

LEARNING MUSICAL INSTRUMENTS FROM MIXTURES OF AUDIO WITH WEAK LABELS

David Little and Bryan Pardo

EECS Department

Northwestern University

Evanston, IL 60208

d-little,pardo@northwestern.edu

ABSTRACT

We are interested in developing a system that learns to recognize individual sound sources in an auditory scene where multiple sources may be occurring simultaneously. We focus here on sound source recognition in music audio mixtures. Many researchers have made progress by using isolated training examples or very strongly labeled training data. We consider an alternative approach: the learner is presented with a variety of weakly-labeled mixtures. Positive examples include the target instrument at some point in a mixture of sounds, and negative examples are mixtures that do not contain the target. We show that it not only *possible* to learn from weakly-labeled mixtures of instruments, but that it works significantly *better* (78% correct labeling compared to 55%) than learning from isolated examples when the task is identification of an instrument in novel mixtures.

1 INTRODUCTION

We are interested in developing a system that can learn to recognize individual sound objects in an auditory scene where multiple sound sources may be occurring simultaneously. A system able to identify what particular sound sources are present in a mixture would enable automatic tagging of audio recordings with meta-data. A system capable of modeling sources in audio mixtures, without having to first learn the sounds in isolation, would be useful to researchers working on separating audio mixtures into their component sources.

In a typical supervised learning paradigm, isolated instances of a particular class of data are presented to a learner. Learning in this case is mostly limited to a stage of the design process. With a more ecologically realistic set of requirements, where learning can occur in an environment after deployment, this isolation of training instances can not be guaranteed. While similar problems have already been studied in the field of computer vision [1], auditory data presents its own unique problems because mixtures of sound can result in a composite of multiple sources at a given time.

In this paper, we focus on identification of musical instruments in a musical mixture. Almost all systems that learn

to identify musical instruments require isolated examples of the target instrument at some stage in the training process (see [9] for a review). This could mean, isolated notes [12], or more recently, solo phrases from an instrument [11]. A number of these systems have been evaluated using polyphonic audio [5, 11, 15], i.e. audio with multiple simultaneous notes, but this work is still limited by the requirement that instruments be *learned* in isolation.

We are aware of two systems that *learn* solely from polyphonic audio. However, they differ significantly from the task we consider here. In [6] the system is trained on polyphonic audio, but each unique combination of instruments must be learned individually. The authors admit this approach is only realistic when learning is done before deployment. Given n instruments, this results in 2^n possible combinations that must be individually learned, significantly increasing the number of required training examples. This approach also leaves open the question of how to recognize individual instruments when presented in novel audio contexts. In addition, the training data requires labels indicating all instruments currently playing for every two second segment of audio, a fairly intensive labeling task. This is true even if a score is available (since it must be correctly aligned with a particular audio recording).

Kitahara et al [10] learn from polyphonic data, avoiding the combinatorial problems of [6]. However, they require the user to input a musical score for the recording that exactly labels every note with pitch, onset time and offset time, i.e. perfect segmentation of each instrument into individual notes is assumed.

What we would really like is a system that can learn from weakly labeled mixtures. We call a label weak if only the presence or absence of the target sound object is indicated for some N second length of audio. A weakly labeled positive example will contain the target object at some point in the example, but a significant portion of the time may not contain audio from the target. A weakly labeled negative example does not contain the target sound object at any point in the example. In this scenario a useful positive training example for the “saxophone” class would be

an acoustic recording of a saxophone and bass duo playing at a gallery opening. A negative example would be some other audio clip that contains no saxophone in the recording. Training could be accomplished on-line by recording a few audio clips that contain the desired sound. Our proposed input and output have important differences from that of previous systems that learn from polyphonic audio [6, 10] (weakly vs. strongly labeled input and multi-class vs. a presence/absence label as output).

As a first step towards our objective, in this paper we evaluate a system that provides presence/absence labels for a single instrument over short time intervals (~2 seconds). We do not require labeling all instruments in the mixtures at short intervals, nor do we require detailed scores. Instead, the system learns from weakly labeled examples that include distractor sounds.

