NONNEGATIVE MATRIX FACTORIZATION

SOFÍA LLAVAYOL

Abstract. This document presents the final project for the course *Numerical Linear Algebra for Statistical Learning* at Universidad de la República, Uruguay. It outlines fundamental concepts of Nonnegative Matrix Factorization based on the reference [2]. A custom implementation of the multiplicative update algorithm by Lee and Seung [3] is developed in Python and applied to the Labeled Faces in the Wild (LFW) dataset. The project includes experiments on image reconstruction, component interpretability, and analysis of reconstruction error and sparsity as functions of the factorization rank.

Contents

1. Introduction

Nonnegative matrix factorization (NMF) is an easily interpretable *linear dimensionality reduction (LDR)* technique for nonnegative data. We first introduce the general concept of LDR, followed by a more detailed discussion of NMF.

1.1. LDR techniques for Data Analysis.

Extracting the underlying structure within data sets is one of the central problems in data science, and numerous techniques exist to perform this task. One of the oldest approaches is LDR. The idea of LDR is to represent each data point as a linear combination of a small number of basis elements.

Date: June 25, 2025.

Mathematically, given a dataset of n data points $x_1, \dots, x_n \in \mathbb{R}^m$, LDR looks for $r \ll \min\{m, n\}$ basis vectors $w_1, \dots, w_r \in \mathbb{R}^m$ such that each data point x_j is well-approximated by a linear combination of these basis vectors:

$$x_j \approx w_1 \cdot h_{1j} + \cdots + w_r \cdot h_{rj} = \begin{bmatrix} w_1 \cdots w_r \end{bmatrix} \begin{bmatrix} h_{1j} \\ \vdots \\ h_{rj} \end{bmatrix} = W h_j,$$

where $h_j = \begin{bmatrix} h_{1j}, \dots, h_{rj} \end{bmatrix}^T \in \mathbb{R}^r$ contains the coordinates of x_j in the reduced basis.

This model can be written compactly as $X \approx WH$, where

- each column of $X \in \mathbb{R}^{m \times n}$ is a data point, $X(:, j) = x_j$;
- each column of $W \in \mathbb{R}^{m \times r}$ is a basis element, $W(:, j) = w_j$;
- each column of $H \in \mathbb{R}^{r \times n}$ contains the coordinates of a data point x_j in the basis W, $H(:, j) = h_j$.

Hence, LDR produces a rank-r approximation $X \approx WH$ of the form:

$$\begin{bmatrix} x_1 \cdots x_n \end{bmatrix} \approx \begin{bmatrix} w_1 \cdots w_r \end{bmatrix} \begin{bmatrix} h_1 \cdots h_n \end{bmatrix},$$

which is a *low-rank matrix approximation (LRMA)*. Conversely, any rank-r matrix $X_r \in \mathbb{R}^{m \times n}$ can be factored as $X_r = WH$ with $W \in \mathbb{R}^{m \times r}$ and $H \in \mathbb{R}^{r \times n}$. Therefore, LDR and LRMA are equivalent formulations of the same problem.

In order to compute W and H given X and r, one needs to define an error measure. For example, when (W, H) minimizes the Frobenius (or Euclidean) norm

$$\|X - WH\|_F^2 = \sum_{i,j} (X - WH)_{ij}^2,$$

then LRMA is equivalent to *principal component analysis (PCA)*, which can be computed via the *singular value decomposition (SVD)*. Recall that in PCA the optimal rank-r approximation X_r of X that minimizes the Frobenius norm is given by the truncated SVD of X, obtained by keeping the top r singular values and the corresponding singular vectors. That is, $X_r = U_r \Sigma_r V_r^T$ where

- $U_r \in \mathbb{R}^{m \times r}$ has the first r left singular vectors as columns;
- $\Sigma_r \in \mathbb{R}^{r \times r}$ has the top r singular values on the diagonal;
- $V_r \in \mathbb{R}^{n \times r}$ has the first r right singular vectors as columns.

In this case, one can set $W = U_r \Sigma_r \in \mathbb{R}^{m \times r}$ and $H = V_r^T \in \mathbb{R}^{r \times n}$ to obtain LDR.

LRMA models are used to compress the data, filter the noise, reduce the computational effort for further manipulation of the data, or to directly identify hidden structure in a data set. Many variants of LRMA have been developed, and they differ in two key aspects: (1) the error measure can vary and should be chosen depending on the noise statistic assumed on the data, (2) different constraints can be imposed on the factors W and H.

