_R(\mathbf{x}^t) + \langle \nabla s_R(\mathbf{x}^t), \mathbf{x} - \mathbf{x}^t \rangle + \sigma_R \mathcal{B}_\kappa(\mathbf{x}, \mathbf{x}^t)$$

where

$$\mathcal{B}_\kappa(\mathbf{x}, \mathbf{x}^t) := \kappa(\mathbf{x}) - \kappa(\mathbf{x}^t) - \langle \nabla \kappa(\mathbf{x}^t), \mathbf{x} - \mathbf{x}^t \rangle.$$

Given that $\kappa(\mathbf{x}) = -\mathbf{1}^\top \log(\mathbf{x})$, we compute

$$\mathcal{B}_\kappa(\mathbf{x}, \mathbf{x}^t) = \sum_i^n \left(\frac{x_i}{x_i^t} - \log\left(\frac{x_i}{x_i^t}\right) - 1 \right)$$

Finally, by continuity, we obtain the 3rd and 4th property of majorizing functions. $\square$

Lemma 6 (Concave majorisation). *Given $s(x)$ a concave function defined on $x \in \mathcal{C} \subset \mathbb{R}$, it's linear approximation at the point x^t*

$$g(x, x^t) = s(x^t) + \frac{\partial s(x^t)}{\partial x} (x - x^t) \quad (14)$$

is a first order majorization function for $x^t \in \mathcal{C}$.

Proof. One can simply observe $s(x^t) = g(x^t, x^t)$. Then by concavity, we have

$$s(x_i) \leq s(x_i^t) + \frac{\partial s(x_i^t)}{\partial x_i} (x_i - x_i^t)$$

Finally, by continuity, we obtain the 3rd and 4th property of majorizing functions. $\square$

A.3 Proof of subproblem updates

In this subsection, we present the proof of the subproblem updates used in the MU and QU algorithms. Let us start with the MU updates.

Proposition 1 (Generalized MU for (10)). *Assuming $\mathbf{x}^t, \mathbf{x}, \mathbf{b}, \mathbf{A} > 0$, the first-order majorizing function defined in (15) is strictly convex, and its global minimum $\mathbf{x}^{t+1}$ is given by*

$$x_j^{t+1} = x_j^t \frac{\alpha_j^t}{\beta_j^t} \quad (16)$$

where

$$\alpha_j^t = \sum_i b_i \frac{a_{ij}}{\sum_k a_{ik} x_k^t} + 2 \left(\max_i x_i^t \right) \sigma_L + \frac{\sigma_R}{x_j^t} \quad \text{and} \quad (17)$$

$$\beta_j^t = \sum_i a_{ij} + \nabla_{x_j} s_L(\mathbf{x}^t) + 2 \left(\max_i x_i^t \right) \sigma_L + \nabla_{x_j} s_R(\mathbf{x}^t) + \frac{\sigma_R}{x_j^t} + \frac{\partial s_C(x_j^t)}{\partial x}. \quad (18)$$

Proof. Assuming $a_{ij}, b_i > 0$, (10) is strictly convex, as the green term is strictly convex, and the remaining terms are convex. Consequently, (10) possesses a global minimum. To identify this minimum, we seek the stationary point $\nabla_{\mathbf{x}} g = \mathbf{0}$. Due to our meticulous selection of majorizing functions, this subproblem becomes separable. When computing the gradient with respect to the variable x_j , we obtain:

$$\begin{aligned} \nabla_{x_j} g(\mathbf{x}, \mathbf{x}^t) &= -\frac{1}{x_j} \sum_i b_i \frac{a_{ij} x_j^t}{\sum_k a_{ik} x_k^t} + \sum_i a_{ij} \\ &\quad + \nabla_{x_j} s_L(\mathbf{x}^t) + 2 \left(\max_i x_i^t \right) \sigma_L \left(1 - \frac{x_j^t}{x_j} \right) \\ &\quad + \nabla_{x_j} s_R(\mathbf{x}^t) + \sigma_R \left(\frac{1}{x_j^t} - \frac{1}{x_j} \right) \\ &\quad + \frac{\partial s_C(x_j^t)}{\partial x} = 0 \end{aligned} \quad (30)$$

where we assume that $x_j, x_j^t > 0$ since $0 \notin \mathcal{C}$. Transforming the above expression, we find a multiplicative update rule (16) for x_j . $\square$

