Towards a Generative Electronica

A PROGRESS REPORT

by Arne Eigenfeldt

This paper was originally presented at TES 11, and presents the state of research at that time by our research group. The Generative Electronica Research Project (GERP) is a combination of scientists – researchers in artificial intelligence, cognitive science, and machine-learning – and creative artists whose aim is to generate stylistically valid Electronic Dance Music (EDM) using human-informed machine-learning. More recent research can be found at www.metacreation.net.

Introduction

What do we mean by generative electronica? I'd like to unpack the ideas, and discuss them individually.

Generative Electronica

One definition of generative music is a system that has no inputs, and is thus not transformational. Another is that it is simply a process, designed by the composer, and set in motion. Terry Riley's *In C* is a classic example of this. Both cases focus upon the fact that each run of the system will produce different results. The notion of varied output suggests a need for random procedures within the system. For example, rather than determining that a particular melody should play at a certain point, an algorithm could be written that chooses notes from a particular pitch class set, a group of rhythms, such that they follow a particular melodic shape, and initiating this algorithm at a particular point in time. The randomness within the algorithm guarantees that every generation should be different; however, the constraints placed on the random selection attempts to guarantee a certain musical success.

One difficulty with such systems is evaluating their success – you can listen to the music, and judge whether you consider it to be successful (whatever that means). We must assume, however, that the producer of the system has accepted it's output as "artistically worthy", and thus representative of his/her ideas. If you, as the listener, don't like the results, are you judging the composer's aesthetic, or the system's ability (or even the composer's ability)? See (Eigenfeldt et al. 2012) for a more detailed discussion.

Validation

In scientific terms (and I find it extremely odd for me to be discussing the scientific point of view), the objective description of any system's success is called validation. How can one validate an artistic production system, separately from the art it produces?

This is an extremely difficult topic, one of the grand challenges of generative art, and the topic of many papers at Generative Art conferences. Generating music within a specific, recognized style, and judging the generated results in relation to that style, is one method that has been used. If analysis can determine tendencies within the music (what we may consider "style"), then these same rules can be used to generate a probabilistic representation of that music.

The most famous of these experiments was by David Cope, who used such a system to write music "in the style of" composers such as Bach, Mozart, Scott Joplin, and himself.

Eigenfeldt_audio01.mp3

audio 1 (:48). David Cope, *Invention 1* (1993) Using his software, EMI, Cope generates new versions of Bach based upon analysis of the original corpus.

Generative Dance Music

One tradition that has evolved over the last decade at computer music conferences is the late night concert. Recognizing that the tools used by academics and popular musicians are not very different, the late night concert allows participants to present their non-academic music in an informal nightclub atmosphere. Many of these performances involve music that relies heavily upon corporeal rhythm - in other words, beats. And all of the systems that I see tend to be generative - no one is using Ableton Live and simply mixing and adding effects. This is, I presume, an effort to approximate improvisational performances.

But creating a generative work that is *influenced* by dance music, and one that is *stylistically representative* of it are two different things. Of course, we can play the "artist" card, and say that we are not actually interested in creating commercially viable dance music - that would be selling out.

But can it actually be done?

Although generative techniques have been used in styles such as ambient and other more experimental areas, they have not been used in genres that are very style specific and commercial. Consider that the rules of House music are fairly straight forward, but the subtleties in separating "good" music within the style from pretenders are hard to codify.

This is the case whenever one uses models: anyone that has written a software program to create imitation Schoenberg, Webern, Miles Davis, or Crystal Method, knows that it is possible to create stylistically correct music that is, say, 60-70% accurate; getting beyond 80% is another matter. This is one of the problems of machine learning (a branch of AI that has produced the most tangible and applicable results): each notch above 80% is exponentially difficult to achieve.

The Corpus

In fine-tuning our model, four genres of electronic dance music were chosen – Breaks, Drum and Bass, Dubstep, and House. 100 individual tracks were chosen to represent diverse characteristics and time periods, ranging between 1994-2010, with only four artists being represented twice. The tracks contain many common formal and structural production traits that are typical of each style and period.

