Transcription of Percussion Instruments in Polyphonic Audio

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Beach Clark | Daniel Ethridge | | | Laney Light |
| Georgia Institute of  Technology  Bclark66@gatech.edu | | Georgia Institute of Technology  dethridge7@gatech.edu | Georgia Institute of  Technology  Llight7@gatech.edu | |

ABSTRACT

This paper presents an algorithm developed for the purposes of automatic drum transcription (ADT). We began with a support vector machine (SVM) as our baseline classification model trained on 10 features. From there, we went through multiple iterations of development and arrived at a total of five different models. Model two utilized the baseline feature set and two additional data sets (ENST and IMDT-SMT), while models three through five utilized the three data sets and different combinations of features. We found that our model tends to work well when the drums sounds are separated out but performs worse when more instruments are introduced.

1. INTRODUCTION

Automatic drum transcription (ADT) can be described as “the reverse of music making. Instead of having musicians perform with their instruments according to a notated sheet music, ADT aims at deriving such symbolic notation from previously recorded music.” [1] The symbolic notation derived from ADT can then be used in other musicological tasks such as creation of karaoke tracks or the creation of large collections of symbolic data for statistical analysis such as the database of features for approximately 40 million songs that Spotify maintains. ADT is still a very challenging task since drum events are dissimilar from the more frequently studied pitched events such as piano or the human voice [16]. There has been good success in transcribing a small number of certain isolated drum sounds, ADT in polyphonic audio remains a challenge. Our aim in this project was to create a baseline system for transcribing drums in a polyphonic texture that would allow for experimentation with multiple combinations of features and feature extraction parameters (e.g. block size) as well as model parameters (e.g. combinations of features to be used in the model). We also experimented with multiple selections of training data.

1. Related Work

Over the past 15 years, considerable research has been devoted to automated drum transcriptions. Wu et. al provide a comprehensive survey of the research along with a detailed quantitative analysis of the methods employed and the results achieved in three broad areas of research: drum transcription without other instruments present, drum transcription with other percussion present, and drum transcriptions with other, pitched instruments present [16]. They found that some form of recurrent neural network (RNN) achieved the best results in all three categories, although the gap between RNNs and other simpler models was less pronounced in the case of full polyphonic music[1].

One of the challenges with neural networks has been the curation of labeled datasets that are representative of the variety of drum sounds and textures in real-world audio. Recently, three datasets have been developed that have supported a large portion of the research in this area. The most recent, MDB Drums, is a collection that was developed for ADT and includes 23 actual polyphonic recordings for drums in a variety of genres [12]. ENST Drums is a large database for ADT that consists of recordings by three professional drummers in multiple contexts such as solo drumming and drumming along with a pre-recorded backing track [15]. IDMT-SMT drums is a dataset that consists solely of snare drum, hi-hat and kick drum sounds from recorded sessions, drum synthesizers and drum sample libraries [17].

1. Algorithm Overview and Description
   1. Data Sources

For our baseline model, we used MDB drums [12]. Over the 23 tracks, the dataset identifies 7,994 percussion instrument onsets in total, but we focused our attention to the tracks containing polyphonic audio. The annotation files contain six instrument classes: snare drum (SD, 33% of total onsets), hi-hat (HH, 33%), kick drum (KD, 19%), cymbal (CY, 13%), tom (TT, 1%), and other percussion (OT, 1%).

Because the MDB Drums dataset is small, we supplemented it using the ENST Drums dataset [15]. The ENST data contains 143 annotation files including onset times and labels for 23 types of percussion instruments. We collapsed the instrument labels into fewer categories to align with the MDB drums categories (for example, in the MDB dataset, bass drum hits are labeled ‘KD’ but in the ENST dataset they are labeled ‘bd’, so we re-labeled all the ‘bd’ as ‘KD’). For each annotation file, there are multiple associated audio files containing different instrumentation and mixing (dry mix, wet mix, snare only, etc.). We used the dry mix files and the instrument-specific files for SD, KD, and HH (tracks recorded with the microphone placed near the specified instrument). We processed the data from both datasets as described in sections 3.2 and 3.3. We then merged the onsets, features, and labels into a single dataset based on the event times (event times for the features were calculated as a function of the hop size and sample rate). Features falling outside a window of 10 milliseconds before or after a label were discarded. We experimented with several different window sizes from 10 to 100 milliseconds and found that 10 milliseconds worked best. Because many of the files have musical events other than the drum hits we are trying to identify with our model, it makes sense to limit the features we use to those falling as close to the annotations as possible.

We also used the IDMT-SMT data to compare our polyphonic results to the same models using monophonic data. The IDMT data also contains multiple .wav files corresponding to a single annotation file, but in the case of IDMT-SMT, only snare, kick, and hi-hat events are included.