Our instrument classes are learned independently of the background sounds: the classifier can be trained in one background environment and used in another. Although all the tools we use to accomplish this task already exist (as described in Section 2), we are not aware of anyone who has yet put these pieces together the way we do. In our experiments (in Section 4), we show that it is not only *possible* to learn from a mixture of instruments (constructed as in Section 3), but that it works significantly *better* than learning from isolated examples when identifying instruments in novel mixtures.

2 CLASSIFICATION METHOD

We use the segment-and-combine [7] framework for our classification system, defined below.

1. **Segment** each example X_i by selecting n potentially overlapping segments $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}, \dots, \mathbf{x}_{in}\} \in X_i$ at random (as per [7]), using the piece extraction operator described below.
2. **Learn** a classifier function $f(\mathbf{x}) \in [0, 1]$ which returns higher values for segments more likely to be part of the target object.
3. **Classify** segments with the learned function $f(\mathbf{x})$.
4. **Combine** each of the classifications of $f(\mathbf{x})$, $F(X_i) = C[f(\mathbf{x}_{i1}), f(\mathbf{x}_{i2}), \dots, f(\mathbf{x}_{in})] \in \{0, 1\}$. The combinator $C[\cdot]$ returns a positive result if the average of $f(\mathbf{x})$ for all segments is greater than 0.5 (as per [7]).

We chose this approach because it has shown promise, even compared to more sophisticated approaches, in similar problems, such as learning to recognize visual objects from cluttered data [7]. We describe the extraction operator in Section 2.1 and the segment classifiers $f(x)$ in Section 2.2.

Feature	Description/Details
MFCCs	Mel frequency cepstral coefficients
Brightness	Ratio of energy above 1500Hz to energy below.
Spectral Stats	First four statistical moments of spectrum
Flatness	Ratio of geometric to arithmetic mean of spectrum
Pitch	Calculated using first peak of autocorrelation function.
Amplitude	Root mean squared average.
Flux	Difference of spectrum between subsequent frames.
Inharmonicity	The strength of the pitch: amount of energy far from $f_0 \cdot n$

Table 1. The features used to represent each audio segment.

2.1 Extraction Operator

Our extraction operator to select possibly overlapping audio segments in Step 1 selects a sub-portion of the audio, based on two parameters: time(t) and length(l). The range of possible values for our start time (t) is from 0 to the length of the file in frames. The length of the extracted segment (l) ranges from 0.1 seconds to 0.5 seconds. Given an audio example X_i , we select 5 segments per second of audio in the example. Thus, for a 10 second example there would be 50 segments. We randomly select these segments, picking values for t and l using a uniform probability distribution over time.

The j th segment drawn from example i is represented as a vector of features $\mathbf{x}_{ij} \in \mathbb{R}^d$. In this case $d = 22$ (13 MFCCs, 4 spectral statistics, plus the other 6 features). We choose to use a set of features found to be commonly useful in past work on musical instrument identification. All features used are listed in Table 1.

2.2 Segment Classifiers

We compare three classifiers: Extra Trees [8], which, as we will explain, should be fairly appropriate for this task, to a very simple classifier, the K-nearest neighbors algorithm [4], and a fairly sophisticated classifier, the Support Vector Machine [2] with a radial basis kernel. This comparison is detailed in Section 4.

It is important to recognize the noisy nature of the labeling data: it is possible that a segment from a positive example, which would be labeled positively, contains no energy from the target instrument. Since labels are weak, a positive example for “saxophone” could be a jazz recording containing many portions where the saxophone is silent.

The results in [3] suggest that methods such as bagging or randomization work well under such noisy labeling con-

ditions. The Extra Tree algorithm [8] is an ensemble method that uses randomized decision trees. It differs from other randomized decision trees, in that, with the lowest parameter setting, the algorithm would completely randomize both the split and the attribute selected at each decision node, yielding a tree completely independent of training data (Extra Tree is short for extremley randomized tree). The construction of a single Extra Tree is summarized below.

