An Iterative Algorithm for Regularized Non-negative Matrix Factorizations

Steven E. Pav [*]

November 4, 2024

Abstract

We generalize the non-negative matrix factorization algorithm of Lee and Seung [2] to accept a weighted norm, and to support ridge and Lasso regularization. We recast the Lee and Seung multiplicative update as an additive update which does not get stuck on zero values.

1 Introduction

The term *non-negative matrix factorization* refers to decomposing a given matrix Y with non-negative elements approximately as the product $Y \approx LR$, for comfortable non-negative matrices L and R of smaller rank than Y. This factorization represents a kind of dimensionality reduction. The quality of the approximation is measured by some "divergence" between Y and LR, or some norm of their difference. Perhaps the most common is the Frobenius (or elementwise ℓ_2) distance between them. Under this objective, non-negative matrix factorization can be expressed as the optimization problem:

$$\min_{L \geq 0,\, R \geq 0} \operatorname{tr}\left((Y - LR)^\top (Y - LR) \right), \tag{1}$$

where $\operatorname{tr}(M)$ is the trace of the matrix M, and $A \geq 0$ means the elements of A are non-negative. [8]

Lee and Seung described an iterative approach to this problem, as well as for a related optimization with a different objective. [2] Their algorithm starts with some unspecified initial iterates $L^{(0)}$ and $R^{(0)}$, then iteratively improves $L^{(k)}$ and $R^{(k)}$ in turn. That is, given $L^{(k)}$ and $R^{(k)}$, an update is made to arrive at $L^{(k+1)}$; then using $L^{(k+1)}$ and $R^{(k)}$, one computes $R^{(k+1)}$, and the process repeats. The iterative update for $L^{(k+1)}$ is

$$L^{(k+1)} \leftarrow L^{(k)} \odot YR^{(k)\top} \oslash \left(L^{(k)}R^{(k)}R^{(k)\top} \right). \tag{2}$$

We use $\odot$ to mean the Hadamard, or 'elementwise', multiplication, and $\oslash$ to mean Hadamard division. The iterative update for $R^{(k+1)}$ is defined *mutatis mutandis*. As should be apparent from inspection, this iterative update maintains non-negativity of estimates. That is, if Y, $L^{(k)}$, and $R^{(k)}$ have only positive

---

[*]shabbychef@gmail.com Application of the multiplicative methods described herein for chemometrics applications may be covered by U.S. Patent. [6]

elements, then $\mathsf{L}^{(k+1)}$ is finite and has only positive elements. It is not clear what should be done if elements of the denominator term $\mathsf{L}^{(k)}\,\mathsf{R}^{(k)}\mathsf{R}^{(k)^\top}$ are zero, and whether and how this can be avoided.

Gonzalez and Zhang describe a modification of the update to accelerate convergence. [1] Their accelerated algorithm updates $\mathsf{L}^{(k)}$ row by row, and $\mathsf{R}^{(k)}$ column by column, using the step length modification of Merritt and Zhang [4].

One attractive feature of the Lee and Seung iterative update is its sheer simplicity. In a high-level computer language with a rich set of matrix operations, such as Julia or octave, the iterative step can be expressed in a single line of code; the entire algorithm could be expressed in perhaps 20 lines of code. Conceivably this algorithm could even be implemented in challenged computing environments, like within a SQL query or a spreadsheet macro language. Simplicity makes this algorithm an obvious choice for experimentation without requiring an all-purpose constrained quadratic optimization solver, or other complicated software.

2 Regularized Non-Negative Matrix Factorization

We generalize the optimization problem of 1 to include a weighted norm, regularization terms, and orthogonality penalties. For a given Y, and non-negative weighting matrices $\mathsf{W}_{0,R}$, $\mathsf{W}_{0,C}$, $\mathsf{W}_{1,L}$, $\mathsf{W}_{1,R}$, $\mathsf{W}_{2,R,L,j}$, $\mathsf{W}_{2,C,L,j}$, $\mathsf{W}_{2,R,R,j}$, and $\mathsf{W}_{2,C,R,j}$, find matrices L and R to minimize

$$
\begin{aligned}
\min_{\mathsf{L}\geq 0,\,\mathsf{R}\geq 0} \quad & \frac{1}{2}\operatorname{tr}\left((\mathsf{Y}-\mathsf{L}\mathsf{R})^\top \mathsf{W}_{0,R}\left(\mathsf{Y}-\mathsf{L}\mathsf{R}\right)\mathsf{W}_{0,C}\right) \quad\quad (3)\\
& + \operatorname{tr}\left(\mathsf{W}_{1,L}^{\top}\mathsf{L}\right) + \operatorname{tr}\left(\mathsf{W}_{1,R}^{\top}\mathsf{R}\right)\\
& + \frac{1}{2}\sum_{1\leq j\leq J}\operatorname{tr}\left(\mathsf{L}^\top \mathsf{W}_{2,R,L,j}\mathsf{L}\mathsf{W}_{2,C,L,j}\right)\\
& + \frac{1}{2}\sum_{1\leq j\leq J}\operatorname{tr}\left(\mathsf{R}^\top \mathsf{W}_{2,R,R,j}\mathsf{R}\mathsf{W}_{2,C,R,j}\right).
\end{aligned}
$$

Essentially we are trying to model Y as $\mathsf{L}\mathsf{R}$. Our loss function is quadratic in the error $\mathsf{Y}-\mathsf{L}\mathsf{R}$, with weights $\mathsf{W}_{0,R}$ and $\mathsf{W}_{0,C}$, and ℓ_1 and ℓ_2 regularization via the terms with W_1 and $\mathsf{W}_{2,R,j}$ and $\mathsf{W}_{2,C,j}$. This optimization problem includes that of 1 as a special case.

We note that a non-orthogonality penalty term on the X can be expressed as

$$
\frac{c}{2}\operatorname{tr}\left(\mathsf{X}^\top \mathsf{I}\mathsf{X}\left(\mathbf{1}\mathbf{1}^\top-\mathsf{I}\right)\right).
$$

Thus a non-orthogonality penalty term can be expressed with the $\mathsf{W}_{2,R,j}$ and $\mathsf{W}_{2,C,j}$. Indeed the motivation for including a sum of the ℓ_2 terms is to allow for both straightforward regularization, where $\mathsf{W}_{2,R,j}=\mathsf{I}$ and $\mathsf{W}_{2,C,j}=c\mathsf{I}$, and orthogonality terms. For simplicity of notation, in the discussion that follows we will often treat the ℓ_2 terms as if there were only one of them, $i.e.$, $J=1$, and omit the j subscripts. Restoring the more general case consists only of expanding to a full sum of J terms.

