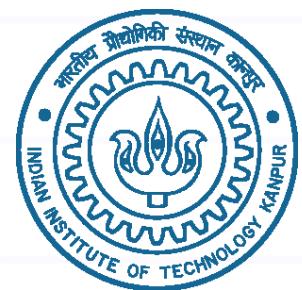


# Non-negative Matrix Factorization III

---

*EE698V - Machine Learning for Signal Processing*

Vipul Arora



# NMF: for Reconstruction

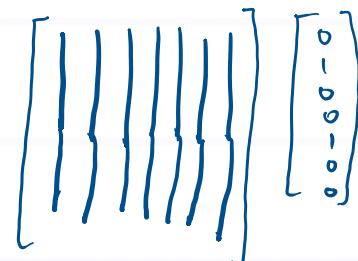
DICTIONARY



cols are  
dictionary  
elements  
↓  
coefficients  
or activations

$$x = Wh$$

$(N \times 1) \quad (N \times 7)(7 \times 1)$



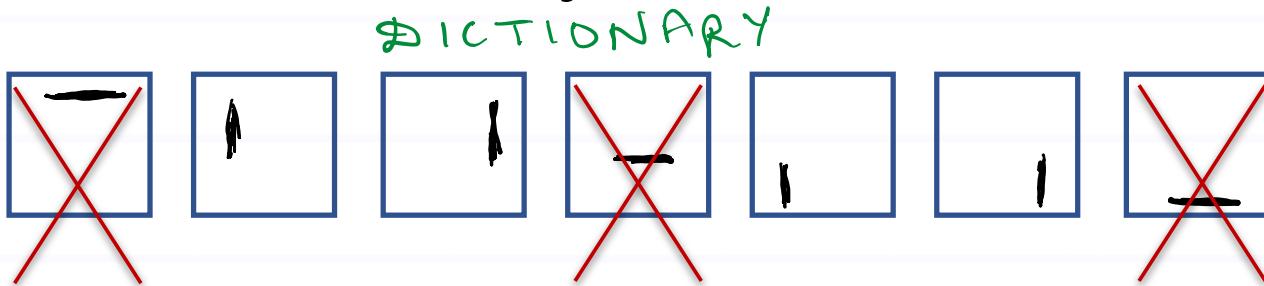
$$\begin{bmatrix} 1 \end{bmatrix} \approx \begin{bmatrix} 1 \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix}$$

$$\begin{bmatrix} 2 \end{bmatrix} \approx \begin{bmatrix} 1 \end{bmatrix} + \begin{bmatrix} - \end{bmatrix} + \begin{bmatrix} - \end{bmatrix} + \begin{bmatrix} - \end{bmatrix} + \begin{bmatrix} - \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix}$$

$$\begin{bmatrix} 3 \end{bmatrix} \approx \begin{bmatrix} 1 \end{bmatrix} + \begin{bmatrix} - \end{bmatrix} + \begin{bmatrix} - \end{bmatrix} + \begin{bmatrix} - \end{bmatrix} + \begin{bmatrix} - \end{bmatrix} + \begin{bmatrix} 1 \end{bmatrix}$$

# NMF: for Controlled Reconstruction

- Construct only the vertical lines of digits



$$\begin{array}{c|c} \boxed{1} & \approx \end{array} \begin{array}{c|c} \boxed{1} & + \end{array} \begin{array}{c|c} \boxed{1} & = \end{array} \begin{array}{c|c} \boxed{1} & \end{array}$$

$$\begin{array}{c|c} \boxed{2} & \approx \end{array} \begin{array}{c|c} \boxed{1} & + \end{array} \begin{array}{c|c} \boxed{1} & = \end{array} \begin{array}{c|c} \boxed{1} & \end{array}$$

$$\begin{array}{c|c} \boxed{3} & \approx \end{array} \begin{array}{c|c} \boxed{1} & + \end{array} \begin{array}{c|c} \boxed{1} & = \end{array} \begin{array}{c|c} \boxed{1} & \end{array}$$

$$h = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$h = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix}$$

$$h = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}$$

# NMF: for Clustering

---

- Similar  $v$  have similar activations

$$V = WH$$

$$\begin{bmatrix} \text{wavy} & \text{wavy} & \text{wavy} & \text{wavy} & \text{wavy} \\ \text{wavy} & \text{wavy} & \text{wavy} & \text{wavy} & \text{wavy} \end{bmatrix} = \begin{bmatrix} \text{wavy} & \text{wavy} & \text{wavy} \\ \text{wavy} & \text{wavy} & \text{wavy} \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$\underbrace{v}_{\sim} \dots$

cluster your  $H$   
with std. clustering  
algo.

