Musical Hit Detection

CS 229 Project Milestone Report

Eleanor Crane
Sarah Houts
Kiran Murthy

December 12, 2008

1 Problem Statement

Musical visualizers are programs that process audio input in order to provide aesthetically-pleasing audio-synchronized graphics. In popular music, musical instrumentation changes known as hits are an important indicator of changes in the music's mood. Ideally, a visualizer should respond to a hit by also changing the mood of the displayed graphics to match the music. This project will focus on using machine learning techniques to detect hits, and therefore mood changes, in a song.

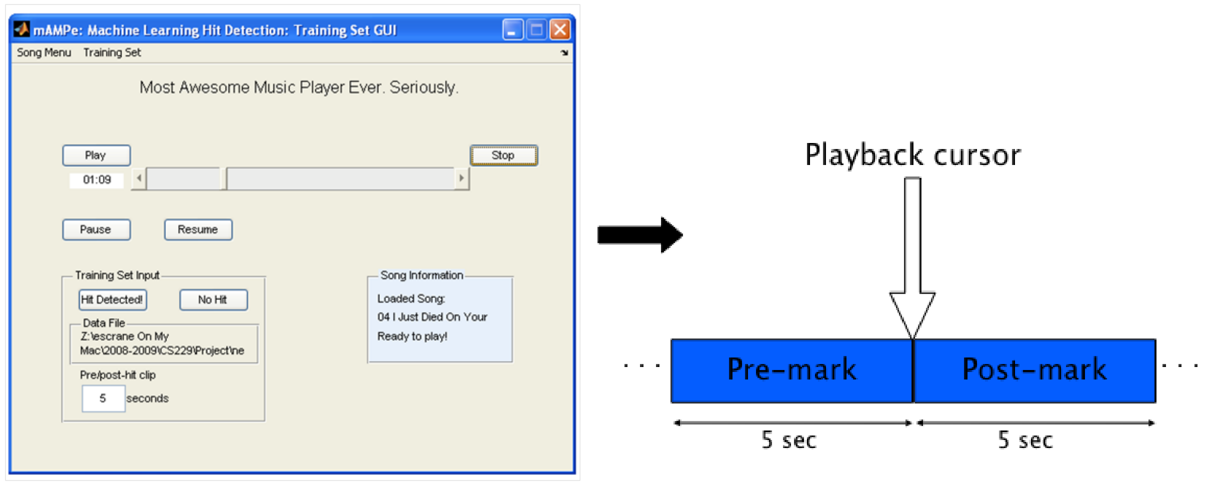
2 Approach

2.1 Data Collection

Our approach utilizes supervised learning to train a hit detection algorithm. The supervised learning approach is used since it is easy for a human operator to label hits within a song. Additionally, a hit detector should operate on only a small segment of music data ahead of the current playback location, facilitating hit detection in streaming music. Thus, the learning algorithm only takes into account music data in the vicinity of hits.

We start by selecting a set of songs containing strong mood changes and another set of songs without mood changes. As the song plays, a human operator marks hit/no-hit locations in the song. At each specified mark, two 5-second musical clips are extracted from the song: one clip ending at the mark (pre-mark clip) and one clip beginning at the mark (post-mark clip). These musical clips and their associated hit/no-hit labels are imported into MATLAB where they are fed into a supervised learning algorithm.

A GUI assists the marking of hit/no-hit locations in songs. Figure 1 shows a screenshot of the GUI.



01:09

Z:\escrane On My Mac\2008-2009\CS229\Project\ne

5

Loaded Song:
04 I Just Died On Your
Ready to play!

Figure 1: Music hit/no-hit marking GUI

Dubbed mAMPe, the GUI allows the user to load mp3 songs and choose a data file to which hit/no-hit labels and clip file names are stored. The GUI also allows the user to play the song, and as the song comes across a hit/no-hit, the user may click on a set of buttons to automatically save the pre-mark/post-mark clips.

2.2 Feature Selection

Changes in song mood generally correspond to changes in:

- Beat frequency
- Beat amplitude
- Instrumentation (current set of instruments playing)

From the pre and post-mark clips, a feature describing the hit/no-hit status of the clips is calculated. We note that musical mood changes correspond to changes in music amplitude and instrumentation. While a change in amplitude can be assessed from the time domain, a change in instrumentation is expressed much more clearly in the frequency domain. In order to capture changes in amplitude and instrumentation with as few calculations as possible, the Power Spectral Densities (PSDs) of the pre and post-mark clips are used to construct the feature vector.