This problem formulation is appropriate for power users who can meaningfully select the various weighting matrices. And while it is easy to conceive of a use for general forms for $W_{0,R}$ and $W_{0,C}$ (the rows of Y have different importances, or the columns of Y are observed with different accuracies, for example), most users would probably prefer simpler formulations for the $W_{1,\cdot}$ and $W_{2,\cdot,\cdot,j}$ matrices. To achieve something like an elasticnet factorization with non-orthogonality penalties, one should set $J = 2$ and take

$$
\begin{aligned}
W_{1,L} &= \lambda_{1,L}1, & W_{1,R} &= \lambda_{1,R}1, \\
W_{2,R,L,1} &= \lambda_{2,L}I, & W_{2,C,L,1} &= I, \\
W_{2,R,R,1} &= \lambda_{2,R}I, & W_{2,C,R,1} &= I, & (4)\\
W_{2,R,L,2} &= I, & W_{2,C,L,2} &= \gamma_{2,L}\left(11^{\top} - I\right), \\
W_{2,R,R,2} &= I, & W_{2,C,R,2} &= \gamma_{2,R}\left(11^{\top} - I\right),
\end{aligned}
$$

where here 1 stands in for an all-ones matrix of the appropriate size. All the λ and γ parameters must be non-negative, and the identity matrices are all appropriately sized. This is the reduced form of the problem that depends mostly on a few scalars. A simple form for the weighted error is to take $W_{0,R}$ and $W_{0,C}$ to be diagonal matrices,

Depending on the regularization terms, the solution to the problem of 3 is unlikely to be unique. Certain the solution to 1 is not unique, since if Q is a permutation matrix of the appropriate size, then LQ and $Q^{-1}R$ are another set of non-negative matrices with the same objective value. Perhaps the regularization terms can help avoid these kinds of ambiguity in the solution.

2.0.1 Split Form

To approximately solve this problem our algorithm starts with some initial estimates of L and R and iteratively alternates between updating the estimate of L with the estimate of R fixed, and updating the estimate of R with the estimate of L fixed. To discuss these half-step updates, we can describe both of them in a single unified formulation. So we consider the optimization problem

$$
\min_{X \geq 0} \phi\left(X\right), \tag{5}
$$

where we define the objective function

$$
\phi\left(X\right) = \frac{1}{2}\operatorname{tr}\left(\left(Y - LXR\right)^{\top}W_{0,R}\left(Y - LXR\right)W_{0,C}\right) \tag{6}
$$
$$
+ \operatorname{tr}\left(W_1{}^{\top}X\right) + \frac{1}{2}\sum_{1 \leq j \leq J}\operatorname{tr}\left(X^{\top}W_{2,R,j}XW_{2,C,j}\right).
$$

We must assume that L has full column rank and R has full row rank, $W_{0,R}$, $W_{0,C}$, $W_{2,R,j}$ and $W_{2,C,j}$ are square, symmetric and positive semidefinite. The $W_{0,R}$ has the same number of rows as Y, and $W_{0,C}$ has the same number of columns as Y. W_1 is the same size as X; we will make further restrictions on W_1 below. The matrices $W_{2,R,j}$ have the same number of rows as X, and $W_{2,C,j}$ have the same number of columns as X. All the $W_{i,\cdot}$ are assumed to have non-negative elements, that is $W_{i,\cdot} \geq 0$.

Our algorithm now consists of filling in an identity for L, and estimate of L for X, and all the L-appropriate weighting matrices into Equation 6 and taking a step to solve Problem 3, getting an update of L. Then we perform a similar operation to get a new estimate of R.

2.1 Factorization in Vector Form

The analysis is simplified if we express the problem as one of finding a vector unknown, so we will 'vectorize' the problem. For matrix M let $\mathrm{vec}\,(\mathsf{M})$ be the vector which consists of the columns of M stacked on top of each other; let $\mathsf{A}\otimes\mathsf{B}$ denote the Kronecker product of A and B. [9] See Section A in the appendix for some helpful identities involving the Kronecker product, matrix trace, and vectorization.

We write $\phi\,(\mathsf{X})$ as a quadratic function of the vector $\mathrm{vec}\,(\mathsf{X})$

$$
\begin{aligned}
\phi\,(\mathsf{X}) &= \frac{1}{2}\left\|\mathrm{vec}\,(\mathsf{Y}) - \left(\mathsf{R}^{\top}\otimes\mathsf{L}\right)\mathrm{vec}\,(\mathsf{X})\right\|^{2}_{\mathsf{W}_{0,C}\otimes\mathsf{W}_{0,R}} \\
&\quad + \frac{1}{2}\sum_{j}\left\|\mathrm{vec}\,(\mathsf{X})\right\|^{2}_{\mathsf{W}_{2,C,j}\otimes\mathsf{W}_{2,R,j}} + \mathrm{vec}\,(\mathsf{W}_{1})^{\top}\mathrm{vec}\,(\mathsf{X}) \\
&= \frac{1}{2}\mathrm{vec}\,(\mathsf{X})^{\top}\mathsf{G}\,\mathrm{vec}\,(\mathsf{X}) + \boldsymbol{d}^{\top}\mathrm{vec}\,(\mathsf{X}) + c,
\end{aligned}
\tag{7}
$$

where

$$
\begin{aligned}
\mathsf{G} &= \left(\mathsf{R}^{\top}\otimes\mathsf{L}\right)^{\top}\left(\mathsf{W}_{0,C}\otimes\mathsf{W}_{0,R}\right)\left(\mathsf{R}^{\top}\otimes\mathsf{L}\right) + \sum_{j}\mathsf{W}_{2,C,j}\otimes\mathsf{W}_{2,R,j}, \\
&= \left(\mathsf{R}\mathsf{W}_{0,C}\mathsf{R}^{\top}\right)\otimes\left(\mathsf{L}^{\top}\mathsf{W}_{0,R}\mathsf{L}\right) + \sum_{j}\mathsf{W}_{2,C,j}\otimes\mathsf{W}_{2,R,j}.
\end{aligned}
\tag{8}
$$

$$
\begin{aligned}
\boldsymbol{d} &= \mathrm{vec}\,(\mathsf{W}_{1}) - \left(\mathsf{R}^{\top}\otimes\mathsf{L}\right)^{\top}\left(\mathsf{W}_{0,C}\otimes\mathsf{W}_{0,R}\right)\mathrm{vec}\,(\mathsf{Y})\,, \\
&= \mathrm{vec}\,(\mathsf{W}_{1}) - \mathrm{vec}\,\left(\mathsf{L}^{\top}\mathsf{W}_{0,R}\mathsf{Y}\mathsf{W}_{0,C}\mathsf{R}^{\top}\right).
\end{aligned}
$$

We write $\|\boldsymbol{x}\|^{2}_{\mathsf{A}}$ to mean $\boldsymbol{x}^{\top}\mathsf{A}\boldsymbol{x}$. Note that $\mathsf{G}\geq 0$, but $\boldsymbol{d}$ will likely have negative elements. In fact, to use Lee and Seung's iterative algorithm, the vector $\boldsymbol{d}$ must have non-positive elements. This is equivalent to the restriction $\mathsf{L}^{\top}\mathsf{W}_{0,R}\mathsf{Y}\mathsf{W}_{0,C}\mathsf{R} - \mathsf{W}_{1}\geq 0$, taken elementwise. This imposes a restriction on W_{1} which is hard to verify for the problem of non-negative matrix factorization. Note that by definition G can be positive definite, depending on the R, L and the $\mathsf{W}_{2,.}$, but is only positive *semi* definite in the general case.

We can thus consider this problem as optimization over a vector unknown. In an abuse of notation, we reuse the objective function $\phi\,(\cdot)$, writing

$$
\phi\,(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\top}\mathsf{G}\boldsymbol{x} + \boldsymbol{d}^{\top}\boldsymbol{x},
\tag{9}
$$

for symmetric, nonnegative positive semidefinite G. So now we consider the positivity-constrained quadratic optimization problem

$$
\min_{\boldsymbol{x}\geq 0}\phi\,(\boldsymbol{x})\,.
\tag{10}
$$

Note that the unconstrained solution to this problem is $-\mathsf{G}^{-1}\boldsymbol{d}$. When G is positive definite, because the constraint space is convex, this solution will be

unique. [5] In general, however, we will not have unique solutions. When the W_1 term is all zero, this a weighted least squares problem, possibly with Tikhonov regularization.