For Breaks, tempi range from 120-138 beats per minute (BPM); drum parts are originally derived from sped-up samples of drum breaks in Soul and Funk music. The beat is moderately syncopated, emphasizing two and four.

Eigenfeldt_audio02.mp3

audio 2 (:28). Junkie XL, *Mushroom* (2006) Excerpt from a typical *Breaks* track.

Compare this to Dubstep, which has a faster BPM, but is felt at half time since it emphasizes the third beat. Dubstep also typically has a "wobble bass", a highly modulated synth bass.

Eigenfeldt_audio03.mp3

audio 3 (:31). Evergreen and Landford, *Jah Rain* (2009) Excerpt from a typical *DubStep* track.

Analysis

Machine listening is a new technique in AI that has been very successful at style identification, using signal-processing analysis to look at dozens of features within the audio itself. Features are low level parameters, such as spectral centroid, spectral noise, zero crossing, etc. But knowing that the spectral centroid in Drum and Bass is higher than DubStep doesn't help for generation purposes, because these are such low level descriptors. We needed more concrete data, and chose to approach it from both a high level and low level at the same time.

Beat Transcription

As the beats were one discriminating factor in this music, we began to transcribe the drum patterns. This is not very difficult to do for an expert listener.

Eigenfeldt_fig01.jpg

Figure 1. The transcribed drum beat for *Jah Rain*

If you listen to audio example #2 again, *Jah Rain*, you may note that the beat is fairly consistent throughout the track. It was found that almost all tracks had only between 1-6 unique beat patterns.

Form Analysis

We also looked at how the tracks were put together in terms of individual sections. Almost all tracks consisted of eight bar phrases, which could be grouped into five different sections:

- Fade in (A)
- Intro (B)
- Verse (C)
- Breakdown (D)
- Outro (E)

You can see this track cut up in Ableton Live (see Figure 2), with each vertical dark line representing an 8 bar phrase. As such, this track would be represented formally as:
AAAAAABBBBCCCCDDDDCCCCDDCCEE

Eigenfeldt_fig02.jpg

Figure 2. *Jah Rain* divided into 8-bar phrases. Different sections are colour-coded.

Machine Analysis

At the same time, we attempted to produce our own machine analysis. The results were good enough to get published (Eigenfeldt and Pasquier 2012), but not good enough to use.

For example, one beat transcription determined that there where 31 unique kick patterns, 40 snare patterns, and 20 hi-hat patterns used in a track, whereas human transcription suggested only 1. A 70-80% success rate is considered reasonable in this area; if we spent the next five years on this aspect alone, we might get up to 90%; however, compare this to the 100% accuracy I get from my paid graduate students doing the transcriptions, the direction was clear.

Generation

So now we get to Beat Generation, which is more interesting. Given transcribed data on beats within the corpus, how can you generate beats that are consistent within that style, yet not a simple selection? We want to avoid, for example, simply choosing one of the 25 patterns in the Breaks music dataset. As was just mentioned, the drum pattern is the defining aspect of the music, and there is a finite set of variations that allow a pattern to be classified as acceptable within that genre.

Beat Generation

For this reason, I created a probability matrix from the corpus, using the Breaks database for this example. Beat patterns are separated into kick, snare, and hi-hat. Dividing the measure into 16ths, each onset at each location within the beat is counted.

What is shown is the relative number of times an onset occurred at that location within the bar (time is left to right), using only the Breaks database.

Eigenfeldt_fig03.jpg

Figure 3. Histogram for onsets: kick, snare, hi-hat.

This histogram can easily be translated into probabilities. Thus, in the kick, there is a 100% chance that it will occur on the downbeat; in the snare, there is a 100% chance that it will occur on beats two and four. Generation is simply a matter of generating a random value between 0 and 100 for each temporal location: if it is lower than the probability, an onset is generated.

Eigenfeldt_video01.mov

Video 1 (:32). Screengrab demonstrating beat generation from probability matrixes.

