Proceedings of ALGORITMY 2016
pp. 312–322

NONNEGATIVE MATRIX FACTORIZATION VIA NEWTON ITERATION FOR SHARED-MEMORY SYSTEMS*

MARKUS FLATZ† AND MARIÁN VAJTERŠIC‡

Abstract. Nonnegative Matrix Factorization (NMF) can be used to approximate a large nonnegative matrix as a product of two smaller nonnegative matrices. This paper shows in detail how an NMF algorithm based on Newton iteration can be derived utilizing the general Karush-Kuhn-Tucker (KKT) conditions for first-order optimality. This algorithm is suited for parallel execution on shared-memory systems. It was implemented and tested, delivering satisfactory speedup results.

Key words. Nonnegative Matrix Factorization, Parallel Computing, Newton Iteration

AMS subject classifications. 68W10, 15A23, 49M15

1. Introduction.

The need to process large amounts of data is prevalent in modern society. One important class of data is represented by nonnegative matrices, which occur in many application areas. The processing and evaluation of such large amounts of data is difficult and time-consuming. Therefore, parallelism is often inevitable to solve such problems in practice.

The goal of Nonnegative Matrix Factorization (NMF) is to represent a large nonnegative matrix in an approximate way as a product of two significantly smaller nonnegative matrices, which are easier to handle and process. The idea of such a factorization was published in 1994 under the name "Positive Matrix Factorization" [7]. In 1999, an article in Nature [5] about Nonnegative Matrix Factorization caught the attention of a wide audience. Several papers were written about NMF since then, discussing its properties, algorithms, modifications and often also possible applications. NMF and was successfully used in a variety of application areas, for example in text mining, document classification, clustering, spectral data analysis, face recognition and computational biology. In contrast to other methods such as singular value decomposition (SVD) or principal component analysis (PCA), NMF has the distinguishing property that the factors are guaranteed to be nonnegative, which allows interpreting the factorization as an additive combination of features.

The paper is organized as follows: First, the NMF problem is defined in section 2. Next, the Karush-Kuhn-Tucker (KKT) conditions for first-order optimality are formulated in section 3. Their application to the NMF problem is discussed in detail in section 4, leading to the parallel NMF algorithm based on Newton iteration in section 5. The algorithm was implemented and tested on a parallel system, and corresponding speedup measurements (on up to 64 processor cores) are included in section 6. Section 7 contains the conclusion.

*The second author was supported by the VEGA grant no. 2/0026/14 from the Scientific Grant Agency of the Ministry of Education and Slovak Academy of Sciences, Slovakia.

†Department of Computer Sciences, University of Salzburg, Salzburg, Austria (mflatz@cosy.sbg.ac.at).

‡Department of Computer Sciences, University of Salzburg, Salzburg, Austria and Mathematical Institute, Department of Informatics, Slovak Academy of Sciences, Bratislava, Slovakia (marian@cosy.sbg.ac.at).

2. The NMF problem. Formally, NMF can be defined as [6]:

DEFINITION 2.1 (NMF). *Given a nonnegative matrix* $\mathbf{Y} \in \mathbb{R}^{I \times T}$ *and a positive integer* J*, find nonnegative matrices* $\mathbf{A} \in \mathbb{R}^{I \times J}$ *and* $\mathbf{X} \in \mathbb{R}^{J \times T}$ *that minimize the functional*

$$\mathbf{f}(\mathbf{A}, \mathbf{X}) = \frac{1}{2} ||\mathbf{Y} - \mathbf{AX}||_F^2 . \tag{2.1}$$

Figure 2.1 illustrates the NMF problem. A matrix is called nonnegative if all its elements are ≥ 0. For an $I \times T$ matrix $\mathbf{M}$, $||\mathbf{M}||_F$ is the Frobenius norm of $\mathbf{M}$. Therefore, $\mathbf{f}(\mathbf{A}, \mathbf{X})$ is the square of the Euclidean distance between $\mathbf{Y}$ and $\mathbf{AX}$ with an additional factor $\frac{1}{2}$.

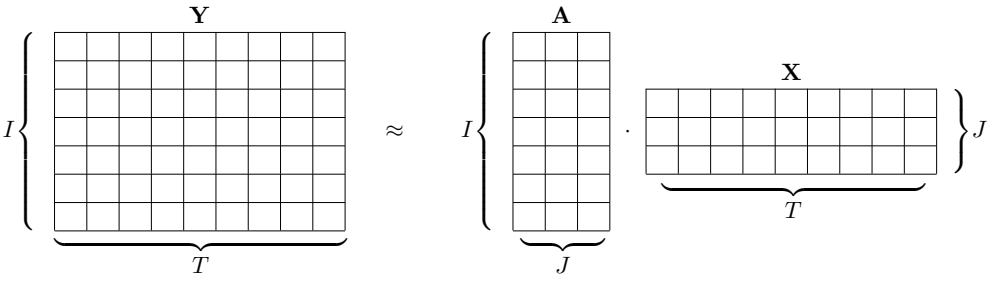

FIGURE 2.1. *Illustration of the NMF problem.*

In practical cases, the chosen J is usually much smaller than I and T. In general, it is not possible to find $\mathbf{A}$ and $\mathbf{X}$ such that $\mathbf{AX} = \mathbf{Y}$, NMF is only an approximation, for this reason it is sometimes called Approximative Nonnegative Matrix Factorization or Nonnegative Matrix Approximation. Thus, $\mathbf{AX}$ can be seen as a compressed representation of $\mathbf{Y}$, with a rank of J or less. The NMF problem does not have a unique solution, for example for any diagonal matrix $\mathbf{D}$ with positive diagonal values, $\mathbf{f}(\mathbf{A}, \mathbf{X}) = \mathbf{f}(\mathbf{AD}, \mathbf{D}^{-1}\mathbf{X})$. The problem is convex in $\mathbf{A}$ and in $\mathbf{X}$ separately, but not in both simultaneously [4]. Every column of $\mathbf{A}$ can be interpreted as a basis feature of size I. In total, $\mathbf{A}$ contains J basis features. The multiplication of $\mathbf{A}$ with the nonnegative matrix $\mathbf{X}$ yields a matrix $\mathbf{AX}$, where every column of $\mathbf{AX}$ is an additive (or non-subtractive) combination of weighted basis features (columns of $\mathbf{A}$).

