Live Drum Separation Using Probabilistic Spectral Clustering Based on the Itakura-Saito Divergence

Eric Battenberg[1,2], Victor Huang[1], David Wessel[1,2]

[1] *University of California, Berkeley, CA, USA*

[2] *Center for New Music and Audio Technologies, Berkeley, CA, USA*

Correspondence should be addressed to Eric Battenberg (`ericb@eecs.berkeley.edu`)

ABSTRACT

We present a live drum separation system for a specific target drumset to be used as a front end in a complete live drum understanding system. Our system decomposes drum note onsets onto spectral drum templates by adapting techniques from non-negative matrix factorization. Multiple templates per drum are computed using a new Gamma mixture model clustering procedure to account for the variety of sounds that can be produced by a single drum. This clustering procedure imposes an Itakura-Saito distance metric on the cluster space. In addition, we utilize "tail" templates for each drum which greatly improve the separation accuracy when cymbals with long decay times are present.

1. INTRO

The purpose of this work is to present the initial steps we have made toward the development of a live drum understanding system, which will incorporate drum-wise source separation, low-level beat tracking, and high-level rhythmic understanding. In this paper, we focus on the first component of the system: live drum separation.

Source separation is an important problem in the field of music information retrieval that attempts to isolate sound sources (instruments, notes, sound samples, noise, etc.) as separate signals[1]. This can greatly enhance music analysis tasks such as genre/artist classification, beat-tracking, lyrics synchronization, and automatic transcription. In this paper we utilize an existing approach created for a source separation technique called non-negative matrix factorization (NMF) [2] to compute the contribution of individual drums to a spectrogram slice.

Our motivation for developing live drum separation techniques stems from the fact that the activation (time and amplitude) of individual drum events is much more informative than non-drum-specific onset locations when determining perceptual beat locations and classifying rhythmic patterns. For example, in rock and jazz music, the drummer typically keeps time on one of the cymbals by playing some sort of repeating pattern (e.g. straight eighth or quarter notes or a swung ride pattern), while the snare and bass drum will typically accentuate certain beats within a measure.

Approaches to feature-based classification of solo drum onsets are covered in [3] and [4]. In [5], Paulus uses NMF techniques to transcribe solo drums. The spectral templates used for the drums are spectrum averages computed from a large number of training samples, so they are more generally applicable but less specific to a particular drum kit. Yoshii [6] uses an adaptive approach to drum transcription by starting with general drum templates, matching the templates to drum onsets, and then refining the templates to better match the input drums.

In our live drum separation system, we have the luxury of a single target drum kit that we can train our templates to. It is assumed that sound samples of the individual drums can be gathered during sound check or sometime before a performance, so we can specialize our templates to the specific drums that will be used.

Our approach trains multiple templates per drum to account for the vast array of sound qualities that can be produced on a particular drum. For example, a snare drum can produce a large variety of sounds depending upon the striking velocity and whether the drum stick hits the center, edge, or rim of the drum.

In addition, our system incorporates the use of "tail" templates to address the problem that occurs when long-decay percussion instruments (e.g. cymbals) overlap with successive drum onsets, thereby degrading the quality of separation.

1.1. System Overview

Our drum separation system is comprised of four primary components: onset detection, spectrogram slice extraction, drum template training, and non-negative vector decomposition (NVD). Onset detection is used to locate significant rhythmic events to be spectrally analyzed. During the training of a particular drum, spectrogram slices at onset locations are clustered using a probabilistic formulation of the Itakura-Saito distance[7]. The cluster centers are then used as time-frequency templates for each trained drum. Live drum separation is accomplished by locating drum events and decomposing the spectrogram slices of each event as a non-negative mixture of drum templates. A block diagram of the system is shown in Figure 1.

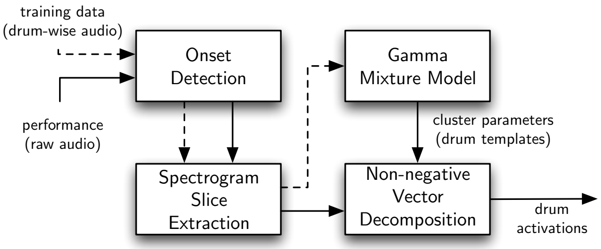


Fig. 1: *System overview. The dotted connections are present during training, while the solid connections are used during live performance*

In the following four sections, we describe the approach used for onset detection, spectrogram slice extraction, drum template training, and non-negative mixture decomposition, respectively.

2. ONSET DETECTION

Our approach to onset detection is similar to that presented in [8], in that we compute our onset detection function using the differentiated log-energy of multiple sub-bands. We used a larger number of sub-bands (20 per channel) spaced according to the Bark scale. The energy in the ith sub-band of the nth frame of samples, $x_i(n)$, is processed using mu-law compression as a robust estimation of pure logarithmic compression, according to eq. (1).

$$y_i(n) = \frac{\log(1+\mu x_i(n))}{\log(1+\mu)} \tag{1}$$

We use a large value of $\mu = 10^8$ in the above in order to enhance the detection of more subtle drum notes. In order to limit the detection of spurious, non-musical onsets, $y_i(n)$ is smoothed using a linear phase Hann window with a 3dB cutoff of 20Hz to produce $z_i(n)$. We time-differentiate and half-wave rectify $z_i(n)$ to get $\mathrm{d}z_i(n)$. To arrive at our final onset detection function, $o(n)$, we take the mean of $\mathrm{d}z_i(n)$ across sub-bands.

