Machine Learning for Signal Processing
Non-negative Matrix Factorization

Class 9. 29 Sep 2016

Instructor: Bhiksha Raj

With examples and slides from Paris Smaragdis
• **Problem:** Given a collection of data $X$, find a set of “bases” $B$, such that each vector $x_i$ can be expressed as a weighted combination of the bases.

\[ x_i = Bw_i \]

\[ x_i = w_{11}B_1 + \cdots + w_{1K}B_K \]
A Quick Recap: Subproblem 1

- **Problem 1**: Finding bases
  - Finding typical faces
  - Finding “notes” like structures
A Quick Recap: Subproblem 2

- Problem 2: Expressing instances in terms of these bases
  - Finding weights of typical faces
  - Finding weights of notes

• **Better Representation**: The weights \( \{w_{ij}\} \) represent the vectors in a *meaningful* way
  – Better suited to semantically motivated operation
  – Better suited for specific statistical models
A Quick Recap: **WHY? 2.**

- **Dimensionality Reduction:** The number of Bases may be fewer than the dimensions of the vectors
  - Represent each Vector using fewer numbers
  - Expresses each vector within a *subspace*
    - Loses information / energy
    - **Objective:** Lose *least* energy

- **Denoising**: Reduced dimensional representation eliminates dimensions
- Can often eliminate *noise* dimensions
  - Signal-to-Noise ratio worst in dimensions where the signal has least energy/information
  - Removing them eliminates noise
A Quick Recap: HOW? PCA

- **Find Eigenvectors of Correlation matrix**
  - These are our “eigen” bases
  - Capture information compactly and satisfy most of our requirements

- **MOST??**
• **What is a negative face?**
  – And what does it mean to subtract one face from the other?
• Problem more obvious when applied to music
  – You would like bases to be notes
  – Weights to be scores
  – What is a negative note? What is a negative score?
Summary

• Decorrelation and Independence are statistically meaningful operations
• But may not be *physically* meaningful

• Next: A physically meaningful constraint
  – Non-negativity
The Engineer and the Musician

Once upon a time a rich potentate discovered a previously unknown recording of a beautiful piece of music. Unfortunately it was badly damaged.

He greatly wanted to find out what it would sound like if it were not.

So he hired an engineer and a musician to solve the problem.
The Engineer and the Musician

The engineer worked for many years. He spent much money and published many papers.

Finally he had a somewhat scratchy restoration of the music..

The musician listened to the music carefully for a day, transcribed it, broke out his trusty keyboard and replicated the music.
Who do you think won the princess?
The search for building blocks

- What composes an audio signal?
  - E.g. notes compose music
The properties of building blocks

- **Constructive composition**
  - A second note does not diminish a first note

- **Linearity of composition**
  - Notes do not distort one another
Looking for building blocks in sound

Can we compute the building blocks from sound itself
The spectrogram of the sum of two signals is the sum of their spectrograms

- This is a property of the Fourier transform that is used to compute the columns of the spectrogram

- The individual spectral vectors of the spectrograms add up
  - Each column of the first spectrogram is added to the same column of the second

- Building blocks can be learned by using this property
  - Learn the building blocks of the “composed” signal by finding what vectors were added to produce it
Another property of spectrograms

- We deal with the *power* in the signal
  - The power in the sum of two signals is the sum of the powers in the individual signals
  - The power of any frequency component in the sum at any time is the sum of the powers in the individual signals at that frequency and time
- The power is strictly non-negative (real)
The building blocks of sound are (power) spectral structures:
- E.g. notes build music
- The spectra are entirely non-negative

The complete sound is composed by constructive combination of the building blocks scaled to different non-negative gains:
- E.g. notes are played with varying energies through the music
- The sound from the individual notes combines to form the final spectrogram

The final spectrogram is also non-negative
Building Blocks of Sound

- Each frame of sound is composed by activating each spectral building block by a frame-specific amount.
- Individual frames are composed by activating the building blocks to different degrees.
  - E.g. notes are strummed with different energies to compose the frame.
Composing the Sound

- Each frame of sound is composed by activating each spectral building block by a frame-specific amount.
- Individual frames are composed by activating the building blocks to different degrees.
  - E.g. notes are strummed with different energies to compose the frame.
Building Blocks of Sound

- Each frame of sound is composed by activating each spectral building block by a frame-specific amount.

