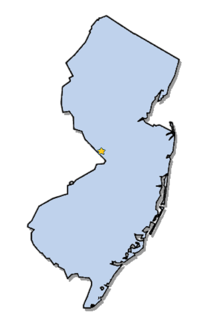
Music Information Retrieval in Polyphonic Mixtures

Steve Tjoa

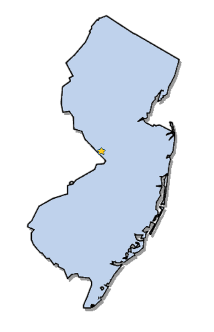
MIR Workshop
CCRMA, Stanford University
iZotope, Inc.
San Francisco, CA, USA

June 27, 2012

A bit about myself...

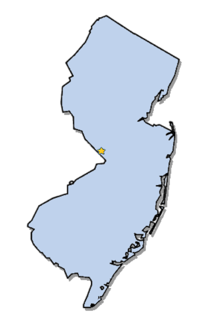


A bit about myself...



UNIVERSITY OF MARYLAND

A bit about myself...



UNIVERSITY OF MARYLAND

iZotope

Quick Review

- What are the three main components of any classification system?

Quick Review

- What are the three main components of any classification system?
- What are some useful features for MIR?

Quick Review

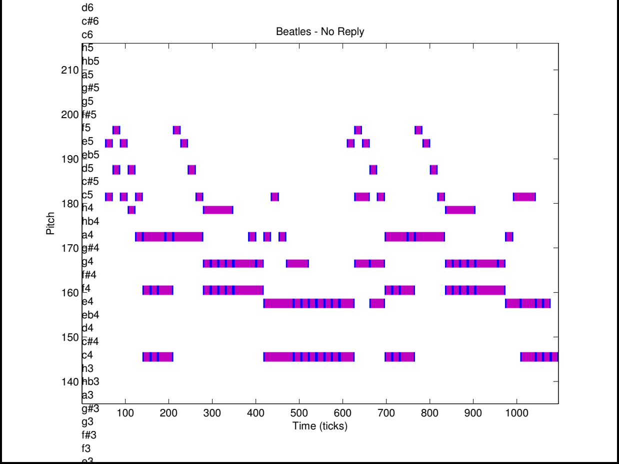
- What are the three main components of any classification system?
- What are some useful features for MIR?
- What are some problems and applications addressed by MIR?

Music Transcription

From this song...

Music Transcription

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



Music Source Separation

Isolate, amplify, or suppress a musical voice/instrument.

Example: From these beats...

Music Source Separation

Isolate, amplify, or suppress a musical voice/instrument.

Example: From these beats... isolate the kick drum and snare drum.

A Really Special Tool

Nonnegative Matrix Factorization (NMF):

- Given $\mathbf{X}$ nonnegative, find $\mathbf{W}$ and $\mathbf{H}$, both nonnegative, that minimize some distance $d(\mathbf{X}, \mathbf{WH})$.

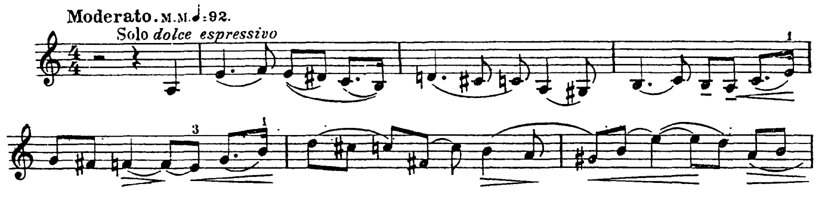
A Really Special Tool

Nonnegative Matrix Factorization (NMF):

- Given $\mathbf{X}$ nonnegative, find $\mathbf{W}$ and $\mathbf{H}$, both nonnegative, that minimize some distance $d(\mathbf{X}, \mathbf{WH})$.
- 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

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

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

$$\mathbf{w} \begin{bmatrix} a & b & c \end{bmatrix} = \begin{bmatrix} a\mathbf{w} & b\mathbf{w} & c\mathbf{w} \end{bmatrix}$$

