

Statistical Pattern Recognition – Spring 2021

Dr. Fardin Akhlaghian



دانشگاه کردستان  
University of Kurdistan  
زانکۆی کوردستان

# Nonnegative Matrix Factorization

TA: A. Seyedi



# Background

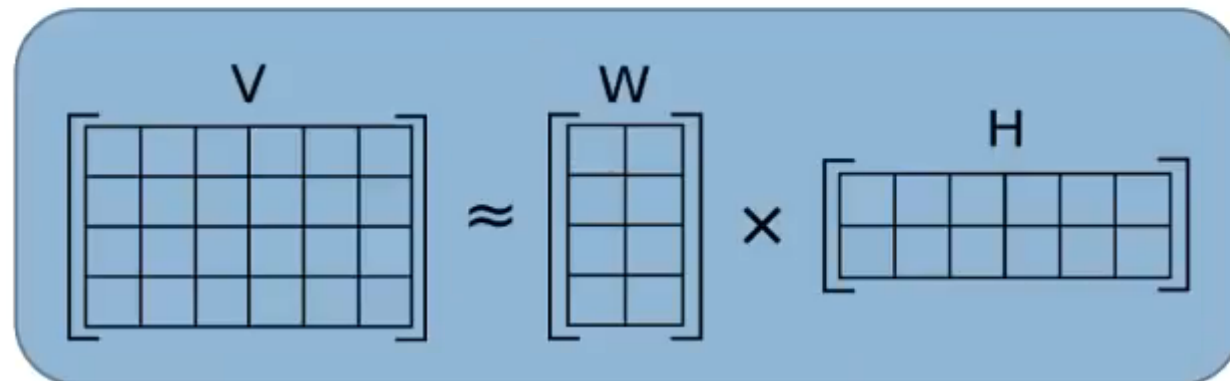
- Non-negative matrix factorization (NMF) is an unsupervised machine learning technique created by Lee & Seung in 1999.
- It is a versatile algorithm
  - Makes a parts-based-representation of its input data
  - Non-negativity of input data allows this
- Uses:
  - Dimensionality reduction
  - Data compression and approximation
  - Audio source separation
  - Text topic extraction

# Definition

- For a matrix  $V$  of dimension  $m \times n$  where each element  $v_{ij} \geq 0$ , NMF decomposes it into two matrices  $W$  and  $H$  of dimensions  $m \times r$  and  $r \times n$ , respectively, where each element  $w_{ij} \geq 0$  and  $h_{ij} \geq 0$  and  $r < \min(m, n)$  such that:

$$V \approx WH$$

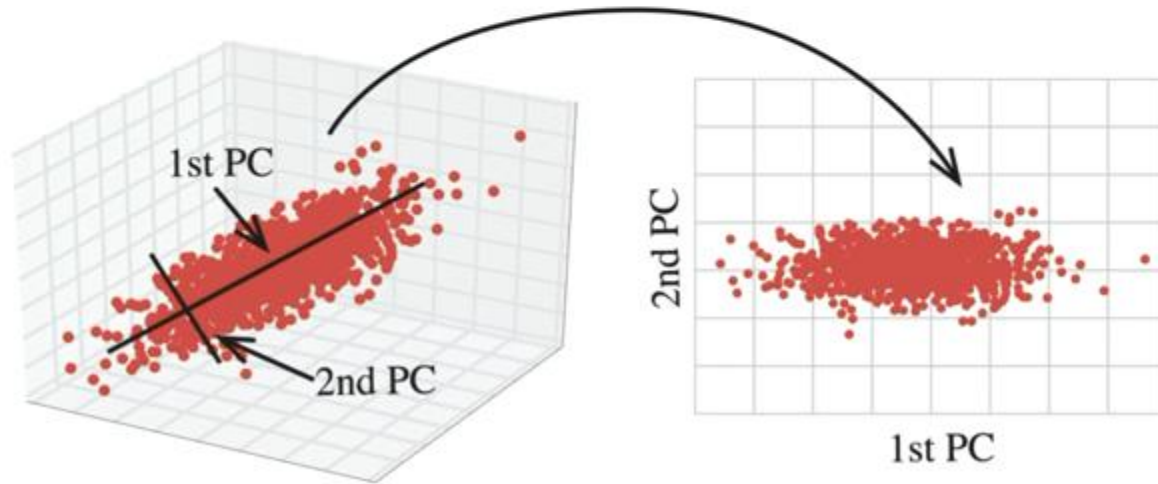
- This problem, however, cannot be solved analytically so it is generally approximated numerically:



# NMF Use Case: Dimensionality Reduction

Dimensionality Reduction: Task of transforming a dataset with many dimensions (or features) into a dataset with fewer dimensions, while losing the least amount of information possible.

PCA Example:

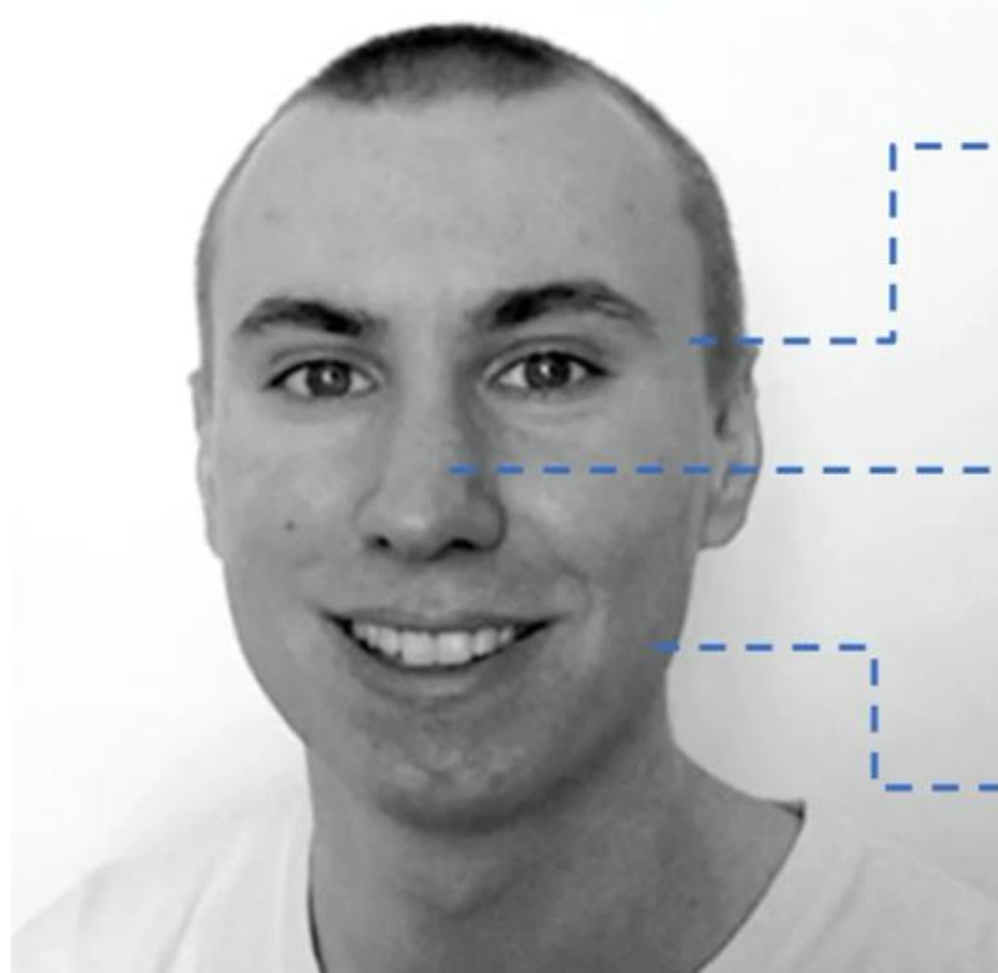


# Example: Face Recognition



# Example: Face Recognition

Identifying  
features



Eyes

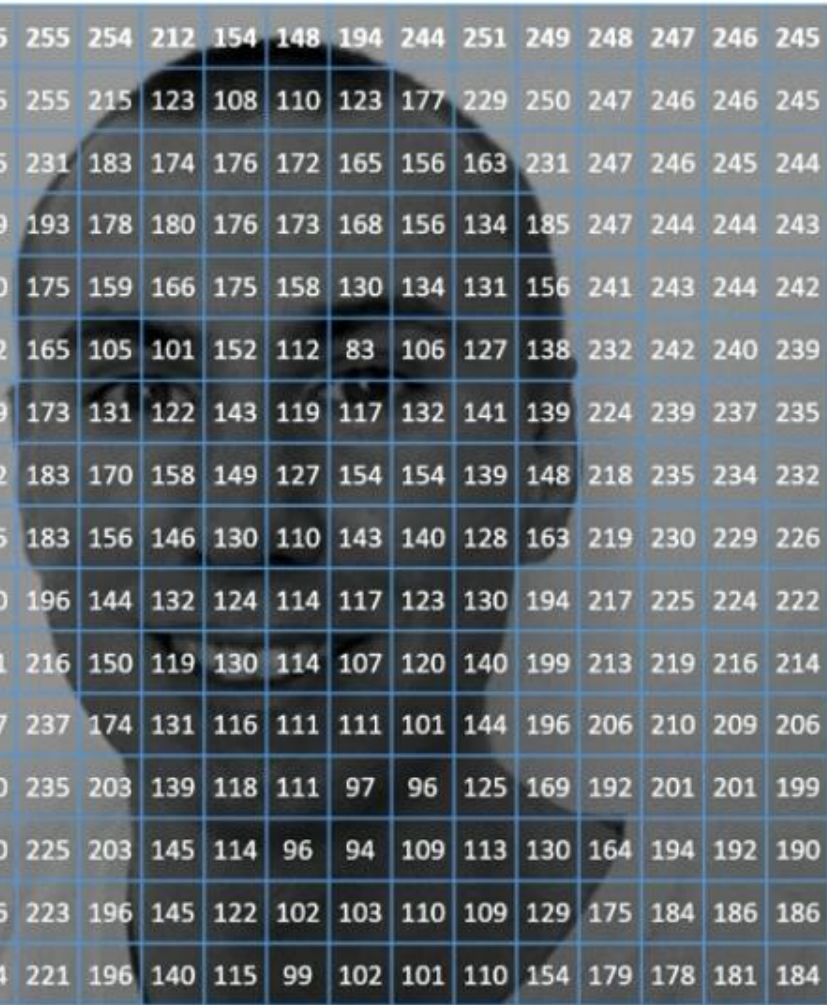


Nose



Mouth

# Example: Face Recognition



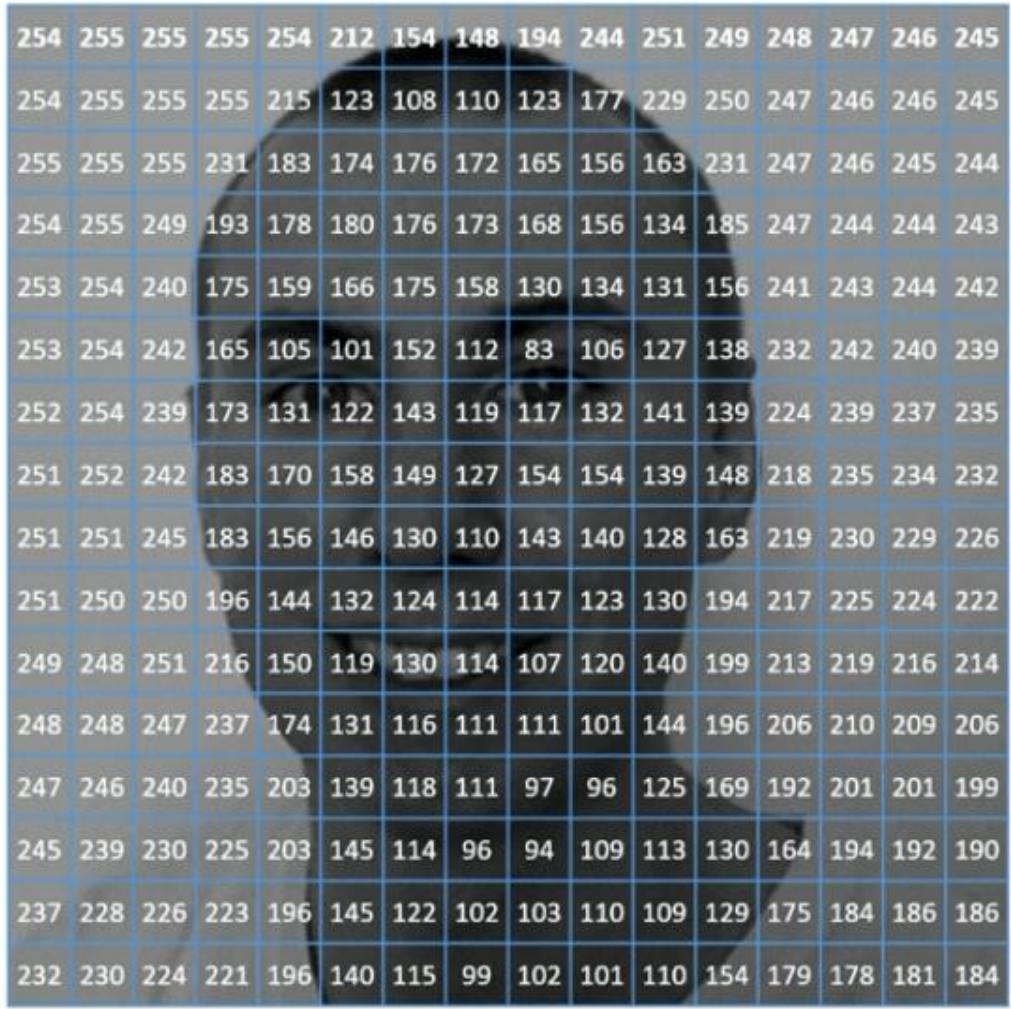
254	255	255	255	254	212	154	148	194	244	251	249	248	247	246	245
254	255	255	255	215	123	108	110	123	177	229	250	247	246	246	245
255	255	255	231	183	174	176	172	165	156	163	231	247	246	245	244
254	255	249	193	178	180	176	173	168	156	134	185	247	244	244	243
253	254	240	175	159	166	175	158	130	134	131	156	241	243	244	242
253	254	242	165	105	101	152	112	83	106	127	138	232	242	240	239
252	254	239	173	131	122	143	119	117	132	141	139	224	239	237	235
251	252	242	183	170	158	149	127	154	154	139	148	218	235	234	232
251	251	245	183	156	146	130	110	143	140	128	163	219	230	229	226
251	250	250	196	144	132	124	114	117	123	130	194	217	225	224	222
249	248	251	216	150	119	130	114	107	120	140	199	213	219	216	214
248	248	247	237	174	131	116	111	111	101	144	196	206	210	209	206
247	246	240	235	203	139	118	111	97	96	125	169	192	201	201	199
245	239	230	225	203	145	114	96	94	109	113	130	164	194	192	190
237	228	226	223	196	145	122	102	103	110	109	129	175	184	186	186
232	230	224	221	196	140	115	99	102	101	110	154	179	178	181	184

**Images are stored as numbers**

**Every image is a matrix**

**Very large number of features**

# Example: Face Recognition



254	255	255	255	254	212	154	148	194	244	251	249	248	247	246	245
254	255	255	255	215	123	108	110	123	177	229	250	247	246	246	245
255	255	255	231	183	174	176	172	165	156	163	231	247	246	245	244
254	255	249	193	178	180	176	173	168	156	134	185	247	244	244	243
253	254	240	175	159	166	175	158	130	134	131	156	241	243	244	242
253	254	242	165	105	101	152	112	83	106	127	138	232	242	240	239
252	254	239	173	131	122	143	119	117	132	141	139	224	239	237	235
251	252	242	183	170	158	149	127	154	154	139	148	218	235	234	232
251	251	245	183	156	146	130	110	143	140	128	163	219	230	229	226
251	250	250	196	144	132	124	114	117	123	130	194	217	225	224	222
249	248	251	216	150	119	130	114	107	120	140	199	213	219	216	214
248	248	247	237	174	131	116	111	111	101	144	196	206	210	209	206
247	246	240	235	203	139	118	111	97	96	125	169	192	201	201	199
245	239	230	225	203	145	114	96	94	109	113	130	164	194	192	190
237	228	226	223	196	145	122	102	103	110	109	129	175	184	186	186
232	230	224	221	196	140	115	99	102	101	110	154	179	178	181	184