- Individual frames are composed by activating the building blocks to different degrees.
  - E.g. notes are strummed with different energies to compose the frame.
Building Blocks of Sound

- Each frame of sound is composed by activating each spectral building block by a frame-specific amount.
- Individual frames are composed by activating the building blocks to different degrees.
  - E.g. notes are strummed with different energies to compose the frame.
Building Blocks of Sound

- Each frame of sound is composed by activating each spectral building block by a frame-specific amount.
- Individual frames are composed by activating the building blocks to different degrees.
  - E.g. notes are strummed with different energies to compose the frame.
The Problem of Learning

- Given only the final sound, determine its building blocks
- From only listening to music, learn all about musical notes!
In Math

- Each frame is a non-negative power spectral vector
- Each note is a non-negative power spectral vector
- Each frame is a non-negative combination of the notes

$$V_1 = w_{11}B_1 + w_{21}B_2 + w_{31}B_3 + \ldots$$
Expressing a vector in terms of other vectors
Expressing a vector in terms of other vectors
Expressing a vector in terms of other vectors

\[ 2a + 5b = 4 \]
\[ 3a - 3b = 2 \]

\[
\begin{bmatrix}
2 & 5 \\
3 & -3
\end{bmatrix}
\begin{bmatrix}
a \\
b
\end{bmatrix} =
\begin{bmatrix}
4 \\
2
\end{bmatrix}
\]

\[
\begin{bmatrix}
a \\
b
\end{bmatrix} =
\begin{bmatrix}
2 & 5 \\
3 & -3
\end{bmatrix}^{-1}
\begin{bmatrix}
4 \\
2
\end{bmatrix}
\]

\[
\begin{bmatrix}
a \\
b
\end{bmatrix} =
\begin{bmatrix}
1.04761905 \\
0.38095238
\end{bmatrix}
\]

\[ V = 1.048B_1 + 0.381B_2 \]
Power spectral vectors: Requirements

- \( V \) has only non-negative components
  - Is a power spectrum
- \( B_1 \) and \( B_2 \) have only non-negative components
  - Power spectra of building blocks of audio
  - E.g. power spectra of notes
- \( a \) and \( b \) are strictly non-negative
  - Building blocks don’t subtract from one another
Learning building blocks: Restating the problem

- Given a collection of spectral vectors (from the composed sound) ...
- Find a set of “basic” sound spectral vectors such that ...
- All of the spectral vectors can be composed through constructive addition of the bases
  - We never have to flip the direction of any basis
Learning building blocks: Restating the problem

\[ V = BW \]

- Each column of \( V \) is one “composed” spectral vector
- Each column of \( B \) is one building block
  - One spectral basis
- Each column of \( W \) has the scaling factors for the building blocks to compose the corresponding column of \( V \)
- All columns of \( V \) are non-negative
- All entries of \( B \) and \( W \) must also be non-negative
Non-negative matrix factorization: Basics

- NMF is used in a *compositional* model
- Data are assumed to be non-negative
  - E.g. power spectra
- Every data vector is explained as a purely constructive linear composition of a set of bases
  - $V = \sum_i w_i B_i$
  - The bases $B_i$ are in the same domain as the data
    - I.e. they are power spectra
- Constructive composition: no subtraction allowed
  - Weights $w_i$ must all be non-negative
  - All components of bases $B_i$ must also be non-negative
Interpreting non-negative factorization

- Bases are non-negative, lie in the positive quadrant
- Blue lines represent bases, blue dots represent vectors
- Any vector that lies between the bases (highlighted region) can be expressed as a non-negative combination of bases
  - E.g. the black dot
Interpreting non-negative factorization

