Source Separation Tutorial Mini-Series II: Introduction to Non-Negative Matrix Factorization

Nicholas Bryan
Dennis Sun

Center for Computer Research in Music and Acoustics,
Stanford University

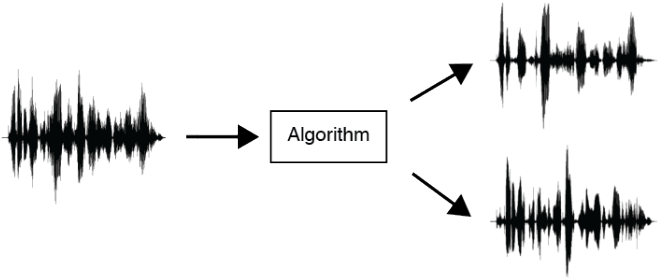
DSP Seminar
April 9th, 2013

Roadmap of Talk

1. Motivation
2. Current Approaches
3. Non-Negative Matrix Factorization (NMF)
4. Source Separation via NMF
5. Algorithms for NMF
6. Matlab Code

Roadmap of Talk

General Idea



Music Remixing and Content Creation

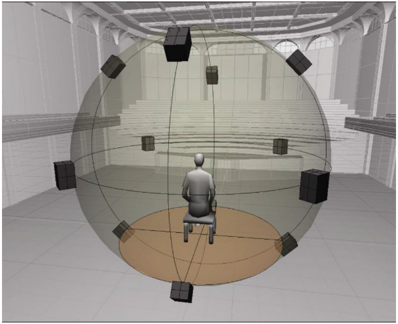
Music remixing and content creation

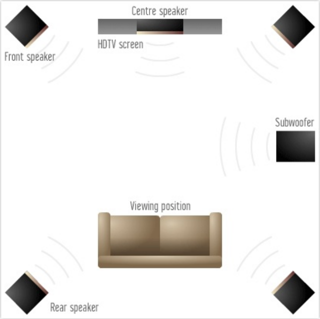
Audio Post-Production and Remastering

Audio post-production and remastering

Spatial Audio and Upmixing

Spatial audio and upmixing





Denoising

Denoising

- Separate noise speech
- Remove background music from music
- Remove bleed from other instruments

Roadmap of Talk

Current Approaches I

- Microphone Arrays
  - Beamforming to “listen” in a particular direction [BCH08]
  - Requires multiple microphones
- Adaptive Signal Processing
  - Self-adjusting filter to remove an unwanted signal [WS85]
  - Requires knowing the interfering signal
- Independent Component Analysis
  - Leverages statistical independence between signals [HO00]
  - Requires N recordings to separate N sources

Current Approaches I

- Microphone Arrays
  - Beamforming to “listen” in a particular direction [BCH08]
  - Requires multiple microphones
- Adaptive Signal Processing
  - Self-adjusting filter to remove an unwanted signal [WS85]
  - Requires knowing the interfering signal
- Independent Component Analysis
  - Leverages statistical independence between signals [HO00]
  - Requires N recordings to separate N sources

Current Approaches I

- Microphone Arrays
  - Beamforming to “listen” in a particular direction [BCH08]
  - Requires multiple microphones
- Adaptive Signal Processing
  - Self-adjusting filter to remove an unwanted signal [WS85]
  - Requires knowing the interfering signal
- Independent Component Analysis
  - Leverages statistical independence between signals [HO00]
  - Requires N recordings to separate N sources

Current Approaches II

- Computational Auditory Scene Analysis
  - Leverages knowledge of auditory system [WB06]
  - Still requires some other underlying algorithm
- Sinusoidal Modeling
  - Decomposes sound into sinusoidal peak tracks [Smi11, Wan94]
  - Problem in assigning sound source to peak tracks
- Classical Denoising and Enhancement
  - Wiener filtering, spectral subtraction, MMSE STSA (Talk 1)
  - Difficulty with time varying noise

Current Approaches II

- Computational Auditory Scene Analysis
  - Leverages knowledge of auditory system [WB06]
  - Still requires some other underlying algorithm
- Sinusoidal Modeling
  - Decomposes sound into sinusoidal peak tracks [Smi11, Wan94]
  - Problem in assigning sound source to peak tracks
- Classical Denoising and Enhancement
  - Wiener filtering, spectral subtraction, MMSE STSA (Talk 1)
  - Difficulty with time varying noise

Current Approaches II

- Computational Auditory Scene Analysis
  - Leverages knowledge of auditory system [WB06]
  - Still requires some other underlying algorithm
- Sinusoidal Modeling
  - Decomposes sound into sinusoidal peak tracks [Smi11, Wan94]
  - Problem in assigning sound source to peak tracks
- Classical Denoising and Enhancement
  - Wiener filtering, spectral subtraction, MMSE STSA (Talk 1)
  - Difficulty with time varying noise

Current Approaches III

- Non-Negative Matrix Factorization & Probabilistic Models
  - Popular technique for processing audio, image, text, etc.
  - Models spectrogram data as mixture of prototypical spectra
  - Relatively compact and easy to code algorithms
  - Amenable to machine learning
  - In many cases, works surprisingly well
  - The topic of today's discussion!

Roadmap of Talk

1. Motivation
2. Current Approaches
3. Non-Negative Matrix Factorization (NMF)
4. Source Separation via NMF
5. Algorithms for NMF
6. Matlab Code

Matrix Factorization

- Decompose a matrix as a product of two or more matrices

$$\mathbf{A} = \mathbf{B}\,\mathbf{C} \qquad \mathbf{A} \approx \mathbf{B}\,\mathbf{C}$$

$$\mathbf{D} = \mathbf{E}\,\mathbf{F}\,\mathbf{G} \qquad \mathbf{D} \approx \mathbf{E}\,\mathbf{F}\,\mathbf{G}$$

- Matrices have special properties depending on factorization
- Example factorizations:
  - Singular Value Decomposition (SVD)
  - Eigenvalue Decomposition
  - QR Decomposition (QR)
  - Lower Upper Decomposition (LU)
  - Non-Negative Matrix Factorization

Matrix Factorization

- Decompose a matrix as a product of two or more matrices

$$\mathbf{A} = \mathbf{B}\,\mathbf{C} \qquad \mathbf{A} \approx \mathbf{B}\,\mathbf{C}$$

$$\mathbf{D} = \mathbf{E}\,\mathbf{F}\,\mathbf{G} \qquad \mathbf{D} \approx \mathbf{E}\,\mathbf{F}\,\mathbf{G}$$

- Matrices have special properties depending on factorization
- Example factorizations:
  - Singular Value Decomposition (SVD)
  - Eigenvalue Decomposition
  - QR Decomposition (QR)
  - Lower Upper Decomposition (LU)
  - Non-Negative Matrix Factorization

Non-Negative Matrix Factorization

$$\overset{\text{Data}}{\left[\quad \mathbf{V} \quad\right]} \approx \overset{\text{Basis Vectors}}{\left[\quad \mathbf{W} \quad\right]} \overset{\text{Weights}}{\left[\quad \mathbf{H} \quad\right]}$$

- A matrix factorization where everything is non-negative
- $\mathbf{V} \in \mathrm{R}_{+}^{F \times T}$ - original non-negative data
- $\mathbf{W} \in \mathrm{R}_{+}^{F \times K}$ - matrix of basis vectors, dictionary elements
- $\mathbf{H} \in \mathrm{R}_{+}^{K \times T}$ - matrix of activations, weights, or gains
- $K < F < T$ (typically)
  - A compressed representation of the data
  - A low-rank approximation to $\mathbf{V}$

Non-Negative Matrix Factorization

$$\overset{\text{Data}}{\begin{bmatrix} & & \mathbf{V} & & \end{bmatrix}} \approx \overset{\text{Basis Vectors}}{\begin{bmatrix} & \mathbf{W} & \end{bmatrix}} \overset{\text{Weights}}{\begin{bmatrix} & & \mathbf{H} & & \end{bmatrix}}$$

- A matrix factorization where everything is non-negative
- $\mathbf{V} \in \mathrm{R}_+^{F \times T}$ - original non-negative data
- $\mathbf{W} \in \mathrm{R}_+^{F \times K}$ - matrix of basis vectors, dictionary elements
- $\mathbf{H} \in \mathrm{R}_+^{K \times T}$ - matrix of activations, weights, or gains
- $K < F < T$ (typically)
    - A compressed representation of the data
    - A low-rank approximation to $\mathbf{V}$

Interpretation of $\mathbf{V}$

$$\overset{\text{Data}}{\begin{bmatrix} & & \\ & \mathbf{V} & \\ & & \end{bmatrix}} \approx \overset{\text{Basis Vectors}}{\begin{bmatrix} & \\ \mathbf{W} \\ & \end{bmatrix}} \overset{\text{Weights}}{\begin{bmatrix} & & \\ & \mathbf{H} & \\ & & \end{bmatrix}}$$

- $\mathbf{V} \in \mathrm{R}_+^{F \times T}$ - original non-negative data
  - Each column is an F-dimensional data sample
  - Each row represents a data feature
  - We will use audio spectrogram data as $\mathbf{V}$

Interpretation of $\mathbf{W}$

$$\underset{\text{Data}}{\begin{bmatrix} & & \\ & \mathbf{V} & \\ & & \end{bmatrix}} \approx \underset{\text{Basis Vectors}}{\begin{bmatrix} & \mathbf{W} & \end{bmatrix}} \underset{\text{Weights}}{\begin{bmatrix} & & \\ & \mathbf{H} & \\ & & \end{bmatrix}}$$

- $\mathbf{W} \in \mathrm{R}_{+}^{F \times K}$ - matrix of basis vectors, dictionary elements
  - A single column is referred to as a basis vector
  - Not orthonormal, but commonly normalized to one

Interpretation of $\mathbf{H}$

$$\overset{\text{Data}}{\begin{bmatrix} \quad & \mathbf{V} & \quad \end{bmatrix}} \approx \overset{\text{Basis Vectors}}{\begin{bmatrix} \mathbf{W} \end{bmatrix}} \overset{\text{Weights}}{\begin{bmatrix} \quad & \mathbf{H} & \quad \end{bmatrix}}$$