Brief Refresher: Matrix Multiplication

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

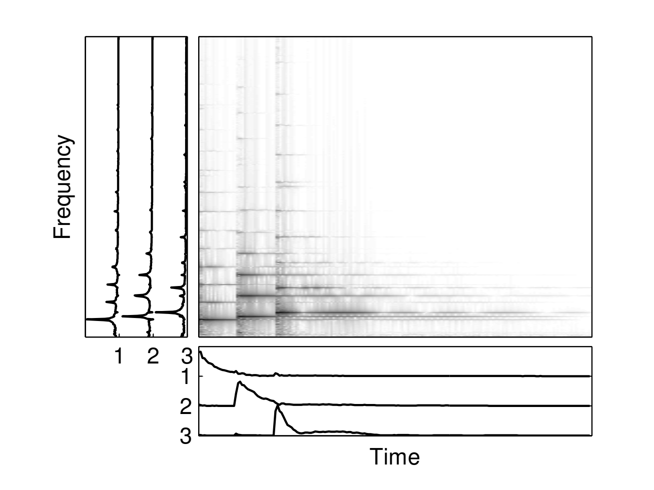
$$\begin{bmatrix} 3 \\ 4 \end{bmatrix} \begin{bmatrix} a & b & c \end{bmatrix} = \begin{bmatrix} 3a & 3b & 3c \\ 4a & 4b & 4c \end{bmatrix}$$

$$\mathbf{w} \begin{bmatrix} a & b & c \end{bmatrix} = \begin{bmatrix} a\mathbf{w} & b\mathbf{w} & c\mathbf{w} \end{bmatrix}$$

$$\begin{bmatrix} 3 \\ 4 \end{bmatrix} \mathbf{h} = \begin{bmatrix} 3\mathbf{h} \\ 4\mathbf{h} \end{bmatrix}$$

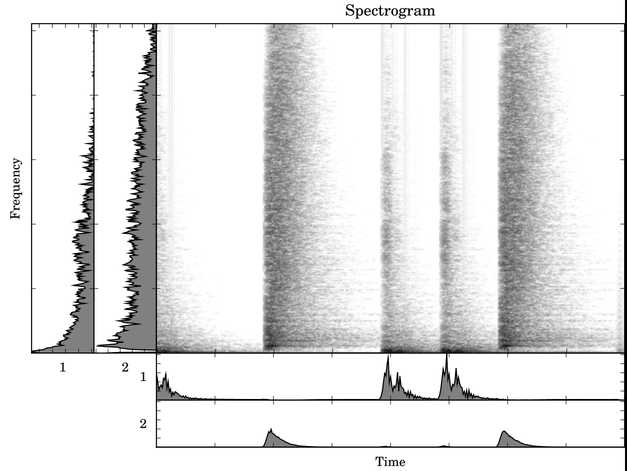
Nonnegative Matrix Factorizaton

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



Nonnegative Matrix Factorizaton

Top right: **X**. Left: **W**. Bottom: **H**. Kick and snare:



NMF Algorithms

Multiplicative update rules:

$$\mathbf{W} \leftarrow \mathbf{W} \cdot \frac{\mathbf{X}\mathbf{H}^T}{\mathbf{W}\mathbf{H}\mathbf{H}^T} \qquad \mathbf{H} \leftarrow \mathbf{H} \cdot \frac{\mathbf{W}^T\mathbf{X}}{\mathbf{W}^T\mathbf{W}\mathbf{H}}$$

See [Lee and Seung, NIPS 2001].