To pick onset locations from the detection function we first find the local maxima of $o(n)$. Then, local maxima are labeled as onsets if they are larger than the dynamic threshold, T_{dyn} [eqs. (2–4)]. In addition, the detection function must have dropped below the local dynamic threshold since the last detected onset. These criteria were chosen with the causality requirement of our system in mind.

We have experimented with many types of dynamic thresholds. Typical ways to compute a dynamic threshold are covered in [9]. The dynamic threshold we utilize here is shown in eq. (2), where $P_i(n)$ represents the ith percentile of the set

$$\vec{O}(n) = \{o(n), o(n-1), \ldots, o(n-N)\},$$

where N is chosen so that $\vec{O}(n)$ contains 1 second worth of detection function values.

$$T_1(n) = 1.5(P_{75}(n) - P_{25}(n)) + P_{50}(n) + 0.05 \tag{2}$$

However, due to causality requirements, we also needed a way to quickly increase the threshold after a passage of silence. This is accomplished using a percentage of the maximum value, $P_{100}(n)$, of $\vec{O}(n)$ as shown in eq. (3).

$$T_2(n) = 0.1P_{100}(n) \tag{3}$$

The final dynamic threshold is calculated as the power mean between $T_1(n)$ and $T_2(n)$ [eq. (4)]. The power mean is chosen to simulate a softened maximum between the two values.

$$T_{dyn}(n) = \left(\frac{T_1(n)^p + T_2(n)^p}{2}\right)^{1/p} \tag{4}$$

We use a value of $p = 2$, which results in a fairly soft maximum. As $p \to \infty$, the power mean approaches the true maximum.

An overview of the complete onset detection system is shown in Figure 2.

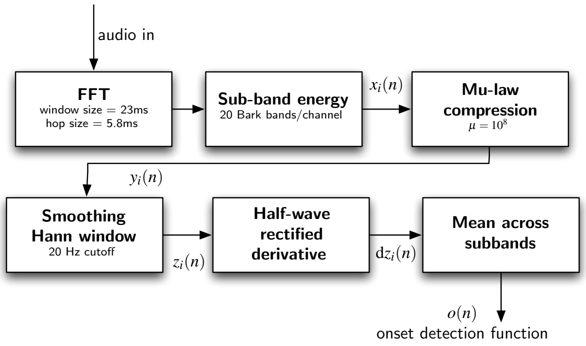


Fig. 2: *Diagram of onset detection system. The audio sample rate is 44.1kHz*

3. EXTRACTION OF SPECTROGRAM SLICES

The primary feature we use to perform drum separation is a spectrogram slice that is local to a detected onset event. In previous work employing non-negative matrix factorization (NMF), the features used in drum separation are simply the local spectra which contain no time-based transient information [5]. Like in Yoshii's template-based approach to drum transcription [6], we use a number of adjacent spectral frames to detect the presence of drums. This approach allows us to capture characteristics of the attack and decay of a particular drum.

The short-time spectra that make up a spectrogram slice are extracted using a 23ms Hann window with hopsize 5.8ms. We then compute the energy of the FFT components of this window of samples. This energy spectrum is then dimensionality-reduced by summing the energy into 40 sub-bands per channel spaced according to the Bark scale. In [10], we have shown that NMF used for drum source separation can be sped up significantly using this sub-band-based, dimensionality-reduction approach with no loss of separation quality.

A spectrogram slice is comprised of 100ms worth of adjacent spectral frames (about 17 frames). The window of frames used in the slice begins about 33ms before the detected onset and ends 67ms after the onset. This offset helps to ensure that the window contains the entire attack of the onset. In addition, during training, we extract a further 100ms of spectrum frames after the initial slice to be used in computing "tail" templates. When performing non-negative vector decomposition (NVD) on input spectrogram slices, these tail templates serve as "decoys" to prevent the long decay of a previous drum onset (e.g. cymbal crash) from incorrectly showing up in the drum activations of the current onset. These tail templates account for the intrusion of these long decays into the current spectrogram slice, and their activations are ignored in the final drum separation.

The head and tail spectrogram slices are both smoothed across frames using a 29ms Hann window and downsampled by a factor of 2 in order to allow the slices to be more robust to variations in microtiming. Finally, the square root is taken to move the data from the energy domain to the magnitude domain. During training, these head and tail slices are fed to the drum template training stage covered in the following section. During live performance, only the head slices are used as input to the drum separation covered in Section 5.

4. TRAINING DRUM TEMPLATES

To model the time-frequency characteristics of individual drums given a target drum set and specific microphone placements, we cluster the spectrogram slices extracted from the training data for each drum. It is assumed that we have access to isolated, single-drum samples for each drum as is typically the case during a pre-performance sound check. Once we have accumulated a training set of spectrogram slices (both head and tail for each detected onset) for a particular drum, we follow the agglomerative clustering procedure outlined in this section to arrive at a small number of spectrogram templates that compactly describe the range of sounds that can be produced by the drum. The head and tail slices for each drum are clustered separately to produce separate sets of templates.

