Improvements to Percussive Component Extraction Using Non-Negative Matrix Factorization and Support Vector Machines

Eric Battenberg

January 29, 2009

Abstract

A system for the automatic extraction of percussive components from polyphonic digital audio is presented. Like some previous work, the system uses an iterative non-negative matrix factorization (NMF) algorithm to decompose a song's spectrogram into components, and then it classifies these components as percussive or non-percussive using a support vector machine (SVM). Our approach attempts to reduce computation time and improve separation results by incorporating a perceptual dimensionality reduction into the NMF step. In addition, we introduce new features – some based on note onset locations – extracted from the spectra and gain signals of each component in order to reduce classification errors. Our NMF approach greatly reduces computation time while retaining the same (or improving) the quality of separation. And using our new features, our component classifier achieves an equal error rate of less than 3.7% on a database of 32 songs.

Contents

List of Figures

List of Tables

Acknowledgements

Thanks to my adviser David Wessel for giving me the opportunity to pursue my music-related research interests and for providing me with creative research ideas. Thank you to my co-adviser Nelson Morgan for being willing to give me advice and manage my degree progress even though I do not work directly with him. Thanks to Markus Cremer at Gracenote for encouraging me to pursue my rhythm and drum-related interests and for his valuable brainstorming sessions. Lastly, thanks to my parents, Terry and Barbara Battenberg, for encouraging and supporting my interests, especially my love for music.

Chapter 1

Introduction

Music Information Retrieval (MIR) is a field concerned with the extraction of supplemental information about a musical audio signal. Typical applications within MIR include music similarity, automatic play-list generation, mood detection, score transcription, melody extraction, audio fingerprinting, and more [1][2][3]. Some researchers use the more general term "machine listening" to refer to techniques that use computers to draw human-like conclusions about audio signals in general.

MIR techniques are becoming increasingly important as it is not uncommon for a typical consumer to have over 10,000 songs in his or her personal digital media collection. In addition, digital music stores are amassing extremely large libraries of audio. With such large collections, it becomes very difficult to navigate and search for a particular type of song using only the provided textual descriptions such as artist, title, and genre.

Some online services such as Pandora and Last.fm have attempted to organize and recommend content using human-produced editorial information or collaborative filtering based on users with similar tastes. Editorial information works for certain tasks but fails to limit the tedious human involvement involved in hand-labeling songs. Recommendation through collaborative filtering is prone to only suggest more popular songs and is susceptible to certain types of misinformation attacks designed to unfairly influence the popularity of a song. In addition, these services do not allow a user to navigate and search the songs in his or her own collections in a similar manner.

MIR techniques will address many of the problems faced by music listeners in today's digital world. Musicians and composers also will benefit due to improvements in recommendation and promotion of content as well as more relevant querying of music archives. Basic MIR technology is already employed by audio players and online services around the world, and soon more advanced techniques will become an indispensable part of the digital music experience.

1.1 Previous Work

In this project, we focus on rhythm. There has been a considerable amount of work done on timbre and spectral models of music [4][5]. These systems work well for automatic genre classification, especially when discriminating between disparate genres such as heavy metal and jazz. When more detailed information about a song, such as mood, feel or groove , is desired, rhythm and other temporal considerations become much more important. Due to these shortcomings of timbre-only analysis, there has been a recent increase in rhythm-related research.

An area that is integral to MIR rhythm research is rhythm summarization and comparison. An important contribution to this area is the work of Foote [6], in which a self-similarity measure is used to compute a representation of the strength of different rhythmic intervals. Other work has used this representation to analyze rhythmic patterns, make rhythmic comparisons, and segment songs into passages [7][8]. Similarly, Peeters [9] uses "spectral rhythm patterns" to classify rhythms.

Automatic drum transcription and extraction has also seen attention. A system presented by Yoshii et al. [10] uses adaptive spectral templates to recognize the occurrence of drum sounds. The work of Uhle [11], which inspired this project, used independent subspace analysis as a source separation technique to perform drum track extraction. Helen and Virtanen [12] use non-negative matrix factorization and support vector machines in their attempt to separate the drums from harmonic instruments. In this project, we focus on this type of percussive source separation.

1.2 Target Applications

The extraction of drum and percussion is a simple source separation task compared to the extraction of pitched instruments that have spectra that vary with pitch. For this reason, there is hope that drum extraction systems can be made to perform well enough to be used by the public in the near future. In addition, a drum track extractor used a preprocessing step for other rhythm tasks, such as drum transcription or rhythm summarization, has the potential to improve the performance of these systems greatly.

In this project, we aim to improve both the performance and speed of existing drum extraction systems. Our work is similar to that in [12] in that we use non-negative matrix factorization (NMF) along with a support vector machine (SVM) in order to separate percussive components from harmonic components. This project improves on previous work by introducing a dimension-reduced version of NMF that performs just as well but runs significantly faster. We also suggest a deterministic initialization of the iterative NMF algorithm that eliminates the inherent randomness of traditional initialization. A new feature set is tested with the SVM classifier, and it yields good results. Some of these features are computed using information from an onset detector, and these features turn out to be very important. Lastly, instead of using artificially produced audio mixes to test our system, we evaluate our results using actual audio data which is either

hand-labeled or comes from an audio archive aimed at source separation.

1.3 Paper Overview

Chapter 2 is an overview of techniques important to audio source separation with regard to rhythm. These techniques include independent component analysis (ICA), independent subspace analysis (ISA), and non-negative matrix factorization. Chapter 3 covers support vector machines and the practical considerations regarding their use as classifiers, such as cross-validation, training, and parameter selection. In Chapter 4, our method of onset detection is covered. Then the specifics of the entire percussion extraction system are detailed in Chapter 5. Finally, our results and analysis are contained in Chapter 6.

Chapter 2

Audio Source Separation

Source separation involves problems in which multiple signals have been combined in some way, and we wish to recover some or all of the original signals from their combination. In the analysis of audio signals, this process is especially useful since the vast majority of real-world audio signals are formed by the combination of several sources. Speech processing is a field in which robust source separation would be incredibly useful. By separating out background noise and other speakers from a target speaker, the accuracy of automatic speech recognition techniques could be vastly improved. However, the ability of audio source separation to perform well under a variety of conditions has been quite lacking, and automatic speech recognition remains a difficult problem.

Music processing is another field in which source separation shows promise. Since musical signals result from the combination of parts produced by separate instruments, the analysis of a passage of music could be greatly simplified if the contribution of each instrument could be processed individually.

Let $\mathbf{x}(t) \in \mathbb{R}^n$ be the observed signal vector and $\mathbf{s}(t) \in \mathbb{R}^k$ the source signal vector, i.e.

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix} \qquad \mathbf{s}(t) = \begin{bmatrix} s_1(t) \\ \vdots \\ s_k(t) \end{bmatrix},$$

$\mathbf{x}(t)$ contains the signals observed at n sensors. For audio, these n sensors are typically an array of microphones distributed in space. $\mathbf{s}(t)$ represents the k source signals we are trying to discover. When $\mathbf{x}(t)$ is assumed to be a linear combination of the elements of $\mathbf{s}(t)$, we can relate $\mathbf{x}(t)$ to $\mathbf{s}(t)$ with the mixing matrix $\mathbf{A} \in \mathbb{R}^{n \times k}$.

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t)$$

2.1 Independent Component Analysis

Independent component analysis (ICA) [13] is a family of source separation techniques that attempt to find an "unmixing" matrix, $\mathbf{B} \in \mathbb{R}^{k \times n}$ that transforms the observed signal vector back into the original source signal vector.

$$\mathbf{s}(t) = \mathbf{B}\mathbf{x}(t)$$

In unsupervised, or blind ICA, typical algorithms use a result of the central limit theorem, which states that the sum of an increasing number of independent random variables tends toward a Gaussian distribution. Therefore, the sum of a finite number of independent variables should be closer to a Gaussian distribution than the individual random variables (as long as they are not Gaussian themselves). Some algorithms try to estimate the unmixing matrix such that it yields source components that are maximally "non-Gaussian", and therefore independent. FastICA [13] is a popular algorithm that does this. Other algorithms attempt to achieve independence amongst source components using information theoretic measures such as mutual information or negentropy.

In order for the original source components in the above model to be recovered, we must have at least as many observations as sources, i.e. $n \geq k$. If this condition isn't satisfied, the system is underdetermined, and there are infinitely many solutions, $\mathbf{s}(t)$. If $n = k$ and the source components are independent, ICA performs very well.

In music information retrieval, we are frequently tasked with analyzing monaural or stereo audio signals. These one or two channel observation signals are definitely insufficient for ICA to separate out an ensemble of instruments. So how do we compensate for this?

2.2 Independent Subspace Analysis

Independent Subspace Analysis (ISA) [14] attempts to reconcile the inability of a single channel audio signal to be of any use in ICA by estimating frequency domain subspaces that each independent source contributes to the observation signal. If we take the short-time Fourier transform (STFT) of the single channel audio signal, we get the matrix $\mathbf{X} \in \mathbb{C}^{N \times F}$, where each column holds the complex-valued FFT of each analysis frame. N is the length of the analysis window used to construct each frame, and F is the number of frames.

The FFT is linear in that the FFT of a sum of signals is equal to the sum of the FFTs of each signal. Therefore, we *could* model $\mathbf{X}$ as:

$$\mathbf{X} = \mathbf{A}\mathbf{S} \qquad \mathbf{S} \in \mathbb{R}^{K \times F},\ \mathbf{A} \in \mathbb{C}^{N \times K}$$

where $\mathbf{S}$ is a matrix which contains the source components for each frame in its columns, and $\mathbf{A}$ is the mixing matrix with columns that describe the frequency subspace of each of the K components.

This model is mathematically sound; however, in practice, this complex spectrogram has an irregular, non-stationary phase due to the shifting time window interacting with the original widely-varying phase of the frequency components, and a complex-valued mixing matrix, $\mathbf{A}$, becomes impossible to estimate. To fix this, we can replace the complex spectrogram, $\mathbf{X}$, with its magnitude, $|\mathbf{X}|$, or squared-magnitude spectrum, $|\mathbf{X}|^2$. The squared-magnitude spectrum sums linearly in its expectation as long as the phase of the sources are independent. We cannot make this claim for the non-squared magnitude spectrum, though it also serves as a useful approximation and actually performs better in our application.

For real-valued signals, the magnitude spectrum is symmetric, so we can get rid of the redundant components corresponding to negative frequencies, and N becomes equal to half the analysis window length. This leaves us with our final ISA model, where $|\mathbf{X}|$ is the magnitude spectrogram (squared or not), $\mathbf{A}$ is the real-valued matrix containing the spectral contributions of each source component in its columns, and $\mathbf{S}$ contains the frame-wise gain values of each source component in its rows. In this model, K is the number of source components, F is the number of frames, and N is the number of frequency components.

$$|\mathbf{X}| = \mathbf{AS} \qquad \text{where } |\mathbf{X}| \in \mathbb{R}^{N \times F},\ \mathbf{A} \in \mathbb{R}^{N \times K},\ \mathbf{S} \in \mathbb{R}^{K \times F} \tag{2.1}$$

2.3 Non-negative Matrix Factorization

Using the magnitude spectrogram ISA model along with an ICA algorithm, we can compute K source components and each of their spectral contributions. The problem with this, however, is that normal ICA algorithms do not restrict the source components or their contributions to each observation to be non-negative. A negative contribution to a magnitude spectrogram only makes sense when we have some sort of significant destructive interference between instruments, and allowing the sources to go negative doesn't make sense physically.

Non-negative independent component analysis [15] solves one of these problems by forcing the sources, $\mathbf{S}$ to be non-negative; however the source contributions, $\mathbf{A}$, can become negative using this method.

