Teaching to computer based on neural network model

A. T. TIMAJI
Neshan-Kaveh–Ahangar Company
#3, Lali St. 25th St. Gesha Ave. Tehran, Iran.

Z. HEIDARY
Naft-Iran Company
#83, Zafar St. Shareati Ave. Tehran, Iran.

B. TAFAGHODINIA
Iranian Research Organization for Science and Technology
#71, Forsat St., Engelab Ave., Theran, Iran. P.O. Box: 15815-3538

Abstract: However digit base in computer is binary and their recognition by 1 and 0 is regain able, but nowadays by using new and advanced programming and pattern' making can teach computer with an algorithm [5]. So that from neuron model can be used in Hopfield neural network. Here, through language of, programming we observe a sample of teaching and as a result. This kind of teaching will result in new field in optimization of artificial intelligence in robot and humanoids. In this model and algorithm, computer without using computer base software can certainly recognize language, words and alphabets.

Key-Words: - Artificial Intelligence, Neural network,

1 Introduction

Artificial Intelligence is a branch of Science which deals with helping machines find solutions to complex problems in a more human-like fashion [1]. This generally involves borrowing characteristics from human intelligence, and applying them as algorithms in a computer friendly way [4] [6]. A more or less flexible or efficient approach can be taken depending on the requirements established, which influences how artificial the intelligent behavior appears.

Artificial intelligence is generally associated with Computer Science, but it has many important links with other fields such as Mathematics, Psychology, Cognition, Biology and Philosophy, among many others [2]. Our ability to combine knowledge from all these fields will ultimately benefit our progress in the quest of creating an intelligent artificial being.

However digit base in computer is binary and their recognition by 1 and 0 is regain able, but nowadays by using new and advanced programming and pattern making can teach computer with an algorithm [5]. So that from neuron model can be used in Hopfield neural network [3]. Here, through language of, programming we observe a sample of teaching and as a result. This kind of training will result

in new field in optimization of artificial intelligence in robot and humanoids. In this model and algorithm, computer without using computer base software can certainly recognize language, words and alphabets.

2 Materials and Methods

The algorithm of program includes 4 stages:

1- Based on neurons of brain neural network, here also is replaced instead of neurons and specific weight is not considered for them. Assume in puts aver X_0, X_1 ,…, X_{m+1} and T_{ij} is weight of function. In this stage the weight of fasteners is specified. If i!=j → $\sum X_{is}$, $\sum X_{js}$ if i=j → (i>=0 , j<=m-1) or i=j=0 s=0, m-1 and j and I will be the weight of fastener knot of I to knot of j. X_{is}, is the sample element of I related to s group and also. X_{js}, is the sample element of j related to s group. In this case, if inputs are in the shape of $X_0,X_1,\ldots,X_{m+1}$, out puts will be $x'_0,x'_1,\ldots,x'_{m+1}$.

2- Second stage is specifying primary numbers by using unknown input model. Whit assu1ption $\mu_i(0)=X_i$ that i=0, …,m-1
network includes m group. Inputs will be only +1,-1. Here $\mu_i(t)$ is the output of knot and I is in the time of t. and X_i that can be +1 to -1 is of i element of input model.

3- Repetition until reaching to permanent.
$\mu_i(t+1)=f_h[\sum T_{ij} *\mu_i(t)]$
that i=0,…,m-1 and t=0,… , j=0,…,m-1
f_h function is one of useable know function in Hopfield net works and stages are repeated till outputs of knot express the sample which are most similar to unknown input.

