

Computational Physics

Project 9: The Neural Network

Michael Ryan

May 25, 1999

Abstract

One of the more interesting software developments in the past 20 years has been the neural network. Unlike most of the projects in this course the neural network is not approximating a purely mathematical method for use in the limited frames of a computer's abilities. The approximation derives from an entirely organic foundation. It was found that neurons could be abstractly thought of in terms of existing only in an on or off state, much like the spins in the Ising model. These 'on or off' states of the one neuron affects all the other neurons to which it's connected. The flow of the 'on or off' states are governed by a configuration parameter which essentially stores information stored and recalled by the neurons. These neural networks provided a way of collecting, storing, and analyzing information by using only software and not having to mess with the addresses and storage hardware of the computer. Not suprising this has become a very useful tool in the analysis of large amounts of data for picking out snippets of information that might overwhelm our own brains.

1 Procedure and Equations

1.1 The Neural Networks and the Ising Model.

The simple model of a neural network that I shall use here is very closely related to the Ising model. In fact, the only real difference between my neural network and the Ising model is the fact that the neural network 'spins' are summed over all the 'spins' and not over only the nearest neighbors like in

the Ising model. Basically, for some number of spins, N , in configuration number m one can find some inherent energy of the system.

$$E(m) = - \sum_{i=1}^N \sum_{j=1}^N J_{ij}(m) \sigma_i \sigma_j \quad (1)$$

where $J_{ij}(m)$ is called the configuration of the system in m . In this the σ 's contain in spin up or down form the input to be analyzed by the program. The $J_{ij}(m)$ contain the previously stored information. The *sigma*'s eventually should become the desired output of the program as commanded by $J_{ij}(m)$. Also note that there are no differences between the σ_i 's and σ_j 's except that they are summed separately. This energy is the goal of the program. What the program attempts to do is to minimize the energy so that the computers $E_c(m)$ matches the ideal energy $E(m)$. The key here is of course $J_{ij}(m)$. There are numerous configuration parameters that can be chosen to accomplish any given task, but the simplest ones are most often the most desired. So if I let

$$J_{ij}(m) = \sigma_i(m) \sigma_j(m) \quad (2)$$

where $\sigma_i(m)$ and σ_j are the i th and j th spin state of the m th configuration, one can see that with *up* = 1 and *down* = -1 I can use the fact that I will get $(-1)^{2n}$ where n is some integer when my input's spins match the configuration's spins. With $E(m)$ being equal to the negative of the sum my energy is thus minimized every time I have matched spins. For all M configurations together I get a total energy of

$$E = - \frac{1}{M} \sum_{m=1}^M \sum_{i=1}^N \sum_{j=1}^N J_{ij}(m) \sigma_i \sigma_j. \quad (3)$$

The goal is then to minimize this energy so that equilibrium points $E(m)$ fall out. I should mention that ideally all the configurations would be orthogonal to one another but that is not always the case. Similar configurations can obviously confuse the program (as you'll see later).

1.2 The Program

So, how to use this. I'll use the Monte Carlo method to skip through the spins measuring the energy at that point and changing it if it can be minimized. Here are the steps:

1. Choose a random number $x \in [0, 1]$.
2. If $x > 0.5$ then continue on to the next spin σ_{j+1} leaving σ_j alone.
3. Otherwise if $x \leq 0.5$ then check ΔE with the j th spin flipped:

$$\Delta E = -\frac{1}{M} \sum_{m=1}^M \sum_{i=1}^N J_{ij}(m) \sigma_i (-\sigma_j - \sigma_j) \quad (4)$$

which can be simplified to

$$\Delta E = \frac{2}{M} \sum_{m=1}^M \sum_{i=1}^N J_{ij}(m) \sigma_i \sigma_j \quad (5)$$

4. If this value is less than zero then set $\sigma_j = -\sigma_j$.
5. If not, then leave σ_j alone and continue on from the top for $j + 1$.

Once each of these N spins has been checked then one Monte Carlo sweep has finished.

What will I be measuring? I am going to input 5 configurations of letters drawn on a 10×10 matrix: a, i, o, x, and h. Please note that the 100 spins are not counted on a matrix but simply placed in order into the matrix counting by columns first and then rows. Here is how they look when read by the program.

