

Nonnegative matrix factorization (NMF)

DSA5103 Lecture 7

Yangjing Zhang

02-Mar-2023

NUS

Today's content¹

1. Dimension reduction
2. Applications of NMF
3. Variants of NMF
4. Optimization algorithms for NMF

¹Most of the content is from [3]

Dimension reduction

- Dimension reduction aims to represent each data point as a linear combination of a small number of basis vectors
- Given n data points $v_1, v_2, \dots, v_n \in \mathbb{R}^m$, dimension reduction looks for basis vectors

$$w_1, w_2, \dots, w_r \in \mathbb{R}^m$$

such that each data point is well-approximated by a linear combination of the basis vectors

$$\begin{aligned} v_j &\approx \sum_{i=1}^r w_i H_{ij} \\ &= w_1 H_{1j} + w_2 H_{2j} + \dots + w_r H_{rj}, \quad j \in [n] \end{aligned} \tag{1}$$

where H_{ij} are scalars.

Illustration $m = 2, r = 1, n = 10$

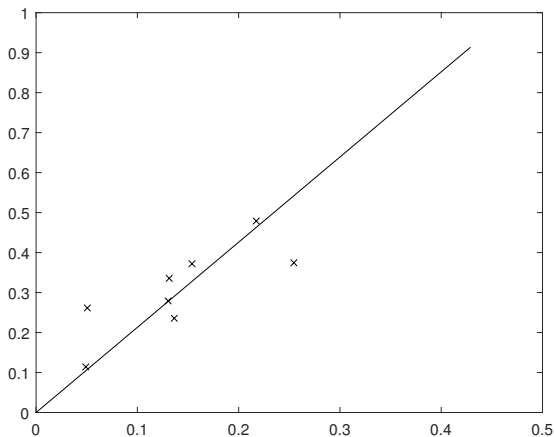


Figure 1: Approximate 2-dimensional data points with a 1-dimensional line generated by w_1

Illustration $m = 3, r = 2, n = 50$

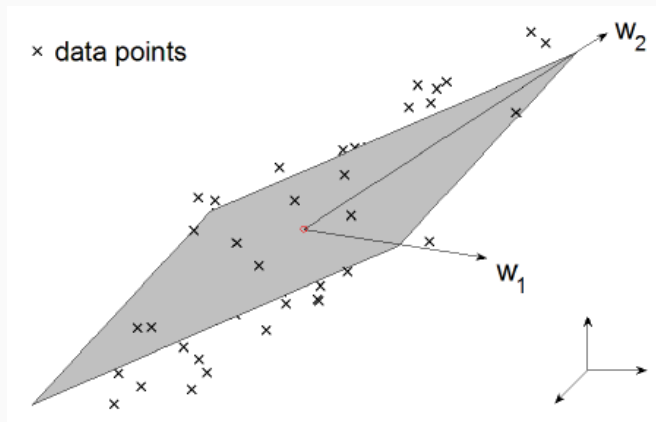


Figure 2: Approximate 3-dimensional data points with a 2-dimensional plane generated by w_1 and w_2 . Image from [3, Fig 1.1]

Dimension reduction

Dimension reduction (1)

$$v_j \approx w_1 H_{1j} + w_2 H_{2j} + \cdots + w_r H_{rj}, \quad j \in [n]$$

is equivalent to

$$\underbrace{[v_1, v_2, \dots, v_n]}_{V \in \mathbb{R}^{m \times n}} \approx \underbrace{[w_1, w_2, \dots, w_r]}_{W \in \mathbb{R}^{m \times r}} \underbrace{\begin{bmatrix} H_{11} & \cdots & H_{1n} \\ \vdots & \ddots & \vdots \\ H_{r1} & \cdots & H_{rn} \end{bmatrix}}_{H \in \mathbb{R}^{r \times n}}$$

- each column of $V \in \mathbb{R}^{m \times n}$ is a **data point**, $V_{\cdot j} = v_j$
- each column of $W \in \mathbb{R}^{m \times r}$ is a **basis vector**, $W_{\cdot j} = w_j$
- each column of $H \in \mathbb{R}^{r \times n}$ contains the **coordinates** of a data point in the basis W
- $\text{rank } r \ll \min(m, n)$

Dimension reduction

Variants of dimension reduction mainly differ in two key aspects:

1. **Error measure** can vary

- ▷ NMF takes $\|V - WH\|^2 = \|V - WH\|_F^2$
- ▷ One can instead take ℓ_1 norm $\|V - WH\|_1$

