Time-Series and Frequency-Series Features for Musical Instrument Classification

Landon Buell

3 June 2020

1 Introduction

2 Time Series Features

Time series features (also called time-domain or time-space features) are pieces of information that can be extracted, derived, or otherwise originate from a signal that in expressed with a dependence on time. This can include weather forecasts, year-to-year population of a given demographic, audio information, or economic stock data. In the context of audio data, this is generally referred to as a *waveform*. Given that time-series data (and all data) stored digitally must contain evenly-spaced, discrete samples, S is used to indicate a signal stored in an array-like object with it's elements S_i (where $i \in \{0, 1, 2, ..., N-2, N-1\}$) make up the waveform.

Additiona

2.1 Rise and Decay Time

Signal *rise time* measures the amount of time it takes for a time-domain signal, S, to leave it's steady-state condition and *rise* to a given maximum $\max\big[|S|\big]$. Signal *decay time* measures the amount of time it takes for a time-domain signal, S, to *decay* from a given maximum value and return to it's steady-state value. Due to the nature of digitally stored audio in both time and amplitude, it may be impossible to find entries in the signal that satisfy exact equilibrium values (usually, 0).

To account for this, we employ a function within our feature-extraction program that measures the 10% to 90% amplitude rise time, and the subsequent 90% to 10% amplitude decay time. This means that we produce two array-like objects for each signal. The first array will hold the position (or index) of all points that have a value that exceeds 90% of the waveform maximum (accounting for negative amplitudes as well). The second array contains similarly index of values that exceed 10% of the maximum, but not 90%.

If we take the difference between the 0-th entry of each array, this is the number of samples that it took for the signal to reach (roughly) 10% of it's maximum amplitude to (roughly) 90% - this *rise time*, Δt_R. Similarly, if we take the difference between the final entry in each array, this gives the *decay time*, Δt_D. Both measurements are in units of the *number of samples*; to convert into more traditional units of time (e.g. seconds), one can multiply by the spacing between samples or divide by the sample rate.

Algorithm 1 Compute amplitude rise and decay time of a discrete time-domain signal S for a give lower bound percentage and upper bound percentage. Note that this uses a brute-force,loop through the full signal. Optimization methods for longer signals are encouraged

```
Require: Discrete time series signal: $S$ of length $N$
Require: Value to consider lower bound percentage (0,1): low_bnd
Require: Value to consider upper bound percentage (0,1): high_bnd
Ensure: low_bnd < high_bnd
  $S \leftarrow |S|$
  max_amp ← maximum($S$)
  above_upper ← {}
  above_lower ← {}
  for $i = 0, 1, 2, ..., N-2, N-1$ do
    iterate through time, and add index values to respective lists
    if $S_i \geq$ high_bnd then
      above_upper.insert($i$)
    else if $S_i \leq$ high_bnd and $S_i \geq$ low_bnd then
      above_lower.insert($i$)
    else
      continue
    end if
  end for
  M ← || above_lower$_{(0)}$ - above_lower$_{(-1)}$ || - Length of active waveform by number of samples
  risetime ← || above_upper$_{(0)}$ - above_lower$_{(0)}$ ||/M - Rise time
  decaytime ← || above_upper$_{(-1)}$ - above_lower$_{(-1)}$ ||/M - Decay time
  return risetime , decaytime
```

Additionally, notice we normalize both rise and decay time with respect to the amount of samples between the initial rise and the final decay. We do this to eliminate dependence on absolute sample difference. If a signal is truncated, contains excessive quiet time or additional zero-padding, the interpretation of rise/decay time is then changed to a percentage or fraction of total active-waveform time.

2.2 Time-Frames

A *frame* is a unit of time discussed by Khan & Al-Kathib and Liu. [1, 2]. Each frame is a collection of points from a discrete time-series waveform that have been concatenated into a new, smaller object. For frames of smaller length, a single waveform can produce hundreds or thousands of frames, where larger frame lengths may only permit dozens of frames from a waveform. The idea is to divide a waveform into sections with lower time resolution in order to extract additional features. A discrete signal S can be decomposed into frames s such that:

$$S = \big[s_0, s_1, s_2, ..., s_{N-2}, s_{N-1}\big] \tag{1}$$

Where a single entry s_i contains a subset of samples from the greater signal S, and can function as it's own discrete time-series object. Frames can also can also be overlapping or non-overlapping. It is recommended to use over-lapping frames when producing a spectrogram-like object or producing some sort of energy-amplitude envelope. We outline an algorithm that extracts an array-like object of frames from a signal S using freame of fixed length:

Algorithm 2 Use discrete signal S to produce an array-like object X of time-frame signals of fixed length.

```
Require: Discrete time series signal: $S$ of length $N$
Require: Number of samples per frame: $k$. Recc. $\log_2(k) \in \mathbb{Z}$
Require: Percentage of frame overlap: $L$ with $L \in [0, 1)$
Ensure: $N/k \in \mathbb{Z}$ - Use zero-padding or truncation. This ensures that no frames are partially filled
  dt $\leftarrow k * (1 - L)$ - integer step size
  X $\leftarrow \{\}$ - array to hold all frame arrays
  for $i = 0$ to $i = (N - k - 1)$ step dt do
    Create frame using index $i$ to $i + k$ from signal $S$
    frame $\leftarrow \big\{S[i : i + k]\big\}$
    X.insert(frame)
  end for
  return X
```

When constructing frames from is signal S of N samples, a designer can choose to use produce a set number of frames, or a set number of sample per frame. When choosing a fixed number of frames, m, there will be N/m samples in each frame. When choosing a fixed number of sample per frame, k, there will be N/k frames. It is necessary to truncate or pad the signal S such that either $N/m \in \mathbb{Z}$ or $N/k \in \mathbb{Z}$. It is important to note that the frames themselves are not classification features, but are a vital step in deriving further classification features.

2.3 Waveform RMS Energy

The Root-mean-squared energy (RMS)of a waveform can be used to approximate the average power in a waveform using Hooke's Law. The RMS of some discrete function S with N samples is given by:

$$S_{RMS} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} S_i^2} \tag{2}$$

2.4 Time Spectral Flux

Time-Spectral-Flux (TSF) is a feature proposed by [1]. It measures the magnitude of the incremental change between adjacent elements in a given sequence. For a discrete signal with unit sample rate, this can be loosely though of as a first-derivative approximation. Given a signal S with N samples, the TSF, Φ, will be an array-like object $N-1$ samples. The i-th sample is given by:

$$\Phi_i = \big(S_{i+1} - S_i\big) \tag{3}$$

TSF can be used with the raw waveform signal S directly, with time-frame-like objects X or any other time-spectral array. Depending on the context, the right side of eqn. (3) can be multiplied by come constant c if values need to be scaled. This could be used to normalize the array, or to account for physical properties such as sample rate. In some cases, it may be beneficial to compute the magnitude of the difference between adjacent elements. Below we show an algorithm to compute TSF:

Algorithm 3 Compute Time-Spectral-Flux Φ of discrete signal S

```
Require: Discrete time series signal: $S$ of length $N$
  $\Phi \leftarrow \{\}$
  for $i = 0, 1, 2, ..., N-3, N-2$ do
    $\Phi$.insert($S[i+1] - S[i]$)
  end for
  Optional: Apply additional functions or scaling. Normalization, scaling, absolute value,
  etc. can also be used and optimized depending on context
  return $\Phi$
```

Physically, TSF describes the frame-by-frame or index-by-index change in the shape of a spectrum. Signals containing speech have shown to have higher average spectral flux values than than of musical notes [1, 5]. Similarly, percussive instruments will demonstrate higher average frame TSF values than instruments that demonstrate greater sustain.

Interval of Zero-Crossings

Kahn & Al-Kathib [1] proposed a feature using the difference of maximum and minimum zero crossing in given time intervals as a feature vector when differentiating between speech and audio. This produced a much more distinct decision boundary than an ordinary zero crossing counter. Building on this, we count the number of zero crossing in each time-frame X_i.