'a'

```

- - - - + + - - - -
- - - + + + + - - -
- - + + - - + + - -
- + + - - - - + + -
- + + + + + + + -
+ + + + + + + + +
+ + + - - - - + + +
+ + - - - - - + +
+ + - - - - - + +
+ + - - - - - + +

```

'h'

```

+ + - - - - - + +
+ + - - - - - + +
+ + - - - - - + +
+ + - - - - - + +
+ + + + + + + + +
+ + + + + + + + +
+ + - - - - - + +
+ + - - - - - + +
+ + - - - - - + +
+ + - - - - - + +

```

'1'

```

- - - + + + + - - -
- - - + + + + - - -
- - - - + + - - - -
- - - - + + - - - -
- - - - + + - - - -
- - - - + + - - - -
- - - - + + - - - -
- - - - + + - - - -
- - - + + + + - - -
- - - + + + + - - -

```

'0'

```

- - + + + + + - -
- + + + + + + + -
+ + + - - - - + + +
+ + - - - - - + +
+ + - - - - - + +
+ + - - - - - + +
+ + - - - - - + +
+ + + - - - - + + +
- + + + + + + + -
- - + + + + + - -

```

'X'

```

+ + - - - - - + +

```

```

- + + - - - - + + -
- - + + - - + + - -
- - - + + + + - - -
- - - - + + - - - -
- - - - + + - - - -
- - - + + + + - - -
- - + + - - + + - -
- + + - - - - + + -
+ + - - - - - - + +

```

where '+' symbolizes 1 and '-' symbolizes -1.

1.3 Measurements.

My input can be distorted by selecting the number of spins of the selected letter subject to random fluctuations. So if 20 percent is selected for 'a' then 'a' will have 20 percent of its spins randomized. This way I can measure the effectiveness of each configuration in being successfully analyzed by the program.

For 'a' my program's ability to resolve the letter stopped at around 31 percent.

Input:

```

- - + - + + - + - +
+ - + + + + + - + -
+ + - + - + + + - -
- + - - + - - + + -
- + + + + + + + + -
+ - - + - - + + + -
+ + + + + - - + + +
+ + - + - + + + + +
+ - - - - - - - + +
+ + + - - + - + + +

```

Output:

```

- - - - + + - - - -
- - - + + + + - - -
- - + + - - + + - -

```

```

- + + - - - - + + -
- + + + + + + + + -
+ + + + + + + + + +
+ + + - - - - + + +
+ + - - - - - - + +
+ + - - - - - - + +
+ + - - - - - - + +

```

For 'i' my program stopped resolving the letter at 15 percent.
Input:

```

- - - + + + + - - -
+ - + + + + + - - -
- - - - + + - - - -
- - + - - + - - - -
- - - - + + - - - -
- + - - - - - - - -
- - - - - + - - - -
- - - - + + - + - -
- + - + + + + - - -
- - + + + - + - - -

```

Output:

```

- - - + + + + - - -
- - - + + + + - - -
- - - - + + - - - -
- - - - + + - - - -
- - - - + + - - - -
- - - - + + - - - -
- - - - + + - - - -
- - - - + + - - - -
- - - + + + + - - -
- - - + + + + - - -

```

For 'o' the program crapped out at 49 percent.
Input:

```

+ + - - + + + - - +
+ + - - - - + + - +

```

```

- - - - - + + + + +
+ + + - + - - - + +
+ + - + - - - - + +
+ - + - + + + - + -
+ - - + + - - - - +
+ + - + - + + - + +
- - + - + + + + - +
- - - - + - - - + -

```

Output:

```

- - + + + + + - -
- + + + + + + + -
+ + + - - - - + + +
+ + - - - - - + +
+ + - - - - - + +
+ + - - - - - + +
+ + - - - - - + +
+ + + - - - - + + +
- + + + + + + + -
- - + + + + + - -

```

'x' stopped at 36 percent.

Input:

```

- + + - - - - + + -
+ + - + + - - + - +
+ + - + - + + + - -
- - + + - + + - - -
- - - + + + - - - -
- + + - - - - - +
- - - - - + + - - -
- - - - - + - - - -
- - + - - - - + + +
+ + + - - + + + + +

```

Output:

```

+ + - - - - - + +
- + + - - - - + + -

```

```

- - + + - - + + - -
- - - + + + + - - -
- - - - + + - - - -
- - - - + + - - - -
- - - + + + + - - -
- - + + - - + + - -
- + + - - - - + + -
+ + - - - - - - + +

```

'h' was an odd case. I found that its configuration was too close to those of 'a' and 'x' for the program to resolve it at any percentage.