2. **Different constraints** can be imposed on W and H , which depends on the applications

- ▷ NMF takes

$$W \geq 0, H \geq 0$$

for nonnegative data analysis

- ▷ For example, orthogonal constraint

$$HH^T = I_r$$

(I_r denotes $r \times r$ identity matrix)

- ▷ For example, symmetric constraint

$$H = W^T$$

in this case, $V \approx WW^T$, $m = n$

Given a nonnegative matrix $V \in \mathbb{R}_+^{m \times n}$ and a rank r , NMF computes two nonnegative matrices $W \in \mathbb{R}_+^{m \times r}$ and $H \in \mathbb{R}_+^{r \times n}$ via

$$\begin{aligned} \min_{W, H} \quad & \frac{1}{2} \|V - WH\|_F^2 \\ \text{s.t.} \quad & W \geq 0, H \geq 0 \end{aligned} \quad (\text{NMF})$$

- The objective function is non-convex w.r.t. (W, H) , but bi-convex (convex in W with H fixed and convex in H with W fixed).
- NMF is one of the dimension reduction techniques
- NMF has different interpretations in applications: $V_{.j} \approx \sum_{i=1}^r W_{.i} H_{ij}$

	columns of V (data)	columns of W (basis)
CBCL face data	images of faces	images of facial features
Two digits data	images of two digits	images of basis digits
One digit data	images of a digit	cluster centroids
Text mining	words in documents	words for different topics


Applications

Grayscale images

- The value of each pixel is the intensity
- Option 1: 0 represents black; 1 represents white (Matlab takes this option)
- Option 2: Sometimes the scale is reversed. 0 represents white; 1 represents black
- Two-dimensional images are transformed into a long one-dimensional vector by stacking the columns

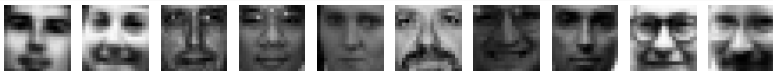


vec


$$= \text{vec} \begin{pmatrix} 0 & 1 & 0.6 & 0.3 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0.6 & 0.3 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 1 \\ 0.6 \\ \vdots \\ 0.6 \\ 0.3 \\ \vdots \\ 0.3 \end{pmatrix}$$

CBCL facial data²

- It contains 2429 grayscale facial images of the **same resolution**
 $19 \times 19 (= 361)$



- The images are **well-aligned**, e.g., pixels corresponding to noses should be located at the same position
- Construct data matrix V such that each column of V corresponds to a vectorized image
- $V \in \mathbb{R}^{361 \times 2429}$, $m = 361$, $n = 2429$. V_{ij} is the intensity of a pixel and nonnegative
- We manually choose $r = 20$ (one may try different values of r)

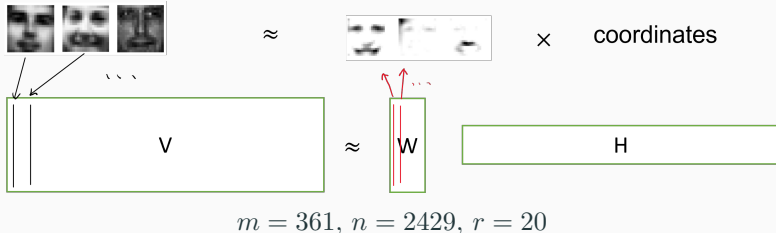
²See reference [4] for more details

CBCL facial data

- We solve the NMF problem by Matlab built-in function “nnmf”

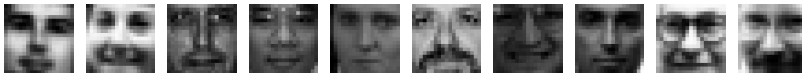
```
[W,H] = nnmf(V,r);
```

- Columns of W are **basis images containing facial features** such as eyes, noses, lips ...
- WH is a **low rank approximation** of V
- Dimension reduction: original data $361 \times 2429 \approx 900,000$, reduced data $361 \times 20 + 20 \times 2429 \approx 60,000$