The proof of the QU updates is similar to the MU updates, expect that we need to solve a quadratic equation to obtain a closed-form solution.

Proposition 2 (QU for (10)). *Assuming $\mathbf{x}^t, \mathbf{x}, \mathbf{b}, \mathbf{A} > 0$, the first-order majorizing function defined in Equation (19) is strictly convex, and its global minimum $\mathbf{x}^{t+1}$ is given by*

$$x_j^{t+1} = \frac{-\beta_j^t + \sqrt{(\beta_j^t)^2 + 4\alpha\zeta_j^t}}{2\alpha} \quad (20)$$

where

$$\alpha = 2\sigma_L \quad \beta_j^t = \sum_i a_{ij} + \nabla_{x_j} s_L(\mathbf{x}^t) - 2\sigma_L x_j^t + \nabla_{x_j} s_R(\mathbf{x}^t) + \frac{\sigma_R}{x_j^t} + \frac{\partial s_C(x_j^t)}{\partial x} \quad \zeta_j^t = \sum_i b_i \frac{a_{ij} x_j^t}{\sum_k a_{ik} x_k^t} + \sigma_R. \quad (21)$$

Proof. Assuming $a_{ij}, b_i > 0$, Equation (19) is strictly convex. This convexity arises from the strict convexity of the green term, coupled with the convexity of the other terms. Consequently, (19) possesses a global minimum. To identify this minimum, we seek the stationary point by computing $\nabla_{\mathbf{x}} g = \mathbf{0}$:

$$\begin{aligned} \nabla_{x_j} g(\mathbf{x}, \mathbf{x}^t) &= -\frac{1}{x_j} \sum_i b_i \frac{a_{ij} x_j^t}{\sum_k a_{ik} x_k^t} + \sum_i a_{ij} \\ &\quad + \nabla_{x_j} s_L(\mathbf{x}^t) + 2\sigma_L (x_j - x_j^t) \\ &\quad + \nabla_{x_j} s_R(\mathbf{x}^t) + \sigma_R \left(\frac{1}{x_j^t} - \frac{1}{x_j} \right) \\ &\quad + \frac{\partial s_C(x_j^t)}{\partial x} = 0 \end{aligned} \quad (31)$$

We observe that it is a separable quadratic function, hence the update rule named Quadratic Update (QU). Solving (31) for $x_j > 0$ can be rewritten as

$$\alpha x_j^2 + \beta_j^t x_j - \zeta_j^t = 0,$$

where α , β_j^t , and ζ_j^t are given in (21). Assuming $\zeta_j^t \neq 0$, we have $\zeta_j^t > 0$ and $4\alpha\zeta_j^t > 0$. Therefore, the previous quadratic equation has two real solutions. Since $\sqrt{(\beta_j^t)^2 + 4\alpha\zeta_j^t} > \beta_j^t$, they are of opposite sign. Due to the constraint $x_j \geq \epsilon > 0$, we select the positive one, leading to the update rule of (20) for x_j . $\square$

B Computation of Lower and Upper Bounds for Dichotomy

In Section 4.3, we introduced modifications to the MU and QU algorithms to incorporate the positivity constraint $\mathbf{x} \geq \epsilon$ and the linear constraint $\mathbf{e}^\top \mathbf{x} = 1$. However, solving for the dual parameter ν in equations (24) for MU or (25) for QU is intractable. To address this, we propose using the dichotomy method to solve for $h(\nu) = 0$. Therefore, this appendix provides the computation of lower bound ν_{low} and upper bound ν_{up} such that $h(\nu_{\text{low}}) < 0$ and $h(\nu_{\text{up}}) > 0$. These bounds will serve as convenient initializations for the dichotomy algorithm.

