Midi Markov Manual

Musical Mimicry and Mixing in Max • Jared Apillanes

Midi Markov is a python-maxpatch blend that analyzes midi files offline to produce pitch-driven Markov chains that can be interpreted by Max and weighted by users to produce and record live midi data that blends differently sampled files together.

The **main.py** python file is used to generate the probability tables and export them as .jxf files interpretable by the jit library in Max. The file provides basic io for interpreting entire tracks of midi files as well as the ability to split the track into multiple tables so that different segments of the composition can be weighted differently during playback. By default, the program is configured to request one midi file and track number, and to export the data to one set of jxf files (one for each characteristic). The main function, **markov_midi**, also accepts measure points at which to break the track and export the data to separate files, an example of which can be seen in the file's comments. The file runs on python version 3.10 and requires the following third-party libraries: numpy (provides matrix support), and mido (used for its midi parsing). The most common way to install these libraries is through pip. main.py works by reading through the selected midi track and recording note-on and note-off messages. It maintains lists of the currently active notes, and last active notes, and adds links in the pitch table between these two sets each time a new set of notes is encountered. All tables hold data for each of the 128 notes in their own row, but differ in column length. Note duration is discretized into 5 lengths (whole, half, quarter, eight, and sixteenth notes) and as occurrences for each note. Notes played at the same time as a given note are considered to be in a chord and are recorded in the note's chord row. The number of notes played at once is also recorded

per occurrence of each active note in the chord size table. The tables are not normalized to provide a more accurate blend when weighting songs in the Max portion of the project. Generated files should either be included in the maxproject or placed within the Max search path. Sample jxf files have been provided in the project's data subdirectory.

The Max portion of the project enables live-sampling, and playback and recording of the produced tables. The **main.maxpatch** file provides the user interface for the project and allows users to import or remove jxf files, weight the added compositions, change playback tempo, start and stop playback, and visualize playback on a keyboard. The project relies on the jit.matrix data structure and its operations, but required slightly more complexity for loading files (see **matrixF.maxpatch**). The program works by summing the product of each loaded table and its specified weights. These tables are then subsampled to obtain the row of the current state and fed to the **table** object as discrete distributions to randomly sample. The new state is exported as live midi data (pitch, duration) and shifted to as the Markov chain's current state. The Markov chain is pitch driven, meaning that only the pitch is a Markov process, while the other attributes are pitch dependent probabilities. The pitch is therefore generated and fed as input to select the appropriate distributions for duration, chord size, and chord notes. To use the file, follow the comments in the presentation mode of **main.maxpatch** file. Again, ensure that all generated .jxf files are either located in the same directory and are included in the project, or are placed in the program's search path.

Unfortunately, the achieved product does not match the proposed project completely. Much of the expected base functionality did not exist within Max and designing and testing supplements took up a lengthy portion of the project's allocated time. Some desired features

were limited by other design choice, such as live input of midi data to modify the tables and smooth weight changing, while others were forgotten, such as recognizing and supporting rests in the midi files. Future work on the project would likely require an overhaul of many of the data structures and subroutines, as well as a reconsideration of the tables' pitch-dependency.