The gradient of the objective $\phi(\boldsymbol{x})$ is is $\nabla\phi(\boldsymbol{x}) = \mathsf{G}\boldsymbol{x} + \boldsymbol{d}$. When considering the matrix form of the problem, this has the value

$$\text{vec}\left(\left[\mathsf{L}^\top \mathsf{W}_{0,R}\mathsf{L}\mathsf{X}\mathsf{R}\mathsf{W}_{0,C}\mathsf{R}^\top + \sum_j \mathsf{W}_{2,R,j}\mathsf{X}\mathsf{W}_{2,C,j} + \mathsf{W}_1 - \mathsf{L}^\top \mathsf{W}_{0,R}\mathsf{Y}\mathsf{W}_{0,C}\mathsf{R}^\top\right]\right).$$

We consider an iterative solution to this problem. We first select some $\boldsymbol{x}^{(0)}$. Then given current iterate $\boldsymbol{x}^{(k)}$, we seek to find the next iterate $\boldsymbol{x}^{(k+1)}$. Define

$$\phi^{(k)}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \mathsf{G}^{(k)}\boldsymbol{x} + \boldsymbol{d}^{(k)^\top}\boldsymbol{x} + c_k,$$

where we will set $\mathsf{G}^{(k)}$, $\boldsymbol{d}^{(k)}$ and c_k such that $\phi^{(k)}(\boldsymbol{x})$ is tangent to $\phi(\boldsymbol{x})$ at $\boldsymbol{x}^{(k)}$, and such that $\phi^{(k)}(\boldsymbol{x}) \geq \phi(\boldsymbol{x})$ for all $\boldsymbol{x}$. We will take $\boldsymbol{x}^{(k+1)}$ as some point which improves $\phi^{(k)}(\boldsymbol{x})$: its minimizer if it is non-negative, or another point if not. If $\boldsymbol{x}^{(k+1)}$ improves $\phi^{(k)}(\boldsymbol{x})$, then it also improves $\phi(\boldsymbol{x})$, as

$$\phi\left(\boldsymbol{x}^{(k+1)}\right) \leq \phi^{(k)}\left(\boldsymbol{x}^{(k+1)}\right) \leq \phi^{(k)}\left(\boldsymbol{x}^{(k)}\right) = \phi\left(\boldsymbol{x}^{(k)}\right).$$

The tangency condition fixes the value of c_k, and also gives the identity

$$\boldsymbol{d}^{(k)} = \boldsymbol{d} + \mathsf{G}\boldsymbol{x}^{(k)} - \mathsf{G}^{(k)}\boldsymbol{x}^{(k)}.$$

To ensure that $\phi^{(k)}(\boldsymbol{x}) \geq \phi(\boldsymbol{x})$ it is sufficient to select select $\mathsf{G}^{(k)}$ such that $\mathsf{G}^{(k)} - \mathsf{G}$ is positive semidefinite. The following lemma gives us a way of selecting diagonal matrices $\mathsf{G}^{(k)}$ that satisfy this condition.

First a bit of notation, for vector $\boldsymbol{v}$, $\text{Diag}(\boldsymbol{v})$ is the diagonal matrix with $\boldsymbol{v}$ for diagonal. For matrix A, $\text{diag}(\mathsf{A})$ is the vector of the diagonal of A. Thus we have $\text{diag}(\text{Diag}(\boldsymbol{v})) = \boldsymbol{v}$.

Lemma 2.1. *Let G be a symmetric matrix with non-negative elements, and full rank, and let $\boldsymbol{b}$ be a vector with strictly positive elements. Then*

$$\text{Diag}(\mathsf{G}\boldsymbol{b})\text{Diag}(\boldsymbol{b})^{-1} \succeq \mathsf{G}.$$

(Here $\mathsf{A} \succeq \mathsf{B}$ means that $\mathsf{A} - \mathsf{B}$ is positive semidefinite.)

Proof. First note that if A is symmetric, and has non-negative elements, then $\text{Diag}(\mathsf{A}\mathbf{1}) \succeq \mathsf{A}$ because the matrix $\text{Diag}(\mathsf{A}\mathbf{1}) - \mathsf{A}$ is symmetric, with non-negative diagonal, and is diagonally dominant, thus it is positive semidefinite.

Apply that fact with $\mathsf{A} = \text{Diag}(\boldsymbol{b})\mathsf{G}\text{Diag}(\boldsymbol{b})$. Then

$$\text{Diag}(\text{Diag}(\boldsymbol{b})\mathsf{G}\text{Diag}(\boldsymbol{b})\mathbf{1}) \succeq \text{Diag}(\boldsymbol{b})\mathsf{G}\text{Diag}(\boldsymbol{b}),$$
$$\text{Diag}(\boldsymbol{b})\text{Diag}(\mathsf{G}\boldsymbol{b}) \succeq \text{Diag}(\boldsymbol{b})\mathsf{G}\text{Diag}(\boldsymbol{b}),$$
$$\text{Diag}(\mathsf{G}\boldsymbol{b})\text{Diag}(\boldsymbol{b})^{-1} \succeq \mathsf{G},$$

as needed.

Note that the proof relies on strict positivity of the elements of $\boldsymbol{b}$. By this we can claim a bijection between $\mathbb{R}^n$ and $\text{Diag}(\boldsymbol{b})\mathbb{R}^n$, which allows us to conclude the last line above. $\square$

Lemma 2.2. *Let* G *be a symmetric matrix with non-negative elements, and full rank. Let $\boldsymbol{b}$ be some vector with non-negative elements such that $\mathsf{G}\boldsymbol{b}$ has strictly positive elements. Letting*

$$\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)} - \operatorname{Diag}\left(\boldsymbol{b}\right) \operatorname{Diag}\left(\mathsf{G}\boldsymbol{b}\right)^{-1}\left(\boldsymbol{d} + \mathsf{G}\boldsymbol{x}^{(k)}\right),$$

then $\phi\left(\boldsymbol{x}^{(k+1)}\right) \leq \phi\left(\boldsymbol{x}^{(k)}\right).$

Proof. First, consider the case where $\boldsymbol{b}$ has strictly positive elements. Letting $\mathsf{G}^{(k)} = \operatorname{Diag}\left(\mathsf{G}\boldsymbol{b}\right) \operatorname{Diag}\left(\boldsymbol{b}\right)^{-1}$, the minimum of $\phi^{(k)}\left(\boldsymbol{x}\right)$ occurs at $-\mathsf{G}^{(k)^{-1}}\boldsymbol{d}^{(k)}$, which has value

$$
\begin{aligned}
-\mathsf{G}^{(k)^{-1}}\boldsymbol{d}^{(k)} &= -\mathsf{G}^{(k)^{-1}}\left(\boldsymbol{d} + \mathsf{G}\boldsymbol{x}^{(k)} - \mathsf{G}^{(k)}\boldsymbol{x}^{(k)}\right), \\
&= \boldsymbol{x}^{(k)} - \mathsf{G}^{(k)^{-1}}\left(\boldsymbol{d} + \mathsf{G}\boldsymbol{x}^{(k)}\right), \quad\quad (11) \\
&= \boldsymbol{x}^{(k)} - \operatorname{Diag}\left(\boldsymbol{b}\right) \operatorname{Diag}\left(\mathsf{G}\boldsymbol{b}\right)^{-1}\left(\boldsymbol{d} + \mathsf{G}\boldsymbol{x}^{(k)}\right) = \boldsymbol{x}^{(k+1)}.
\end{aligned}
$$

By Lemma 2.1, $\phi^{(k)}\left(\boldsymbol{x}\right)$ dominates $\phi\left(\boldsymbol{x}\right)$, and so $\phi\left(\boldsymbol{x}^{(k+1)}\right) \leq \phi\left(\boldsymbol{x}^{(k)}\right)$.

