Open in app ↗

Published in Towards Data Science

mlearnere Follow

Mar 13, 2021 · 5 min read · ✨ · Listen

Save

Learning from Audio: Time Domain Features

Going deeper in time.

Introduction

While Deep Learning often utilizes processes in the frequency domain, there are still many relevant features to be leveraged within the time domain that are relevant to many Machine Learning techniques. Simply put, these features can be extracted and analyzed to understand the wave form's properties. When extracting features within the time-domain, we will generally study the amplitude of each sample. How we manipulate the amplitude gives us certain details about the signal in question.

Related Articles:

- Learning from Audio: Wave Forms
- Learning from Audio: Fourier Transformations
- Learning from Audio: Spectrograms
- Learning from Audio: The Mel Scale, Mel Spectrograms, and Mel Frequency Cepstral Coefficients
- Learning from Audio: Pitch and Chromagrams

Before beginning I would like to establish some notation:

43

$s(k)$ refers to the Amplitdue of the k^{th} sample
K is the frame size, or the number of samples within each frame.
t represents the frame number.

Within these examples, I will elaborate on what the feature is, how to formally define it, and show how to extract the features in Python. In the previous examples, we looked at certain sounds of instruments. However, in this example, we will look at random 7 second snippets of songs across different genres (specifically R&B, Rap, and Rock) as we will be able to better see the properties of these features.

For copyright purposes, I will not be able to share the songs in question, however I will share the output plots as well as the genre of the songs. This will allow us to study nuances between genres.

As always, the repository of the all the code can be found on the Learning from Audio GitHub repository.

Note: all figures are by author.

Amplitude Envelope

The Amplitude Envelope (AE) aims to extract the maximum amplitude within each frame and string them all together. It is important to remember that the amplitude represents the volume (or loudness) of the signal. First, we split up the signal into its constituent windows and find the maximum amplitude within each window. From there, we plot the maximum amplitude in each window along time.

We can use the AE for onset detection, or the detection of the beginning of a sound. In various speech processing applications this could be someone speaking or external noise, whereas in Music Information Retrieval (MIR) this could be the beginning of a note or instrument.

The main downfall of the AE is that is not as robust to outliers as Root-Mean-Square Energy which will we study soon.

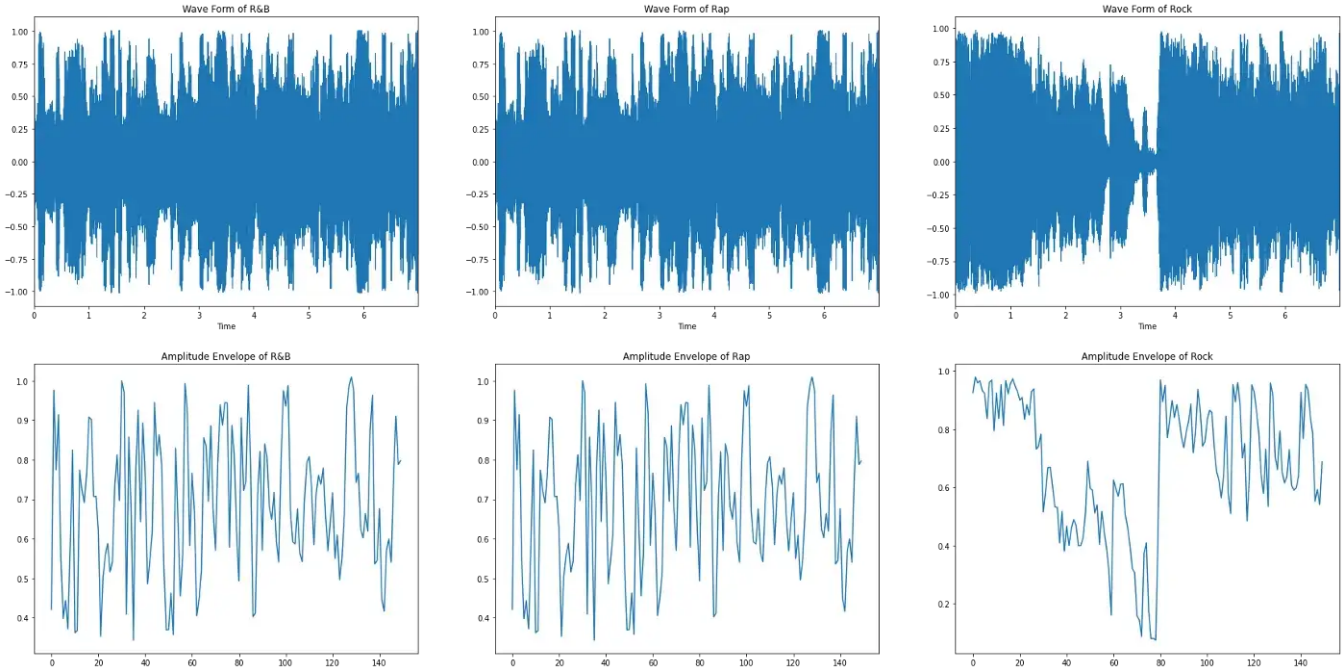
Here is how we can formalize this concept:

$$\mathrm{AE}_t = \max_{k=t\cdot K}^{(t+1)\cdot(K-1)} s(k)$$

Upon searching for a defined method in Python that could accomplish this task, I could not find it. Therefore, we will define it from scratch as it will be quite easy to do. The other feature extraction methods we will be looking at have already been defined in `librosa`, so we will be using those functions after formally defining them.