3. Karush-Kuhn-Tucker conditions for first-order optimality. Karush-Kuhn-Tucker conditions are necessary conditions for first-order optimality. We present these conditions after introducing some prerequisites. This section is based on [1]. The goal is to

$$\begin{array}{lll} \text{minimize} & \mathbf{f}_0(\mathbf{x}) & \\ \text{subject to} & \mathbf{f}_i(\mathbf{x}) \leq 0, & i = 1, \ldots, m \\ & \mathbf{h}_i(\mathbf{x}) = 0, & i = 1, \ldots, p, \end{array} \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the variable, whose domain $\mathcal{D} = \bigcap_{i=0}^{m} \mathbf{dom}(\mathbf{f}_i) \cap \bigcap_{i=1}^{p} \mathbf{dom}(\mathbf{h}_i)$ is nonempty. The optimal value of (3.1) is denoted by $\mathbf{p}^\star$. The Lagrangian $\mathbf{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is the function

$$\mathbf{L}(\mathbf{x}, \lambda, \nu) = \mathbf{f}_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \mathbf{f}_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i \mathbf{h}_i(\mathbf{x}), \tag{3.2}$$

with $\mathbf{dom}(\mathbf{L}) = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$. The Lagrange dual function $\mathbf{g} : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is defined as

$$\mathbf{g}(\lambda, \nu) = \inf_{\mathbf{x} \in \mathcal{D}} \mathbf{L}(\mathbf{x}, \lambda, \nu) = \inf_{\mathbf{x} \in \mathcal{D}} \left(\mathbf{f}_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i \mathbf{f}_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i \mathbf{h}_i(\mathbf{x}) \right) . \tag{3.3}$$

For any $\lambda \geq \mathbf{0}$ and any ν, $\mathbf{g}(\lambda, \nu) \leq \mathbf{p}^\star$. The search for the best lower bound for $\mathbf{p}^\star$ given by the Lagrange dual function $\mathbf{g}$ leads to the optimization problem

$$\begin{array}{ll} \text{maximize} & \mathbf{g}(\lambda, \nu) \\ \text{subject to} & \lambda \geq \mathbf{0} \, . \end{array} \tag{3.4}$$

This is the Lagrange dual problem, while (3.1) is the primal problem. $(\lambda^\star, \nu^\star)$ are called dual optimal if they are optimal for (3.4), and $\mathbf{d}^\star = \mathbf{g}(\lambda^\star, \nu^\star)$ is the resulting optimal value of (3.4). The difference $\mathbf{p}^\star - \mathbf{d}^\star$ is called the optimal duality gap. If the optimal duality gap is zero, so-called strong duality holds.

Let $\mathbf{x}^\star$ be primal optimal, i.e., $\mathbf{f}_0(\mathbf{x}^\star) = \mathbf{p}^\star$, and $(\lambda^\star, \nu^\star)$ dual optimal. If strong duality holds, then

$$\begin{aligned} \mathbf{f}_0(\mathbf{x}^\star) = \mathbf{g}(\lambda^\star, \nu^\star) &= \inf_{\mathbf{x} \in \mathcal{D}} \left(\mathbf{f}_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i^\star \mathbf{f}_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i^\star \mathbf{h}_i(\mathbf{x}) \right) \\ &\leq \mathbf{f}_0(\mathbf{x}^\star) + \sum_{i=1}^{m} \lambda_i^\star \mathbf{f}_i(\mathbf{x}^\star) + \sum_{i=1}^{p} \nu_i^\star \mathbf{h}_i(\mathbf{x}^\star) \leq \mathbf{f}_0(\mathbf{x}^\star) \, . \end{aligned} \tag{3.5}$$

because of the definitions of strong duality, of the dual function and of the infimum, and the last inequality holds because for all $i = 1, \ldots, p$, $\mathbf{h}_i(\mathbf{x}^\star) = 0$, and for all $i = 1, \ldots, m$, $\mathbf{f}_i(\mathbf{x}^\star) \leq 0$ and $\lambda_i^\star \geq 0$ and therefore $\lambda_i^\star \mathbf{f}_i(\mathbf{x}^\star) \leq 0$. Since we have a chain of $=$ and $\leq$ with $\mathbf{f}_0(\mathbf{x}^\star)$ at the beginning and at the end, all the $\leq$ are in fact $=$. Hence, we can conclude that $\sum_{i=1}^{m} \lambda_i^\star \mathbf{f}_i(\mathbf{x}^\star) = 0$ and, since all $\lambda_i^\star \mathbf{f}_i(\mathbf{x}^\star) \leq 0$,

$$\lambda_i^\star \mathbf{f}_i(\mathbf{x}^\star) = 0, \quad i = 1, \ldots, m \, . \tag{3.6}$$

This is called complementary slackness, and we have $\lambda_i^\star > 0 \Rightarrow \mathbf{f}_i(\mathbf{x}^\star) = 0$ and $\mathbf{f}_i(\mathbf{x}^\star) < 0 \Rightarrow \lambda_i^\star = 0$.