B.1 Case 1: MU

For MU, we aim to solve equation (24) for ν :

$$h_1(\nu) = \sum_j e_j \frac{x_j^t \alpha_j}{\min\left(\frac{x_j^t \alpha_j}{\epsilon}, \beta_j^t + \nu e_j\right)} - 1 = 0$$

Terms where $e_j = 0$ can be ignored since they do not contribute to the sum. Assuming $\epsilon > 0$, the function h_1 is well-defined for $\nu \in \mathbb{R}$. Since $x_j^t \alpha_j > 0$, we have $h_1(\nu) = \frac{\|\mathbf{e}\|_1}{\epsilon} - 1$ for $\nu \leq \nu_{\text{lim}} = \min_j \left(\frac{\frac{x_j^t \alpha_j}{\epsilon} - \beta_j^t}{e_j} \right)$, and h_1 is monotonically decreasing for $\nu \in [\nu_{\text{lim}}, \infty[$. Assuming $\frac{\|\mathbf{e}\|_1}{\epsilon} - 1 \geq 0$ (which ensures the feasibility of the constraints $\mathbf{x} \geq \epsilon$ and $\mathbf{e}^\top \mathbf{x} = 1$), we have $h_1(\nu_{\text{lim}}) \geq 0$ and $\lim_{\nu \rightarrow \infty} h_1(\nu) = -1$. Thus, there exists exactly one root for the function h_1 .

Negative bound First, let's find ν_{low} such that $h_1(\nu) < 0$ for $\nu_{\text{low}} \leq \nu < \infty$. We can bound h_1 as follows:

$$\begin{aligned} h_1(\nu) &= \sum_j \frac{x_j^t \alpha_j}{\min\left(\frac{x_j^t \alpha_j}{\epsilon e_j}, \frac{\beta_j^t}{e_j} + \nu\right)} - 1 \\ &\leq n \frac{\max_j x_j^t \alpha_j}{\min_j \min\left(\frac{x_j^t \alpha_j}{\epsilon e_j}, \frac{\beta_j^t}{e_j} + \nu\right)} - 1 < 0 \end{aligned}$$

Therefore one possible bound is

$$\nu_{\text{low}} = n \max_j x_j^t \alpha_j - \min_j \frac{\beta_j^t}{e_j}.$$

Positive bound Similarly, let's find ν_{up} such that $h_1(\nu) \geq 0$ for $\nu_{\text{up}} \geq \nu \geq \nu_{\text{lim}}$. Note that ν_{lim} is not a good bound when ϵ is small. We can bound h_1 as follows:

$$\begin{aligned} h_1(\nu) &= \sum_{j=1}^n \frac{x_j^t \alpha_j}{\min\left(\frac{x_j^t \alpha_j}{\epsilon e_j}, \frac{\beta_j^t}{e_j} + \nu\right)} - 1 \\ &\geq \max_j \frac{x_j^t \alpha_j}{\min\left(\frac{x_j^t \alpha_j}{\epsilon e_j}, \frac{\beta_j^t}{e_j} + \nu\right)} - 1 \\ &\geq \max_j \frac{x_j^t \alpha_j}{\frac{\beta_j^t}{e_j} + \nu} - 1 > 0, \end{aligned}$$

As a result, we have

$$\nu_{\text{up}} = \max_j \left(x_j^t \alpha_j - \frac{\beta_j^t}{e_j} \right)$$

To improve numerical stability, one could use $\nu'_{\text{low}} = 2n \max_j x_j^t \alpha_j - \min_j \frac{\beta_j^t}{e_j}$ and $\nu'_{\text{up}} < \max_j \frac{x_j^t \alpha_j}{2} - \frac{\beta_j^t}{e_j}$.