As Figure 2 shows, the absolute value of the difference of the 129-point pre- and post-mark PSDs serves as our feature vector. It should be noted that deriving the feature from the PSD

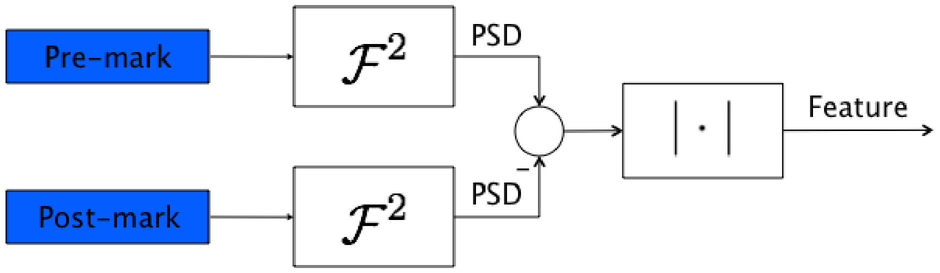


Figure 2: Feature creation algorithm

yields better performance than deriving the feature from the Fast Fourier Transform (FFT). In particular, Figure 3 shows that the PSD-feature's learning curve shows improvement over test set error as more training samples are added to the training set, while the FFT-feature's learning curve does not show this improvement. This result implies that the PSD contains information more relevant to hits than the FFT.

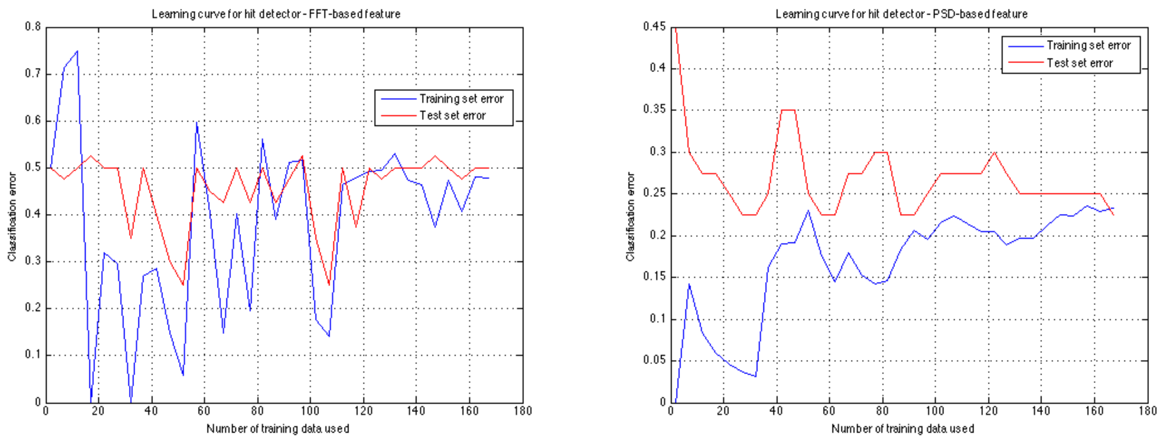


Figure 3: Learning curve with FFT-based feature (left) and PSD-based feature (right)

A likely explanation for the performance improvement is that the differenced PSD better captures changes in musical intensity (waveform energy) than the FFT, and sudden changes in musical intensity are very indicative of hits.

2.3 Training the Classifier

The hit detector takes the form of a Support Vector Machine (SVM), trained using a regularized, kernelized SMO algorithm. To test the trained SVM, hold-out cross-validation was used - 20% of the total data set was randomly held out of the training data in order to calculate test error.

The kernelization was chosen after our initial SVM training using non-kernelized features yielded 30% training and test set error - an unacceptable level of performance. Since the training and test set error were similar, performance improvements could only be made by reducing bias. Thus, kernelization was used as a method to increase the dimensionality of our feature space. Several different kernels were evaluated, and a Gaussian kernel resulted in the best performance with 22% training and test set error. This performance was adequate to reliably detect hits.