Now, assume the functions $\mathbf{f}_0, \ldots, \mathbf{f}_m, \mathbf{h}_1, \ldots, \mathbf{h}_p$ are differentiable. Let $\mathbf{x}^\star$ and $(\lambda^\star, \nu^\star)$ be any primal and dual optimal points with zero duality gap. Since $\mathbf{x}^\star$ minimizes $\mathbf{L}(\mathbf{x}, \lambda^\star, \nu^\star)$ over $\mathbf{x}$, its gradient must vanish at $\mathbf{x}^\star$:

$$\nabla \mathbf{f}_0(\mathbf{x}^\star) + \sum_{i=1}^{m} \lambda_i^\star \nabla \mathbf{f}_i(\mathbf{x}^\star) + \sum_{i=1}^{p} \nu_i^\star \nabla \mathbf{h}_i(\mathbf{x}^\star) = 0 \, . \tag{3.7}$$

Following the definitions and statements above, the following conditions, called Karush-Kuhn-Tucker (KKT) conditions, have to hold:

$$\begin{aligned} \mathbf{f}_i(\mathbf{x}^\star) &\leq 0, \quad i = 1, \ldots, m \\ \mathbf{h}_i(\mathbf{x}^\star) &= 0, \quad i = 1, \ldots, p \\ \lambda_i^\star &\geq 0, \quad i = 1, \ldots, m \\ \lambda_i^\star \mathbf{f}_i(\mathbf{x}^\star) &= 0, \quad i = 1, \ldots, m \\ \nabla \mathbf{f}_0(\mathbf{x}^\star) + \textstyle\sum_{i=1}^{m} \lambda_i^\star \nabla \mathbf{f}_i(\mathbf{x}^\star) + \sum_{i=1}^{p} \nu_i^\star \nabla \mathbf{h}_i(\mathbf{x}^\star) &= 0 \, . \end{aligned} \tag{3.8}$$

Additionally, if the primal problem is convex, then the KKT conditions are also sufficient for the points to be primal and dual optimal.

4. Karush-Kuhn-Tucker conditions for the NMF problem. The approach depicted here is based on the technical report [6] and [2]. Starting from the definition of NMF in Theorem 2.1, we introduce

$$\mathbf{E} = \mathbf{Y} - \mathbf{A}\mathbf{X}\,. \tag{4.1}$$

The matrix $\mathbf{E}$ can be seen as error matrix, the goal, minimizing (2.1), can be written as minimizing $\frac{1}{2}||\mathbf{E}||_F^2$. According to the definition of the Frobenius norm and of the standard matrix multiplication,

$$\mathbf{f}(\mathbf{A},\mathbf{X}) = \frac{1}{2}\sum_{i=1}^{I}\sum_{t=1}^{T} {e_{i,t}}^2 = \frac{1}{2}\sum_{i=1}^{I}\sum_{t=1}^{T}\left(y_{i,t} - \sum_{j=1}^{J} a_{i,j}\cdot x_{j,t}\right)^2 . \tag{4.2}$$

The partial derivative of $\mathbf{f}$ with respect to a single element $a_{k,l}$ of $\mathbf{A}$, with $1 \le k \le I$ and $1 \le l \le J$, is

$$\begin{aligned}\frac{\partial \mathbf{f}}{\partial a_{k,l}}(\mathbf{A},\mathbf{X}) &= \frac{1}{2}\sum_{t=1}^{T}\left(2\cdot\left(y_{k,t} - \sum_{j=1}^{J} a_{k,j}\cdot x_{j,t}\right)\cdot(-x_{l,t})\right) = \\ &= -\sum_{t=1}^{T}\left(e_{k,t}\cdot(\mathbf{X}^T)_{t,l}\right) = \left(-\mathbf{E}\mathbf{X}^T\right)_{k,l}\end{aligned} \tag{4.3}$$

where $(\mathbf{X}^T)_{t,l}$ denotes the element of the (transposed) matrix $\mathbf{X}^T$ with row index t and column index l, and $\left(-\mathbf{E}\mathbf{X}^T\right)_{k,l}$ the element of $-\mathbf{E}\mathbf{X}^T$ with indices k and l. All the $\frac{\partial \mathbf{f}}{\partial a_{k,l}}$ can be combined to the $I \times J$ matrix

$$\frac{\partial \mathbf{f}}{\partial \mathbf{A}}(\mathbf{A},\mathbf{X}) = \begin{pmatrix} \frac{\partial \mathbf{f}}{\partial a_{1,1}} & \frac{\partial \mathbf{f}}{\partial a_{1,2}} & \cdots & \frac{\partial \mathbf{f}}{\partial a_{1,J}} \\ \frac{\partial \mathbf{f}}{\partial a_{2,1}} & \frac{\partial \mathbf{f}}{\partial a_{2,2}} & \cdots & \frac{\partial \mathbf{f}}{\partial a_{2,J}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{f}}{\partial a_{I,1}} & \frac{\partial \mathbf{f}}{\partial a_{I,2}} & \cdots & \frac{\partial \mathbf{f}}{\partial a_{I,J}} \end{pmatrix} = -\mathbf{E}\mathbf{X}^T\,. \tag{4.4}$$

The partial derivative of $\mathbf{f}$ with respect to a single element $x_{l,m}$ of $\mathbf{X}$, with $1 \le l \le J$ and $1 \le m \le T$, is

$$\begin{aligned}\frac{\partial \mathbf{f}}{\partial x_{l,m}}(\mathbf{A},\mathbf{X}) &= \frac{1}{2}\sum_{i=1}^{I}\left(2\cdot\left(y_{i,m} - \sum_{j=1}^{J} a_{i,j}\cdot x_{j,m}\right)\cdot(-a_{i,l})\right) = \\ &= -\sum_{i=1}^{I}\left((\mathbf{A}^T)_{l,i}\cdot e_{i,m}\right) = \left(-\mathbf{A}^T\mathbf{E}\right)_{l,m}.\end{aligned} \tag{4.5}$$