1.2. **NMF, an LDR technique for nonnegative data.** Among LRMA models, nonnegative matrix factorization requires the factor matrices W and H to be componentwise nonnegative, which we denote $W \geq 0$ and $H \geq 0$. In Section 2, we discuss an application where these nonnegativity constraints are natural and meaningful.

Formally, the NMF problem is defined as follows: find matrices (W, H) that minimize

$$\min_{\substack{W \in \mathbb{R}^{m \times r} \\ H \in \mathbb{R}^{r \times n}}} d(X, WH) \quad \text{subject to } W \geq 0 \text{ and } H \geq 0, \tag{1}$$

where $d(\cdot, \cdot)$ is a measure of approximation error (e.g., the Frobenius norm).

In Section 3, we discuss an algorithm to approximately solve this problem when the distance measure is given by the Frobenius norm. An application of this algorithm to image processing is presented in Section 2.

1.2.1. *Sparcity.* Because of the nonnegativity constraints, NMF solutions (W, H) are expected to contain zero entries and hence to naturally have some degree of sparsity; see Figure 1 for examples. Mathematically, this is explained by the *first-order optimality conditions* of a smooth optimization problem with nonnegativity constraints

$$\min_{x \in \mathbb{R}^M} f(x) \quad \text{subject to } x \geq 0, \tag{2}$$

which are given by

$$x \geq 0, \quad \nabla f(x) \geq 0 \quad \text{and} \quad x_i \cdot (\nabla f(x))_i = 0 \text{ for all } i. \tag{3}$$

This enforces $x_i = 0$ whenever $(\nabla f(x))_i > 0$.

Let us see that (3) are indeed the conditions given in (8) associated to the problem (2). The constraints are given by $g_i(x) = -x_i \leq 0$ for $i = 1, \ldots, M$, and so the gradient $\nabla g_i(x)$ equals the oposite of the canonical basis vector, $-e_i \in \mathbb{R}^M$. Hence, the stationarity condition reads

$$0 = \nabla f(x) - \sum_{i=1}^{M} \lambda_i e_i = \sum_{i=1}^{M} \Big((\nabla f(x))_i - \lambda_i \Big) e_i.$$

As $(\nabla f(x))_i = \lambda_i$ for all $i = 1, \ldots, M$, the dual feasibility condition gives $(\nabla f(x))_i \geq 0$, and the complementary slackness gives $0 = x_i \cdot (\nabla f(x))_i$.

2. Application on Facial Feature Extraction

Given a set of gray-scale images of the same dimensions, let us construct the matrix X such that each column of X corresponds to a vectorized gray-level image. Vectorization means that the two-dimensional images are transformed into a long one-dimensional vector, for example, by stacking the columns of the image on top of each other. Hence, the entry $X(i, j)$ is equal to the intensity of the ith pixel within the jth image, which is nonnegative.

In this work, we used the *Labeled Faces in the Wild (LFW)* dataset, a well-known collection of face photographs designed for studying face recognition under unconstrained conditions.[1] We

[1]Face recognition under unconstrained conditions refers to the process of identifying or verifying a person's identity using facial features when the environment and subject conditions are not controlled. This contrasts with constrained conditions, like passport photos, where lighting, pose, and facial expression are standardized.