- Vectors outside the shaded enclosed area can only be expressed as a linear combination of the bases by reversing a basis
  - I.e. assigning a negative weight to the basis
  - E.g. the red dot
- Alpha and beta are scaling factors for bases
- Beta weighting is negative
Interpreting non-negative factorization

- If we approximate the red dot as a non-negative combination of the bases, the approximation will lie in the shaded region
  - On or close to the boundary
  - The approximation has error
The NMF representation

- The representation characterizes all data as lying within a compact convex region
  - “Compact” → enclosing only a small fraction of the entire space
  - The more compact the enclosed region, the more it localizes the data within it
    - Represents the boundaries of the distribution of the data better
      - Conventional statistical models represent the mode of the distribution

- The *bases* must be chosen to
  - Enclose the data as compactly as possible
  - And also enclose as much of the data as possible
    - Data that are not enclosed are not represented correctly
Data need not be non-negative

- The general principle of enclosing data applies to any one-sided data whose distribution does not cross the origin.
- The only part of the model that must be non-negative are the weights.
- Examples
  - Blue bases enclose blue region in negative quadrant
  - Red bases enclose red region in positive-negative quadrant
- Notions of compactness and enclosure still apply
  - This is a generalization of NMF
  - We won't discuss it further
Given a collection of data vectors (blue dots)

Goal: find a set of bases (blue arrows) such that they enclose the data.

Ideally, they must simultaneously enclose the smallest volume

This “enclosure” constraint is usually not explicitly imposed in the standard NMF formulation
NMF: Learning Bases

- Express every training vector as non-negative combination of bases
  - \( V = \sum_i w_i B_i \)

- In linear algebraic notation, represent:
  - Set of all training vectors as a data matrix \( V \)
    - A \( D \times N \) matrix, \( D \) = dimensionality of vectors, \( N \) = No. of vectors
  - All basis vectors as a matrix \( B \)
    - A \( D \times K \) matrix, \( K \) is the number of bases
  - The \( K \) weights for any vector \( V \) as a \( K \times 1 \) column vector \( W \)
  - The weight vectors for all \( N \) training data vectors as a matrix \( W \)
    - A \( K \times N \) matrix

- Ideally \( V = BW \)
**NMF: Learning Bases**

- \( \mathbf{V} = \mathbf{BW} \) will only hold true if all training vectors in \( \mathbf{V} \) lie inside the region enclosed by the bases.

- Learning bases is an iterative algorithm.

- Intermediate estimates of \( \mathbf{B} \) do not satisfy \( \mathbf{V} = \mathbf{BW} \).

- Algorithm updates \( \mathbf{B} \) until \( \mathbf{V} = \mathbf{BW} \) is satisfied as closely as possible.
NMF: Minimizing Divergence

- Define a *Divergence* between data $V$ and approximation $BW$
  - Divergence($V$, $BW$) is the total error in approximating all vectors in $V$ as $BW$
  - Must estimate $B$ and $W$ so that this error is minimized

- Divergence($V$, $BW$) can be defined in different ways
  - L2: $\text{Divergence} = \sum_{i\sum_j} (V_{ij} - (BW)_{ij})^2$
    - Minimizing the L2 divergence gives us an algorithm to learn $B$ and $W$
  - KL: $\text{Divergence}(V,BW) = \sum_{i\sum_j} V_{ij} \log(V_{ij} / (BW)_{ij}) + \sum_{i\sum_j} V_{ij} - \sum_{i\sum_j} (BW)_{ij}$
    - This is a *generalized* KL divergence that is minimum when $V = BW$
    - Minimizing the KL divergence gives us another algorithm to learn $B$ and $W$

- Other divergence forms can also be used
NMF: Minimizing Divergence

- Define a *Divergence* between data \( V \) and approximation \( BW \)
  - Divergence\((V, BW)\) is the total error in approximating all vectors in \( V \) as \( BW \)
  - Must estimate \( B \) and \( W \) so that this error is minimized