CBCL facial data

(2429 images) $V =$



(20 basis images) $W =$



(2429 approx. images) $WH =$



Listing 1: Matlab codes applying NMF to CBCL and Swimmer data sets

```
clear; close all; rng(1);
option = 1; % {1,2}
if option == 1
    load DataCBCL; % load CBCL dataset
    pr = 19; pc = 19; % pixel of row and column
    r = 20; % set the rank
elseif option == 2
    load DataSwimmer;
    pr = 20; pc = 11; r = 10;
end
[m,n] = size(V);
%% show 10 original images
figure; tiledlayout(1,10);
rand_plot = randperm(n,10); % select 10 images randomly
for i = rand_plot
    nexttile;
    imshow(1-reshape(V(:,i),pr,pc)/max(V(:,i)),[0 1]);
end
%% NMF
[W,H] = nnmf(V,r);
W = max(W,0); H = max(H,0); % make sure nonnegative
```

```

%% show basis images
figure; tiledlayout(2,10);
for i = 1:min(20,r)
    nexttile;
    imshow(1-reshape(W(:,i),pr,pc)/max(W(:,i)),[0 1]);
end

%% show 10 approximate images
figure; tiledlayout(1,10);
for i = rand_plot
    nexttile;
    imshow(1-reshape(W*H(:,i),pr,pc)/max(W*H(:,i)),[0 1]);
end

```

- “imshow” displays the grayscale image
- We normalize a vectorized image such that the entries $\in [0, 1]$
- We show the value “1 – pixel value”. In Matlab, 0 = black, but in this application, we display 0 = white

Two digits data³

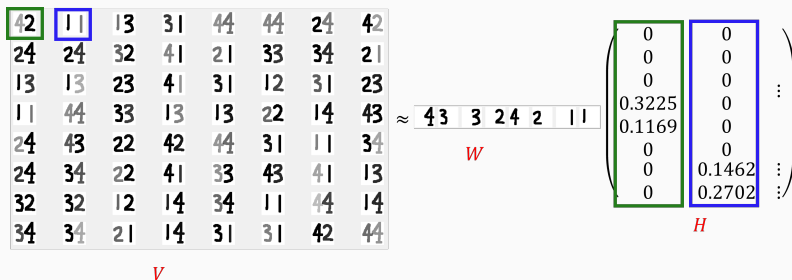
- It contains 64 images, and each image contains two digits.
Resolution $64 \times 64 (= 4096)$
- $V \in \mathbb{R}^{4096 \times 64}$, $m = 4096$, $n = 64$
- We manually choose $r = 8$ ($\{1, 2, 3, 4\} \times$ two digits)

42	11	13	31	44	44	24	42
24	24	32	41	21	33	34	21
13	13	23	41	31	12	31	23
11	44	33	13	13	22	14	43
24	43	22	42	44	31	11	34
24	34	22	41	33	43	41	13
32	32	12	14	34	11	44	14
34	34	21	14	31	31	42	44

³https://github.com/FLYYZJ/machine_learning_algorithms/tree/master/Matrix_Factorization

Two digits data

- Columns of W are basis images of left/right aligned numbers 1, 2, 3, 4
- In the first image V_{11} , intensity of digit "4" < intensity of digit "2". Correspondingly, $H_{51} = 0.1169 < H_{41} = 0.3225$



$$m = 4096, n = 64, r = 8$$

Two digits data

- If choosing $r = 7$, WH from NMF fails to give a good approximation of input V .

(64 images) $V =$

42	11	13	31	44	44	24	42
24	24	32	41	21	33	34	21
13	13	23	41	31	12	31	23
11	44	33	13	13	22	14	43
24	43	22	42	44	31	11	34
24	34	22	41	33	43	41	13
32	32	12	14	34	11	44	14
34	34	21	14	31	31	42	44

(64 approx. images) $WH =$

42	11	13	31	44	44	24	42
24	24	32	41	21	33	34	21
13	13	23	41	31	12	31	23
11	44	33	13	13	22	14	43
24	43	22	42	44	31	11	34
24	34	22	41	33	43	41	13
32	32	12	14	34	11	44	14
34	34	21	14	31	31	42	44

($r = 7$ basis images) $W =$ 

```
clear; close all; rng(1);
load Data2digits; % load dataset
pr = 64; pc = 64; r = 7; [m,n] = size(V);
%% show original images
figure; tiledlayout(8,8);
for i = 1:n
    nexttile;
    imshow(1-reshape(V(:,i),pr,pc)/max(V(:,i)),[0 1]);
end
%% NMF
[W,H] = nnmf(V,r); W = max(W,0); H = max(H,0);
%% show basis images
figure; tiledlayout(1,r);
for i = 1:r
    nexttile;
    imshow(1-reshape(W(:,i),pr,pc)/max(W(:,i)),[0 1]);
end
%% show approximate images
figure; tiledlayout(8,8);
for i = 1:n
    nexttile;
    imshow(1-reshape(W*H(:,i),pr,pc)/max(W*H(:,i)),[0 1]);
end
```