Non-negative matrix factorization (NMF) [16] solves both of the negativity problems by attempting to find non-negative $\mathbf{A}$ and $\mathbf{S}$. Basic NMF techniques do not take into account the independence of the sources or any other statistical considerations as does ICA. Though in musical analysis, independence between musical instruments is an invalid assumption anyway.

Given the non-negative matrix $\mathbf{Y}$ and the inner dimension, K, of its factorization, $\mathbf{AS}$, NMF tries to find $\mathbf{A}$ and $\mathbf{S}$ such that they are non-negative and yield the minimum reconstructive error with respect to $\mathbf{Y}$. Different cost functions can be used to emphasize different types of errors in the reconstruction.

Two common cost functions used in NMF are the square of the Euclidean distance

$$\|\mathbf{Y} - \mathbf{AS}\|^2 = \sum_{ij} \left(\mathbf{Y}_{ij} - (\mathbf{AS})_{ij}\right)^2$$

and a matrix version of the Kullback-Leibler divergence n

$$D(\mathbf{Y}\|\mathbf{AS}) = \sum_{ij} \left(\mathbf{Y}_{ij} \log \frac{\mathbf{Y}_{ij}}{(\mathbf{AS})_{ij}} - \mathbf{Y}_{ij} + (\mathbf{AS})_{ij}\right) \tag{2.2}$$

The above divergence cost function has been shown to perform better for separating percussive and subtle background components than the Euclidean cost function [17], which is why it is used in this project.

Lee and Seung [16] introduce multiplicative update rules for minimizing the above cost functions with respect to $\mathbf{A}$ and $\mathbf{S}$ with $\mathbf{A}, \mathbf{S} \geq 0$. The functions are not convex in both $\mathbf{A}$ and $\mathbf{S}$, so the updates converge to local (not global) minima. Lee and Seung point out that the multiplicative rules are faster to converge than gradient-based updates and easier to implement. For these reasons, they have been used frequently in audio source separation tasks.

The divergence cost function is non-increasing under the following update rules

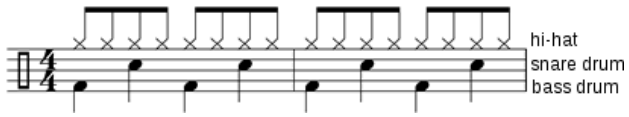
$$\mathbf{S}_{\alpha\mu} \leftarrow \mathbf{S}_{\alpha\mu} \frac{\sum_i \mathbf{A}_{i\alpha} \mathbf{Y}_{i\mu} / (\mathbf{AS})_{i\mu}}{\sum_k \mathbf{A}_{k\alpha}} \qquad \mathbf{A}_{i\alpha} \leftarrow \mathbf{A}_{i\alpha} \frac{\sum_\mu \mathbf{S}_{\alpha\mu} \mathbf{Y}_{i\mu} / (\mathbf{AS})_{i\mu}}{\sum_\nu \mathbf{S}_{\alpha\nu}} \tag{2.3}$$

Written in Matlab syntax, these updates are

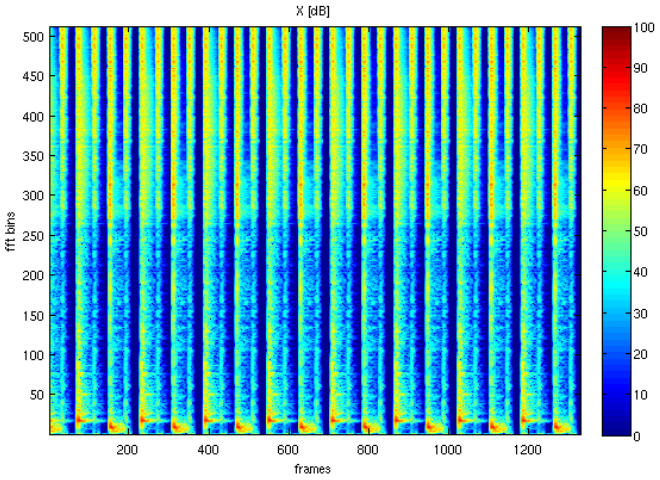
```
Z = Y./(A*S+eps);
S = S.*(A'*Z)./(repmat(sum(A)',1,F));
Z = Y./(A*S+eps);
A = A.*(Z*S')./(repmat(sum(S,2)',N,1));

% where
% [N,F] = size(Y);
% and 'eps' is a small constant added to the denominator of Z to avoid dividing by zero
```

To carry out NMF according to our ISA model, we set $\mathbf{Y} = |\mathbf{X}|$, so that we will be factoring the magnitude spectrogram of the input signal. A simple example of NMF performed on a 3 instrument drum track is shown in Figure 2.1. The three instruments are hi-hat, snare drum, and bass drum. Because of the lack of background interference and the fact that the spectra of each instrument are fairly stationary, the above algorithm performs very well, when $\mathbf{A}$ and $\mathbf{S}$ are initialized with rectified white noise.

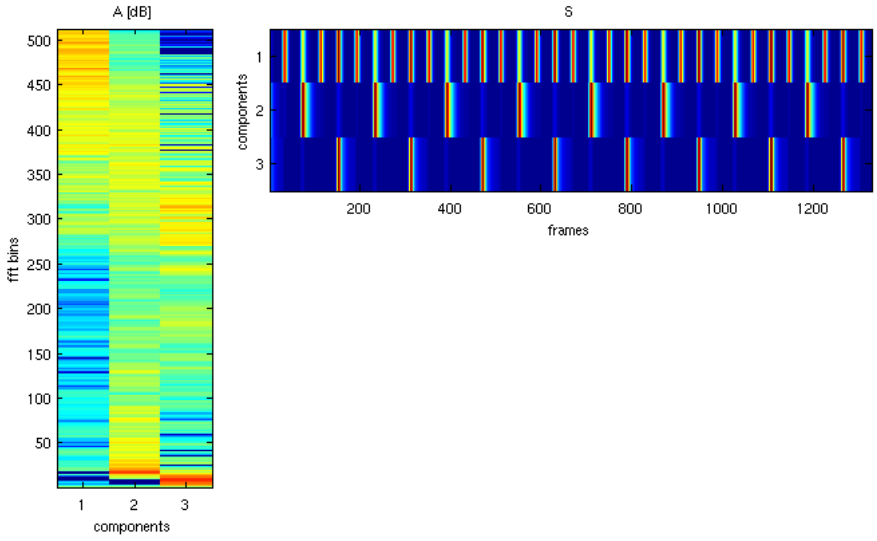
We can see that the rows of $\mathbf{S}$ in Figure 2.1b correspond very well with the occurrences of individual instruments in the drum score in Figure 2.1a. Having access to an instrument-wise decomposition such as this is very useful for all kinds of musical or rhythmic analysis as well as automatic transcriptions and content summarization.



a.



b.



c.

Figure 2.1: Example NMF: a) input drum track, b) spectrogram of input audio, c) resulting non-negative matrix factorization for $K = 3$ sources

2.4 Problems with NMF

The primary problems concerning the use of NMF for audio source separation include its significant computational complexity, the non-stationary spectra of many musical instruments, and the uncertainty caused by non-deterministic initialization when using iterative algorithms.

Computational complexity can be reduced either by developing faster algorithms or reducing the size of the problem. In this project, the approach we take is problem size reduction through perceptual dimensionality reduction. In addition to a shorter running time, this method actually yields somewhat better decomposition quality. The details of this approach are covered in Chapter 5.

Because melodic instruments have much more dynamic spectra than percussive instruments, they usually cannot be described by a single spectral contribution scaled by a time-varying gain. In order to limit the number of components in the decomposition used by a single melodic instrument, we can use the assumption that instrument-wise spectra are approximately stationary over a small time interval [14]. This way we will be decomposing multiple short spectrograms that make up an entire audio track, and addressing the non-stationarity issue.

The most frequent initialization procedure used with these iterative NMF algorithms is to set $\mathbf{A}$ and $\mathbf{S}$ to rectified white noise. While this method can work well on average, a deterministic initialization is desired in order to make the NMF decomposition repeatable. Our method is covered in Chapter 5.

Chapter 3

Classification with Support Vector Machines

Support Vector Machines (SVMs) are a group of supervised learning techniques used in classification and regression. We employ them in this project in order to make automatic decisions about whether a particular component of the spectrogram NMF is percussion or not. The classification performance we achieve using SVMs is more than satisfactory.

3.1 Maximum Margin Classification

We start with a set of training data

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_{i=1}^n \tag{3.1}$$

where the vectors $\mathbf{x}_i$ are p-dimensional feature vectors, and the boolean-valued y_i are corresponding class labels. If the classes are linearly separable, we can find $\mathbf{w} \in \mathbb{R}^p$ and scalar b such that the hyperplane, $\mathbf{w}^{\mathrm{T}}\mathbf{x} - b = 0$, separates all the points $\mathbf{x}_i$ with label $y_i = 1$ from those with label $y_i = -1$. Once we have calculated such a hyperplane, we can use it to classify new data points, $\mathbf{x}$, into classes according to the sign of $\mathbf{w}^{\mathrm{T}}\mathbf{x} - b$. In general, if the data is separable, we can find many such separating hyperplanes. In order to choose a specific hyperplane, we can choose the one that maximizes the margin between the two classes. With the original separating hyperplane at the center of the margin, the parallel hyperplanes that represent the boundaries of the margin are

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} - b = \pm 1 \tag{3.2}$$

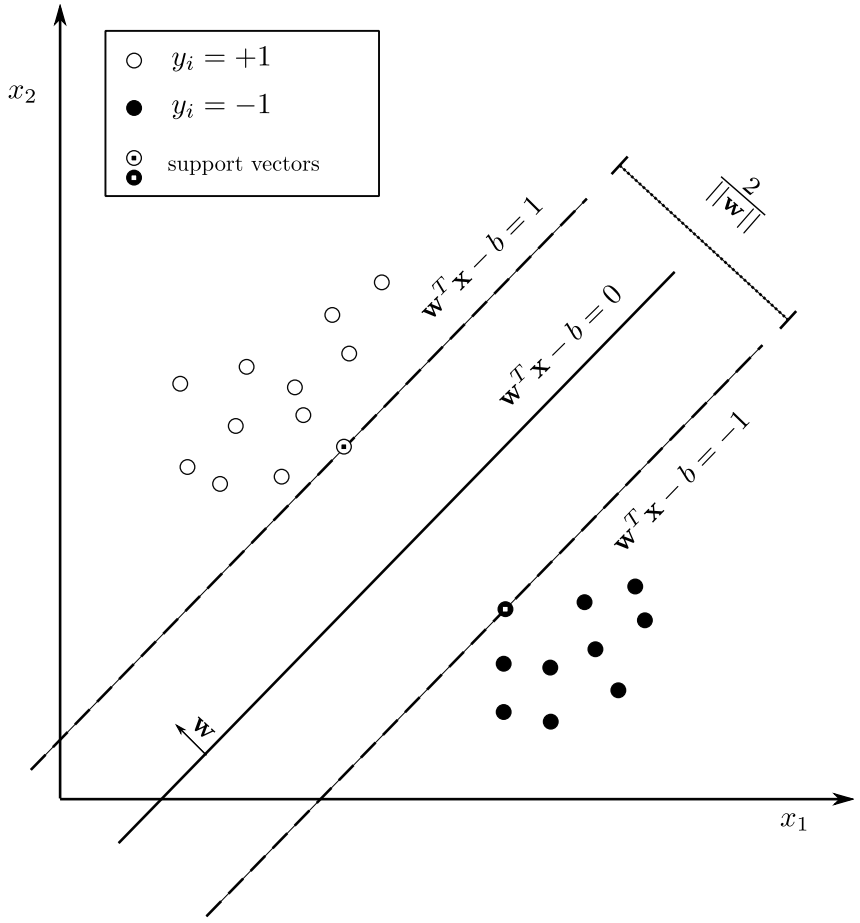


Figure 3.1: Maximum margin separating hyperplane

With the data from the two classes constrained to the half-spaces

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \geq +1, \quad \text{for } y_i = +1 \tag{3.3}$$

$$\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b \leq -1, \quad \text{for } y_i = -1 \tag{3.4}$$

An illustration of a maximum margin hyperplane is shown in Figure 3.1. We can see that all of the data points lie outside of the margin and the two classes are separated by it. The width of the margin is $\frac{2}{||\mathbf{w}||}$, the value we wish to maximize. Equivalently, we can minimize $||\mathbf{w}||$ or $\frac{1}{2}\,||\mathbf{w}||^2$, subject to the constraints in (3.3). Our optimization problem can be written

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{minimize}} && \tfrac{1}{2}\,||\mathbf{w}||^2 \\ &\text{subject to} && y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b) \geq 1 \quad \forall i \in \{1, \ldots, n\} \end{aligned} \tag{3.5}$$

The optimization problem in (3.5) above is a convex quadratic program (QP), i.e. it has a quadratic objective and linear constraints, and can be solved using standard QP optimization algorithms. We can reformulate the problem as its important equivalent dual form using a Lagrangian [18].