# NMF: for Dimensionality Reduction

---

- Learn  $W$  from a huge database

$$V = WH$$

$$\begin{bmatrix} \text{z} & \text{z} & \text{z} & \text{z} & \text{z} \\ \text{z} & \text{z} & \text{z} & \text{z} & \text{z} \\ \text{z} & \text{z} & \text{z} & \text{z} & \text{z} \\ \text{z} & \text{z} & \text{z} & \text{z} & \text{z} \end{bmatrix} = \begin{bmatrix} \text{z} & \text{z} \\ \text{z} & \text{z} \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

- Use  $W$  to convert a new vector  $v$  into  $h$

$$v = Wh$$

# NMF: for Classification

class 1:

$$\begin{bmatrix} \{ \} & \{ \} & \{ \} & \{ \} \end{bmatrix}_{N \times 4} = \begin{bmatrix} > \{ & \{ \\ > \{ & \{ \end{bmatrix}_{N \times 3} \begin{bmatrix} 0.1 & 0.8 & \dots \\ 0.9 & 1.7 & \\ 1.0 & 0.2 & \end{bmatrix}_{3 \times 4}$$

Learn  $W_1$

class 2:

$$\begin{bmatrix} \{ \} & \{ \} & \{ \} & \{ \} \end{bmatrix} = \begin{bmatrix} \{ \} & \{ \} \\ \{ \} & \{ \} \end{bmatrix} \begin{bmatrix} 0.2 & 0.7 & \dots \\ 1.2 & 1.4 & \\ 0.3 & 0.3 & \end{bmatrix}_{3 \times 4}$$

Learn  $W_2$

Test:

$$\begin{bmatrix} \{ \} & \{ \} \end{bmatrix} = \begin{bmatrix} > \{ & \{ \\ > \{ & \{ \end{bmatrix} \begin{bmatrix} W_1 & W_2 \\ \{ \} & \{ \} \end{bmatrix} \begin{bmatrix} 0.1 & 0 & \\ 0.9 & 0.05 & \\ 0.5 & 0 & \\ 0 & 1.1 & \\ 0.1 & 0.7 & \\ 0 & 0.3 & \end{bmatrix}_{3 \times 4} \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}$$

- Use  $W_1, W_2$  to estimate  $H_1, H_2$
- classify using  $H_1, H_2$

# NMF: for Separation

- Linearly added vectors from different classes can be separated

$$\begin{bmatrix} \boxed{1} & \boxed{2} & \boxed{3} & \dots \end{bmatrix} \approx \begin{bmatrix} - & 1 & 1 & - & 1 & 1 & - \end{bmatrix} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}}_{w_1} \begin{bmatrix} 0 & 1 & 0 & \dots \end{bmatrix}$$

• Learn  $w_1$

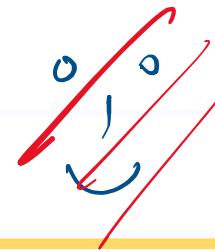
$$\begin{bmatrix} \checkmark & \times & \dots \end{bmatrix} \approx \begin{bmatrix} / & \backslash & \backslash \end{bmatrix} \underbrace{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}}_{w_2} \begin{bmatrix} 1 & 1 & \dots \end{bmatrix}$$

• Learn  $w_2$

$$\begin{bmatrix} \checkmark & \times \end{bmatrix} \approx \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \underbrace{w_1 h_1}_{\text{digit}} + \underbrace{w_2 h_2}_{\text{checkmark}} = \boxed{3} + \boxed{\times}$$

• use  $[w_1, w_2]$  to estimate  $h_1, h_2$  & reconstruct

# NMF: for In-painting



- Missing parts could be refilled or reconstructed using dictionary learnt beforehand