One digit data

- It is constructed from two digits data by splitting a image into two
- It contains 128 images, and each image contains a digit. Resolution $64 \times 32 (= 2048)$
- $V \in \mathbb{R}^{2048 \times 128}$, $m = 2048$, $n = 128$.
- We manually choose $r = 4$ ($\{1, 2, 3, 4\}$)

4	1	1	3	4	4	2	4	2	2	3	4	2	3	3	2
1	1	2	4	3	1	3	2	1	4	3	1	1	2	1	4
2	4	2	4	4	3	1	3	2	3	2	4	3	4	4	1
3	3	1	1	3	1	4	1	3	3	2	1	3	3	4	4
2	1	3	1	4	4	4	2	4	4	2	1	1	3	4	1
3	3	3	1	1	2	1	3	1	4	3	3	3	2	4	3
4	3	2	2	4	1	1	4	4	4	2	1	3	3	1	3
2	2	2	4	4	1	4	4	4	4	1	4	1	1	2	4

One digit data

- For this data, NMF has an inherent clustering property. It automatically clusters the data points
- Columns of W give the cluster centroids
- Columns of H give the cluster membership: j -th data point $\in k$ -th cluster if $H_{kj} > H_{ij}, \forall i \neq k$

(128 images) $V =$

4	1	1	3	4	4	2	4	2	2	3	4	2	3	3	2
1	1	2	4	3	1	3	2	1	4	3	1	1	2	1	4
2	4	2	4	4	3	1	3	2	3	2	4	3	4	4	1
3	3	1	1	3	1	4	1	3	3	2	1	3	3	4	4
2	1	3	1	4	4	4	2	4	4	2	1	1	3	4	1
3	3	3	1	1	2	1	3	1	4	3	3	3	2	4	3
4	3	2	2	4	1	1	4	4	4	2	1	3	3	1	3
2	2	2	4	4	1	4	4	4	4	1	4	1	1	2	4

(128 approx. images) $WH =$

4	1	1	3	4	4	2	4	2	3	4	2	3	3	2
1	1	2	4	3	1	3	2	1	4	3	1	1	2	1
2	4	2	4	4	3	1	3	2	3	2	4	3	4	1
3	3	1	1	3	1	4	1	3	3	2	1	3	3	4
2	1	3	1	4	4	4	2	4	4	2	1	1	3	4
3	3	3	1	1	2	1	3	1	4	3	3	3	2	4
4	3	2	2	4	1	1	4	4	4	2	1	3	3	1
2	2	2	4	4	1	4	4	4	4	1	4	1	1	2

($r = 4$ basis images) $W =$ 

Cluster membership

- From $H_{.1}$, we can see the first image \in 2nd cluster (“digit 2”)
- From $H_{.2}$, we can see the second image \in 4th cluster (“digit 1”)

(7 images) $[V_1, \dots, V_7] =$ 

($r = 4$ basis images) $W =$ 

(7 columns) $[H_{.1}, \dots, H_{.7}] =$

0.0290	0	0	0.1562	0.0368	0.0407	0
0.0369	0	0	0	0.0460	0.0510	0.0500
0.0028	0	0	0.0066	0.0041	0.0044	0.2445
0.0037	0.2298	0.1282	0.0033	0.0056	0.0060	0

Synthetic swimmer data set⁴

- It contains 256 images of a “swimmer”. Resolution $20 \times 11 (= 220)$



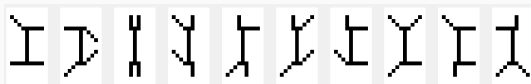
- Four limbs take four different positions
- $V \in \mathbb{R}^{220 \times 256}$, $m = 220$, $n = 256$
- We manually choose $r = 17$ (four different positions \times four limbs + body)

⁴[2]

Synthetic swimmer data set

- NMF by Matlab built-in function “nnmf”⁵ fails to give a good decomposition.