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & \textstyle\sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j \\ \text{subject to} \quad & \lambda_i \geq 0, \quad \forall i \in \{1, \ldots, n\} \\ & \textstyle\sum_{i=1}^{n} \lambda_i y_i = 0 \end{aligned} \tag{3.6}$$

Solving the dual problem in (3.6) yields the same solution as the original problem with

$$\mathbf{w} = \sum_i \lambda_i y_i \mathbf{x}_i \tag{3.7}$$

$$b = y_K - \mathbf{w}^{\mathrm{T}} \mathbf{x}_K, \quad \text{where } K = \text{any } k \text{ such that } \lambda_k \neq 0 \tag{3.8}$$

We see that the hyperplane now only depends on the pairs $(\mathbf{x}_i, y_i)$ with $\lambda_i \neq 0$. The $\mathbf{x}_i$ for which $\lambda_i \neq 0$ are called *support vectors* because they actually lie on the borders of the margin. An important observation is that we can remove any of the data points that are not support vectors and still end up with the same maximum margin hyperplane.

This formulation of a support vector machine works well when the data is separable; however, in real world applications, this is rarely the case. The problem of non-separable data is handled by using *soft margins*.

3.2 Soft Margins

The use of soft margins in support vector machines was introduced by Cortes and Vapnik in [19]. What the soft margin mechanism does is to allow data points to cross into the margin or even to the other side of it. By introducing non-negative noise variables, ξ_i, that are proportional to the distance a data point, $\mathbf{x}_i$, is from its correct side of the margin, we can reformulate the problem in 3.5 as

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \tfrac{1}{2} ||\mathbf{w}||^2 + C \textstyle\sum_i \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^{\mathrm{T}} \mathbf{x} + b) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \ \forall i \end{aligned} \tag{3.9}$$

Where C is a parameter which controls the softness of the margin. With $C = +\infty$, the ξ_i are forced to zero, and the problem is equivalent to the original hard margin problem. An example soft margin and its associated noise terms are illustrated in Figure 3.2.

This problem can also be reformulated as its equivalent dual problem in order to express it in terms of the support vectors. Below we see that the only change between the soft margin dual problem and the hard margin dual problem in 3.6 is the addition of the upper bound C to the variables ξ_i.

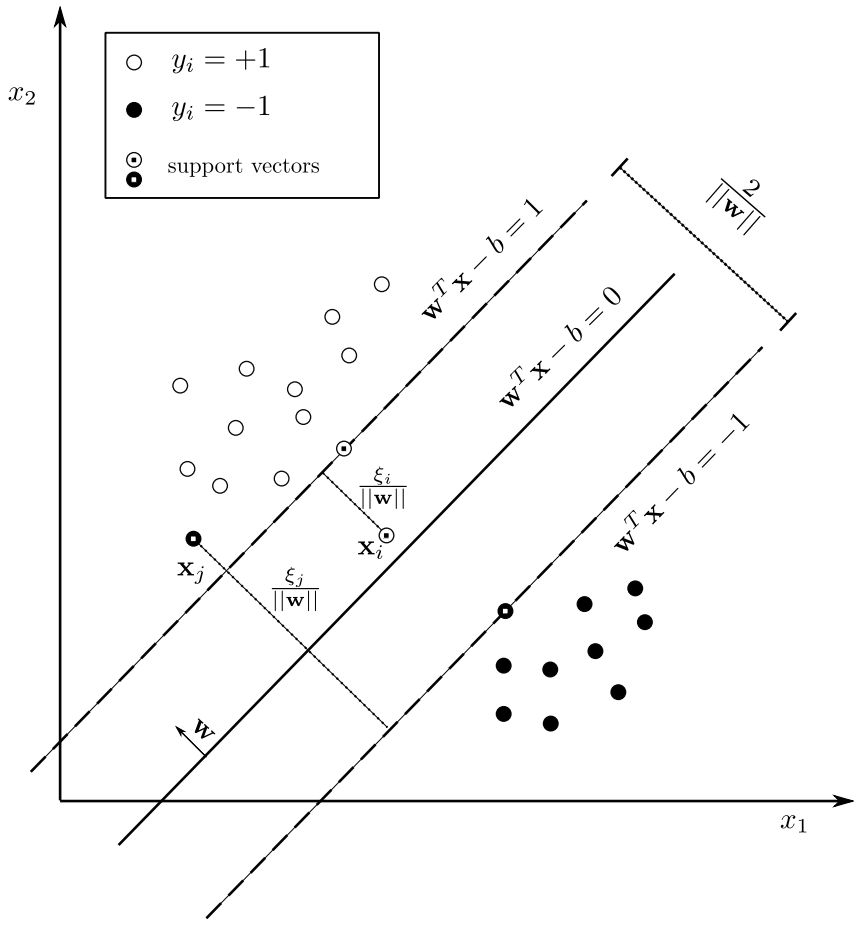


Figure 3.2: Soft margin separating hyperplane

$$\begin{aligned} \underset{\lambda}{\text{maximize}} \quad & \textstyle\sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j \\ \text{subject to} \quad & 0 \leq \lambda_i \leq C, \quad \forall i \in \{1, \ldots, n\} \\ & \textstyle\sum_{i=1}^{n} \lambda_i y_i = 0 \end{aligned} \tag{3.10}$$

After solving the above dual soft margin problem using a standard QP routine, we compute $\mathbf{w}$ as we did in (3.7) as a linear combination of the support vectors. We compute b using any $\mathbf{x}_i$ that lies on the border of the margin, which turns out to be those for which $0 < \lambda_i < C$ (a strict inequality) holds. Those for which $\lambda_i = C$ are the x_i which cross into or past the margin, i.e. $\xi_i > 0$. They are included in the calculation of $\mathbf{w}$ but cannot be used to compute b since they do not lie on the margin borders.

This soft margin formulation allows the SVM to be more robust in cases of misclassified training data or noisy feature measurements. However, if the data lie in some non-linear manifold, the linear SVM we have used so far would be ineffective. In the next section, the dual formulation of the SVM problem is employed along with the "kernel trick" to create non-linear SVM classifiers.

3.3 Non-linear Kernels in Support Vector Machines

In 3.10 we see the only dependence the dual problem has on each data point, $\mathbf{x}_i$, is through pair-wise dot products. Likewise, evaluating the sign of $\mathbf{w}^{\mathrm{T}}\mathbf{x} - b$ during classification can be done using a linear combination of pair-wise dot products (due to the definition in 3.7).

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} - b = \sum_i \lambda_i y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x} - b \tag{3.11}$$

In order to deal with data that is not linearly separable, we can map the data onto a set of non-linear basis functions. In Figure 3.3, we see two classes that can be separated by a parabola. However, because the SVM formulation is in the form of a linear projection, it doesn't allow for this type of division. Remapping the data from Figure 3.3 so that

$$phi(\mathbf{x}) : \mathbf{x} \rightarrow \mathbf{z} = \begin{bmatrix} x_1^2 \\ x_2 \end{bmatrix} \tag{3.12}$$

results in the data layout in Figure 3.4 which is easily separable by a line.

Using this "kernel trick", we can easily implement many types of non-linear classifiers simply by using a non-linear map on the data before running the SVM. The important pair-wise dot products $\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$ become

$$\mathrm{K}(\mathbf{x}_i, \mathbf{x}_j) \triangleq \phi(\mathbf{x}_i)^{\mathrm{T}} \phi(\mathbf{x}_j)$$

where $\mathrm{K}(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel function chosen with the data layout in mind.

Typical non-linear kernels are shown in Table 3.1. The transformation in (3.12) is a special case of the inhomogeneous polynomial kernel with $d = 2$. The radial basis function kernel is very diverse in that it can

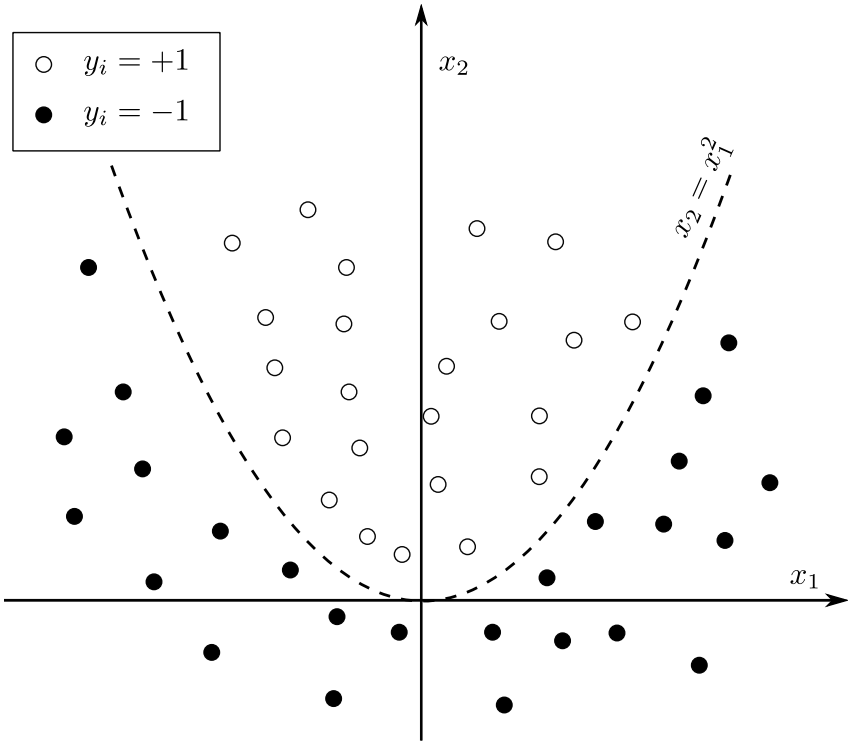


Figure 3.3: Data with a non-linear parabolic separation

be effectively used with a wide range of data topologies. For this reason, it is recommended as a first resort when dealing with large feature vectors. We use this type of kernel function in this project, and it performs very well.

