Week 3: Bachelor's Thesis

Saurabh Vaishampayan

November 2020

1 Introduction to the problem and challenges

1. Our problem is as follows: Given a sound sample for a note played in an instrument A, find a way to transform sound of note on instrument B to that corresponding to A, preferably in real time. We want the system to be time invariant, if the input is delayed, the output should be too.

2. The perceived sound of a note on an instrument depends on two factors: the distribution of harmonic amplitudes and the note envelope(attack, decay, sustain, release).

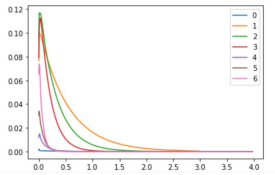
3. First we focus on the harmonic amplitudes aspect

4. To transform the sound from one instrument to other, one has to come up with a function that allows for enough parameters and **"choreograph"** the harmonic amplitudes of input to match the output.

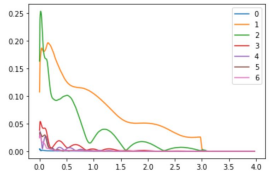
5. We aim to do this by using the diagonal Volterra model, given below. Q is order of nonlinearity and $h_k[n]$ are FIR filters

$$y[n] = \sum_{k=1}^{Q} h_k[n] * x^k[n]$$

6. Now, coming to the envelope part. In the figure below you can see envelope profiles for two different instruments for the same note,



(a) Acoustic guitar



(b) Electric guitar

Figure 1: Envelopes comparison for electric and acoustic guitars

7. These instruments are played in different settings, there is no reason to expect to find a diagonal Volterra model that directly produces the transformation.

8. **There is an urgent need to "cast" the sound of target in the envelope of the input, and then proceed with find the Diagonal Volterra Model coefficients for this intermediate "cast".** These coefficients, once learnt can be applied to an input in real time, since simple signal convolution is pretty fast.

2 Stages involved in the solution

1. Find the time invariant features of the target. Find the time varying features of the target. These should be found in such a way that reconstructing the target audio using these should be possible with a high fidelity. Mathematically, this can be stated as:

$$\mathbf{b} = f(\mathbf{A}(\mathbf{t})) \tag{1}$$

$$\mathbf{e}(\mathbf{t}) = g(\mathbf{A}(\mathbf{t})) \tag{2}$$

$$\mathbf{A}(\mathbf{t}) = h(\mathbf{b}, \mathbf{e}(\mathbf{t})) \tag{3}$$

   Here $\mathbf{A}(\mathbf{t}), \mathbf{b}, \mathbf{e}(\mathbf{t})$ are the harmonic envelope amplitudes, time invariant features, time varying features respectively.

2. Do the same for the input.

3. Now use the time varying features of the input and the time invariant features of the target to make an intermediate "cast" audio.

4. Now find the coefficients for the diagonal Volterra model. We will use this for real time timber transfer later. Note that our transformation is limited to the intermediate cast only, one cannot achieve the same attack, delay, sustain, release features as the original, if one hopes to do this in real time with a small delay.

5. We summarise this in a figure below:

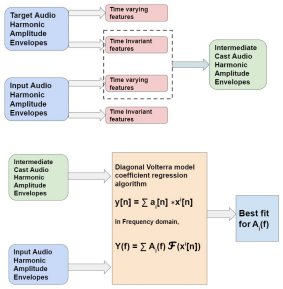


Figure 2: Algorithm for identification of diagonal Volterra coefficients

3 A preliminary heuristic algorithm for extracting and separating time invariant features, and time varying features

The goals for this section are as follows: Given an audio segment of a musical note, find a way to extract time varying and time invariant features of the audio. Also find a function that is able to reconstruct the original audio with high fidelity. Mathematically, we repeat the equations 1 to 3 here for convenience of reading:

$$\mathbf{b} = f(\mathbf{A}(\mathbf{t}))$$

$$\mathbf{e(t)} = g(\mathbf{A(t)})$$
$$\mathbf{A(t)} = h(\mathbf{b}, \mathbf{e(t)})$$

We have tried a method before for this, which was based on learning a polynomial regression from fundamental harmonic envelopes to higher harmonic envelopes. The results were insightful, but audio generated was not very good perceptually.
We now propose a new heuristic method, inspired from PCA/SVD to prepare this "cast" of the audio, and one can see from audio samples given that perceptually, it is much closer than the polynomial regression method.