Images are stored as numbers

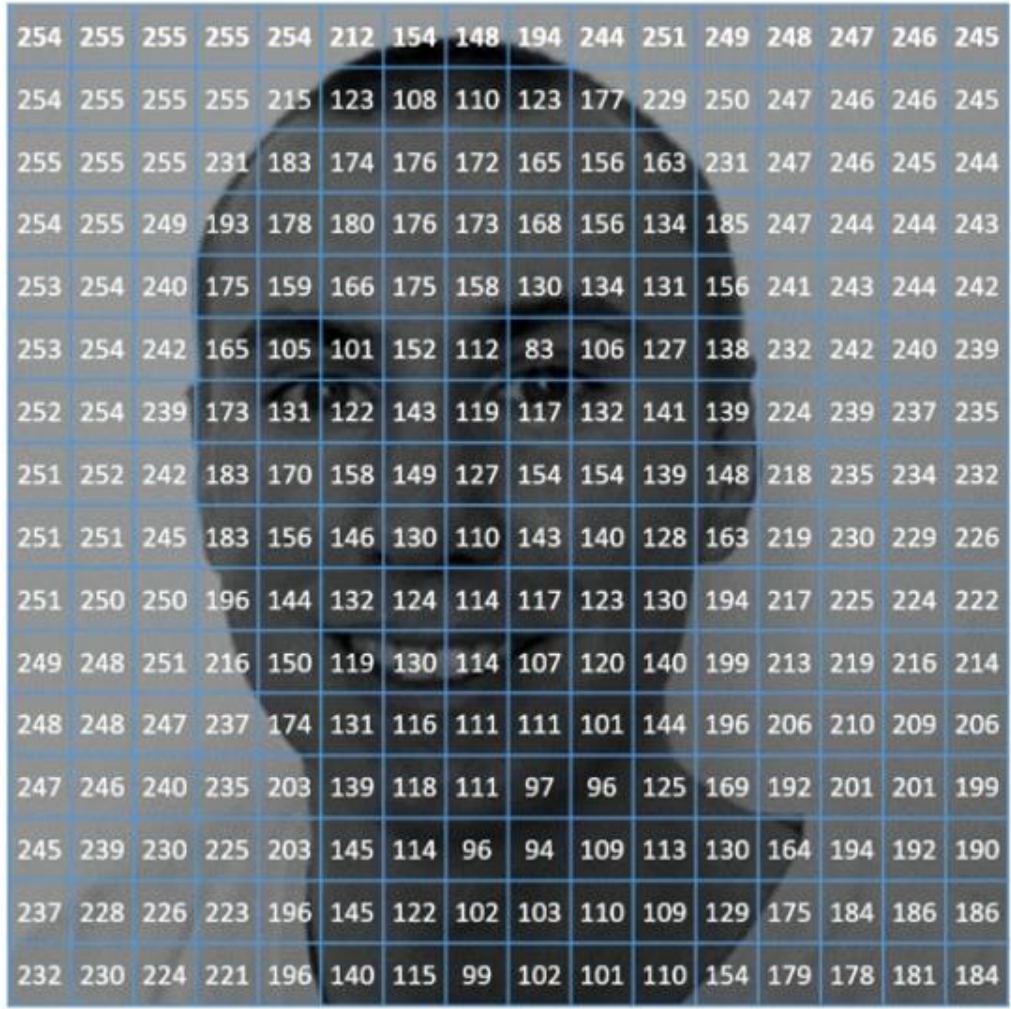
Every image is a matrix

Very large number of features

Need to **reduce the number of features**



# Example: Face Recognition



254	255	255	255	254	212	154	148	194	244	251	249	248	247	246	245
254	255	255	255	215	123	108	110	123	177	229	250	247	246	246	245
255	255	255	231	183	174	176	172	165	156	163	231	247	246	245	244
254	255	249	193	178	180	176	173	168	156	134	185	247	244	244	243
253	254	240	175	159	166	175	158	130	134	131	156	241	243	244	242
253	254	242	165	105	101	152	112	83	106	127	138	232	242	240	239
252	254	239	173	131	122	143	119	117	132	141	139	224	239	237	235
251	252	242	183	170	158	149	127	154	154	139	148	218	235	234	232
251	251	245	183	156	146	130	110	143	140	128	163	219	230	229	226
251	250	250	196	144	132	124	114	117	123	130	194	217	225	224	222
249	248	251	216	150	119	130	114	107	120	140	199	213	219	216	214
248	248	247	237	174	131	116	111	111	101	144	196	206	210	209	206
247	246	240	235	203	139	118	111	97	96	125	169	192	201	201	199
245	239	230	225	203	145	114	96	94	109	113	130	164	194	192	190
237	228	226	223	196	145	122	102	103	110	109	129	175	184	186	186
232	230	224	221	196	140	115	99	102	101	110	154	179	178	181	184