To prove the theorem for the general case where $\boldsymbol{b}$ is simply non-negative, consider a sequence of strictly positive vectors which converge to $\boldsymbol{b}$, and apply the argument above, then appeal to continuity of $\phi\left(\cdot\right)$. $\qquad\square$

Theorem 2.3 (Lee and Seung). *Let G be a symmetric matrix with non-negative elements, and full rank. Assume $\boldsymbol{x}^{(k)}$ has non-negative elements, and $\mathsf{G}\boldsymbol{x}^{(k)}$ has strictly positive elements. If*

$$\boldsymbol{x}^{(k+1)} \leftarrow -\boldsymbol{x}^{(k)} \odot \boldsymbol{d} \oslash \left(\mathsf{G}\boldsymbol{x}^{(k)}\right), \quad\quad (12)$$

then $\phi\left(\boldsymbol{x}^{(k+1)}\right) \leq \phi\left(\boldsymbol{x}^{(k)}\right)$. Moreover, the update preserves non-negativity of iterates as long as $\boldsymbol{d}$ has non-positive elements.

Proof. Let $\boldsymbol{b} = \boldsymbol{x}^{(k)}$ in Lemma 2.2. Then note that

$$
\begin{aligned}
\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} - \operatorname{Diag}\left(\boldsymbol{x}^{(k)}\right) \operatorname{Diag}\left(\mathsf{G}\boldsymbol{x}^{(k)}\right)^{-1}\left(\boldsymbol{d} + \mathsf{G}\boldsymbol{x}^{(k)}\right), \\
&= \boldsymbol{x}^{(k)} - \operatorname{Diag}\left(\boldsymbol{x}^{(k)}\right) \operatorname{Diag}\left(\mathsf{G}\boldsymbol{x}^{(k)}\right)^{-1}\boldsymbol{d} - \boldsymbol{x}^{(k)}, \\
&= -\operatorname{Diag}\left(\boldsymbol{x}^{(k)}\right) \operatorname{Diag}\left(\mathsf{G}\boldsymbol{x}^{(k)}\right)^{-1}\boldsymbol{d}, \\
&= -\boldsymbol{x}^{(k)} \odot \boldsymbol{d} \oslash \left(\mathsf{G}\boldsymbol{x}^{(k)}\right).
\end{aligned}
\quad\quad (13)
$$

$\qquad\square$

Returning to the original problem of minimizing the objective of Equation 6, the iterative update is

$$\mathsf{X}^{(k+1)} \leftarrow -\mathsf{X}^{(k)} \odot \left(\mathsf{W}_1 - \mathsf{L}^\top\mathsf{W}_{0,R}\mathsf{Y}\mathsf{W}_{0,C}\mathsf{R}^\top\right) \oslash$$

$$\left(\mathsf{L}^\top\mathsf{W}_{0,R}\mathsf{L}\mathsf{X}^{(k)}\mathsf{R}\mathsf{W}_{0,C}\mathsf{R}^\top + \sum_j \mathsf{W}_{2,R,j}\mathsf{X}^{(k)}\mathsf{W}_{2,C,j}\right). \quad (14)$$

2.1.1 As a Matrix Factorization Algorithm

Now consider using this iterative update in the solution of Problem 3. One starts with initial guesses $\mathsf{L}^{(0)}$ and $\mathsf{R}^{(0)}$, which are strictly positive, then computes in turn

$$\mathsf{L}^{(1)} \leftarrow -\mathsf{L}^{(0)} \odot \left(\mathsf{W}_{1,L} - \mathsf{W}_{0,R} \mathsf{Y} \mathsf{W}_{0,C} \mathsf{R}^{(0)\top} \right) \oslash \tag{15}$$

$$\left(\mathsf{W}_{0,R} \mathsf{L}^{(0)} \mathsf{R}^{(0)} \mathsf{W}_{0,C} \mathsf{R}^{(0)\top} + \sum_j \mathsf{W}_{2,R,L,j} \mathsf{L}^{(0)} \mathsf{W}_{2,C,L,j} \right),$$

$$\mathsf{R}^{(1)} \leftarrow -\mathsf{R}^{(0)} \odot \left(\mathsf{W}_{1,R} - \mathsf{L}^{(1)\top} \mathsf{W}_{0,R} \mathsf{Y} \mathsf{W}_{0,C} \right) \oslash \tag{16}$$

$$\left(\mathsf{L}^{(1)\top} \mathsf{W}_{0,R} \mathsf{L}^{(1)} \mathsf{R}^{(0)} \mathsf{W}_{0,C} + \sum_j \mathsf{W}_{2,R,R,j} \mathsf{R}^{(0)} \mathsf{W}_{2,C,R,j} \right).$$

Then one computes estimates of $\mathsf{L}^{(2)}, \mathsf{R}^{(2)}, \mathsf{L}^{(3)}, \mathsf{R}^{(3)}, \ldots$

The restriction on the W_1 from the symmetric form of the problem translates into the requirements that

$$\mathsf{W}_{1,L} - \mathsf{W}_{0,R} \mathsf{Y} \mathsf{W}_{0,C} \mathsf{R}^{(k)\top} \le 0, \quad \text{and} \tag{17}$$

$$\mathsf{W}_{1,R} - \mathsf{L}^{(k)\top} \mathsf{W}_{0,R} \mathsf{Y} \mathsf{W}_{0,C} \le 0, \tag{18}$$

elementwise. This restriction suffices to keep iterates strictly positive if the initial iterates are also strictly positive. Note, however, it is not clear how this can be guaranteed *a priori*, as the iterates $\mathsf{R}^{(k)}$ and $\mathsf{L}^{(k)}$ may become small, and the $\mathsf{W}_{1,\cdot}$ may be large. In fact, the *entire point* of ℓ_1 regularization is to encourage elements of $\mathsf{R}^{(k)}$ and $\mathsf{L}^{(k)}$ to take value zero. As a practical matter, then, code which allows arbitrary $\mathsf{W}_{1,L}, \mathsf{W}_{1,R}$ should take the liberty of overriding the input values and always guaranteeing that the numerators are bounded elementwise by

$$\left(\mathsf{W}_{1,L} - \mathsf{W}_{0,R} \mathsf{Y} \mathsf{W}_{0,C} \mathsf{R}^{(k)\top} \right) \le -\epsilon \ge \left(\mathsf{W}_{1,R} - \mathsf{L}^{(k)\top} \mathsf{W}_{0,R} \mathsf{Y} \mathsf{W}_{0,C} \right) \tag{19}$$

for some ϵ. This is a compromise for simplicity, and thus this form of the algorithm does not always solve for the input problem. In the next section we describe a more principled approach that respects the user input.