3.1 The algorithm

The algorithm we use here is insipired from an interpretation of Principal Component Analysis and Singular Value Decomposition.
Assume we have an $N \times M$ data matrix, with the rows representing time and columns the index of the harmonic.
i.e.

$$\mathbf{A(t)} = A_{N\times M}; A^{(m)}(t_n) = A_{nm}$$

Here n represents the time instant t_n, m represents harmonic number, N is the total duration of audio and M is total number of harmonics in consideration.
The matrix A has a singular value decomposition given by:

$$\mathbf{A} = \mathbf{U\Sigma V^H} = \mathbf{u_1}\sigma_1\mathbf{v_1^H} + \mathbf{u_2}\sigma_2\mathbf{v_2^H} + ...$$

Each of the $\mathbf{u_i}, \mathbf{v_i^H}$ are $N \times 1$ and $1 \times M$ matrices respectively.
$\sigma_\mathbf{i}\mathbf{v_i^H}$ are the principal axes for the data, and are the time invariant features, while $\mathbf{u_i}$, which are of shape $N \times 1$ are the time dependent features. Combining these two perfectly reconstructs the original data, if one considers all the principal values.
If one only considers the first principal component, we get the following: all harmonic amplitudes as a function of time are proportional to some vector(v_i^H). This usually does not always capture the full sound, because harmonic amplitudes may not rise and fall together, infact, different peak times give a sharp attack effect. Considering only one gives a "horn-like" effect.
We show plots of an example sound data, with original and reconstruction using some and all of the principal values.

Now what we propose is the following, for preparing the "cast" ie constructing an audio with time dependent features of the input and time independent features of the target.

- Perform SVD for both the target and input.

$$\mathbf{A^{target}(t)} = A_{N\times M}^{target} \tag{4}$$
$$\mathbf{A^{input}(t)} = A_{N\times M}^{input} \tag{5}$$

- Construct the "cast" using $\mathbf{U_{input}}$ and $\mathbf{\Sigma_{target}V_{target}^H}$ as follows:

$$\mathbf{A^{cast}(t)} = \mathbf{U_{input}\Sigma_{target}V_{target}^H} \tag{6}$$

- Now we have harmonic envelopes for the cast. Multiply each by a sinusoid of frequency nf_0 and add to get the audio.

We have provided some audio samples for the above transformation on different instruments for a fixed musical note.

3.2 Possible improvements

Are there any other better ways to do this sequence of extraction of time variant and invariant features of the data and reconstruction?

One could use autoencoders to perform this same task, in hopefully a much better way. **Google Brain's Magenta group has a project called DDSP where they have achieved similar functionality using autoencoders, so this should be possible.** Autoencoders are essentially a cascade of an encoder neural network which learns the hidden representation of the data and a decoder which tries to reconstruct the original from this hidden representation. The problem is that we might require a lot of data. While we technically have access to hundreds of thousands of points(each point is a vector that represents value of amplitude for a particular harmonic for a particular instant of time), but these amplitudes are baseband signals, and vary very slowly. How to train this autoencoder to extract time varying and time invariant features also requires thought, and I require some time and help from a Machine Learning specialist for this.

4 Finding the diagonal Volterra Kernel coefficients

- We write the equation for diagonal Volterra coefficients as follows:

$$y[n] = \sum_{k=1}^{Q} h_k[n] * x^k[n]$$

- If $x[n]$ is a sinuoid, its powers will give harmonics of the sinusoid. If $x[n]$ is a sum of harmonics,it provides a way to **"choreograph"** the different harmonic amplitudes so that we get the features of the target**(the distribution of harmonic amplitude envelope are indicative of the time independent features of the instrument)**.

- However there is a major observation one needs to point out. The diagonal Volterra model that deals with nonlinearity will work well only if the "cast" is generated using some nonlinear function. The SVD inspired algorithm we have used is a linear function of the harmonic amplitude envelope(mixing is allowed). So there is this tension already between the two methods, and this may result in a suboptimal performance. If one uses some nonlinear transformation to construct the intermediate "cast", then the performance may be better.

We are interested in doing the transformation in real time using short FIR filters. Keeping this in mind, we propose a heuristic algorithm for identifying the filter coefficients, from a frequency domain analysis

- Extract the harmonic amplitude envelopes for all $x^q[n]; q \in Q$, where Q is the order of nonlinearity.

$$x[n] = \sum_{k=1}^{M} A_k[n] cos(2\pi kn \frac{f_0}{f_s} + \phi_k[n]) \tag{7}$$

  Here M is the order of the harmonics we are interested upto. $A_k[n], \phi_k[n]$ are baseband signals.

- Choose a slice width, greater than FIR size and smaller than bandwidth of the envelope profiles.

- FIR filter length is much smaller than harmonic envelopes, so it will appear as if instantaneous.

- Randomly choose time slice instants and separate into train and validation dataset.

- The diagonal volterra model written for one particular frequency is given as:

$$Y(nf_0) = H_1(nf_0)X(nf_0) + H_2(nf_0)X^{(2)}(nf_0) + ...H_Q(nf_0)X^{(Q)}(nf_0) \tag{8}$$

We have access to multiple values for $X(nf_0)$, so this really becomes a problem of complex linear least squares fitting.