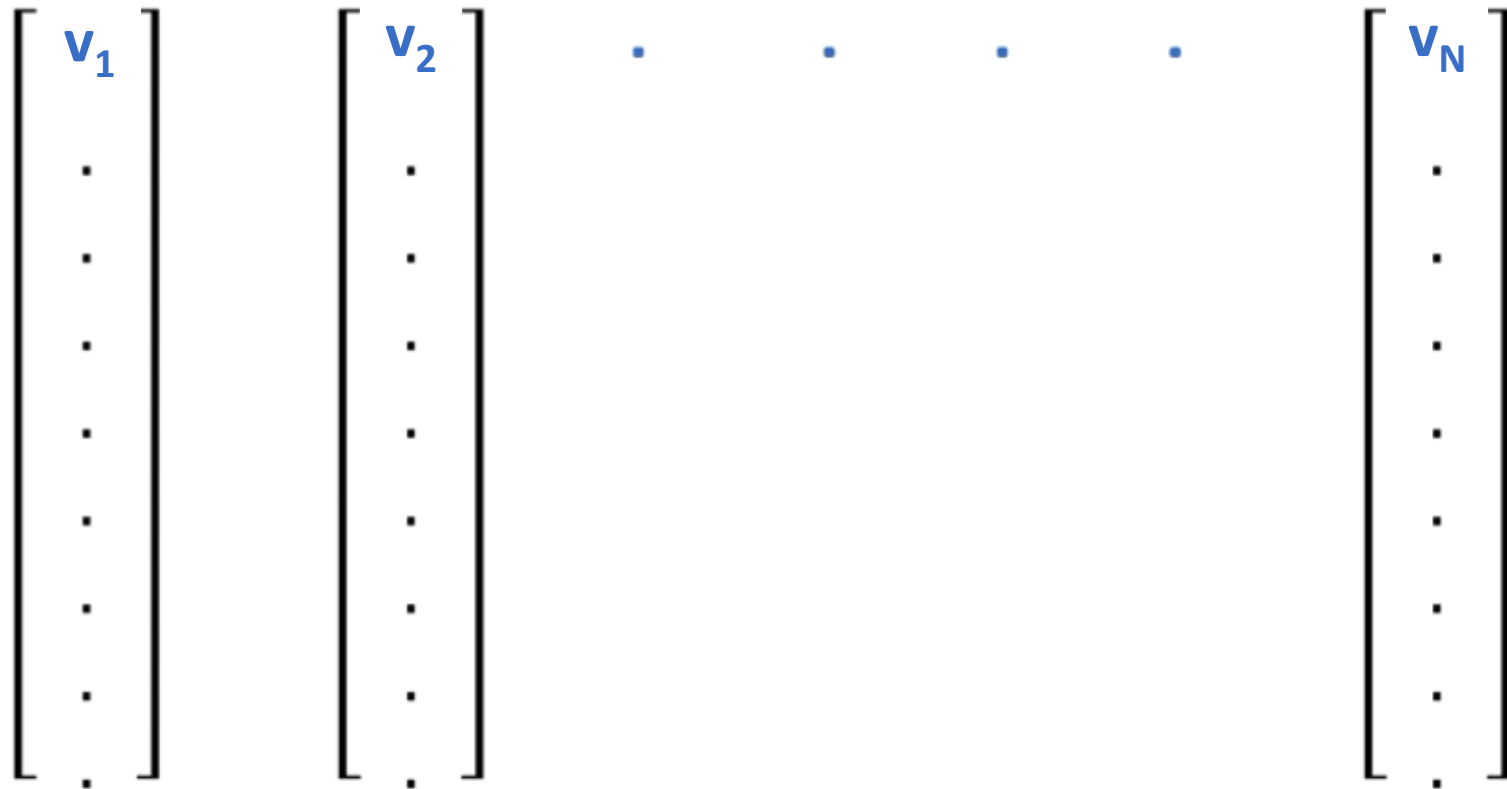
## Techniques for feature Reduction

Principal Component Analysis

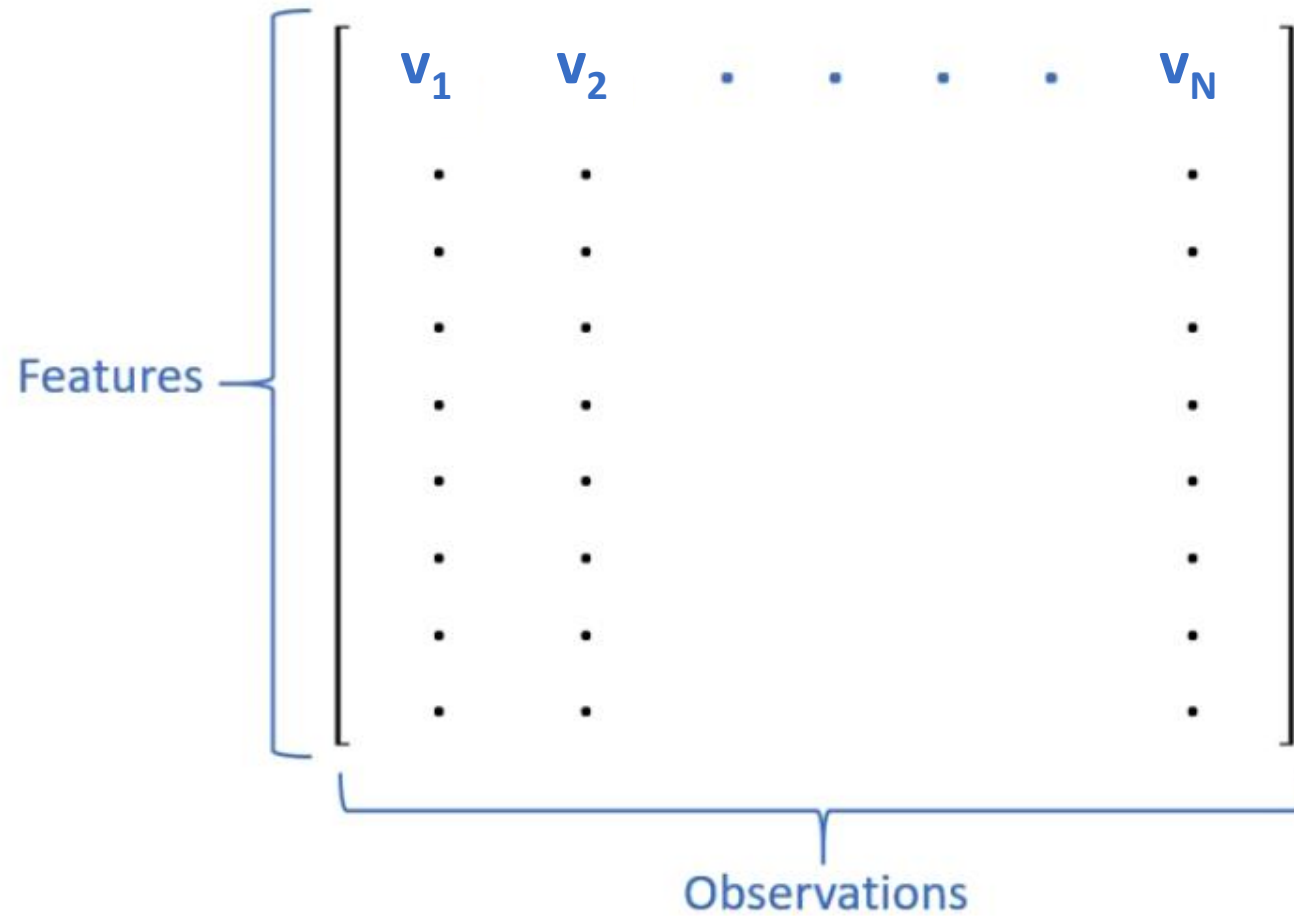
Independent Component Analysis

Non-Negative Matrix Factorization

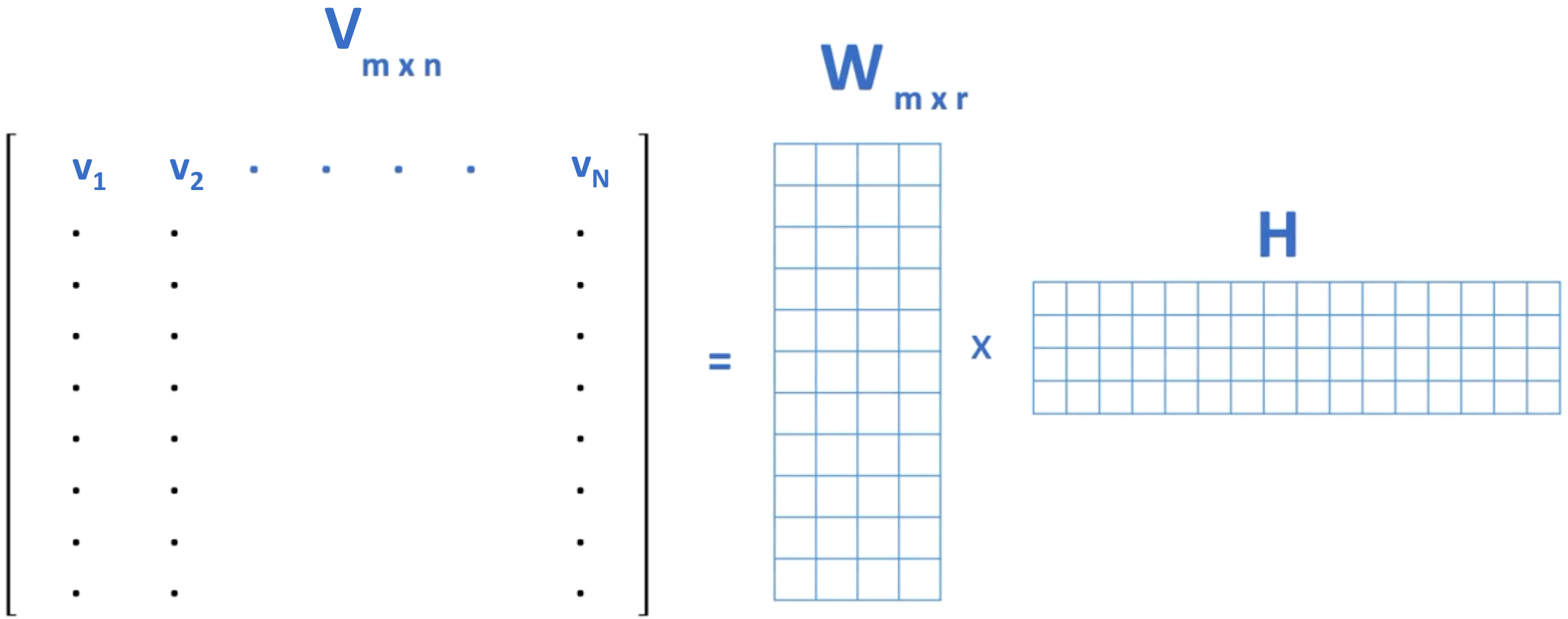




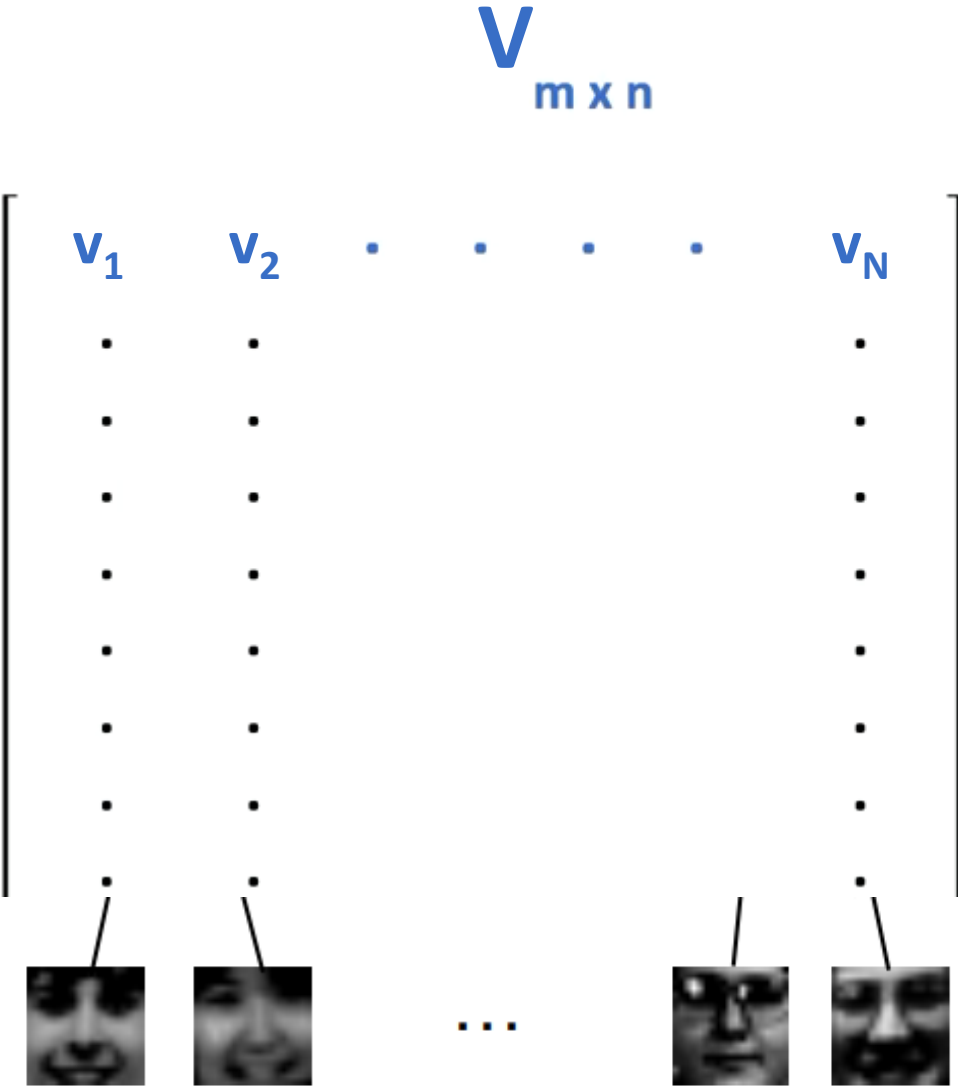
**Flattening all images**



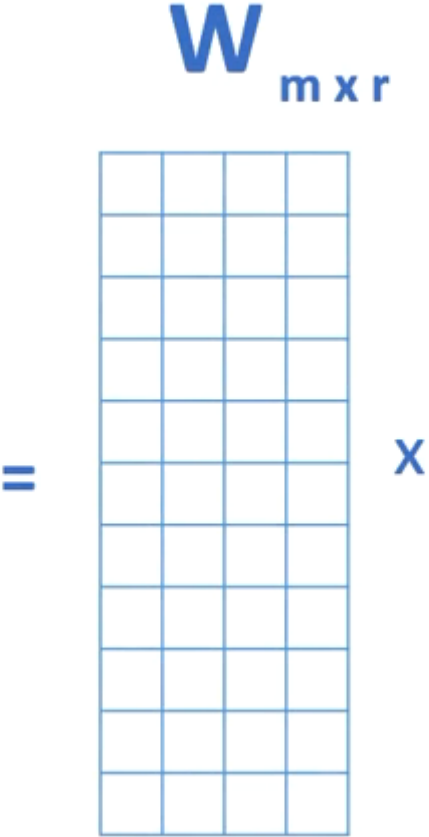
**Concatenating all vectors to form a matrix**

$$\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdot & \cdot & \cdot & \cdot & \mathbf{v}_N \\ \cdot & \cdot & & & & & \cdot \\ \cdot & \cdot & & & & & \cdot \\ \cdot & \cdot & & & & & \cdot \\ \cdot & \cdot & & & & & \cdot \\ \cdot & \cdot & & & & & \cdot \\ \cdot & \cdot & & & & & \cdot \\ \cdot & \cdot & & & & & \cdot \\ \cdot & \cdot & & & & & \cdot \end{bmatrix} = \begin{matrix} \mathbf{W} \\ \text{m} \times \text{r} \end{matrix} \times \begin{matrix} \mathbf{H} \\ \text{r} \times \text{n} \end{matrix}$$