- $\mathbf{H} \in \mathrm{R}_{+}^{K \times T}$ - matrix of activations, weights, or gains
  - A row represents the gain of corresponding basis vector
  - Not orthonormal, but commonly normalized to one

NMF With Spectrogram Data

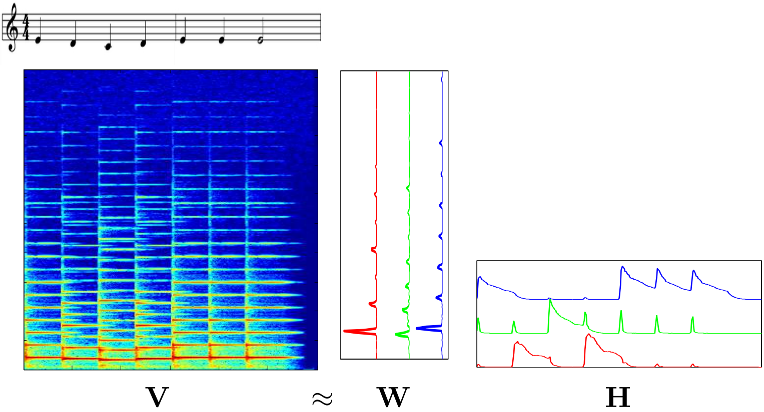


Figure : NMF of *Mary Had a Little Lamb* with $K = 3$ play stop

- The basis vectors capture prototypical spectra [SB03]
- The weights capture the gain of the basis vectors

NMF With Spectrogram Data

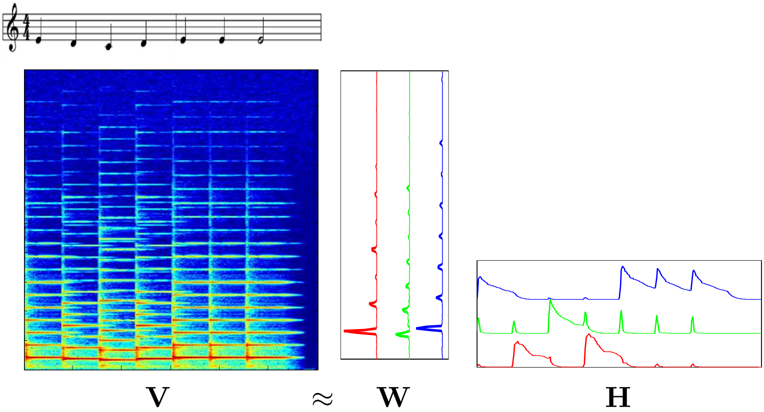
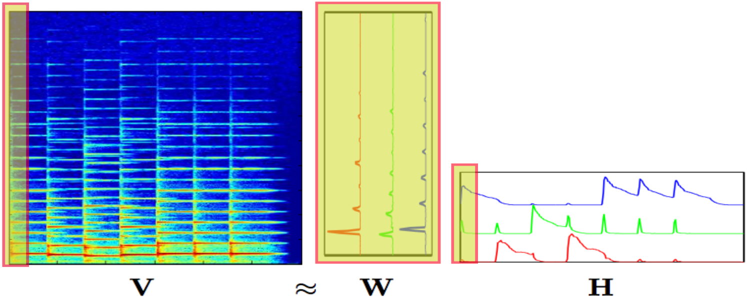


Figure : NMF of *Mary Had a Little Lamb* with $K = 3$ play stop

- The basis vectors capture prototypical spectra [SB03]
- The weights capture the gain of the basis vectors

Factorization Interpretation I

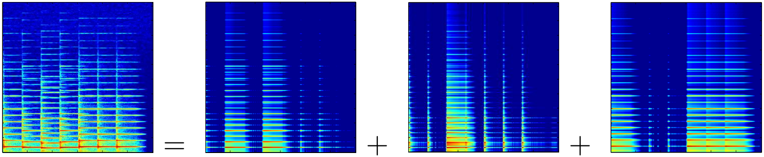
Columns of $\mathbf{V} \approx$ as a weighted sum (mixture) of basis vectors



$$\begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_T \\ | & | & & | \end{bmatrix} \approx \begin{bmatrix} \sum_{j=1}^{K} \mathbf{H}_{j1}\, \mathbf{w}_j & \sum_{j=1}^{K} \mathbf{H}_{j2}\, \mathbf{w}_j & \dots & \sum_{j=1}^{K} \mathbf{H}_{jT}\, \mathbf{w}_j \end{bmatrix}$$

Factorization Interpretation II

$\mathbf{V}$ is approximated as sum of matrix "layers"



$$\begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_T \\ | & | & & | \end{bmatrix} \approx \begin{bmatrix} | & | & & | \\ \mathbf{w}_1 & \mathbf{w}_2 & \ldots & \mathbf{w}_K \\ | & | & & | \end{bmatrix} \begin{bmatrix} — & \mathbf{h}_1^{\mathrm{T}} & — \\ — & \mathbf{h}_2^{\mathrm{T}} & — \\ & \vdots & \\ — & \mathbf{h}_K^{\mathrm{T}} & — \end{bmatrix}$$

$$\mathbf{V} \approx \mathbf{w}_1\,\mathbf{h}_1^{\mathrm{T}} + \mathbf{w}_2\,\mathbf{h}_2^{\mathrm{T}} + \ldots + \mathbf{w}_K\,\mathbf{h}_K^{\mathrm{T}}$$

Questions

- How do we use $\mathbf{W}$ and $\mathbf{H}$ to perform separation?

- How do we solve for $\mathbf{W}$ and $\mathbf{H}$, given a known $\mathbf{V}$?

Questions

- How do we use $\mathbf{W}$ and $\mathbf{H}$ to perform separation?

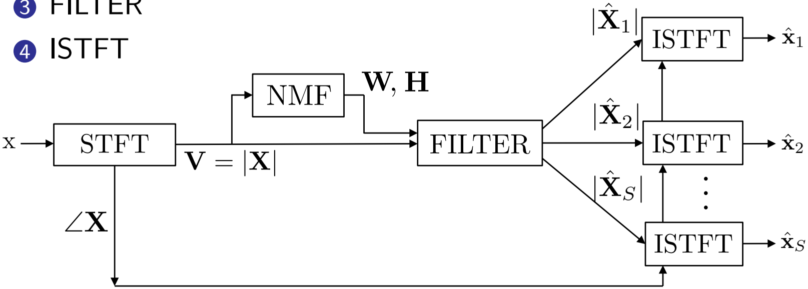
- How do we solve for $\mathbf{W}$ and $\mathbf{H}$, given a known $\mathbf{V}$?

Roadmap of Talk

1. Motivation
2. Current Approaches
3. Non-Negative Matrix Factorization (NMF)
4. Source Separation via NMF
5. Algorithms for NMF
6. Matlab Code

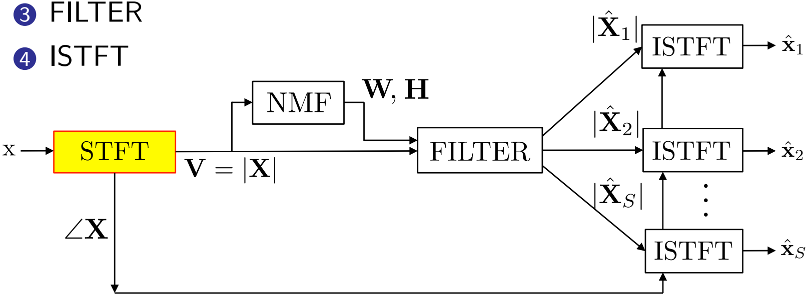
General Separation Pipeline

1. STFT
2. NMF
3. FILTER
4. ISTFT

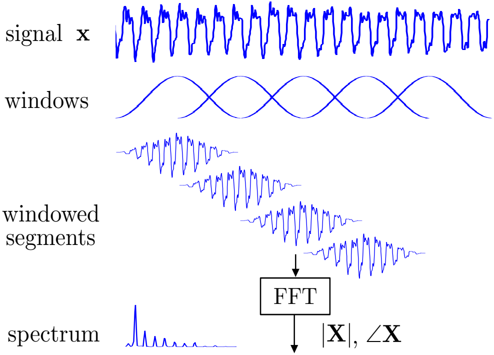


General Separation Pipeline

1. STFT
2. NMF
3. FILTER
4. ISTFT



Short-Time Fourier Transform (STFT)



- Inputs time domain signal $\mathbf{x}$
- Outputs magnitude $|\mathbf{X}|$ and phase $\angle\mathbf{X}$ matrices

Short-Time Fourier Transform (STFT)

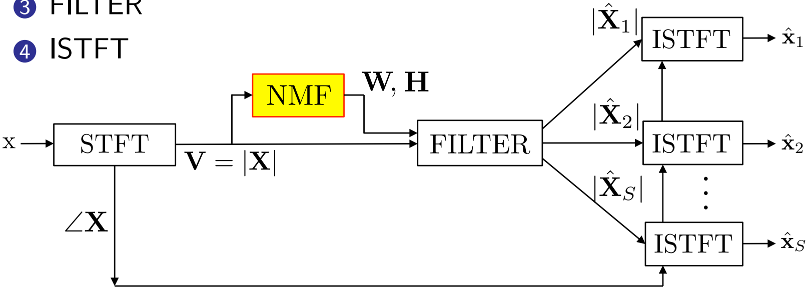
$$X_m(\omega_k) \;\; = \;\; e^{-j\omega_k mR} \sum_{n=-N/2}^{N/2-1} x(n+mR)w(n)e^{-j\omega_k n}$$

$$\begin{aligned}
x(n) &= \text{input signal at time } n \\
w(n) &= \text{length } M \text{ window function (e.g. Hann, etc.)} \\
N &= \text{DFT size, in samples} \\
R &= \text{hop size, in samples, between successive DFT} \\
M &= \text{window size, in samples} \\
w_k &= 2\pi k/N,\; k = 0, 1, 2, \ldots, N-1
\end{aligned}$$

- Choose window, window size, DFT size, and hop size
- Must maintain constant overlap-add COLA(R) [Smi11]