- Divergence\((V, BW)\) can be defined in different ways
  - L2:  \[ \text{Divergence} = \sum_i \sum_j (V_{ij} - (BW)_{ij})^2 \]
    - Minimizing the L2 divergence gives us an algorithm to learn \( B \) and \( W \)
  - KL: \[ \text{Divergence}(V,BW) = \sum_i \sum_j V_{ij} \log(V_{ij} / (BW)_{ij}) + \sum_i \sum_j V_{ij} - \sum_i \sum_j (BW)_{ij} \]
    - This is a *generalized* KL divergence that is minimum when \( V = BW \)
    - Minimizing the KL divergence gives us another algorithm to learn \( B \) and \( W \)

- Other divergence forms can also be used
NMF: Minimizing $L_2$ Divergence

- Divergence($V$, $BW$) is defined as
  \[ E = ||V - BW||_F^2 \]
  \[ E = \sum_i \sum_j (V_{ij} - (BW)_{ij})^2 \]

- Iterative solution: Minimize $E$ such that $B$ and $W$ are strictly non-negative
NMF: Minimizing $L_2$ Divergence

- Learning both $\mathbf{B}$ and $\mathbf{W}$ with non-negativity
- Divergence($\mathbf{V}$, $\mathbf{BW}$) is defined as
  - $E = \| \mathbf{V} - \mathbf{BW} \|_F^2$
  - $\mathbf{V} \approx \mathbf{BW}$

- Iterative solution:
  - $\mathbf{B} = [\mathbf{V} \text{Pinv}(\mathbf{W})]_+$
  - $\mathbf{W} = [\text{Pinv}(\mathbf{B}) \mathbf{V}]_+$
  - Subscript + indicates thresholding –ve values to 0
**NMF: Minimizing Divergence**

- **Define a** *Divergence* **between data** $V$ **and approximation** $BW$
  - $\text{Divergence}(V, BW)$ is the total error in approximating all vectors in $V$ as $BW$
  - Must estimate $B$ and $W$ so that this error is minimized

- **Divergence($V$, $BW$) can be defined in different ways**
  - **L2**: $\text{Divergence} = \sum_i \sum_j (V_{ij} - (BW)_{ij})^2$
    - Minimizing the L2 divergence gives us an algorithm to learn $B$ and $W$
  - **KL**: $\text{Divergence}(V,BW) = \sum_i \sum_j V_{ij} \log(V_{ij} / (BW)_{ij}) + \sum_i \sum_j V_{ij} - \sum_i \sum_j (BW)_{ij}$
    - This is a *generalized* KL divergence that is minimum when $V = BW$
    - Minimizing the KL divergence gives us another algorithm to learn $B$ and $W$

- **For many kinds of signals, e.g. sound, NMF-based representations work best when we minimize the KL divergence**
NMF: Minimizing KL Divergence

- Divergence($V, BW$) defined as
  \[ E = \sum_i \sum_j V_{ij} \log(V_{ij} / (BW)_{ij}) + \sum_i \sum_j V_{ij} - \sum_i \sum_j (BW)_{ij} \]

- Iterative update rules
- Number of iterative update rules have been proposed
- The most popular one is the multiplicative update rule.
The algorithm to estimate $B$ and $W$ to minimize the KL divergence between $V$ and $BW$:

- Initialize $B$ and $W$ (randomly)
- Iteratively update $B$ and $W$ using the following formulae

$$
B = B \times \left( \frac{V}{BW} \right) W^T \\
W = W \times \left( \frac{V}{BW} \right) B^T 1
$$

- Iterations continue until divergence converges
  - In practice, continue for a fixed no. of iterations
NMF learns the *optimal set of basis vectors* $B_k$ to approximate the data in terms of the bases.

It also learns how to compose the data in terms of these bases.