Input:

```

+ + - - - - - - + +
+ + - - - - - - + +
+ + - - - - - - + +
+ + - - - - - - + +
+ + + + + + + + + +
+ + + + + + + + + +
+ + - - - - - - + +
+ + - - - - - - + +
+ + - - - - - - + +
+ + - - - - - - + +

```

Output (with zero distortion):

```

+ + - - - - - - + +
+ + - - - - - - + +
+ + - + - - + - + +
+ + + - - - - + + +
+ + + + + + + + + +
+ + + + + + + + + +
+ + + - - - - + + +
+ + - - - - - - + +
+ + - - - - - - + +
+ + - - - - - - + +

```

2 Conclusion

Overall the results went well. The program was able to resolve letters out of patterns that were very hard for me to even identify. The one weak point was obviously 'h'. Perhaps with more spins 'h' could be identified by the program. As it was 'h' was too close between 'a' and 'x' that the program would confuse the two as shown in the output above. One thing that could strongly affect the results for these programs is the psuedo-random code generator. There are certain spins which are integral to identifying the letters. If those spins get changed the program's job becomes much more difficult. Should the random number generator on another computer be slightly different the resulting change will affect which spins are randomized. Thus resolution rates should vary quite a bit on different platforms.

3 Appendices: The Programs

Letter Creation

This program writes and randomizes the letters one chooses. It also writes a fresh list of configurations for $J_{ij}(m)$

```
#include <iostream>
using namespace std; //introduces namespace std
#include <cstdlib>
#include <fstream>
#include <cmath>

int m ;
char inp ;
double percent ;
int let[10][10][6] ;

class Letter
{
private:
struct storage
{
        int pattern1[10];
```

```

        int pattern2[10];
        int updown;
    };
public:
    void a();
    void h();
    void i();
    void o();
    void x();
    void Construction( storage out );
    void alphabet();
    void output();
};

void Letter::alphabet( )
{
    Letter call;

    switch( inp )
    {
        case 'a':
        case 'A':
            call.a();
            break;

        case 'h':
        case 'H':
            call.h();
            break;

        case 'i':
        case 'I':
            call.i();
            break;

        case 'o':
        case 'O':
            call.o();
            break;

        case 'x':
        case 'X':

```

```

        call.x();
        break;
    }
}

void Letter::a()
{
    storage a1 =
    {
        {4, 3, 2, 1, 1, 5, 3, 2, 2, 2},
        {1, 2, 2, 2, 2, 4, 0, 2, 3, 3, 3},
        1
    };
    Construction( a1 );
}

void Letter::h()
{
    storage h1 =
    {
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
        {2, 2, 2, 2, 5, 0, 0, 0, 0, 0},
        0
    };
    Construction( h1 );
}

void Letter::i()
{
    storage i1 =
    {
        {3, 3, 4, 4, 4, 0, 0, 0, 0, 0},
        {2, 2, 1, 1, 1, 0, 0, 0, 0, 0},
        0
    };
    Construction( i1 );
}

```

```

void Letter::o()
{
    storage o1 =
    {
        {2, 1, 0, 0, 0, 0, 0, 0, 0, 0 },
        {3, 4, 3, 2, 2, 0, 0, 0, 0, 0 },
        0
    };
    Construction( o1 );
}

void Letter::x()
{
    storage x1 =
    {
        {0, 1, 2, 3, 4, 0, 0, 0, 0, 0 },
        {2, 2, 2, 2, 1, 0, 0, 0, 0, 0 },
        0
    };
    Construction( x1 );
}

void Letter::Construction( storage out )
{
    int i, j, symmetry = 5 ;
    int up, down ;

    if( out.updown != 0 )
        symmetry = 10 ;

    for(i = 0; i < symmetry; i++)
    {
        for( j = 0; j < 5; j++)
        {

            if( i < 5 || out.updown == 0 )
            {
                up = 1;

```

```

        down = -1;
    }
    else if( out.updown == 1 )
    {
        up = -1;
        down = 1;
    }

    if( j < out.pattern1[i] )
        let[i][j][m] = down ;
    else
    {
        if( j < out.pattern1[i] + out.pattern2[i] )
            let[i][j][m] = up ;
        else
            let[i][j][m] = down ;
    }

    if( out.updown != 1 )
    {
        let[9-i][j][m] = let[i][j][m] ;
        let[9-i][9-j][m] = let[9-i][j][m] ;
    }

    let[i][9-j][m] = let[i][j][m] ;

    }

}

void Letter::output()
{
    ofstream fout1("J.dat");
    ofstream fout2("sigma.dat");
    double x ;

```

```

int i, j ;

cout << "\nOutput:\n";

for( m = 0; m < 6; m++ )
    for( i = 0; i < 10; i++ )
    {
        for( j = 0; j < 10; j++ )
        {
            if( m < 5 )
            {
                if( let[i][j][m] == 1 )
                    fout1 << "+\n" ;
                else
                    fout1 << "-\n" ;
            }
            else
            {
                x = drand48();

                if( x > percent )
                {
                    if( let[i][j][m] == 1 )
                    {
                        fout2 << "+\n" ;
                        cout << "+ " ;
                    }
                    else
                    {
                        fout2 << "-\n" ;
                        cout << "- " ;
                    }
                }
            }
            else
            {
                if( let[i][j][m] == 1 )

