REPORT

Introduction to Artificial Intelligence Assignment 2: Music Generation

Adela Krylova

a.krylova@innopolis.university

07. 05. 2022

1. Task Description

We are given a .mid file with main song melody. The goal of the program is to generate melodic complement for the melody using genetical algorithm.

2. Terminology

Note – numerical representation of note for midi files – an integer value from 0 to 128

Chord – one to three tones (notes) played concurrently by the left hand

Complement – set of chords in order as they are played during the song (they might repeat themselves)

Generation / population – several complements

Epoch – one iteration in the genetic algorithm (steps 3, 4, 5)

Crossover – making new complement based on swapping some part of two old complements

Mutation – randomly changing couple of chords in the complement

Scale – set of notes which sounds good

Beat – in our case ¼ of a bar

Bar – for simplicity we divide songs to smaller periods of the same length, they are called "bars"

Winner – one of the 3 best complements within a generation

3. Solution

The solution consists of several smaller subproblems:

1) Read the midi file and analyze the melody

Every song is written in some scale. For simplicity I have considered only two different scales – *minor* and *major* scales. Using *music21* library for python I have analyze the scale, tempo and rhythm of the given song. Moreover, using this library I was able to parse the file into integer values representing each note and its length.

2) Randomly generate first population of complements – *"Pullum-et-Ovum"*

The genetic algorithm starts with a randomly generated population. Every scale consists of notes, a song written in one scale uses notes which belongs to this particular scale. Therefore, the first complements are generated based on this principle – there are lists of notes for every scale, from which I am choosing randomly a new chord.

3) Evaluate the generation and choose the best ones – *"Pollice Verso"*

For that purpose, I have written a *fitness function.* This function evaluates all complements within the given generation and chooses 3 best one individuals. The function evaluates the generation based on several different criterions:

a) How far are the notes within the chord from each other (neighbor notes are not melodic)
b) Penalizing small variety of different chords
c) Penalizing chords with the same notes

4) Make some mutation of the complements – "Mutatio vivere"

Mutation takes as a parameter only one old complement – a winner based on the fitness function. The function chooses some interval of the complement and changes it randomly. The number of mutations is chosen randomly.

5) Perform a crossover between the best complements – *"Gemino vivere"*

A crossover is a process of generating a new complement based on two old complements. The function takes two winners and chooses randomly some interval (the length of the interval is chosen randomly). This interval is swapped and newly created complement is returned.

6) Repeat 3, 4, 5, 6 steps – *"Per Aspera ad Astra (ad Infinita…) "*

After generating the first generation, the evaluation, mutation and crossover are performer again. The number of epochs strongly influence the overall quality of the given complement. However, the more the better does not work though. With too large number of epochs the results are spoiled (see the results chapter for more details about choosing optimal parameters for the learning).

7) Generate the midi file

The notes are represented as a list of integer values. In order to get audible song at the end, the song must be generated.

4. Results

I found out (empirically) that the best setting of the evolution algorithm is as follows:

```
Number of winners per generation:        3
Number of individuals per generation:    10
Number of chords per bar:                2
Number of epochs:                        100
```

After setting these values the algorithm provided with satisfactory results (the music is audible); however, sometimes some inappropriate chords might be chosen. Overall, I would say that the algorithm provides sufficient results.