

Nonnegative Matrix Factorization On Image Compression

Nam Pham

BS19BDS025

SP Jain School of Global
Management

Overview

In the age of big data explosion, it is very essential to handle large amounts of data accurately. If we take non-negative values, then it is not possible to perform analysis with classical tools as they would not guarantee to remain non-negative. In this connection, a non-negative matrix factorization (NMF) can be applied, since it provides a low-rank approximation of the target matrix, while also extracting its significant values. This report presents theoretical details about NMF, the main numerical algorithm and implemented Python code used to solve the image compression problem, and an overview of the most recent experimental applications in the field of image processing.

TABLE OF CONTENTS

Table of Contents

Nonnegative Matrix Factorization 3

 1 Introduction 3

 2 Definition..... 5

Image Compression 7

Conclusion 12

Nonnegative Matrix Factorization

I Introduction

Over the past decades, newest improvements in high-technology have lead to an increasing amount of data, quickly overwhelming many existing classical analytical tools. Processing this enormous amount of data requires powerful methods of representing and reducing their size.

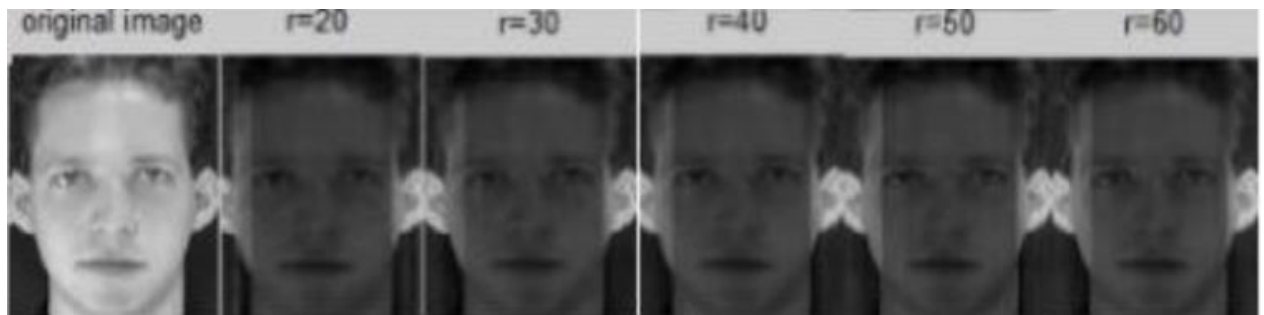
In a few practical applications, it is common to deal with non-negative data, for example the pixel intensity that generates the image, the amplitude spectrum, the number of occurrences of a given event, the scores taken by some users, etc. However, analysis of this data type cannot be performed using regular matrix factorization techniques - such as singular value decomposition (SVD) - and other popular tools for dimensional reduction and detection of latent structures - such as factor analysis, major component analysis, and cluster analysis - have been studied in numerical linear algebra, as they cannot ensure the maintenance of non-negativity. In fact, despite its various strengths (optimal properties, fast and powerful computation, etc.), SVD reveals nothing about the data collection because the factors do not provide any interpretability. Such a constraint apparently makes much of the previous work inapplicable to the present situation and paves the way for the development of **non-negative matrix factorization** (NMF).

Today, NMF has proven to be one of the most popular low-level approximations, used for compression, rendering, feature selection, and noise filtering. Its purpose is to replace the original data with a lower dimensional representation obtained through subspace approximation, in which constructs a set of fundamental elements such that the linear space that they span gets as close to the data points as possible. Those fundamental elements could later be used for cation identification and classification,

making NMF a really powerful unsupervised learning technique that enables automatic clustering of similar elements into groups and therefore a valuable alternative to common cluster algorithms.

The fields for which NMF has been successfully applied to dimensional data analysis are very high. Some demonstrations include image processing (for example, to detect key features of a face), text extraction (to restore the main contents of a set of documents), bioinformatics, signal processing, gas emission control, analysis and transcription of music, portfolio diversification and so on. Yet, many other industries are being looked forward to using NMF in no time, due to its innate ability to automatically extract sparse, useful, and easy-to-understand features from a set of non-negative data.

Within the limits of this study, we will be using NMF to compress an image for different r dimensions. An algorithm of NMF has to be chosen and implemented using a software such as Python. Additionally, two images with different features shall be selected from a website on internet for being compressed using the NMF implementation. An example is in the figure below that shows an original image and compressed images for distinct value r .



