MATHEMATICAL MODELING OF PROCESSES AND SYSTEMS

UDC 681.5.013 (045)

A. V. Vishnevsky

THE BINARY WEIGHT MATRIX OF AN ELECTRONIC COMPOSER ALGORITHM

National Aviation University, Ukraine, Kyiv
E-mail: av@nau.edu.ua

Abstract*—*The binary realization of logical structure, that can be represented as weight matrix of a neural network electronic composer – has been offered.

Index Terms—Digital music; electronic composer; neural network; information processing; artificial intelligence.

I. Introduction

The work of neural network is determined by its weight matrix – states the principle of connectionism. And we can say, that neural network (NN) based musical processing routine can be described by this matrix (or these matrices), too.

So, for our study it can be undoubtedly said, that the weight coefficients matrix (WM) has paramount importance. Let's examine closer this matrix, and take care of it's structure and role for digital music processing.

II. Analysis of Researches and Publications

A very solid basement for the future building of the artificial intelligence (Al) model of an electronic composer has been proposed in [1]. A good example of real achievements in this direction are works [2], [3]. Source of inspiration for current paper is [4-6]. [7] is a preceding publication of the author of this paper on the topic of research.

III. The Neural Network Mechanism

Before beginning composing digital music, you need to have an exact notion of how to do it.

This notion gives a direction, in which in future is developed a specific programming digital music composing routine. The routine presented in this paper is based on the following structural scheme (Mathlab generated).

Figure 1 shows a generalized view of the neural network digital music processing scheme realization. Here **TDL** represents a time delay unit, **tansig** is a transfer function being used, **bias** is the an integer value in range [-255; 255], **netsum** is the adder of the weight coefficients and the bias. In this scheme the WM circuit block, which is the most important element for this scheme, hasn't been shown yet. It's revealed in Fig. 2. here **Mux** stands for multiplexor unit, **dotprod** is a logical operation of dot product, **weights** is the NN weight coefficients (for case of 10-neuroned feedforward network).

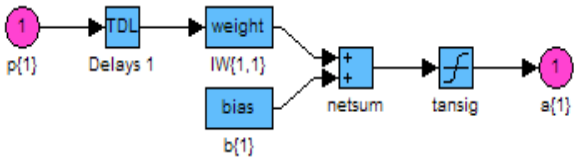

Fig. 1. The neural scheme

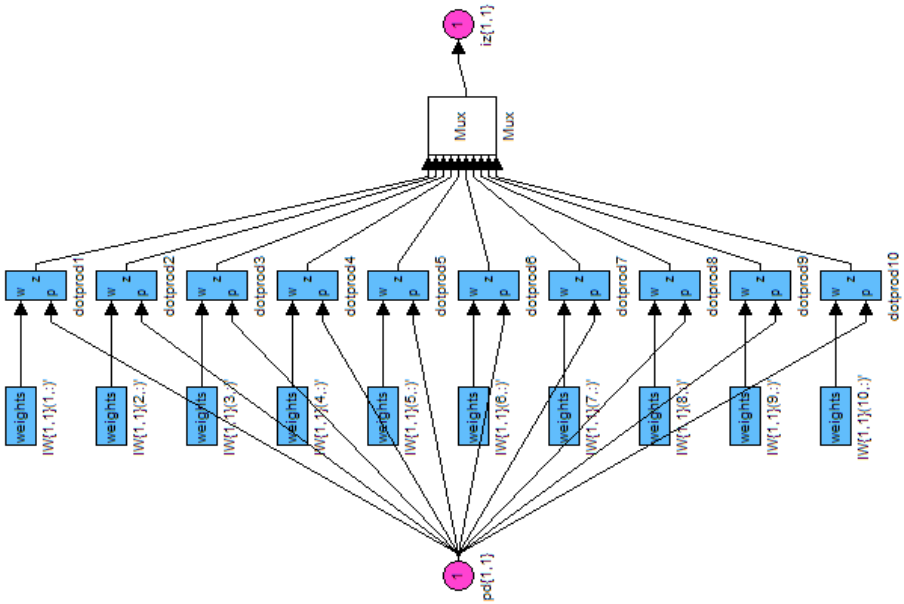

Fig. 2. The weight coefficients block

At this picture the inner philosophy of two digital music processes routing has been delivered, of which the first being vector $p[i]$ (our initial signal, coming from the output of generator of control signals), when the second being $w[i]$ (vector of the weight coefficients of WM).