NMF Algorithms

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))
```

NMF Algorithms

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]

oboe 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]

oboe and horn:

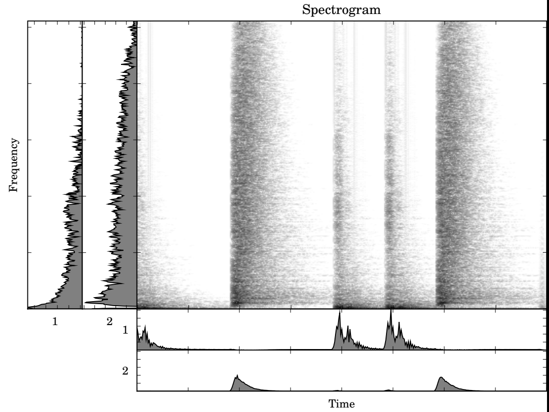
- Duan et. al: [oboe] and [horn]
- Wang et. al: [oboe] and [horn]
- Tjoa and Liu: [oboe] and [horn]

Vivaldi, *Winter, Four Seasons*:

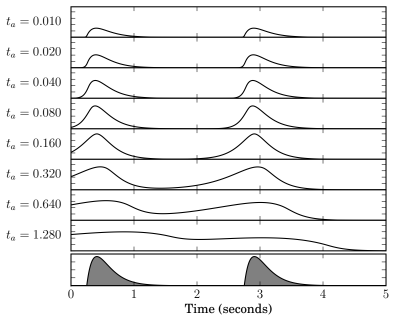
- [solo] and [accompaniment]

Example: Instrument Recognition

Use NMF to identify the instruments in a musical signal.
Observe these atoms:

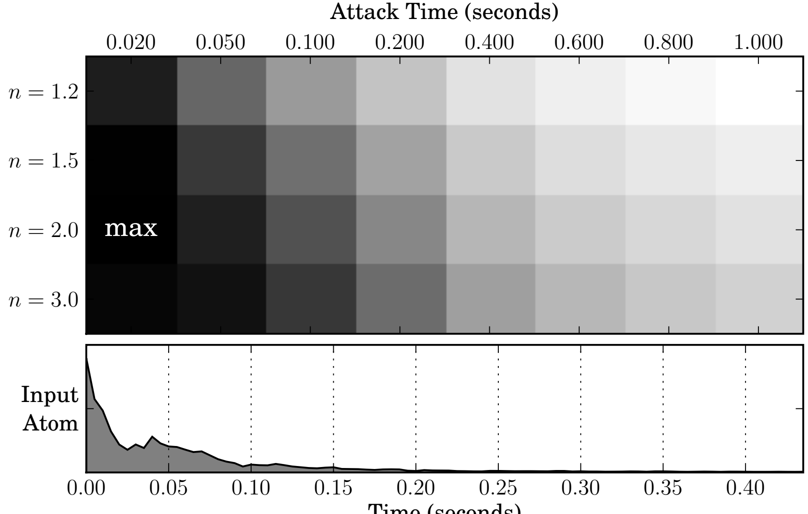


Filter the temporal atoms from NMF [Tjoa and Liu, 2010]:

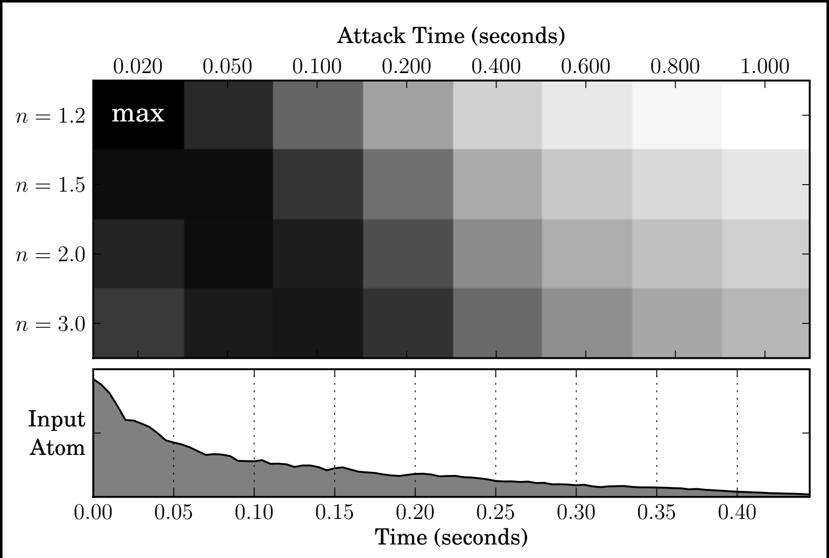


- Use support vector machine (SVM) to classify the processed spectral and temporal atoms.

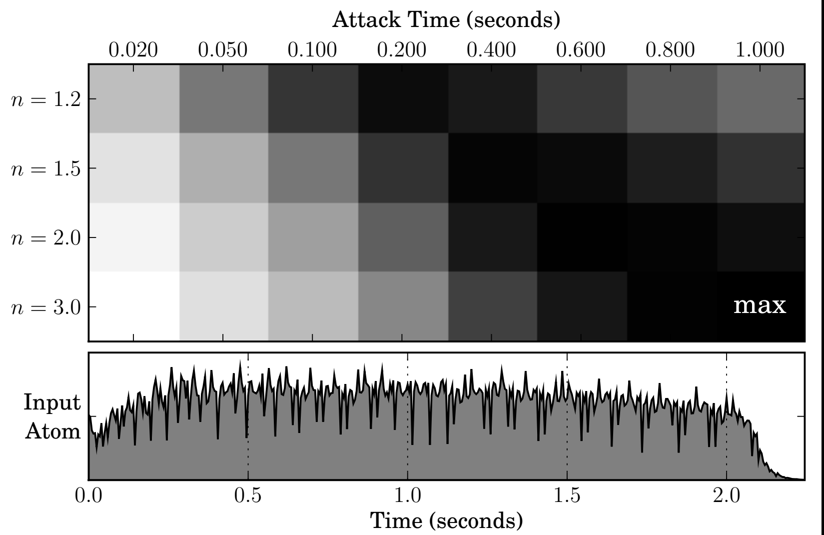
Feature Vector of Kick Drum



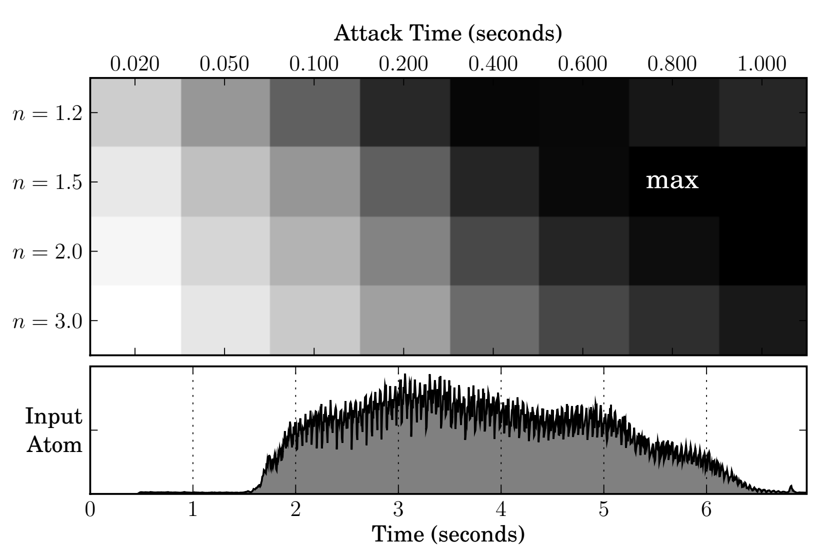
Feature Vector of Snare Drum



Feature Vector of Trumpet



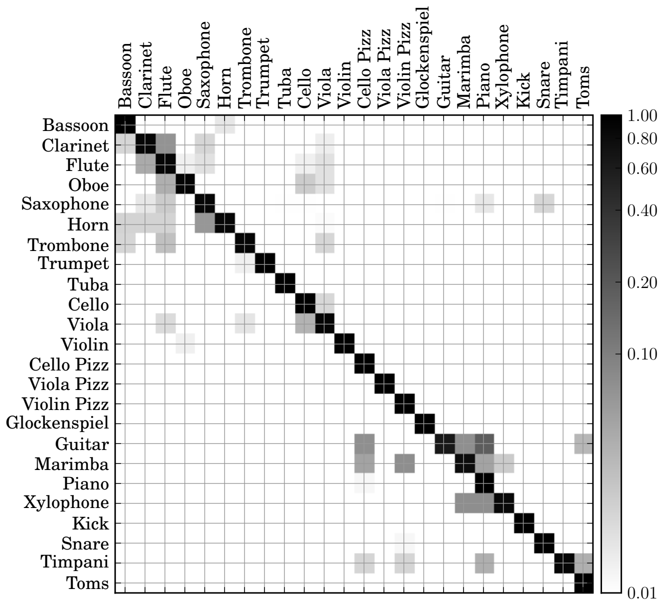
Feature Vector of Violin



Results: Isolated Instrument Recognition

Experiments on isolated instrument sounds:

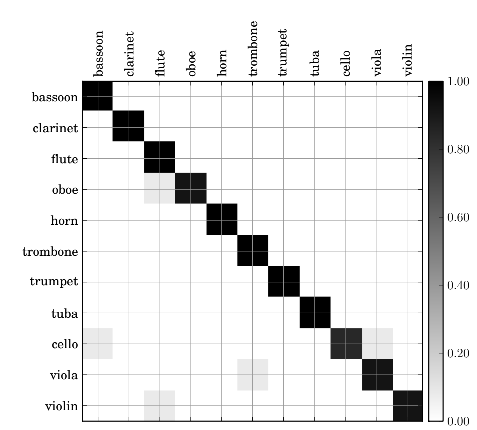
- Accuracy: **92.3%**
- Reflect state-of-the-art performance for isolated instrument recognition among as many as 24 classes.



Bassoon
Clarinet
Flute
Oboe
Saxophone
Horn
Trombone
Trumpet
Tuba
Cello
Viola
Violin
Cello Pizz
Viola Pizz
Violin Pizz
Glockenspiel
Guitar
Marimba
Piano
Xylophone
Kick
Snare
Timpani
Toms
1.00
0.80
0.60
0.40
0.20
0.10
0.01

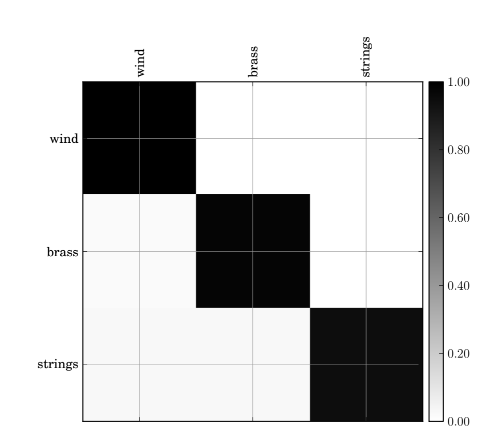
Results: Solo Melodic Phrases

Instrument classifications. One decision per signal.
Accuracy: **96.2%**.



Results: Solo Melodic Phrases

Family classifications. One decision per signal.
Accuracy: **97.4**%.



Current and Future Work

Existing algorithms cannot handle "**complicated**" music.

Related work:

- smoothness
- harmonicity
- statistical priors

Sparse Coding

What if you already have a large dictionary?

$$\min_{\mathbf{s}} d(\mathbf{x}, \mathbf{A}\mathbf{s})$$

- Solution: Impose sparsity on $\mathbf{s}$.
- Benefits: guaranteed spectral structure; labels already known.

Sparse Coding

Related work:

- matching pursuit (MP)
- orthogonal matching pursuit (OMP)
- basis pursuit (BP)