B.2 Case 2: QU

For QU, we aim to solve equation (25) for ν :

$$h_2(\nu) = \sum_j e_j \max \left(\frac{-\beta_j^t - \nu e_j + \sqrt{(\beta_j^t + \nu e_j)^2 + 4\alpha \zeta_j^t}}{2\alpha}, \epsilon \right) - 1 = 0.$$

Let's analyze the function h_2 . We observe that the function $-x + \sqrt{x^2 + \delta}$ is strictly decreasing over $\mathbb{R}$ for $\delta > 0$, since its derivative $-1 + \frac{x}{\sqrt{x^2 + \delta}}$ is strictly negative for $\delta > 0$. Therefore, each term of the sum is decreasing, and h_2 is a decreasing function. Note that once at least for one j , we have

$$-\beta_j^t - \nu e_j + \sqrt{(\beta_j^t + \nu e_j)^2 + 4\alpha \zeta_j^t} \geq 2\alpha\epsilon,$$

and therefore the function h_2 becomes strictly decreasing. In the limit, we have $\lim_{\nu \rightarrow -\infty} h_2(\nu) = \infty$ and $\lim_{\nu \rightarrow \infty} h_2(\nu) = \epsilon \|\mathbf{e}\|_1 - 1$. Assuming $\epsilon \|\mathbf{e}\|_1 > 1$ (which ensures the feasibility of the constraints $\mathbf{x} \geq \epsilon$ and $\mathbf{e}^\top \mathbf{x} = 1$), we know that the function h_2 has exactly one root.

Negative bound Let's find ν_{low} such that $h_2(\nu) \leq 0$ for $\nu \geq \nu_{\text{low}}$. We start by bounding the term

$$-\beta_j^t - \nu e_j + \sqrt{(\beta_j^t + \nu e_j)^2 + 4\alpha \zeta_j^t} \leq \delta_j,$$

where $\delta_j > \epsilon > 0$. We move $-\beta_j^t - \nu e_j$ to the left and square the inequality to remove the square root:

$$(\beta_j^t + \nu e_j)^2 + 4\alpha \zeta_j^t \leq (\beta_j^t + \nu e_j + \epsilon_j)^2$$

Eventually, we can extract a bound for ν to ensure that the inequality is satisfied for a chosen δ_j :

$$\nu \geq \frac{4\alpha \zeta_j^t - \epsilon_j^2}{2\delta_j e_j} - \frac{\beta_j^t}{e_j}.$$

Let's set $\delta_j = \frac{2\alpha}{m e_j}$, where m is the number of elements in the sum, and take the maximum over j to obtain the bound:

$$\nu_{\text{low}} \geq \max_j \left(m \zeta_j^t - \frac{\alpha}{m e_j^2} - \frac{\beta_j^t}{e_j} \right)$$

We observe the validity of this bound provided that $\delta_j = \frac{2\alpha}{m e_j} \geq \epsilon$ for all j . Then, it can be verified that

$$\begin{aligned} h_2(\nu_{\text{low}}) &= \sum_j e_j \max \left(\frac{-\beta_j^t - \nu_{\text{low}} e_j + \sqrt{(\beta_j^t + \nu_{\text{low}} e_j)^2 + 4\alpha \zeta_j^t}}{2\alpha}, \epsilon \right) - 1 \\ &= \sum_j e_j \frac{-\beta_j^t - \nu_{\text{low}} e_j + \sqrt{(\beta_j^t + \nu_{\text{low}} e_j)^2 + 4\alpha \zeta_j^t}}{2\alpha} - 1 \\ &\leq \sum_j e_j \frac{\delta_j}{2\alpha} - 1 \\ &= \sum_j e_j \frac{\frac{2\alpha}{m e_j}}{2\alpha} - 1 = 0, \end{aligned}$$

which proves that ν_{low} is a valid negative bound.