(256 images) $V =$



(17 basis images) $W =$



(256 approx. images) $WH =$



⁵under default setting

Term-document matrix

- Term document matrix is a method for representing documents
- Documents are the columns and terms are the rows
- The (i, j) entry is the number of times word i appears in document j
- The order of the words can be arbitrary

For example if one has two short documents

Doc 1 "I like optimization"

Doc 2 "Do you like optimization"

then the term-document matrix would be

	Doc 1	Doc 2
i	1	0
you	0	1
like	1	1
optimization	1	1
do	0	1

Text mining

- V is a term-document matrix, a column of V corresponds to a document
- Columns of W correspond to different topics
- Columns of H indicate the importance of the topics discussed in the corresponding documents
- Applied for topic recovery and document classification

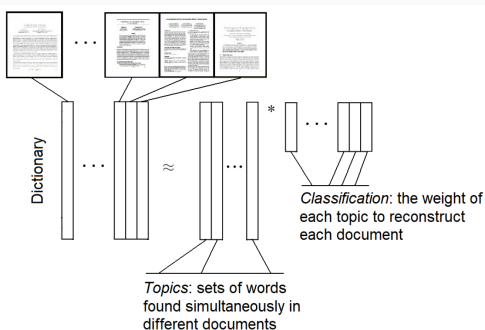


Image from [3, Fig 1.7]

Exact NMF

Definition (Exact NMF)

Given $V \in \mathbb{R}_{+}^{m \times n}$. If there exist $W \in \mathbb{R}_{+}^{m \times r}$ and $H \in \mathbb{R}_{+}^{r \times n}$ such that

$$V = WH$$

then we say WH is an **exact NMF** of V of rank r .

- Exact NMF may not be unique; Exact NMF may not exist for some r
- The **nonnegative rank**, denoted as $\text{rank}_{+}(V)$, refers to the smallest r such that V admits an exact NMF, i.e.,

$$V = WH, W \in \mathbb{R}_{+}^{m \times r}, H \in \mathbb{R}_{+}^{r \times n}$$

- Recall the rank of V is the smallest r such that V can be factorized as

$$V = WH, W \in \mathbb{R}^{m \times r}, H \in \mathbb{R}^{r \times n}$$

- Obviously, $\text{rank}(V) \leq \text{rank}_{+}(V) \leq \min(m, n)$ for any $V \in \mathbb{R}_{+}^{m \times n}$
(The second inequality follows from the trivial factorizations
 $V = VI_n = I_m V$)

Results on rank_+

We give two results from [6] without proof.

Result 1

If $V \geq 0$, $\text{rank}(V) \leq 2$, then $\text{rank}(V) = \text{rank}_+(V)$.

- $\text{rank}(V) = 1$ trivial. $V = [t_1 a, t_2 a, \dots, t_n a]$, $t_i \geq 0$, $a \in \mathbb{R}_+^m$
- $\text{rank}(V) = 2$ non-trivial. Proof omitted

Result 2

$$\text{Let } V = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \text{rank}_+(V) = 4, \text{rank}(V) = 3.$$

One result from [7] implies that exact NMF are NP-hard. We instead consider approximate NMF in practice.

Result 3

It is NP-hard to check whether $\text{rank}_+(V) = \text{rank}(V)$ for $V \geq 0$.

Variants of NMF

Variants of NMF


- Standard NMF model can be modified by the choice of error measure, or the inclusion of constraints or regularizers.
- Variants of NMF be more suitable for specific scenarios or applications
- We simply present the following 3 variants and explain in which situations they are meaningful


	Objective function	Constraints
NMF	$\frac{1}{2} \ V - WH\ _F^2$	$W \geq 0, H \geq 0$
Orthogonal NMF	$\frac{1}{2} \ V - WH\ _F^2$	$W \geq 0, H \geq 0, HH^T = I_r$
Symmetric NMF	$\frac{1}{2} \ V - WH\ _F^2$	$W \geq 0, H \geq 0, H = W^T$
Sparse NMF	$\frac{1}{2} \ V - WH\ _F^2$ $+ \lambda_W \ W\ _1 + \lambda_H \ H\ _1$	$W \geq 0, H \geq 0$

Orthogonal NMF

$$\begin{aligned} \min_{W, H} \quad & \frac{1}{2} \|V - WH\|_F^2 \\ \text{s.t.} \quad & W \geq 0, H \geq 0 \\ & HH^T = I_r \end{aligned}$$

Lemma: $H \geq 0$ and $HH^T = I_r$ imply that each column of H has at most a single positive entry.



$H_i.$  $a \geq 0, b \geq 0, a^T b = 0$

$H_j.$  implies that the two vectors has disjoint supports.

Orthogonal NMF

$$\begin{aligned} \min_{W, H} \quad & \frac{1}{2} \|V - WH\|_F^2 \\ \text{s.t.} \quad & W \geq 0, H \geq 0 \\ & HH^T = I_r \end{aligned}$$

Lemma: $H \geq 0$ and $HH^T = I_r$ imply that each column of H has at most a single positive entry.