4.1. Clustering with the Itakura-Saito Divergence

The speech recognition community has made frequent use of Gaussian Mixture Models (GMMs) for modeling speaker-specific data[11][12]. Clustering data using GMMs enforces a squared Euclidean distance measure when determining the cluster membership of individual observations. As shown by Banerjee [13], mixture models built from other exponential-family priors utilize other distance measures from the family of Bregman divergences. An important Bregman divergence to the speech and music community is the Itakura-Saito distance (IS distance), which can be used as a measure of the perceptual distance between audio spectra[7][14]. Banerjee shows that the IS distance is the Bregman divergence corresponding to a mixture model composed of exponential distributions[13]. In this work we utilize the family of Gamma distributions, which can also be shown to enforce an IS distance measure, as our mixture model prior.

4.2. The Gamma Distribution

The probability density function of the univariate Gamma distribution can be written as follows:

$$\begin{aligned} \mathrm{p}(y|\lambda,k) &= y^{k-1}\frac{\lambda^k e^{-\lambda y}}{\Gamma(k)}, \quad y \geq 0;\ \lambda, k > 0 & (5) \\ E[y] = \mu &= k/\lambda & (6) \end{aligned}$$

Where the mean is shown in eq. (6). The Gamma distribution generalizes the Erlang distribution (which models the sum of k iid exponential random variables) to continuous $k > 0$. Figure 3 shows the Gamma distribution with constant mean for various values of shape parameter k.

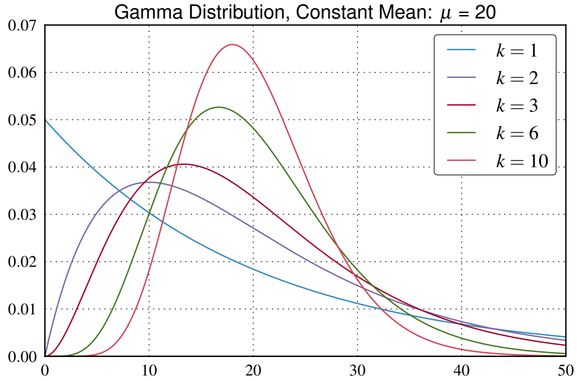


Fig. 3: *The Gamma distribution for various values of k and constant mean.*

As shown in [13], we can write the Gamma distribution in its Bregman divergence form as shown in eq. (7). The Bregman divergence inside of the exponential, $d_{\mathrm{IS}}(y,\mu)$, is the IS distance.

$$\begin{aligned} \mathrm{p}(y|\lambda,k) &= \exp\left(-k d_{\mathrm{IS}}(y,\mu)\right) b_0(y) & (7) \\ &= \exp\left\{-k\left(\tfrac{y}{\mu} - \log\tfrac{y}{\mu} - 1\right)\right\} b_0(y) & \\ b_0(y) &= \frac{e^{-k}k^k y^{-1}}{\Gamma(k)}, \quad \mu = k/\lambda & (8) \end{aligned}$$

To model multidimensional data, we construct a multivariate Gamma distribution from independent Gamma distributions:

$$\begin{aligned} \mathrm{p}(\vec{y}|\vec{\lambda},k) &= \prod_{i=1}^{M}\frac{\lambda_i^k y_i^{k-1} e^{-\lambda_i y_i}}{\Gamma(k)} & (9) \\ &= \frac{|\vec{\lambda}|^k |\vec{y}|^{k-1}\exp(-\vec{\lambda}\cdot\vec{y})}{(\Gamma(k))^M} & (10) \\ E[\vec{y}] = \vec{\mu} &= k/\vec{\lambda} & (11) \end{aligned}$$

where $|\cdot|$ denotes the product of the elements in a vector, $\vec{a}\cdot\vec{b}$ denotes a dot product between vectors, and division involving vectors is performed element-wise.

The log-likelihood is:

$$\begin{aligned} \log\left(\mathrm{p}(\vec{y}|\vec{\lambda},k)\right) &= k\log|\vec{\lambda}| + (k-1)\log|\vec{y}| & \\ &\quad -\vec{\lambda}\cdot\vec{y} - M\log\left(\Gamma(k)\right) & (12) \end{aligned}$$

4.3. The Gamma Mixture Model

Our Gamma mixture model (ΓMM) has the following distribution for observations, $\vec{y}_n$:

$$\begin{aligned} \mathrm{p}(\vec{y}_n|\theta) &= \sum_{l=1}^{K}\pi_l \mathrm{p}(\vec{y}_n|\vec{\lambda}_l,k) & (13) \\ \pi_l &= \mathrm{p}(x_n = l) & (14) \end{aligned}$$

where $\theta = \{\vec{\lambda}_l, \pi_l\}_{l=1}^K$, $\vec{y}_n \in \mathbb{R}^M$, and x_n is a hidden variable that denotes which mixture component $\vec{y}_n$ was generated from.

In order to learn the ΓMM parameters, θ, from a set of training data, $\mathbf{Y} = \{\vec{y}_n\}_{n=1}^N$, we employ the Expectation-Maximization (EM) algorithm[15]. Because maximizing the log-likelihood of the parameters, $\log \mathrm{p}(\mathbf{Y}|\theta)$, is intractable, instead the EM algorithm iteratively optimizes an auxiliary function:

$$Q(\theta|\theta^{(t)}) = E\left[\log \mathrm{p}(\mathbf{Y},\mathbf{X}|\theta)\middle|\mathbf{Y},\theta^{(t)}\right] \qquad (15)$$

where $\theta^{(t)}$ are the current parameters at iteration t, θ are the parameters to be optimized during the current iteration, and $\mathbf{X} = \{x_n\}_{n=1}^N$. For our ΓMM, we eventually arrive at the simplified expression:

$$\begin{aligned} &Q(\theta|\theta^{(t)}) = & (16) \\ &\quad N_l^* \sum_{l=1}^{K}\left\{\log\pi_l + k\log|\vec{\lambda}_l| - k\left(\vec{\lambda}_l\cdot\frac{1}{\vec{\lambda}_l^*}\right)\right\} & \\ &\quad +(k-1)\sum_{n=1}^{N}\log|\vec{y}_n| - NM\log\Gamma(k) & \end{aligned}$$

where

$$N_l^* = \sum_{n=1}^{N} \mathrm{p}(x_n = l|\vec{y}_n, \theta^{(t)}) \quad (17)$$

$$\vec{\lambda}_l^{\,*} = \frac{kN_l^*}{\sum_{n=1}^{N} \vec{y}_n \mathrm{p}(x_n = l|\vec{y}_n, \theta^{(t)})} \quad (18)$$

Eqs. (17,18) rely on the posterior $\mathrm{p}(x_n = l|\vec{y}_n, \theta^{(t)})$ which can be calculated using Bayes' rule:

$$\mathrm{p}(x_n = l|\vec{y}_n, \theta^{(t)}) = \frac{\mathrm{p}(\vec{y}_n|x_n = l, \theta^{(t)})\mathrm{p}(x_n = l|\theta^{(t)})}{\mathrm{p}(\vec{y}_n|\theta^{(t)})} \quad (19)$$

$$= \frac{\pi_l \exp\left(-k\, d_{\mathrm{IS}}(\vec{y}_n, \vec{\mu}_l)\right)}{\sum_{j=1}^{K} \pi_j \exp\left(-k\, d_{\mathrm{IS}}(\vec{y}_n, \vec{\mu}_j)\right)} \quad (20)$$

$$= \frac{\pi_l |\vec{\lambda}_l|^k \exp(-\vec{\lambda}_l \cdot \vec{y}_n)}{\sum_{j=1}^{K} \pi_j |\vec{\lambda}_j|^k \exp(-\vec{\lambda}_j \cdot \vec{y}_n)} \quad (21)$$

where $\vec{\mu}_l = k/\vec{\lambda}_l$. Notice that in eq. (20), the posterior (or cluster membership probabilities) can be viewed as a weighted sum of the IS distance (rescaled by the exponential function) between an observation and the cluster mean. Banerjee shows that clustering in this way aims to minimize the expected IS distance between a cluster mean and the cluster members[13]. The more simplified posterior expression that we employ is shown in eq. (21).

Now we can compute updated optimal values of the parameters using $Q(\theta|\theta^{(t)})$, which is maximized with the following parameter values:

$$\pi_l = \frac{N_l^*}{N}, \quad \vec{\lambda}_l = \vec{\lambda}_l^{\,*} \quad (22)$$

with N_l^* and $\vec{\lambda}_l^{\,*}$ from eqs. (17,18).

We continue to alternate between updating the posterior using eq. (21) and updating the parameters using eqs. (17,18,22) until $Q(\theta|\theta^{(t)})$ converges.

4.4. Agglomerative Clustering with Gamma Mixture Models

The ΓMM training procedure covered in 4.3 relies on a fixed number (K) of mixture components or clusters. In order to choose a value of K that complements the training data, we use the agglomerative clustering approach in [16]. This approach starts with an initial maximum value of $K = K_0$ and iteratively merges similar clusters until we are left with a single cluster ($K = 1$).

The agglomerative procedure begins by initializing the K_0 cluster means, $\vec{\mu}_l = k/\vec{\lambda}_l$, to be equal to randomly chosen data points from the training set, $\mathbf{Y}$. The initial cluster prior probabilities, π_l, are uniformly initialized.

Then the parameters are iteratively optimized for this value of K until convergence. Upon convergence, we merge the two most similar clusters, decrement K, and again update the parameters until convergence. The similarity of two clusters (l,m) is computed using the following distance function:

$$D(l,m) = Q(\theta^*|\theta^{(t)}) - Q(\theta^*_{(l,m)}|\theta^{(t)}) \quad (23)$$

where θ^* is the set of parameters that optimizes $Q(\theta|\theta^{(t)})$, and $\theta^*_{(l,m)}$ is the optimal set of parameters with the restriction that:

$$\vec{\mu}_l^{\,*} = \vec{\mu}_m^{\,*} = \frac{\pi_l \vec{\mu}_l + \pi_m \vec{\mu}_m}{\pi_l + \pi_m} \quad (24)$$

Using eqs. (11,17,17,18,24) along with the convergence assumption of $\theta^* = \theta^{(t)}$, eq. (23) simplifies to:

$$\begin{aligned} &D(l,m) = \\ &k\left\{(N_l^* + N_m^*)\left[\log\left|\frac{N_l^*}{\vec{\lambda}_l} + \frac{N_m^*}{\vec{\lambda}_m}\right| - M\log(N_l^* + N_m^*)\right]\right\} \quad (25) \\ &\quad + N_l^* \log|\vec{\lambda}_l| + N_m^* \log|\vec{\lambda}_m| \end{aligned}$$

The two clusters that minimize $D(l,m)$ are deemed the most similar and are merged by setting the merged cluster's parameters to:

$$\vec{\mu_{l,m}} = \frac{\pi_l \vec{\mu}_l + \pi_m \vec{\mu}_m}{\pi_l + \pi_m}, \quad \pi_{l,m} = \pi_l + \pi_m \quad (26)$$