Again the restriction on the W_1 from the symmetric problem cannot be guaranteed *a priori*; instead an implementation must keep the numerators bounded away from zero. We present the method in Algorithm 1. We call it a "multiplicative update" since the update steps are elementwise multiply and divide operations on the initial estimates of L and R. For this reason, it has the same limitations as the Lee and Seung algorithm, namely that once an element of L or R takes value zero, it can never take a non-zero value. This algorithm assumes the simplified form of Equations 4; the code for the more general form of the problem is an obvious modification of the update steps.

3 Additive Steps and Convergence

Theorem 2.3 only tells us that the sequence $\phi\left(\boldsymbol{x}^{(k)}\right)$ is non-decreasing. It does *not* guarantee convergence to a global, or even a local, minimum. In fact, if we

---

Algorithm 1 Multiplicative Update Regularized Non-Negative Matrix Factorization.

---

function MURNMF(Y, d, $\mathsf{W}_{0,R}$, $\mathsf{W}_{0,C}$, $\lambda_{1,L}$, $\lambda_{1,R}$, $\lambda_{2,L}$, $\lambda_{2,R}$, $\gamma_{2,L}$, $\gamma_{2,R}$, $\epsilon > 0$)

    Initialize random matrices $\mathsf{L}^{(0)} > 0$ and $\mathsf{R}^{(0)} > 0$ with d columns and rows, respectively.

    Let $k \leftarrow 0$.

    while not converged **do**

        Compute numerator $\mathsf{H} \leftarrow \left(\lambda_{1,L}\mathbf{1} - \mathsf{W}_{0,R}\mathsf{Y}\mathsf{W}_{0,C}\mathsf{R}^{(k)}{}^{\top} \right)$.

        Clip the numerator $\mathsf{H} \leftarrow \mathsf{H} \wedge -\epsilon\mathbf{1}$.

        Let $\mathsf{F} \leftarrow \left(\mathsf{W}_{0,R}\mathsf{L}^{(k)}\mathsf{R}^{(k)}\mathsf{W}_{0,C}\mathsf{R}^{(k)}{}^{\top} + \lambda_{2,L}\mathsf{L}^{(k)} + \gamma_{2,L}\mathsf{L}^{(k)}\left(\mathbf{1}\mathbf{1}^{\top} - \mathsf{I} \right) \right)$.

        Let $\mathsf{L}^{(k+1)} \leftarrow -\mathsf{L}^{(k)} \odot \mathsf{H} \oslash \mathsf{F}$.

        Compute numerator $\mathsf{J} \leftarrow \left(\lambda_{1,R}\mathbf{1} - \mathsf{L}^{(k+1)}{}^{\top}\mathsf{W}_{0,R}\mathsf{Y}\mathsf{W}_{0,C} \right)$.

        Clip the numerator $\mathsf{J} \leftarrow \mathsf{J} \wedge -\epsilon\mathbf{1}$.

        Let $\mathsf{F} \leftarrow \left(\mathsf{L}^{(k+1)}{}^{\top}\mathsf{W}_{0,R}\mathsf{L}^{(k+1)}\mathsf{R}^{(k)}\mathsf{W}_{0,C} + \lambda_{2,R}\mathsf{R}^{(k)} + \gamma_{2,R}\mathsf{R}^{(k)}\left(\mathbf{1}\mathbf{1}^{\top} - \mathsf{I} \right) \right)$.

        Let $\mathsf{R}^{(k+1)} \leftarrow -\mathsf{R}^{(k)} \odot \mathsf{J} \oslash \mathsf{F}$.

        Increment $k \leftarrow k + 1$.

        Check convergence.

    end while

    return $\boldsymbol{x}^{(k)}$

end function

---

only restrict $\boldsymbol{d}$ to be non-positive, it is easy to construct cases where $\boldsymbol{x}^{(k)}$ will not converge to the constrained optimum. For if an element of $\boldsymbol{d}$ is zero, then that same element of $\boldsymbol{x}^{(k)}$ will be zero for all $k > 0$. However, the optimum may occur for $\boldsymbol{x}$ with a non-zero value for that element. (Certainly the *unconstrained* solution $-\mathsf{G}^{-1}\boldsymbol{d}$ may have a non-zero value for that element.)

Here we analyze the Lee and Seung update as a more traditional additive step, rather than a multiplicative step. Expressing the iteration as an additive step, we have

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{h}^{(k+1)}, \tag{20}$$

where

$$\boldsymbol{h}^{(k+1)} = -\boldsymbol{x}^{(k)} \odot \left(\boldsymbol{d} \oslash \left(\mathsf{G}\boldsymbol{x}^{(k)} \right) + \mathbf{1} \right), \tag{21}$$

$$= -\left(\mathsf{G}\boldsymbol{x}^{(k)} + \boldsymbol{d} \right) \odot \boldsymbol{x}^{(k)} \oslash \left(\mathsf{G}\boldsymbol{x}^{(k)} \right),$$

$$= -\nabla\phi\left(\boldsymbol{x}^{(k)} \right) \odot \boldsymbol{x}^{(k)} \oslash \left(\mathsf{G}\boldsymbol{x}^{(k)} \right). \tag{22}$$

This may not be the optimal length step in the direction of $\boldsymbol{h}^{(k+1)}$. If we instead take

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + c\boldsymbol{h}^{(k+1)}, \tag{23}$$

then $\phi\left(\boldsymbol{x}^{(k+1)} \right)$ is a quadratic function of c with optimum at

$$c^* = \frac{-\left(\mathsf{G}\boldsymbol{x}^{(k)} + \boldsymbol{d} \right)^{\top}\boldsymbol{h}^{(k+1)}}{\boldsymbol{h}^{(k+1)}{}^{\top}\mathsf{G}\boldsymbol{h}^{(k+1)}}. \tag{24}$$

8

However, if $c^* > 1$ elements of $\boldsymbol{x}^{(k)} + c^* \boldsymbol{h}^{(k+1)}$ could be negative. One would instead have to select a c that results in a strictly non-negative $\boldsymbol{x}^{(k+1)}$. However, if $c^* < 1$, then the original algorithm would have overshot the optimum.

Lemma 3.1. *Let* G *be a symmetric matrix with non-negative elements, and full rank. Let* c^* *be as defined in Equation 24. Let* c' *be the largest number in* $[0, c^*]$ *such that all elements of* $\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)} + c' \boldsymbol{h}^{(k+1)}$ *are non-negative, where* $\boldsymbol{h}^{(k+1)}$ *is as in Equation 22. Then* $\phi\left(\boldsymbol{x}^{(k+1)}\right) \leq \phi\left(\boldsymbol{x}^{(k)}\right)$.

Proof. Letting $f(c) = \phi\left(\boldsymbol{x}^{(k)} + c \boldsymbol{h}^{(k+1)}\right)$, $f(c)$ is quadratic in c with positive second derivative. Then since c' is between 0 and c^*, we have $f(c') \leq f(0)$, which was to be proven. By construction the elements of $\boldsymbol{x}^{(k+1)}$ are non-negative. $\square$

We note that if $c^* \leq 1$ then $c' = c^*$, since non-negativity is sustained for smaller step sizes.

