Music Information Retrieval in Polyphonic Mixtures

Steve Tjoa

MIR Workshop
CCRMA, Stanford University

June 27, 2013
Quick Review

- What are the three main components of any classification system?
- What are some useful features for MIR?
- What are some problems and applications addressed by MIR?
Music Transcription

From this song...
From this song... get the “piano roll”:

![Piano Roll Diagram](image-url)
Isolate, amplify, or suppress a musical voice/instrument.

Example: From these beats...
Isolate, amplify, or suppress a musical voice/instrument.

Example: From these beats... isolate the kick drum and snare drum.
Nonnegative Matrix Factorization (NMF):

- Given $X$ nonnegative, find $W$ and $H$, both nonnegative, that minimize some distance $d(X, WH)$. 
Nonnegative Matrix Factorization (NMF):
- Given $X$ nonnegative, find $W$ and $H$, both nonnegative, that minimize some distance $d(X, WH)$.
- Easy! And it works.
- *Meaningful* to humans.
- Widely used.
Why NMF?

Energy of musical events are nonnegative.
Brief Refresher: Matrix Multiplication

\[
\begin{bmatrix}
1 & 2 \\
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
\end{bmatrix} = a + 2b
\]
Brief Refresher: Matrix Multiplication

\[
\begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = a + 2b
\]

\[
\begin{bmatrix} 3 \\ 4 \end{bmatrix} \begin{bmatrix} a & b & c \end{bmatrix} = \begin{bmatrix} 3a & 3b & 3c \\ 4a & 4b & 4c \end{bmatrix}
\]
Brief Refresher: Matrix Multiplication

\[
\begin{bmatrix}
1 & 2
\end{bmatrix}
\begin{bmatrix}
a \\
b
\end{bmatrix} = a + 2b
\]

\[
\begin{bmatrix}
3 \\
4
\end{bmatrix}
\begin{bmatrix}
a & b & c
\end{bmatrix} =
\begin{bmatrix}
3a & 3b & 3c \\
4a & 4b & 4c
\end{bmatrix}
\]

\[
w
\begin{bmatrix}
a & b & c
\end{bmatrix} =
\begin{bmatrix}
a \cdot w & b \cdot w & c \cdot w
\end{bmatrix}
\]
Brief Refresher: Matrix Multiplication

\[
\begin{bmatrix}
1 & 2 \\
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
\end{bmatrix}
= a + 2b
\]

\[
\begin{bmatrix}
3 & \\
4 & \\
\end{bmatrix}
\begin{bmatrix}
a & b & c \\
\end{bmatrix}
= 
\begin{bmatrix}
3a & 3b & 3c \\
4a & 4b & 4c \\
\end{bmatrix}
\]

\[
w
\begin{bmatrix}
a & b & c \\
\end{bmatrix}
= 
\begin{bmatrix}
aw & bw & cw \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
3 & \\
4 & \\
\end{bmatrix}
h
= 
\begin{bmatrix}
3h \\
4h \\
\end{bmatrix}
\]
Nonnegative Matrix Factorization

Top right: $X$. Left: $W$. Bottom: $H$. Three piano notes:

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Top right: $\mathbf{X}$. Left: $\mathbf{W}$. Bottom: $\mathbf{H}$. Kick and snare:
NMF Algorithms

Multiplicative update rules:

\[ W \leftarrow W \cdot \frac{XH^T}{WHH^T} \quad H \leftarrow H \cdot \frac{W^TX}{W^TWH} \]

See [Lee and Seung, NIPS 2001].
NMF Algorithms

Easy to implement!

Python:

```python
for iter in range(maxiter):
    W = multiply(W, (X*H.T)/(W*H*H.T))
    H = multiply(H, (W.T*X)/(W.T*W*H))
```

Matlab:

```matlab
for iter=1:maxiter
    W = W.*(X*H')./(W*H*H');
    H = H.*(W'*X)./(W'*W*H);
end
```
Easy to implement!