Type	$\mathrm{K}(\mathbf{x}_i, \mathbf{x}_j)$
Polynomial (homogeneous)	$(\mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j)^d$
Polynomial (inhomogeneous)	$(\mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j + 1)^d$
Gaussian Radial Basis Function	$\exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$

Table 3.1: Typical non-linear kernels

3.4 Cross-validation and Testing

Deciding to use an SVM along with a particular kernel doesn't completely specify our classifier. For a soft margin SVM, we have to choose the softness parameter C, and our non-liner radial basis kernel has the gaussian standard deviation parameter σ to choose. Unless we have some sort of intimate knowledge of our data and already know which parameters would work best, we have to rely on just testing different

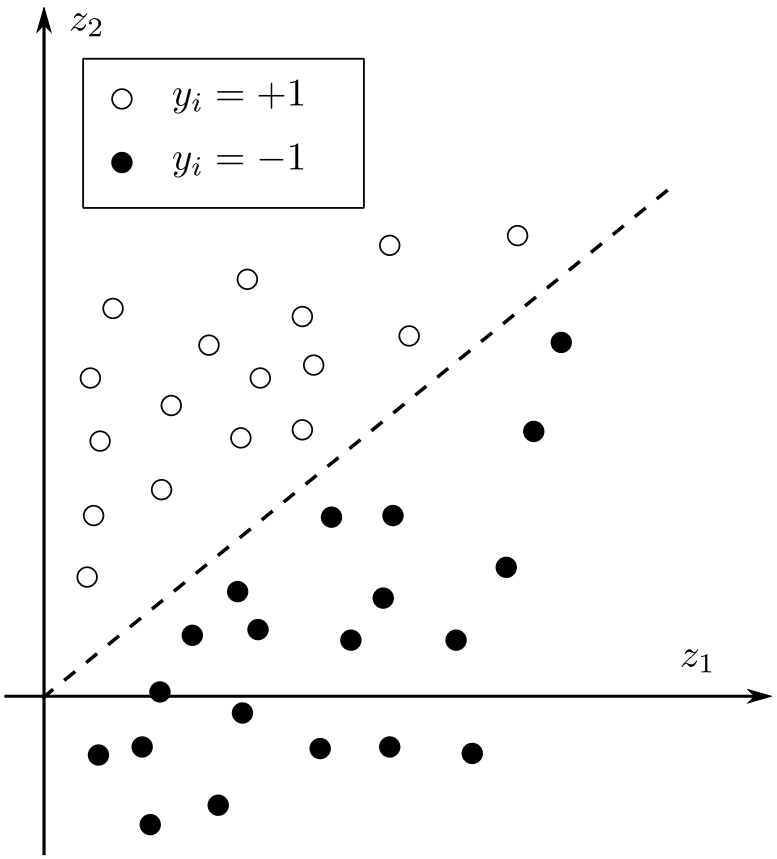


Figure 3.4: Data similar to that in Figure 3.3 remapped using a non-linear transformation

parameters to evaluate which works best. A popular and effective method of testing the performance of any supervised learning method is called cross-validation.

In K-fold cross-validation, we start with a set of labeled data, $\mathcal{D}$ as in (3.1, and partition it into K equal-sized groups. For each group $k \in \{1, \ldots, K\}$, we train the SVM using the data in all groups besides k, and then test the performance of the classifier using the labeled data in group k. We then average the number of classification errors that occurred when testing with each of the K groups in order to arrive at our final performance estimate.

Cross-validation is an important testing technique that estimates the performance of a learning method when tested using data it has not yet seen. It prevents over-fitting of the model to the test data and better simulates real-world conditions. In order to use cross-validation when deciding on parameters, we can do an exhaustive "grid search" that tests all feasible pairs of (C, σ) and determines the pair that gives the best performance. Cross-validation is also used to test the final performance of the classifier.

Chapter 4

Onset Detection

The detection of onsets in audio signals refers to the process of locating the beginning of musical notes, percussive attacks, or other sounds. Onset detection is still an active research area in music information retrieval, but certain existing techniques are very helpful when extracting information about percussive sounds.

In this project, our method for onset detection is based on the system proposed by Klapuri in [20] which is partly based on work by Scheirer [21]. The main premise of these approaches is that important perceptual onset information is contained in the first derivative of band-wise power signals. The specifics of this approach along with other intermediate steps such as smoothing and the final onset decisions are covered in the following sections.

4.1 Band Decomposition

Performing a band decomposition on the signal to undergo onset detection is a necessary first step. By breaking the signal up into its frequency components we can more easily detect onsets within each band. We are also able to pick out when an instrument undergoes a significant change in pitch, since its energy will be moved to a different band.

The spectral decomposition that we use to compute band-wise power signals is basically a perceptually-based spectrogram. Since we already compute the magnitude spectrogram, $|\mathbf{X}|$, in our independent subspace analysis model 2.1 used in source separation, we will start from there.

In order to transform the spectrogram $|\mathbf{X}|$ into a more perceptually meaningful form, we can group adjacent frequency components according to an approximately uniform partitioning of the Bark scale, which was designed to correspond to the critical bands of human hearing . The Bark scale frequency warping is shown below.

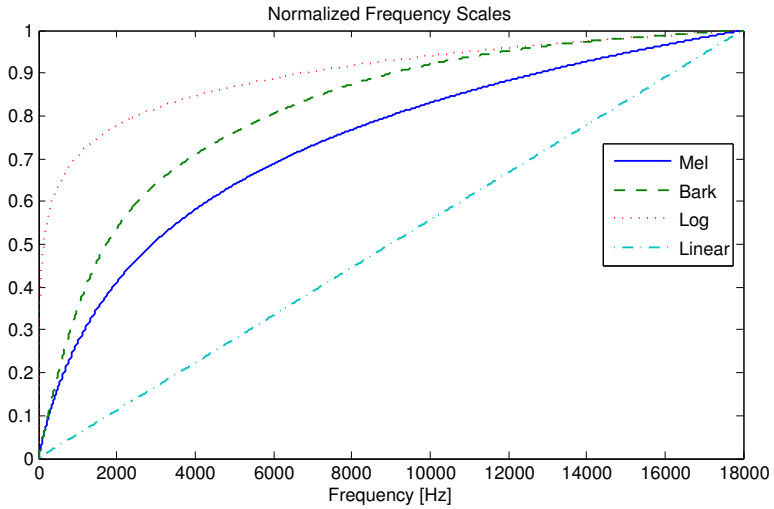


Figure 4.1: Comparison of different frequency scales

$$\text{Bark}(f) = 13\arctan(0.00076f) + 3.5\arctan\left((f/7500)^2\right) \tag{4.1}$$

Alternatively, we can use the mel scale, which was designed with the perceptual similarity of pitches in mind. In this project we use the mel scale due to its familiarity and frequent use in MIR applications.

$$\text{mel}(f) = 1127.01048\log(1 + f/700) \tag{4.2}$$

Using half-overlapping triangular-shaped filters with widths proportional to one of the perceptual scales above, we can form the matrix $\mathbf{F}$ with rows containing the filters which sum adjacent frequency bands. A plot of the rows of such a matrix with 40 mel-spaced filters applied to a 512 point FFT of a signal with sample rate 44.1kHz is shown in Figure 4.2.

To apply the band grouping transformation, we simply multiply by $\mathbf{F}$ to get the modified spectrogram $|\mathbf{X}|_{\text{BG}} = \mathbf{F}|\mathbf{X}|$. This transformation performs both dimensionality reduction and perceptual warping on the spectrogram. A transformation using a 128-band mel-scale grouping is shown in Figure 4.3.

4.2 Energy Differentiation

After performing the perceptual band decomposition on the input signal, we now compute the half-wave rectified energy differential in each band. This shows us the extent to which the energy in a band undergoes an increase in level. Before we can apply the differentiation, however, we must do a bit of preprocessing to emphasize perceptually-meaningful dynamics changes and modulation frequencies.

First we process each band signal using μ-law compression in order to mimic the log-like amplitude

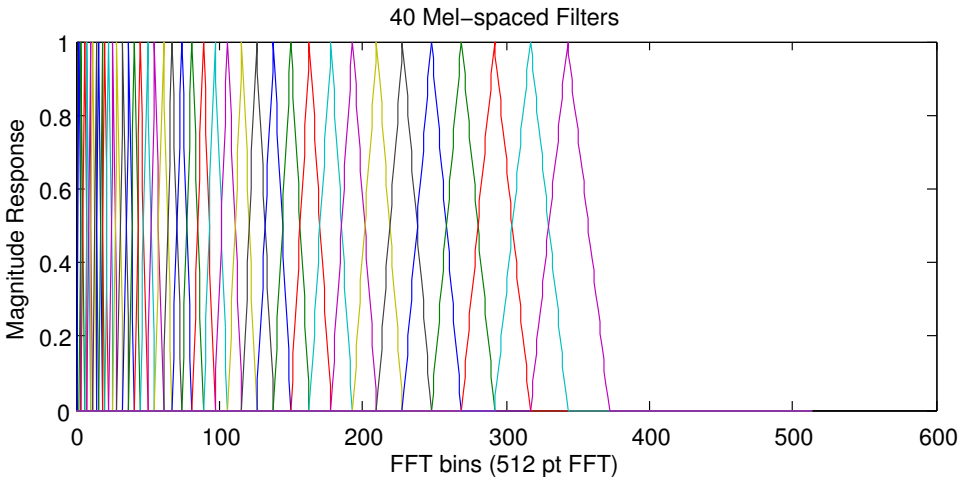


Figure 4.2: Rows of a mel-spaced filter matrix

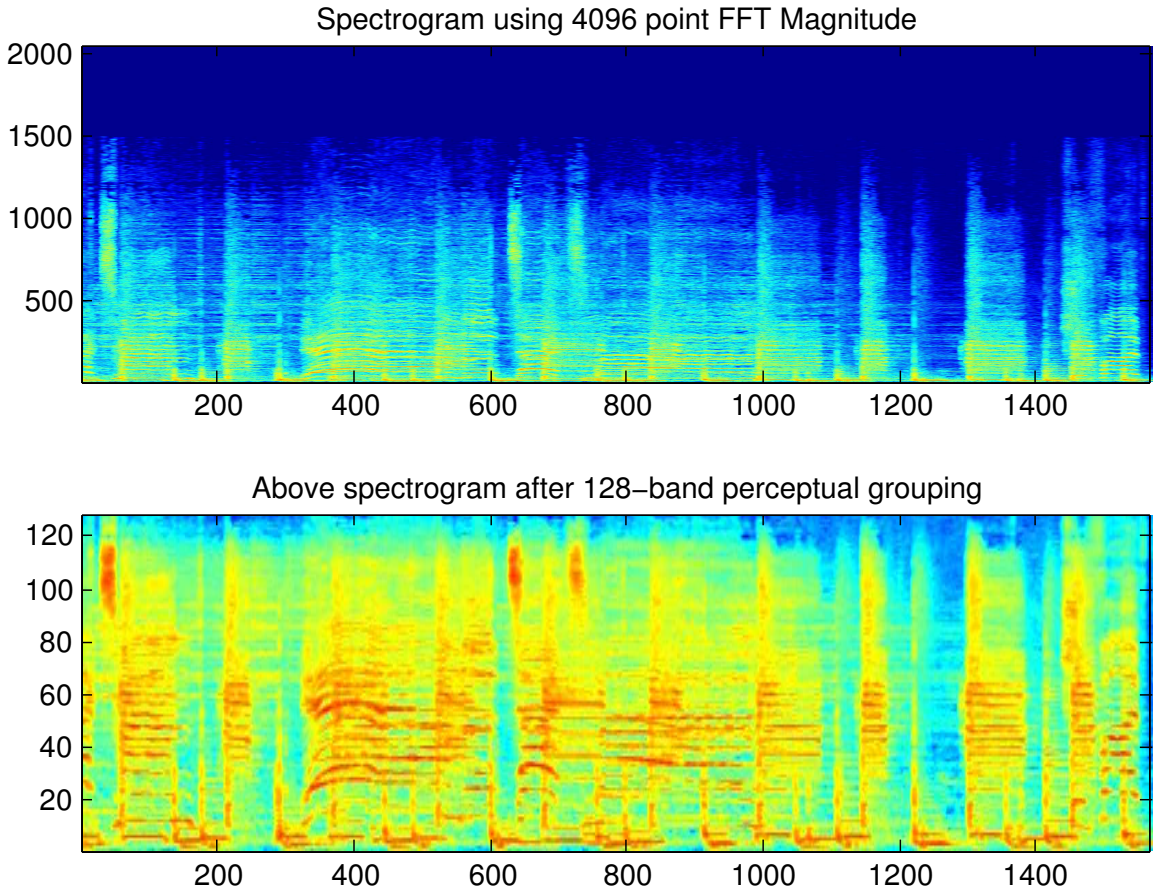


Figure 4.3: 128-band mel-scale grouping performed on a spectrogram with FFT size 4096.

sensitivity of the human ear without allowing the signal to go to $-\infty$ at low levels. The μ-law function is

$$F_\mu(x) = \frac{\log(1+\mu x)}{\log(1+\mu)} \tag{4.3}$$

The parameter μ sets the degree of compression of the function with smaller μ causing it to behave more linearly and larger μ making it more log-like. We use a larger value of $\mu = 10^8$ to achieve a near log curve.