Also in the project we will estimate the distortion that the image suffers after the compression. This is measured using the Compression ratio, which is defined as:

$$C_r = \frac{\text{Uncompressed Image Size}}{\text{Compressed Image Size}}$$

II Definition

Non-negative matrix factorization (NMF) is a method that for a given matrix \mathbf{V} of n rows and m columns, NMF will find 2 (or 3) matrices \mathbf{H} , \mathbf{W} such that, the product $\mathbf{H} \times \mathbf{W}$ will return a value approximately equal to \mathbf{V} , or in another word, with a given matrix \mathbf{V} , find \mathbf{H} , \mathbf{W} such that:

$$\mathbf{V} \approx \mathbf{W}\mathbf{H} \quad \text{with } (\mathbf{W}, \mathbf{H}) \geq 0$$

where the columns in $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_\mu \dots \mathbf{v}_m]$ represent the data points, $\mathbf{W} = [\mathbf{w}_1 \dots \mathbf{w}_a \dots \mathbf{w}_r]$ represents the latent features, and those in $\mathbf{H} = [\mathbf{h}_1 \dots \mathbf{h}_\mu \dots \mathbf{h}_m]$ represents the coordinates of each data point in the factor matrix \mathbf{W} .

$$\begin{array}{c}
 \begin{array}{c} \text{variable 1} \\ \vdots \\ \text{variable } i \\ \vdots \\ \text{variable } n \end{array} \mathbf{V} = \begin{array}{c} \text{entity 1} \quad \dots \quad \text{entity } \mu \quad \dots \quad \text{entity } m \\ \left[\begin{array}{cccccc} v_{11} & \dots & v_{1\mu} & \dots & v_{1m} \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ v_{i1} & \dots & v_{i\mu} & \dots & v_{im} \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ v_{n1} & \dots & v_{n\mu} & \dots & v_{nm} \end{array} \right] \end{array} \approx
 \end{array}$$

$$\begin{array}{c}
 \text{Basis vectors} \\
 \left[\begin{array}{cccc} w_{11} & \dots & w_{1a} & \dots & w_{1r} \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ w_{i1} & \dots & w_{ia} & \dots & w_{ir} \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots \\ w_{n1} & \dots & w_{na} & \dots & w_{nr} \end{array} \right] \times
 \end{array}$$

$$\begin{array}{c}
 \text{Mixture coefficients} \\
 \left[\begin{array}{cccc} h_{11} & \dots & h_{1\mu} & \dots & h_{1m} \\ \vdots & & \vdots & & \vdots \\ h_{a1} & \dots & h_{a\mu} & \dots & h_{am} \\ \vdots & & \vdots & & \vdots \\ h_{r1} & \dots & h_{r\mu} & \dots & h_{rm} \end{array} \right] = \mathbf{W}\mathbf{H}
 \end{array}$$

Figure 2.1: Decomposition of the target matrix in nonnegative matrix factorization

How to find W and H ?

We need to optimize the following objective function $L(V, WH)$, qualifying the quality of the approximation, based on the square Euclidean distance between the two nonnegative matrices V and WH :

$$L(V, WH) = \|V - WH\|^2 = \sum_{i=1}^n \sum_{\mu=1}^m \left(v_{i\mu} - (WH)_{i\mu} \right)^2.$$

When this target function is optimized, meaning, progressing to 0, then we have $V \approx HW$.

To optimize the objective function L , we can use *gradient descent* or *multiplicative update rule*. The essence of both optimization methods is similar in that the algorithm will start by initializing a random non-negative value for W , H and then repeating the process of updating the value for H and W until the value of the objective function converges (convergence means that the value of the objective function changes little to none after each iteration).

The matrix W and H are respectively called the **base matrix** and the **mixture coefficients matrix**. In NMF technique, the matrix H is also proven to be a matrix representing V through the **base matrix** W , which means that through the NMF algorithm, we could find a matrix H , whose characteristics approximate the matrix V but for a much smaller dimension.

Image Compression

The first areas of NMF's application are the image processing and facial recognition. When Lee and Seung first created NMF pointed out that this revolutionary technique would provide a convenient way to learn parts of the face (such as the nose, eyes, facial structure and mouth), which can later be used correspondingly for pictures.

Let us now consider an application of NMF for image compression. Our goal is to apply NMF for obtaining an optimized compressed version of the original image while retaining its key features.

- ❖ During the image processing, we will be using Python to implement some codes.
- ❖ There are plenty of algorithms that can be used for this study, whereas **gradient descent** is probably the easiest technique to use, yet convergence is quite slow on the other hand. Other methods like **conjugate gradient** are faster when it comes to convergence but more complicated to proceed.
- ❖ Therefore, we can try applying **multiplicative update rule** for this image compression problem. It is a good balance between speed and convenience of implementation.

Theory

- ❖ The Euclidean distance $\|V - WH\|$ is non-increasing if it satisfies:

$$W_{i,j} \leftarrow W_{ij} \frac{(VH^T)_{ij}}{(WHH^T)_{ij}}$$
$$H_{i,j} \leftarrow H_{ij} \frac{(W^TV)_{ij}}{(W^TWH)_{ij}}$$

The Euclidean distance is only invariant when W and H are at a steady point of distance.

Images applied in this study

Two example images we will be using include the stunning view of the bustling Ho Chi Minh city in Vietnam and a romantic acrylic painting from some artist. Both are taken randomly from the Internet.