Python:

```
  for iter in range(maxiter):
    W = multiply(W, (X*H.T)/(W*H*H.T))
    H = multiply(H, (W.T*X)/(W.T*W*H))
```

Matlab:

```
  for iter=1:maxiter
    W = W.*(X*H')./(W*H*H');
    H = H.*(W'*X)./(W'*W*H);
  end
```
Example: Source Separation

kick and snare:

- [kick drum] and [snare drum]
Example: Source Separation

kick and snare:
- [kick drum] and [snare drum]

obo and horn:
- Duan et. al: [oboe] and [horn]
- Wang et. al: [oboe] and [horn]
- Tjoa and Liu: [oboe] and [horn]
Example: Source Separation

kick and snare:
  - [kick drum] and [snare drum]

oboie and horn:
  - Duan et. al: [oboe] and [horn]
  - Wang et. al: [oboe] and [horn]
  - Tjoa and Liu: [oboe] and [horn]

Vivaldi, Winter, Four Seasons:
  - [solo] and [accompaniment]
Example: Instrument Recognition

Use NMF to identify the instruments in a musical signal. Observe these atoms:
Filter the temporal atoms from NMF [Tjoa and Liu, 2010]:

- Use support vector machine (SVM) to classify the processed spectral and temporal atoms.
Feature Vector of Kick Drum
Feature Vector of Snare Drum

- Input Atom
- Attack Time (seconds)

- $n = 1.2$
- $n = 1.5$
- $n = 2.0$
- $n = 3.0$

- max

Time (seconds)
Feature Vector of Violin

\begin{align*}
\text{Attack Time (seconds)} & \\
0.020 & 0.050 & 0.100 & 0.200 & 0.400 & 0.600 & 0.800 & 1.000 \\
\end{align*}

\begin{align*}
n &= 1.2 \\
n &= 1.5 \\
n &= 2.0 \\
n &= 3.0 \\
\text{max} \\
\end{align*}

\begin{align*}
\text{Input Atom} & \\
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\end{align*}

\begin{align*}
\text{Time (seconds)} & \\
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\end{align*}
Experiments on isolated instrument sounds:

- Accuracy: **92.3%**
- Reflect state-of-the-art performance for isolated instrument recognition among as many as 24 classes.
Results: Solo Melodic Phrases

Instrument classifications. One decision per signal. Accuracy: 96.2%.
Results: Solo Melodic Phrases

Family classifications. One decision per signal.
Accuracy: 97.4%.
Existing algorithms cannot handle “complicated” music.
Related work:
- smoothness
- harmonicity
- statistical priors
Sparse Coding

What if you already have a large dictionary?

$$\min_{s} d(x, As)$$

- Solution: Impose sparsity on $s$.
- Benefits: guaranteed spectral structure; labels already known.
Related work:
- matching pursuit (MP)
- orthogonal matching pursuit (OMP)
- basis pursuit (BP)

Disadvantages:
- Complexity that is \textit{linear} in the dictionary size.
- \textbf{Neither fast nor scalable.}
Example: Orthogonal Matching Pursuit

OMP [Pati et al., 1993]:

- **Input:** \( \mathbf{x} \in \mathbb{R}^M; \mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_K] \in \mathbb{R}^{M \times K} \) s.t. \( \|\mathbf{a}_k\|_2 = 1 \) for all \( k \).
- **Output:** \( \hat{\mathbf{s}} \in \mathbb{R}^K \)
- **Initialize:** \( \mathcal{S} \leftarrow \emptyset; \mathbf{s} \leftarrow \mathbf{0}; \mathbf{r} \leftarrow \mathbf{x}; \epsilon > 0 \).
- **While** \( \|\mathbf{r}\| > \epsilon \):
  1. \( k \leftarrow \text{argmax}_j \mathbf{a}_j^T \mathbf{r} \)
  2. \( \mathcal{S} \leftarrow \mathcal{S} \cup k \)
  3. Solve for \( \{s_j|j \in \mathcal{S}\} \): \( \min_{s_j|j \in \mathcal{S}} \| \mathbf{x} - \sum_{j \in \mathcal{S}} \mathbf{a}_j s_j \| \)
  4. \( \mathbf{r} \leftarrow \mathbf{x} - \mathbf{A}s \)