General Separation Pipeline

1. STFT
2. NMF
3. FILTER
4. ISTFT

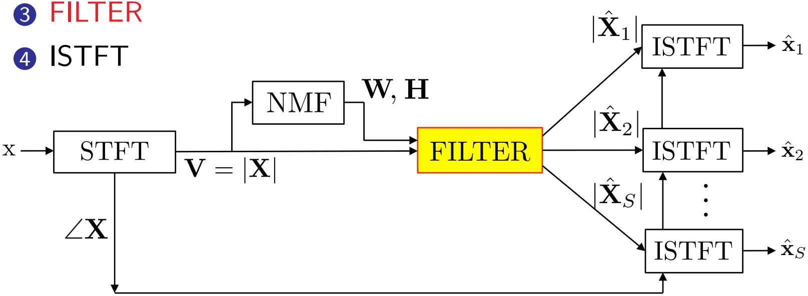


Non-Negative Matrix Factorization

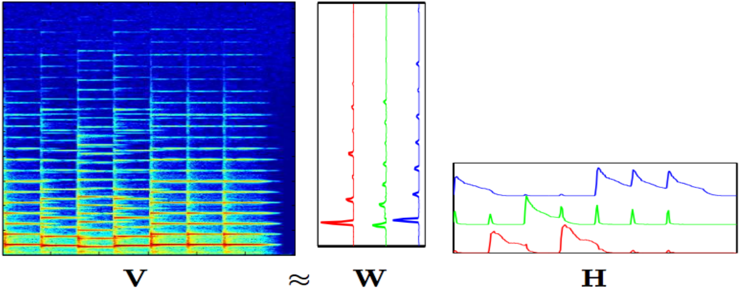
- Inputs $|\mathbf{X}|$, outputs $\mathbf{W}$ and $\mathbf{H}$
- Algorithm to be discussed

General Separation Pipeline

1. STFT
2. NMF
3. FILTER
4. ISTFT



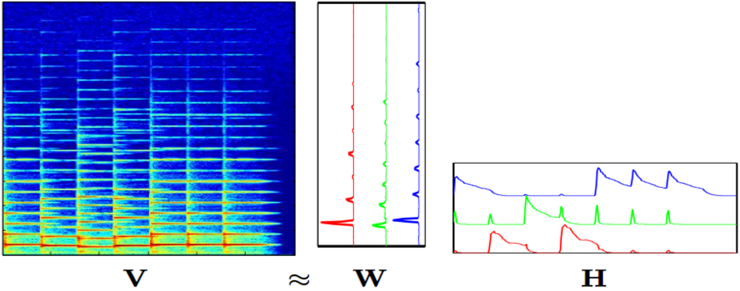
Source Synthesis I



- Choose a subset of basis vectors $\mathbf{W}_s$ and activations $\mathbf{H}_s$ to reconstruct source s
- Estimate the source s magnitude:

$$|\hat{\mathbf{X}}_s| = \mathbf{W}_s\,\mathbf{H}_s = \sum_{i \in s} (\mathbf{w}_i\,\mathbf{h}_i^{\mathrm{T}})$$

Source Synthesis I

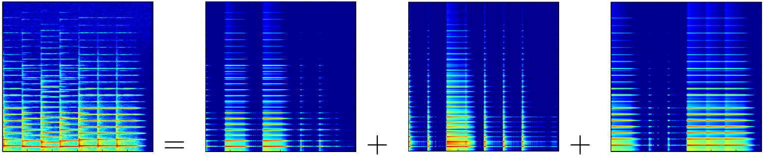


- Choose a subset of basis vectors $\mathbf{W}_s$ and activations $\mathbf{H}_s$ to reconstruct source s
- Estimate the source s magnitude:

$$|\hat{\mathbf{X}}_s| = \mathbf{W}_s\,\mathbf{H}_s = \sum_{i \in s} (\mathbf{w}_i\,\mathbf{h}_i^{\mathrm{T}})$$

Source Synthesis II

Example 1: “D” pitches as a single source

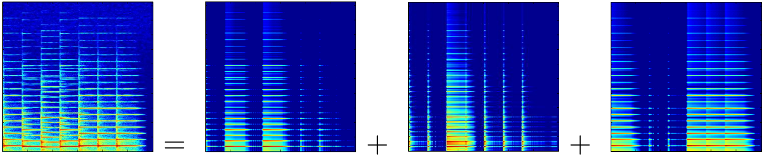


$$\mathbf{V} \approx \mathbf{w}_1\,\mathbf{h}_1^{\mathrm{T}} + \mathbf{w}_2\,\mathbf{h}_2^{\mathrm{T}} + \mathbf{w}_3\,\mathbf{h}_3^{\mathrm{T}}$$

- $|\hat{\mathbf{X}}_s| \approx \mathbf{w}_1\,\mathbf{h}_1^{\mathrm{T}}$
- Use one basis vector to reconstruct a source

Source Synthesis III

Example 2: “D” and “E” pitches as a source



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

- $|\hat{\mathbf{X}}_s| \approx \mathbf{w}_1\,\mathbf{h}_1^{\mathrm{T}} + \mathbf{w}_2\,\mathbf{h}_2^{\mathrm{T}}$
- Use two (or more) basis vector to reconstruct a source

Source Filtering I

Alternatively, we can estimate $|\hat{\mathbf{X}}_s|$ by filtering $|\,\mathbf{X}\,|$ via:

1. Generate a filter $\mathbf{M}_s,\ \forall s$

$$\mathbf{M}_s = \frac{(\mathbf{W}_s\,\mathbf{H}_s)^\alpha}{\sum_{i=1}^{K}(\mathbf{W}_i\,\mathbf{H}_i)^\alpha} = \frac{|\hat{\mathbf{X}}_s|^\alpha}{\sum_{i=1}^{K}|\hat{\mathbf{X}}_i|^\alpha} = \frac{\sum_{i\in s}(\mathbf{w}_i\,\mathbf{h}_i^{\mathrm{T}})^\alpha}{\sum_{i=1}^{K}(\mathbf{w}_i\,\mathbf{h}_i^{\mathrm{T}})^\alpha}$$

   where $\alpha \in \mathrm{R}_+$ is typically set to one or two.

2. Estimate the source s magnitude $|\,\mathbf{X}_s\,|$

$$|\hat{\mathbf{X}}_s| = \mathbf{M}_s \odot |\,\mathbf{X}\,|$$

   where $\odot$ is an element-wise multiplication

Source Filtering I

Alternatively, we can estimate $|\hat{\mathbf{X}}_s|$ by filtering $|\,\mathbf{X}\,|$ via:

1. Generate a filter $\mathbf{M}_s,\ \forall s$

$$\mathbf{M}_s = \frac{(\mathbf{W}_s\,\mathbf{H}_s)^\alpha}{\sum_{i=1}^{K}(\mathbf{W}_i\,\mathbf{H}_i)^\alpha} = \frac{|\hat{\mathbf{X}}_s|^\alpha}{\sum_{i=1}^{K}|\hat{\mathbf{X}}_i|^\alpha} = \frac{\sum_{i\in s}(\mathbf{w}_i\,\mathbf{h}_i^{\mathrm{T}})^\alpha}{\sum_{i=1}^{K}(\mathbf{w}_i\,\mathbf{h}_i^{\mathrm{T}})^\alpha}$$

   where $\alpha \in \mathrm{R}_+$ is typically set to one or two.

2. Estimate the source s magnitude $|\,\mathbf{X}_s\,|$

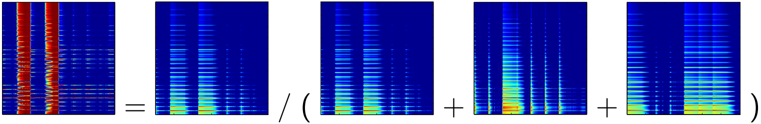
$$|\hat{\mathbf{X}}_s| = \mathbf{M}_s \odot |\,\mathbf{X}\,|$$

   where $\odot$ is an element-wise multiplication

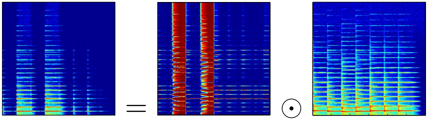
Source Filtering II

Example: Choose "D" pitches as a single source w/one basis vector

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



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

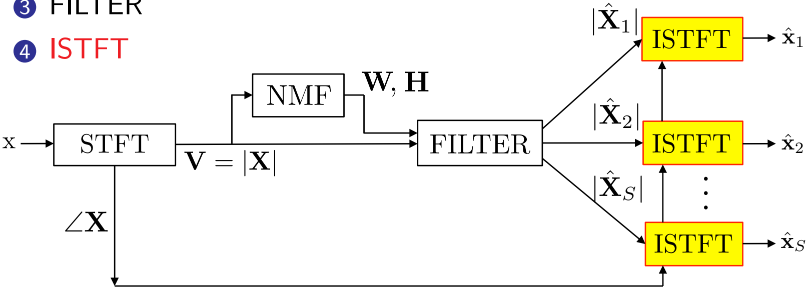


Source Filtering III

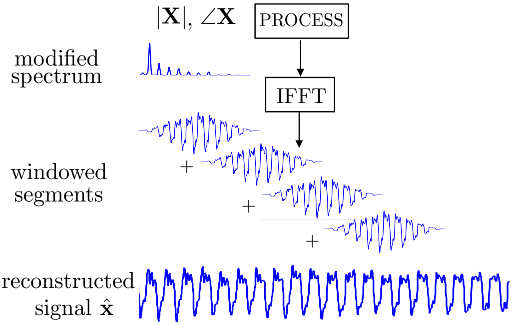
- The filter $\mathbf{M}$ is referred to as a *masking filter* or *soft mask*
- Tends to perform better than the reconstruction method
- Similar to Wiener filtering discussed in Talk 1

General Separation Pipeline

1. STFT
2. NMF
3. FILTER
4. ISTFT



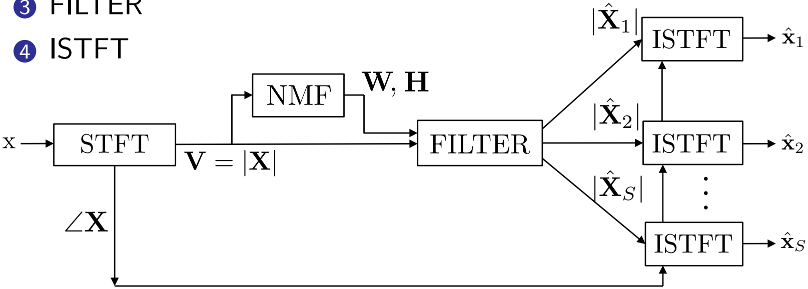
Inverse Short-Time Fourier Transform (ISTFT)



- Inputs $|\mathbf{X}|$ and phase $\angle\mathbf{X}$ matrices
- Outputs time domain signal $\mathbf{x}$

General Separation Pipeline

1. STFT
2. NMF
3. FILTER
4. ISTFT



Roadmap of Talk

1. Motivation
2. Current Approaches
3. Non-Negative Matrix Factorization (NMF)
4. Source Separation via NMF
5. Algorithms for NMF
6. Matlab Code

Algorithms for NMF

- Question: How do we solve for $\mathbf{W}$ and $\mathbf{H}$, given a known $\mathbf{V}$?
- Answer: Frame as optimization problem

$$\underset{\mathbf{W},\mathbf{H}\geq 0}{\text{minimize}} \;\; D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H})$$

where D is a measure of "divergence".