Next we low pass filter each band signal in order to prevent noisy, high frequency modulations from dominating the differentiation. Perceptually-meaningful modulations are present below about 10Hz, so we filter each band using a squared Hann window with a cutoff of 10Hz in order to suppress the noisy modulations. As a filter, this window has linear phase which gives it a frequency independent group delay.

After the smoothing filter is applied, the difference between adjacent frames is computed to approximate the first derivative. Then we zero all of the negative values so that we're detecting only positive energy slopes. This final step leaves us with our band-wise accent signals. A before and after example showing the result of the compression, smoothing, and half-wave rectified differentiation is shown in Figure 4.4.

Once these band-wise accent signals are computed, we combine them into a single accent signal by summing all of the bands together.

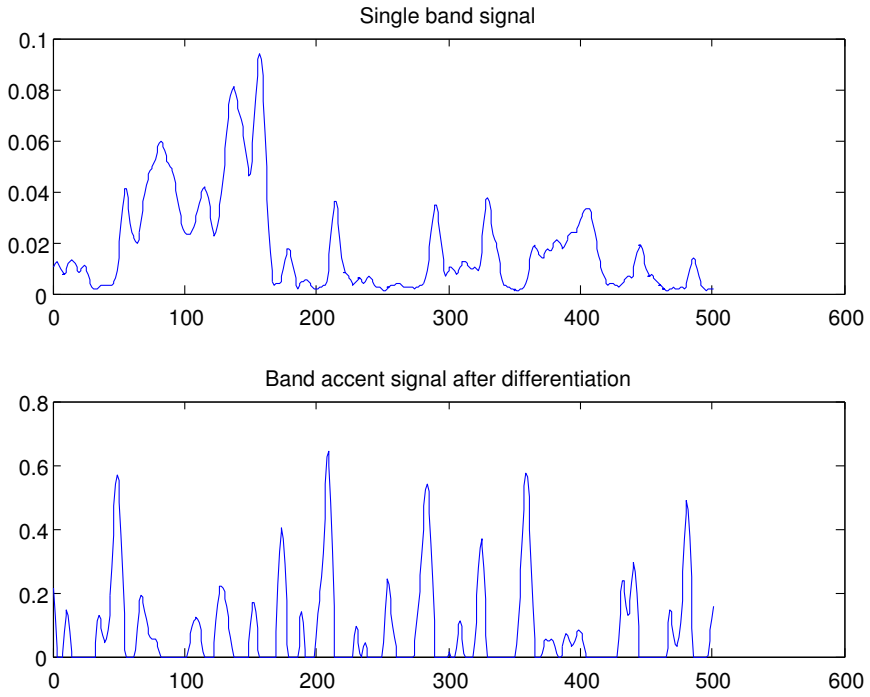


Figure 4.4: Band-wise signal shown before and after compression, smoothing, and differentiation.

4.3 Peak Detection

The final step in onset detection is to pick out the significant peaks in the accent signal and label them as onsets. We do this using a series of steps that narrow down the prospective peaks. After the accent signal is scaled so that one is the second largest peak, we perform the following:

1. *Local maxima detection*

   First peaks are detected by choosing points which are greater than or equal two both neighboring points.

2. *Thresholding*

   We keep only the peaks which exceed a minimum threshold of 0.2.

3. *Minimum Interval*

   Only peaks which are the largest within a certain time interval are kept. We use an interval of 68ms.

4. *Relative Rise*

   Valid peaks must rise a certain amount above the trailing local minimum. The amount we use is 0.2.

Once we have discovered all of the valid peaks in the accent signal, we adjust the peak locations for the delay of the squared Hann filter from Section 4.2. This leaves us with our final onset locations. The entire onset detection process is outlined in Figure 4.5

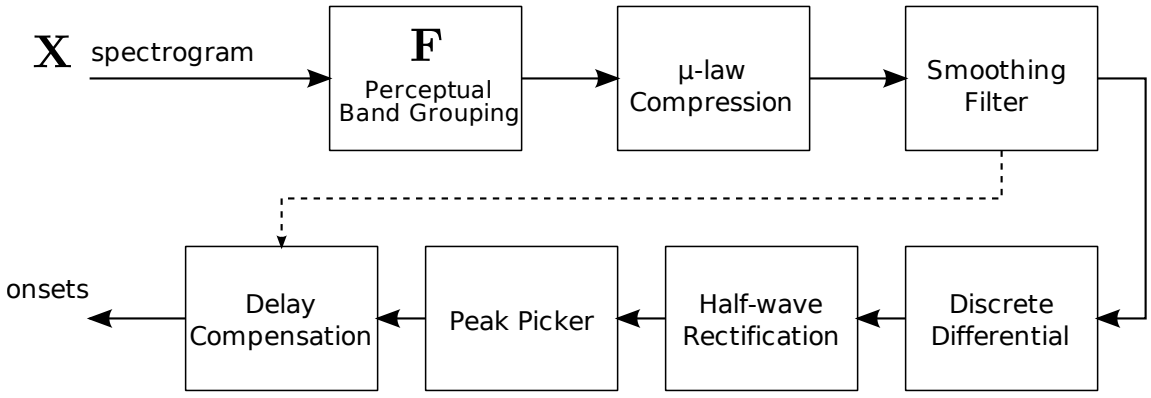


Figure 4.5: Onset detection system

It is also desirable to find the locations of the perceptual peaks associated with each perceptual onsets. We will refer to these locations as onset-peaks as opposed to simply onsets. Whereas an onset location is a frame where there is a local maximum in the energy derivative, an onset-peak location would be a frame following the onset where there is a local maximum in energy. Finding a local maximum in energy can be as simple as finding the local minimum location in the accent signal following the onset. However, for reasons

that will be made clear in Section 5.5, we would like to find the frame that contains the largest portion of the musical event that caused the onset.

To do this, instead of simply finding the first zero location in the accent signal we search for the location where many of the significant band-wise accent signals associated with the onset reach zero. In our algorithm we look for the first location where 8 of the 10 most significant bands reach zero. If no such location occurs, we simply find the frame that has the most zero bands. In our experiments, this method aligned the peak locations better with the frames that contain the largest portion of a musical event.

Chapter 5

Percussion Extraction System

We now focus on the specifics of our percussion separation system. A block diagram of the entire process is shown in Figure 5.1. The specifics of each block will be explained in the following sections.

5.1 Spectrogram Extraction and Onset Detection

5.1.1 Spectrogram Extraction

Extracting a spectrogram from an input audio signal involves taking the short-time Fourier transform (STFT) of that signal. In our case, this involves computing the discrete Fourier transform of every windowed frame of the input signal in order to compute the local frequency content. Important parameters that must be chosen include the analysis window size, the hopsize, and the window function.

A larger window size increases the frequency resolution of each frame while having a time smearing effect. We felt that by allowing each frame to contain about 100ms of audio, we would capture the entire percussive attack and a portion of the decay of most instruments, thereby increasing the chance that a percussive source would be represented as a single component rather than having an attack and release component. This window length comes out to about 4096 at 44.1kHz sampling rate.

The hopsize, or offset between frames, was chosen to be less than 10ms in order to allow for sufficient time resolution to accurately locate percussive peaks. We used a hopsize of 256 at 44.1kHz. For the analysis window function, we use the popular Hann function.

When computing the spectrogram we place the FFT components of each frame corresponding to the 2049 non-negative frequencies in the columns of a matrix $\mathbf{X}$. We send the magnitude, $|\mathbf{X}|$ on to the next step in the system, and retain the phase information $\angle\mathbf{X}$ for the final signal reconstruction step.

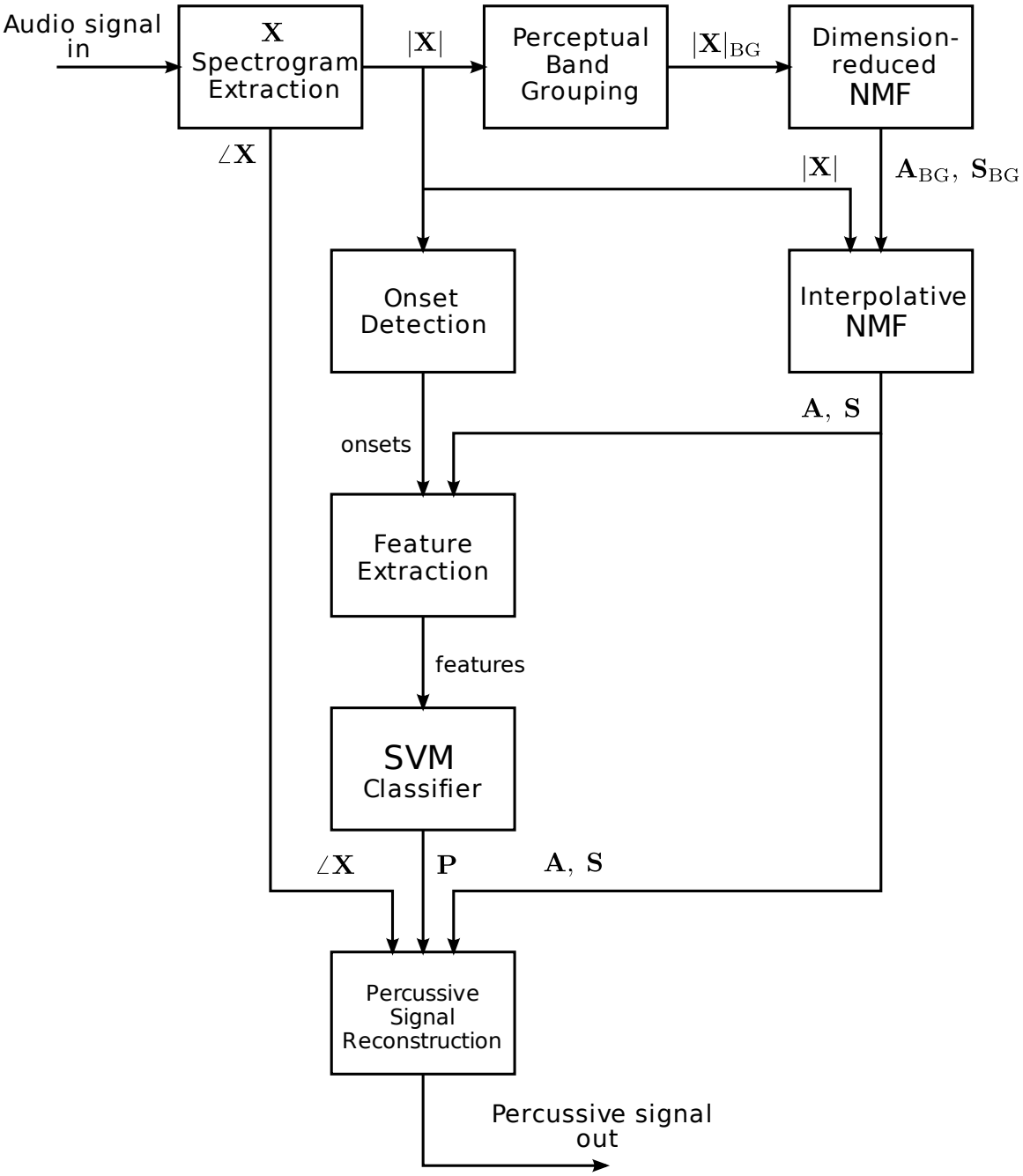


Figure 5.1: Block diagram of entire percussion extraction system

5.1.2 Onset Detection

Onset detection is carried out using the spectrogram magnitude, $|\mathbf{X}|$, and the procedure covered in Chapter 4 and outlined in Figure 4.5.

