Various Artificial Intelligence Techniques For Automated Melody Generation

Nikahat Kazi
Computer Engineering Department,
Thadomal Shahani Engineering College, Mumbai, India

Shalini Bhatia
Assistant Professor, Computer Engineering Department,
Thadomal Shahani Engineering College, Mumbai, India

Abstract

Music Composition involves different steps like melody creation, adding variations, rhythm, basses, etc. Out of these, melody generation is the basis of the musical piece created. This paper discusses generating novel pieces of monophonic melodies using three different techniques, namely, Elman Recurrent Neural Networks, a hybrid approach of Genetic Algorithms with Neural Networks as fitness evaluator and Genetic Algorithms with fitness evaluation by musical rules.

Keywords: *Melody generation, Elman recurrent neural network, genetic algorithm, genetic algorithm with neural network as fitness evaluator.*

1. Introduction

Music is a broad field in its own. Composing music is an art. It is a difficult task even for human beings. When a music composer manually writes music compositions, he has reason, the intention in his music, as well as his creativity. Automating the task of musical composition is an interesting application. Since composing music requires human intellect, the idea of using artificial intelligence to achieve the same task comes into being.

The aim of this project is the generation of a new "musical idea" which is a sequence of musical notes forming a basis for composing music. The inputs to the system are various melodies of various genres in the Ringtone Text and Transfer Language (RTTTL) format [1].

Two important areas in artificial intelligence have been used to achieve this task, namely, neural networks and genetic algorithms. There are three basic approaches that have been used. In the first approach, a neural network is trained to learn on the collected music samples. Several such trained neural networks are saved in the database. Later these networks are used in composition mode to create new melodies. In the second approach, one of the trained neural networks is picked up from the database and is used as a fitness evaluator in a genetic algorithm to generate new musical pieces similar to the ones learned by the neural network. The third method makes use of a genetic algorithm with musical rules as fitness evaluator.

The type of neural network that is used is an Elman recurrent neural network [2]. Elman networks are partially recurrent neural networks with an additional layer apart from the input, output and hidden layers called the context layer. The number of neurons in the context layer is equal to the number of hidden layer neurons. The feedback connections are from the hidden layer to the context layer. Elman recurrent networks can be used to learn temporal patterns as they have some amount of memory associated with them. The standard back-propagation algorithm is used to train the network with some modification to suit the Elman network used.

2. Background of Music Theory

Music theory is the study of how music works. It examines the language and notation of music. A simple set of definitions for musical terms are discussed below.

A melody is composed of an array of pitch - duration value pairs. A pitch comprises of two parts, the note and octave. Notes are depicted by seven letters of the English alphabet: A to G. A note, on its own, does not mean anything. When a note is combined with an octave, it becomes a pitch. An octave is the interval between one musical pitch and another with half or double its frequency. For e.g. A4-440 Hz and A5-880Hz are two pitches, one octave apart. An octave is an integer ranging from 0

to 8 on a piano keyboard. A complete musical note must also contain the duration value to be meaningful. The duration of a musical note is the length of time that a note is played. There are different durations which categorize notes as whole note, half note, quarter note, 8^{th} note, etc. The actual durations of notes are seen when the tempo value of the song is known, which gives how many 4^{th} (quarter) note durations are in a minute [3].

3. Literature Review

Research that has been done so far in automating music has been discussed in the sections below.

3.1. Music Generation using Neural Networks

The use of neural networks in music learning and composition has attracted researchers and many approaches have been developed.

The neural network proposed by Todd had three layers (context, hidden and output), as well as a set of "plan" inputs. The plan units were held at a set value for the full sequence, and were used to indicate to the network which sequence was being trained. Consequently, the network was required to learn each training sequence with only the plan as a stimulus [4].

The CONCERT architecture suggested by Mozer consisted of four layers in a linear arrangement: a current note layer, a context layer, a next-note-distributed (NND) layer, and a next-note-local (NNL) layer. The context layer used modifiable feedback weights, and, during composition, the output of the NNL layer was fed into the current note layer via a note selector. Since the network had modifiable backward connections, standard back-propagation could not be used for training; a method called back-propagation through time was employed instead [5].

3.2. Genetic Algorithms for Music Generation

Genetic algorithms in algorithmic composition have a short but interesting history, largely lying in the school of style replication. John A. Biles utilized genetic algorithms to generate jazz solos [6].