Algorithms for NMF

- Question: How do we solve for $\mathbf{W}$ and $\mathbf{H}$, given a known $\mathbf{V}$?
- Answer: Frame as optimization problem

$$\underset{\mathbf{W},\mathbf{H}\geq 0}{\text{minimize}} \quad D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H})$$

where D is a measure of "divergence".

Algorithms for NMF

Some choices for D:

- **Euclidean**: $D(\mathbf{V}\,||\hat{\mathbf{V}}) = \sum_{i,j} (\mathbf{V}_{ij} - \hat{\mathbf{V}}_{ij})^2$
- **Kullback-Leibler**:
  $$D(\mathbf{V}\,||\hat{\mathbf{V}}) = \sum_{i,j} \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{\hat{\mathbf{V}}_{ij}} - \mathbf{V}_{ij} + \hat{\mathbf{V}}_{ij} \right)$$

We will focus on KL divergence in today's lecture.

Algorithms for NMF

Some choices for D:

- **Euclidean**: $D(\mathbf{V}\,||\hat{\mathbf{V}}) = \sum_{i,j}(\mathbf{V}_{ij} - \hat{\mathbf{V}}_{ij})^2$
- **Kullback-Leibler**:
$$D(\mathbf{V}\,||\hat{\mathbf{V}}) = \sum_{i,j}\left(\mathbf{V}_{ij}\log\frac{\mathbf{V}_{ij}}{\hat{\mathbf{V}}_{ij}} - \mathbf{V}_{ij} + \hat{\mathbf{V}}_{ij}\right)$$

We will focus on KL divergence in today's lecture.

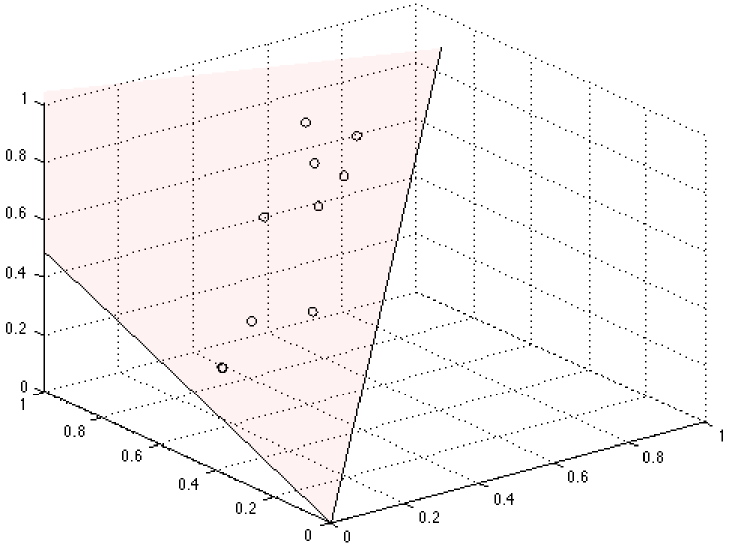
Algorithms for NMF

Some choices for D:

- **Euclidean**: $D(\mathbf{V}\,||\hat{\mathbf{V}}) = \sum_{i,j} (\mathbf{V}_{ij} - \hat{\mathbf{V}}_{ij})^2$
- **Kullback-Leibler**:
  $$D(\mathbf{V}\,||\hat{\mathbf{V}}) = \sum_{i,j} \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{\hat{\mathbf{V}}_{ij}} - \mathbf{V}_{ij} + \hat{\mathbf{V}}_{ij} \right)$$

We will focus on KL divergence in today's lecture.

Geometric View of NMF



Algorithms for NMF

How does one solve

$$\underset{\mathbf{W},\mathbf{H}\geq 0}{\text{minimize}} \quad \sum_{i,j} \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{(\mathbf{W}\,\mathbf{H})_{ij}} - \mathbf{V}_{ij} + (\mathbf{W}\,\mathbf{H})_{ij} \right)?$$

Tricks of the trade for minimizing a function $f(\mathbf{x})$.

- closed-form solutions: solve $\nabla f(\mathbf{x}) = 0$.
- gradient descent: iteratively move in steepest descent dir.

$$\mathbf{x}^{(\ell+1)} \leftarrow \mathbf{x}^{(\ell)} - \eta \nabla f(\mathbf{x}^{(\ell)}).$$

- Newton's method: iteratively minimize quadratic approx.

$$\mathbf{x}^{(\ell+1)} \leftarrow \underset{\mathbf{x}}{\text{argmin}} \; f(\mathbf{x}^{(\ell)}) + \nabla f(\mathbf{x}^{(\ell)})^T (\mathbf{x} - \mathbf{x}^{(\ell)})$$
$$+ \frac{1}{2} (\mathbf{x} - \mathbf{x}^{(\ell)})^T \nabla^2 f(\mathbf{x}^{(\ell)}) (\mathbf{x} - \mathbf{x}^{(\ell)})$$

Algorithms for NMF

How does one solve

$$\underset{\mathbf{W},\mathbf{H}\geq 0}{\text{minimize}} \quad \sum_{i,j}\left(\mathbf{V}_{ij}\log\frac{\mathbf{V}_{ij}}{(\mathbf{W}\,\mathbf{H})_{ij}} - \mathbf{V}_{ij} + (\mathbf{W}\,\mathbf{H})_{ij}\right)?$$

Tricks of the trade for minimizing a function $f(\mathbf{x})$.

- closed-form solutions: solve $\nabla f(\mathbf{x}) = 0$.
- gradient descent: iteratively move in steepest descent dir.

$$\mathbf{x}^{(\ell+1)} \leftarrow \mathbf{x}^{(\ell)} - \eta \nabla f(\mathbf{x}^{(\ell)}).$$

- Newton's method: iteratively minimize quadratic approx.

$$\mathbf{x}^{(\ell+1)} \leftarrow \underset{\mathbf{x}}{\text{argmin}}\; f(\mathbf{x}^{(\ell)}) + \nabla f(\mathbf{x}^{(\ell)})^T(\mathbf{x}-\mathbf{x}^{(\ell)}) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^{(\ell)})^T\nabla^2 f(\mathbf{x}^{(\ell)})(\mathbf{x}-\mathbf{x}^{(\ell)})$$

Algorithms for NMF

How does one solve

$$\underset{\mathbf{W},\mathbf{H}\geq 0}{\text{minimize}} \quad \sum_{i,j} \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{(\mathbf{W}\,\mathbf{H})_{ij}} - \mathbf{V}_{ij} + (\mathbf{W}\,\mathbf{H})_{ij} \right)?$$

Tricks of the trade for minimizing a function $f(\mathbf{x})$.

- closed-form solutions: solve $\nabla f(\mathbf{x}) = 0$.
- gradient descent: iteratively move in steepest descent dir.

$$\mathbf{x}^{(\ell+1)} \leftarrow \mathbf{x}^{(\ell)} - \eta \nabla f(\mathbf{x}^{(\ell)}).$$

- Newton's method: iteratively minimize quadratic approx.

$$\mathbf{x}^{(\ell+1)} \leftarrow \underset{\mathbf{x}}{\text{argmin}} \; f(\mathbf{x}^{(\ell)}) + \nabla f(\mathbf{x}^{(\ell)})^T (\mathbf{x} - \mathbf{x}^{(\ell)}) + \frac{1}{2} (\mathbf{x} - \mathbf{x}^{(\ell)})^T \nabla^2 f(\mathbf{x}^{(\ell)}) (\mathbf{x} - \mathbf{x}^{(\ell)})$$

Algorithms for NMF

How does one solve

$$\underset{\mathbf{W},\mathbf{H}\geq 0}{\text{minimize}} \quad \sum_{i,j} \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{(\mathbf{W}\,\mathbf{H})_{ij}} - \mathbf{V}_{ij} + (\mathbf{W}\,\mathbf{H})_{ij} \right)?$$

Tricks of the trade for minimizing a function $f(\mathbf{x})$.