Positive bound Similarly, let's find ν_{up} such that $h_2(\nu) \geq 0$ for $\nu \leq \nu_{\text{up}}$. This time, we will bound h_2 from below and obtain

$$\begin{aligned}
h_2(\nu) &= \sum_j e_j \max \left(\frac{-\beta_j^t - \nu e_j + \sqrt{(\beta_j^t + \nu e_j)^2 + 4\alpha\zeta_j^t}}{2\alpha}, \epsilon \right) - 1 \\
&\geq \sum_j e_j \frac{-\beta_j^t - \nu e_j + \sqrt{(\beta_j^t + \nu e_j)^2 + 4\alpha\zeta_j^t}}{2\alpha} - 1 \\
&\geq \frac{1}{2\alpha} \sum_j e_j (-\beta_j^t - \nu e_j) - 1 \\
&= -\frac{1}{2\alpha} \left(\sum_j e_j \beta_j^t + \nu \sum_j e_j^2 \right) - 1 > 0.
\end{aligned}$$

We can, therefore, define

$$\nu_{\text{up}} = -\frac{2\alpha + \sum_j e_j \beta_j^t}{\sum_j e_j^2}.$$

References

- [1] Hedy Attouch, Jérôme Bolte, Patrick Redont, and Antoine Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyk-lojasiewicz inequality. *Mathematics of operations research*, 35(2):438–457, 2010.
- [2] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1):459–494, 2014.
- [3] Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.
- [4] Jean-Philippe Brunet, Pablo Tamayo, Todd R Golub, and Jill P Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the national academy of sciences*, 101(12):4164–4169, 2004.
- [5] Stefania Cacovich, Fabio Matteocci, Mojtaba Abdi-Jalebi, Samuel D Stranks, Aldo Di Carlo, Caterina Ducati, and Giorgio Divitini. Unveiling the chemical composition of halide perovskite films using multivariate statistical analyses. *ACS Applied Energy Materials*, 1(12):7174–7181, 2018.
- [6] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1548–1560, 2010.
- [7] Andrzej Cichocki and Anh-Huy Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 92(3):708–721, 2009.
- [8] Andrzej Cichocki, Rafal Zdunek, and Shun-ichi Amari. Hierarchical als algorithms for nonnegative matrix and 3d tensor factorization. In *International Conference on Independent Component Analysis and Signal Separation*, pages 169–176. Springer, 2007.
- [9] Yu-Hong Dai and Yaxiang Yuan. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM Journal on optimization*, 10(1):177–182, 1999.
- [10] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis. *Neural computation*, 21(3):793–830, 2009.

