From Bases to Exemplars, from Separation to Understanding

Paris Smaragdis – paris@illinois.edu
Some Motivation

- Why do we separate signals?
  - I don’t really know ...

- Is there an all-conquering algorithm?
  - I suspect, not really ...

- So why are you working on both of the above Paris?
  - It’s a good exercise for what’s to come
Outline

- Low-rank models
  - Learning to listen

- Nearest subspace approaches
  - Using data, not fancy algorithms

- Taking advantage of semantic information
  - Explaining mixtures, not decomposing them
How do we deal with mixtures?

- We find coherent structure
  - We can mimic humans
  - Or we can use statistics
Learning what to separate

- We cannot make up data, we need something to learn from

- **Original speech**
- **Original interference**
- **Mixture**
- **Training speech**
- **Training interference**
- **Unknown data**
- **Observed data**

Diagram showing the process of separating original speech from interference.
Describing sounds

- A probabilistic interpretation of the spectrum
  - Why?
    - We don’t care for scale and phase
    - Allows us to perform sophisticated reasoning
The space we deal with

- These distributions live in a simplex
The space we deal with

- These distributions live in a simplex
Modeling one sound

- Use a dictionary representation

\[ P_t(f) = \sum_z P(f \mid z) P_t(z) \]

- \( z \) is the index of the dictionary
- Everything is a “distribution”
- We can estimate dictionary/weights using EM
For the matrix inclined

- It’s a linear transform

\[ P_t(f) = \sum_{z} P(f \mid z) P_t(z) \]

\[ f_t = f \cdot z_t \]

Input spectra

Dictionary elements

Weights
Huh?
Represented as frequency distributions

- Each column is normalized
  - Each column is now $P_t(f)$
A 2-element dictionary approximation

\[ P_{i}(f) \]

\[ P(f | z) \]

\[ P_{i}(z) \]
Complex sounds = large dictionaries

- Frequency distributions capture spectral character
Or we can see this as

- Different areas of the simplex are different “sounds”
- Learned dictionary elements form convex hulls around them
Modeling mixtures

- A mix of two normed spectra lies on connecting subspace
Huh again?

Learn speech dictionary

Learn chime dictionary

Speech and Chimes

Extracted speech

Extracted chimes

Keep fixed

Mixture of speech and chimes

Learn only the weights
A problem

- Convex hulls are a bad idea, sounds can overlap
Nearest subspace search

- Search for all possible solutions given training data
  - i.e. exemplars training
The bad news

- **Very high computational complexity**
  - $M^N$ searches per query
  - For $N$ sources and $M$ training data points
  - 8 min, 5 sources $\rightarrow$ 206,719 training data points
  - $206,719^5 = 377,486,980,238,462,848,824,329,599$ searches
    - For each input spectrum!

- **Approximate algorithms**
  - Somewhat faster search, unrealistic memory requirements
    - A few Petabytes
Avoiding the search

- We can still use the previous model

\[ P_t(f) = \sum_z P(f \mid z) P_t(z) \]

- If we force weights \( P_t(z) \) to be sparse we approximate the nearest subspace search

Input spectra

Consolidated training data

Sparse weights
Enforcing sparsity

- **The hard way: Entropic priors**
  - We can tune each distribution’s entropy
  - For sparse $P_t(z)$ we minimize its entropy
    - Pain in the @#$!

- **The easy way: Maximum $\ell_2$-norm**
  - Since $0 \leq P_t(z) \leq 1$ max $\ell_2$-norm results in sparsity
  - Corresponds to Simpson’s diversity index

- **Both plug seamlessly in EM estimation**
Computation gains

- Proposed method is substantially faster for a realistic number of training data ( > 1,000)
How this looks

- Finds points whose connecting subspace passes closest to the observed mixture point

*Using learned dictionaries*

- Source A
- Source B
- Mixture
- Convex Hull A
- Convex Hull B
- Estimate for A
- Estimate for B
- Approximation of mixture

*Using exemplars*

- Source A
- Source B
- Mixture
- Estimate for A
- Estimate for B
- Approximation of mixture
And some results

- **TIMIT speech mixes**
  - \( \approx 20\text{dB} \) SIR on average
  - \( \approx 30\text{dB} \) SIR with post-process

- **Exemplars beat dictionaries**
  - By a lot!

![Signal to Interference Ratio](chart1.png)

![Signal to Distortion Ratio](chart2.png)
A practical extension

- We can’t know all sources
  - But we usually know one (target or interference)

- All mixing problems are binary
  - Target vs. all else

- We need to learn extra bases
  - Describe all that we don’t know
  - Straightforward extension
In practice

- **Selective parameter updates**

\[
P_t(f) = P_t(s_1) \sum_z P(f \mid z, s_1) P_t(z \mid s_1) + P_t(s_2) \sum_z P(f \mid z, s_2) P_t(z \mid s_2)
\]

Learn extra frequency elements that explain other sounds

Target or interference example

Keep fixed

Mixture recording

Learn the weights

Mixtures and extracted sources
Fun things to do

Original drum loop

Extracted layers

No tambourine

No congas

Congas!

Remixer

Music layer

Selective pitch shifting

Voice layer

Original drum loop

Soprano layer

Remixed layers

Piano + Soprano

Soprano layer

Remixed layers

Piano layer
More fun things

Smart Audio User Interfaces
Paris Smaragdis, University of Illinois
Gautham Mysore, Adobe Systems Inc.
But the objective is not to separate!!