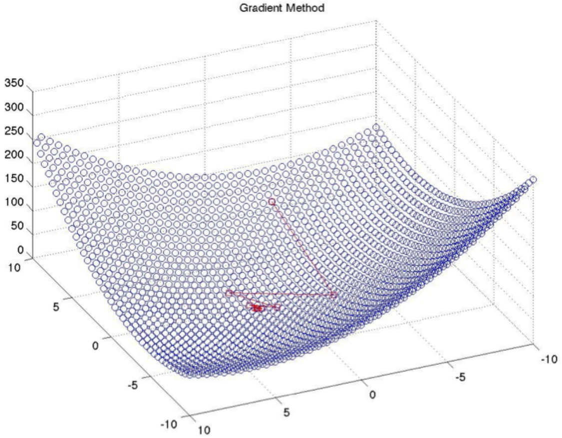
- closed-form solutions: solve $\nabla f(\mathbf{x}) = 0$.
- gradient descent: iteratively move in steepest descent dir.

$$\mathbf{x}^{(\ell+1)} \leftarrow \mathbf{x}^{(\ell)} - \eta \nabla f(\mathbf{x}^{(\ell)}).$$

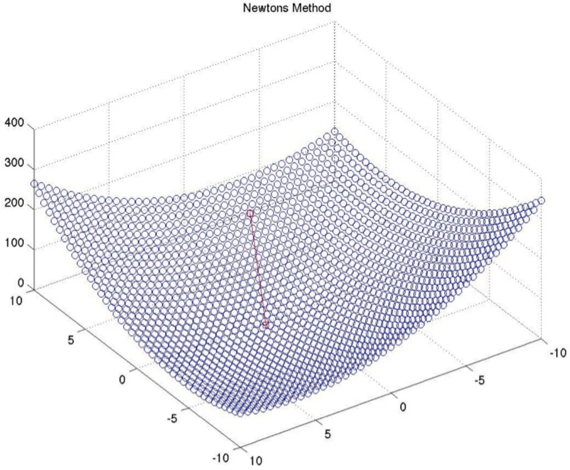
- Newton's method: iteratively minimize quadratic approx.

$$\mathbf{x}^{(\ell+1)} \leftarrow \underset{\mathbf{x}}{\text{argmin}}\ f(\mathbf{x}^{(\ell)}) + \nabla f(\mathbf{x}^{(\ell)})^T(\mathbf{x} - \mathbf{x}^{(\ell)}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(\ell)})^T \nabla^2 f(\mathbf{x}^{(\ell)})(\mathbf{x} - \mathbf{x}^{(\ell)})$$

Gradient Descent



Newton's Method



Meta Algorithms

Coordinate descent

- Instead of minimizing $f(\mathbf{x})$, minimize $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ and cycle over i.
- Useful when $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ can be minimized in closed form.

Majorization-minimization

1. Find a majorizing function g for f at current iterate $\mathbf{x}^{(\ell)}$.
   - $f(\mathbf{x}) < g(\mathbf{x}; \mathbf{x}^{(\ell)})$ for all $\mathbf{x} \neq \mathbf{x}^{(\ell)}$
   - $f(\mathbf{x}^{(\ell)}) = g(\mathbf{x}^{(\ell)}; \mathbf{x}^{(\ell)})$
2. Minimize the majorizing function to obtain $\mathbf{x}^{(\ell+1)}$.

Meta Algorithms

Coordinate descent

- Instead of minimizing $f(\mathbf{x})$, minimize $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ and cycle over i.
- Useful when $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ can be minimized in closed form.

Majorization-minimization

1. Find a majorizing function g for f at current iterate $\mathbf{x}^{(\ell)}$.
   - $f(\mathbf{x}) < g(\mathbf{x}; \mathbf{x}^{(\ell)})$ for all $\mathbf{x} \neq \mathbf{x}^{(\ell)}$
   - $f(\mathbf{x}^{(\ell)}) = g(\mathbf{x}^{(\ell)}; \mathbf{x}^{(\ell)})$
2. Minimize the majorizing function to obtain $\mathbf{x}^{(\ell+1)}$.

Meta Algorithms

Coordinate descent

- Instead of minimizing $f(\mathbf{x})$, minimize $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ and cycle over i.
- Useful when $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ can be minimized in closed form.

Majorization-minimization

1. Find a majorizing function g for f at current iterate $\mathbf{x}^{(\ell)}$.
   - $f(\mathbf{x}) < g(\mathbf{x}; \mathbf{x}^{(\ell)})$ for all $\mathbf{x} \neq \mathbf{x}^{(\ell)}$
   - $f(\mathbf{x}^{(\ell)}) = g(\mathbf{x}^{(\ell)}; \mathbf{x}^{(\ell)})$
2. Minimize the majorizing function to obtain $\mathbf{x}^{(\ell+1)}$.

Meta Algorithms

Coordinate descent

- Instead of minimizing $f(\mathbf{x})$, minimize $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ and cycle over i.
- Useful when $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ can be minimized in closed form.

Majorization-minimization

1. Find a majorizing function g for f at current iterate $\mathbf{x}^{(\ell)}$.
   - $f(\mathbf{x}) < g(\mathbf{x}; \mathbf{x}^{(\ell)})$ for all $\mathbf{x} \neq \mathbf{x}^{(\ell)}$
   - $f(\mathbf{x}^{(\ell)}) = g(\mathbf{x}^{(\ell)}; \mathbf{x}^{(\ell)})$
2. Minimize the majorizing function to obtain $\mathbf{x}^{(\ell+1)}$.

Meta Algorithms

Coordinate descent

- Instead of minimizing $f(\mathbf{x})$, minimize $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ and cycle over i.
- Useful when $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ can be minimized in closed form.

Majorization-minimization

1. Find a majorizing function g for f at current iterate $\mathbf{x}^{(\ell)}$.
   - $f(\mathbf{x}) < g(\mathbf{x}; \mathbf{x}^{(\ell)})$ for all $\mathbf{x} \neq \mathbf{x}^{(\ell)}$
   - $f(\mathbf{x}^{(\ell)}) = g(\mathbf{x}^{(\ell)}; \mathbf{x}^{(\ell)})$
2. Minimize the majorizing function to obtain $\mathbf{x}^{(\ell+1)}$.

Meta Algorithms

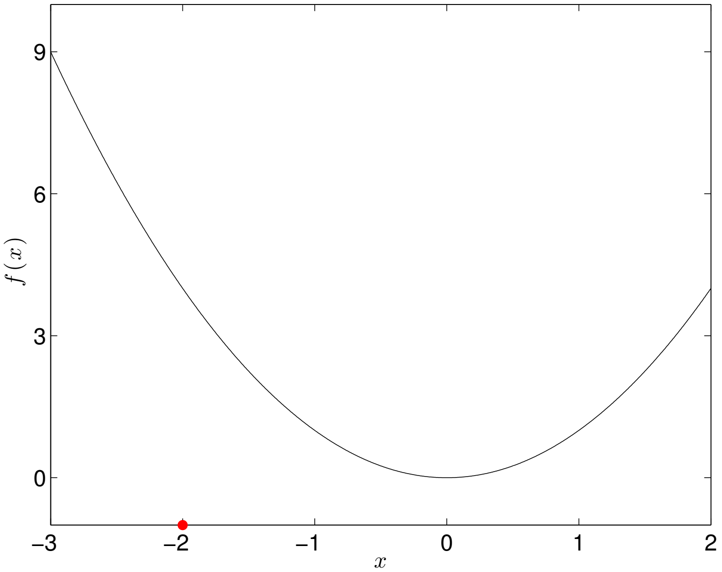
Coordinate descent

- Instead of minimizing $f(\mathbf{x})$, minimize $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ and cycle over i.
- Useful when $f(\mathbf{x}_i; \mathbf{x}_{-i}^{(\ell)})$ can be minimized in closed form.

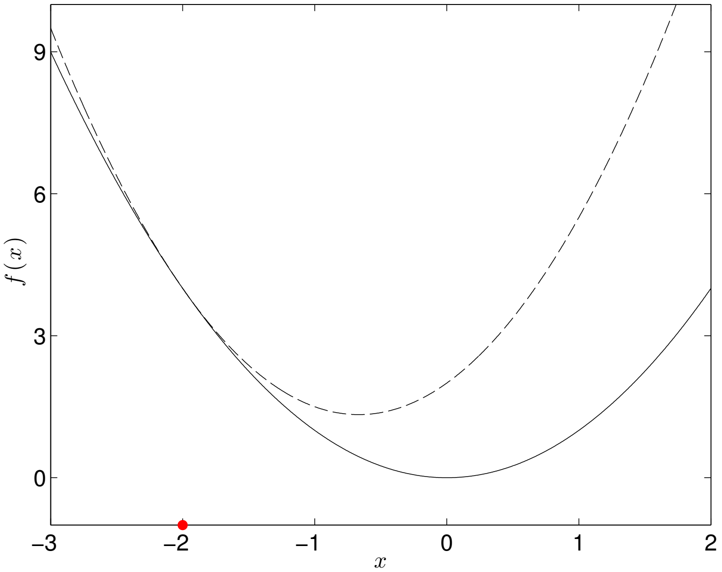
Majorization-minimization

1. Find a majorizing function g for f at current iterate $\mathbf{x}^{(\ell)}$.
   - $f(\mathbf{x}) < g(\mathbf{x}; \mathbf{x}^{(\ell)})$ for all $\mathbf{x} \neq \mathbf{x}^{(\ell)}$
   - $f(\mathbf{x}^{(\ell)}) = g(\mathbf{x}^{(\ell)}; \mathbf{x}^{(\ell)})$
2. Minimize the majorizing function to obtain $\mathbf{x}^{(\ell+1)}$.

