Drum-Off, a Drum Playing Game

Mikko Hakila

June 6, 2018

Contents

Abstract

Drum-Off is a computer game where a drummer competes against a computer in a Drum-off. Drummers take turns in playing the most attractive beats possible trying to put the other drummer in shame. This thesis describes the architecture and algorithms used in such a game.

1 Introduction

I will describe and implement a system that transcribes annotation for all drums of a specific drum set. From this annotation the system will learn rhythmic patters and generate new ones. These rhythmic patterns will then be played back to the performing drummer as a comeback of the players performance.

I will research the area of music information retrieval, particularly that which concentrates on rhythmic instruments. I will decide on the approach to use in implementing the system during the thesis plan phase of the thesis process. I have already studied the subject and have a pretty good idea on the direction I will take.

The task of listening to a live drum audio and transcribing the drum notes in a way that a computer can learn from it is a machine learning -heavy process. Two main obstacles have to be solved for the system to perform adequately.

$\neq$ Drum-Off is a gamelike platform for competing in music with a computer opponent. Learning to play whilst battling with an opponent adds an another layer of interest on top of normal learning programs.

First the user is asked to perform a soudcheck

•Record Soundcheck , store for each drumhit the frequency mean and four frames of MFCC for NN to learn from. •Estimate f from recorded drum templates •Estimate filters to use for each drum •Train NN to recognise different drums from each other •Store a drum object that holds the relevant data of each drum in the drumkit

When the soundcheck is finished and the system is able to recognise different drums adequately the game can begin.

•Record the user playing •Transcribe performance •Train NN based on recorded performances •Play comeback

1.1 The Drum Kit

A normal drum kit has two types of drums, skinned membranophones ,where the skin is usually plastic. Common membranophones are the kick drum, snare or tom-toms, and plate drums such as the hi-hat, ride cymbal or crash cymbals. The kit can have special percussive parts also, but many of these still fall into either category.

Drums are usually played with sticks or brushes, the kick drum is played with feet and the hi-hat is also controlled or played with feet

The kick drum is the drum that usually defines the up-beat or the "one" in drum sequences. It is usually the largest and the lowest frequency drum of the kit with the maximum energy usually between 20 and 120 hz.

The Snare drum is a skinned drum that has snare wires spanning the bottom skin. It is usually played on the down beat of a basic drum sequence. It is a mid range frequency skinned drum with the majority of the energy between 100 and 2000hz. The snares make a white noise like layer on top of a normal skinned drum sound and shorten the resonance of the drum. Snare drum is usually tuned very high compared to i.e. the tom toms, this adds high transient ringing to the sound which is sometimes attenuated by damping the top head in some fashion.

Tom-toms range from small to large skinned drums, they can be very small with high tune or as large as a kick drum with a similar frequency range. Tom-toms are usually played in drum fills, short expressive bursts of drumming between beats or instead of plate drums in beats with kick and snare drums.

Hi-hat is two metallic plates stacked on top of each other. It is usually played with kick and snare drums in beats as quarter, eight or sixteenth notes. Hi-hat is usually also played between up and down beat. Hi-hat can be played in open or closed position and also by stomping the pedal. Closed sound is short "tsk" and the open sound is a long "tsss", stomp is a short "chak". All hi-hat sounds are high pitched with a lot of random transient noise.

Cymbals are metal plates used for accenting hits or as an alternative to the hi-hat. The Ride cymbal is usually used as an alternative to the hi-hat and it delivers a clear ping if played with a drumstick tip. Crash cymbals are usually used as accent drums and they deliver a long transient rich sound, hence the name "crash."

1.2 Automatic Drum Transcription

Automatic Drum Transcription (ADT) is fairly succesful from drum only audio. This special case makes the task even easier with the assumption that a single drummer plays only a single drum kit for the duration of the game. So transcribing said drum set can be done with a relatively low amount of training data. We request a set of samples from the player and from these samples we construct a prior vector that represents each drums mean frequency response. These prior vectors can be used to deconstruct source spectra with for instance Principal Component Analysis (PCA), Independent Component Analysis (ICA), Independent Subspace Analysis(ISA) or Non-Negative Matrix Factorization (NMF.)

1.2.1 Source separation techniques

Source Separation techniques approach ADT accordingly:

1. Transform the signal to magnitude spectrum.

2. Separate components corresponding to different drums present in the spectrum.

3. Detect onsets in spectrums of the separated components.

PCA aims to find the underlying components the signal is made of.

ICA Does the same but independent yo!