IV. THE EQUATION

A matrix equation connecting input (P), weights (W) and output (A) is showed below. It works for any NN-structure imaginable (it's written out here without taking into account biases vector **B**).

$$\begin{vmatrix} w_{1,1} & w_{1,\dots} & w_{1,i} \\ w\dots & w\dots & w\dots \\ w_{1,j} & w_{\dots,j} & w_{i,j} \end{vmatrix} \bullet \begin{vmatrix} P_1 \\ P\dots \\ P_j \end{vmatrix} = \begin{vmatrix} A_1 \\ A\dots \\ A_{i,j} \end{vmatrix}.$$

Or, alternatively, the neural network based musical sequence processing algorithm, conveyed in this paper, can is written by formula

$$|\mathbf{W}| \bullet |\mathbf{P}| = |\mathbf{A}|, \qquad (1)$$

where **W** is the matrix of weight coefficients; **P** is the input musical sequence (pattern); **A** is the output musical sequence (abaculus).

Let's designate by symbol of logical "1" note *D*, and by symbol of logical "0" note *A*. The input sequence *P* will look like is shown in Fig. 3.

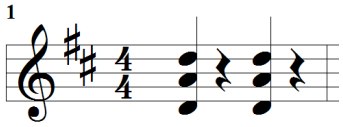

Fig. 3 Initial sequence for example one

The matrix of weight coefficients **W** for example one will have the following form (if we continue to use same notes for "1" and "0") (Fig. 4).

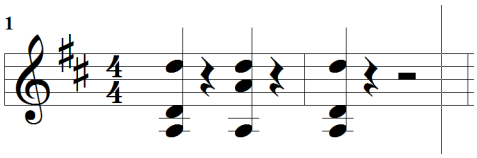

Fig. 4 The weight coefficients matrix for example one in staff view

As we can see, as a result of input sound sequence neural-network-based processing, the output sequence A will be different from the initial one. Figure 5 represents this transformation:

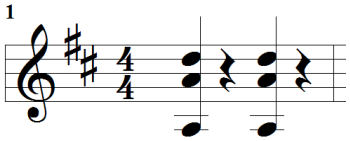

Fig. 5 Output sequence for example one

The mathematical notation for this example will have the classical matrix-multiplication-style form, derived from (1):

$$\begin{vmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{vmatrix} \bullet \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} 1+1+1 \\ 0+0+0 \\ 0+0+0 \end{vmatrix} = \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix}.$$

So, we've processed our music!

Let's add, that **W**-matrix from example one had "zero-triangle" form. This, and different other forms that we will use or we could use (but won't in this paper because of lack of space and time) remind very well the Kohonen's neural network weight matrix construction concept. For this reason one can think of presented in this paper algorithm as a partially Kohonen's network related algorithm.

Two next **W**-matrices can be referred to as "zero-rhomb" matrix (first), and "one-rhomb" matrix (second).

$$\begin{vmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{vmatrix} \bullet \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} 1+0+1 \\ 0+0+0 \\ 1+0+1 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix},$$

$$\begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{vmatrix} \bullet \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} 0+1+0 \\ 0+0+0 \\ 0+1+0 \end{vmatrix} = \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix}.$$

A collection of these **W**-matrices can be used by an electronic composer like a kit of lead letters can be used by a compositor:

$$\begin{vmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{vmatrix} \bullet \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} 1+1+1 \\ 0+0+0 \\ 1+1+1 \end{vmatrix} = \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix} \text{ ("1"-comb),}$$

$$\begin{vmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{vmatrix} \bullet \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} 0+0+0 \\ 0+0+0 \\ 0+0+0 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix} \text{ ("0"-comb),}$$

$$\begin{vmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{vmatrix} \bullet \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} 1+0+1 \\ 0+0+0 \\ 1+0+1 \end{vmatrix} = \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix} \text{ (mirror "1"-comb),}$$

$$\begin{vmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{vmatrix} \bullet \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} 0+1+0 \\ 0+0+0 \\ 0+1+0 \end{vmatrix} = \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix} \text{ (mirror "0"-comb), etc.}$$

It is an interesting fact, that the WM can be thought of as P column-matrix, when appropriate **W** matrix from (1) may be referred to as input musical sequence. In this case WM instantaneously processes not only harmony, but the melody, too. The size of melodic line being processed is determined by size of **W** matrix. And not only one

melody can be processed, but several. The number of melodies treated equal to WM size. If this number is too big, WM can be filled with zeros at lower rows.