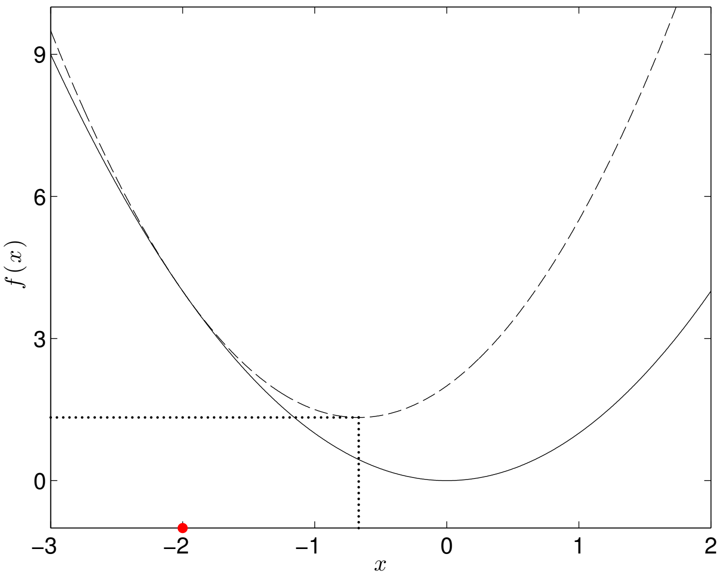
Majorization-minimization



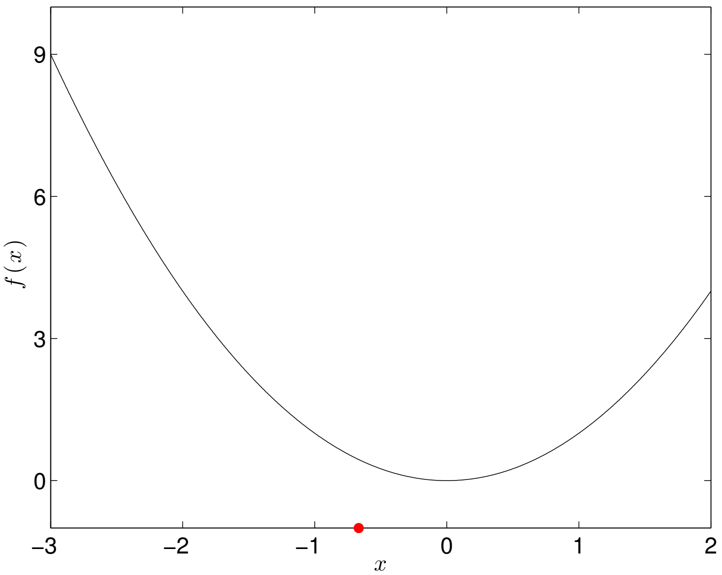
Majorization-minimization



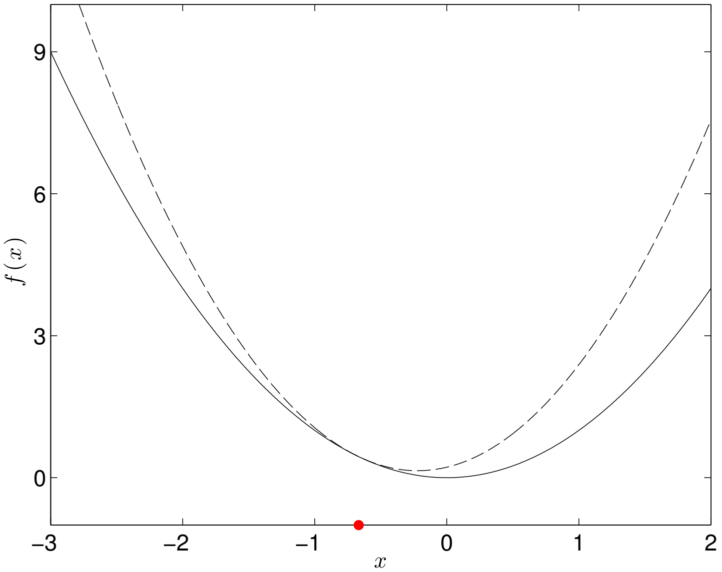
Majorization-minimization



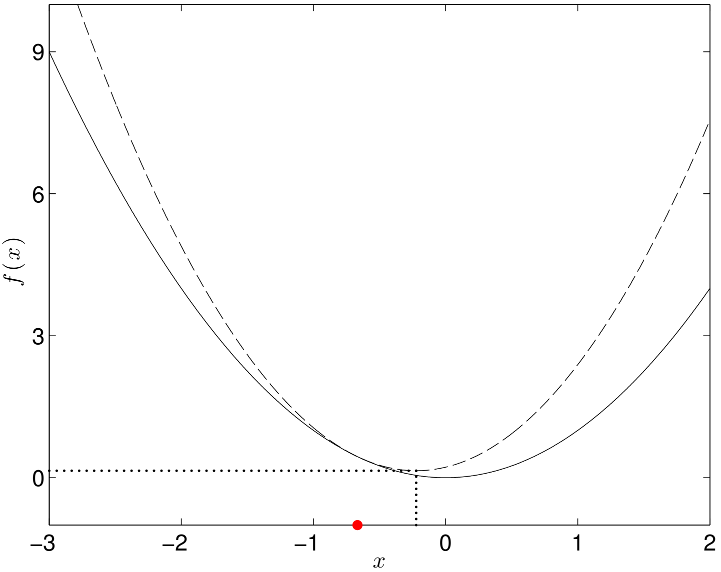
Majorization-minimization



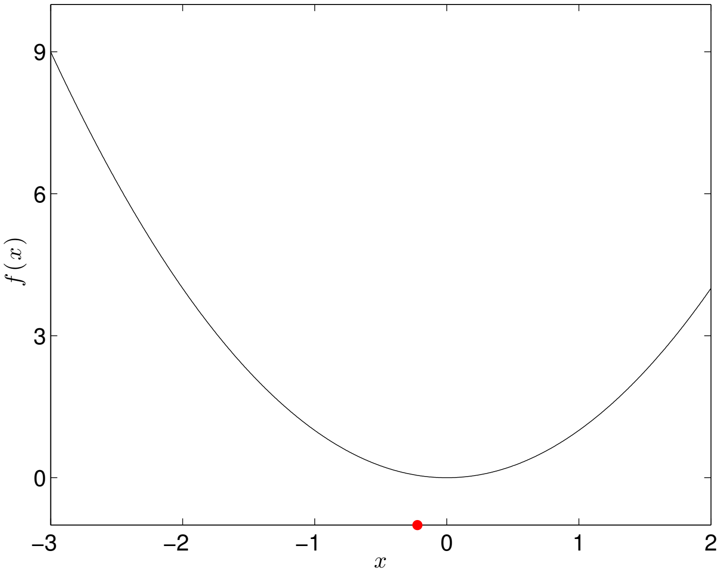
Majorization-minimization



Majorization-minimization



Majorization-minimization



Algorithms for NMF

To minimize

$$D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) = \sum_{i,j} \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{(\mathbf{W}\,\mathbf{H})_{ij}} - \mathbf{V}_{ij} + (\mathbf{W}\,\mathbf{H})_{ij} \right)$$

$$\stackrel{\text{cst.}}{=} \sum_{i,j} -\mathbf{V}_{ij} \log \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} + \sum_{i,j} \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}$$

we use **(block) coordinate descent**: optimize $\mathbf{H}$ for $\mathbf{W}$ fixed, then optimize $\mathbf{W}$ for $\mathbf{H}$ fixed (rinse and repeat).

Can we optimize this in closed form?

Algorithms for NMF

To minimize

$$D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) = \sum_{i,j} \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{(\mathbf{W}\,\mathbf{H})_{ij}} - \mathbf{V}_{ij} + (\mathbf{W}\,\mathbf{H})_{ij} \right)$$

$$\overset{\text{cst.}}{=} \sum_{i,j} - \mathbf{V}_{ij} \log \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} + \sum_{i,j} \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}$$

we use **(block) coordinate descent**: optimize $\mathbf{H}$ for $\mathbf{W}$ fixed, then optimize $\mathbf{W}$ for $\mathbf{H}$ fixed (rinse and repeat).

Can we optimize this in closed form?

Algorithms for NMF

To minimize

$$D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) = \sum_{i,j} \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{(\mathbf{W}\,\mathbf{H})_{ij}} - \mathbf{V}_{ij} + (\mathbf{W}\,\mathbf{H})_{ij} \right)$$

$$\overset{\text{cst.}}{=} \sum_{i,j} - \mathbf{V}_{ij} \log \sum_{k} \mathbf{W}_{ik}\,\mathbf{H}_{kj} + \sum_{i,j} \sum_{k} \mathbf{W}_{ik}\,\mathbf{H}_{kj}$$

we use **(block) coordinate descent**: optimize $\mathbf{H}$ for $\mathbf{W}$ fixed, then optimize $\mathbf{W}$ for $\mathbf{H}$ fixed (rinse and repeat).

Can we optimize this in closed form?

Algorithms for NMF

To minimize

$$D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) = \sum_{i,j} \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{(\mathbf{W}\,\mathbf{H})_{ij}} - \mathbf{V}_{ij} + (\mathbf{W}\,\mathbf{H})_{ij} \right)$$

$$\stackrel{\text{cst.}}{=} \sum_{i,j} - \mathbf{V}_{ij} \log \sum_{k} \mathbf{W}_{ik}\,\mathbf{H}_{kj} + \sum_{i,j} \sum_{k} \mathbf{W}_{ik}\,\mathbf{H}_{kj}$$

we use **(block) coordinate descent**: optimize $\mathbf{H}$ for $\mathbf{W}$ fixed, then optimize $\mathbf{W}$ for $\mathbf{H}$ fixed (rinse and repeat).

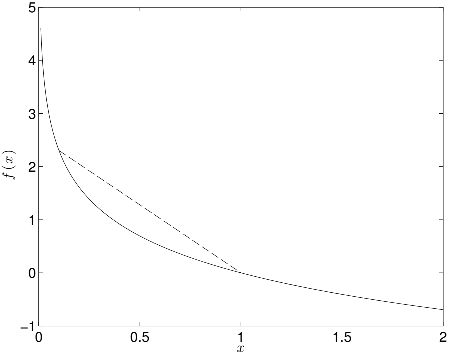
Can we optimize this in closed form?