Disadvantages:

- Complexity that is **linear** in the dictionary size.
- **Neither fast nor scalable.**

Example: Orthogonal Matching Pursuit

OMP [Pati et al., 1993]:

- Input: $\mathbf{x} \in \mathbb{R}^M$; $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_K] \in \mathbb{R}^{M \times K}$ 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. $\mathbf{r} \leftarrow \mathbf{x} - \mathbf{A}\mathbf{s}$
- $\hat{\mathbf{s}} \leftarrow \mathbf{s}$

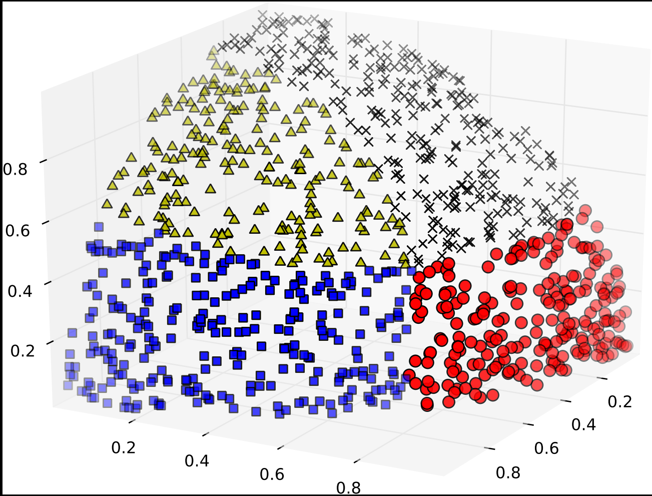
Proposed Algorithm: Approximate Matching Pursuit

AMP [Tjoa and Liu]:

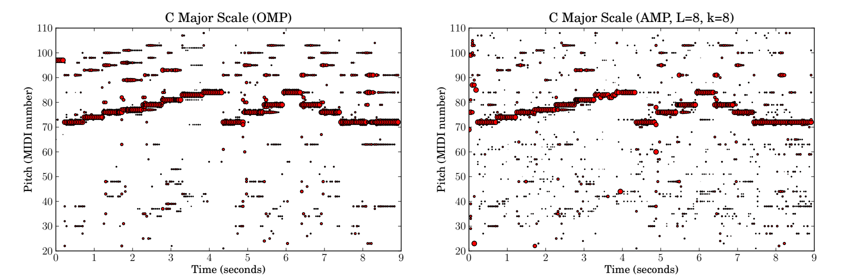
- Input: $\mathbf{x} \in \mathbb{R}^M$; $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_K] \in \mathbb{R}^{M \times K}$ s.t. $||\mathbf{a}_k||_2 = 1$ for all k.
- Output: $\hat{\mathbf{s}} \in \mathbb{R}^K$
- Initialize: $\mathcal{S} \leftarrow \emptyset$; $\mathbf{s} \leftarrow \mathbf{0}$; $\mathbf{r} \leftarrow \mathbf{x}$; $\epsilon > 0$.
- While $||\mathbf{r}|| > \epsilon$:
  1. Find any k such that $\mathbf{a}_k$ and $\mathbf{r}$ are near neighbors.
