Musical Instrument Modeling and Classification

Christopher N. Copeland, Sameep Mehrotra
CS 229: Machine Learning, Stanford University
chrisnc@stanford.edu, sameep13@stanford.edu

Abstract

Independent Components Analysis is commonly used to separate different sources of sounds in a set of recordings. ICA produces the original unmixed input signals as output, but not in a known order. It is up to the listener to identify or interpret the sounds separately. This task is easy for a human who is familiar with the nature of the sounds being mixed (e.g., human voices), but may be more difficult if the sounds are not already known to the speaker. We propose and implement a technique to label the unmixed sounds produced by ICA. Specifically, we were able to use a set of Support Vector Machines trained to identify each of four different musical instruments. To find a feature mapping the SVMs can use to label arbitrary sounds beyond those similar to its training set, we used "Yet Another Audio Feature Extractor" or Yaafe, an open-source audio feature toolkit [1]. We tested our instrument SVMs on randomly generated sounds from known instruments and found that, using the appropriate set of features from Yaafe, we obtained a 95% prediction accuracy. We also tested the instrument SVMs on the outputs of ICA, which have some lingering components of other instruments mixed in due to an imperfect unmixing matrix, and found a prediction accuracy of 75%.

I. Overview

Independent Component Analysis is an excellent tool for analyzing and separating multiple sources of sounds using recordings from different locations in a room. Often, the resulting sounds from this separation are readily useful just by listening to them. Speech, for example, is easy to discern after ICA if the unmixing matrix is sufficiently accurate. For some tasks, though, further processing after ICA can give valuable information or otherwise save humans the trouble of finding information from the sound samples by directly listening to them. One such example is the task of identifying the instrument that produced each of the unmixed tracks found from ICA. ICA normally does not attempt to produce any particular ordering of the unmixed sound outputs, so ICA alone is not sufficient for this task.

To solve this problem, we used multiple Support Vector Machines, with each trained to classify one of the instruments we used in our data sets. A prediction consists of determining which SVM determines that a sound sample belongs in its class, and returning an error if none, or more than one, of the SVMs give a positive result. We used the LIBLINEAR to create and train these SVMs [4].

In our task of classifying instruments given only short sound samples, we needed to choose features that would provide enough information to uniquely identify each instrument, without containing too much (or any, if possible) information about the notes being played, as these will generally be different between any two sound samples. In particular, if features pertaining to the notes being played are captured by the feature mapping, then an instrument playing some set of notes is more likely to be misclassified as a different instrument, if that instrument happened to play similar notes in the training data set. In order to solve this problem, we looked at many different features available in Yaafe, and tested our SVMs using different subsets of these features. We found that classifying randomly generated sounds using the appropriate subset of features gives very good performance, while using features that depend heavily on the notes played in

a particular sample leads to very poor performance, even used in conjunction with the highly predictive features.

II. Sound Generation

To produce our sound data, we generated multiple waveforms for each of four instruments (clarinet, mandolin, saxophone, and sitar). These waveforms consist of randomly generated "music" in the form of chords, single notes, and rests of varying lengths interspersed together. The notes were randomly selected from the one octave of 12 semitone notes between C4 (middle C) and B5, inclusive. We limited the chords to just two or three simultaneous notes, while varying the length of the notes between roughly 100ms and one second, depending on the instrument. Each waveform is roughly eight seconds of sound data.

This data was generated in the ChucK programming language, which is a strongly-typed, strongly-timed concurrent audio and multimedia programming language created by Ge Wang of the Stanford Center for Computer Research in Music and Acoustics (CCRMA) [2]. The instrument models in ChucK are based off of those found in Perry Cook and Gary Scavone's Synthesis Took Kit (STK), which is a set of open source audio signal processing and algorithmic synthesis tools written in C++. The STK instrument classes utilize the method of digital waveguide synthesis, which is a form of physical modeling synthesis in which a mathematical model of the instrument is approximated based on physical features of that instrument. Digital waveguides are efficient computational models for physical media through which acoustic waves propagate. For example, the STK mandolin uses two "twang" models and commuted synthesis techniques in order to model the mandolin instrument.