```

```

        {
            fout2 << "-\n" ;
            cout << "- " ;
        }
        else
        {
            fout2 << "+\n" ;
            cout << "+ " ;
        }
    }

    if( j == 9 )
        cout << "\n" ;
}

}

}

fout1.close() ;
fout2.close() ;
}

main()
{
    Letter step;
    int iseed = 25969 ;
    int flag=0;

    srand48(iseed) ;

    m = 0 ;
    step.a();
    m = 1 ;
    step.h();
    m = 2 ;
    step.i();
    m = 3 ;
    step.o();
    m = 4 ;
}

```

```

        step.x();

        m = 5;

        do{
                cout << "Enter the letter you wish to have identified and
randomized ( a, h, i, o, x). \n" ;
                cin >> inp ;

                if( inp != 'a' && inp != 'h' && inp != 'i' && inp != 'o' && inp
                if( inp != 'A' && inp != 'H' && inp != 'I' && inp != 'O
'X' )

                        flag = 1;

        } while( flag == 1 );

        step.alphabet();

        flag = 1;

        do{
                cout << "Enter the percentage of the spins you wish subject to
randomization. \n" ;
                cin >> percent ;

        } while( percent > 100 || percent < 0 );

        percent /= 100.0 ;

        m = 0;
        step.output( );
}

```

The Brain

This program is the neural network that reads file “J.dat” for its configuration and “sigma.dat” for its input.

```
#include <iostream>
```

```

using namespace std; //introduces namespace std
#include <cstdlib>
#include <fstream>
#include <cmath>

class Brain
{
    private:
        int sigma[100] ;
        int J[100][100][5] ;
    public:
        void input();
        void output();
        int calc();
};

void Brain::input()
{
    int i, j, m, itemp[100];
    char ch1, ch2, chtemp ;
    ifstream fin1("J.dat") ;
    ifstream fin2("sigma.dat") ;

    for( m = 0; m < 5; m++ )
    {

        for( i = 0; i < 100; i++ )
        {
            fin1 >> ch1 ;

            if( ch1 == '+' )
                itemp[i] = 1;
            else if( ch1 == '-' )
                itemp[i] = -1;
            else
            {
                cout << "There is a problem with the data file"
                break;
            }
        }
    }
}

```

```

    }

    if( m == 0 )
    {
        fin2 >> ch2 ;

        if( ch2 == '+' )
            sigma[i] = 1;
        else if( ch2 == '-' )
            sigma[i] = -1;
        else
        {
            cout << "There is a problem with the
file.\n";

            break;
        }
    }

    for( i = 0; i < 100; i++ )
        for( j = 0; j < 100; j++ )
            J[i][j][m] = itemp[i]*itemp[j];
}

void Brain::output()
{
    int i ;
    ofstream fout("Identified.dat");

    cout << "\nIdentified as:\n";

    for( i = 0; i < 100; i++ )
    {
        if( sigma[i] == 1 )
        {
            fout << "+ " ;
            cout << "+ " ;

```

```

        }
        else
        {
            fout << "- " ;
            cout << "- " ;
        }

        if( ((i+1) % 10) == 0 )
        {
            fout << "\n" ;
            cout << "\n" ;
        }

    }

    fout.close();
}

int Brain::calc()
{
    int i, j, m ;
    double temp1 = 0;
    double temp2 = 0;
    double deltaE ;
    double x;

    for( j = 0; j < 100; j++ )
    {
        temp1 = temp2 = 0 ;
        x = drand48();

        if( x <= 0.5 )
        {
            for( i = 0; i < 100; i++ )
            {
                if( i != j )
                {
                    for( m = 0; m < 5; m++ )

```

```

                                {
                                temp1 +=
(double)(J[i][j][m]*sigma[i])/5.0;
                                temp2 +=
(double)(J[j][i][m]*sigma[i])/5.0;
                                }
                                }

                                deltaE = ((2*temp1) + (2*temp2))*(double)sigma[j];

                                if( deltaE < 0 )
                                    sigma[j] = -sigma[j];
                                }
                                }

                                return 0;
                                }

int main()
{
    int i;
    int iseed = 27498;

    srand48(iseed);

    Brain step ;
    step.input();

    for( i = 0; i < 1000; i++ )
        step.calc();

    step.output();

    return 0;
}

```