- \( \hat{\mathbf{s}} \leftarrow \mathbf{s} \)
Proposed Algorithm: Approximate Matching Pursuit

AMP [Tjoa and Liu]:

- **Input:** \( \mathbf{x} \in \mathbb{R}^M; \mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_K] \in \mathbb{R}^{M \times K} \) s.t. \( \|\mathbf{a}_k\|_2 = 1 \) for all \( k \).
- **Output:** \( \hat{\mathbf{s}} \in \mathbb{R}^K \)
- **Initialize:** \( S \leftarrow \emptyset; \mathbf{s} \leftarrow \mathbf{0}; \mathbf{r} \leftarrow \mathbf{x}; \epsilon > 0. \)
- **While** \( \|\mathbf{r}\| > \epsilon \):
  1. Find any \( k \) such that \( \mathbf{a}_k \) and \( \mathbf{r} \) are near neighbors.
  2. \( S \leftarrow S \cup k \)
  3. Solve for \( \{s_j \mid j \in S\} \): \( \min_{s_j \mid j \in S} \|\mathbf{x} - \sum_{j \in S} \mathbf{a}_j s_j\| \)
  4. \( \mathbf{r} \leftarrow \mathbf{x} - \mathbf{A} \mathbf{s} \)
- \( \hat{\mathbf{s}} \leftarrow \mathbf{s} \)
Locality Sensitive Hashing

Idea: Hash nearby points into the same bin.
Experiments: Music Transcription

C Major Scale (OMP)

C Major Scale (AMP, L=8, k=8)
Experiments: Music Transcription

Debussy Clair de Lune, mm. 1-4 (OMP)

Debussy Clair de Lune, mm. 1-4 (AMP, L=8, k=8)

### Debussy Clair de Lune, mm. 1-4 (OMP)

### Debussy Clair de Lune, mm. 1-4 (AMP, L=8, k=8)
Experiments: Music Transcription

Debussy Clair de Lune, mm. 5-8 (OMP)

Debussy Clair de Lune, mm. 5-8 (AMP, L=8, k=8)
Execution times in seconds.

<table>
<thead>
<tr>
<th>Song</th>
<th>OMP</th>
<th>$\text{AMP}_{8,8}$</th>
<th>$\text{AMP}_{10,10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-major scale</td>
<td>81.05</td>
<td>43.63</td>
<td>21.03</td>
</tr>
<tr>
<td>Debussy mm. 1-4</td>
<td>118.57</td>
<td>88.45</td>
<td>29.01</td>
</tr>
<tr>
<td>Debussy mm. 5-8</td>
<td>123.05</td>
<td>121.73</td>
<td>121.84</td>
</tr>
</tbody>
</table>
Where to Learn More

Conferences:
- Int. Society of Music Information Retrieval (ISMIR)
- MIR Evaluation Exchange (MIREX)
- Int. Computer Music Conference (ICMC)
- IEEE Int. Conf. Audio, Speech, Signal Processing (ICASSP)
- ACM Multimedia

Journals:
- Journal of New Music Research
- Computer Music Journal
Lab 4
Lab 4: Summary

Summary:

3.1 Separate sources.
3.2 Separate noisy sources.
3.3 Classify separated sources.
Pressing the up and down arrows let you scroll through command history.

A semicolon at the end of a line simply means “suppress output”.