$H_i.$  $a \geq 0, b \geq 0, a^T b = 0$
implies that the two vectors
 $H_j.$  has disjoint supports.

Proof: For $i \neq j$, the (i, j) entry of HH^T is $H_i.H_j^T = 0$. Every two rows of H have disjoint supports. Namely, each column of H can have at most a single positive entry.

Orthogonal NMF

- Orthogonal NMF is a clustering problem. Each column has at most a single positive entry indicating the membership
- Columns of W are cluster centroids
- Columns of H give the cluster membership: j -th data point $\in k$ -th cluster if $H_{kj} > 0$

Penalized version of orthogonal NMF

$$\begin{aligned} \min_{W, H} \quad & \frac{1}{2} \|V - WH\|_F^2 + \lambda \|HH^T - I_r\|_F^2 \\ \text{s.t.} \quad & W \geq 0, H \geq 0 \end{aligned}$$

where $\lambda > 0$ is the penalty parameter. The penalized version is relatively easy to optimize compared with the constrained version ($HH^T = I_r$)

$$\begin{aligned} \min_{W,H} \quad & \frac{1}{2} \|V - WH\|_F^2 + \lambda_W \|W\|_1 + \lambda_H \|H\|_1 \\ \text{s.t.} \quad & W \geq 0, H \geq 0 \end{aligned}$$

where $\lambda_W, \lambda_H > 0$ are the penalty parameters.

- For synthetic swimmer data set, recall that NMF by Matlab built-in function “nnmf” fails to give a good decomposition.
- We apply sparse NMF to swimmer data set (call the Matlab function “sparseNMF” [3]⁶)

⁶[https:](https://gitlab.com/ngillis/nmfbook/-/tree/master/algorithms/sparse%20NMF)

[//gitlab.com/ngillis/nmfbook/-/tree/master/algorithms/sparse%20NMF](https://gitlab.com/ngillis/nmfbook/-/tree/master/algorithms/sparse%20NMF)

Apply sparse NMF to swimmer data set

- Sparse NMF extracts the basis features well and the low rank factorization WH almost exactly recovers V

(256 images) $V =$



(17 basis images) $W =$



(256 approx. images) $WH =$



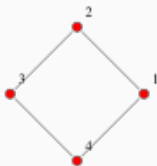
Adjacency matrix

of an undirected graph: $V_{ij} = 0$ no edge connecting node i and node j ;
 $V_{ij} = 1$ an edge connecting node i and node j .

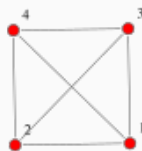
Instead of 1, the weight can be any scalars indicating the strength of the connection.



$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$



$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$



$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Membership matrix

The membership matrix of the data with label

Data points	label
A	1
B	0
C	1
D	0

is 4×2 (data points are rows, clusters are columns)

	class 0	class 1
A	0	1
B	1	0
C	0	1
D	1	0

Membership matrix

The membership matrix of the data with label

Data points	label
A	1
B	0
C	1
D	0

is 4×2 (data points are rows, clusters are columns)

	class 0	class 1
A	0	1
B	1	0
C	0	1
D	1	0

On the other hand, given an approximate membership matrix

	class 0	class 1
A	0.05	0.9
B	0.55	0.35
C	0.4	0.9
D	0.8	0

we label the data points as follows

Data points	label
A	1
B	0
C	1
D	0

Symmetric NMF

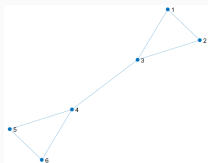
- Symmetric NMF requires $H = W^T$ ($m = n$), that is, $V \approx WW^T$.
- Symmetric NMF makes sense only when V is symmetric since WW^T is symmetric.

$$\begin{aligned} \min_W \quad & \frac{1}{2} \|V - WW^T\|_F^2 \\ \text{s.t.} \quad & W \geq 0 \end{aligned}$$

- $V \in \mathbb{R}_+^{n \times n}$ is nonnegative and symmetric. It can be thought of as **the adjacency matrix** of an undirected graph where V_{ij} indicated the similarity/correlation between nodes i and j
- Symmetric NMF is applicable for **community detection**: finding subsets of nodes that are densely connected. A subset of densely connected nodes is referred to as a community
- r = number of communities. W is an approximate membership matrix. Node $j \in k$ -th community if $W_{jk} > W_{ji}, \forall i \neq k$