5.2 Perceptual Band Grouping

Before we begin NMF on $|\mathbf{X}|$, we carry out a perceptual band grouping on the columns of $|\mathbf{X}|$. This process is explained in Section 4.1 and can be represented as $|\mathbf{X}|_{\text{BG}} = \mathbf{F}|\mathbf{X}|$ where $\mathbf{F}$ mel-spaced triangular filters in its rows.

This step is done for two reasons. First, it allows the spectrogram $|\mathbf{X}|_{\text{BG}}$ to more closely mimic the frequency sensitivity of the human auditory system. Second, by reducing the number of rows in the matrix to undergo NMF, we greatly decrease the computation complexity of the system.

In our system we use 512 mel-spaced bands, which reduces the number of rows by a factor of 4 for a window size of 4096. In our trials, the quality of factorization was diminished when using 256 or 1024 bands.

One problem to address is fitting the integer-width filters to the continuous-valued mel warping. This isn't typically a problem when computing 40 or so mel filters for a length 1024 FFT as is done frequently in speech processing, but in our case it requires a little work. At low frequencies, the mel filters may be less than one FFT bin wide. In this case, we prevent oddly shaped filters by setting the width of the first filter to one and only increasing filter width by an integer amount when the mel curve catches up to the center frequency of the current filter. The plot in Figure 5.2 shows the result of this method of approximation.

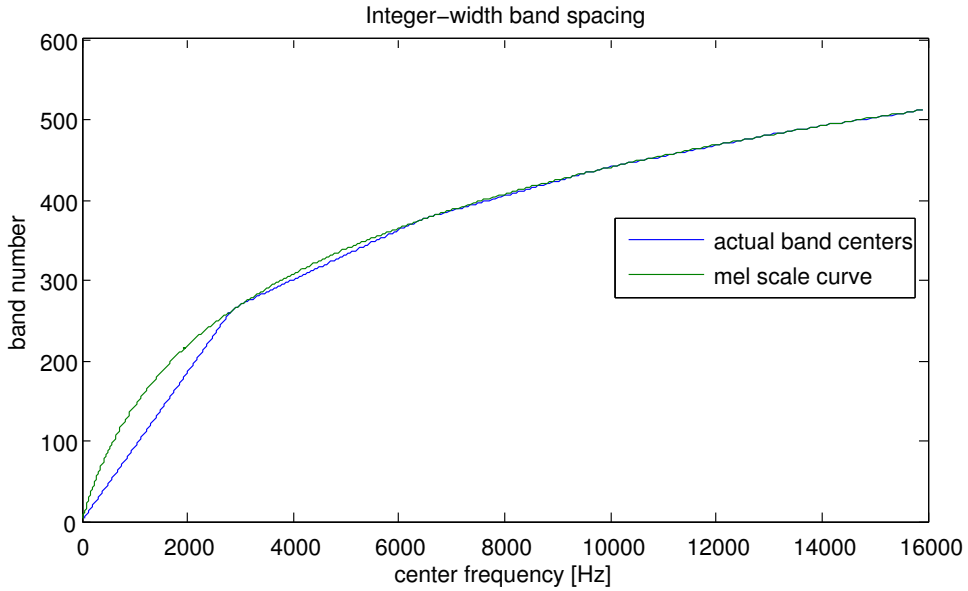


Figure 5.2: Integer bin width filters compared to actual mel curve

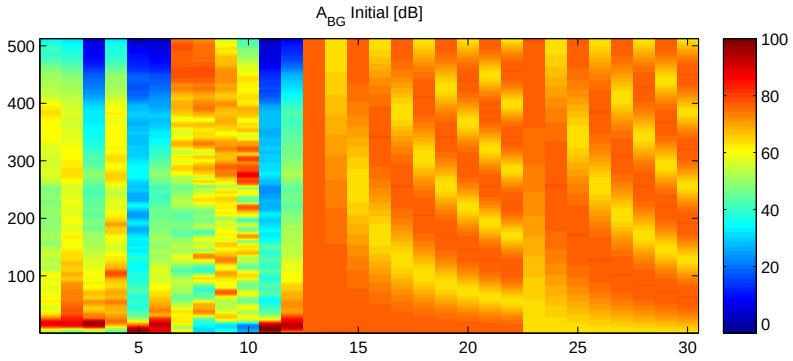


Figure 5.3: Initial value of $\mathbf{A}_{\mathrm{BG}}$

5.3 Dimension-reduced NMF

After the band grouping step, we can begin our iterative non-negative matrix factorization of $|\mathbf{X}|_{\mathrm{BG}}$. In order to perform the first update using eq. (2.3), we must start with an initial guess of the matrices $\mathbf{A}_{\mathrm{BG}}$ and $\mathbf{S}_{\mathrm{BG}}$. When using NMF, it is common to use rectified white noise to initialize these matrices. In this project we sought a more deterministic and music-based approach to initialization. When using $K = 30$ components in our factorization, we devote 12 of the components to typical spectra of percussive instruments common in popular music. These include snare drum with snare on and off, snare drum side stick, bass drum, tom toms, open and close hi-hat, ride cymbal, and ride cymbal bell. For the snare drum (snares on), bass drum, and toms, we count the close-mic and room-mic versions as separate instruments due to their distinct sound qualities. We compute the typical spectra for each instrument by computing the median spectra over a number of different audio samples.

The remaining 18 columns of $\mathbf{A}_{\mathrm{BG}}$ are made up of DCT basis functions 2 through 10 and their vertically flipped versions. A constant is added to each function so they are positive everywhere. This is done with the intent of spanning the remaining musical timbre space not covered by the drum spectra. The initial $\mathbf{A}_{\mathrm{BG}}$ we use is shown in Figure 5.3.

The rows of $\mathbf{S}_{\mathrm{BG}}$ are initialized to a constant value. The columns of $\mathbf{A}_{\mathrm{BG}}$ are the sole source of initial diversity in the updates. We then begin updating $\mathbf{S}_{\mathrm{BG}}$ and $\mathbf{A}_{\mathrm{BG}}$ by alternating the updates in (2.3). Since these updates are meant to reduce the divergence in (2.2, we stop the iterations when the change in divergence over a number of iterations is less than a certain percentage of the current divergence. We found that a relative change of 0.005 over 25 iterations was a good indicator of sufficient convergence. For a 20 second spectrogram, we saw on average about 250 iterations until convergence using this value.

After convergence we normalize the maximum value of each row of $\mathbf{S}_{\mathrm{BG}}$ to one and scale the corresponding columns of $\mathbf{A}_{\mathrm{BG}}$ accordingly. It is also desirable to sort the components (by reordering the corresponding rows

and columns of the factor matrices) by their overall energy contribution to the reconstructed spectrogram. This makes viewing the most important components much easier.

5.4 Interpolative NMF

In order to reconstruct an audio signal which contains only the percussive components, we must be able reconstruct a full spectrogram made up of the FFT components of each analysis frame in its columns. If we use $\mathbf{A}_{\text{BG}}$ and $\mathbf{S}_{\text{BG}}$ from the previous NMF step, we end up with only a dimension-reduced perceptual spectrogram. For this reason, we now interpolate the 2049 values of each FFT magnitude from the 512-band groupings. This interpolation is done using the information in the original magnitude spectrogram, $|\mathbf{X}|$, as a target for further NMF updates.

We initialize this second round of NMF updates with a matrix $\mathbf{A}$ formed from a linear interpolation of the $\mathbf{A}_{\text{BG}}$ matrix. To do this, we must retain the band centers of each of the filters used to create $|\mathbf{X}|_{\text{BG}}$ from $|\mathbf{X}|$. The interpolated initial $\mathbf{A}$ matrix is then iteratively updated using the divergence update formula along with $|\mathbf{X}|$ and $\mathbf{S}$. We choose to set $\mathbf{S} = \mathbf{S}_{\text{BG}}$ and not update it at all. We do this to prevent any updates on $\mathbf{S}$ from undoing the improvements caused by the perceptual band-grouping.

After $\mathbf{A}$ converges we have the final spectral contributions of each of the components. $\mathbf{S}$ shows when each of these components is present in the audio signal.

5.5 Feature Extraction

In order to train our SVM and use it to classify components as percussive or not, we must first compute features to be used in the classification.

For computing certain spectral features, it was helpful to compute typical spectra for each component. This augments the spectral contribution information, $\mathbf{a}_i$. To compute the typical spectra we first compute a component's "significance" in each frame, which is its gain signal squared divided by the sum of the gain signals of all components in that frame.

$$\text{significance}(i, j) = \frac{S_{ij}^2}{\sum_k S_{kj}}, \qquad \text{for component } i, \text{ frame } j \tag{5.1}$$

Once we have the significance, we compute the typical spectrum of a component by taking the element-wise median of the columns of $|\mathbf{X}|$ that correspond to frames with a significance in the top 5% for that component.

We use features from previous work as well as a few of our own. From the work of Uhle et al. [11] we use the percussiveness, noise-likeness, and spectral flatness features that were used to classify ICA components as drums or not. Helen and Virtanen [12] use features common in music information retrieval, including spectral centroid and mel-frequency cepstral coefficients (MFCCs), to classify drum components. The important

features used in this project are outlined below. It is helpful to refer to the ith column of $\mathbf{A}$ as $\mathbf{a}_i$ and the ith row of $\mathbf{S}$ as $\mathbf{s}_i$.

Features

1. *Percussiveness*

   This measures the degree to which the gain signal of a component, $\mathbf{s}_i$, exhibits a percussive attack with a quick exponential decay. The details of this feature can be found in [11]. For each gain signal, we first compute legitimate local max locations. These are local maxes that occur after the trailing local minimum has dropped to a percentage (50%) of the previous and current maximum, i.e. there is a sufficient "dip" between maxes. Also, a legitimate max has the largest value within 68ms of itself. Once we have found the legitimate maxes, we convolve an impulse at each max location with a percussively-shaped kernel. The result of this convolution is then correlated with the original gain signal to get the percussiveness feature.

2. *Noise-likeness*

   This feature describes how much the spectral contribution of a component, $\mathbf{a}_i$ resembles a noise spectrum. The details can be found in [11]. It is computed similarly to percussiveness, though a Gaussian kernel is used. This feature is used because many percussive instruments have noise-like spectra compared to the harmonic spectra of pitched instruments.

3. *Typical Spectrum Noise-Likeness*

   This is computed from the typical spectrum of a component using the procedure above.

4. *Spectral Flatness*

   Spectral flatness is the ratio of the geometric mean to the arithmetic mean of the values contained in $\mathbf{a}_i$. As its name implies, it aims to measure how flat, or "non-peaky", the spectrum of a component is. This feature also correlates well with the presence of noise-like spectra.

5. *Typical Spectrum Flatness*

   We also compute the flatness of the typical spectrum of a component.

6. *Peak Gain Ratio*

   This is the ratio of the gain signal energy around the perceptual peaks associated with onsets (as described at the end of Section 4.3) to the total energy in the gain signal. It correlates very well with percussive components since their significant frames tend to occur at these locations.

7. *Max Gain Ratio*

   The max gain ratio measures how much of the energy of a gain signal occurs around legitimate maxes (as found for the percussiveness feature). We take the ratio of the energy contained within a small window of each max to the overall energy of the gain signal.