- Compositions can be inexact.

\[
V_{D\times N} \approx B_{D\times K} W_{K\times N}
\]

\[
V_L \approx \sum_k w_{L,k} B_k
\]
Learning building blocks of sound

From Bach’s Fugue in Gm

\[ V = BW \]

- Each column of \( V \) is one spectral vector
- Each column of \( B \) is one building block/basis
- Each column of \( W \) has the scaling factors for the bases to compose the corresponding column of \( V \)
- All terms are non-negative
- Learn \( B \) (and \( W \)) by applying NMF to \( V \)
Learning Building Blocks

Speech Signal

bases

Basis-specific spectrograms
What about other data

- **Faces**
  - Trained 49 multinomial components on 2500 faces
    - Each face unwrapped into a 361-dimensional vector
  - Discovers parts of faces
There is no “compactness” constraint

- No explicit “compactness” constraint on bases
- The red lines would be perfect bases:
  - Enclose all training data without error
  - Algorithm can end up with these bases
  - If no. of bases $K \geq$ dimensionality $D$, can get uninformative bases

- If $K < D$, we usually learn compact representations
  - NMF becomes a dimensionality reducing representation
    - Representing $D$-dimensional data in terms of $K$ weights, where $K < D$
Representing Data using *Known* Bases

- If we already have bases $B_k$ and are given a vector that must be expressed in terms of the bases: $V \approx \sum_k w_k B_k$

- Estimate weights as:
  - Initialize weights
  - Iteratively update them using

\[
W = W \otimes \frac{B^T \left( \frac{V}{BW} \right)}{B^T 1}
\]
What can we do knowing the building blocks

- Signal Representation
- Signal Separation
- Signal Completion
- Denoising
- Signal recovery
- Music Transcription
- Etc.
Signal Separation

- Can we separate mixed signals?
Given two distinct sets of building blocks, can we find which parts of a composition were composed from which blocks.
Separating Sounds

- From example of A, learn blocks A (NMF)

\[ V_1 = B_1 W_1 \]

given estimate

estimate
Separating Sounds

- From example of A, learn blocks A (NMF)
- From example of B, learn B (NMF)

\[ V_2 = B_2 W_2 \text{ estimate} \]

given estimate

59
Separating Sounds

From mixture, separate out (NMF)

- Use known “bases” of both sources
- Estimate the weights with which they combine in the mixed signal

\[ V = BW \]

\[
\begin{bmatrix}
B_1 & B_2
\end{bmatrix}
\begin{bmatrix}
W_1 \\
W_2
\end{bmatrix}
\]

given

given

estimate
Separating Sounds

Separated signals are estimated as the contributions of the source-specific bases to the mixed signal.
Separating Sounds

It is sometimes sufficient to know the bases for only one source

- The bases for the other can be estimated from the mixed signal itself.
Separating Sounds

- “Raise my rent” by David Gilmour
- Background music “bases” learnt from 5-seconds of music-only segments within the song
- Lead guitar “bases” bases learnt from the rest of the song
- Norah Jones singing “Sunrise”
- Background music bases learnt from 5 seconds of music-only segments
Predicting Missing Data

- Use the building blocks to fill in “holes”
Filling in

- Some frequency components are missing (left panel)
- We know the bases
  - But not the mixture weights for any particular spectral frame
- We must “fill in” the holes in the spectrogram
  - To obtain the one to the right
Learn building blocks

- Learn the building blocks from other examples of similar sounds
  - E.g. music by same singer
  - E.g. from undamaged regions of same recording
Predict data

- “Modify” bases to look like damaged spectra
  - Remove appropriate spectral components
- Learn how to compose damaged data with modified bases
- Reconstruct missing regions with complete bases

\[ \hat{V} = \hat{B}W \] estimate

\[ V = BW \] estimate

Modified bases (given)

Full bases
Filling in: An example

- Madonna...
- Bases learned from other Madonna songs
A more fun example

- Reduced BW data

- Bases learned from this

- Bandwidth expanded version
For K-dimensional data, can learn no more than K-1 bases meaningfully

- At K bases, simply select the axes as bases
- The bases will represent all data exactly
Its an unnatural restriction

- For K-dimensional spectra, can learn no more than K-1 bases
- Nature does not respect the dimensionality of your spectrogram
- E.g. Music: There are tens of instruments
  - Each can produce dozens of unique notes
  - Amounting to a total of many thousands of notes
  - Many more than the dimensionality of the spectrum
- E.g. images: a 1024 pixel image can show millions of recognizable pictures!
  - Many more than the number of pixels in the image
Fixing the restriction: Updated model