Self-organising Map for Drum Timbres

Auditioning your beats through various drum patches can be time consuming – what did *Afterhours* sound like again? We're using a Self-organizing Map to correlate timbres of drum patches. This process is explained thoroughly in (Eigenfeldt and Pasquier 2010). Basically, all drum kit patches were analyzed for a variety of spectral features, including:

- 24 Bark bands,
- 24 Mel Coefficients,
- Spectral Flux,
- Spectral Centroid,
- Spectral Roll-off (frequency at which 95% of spectrum is below),
- Skewness (asymmetry of spectrum),
- Spectral Kurtosis (flatness),
- Spectral Spread,
- Decrease in Spectral Magnitude

A self organising map is a special type of neural network in which the trained data organizes itself so that similar entries are move close to one another on a 2D map. The end result allows users to click through an organised representation of the data space, with similar data points being close together. In our case, the user can select overall similarity, or individual instrument similarity.

Eigenfeldt_video02.mov

Video 2 (:34). Using a self-organising map (SOM) to navigate the timbral space of drums.

Auxiliary Percussion Generation

The drum pattern generation is limited to kick, snare, and hi-hat. It was found that this combination (or functionally related parts) was present in virtually all the tracks, in all the styles. However, it was also found that most tracks contained additional percussion parts, from 1 to 3 more. I posited that the structural importance of these parts was less concerned with their placement within the bar than in their relationship to the other parts. For example, a second kick (sometimes called a ghost kick) existed that interacted with the regular kick in complex ways.

Eigenfeldt_fig04.jpg

Figure 4. Typical kick and ghost-kick interrelationship.

Therefore, I created a genetic algorithm to evolve patterns that could co-exist.

It isn't possible to fully describe this process here, but it should be noted that this takes the system out of realtime. I'll talk a bit about genetic algorithms later in discussing form generation.

Eigenfeldt_video03.mov

Video 3 (:08). Screengrab demonstrating auxiliary percussion generation.

Displayed is the screen for percussion generation, with the result of the GA at the bottom (red squares are on the beat). Note that the user has some influence over results.

The next video demonstrates the interrelationship between the two parts.

Eigenfeldt_video04.mov

Video 4 (:32). Combining the drum pattern with the generated percussion pattern.

Bassline Generation

Bass line generation followed a similar strategy to beat pattern, in that a probability matrix was created based upon the corpus. In this case, time (onset location) is top to bottom - the first 16th at the top, the last at the bottom. Individual pitch classes occur left to right.

Thus, in the first 16th, there is a 100% possibility that an onset occurs, and that onset has a high percentile of being PC 0 (the root), and a very low chance of being PC 7 (the 5th) - note that percentile can total more than 100.

As with drum patterns, this type of generation is extremely fast, and many possibilities can be auditioned.

Lastly, the (Breaks) corpus is displayed to the left, and every track is included in the database. Users can select which tracks are included, and thus influence the generation.

Eigenfeldt_video05.mov

Video 5 (:26). Generating basslines using a probability matrix for both onsets and pitches.

Form Generation

The system is able to generate an overall formal structure, which is, naturally, derived from the analysis. Given a user specified length (in terms of number of sections) and phrases (8 bar units within each section), the system randomly generates 100 combinations of the phrase descriptors, called individuals (i.e. one individual might be CBABCCADA...). Next, a function is run to rate how close each individual is to the overall requested number of sections (which you could consider a fitness function), and another compares the combinations to a 1st order Markov probability table of phrase relationships. The highest ranked individual is presented to the user.

Eigenfeldt_fig05.jpg

Figure 5. Generating overall forms.

In this case, the result produced 24 phrases, as requested, but only 6 sections.

Given a form, it was necessary to discover the relationships of specific parts between phrases and sections. For example, should the drums be in a given section? If the drums don't come in at the beginning, when should they come in? Can a part drop out of a section before it is over?

Analysis was done that mapped these relationships - this is an example of a Google document that my research team uses. It simply uses "switches" to determine whether any of the 16 parts is active in a phrase.