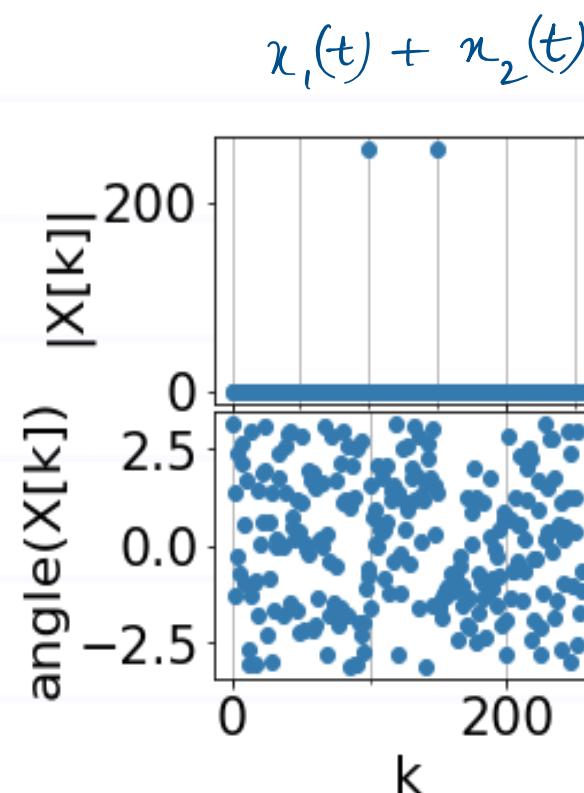
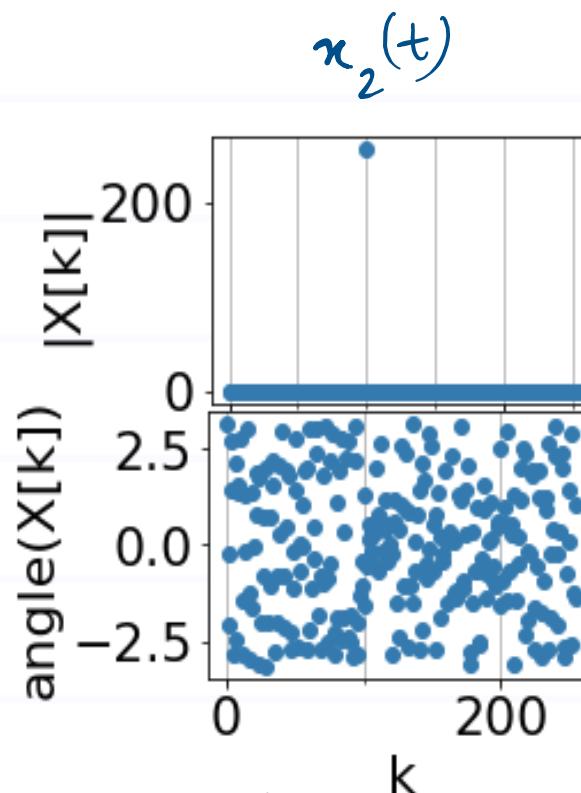
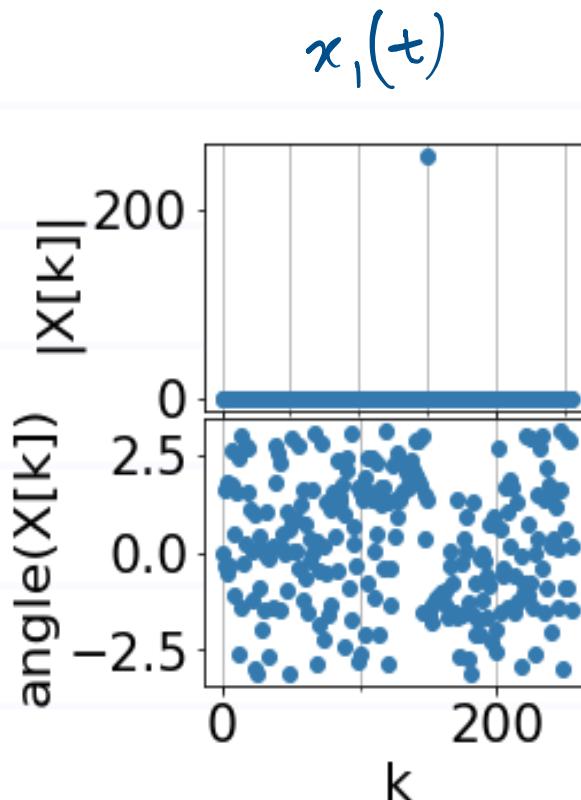
$$\begin{bmatrix} ; \\ ; \end{bmatrix} \approx \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} - \\ J \\ - \end{bmatrix} \approx \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} - \\ - \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

# Time Series

$$X[k] \propto \sum_n \sin\left(2\pi f \frac{n}{f_s}\right) e^{-j 2\pi \frac{kn}{N}}$$
$$\bar{f} = \frac{k}{N}$$