In [7], a genetic algorithm for making music compositions is presented. Position based representation of rhythm and relative representation of pitches, based on measuring relation from starting pitch, allow for a flexible and robust way for encoding music compositions. This approach includes a pre-defined rhythm applied to initial population, giving good starting solutions. Modified genetic operators enable significantly changing scheduling of pitches and breaks, which can restore good genetic material and prevent from premature convergence in bad suboptimal solutions. Beside main principles of the algorithm and methodology of development, in this paper, some solutions are presented in the musical score.

3.3. Music using a combination of Neural Networks and Genetic Algorithms

The combination of GAs and neural networks is a powerful tool for composing music. A hybrid method that adopts BP neural network for evaluation of emotions in music and genetic algorithm as an appropriate method for nominating creativity is presented in [8].

A music generation system using evolutionary algorithms and recurrent neural networks as the fitness evaluator is developed. The music generation process is fully automatic and requires no human interaction during the evolution phase [9].

4. Design

The design of the project, i.e. basically, the kind of data, the neural network architecture, chromosome structure, the genetic operators and the musical rules used in the genetic algorithm are discussed in the following sections.

4.1. Data Description

The input data to be processed is in the Ring Tone Text and Transfer Language (RTTTL). RTTTL was developed by Nokia to be used to transfer ringtones to cell-phone by Nokia. The RTTTL format is a string divided into three sections: name, default value, and data. The name section consists of a string describing the name of the ringtone. It can be no longer than 10 characters, and cannot contain a colon ":" character. The default value section is a set of values separated by commas, where each value contains a key and a value separated by an = character, which describes certain defaults which should be adhered to during the execution of the ringtone. Possible names are

d - duration
o - octave
b - beat, tempo

The data section consists of a set of character strings separated by commas, where each string

contains a duration, note, octave and optional dotting (which increases the duration of the note by one half).

Barbiegirl:d=4,o=5,b=125:8g#,8e,8g#,8c6,a,p,8f#, 8d#,8f#,8b,g#,8f#,8e,p,8e,8c#,f#,c#,p,8f#,8e,g#,f#

Figure 1. A melody in the Rtttl format

As an example, consider the melody in Figure 1. The name of the melody is Barbiegirl, followed by the default section which tells that the default duration of the melody is 4(quarter note), the default octave is 5 and the tempo of the melody is 125 beats/minute. Following the default section is the data section, which is the sequence of musical notes of the melody. The default values are used to fill in the missing values in the data section. For example, consider the first note in the melody, i.e., 8g#. It means that the duration of the note is 8(8^{th} note), and the note itself is g#, but there is no value for octave, so the octave in this note, comes from the default section, i.e. 5. So, this musical note is 8g#5. Similarly, the default duration can also be used if it is missing in a musical note in the data section [1].

A database of melodies in the RTTL format has been created using three different genres, namely, pop (10 melodies), hip-hop (10 melodies) and R&B (10 melodies).

4.2. Input Pre-processing

To be able to train the neural network with the melodies, they need to be converted to a format which the neural network would understand. Each input training vector to the network identifies the duration, note and octave of the current musical note. So, for that purpose, the input should be first converted into a sequence of musical notes of this form.

After each musical note appears in the required form, the next step in this stage is to separate each musical note into duration, note and octave. For example, consider the musical note 16d#5.After separating, duration=16, note=d#, octave=5.This must be done for every musical note in the melody sequence.

4.3. Neural Network Architecture

The neural network used for the project is an Elman recurrent neural network in Fig. 2, with one input layer consisting of 25 units, a hidden layer with variable units, a context layer comprising of the same number of units in the hidden layer to which output from the hidden layer of the previous time step is fed, and an output layer consisting of 25 units.

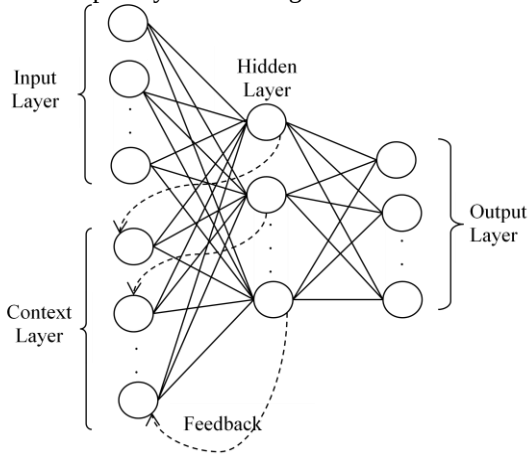

Figure 2. Elman Neural Network Architecture

The inputs to the network are duration, note and octave of the current time step and the target outputs are the duration, note and octave of the next musical note in the melody sequence. So, at each step, the neural network tries to predict the next musical note in sequence.

