

Non-negative matrix
factorization (NMF)

1. Introduction

NMF is a group of algorithms where a matrix V can be decomposed into two matrices W and H , each of which are easier to work with and when multiplied together, yield the original matrix.

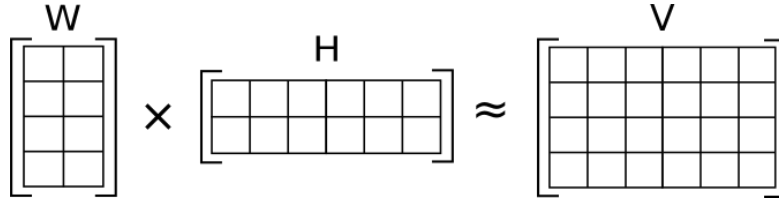


Fig. 1.1 V (4 X 6) is approximated as W (4 X 2) multiplied by H (2 X 6)

(Source: https://en.wikipedia.org/wiki/Non-negative_matrix_factorization)

Given, a matrix V of dimension $m \times n$ and $v_{ij} \geq 0$, NMF decomposes it into 2 matrices W and H of dimension $m \times r$ and $r \times n$ where

$$W_{ij} \geq 0$$

$$H_{ij} \geq 0$$

$$r < \min(m, n)$$

Thus, V is decomposed into a tall, skinny matrix W and a short, wide matrix H . The user can specify r as the inner dimension of W and H as long as $r < \min(m, n)$.

Each column of V , v_i can be calculated as:

$$v_i = W * h_i$$

Thus, each column of W is weighted by its corresponding row in h_i , which are then added together to form columns of V .

2. Purpose

Suppose V is a large dataset where each column is an observation and each row is a feature. For example, in a database of images, a column might represent some image and a row can represent a pixel. In machine learning, it is necessary to reduce the feature space for easy computation. In the above example, it is difficult to consider each pixel value every time an image is handled, so it is worthwhile to break it down into fewer components. Thus, NMF is used as a new way of reducing the dimensionality of data.

Since NMF has a non-negative constraint, it is used to represent data with positive features. This advantage can be used in image processing since each image has a positive pixel value.

NMF is similar to PCA where each base is assigned a weight. But in NMF, the weights are constrained to be positive.

3. Applications

3.1. Computer vision

NMF is beginning to be used in many fields. It is used in computer vision to reduce the feature space in images. This can be useful in identifying and classifying images.

3.2. Text mining

NMF is also used in text mining. For example, you might organize a series of documents into a matrix where each column may represent the frequency a particular word and a row might represent the document. Then you would extract semantic features about the data.

3.3. Speech denoising

NMF is used to break audio recordings of speech into speech parts and noise parts so that the speech parts alone can be isolated.

4. The problem

A fundamental model in NMF utilizes the least squares cost function to measure the closeness of matrices, resulting in the following standard NMF problem:

$$\text{Minimize } \|V - WH\|^2 \text{ subject to } W, H \geq 0 \quad (1)$$

where $\|\cdot\|$ is Frobenius norm, and the inequalities are component-wise [1].

The conventional approach to find W and H is by minimizing the difference between V and WH :

$$f(W, H) \equiv \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^m (v_{ij} - (WH)_{ij})^2 \quad (2)$$

$$\text{where } W_{ia} \geq 0, H_{bj} \geq 0, \quad \forall i, a, b, j$$

Moreover, the problem is considered non-convex which makes the problem difficult and hence is of type NP(hard). Non-convexity is a concept from optimization where a problem is considered to be non-convex if the objective or any of the constraints are non-convex or unbounded. Eg. A sine wave

5. Framework

Many algorithms have been proposed for this. But all of them use a concept from optimization theory called block coordinate descent. This framework is used mainly because of its nice convergence property. This is based on alternating updates between W and H until a tolerance is met. So, when it converges, it will converge at a local minimum and because the problem is non-convex, this is what most algorithms should achieve.

```

while (tolerance met) {
    fix H
    update W

    fix W
    update H
}