- [11] Dan Fu, Gary Holtom, Christian Freudiger, Xu Zhang, and Xiaoliang Sunney Xie. Hyperspectral imaging with stimulated raman scattering by chirped femtosecond lasers. *The Journal of Physical Chemistry B*, 117(16):4634–4640, 2013.
- [12] Nicolas Gillis. The why and how of nonnegative matrix factorization. In *Regularization, Optimization, Kernels, and Support Vector Machines*, pages 275–310. Chapman and Hall/CRC, 2014.
- [13] Prem Gopalan, Jake M Hofman, and David M Blei. Scalable recommendation with poisson factorization. *arXiv preprint arXiv:1311.1704*, 2013.
- [14] Filip Hanzely, Peter Richtarik, and Lin Xiao. Accelerated bregman proximal gradient methods for relatively smooth convex optimization. *Computational Optimization and Applications*, 79:405–440, 2021.
- [15] Niao He, Zaid Harchaoui, Yichen Wang, and Le Song. Fast and simple optimization for poisson likelihood models. *arXiv preprint arXiv:1608.01264*, 2016.
- [16] Le Thi Khanh Hien and Nicolas Gillis. Algorithms for nonnegative matrix factorization with the kullback–leibler divergence. *Journal of Scientific Computing*, 87(3):1–32, 2021.
- [17] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.
- [18] Cho-Jui Hsieh and Inderjit S Dhillon. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1064–1072, 2011.
- [19] BR Jany, Arkadiusz Janas, and Franciszek Krok. Retrieving the quantitative chemical information at nanoscale from scanning electron microscope energy dispersive x-ray measurements by machine learning. *Nano letters*, 17(11):6520–6525, 2017.
- [20] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1724–1732. PMLR, 2017.
- [21] Ramakrishnan Kannan, AV Ievlev, Nouamane Laanait, Maxim A Ziatdinov, Rama K Vasudevan, Stephen Jesse, and Sergei V Kalinin. Deep data analysis via physically constrained linear unmixing: universal framework, domain examples, and a community-wide platform. *Advanced Structural and Chemical Imaging*, 4(1):1–20, 2018.
- [22] Hideaki Kano, Hiroki Segawa, Masanari Okuno, Philippe Leproux, and Vincent Couderc. Hyperspectral coherent raman imaging—principle, theory, instrumentation, and applications to life sciences. *Journal of Raman Spectroscopy*, 47(1):116–123, 2016.
- [23] Hyunsoo Kim and Haesun Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM journal on matrix analysis and applications*, 30(2):713–730, 2008.
- [24] Jingu Kim, Yunlong He, and Haesun Park. Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework. *Journal of Global Optimization*, 58(2):285–319, 2014.
- [25] Jingu Kim and Haesun Park. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, 33(6):3261–3281, 2011.
- [26] Nikos Komodakis and Jean-Christophe Pesquet. Playing with duality: An overview of recent primal? dual approaches for solving large-scale optimization problems. *IEEE Signal Processing Magazine*, 32(6):31–54, 2015.
- [27] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

- [28] Paul G Kotula, Michael R Keenan, and Joseph R Michael. Automated analysis of sem x-ray spectral images: A powerful new microanalysis tool. *Microscopy and Microanalysis*, 9(1):1–17, 2003.
- [29] Daniel Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 556–562. MIT Press, 2001.
- [30] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.
- [31] Qiuwei Li, Zhihui Zhu, Gongguo Tang, and Michael B Wakin. Provable bregman-divergence based methods for nonconvex and non-lipschitz problems. *arXiv preprint arXiv:1904.09712*, 2019.
- [32] Xinghua Li, Liyuan Wang, Qing Cheng, Penghai Wu, Wenxia Gan, and Lina Fang. Cloud removal in remote sensing images using nonnegative matrix factorization and error correction. *ISPRS journal of photogrammetry and remote sensing*, 148:103–113, 2019.
- [33] Chih-Jen Lin. On the convergence of multiplicative update algorithms for nonnegative matrix factorization. *IEEE Transactions on Neural Networks*, 18(6):1589–1596, 2007.
- [34] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.
- [35] Haihao Lu, Robert M Freund, and Yurii Nesterov. Relatively smooth convex optimization by first-order methods, and applications. *SIAM Journal on Optimization*, 28(1):333–354, 2018.
- [36] Xiaoqiang Lu, Hao Wu, Yuan Yuan, Pingkun Yan, and Xuelong Li. Manifold regularized sparse nmf for hyperspectral unmixing. *IEEE Transactions on Geoscience and Remote Sensing*, 51(5):2815–2826, 2012.
- [37] Julien Mairal. Incremental majorization-minimization optimization with application to large-scale machine learning. *SIAM Journal on Optimization*, 25(2):829–855, 2015.
- [38] Qiaozhu Mei and ChengXiang Zhai. Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 198–207, 2005.
- [39] Pentti Paatero. Least squares formulation of robust non-negative factor analysis. *Chemometrics and intelligent laboratory systems*, 37(1):23–35, 1997.
- [40] Ioannis Panageas, Georgios Piliouras, and Xiao Wang. First-order methods almost always avoid saddle points: The case of vanishing step-sizes. *Advances in Neural Information Processing Systems*, 32, 2019.
- [41] Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.
- [42] Clément W Royer, Michael O’Neill, and Stephen J Wright. A newton-cg algorithm with complexity guarantees for smooth unconstrained optimization. *Mathematical Programming*, 180(1):451–488, 2020.
- [43] Clément W Royer and Stephen J Wright. Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1448–1477, 2018.
- [44] Joseph Salmon, Zachary Harmany, Charles-Alban Deledalle, and Rebecca Willett. Poisson noise reduction with non-local pca. *Journal of mathematical imaging and vision*, 48(2):279–294, 2014.
- [45] Motoki Shiga, Kazuyoshi Tatsumi, Shunsuke Muto, Koji Tsuda, Yuta Yamamoto, Toshiyuki Mori, and Takayoshi Tanji. Sparse modeling of eels and edx spectral imaging data by nonnegative matrix factorization. *Ultramicroscopy*, 170:43–59, 2016.