- Can have a *very large* number of building blocks (bases)
  - E.g. notes

- But any *particular* frame is composed of only a small subset of bases
  - E.g. any single frame only has a small set of notes
The Modified Model

\[ V = BW \quad \text{For one vector} \]

- **Modification 1:**
  - In any column of \( W \), only a small number of entries have non-zero value
  - I.e. the columns of \( W \) are *sparse*
  - These are *sparse* representations

- **Modification 2:**
  - \( B \) may have more columns than rows
  - These are called *overcomplete* representations

Sparse representations need not be overcomplete, but the reverse will generally not provide useful decompositions
Imposing Sparsity

\[ V = BW \]

\[ E = Div(V, BW) \]

\[ Q = Div(V, BW) + \lambda |W|_0 \]

- Minimize a modified objective function
- Combines divergence and ell-0 norm of \( W \)
  - The number of non-zero elements in \( W \)
- Minimize \( Q \) instead of \( E \)
  - Simultaneously minimizes both divergence and number of active bases at any time
Imposing Sparsity

\[ V = BW \]

\[ Q = Div(V, BW) + \lambda |W|_0 \]

- Minimize the ell-0 norm is hard
  - Combinatorial optimization
- Minimize ell-1 norm instead
  - The sum of all the entries in W
  - Relaxation
- Is equivalent to minimize ell-0
  - We cover this equivalence later
- Will also result in sparse solutions
Update Rules

- Modified Iterative solutions
  - In gradient based solutions, gradient w.r.t any $W$ term now includes $\lambda$
  - I.e. if $\frac{dQ}{dW} = \frac{dE}{dW} + \lambda$

- For KL Divergence, results in following modified update rules

\[
B = B \otimes \left( \frac{V}{BW} \right) W^T
\]

\[
W = W \otimes \frac{B^T \left( \frac{V}{BW} \right)}{B^T 1 + \lambda}
\]

- Increasing $\lambda$ makes the weights increasingly sparse
Update Rules

- **Modified Iterative solutions**
  - In gradient based solutions, gradient w.r.t any $W$ term now includes $\lambda$
  - i.e. if $dQ/dW = dE/dW + \lambda$

- **Both $B$ and $W$ can be made sparse**

\[
B = B \otimes \frac{\left( \frac{V}{BW} \right) W^T}{1W^T + \lambda_b}
\]

\[
W = W \otimes \frac{B^T \left( \frac{V}{BW} \right)}{B^T 1 + \lambda_w}
\]
What about Overcompleteness?

- Use the same solutions
- Simply make $B$ wide!
  - $W$ must be made sparse

\[
B = B \otimes \frac{V}{BW} W^T
\]

\[
W = W \otimes \frac{B^T \left( \frac{V}{BW} \right)}{B^T 1 + \lambda_w}
\]
Sparsity: What do we learn

- Without sparsity: The model has an implicit limit: can learn no more than $D-1$ useful bases
  - If $K \geq D$, we can get uninformative bases

- Sparsity: The bases are “pulled towards” the data
  - Representing the distribution of the data much more effectively
Sparsity: What do we learn

- **Top and middle panel:** Compact (non-sparse) estimator
  - As the number of bases increases, bases migrate towards corners of the orthant
- **Bottom panel:** Sparse estimator
  - Cone formed by bases shrinks to fit the data

Each dot represents a location where a vector “pierces” the simplex
The Vowels and Music Examples

- Left panel, Compact learning: most bases have significant energy in all frames
- Right panel, Sparse learning: Fewer bases active within any frame
  - Decomposition into basic sounds is cleaner
Sparse Overcomplete Bases: Separation

- 3000 bases for each of the speakers
  - The speaker-to-speaker ratio typically doubles (in dB) w.r.t compact bases

Regular bases

Sparse bases

Panels 2 and 3: Regular learning

Panels 4 and 5: Sparse learning
Sparseness: what do we learn

- As solutions get more sparse, bases become more informative
  - In the limit, each basis is a complete face by itself.
  - Mixture weights simply select face

Sparse bases

Dense bases

“Dense” weights

Sparse weights
Filling in missing information

- 19x19 pixel images (361 pixels)
- 1000 bases trained from 2000 faces
- SNR of reconstruction from overcomplete basis set more than 10dB better than reconstruction from corresponding “compact” (regular) basis set
Extending the model

- In reality our building blocks are not spectra
- They are spectral patterns!
  - Which change with time
The building blocks of sound are spectral patches!
The building blocks of sound are spectral patches!