- Linear addition of periodic signals

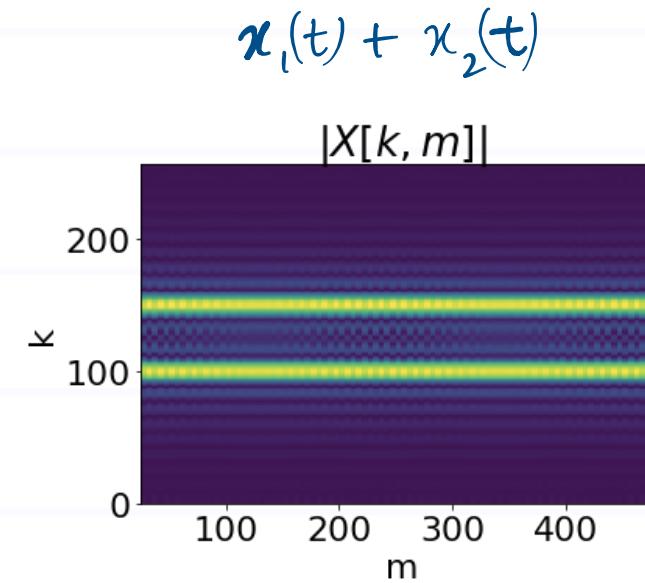
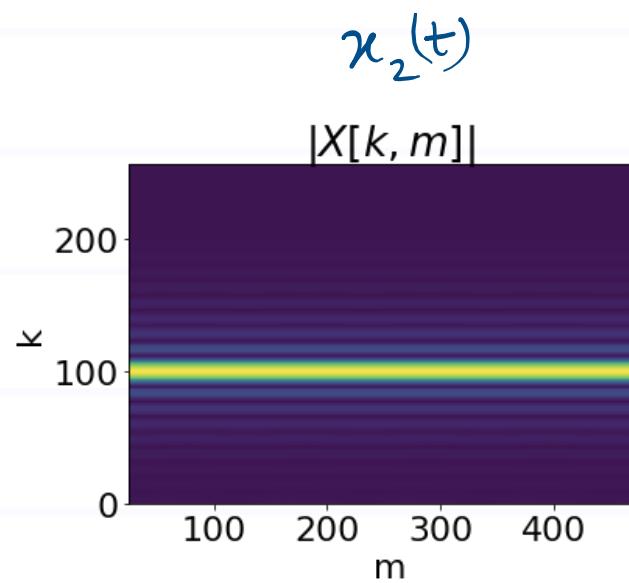
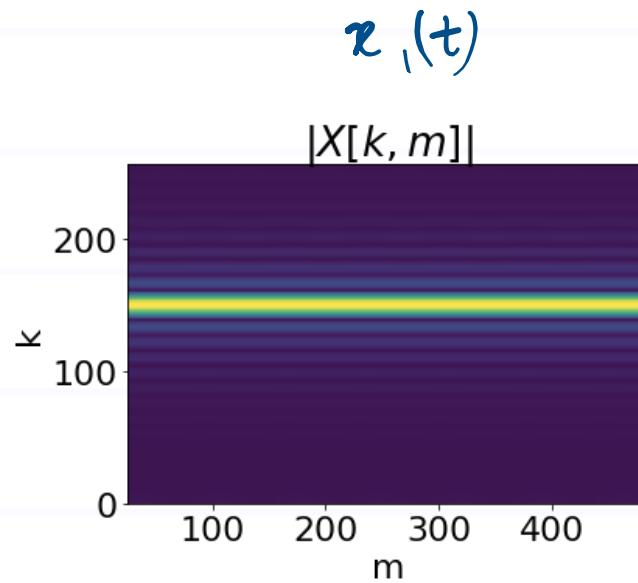


Notice that:  $|x_1[k] + x_2[k]| \approx |x_1[k]| + |x_2[k]|$

# Time Series

---

- Note that  $|X[k]|$  can be assumed to add linearly
- Same is observed with  $|X[k, m]|$  as well



# NMF for time series

---

- We can use this assumption to use NMF for time series analysis
- A very popular application is audio source separation