Toy example

- Adjacency matrix $V \in \mathbb{R}_+^{6 \times 6}$, $V_{ij} \in \{0, 1\}$
- Choose $r = 2$ and apply symmetric NMF (call the Matlab function “symNMF” [3]⁷)

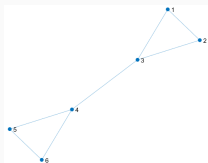


⁷[https://](https://gitlab.com/ngillis/nmfbook/-/tree/master/algorithms/symmetric%20NMF)

gitlab.com/ngillis/nmfbook/-/tree/master/algorithms/symmetric%20NMF

Toy example

- Adjacency matrix $V \in \mathbb{R}_+^{6 \times 6}$, $V_{ij} \in \{0, 1\}$
- Choose $r = 2$ and apply symmetric NMF (call the Matlab function “symNMF” [3]⁷)


$$W =$$

0.8005	0
0.8005	0
0.8472	0.2425
0.2425	0.8472
0	0.8005
0	0.8005

- Nodes 1, 2 \in community 1; Nodes 5, 6 \in community 2
- Nodes 3, 4 belong to the two communities with different intensities
- We say node 3 belongs mostly to community 1 since $W_{31} = 0.8472 > W_{32} = 0.2425$
- We say node 4 belongs mostly to community 2 since $W_{41} = 0.2425 < W_{42} = 0.8472$

⁷[https://](https://gitlab.com/ngillis/nmfbook/-/tree/master/algorithms/symmetric%20NMF)

gitlab.com/ngillis/nmfbook/-/tree/master/algorithms/symmetric%20NMF

Algorithms for NMF

1. PG and APG
2. Multiplicative update (MU) algorithm
3. Alternating least squares (ALS) algorithm
4. Hierarchical alternating least squares (HALS)

$$\min_{W,H} \underbrace{\frac{1}{2} \|V - WH\|_F^2}_{:=f(W,H) \text{ smooth}} + \underbrace{\delta_{\mathbb{R}_+^{m \times r}}(W) + \delta_{\mathbb{R}_+^{r \times n}}(H)}_{\text{non-smooth}}$$

- Denote $\Pi_+(\cdot) = \Pi_{\mathbb{R}_+^{m \times r}}(\cdot)$ or $\Pi_{\mathbb{R}_+^{r \times n}}(\cdot)$ (omiting dimensions).

$$W^{(k+1)} = \Pi_+ \left(W^{(k)} - \alpha_k \nabla_W f(W^{(k)}, H^{(k)}) \right)$$

- PG iteration:

$$H^{(k+1)} = \Pi_+ \left(H^{(k)} - \alpha_k \nabla_H f(W^{(k)}, H^{(k)}) \right)$$

- Gradients (details omitted)

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

- Therefore, PG iteration (APG iteration is similar):

$$R^{(k)} = V - W^{(k)} H^{(k)}$$

$$W^{(k+1)} = \Pi_+ \left(W^{(k)} + \alpha_k R^{(k)} (H^{(k)})^T \right)$$

$$H^{(k+1)} = \Pi_+ \left(H^{(k)} + \alpha_k (W^{(k)})^T R^{(k)} \right)$$

Multiplicative update (MU) algorithm [5]

- MU algorithm for NMF can be called via

```
[W,H] = nnmf(V,r,'algorithm','mult');
```

- This algorithm can be regarded as a variant of PG. The entries of the matrices W and H are all updated with **individually selected step sizes**.
- S_W (resp. S_H) matrix of step sizes for W (resp. H).
- Iteration (for brevity k omitted)

$$R \leftarrow V - WH$$

$$W \leftarrow \Pi_+ (W + S_W \odot RH^T)$$

$$H \leftarrow \Pi_+ (H + S_H \odot W^T R)$$

where \odot denotes element-wise multiplication.

MU algorithm

- The step sizes in [5] are taken as

$$S_W = W \oslash [W H H^T], \quad S_H = H \oslash [W^T W H]$$

where \oslash denotes element-wise division.