III. Feature Selection

Our intuition about the nature of sound initially suggested that a frequency representation is most helpful for identifying the properties of a musical instrument. The strongest features in a given sound sample will be the frequencies of the note being played (the fundamental frequency), and the other strong features will describe the harmonics associated with the instrument playing the note. For classifying arbitrary sequences of sounds, however, this is not sufficient, as the frequency profile of a sound sample depends heavily on the fundamental frequencies being played, and not just the instrument playing them, which primarily influences the harmonics.

A useful property of this feature mapping is that it is time-invariant. Specifically, because we are taking the magnitude of the Fourier transform, starting a sound sample with some delay t simply multiplies the Fourier domain representation by, $\exp(-2\pi it)$, which has a magnitude of 1, and therefore the representation will be the same. This means that, ideally, we should be able to detect a particular profile regardless of when the note begins in any given sample. Despite other drawbacks of using a frequency representation, any other ideal feature mapping should also have this property. Shifting sound samples in time should not affect the instrument classification of that sound sample.

After experimenting with pure frequency representation as a feature mapping and finding the results to be poor, we turned to more sophisticated signal processing techniques provided by Yaafe (Yet Another Audio Feature Extractor). An efficient and easy-to-use audio features extraction toolbox, Yaafe was developed at Telecom Paristech by the AAO (Audio, Acoustique et Ondes) group. Yaafe conveniently includes Python and Matlab bindings, and offers a wide variety of core audio features. We found some of the most useful and successful of these features to be the autocorrelation coefficients, the spectral shape statistics (including centroid, spread, skewness, and kurtosis) [3], and the Octave Band Signal Intensities (OBSI) [5], which is a rough estimator of harmonic power distribution.

IV. Instrument Separation

As described in our overview, we tested our classification technique on the outputs of Independent Component Analysis, which contain a single primary instrument and one or more instruments with reduced amplitude in the background. These background instruments are the result of imperfect separation in ICA, which is not gauranteed to find the perfect unmixing matrix. As expected, this will degrade the performance of the classification somewhat. Refer to figure 1 for an example of sound samples that have gone through the mixing and unmixing process.