The input layer is linear. So, the input vector designed was binary of size 25 × 1.

The possible no. of durations are 7, the no. of notes are 13 including rest note and there are 4 octaves possible, which sums up to 24. To take into account the dotted notes, an additional bit is used.

The first 7 bits represent the duration, next 13 bits the note, the next 4 bits the octave and, the last bit is for representing the dotted note.

A "1" means that particular note is active at the current time step, otherwise the bit for the same is set to a "0".

4.4. Chromosome Structure

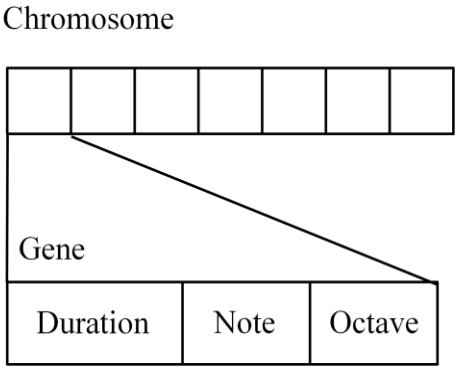

Figure 3. Chromosome structure

Each individual consists of a chromosome and an associated fitness value. The structure of the chromosome for this project is shown in Figure 3 [9]. Each chromosome is an individual melody. The chromosome consists of several genes. Each gene is a musical note which further comprises of three parts, duration, note and octave.

Considering the melody of Fig. 1, the first musical note of the melody is 8g#5, which translates into the first gene as duration=8, note=g# and octave=5, the next note is 8e5 which translates into the first gene as duration=8, note=e and octave=5, as per Figure 3.

4.5. Genetic operators used

In the GA, at the end of a generation, the genetic operators namely, selection, mutation and crossover, work on a set of individuals forming the population.

Roulette wheel section was used as the selection operator. In this scheme, the fitness level of a chromosome is used to associate a probability of selection with each individual chromosome. If f_i is the fitness of individual 'i' in the population, its probability of being selected is $p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$, where N is the number of individuals in the population.

Swap, scramble, inversion and replace were the mutation operators used [9]. The swap mutation is used to swap two randomly chosen musical genes. Scramble operator shuffles the genes between 2 randomly chosen points. Inversion operator inverts a chromosome between chosen points. Replace is used to replace a randomly selected gene with a new, randomly generated gene.

The operators used for crossover were one-point, two-point and uniform crossover [9]. One point crossover selects a random point, slices two different genomes around this point, and creates children genomes with those slices. Two point crossover does the same thing by selecting two random points. Uniform crossover does not limit the number of slice points, distributing all genes from the parents to the offspring uniformly randomly.

Further, the crossover probability was set to 0.7 and the mutation probability was set to 0.1. These operators were then used according to the above mentioned probabilities.

4.5. Musical Rules in the genetic algorithm

The musical rules approach is a basic set of rules that a composer must follow for making good melodies. These were formed after discussions with several musicologists. They are listed below.

1) Scale Rule

This rule calculates if a given melody adheres to a specific scale, for example C major (C D E F G A B). The scale can be chosen by the user.

2) Adjacent Notes Rule

It calculates and verifies whether the difference between most adjacent notes (around 70%) is not more than a step and whether leaps are limited to less than 4 semitones.

3) Proportion between Rests and Notes

It finds the proportion between the notes and rests in a given melody. The proportion of notes/rests is set as a parameter.

4) Repeating Notes Rule

If a given melody has repeating notes or rests, it must be observed that it does not exceed a threshold, lest the melody may sound repetitive and boring. The maximum number of repeating notes and/or rests can be set as parameters.

5) Global Pitch Distribution Rule

This rule validates if the lowest and highest pitch of a given melody fall within the margins specified. The margin is indicated as the number of semitones.