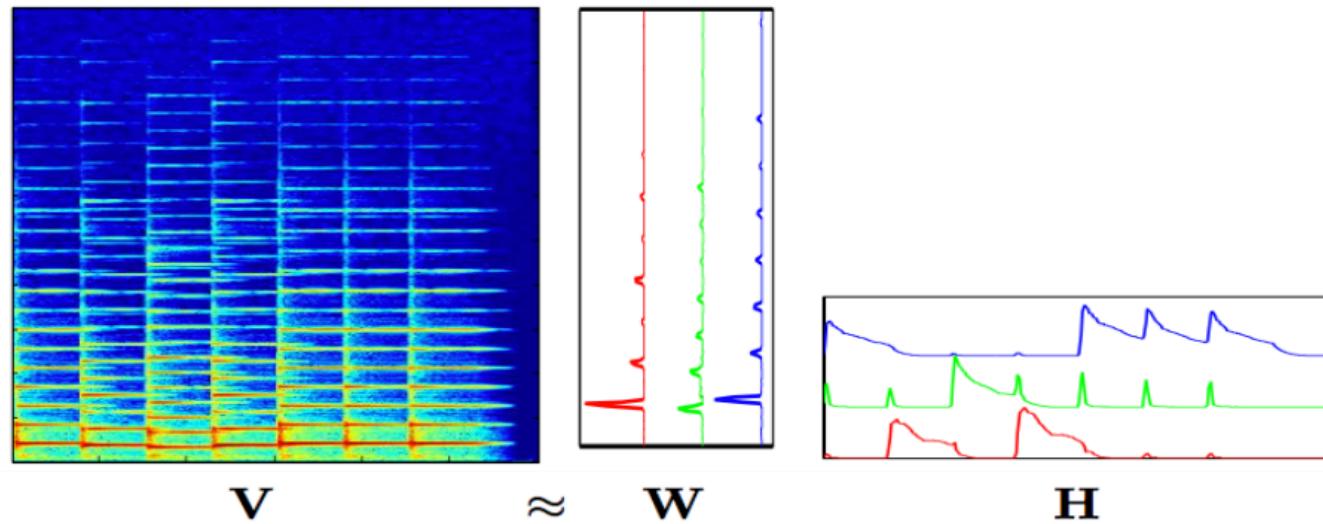
# NMF for Audio

---

- Given an audio recording  $x[n]$  sampled at  $F_s$
- Obtain  $|X[k, m]|$

# NMF on STFT: $|X[k]|$ as a feature vector

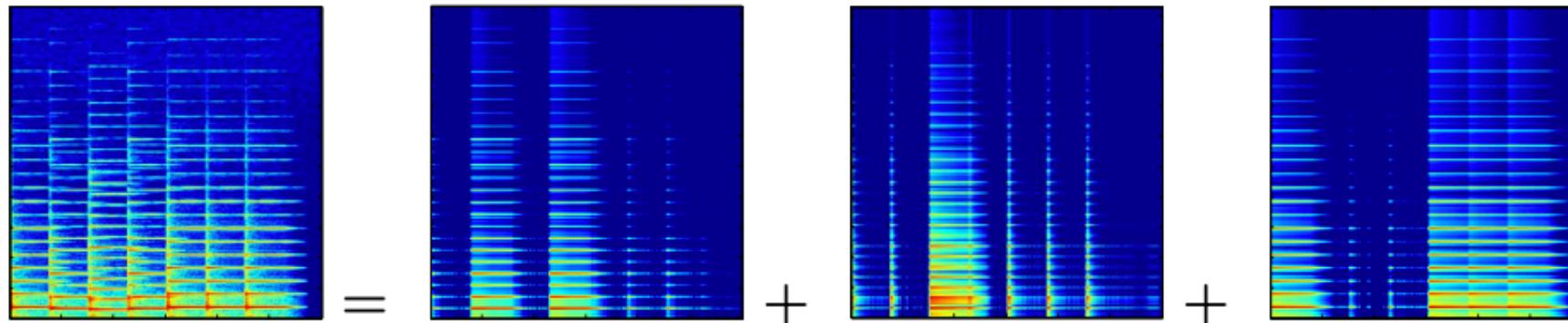
---



Courtesy: <https://ccrma.stanford.edu/~njb/teaching/sstutorial/part2.pdf>

# Reconstruct each pitch separately

---



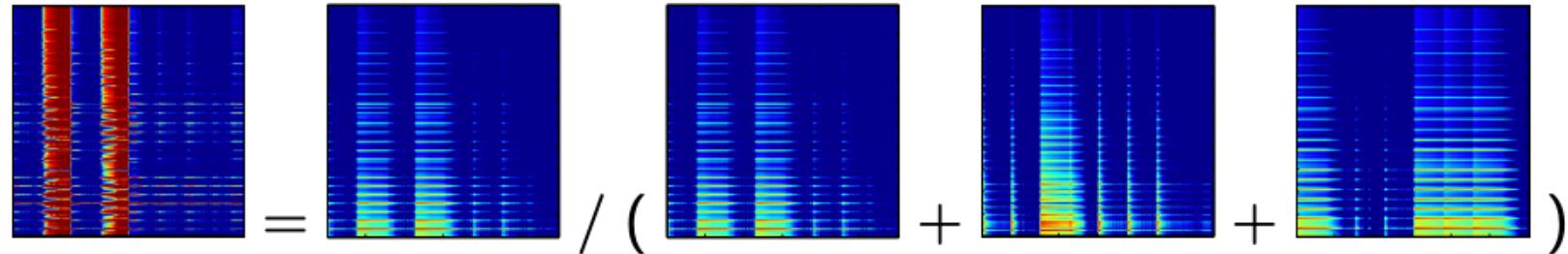
$$\mathbf{V} \approx \mathbf{w}_1 \mathbf{h}_1^T + \mathbf{w}_2 \mathbf{h}_2^T + \mathbf{w}_3 \mathbf{h}_3^T$$

