Neural Networks Paper Title

Thesis Subtitle

Sneha Lodha
Marco Gallo
Max Falziri
Yasmin de Groot

Neural Networks
University of Groningen
7th of July 2020

Abstract

abstract text (10 lines)

Contents

1 Introduction

Classical music generation outside the domain of AI is mainly inspired by creativity and the knowledge of some musical elements. There are several methods that have been used in the past to perform similar tasks, some of which include the use of algorithmic composition and Markov chains. Composing music through algorithmic rules can be restrictive in terms of originality, and user feedback. [1] Once the algorithms to generate the music have been defined, there is little to modify or fix. When using Markov chains for the generation of music, although the presented output has more variation, drawbacks such as the violation of the Markov property come hand in hand. [2] This is mainly due to the fact that Markov property relies on simply the previous output which is not enough when it comes to generating musical patterns. In this project, we will be attempting to generate musical drum patterns using an Echo State Network, also referred to as an ESN, and measure how close the generated pattern is to actual music.

An ESN is a type of Recurrent Neural Network (RNN) with an input layer, a hidden layer, also known as a reservoir and an output layer. The interesting aspect of the ESN is that the reservoir to output weights are the only ones that need to be learned, and the weights of the reservoir itself are fixed. Due to this property of the learning the ESN is considered faster than some other RNNs. We decided to use

2 Data

In this section the data used in this project is described and explained. The first subsection will explain how the drums are represented, and the second subsection will describe the generation of the music.

2.1 Representation of the drums

The drum beats are represented in the network in the form of a matrix. In Table 1 an example of such a matrix can be seen. All the matrices will have 9 columns, each representing different notes as seen in the table. Each row in the matrix this matrix will represent a quarter, and 4 row will represent a bar. If the tempo of the song is for example in 8th, 8 rows will together represent a bar. This is thus dependent on the tempo of the song.

The matrix seen in Table 1 represents 1 bar of music. Longer files will consist of more rows. Therefore the time is thus represented in the rows of the matrix. In this table all the column are set to 0 instead of the bass drum. This will therefore produce one bar of music with a bass drum playing on the first and third quarter.

	Hi-hat-closed	Hi-hat-open	Bass drum	Crash	Snare	High-tom	Mid-tom	Floor-tom	Ride
Quarter 1	0	0	1	0	0	0	0	0	0
Quarter 2	0	0	0	0	0	0	0	0	0
Quarter 3	0	0	1	0	0	0	0	0	0
Quarter 4	0	0	0	0	0	0	0	0	0

Table 1: An example of how the input matrix is structured.

2.2 Generation of the music

The database used for training is generated by hand in this project. In order te generate music, two functions were implemented. The first one being `note_replicator` and the second one being `note_generator`. The replicator generates a note in a bar using the tempo, the note, and in what quarter it plays (this can also be a list of quarters / eight's etc). This bar is then repeated during the course of the whole dataset. The generated works in a similar way, only will this also need a variable representing how many bars this note is played. Besides the two generating function, another function is implements that allows the user to concatenate matrices. This is used to combine different "songs" into one long song in such a way that the network can use it to train. This way ... bars of music were generated to train the network.

3 Methods

In this experiment, one pipeline is used. This consists of the learning algorithm, the post processing and the performance metrics. All of these steps will be explained in detail in this section. The reason there is no pre-process present is because the data is generated in such a way that it does not need to be pre-processed for the network. The way this data is generated was explained in the previous section. The code of this project is mainly written in Python, while the statistical analysis is done in R.

3.1 Echo State Network

In this music generating task an ESN is used. The input signal of the ESN is created as described in the pre-processing. The set up of this ESN is based on "GUIDE REFERENCE". This guide contained the base code of this project. It also provided the needed knowledge about the network and its parameters as described below. The given code has been rewritten to fit this project. The transformers, sparse matrix and noise vector were added as an addition. The documentation of the code can be found *here*. Regarding the ESN, the following parameters have been set:

The reservoir size

Within ESNs, it is severely important that the reservoir is big enough, such that is it possible to obtain the target output $y^{target}(n)$ from a linear combination of this signal space. The reservoir in this project has been set to ...

The reservoir density

The density of a reservoir is mainly dependent on the distribution of the nonzero elements in the reservoir. In this project a basic uniform distribution is used. Besides the distribution, the density of the reservoir is set to ....

Spectral radius

Another main parameter for fitting the ESN is the spectral radius ϱ. This spectral radius is a parameter that will scale the reservoir matrix $\mathbf{W}$. The effect this parameter has is mainly seen on the learning accuracy of the the network. The new scaled matrix $\mathbf{W}$ is calculated using

$$\boldsymbol{W_{new}} = \mathbf{W} * (\frac{\varrho}{\max(|\lambda|)})$$

where λ represents the eigenvalues of the reservoir matrix $\mathbf{W}$, and $\mathbf{W_{new}}$ represents the updated reservoir matrix. After experimenting, the ϱ has been set to ...

Leaking rate

The leaking rate α for an ESN determines how well a reservoir unit maintains its value and how much it gets updated. Therefore α is one of the main parameters regarding the training process of this project. In this project α is set to ...

Regularization

The regularization of this project is implemented using a ridge regression in the learning step of the network. This is used to stabilize the output in the long run. The parameter that scales the identity matrix in the ridge regression is the actual set parameter in this situation. In this code this parameter is set to e^{-8}.

Besides the ridge regression, a noise vector is also present in the code. This is not used during this project.

Transformers

@Max :) Transformer (the 3 mentioned is the intro paper: treshold, sigmoid, sigmoid probability) (?) each transformer has a parameter and a squeezing function (for now).

3.2 Post-processing

Output

not sure what to write here yet

3.3 Fitting

evaluation function (MLP/Jaeger idea). expanding parameters: bfs with gradient decent.

4 Results

5 Discussion