```

Fig. 5.1 The BCD framework

6. Algorithms

We need certain properties of the NMF problem (2). The gradient of the function $f(W, H)$ consists of 2 parts:

$$\nabla_W f(W, H) = (WH - V)H^T \quad (3)$$

$$\nabla_H f(W, H) = (WH - V)W^T \quad (4)$$

From the Karush-Kuhn-Tucker (KKT) optimality condition (Bertsekas, 1999), (W, H) is a stationary point of (2) if and only if

$$W_{ia} \geq 0$$

$$H_{bj} \geq 0$$

$$\nabla_W f(W, H)_{ia} \geq 0 \quad (5)$$

$$\nabla_H f(W, H)_{bj} \geq 0 \quad (6)$$

$$W_{ia} \cdot \nabla_W f(W, H)_{ia} = 0$$

$$H_{bj} \cdot \nabla_H f(W, H)_{bj} = 0 \quad \forall a, i, b, j$$

Thus, problem (2) is considered to be non-convex and may have several local minima. Most non-convex optimization methods guarantee only the stationarity of the limit points. Such a property is still useful, as any local minimum must be a stationary point.

6.1. Multiplicative update methods

The most widely used approach to solving (2) is a multiplicative update method proposed by Lee and Seung (2001).

Algorithm 1: Multiplicative Update

1. Initialize $W_{ia}^1 \geq 0, H_{bj}^1 \geq 0, \forall i, a, b, j$.

2. For $k = 1, 2, \dots$

$$W_{ia}^{k+1} = W_{ia}^k \frac{(V(H^k)^T)_{ia}}{(W^k H^k (H^k)^T)_{ia}}, \quad \forall i, a$$

$$H_{bj}^{k+1} = H_{bj}^k \frac{((W^{k+1})^T V)_{bj}}{((W^{k+1})^T W^{k+1} H^k)_{bj}}, \quad \forall b, j$$

Fig. 6.1 The multiplicative update algorithm

This is a fixed-point type method. If $(W^k H^k (H^k)^T)_{ia} \neq 0$ and $W_{ia}^{k+1} = W_{ia}^k > 0$, then

$$(V(H^k)^T)_{ia} = (W^k H^k (H^k)^T)_{ia} \rightarrow \nabla_W f(W^k, H^k)_{ia} = 0$$

which is part of the KKT condition (5) and (6). Lee and Seung (2001) have shown that the function value is non-increasing after every update [3]:

$$f(W^{k+1}, H^k) \leq f(W^k, H^k) \text{ and} \quad (7)$$

$$f(W^{k+1}, H^{k+1}) \leq f(W^{k+1}, H^k) \quad (8)$$

They claim that the limit point satisfies the KKT condition. However, this claim is wrong as having (7) and (8) does not mean convergence. Therefore, this method lacks optimization properties.

The overall cost of algorithm 1 is:

$$\# \text{ of trials} * O(nmr)$$

6.2. Alternating non-negative least squares (ANLS)

The method used for solving the above problem seems to be the alternating least squares (ALS) algorithm utilized by Paatero and Tapper in 1994 [2]. It minimizes the least squares cost function with respect to either W or H , one at a time, while fixing the other and disregarding non-negativity, and then sets any negative entries to zero after each least squares step.

From the non-increasing properties (7) and (8), the multiplicative update algorithm is a special case of a general framework, which alternatively fixes one matrix and improves the other:

Find W^{k+1} such that $f(W^{k+1}, H^k) \leq f(W^k, H^k)$ and
Find H^{k+1} such that $f(W^{k+1}, H^{k+1}) \leq f(W^{k+1}, H^k)$

Algorithm 2: Alternating non-negative least squares

1. Initialize $W_{ia}^1 \geq 0, H_{bj}^1 \geq 0, \forall i, a, b, j$.
2. For $k = 1, 2, \dots$

$$W^{k+1} = \arg \min_{W \geq 0} f(W, H^k) \quad (9)$$

$$H^{k+1} = \arg \min_{H \geq 0} f(W^{k+1}, H) \quad (10)$$

Fig. 6.2 The ANLS algorithm

This approach is the “block coordinate descent” method in bound-constrained optimization (Bertsekas, 1999) [5], where sequentially one block of variables is minimized under corresponding constraints and the remaining blocks are fixed. For NMF, we have the simplest case of only two block variables W and H .