* 1. Onset Detection

We blocked each audio signal using a block size of 4096 and a hop size of 2048 and used the Madmom audio signal processing library in Python [14] to identify onsets with an RNN onset processor. Our use of an onset detector was predicated on the system design in [2]. However, on further examination of the datasets we were using which contained musical events for other instruments, we discontinued using the onset detector. Adding back the onset detector is something we can consider for future work.

**3.3 Feature Identification**

For our baseline system, we calculated ten features for each window of each file in which an onset was annotated: the first five Mel-Frequency cepstral coefficients (MFCCs), root-mean-square (RMS), spectral flux, spectral crest, zero crossings, and spectral centroid. We also computed spectral flatness, spectral rolloff, and spectral contrast using the Python librosa module [20]. Spectral contrast comprised seven features (the contrast between the spectral peak and spectral valley as well as for each frequency sub-band). We applied several filters to the signal and computed the RMS and several other RMS-related statistics to try to improve the instrument-specific results (for example, we predicted that the kick drum detection could be improved by applying a low pass filter). The RMS-related statistics were the RMS calculated over low pass, band pass, and high pass filters as well as the relative differences between each of the filtered RMS statistics and the unfiltered RMS. Also included were the differences between each of the filtered RMS statistics with each other [2]. In total, this amounted to 29 features. We normalized each feature across all files in each data source we tested.

1. Evaluation
   1. Classification Models

We implemented support vector machine (SVM) classification models in Python using scikit-learn [18] and the annotated instrument classes as the ground truth. We tested incremental changes to the model, varying the testing and training data, kernel shape, model parameterization, and features. For each model, we calculated the precision, recall, and F-score for each instrument class and for the overall model. Our primary evaluation metric was the F-score for the overall model, weighted by the instrument sample sizes.

* + 1. Model 1: Baseline, MDB dataset

We divided the MDB Drums files into a testing set (26% of files and 31% of annotated onsets) and a training set (74% of files and 69% of onsets) and used the set of 10 features described in section 3.3 to train our SVM. We tested several different options for the kernel shape and found that the sigmoid kernel provided the best results. The linear model predicted that every instrument was HH, and the polynomial predicted that only HH or SD. We also ran the baseline model with 10-fold cross-validation with the sigmoid kernel and found that the F-score was equivalent to the model that used a 76/24 test/train split.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instrument | Precision | Recall | F score | N  Onsets |
| CY | 0.03 | 0.21 | 0.05 | 61 |
| HH | 0.31 | 0.52 | 0.39 | 484 |
| KD | 0 | 0 | 0 | 284 |
| OT | 0 | 0 | 0 | 25 |
| SD | 0.5 | 0.22 | 0.31 | 802 |
| TT | 0 | 0 | 0 | 30 |
| macro avg | 0.14 | 0.16 | 0.12 | 1686 |
| weighted avg | 0.33 | 0.26 | 0.26 | 1686 |

**Table 1.** Baseline Model Results with 76/24 Split and Sigmoid Kernel

* + 1. Model 2: Baseline feature set, all datasets, 3 instruments

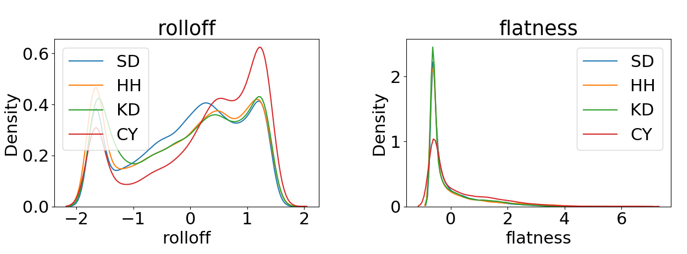
Expanding on the baseline model, we fit the SVM with the same set of 10 features to each of our other datasets and used the best-fitting kernel identified in Model 1. For the MDB Drums dataset, we used the same test/train split as in Model 1. We also fit the same model to the ENST Drums data only, using half of the dry mix files as the test set and the remainder of the files as the training set.

We also combined the ENST and MDB Drums into one dataset, using the ENST data as the training set and MDB as the testing set.

*4.1.4 Models 3-5: Alternate feature sets*

Starting with the full set of 29 features described in section 3.3, we determined which features to include in the model. The large number of files and features prevented us from doing an exhaustive comparison of all combinations of features because of the long run time, so we used an alternate approach for feature selection.

First, we plotted each feature distribution by annotated instrument class to determine visually which features appeared to distinguish well between instruments [see Figure 1]. We observed that the spectral rolloff distribution was substantially different between instruments. Spectral flatness appeared to distinguish cymbals from the other instruments. Spectral centroid, zero crossings, RMS, crest, and flux had a spike for the hi-hat that was not present for the other instruments.