V. PROGRAM REALIZATION

Console application C++ code for main() function is listed below.

```
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "C:\...\lw.h"
#include "C:\...\dw.h"

int q[100];
int d[120];
int l[100];
int fin[4];

FILE *stream;

int _tmain(int argc, _TCHAR* argv[])
{
{
q[1]='M';q[2]='T';q[3]='h';q[4]='d';q[5]=0;
q[6]=0;q[7]=0;q[8]=6;q[9]=0;q[10]=1;q[11]=0
;
q[12]=2;q[13]=0;q[14]=120;q[15]=77;q[16]=84
;q[17]=114;q[18]=107;q[19]=0;q[20]=0;q[21]=
0;
q[22]=46;q[23]=0;q[24]=255;q[25]=3;q[26]=8;
q[27]='u';q[28]='n';q[29]='t';q[30]='i';q[3
1]='t';q[32]='l';q[33]='e';q[34]='d';
q[35]=0;q[36]=255;q[37]=1;q[38]=5;q[39]='S'
;q[40]='a';q[41]='s';q[42]='h';q[43]='a';q[
44]=0;q[45]=255;q[46]=88;q[47]=4;q[48]=4;q[
49]=2;
q[50]=24;q[51]=8;q[52]=0;q[53]=255;q[54]=89
;q[55]=2;q[56]=0;q[57]=0;
q[58]=0;q[59]=255;q[60]=81;q[61]=3;q[62]=9;
q[63]=39;q[64]=192;
q[65]=0;q[66]=255;q[67]=47;q[68]=0;q[69]=77
;q[70]=84;q[71]=114;
q[72]=107;q[73]=0;q[74]=0;q[75]=0;q[76]=61;
q[77]=0;q[78]=144;

fin[1]=255;
fin[2]=47;
fin[3]=0;

stream = fopen("DUMMY.mid", "w+");

for(int i=1;i<=78;i++)
{
fprintf(stream, "%c", q[i]);
}
srand(29);
fclose(stream);

////////////////////////////tgt sequence
srand(12);lw._32(4);srand(6);lw._32(8);sran
d(7);lw._32(8);srand(50);dw._32(4);srand(50
0);dw._32(4);srand(19);dw._32(4);srand(46);
lw._32(8);srand(99);lw._32(8);srand(1);lw._
32(8);srand(30);
////////////////////////////

stream = fopen("DUMMY.mid", "a+");
for(int i=1;i<=3;i++)
{
fprintf(stream, "%c", fin[i]);
}

   /* close the file */
   fclose(stream);

}
 return 0;
}
```

This code represents by itself an “engine” for the AI NN electronic composer. Grafical user interface (GUI) implementation is not given here, though it has been developed successfully.

A part of one of header files (for diatonic composing) where musical information processing classes are stored, is included.

```
#ifndef lwH
#define lwH

class light_water
 {
 public:
 int w,w1,w2;
 int l[52];
 double W[3][3];
 int b;
 int tonic_accord[4];
 int boo;
 void _32(int length);
 void _16(int length);
 void _16W(int length);
 void _8(int length);
 void _4(int length);
 . . .
 void _tonic16(int length);
 light_water();
 FILE *stream;
 }lw;

 light_water::light_water()
 {
//////////////////tonic_accord[3]
tonic_accord[1]=48;
tonic_accord[2]=52;
tonic_accord[3]=55;

//for (int i=1; i<=3; i++)
//{
//for (int j=1; j<=3; j++)
//{
//W[i][j]=(double)rand() / RAND_MAX;
//}
//}

//weight coefficients

W[1][1]=0.22;//(double)rand() / RAND_MAX;
W[1][2]=0.22;//(double)rand() / RAND_MAX;
W[1][3]=0.22;//(double)rand() / RAND_MAX;

 ///////////////light water array
```

```
        l[1]=40;      l[2]=41;      l[3]=43;
l[4]=45;  l[5]=47;  l[6]=48;  l[7]=50;
        l[8]=52;      l[9]=43;      l[10]=55;
l[11]=57; l[12]=59; l[13]=60; l[14]=62;
        l[15]=64;     l[16]=65;     l[17]=67;
l[18]=69; l[19]=71; l[20]=72; l[21]=74;
        l[22]=76;     l[23]=77;     l[24]=79;
l[25]=81; l[26]=83; l[27]=84; l[28]=86;
        l[29]=88;     l[30]=89;     l[31]=91;
l[32]=93; l[33]=95; l[34]=96; l[35]=98;

l[36]=100;l[37]=101;l[38]=103;l[39]=105;l[4
0]=107;l[41]=108;l[42]=110;

l[43]=112;l[44]=113;l[45]=115;l[46]=117;l[4
7]=119;l[48]=120;l[49]=122;
        l[50]=124;l[51]=125;l[52]=127;
        }
        void light_water::_16(int length)
        {
        stream = fopen("DUMMY.mid", "a+");

        //light water
       length=length*16;
       for(int i=1;i<=length;i++)
       {
       w=random(32)+1;
       . . .
       }
       fclose(stream);
        }//end_16

         void light_water::_16W(int length)
        {
        stream = fopen("DUMMY.mid", "a+");

        //light water
       length=length*16;
       for(int i=1;i<=length;i++)
       {

       //this is the genetic algorithm
       b=22;
       w=  (int)(  (random(51)*W[1][1]+b)  +
(random(51)*W[1][2]+b)                      +
(random(51)*W[1][3]+b) )/3 ;
       . . .
       }

       fclose(stream);
        }//end_16W


         void light_water::_8(int length)
        {
        stream = fopen("DUMMY.mid", "a+");

        //light water
       length=length*8;
       for(int i=1;i<=length;i++)
       {
       w=random(42)+1;
       . . .
       }
       fclose(stream);
        }//end_8
       . . .
       #endif
```