Suppose (9) and (10) are considered sub-problems in algorithm 2. If one variable (eg. W), then the sub problem is a collection of non-negative least squares. Thus, from (10)

$$H^{k+1'} s^{j^{th}} column = \min_{H \geq 0} \|v - W^{k+1} h\|^2 \quad (11)$$

where v is the j^{th} column of V and h is a vector variable. But solving problems (9) and (10) for every iteration would be slower than the multiplicative update algorithm 1. Efficient methods to solve them are required. Therefore, we use the projected gradient method described next.

7. ANLS using projected gradient

Algorithm 2 requires solving sub-problems (9) and (10). Thus, use projected gradient methods. Sub-problem (10) consists of several non-negative least squares (11), so one can solve them separately but only in parallel environments. In serial environments, we solve them together.

To obtain H^{k+1} , rewrite (10) as:

$$\min_H \bar{f}(H) \equiv \frac{1}{2} \|V - WH\|^2$$

$$\text{subject to } H_{bj} \geq 0 \quad \forall b, j$$

where both V and H are constant matrices. Similarly, we can obtain W^{k+1} by rewriting (10) as:

$$\bar{f}(W) \equiv \frac{1}{2} \|V^T - W^T H^T\|^2$$

$$\text{subject to } W_{ia} \geq 0 \quad \forall a, i$$

where both V^T and H^T are constant matrices.

The total computation cost is:

$$\# \text{ of trials} * O(\text{tmnr})$$

8. Data

We consider an image problem:

CBCL face image database [6]

<http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html>

This data set consists of 472 faces, 23,573 non-faces as 19 x 19 images. The idea is to reduce the dimensionality of the above data so that it can be used for further analysis and computation.

9. MATLAB Code [7]

The MATLAB code used for solving the NMF problem consists of a function call described as:

```
function [W,H] = nmf(V, Winit, Hinit, tol, timelimit, maxiter)
```

where

W, H	=	output matrices;	V	=	input matrix
$Winit$	=	<code>abs(randn(size(V, 1), r));</code>	$Hinit$	=	<code>abs(randn(r, size(V, 2)));</code>
tol	=	tolerance for stopping condition;	$timelimit$	=	stopping condition
$maxiter$	=	maximum iterations; will break if tolerance or time limit reached			

The initial values for W and H plays an important role in the final result. Larger the value of r , the more accurate will be the result. But this defeats the main purpose of dimension reducing algorithms.

9.1. Stopping conditions

In the above MATLAB code, `tolerance` and `timelimit` are used as stopping conditions. An initial value of the gradient is calculated by computing forbenius norm between `gradW` and `gradH`. During iteration, another value of project gradient is computed as `norm([gradW(gradW<0 | W>0); gradH(gradH<0 | H>0)])`. If this value is less than multiplication of `tolerance` and inital gradient, it terminates.

`timelimit` is another stopping condition. An initial value of `cputime` is computed. Then during iteration, the current value of `cputime` is subtracted from the earlier value. If it is greater than `timelimit`, it terminates.

`maxiter` is the number of iterations, but again this depends on values of `tolerance` and `timelimit`.

10. Output




W (19 x 9)	H (9 x 19)	W*H (19 x 19)	V (19 x 19)
			

