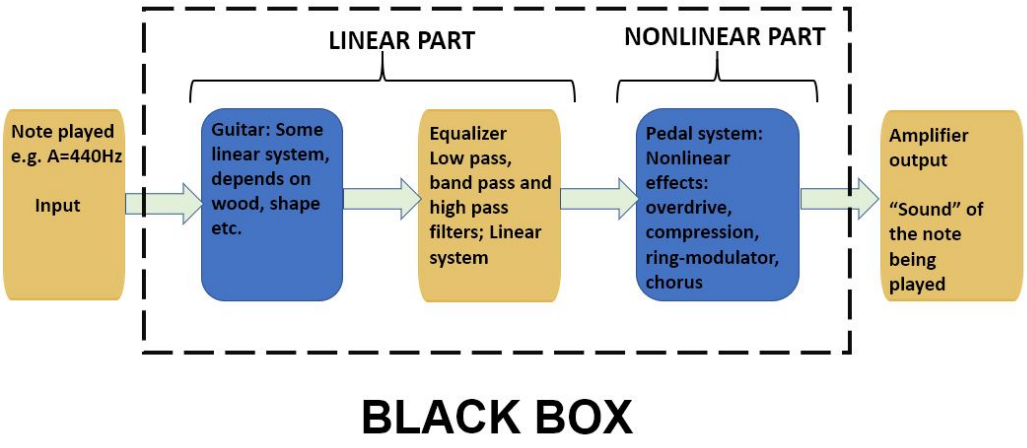
Musical Timbre Transform using Hammerstein Kernels

Saurabh Vaishampayan: EP17B028

June 2020

1 Introduction

1. I wish to propose an ML based program to learn the characteristic sound of an instrument from a sample of a song.
2. For example when Eddie van Halen plays a piece on his guitar-pedal-processor-amplifier setup it sounds very different to mine played on a cheap setup, even if the notes played are the same.
3. While processing there are nonlinear elements involved as well like clipping/saturation of op-amps etc. The goal is to create a program that takes a small sample of the song as input,identifies the note being played and isolates it, learns both the linear and non-linear transfer function and when I play my guitar and feed it to the program it tries to make it sound like the original guitar.
4. I may play a different scale or a different song entirely, but the program should make my guitar sound like the original.
5. While the (rough) algorithm I propose is designed with an electric guitar in mind, one should in principle be able to apply it to any instrument.
6. I may take a small segment of any Pandit Shiv Kumar Sharma's santoor performance and try to play a song on piano and the program should make it sound that it that it was played on his own santoor.



We describe the aspects of the individual sections of the algorithm below.A flowchart summary of the algorithm is given at the end

2 Note identification

1. In the standard tuning, A = 440Hz. All of the other notes can be derived by

$$\boxed{f_r = (2^{\frac{r}{12}})(440Hz)} \tag{1}$$

   where r is any integer.

2. We take a moving window of the input and calculate the FFT for each window. One determines the peak frequency of each FFT and then tries to fit it to the nearest note given by equation (1)

3. At the end of this we now know the note played as a function of time which we feed as input to the next step. This step is essential as we want to establish a relationship between the sound of the note and the note and one can do a supervised ML with this. Note that we also have to extract the amplitude and the phase of the note, along with the frequency of the note.

2.1 Algorithms

We need to identify and implement algorithms that identify notes in as minimum time as possible. There are two important facts we know: First, that musical notes come in a countable set and have very specific frequencies. I can sacrifice frequency resolution for speed as long as frequency resolution is not so bad as to make wrong identification. Also adjacent notes have constant frequency ratio, meaning that fractional error stays same(higher resolution for base notes needed). One could perform binary search like algorithms to first roughly determine the frequency range and then adjust window size accordingly.

3 Nonlinearity

We assume that the signal has zero DC component(valid assumption as the input is an oscillation)

1. In the electric guitar setup, the guitar, along with treble, base knobs act as a linear system.

2. The amplifier/pedal adds nonlinear effects like overdrive/distortion, compression, chorus, ring modulator etc.

3. Modelling a nonlinear effect by Taylor expansion:
   $f(x) = h_1 x + h_2 x^2 + h_3 x^3 ....$

4. This can be generalised to the following:
   $y(t) = f(x(t)) = h_1(t) * x(t) + h_2(t) * x^2(t) + h_3(t) * x^3(t) + ...$, where $*$ represents convolution

$$\boxed{y(t) = f(x(t)) = \sum_{n=1}^{M} h_n(t) * x^n(t)} \tag{2}$$

where we are truncating the nonlinearity upto order Q. For better approximations, we can take larger values of Q.
Our goal is to determine the coefficients of FIR filters $\{a_n(\mathrm{t})\}_{n=1}^{M}$.
I later found that this is called the Hammerstein kernel, which is the diagonal form of the Volterra Kernel.