- Source separation is a useless pursuit
  - There is almost never a reason to separate

- The real holy grail:
  - Understand mixtures, don’t separate them

- Harder proposition, and rather unexplored
Making Direct Use of Exemplars

- Polyphonic pitch tracking
  - Difficult mixture problem

- Some observations
  - It can be learned, it shouldn’t be user-specified

- We can adapt what we’ve done to do so
  - We should avoid to separate!
Mono pitch tracking by example

**Training data**

\[ f_{x_1}(t) \]

**Data to pitch track**

\[ f_{y_1}(t) \]
Representation and matching

- **Nearest Neighbor match**
  - Find closest spectrum
  - Use neighbor’s pitch tag

- **Normalized warped spectrograms**
  - Provide gain invariance
  - Clarify harmonic structure
How well does that work?

- **Proposed pitch tracking accuracy**
  - Error mean $\mu = 0.02$ Hz
  - Error deviation $\sigma = 1.1$ Hz

- **With popular pitch trackers**
  - Error mean $\mu = 0.1$ Hz
  - Error deviation $\sigma = 1.2$ Hz

- **So we’re on to something**
  - But ...
The polyphonic case

- Nearest neighbors are insensitive to additivity
  - Therefore can’t resolve mixture sounds
- For mixtures we have to search for the nearest subspaces
  - Aha! We know how to do that!

**Nearest Neighbor Search**

**Nearest Subspace Search**

- Untagged input
- Pitch-tagged data
Dealing with a duet

- Training on two instruments

\[ f_{x_1}(t) \]

\[ f_{x_2}(t) \]

\[ f_{y_1}(t) + f_{y_2}(t) \]
Duet results

- Still works well
  - Pitch error stats: \( \mu = 0.003 \text{ Hz}, \sigma = 2.05 \text{ Hz} \)
A more beefy example

- **Wind quintet recording**
  - Bassoon, Clarinet, Flute, Horn, Oboe

- **Training data**
  - 7m41s per source → 198,535 training vectors
  - Removing unpitched vectors → 50,000 training vectors

- **Test data**
  - 1m10s of simultaneous performance → 6,000 input spectra
  - Data tested as duet, trio, quartet and quintet
Ratio of true vs. estimated pitch

Duet $p_y/\hat{p}_y$

- $\mu = -18.5$ Hz
- $\sigma = 89.5$ Hz
- $t = 37$ sec

Trio $p_y/\hat{p}_y$

- $\mu = -32.6$ Hz
- $\sigma = 109.5$ Hz
- $t = 49$ sec

Quartet $p_y/\hat{p}_y$

- $\mu = -28.7$ Hz
- $\sigma = 1.6$ Hz
- $t = 60$ sec

Quintet $p_y/\hat{p}_y$

- $\mu = -42.8$ Hz
- $\sigma = 112.8$ Hz
- $t = 114$ sec
Most errors are “human”
- Transition problems
- Occasional confusion with other instruments

Correct over majority samples of a note
What happened here?

- **Input:**
  - Some listening experience
  - Mixture of five sounds

- **Output:**
  - Pitch values for each instrument (dictionary elements used)
  - Kind of instrument (dictionary elements again)
  - Amplitude of each source (presence of these elements)

- **What more is there to do?**
  - No need to separate
“Human”-ish side effects

- Graceful degradation with increasing number of sources
  - Duets easier than trios, easier than quartets, ...

- Can “pitch track” pitch-less sounds
  - Inharmonic, quasi-periodic, etc. ...

- The more you know the better you do
A more realistic take

- Just as before, we can’t know everything
  - But we know something

- Semi-exemplar learning
  - Mix exemplar model with basis decomposition

- Applies to target/background cases
  - Which are most of the interesting cases anyway
Step 1. Learn the target source

- Like before, each exemplar comes with feature labels
  - In this case pitch, can also be phoneme, stress, etc.

- We also learn temporal dynamics
  - How exemplars/bases appear in time
  - Also can apply for features too

- Use a transition matrix for $z$

$$P(z_{t+1} \mid z_t)$$

Target dictionary $P(f \mid z)$

Transition matrix $P(z_{t+1} \mid z_t)$
Step 2. Learn the rest from a mixture

- Keep target exemplars fixed
  - Adapt a new set of bases, while obeying transitions

- Explain mixture as target + “rest”
  - We don’t care about “rest” accuracy

- Use pitch from exemplars
  - Same as before
Example pitch tracking

- Results in very accurate target following
  - In a challenging and highly correlated case

Estimated $P_t(a) P_t^{(a)}(q)$ with $C = 0.0015$
Delving deeper in temporal dynamics

- Previous model was a linear predictor of sorts
  - Short-term effects, minimal structure

- Extending this idea to stricter models
  - Hidden Markov Model formulation

- Can come in many flavors
  - Markov Model Selection
  - Non-Negative HMMs
  - ...

One last example

- **Structured speech mixtures**
  - Each speaker follows a language model
    - i.e. we hear words in sentences that make sense

- **Use an HMM of course**
  - Encode domain structure knowledge
  - Structured model replaces exemplars
The “non-negative” HMM

- Temporal model using exemplars/bases
Non-factorial learning

- State model additivity results in decoupled chains
  - Fast state estimation, doesn’t require factorial model
- Previous extensions apply
Results on the Speech Separation Challenge

- Yes, we can separate, but we don’t have to!
  - HMM state paths transcribe speech
  - Results are quite competitive

![Bar Chart]

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Speaker 2</th>
<th>Speaker 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-6dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-9dB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clean</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
My parting messages

- Don’t separate!
  - Separation algorithms are laying the foundation for mixed signal processing and analysis, treat them as such!
My parting messages

- Don’t separate!
  - Separation algorithms are laying the foundation for mixed signal processing and analysis, treat them as such!

- Keep separating!
  - We’re learning a ton of new things, that’s great! 😊
References