- Humans cannot perceive phase differences between harmonics, so we don't really care about global phase of the harmonic at an instant, just its absolute value. This is a gauge degree of freedom which loosens the restrictions on $H_1(nf_0), H_2(nf_0)$ etc upto a global phase, achieving better accuracy in least squares fitting.

- Writing all this mathematically, one has:

$$\begin{pmatrix} Y_1(nf_0) \\ Y_2(nf_0) \\ Y_3(nf_0) \\ . \\ . \\ Y_T(nf_0) \end{pmatrix} = \begin{pmatrix} e^{j\phi_1} \\ e^{j\phi_2} \\ e^{j\phi_3} \\ . \\ . \\ e^{j\phi_T} \end{pmatrix} \circ \begin{pmatrix} X_1^{(1)}(nf_0) & X_1^{(2)}(nf_0) & X_1^{(3)}(nf_0) & . & . & X_1^{(M)}(nf_0) \\ X_2^{(1)}(nf_0) & X_2^{(2)}(nf_0) & X_2^{(3)}(nf_0) & . & . & X_2^{(M)}(nf_0) \\ X_3^{(1)}(nf_0) & X_3^{(2)}(nf_0) & X_3^{(3)}(nf_0) & . & . & X_3^{(M)}(nf_0) \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ X_T^{(1)}(nf_0) & X_T^{(2)}(nf_0) & X_T^{(3)}(nf_0) & . & . & X_T^{(M)}(nf_0) \end{pmatrix} \begin{pmatrix} H_1(nf_0) \\ H_2(nf_0) \\ H_3(nf_0) \\ . \\ . \\ H_M(nf_0) \end{pmatrix} \tag{9}$$

- Here each of ϕ_i are gauge degree of freedom, since we do not care about global phase and are chosen so that minimum error is acheived.

- Writing in compact form this becomes:

$$Y = UXH \tag{10}$$

Here $U = diag(e^{j\phi_i}), i \in T$

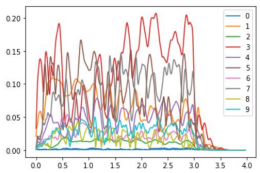
- Using least squares and writing gradient, and adding a regularisation term, we get

$$\nabla_H = (X^H X + \lambda I) - X^H U^H Y \tag{11}$$

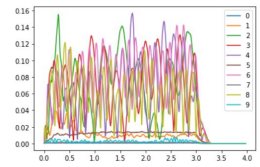
$$\nabla_\phi = imag(Y^* \circ (UXH)) \tag{12}$$

5 Observations

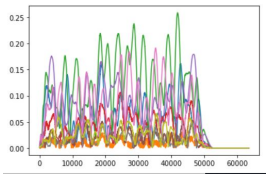
Transformation from organ to reed for note G4 with pitch 391 Hz
Hyperparameters that give least error on validation dataset:
Order of nonlinearity: 5
Regularisation parameter: 1e-7
Error(fraction): 0.45728904502699813
i.e. around 55 percent test accuracy.
Plots:



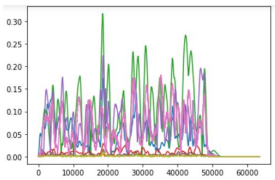
(a) G note on reed(instrument)



(b) G note on organ



(c) Intermediate cast from reed to organ



(d) Transformed output using Volterra from organ to reed sound

Figure 3: Harmonic envelopes for reed, organ, intermediate transform, and transform using Volterra

6 Conclusion and future improvements

As mentioned before there are two components of the algorithm: First is identification of time variant and invariant features of input and target audio and preparing an intermediate array of the target in the cast of the input. Then using this to perform regression of the Diagonal Volterra model.
The method we have used is inspired from SVD/PCA and is hence linear. However it is linear in the sense of harmonic amplitudes. Mixing of envelopes of different harmonics is allowed(which is forbidden in LTI systems)**The tension between this linearity and the nonlinearity of the diagonal Volterra model is responesible for suboptimal performace**
Stating this mathematically, we have:

$$\mathbf{A_{cast}} = \mathbf{U_{input}}\mathbf{\Sigma_{target}}\mathbf{V^{H}_{target}}$$

Rearranging this in a different form we get:

$$\mathbf{A_{cast}} = \mathbf{A_{input}}\mathbf{V_{input}}\mathbf{\Sigma^{-1}_{input}}\mathbf{\Sigma_{target}}\mathbf{V_{target}} = \mathbf{A_{input}}\mathbf{W}$$

$$\mathbf{W} = \mathbf{V_{input}}\mathbf{\Sigma^{-1}_{input}}\mathbf{\Sigma_{target}}\mathbf{V_{target}} \tag{13}$$

Only if W is diagonal, then the transformation can be achieved by LTI system, else we get mixing of harmonics.
So we need to look at other nonlinear systems like autoencoders to achieve the transformation