Eigenfeldt_fig06.jpg

Figure 6. States (on or off switches) for instruments in a track, by phrase.

This could now produce probability tables that answered all my questions, both the "horizontal" probabilities (which parts are active in a given section) as well as the "vertical" probabilities (how parts change over time).

However, generating a "part score" produced some very illogical results. For example, phrases in which no part played (which is entirely possible if each part is less than 100% active in ever section) – my model did not take into account the relationships between parts.

Philippe Pasquier, my collaborator whose specialty is artificial intelligence, suggested I needed to use a grammar, or associative rule learning, to accomplish this. Unfortunately, this is something that I've never done, and had no idea how to go about doing.

Right at this time, I attended the evolutionary art conference, EvoMusArt, and saw a lecture by Craig Reynolds (the creator of the flocking algorithm used in almost every film since Jurassic Park), a talk which showed a very pragmatic use of genetic algorithms to solve problems, the results of which are known, but the methods used to get there, are not. This is exactly what I needed, and I was amazed to see that in usually less than 15 generations of an initial population of 25 individuals (initially generated using the above described probabilistic system), successful candidates emerged.

The trick was finding the right fitness functions, which amounted to about ten different functions that were generated after looking at outputs of the system. For example, any individual with a completely empty phrase gets a penalty point; any individual where the drum comes in at A, but doesn't continue in B, gets a penalty point.

Each individual is rated, the bad ones culled, and the good ones mated to replace them (mating involves randomly combining sections). The genetic algorithm (which takes about 90 seconds to run) generates scores that look very much like the transcribed part score analysis.

Eigenfeldt_fig07.jpg

Figure 7. A generated score.

In this example, not all parts are present (no midfreq percussion, no breaks, no 2nd synth lead, etc.) Part generation is only required for those parts in the generated score. Here is an excerpt of this particular score.

Eigenfeldt_audio04.aiff

audio 3 (1:45). Example generation from system.

There are two obvious omissions to this generated music: fills, and signal processing. Both of these are yet to be tackled, but both will be based upon similar corpus-based generative methods.

Future Directions

I've already mentioned that certain generative elements need to be improved - foreground SFX, pad (which include all potential harmonic information). Also, fill generation, and signal processing.

When these are complete (hopefully within the next six monthes), we plan to tackle validation. We have a concert scheduled for the winter, in which several generated tracks will be presented, together with tracks composed by my students using the same tools (Reason / Live), as well as a few tracks from the corpus.

Audience members will not be told which ones are which, and will be asked to rate the music. Some of the audience will be aware that some of the music was machine composed, others will not. Voting will occur via a website, which audience members will connect to via their phones. We hope to determine whether audiences can, in fact, tell the difference between machine composed music, and human composed.

Hopefully next year, I'll be able to play some generated music, together with the validation results

Bibliography

Eigenfeldt, Arne and Philippe Pasquier. “Realtime Timbral Organisation: Selecting Samples Based Upon Similarity.” Organized Sound, vol 15 issue 2, 2010.

Eigenfeldt, Arne, Philippe Pasquier and Adam Burnett. “Evaluating Musical Metacreation.” International Conference of Computation Creativity, Dublin, 2012.

Eigenfeldt, Arne and Philippe Pasquier. Towards a Generative Electronica: Human-Informed Machine Transcription and Analysis in MaxMSP. Proceedings of Sound and Music Computing Conference, Padua, 2011.

Biography

Arne Eigenfeldt is a composer of live electroacoustic music, and a researcher into intelligent realtime music systems. His music has been performed around the world, and his collaborations range from Persian Tar masters to contemporary dance companies to musical robots. His research has been presented at conferences such as ICMC, NIME, SEAMUS, ISMIR, EMS, and SMC. He is an associate professor of music and technology at Simon Fraser University, Canada, and is the co-director of the MetaCreation research group (metacreation.net), which aims to endow computers with creative behaviour.

http://www.sfu.ca/~eigenfel

http://metacreation.net