3 Frequency Series Features

Frequency series features (also called frequency-domain or frequency-space features) are peices of information that can be extracted, derived or otherwise originate from a signal that is expressed with a dependence on frequency. This is commonly done when needing to break down a give signal into it's simplest possible constituent frequency [3]. We move a discrete signal of length N that is a function of time, S, into a signal that is a function of frequency F, by applying the *Discrete Fourier Transform* (DFT). This produces an array of N complex-values numbers, the k-th element is given by:

$$F_k = \sum_{n=0}^{N-1} S_n \cdot e^{-2\pi i \frac{kn}{N}} \tag{4}$$

Where F is an array Similarly, we recover the original time-basis signal S with the *Inverse Discrete Fourier transform* (IDFT). The k-th element is given by:

$$S_k = \frac{1}{N} \sum_{n=0}^{N-1} F_n \cdot e^{-2\pi i \frac{kn}{N}} \tag{5}$$

Equations (4) and (5) create a *transform pair.*

3.1 Number of Unique, Isolated Peak

3.2 Energy Spectral Density of Frequency Bands

One of the most defining characteristics of the timbre of a sound is the it's distribution of power in frequency space. To do this, we divide a section of frequency space into multiple frequency *banks.* Each bank is a subset of frequency space. Given the physical properties of vibrating strings, membranes and air columns, we know that successive overtones which appear are large isolated *spikes* in the frequency domain tend to occur at positions that are integer multiples of the fundamental frequency.

By dividing frequency space into k unique, non-overlapping banks, we can produce an estimate for the power of a musical instrument in each given band. For some signal S, represented in a frequency-basis F, we can compute the power *energy spectral density* (ESD), $\mathbb{E}$ in the band from frequency f'_a to f'_b as given by:

$$\mathbb{E} = \sum_{i=f'_a}^{f'_b} |F_i|^2 df \tag{6}$$

To improve numerical accuracy,this direct Reimann Summation can be replaced with a different integration method if desired. The resulting real floating-point number is the energy density of that signal in that given frequency band. We show an algorithm for computing the energy spectral density in of a frequency-space signal F.

Algorithm 4 Compute energy spectral density of banks in frequency space array F

```
Require: Discrete frequency series signal: $F$ of length $N$
Require: Array-like object: $x$, length $N$, maps each index of $F$ to a real-valued frequency bin.
Require: Array-like object: $P$, containing $k$ upper bound / lower bound pairs.
Require: Integration function, method or operation: $INTG$
Ensure: $P$ array-object has shape $(k \times 2)$, with format resembling:
  $P = \Big[\{P_{(0,0)}, P_{(0,1)}\}, \{P_{(1,0)}, P_{(1,1)}\}, ..., \{P_{(k-1,0)}, P(k-1,1)\}\Big]$
  Where $P_{(i,0)}$ is the lower bound on bank $i$ and $P_{(i,1)}$ is the upper bound on bank $i$.
  Initialize array-like object to the the Energy-Spectral-Density (ESD) of each frequency bank:
  Bin_ESDs ← {}
  for $i = 0, 1, 2, ..., k-2, k-1$ do
    Iterate through each bank pair, find indecies where
    low_bnd , high_bnd ← $P_{(i,0)}$ , $P_{(i,1)}$
    idxs ← {where $x[j] \geq$ low_bnd and $x[j] \leq$ high_bnd }
    Use entries in "idx" array to find the energy at the frequency band values, integrate the subset of the array to compute ESD of
    E ← INTG$\Big[|F[\text{idxs}|^2\Big]$
    Bin_ESDs.insert(E)
  end for
  return Bin_ESDs
```

4 Combined Series Features

4.1 Spectrogram Matrix

4.2 Phase-Space Matrix

5 Additional Features

Summary

References

[1] Khan, M. Kashif Saeed, and Wasfi G. Al-Khatib. “Machine-Learning Based Classification of Speech and Music.” Multimedia Systems, vol. 12, no. 1, 2006, pp. 55–67., doi:10.1007/s00530-006-0034-0.

[2] Liu, Zhu, et al. “Audio Feature Extraction and Analysis for Scene Classification.” Proceedings of First Signal Processing Society Workshop on Multimedia Signal Processing, 1998, pp. 61–79., doi:10.1109/mmsp.1997.602659.

[3] Peatross, Justin, and Michael Ware. *Physics of Light and Optics*. Brigham Young University, Department of Physics, 2015.

[4] Short, Garcia, “Signal Analysis Using the Complex Spectral Phase Evolution (CSPE) Method”. Journal of the Audio Engineering Society.

[5] Zhang, Tong, and C.-C. Jay Kuo. “Content-Based Classification and Retrieval of Audio.” Advanced Signal Processing Algorithms, Architectures, and Implementations VIII, 2 Oct. 1998, pp. 432–443., doi:10.1117/12.325703.