4- Repeat from step 2

Text of program (C++)

```
#include <stdlib.h>
#include <stdio.h>

#define FALSE    0
#define TRUE     1

#define LOW      1
#define HIGH     +1

#define BINARY(X) ((X)==LOW ? FALSE : TRUE)
#define TWOWAY(X) ((X)==FALSE ? LOW: HIGH)

typedef struct {
/*OUR NET:*/
int Units;
/*- number of terms in this net*/
int* Output;
/*- output of term i*/
int* Threshold;
/*-threshold of term j*/
int** Weight;
/*-connection weights to term i*/
}NET;

/****RANDOMS DRAWN FROM DISTRIBUTIONS ****/

void InitializeRandoms()
{
srand(4711);}
int RandomEquaLINT(int LOW,int HIGH) {
return rand() %(HIGH-LOW+1)+LOW; }

/****SET UP OUR ASSUMPTION ****/
#define NoOfData  7
#define X         10
#define Y         10
#define N       (X*Y) /*100*/

char Pattern[NoOfData][Y][X]={
{
"   000000   ",
" 00000000 ",
"          00 ",
"          00 ",
"  000000    ",
"  000000    ",
"          00 ",
"          00 ",
" 00000000 ",
"  000000    "
},
```

```
{

"   000000   ",
"00000000   ",
"00      000  ",
"         000  ",
"        000   ",
"       000    ",
"     000      ",
"  000         ",
"000000000 ",
"000000000 "
},

{
"000000000 ",
"000000000 ",
"000           ",
"000           ",
"00000000   ",
"     000000 ",
"          000",
"          000",
"00       000 ",
"  000000    "
},

{
"    000    ",
"  0000    ",
"    000    ",
"    000    ",
"    000    ",
"    000    ",
"    000    ",
"    000    ",
"    000    ",
"  00000   "
},

{
"  00000000  ",
"0000000000",
"00              ",
"00              ",
" 000000000 ",
"0000000000",
"00           00",
"00           00",
"0000000000",
"  00000000 "
},

{
"0000000000",
"0000000000",
"00          000",
"            000 ",
"           000  ",
"          000   ",
"         000    ",
"        000     ",
"       000      ",
"       000     "
},

{
"  00000000  ",
"0000000000",
"00           00",
"00           00",
"  00000000  ",
"0000000000",
"00           00",
"00           00",
"0000000000",
"  00000000  "
}};

int Input[NoOfData][N];
FILE*  f;
/**** INITIALISE NETWORK VALUES ****/

void LetsInitialise(NET* Net){
int n,i,j;
for (n=0;n<NoOfData;n++){
for (i=0;i<Y;i++){
for(j=0; j<X; j++){
Input [n][i*X+j]=TWOWAY(Pattern [n][i][j]=='0');
}}}
f=fopen("HOPOUT.txt","W");
}}

/**** WRITE RESULT TO FILE ****/
void WriteNet(NET*Net){
int i,j;
for (i=0;i<Y;i++){
for(j=0;j<X;j++){
fprintf(f,"%c",BINARY(Net->Output[i*X+j])  ?  '0'  :
");}
fprintf(f,"\n");}
fprintf(f,"\n");}

/*************************/

void LetsClose(NET*Net){
```

```
fclose(f);}

/**** LETS SETUP OUR DATA STRUCTURE ****/

void MakeNet(NET*Net){
int i;
Net->Units=N;
Net->Output=(int*) calloc(N,sizeof(int));
Net->Threshold=(int*) calloc(N,sizeof(int));
Net->Weight=(int**) calloc(N,sizeof(int*));
for(i=0;i<N;i++){
Net->Threshold[i] =0;
Net->Weight[i]=(int*)calloc(N,sizeof(int));}}

/**** CALCULATION OF CONNECTION WEIGHTS ****/

void CalculateWeights(NET*Net){
int i,j,n;
int Weight;
for(i=0;i<Net->Units;i++){
for(j=0;j<Net->Units;j++){
Weight=0;
if(i!=j){
for(n=0;n<NoOfData;n++){
Weight+=Input[n][i]*Input[n][j];}}
Net->Weight[i][j]=Weight;
}}}

/**** USE OUTPUT AS NEXT INPUT ****/
void SetInput(NET*Net,int*Input){
int i;
for(i=0;i<Net->Units;i++){
Net->Output[i]=Input[i];}
WriteNet(Net);}

/**** GET OUTPUT FROM NETWORK ****/
void GetOutput(NET*Net, int*Output){
int i;
for(i=0; i<Net->Units;i++){
Output[i]=Net->Output[i];}
WriteNet(Net);}

/**** PROCESS OUR SIGNALS ****/
int Sigproc(NET*Net,int i){
int j;
int sum, Out;
int changed;
changed=FALSE;
sum=0;
for(j=0;j<Net->Weight[i][j]* Net->Output[j];{
}
if (sum!=Net->Threshold[i])
{
if (sum < Net->Threshold[i]) Out=LOW;
if (sum > Net->Threshold[i]) Out=HIGH;
if (Out!=Net->Output[i]){
changed=TRUE;
Net->Output[i]=Out;}
return changed;}

/**** PROCESS THE NETWORK ****/
void Netproc(NET*Net){
int Repeat, RepeatOfLastChange;
Repeat=0;
RepeatOfLastChange=0;
do{
Repeat++;
if (Sigproc(Net, RandomEquaLINT(0,Net->Units-1)))
RepeatOfLastChange=Repeat;}
while(Repeat-RepeatOfLastChange<10*Net->Units);}

/**** SIMULATING THE NET ****/
void SimulateNet(NET*Net, int* Input){
int Output[N];
SetInput(Net,Input);
Netproc(Net);
GetOutput(Net,Output);}

/**** MAIN ****/
void main(){
NET Net;
int n;
InitializeRandoms();
MakeNet(&Net);
LetsInitialise(&Net);
CalculateWeights(&Net);
for (n=0;n<NoOfData;n+1){
SimulateNet(&Net, Input[n]);}
LetsClose(&Net);}
```