However, before merging, the parameter set θ^* for the current value of K is saved along with a measure of how efficiently the parameters describe the training data. As in [16], we use the Minimum Description Length (MDL) introduced by Rissanen[17].

$$\begin{aligned} &\mathrm{MDL}(K,\theta) = \\ &\quad -\sum_{n=1}^{N} \log\left(\sum_{l=1}^{K} \mathrm{p}(\vec{y}_n|\vec{\lambda}_l)\pi_l\right) + \tfrac{1}{2}L\log(NM) \quad (27) \end{aligned}$$

with L equal to the number of free parameters.

$$L = KM + (K-1) \quad (28)$$

For our ΓMM, eq. (27) simplifies to:

$$\begin{aligned} &\mathrm{MDL}(K,\theta) = \quad (29) \\ &-\sum_{n=1}^{N}\left[\log\left(\sum_{l=1}^{K} \pi_l |\vec{\lambda}_l|^k \exp\left(-\vec{\lambda}_l \cdot \vec{y}_n\right)\right)\right] \\ &-\sum_{n=1}^{N}\left[(k-1)\log|\vec{y}_n|\right] + NM\log\Gamma(k) + \tfrac{1}{2}L\log(NM) \end{aligned}$$

So for each K, we run the EM algorithm until $Q(\theta|\theta^{(t)})$ converges, then save the parameters θ^* along with MDL(K,θ^*). We merge the two most similar clusters using eqs. (25,26), and then repeat the EM algorithm for the new smaller K. Once we reach $K=1$, we choose the K and corresponding set of parameters with the minimum MDL.

We then compute the mean vector of each cluster, $\vec{\mu}_l$, from the winning cluster parameters, $\vec{\lambda}_l$ using eq. (11). These mean vectors are used as our drum templates.

5. DRUM SEPARATION

Now that we have trained a number of head and tail templates for each drum, we can begin to compute the activations of these templates from live drum audio input. In order to make the amplitude of each template activation meaningful, we normalize the head templates so that all of the templates for a particular drum have the same approximate loudness. To do this, we normalize the head templates for a single drum to have the same energy as the template with the maximum energy for that drum. No normalization is required for the tail templates, since as you will see, we discard their activations.

5.1. Non-negative matrix factorization

Using the tail templates and normalized head templates, we can decompose live drum onsets as non-negative linear combinations of the templates. The activations of all the head templates corresponding to a single drum are summed to get the activation of that drum, and the activations of tail templates are discarded. To determine the activation of each template for a particular input spectrum, we employ multiplicative algorithms used in non-negative matrix factorization (NMF)[2]. NMF poses the problem:

$$\min_{W,H} D(X||WH), \quad W \geq 0, H \geq 0 \tag{30}$$

$$X \in \mathbb{R}^{M\times F},\, W \in \mathbb{R}^{M\times T},\, H \in \mathbb{R}^{T\times F} \tag{31}$$

where the matrix inequality constraint applies to every element, and X is the matrix to be factorized. $D(X||WH)$ is an associated cost function that measures the error between X and its factorization WH. Typical NMF cost functions include Euclidean distance, KL divergence, and IS distance [14]. These cost functions all belong to the family of Bregman divergences and, additionally, belong to the subset of the Bregman divergence called the *Beta divergence*[18].

$$d_\beta(x||y) = \begin{cases} \frac{x}{y} - \log\frac{x}{y} - 1, & \beta = 0 \\ x(\log\frac{x}{y} + y - x), & \beta = 1 \\ \frac{x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}}{\beta(\beta-1)}, & \beta \in \mathbb{R}\backslash\{0,1\} \end{cases} \tag{32}$$

The Eucliden distance, KL divergence, and IS distance are Beta divergences with $\beta = 2$, 1, and 0, respectively. In order to compute the Beta divergence between two matrices, we simply sum the Beta divergence between corresponding elements in the matrices.

5.2. Non-negative vector decomposition

Our drum separation task does explicitly carry out NMF on the input drum spectra. Because we have already trained drum spectra templates, we have already determined the matrix W, with the templates contained in its columns. Because W is not being optimized, the problem in (30) decouples into F independent optimization problems

$$\min_{\vec{h}_i} D(\vec{x}_i||W\vec{h}_i), \quad \vec{h}_i \geq 0 \tag{33}$$

$$\vec{x}_i \in \mathbb{R}^M,\, W \in \mathbb{R}^{M\times T},\, \vec{h}_i \in \mathbb{R}^T \tag{34}$$

where $\vec{x}_i$ and $\vec{h}_i$ are the ith columns of X and H, respectively.

To pursue an optimal solution to (33), we use multiplicative, gradient-based updates first introduced by Lee and Seung [2] for Euclidean and KL divergence cost functions and later generalized to Beta divergences [18][19].

$$\vec{h}_i \leftarrow \vec{h}_i . \frac{W^T((W\vec{h}_i)^{.\beta-2} . \vec{x}_i)}{W^T(W\vec{h}_i)^{.\beta-1}} \tag{35}$$

where dotted operations and division are carried out element-wise. In order to pursue an optimal mixture of templates in terms of the IS distance, we can iterate on the above update with $\beta = 0$; however, $d_\beta(x||y)$ is not convex in terms of y for $\beta = 0$, so there is the possibility of convergence to local minima. For $1 \leq \beta \leq 2$, $d_\beta(x||y)$ is convex in y. Bertin el al.[20] use this fact to design a "tempering" algorithm which begins running the updates with β in the convex range and then slowly reduces β to 0. This approach is shown to help avoid local minima and reduce the final converged cost.