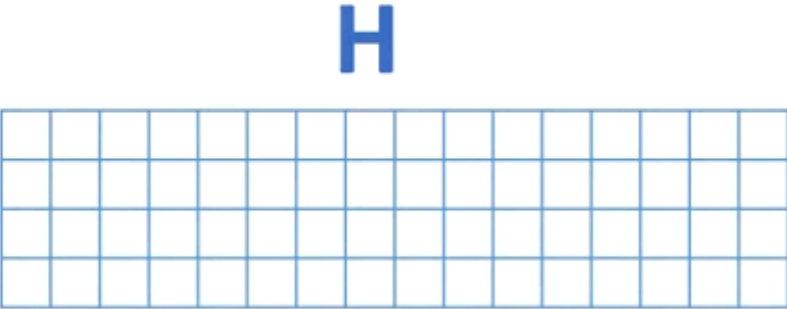
# Vectorised images



# Facial features



# Importance of features in each image



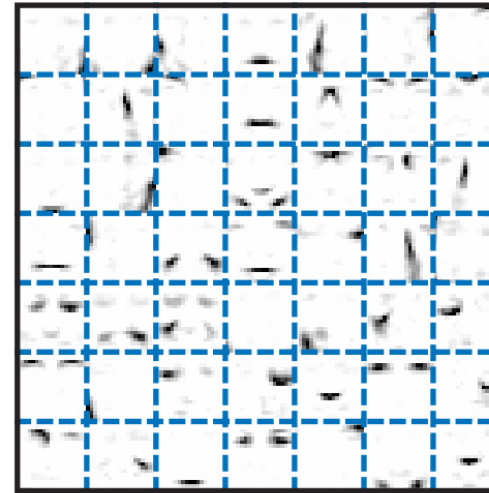
# Example

- Let's take, for example, a database of face images organized into a matrix  $V$ .
  - $V$  is  $361 \times 2,429$ , as in there are 2,429 faces that are composed of  $19 \times 19 = 361$  pixels, each
- We assign  $V$  49 bases ( $r = 49$ ) and decompose it into  $W$  and  $H$ .

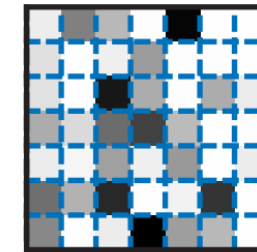
# NMF vs. PCA

- We see that NMF (top) forms bases that are parts of a face
  - ▣ The basis images are mostly empty space (sparse) and we see from the weighting (also sparse) that not all parts are used when adding together a representative face
  
- ▣ PCA formed bases of positive and negative pixels and the weights blend them together (neither is sparse)

NMF

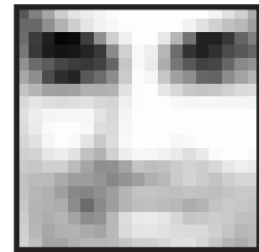
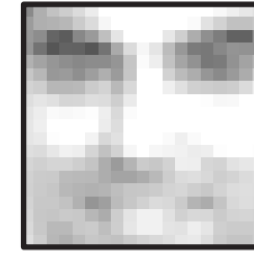


