Algorithms for Non-negative Matrix Factorization

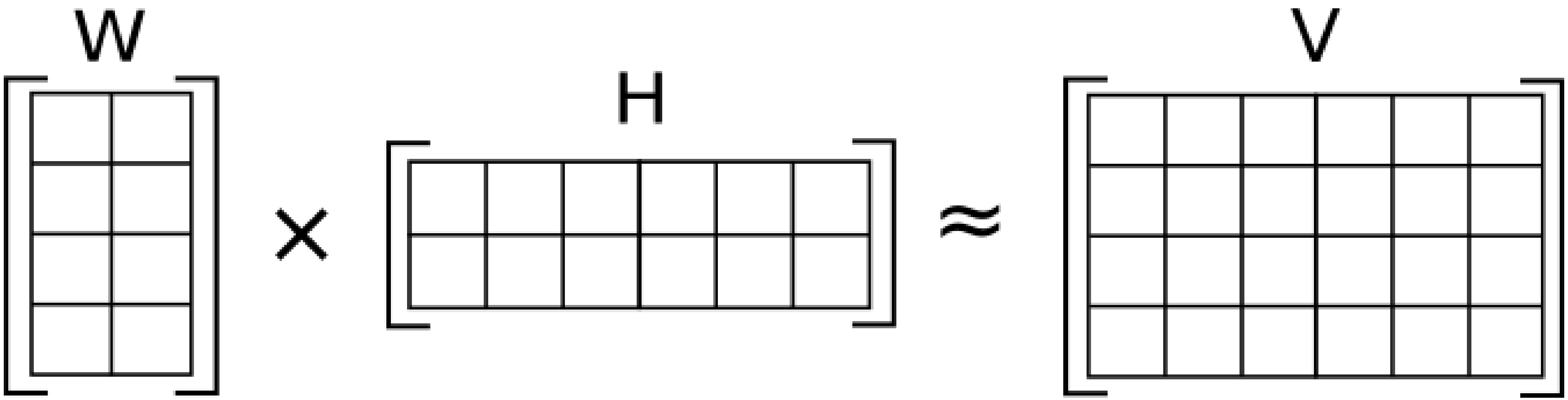
Zhang Jiyuan

zhangjiyuan2017@ia.ac.cn

Data is often non-negative by nature

- Pixel intensities
- Amplitude spectra
- Occurrence counts
- Food or energy consumption
- User scores
- Stock market values
- …

Non-negative Matrix Factorization



$W = [w_{fk}] \quad s.t. \;\; w_{fk} \geq 0$

$V = [v_{fn}] \quad s.t. \;\; v_{fn} \geq 0$

$H = [h_{kn}] \quad s.t. \;\; h_{kn} \geq 0$

V ≈ WH

- V : the $F \times N$ data matrix
  - N features (rows), M examples (columns)

- W : the $F \times K$ dictionary matrix
  - w_k is a basis vector among K elements

- H : the $K \times N$ activation/expansion matrix
  - h_k is the row vector of activation coefficients relating to basis vector w_k

Cost functions

- Minimize the distance between V and WH
- Measures of distance between two non-negative matrices A and B
  - Euclidean distance $$||A - B||^2 = \sum_{ij} (A_{ij} - B_{ij})^2$$
  - KL divergence $$D(A||B) = \sum_{ij} \left(A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij} \right)$$

Alternating optimization strategy

- The problem is usually easier to optimize over one matrix given the other matrix is known and fixed.

- $D(V|W,H)$ is even convex separately.