6. RESULTS

6.1. Test Setup

In order to generate audio signals with solid ground truth information, we created our test data using Roland electronic *V-drums*[21] to record MIDI data for each drum performance. The MIDI data was then used to generate a two-channel, 44.1kHz, 24-bit audio version of the performance using Toontrack's high-quality, multi-sampled, multi-microphone drum sampling engine *Superior Drummer 2.0* [22]. The MIDI data generated by the V-drums includes separate MIDI notes for snare and snare rim, hi-hat bow and edge (both open and closed), bass drum, and ride cymbal bow and bell. The test audio signals were generated with each drum panned to correspond to its physical location within the stereo field. No compression or filtering was done on any of the drum samples.

The training data is divided into 5 distinct classes: bass drum, snare drum, closed hi-hat, open hi-hat, and ride cymbal. For each class, we train on notes played at a variety of dynamic levels and articulations (e.g. hi-hat bow vs. edge). The number of training examples for each drum class is shown in Table 1.

The drum templates were trained with Gamma shape parameter $k = 2$ for head templates and $k = 1$ for tail templates. Larger values of k tend to produce a larger number of clusters, and we decided that a smaller number of tail templates was reasonable for our purposes. We test separation quality for a few different training configurations. The parameters we varied during training were as follows:

$K_{0,H}$: Initial head clusters per drum

$K_{0,T}$: Initial tail clusters per drum

First, we test using the optimal number of clusters determined by the agglomerative clustering process, with the initial number of clusters equal to the number of training observations ($K_{0,H} = K_{0,T} = N$). The optimal number of templates for each drum class, K_H^*, K_T^*, are shown in Table 1. To test the influence of using multiple templates per drum, we also test with the number of head and/or tail templates limited to one per drum ($K_{0,H} = 1$, $K_{0,T} = 1$). With a single cluster, the optimal template is simply the mean of all of the training data. To test the benefit of the tail templates, we also run the tests without the use of tail templates ($K_{0,T} = 0$). For the NVD drum separation, the best results were obtained using the updates from eq. (35) with $\beta = 0$, which corresponds to the Itakura-Saito distance; so this is the approach we used in our test system. We found no practical benefit to using the tempering algorithm described in 5.2, where β is gradually reduced to its target value.

Drum Class	Bass	Snare	Hi-hat		Ride
			Open	Closed	
Training Notes (N)	40	84	74	52	55
Head Templates (K_H^*)	3	4	3	2	2
Tail Templates (K_T^*)	2	4	3	2	2

Table 1: *Number of training notes per drum class and optimal number of head and tail templates.*

6.2. Test Data

A total of 10 drum beats were performed for our test data, containing a total of 2922 drum notes spaced over about 10 minutes of audio. The tempo of the beats varies between 75 and 160 beats per minute. Table 2 shows the occurrences of each drum class within each beat. The performed rhythms have a rock music bias with a fair amount of syncopation. Many of the beats include 16th note cymbal patterns, open hi-hats, or ghosted snare hits. Beat 4 is the lone track with a jazz/swing feel. Any notes with velocity below 40 were discarded from the MIDI tracks due to their limited audibility.

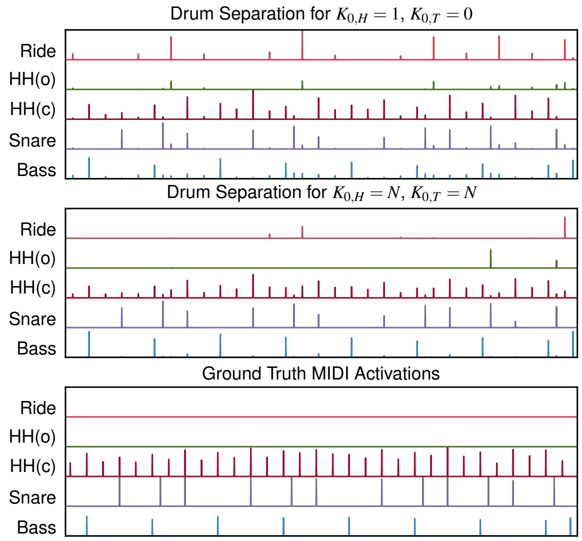
Beat	Tempo	Bass	Snare	Hi-hat		Ride	Total
				Open	Closed		
1	80	120	112	0	237	0	469
2	80	84	112	0	0	162	358
3	80	68	64	90	0	0	222
4	140	31	36	1	5	191	264
5	100	107	32	16	48	0	203
6	75	88	48	10	338	0	484
7	90	51	210	0	47	0	308
8	160	38	41	0	128	0	207
9	85	62	36	0	9	42	149
10	85	52	40	1	20	145	258
Total	-	701	731	118	832	540	2922

Table 2: *Number of ground truth drum notes by beat.*

For each test audio track and configuration of training parameters, we compute the activation of each drum class at each detected onset location using the NVD procedure outlined in Section 5. These drum activations are stored in a matrix where each row contains the activations associated with a single drum class and the columns correspond to audio frames. Example drum activation matrices are shown in Figure 4.

Ground truth drum notes were defined by the *Note On* events in the MIDI files used to generate the audio for each beat. To obtain a ground truth "drum activation matrix", the time of each event was quantized to the frame hop size, and the corresponding velocity values were scaled to the interval [0,1].

