NON-NEGATIVE MATRIX FACTORIZATION AND APPLICATIONS

PIPER MORRIS, BRANDON BONIFAZ-REYES & RAD MALLARI[1]

May 27, 2022

[1] *Department of Mathematics, University of California, Santa Barbara*

Contents

List of Figures

Abstract

Non-negative matrix factorization, often referred to as NMF, is one of many unsupervised learning algorithms used to extract significant figures from large sets of data. It is a relatively new process that has become an important tool in dimensionality reduction, and is comparable to other factorization techniques such as **principal component analysis** (PCA) and **vector quantification** (VQ). NMF is unique due to its constraint on non-negative elements resulting in an often preferable outcome. NMF has garnered popularity in machine-learning industries because the algorithm is able to efficiently extract sparse data and provide respective factors that one can easily interpret.

1 Introduction

Given an $n \times m$ non-negative matrix A, NMF aims to express A as two similarly non-negative matrices of smaller dimensions, W and H. Note that when referring to a non-negative matrix, we are discussing a matrix in which all its entries are real and greater than or equal to zero. W is an $m \times k$ matrix consisting of k *basis elements* of A and H is a $k \times n$ matrix consisting of the coefficients, or *weights*, related to the entries of W. We denote some $k > 0$ as the inner dimension of W and H and the smallest such instance where k holds is called the *factorization rank*[1]. Because our resulting NMF is a nonnegative approximation, W, H (holding the same properties) may not be unique either. Nonetheless, we utilize methods for measuring the error of our approximations.

2 Cost Functions

As mentioned above, the reason NMF is so complex is because its algorithms attempt to replicate the original matrix, namely A. Thus, the product of WH will be a representation of A, and not an exact replica. As a result, we implement various methods to go about solving an NMF problem. One of the most common ways to execute this is by using a cost function, of which there are several. Cost functions share a common goal of attempting to minimize the error between our original matrix and the product of our factorization. One useful method is accomplished by minimizing the square of the Euclidean distance, which is often referred to as minimizing the Frobenius norm as it relates specifically to matrices. See the Frobenius norm below[1]:

$$\|A - WH\|_F^2 = \sum_{ij} (A - WH)_{ij}$$

This measure is often chosen because it is not as expensive as other cost functions in terms of calculation. Additionally, it has a key property in that it remains invariant under rotations/orthogonal transformations. Secondly, it accounts for the presence of Gaussian noise and is recognized as the Euclidean norm because it compiles all the rows/columns of a matrix and concatenates them to produce one vector. While applying the frobenius norm, iterative update rules are used to exchange a randomized WH matrix with one that has a smaller error. We continue to do this until the error meets a predetermined requirement. An alternative norm used in the process of decomposition is the Kullback Liebler divergence[2]:

$$D_{KL}(A|WH) := \sum_{i=1}^{m} \sum_{j=1}^{n} ((WH)_{ij} - A_{ij} \log(WH)_{ij}) + \sum_{i=1}^{m} \sum_{j=1}^{n} (A_{ij} \log A_{ij} - A_{ij})$$

Where $D_{KL}(A|WH)$ is a scalar cost function and the new goal is to minimize the scalar cost function, i.e.

$$\min_{W,H} D_{KL}(A|WH)$$

3 Multiplicative Update Rule

A common approach to NMF problems is Lee and Seung's multiplicative update rule. The multiplicative update rule became popular due to its simplicity, however, it is criticized for its slow convergence and lack of guarantee of convergence to a stationary point, which is necessary to find a local minimum. The formula is given by[1]:

$$W \leftarrow W \cdot \frac{(VH^T)}{(WHH^T)}$$

$$H \leftarrow H \cdot \frac{(W^T V)}{W^T WH}$$

This solves the minimization problem from the previous section. It is a result of a proof that is beyond the scope of this paper, but application is surprisingly simple as discussed in the following section.

4 Applications

We created an application of NMF which decomposes an image matrix, then returns W, H, and the resulting WH which then is plotted. Since WH are not exact, we see that the image return is only approximately the original image 2 on the following page. The script[2] is written using Python, and deploys the multiplicative update rule to iterate until the error between the original matrix A and W, H are less than a specified error or a specified number of iterations. From here the following will convey the implications of the results.

