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.

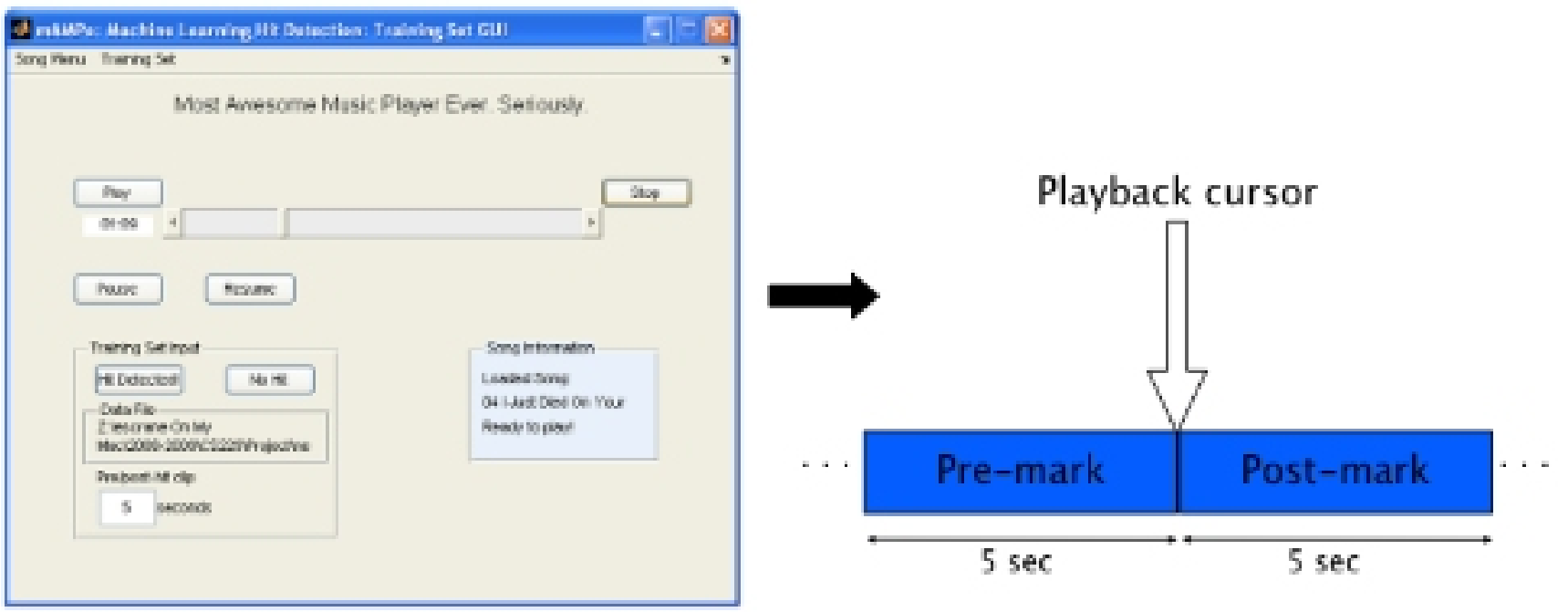

Figure 1: Music hit/no-hit marking GUI

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

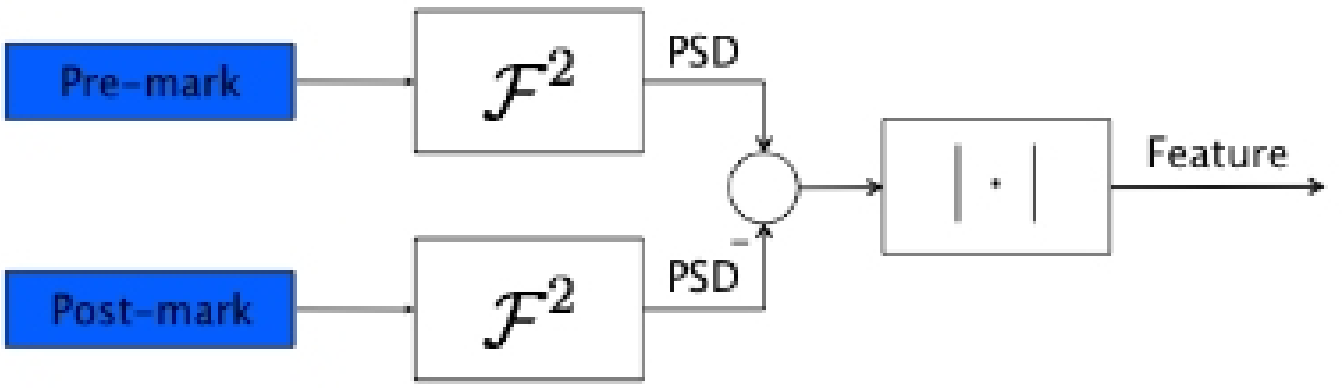

Figure 2: Feature creation algorithm