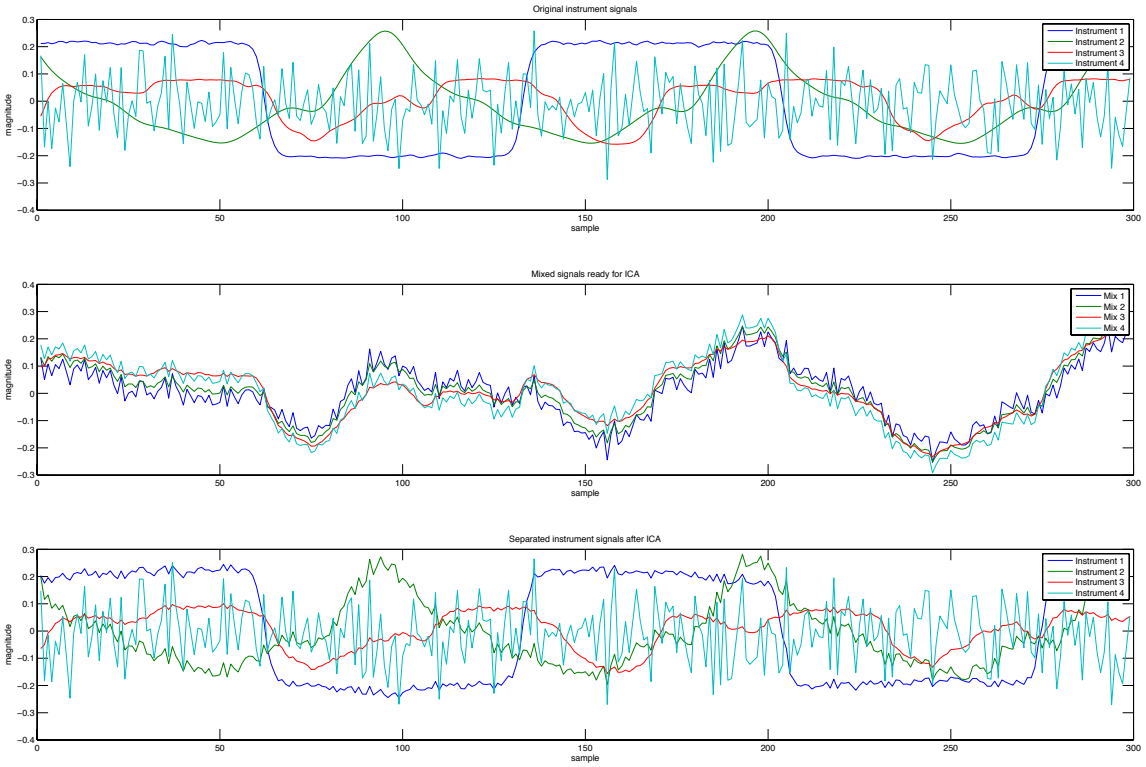

Figure 1: *Original sound samples, mixed recordings, and unmixed results of ICA.*

V. Results

We measured our system's performance by counting the number of correctly identified instruments from randomly generated sound samples. In these eight-second samples, the simulated instrument plays random sets of notes for random durations, in combinations that are unlikely to appear in training data. Overall we found that some features performed very weakly and could even detract from the performance of more predictive features, suggesting some level of overfitting. We achieved an excellent prediction accuracy of 95% when testing our SVMs on randomly generated sounds from known instruments (see figure 3). The task of identifying instruments from samples obtained via ICA is a more difficult one, but we were still able to reach 75% accuracy when using autocorrelation coefficients, autocorrelation peaks integrator, and the OBSI. Refer to figures 2 and 3 for accuracy results using on original sound samples and ICA-unmixed sound samples, respectively.

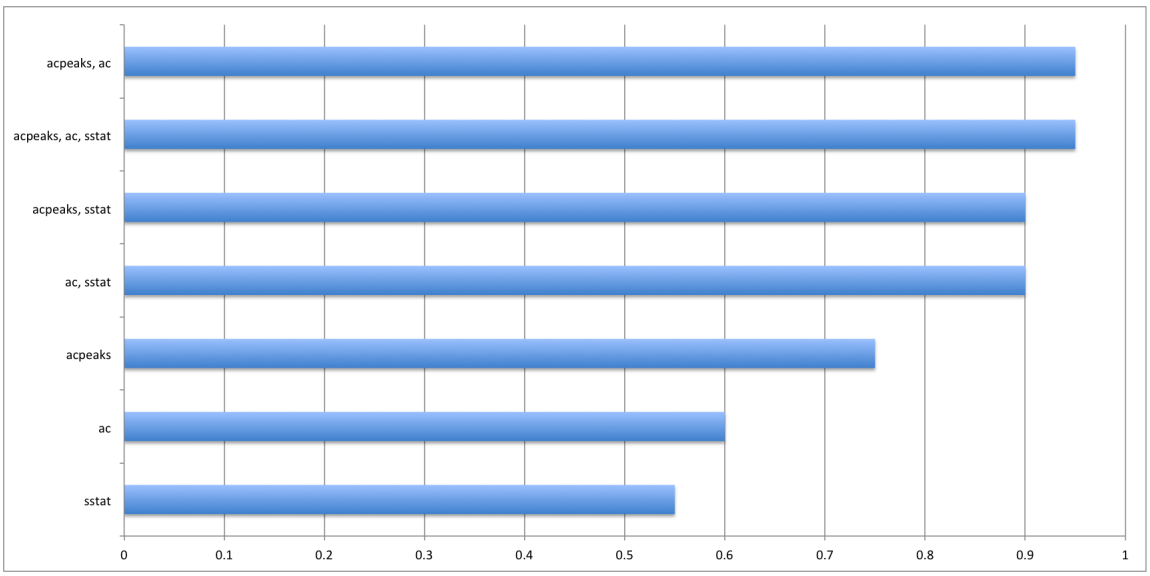

Figure 2: *Instrument Prediction Accuracy using different sets of Yaafe features. Key: acpeaks = AutoCorrelationPeaksIntegrator, ac = AutoCorrelation, sstat = SpectralShapeStatistics*