8. *Onset-onset Ratio*

   We compute this feature by first finding the half-wave rectified difference signal of $\mathbf{s}_i$, and then finding how much of the energy of this positive derivative lies around onset locations found by the onset detection process covered in Chapter 4

9. *Periodicity*

   Percussive instruments very often play repetitive parts. Because of this, the gain signal of a percussive component is usually very periodic. We compute periodicity by computing the autocorrelation of $\mathbf{s}_i$ and then finding the ratio of the maximum value with non-zero lag to the average power of the signal.

10. *Harmonicity*

    In order to capture another measure of periodicity that would not favor a pure sinusoid, we compute a ratio using the power spectrum of the gain signal. Within the power spectrum, we find the maximum value besides the DC value. We then sum this value with the local maxima in the vicinity of multiples of the frequency of the maximum. This sum is then divided by the DC value to get the harmonicity feature.

11. *Noisiness*

    This feature measures how much the power spectrum of a gain signal resembles noise. This is done using the method used to measure noise-likeness for the spectral contribution of a component. The idea is that percussive signals with regular intervals have gain signals that are less similar to noise.

12. *Presence*

    Our presence feature measures how consistently a component occurs. In this regard, percussion instruments are more likely to be ever-present throughout a song, while specifics tones of harmonic instruments more sparse. To find this, we find the maximum of the gain signal within a sliding window of 100 frames. Then we find the mean of this resulting max signal.

13. *Impulsiveness*

    Impulsiveness is the ratio of the above presence feature to the mean of the gain signal. This measure how "spiky" the signal is.

14. *Energy*

    We compute the "energy" of a component as the sum of all elements of the component spectrogram matrix obtained by taking the outer product of a gain signal with its spectral contribution, i.e. $\mathbf{a}_i\mathbf{s}_i^{\mathrm{T}}$. This is the L1-norm of spectrogram matrix when all other components are set to zero.

15. *Heuristic Product*

    A heuristically determined product of features that correlate well with the occurrence of percussive

components was found to improve classification. This feature is actually a weighted geometric mean and was computed from the other features, where f_i is the ith feature on this list.

$$f_{15} = \left(\prod_i f_i^{p_i} \right)^{1/P}, \qquad P = \sum_i p_i \tag{5.2}$$

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15+
p_i	2	1	1	1	0	4	2	2	2	2	0	1	0	2	0

Table 5.1: Feature weights for heuristic product

16. *Spectral Centroid*

    The spectral centroid is the first moment of the spectral contribution, $\mathbf{a}_i$ of a component. We use the FFT bin frequencies as the location of each value in $\mathbf{a}_i$.

17. - 26. *MFCCs*

    The final 10 features are the first 10 mel-frequency cepstal coefficients [22] of $\mathbf{a}_i$. These describe the perceptual spectral envelope of a component.

5.6 SVM Classifier

Using the features from the previous section, the SVM classifier decides whether a component should be considered percussive or not. We normalize the features, so that across all data in the training set, each feature has a maximum of 1 and a minimum of 0. The radial basis kernel SVM is trained using hand-labeled components extracted from snippets of actual songs. The soft margin parameter, C, and the radial basis spread, σ, are chosen by a grid search using cross-validation.

When cross-validating, we use a song-wise grouping, that is, we train the SVM using the components from all songs except for the one we test, then repeat for each song and average the error rate.

The performance of the classifier is covered in Chapter 6.

5.7 Percussive Signal Reconstruction

After classifying components as percussive or not, we can reconstruct an audio signal containing only the percussive components (or only the non-percussive ones). We first compute a percussive magnitude spectrum by zeroing the contribution of non-percussive components. We can do this using the diagonal matrix $\mathbf{P}$, for which $\mathbf{P}_{ii} = 1$ if component i is classified as percussive and $\mathbf{P}_{ii} = 0$ otherwise. Then we have

$$|\mathbf{X}_p| = \mathbf{APS} \tag{5.3}$$

where $|\mathbf{X}_p|$ is the reconstructed percussive magnitude spectrogram. To estimate the complex percussive spectrogram, we use the phase information, $\angle\mathbf{X}$, retained from the spectrogram extraction step to get

$$\mathbf{X}_p = |\mathbf{X}_p| \exp(j\angle\mathbf{X}) \tag{5.4}$$

where the complex exponentiation is carried out element-wise. Because the window function used during spectrogram extraction was chosen appropriately, we can take the inverse FFT of each column of the percussive spectrogram and use overlap-add to reconstruct the percussive audio signal. This completes the percussion extraction process.

Chapter 6

Results and Conclusion

Now we deal with the performance of the percussion extraction system. First we address the quality of the percussive components extracted from the audio. Then the accuracy of the component classifier is evaluated. Last we mention directions for future work in percussive source separation and summarize our work.

6.1 Quality of Separation

To assess the quality of separation achieved by the iterative NMF algorithm, we utilized the MTG MASS Resources archive [23] of song snippets. Each song snippet in the archive includes all the unmixed individual tracks that comprise the multi-track recording. Most importantly, percussion instruments (mainly drums) are isolated on their own tracks. We use these percussion tracks as ground truth signals for ideal percussion extraction.

From the archive, we use 10 second excerpts from the 8 snippets containing suitable percussion. For each snippet used, we create a mix signal that contains all the tracks from the snippet mixed into one, and we create a percussion signal that is a mix of the tracks containing percussion.

We evaluate the separation by running one of three NMF routines on each mix signal. We then hand select the percussive components and reconstruct a percussion signal. The signal-to-noise ratio with respect to the ground truth percussion signal is used to measure the quality of separation.

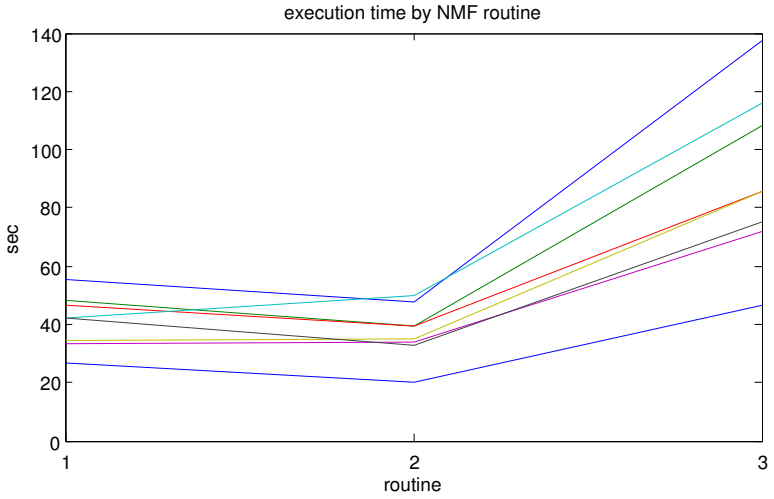
The first of the three NMF routines evaluated is our band-grouping interpolative scheme outlined in Sections 5.1-5.4.

The second method uses the band-grouping interpolative method but runs preconditioning iterations of the NMF updates on the matrix constructed from the columns of $|\mathbf{X}|_{\mathrm{BG}}$ corresponding to onset-peaks. These preconditioning iterations yield $\mathbf{A}$ and $\mathbf{S}$ from which we use $\mathbf{A}$ as the starting point for our NMF iterations on the complete $|\mathbf{X}|_{\mathrm{BG}}$ matrix. The motivation behind this last scheme is mostly speed-based. The NMF iterations on the onset-peak band-grouped magnitude spectrogram are very fast and hopefully result in a

matrix **A** that allows faster convergence of NMF in subsequent steps.

The third method is bare bones NMF on the full magnitude spectrogram matrix, $|\mathbf{X}|$, without any band-grouping or interpolation.

For each of the three methods we evaluate total execution time until convergence and signal-to-noise ratio (SNR). The execution time data is shown in Figure 6.1. The SNR results are shown in Figure 6.2. In Figure 6.1 we see that standard NMF performed on the entire FFT-bin magnitude spectrogram takes more than twice the time for conversion as the other two methods. Although onset-peak preconditioning performs slightly better than routine 1, the difference for this small sample size is not statistically significant.



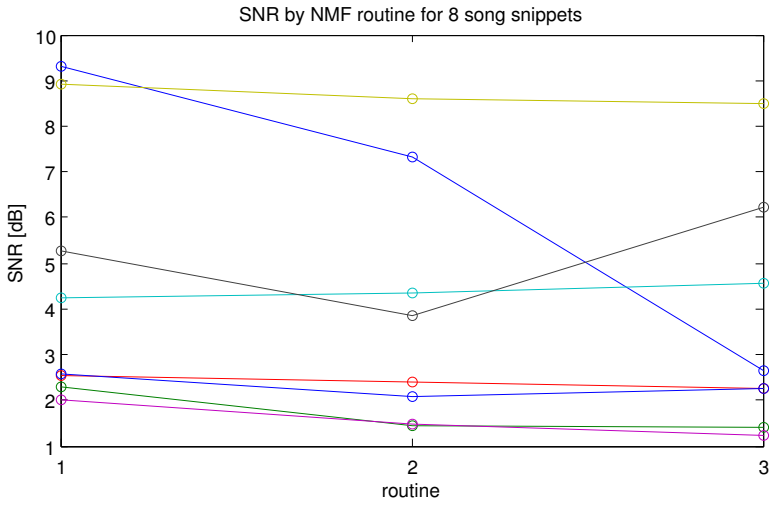
Routine	Description	Mean Execution Time
1	Band-grouped interpolative	41.1 sec
2	Routine 1 with onset-peak preconditioning	37.3 sec
3	Bare bones NMF	90.9 sec

Figure 6.1: Execution time until convergence for 3 NMF routines used on 8 song snippets

Figure 6.2 shows that routine 1, the band-grouped interpolative method without onset-peak preconditioning, achieves the best mean and median SNR. While the difference is not statistically significant, we can see that routine 1 gives better separation than routine 3 in 6 of the 8 songs while completing in less than half the time.

6.2 Classification Accuracy

To evaluate classification accuracy, we use a set of 20 second snippets from 32 songs. We restricted our song selection to rock songs which contain mainly guitar, vocals, and acoustic drums. This was done to



Routine	Mean SNR	Median SNR
1	5.63	3.49
2	4.76	3.20
3	4.41	2.46

Figure 6.2: Reconstructed percussion signal SNR for 3 NMF routines used on 8 song snippets

compensate for the small number of songs in our set which would not allow for a sufficient amount of training data to cover many genres.

Each song snippet is processed by the band-grouping interpolative NMF procedure using $K = 30$ sources. Each of the sources are hand labeled as percussive or not, and a feature vector is computed for each. If it is unclear whether a component is percussive or not, we always label it as not percussive. To evaluate classification accuracy, we use song-wise cross validation, that is, we train the SVM classifier with the components from all songs except for those from the test song. This is repeated for each of the 32 songs. Using this song-wise testing procedure, we can search for the best features and SVM parameters to use in our classifier.

6.2.1 Feature Selection

Of the 26 features we outline in Section 5.5, we must decide which features achieve the best classification results. Our feature selection measures are evaluated using the same set of 32 songs that we use to measure classification accuracy. A first measure of feature usefulness is the correlation coefficient, ρ, between each feature and the class labels. We use the absolute value, $|\rho|$, since a strong negative correlation would be a valuable source of information as well. If $\mathbf{f}$ is a vector of observations of a single feature and $\mathbf{y}$ is the vector

of corresponding class labels, the unbiased estimator of ρ is

$$\rho(\mathbf{f}, \mathbf{y}) = \frac{\sum_i (f_i - \bar{f})(y_i - \bar{y})}{(n-1)s_{\mathbf{f}} s_{\mathbf{y}}}, \qquad \text{where } s_{\mathbf{z}} = \sqrt{\frac{\sum_i (z_i - \bar{z})^2}{N-1}} \tag{6.1}$$

The rectified correlation coefficient, $|\rho|$, for each of the 26 features is shown in Figure 6.3. We can see that the heuristic product (15), peak gain ratio (6), and onset-onset ratio (8) all correlate well with percussive components.