At each time, they combine to compose a patch starting from that time.

Overlapping patches add
Convolutive NMF

- The building blocks of sound are spectral patches!
- At each time, they combine to compose a patch starting from that time
- Overlapping patches add
Convolutive NMF

- The building blocks of sound are spectral patches!
- At each time, they combine to compose a patch starting from that time
- Overlapping patches add
The building blocks of sound are spectral patches!

At each time, they combine to compose a patch starting from that time

Overlapping patches *add*
The building blocks of sound are spectral patches!

At each time, they combine to compose a patch starting from that time

Overlapping patches add
In Math

Each spectral frame has contributions from several previous shifts

\[
S(t) = \sum_i w_i(0)B_i(t) + \sum_i w_i(1)B_i(t-1) + \sum_i w_i(2)B_i(t-2) + \ldots = \sum_i \sum_\tau w_i(\tau)B_i(t-\tau)
\]

\[
S(t) = \sum_i B_i(t) \otimes w_i(t)
\]
An Alternate Representation

- **B(t)** is a matrix composed of the $t$-th columns of all bases
  - The $i$-th column represents the $i$-th basis
- **W** is a matrix whose $i$-th row is sequence of weights applied to the $i$-th basis
  - The superscript $t\rightarrow$ represents a right shift by $t$

\[
S(t) = \sum_i \sum_{\tau} w_i(\tau)B_i(t-\tau) = \sum_i \sum_{\tau} w_i(t-\tau)B_i(\tau)
\]

\[
S(t) = \sum_{\tau} \sum_i B_i(\tau)w_i(t-\tau)
\]

\[
S = \sum_{\tau} B(\tau)\rightarrow W
\]
**Convolutive NMF**

- Simple learning rules for $\mathbf{B}$ and $\mathbf{W}$
- Identical rules to estimate $\mathbf{W}$ given $\mathbf{B}$
  - Simply don’t update $\mathbf{B}$
- Sparsity can be imposed on $\mathbf{W}$ as before if desired
The Convolutive Model

- An Example: Two distinct sounds occurring with different repetition rates within a signal
  - Each sound has a time-varying spectral structure

INPUT SPECTROGRAM

Discovered “patch” bases

Contribution of individual bases to the recording
Example applications: Dereverberation

- From “Adrak ke Panje” by Babban Khan
- Treat the reverberated spectrogram as a composition of many shifted copies of a “clean” spectrogram
  - “Shift-invariant” analysis
- NMF to estimate clean spectrogram
Pitch Tracking

- Left: A segment of a song
- Right: Smoke on the water
  - “Impulse” distribution captures the “melody”!
Simultaneous pitch tracking on multiple instruments

Can be used to find the velocity of cars on the highway!!

“Pitch track” of sound tracks Doppler shift (and velocity)
Example: 2-D shift invariance

- Sparse decomposition employed in this example
  - Otherwise locations of faces (bottom right panel) are not precisely determined
Example: 2-D shift invariance

- The original figure has multiple handwritten renderings of three characters
  - In different colours
- The algorithm learns the three characters and identifies their locations in the figure
Example: Transform Invariance

- Top left: Original figure
- Bottom left – the two bases discovered
- Bottom right –
  - Left panel, positions of “a”
  - Right panel, positions of “l”
- Top right: estimated distribution underlying original figure
Example: Higher dimensional data

- Video example
Lessons learned

- Linear decomposition when constrained with semantic constraints e.g. non-negativity can result in semantically meaningful bases.

- NMF: Useful *compositional* model of data.

- Really effective when the data obey compositional rules.