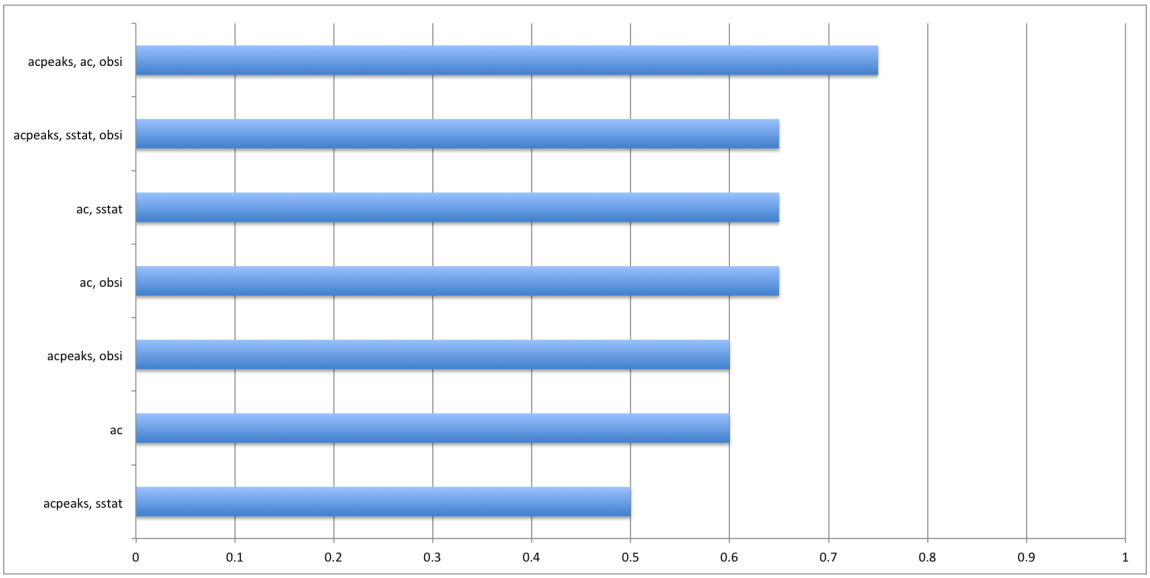

Figure 3: *Instrument Prediction Accuracy using different sets of Yaafe features, using data that has gone through mixing and unmixing. Key: acpeaks = AutoCorrelationPeaksIntegrator, ac = AutoCorrelation, sstat = SpectralShapeStatistics, obsi = OctaveBandSignalIntensity*

VI. Future Work

There are several opportunities to extend and improve our project. Given more time, we would have liked to work toward more ambitious goals such as:

- Exploring other features to include in training the SVM, and doing a more in-depth analysis of their relative effectiveness
- Replicating a simple version of instrument synthesis using parameters extracted from a generative instrument model
- Comparing machine classification of instruments to human classification, especially in the absence of an attack window, which is well known to make this task difficult for humans

References

[1] B.Mathieu, S.Essid, T.Fillon, J.Prado, G.Richard, YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software.Proceedings of the 11th ISMIR Conference, Utrecht, Netherlands, 2010.

[2] Ge Wang, The ChucK Audio Programming Language: A Strongly-timed and On-the-fly Environ/mentality. PhD Thesis, Princeton University, 2008.

[3] O.Gillet, G.Richard, Automatic transcription of drum loops. in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Montreal, Canada, 2004.

[4] R Fan, K Chang, C Hsieh, X Wang, C Lin, LIBLINEAR: A Library for Large Linear Classification. Journal of Machine Learning Research, 2008.

[5] S.Essid, G. Richard, B. David, Musical Instrument Recognition by Pairwise ClassiiňĄcation Strategies. IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 14, 2006.