All the $\frac{\partial \mathbf{f}}{\partial x_{l,m}}$ can be combined to the $J \times T$ matrix

$$\frac{\partial \mathbf{f}}{\partial \mathbf{X}}(\mathbf{A},\mathbf{X}) = -\mathbf{A}^T\mathbf{E}\,. \tag{4.6}$$

Combining (4.4) and (4.6), we get the gradient

$$\nabla\mathbf{f}(\mathbf{A},\mathbf{X}) = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{A}}(\mathbf{A},\mathbf{X}),\, \frac{\partial \mathbf{f}}{\partial \mathbf{X}}(\mathbf{A},\mathbf{X})\right) = \left(-\mathbf{E}\mathbf{X}^T,\, -\mathbf{A}^T\mathbf{E}\right)\,. \tag{4.7}$$

To use the Karush-Kuhn-Tucker conditions from section 3, we express the NMF problem, minimizing $\mathbf{f}(\mathbf{A}, \mathbf{X})$, as minimizing a function $\mathbf{f}_0(\mathbf{n})$, where $\mathbf{n}$ is a vector of length $I \cdot J + J \cdot T$ containing the elements of the matrices $\mathbf{A}$ and $\mathbf{X}$, with

$$\begin{aligned}\mathbf{n} = (&a_{1,1}\ a_{1,2}\ \ldots\ a_{1,J}\ a_{2,1}\ a_{2,2}\ \ldots\ a_{2,J}\ a_{3,1}\ \ldots\ a_{I,J}\ x_{1,1}\ x_{1,2}\ \ldots \\ &x_{1,T}\ x_{2,1}\ x_{2,2}\ \ldots\ x_{2,T}\ x_{3,1}\ \ldots x_{J,T})^T.\end{aligned} \tag{4.8}$$

We define a restructuring function $\mathbf{r} : \mathbb{R}^{IJ+JT} \to \mathbb{R}^{I\times J} \times \mathbb{R}^{J\times T}$ that maps such a vector $\mathbf{n}$ back to a pair of matrices $\mathbf{A}$ and $\mathbf{X}$. Thus we can define $\mathbf{f}_0$ based on this restructuring function $\mathbf{r}$ and $\mathbf{f}$ from (2.1) as

$$\mathbf{f}_0(\mathbf{n}) = \mathbf{f}(\mathbf{r}(\mathbf{n}))\,. \tag{4.9}$$

Based on (4.3) and (4.5), the partial derivatives are

$$\begin{aligned}
i = 1, \ldots, IJ: \quad & \frac{\partial \mathbf{f}_0}{\partial n_i}(\mathbf{n}) &&= \left(-\mathbf{E}\mathbf{X}^T\right)_{p,q} \\
& \text{where} && p = ((i-1) \text{ div } J) + 1 \\
& \text{and} && q = ((i-1) \bmod J) + 1 \\
i = IJ+1, \ldots, IJ+JT: \quad & \frac{\partial \mathbf{f}_0}{\partial n_i}(\mathbf{n}) &&= \left(-\mathbf{A}^T\mathbf{E}\right)_{r,s} \\
& \text{where} && r = ((i-IJ-1) \text{ div } T) + 1 \\
& \text{and} && s = ((i-IJ-1) \bmod T) + 1\,.
\end{aligned} \tag{4.10}$$

The required nonnegativity of the matrices $\mathbf{A}$ and $\mathbf{X}$ in the definition of NMF means that all elements of $\mathbf{n}$ have to be nonnegative, which can be expressed using $IJ + JT$ constraint functions $\mathbf{f}_i : \mathbb{R}^{IJ+JT} \to \mathbb{R}$, where for $i = 1, \ldots, (IJ + JT)$

$$\mathbf{f}_i(\mathbf{n}) = -n_i \tag{4.11}$$

with the requirement

$$\mathbf{f}_i(\mathbf{n}) \leq 0\,. \tag{4.12}$$

This way, the nonnegativity constraints fit the structure of the minimization problem in (3.1). The partial derivatives of these constraint functions are

$$\begin{aligned}
&\frac{\partial \mathbf{f}_i}{\partial n_i}(\mathbf{n}) = -1, \qquad i = 1, \ldots, (IJ + JT) \\
&\frac{\partial \mathbf{f}_i}{\partial n_j}(\mathbf{n}) = 0, \qquad i, j = 1, \ldots, (IJ + JT), \qquad i \neq j.
\end{aligned} \tag{4.13}$$

Based on the mapping from $\mathbf{n}$ to the matrices, we can express these constraints as

$$\begin{aligned}
i = 1, \ldots, IJ: \quad & \mathbf{f}_i(\mathbf{n}) &&= -a_{p,q} \\
& \text{where} && p = ((i-1) \text{ div } J) + 1 \\
& \text{and} && q = ((i-1) \bmod J) + 1 \\
i = IJ+1, \ldots, IJ+JT: \quad & \mathbf{f}_i(\mathbf{n}) &&= -x_{r,s} \\
& \text{where} && r = ((i-IJ-1) \text{ div } T) + 1 \\
& \text{and} && s = ((i-IJ-1) \bmod T) + 1\,.
\end{aligned} \tag{4.14}$$

For the NMF problem, there are no constraints $\mathbf{h}_i(\mathbf{n}) = 0$. The Lagrangian is therefore $\mathbf{L} : \mathbb{R}^{IJ+JT} \times \mathbb{R}^{IJ+JT} \to \mathbb{R}$ with

$$\mathbf{L}(\mathbf{n}, \lambda) = \mathbf{f}_0(\mathbf{n}) + \sum_{i=1}^{IJ+JT} \lambda_i \mathbf{f}_i(\mathbf{n}) \,. \tag{4.15}$$