$$|\hat{\mathbf{X}}_s| \approx \mathbf{w}_1 \mathbf{h}_1^T$$

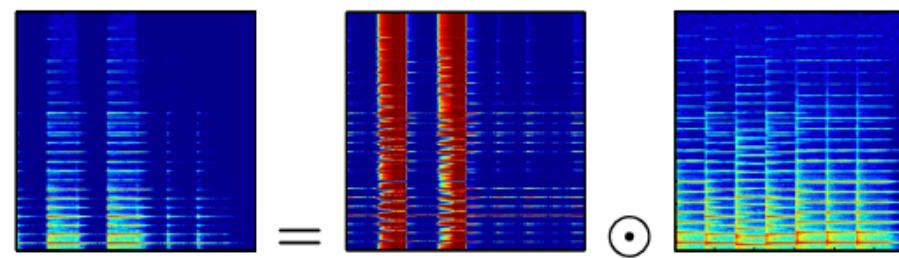
# Reconstruction with masking

- ① Compute filter  $\mathbf{M}_s = \frac{\mathbf{w}_1 \mathbf{h}_1^T}{\sum_{i=1}^K \mathbf{w}_i \mathbf{h}_i^T}$ , with  $\alpha = 1$

$$\mathbf{M}_s = \frac{(\mathbf{W}_s \mathbf{H}_s)^\alpha}{\sum_{i=1}^K (\mathbf{W}_i \mathbf{H}_i)^\alpha} = \frac{|\hat{\mathbf{X}}_s|^\alpha}{\sum_{i=1}^K |\hat{\mathbf{X}}_i|^\alpha}$$



- ② Multiply with  $|\hat{\mathbf{X}}_s| = \mathbf{M}_s \odot |\mathbf{X}|$



# Spectrum to $x[n]$

---

- Inverse DFT to get  $x[n]$ 
  - Use  $|\hat{X}_s[k]|$  and  $\angle X[k]$

# Source Separation

---

- Train  $W_1$  from source 1
- Train  $W_2$  from source 2
- Decompose mixture as  $V = [W_1 \ W_2] \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}$

# References: optional, for advanced study

---

- Smaragdis, P., & Brown, J. C. (2003, October). Non-negative matrix factorization for polyphonic music transcription. In 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No. 03TH8684) (pp. 177-180). IEEE.
- Smaragdis, P., Raj, B., & Shashanka, M. (2011). Missing data imputation for time-frequency representations of audio signals. *Journal of signal processing systems*, 65(3), 361-370.
- Wang, Y. X., & Zhang, Y. J. (2011, September). Image inpainting via weighted sparse non-negative matrix factorization. In 2011 18th IEEE International Conference on Image Processing (pp. 3409-3412). IEEE.
- Benzi, K., Kalofolias, V., Bresson, X., & Vandergheynst, P. (2016, March). Song recommendation with non-negative matrix factorization and graph total variation. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2439-2443). Ieee.