From these drum activation matrices, we compute "onset arrays" by summing the activation matrices across rows then setting any positive values to one. This gives us arrays containing binary values indicating the existence of an onset at the corresponding frame.


Drum Separation for $K_{0,H} = 1$, $K_{0,T} = 0$

Drum Separation for $K_{0,H} = N$, $K_{0,T} = N$

Ground Truth MIDI Activations

Fig. 4: *Drum activation matrices for a portion of Beat 1 (two template configurations with ground truth on bottom).*

6.3. Onset Detection Accuracy

Because the onset detector chooses which spectrogram slices to analyze, its accuracy greatly influence the efficacy of the overall system. We evaluated onset detection accuracy using precision and recall rates:

$$\text{Precision} = \left(\frac{\text{\# correctly detected onsets}}{\text{\# detected onsets}}\right) \quad (36)$$

$$\text{Recall} = \left(\frac{\text{\# correctly detected onsets}}{\text{\# actual onsets}}\right) \quad (37)$$

To count correctly detected onsets, we iterated through the ground truth onset array and attempted to match onsets in the test onset array. If the difference between a detected onset time and an actual onset time was within 4 frames (roughly 23ms), it was counted as correct. The onset detection results are shown in Table 3. The overall precision is very high indicating that a more sensitive threshold would most likely yield improved accuracy results. Beats 6, 7, and 8 contain a significant number of 16th notes played at low volume or high tempo which explains their lower recall rates.

Beat	Correct	Detected	Total	Recall	Precision
1	305	307	315	96.8	99.3
2	227	227	267	85.0	100
3	136	136	137	99.3	100
4	228	228	255	89.4	100
5	158	158	188	84.0	100
6	327	327	413	79.2	100
7	195	196	278	70.1	99.5
8	122	122	168	72.6	100
9	126	126	134	94.0	100
10	224	224	251	89.2	100
Total	2048	2051	2406	85.1	99.9

Table 3: *Onset detection accuracy*

6.4. Separation Quality

To compare the activation amplitude of correctly matched drum onsets, we used the cosine similarity, which is basically a normalized cross-correlation between vectors containing the drum activation amplitudes of matching onsets. It also represents the cosine of the angle between the two vectors.

$$S_{\cos}(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{||\vec{x}||\,||\vec{y}||} \quad (38)$$

where $||\cdot||$ indicates the 2-norm.

We computed the cosine similarity between matching onsets for each drum class. We average (weighted by number of onsets) the cosine similarity results across drums and beats to get an overall amplitude similarity for each configuration of training parameters. These results are shown in Figure 5. These results show an apparent advantage in amplitude quality when using more than one head template per drum ($K_{0,H} = N$); however; highest average cosine similarity is achieved with a single tail template.

To measure the amplitudinal significance of false drum activations, we collected all drum activations in the test matrix of amplitude greater than 0.05 that did not have a matching ground truth activation greater than 0.05 in the

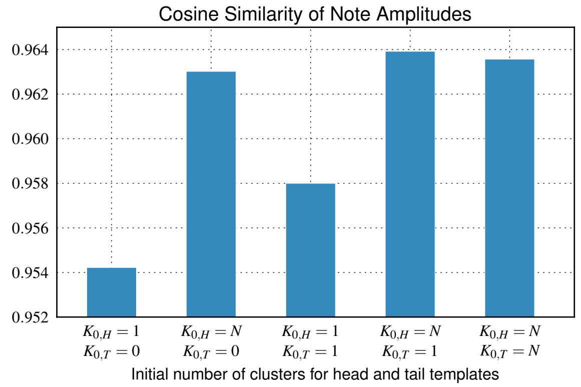


Fig. 5: *Average cosine similarity of true positive note amplitudes for different numbers of initial clusters.*

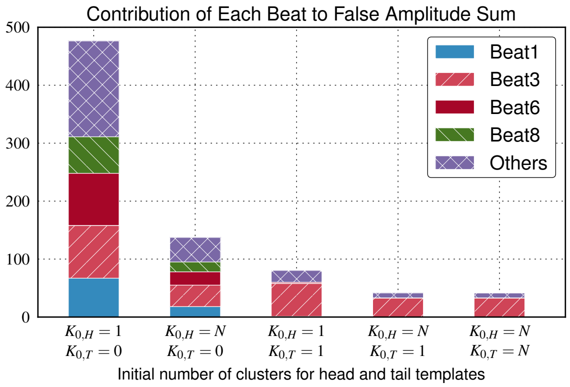


Fig. 7: *The contribution of each beat to the overall sum of false amplitudes. Beats that made up less than 10% of the total are lumped into "Others".*

same drum class. These "false activations" were counted and their amplitudes were summed. The results for each template configuration are shown in Figure 6. It is clear that using at least one tail template per drum produces significantly better results. Also, using more than one head template seems to be of significant benefit as well.

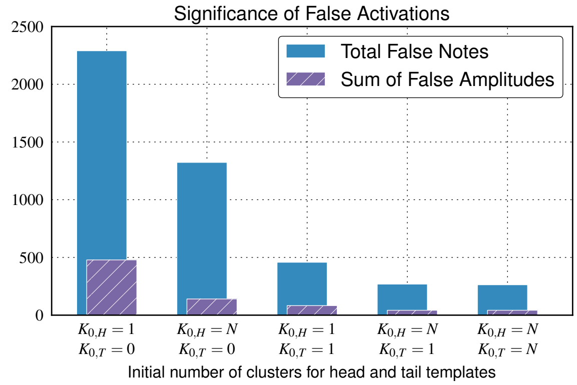


Fig. 6: *The total number of false positive notes and the sum of the corresponding activations for various template configurations.*

To get a sense for what types of false activations were being added, we present the beat-wise contribution to the sum of false amplitudes for each template configuration in Figure 7. Any contribution less than 10% of the total was lumped into the "Other" category.