  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. $\mathbf{r} \leftarrow \mathbf{x} - \mathbf{A}\mathbf{s}$
- $\hat{\mathbf{s}} \leftarrow \mathbf{s}$

Locality Sensitive Hashing

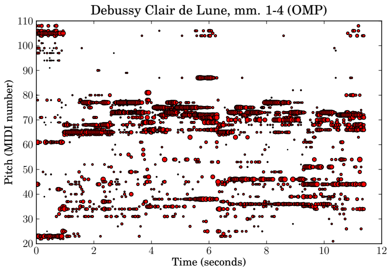
Idea: Hash nearby points into the same bin.

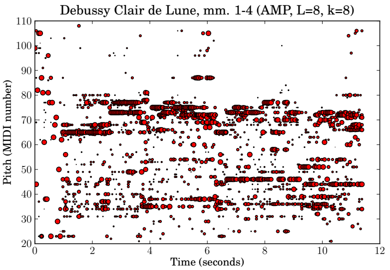


Experiments: Music Transcription

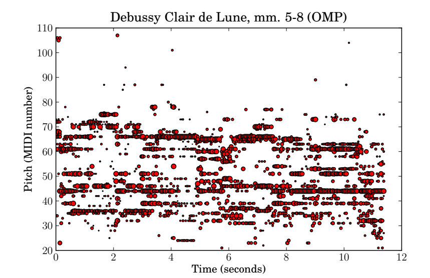


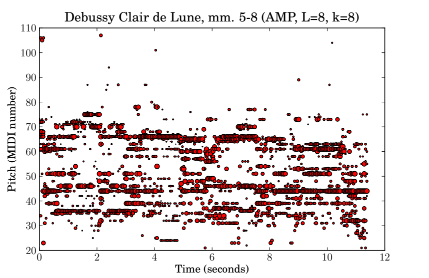
Experiments: Music Transcription

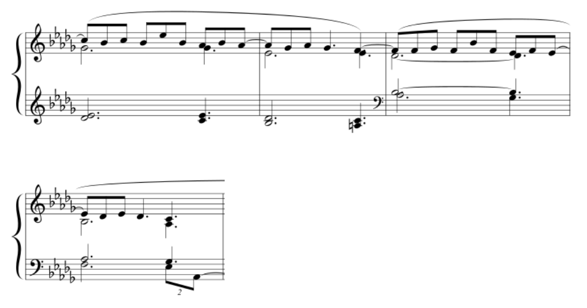




Experiments: Music Transcription







Experiments: Music Transcription

Execution times in seconds.

Song	OMP	$AMP_{8,8}$	$AMP_{10,10}$
C-major scale	81.05	43.63	21.03
Debussy mm. 1-4	118.57	88.45	29.01
Debussy mm. 5-8	123.05	121.73	121.84

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:

- IEEE Trans. Audio, Speech, Language, Processing
- Journal of New Music Research
- Computer Music Journal

Lab 3

Lab 3: Summary

Summary:

3.1 Separate sources.

3.2 Separate noisy sources.

3.3 Classify separated sources.

Lab 3: Matlab Programming Tips

- 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 3.1: 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)')
```