# Supervised Dimensionality Reduction

Reference: PRML Section 4.1.4

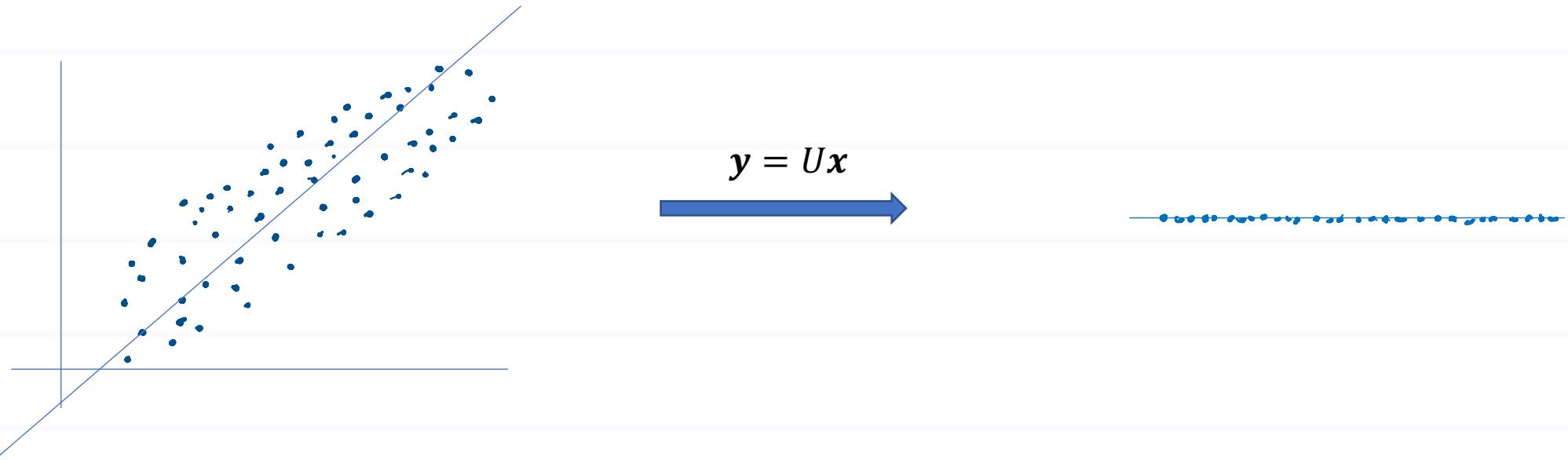
# Linear Transformation

---

- PCA:
  - Find eigen vectors and project along the eigenvector(s) with maximum eigenvalues
  - $y = Ux$

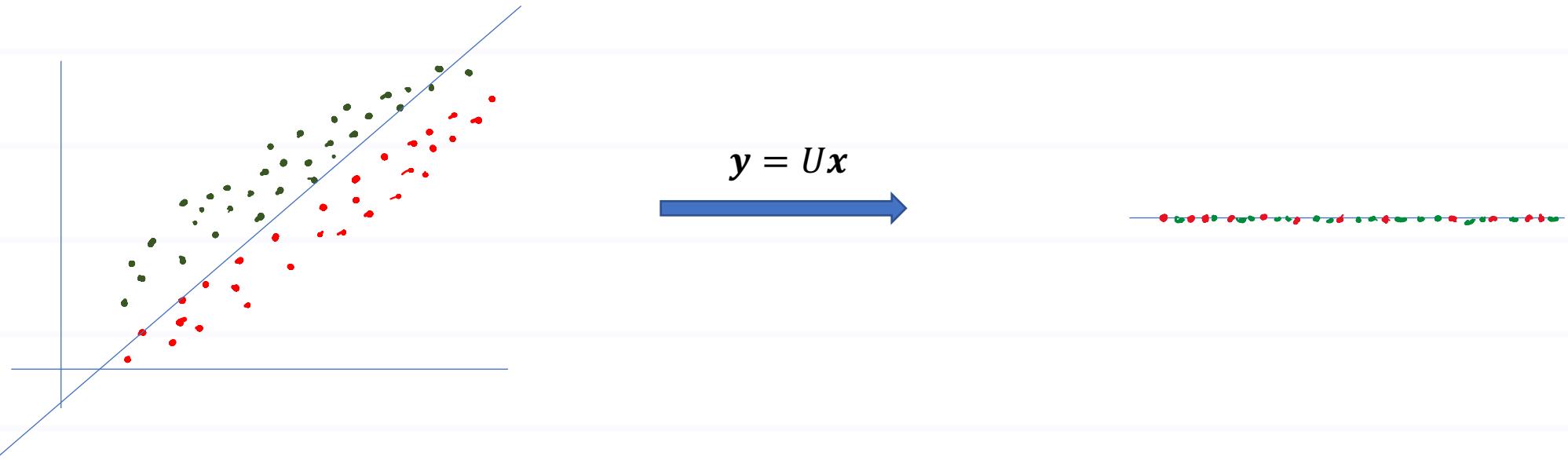
# PCA for dim. red.