The only beat that doesn't seem to significantly improve with the inclusion of tail templates or additional head templates is Beat 3. This beat features a prominent and loudly played open hi-hat which was frequently interpreted by the system as a mixture of different cymbals. Without the use of tail templates, all three cymbal classes (open and closed hi-hat and ride cymbal) are present. When including tail templates, each open hi-hat is interpreted as a combination of open and closed hi-hats, which is not necessarily a regrettable interpretation, since they are notes from the same cymbal and – with the exception of decay time – possess very similar timbral characteristics. A simple fix to this interpretation would be to assume the impossibility of simultaneous open and closed hi-hat notes from the same cymbal and zero out the least likely activation.

Without tail templates, the separation results for Beat 1 include many incorrect ride cymbal activations during what should be closed hi-hat activations. It seems that the ride cymbal template is used to explain the spectral influence of the decaying remnants of previous drum onsets. Beat 6 suffers the same phenomenon; without tail templates, the decomposition of its quick 16th note hi-hat pattern is rife with ride cymbal activations and sprinkled with open hi-hat. Beat 8 is another victim of similar unwarranted ride cymbal activations.

6.5. Closing Words

The results we have presented suggest that the non-negative decomposition of drum onsets onto spectral templates can be greatly improved by the use of at least one tail template per drum class. In addition, using multiple head templates per drum decreases false activations and improves amplitude quality in correct activations. To determine these multiple drum templates, we have proposed a Gamma mixture model, a probabilistic agglomerative clustering approach that takes the perceptual spectral similarity of training data into account and does not rely on the computationally expensive and possibly unstable covariance calculation required by Gaussian mixture models.

7. REFERENCES

[1] M. D. Plumbley *et al.*, "Automatic Music Transcription and Audio Source Separation," *Cybernetics and Systems*, vol. 33, no. 6, pp. 603–627, Sep. 2002.

[2] D. Lee and H. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems*, 2001.

[3] O. Gillet and G. Richard, "Automatic transcription of drum loops," in *Acoustics, Speech, and Signal Processing, (ICASSP '04). IEEE*, 2004, pp. iv–269– iv–272 vol.4.

[4] F. Gouyon and P. Herrera, "Exploration of techniques for automatic labeling of audio drum tracks instruments," in *Proceedings of MOSART: Workshop on Current Directions in Computer Music*, 2001.

[5] J. Paulus and T. Virtanen, "Drum transcription with non-negative spectrogram factorisation," in *Proc. of 13th European Signal Processing Conference (EUSIPCO2005)*, 2005.

[6] K. Yoshii, M. Goto, and H. G. Okuno, "Drum Sound Recognition for Polyphonic Audio Signals by Adaptation and Matching of Spectrogram Templates With Harmonic Structure Suppression," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 1, pp. 333–345, 2007.

[7] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, 1st ed. Prentice Hall, Apr. 1993.

[8] A. Klapuri, "Sound onset detection by applying psychoacoustic knowledge," *ICASSP*, 1999.

[9] J. Bello *et al.*, "A tutorial on onset detection in music signals," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, p. 1035, 2005.

[10] E. Battenberg, "Improvements to Percussive Component Extraction Using Non-Negative Matrix Factorization and Support Vector Machines," Master's thesis, University of California, Berkeley, Dec. 2008.

[11] D. Reynolds, T. Quatieri, and R. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[12] D. Reynolds, "Approaches and applications of audio diarization," *Acoustics*, 2005.

[13] A. Banerjee, S. Merugu, and I. Dhillon, "Clustering with Bregman Divergences," *Journal of Machine Learning Research*, 2005.

[14] C. Févotte, N. Bertin, and J. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural Computation*, vol. 21, no. 3, pp. 793–830, 2009.

[15] J. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," International Computer Science Institute, Tech. Rep. TR-97-021, 1998.

[16] C. Bouman, "Cluster: An unsupervised algorithm for modeling gaussian mixtures," https://engineering.purdue.edu/~bouman/software/cluster/, Sept 1998.

[17] J. Rissanen, "A universal prior for integers and estimation by minimum description length," *The Annals of statistics*, 1983.

[18] R. Hennequin, B. David, and R. Badeau, "Beta-Divergence as a Subclass of Bregman Divergence," *Signal Processing Letters, IEEE*, vol. 18, no. 2, pp. 83–86, 2011.

[19] A. Cichocki, R. Zdunek, and S. Amaria, "Csiszar's divergences for non-negative matrix factorization: Family of new algorithms," in *International Conference on Independent Component Analysis and Blind Signal Separation*, 2006.

[20] N. Bertin, C. Févotte, and R. Badeau, "A tempering approach for Itakura-Saito non-negative matrix factorization. With application to music transcription," in *ICASSP 2009*, 2009, pp. 1545–1548.

[21] Roland. V-Drums. [Online]. http://rolandus.com

[22] Toontrack. Superior Drummer 2.0. [Online]. http://toontrack.com