3 Results and Conclusion

Program Input:

```
"   000000   ",
" 00000000 ",
"          00 ",
"          00 ",
```

```
"  000000   ",
"  000000   ",
"         00 ",
"         00 ",
" 00000000 ",
"  000000   "
```

Program Output:
exit:

```
"   000000  ",
" 00000000 ",
" 00      00 ",
" 00      00 ",
"  000000   ",
"  000000   ",
"         00 ",
"         00 ",
" 00000000 ",
"  000000   "
```

Numerating of the machine happens with numbers that have extreme digits and as a result a lot of calculations are done by approximate presentation of this number in today's computer. In today's computer for presentation of all the real numbers only a small relative subset of set of real number is used. This subset only includes rational number either positive or negative and stores a fixed point part by the name (title) of mantis along with an indicial point part by the name (title) of specifications for example, a number with usual care and with a lot of decimal point which used in the series of 360 or 370 is compound from a 24 digit binary and a 7 digit medical binary with standard of 16. Since 24 binary digits are isomorphic of 7 to 8 decimal number, it is assume that IBM series of 360 or 370 for set with exact decimals have at least 7 decimal digits. Presentation of 7 binary digit shows range of 0 to 127.
But be cause of

0	1000010	10110011000001000000000

$+((1/2)^1+(1/2)^3+(1/2)^4+(1/2)^7+(1/2)^8+(1/2)^{24})*16^{66}$=179.015625range of -64 to +63, namely 64 automatically is deducted from positive presentation mechanical number is the exact presentation of because the first binary digit is the presentation of a number. 0 for + and 1 for – of 7 binary digit after presentation of …. And 24 binary digit of the end of presentation of mantis in fact is used for in [179.01561737060546875,179.01563262939453125] in this presentation the least positive number is 16^{63} =10^{75} and the biggest number is 1063=1075. +/- $d_1d_2\ldots d_{24}*16$ the number less than 16^{-64} results to under flow and after assume there zero and number bigger than 1663 results to over flow and in conclusion stops the 0 calculations. +/- $d_1d_2\ldots d_k*10^n$ $1<=d_1<=9$, i=2,…,k which is -77<=n<=75 , k=7 ibm and whenever p* is approximate of p, implicit error equivalent to |p-p*| and proportional error equivalent to p-p*/|p| on the condition that p=0 can be considered. The probility of error can be calculated by idiographic the ory. Whenever p of idiotrapie compound is from extreme grade of n with kots in Tn+1, in that case for each max|f(x)-p(x)|<=$1/2^{n(n+1)}$! Max|f^{n+1}(x)| that f€c^{n+1}[-1,1] , x€[-1,1] of compound of Chipchof can be used for decreeing the grade of one scant making compound with the least loss in care.

References:

1. T. S. Bellows, R. G. Vandriesche, and J. S. Elkinton, Life table analysis construction and analysis in the evaluation of natural enemies, Ann. Rev. Entomol. 37: 587-614 (1992)
2. P. Jackson , Introduction to Expert Systems, Addison-Wesley, (1999)
3. S. Holtzman, Intelligent Decision Systems, Addison-Wesley,(1989)
4. A. Blum, Neural Networks Programming in C++: An Object-Oriented Framework for Building Connectionist Systems, John Wiley, (1992)
5. Edward R. Dougherty, Charles Robert Giardina, Mathematical Methods for Artificial Intelligence and Autonomous Systems, Prentice Hall, (1988)
6. James A. Freeman, David M. Skapura, Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, (1991)