**Figure 1**. Sample of Feature Distribution Plots

Next, we fit logistic regression models (one per instrument class) using the statsmodels module in Python [19]. We fit the SVM using a set of nine features[[1]](#footnote-1) that had a p-value less than 0.01 for at least one of the models (Model 3).

Then we fit the SVM to the same datasets using the set of 13 features identified by the logistic regression and/or the distribution plots[[2]](#footnote-2) (Model 4) and tested the same feature set with a 4th degree polynomial kernel (Model 5). We used the same test/train splits as in Model 2. We found that model 3 provided the best F-score for the MDB data only and combined MDB/ENST, but model 5 provided the best results for the ESNT data only and IDMT data only (see Table 2).

We found that the IDMT data resulted in the best model fit, because this data consists only of monophonic audio. With datasets that contain only polyphonic audio, the F-scores were generally lower. We hypothesized that our MDB predictions would improve by adding the instrument-specific audio files from the ENST data into the training set (see MDB only and MDB+ESNT columns in Table 2). However, we found that the MDB predictions were better when the model was trained on only MDB data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **F-score** | | | |
| MDB only | MDB+  ENST | ESNT only | IDMT only |
| 1. Baseline: 10 features, 6 instruments, sigmoid kernel | 0.26 | NA | NA | NA |
| 2. Baseline set of 10 features, 3 instruments, sigmoid | 0.26 | 0.32 | 0.24 | .43 |
| 3. Set of 9 features from logistic regression, sigmoid | **0.37** | **0.28** | 0.31 | .49 |
| 4. Set of 13 features from logistic regression/distribution plots, sigmoid | 0.30 | 0.27 | 0.32 | .55 |
| 5. Set of 13 features from (5), polynomial kernel degree 4 | 0.36 | 0.27 | **0.35** | **.81** |

**Table 2.** F scores by Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instrument | Precision | Recall | F score | N  Onsets |
| CY |  |  |  |  |
| HH |  |  |  |  |
| KD |  |  |  |  |
| SD |  |  |  |  |
| macro avg |  |  |  |  |
| weighted avg |  |  |  |  |

**Table 3**. Results for Best Fitting Model using MDB data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | CY | HH | KD | SD |
| CY |  |  |  |  |
| HH |  |  |  |  |
| KD |  |  |  |  |
| SD |  |  |  |  |

**Table 4**. Confusion Matrix for Best Fitting Model

1. conclusion

Our model works well when the drum sounds are separated out, but the fit gets progressively worse as we add in more instruments. These findings are consistent with [1] where the authors also found that results get progressively worse when the target drum sounds (in our case snare, kick and hi-hat) were mixed with other percussion and even worse when pitched instruments are included. Using many of the approaches suggested by … Transcribing percussion in a polyphonic texture is still a difficult problem, and more research into alternative methods is needed.

1. REFERENCES
2. C.-W. Wu et al., “A Review of Automatic Drum Transcription,” IEEE/ACM Trans. Audio Speech Lang. Process., vol. 26, no. 9, pp. 1457–1483, Sep. 2018.
3. K. Tanghe, S. Degroeve, and B. D. Baets, “An Algorithm for Detecting and Labeling Drum Events in Polyphonic Music,” p. 8.
4. R. Vogl, M. Dorfer, and P. Knees, “Drum transcription from polyphonic music with recurrent neural networks,” in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, 2017, pp. 201–205.
5. Bock, Sebastian and Korzeniowski, Filip and Schulter, Jan and Krebs, Florian and Widmer, Gerhard, “madmom: a new Python Audio and Music Signal Processing Library.”
6. C. Southall, C.-W. Wu, A. Lerch, and J. Hockman, “MDB Drums -- An Annotated Subset of MedleyDB for Automatic Drum Transcription,” 2017.
7. S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “madmom: a new Python Audio and Music Signal Processing Library,” *arXiv:1605.07008 [cs]*, May 2016.
8. O. Gillet and G. Richard, “ENST-Drums: an extensive audio-visual database for drum signals processing,” p. 4.
9. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
10. Seabold, Skipper, and Josef Perktold. “statsmodels: Econometric and statistical modeling with python.” Proceedings of the 9th Python in Science Conference. 2010.
11. B. McFee et al., “librosa: Audio and Music Signal Analysis in Python,” presented at the Python in Science Conference, Austin, Texas, 2015, pp. 18–24.

1. Spectral flux, first four MFCCs, flatness, rolloff, RMSF, and RMS with low pass filter [↑](#footnote-ref-1)
2. Set of nine features identified by logistic regression, plus spectral centroid, zero crossings, RMS, and spectral crest [↑](#footnote-ref-2)