The partial derivative of the Lagrangian with respect to an element n_i of $\mathbf{n}$ is, based on (4.10) and (4.13)

$$\begin{array}{rlcl} i = 1, \ldots, IJ: & \dfrac{\partial \mathbf{L}}{\partial n_i}(\mathbf{n}, \lambda) & = & \left(-\mathbf{E}\mathbf{X}^T\right)_{p,q} - \lambda_i \\ & \text{where} & & p = ((i-1) \text{ div } J) + 1 \\ & \text{and} & & q = ((i-1) \bmod J) + 1 \\ i = IJ+1, \ldots, IJ+JT: & \dfrac{\partial \mathbf{L}}{\partial n_i}(\mathbf{n}, \lambda) & = & \left(-\mathbf{A}^T\mathbf{E}\right)_{r,s} - \lambda_i \\ & \text{where} & & r = ((i-IJ-1) \text{ div } T) + 1 \\ & \text{and} & & s = ((i-IJ-1) \bmod T) + 1 \,. \end{array} \tag{4.16}$$

For a primal optimal $\mathbf{x}^\star$ and a dual optimal $\lambda^\star$ with zero duality gap, all the $\frac{\partial \mathbf{L}}{\partial n_i}(\mathbf{n}^\star, \lambda^\star)$ are 0 and therefore

$$\begin{array}{rlcl} i = 1, \ldots, IJ: & \lambda_i^\star & = & \left(-\mathbf{E}\mathbf{X}^T\right)_{p,q} \\ & \text{where} & & p = ((i-1) \text{ div } J) + 1 \\ & \text{and} & & q = ((i-1) \bmod J) + 1 \\ i = IJ+1, \ldots, IJ+JT: & \lambda_i^\star & = & \left(-\mathbf{A}^T\mathbf{E}\right)_{r,s} \\ & \text{where} & & r = ((i-IJ-1) \text{ div } T) + 1 \\ & \text{and} & & s = ((i-IJ-1) \bmod T) + 1 \,. \end{array} \tag{4.17}$$

For two matrices $\mathbf{A}$ and $\mathbf{B}$ of size $I \times T$, their Hadamard product $\mathbf{C} = \mathbf{A} \circledast \mathbf{B}$ is another matrix of size $I \times T$, where for all indices i and t, $c_{i,t} = a_{i,t} \cdot b_{i,t}$. Using this notation, we get, based on (4.14) and (4.17),

$$\begin{array}{rlcl} i = 1, \ldots, IJ: & \lambda_i^\star \, \mathbf{f}_i(\mathbf{n}^\star) & = & \left(-\mathbf{A} \circledast \left(\mathbf{E}\mathbf{X}^T\right)\right)_{p,q} \\ & \text{where} & & p = ((i-1) \text{ div } J) + 1 \\ & \text{and} & & q = ((i-1) \bmod J) + 1 \\ i = IJ+1, \ldots, IJ+JT: & \lambda_i^\star \, \mathbf{f}_i(\mathbf{n}^\star) & = & \left(-\mathbf{X} \circledast \left(\mathbf{A}^T\mathbf{E}\right)\right)_{r,s} \\ & \text{where} & & r = ((i-IJ-1) \text{ div } T) + 1 \\ & \text{and} & & s = ((i-IJ-1) \bmod T) + 1 \,. \end{array} \tag{4.18}$$

Following (3.8), the KKT optimality conditions are

$$\begin{array}{rcll} \mathbf{f}_i(\mathbf{n}^\star) & \leq & 0, & i = 1, \ldots, m \\ \lambda_i^\star & \geq & 0, & i = 1, \ldots, m \\ \lambda_i^\star \mathbf{f}_i(\mathbf{n}^\star) & = & 0, & i = 1, \ldots, m \\ \nabla \mathbf{f}_0(\mathbf{n}^\star) + \sum_{i=1}^m \lambda_i^\star \nabla \mathbf{f}_i(\mathbf{n}^\star) & = & 0 \,. & \end{array} \tag{4.19}$$

The first and the fourth condition were already expressed and used above. The second condition means that we have, based on (4.17) and the definition of $\mathbf{E}$ in (4.1),

$$\mathbf{E}\mathbf{X}^T = \mathbf{Y}\mathbf{X}^T - \mathbf{A}\mathbf{X}\mathbf{X}^T \leq 0 \tag{4.20}$$

and

$$\mathbf{A}^T\mathbf{E} = \mathbf{A}^T\mathbf{Y} - \mathbf{A}^T\mathbf{A}\mathbf{X} \leq 0\,. \tag{4.21}$$

The third condition, (4.18) and the definition of $\mathbf{E}$ in (4.1) lead to the two equations

$$\mathbf{A} \circledast \left(\mathbf{Y}\mathbf{X}^T - \mathbf{A}\mathbf{X}\mathbf{X}^T\right) = \mathbf{0}\,. \tag{4.22}$$

$$\mathbf{X} \circledast \left(\mathbf{A}^T\mathbf{Y} - \mathbf{A}^T\mathbf{A}\mathbf{X}\right) = \mathbf{0}\,. \tag{4.23}$$

Note that the simply setting $\mathbf{A} = \mathbf{0}$ to fulfill (4.22) is, in general, no viable solution. Because $\mathbf{Y}$ and $\mathbf{X}$ are both nonnegative, $\mathbf{Y}\mathbf{X}^T$ would then have to be $\mathbf{0}$ as well to satisfy (4.20). Similarly, setting $\mathbf{X} = \mathbf{0}$ to fulfill (4.23) would require $\mathbf{A}^T\mathbf{Y} = \mathbf{0}$ to satisfy (4.21). More importantly, although setting both $\mathbf{A} = \mathbf{0}$ and $\mathbf{X} = \mathbf{0}$ trivially satisfies all the KKT conditions, it is not an optimal solution, since the KKT conditions are necessary, but not sufficient, optimality conditions for the non-convex NMF problem.