Update W, given H fixed
Update H, given W fixed
Until converge

Multiplicative update rules

- Measure with Euclidean distance

$$H_{\alpha\mu} \leftarrow H_{\alpha\mu} \frac{(W^T V)_{\alpha\mu}}{(W^T W H)_{\alpha\mu}} \qquad W_{i\alpha} \leftarrow W_{i\alpha} \frac{(V H^T)_{i\alpha}}{(W H H^T)_{i\alpha}}$$

- Measure with KL-divergence

$$H_{\alpha\mu} \leftarrow H_{\alpha\mu} \frac{\sum_i W_{i\alpha} V_{i\mu}/(WH)_{i\mu}}{\sum_k W_{k\alpha}} \qquad W_{i\alpha} \leftarrow W_{i\alpha} \frac{\sum_\mu H_{\alpha\mu} V_{i\mu}/(WH)_{i\mu}}{\sum_\nu H_{\alpha\nu}}$$

Why use NMF ?

- Non-negativity induces sparsity
  - NMF constructs sparse bases and sparse weightings
- Non-negativity leads to part-based decompositions
  - . Combine parts to form a whole

.

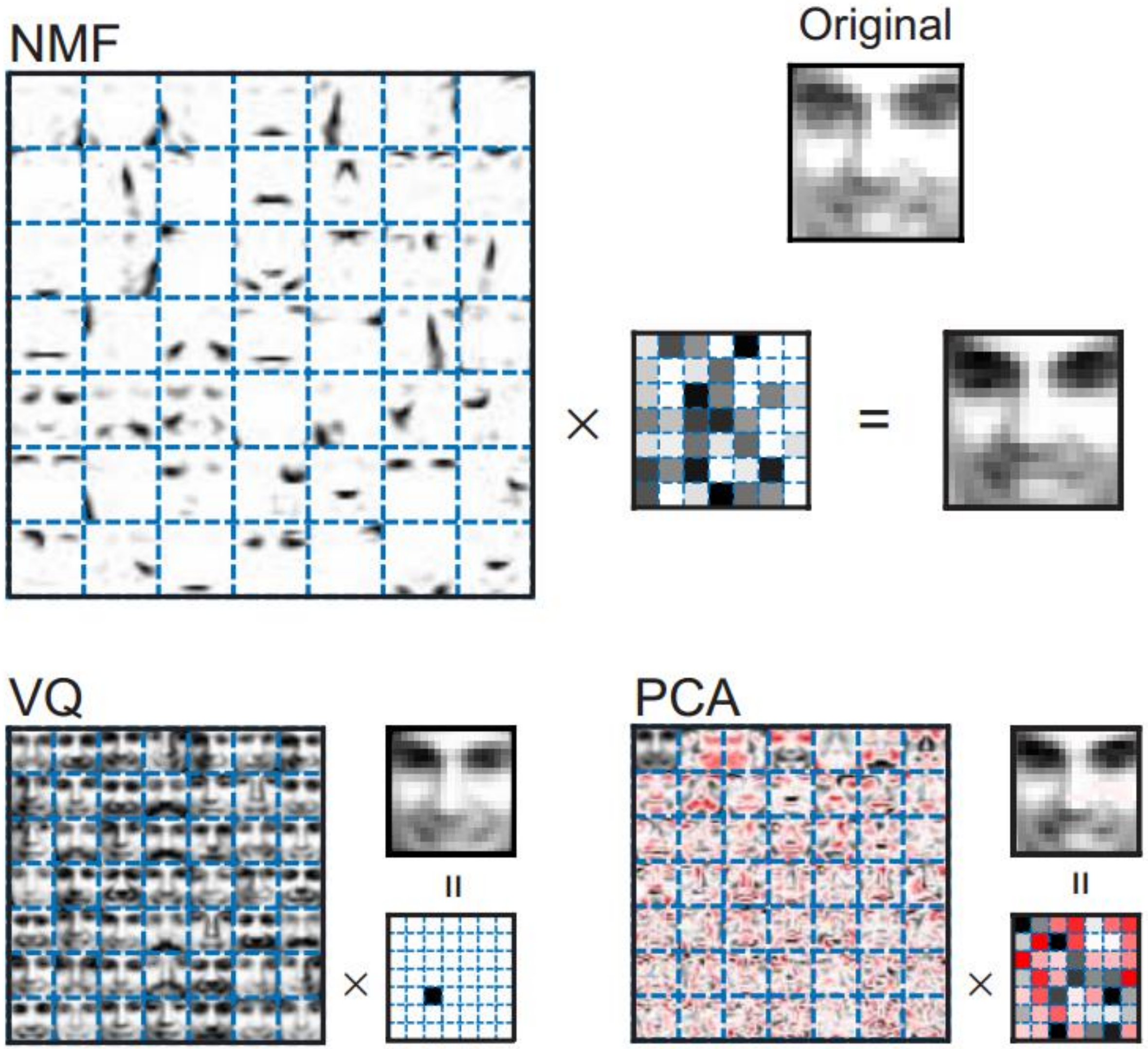
Learning the parts of objects

NMF

- The basis images are mostly empty space(sparse)
- The weighting images shows that not all parts are used

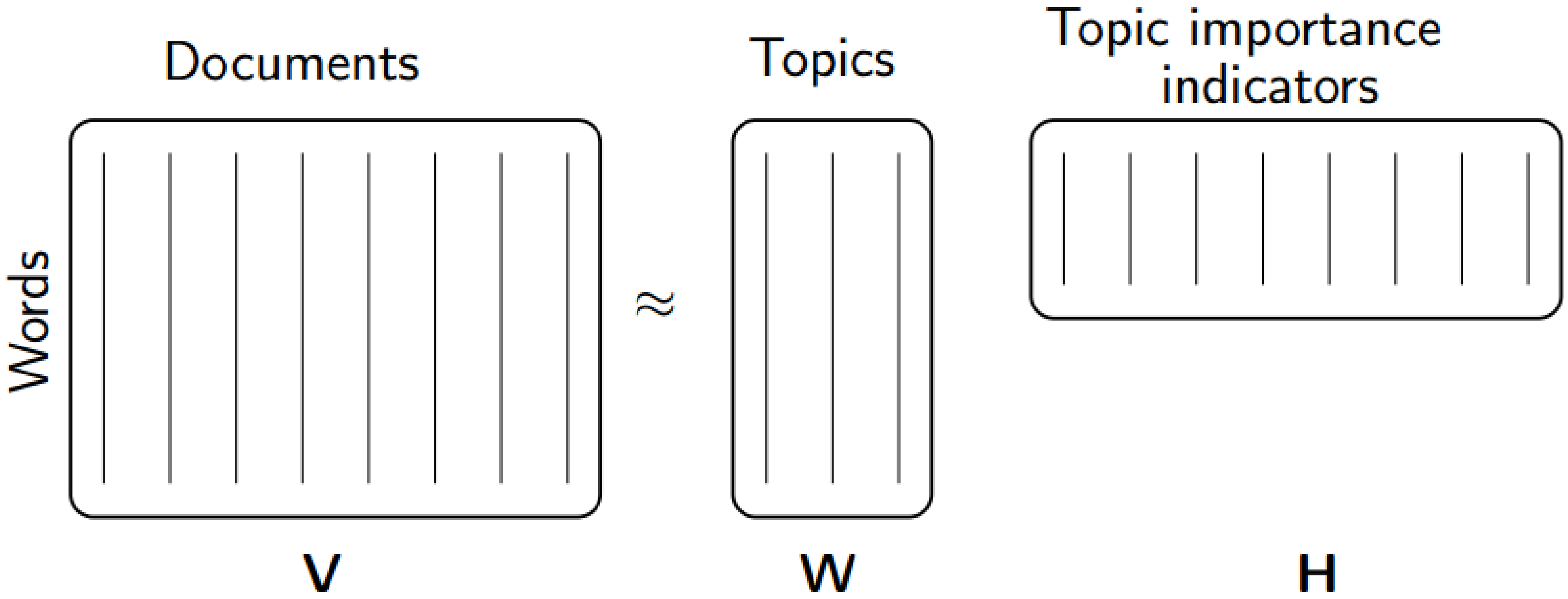
VQ formed bases of prototype faces from the dataset and the nearest one to the new face was selected

PCA formed bases of positive and negative pixels and the weights blend them together