×

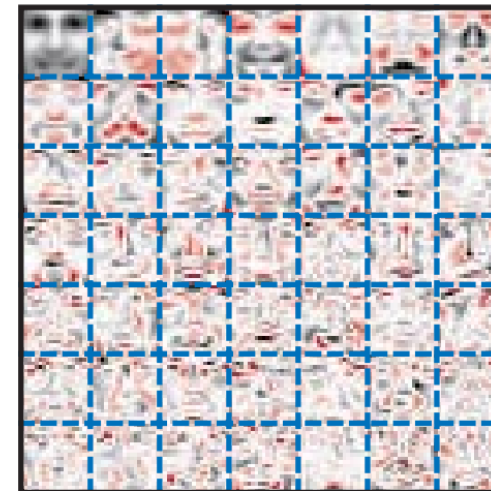


=

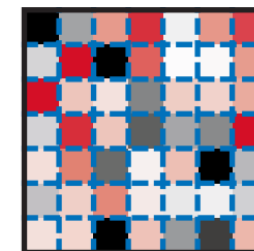
Original



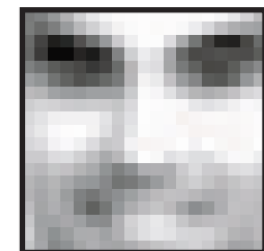
PCA



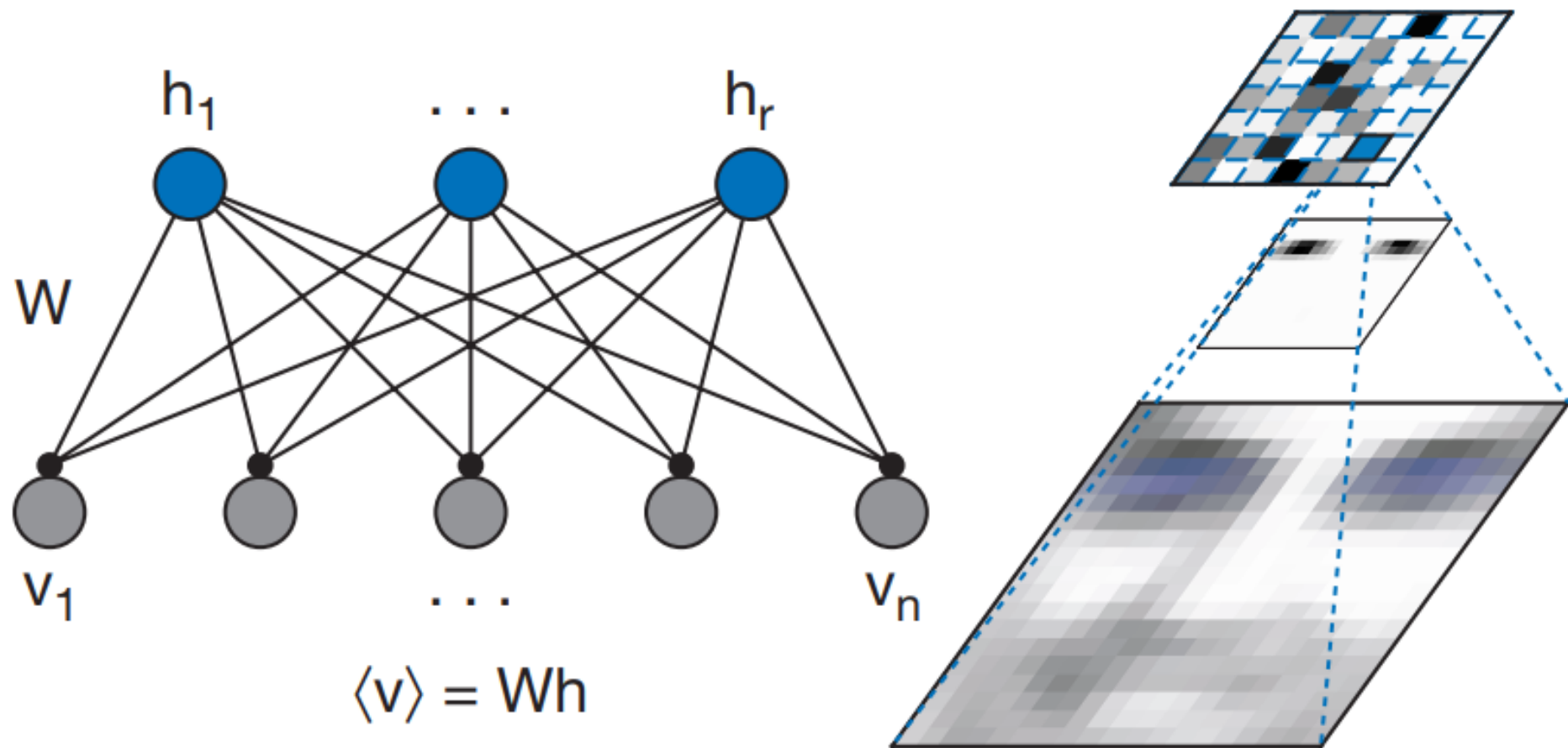
×



=







# NMF vs. SVD

Property	NMF	SVD
Formulation	$V = WH$	$V = U\Sigma V^T$
Optimality (in terms of squared distance)	⊘	✓
Speed & robustness	⊘	✓
Uniqueness	⊘	✓
Sensitivity to initialization	✓	⊘
Orthogonality	⊘	✓
Sparsity	✓	⊘
Non-negativity	✓	⊘
Interpretability	✓	⊘

# Optimization

*Problem: Minimize  $\|V - WH\|^2$  with respect to  $W$  and  $H$ , subject to the constraints  $W, H \geq 0$ .*

- ▣ Often, the Eucliden or Frobenius distance is used
- ▣ Possible numerical methods:
  - ▣ Coordinate Descent (alternate: fix  $W$  and optimize  $H$ , fix  $H$  and optimize  $W$  until tolerance is met)
  - ▣ Multiplicative Update using the following update step:

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^T V)_{a\mu}}{(W^T W H)_{a\mu}} \quad W_{ia} \leftarrow W_{ia} \frac{(V H^T)_{ia}}{(W H H^T)_{ia}}$$

- ▣ Note that both of these methods merely find a local minimum, but that is often useful enough

# How to calculate Multiplicative Update Rules (MURs)?

$$\min_{\mathbf{W}, \mathbf{H}} \|\mathbf{V} - \mathbf{W}\mathbf{H}\|_F^2, \text{ s.t. } \mathbf{W} \geq 0, \mathbf{H} \geq 0.$$

$$C = \|\mathbf{V} - \mathbf{W}\mathbf{H}\|_F^2 = \text{Tr} \left( (\mathbf{V} - \mathbf{W}\mathbf{H})^\top (\mathbf{V} - \mathbf{W}\mathbf{H}) \right)$$

$$C = \text{Tr} (\mathbf{V}^\top \mathbf{V} - \mathbf{V}^\top \mathbf{W}\mathbf{H} - \mathbf{H}^\top \mathbf{W}^\top \mathbf{V} + \mathbf{H}^\top \mathbf{W}^\top \mathbf{W}\mathbf{H})$$

$$\|\mathbf{A}\|_F = \sqrt{\text{Tr}(\mathbf{A}^\top \mathbf{A})}$$

$$\|\mathbf{A}\|_F^2 = \text{Tr}(\mathbf{A}^\top \mathbf{A})$$

Problem can be solved by alternating minimization

$$C = \text{Tr} (V^T V - \underset{\uparrow}{V^T W H} - \underset{\uparrow}{H^T W^T V} + \underset{\uparrow}{H^T W^T} \underset{\uparrow}{W H})$$

By fixing  $H$

$$\frac{\partial C}{\partial W} = -\underset{\uparrow}{V H^T} - \underset{\uparrow}{V H^T} + \underset{\uparrow}{W H H^T} + \underset{\uparrow}{W H H^T}$$

$$\frac{\partial C}{\partial W} = -2V H^T + 2W H H^T$$

$$W \leftarrow W \odot \frac{2V H^T}{2W H H^T}$$

$$W \leftarrow W \odot \frac{V H^T}{W H H^T}$$

By fixing  $W$

$$\frac{\partial C}{\partial H} = -W^T V - W^T V + W^T W H + W^T W H$$

$$\frac{\partial C}{\partial H} = -2W^T V + 2W^T W H$$

$$H \leftarrow H \odot \frac{2W^T V}{2W^T W H}$$

$$H \leftarrow H \odot \frac{W^T V}{W^T W H}$$

Oja Rule

$$W \leftarrow W \odot \frac{\{negative\_terms\}}{\{positive\_terms\}}$$

# Gradient descent algorithm

*Input:  $V$  dataset,  $r$  factor*

1. *Initializing  $W, H$  randomly*

2. *for  $i = 1 : \text{maxiter}$*

3.