- [46] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658, 2008.
- [47] Paris Smaragdis and Judith C Brown. Non-negative matrix factorization for polyphonic music transcription. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No. 03TH8684)*, pages 177–180. IEEE, 2003.
- [48] Dennis L Sun and Cedric Fevotte. Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6201–6205. IEEE, 2014.
- [49] Leo Taslaman and Björn Nilsson. A framework for regularized non-negative matrix factorization, with application to the analysis of gene expression data. *PloS one*, 7(11):e46331, 2012.
- [50] Marc Teboulle and Yakov Vaisbourd. Novel proximal gradient methods for nonnegative matrix factorization with sparsity constraints. *SIAM Journal on Imaging Sciences*, 13(1):381–421, 2020.
- [51] Adrien Teurtrie, Nathanaël Perraudin, Thomas Holvoet, Hui Chen, Duncan TL Alexander, Guillaume Obozinski, and Cécile Hébert. espm: A python library for the simulation of stem-edxs datasets. *Ultramicroscopy*, page 113719, 2023.
- [52] Adrien Teurtrie, Nathanaël Perraudin, Thomas Holvoet, Hui Chen, Duncan TL Alexander, Guillaume Obozinski, and Cécile Hébert. From stem-edxs data to phase separation and quantification using physics-guided nmf. *To appear*, 2024.
- [53] Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109:475–494, 2001.
- [54] Musundi B Wabuyele, Fei Yan, Guy D Griffin, and Tuan Vo-Dinh. Hyperspectral surface-enhanced raman imaging of labeled silver nanoparticles in single cells. *Review of scientific instruments*, 76(6):063710, 2005.
- [55] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, 6(3):1758–1789, 2013.
- [56] Felipe Yanez and Francis Bach. Primal-dual algorithms for non-negative matrix factorization with the kullback-leibler divergence. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2257–2261. IEEE, 2017.
- [57] Andrew B Yankovich, Chenyu Zhang, Albert Oh, Thomas JA Slater, Feridoon Azough, Robert Freer, Sarah J Haigh, Rebecca Willett, and Paul M Voyles. Non-rigid registration and non-local principle component analysis to improve electron microscopy spectrum images. *Nanotechnology*, 27(36):364001, 2016.
- [58] Minchao Ye, Yuntao Qian, and Jun Zhou. Multitask sparse nonnegative matrix factorization for joint spectral-spatial hyperspectral imagery denoising. *IEEE Transactions on Geoscience and Remote Sensing*, 53(5):2621–2639, 2014.
- [59] Chenyu Zhang, Rungang Han, Anru R Zhang, and Paul M Voyles. Denoising atomic resolution 4d scanning transmission electron microscopy data with tensor singular value decomposition. *Ultramicroscopy*, 219:113123, 2020.
- [60] Changzhong Zou and Youshen Xia. Restoration of hyperspectral image contaminated by poisson noise using spectral unmixing. *Neurocomputing*, 275:430–437, 2018.