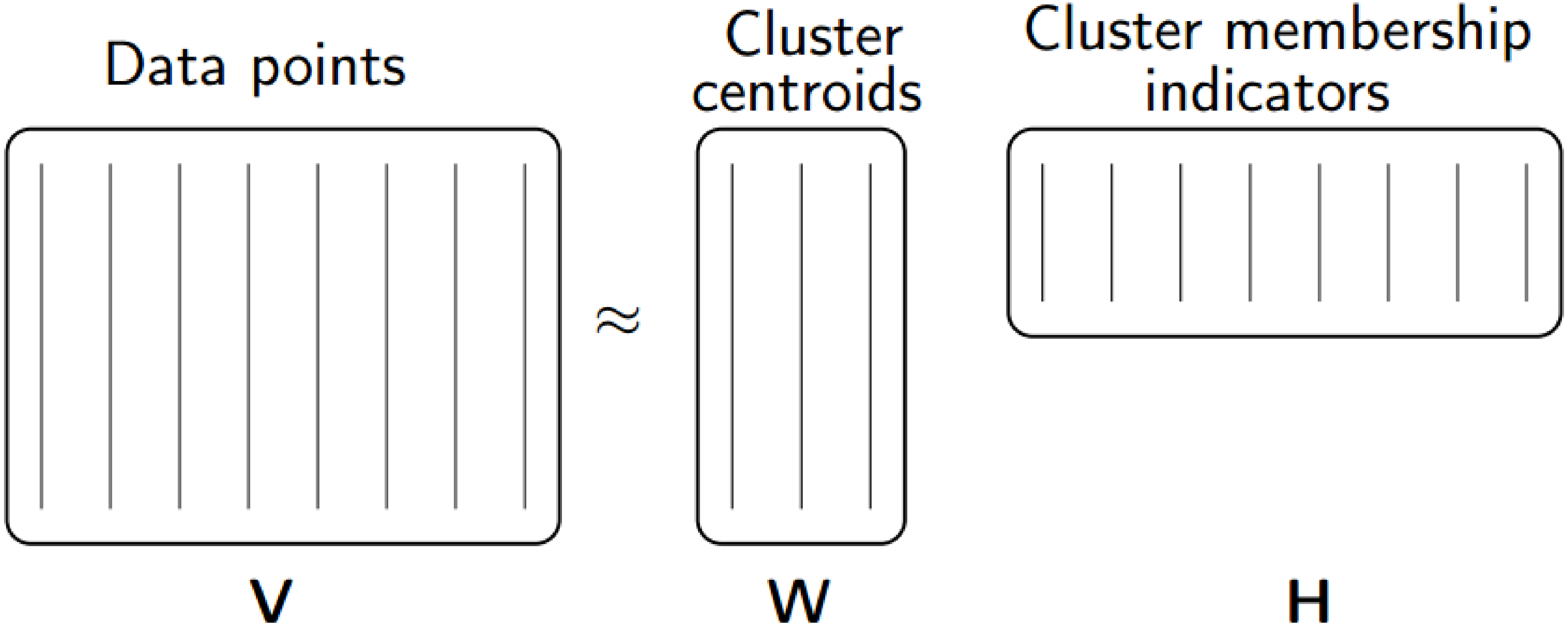
Usages of NMF: Topics recovery

assume $V = [v_{fn}]$ is a (scaled) term-document co-occurrence matrix:
v_{fn} is the frequency of occurrences of word m_f in document d_n