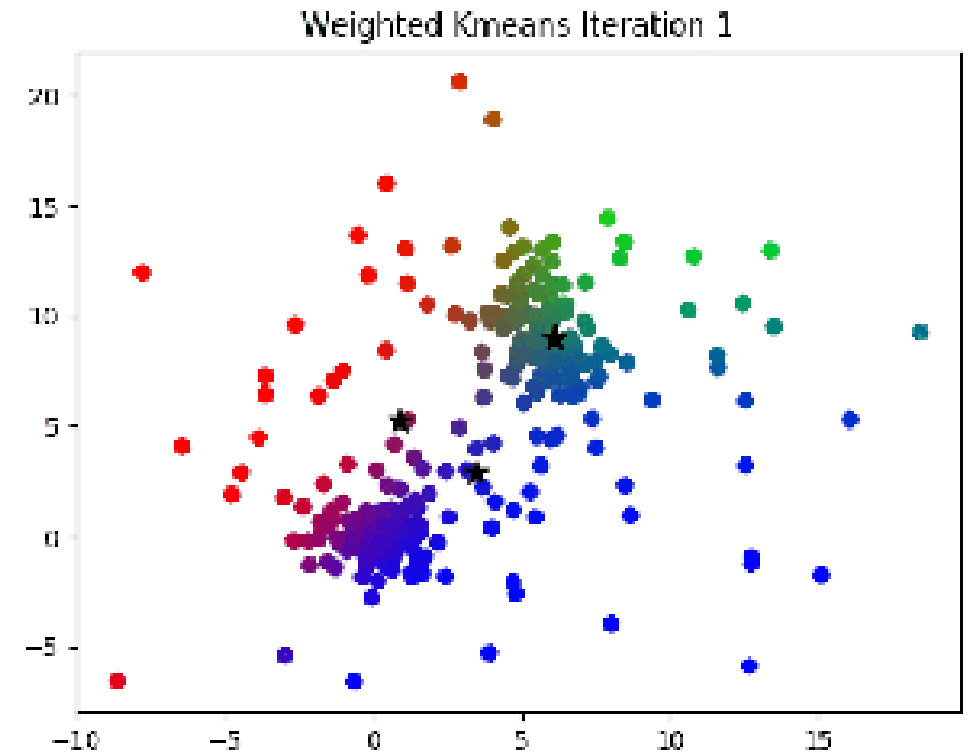
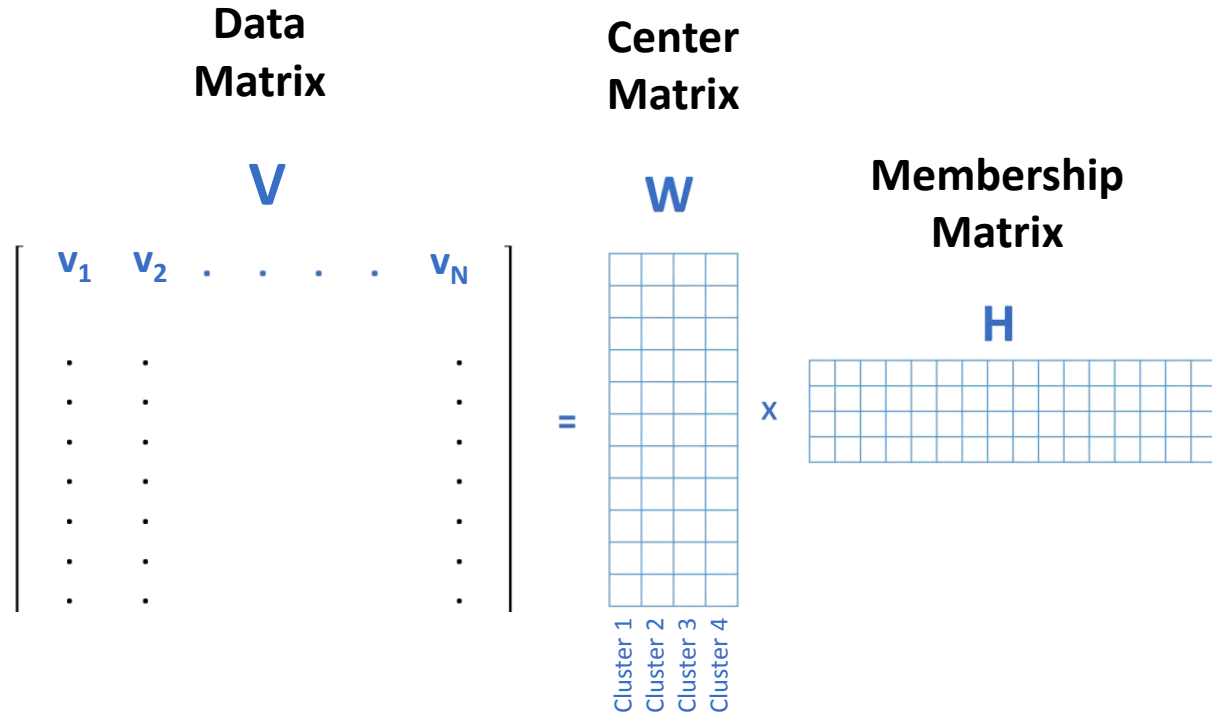
4. *Updating  $W$  by* 
$$W \leftarrow W \odot \frac{VH^T}{WHH^T}$$

5. *Updating  $H$  by* 
$$H \leftarrow H \odot \frac{W^T V}{W^T W H}$$

6. *end*

*Output:  $W, H$*

# NMF as a Data Clustering Algorithm



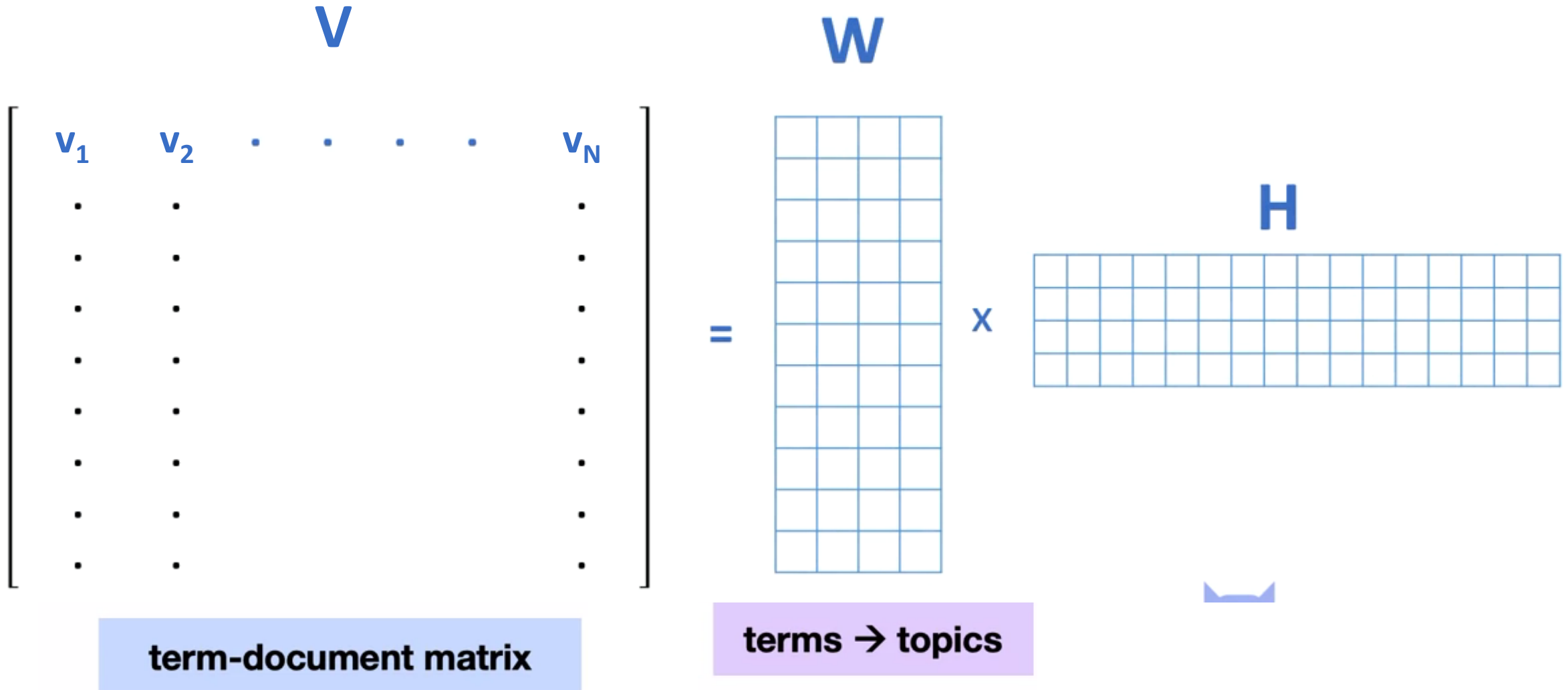
$$\min_{W, H} \|V - WH\|_F^2 = \sum_{i=1}^r \sum_{j=1}^n \|x_j - w_i\|_2^2 h_{ij}$$

# Example: Text Clustering

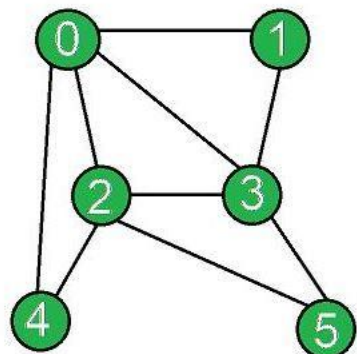
Dictionary	Document 1	Document 2
a	1	0
brown	1	0
dog	0	1
fox	1	0
jumped	0	1
lazy	0	1
over	0	1
quick	1	0
the	0	1