5. NMF via Newton iteration. Defining the four matrices

$$\mathbf{B} = \mathbf{Y}\mathbf{X}^T\,,\quad \mathbf{C} = \mathbf{X}\mathbf{X}^T\,,\quad \mathbf{R} = \mathbf{A}^T\mathbf{Y}\,,\quad \mathbf{S} = \mathbf{A}^T\mathbf{A}\,, \tag{5.1}$$

the conditions (4.20), (4.21), (4.22) and (4.23) can be written as

$$\mathbf{B} - \mathbf{A}\mathbf{C} \leq 0\,, \tag{5.2}$$

$$\mathbf{R} - \mathbf{S}\mathbf{X} \leq 0\,, \tag{5.3}$$

$$\mathbf{A} \circledast (\mathbf{B} - \mathbf{A}\mathbf{C}) = \mathbf{0}\,, \tag{5.4}$$

$$\mathbf{X} \circledast (\mathbf{R} - \mathbf{S}\mathbf{X}) = \mathbf{0}\,. \tag{5.5}$$

In general, it is not possible to solve (5.4) by simply setting $\mathbf{A} = \mathbf{B}\mathbf{C}^{-1}$, because $\mathbf{C}^{-1}$, the inverse of the nonnegative matrix $\mathbf{C}$, may have negative elements, so $\mathbf{B}\mathbf{C}^{-1}$ may have negative elements, too, but $\mathbf{A}$ is required to be nonnegative. For the same reason, $\mathbf{X} = \mathbf{S}^{-1}\mathbf{R}$ is in general no valid solution of (5.5).

The NMF algorithm based on Newton iteration approximates equations (5.4) and (5.5) via alternating direction iteration, i.e., first the matrix $\mathbf{X}$ is fixed to compute a new $\mathbf{A}$ with the goal of fulfilling (5.4) while also taking (5.2) into account, then this $\mathbf{A}$ is fixed to compute a new $\mathbf{X}$ to approximate (5.5) with consideration to (5.3). This process is then repeated until some termination condition is met.

We want to focus on the computation of $\mathbf{A}$ first: (5.4) has the useful property that every row of $\mathbf{A}$ is independent of the others, so all rows can be processed separately and in parallel. So, every row $\mathbf{a}$ of $\mathbf{A}$ has to satisfy

$$\mathbf{a} \circledast (\mathbf{b} - \mathbf{a}\mathbf{C}) = \mathbf{0} \tag{5.6}$$

where $\mathbf{b}$ is the corresponding row of $\mathbf{B}$. The size of the matrix $\mathbf{A}$ is $I \times J$, where I is the number of rows of the input matrix, typically a large value, and J is the chosen parameter for NMF, which is usually much smaller. Therefore, the problem can be split up into many small problems, each of them requires only working with vectors (matrix rows) of size J and the matrix $\mathbf{C}$ of size $J \times J$. The vector $\mathbf{a}$ can be written

as $\mathbf{a} = \mathbf{z} \circledast \mathbf{z}$, where $\mathbf{z}$ does not have to fulfill a nonnegativity constraint. Because $\mathbf{C} = \mathbf{X}\mathbf{X}^T$ is symmetric and interpreting $\mathbf{z}$ and $\mathbf{b}$ as column vectors, the condition can be written as

$$(\mathbf{C}\,(\mathbf{z} \circledast \mathbf{z}) - \mathbf{b}) \circledast \mathbf{z} =: \mathbf{d}(\mathbf{z}) = \mathbf{0}\,. \tag{5.7}$$

The part of the constraint (5.2) concerning this single row is transformed into

$$\mathbf{C}\,(\mathbf{z} \circledast \mathbf{z}) - \mathbf{b} \geq \mathbf{0}\,. \tag{5.8}$$

The goal is now to find a root of the function $\mathbf{d}$. To use Newton iteration, we need the associated Jacobi matrix. For $j = 1, 2, \ldots, J$, component d_j of $\mathbf{d}$ can be written as

$$d_j = \left(\left(\sum_{k=1}^{J} c_{j,k} \cdot z_k^2\right) - b_j\right) \cdot z_j\,. \tag{5.9}$$

Therefore, we get for $i, j = 1, 2, \ldots, J$ with $i \neq j$

$$\frac{\partial d_j}{\partial z_i} = 2 \cdot c_{j,i} \cdot z_i \cdot z_j = (2\; diag(\mathbf{z})\,\mathbf{C}\; diag(\mathbf{z}))_{j,i} \tag{5.10}$$

which is the element of the matrix $(2\; diag(\mathbf{z})\,\mathbf{C}\; diag(\mathbf{z}))$ with row index j and column index i, where $diag(\mathbf{z})$ is a diagonal matrix of size $J \times J$ with the elements of the vector $\mathbf{z}$ on the main diagonal.

For $i = j$, the partial derivative is

$$\begin{aligned} \frac{\partial d_j}{\partial z_j} &= 2 \cdot c_{j,j} \cdot z_j \cdot z_j + \left(\left(\sum_{k=1}^{J} c_{j,k} \cdot z_k^2\right) - b_j\right) = \\ &= (2\; diag(\mathbf{z})\,\mathbf{C}\; diag(\mathbf{z}))_{j,j} + (\mathbf{C}\,(\mathbf{z} \circledast \mathbf{z}) - \mathbf{b})_j = \\ &= (2\; diag(\mathbf{z})\,\mathbf{C}\; diag(\mathbf{z}) + diag(\mathbf{C}\,(\mathbf{z} \circledast \mathbf{z}) - \mathbf{b}))_{j,j}\,. \end{aligned} \tag{5.11}$$