5. Algorithms Used

Various algorithms which are used as part of the project implementation are discussed below.

5.1. Steps in Input Pre-processing

This is the first step in the implementation.

Input

Melodies in the rtttl format.

Output

Melodies in the form of array of duration, note, octave triplets

Algorithm

For each melody in the training set
Do

1. Extract the musical notes sequence from the rtttl format.
2. Split the sequence into individual musical notes
3. Separate each musical note into duration, note and octave form.

End.

5.2. Algorithm for Training the Recurrent Neural Network

The following standard back-propagation algorithm was used to train the neural network.

Input

Melodies in the form of array of duration, note, octave triplets

Output

Trained Neural Network with Final Weights and Emax

Algorithm

1. Initialize the weights in the network (small random values)
2. Do
   For each example e in the training set
   a. Compute O = neural-network-output (network, e) //forward pass
   b. Copy hidden layer output to context layer
   c. Compute error (T - O) at the output units [T = teacher output for e]
   d. Compute delta_wh for all weights from hidden layer to output layer // backward pass
   e. Compute delta_wi for all weights from input layer to hidden layer // backward pass continued
   f. Update the weights in the network
   Until stopping criterion satisfied
3. Return the network [10].

5.3. Genetic Algorithm using Elman Neural Network as Fitness Evaluator

The genetic algorithm combining a neural network as applicable to the project has been discussed below.

Input

Trained Neural Network with Final Weights and Emax

Output

Composed melody as MIDI file

Algorithm

1. The initial population of melodies is created in multiples of 2 for practical reasons, which is created using a random sequence of musical elements.
2. Generate the chromosomes to form the current population of the current generation.
3. Evaluate the fitness of the various chromosomes using a trained RNN. Each value pair is given to the NN as input value, and the consequent pair is used to compare to the output value and the difference is calculated. The differences for all pairs are accumulated to obtain a final error value, which is used to calculate the fitness value of the individual.
   Mathematically, it is given by,
   $$f_j = \sum_{i=1}^{m} (t_i - a_i) \quad ; j = 1,2, \dots, n$$
   Where m is the total number of value pairs, t_i is the target value i.e. the next consecutive value pair and a_i is the actual value and n is the number of individuals.
4. If any chromosome is fit enough, stop here and output the chromosome as the melody of the current generation else goto Step 5.
   Here, compare each f_j computed earlier with the desired minimum fitness value F_{min}.
   If $f_j < F_{min}$, save fitness value and the corresponding individual.
   Output melodies for all such saved fitnesses and stop the Genetic Algorithm.
5. Perform Selection using roulette wheel selection.
6. Perform Mutation and Crossover.
7. Goto Step 2.

5.4. Genetic Algorithm using Musical Rules as Fitness Evaluator

In this variant, the genetic algorithm uses musical rules from music theory as a fitness function.

Input

Melodies with random musical notes sequence

Output

Composed melody as MIDI file

Algorithm

The genetic algorithm for melodies uses simple musical rules from music theory. The fitness function is composed of checking the validity of these rules as to how the initial melodies differ from the ideal ones.

1. Create initial population of melodies. The initial population is created using a random sequence of musical elements.
2. Generate the chromosomes to form the current population of the current generation.

3. Evaluate the fitness of the various chromosomes using musical rules. All rules calculate the deviation between the generated melody and the specified rules. A lower fitness value means less deviation which means a better melody.

$$f_j = \sum_{i=1}^{m} errorCount_i \;\; ; j = 1,2,\dots,n$$

Where m is the number of musical rules and n is the number of individual chromosomes.

4. If any chromosome is fit enough, stop here and output the chromosome as the melody of the current generation else goto 5.
   a. Compare each f_j computed earlier with the desired minimum fitness value F_{min}.
   b. If $f_j < F_{min}$, save fitness value and the corresponding individual.
   c. Output melodies for all such saved fitnesses and stop the Genetic Algorithm.
5. Perform Selection using roulette wheel selection.
6. Perform Mutation and Crossover.
7. Goto Step 2.

6. Results and Discussion

The results obtained as part of the project have been discussed including the neural network training, genetic algorithm using neural network as fitness evaluator and a genetic algorithm using musical rules as fitness function.