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.

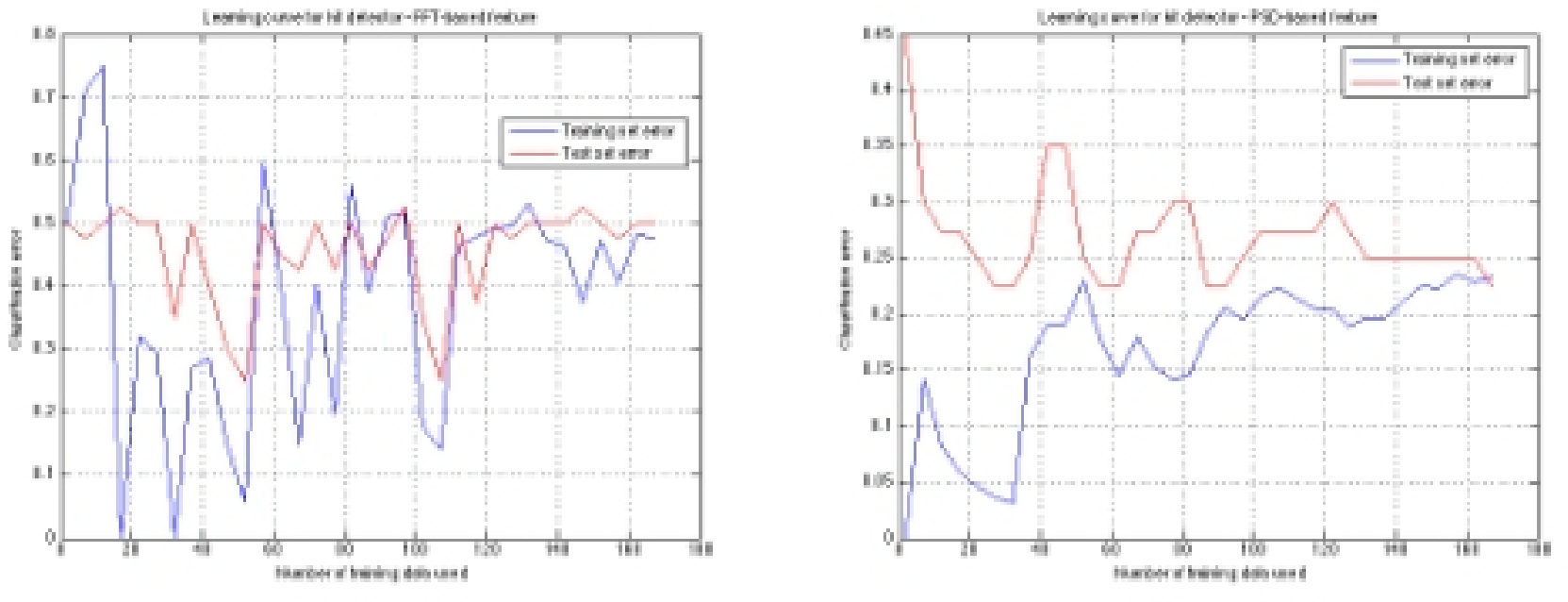

Figure 3: Learning curve with FFT-based feature (left) and PSD-based feature (right)

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.