Algorithms for NMF

$$D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) \stackrel{\text{cst.}}{=} \sum_{i,j} -\mathbf{V}_{ij} \log \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} + \sum_{i,j} \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}$$

Not quite, so let's try to majorize the function. A useful tool is **Jensen's inequality**, which says that for **convex** functions f:

$$f(\text{average}) \leq \text{average of } f$$



Algorithms for NMF

$$D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) \overset{\text{cst.}}{=} \sum_{i,j} -\mathbf{V}_{ij} \log \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} + \sum_{i,j} \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}$$

To apply Jensen's inequality, we introduce weights $\sum_k \pi_{ijk} = 1$.

$$= \sum_{i,j} \left(-\mathbf{V}_{ij} \log \sum_k \pi_{ijk} \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

$$\leq \sum_{i,j} \left(-\mathbf{V}_{ij} \sum_k \pi_{ijk} \log \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

Now this function *can* be minimized exactly!

$$\mathbf{H}^*_{kj} = \frac{\sum_i \mathbf{V}_{ij}\,\pi_{ijk}}{\sum_i \mathbf{W}_{ik}}$$

Algorithms for NMF

$$D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) \overset{\text{cst.}}{=} \sum_{i,j} -\mathbf{V}_{ij} \log \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} + \sum_{i,j} \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}$$

To apply Jensen's inequality, we introduce weights $\sum_k \pi_{ijk} = 1$.

$$= \sum_{i,j} \left(-\mathbf{V}_{ij} \log \sum_k \pi_{ijk} \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

$$\leq \sum_{i,j} \left(-\mathbf{V}_{ij} \sum_k \pi_{ijk} \log \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

Now this function *can* be minimized exactly!

$$\mathbf{H}^*_{kj} = \frac{\sum_i \mathbf{V}_{ij}\,\pi_{ijk}}{\sum_i \mathbf{W}_{ik}}$$

Algorithms for NMF

$$D(\mathbf{V}\,||\,\mathbf{W}\,\mathbf{H}) \overset{\text{cst.}}{=} \sum_{i,j} -\mathbf{V}_{ij} \log \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} + \sum_{i,j} \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}$$

To apply Jensen's inequality, we introduce weights $\sum_k \pi_{ijk} = 1$.

$$= \sum_{i,j} \left(-\mathbf{V}_{ij} \log \sum_k \pi_{ijk} \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

$$\leq \sum_{i,j} \left(-\mathbf{V}_{ij} \sum_k \pi_{ijk} \log \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

Now this function *can* be minimized exactly!

$$\mathbf{H}^*_{kj} = \frac{\sum_i \mathbf{V}_{ij}\,\pi_{ijk}}{\sum_i \mathbf{W}_{ik}}$$

Algorithms for NMF

$$D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) \overset{\text{cst.}}{=} \sum_{i,j} -\mathbf{V}_{ij} \log \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} + \sum_{i,j} \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}$$

To apply Jensen's inequality, we introduce weights $\sum_k \pi_{ijk} = 1$.

$$= \sum_{i,j} \left(-\mathbf{V}_{ij} \log \sum_k \pi_{ijk} \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

$$\leq \sum_{i,j} \left(-\mathbf{V}_{ij} \sum_k \pi_{ijk} \log \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

Now this function *can* be minimized exactly!

$$\mathbf{H}^*_{kj} = \frac{\sum_i \mathbf{V}_{ij}\,\pi_{ijk}}{\sum_i \mathbf{W}_{ik}}$$

Algorithms for NMF

$$D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) \stackrel{\text{cst.}}{=} \sum_{i,j} \left(-\mathbf{V}_{ij} \log \sum_k \pi_{ijk} \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

$$\leq \sum_{i,j} \left(-\mathbf{V}_{ij} \sum_k \pi_{ijk} \log \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

$$\mathbf{H}^*_{kj} = \frac{\sum_i \mathbf{V}_{ij}\,\pi_{ijk}}{\sum_i \mathbf{W}_{ik}}$$

But I haven't told you what π_{ijk} is. We have to choose π_{ijk} to make the function a *majorizing* function.

$$\pi_{ijk} = \frac{\mathbf{W}_{ik}\,\mathbf{H}^{(\ell)}_{kj}}{\sum_k \mathbf{W}_{ik}\,\mathbf{H}^{(\ell)}_{kj}} \text{ does the trick.}$$

Algorithms for NMF

$$D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) \overset{\text{cst.}}{=} \sum_{i,j} \left(-\mathbf{V}_{ij} \log \sum_k \pi_{ijk} \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

$$\leq \sum_{i,j} \left(-\mathbf{V}_{ij} \sum_k \pi_{ijk} \log \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

$$\mathbf{H}^*_{kj} = \frac{\sum_i \mathbf{V}_{ij}\,\pi_{ijk}}{\sum_i \mathbf{W}_{ik}}$$

But I haven't told you what π_{ijk} is. We have to choose π_{ijk} to make the function a *majorizing* function.

$$\pi_{ijk} = \frac{\mathbf{W}_{ik}\,\mathbf{H}^{(\ell)}_{kj}}{\sum_k \mathbf{W}_{ik}\,\mathbf{H}^{(\ell)}_{kj}} \text{ does the trick.}$$

Algorithms for NMF

$$D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) \overset{\text{cst.}}{=} \sum_{i,j} \left(-\mathbf{V}_{ij} \log \sum_k \pi_{ijk} \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

$$\leq \sum_{i,j} \left(-\mathbf{V}_{ij} \sum_k \pi_{ijk} \log \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

$$\mathbf{H}^*_{kj} = \frac{\sum_i \mathbf{V}_{ij}\,\pi_{ijk}}{\sum_i \mathbf{W}_{ik}}$$

But I haven't told you what π_{ijk} is. We have to choose π_{ijk} to make the function a *majorizing* function.

$\pi_{ijk} = \frac{\mathbf{W}_{ik}\,\mathbf{H}^{(\ell)}_{kj}}{\sum_k \mathbf{W}_{ik}\,\mathbf{H}^{(\ell)}_{kj}}$ does the trick.

Algorithms for NMF

$$D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) \overset{\text{cst.}}{=} \sum_{i,j} \left(-\mathbf{V}_{ij} \log \sum_k \pi_{ijk} \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

$$\leq \sum_{i,j} \left(-\mathbf{V}_{ij} \sum_k \pi_{ijk} \log \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}}{\pi_{ijk}} + \sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj} \right)$$

$$\mathbf{H}^*_{kj} = \frac{\sum_i \mathbf{V}_{ij}\,\pi_{ijk}}{\sum_i \mathbf{W}_{ik}}$$

But I haven't told you what π_{ijk} is. We have to choose π_{ijk} to make the function a *majorizing* function.

$$\pi_{ijk} = \frac{\mathbf{W}_{ik}\,\mathbf{H}^{(\ell)}_{kj}}{\sum_k \mathbf{W}_{ik}\,\mathbf{H}^{(\ell)}_{kj}} \text{ does the trick.}$$

Algorithms for NMF

If we substitute $\pi_{ijk} = \frac{\mathbf{W}_{ik}\,\mathbf{H}^{(\ell)}_{kj}}{\sum_k \mathbf{W}_{ik}\,\mathbf{H}^{(\ell)}_{kj}}$, we obtain the updates:

$$\mathbf{H}^{(\ell+1)}_{kj} \leftarrow \frac{\sum_i \mathbf{V}_{ij} \frac{\mathbf{W}_{ik}\,\mathbf{H}^{(\ell)}_{kj}}{\sum_k \mathbf{W}_{ik}\,\mathbf{H}^{(\ell)}_{kj}}}{\sum_i \mathbf{W}_{ik}}$$

$$= \mathbf{H}^{(\ell)}_{kj} \cdot \frac{\sum_i \left(\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}^{(\ell)}}\right)_{ij} \mathbf{W}_{ik}}{\sum_i \mathbf{W}_{ik}}$$

These are **multiplicative updates**. In matrix form:

$$\mathbf{H}^{(\ell+1)} \leftarrow \mathbf{H}^{(\ell)} \,.\!* \frac{\mathbf{W}^T \frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}^{(\ell)}}}{\mathbf{W}^T\,\mathbf{1}}$$

Algorithms for NMF

If we substitute $\pi_{ijk} = \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}^{(\ell)}}{\sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}^{(\ell)}}$, we obtain the updates:

$$\mathbf{H}_{kj}^{(\ell+1)} \leftarrow \frac{\sum_i \mathbf{V}_{ij}\,\frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}^{(\ell)}}{\sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}^{(\ell)}}}{\sum_i \mathbf{W}_{ik}}$$

$$= \mathbf{H}_{kj}^{(\ell)} \cdot \frac{\sum_i \left(\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}^{(\ell)}}\right)_{ij} \mathbf{W}_{ik}}{\sum_i \mathbf{W}_{ik}}$$

These are **multiplicative updates**. In matrix form:

$$\mathbf{H}^{(\ell+1)} \leftarrow \mathbf{H}^{(\ell)} .\!* \frac{\mathbf{W}^T\,\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}^{(\ell)}}}{\mathbf{W}^T\,\mathbf{1}}$$

Algorithms for NMF

If we substitute $\pi_{ijk} = \dfrac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}^{(\ell)}}{\sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}^{(\ell)}}$, we obtain the updates:

$$\mathbf{H}_{kj}^{(\ell+1)} \leftarrow \frac{\sum_i \mathbf{V}_{ij}\,\frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}^{(\ell)}}{\sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}^{(\ell)}}}{\sum_i \mathbf{W}_{ik}}$$

$$= \mathbf{H}_{kj}^{(\ell)} \cdot \frac{\sum_i \left(\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}^{(\ell)}}\right)_{ij} \mathbf{W}_{ik}}{\sum_i \mathbf{W}_{ik}}$$

These are **multiplicative updates**. In matrix form:

$$\mathbf{H}^{(\ell+1)} \leftarrow \mathbf{H}^{(\ell)} .* \frac{\mathbf{W}^T \frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}^{(\ell)}}}{\mathbf{W}^T\,\mathbf{1}}$$

Algorithms for NMF

If we substitute $\pi_{ijk} = \frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}^{(\ell)}}{\sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}^{(\ell)}}$, we obtain the updates:

$$\begin{aligned}
\mathbf{H}_{kj}^{(\ell+1)} \leftarrow \frac{\sum_i \mathbf{V}_{ij}\,\frac{\mathbf{W}_{ik}\,\mathbf{H}_{kj}^{(\ell)}}{\sum_k \mathbf{W}_{ik}\,\mathbf{H}_{kj}^{(\ell)}}}{\sum_i \mathbf{W}_{ik}} \\
= \mathbf{H}_{kj}^{(\ell)} \cdot \frac{\sum_i \left(\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}^{(\ell)}}\right)_{ij} \mathbf{W}_{ik}}{\sum_i \mathbf{W}_{ik}}
\end{aligned}$$

These are **multiplicative updates**. In matrix form:

$$\mathbf{H}^{(\ell+1)} \leftarrow \mathbf{H}^{(\ell)} \,.\!*\, \frac{\mathbf{W}^T\,\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}^{(\ell)}}}{\mathbf{W}^T\,\mathbf{1}}$$

Algorithms for NMF

Using $D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) = D(\mathbf{V}^T \,||\, \mathbf{H}^T\,\mathbf{W}^T)$, we obtain a similar update for $\mathbf{W}$.