Since $(diag(\mathbf{C}\,(\mathbf{z} \circledast \mathbf{z}) - \mathbf{b}))_{j,i} = 0$ for $i \neq j$, we can combine (5.10) and (5.11), so for all $i, j = 1, 2, \ldots, J$

$$\frac{\partial d_j}{\partial z_i} = (2\; diag(\mathbf{z})\,\mathbf{C}\; diag(\mathbf{z}) + diag(\mathbf{C}\,(\mathbf{z} \circledast \mathbf{z}) - \mathbf{b}))_{j,i}\,. \tag{5.12}$$

Therefore, the Jacobi matrix is

$$\begin{aligned} \mathbf{J}_{\mathbf{d}}(\mathbf{z}^{(i)}) = \frac{\partial \mathbf{d}}{\partial \mathbf{z}}(\mathbf{z}^{(i)}) &= \frac{\partial(d_1, d_2, \ldots, d_J)}{\partial(z_1, z_2, \ldots, z_J)}(\mathbf{z}^{(i)}) = \\ &= 2\; diag(\mathbf{z})\,\mathbf{C}\; diag(\mathbf{z}) + diag(\mathbf{C}\,(\mathbf{z} \circledast \mathbf{z}) - \mathbf{b})\,, \end{aligned} \tag{5.13}$$

so it is possible to use Newton iteration to find the root for this row, and also for the other rows in parallel. For every row a standard Newton iteration step consists of solving the following linear system for $\Delta\mathbf{z}^{(i)}$:

$$\mathbf{J}_{\mathbf{d}}(\mathbf{z}^{(i)}) \cdot \Delta\mathbf{z}^{(i)} = -\mathbf{d}(\mathbf{z}^{(i)}) \tag{5.14}$$

and updating $\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} + \Delta\mathbf{z}^{(i)}$. Taking the constraint (5.8) into account, we get Algorithm 1. In line 2 of the algorithm, an initial $\mathbf{z}$ that satisfies this constraint has

to be generated. One possible way to achieve this is by assigning random values, checking whether the constraint is fulfilled, and repeating the assignment and the constraint check as often as necessary. Since $\mathbf{C}$ and $\mathbf{b}$ are both nonnegative, doubling the range of the random number generator with each repetition makes the fulfillment of the constraint increasingly likely, so usually the needed number of repetitions is low. In line 6, a suitable α can be determined by starting with $\alpha = 1$ and checking the constraint, and, if necessary, repeatedly dividing α by 2 and checking again. If α falls below a threshold, the loop from line 3 to 8 can be ended since in this case the solution for this row is stable. A fixed maximum number of iterations is an additional suitable local termination criterion with low overhead. In principle, it is also possible to use the global goal (2.1) as termination criterion, but the cost of such a check makes this inadvisable.

Algorithm 1: Newton iteration for the rows of $\mathbf{A}$

```
1  for every row a of A do in parallel
       // Treat z and the row b of B as column vectors.
2      Find z such that C (z ⊛ z) − b ≥ 0;
3      repeat
4          g ← C (z ⊛ z) − b;
           // Solve linear system from (5.14):
5          Solve for h: (2 diag(z) C diag(z) + diag(g)) h = −g ⊛ z;
6          Find α > 0 such that C ((z + αh) ⊛ (z + αh)) − b ≥ 0;
7          z ← z + αh;
8      until local termination criterion is met;
       // Treat z as row vector.
9      a ← z ⊛ z;
10 end
```

The computation of $\mathbf{X}$ works in a similar way. Here, all columns of $\mathbf{X}$ can be computed independently (as opposed to $\mathbf{A}$, where we had independent rows). Based on (5.5), we get for every column $\mathbf{x}$ of $\mathbf{X}$

$$\mathbf{x} \circledast (\mathbf{r} - \mathbf{S}\mathbf{x}) = \mathbf{0} \tag{5.15}$$

where $\mathbf{r}$ is the corresponding column of $\mathbf{R}$. The matrix $\mathbf{X}$ has the size $J \times T$, where T is the (potentially very large) number of columns of the input matrix, and J is the (usually significantly smaller) chosen parameter for NMF. Therefore, the computation of $\mathbf{X}$ can again be split up into many (up to T) small problems, each of them requiring only vectors of size J and the matrix $\mathbf{S}$ of size $J \times J$. Again, we use the trick to express the nonnegative vector $\mathbf{x}$ as $\mathbf{x} = \mathbf{u} \circledast \mathbf{u}$, with no nonnegativity constraint on $\mathbf{u}$. Then we get

$$(\mathbf{S}\,(\mathbf{u} \circledast \mathbf{u}) - \mathbf{r}) \circledast \mathbf{u} =: \mathbf{t}(\mathbf{u}) = \mathbf{0}\,. \tag{5.16}$$

The part of the constraint (5.3) concerning this single column is transformed into

$$\mathbf{S}\,(\mathbf{u} \circledast \mathbf{u}) - \mathbf{r} \geq \mathbf{0}\,. \tag{5.17}$$

Since the function $\mathbf{t}$ from (5.16) has the same structure as $\mathbf{d}$ from (5.7), we can easily infer its Jacobi matrix from (5.13):

$$\mathbf{J}_{\mathbf{t}}(\mathbf{u}^{(i)}) = \frac{\partial \mathbf{t}}{\partial \mathbf{u}}(\mathbf{u}^{(i)}) = 2\, diag(\mathbf{u})\, \mathbf{S}\, diag(\mathbf{u}) + diag(\mathbf{S}\,(\mathbf{u} \circledast \mathbf{u}) - \mathbf{r})\,. \tag{5.18}$$

Thus we can express the algorithm to compute the columns of $\mathbf{X}$ as shown in Algorithm 2. For finding the initial $\mathbf{u}$ in line 2, α in line 6 and for the termination criterion, the ideas discussed for Algorithm 1 can be reused.