Lab 3.1: Source Separation

3. Compute and plot a short-time Fourier transform, i.e., the Fourier transform over consecutive frames of the signal.

   ```
   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 3.1: Source Separation

4. Let's separate sources!

```
K = 2;
[y, W, H] = sourcesep(x, fs, K);
```

   Type `help sourcesep` for more information.

5. Plot and listen to the separated signals.

```
plot(t, y)
xlabel('Time (seconds)')
legend('Signal 1', 'Signal 2')
sound(y(:,1), fs)
sound(y(:,2), fs)
```

   Feel free to replace `Signal 1` and `Signal 2` with `Kick` and `Snare` (depending upon which is which).

Lab 3.1: Source Separation

6. Plot the outputs from NMF.

```
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 3.1: 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?

Lab 3.2: Noise Robustness

Begin with `simpleLoop.wav`. Then try others.

1. Add noise to the input signal, plot, and listen.

```
xn = x + 0.01*randn(length(x),1);
plot(t, xn)
sound(xn, fs)
```

Lab 3.2: Noise Robustness

2. Separate, plot, and listen.

   ```
   [yn, Wn, Hn] = sourcesep(xn, fs, K);
   plot(t, yn)
   sound(yn(:,1), fs)
   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 3.3: 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.

```
labels=[[ones(10,1) zeros(10,1)];
        [zeros(10,1) ones(10,1)]];
model_snare =
  knn(5, 2, 1, trainingFeatures, labels);
[voting, model_output] =
  knnfwd(model_snare, featuresScaled)
```

Lab 3.3: Classification

3. Extract features from the drum signals that you separated in Lab 3.1.
   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 `sum`. The listen to the sum signal. Is it coherent, i.e., does it sound like a single separated drum?

...and more!

- 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
iZotope, Inc.
San Francisco, CA, USA