4 Learning

We consider x[n] to be a sum of harmonics with time varying amplitudes and phases given by: ¡center¿$x[n] = \sum_{k=1}^{M} A_k[n] cos(2\pi kn \frac{f_0}{f_s} + \phi_k[n])$¡/center¿ $A_k[n]$ and $\phi_k[n]$ are baseband signals(harmonic envelope amplitude and phase). Different relative phases in harmonics is perceptually indistinct to the human ear, so we can maybe account for this by putting a gauge degree of freedom(ie not caring for this in fitting) later.
Our goal is to find $h_k[n]$ for all k given an input output sample.
One way to do this is the following:

1. Split the input and output envelope and phase data into slices. The slice width should be larger than the filter length but much smaller than 1/BW of envelope. Like we did in the polynomial fitting case, make a training dataset from this.

2. Write the Hammerstein Kernel in Frequency domain. This looks like:

$$\boxed{Y(f) = \sum_{k=1}^{Q} H_k(f)[X^{(k)}(f)]} \tag{3}$$

   Here $X^{(k)}(f)$ is convolution of $X(f)$ with itself k times.

Since the signals only have peaks at the Harmonics, we can write $X(f)$ in compact form. We just need to convolve the vector $[X(-Mf_0), X(-(M-1)f_0), ...X(0), ...X(Mf_0)]^T$ k times, for each of the input slices(during which amplitude and phase can be assumbed to be constant).

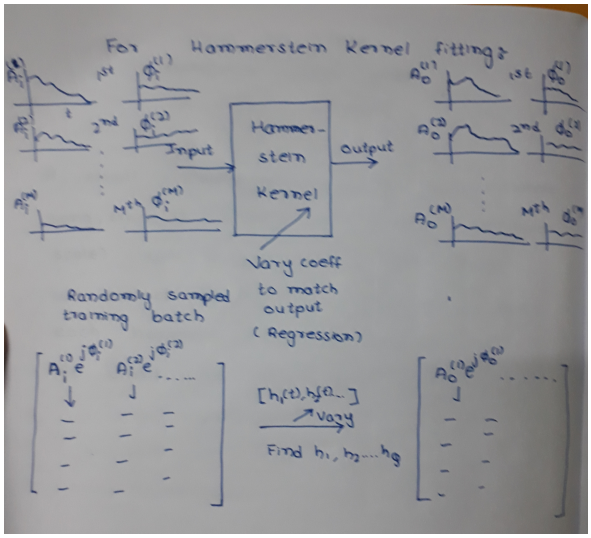
We can write the Hammerstein equation, for each of the slices, as the following:

$$\boxed{\begin{pmatrix} Y(-Mf_0) \\ . \\ . \\ Y(0) \\ . \\ . \\ Y(Mf_0) \end{pmatrix} = \begin{pmatrix} H_k(-Mf_0) \\ . \\ . \\ H_k(0) \\ . \\ . \\ H_k(Mf_0) \end{pmatrix} \circ \begin{pmatrix} X^{(k)}(-Mf_0) \\ . \\ . \\ X^{(k)}(0) \\ . \\ . \\ X^{(k)}(Mf_0) \end{pmatrix}} \tag{4}$$

Here $\circ$ indicates element wise product(Hadamard product) and repeated index k indicates summation over k from 1 to Q. Also since $x^k[n]$ will generate frequencies upto MQf_0, we need to ensure that higher harmonic amplitudes become small. Also since Y X and H are real, conjugate symmetry holds so we only need to care about half of the rows.
Writing the equation for one row ie one particular frequency, for each of the slices, it becomes:

$$Y(nf_0) = H_1(nf_0)X(nf_0) + H_2(nf_0)X^{(2)}(nf_0) + ...H_Q(nf_0)X^{(Q)}(nf_0)$$

We have amplitude and phase data for many slices ie we know $X(nf_0)$ and $Y(nf_0)$. We can determine $H_k(nf_0)$ by linear fitting over this large collection of data. Graphically,



4.1 Possible directions:

A guitar has around 48 playable notes. But we want to learn from a song sample. Any song is often in one particular scale, which itself rules out 5 frequencies per scale multiplied by four octaves which gives atleast 20 notes missed(unless the sample is some exotic music). Also the sample will usually contain notes in only one or at max 2 octaves generally, so we will only have access to about 10-12 notes out of 48 notes and our task is to figure out the "sound" of all notes of the guitar out of this very sparse data.

1. The simplest way is possibly a polynomial spline interpolation for magnitude and phase response of each of the filters.

2. Another attractive way is to use Deep Neural Networks to **find the spectrum for the missing frequencies.** The task is analogous to completing an image, with knowledge of only some pixels and rest are missing. Then one figures out the filter coefficients(we also want to minimise the length of these FIR sequences). For example, one train the network by blocking off the sound data of multiple notes and try to deduce it given the sound of few notes. **This can be done by using autoencoders** trained on complete batches of sound envelope data for each note of a particular guitar. Then when we are given sound envelopes for few notes, we could ask decoder to produce envelopes for other unplayed notes

5 Conclusions

Since Hammerstein kernels have the form that naturally produces overtones, by using this model we can understand better the mechanisms and characteristics of that particular instrument. One could also extend this to speech. This method could be potentially used for timbre transform in case of all musical instruments like piano to guitar, voice to cello etc.