Now we just iterate between:

1. Updating $\mathbf{W}$.
2. Updating $\mathbf{H}$.
3. Checking $D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H})$. If the change since the last iteration is small, then declare convergence.

The algorithm is summarized below:

Algorithm KL-NMF

initialize $\mathbf{W}, \mathbf{H}$

repeat

$$\mathbf{H} \leftarrow \mathbf{H} \,.\!* \frac{\mathbf{W}^T \frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}}}{\mathbf{W}^T\,\mathbf{1}}$$

$$\mathbf{W} \leftarrow \mathbf{W} \,.\!* \frac{\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}}\,\mathbf{H}^T}{\mathbf{1}\,\mathbf{H}^T}$$

until convergence **return** $\mathbf{W}, \mathbf{H}$

Algorithms for NMF

Using $D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) = D(\mathbf{V}^T \,||\, \mathbf{H}^T\,\mathbf{W}^T)$, we obtain a similar update for $\mathbf{W}$.

Now we just iterate between:

1. Updating $\mathbf{W}$.
2. Updating $\mathbf{H}$.
3. Checking $D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H})$. If the change since the last iteration is small, then declare convergence.

The algorithm is summarized below:

Algorithm KL-NMF

initialize $\mathbf{W}, \mathbf{H}$

repeat

$$\mathbf{H} \leftarrow \mathbf{H} \,.\!*\, \frac{\mathbf{W}^T \frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}}}{\mathbf{W}^T\,\mathbf{1}}$$

$$\mathbf{W} \leftarrow \mathbf{W} \,.\!*\, \frac{\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}}\,\mathbf{H}^T}{\mathbf{1}\,\mathbf{H}^T}$$

until convergence **return** $\mathbf{W}, \mathbf{H}$

Algorithms for NMF

Using $D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) = D(\mathbf{V}^T \,||\, \mathbf{H}^T\,\mathbf{W}^T)$, we obtain a similar update for $\mathbf{W}$.

Now we just iterate between:

1. Updating $\mathbf{W}$.
2. Updating $\mathbf{H}$.
3. Checking $D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H})$. If the change since the last iteration is small, then declare convergence.

The algorithm is summarized below:

Algorithm KL-NMF

---

initialize $\mathbf{W}, \mathbf{H}$

repeat

$$\mathbf{H} \leftarrow \mathbf{H} \,.\!* \frac{\mathbf{W}^T \frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}}}{\mathbf{W}^T\,\mathbf{1}}$$

$$\mathbf{W} \leftarrow \mathbf{W} \,.\!* \frac{\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}}\,\mathbf{H}^T}{\mathbf{1}\,\mathbf{H}^T}$$

until convergence **return** $\mathbf{W}, \mathbf{H}$

Algorithms for NMF

Using $D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) = D(\mathbf{V}^T \,||\, \mathbf{H}^T\,\mathbf{W}^T)$, we obtain a similar update for $\mathbf{W}$.

Now we just iterate between:

1. Updating $\mathbf{W}$.
2. Updating $\mathbf{H}$.
3. Checking $D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H})$. If the change since the last iteration is small, then declare convergence.

The algorithm is summarized below:

Algorithm KL-NMF

---

initialize $\mathbf{W}, \mathbf{H}$

repeat

$$\mathbf{H} \leftarrow \mathbf{H} \,.\!* \frac{\mathbf{W}^T \frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}}}{\mathbf{W}^T\,\mathbf{1}}$$

$$\mathbf{W} \leftarrow \mathbf{W} \,.\!* \frac{\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}}\,\mathbf{H}^T}{\mathbf{1}\,\mathbf{H}^T}$$

until convergence **return** $\mathbf{W}, \mathbf{H}$

Algorithms for NMF

Using $D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) = D(\mathbf{V}^T \,||\, \mathbf{H}^T\,\mathbf{W}^T)$, we obtain a similar update for $\mathbf{W}$.

Now we just iterate between:

1. Updating $\mathbf{W}$.
2. Updating $\mathbf{H}$.
3. Checking $D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H})$. If the change since the last iteration is small, then declare convergence.

The algorithm is summarized below:

Algorithm KL-NMF

initialize $\mathbf{W}, \mathbf{H}$

repeat

$$\mathbf{H} \leftarrow \mathbf{H}\,.{*}\,\frac{\mathbf{W}^T\,\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}}}{\mathbf{W}^T\,\mathbf{1}}$$

$$\mathbf{W} \leftarrow \mathbf{W}\,.{*}\,\frac{\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}}\,\mathbf{H}^T}{\mathbf{1}\,\mathbf{H}^T}$$

until convergence **return** $\mathbf{W}, \mathbf{H}$

Algorithms for NMF

Using $D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H}) = D(\mathbf{V}^T \,||\, \mathbf{H}^T\,\mathbf{W}^T)$, we obtain a similar update for $\mathbf{W}$.

Now we just iterate between:

1. Updating $\mathbf{W}$.
2. Updating $\mathbf{H}$.
3. Checking $D(\mathbf{V} \,||\, \mathbf{W}\,\mathbf{H})$. If the change since the last iteration is small, then declare convergence.

The algorithm is summarized below:

Algorithm KL-NMF

---

initialize $\mathbf{W}, \mathbf{H}$

repeat

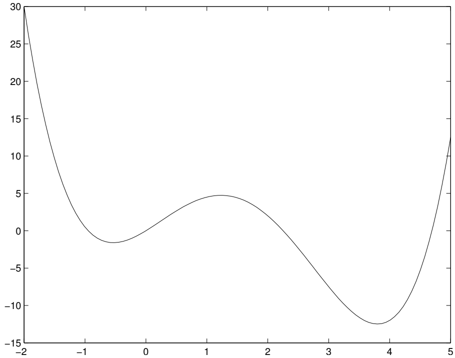
$$\mathbf{H} \leftarrow \mathbf{H} \,.\!* \frac{\mathbf{W}^T \frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}}}{\mathbf{W}^T\,\mathbf{1}}$$

$$\mathbf{W} \leftarrow \mathbf{W} \,.\!* \frac{\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}}\,\mathbf{H}^T}{\mathbf{1}\,\mathbf{H}^T}$$

until convergence **return** $\mathbf{W}, \mathbf{H}$

Caveats

- The NMF problem is **non-convex**.



- The algorithm is only guaranteed to find a local optimum.
- The algorithm is sensitive to choice of initialization.

Roadmap of Talk

1. Motivation
2. Current Approaches
3. Non-Negative Matrix Factorization (NMF)
4. Source Separation via NMF
5. Algorithms for NMF
6. Matlab Code

STFT

```
FFTSIZE = 1024;
HOPSIZE = 256;
WINDOWSIZE = 512;

X = myspectrogram(x,FFTSIZE,fs,hann(WINDOWSIZE),-HOPSIZE);
V = abs(X(1:(FFTSIZE/2+1),:));
F = size(V,1);
T = size(V,2);
```

- https://ccrma.stanford.edu/~jos/sasp/Matlab_listing_myspectrogram_m.html
- https://ccrma.stanford.edu/~jos/sasp/Matlab_listing_invmyspectrogram_m.html

NMF

```
function [W, H] = nmf(V, K, MAXITER)

F = size(V,1);
T = size(V,2);

rand('seed',0)
W = 1+rand(F, K);
H = 1+rand(K, T);

ONES = ones(F,T);

for i=1:MAXITER
    % update activations
    H = H .* (W'*( V./(W*H+eps))) ./ (W'*ONES);
    % update dictionaries
    W = W .* ((V./(W*H+eps))*H') ./(ONES*H');
end

% normalize W to sum to 1
sumW = sum(W);
W = W*diag(1./sumW);
H = diag(sumW)*H;
```

FILTER & ISTFT

```
phi = angle(X);
% reconstruct each basis as a separate source
for i=1:K

    XmagHat = W(:,i)*H(i,:);

    % create upper half of frequency before istft
    XmagHat = [XmagHat; conj( XmagHat(end-1:-1:2,:))];

    % Multiply with phase
    XHat = XmagHat.*exp(1i*phi);

    xhat(:,i) = real(invmyspectrogram(XHat,HOPSIZE))';

end
```

References I

- Jacob Benesty, Jingdong Chen, and Yiteng Huang, *Microphone array signal processing*, Springer, 2008.
- C. Févotte, N. Bertin, and J.-L. Durrieu, *Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis*, Neural Computation **21** (2009), no. 3, 793–830.
- C. Févotte and J. Idier, *Algorithms for nonnegative matrix factorization with the β-divergence*, Neural Computation **23** (2011), no. 9, 2421–2456.
- A. Hyvärinen and E. Oja, *Independent component analysis: algorithms and applications*, Neural Netw. **13** (2000), no. 4-5, 411–430.
- D. D. Lee and H. S. Seung, *Algorithms for non-negative matrix factorization*, Advances in Neural Information Processing Systems (NIPS), MIT Press, 2001, pp. 556–562.

References II

- P. Smaragdis and J.C. Brown, *Non-negative matrix factorization for polyphonic music transcription*, IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), oct. 2003, pp. 177 – 180.
- J. O. Smith, *Spectral audio signal processing*, `http://ccrma.stanford.edu/~jos/sasp/`, 2011, online book.
- Avery Li-chun Wang, *Instantaneous and frequency-warped signal processing techniques for auditory source separation*, Ph.D. thesis, Stanford University, 1994.
- DeLiang Wang and Guy J. Brown, *Computational auditory scene analysis: Principles, algorithms, and applications*, Wiley-IEEE Press, 2006.

References III

Bernard Widrow and Samuel D. Stearns, *Adaptive signal processing*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1985.