---



# PCA for dim. red.

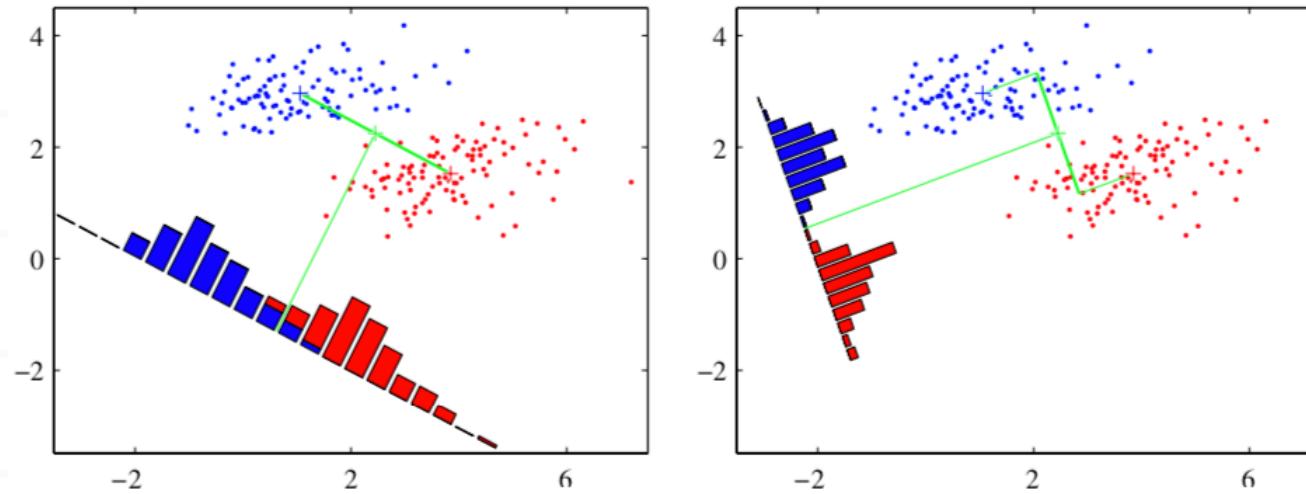
---



# How to find a good direction

---

- $y = wx$



- We should **maximize** the separation between means, but **minimize** the intra-class variance

# $y = \mathbf{w}^T \mathbf{x}$ : how to find $\mathbf{w}$

- Mean of each class in  $\mathbf{x}$

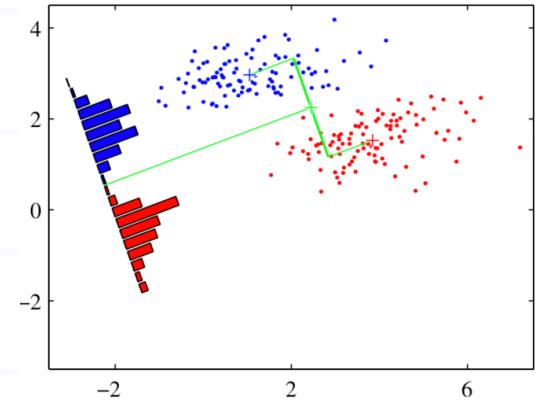
$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n$$

- Mean of each class in  $\mathbf{y}$

$$m_k = \mathbf{w}^T \mathbf{m}_k \quad k \in \{1, 2\}$$

- Intra-class variance:

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$$



# $y = \mathbf{w}^T \mathbf{x}$ : how to find $\mathbf{w}$

---

- Thus, find  $\mathbf{w}$  which maximizes

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

$$J(\mathbf{w}) = \frac{(\mathbf{w}^T \mathbf{m}_2 - \mathbf{w}^T \mathbf{m}_1) (\mathbf{w}^T \mathbf{m}_2 - \mathbf{w}^T \mathbf{m}_1)^T}{\sum_k \sum_{n \in C_k} (\mathbf{w}^T \mathbf{x}_n - \mathbf{w}^T \mathbf{m}_k) (\mathbf{w}^T \mathbf{x}_n - \mathbf{w}^T \mathbf{m}_k)^T} = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

Magnitude of  $\mathbf{w}$  is not important; it could be set to 1

where,  $S_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$

$$S_W = \sum_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

Put

$$\frac{\partial J(\omega)}{\partial \omega} = 0 \Rightarrow \underbrace{(\omega^\top S_B \omega)}_{\text{scalar}} S_W \omega - \underbrace{(\omega^\top S_W \omega)}_{\text{scalar}} S_B \omega = 0$$

$$S_W \omega = (\mu_2 - \mu_1) \underbrace{(\mu_2 - \mu_1)^\top \omega}_{\text{scalar}} \times \text{scalar}$$

$$\Rightarrow \omega = S_W^{-1} (\mu_2 - \mu_1)$$

This method is called **Fisher's Linear Discriminant**