2.4 Classification

Given a set of contiguous pre-mark and post-mark sound clips, the classification will classify the mark as a hit based on the result of kernelized classification expression:

$$\sum_{i=1}^{m} \alpha_i y^{(i)} K(x^{(i)}, x) + b \geq 0$$

During real-time song playback, the hit detector periodically computes the feature vector from pre- and post-mark clips relative to the current playback location, then uses the kernelized SVM classification to classify whether that particular point in the song is a hit or a non-hit. Additionally, the “confidence” of the hit can be judged by the functional margin of the real-time feature.

3 Results

In order to test the hit detection algorithm, we developed a MATLAB GUI to display hit information via a simple visualizer while playing test songs, shown in Figure 4. The visualizer shows the low frequency portion of the FFT of song data for the next second. The colors correspond to the hit or non-hit label that the algorithm has assigned to that moment during the song. A non-hit will be displayed in green and blue shades, whilst a hit will be shown as a shade between yellow and red, increasing from yellow to red with increasing confidence levels.

The online classifier correctly identified hits from many songs from popular music. However, the classifier also produced erroneous hits/no-hits, leading to two key observations:

1. Though musical hits are instantaneous events, there are frequently build-ups, such as drum rolls or guitar riffs, leading up to that instant. As a result, the algorithm will detect hits during this build-up period with increasing confidence. For a brief period after a hit, the change can still be apparent in the feature vector, causing a taper-off period back down from a hit classification to a non-hit classification in the new section after the hit.

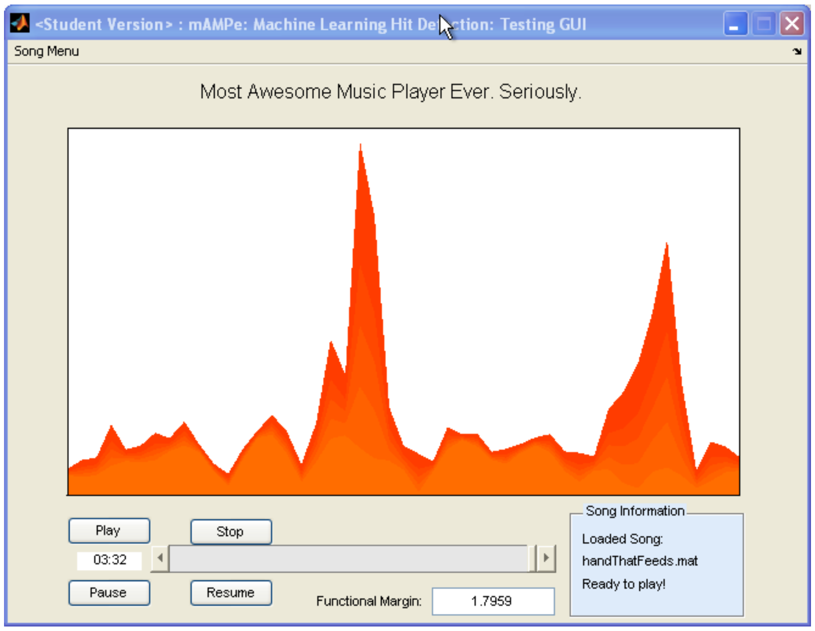


Figure 4: Hit Display GUI

2. A high-volume vocal track without any other change in underlying beat or instrumentation would frequently result in false hit identifications. This may have stemmed from a lack of vocals in our pre- and post-mark clips in our training data.

As MATLAB is not a real-time operating environment, we were not able to identify hits in real-time. Thus, we pre-computed at hit classification of each second of the song to provide a demonstration of how well the algorithm works while playing the song. A future version of the classifier should include a multi-threaded process which could easily perform the same classification in real-time.

4 Future Work

There are a number of areas that could be investigated to further enable correct hit-classification. As previously mentioned, hits are sometimes preceded by a crescendo in the music which can decrease the probability of a correct detection. The build-up error could possibly be rectified by inserting a gap between the pre-mark and post-mark clips and increasing the clip length. This would make the pre- and post-mark clips more distinct, allowing for better classification.

In order to implement a hit detection algorithm with a visualizer, the hit detection's functional margin can be fused with other musical property sensors, such as beat detection in order to detect tempo changes. For actual usage, this algorithm would need to be recast in C or C++ to enable real-time hit detection and appropriate visualization.