- With this choice of step sizes, the iteration

$$R \leftarrow V - W H$$

$$W \leftarrow \Pi_+ (W + S_W \odot R H^T)$$

$$H \leftarrow \Pi_+ (H + S_H \odot W^T R)$$

becomes the MU iteration

$$W \leftarrow [W \odot V H^T \oslash [W H H^T]]$$

$$H \leftarrow [H \odot W^T V \oslash [W^T W H]]$$

Alternating least squares (ALS) algorithm

- ALS algorithm for NMF can be called via

```
[W,H] = nnmf(V,r); % same as nnmf(V,r,'algorithm','als')
```

- Idea: minimize H (then W) without nonnegative constraints and project the solution onto the nonnegative orthant
- Iteration (for brevity k omitted)

$$H \leftarrow \arg \min_H \|V - WH\|_F^2$$

$$H \leftarrow \Pi_+(H)$$

$$W \leftarrow \arg \min_W \|V - WH\|_F^2$$

$$W \leftarrow \Pi_+(W)$$

- Generally, people think MU is better than ALS.


```
rng(1);  
V = rand(100); r = 50;  
[W,H] = nnmf(V,r,'algorithm','mult');  
norm(W*H - V,'fro')  
ans = 15.9377  
[W,H] = nnmf(V,r,'algorithm','als');  
norm(W*H - V,'fro')  
ans = 37.1920
```

- The error of MU is 15.9, that of ALS is 37.2
- One should be careful and try different options when calling a method

MU vs ALS

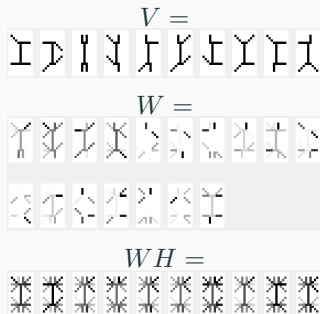


Figure 3: Swimmer data. ALS algorithm for NMF $r = 17$.

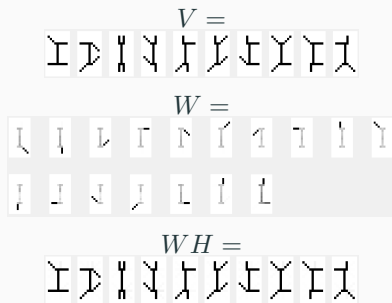


Figure 4: Swimmer data. MU algorithm for NMF $r = 17$.

- MU is much better for decomposing the synthetic swimmer data.

Hierarchical alternating least squares (HALS) [1]

In the iterate, it is called HALS, which is essentially BCD updating the rows of H and columns of W cyclically (see last lecture)

For $i = 1, \dots, r$

$$W_{\cdot i} \leftarrow \Pi_+ \left(\frac{(V - W_{\cdot(-i)} H_{(-i)\cdot}) H_{i\cdot}^T}{\|H_{i\cdot}\|^2} \right)$$

For $i = 1, \dots, r$

$$H_{i\cdot} \leftarrow \Pi_+ \left(\frac{W_{\cdot i}^T (V - W_{\cdot(-i)} H_{(-i)\cdot})}{\|W_{\cdot i}\|^2} \right)$$

- The choice of r is also known as model selection. Most data models face this issue of model selection
- We mentioned some practical strategies:
 1. Prior information. In some applications, one may have prior information on the value of r . For example, in two digits data, a reasonable r should be 8
 2. Cross validation. A widely used strategy for model selection in various data models. It selects r leading to the best performance for a particular task (e.g., clustering)

Compare with PCA

```
[W,H] = nnmf(V,r); % NMF  
  
[Q,Xnew] = pca(V,'NumComponents',r); % PCA  
W = Xnew;  
H = Q';
```



Figure 5: Swimmer data. MU algorithm for NMF $r = 17$.



Figure 6: Swimmer data. PCA with $r = 17$ components.



A. Cichocki, R. Zdunek, and S.-i. Amari.

Hierarchical ALS algorithms for nonnegative matrix and 3d tensor factorization.

In Lecture Notes in Computer Science, pages 169–176, 2007.



D. Donoho and V. Stodden.

When does non-negative matrix factorization give a correct decomposition into parts?

Advances in neural information processing systems, 16, 2003.



N. Gillis.

Nonnegative matrix factorization.

SIAM, 2020.



D. D. Lee and H. S. Seung.

Learning the parts of objects by non-negative matrix factorization.

Nature, 401(6755):788–791, 1999.



D. Seung and L. Lee.

Algorithms for non-negative matrix factorization.

Advances in neural information processing systems, 13:556–562, 2001.



L. Thomas.

Rank factorization of nonnegative matrices (a. berman).

SIAM Review, 16(3):393, 1974.



S. A. Vavasis.

On the complexity of nonnegative matrix factorization.

SIAM Journal on Optimization, 20(3):1364–1377, 2010.