Source: https://webcdn.executivecentre.com/wp-content/uploads/2018/05/2018_5_2_Blog-image.jpg



Source: <https://wallpaperaccess.com/full/206378.jpg>

The implementation will be conducted on Jupiter Notebook, using Python programming language. First, we insert necessary libraries:

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io
from sklearn.decomposition import NMF
```

Next, the following function can be used to generate an image compressed according to NMF:

```
def NMFImage(i2, n_components):
    w, h, c = i2.shape
    new_img = i2.copy()
    for i in range(c):
        nmf = NMF(n_components=n_components)
        P = nmf.fit_transform(i2[:, :, i])
        Q = nmf.components_
        new_img[:, :, i] = np.clip(P @ Q, 0, 1)
    return {'new_image': new_img}
```




Running the above function, we can see the effect of image compression under different values of r:

```
plt.figure(figsize=(12, 9))
plt.imshow(i2); plt.axis('off')
plt.show()
for i in [1,3,5,10,30]:
    print('Value of r:', i)
    out = NMFImage(i2, i)
    new_image = out['new_image']
    plt.figure(figsize=(12, 9))
    plt.imshow(new_image); plt.axis('off')
    plt.show()
```


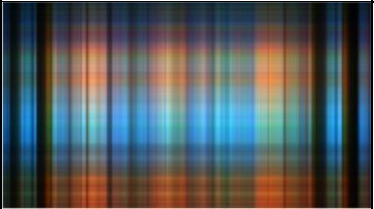

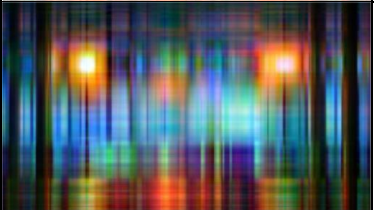
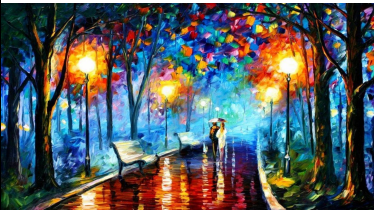
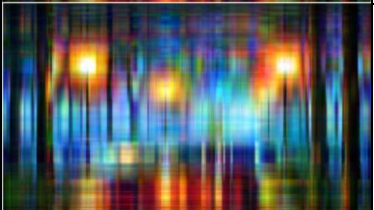
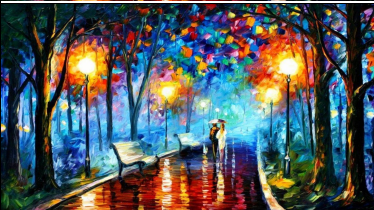

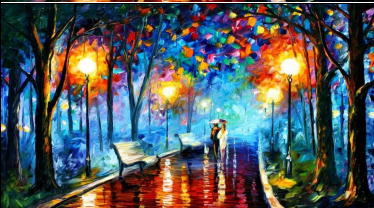

Numerical Experiment

The city image

Original size: 366 KB

Original Image	Value of r	Compressed Image	Compression Ratio
	1		$C_r = \frac{366 \text{ KB}}{172 \text{ KB}} = 2.13$
	3		$C_r = \frac{366 \text{ KB}}{232 \text{ KB}} = 1.58$
	5		$C_r = \frac{366 \text{ KB}}{270 \text{ KB}} = 1.36$
	10		$C_r = \frac{366 \text{ KB}}{300 \text{ KB}} = 1.22$
	30		$C_r = \frac{366 \text{ KB}}{340 \text{ KB}} = 1.08$

The painting image
Original size: 585 KB

Original Image	Value of r	Compressed Image	Compression Ratio
	1		$C_r = \frac{585 \text{ KB}}{248 \text{ KB}} = 2.36$
	3		$C_r = \frac{585 \text{ KB}}{319 \text{ KB}} = 1.83$
	5		$C_r = \frac{585 \text{ KB}}{352 \text{ KB}} = 1.66$
	10		$C_r = \frac{585 \text{ KB}}{392 \text{ KB}} = 1.49$
	30		$C_r = \frac{585 \text{ KB}}{453 \text{ KB}} = 1.29$

Conclusion

The image compression problem presented in this report has shown one of the important applications of NMF in everyday life, from facial recognition, recommendation systems to damaged image restoration. With a smaller compression ratio, the compressed image gets much clearer and closer to the original as the compression ratio reaches to 1, allowing us to evaluate the impact of the rating selection on the restoration of the original image.

To summarize, in the age of Big Data, there is an essential need to accurately process large amounts of data by identifying characteristics of interest in order to make informed decisions. In many disciplines (e.g. image recognition and text extraction), non-negative data are frequently processed, which cannot be implemented through classical tools, as they are not ensured to remain non-negativity.

Hence, a non-negative matrix factorization was developed. This method enables one to analyze non-negative data excellently by providing a low-rank approximation of a potentially very large target matrix while extracting its significant characteristics.

This powerful approach has made great strides and will shave a significant impact on new progress in the not too distant future.