6.1. Neural Network Training Results

An adaptive learning constant was used for training whose values were varied between 0.01 and 1.0. Hidden layer neurons were kept variable. First, training was performed on a single melody chosen from the database of each genre. Next, training was performed, taking 10 melodies from each genre. The trained neural networks were then saved to the database. After the training, these trained networks were picked up from the database one by one, and were run in composition mode [11] by feeding the current output as input for the next time step. Thus, based on the learned training set, a new melody was composed each time by the neural network. A summary of the results have been shown in Table 1.

Table 1. Summary of Neural Network Training Results with Emax = 0.01 and hidden neurons = 90

Genre	No. of Songs	No. of Steps	Accuracy
Pop	1	910	100%
Pop	10	505000	88%
Hip-Hop	1	1270	100%
Hip-Hop	10	399400	90%
R&B	1	1010	100%
R&B	10	78000	94%

Accuracy of prediction was measured according to the formula given below:

$$Accuracy = \frac{No.of\ notes\ correctly\ predicted}{Total\ no.of\ notes} \times 100$$

6.2. Results of Genetic Algorithm with Neural Network

While the first generations produce completely random series, more "music aware" results are obtained in the next steps throughout the evolution process. A summary of the GA evolution results is given in Table 2.

Table 2. Summary of Genetic Algorithm (using NN) results with population size 50 and stopping fitness Fmin= 0.09

Genre	No. of Generations	Running Time (seconds)
Pop	5754	5560
Hip-Hop	785	3321
R&B	264	1458

6.3. Results of Genetic Algorithm using Musical Rules

The simple GA with musical rules produced melodies with minor glitches. It was observed that with increasing population size, the GA converged faster. The results are summarized in Table 3.

Table 3. Summary of Genetic Algorithm (using musical rules) results with stopping fitness Fmin=0

Population Size	No. of Generations	Running Time (seconds)
10	5940	68
50	1044	63
100	173	19

7. Conclusion and Further Work

The melodies generated from Elman neural network, when run in the composition mode, were pleasant to hear. However, the training time was, typically, a few hours.

The genetic algorithm used each of these trained networks as fitness evaluators to generate melodies. It generated melodies with a few minor glitches. The running time was in minutes, but this approach assumes that there are trained neural networks available.

The algorithmic musical composition using genetic algorithm produced melodies in quick time (typically, few seconds to a few minutes), and these melodies were pleasant. The only musical knowledge imparted was the set of musical rules. The melodies were simple, yet pleasant. Perhaps, adding a few more musical rules would structure melodies more appropriately.

As future work, this idea can be extended to generating polyphonic melodies which have more variations and sound like a more complete musical piece.

References

[1] RTTTL Format [Online]. Available: http://www.srtware.com/index.php?/ringtones/rtttlformat.php
[2] Jeffrey L. Elman, "Finding Structure in Time", Cognitive Science: Vol.14, pp 179-211, 1990.
[3] Ricci Adams' *musictheory.net* [Online]. Available: http://www.musictheory.net /lessons
[4] Todd, P. M. A. "Connectionist Approach to Algorithmic Composition." Computer Music Journal: Vol.13, No. 4, 1989.
[5] Mozer, M. C., "Neural Network Music Composition by Prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing." Connection Science, 6(2-3), pp. 247 – 280, 1994.
[6] Biles, J.A., "GenJam: A genetic algorithm for generating jazz solos", In ICMC Proceedings 1994, The Computer Music Association, 1994.
[7] Dragan MATIĆ, "A Genetic Algorithm for Composing Music", Yugoslav Journal of Operations Research, Volume 20 (2010), No. 1, pp. 157-177,2010.
[8] Minjun Jiang and Changle Zhou, "Automated Composition System based on GA", IEEE, pp. 380-383, 2010.
[9] Ali Çağatay Yüksel, Mehmet Melih Karcı and A. Şima Uyar, "Automatic Music Generation Using Evolutionary Algorithms and Neural Networks", IEEE, pp. 354-358, 2011.
[10] Paul J. Werbos, " The Roots of Backpropagation. From Ordered Derivatives to Neural Networks and Political Forecasting", New York, NY: John Wiley & Sons, Inc., 1994.
[11] Débora C. Corrêa, Alexandre L. M. Levada, José H. Saito and João F. Mari, "Neural Network Based Systems for Computer-Aided Musical Composition: Supervised x Unsupervised Learning", ACM, SAC'08, pp. 1738-1742, 2008.