Semester Project Suggestion "Finishing Johann Sebastian Bach's Unfinished Masterpiece"

Overview. Long time ago (1993) there was a time series prediction competition organized by the Santa Fe Institute which became rather famous in the community. The 6 data sets used for that competition have entered the *folklore* of the time series prediction community. Among the data sets, the last – and by far most difficult one – is an unfinished fugue written by Bach. The task was to use machine learning to complete the composition of this fugue. Achieving that would amount to finishing a genius' work… In retrospect, one can marvel at the audacity and innocence of the competition organizers; this task is of an unfathomable difficulty and far from being solved even today. But it is fun to try one's best and see what piece of artificial music art one can get with the machine learning tools that one masters.

Data. I suggest that before you think more about this you watch a youtube video that plays this piece of Bach known as *Contrapunctus XIV*: https://www.youtube.com/watch?v=JbM3VTIvOBk (if you search for "Bach Contrapunctus 14" on youtube you find dozens of other recordings, instrumental or vocal). The Wikipedia article https://en.wikipedia.org/wiki/The_Art_of_Fugue#The_Unfinished_Fugue gives historical and musicological background.

A detailed description of the dataset and some mathematical modeling background information is given in the book chapter

Dirst, M., Weigend, A. S., *Baroque forecasting: on completing J. S. Bach's last fugue*. In: Weigend, A.S. and Gershenfeld, N.A. (eds.), *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley 1993 (SFI Studies in the Sciences of Complexity). (A scan of this chapter is included in the project materials, file 3248_DirstWeigend93.pdf)

The data format delivered by the Santa Fé competition consisted of 4 voices, each encoding the music score as a sequence of integers which represent pitch. A copy is attached (file F.txt). The file F.txt represents each of the 4 voices as a (long, column) vector made from 0's and piano key indices. A typical part of a voice vector looks like

... 0 0 0 0 56 56 56 56 58 58 58 58 58 58 58 58 70 70 70 70 ....

The 0's denote breaks (no sound) and the other integers denote tones. The pitch of a tone is given by the integer (think of them as numbers of the black and white keys on a keyboard), the duration is represented by repeating a key number. Thus, in the above example, the "58" note is heard twice as long as the "56" note.

The Matlab script file displayOriginal4Voices.m (attached) provides a startup kit for reading the F.txt file and converting it to a Matlab sound vector that can be played. If you don't use Matlab, this script may still contain helpful hints for sonifying the F.txt data.

The **task description** is simple: use that dataset to train a model of this fugue (or rather, its beginning). Use the trained model to generate about 25 seconds of pseudo-Bach music. Enjoy...

Degree of difficulty: this is an unlimited-difficulty challenge. But with elementary methods you will already be able to obtain if not an "almost Bach-like" continuation of the unfinished fugue, but … at the very least, something that doesn't sound entirely like composed by monkeys.

Hints:

- The dataset consists of 4 voices. Don't aim too high for starters – just use one of the voices and create a continuation of that one voice. You will soon find that this is as difficult as can be.
- To have fun, you want to *listen* to the music. That is, one of the first things to do is to implement some method to create audio output from the key digit encoding in F.txt. If you are familiar with the `midi` format of music – that would be a fine intermediate data format which gives you all kinds of options for nice sonifications.

- Notice that the *Contrapunctus XIV* is a sequence of three fugues (listen to a youtube recording) of which only the last one is unfinished. You might want to use only the existing beginning of this last one as training data for your composition engine.
- A more technical hint: you need to train a timeseries prediction system which does more than just generate a single predicted note at each time. Instead, your predictor must generate a *probability distribution over the possible next notes*. You get this automatically if you implement a stochastic predictor like a hidden Markov model. But also if you use something as simple as a window-based linear regression predictor, you can tweak it to give you a probability vector output, like this: If there are 20 possible keys in your chosen voice (I didn't count them), the output of the predictor must be a 20-dimensional probability vector. A standard way to get this is to set up the teacher data $\mathbf{y}(t)$ in $(\mathbf{u}(t), \mathbf{y}(t))$ as follows:

  $\mathbf{y}(t)$ is a 20-dimensional, binary vector that is zero everywhere except at the position that encodes the note that is found in the training voice at time $t+1$. $\mathbf{y}(t)$ has a value of 1 in that component.

  If you train a linear regression in this way, magically you will obtain predicted output vectors $\mathbf{y}(t)$ at test time which sum to one precisely; they may have small negative entries though which you have to normalize away in some manner in order to get a probability vector.

- Once you have trained a predictor that gives you a probability distribution over next notes, you can use this for creating a "composition" by doing the following:
  1. at each time t, select one of the notes using the probabilities in $\mathbf{y}(t)$. This can be done, for instance, by picking that note with the highest probability; or by picking a random note with probabilities indicated by $\mathbf{y}(t)$; or by an intermediate strategy (which will likely be the best option).
  2. Once you have selected a note, append it to your "composition" and feed it back into the predictor as input (possibly transforming it to another data format, depending on which data format you used for $\mathbf{u}(t)$), in order to get $\mathbf{y}(t+1)$.
- This may sound complicated but once you understand the principle it's in fact straightforward – and it's the standard way to set up machine learning systems that can generate interesting non-deterministic output (music, poetry, robot dance patterns…).
- An important success factor is how you encode the scores given in key numbers in F.txt. Using those numbers directly is certainly not the best way to prepare an input signal $\mathbf{u}(t)$ for your predictor. I attach a BSc thesis which discusses the topic of good input encoding for automated music generation (Aulon Kuqi, *Art in Echo State Networks: Music Generation*, BSc thesis, Jacobs University 2017, attached file AulonKuqi_gr-thesis.pdf)
- A really non-trivial problem with machine music composition (or poetry generation…) is how to *evaluate* the quality of the generated "composition". The artistic quality is not at all the same as the technical loss function that you will use in your machine learning algorithm. In fact, the only evaluation that will be useful is to listen to what your algorithm composed, and judge for yourself.

The project report for this kind of project should be accompanied by some audiofile(s) that give a demo of your system's creation. – You are of course free to choose some other composer or some other piece (or pieces) of music for this composition project.