1. Let the subset of training examples be S , the feature-split pool size be K and the minimum subset be n_m .
2. If $|S| < n_m$ or if all features values are constant, or all examples in S share the same label.
 - (a) Define this tree's output $f(\mathbf{x})$ as the value of the most frequent label in S .
3. Otherwise:
 - (a) Select K randomly uniform split s_i and feature f_i pairs.
 - (b) Select the best split-feature pair s_{max}, f_{max} by entropy measures.
 - (c) Let the output $f(\mathbf{x})$ be the output of the right tree if $f_{max} < s_{max}$ for \mathbf{x} and the output of the left tree if $f_{max} \geq s_{max}$.
 - (d) Create the right tree, with S equal to the subset of examples with $f_{max} < s_{max}$.
 - (e) Create the left tree, with S equal to the subset of examples with $f_{max} \geq s_{max}$.

Results in [8] suggest that with appropriate parameter settings, as detailed in Section 4.1.1, the Extra Tree ensemble outperforms other random tree ensemble methods.

3 DATA

For our initial tests we wanted to tightly control the difficulty of our examples, allowing us to observe how each of our classifiers degrades as the data becomes more challenging. So, rather than use commercial recordings, we trained and tested on synthesized mixtures of randomly selected notes from four instruments. Each positive example contains a mixture of instruments, one of which corresponds to the label the system should learn. Thus, a positive example for "Cello" might contain a Cello, Violin and Oboe. The four instrument classes used for our source sounds were Cello, Violin, Oboe, and Bb Clarinet. Instrument recordings were taken from a set of instrument samples made available by the University of Iowa¹. The ranges of pitches used in our experiments for the instruments were as follows: for

¹<http://theremin.music.uiowa.edu/index.html>

Cello we used notes from C2 to C7, for Violin from G3 to Db7, for Clarinet from D3 to B6, and for Oboe from Bb3 to C6. These were played at three different dynamic levels, *pp*, *mf* and *ff*. All samples were encoded as linear PCM 16bit wav files at a sampling rate of 44100Hz. The Iowa recordings were segmented into individual notes. Training and testing examples were constructed as follows.

1. Select a set of instruments
2. For each instrument, create a melody by randomly picking a series of note recordings from the database
3. For each instrument melody, intersperse the notes with silences of random length from 0 to some maximum value.
4. Adjust instrument volumes to obfuscate the target instrument to the desired level.
5. Sum the melodies to create a mixture.

Examples were a minimum of 10 seconds long. The average length of an example was 11.5 seconds and the longest example was 17.3 seconds. To vary the a level of target obfuscation we varied the maximum spacing between notes in the target instrument melody (s_t) and notes in the distractor instrument melodies(s_d), as well as the relative amplitude of the distractors notes (a_d) across three target obfuscation conditions. These values were selected so that we would obtain a range of target to mixture ratios, as defined in Section 4.2. For **easy obfuscation**, $s_t = 0.5$, $s_d = 1$ and $a_d = 1$; for **medium obfuscation**, $s_t = 1.5$, $s_d = 1$ and $a_d = 1$; and for **hard obfuscation** $s_t = 1.5$, $s_d = 0.5$, and $a_d = 1.5$. Thus, for easy obfuscation, target instrument notes were more frequent than distractors and of equal amplitude. For hard obfuscation, distractors were significantly more frequent and louder than targets.

Conditions varied along three dimensions: target instrument (one of the four instruments), ensemble of distractor instruments (one to two of the other instruments in all possible combinations), and target obfuscation (from easy to hard): this resulted in a total of $4 \times 6 \times 3 = 72$ data conditions. We created 10 positive examples and 10 negative examples for each of the conditions, yielding a total of 1440 example audio files.

Some examples of typical correctly and incorrectly classified examples for the mixed condition (from Section 4.1) can be found at the following URL:

<http://www.cs.northwestern.edu/~df1909/ismir2008sounds>

4 EVALUATION

In our evaluation we sought to answer to the following questions.

1. What effect does training on mixtures, rather than isolated notes have on accuracy when identifying instruments in novel mixtures of sound?
2. How does accuracy degrade across our three segment classifiers as the target class is increasingly obfuscated by the distractors in the mixture?
3. How does performance using weak labels compare to performance using stronger labels?