This iterative update was originally described by Merritt and Zhang for solution of "Totally Nonnegative Least Squares" problems. [4] Their algorithm has guaranteed convergence for minimization of $\|\mathsf{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$, without the weighting or regularization terms, under certain conditions. They assume strict positivity of $\boldsymbol{b}$, which we cannot easily express in terms of G and $\boldsymbol{d}$, and it is not obvious that their proof can be directly used to guarantee convergence. Without guarantees of convergence, we express their method in our nomenclature as Algorithm 2, computing the step as $\boldsymbol{h}^{(k+1)} \leftarrow -\nabla\phi\left(\boldsymbol{x}^{(k)}\right) \odot \boldsymbol{x}^{(k)} \oslash \left(\mathsf{G}\boldsymbol{x}^{(k)}\right)$. The vague language around selecting the step length τ_k is due to Merritt and Zhang; presumably one can choose it randomly, or always take $\tau_k = (\tau + 1)/2$.

3.1 Other Directions

Imagine, instead, plugging in different values of $\boldsymbol{b}$ into the $\boldsymbol{x}^{(k+1)}$ given by Lemma 2.2. We claim that setting $\boldsymbol{b} = \boldsymbol{x}^{(k)} + \mathsf{G}^{-1}\boldsymbol{d}$ would yield the global unconstrained minimizer for $\phi(\boldsymbol{x})$:

$$
\begin{aligned}
\boldsymbol{x}^{(k+1)} &\leftarrow \boldsymbol{x}^{(k)} - \mathrm{Diag}\left(\boldsymbol{x}^{(k)} + \mathsf{G}^{-1}\boldsymbol{d}\right) \mathrm{Diag}\left(\mathsf{G}\boldsymbol{x}^{(k)} + \boldsymbol{d}\right)^{-1}\left(\boldsymbol{d} + \mathsf{G}\boldsymbol{x}^{(k)}\right), \\
&= \boldsymbol{x}^{(k)} - \mathrm{Diag}\left(\boldsymbol{x}^{(k)} + \mathsf{G}^{-1}\boldsymbol{d}\right)\mathbf{1}, \\
&= -\mathsf{G}^{-1}\boldsymbol{d}.
\end{aligned}
\tag{25}
$$

This would seem to be a totally useless computation because we cannot efficiently compute $-\mathsf{G}^{-1}\boldsymbol{d}$, and it likely violates the positivity constraint. However, this suggests an alternative iterative step that might have accelerated convergence. For example, it might be the case that setting $\boldsymbol{b} = \left(\boldsymbol{x}^{(k)} + \mathrm{Diag}\left(\mathrm{diag}\left(\mathsf{G}\right)\right)^{-1}\boldsymbol{d}\right)^+$, might give quicker convergence to a solution, where x^+ is the non-negative part of x.

We can also imagine an iterative update based on different descent directions altogether. For example steepest descent, where $\boldsymbol{h}^{(k+1)} = -\nabla\phi\left(\boldsymbol{x}^{(k)}\right)$. Both of these and the method above can be couched as the general iterative method of Algorithm 2. We note, however, that steepest descent will fail to make progress in this formulation when an element of $\boldsymbol{x}^{(k)}$ is zero and the corresponding element of the gradient is positive. In this case, the step length would

be computed as zero and the algorithm would terminate early. The Lee and Seung step naturally avoids this problem by scaling the step direction proportional element-wise to $\boldsymbol{x}^{(k)}$. It is not clear, however, whether the denominator part of the Lee and Seung step is necessary, and perhaps taking steps in the direction of $\boldsymbol{h}^{(k+1)} = -\nabla\phi\left(\boldsymbol{x}^{(k)}\right) \odot \boldsymbol{x}^{(k)}$ sufficiently good convergence.

Algorithm 2 The General Iterative Quadrative Programming Method

    function GIQPM_STEP($\boldsymbol{x}^{(k)}$, G, $\boldsymbol{d}$, $\tau \in (0,1)$)

        Compute the gradient at $\boldsymbol{x}^{(k)}$: $\nabla\phi\left(\boldsymbol{x}^{(k)}\right) \leftarrow \mathsf{G}\boldsymbol{x}^{(k)} + \boldsymbol{d}$.

        Somehow choose step $\boldsymbol{h}^{(k+1)}$ such that ${\boldsymbol{h}^{(k+1)}}^{\top}\nabla\phi\left(\boldsymbol{x}^{(k)}\right) < 0$.

        Compute maximum allowable step length:

            $\hat{\alpha}_k \leftarrow \min\left\{\alpha : \boldsymbol{x}^{(k)} + \alpha\boldsymbol{h}^{(k+1)} \geq 0\right\}$.

        Compute optimal step length:

            $\alpha_k^* \leftarrow -\nabla\phi\left(\boldsymbol{x}^{(k)}\right)^{\top}\boldsymbol{h}^{(k+1)} / \left({\boldsymbol{h}^{(k+1)}}^{\top}\mathsf{G}\boldsymbol{h}^{(k+1)}\right)$.

        Choose $\tau_k \in [\tau, 1)$ and let α_k be the minimum of $\tau_k\hat{\alpha}_k$ and α_k^*.

        Let $\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)} + \alpha_k\boldsymbol{h}^{(k+1)}$.

        return $\boldsymbol{x}^{(k+1)}$

    end function

    function GIQPM(G, $\boldsymbol{d}$, $\tau \in (0,1)$)

        Initialize $\boldsymbol{x}^{(0)} > 0$ and set $k \leftarrow 0$.

        while not converged **do**

            Let $\boldsymbol{x}^{(k+1)} \leftarrow$ GIQPM_STEP$\left(\boldsymbol{x}^{(k)}, \mathsf{G}, \boldsymbol{d}, \tau\right)$.

            Increment $k \leftarrow k + 1$.

            Check convergence.

        end while

        return $\boldsymbol{x}^{(k)}$

    end function

Converting this to the NMF problem as formulated in Section 2.1.1 is a straightforward, though somewhat tedious, exercise. We give the algorithm in Algorithm 3 for the case where the weighting matrices satisfy the conditions of Equations 4. We note that the restrictions on $\mathsf{W}_{1,R}$ and $\mathsf{W}_{1,L}$ that caused so much headache above are swept under the rug here with the check for maximum allowable step length, $\hat{\alpha}_k$, in GIQPM_STEP.

It is not clear *a priori* whether this algorithm converges quickly. It is very unlikely that Algorithm 2 is competitive for the constrained optimization problem on vectors given in 10. Converting to steepest descent is not recommended due to slow convergence, as noted above. One could swap out the iterative updates of $\mathsf{L}^{(k)}$ and $\mathsf{R}^{(k)}$ in Algorithm 3 with something like a constrained conjugate gradient. [3, 5] However, one should take several steps of conjugate gradient for each k as the optimization problem changes as we update the $\mathsf{L}^{(k)}$ and $\mathsf{R}^{(k)}$.

4 Simulations

The multiplicative and additive algorithms are implemented in the R package RNNMF written by the author. [7] Here we briefly summarize the results of using the code on generated data. We first generate a Y the exactly equals LR for

---
Algorithm 3 Additive Update Regularized Non-Negative Matrix Factorization, II.

---
function GIQPM_STEP($\mathsf{X}^{(k)}$, $\nabla\phi\left(\mathsf{X}^{(k)}\right)$, $\mathsf{H}^{(k+1)}$, $\mathsf{K}^{(k+1)}$, $\tau_k \in (0,1)$)

    Compute maximum allowable step length:

      $\hat{\alpha}_k \leftarrow \min\left\{\alpha : \mathsf{X}^{(k)} + \alpha\mathsf{H}^{(k+1)} \geq 0\right\}.$

    Compute optimal step length:

      $\alpha_k^* \leftarrow -\operatorname{tr}\left(\nabla\phi\left(\mathsf{X}^{(k)}\right)^\top\mathsf{H}^{(k+1)}\right) / \operatorname{tr}\left(\mathsf{H}^{(k+1)^\top}\mathsf{K}^{(k+1)}\right).$

    (compute these traces without performing matrix multiplies.)