It is important to note that by the setup in this `for` loop, that we do not designate `hop length`. This means that when we create our upper and lower bounds, the windows do NOT overlap, making the `hop length` and the `frame length` the same.

Now, to visualize and compare the AE among genres:


Wave Form of R&B
Wave Form of Rap
Wave Form of Rock
Time
Amplitude Envelope of R&B
Amplitude Envelope of Rap
Amplitude Envelope of Rock

Root-Mean-Square Energy

As mentioned previously, the Root-Mean-Square (RMS) Energy is quite similar to the AE. As opposed to onset detection, however, it attempts to perceive loudness, which can be used for event detection. Furthermore, it is much more robust against outliers, meaning if we segment audio, we can detect new events (such a a new instrument, someone speaking, etc.) much more reliably.

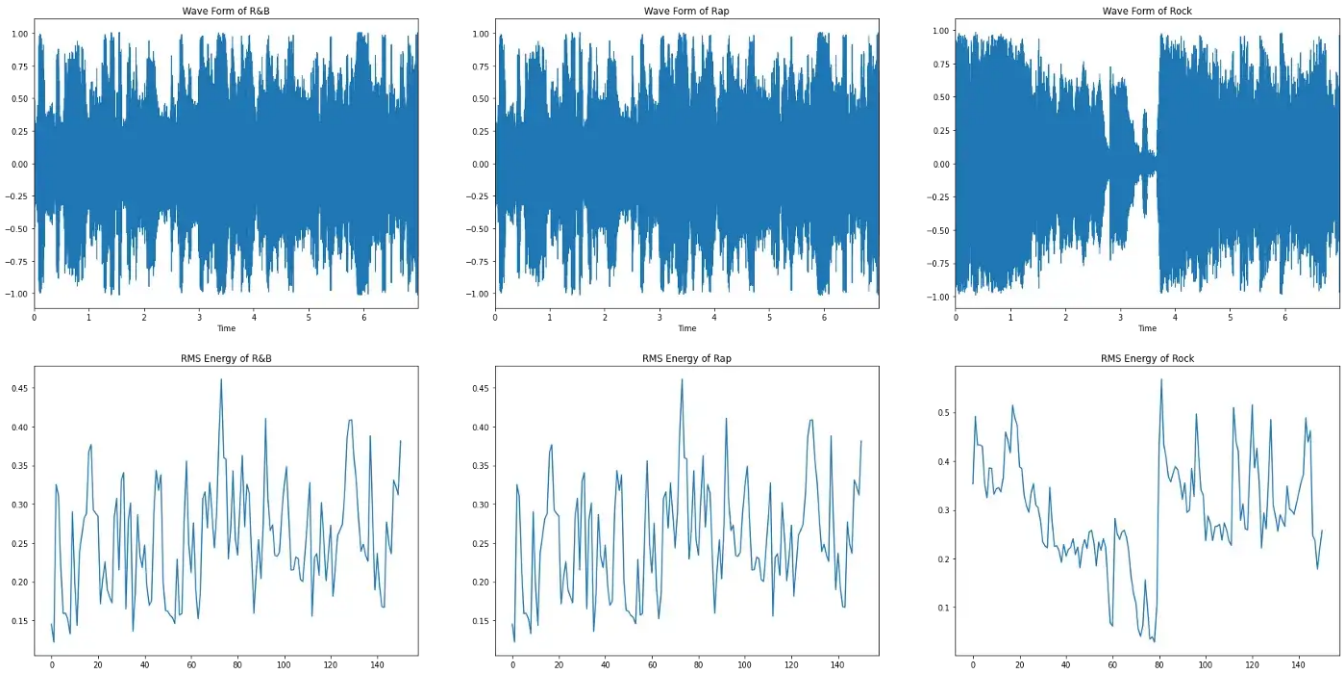
The formal definition of RMS Energy:

$$\mathrm{RMS}_t = \sqrt{\frac{1}{K} \cdot \sum_{k=t \cdot K}^{(t+1)\cdot(K-1)} s(k)^2}$$

If you are familiar with the concept of the Root Mean Square, this will not be too new to you. However, if you are not, do not worry.