Graphical user interface on one hand makes a better performance of the program for a user (intuitive simplicity of work), on the other hand – allows to start visualisations, especially needed when dealing with musical therapy applications (it has been studied that music lowers blood pressure, boosts immunity, eases muscle tension, reduces stress, increases/decreases energy, influences emotion, produces changes in brain wave activity, lowers the breathing /heart rate, relieves repression /anxiety).

Conclusion

The AI NN model of an electronic composer scheme and the WMs has been given in this paper. This neural scheme has been translated into a PC program, that helps to compose digital music. It is a feedforward NN that can be realized both in one- or multilayered versions.

The number of hidden layers depends on a kind of musical task set. A lack of feedbacks and target vectors in this model leads to a very quick “on-line” processing mode, that allows to compose large musical texts very rapidly. This helpful instrument is able to significantly reduce a lot of monotonous work when creating musical texts of a steady emotive temper.

The weight matrix model for case of binary form of weight coefficients is proposed in two different variants.

Acknowledgment

The author would like to acknowledge Dr. Sc., Prof. V. V. Vasilyev, Dr. Sc., Prof. A. Y.Beletsky and Dr. Sc., Prof. A. V. Solomentsev for any support of this research.

References

[1] Gloushkov, V. M.; Introduction to Cybernetics. Academic Press, New York, 1966. Published in Russian, 1964., 324 p.

[2] David, Cope. “Experiments in Music Intelligence”. *In Proceedings of the International Computer Music Conference*, San Francisco: Computer Music Assn, 1987.

[3] Wiggins, G. A. Computer Models of Musical Creativity: A Review of Computer Models of Musical Creativity by David Cope. Literary and Linguistic Computing, 2007. no. 23 (1), pp. 109–116.

[4] http://en. wikipedia. orz/wiki/The Well-Tempered Clavier

[5] Johann Sebastian Bach. Das Wohltemperierte Klavier I BWV 846—869 / Nach dem Autograph und Abschriften hrsg. von W. Dehnhard. Wien: Wiener Urtext Edition, 1977.

[6] Johann Sebastian Bach. Das Wohltemperierte Klavier II / hrsg. von W. Dehnhard. Wien: Wiener Urtext Edition, 1983.

[7] Vishnevsky, A. V. “The neural scheme of an electronic composer”. *Electronics and control systems*. 2013, no. 1 (35), pp. 107–110.

Received 06 February 2015.

Vishnevsky Alexander. Ph.D. Associate professor.
Institute of Aeronavigation, National Aviation University, Kyiv, Ukraine.
Education: Kiev International University of Civil Aviation, Kyiv, Ukraine (1995).
Research area: information processing
Publications: 32.
E-mail: av@nau.edu.ua

О. В. Вишнівський. Бінарна вагова матриця алгоритму електронного композитора

Запропоновано бінарну реалізацію логічної структури, яку можна представити у вигляді вагової матриці нейромережевого електронного композитора.

Ключові слова: цифрова музика; електронний композитор; нейронна мережа; обробка інформації; штучний інтелект.

Вишнівський Олександр Володимирович. Кандидат технічних наук. Доцент.
Інститутут аеронавігації, Національний Авіаційний Університет, Київ, Україна.
Освіта: Київський міжнародний університет цивільної авіації, Київ, Україна (1995).
Напрямок наукової діяльності: обробка інформації
Кількість публікацій 32.
E-mail: av@nau.edu.ua

А. В. Вишневский. Бинарная весовая матрица алгоритма электронного композитора

Предложена бинарная реализация логической структуры, которую можно представить в виде весовой матрицы нейросетевого электронного композитора.

Ключевые слова: цифровая музыка; электронный композитор; нейронная сеть; обработка информации; искусственный интеллект.

Вишневский Александр Владимирович. Кандидат технических наук. Доцент.
Институтут информационных диагностических систем, Национальный Авиационный Университета, Киев, Украина.
Образование: Киевский международный университет гражданской авиации, Киев, Украина (1995).
Направление научной деятельности: обработка информации.
Количество публикаций: 32.
E-mail: av@nau.edu.ua