    Let α_k be the minimum of $\tau_k\hat{\alpha}_k$ and α_k^*.

    Let $\mathsf{X}^{(k+1)} \leftarrow \mathsf{X}^{(k)} + \alpha_k\mathsf{H}^{(k+1)}.$

    return $\mathsf{X}^{(k+1)}$

end function

function PICK_DIRECTION($\mathsf{X}^{(k)}$, $\nabla\phi\left(\mathsf{X}^{(k)}\right)$, F)

    Initialize $\mathsf{H}^{(k)} \leftarrow -\nabla\phi\left(\mathsf{X}^{(k)}\right) \odot \mathsf{X}^{(k)} \oslash \mathsf{F}$.

    Elements of $\mathsf{H}^{(k)}$ for which F and $\mathsf{X}^{(k)}$ are both zero set to $max\left(-\nabla\phi\left(\mathsf{X}^{(k)}\right), 0\right)$.

    Elements of $\mathsf{H}^{(k)}$ for which F is zero but $\mathsf{X}^{(k)} > 0$ set to $-\nabla\phi\left(\mathsf{X}^{(k)}\right) \odot \mathsf{X}^{(k)}$.

    return $\mathsf{H}^{(k+1)}$

end function

function AURNMF(Y, d, $\mathsf{W}_{0,R}$, $\mathsf{W}_{0,C}$, $\lambda_{1,L}$, $\lambda_{1,R}$, $\lambda_{2,L}$, $\lambda_{2,R}$, $\gamma_{2,L}$, $\gamma_{2,R}$, $\tau \in (0,1)$)

    Initialize random $\mathsf{L}^{(0)} > 0$ with d columns and $\mathsf{R}^{(0)} > 0$ with d rows.

    Let $k \leftarrow 0$.

    while not converged **do**

        Choose $\tau_k \in [\tau, 1)$.

        $\mathsf{D} \leftarrow \left(\lambda_{1,L}\mathbf{1} - \mathsf{W}_{0,R}\mathsf{Y}\mathsf{W}_{0,C}\mathsf{R}^{(k)^\top}\right).$

        $\mathsf{F} \leftarrow \mathsf{W}_{0,R}\mathsf{L}^{(k)}\mathsf{R}^{(k)}\mathsf{W}_{0,C}\mathsf{R}^{(k)^\top} + \lambda_{2,L}\mathsf{L}^{(k)} + \gamma_{2,L}\mathsf{L}^{(k)}\left(\mathbf{11}^\top - \mathsf{I}\right)$

        $\nabla\phi\left(\mathsf{L}^{(k)}\right) \leftarrow \mathsf{F} + \mathsf{D}$.

        $\mathsf{H}^{(k+1)} \leftarrow$ PICK_DIRECTION $\left(\mathsf{L}^{(k)}, \nabla\phi\left(\mathsf{L}^{(k)}\right), \mathsf{F}\right)$.

        $\mathsf{K}^{(k+1)} \leftarrow \mathsf{W}_{0,R}\mathsf{H}^{(k+1)}\mathsf{R}^{(k)}\mathsf{W}_{0,C}\mathsf{R}^{(k)^\top} + \lambda_{2,L}\mathsf{H}^{(k+1)}$.

        Let $\mathsf{L}^{(k+1)} \leftarrow$ GIQPM_STEP $\left(\mathsf{L}^{(k)}, \nabla\phi\left(\mathsf{L}^{(k)}\right), \mathsf{H}^{(k+1)}, \mathsf{K}^{(k+1)}, \tau_k\right)$.

        $\mathsf{D} \leftarrow \left(\lambda_{1,R}\mathbf{1} - \mathsf{L}^{(k+1)^\top}\mathsf{W}_{0,R}\mathsf{Y}\mathsf{W}_{0,C}\right).$

        $\mathsf{F} \leftarrow \mathsf{L}^{(k+1)^\top}\mathsf{W}_{0,R}\mathsf{L}^{(k+1)}\mathsf{R}^{(k)}\mathsf{W}_{0,C} + \lambda_{2,R}\mathsf{R}^{(k)} + \gamma_{2,R}\mathsf{R}^{(k)}\left(\mathbf{11}^\top - \mathsf{I}\right)$

        $\nabla\phi\left(\mathsf{R}^{(k)}\right) \leftarrow \mathsf{F} + \mathsf{D}$.

        $\mathsf{H}^{(k+1)} \leftarrow$ PICK_DIRECTION $\left(\mathsf{R}^{(k)}, \nabla\phi\left(\mathsf{R}^{(k)}\right), \mathsf{F}\right)$.

        $\mathsf{K}^{(k+1)} \leftarrow \mathsf{L}^{(k+1)^\top}\mathsf{W}_{0,R}\mathsf{L}^{(k+1)}\mathsf{H}^{(k+1)}\mathsf{W}_{0,C} + \lambda_{2,R}\mathsf{H}^{(k+1)}$

        Let $\mathsf{R}^{(k+1)} \leftarrow$ GIQPM_STEP $\left(\mathsf{R}^{(k)}, \nabla\phi\left(\mathsf{R}^{(k)}\right), \mathsf{H}^{(k+1)}, \mathsf{K}^{(k+1)}, \tau_k\right)$.

        Increment $k \leftarrow k + 1$.

        Check convergence.

    end while

    return $\boldsymbol{x}^{(k)}$

end function

---

some non-negative uniformly random L and R. In the first simulations we take L to be 30×2, and R to be 2×8. We randomly generate starting iterates, 30×2 matrix $\mathsf{L}^{(0)}$ and 2×8 matrix $\mathsf{R}^{(0)}$. We use the same starting point for both the additive and multiplicative algorithms. We compute the Frobenius norm of the error, which is to say

$$\sqrt{\operatorname{tr}\left(\left(\mathsf{Y} - \mathsf{L}^{(k)}\mathsf{R}^{(k)}\right)^{\top}\left(\mathsf{Y} - \mathsf{L}^{(k)}\mathsf{R}^{(k)}\right)\right)},$$

for each iteration.