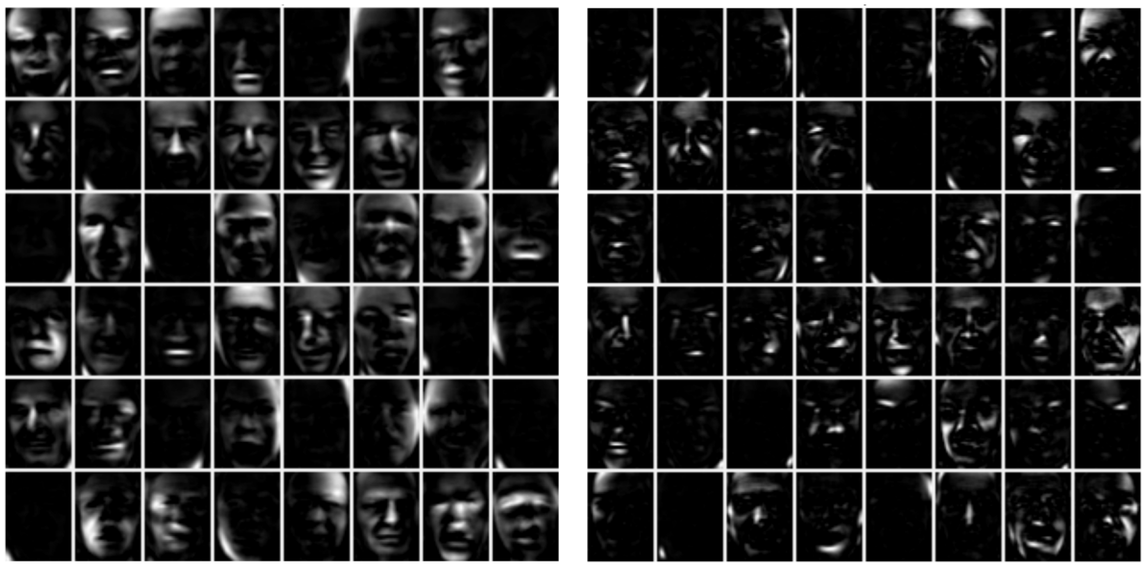

FIGURE 1. Visualization of selected components from the NMF decomposition of facial images. Left: components from a rank-50 factorization. Right: components from a rank-200 factorization. Each component is reshaped into the original image size of (50, 37).

loaded the dataset using the `fetch_lfw_people` function from `scikit-learn`, with the following parameters:

- `min_faces_per_person=70`: this restricts the dataset to individuals who appear in at least 70 photographs, ensuring a sufficient number of images per class and reducing data imbalance.
- `resize=0.4`: the original images were resized to 40% of their original size in both dimensions to reduce computational cost. As a result, each image has a shape of $(50, 37)$ pixels.

With these settings, the resulting dataset consists of $n = 1288$ grayscale images, each represented as a vector of $m = 1850$ features (i.e., flattened 50×37 pixel arrays). We call X the resulting matrix.

2.1. **The reconstruction error.** We applied NMF on X for different ranks. The algorithm used is described in Section 3. In Figure 2 we can observe the relation between the error and the rank. As expected, the error decreases with the rank.

The curve exhibits a convex shape, with the error decreasing rapidly at first and then more gradually, suggesting diminishing returns as the rank increases. According to the *elbow method*, the knee point appears at rank 50, indicating a good trade-off between reconstruction accuracy and model complexity.

Figure 3 shows examples of reconstructed images for ranks 20, 50, and 200, alongside the original images. As expected, increasing the rank improves the visual quality of the reconstruction.

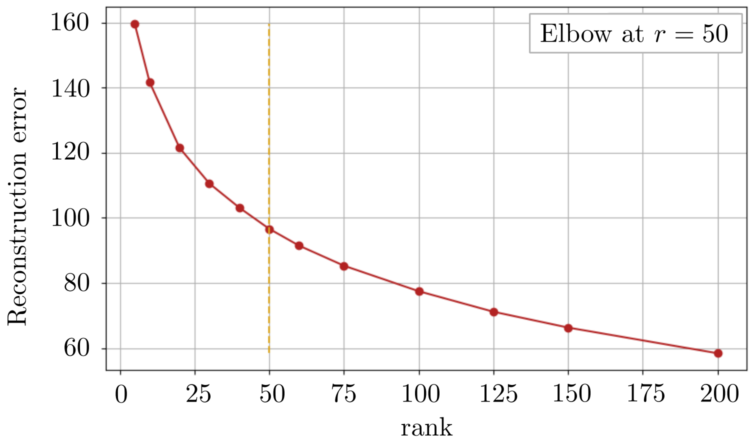

FIGURE 2. Reconstruction error versus rank for the LFW dataset using NMF. The error is measured as $\|X - WH\|_F$, where $X \approx WH$ is the rank-r NMF approximation. An elbow point is observed at rank 50.