# Example: Text Clustering



# Example: Graph Clustering



	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	1	1
3	1	1	1	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	0	0

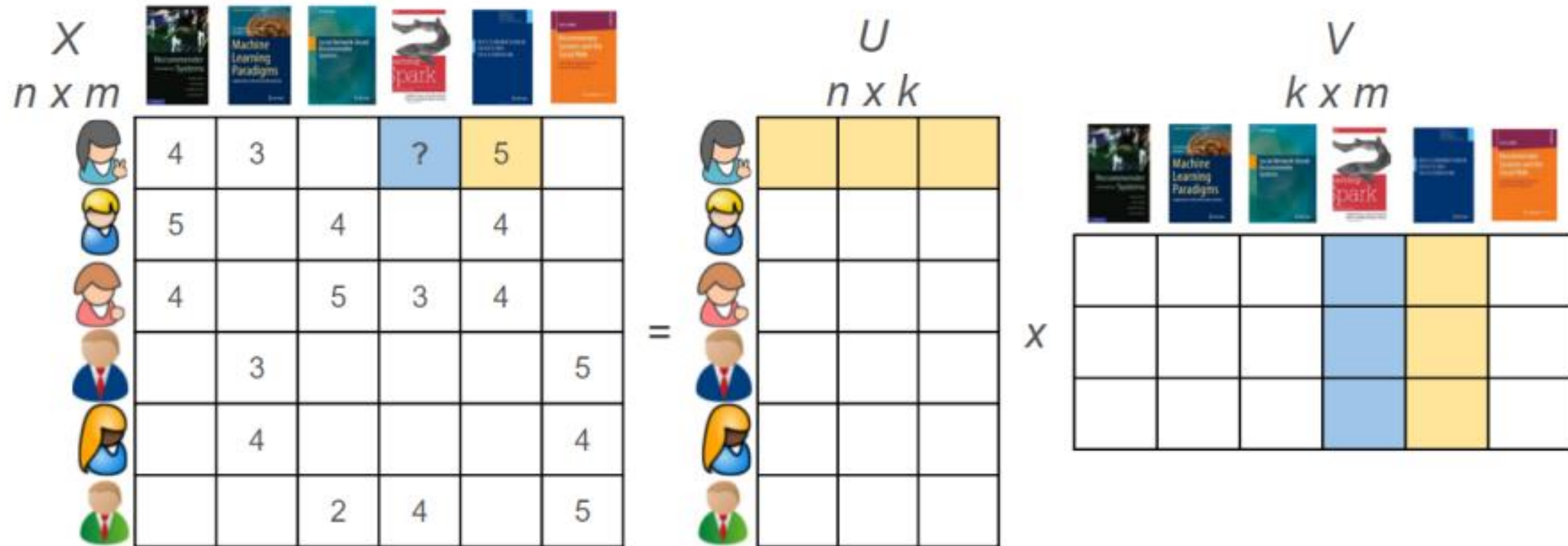
$A$

$$\min_{W, H} \|A - WH\|_F^2$$

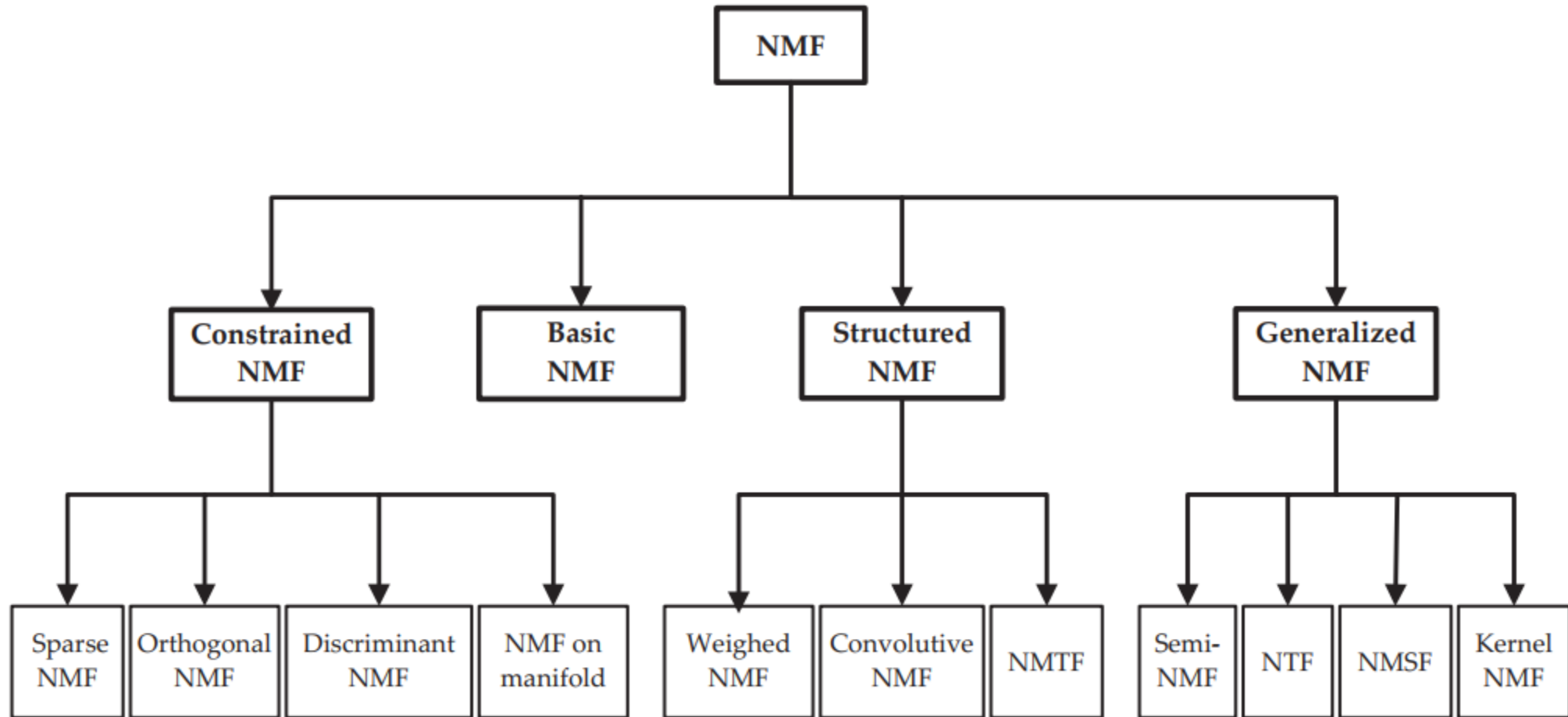
$$\min_H \|A - HH^\top\|_F^2$$

$$\min_{W, H} \|A - HWH^\top\|_F^2$$

# Example: Recommendation System



# categorization of NMF models



# Regularized and Constraint NMF

## Sparse NMF

$$\min_{\mathbf{W}, \mathbf{H}} \|\mathbf{A} - \mathbf{W}\mathbf{H}\|_F^2 + \lambda \sum_{i=1}^n \|\mathbf{h}_i\|_1$$

$$\min_{\mathbf{W}, \mathbf{H}} \|\mathbf{A} - \mathbf{W}\mathbf{H}\|_F^2 + \lambda \sum_{i=1}^n \|\mathbf{w}_i\|_2^2$$

$$\min_{\mathbf{W}, \mathbf{H}} \|\mathbf{A} - \mathbf{W}\mathbf{H}\|_F^2 + \lambda \|\mathbf{W}\|_F^2$$

## Orthogonal NMF

$$\text{ONMF-H} : \min_{\mathbf{W}, \mathbf{H}} \|\mathbf{A} - \mathbf{W}\mathbf{H}\|_F^2$$

$$\text{s.t. } \mathbf{W}, \mathbf{H} \geq 0, \mathbf{H}^\top \mathbf{H} = \mathbf{I}.$$

$$\text{ONMF-W} : \min_{\mathbf{W}, \mathbf{H}} \|\mathbf{A} - \mathbf{W}\mathbf{H}\|_F^2$$

$$\text{s.t. } \mathbf{W}, \mathbf{H} \geq 0, \mathbf{W}\mathbf{W}^\top = \mathbf{I}.$$

# NMF: the Syntax

Import the class containing the clustering method.

```
from sklearn.decomposition import NMF
```

Create an instance of the class.

```
nmf = NMF(n_components=3, init='random')
```

Fit the instance and create transformed version of the data:

```
X_nmf = NMF.fit(X)
```