ISA Is PCA followed by ICA

PSA removes PCA step by learning prior f from a set of drum samples and then applying ICA to derive independent prior subspaces.

Independent Component Analysis (ICA) was proposed by it starts from the assumption that a signal is a composite of n subsignals that are mixed together. Principal Component Analysis is first applied to separate defining templates from the audio and ISA is then used to derive independent ISA approach to DT

1.2.2 Nonnegative Matrix Factorization of a Magnitude Spectrogram

Non-negative Matrix Factorization presents the following problem:

Given the non-negative matrix **X** find the factorization to non-negative matrix factors **W** and **H** that minimizes the cost function D associated with the factorization:

$$\min_{W,H} D(X||WH) \tag{1}$$

Where $X \in \mathbb{R}^{I \times T}$, $W \in \mathbb{R}^{I \times K}$ and $H \in \mathbb{R}^{K \times T}$.
The cost functions most commonly used include the Eulidean Distance EU, Kullback-Leibler Divergence KL and the Itakura-Saito Divergence IS. All of these cost functions belong to a subset of Bregman Divergences known as beta-divergences:

$$D_\beta = \begin{cases} \frac{x}{y} - \log\frac{x}{y} - 1, & \beta = 0 \\ x(\log\frac{x}{y} + y - x), & \beta = 1 \\ \frac{x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}}{\beta(\beta-1)}, & \beta \in \mathbb{R} \setminus \{0,1,2\} \end{cases} \tag{2}$$

$$D(X||\hat{X}) = \sum (X(i,k) \log \frac{X(i,k)}{\hat{X}(i,k)} - X(i,k) + \hat{X}(i,k)) \tag{3}$$

Update rules according to Lee and Seung:

$$H \leftarrow H.* \frac{W^\top (X./(WH))}{W^\top \boldsymbol{1}} \tag{4}$$

My update formula

$$H \leftarrow H.* \frac{W^\top X.*(WH)^{\beta-2}}{W^\top (WH)^{\beta-1} + \sigma} \tag{5}$$

1.2.3 Arbitrary separation of source audio

1.2.4 Match and adapt

1.2.5 Segment and Classify

Features to use[viite] MFCC?, MFCCDelta, Kurtosis, Skewness, ZCR ,Spectral Centroid.

1.2.6 Onset detection

How onsets are detected

1.3 Learning from the Drumming and creating new patterns in a creative fashion

2 Motivation

I myself am a professional drummer with more than 20 years, 7 albums and 1000 live performances of experience. I have also worked as a sound engineer for other bands and recorded a plethora of different acoustic and electronic performances. I am also a CS major of 20 years at the end of my studies, which have been interrupted twice by my pursuits in the field of music.

I have always felt that music learning software is mainly focused on learning to repeat existing performances. Therefore I set out to make an alternative learning tool where the main idea is not to teach people to play songs with an instrument. The aim is to teach people to play the instrument itself.

I have as a pass time activity followed different machine learning fields and naturally felt drawn to any that have something to do with music. As I studied how to read and annotate live music I found that my interests are already widely studied in the Musical Information Retrieval community ([5], [6]). As I read through some of the recent papers I felt that this is something I understand and something that I can in the future contribute to.

3 Related Work

There are no similar gamelike systems commercially available at the time of writing this thesis. Most of the algorithms and approaches used in programming this system are widely studied but with a slightly more general and analytical approach. The boundaries this game sets on the data it uses makes is a special case in the research and therefore a novel template to test ideas on a small subset of the general problem.

3.1 Automatic Drum Transcription

References references...

4 Drum-Off System Architecture Overview

4.1 Game Setup Process

Before the player can compete with the computer the computer has to learn the drummers playing style and equipment. The system must also be controlled by the user either by touch, keyboard or mouse; or by audio.

4.1.1 Audio Controlled UI

Using audio as input allows the player to leave the microphone in stationary position so that the audio levels are constant throughout the session. The instrument most suitable for the task is the hi.hat, more precisely the hi-hat stomp. A hi-hat stomp is mainly used to keep or communicate tempo and in a drum-off it is commonly used to relay information to the other participant.. When a drum pattern reduces to consecutive hi-hat stomps the player hands over the turn to the other player.