We begin our evaluation by addressing questions 1 and 2 in Section 4.1. Questions 3 is addressed in Section 4.2.

4.1 Experiment 1

In this experiment, we wanted to compare performance across isolated and mixture training conditions, across our three different segment classifiers (ExtraTrees, K-nearest neighbor and SVM).

We constructed the various ensembles of instruments as described in Section 3. We then ran an experiment under two conditions, *mixed* and *isolated* training, using the segment-and-combine algorithm described in Section 2 with $f(\mathbf{x})$ learned using one of the three segment classifiers (Section 2.2).

4.1.1 Parameter Settings

To select parameters for our K-nearest-neighbor (KNN) and the radial basis kernel SVM classifiers we evaluated a range of parameters using the data for our most challenging target obfuscation condition (the hard obfuscation condition as described in Section 3). We evaluated the KNN classifier with $K = 1, 2, 5, 10, 20,$ or 50 , of which 5 was the best K we found. For the parameters of the SVM, γ (variance of the Gaussian in the radial basis kernel) and C (the error penalty), we considered $\gamma = 0.001, 0.01, 0.1, 1, 10, 100,$ or 1000 , and $C = 0.001, 0.01, 1, 10, 100,$ or 1000 (yielding $6 \times 6 = 36$ possible parameters settings for the SVM), we found $\gamma = 0.1$ and $C = 100$ was the best setting.

For our ensemble of ExtraTree’s we used the recommended parameters, as described in [8]: thus $K = \sqrt{N}$ where N is the number of features (22), and $n_{min} = 5$. Generally the more trees in the ensemble the better this algorithm, so we chose a large number that did not unduely compromise speed (100).

We extracted segments from the audio files at a rate of 5 segments per second. This value was chosen based on a run of the Extra Tree algorithm using the hard obfuscation condition.

4.1.2 Procedure

In the first (*mixed*) training condition, we used the following procedure for each target instrument class T and each target obfuscation condition.

1. Train the learner on the example from 5 of the 6 possible ensembles of distractors using target T , totaling 50 positive and 50 negative examples.
2. Test on the remaining 20 examples from the final ensemble condition.
3. Repeat steps 1-2, excluding a different condition each time.

In the second (*isolated*) training condition, we used this procedure for each target instrument class T and each target obfuscation condition:

1. Train the learner on all isolated notes of an instrument. For negative examples, we use an equal number of randomly selected notes from the remaining instruments.
2. Test on the 10 negative and 10 positive mixture examples with target T , for each of the 6 ensemble conditions, for a total of 120 examples tested.

This condition included from 138-170 unique isolated notes (depending on the instrument) during training: this is as many or more notes than used in the *polyphonic* training condition.

Thus for each classifier we had a total of $24 \times 3 = 72$ trials of our system (24 mixture conditions \times 3 target obfuscation conditions).

4.1.3 Results

We calculated system accuracy in each trial based on how well the system could identify approximately every two seconds of audio: each example being tested was divided into sixths: for every sixth (or about two seconds) of an example given a correct label we noted that as a success. Successes were then tallied across all examples in a trial. Note that for all obfuscation conditions there was no more than 1.5 second silence between target notes.

Since this is a balanced binary labeling task, random chance performance for a given condition is 50%. Overall performance for the systems across the various target obfuscation conditions is shown in Figure 1.

We performed a $2 \times 3 \times 3$ repeated measures ANOVA within the 24 mixture conditions, across the training condition, segment classification method, and target obfuscation condition. This analysis showed main effects for training condition ($F = 124.90, p < 0.001$), classification method ($F = 48.51, p < 0.001$), and target obfuscation condition ($F = 57.48, p < 0.001$), as well as interactions between training condition with classification method ($F = 12.27, p < 0.001$) and training condition with target obfuscation ($F = 7.72, p = 0.006$). Since results were poor for the