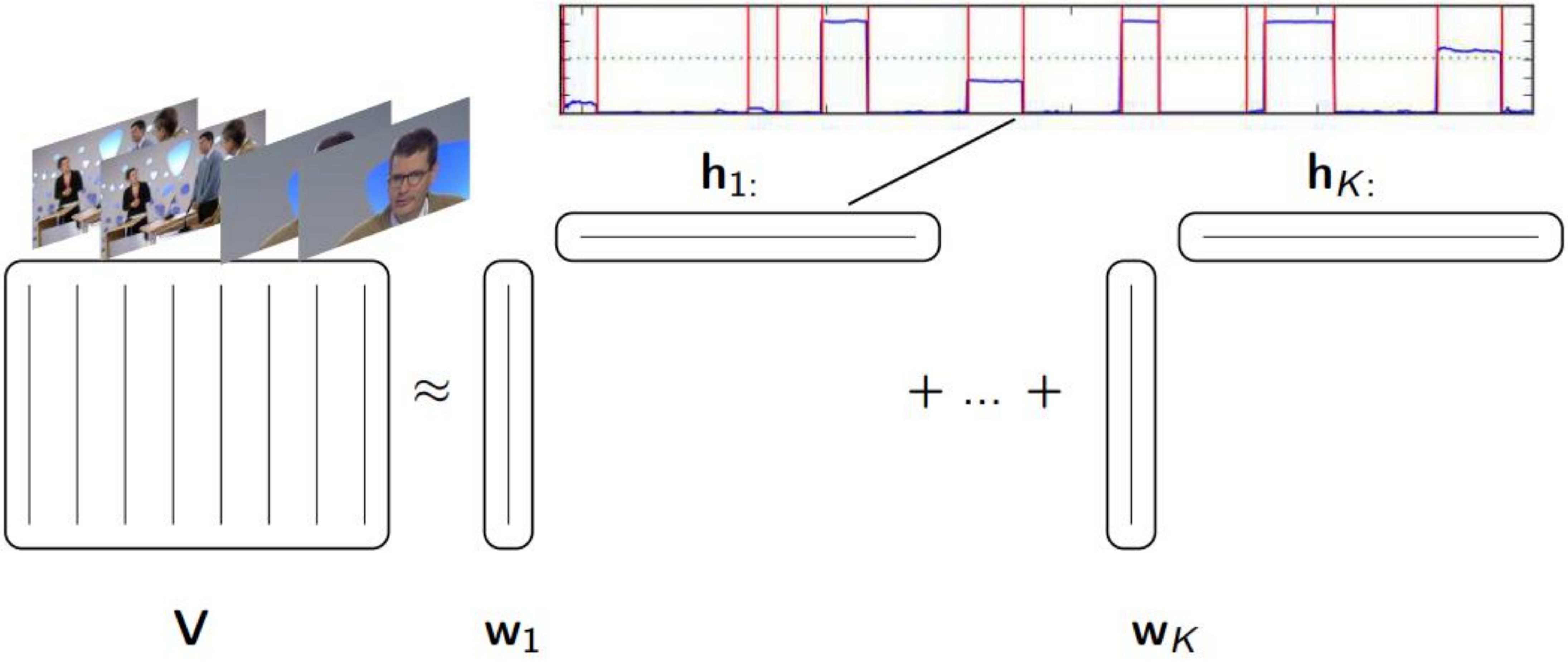
Usages of NMF: Clustering

NMF can handle overlapping clusters and provides soft cluster membership indications.