Type `help <command>` for instant documentation. For example, `help wavread`, `help plot`, `help sound`. Use `help` liberally!
Lab 4.1: Source Separation

1. In Matlab: Select File → Set Path.
   Select “Add with Subfolders”.
   Select /usr/ccrma/courses/mir2011/lab3skt.

2. As in Lab 1, load the file, listen to it, and plot it.
   
   ```
   [x, fs] = wavread('simpleLoop.wav');
   sound(x, fs)
   t = (0:length(x)-1)/fs;
   plot(t, x)
   xlabel('Time (seconds)')
   ```
3. Compute and plot a short-time Fourier transform, i.e., the Fourier transform over consecutive frames of the signal.

```matlab
frame_size = 0.100;
hop = 0.050;
X = parsesig(x, fs, frame_size, hop);
imagesc(abs(X(200:-1:1,:)))
```

Type `help parsesig`, `help imagesc`, and `help abs` for more information.
This step gives you some visual intuition about how sounds (might) overlap.
Lab 4.1: Source Separation

4. Let’s separate sources!
   1. `K = 2;`
   2. `[y, W, H] = sourcesep(x, fs, K);`

   Type `help sourcesep` for more information.

5. Plot and listen to the separated signals.
   1. `plot(t, y)`
   2. `xlabel('Time (seconds)')`
   3. `legend('Signal 1', 'Signal 2')`
   4. `sound(y(:,1), fs)`
   5. `sound(y(:,2), fs)`

   Feel free to replace Signal 1 and Signal 2 with Kick and Snare (depending upon which is which).
6. Plot the outputs from NMF.

```matlab
figure
plot(W(1:200,:))
legend('Signal 1', 'Signal 2')
figure
plot(H')
legend('Signal 1', 'Signal 2')
```

What do you observe from $W$ and $H$?
Does it agree with the sounds you heard?
Lab 4.1: Source Separation

7. Repeat the earlier steps for different audio files.
   - 125BOUNC-mono.WAV
   - 58BPM.WAV
   - CongaGroove-mono.wav
   - Cstrum chord_mono.wav
   ... and more.

Experiment with different values for the number of sources, $K$. Where does this separation method succeed? Where does it fail?
Begin with simpleLoop.wav. Then try others.

1. Add noise to the input signal, plot, and listen.
   
   1. \( x_n = x + 0.01 \cdot \text{randn}(\text{length}(x), 1); \)
   2. \( \text{plot}(t, x_n) \)
   3. \( \text{sound}(x_n, \text{fs}) \)
Lab 4.2: Noise Robustness

2. Separate, plot, and listen.
   
   1. `[yn, Wn, Hn] = sourcesep(xn, fs, K);`
   2. `plot(t, yn)`
   3. `sound(yn(:,1), fs)`
   4. `sound(yn(:,2), fs)`

   How robust to noise is this separation method?
   Compared to the noisy input signal, how much noise is left in the output signals?
   Which output contains more noise? Why?
Lab 4.3: Classification

Follow the K-NN example in Lab 1, but classify the separated signals.

1. As in Lab 1, extract features from each training sample in the kick and snare drum directories.

2. Train a K-NN model using the kick and snare drum samples.

```matlab
labels = [[ones(10,1) zeros(10,1)];
           [zeros(10,1) ones(10,1)]];
model_snare = knn(5, 2, 1, trainingFeatures, labels);
[voting, model_output] = knnwd(model_snare, featuresScaled)
```
3. Extract features from the drum signals that you separated in Lab 4.1. Classify them using the K-NN model that you built. Does K-NN accurately classify the separated signals? Repeat for different numbers of separated signals (i.e., the parameter $K$ in NMF).

4. Overseparate the signal using $K = 20$ or more. For those separated components that are classified as snare, add them together using $\text{sum}$. The listen to the sum signal. Is it coherent, i.e., does it sound like a single separated drum?
If you have another idea that you would like to try out, please ask me!

Please collaborate with a partner.
Together, brainstorm your own problems, if you want!
Good luck!

Music Information Retrieval in Polyphonic Mixtures

Steve Tjoa

MIR Workshop
CCRMA, Stanford University