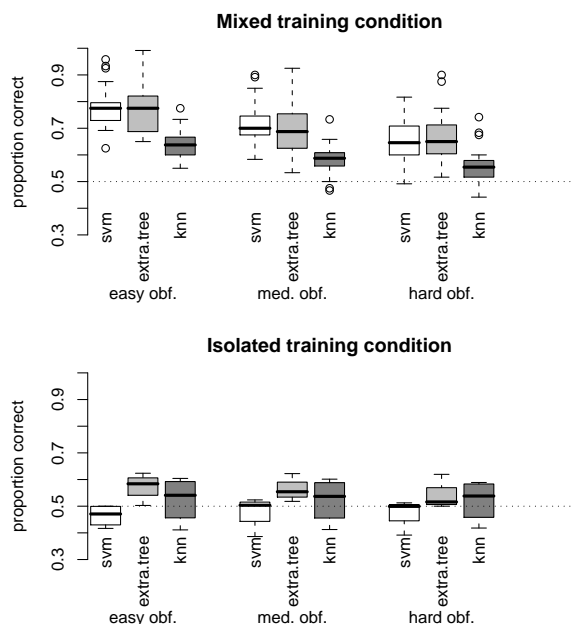


Figure 1. Performance across the three segment classification methods for both the isolated and mixed training conditions. Chance performance is indicated by the dotted line. Here, “obf.” stands for “obfuscation.”

isolated conditions, it is not surprising that the other main effects should interact with the training condition.

Post-hoc permutation tests [13] showed that there was no statistically significant difference in performance between the ExtraTree and SVM segment classifiers ($p = 0.845$) on the mixture case, but performance with the KNN classifier was worse from both SVMs and Extra Trees ($p < 0.001$). The the KNN method is known to be sensitive to irrelevant features, since we have yet to consider feature selection, this is one possible reason the KNN method failed to work. All three obfuscation conditions differed from each other ($p < 0.02$). We also confirmed the that the isolated condition was worse than the mixed training condition ($p < 0.001$). We also examined the pattern of performance across different sorts of mixtures: not surprisingly the more similar a target is to its distractors the worse performance is: due to space considerations we have left the breakdown of performance by instrument out.

Only in the mixed training condition was performance better than chance (0.5), and then only for the SVM and Extra Tree classifier ($p < 0.001$). Taken together, these results suggest that the SVM and the Extra Tree algorithm can be used to classify instruments in novel mixtures using the segment-and-combine algorithm when the training data are mixtures of sound rather than isolated notes. As obfuscation of the target became greater, accuracy dropped similarly across the three segment classifiers.

4.2 Experiment 2

We performed a second experiment to determine how a more exact labeling of segments would affect our results. To do this we determined the ratio of the target instrument to the total mixture (R) in each segment selected for classification. This was found using the following equation:

$$R = \frac{\|proj_t \mathbf{m}\|^2}{\|\mathbf{m}\|^2} \quad (1)$$

The linear projection of the target onto the mixture, $proj_t \mathbf{m}$, is used as an estimate of the target that remains in the mixture as per [14]. An R of 0.5 means half the energy in a segment is from the target. Given an R for each segment our system is provided a positive label only if it is above some threshold t . This new labeling will be more accurate because segments from positive examples that contain no energy from the target instrument will not be included as a labeled segment. Using this procedure, we trained the Extra-Tree classifier as per the mixture condition in Experiment 1 with the data from the hardest target obfuscation condition. Thus each test at a given threshold consists of 24 trials: 4 instruments with 6 possible arrangements of distractors. Results across various values for the threshold t can be seen in Figure 2.

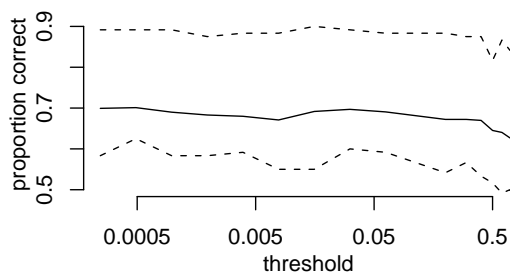


Figure 2. Classification accuracy as a function of the threshold R used to label segments. Dotted lines indicate the maximum and minimum; the solid line indicates the mean.