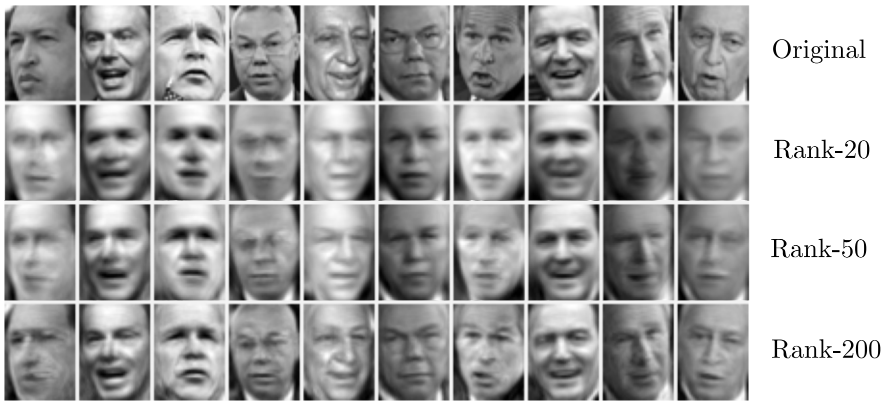

FIGURE 3. Original images (top row) and their reconstructions using NMF with ranks 20, 50, and 200 (subsequent rows). Even with a low-rank approximation, the reconstructions preserve key facial features and expressions.

However, even with a low rank such as 20, the model captures the essential facial structure and expressions—such as the position of the eyes, mouth, and eyebrows.

2.2. **The basis elements.** On the other hand, we are interested in analyzing the r components $w_1, \dots, w_r$ obtained through NMF. Recall that each component has the same dimensionality as the columns of X, where each column represents a $(50, 37)$ image. Therefore, each component can also be interpreted as an image of shape $(50, 37)$. In Figure 1, we display the basis elements reshaped to $(50, 37)$ for ranks $r = 50$ and $r = 200$.

In the case of rank $r = 200$, we observe that many components correspond to interpretable facial features such as lips, cheeks, nose segments, eyebrows, chin, and forehead. In addition, several

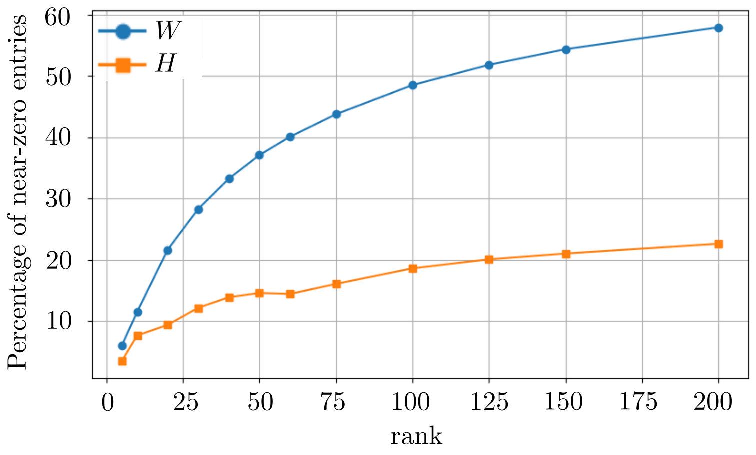

FIGURE 4. Sparsity of W and H versus rank, measured as the fraction of entries below a threshold of 10^{-3}.

components capture lighting or shadow patterns and other structural elements of faces. Since the number of components is relatively large, the decomposition is able to represent fine-grained details and localized features. In contrast, with rank $r = 50$, we observe a similar decomposition, but the components tend to be more diffuse, and the facial parts are less distinctly separated. This reflects the reduced expressive capacity of the model with fewer components.

2.3. **Sparcity.**

To assess the sparsity of the factorized matrices, we computed the fraction of near-zero entries (defined as values smaller than 10^{-3}) in both W and H for different ranks. The plots in Figure 4 show an increase in the fraction of near-zero entries as the rank increases. However, the rate of increase diminishes as the rank becomes higher. This results in a concave trend, where most of the sparsity gain is achieved at lower to moderate ranks. Beyond that point, adding more components yields smaller increases in sparsity. This insight can be useful when selecting the rank in practice: it may be advantageous to choose a lower rank that captures sufficient structure while encouraging sparsity and interpretability.

3. ALGORITHM WITH MULTIPLICATIVE UPDATES

Let $X \in \mathbb{R}^{m\times n}$ and $r \ll \min\{m, n\}$ be given. In this section, we focus on the following constrained optimization problem

$$\min_{\substack{W\in\mathbb{R}^{m\times r}\\ H\in\mathbb{R}^{r\times n}}} f(W,H) \quad \text{subject to } W \geq 0 \text{ and } H \geq 0, \tag{4}$$

where $f(W,H) = \frac{1}{2}\|X - WH\|_F^2$. Note that this problem is equivalent to (1) for the case where $d(X,WH) = \|X - WH\|_F$ is the Euclidean distance. We will apply the KKT conditions, as described in Appendix A.

