Music Information Retrieval in Python

Steve Tjoa
Tuesday, October 21, 2014
Hackbright Academy
Acknowledgements

- Angie Chang, Wendy Dherin, Hackbright Academy
- Jay LeBoeuf (realindustry.org, iZotope, Imagine Research)
- Owen Campbell (Humtap, UCSB)
## Music recommendation

<table>
<thead>
<tr>
<th>Name</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>M0112708_06 Overkill.wav</td>
<td>2:12</td>
</tr>
<tr>
<td>M0112707_05 Blood Lust.wav</td>
<td>2:12</td>
</tr>
<tr>
<td>M0112705_04 Napalm Blitz.wav</td>
<td>2:12</td>
</tr>
<tr>
<td>M0112711_08 Demolition Barbie.wav</td>
<td>2:23</td>
</tr>
<tr>
<td>M0112702_01 Axephetamine.wav</td>
<td>2:34</td>
</tr>
<tr>
<td>M0112703_02 Dimebag Damage.wav</td>
<td>2:25</td>
</tr>
<tr>
<td>Mermaid in Japan</td>
<td>5:06</td>
</tr>
<tr>
<td>M0112953_10 Hallowed By Thy Flame.wav</td>
<td>2:46</td>
</tr>
<tr>
<td>M0112713_09 Headlong Heracy.wav</td>
<td>3:01</td>
</tr>
<tr>
<td>M0112716_11 No Holds Barred.wav</td>
<td>2:37</td>
</tr>
<tr>
<td>M0112717_12 Billy Whizz.wav</td>
<td>2:40</td>
</tr>
<tr>
<td>M0112996_01 The Beast.wav</td>
<td>2:27</td>
</tr>
<tr>
<td>M0112704_03 Terrorize.wav</td>
<td>2:35</td>
</tr>
<tr>
<td>M0113007_07 Speed.wav</td>
<td>2:28</td>
</tr>
<tr>
<td>Bad Attraction – Earjamm Mix (Hipcola)</td>
<td>5:35</td>
</tr>
<tr>
<td>Show Me Fear</td>
<td>3:59</td>
</tr>
<tr>
<td>M0113004_05 Slow Death.wav</td>
<td>2:04</td>
</tr>
<tr>
<td>I Am</td>
<td>4:59</td>
</tr>
<tr>
<td>M0112544_15 Fastball Special.wav</td>
<td>3:51</td>
</tr>
<tr>
<td>Whispers and Knives (Yongen)</td>
<td>5:45</td>
</tr>
</tbody>
</table>
stanford-mir

Instructional material for the Music Information Retrieval Workshop at CCRMA, Stanford University, 2014.

How to Use This Repo

This repo contains a bunch of IPython notebooks related to music information retrieval.

If you're a visitor, browse this repo at nbviewer.ipython.org which renders the notebooks in this repo so they can be viewed in a web browser.

If you're a workshop participant, follow the steps below to get started with Git, Vagrant, and IPython.

For GitHub repository, see https://github.com/stevetjoa/stanford-mir.
A bit about me
Humtap is a collaborative music-making app for everyone.

Make real music instantly. Just hum and tap.
What is MIR?
• fingerprinting
• cover song detection
• genre recognition
• transcription
• recommendation
• symbolic melodic similarity
• onset detection
• mood
• source separation
• instrument recognition
• pitch tracking
• tempo estimation
• score alignment
• song structure/form
• beat tracking
• key detection
• query by humming
• query by tapping
Why Python?

• Because it’s not Matlab.
Why Python?

• free
• general purpose
• nice syntax
• easy to learn
• fast to develop

• popular
• lots of libraries
• good for signal processing (NumPy, SciPy)
• high demand
MIR System Architecture

Audio

Segmentation; Preprocessing → Feature Extraction → Machine Learning

Musical Information
Onset segmentation → Analysis frame
ZERO CROSSING RATE

Zero crossing rate = 9
IPython: Interactive Computing

IPython provides a rich architecture for interactive computing with:

- Powerful interactive shells (terminal and Qt-based).
- A browser-based notebook with support for code, text, mathematical expressions, inline plots and other rich media.
- Support for interactive data visualization and use of GUI toolkits.
- Flexible, embeddable interpreters to load into your own projects.
- Easy to use, high performance tools for parallel computing.
ABSTRACT
Factorization of polyphonic musical signals remains a difficult problem due to the presence of overlapping harmonics. Existing dictionary learning methods cannot guarantee that the learned dictionary atoms are semantically meaningful. In this paper, we explore the factorization of harmonic musical signals when a fixed dictionary of harmonic sounds is already present. We propose a method called approximate matching pursuit (AMP) that can efficiently decompose harmonic sounds by using a known predetermined dictionary. We illustrate the effectiveness of AMP by decomposing polyphonic musical spectra with respect to a large dictionary of instrumental sounds. AMP executes faster than orthogonal matching pursuit yet performs comparably based upon recall and precision.

1. INTRODUCTION
Dictionary learning, sparse coding, and constrained factorization algorithms have recently revolutionized the way we perform music transcription and source separation. Many researchers have reported success when decomposing simple musical signals using nonnegative matrix factorization (NMF) [23] or methods based upon sparse coding such as K-SVD [1,2]. Unfortunately, problems remain for intricate, polyphonic musical signals. When musical notes overlap in time and frequency, the separation and transcription performance of these basic dictionary learning methods diminishes rapidly. In such a case, the algorithm will usually learn a dictionary where each individual atom contains information from multiple musical sources, thus hindering our attempts at decomposition.

Researchers have slowly improved upon the original dictionary learning methods by adding constraints to the learning process. By restricting the dictionary atoms to reside within a predetermined feasible set, we can ensure that the learned atoms will be useful at the conclusion of the learning process. For example, existing solutions include adding constraints to the dictionary learning process such as harmonicity [3, 25] or smoothness [3, 26].

Another solution is to add structure to the dictionary. For example, one can construct and use a large, predefined, overcomplete dictionary where each atom is already labeled and assumed to contain information from only one musical source. Instead of learning an optimal dictionary for a given musical signal, it may suffice to match the signal to this large set of precomputed, labeled dictionary atoms. Then, by decomposing a signal with respect to this fixed dictionary, classification is easily achieved by simply reading the label of the atom. As musical databases become more available, construction of predefined dictionaries will become easier, thus reducing the need for adaptive dictionary learning.

Of course, the performance of such an algorithm depends upon the breadth of the dictionary. When atoms from more musical sources are added to the dictionary, the dictionary’s ability to decompose polyphonic music will improve. However, dictionary growth introduces concerns related to scalability and computational complexity. While the aforementioned algorithms have significantly advanced the state of the art, they remain slow and difficult to scale as the dictionary size increases. Most of the original factorization methods such as matching pursuit (MP) [18] and NMF with multiplicative updates [17] have complexity that is linear in the size of the dictionary. As a result, when dictionary sizes grow, the transcription efficiency of these algorithms diminishes.

To summarize the problem: how can we make use of a large, precomputed, overcomplete dictionary to factorize overlapping harmonic sounds accurately and efficiently? We address this problem by proposing a variant of MP called approximate matching pursuit (AMP). Unlike MP and NMF, AMP can decompose signals into a sparse combination of atoms with complexity that is sublinear in the dictionary size while maintaining accuracy. To do this, AMP uses...
For more...
• IEEE ICASSP
• IEEE Trans. Audio Speech Language Processing
• ACM Multimedia
• Computer Music Journal
• Journal of New Music Research
• NumPy
• SciPy
• IPython notebook
• scikit-learn
What is the range of a viola?

• As far as you can kick it.