As we window across our wave form, we square the amplitudes within the window and sum them up. Once that is complete, we will divide by the frame length, take the square root, and that will be the RMS energy of that window.

To extract the RMS, we can simply use `librosa.feature.rms`. Now, we visualize it:


Wave Form of R&B
Wave Form of Rap
Wave Form of Rock
Time
RMS Energy of R&B
RMS Energy of Rap
RMS Energy of Rock

As we can see the differences from RMS and AE, the RMS does not fluctuate as drastically as the AE. This property is what makes the RMS of the amplitude much more robust to outliers.

Zero-Crossing Rate

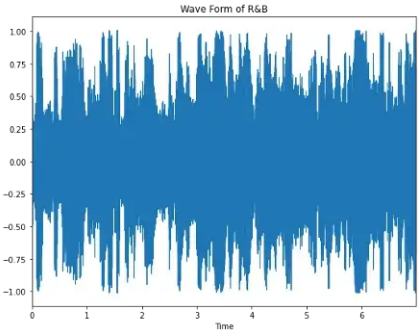
The Zero-Crossing Rate (ZCR) aims to study the the rate in which a signal's amplitude changes sign within each frame. Compared to the previous two features, this one is quite simple to extract.

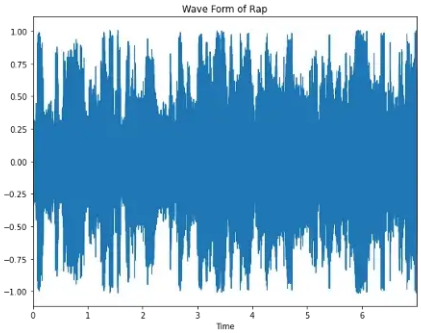
The formal definition of ZCR is the following:

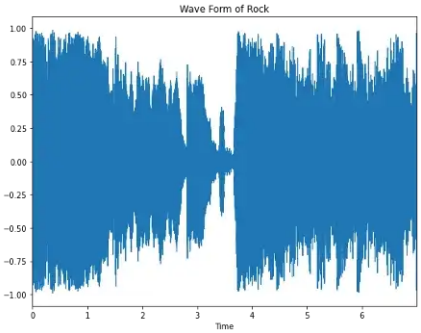
$$\mathrm{ZCR}_t = \frac{1}{2} \cdot \sum_{k=t \cdot K}^{(t+1) \cdot (K-1)} \left| \mathrm{sgn}(s(k)) - \mathrm{sgn}(s(k+1)) \right|$$

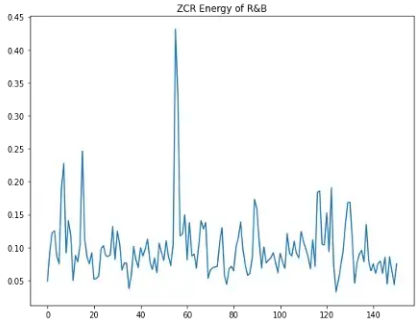
For MIR, this feature is relevant for identifying percussion sound as they often have fluctuating signals that can ZCR can detect quite well as well as pitch detection. However, this feature is generally used as a feature in speech recognition for voice activity detection.

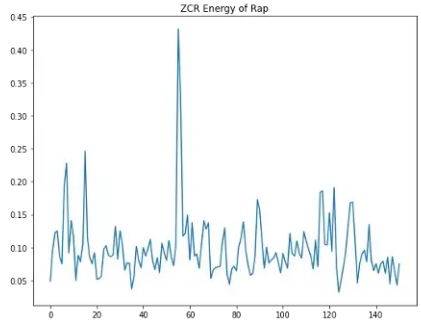
Using `librosa`, we can extract the ZCR using `librosa.feature.zero_crossing_rate`.

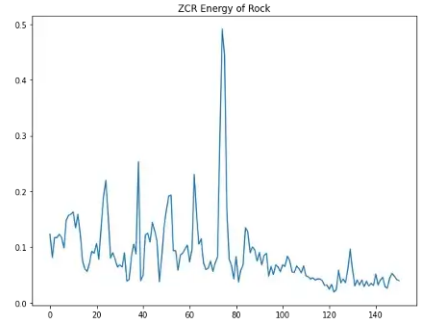

Wave Form of R&B
Time


Wave Form of Rap
Time


Wave Form of Rock
Time


ZCR Energy of R&B


ZCR Energy of Rap


ZCR Energy of Rock

Conclusion

By now, you should have an idea of how time feature extraction works, how it can be utilized in various audio based applications, and how to develop the feature extraction methods yourself. By leveraging the amplitude within specific windows, we open up numerous insights into various applications in MIR and ASR. Thank you for reading, and if you have any questions feel free to ask!

Machine Learning Data Science Digital Signal Processing Artificial Intelligence

Tutorial

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. Take a look.

Emails will be sent to shivakmuddam25@gmail.com. Not you?

Get this newsletter