Algorithm 2: Newton iteration for the columns of $\mathbf{X}$

```
1  for every column x of X do in parallel
2  |  Find u such that S(u ⊛ u) − r ≥ 0;
3  |  repeat
4  |  |  g ← S(u ⊛ u) − r;
5  |  |  Solve linear system for h: (2 diag(u) S diag(u) + diag(g)) h = −g ⊛ u;
6  |  |  Find α > 0 such that S((u + αh) ⊛ (u + αh)) − r ≥ 0;
7  |  |  u ← u + αh;
8  |  until local termination criterion is met;
9  |  x ← u ⊛ u;
10 end
```

All that remains to be done in between these alternating parallel Newton iterations is the computation of the parameter matrices $\mathbf{B}$, $\mathbf{C}$, $\mathbf{R}$ and $\mathbf{S}$. The complete NMF algorithm is shown below in Algorithm 3, where we have parallel computations of $\mathbf{A}$ and $\mathbf{X}$ interspersed with short sequential parts to compute the parameter matrices. This can be accomplished by starting with a single thread that is at some point split up into multiple threads to compute a parallel part. Afterwards, these threads are joined together into a single thread again. This splitting and joining is repeated multiple times. Such a behavior is very well-suited for a shared-memory programming model such as OpenMP, and easy to implement using such an interface. Possible global termination criteria are a fixed number of iterations or a threshold for the total approximation error as given by (2.1).

Algorithm 3: Parallel NMF via Newton iteration using shared memory

```
1  X ← rand(J, T);
2  repeat
3  |  B ← YX^T;
4  |  C ← XX^T;
5  |  Compute new A in parallel as shown in Algorithm 1;
6  |  R ← A^T Y;
7  |  S ← A^T A;
8  |  Compute new X in parallel as shown in Algorithm 2;
9  until global termination criterion is met;
```

We note that it is also possible to compute an NMF via Newton iteration in a distributed environment using message passing, as discussed in [6] and [3]. A comparison of the two parallelization methods would require a detailed analysis of the MPI algorithm. This is beyond the scope of this paper, so it will be a matter of a future publication.

6. Experiments. The algorithm described above was implemented using the programming language C, OpenMP and the AMD Core Math Library (ACML). While the implementation also allows checking the error $||A - WH||_F$ as termination criterion, for the measurements, we chose to use a fixed number of iterations. The measurements were done with an input matrix of size $24\,576 \times 24\,576$, with $J = 1\,536$. The program runs on a single node, which means that we could use up to 64 processor cores in our setup (four AMD Opteron 6274 processors with 16 cores each). Table 6.1 contains

the results, and Figure 6.1 shows their graphical representation. All measurements where executed at least three times, the table shows the median results. To solve this relatively large problem, a single core needs about $1.1 \cdot 10^6$ seconds (13 days). On 64 cores, the runtime is approximately 10 hours. The efficiency continuously decreases with the number of processor cores, but the worst-case efficiency of 47.7% for 64 cores (a speedup of 30.5) is probably still acceptable for practical purposes.

TABLE 6.1
Speedup and efficiency for shared-memory NMF for $I = 24\,576$, $T = 24\,576$, $J = 1\,536$.

cores	duration [s]	speedup	efficiency
1	1 147 090	1.00	1.000
2	595 005	1.93	0.964
4	306 629	3.74	0.935
8	155 077	7.40	0.925
16	83 284	13.8	0.861
32	53 946	21.3	0.664
64	37 561	30.5	0.477

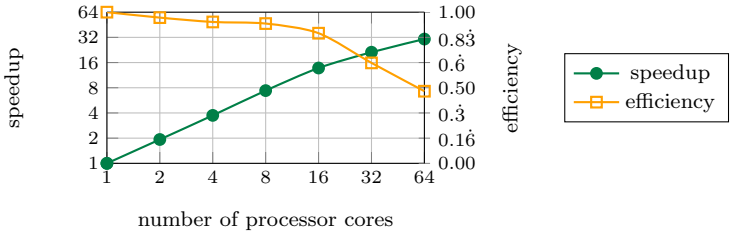

FIGURE 6.1. *Speedup and efficiency for shared-memory NMF (graph).*

7. Conclusion. We showed in detail how the application of the general Karush-Kuhn-Tucker conditions for first-order optimality to the NMF problem can lead to an NMF algorithm based on Newton iteration. This algorithm can be executed in parallel on shared-memory systems, in addition to its use for message-passing systems that was presented previously. Speedup measurements show acceptable performance for relatively large workloads.

REFERENCES

[1] S. BOYD AND L. VANDENBERGHE, *Convex Optimization*, Cambridge University Press, 7th ed., 2009.
[2] M. CHU, F. DIELE, R. PLEMMONS, AND S. RAGNI, *Optimality, Computation, and Interpretations of Nonnegative Matrix Factorizations*, Unpublished Report, (2004).
[3] M. FLATZ AND M. VAJTERŠIC, *A parallel algorithm for Nonnegative Matrix Factorization based on Newton iteration*, in Proceedings of the IASTED International Conference Parallel and Distributed Computing and Networks (PDCN 2013), ACTA Press, 2013, pp. 600–607.
[4] P. O. HOYER., *Non-negative matrix factorization with sparseness constraints*, Journal of Machine Learning Research, 5 (2004), pp. 1457–1469.
[5] D. D. LEE AND H. S. SEUNG, *Learning the parts of objects by non-negative matrix factorization*, Nature, 401 (1999), pp. 788–791.
[6] G. OKŠA, M. BEČKA, AND M. VAJTERŠIC, *Nonnegative Matrix Factorization: Algorithms and Parallelization*, Tech. Report 2010-05, University of Salzburg, Department of Computer Sciences, 2010.
[7] P. PAATERO AND U. TAPPER, *Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values*, Environmetrics, 5 (1994), pp. 111–126.