We can conclude from this that although performance can be improved if we select a fairly inclusive t (i.e. one that includes any segment that has even a small fraction of the target), this will not yield performance gains of much more than about 4-6%. We would note that the poor performance for high values of t may be attributable to the small sample of segments that actually have a high enough R . For example with $t = 0.5$ only about 19% of the segments from positive examples were included.

5 CONCLUSIONS AND FUTURE WORK

Many researchers have tried to address the problem of identifying instruments in mixtures of sound by trying to mitigate the effect the other instruments have on the target in-

strument to be identified, through, for instance, source separation methods [15], missing feature approaches [5] or weighting of features [10]. Here we presented an alternative approach: rather than being smart about how we use isolated training data, we have simply used a variety of mixtures as our training data.

This simple method makes classification possible in mixtures of random notes using only weakly labeled data, when it is not possible to do so when trained with isolated training data, without one of the above mentioned methods. We believe this is because the training data, in the mixed case, is more representative of the testing data, even when the training mixtures do not use the same set of instruments as the testing mixtures. It's possible that the use of mixtures during training could be used in conjunction with these other methods to further improve performance.

We found that when using more strongly labeled data, our results did not improve much (Section 4.2). We believe the most likely explanation of this result is that we have hit a performance ceiling for the chosen features set. This may also explain the similar performance of Extra Trees and the SVM. In future work, now that we have identified a potentially useful training method and learning algorithm for this task, we will begin to explore a more effective set of features.

We will also need to confirm these results on more musical arrangements, and more challenging recordings (e.g. varied articulation, echoes and non-instrumental background, etc...). Since our algorithm does not a priori depend on these simplifications, we suspect the conclusions drawn here will be meaningful in more challenging environments.

6 ACKNOWLEDGEMENTS

We'd like to thank Josh Moshier for his time preparing instrument data. This work was funded by National Science Foundation Grant number IIS-0643752.

7 REFERENCES

- [1] P. Carbonetto, G. Dorko, C. Schmid, H. Kück, N. de Freitas, and M. Vision. Learning to recognize objects with little supervision. *International Journal of Computer Vision*, 2007.
- [2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [3] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.
- [4] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*, chapter 4.4. Wiley-Interscience, 2000.
- [5] J. Eggink and G. J. Brown. A missing feature approach to instrument identification in polyphonic music. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, volume 5, pages 553–556, 2003.
- [6] S. Essid, G. Richard, and B. David. Instrument recognition in polyphonic music based on automatic taxonomies. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):68–80, 2006.
- [7] P. Geurts, R. Maree, and L. Wehenkel. Segment and combine: a generic approach for supervised learning of invariant classifiers from topologically structured data. *Machine Learning Conference of Belgium and The Netherlands (Benelearn)*, 2006.
- [8] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.
- [9] Perfecto Herrera-Boyer, Anssi Klapuri, and Manuel Davy. Automatic classification of pitched musical instrument sounds. In Anssi Klapuri and Manuel Davy, editors, *Signal Processing Methods for Music Transcription*, pages 163–200. Springer Sciences+Business Media LLC, New York, NY, 2006.
- [10] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno. Instrument identification in polyphonic music: Feature weighting to minimize influence of sound overlaps. *EURASIP Journal on Advances in Signal Processing*, 2007.
- [11] A. Livshin and X. Rodet. Musical instrument identification in continuous recordings. In *7th International Conference on Digital Audio Effects*, 2004.
- [12] Keith D. Martin and Youngmoo E. Kim. 2pmu9. musical instrument identification: A pattern-recognition approach. In *136th meeting of the Acoustical Society of America*, 1998.
- [13] J. Menke and TR Martinez. Using permutations instead of student's t distribution for p-values in paired-difference algorithm comparisons. *IEEE International Joint Conference on Neural Networks*, 2, 2004.
- [14] E. Vincent, R. Gribonval, and C. Fevotte. Performance measurement in blind audio source separation. *Audio, Speech and Language Processing, IEEE Transactions on [see also Speech and Audio Processing, IEEE Transactions on]*, 14(4):1462–1469, 2006.
- [15] Emmanuel Vincent and Xavier Rodet. Instrument identification in solo and ensemble music using independent subspace analysis. In *International Conference on Music Information Retrieval*, 2004.