3.1. **KKT conditions.**

We will start by applying the conditions in (8) for the variables $W = (W_{ik})$. The equations and reasoning will be similar to the ones given in (3). Note that the constraints for

```python
import numpy as np

def multiplicative_updates(X, r, max_iter=2000, random_state=0):
    m, n = X.shape

    rng = np.random.default_rng(random_state) # Initialize local random generator

    # Initialize W and H with values in [0, 1)
    W = rng.random((m, r))
    H = rng.random((r, n))

    eps = 1e-10 # Small constant to avoid division by zero

    for i in range(max_iter):
        # Update rules for W and H
        W *= (X @ H.T) / np.maximum(W @ H @ H.T, eps)
        H *= (W.T @ X) / np.maximum(W.T @ W @ H, eps)

    return W, H
```

LISTING 1. Algorithm with multiplicative updates.

this problem are given by $g_{ik}(W) = -W_{ik} \leq 0$. The stationarity condition then reads

$$0 = \frac{\partial f}{\partial W_{ik}} + \sum \lambda_{rs} \cdot \frac{\partial g_{rs}}{\partial W_{ik}} = \frac{\partial f}{\partial W_{ik}} + \lambda_{ik} \cdot (-1) = (\nabla_W f)_{ik} - \lambda_{ik}$$

for all i, k. Hence, writing $\Lambda_W = (\lambda_{ik})$ and computing the gradient $\nabla_W f$, we obtain

$$\Lambda_W = \nabla_W f = -(X - WH)H^T = WHH^T - XH^T.$$

Also, the complementary slackness conditions $\lambda_{ik} \cdot g_{ik}(W) = 0$ in matrix form reads

$$0 = \Lambda_W \circ W = \nabla_W f \circ W$$

where $\circ$ is the component-wise product of two matrices.

Similarly, for the variables $H = (H_{kj})$, the constraints are $-H_{kj} \leq 0$, and applying the same reasoning yields

$$\Lambda_H = \nabla_H f = -W^T(X - WH) = W^T WH - W^T X,$$

$$0 = \nabla_H f \circ H.$$

To finish, adding the feasibility conditions $\Lambda_W \geq 0$ and $\Lambda_H \geq 0$, the KKT conditions read

$$\begin{cases} W \geq 0, \quad \nabla_W f = WHH^T - XH^T \geq 0, \quad W \circ \nabla_W f = 0, \\ H \geq 0, \quad \nabla_H f = W^T WH - W^T X \geq 0, \quad H \circ \nabla_H f = 0. \end{cases} \tag{5}$$

These conditions characterize first-order optimality for the constrained problem (4), and are satisfied at any local minimum.

3.2. **Multiplicative updates.** Given X, W and H, the goal is to iteratively update W and H according to the rules in (6). Listing 1 shows the multiplicative update algorithm used in this work, implemented in Python.

The multiplicative updates (MU) modify W and H as follows:

$$W \leftarrow W \circ \frac{[XH^T]}{[WHH^T]} \qquad \text{and} \qquad H \leftarrow H \circ \frac{[W^T X]}{[W^T WH]}, \tag{6}$$

where $\frac{[\,]}{[\,]}$ denotes the componentwise division between two matrices.

It is straightforward to verify that if a point (W, H) satisfies the first-order optimality conditions (5), then the update rules (6) do not alter the values of W and H.

In Theorem 1 of [3], the authors prove the following:

Theorem 1. *The Euclidean distance* $\|X - WH\|_F$ *is nonincreasing under the update rules* (6)*.*

Hence, the MU lead to an algorithm for which f does not increase.

An intuitive interpretation of the update rules in (6) is as follow. Observe that from the gradient expression $\nabla_W f = WHH^T - XH^T$, we have that

$$\frac{(XH^T)_{ik}}{(WHH^T)_{ik}} \geq 1 \quad \Leftrightarrow \quad (\nabla_W f)_{ik} \leq 0.$$

Therefore, in order to look for matrices W and H that satisfy (5), for each entry of W, the multiplicative update behaves as follows: (i) increase the entry if its partial derivative is negative, (ii) decrease the entry if its partial derivative is positive, or (iii) leave the entry unchanged if its partial derivative is equal to zero. The same logic applies to the updates for H.