In Figure 1 we plot the error versus step for the additive and multiplicative methods. The additive method uses the optimal step size in the chosen direction. While not shown here, choosing the naive step without optimizing converges slower in iterations but may ultimately be faster computationally. The additive method outperforms the multiplicative method in convergence per step. Convergence of both is "lumpy", with apparent phase changes where, we suspect, different directions become the dominant source of approximation error, and then are smoothed out.
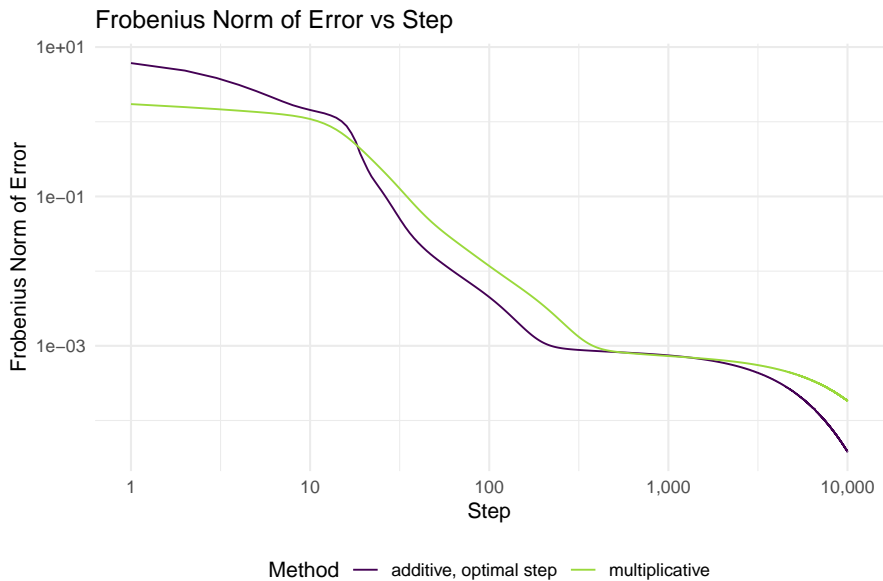
Factoring 30 x 8 matrix down to 3 dimensions. Y matrix has rank 2.

Figure 1: The Frobenius norm is plotted versus step for two methods for a small problem.

We then apply the code to another test case. Here we generate L to be 40×3, and R to be 3×10. Again we test both the additive and multiplicative algorithms, but test the effect of sparse starting iterates. We randomly generate starting iterates, 40×4 matrix $\mathsf{L}^{(0)}$ and 4×10 matrix $\mathsf{R}^{(0)}$. First we do this where about $\frac{1}{3}$ of the elements of $\mathsf{L}^{(0)}$ and $\mathsf{R}^{(0)}$ are zero, testing both algorithms. We repeat the experiment for the same Y, but generate strictly positive $\mathsf{L}^{(0)}$ and $\mathsf{R}^{(0)}$.

In Figure 2 we plot the error versus step for both methods and both choices of starting iterates. Again we see somewhat slower convergence for the multiplicative method, at least measured in iterates. Moreover, the multiplicative method fails to converge for the sparse starting iterates. As we noted above, the multiplicative method cannot change an element from zero to non-negative, thus the lack of convergence for sparse initial estimates is not surprising.
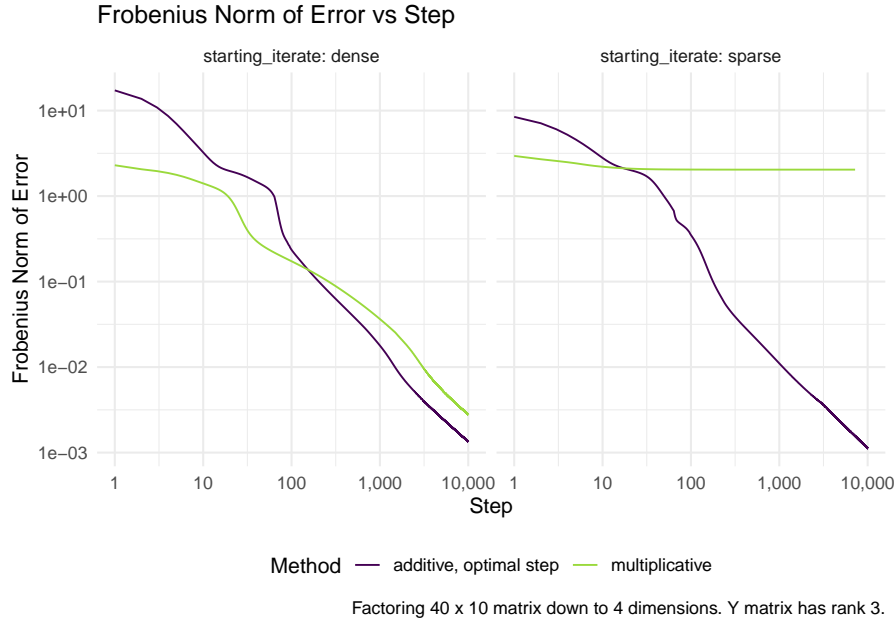

Factoring 40 x 10 matrix down to 4 dimensions. Y matrix has rank 3.

Figure 2: The Frobenius norm is plotted versus step for two methods for a small problem. Starting iterates are taken to be sparse or dense.

References

[1] Edward F. Gonzalez and Yin Zhang. Accelerating the Lee-Seung algorithm for nonnegative matrix factorization. Technical Report TR05-02, Rice University, 2005. URL https://scholarship.rice.edu/bitstream/handle/1911/102034/TR05-02.pdf.

[2] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press, 2001. URL http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf.

[3] Can Li. A conjugate gradient type method for the nonnegative constraints optimization problems. *Journal of Applied Mathematics*, 2013:986317, 2013. doi: 10.1155/2013/986317. URL https://doi.org/10.1155/2013/986317.

[4] Michael Merritt and Yin Zhang. Interior-point gradient method for large-scale totally nonnegative least squares problems. *Journal of optimization*

theory and applications, 126(1):191–202, 2005. URL https://scholarship.rice.edu/bitstream/handle/1911/102020/TR04-08.pdf.

[5] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer series in operations research and financial engineering. Springer, 2006. ISBN 9780387400655. URL http://books.google.com/books?id=VbHYoSyelFcC.

[6] Steven E. Pav. System and method for unmixing spectroscopic observations with nonnegative matrix factorization, 2012. URL https://patentscope.wipo.int/search/en/detail.jsf?docId=US42758160.

[7] Steven E. Pav. *rnnmf: Regularized Non-negative Matrix Factorization*, 2024. URL https://github.com/shabbychef/rnnmf. R package version 0.3.0.

[8] Farid Saberi-Movahed, Kamal Berahman, Razieh Sheikhpour, Yuefeng Li, and Shirui Pan. Nonnegative matrix factorization in dimensionality reduction: A survey, 2024. URL https://arxiv.org/abs/2405.03615.

[9] Kathrin Schäcke. On the Kronecker product, 2013. URL https://www.math.uwaterloo.ca/~hwolkowi/henry/reports/kronthesisschaecke04.pdf.

A Matrix Identities

The following identities are useful for switching between matrix and vectorized representations. [9]

$$\operatorname{vec}(\mathsf{BCD}) = \left(\mathsf{D}^\top \otimes \mathsf{B}\right)\operatorname{vec}(\mathsf{C}). \tag{26}$$

$$\operatorname{tr}\left(\mathsf{A}^\top \mathsf{C}\right) = \operatorname{vec}(\mathsf{A})^\top \operatorname{vec}(\mathsf{C}). \tag{27}$$

$$\operatorname{tr}\left(\mathsf{A}^\top \mathsf{BCD}\right) = \operatorname{vec}(\mathsf{A})^\top \left(\mathsf{D}^\top \otimes \mathsf{B}\right)\operatorname{vec}(\mathsf{C}). \tag{28}$$

$$\operatorname{tr}\left(\mathsf{A}^\top \mathsf{BAD}\right) = \|\operatorname{vec}(\mathsf{A})\|_{\mathsf{D}^\top \otimes \mathsf{B}}^2. \tag{29}$$

$$\left(\mathsf{A} \otimes \mathsf{B}\right)^\top = \mathsf{A}^\top \otimes \mathsf{B}^\top. \tag{30}$$