The user is asked record n hi-hat stomps, we chose n to be 32. From these samples we learn a prior frequency subspace f_p by k-means classifier over a 24 band bark frequency scaled spectrum. We get the frequencies of each sample by taking f from each frame where onset is detected. The Shot Time Fourier Transform (STFT) is a product of frequency and energy at time t, $X_t = \sum_t^n f_t t_t$. From the STFT we can extract the frequency vector at the onset times $f_t = X_t$

4.1.2 Sound Check Phase

The player is first asked to specify the number and type of drum in the drum kit used in the game. The player is then asked to play every drum for a period of time. Each separate take is recorded and the signal is transformed into a Short Time Fourier Transform where the frame size is 2048 and hop size is 441. This spectrogram is reduced to 24 frequency bins via rectangular filters scaled to corresponding Bark frequencies, this reduction has been found useful for reducing computation time without affecting the system performance[viitteet]. Onsets are extracted from this filtered spectrogram with [madmom]. Onset algorithm is run several times while annealing the threshold of the algoritm. When a previously fixed number of onsets are found the threshold is kept for later use. In the sound check phase we only allow onsets that are at least 50 ms from each other. The sound check phase is not yet competition and the drummer is expected to know how a sound check is done.

For every drum a prior frequency template is extracted from found onset events. Ten consecutive frames frequency bins energies are averaged to form the frequency prior. This prior is used as Wp in the NMF algoritm.

From every onset time we gather a set of features to use in our NN classifier. From four frames starting at the onset time MFCC, MFCCDelta, Skewness, Kurtosis, Spectral Centroid and Zero Crossing Rate are calculated.

Drum samples are also stored for generating composite drum hits. These composites are used for training the NN classifier and are generated automatically. The player coud be asked to play all combinations of drums as well but that would make the sound check tedious. Drum composite signals are generated by summing the signal extracted from combinations of two recorded drums and dividing the result by 2. The same feature set is then calculated for each of these composites and added to the training set of the NN classifier.

The Drum Object stores all of the gathered information :

threshold The threshold for onset detection

name The system name of the drum

probability_threshold The threshold for the NN classifier

templates The features of all the hits of the drum

midinote The midi note roughly corresponding to the drum

samples The audio samples extracted at onset times

frequency_pre The prior frequency template

4.2 Non Negative Matrix Factorization with sparsity constraint

We use Non Negative Matrix Factorization with a sparsity constraint to decompose the spectrogram to amplitude and frequency vectors corresponding to each drum.

4.3 Neural Network classifier

We prune the onset activations further with a NN classifier.This classifier is trained on 192 features that are MFCC0 MFCC1 MFCC2 MFCC3 Deltas.. Spectral kurtosis

4.4 Drum Controlled User Interface

The user interface will feature full audio control in order to avoid using hands for other than drumming. Stomping the Hi-Hat number of times will activate shit in the UI.

4.5 Capturing Drum Audio

The audio is captured and analysed as the audio arrives. When an onset is detected the relevant features are extracted at that position

4.6 Transcribing the Captured Audio

The stored onsets and their relevant features are submitted to a NN and the results are stored to a midi file.

4.7 Quantizing the transcription

The output of the transcription is quantized to a fixed tempo so that all transcribed parts can be compared to each other.

4.8 Learning New Patterns Based On Stored Transcriptions

A Long Short-Term Memory Neural network is used to traverse the midinote space and learn to make new patterns based on that.

4.9 Playing the Pattern to the User

5 Automatic Drum Transcription

Results for 4 drums, kick, sn, chh, ohh
Precision: 0.9660
Recall: 0.9297
F-score: 0.9468

Results for 8 drums, set - stomp
Precision: 0.9091
Recall: 0.8281
F-score: 0.8645

Results for all drums:
Precision: 0.8528
Recall: 0.8299
F-score: 0.8245

6 Learning New Patterns From Transcriptions

7 Results

References

[1] Richard Vogl, Matthias Dorfer, Gerhard Widmer, Peter Knees. Drum Transcription via Joint Beat and Drum Modeling using Convolutional Recurrent Neural Networks. In *18th International Society for Music Information Retrieval Conference, Suzhou, China, 2017.*

[2] Christian Dittmar, Daniel Gärtner. Real-time Transcription and Separation of Drum Recordings Based On NMF Decomposition. In *Proc. of the 17thInt. Conference on Digital Audio Effects (DAFx-14), Erlangen, Germany, September 1-5, 2014*

[3] Böck, Sebastian, Korzeniowski, Filip, Schlüter, Jan, Krebs, Florian, Widmer, Gerhard. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia, pages 1174-1178, October, 2016*

[4] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, Oriol Nieto. librosa: Audio and Music Signal Analysis in Python. In *Proc. of the 14th Python In Science Conf. (SCIPY 2015)*

[5] http://www.ismir.net/

[6] http://www.music-ir.org/mirex/wiki/