Gray Davidson Project 5 Summary

Project Design

Melody Ex Machina is an exploratory work in computer music generation using dep learning. The project proposed to seek a novel dataset of midi files from the internet, use these to train a neural network and in turn, generate music with the newly trained network. The domain specific aspects of the project were only one of several goals. The complete set is listed here:

a) to understand and replicate the process of generating music with deep learning from end to end,

b) to explore Keras, a tool I had not used before, experimenting with various inputs, outputs and layers,

c) to build on the work I've done in prior projects to create a concise, clear, impactful presentation.

d) to produce a well-organized, easily maintained project repository on Git-Hub.

I was largely successful in most of these, but mention must be made of the incredible work of Sigurður Skúli Sigurgeirsson (https://github.com/Skuldur) which springboarded my own investigation. In particular, my work relied heavily on Sigurður's use of the MIT-based Python library Music21 and this reliance was responsible for much of the ease with which I manipulated midi files in this project.

A final aspect of the project design which was partially realized in the work, though not in the presentation, was to explore the stranger melodies produced by the network and to manipulate these with audio editing softwares such as Ableton. Because this exploration was limited to melodic creativity the timbre of the resultant melodies is an open area for future work using projects like Magenta's NSynth (https://magenta.tensorflow.org/nsynth).

Tools

I chose to use Musical Instrument Digital Interface (MIDI) files for this project because they carry musical information (pitch and duration in particular) but are lightweight and easy to find.

Music21 is a library written by Michael Scott Cuthbert at MIT specifically for manipulating MIDI in Python. It can be found here: http://web.mit.edu/music21/. I used Music21 to parse the MIDI files into sequences of notes, determine dominant key, detect rests, extract durations, detect chords, etc. Only some of this information was directly used in training the network.

Other Programmers' Github Code was instrumental to this process, as mentioned above, Sigurður's work on classical piano composition was very helpful, and in a less direct way, several of the repositories for Magenta projects provided help and inspiration. Using others' code seems to be an essential tool and I was glad to have this opportunity to explore what level of borrowing feels appropriate.

Adapting my work from project 4 I used Gensim's Word2Vec constructor to train an embedding for the musical sequences I had extracted. The embedding vocabulary was 60 pitches and the number of notes played total was in the tens of thousands. In one version of the network the embedding was pre-trained and the weights loaded. In a different version the weights were trained as a part of the network layer.

The core toolset for this work was Keras, which provided the neural network architectures. Using Keras was one of my primary goals for project 5, regardless of subject matter.

Finally I used Garage Band and Ableton Live 10 to adjust timbres of the finished MIDI tracks which were created byt he network.

Data

The data I used were MIDI files for 120 irish folk songs from Frank Lennon's website (http://www.irishmidifiles.ie/midifiles.htm). For each of these I loaded the file into Garage Band and manually separated the melody from the accompaniment, saving the melody as a separate file (in .aif format) and with a small software utility I downloaded called GB2MIDI I was able to convert these to monophonic .mid files. Monohonic means there is only one line of music with no accompaniment.

I loaded monophonic MIDI melodies into Python using Music21. The melodic sequences were converted from music21 stream objects to lists of notes (in the process, rests were differentiated from other notes in the stream). The dominant key of the song was determined and stored for future use. A vocabulary was created and the note sequences were converted to corresponding integer indices. The notes (now integers) from all songs were concatenated and re-segmented into 100-note sequences followed by 1-note outputs by a moving window. This resulted in, again, some tens of thousands of input sequences.

After the network was trained and final weights were loaded to each layer, new oututs were produced. In all cases the network terminated in a softmax layer the length of the vocabulary. An argmax was applied to the output vectors to select the most probable next note. Assemblies of 500 of these winning notes were colated and exported as new midi tracks. Repeated notes were tied (converted from independently sounding notes to sustained sounding of the previous note) with an 85% probability.

Algorithms

By the end of the project I had tried about 25 different networks with variously assembled layers/layer option in Keras. Of these, I exported meaningful bodies of composed musical work from perhaps 12. In general my final model had the following characteristics:

```
def create_network(network_input, n_vocab): model = Sequential() model.add(CuDNNGRU(
256, input_shape=(network_input.shape[1], network_input.shape[2]) ))
model.add(Dropout(0.3)) model.add(Dense(n_vocab)) model.add(Activation('softmax')) opt
= RMSprop(lr=0.005) model.compile(loss='categorical_crossentropy', optimizer=opt)
model.load_weights('weights-improvement-42-1.2017-bigger.hdf5') return model
```

The defining characteristics here are the GRU layer, followed by Dropout (0.3) followed by Dense and finally the SoftMax activation layer.

Removing the dropout layers completely resulted in particularly boring, repetitive melodies, as did the inclusion of too many dense layers or layers with too many neurons (LSTM layers with 512 neurons, for instance).

The embedding layer was an interesting question. Neither the pre-trained Gensim Word2Vec embedding nor the embedding trained within the network gave any successful music generation. The loss did not decrease and the resulting melodies were strings of the same note. This was perplexing because no error occured. On Sunday before presenting I asked my housemate, (a Harvard music school graduate and a machine learning engineer at Google working on language models) and she suggested this was not uncommon and that it was likely that just the right tuning of parameters could result in a desirable output. I had tried many different parameter combinations already so I did not revisit the embedding during this time but I did find it odd that nothing resulted from this track and I am excited to return to it in future.

Lessons Learned (What to do dofferently)

In a general sense the lesson I learned (what to do differently) is to clearly state the project goals. I think part of why I had a difficult time emotionally on this project was a lack of a clearly stated limited set of goals. Secondarily I would have spent more time researching ahead of getting to work and designing the project to fit the scope of the abilities and time I had. Some particulars I wish I had addressed (which now become future directions) are:

a) I would like to feed the duration data into the neural network along with the pitch data so each 'note' becomes a tuple of pitch and duration (and, indeed, for future projects, I can also include a loudness characteristic).

b) As noted above I intend to continue struggling with the embedding. I believe there is a successful implementation and I intend to approach it by pulling intermediate data from the network to see at what point the signal is lost in noise.

c) I think it would be extremely interesting to compare different cultures' music - irish vs. greek for instance, or even different pop singers (Taylor Swift, for instance, typically sings around the tonic of the scale while Drake sings around the 4th).

d) In retrospect I would have preferred to follow Magenta's Performance RNN for this work rather than Skuldur's Classical Piano Composer. Magenta's project uses midi recordings of real performances and successfully captures the nuances of real human timing in the playing. This also imbues the network output with much greater realism than I can achieve.

e) Finally, I would have preferred to keep my project artifacts much more organized throughout the process. As I got more and more stressed my version tracking, file structure, commenting, etc. all became quite disorganized and I think this contributed heavily to the feelings of disorientation during that period.