4.1 Facial Recognition

Assuming we have a collection of the same face, we can flatten each image to a single vector. This would be our training set, denoted by $X_1, X_2, ..., X_n$ respectively. The collection of these flattened images could be written as matrix X, i.e.

$$X = \begin{bmatrix} \vdots & \vdots & & \vdots \\ X_1 & X_2 & \dots & X_N \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

By applying NMF to this matrix, we have $X \approx WH$, we can identify prominent features on the face with the W matrix and their weights with H. By taking an image matrix outside of the set, then applying NMF to the new image matrix, say $X' \approx W'H'$, we can compare and find the best fit to the collection of original WH pairs. This allows us to predict approximately what the new image is most similar to in the collection of WH pairs.

4.2 Image Processing

NMF often begins its canonical relationship with the real world in the field of image processing. We take a simple image of a face with p pixels, and reduce the dimensions of the data to a single vector such that the i-th entry represents the i-th pixel. Then, we can allow the rows of our matrix A to represent the p pixels and n columns respectively to output one image. The process of NMF will provide W and H such that the columns of W embody the basis images while H gives instructions on how to add up the aforementioned images. This summation process allows us to reconstruct an approximation of the originally provided face[3]. In this example of image processing, we can understand basis images to be learned key features of the face. (See Figure 1 on the next page).

[2] https://github.com/radimallari/108C_Final_Project

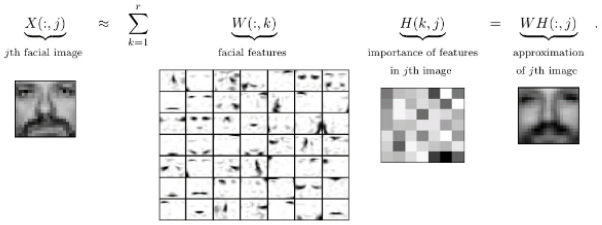

Figure 1: Face decomposition with k = 49. [3]

(a) Pre NMF image. **(b)** Post NMF k = 35.

Figure 2: Result of NMF

4.3 Text Mining

We utilize NMF in the field of topic recovery/document classification. In this context, one is tasked with extracting useful information from a dataset that is high-dimensional and unstructured Let each column of A correspond to a text and each row relate to a keyword. Construct every (i, j)-th entry in such a way where it represents every time a given word (i) comes up in a document (j). [3] This is a type of quality of term frequency construction in such a model, namely the bag-of-words model, where each document being analyzed has its respective set of words. The ordering of words here is not accounted for. The concept of sparseness appears once more in A due to the nature of texts/documents which only use a portion of the dictionary. In any case, NMF provides a decomposition of rank k as follows:

$$A(i,j) \approx \sum_{k=1}^{r} W(i,k) \cdot \sum H(k,j)$$

Note that in the above A is our j-th document, W our k-th topic, and H conveys the relevance of a topic in a given document. The intuition here is that our method of factorization applies *weight* to words with more or less necessity.

5 Conclusion

As a result of our research, we established NMF as it differs from other useful factorization methods in the field of mathematics and machine learning. NMF utilizes important cost functions such as the Frobenius norm in an attempt to minimize the error accrued in the various algorithms NMF implements. This allows us to effectively express a given matrix A as the product of two matrices, WH. The multiplicative update rule is one solution to the minimization problem that NMF presents. We discussed several ways of computing NMF, a crucial piece to learning algorithms in the industry. While computationally complex, NMF has shown to be relatively easy to implement. The application we created, while seemingly meaningless, can be used in powerful implementations such as facial recognition as outlined in section 4. We explored image processing and text mining, but some further subjects are voice and music recognition since they are expressed as positive matrices in a computer.

REFERENCES

[1] Seung Lee. Algorithms for non-negative matrix factorization.

[2] Zhirong Yang, He Zhang, Zhijian Yuan, and Erkki Oja. Kullback-leibler divergence for nonnegative matrix factorization. volume 6791, pages 250–257, 06 2011.

[3] Adrian Colyer. The why and how of nonnegative matrix factorization, Feb 2019.