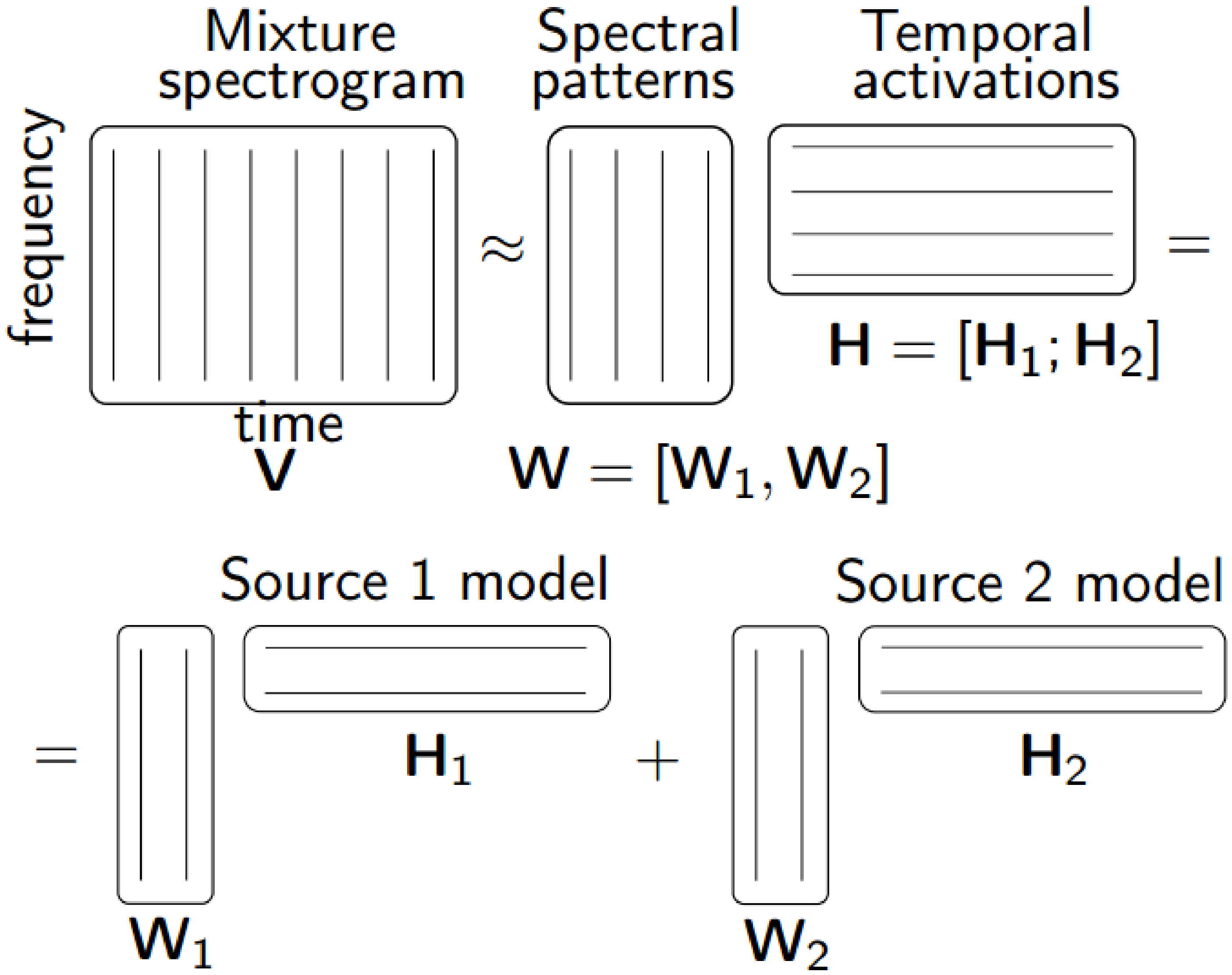


Usages of NMF: temporal segmentation

analysing temporal data sequences, e.g., videos:



Usages of NMF: filtering and source separation



Easy implement

In [1]:
```
import numpy as np
from sklearn.decomposition import NMF
```

In [2]:
```
V = np.array([[1, 1], [2, 1], [3, 1.2], [4, 1], [5, 0.8], [6, 1]])
model = NMF(n_components=2, init='random', random_state=0)
W = model.fit_transform(V)
H = model.components_
```

In [3]:
```
np.dot(W,H)
```

Out[3]:
```
array([[ 1.00063558,  0.99936347],
       [ 1.99965977,  1.00034074],
       [ 2.99965485,  1.20034566],
       [ 3.9998681 ,  1.0001321 ],
       [ 5.00009002,  0.79990984],
       [ 6.00008587,  0.999914  ]])
```

In [4]:
```
V
```

Out[4]:
```
array([[ 1. ,  1. ],
       [ 2. ,  1. ],
       [ 3. ,  1.2],
       [ 4. ,  1. ],
       [ 5. ,  0.8],
       [ 6. ,  1. ]])
```

Thanks!