Fig 10.1 W , H , $W*H$ and the original image

10.1. Verification

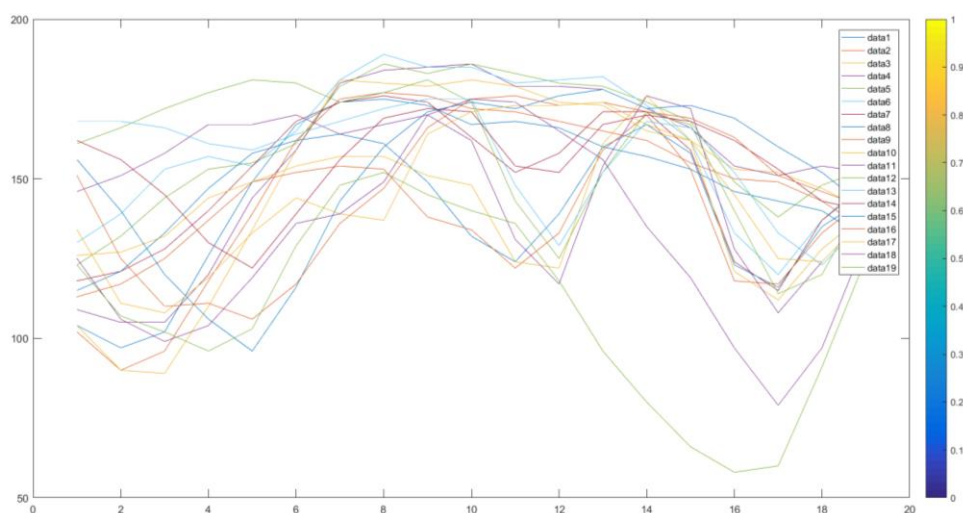


Fig 10.2 A plot of $W*H$

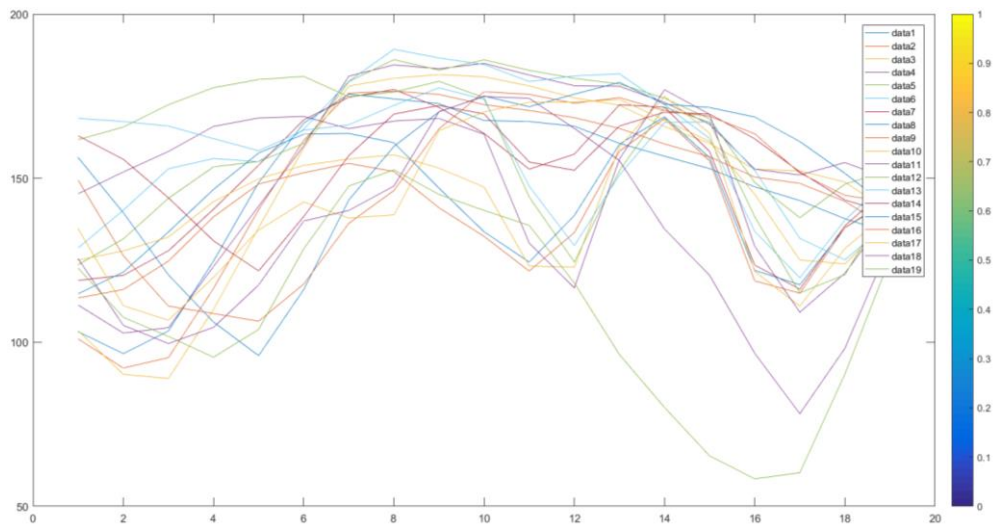


Fig 10.3 A plot of V

10.2. Varying r

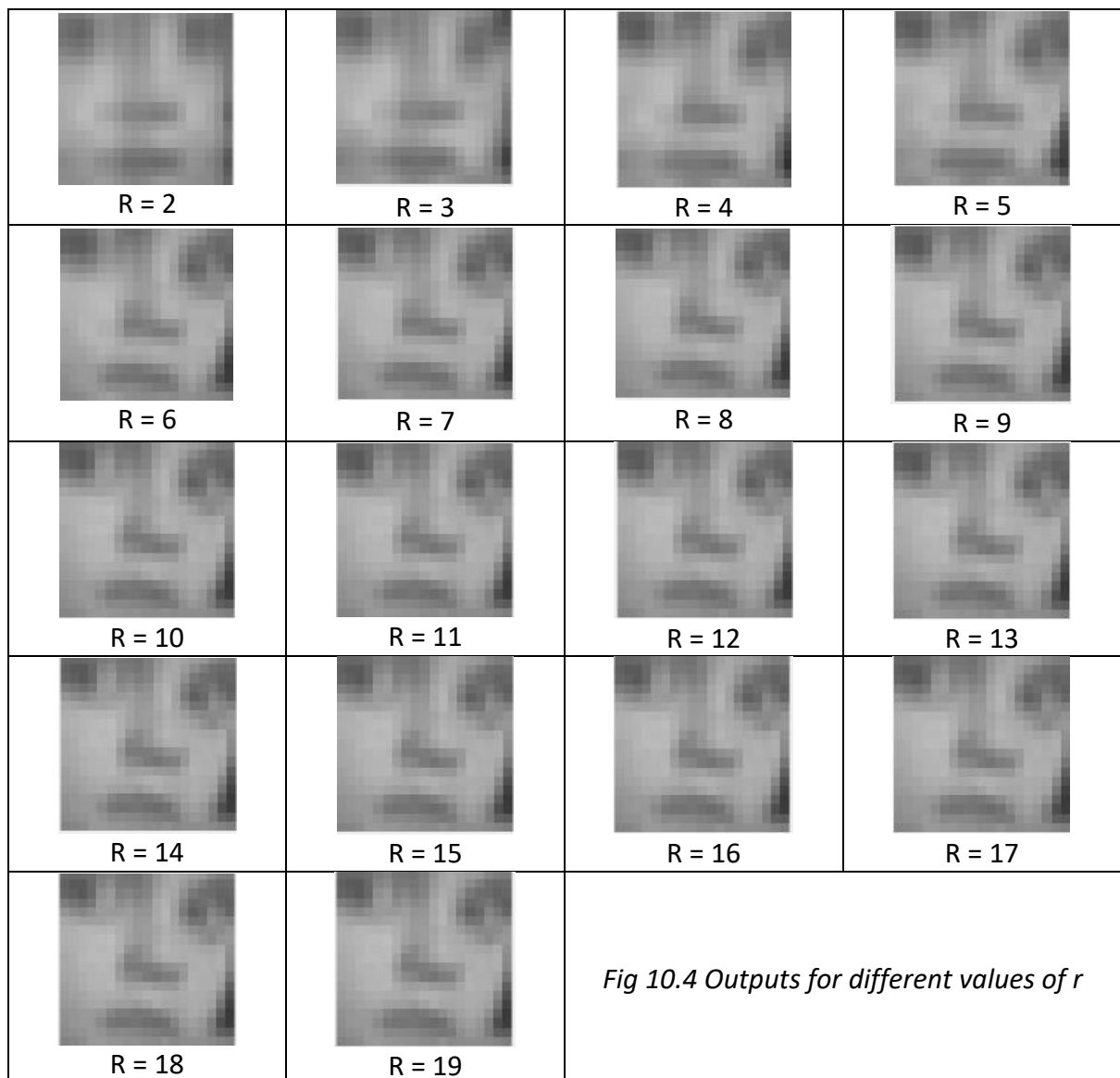


Fig 10.4 Outputs for different values of r

11. Conclusion

Non negative matrix factorization reduces the dimensionality of an object (V) by decomposing it into 2 matrices (W and H), each of which are easier to work with. When these decomposed matrices are multiplied together, it yields the original matrix.

The decomposed matrices (W or H) can be used as an input to a classifier algorithm viz. CNN. Furthermore, since these matrices are smaller than the original matrix, the classifier algorithm learns faster and more efficiently. Thus, the algorithm learns from a set of NMF basis images. In turn, this improves the learning rate and the accuracy whilst testing.

12. References

- [1] Yin Zhang, "An Alternating Direction Algorithm for Nonnegative Matrix Factorization", Rice University, January 2010
- [2] Paatero, P. and Tapper, U., "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values", Environmetrics, 1994
- [3] Lee, D. and Seung, H., "*Algorithms for Non-Negative Matrix Factorization*". Advances in Neural Information Processing Systems, 2001
- [4] Chih-Jen Lin, "Projected Gradient Methods for Non-Negative Matrix Factorization", Neural Computation, MIT Press, 2007
- [5] Dimitri P. Bertsekas. "*Nonlinear Programming*", Athena Scientific, 1999
- [6] CBCL Face Database #1, MIT Center for Biological and Computation Learning, <http://www.ai.mit.edu/projects/cbcl>
- [7] Chih-Jen Lin, National Taiwan University, <https://www.csie.ntu.edu.tw/~cjlin/nmf/>