However, if an entry of W is zero, the MU cannot change it, regardless of the sign of the corresponding partial derivative. As a result, it is possible for an entry of W to be zero while its partial derivative is negative, which violates the optimality conditions in (5). Therefore, the MU are not guaranteed to converge to a point that satisfies the first-order optimality conditions.

This issue highlights a limitation of the MU algorithm. To address it, several alternative optimization algorithms have been proposed, which can ensure convergence to stationary points under appropriate conditions (see [2] for details).

Appendix A. Constrained optimization methods

In many practical optimization problems, the solution is required to satisfy certain constraints. This section introduces two fundamental approaches for handling constraints: the method of Lagrange multipliers for equality constraints, and the *Karush-Kuhn-Tucker (KKT)* conditions for inequality constraints.

A.1. **Lagrange multipliers.** Consider the following optimization problem with one equality constrain

$$\min_{x,y} f(x, y) \quad \text{subject to } g(x, y) = 0. \tag{7}$$

We assume that f and g have continuous first partial derivatives.

Suppose that the point (x_0, y_0) satisfies the constraint $g(x_0, y_0) = 0$ and that the gradient $\nabla g(x_0, y_0) \neq 0$. Recall that the gradient $\nabla g(x_0, y_0)$ is orthogonal to the level set defined by

$g(x,y)=0$. Therefore, if $f(x_0,y_0)$ is a minimum of the constrained problem (7), then the gradient $\nabla f(x_0,y_0)$ must be parallel to $\nabla g(x_0,y_0)$. Otherwise, one could move along the constraint set $g(x,y)=0$ in a direction that decreases f, contradicting the minimality of $f(x_0,y_0)$.

In summary, if $f(x_0,y_0)$ is a minimum of the constrained problem (7) and $\nabla g(x_0,y_0)\neq 0$, then there exists $\lambda_0\in\mathbb{R}$ such that

$$\nabla f(x_0,y_0)=\lambda_0\cdot\nabla g(x_0,y_0).$$

Defining the *Lagrange function* as

$$\mathcal{L}(x,y,\lambda)=f(x,y)+\lambda\cdot g(x,y).$$

Then, the gradient of $\mathcal{L}$ is given by

$$\nabla\mathcal{L}(x,y,\lambda)=\Big(\nabla f(x,y)+\lambda\cdot\nabla g(x,y),\quad g(x,y)\Big).$$

Thus, the condition $\nabla\mathcal{L}(x_0,y_0,\lambda_0)=0$ encodes the necessary conditions for (x_0,y_0) to be a solution of the constrained optimization problem (7), as discussed above.

To solve the original constrained optimization problem (7), we look for points (x,y,λ) such that $\nabla\mathcal{L}(x,y,\lambda)=0$, that is to say

$$\begin{cases}\nabla f(x,y)+\lambda\cdot\nabla g(x,y)=0\\ g(x,y)=0\end{cases}$$

In other words, we reduce the problem to solving a system of equations given by the vanishing of the gradient of the Lagrange function. Any solution (x_0,y_0,λ_0) of this system provides a candidate for a constrained extremum of f subject to $g(x,y)=0$.

A.2. **Karush-Kuhn-Tucker (KKT) conditions.** The method of Lagrange multipliers extends naturally to problems with multiple constraints—both equality and inequality. Suppose we aim to minimize $f(x,y)$ subject to $g_i(x,y)\leq 0$ and $h_j(x,y)=0$ for $i=1,\dots,M$ and $j=1,\dots,N$. The KKT conditions generalize the necessary optimality conditions and state that, under regularity assumptions, a candidate solution (x,y) must satisfy:

$$\begin{cases}\nabla f(x,y)+\sum_{i=1}^{M}\lambda_i\nabla g_i(x,y)+\sum_{j=1}^{N}\mu_j\nabla h_j(x,y)=0 & \text{(stationarity)},\\ g_i(x,y)\leq 0,\quad h_j(x,y)=0 & \text{(primal feasibility)},\\ \lambda_i\geq 0 & \text{(dual feasibility)},\\ \lambda_i\cdot g_i(x,y)=0 & \text{(complementary slackness)}.\end{cases}\tag{8}$$

These conditions form a system whose solutions are candidates for local optima of the constrained optimization problem. For further details, see [1].

References

[1] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.

[2] Gillis, N. (2021). Nonnegative matrix factorization. Society for Industrial and Applied Mathematics. https://lccn.loc.gov/2020042037

[3] Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. Advances in Neural Information Processing Systems, 13, 556–562.

Email address: `so.llavayol@gmail.com`