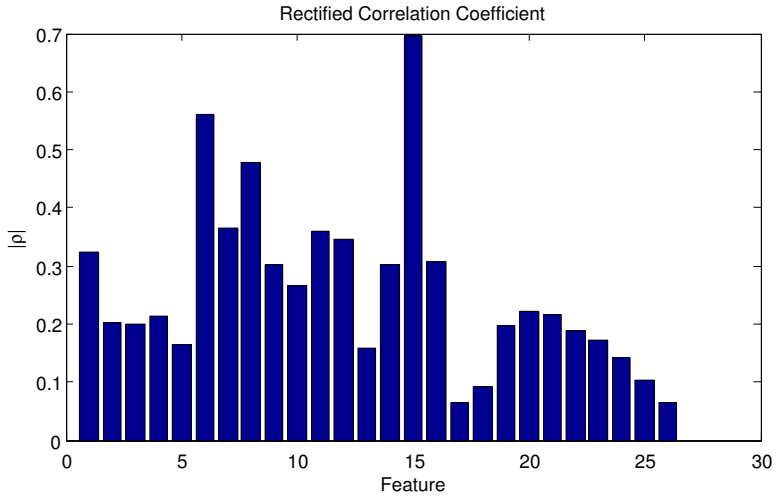


Figure 6.3: Absolute value of the correlation coefficient between a feature and the class labels

With this correlation data in hand, we could exclude the features with very small correlation coefficients from use in the SVM. This doesn't always turn out to be the best approach in practice. In applications where we desire a low (5%) error rate, features with a small $|\rho|$ may contain important information for borderline cases. As we will see, features with large $|\rho|$ can be a primary source of error as well.

In this project, we achieve the best results by excluding features if the SVM performs better without them, rather than by using some sort of statistical or information-based test. Using a sort of greedy algorithm, we iteratively eliminate the feature which causes the greatest increase in classification accuracy when it is removed. We stop when removing any of the remaining features does not improve performance.

For the 32 song dataset, the best results are achieved when we eliminated features {6, 11, 12, 13, 17, 18, 21, 25}. This is interesting because we exclude peak gain ratio (6) even though it has the second largest correlation coefficient. It seems that this feature was causing many of the borderline false positives due to an over-reliance of the SVM on it. Also excluded are noisiness, presence, and impulsiveness (11,12,13) and 4 of the 10 MFCCs. We are then left with an 18-dimensional feature vector for each component. In order to improve training speed, we could eliminate features that do not have any effect on the error; however, removing any of the remaining features increases the error rate.

6.2.2 Parameter Selection

We choose the soft margin parameter, C, and the kernel function spread, σ, using a grid search over possible values. The values which work best for the dataset used are $C = 8.2$ and $\sigma = 1.18$.

6.2.3 Classification Results

Because non-percussive components occur much more frequently than percussive components (about 8 to 1) in the test dataset, we report our system's classification error using the equal error rate (EER). For an SVM, the EER is the error rate achieved when the threshold parameter, b, is adjusted such that the rate of false positives and the rate of false negatives are equal. In this project, false positives occur when non-percussive components are classified as percussive, and false negatives occur when percussive components are classified as non-percussive.

Depending on the application, it may be desirable to adjust the threshold to favor one of the above types of errors at the expense of the other. We balance this trade-off by setting the threshold such that both error rates are equal and achieve a song-wise cross-validation EER of $\leq 3.7\%$. Of the 107 percussive components, only 4 (3.7%) are misclassified, and 20 (2.3%) of the 853 non-percussive components were classified as percussive. Because of the small size of the dataset, the threshold cannot be adjusted to find the actual EER, but 3.7% provides a safe upper bound for this particular data.

Because hand-labeling components requires considerable effort, our results are starved for additional data. A more automated approach to creating training data is taken by Helen and Virtanen [12] in which test audio signals are generated by mixing harmonic and percussive signals from various sources. In this approach, NMF components are extracted from each of the separate signals combined in the mix. Then the NMF components are labeled according to whether they came from a percussive signal or not.

Generating training data in this way is much more efficient than doing it by hand; however, it may fail to reproduce real-world conditions. First, harmonic instruments may have percussive transients that are important to rhythmic analysis. These transients may be desirable but would be labeled non-percussive by this procedure. Second, the way in which components are extracted and interact when a signal consists of both harmonic and percussive instruments is different than when they are isolated. Extracting training data in this way seems insufficient. Third, mixing arbitrary signals that are not necessarily rhythmically or harmonically related may yield unrealistic separation results.

For these reasons, we prefer our method of generating training data, despite the fact that it is less efficient.

6.3 Subjective Evaluation

While the engineer may actually care about the above error and SNR figures, the musician desperately wants to know how it sounds and how well it works in practice. It may be hard to believe that percussion

extraction system based on something as simple as spectrogram factorization would even work at all. In the majority of cases, the process yields a faithful representation of the important rhythmic content. The resulting percussion signal is by no means a high fidelity drum track, but it is definitely sufficient for rhythm matching or summarization.

In modern recordings with low temporal or spectral congestion (much of popular music excluding heavy metal), the extracted percussion signal can mimic a noisy drum track. In older low-fi recordings, the separation can be almost useless. The only example in our test data to fail so miserably is Dick Dale's Miserlou, which features a frenetic, distorted guitar that dominates the drums. In this song, the fast transients from the tremolo guitar picking are the only components resembling percussion. These transients are noisy and inconsistent so they do not work well as a rhythmic summary.

Recordings such as AC/DC's *Back in Black* and Pearl Jam's *Evenflow*, in which the drums are clearly audible to a human listener, yield percussive components that are identifiable as individual drums and give a faithful representation of the rhythm being played. Even though most songs do not reach this level of quality in their extracted drum tracks, there were only about 3-4 songs of the 32 tested that failed to produce a usable percussive signal. Overall, the system effectively provides useful rhythmic information about the percussion contained in short audio clips.

6.4 Future Work

A first step in improving the understanding of the usefulness of this system would be to run it on a much larger database of music. In order for the techniques presented here to work with a wide variety of music, a more larger, more diverse training set would be required. Since rough genre information is readily available for most digital music, a separate classifier could be used for separate genres in order to eliminate the need for a single general-purpose classifier.

To improve the performance of the NMF stage, one could use stereo audio signals as input to the system. In this project, we use monophonic signals to reduce computation time. Because separate instruments usually are panned to produce a stereo image, source separation should be improved by using a stereo spectrogram.

Another way to improve NMF for audio applications is to incorporate musical properties into the cost function to be minimized. Virtanen introduces one such method that introduces sparseness and temporal continuity into the cost function in hopes of improving audio source separation [17], but the results aren't improved much over simply using the divergence as the cost function. Along these lines, it could be beneficial to use separate harmonic and percussive cost functions for different portions of the $\mathbf{A}$ and $\mathbf{S}$ matrices.

In much of popular music, the spectral contributions of percussive instruments are fairly consistent throughout a song. With this in mind, segregating percussive from harmonic components so that the spectral contributions of harmonic components are allowed to change for each small snippet but percussive

components remain constant might further improve NMF results.

Real-world usefulness should be of primary concern. Evaluating the ability of the output percussion signal to improve drum transcription or serve as a basis for rhythmic comparisons would be important to the field. Also, computation time is important to the end user, so efforts could be made to produce faster algorithms or parallel implementations that would take advantage of today's multi-core architectures.

6.5 Conclusion

We have presented a system for the automatic extraction of percussive components from a digital audio signal. Our method improves upon methods which use NMF and SVMs. Novel contributions to the component extraction step include a band-group NMF routine that improves convergence time of existing iterative algorithms and a complementary interpolative NMF routine that transforms the primary source separation into a usable form. When coupled together, these two routines converge much faster than naive full-dimensionality NMF.

When classifying source separated components as percussive or non-percussive, we introduce new features to be used with the SVM. Some of these features rely on our onset detector, which provides valuable information about the perceptual location of sound events. Using a set of 32 song snippets, our our percussive component classifier achieves an equal error rate of less than 3.7%.

Our results should be tested on a much larger database of labeled music in order to confirm our findings and make improvements to the system.

Bibliography

[1] R. Typke, F. Wiering, and R. Veltkamp, "A Survey of Music Information Retrieval Systems," *Proceedings of the International Conference on Music Information Retrieval*, pp. 153–160, 2005.

[2] J. Downie, "Music information retrieval," *Annual Review of Information Science and Technology*, vol. 37, no. 1, pp. 295–340, 2003.

[3] J. Foote, "An overview of audio information retrieval," *Multimedia Systems*, vol. 7, no. 1, pp. 2–10, 1999.

[4] J. Aucouturier and F. Pachet, "Improving timbre similarity: How high is the sky," *Journal of Negative Results in Speech and Audio Sciences*, vol. 1, no. 1, pp. 1–13, 2004.

[5] E. Pampalk, "Computational models of music similarity and their application in music information retrieval," Ph.D. dissertation, Vienna University of Technology, Vienna, Austria, March 2006. [Online]. Available: http://www.ofai.at/ elias.pampalk/publications/pampalk06thesis.pdf

[6] J. Foote and S. Uchihashi, "The beat spectrum: a new approach to rhythm analysis," in *Multimedia and Expo, 2001. ICME 2001. IEEE International Conference on*, 2001, pp. 881–884.

[7] J. Foote, M. Cooper, and U. Nam, "Audio retrieval by rhythmic similarity," in *Proceedings of the International Conference on Music Information Retrieval*, vol. 3, 2002, pp. 265–266.

[8] J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, vol. 1, 2000.

[9] G. Peeters, "Rhythm classification using spectral rhythm patterns," in *Proceedings of the International Conference on Music Information Retrieval*, 2005, pp. 644–647.

[10] K. Yoshii, M. Goto, and H. Okuno, "AdaMast: A Drum Sound Recognizer based on Adaptation and Matching of Spectrogram Templates," *1st Annual Music Information Retrieval Evaluation eXchange (MIREX)*, 2005.

[11] C. Uhle, C. Dittmar, and T. Sporer, "Extraction of drum tracks from polyphonic music using independent subspace analysis," *Fourth International Symposium on Independent Component Analysis and Blind Signal Separation, Nara, Japan*, pp. 1–4, 2003.

[12] M. Helen and T. Virtanen, "Separation of drums from polyphonic music using nonnegative matrix factorization and support vector machine," *Proc. EUSIPCO*, vol. 2005, 2005.

[13] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, vol. 13, no. 4-5, pp. 411–430, 2000.

[14] M. Casey and A. Westner, "Separation of mixed audio sources by independent subspace analysis," *Proceedings of the International Computer Music Conference*, 2000.

[15] M. Plumbley, "Algorithms for nonnegative independent component analysis," *Neural Networks, IEEE Transactions on*, vol. 14, no. 3, pp. 534–543, 2003.

[16] D. Lee and H. Seung, "Algorithms for Non-negative Matrix Factorization," *Advances In Neural Information Processing Systems*, pp. 556–562, 2001.

[17] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, 2007.

[18] C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] A. Klapuri, "Sound onset detection by applying psychoacoustic knowledge," *Acoustics, Speech, and Signal Processing, 1999. ICASSP'99. Proceedings., 1999 IEEE International Conference on*, vol. 6, 1999.

[21] E. Scheirer, "Tempo and beat analysis of acoustic musical signals," *The Journal of the Acoustical Society of America*, vol. 103, p. 588, 1998.

[22] L. Rabiner and B. Juang, *Fundamentals of speech recognition*, 1993.

[23] M. Vinyes, "MTG MASS Resources," http://www.mtg.upf.edu/static/mass/resources, 2008.