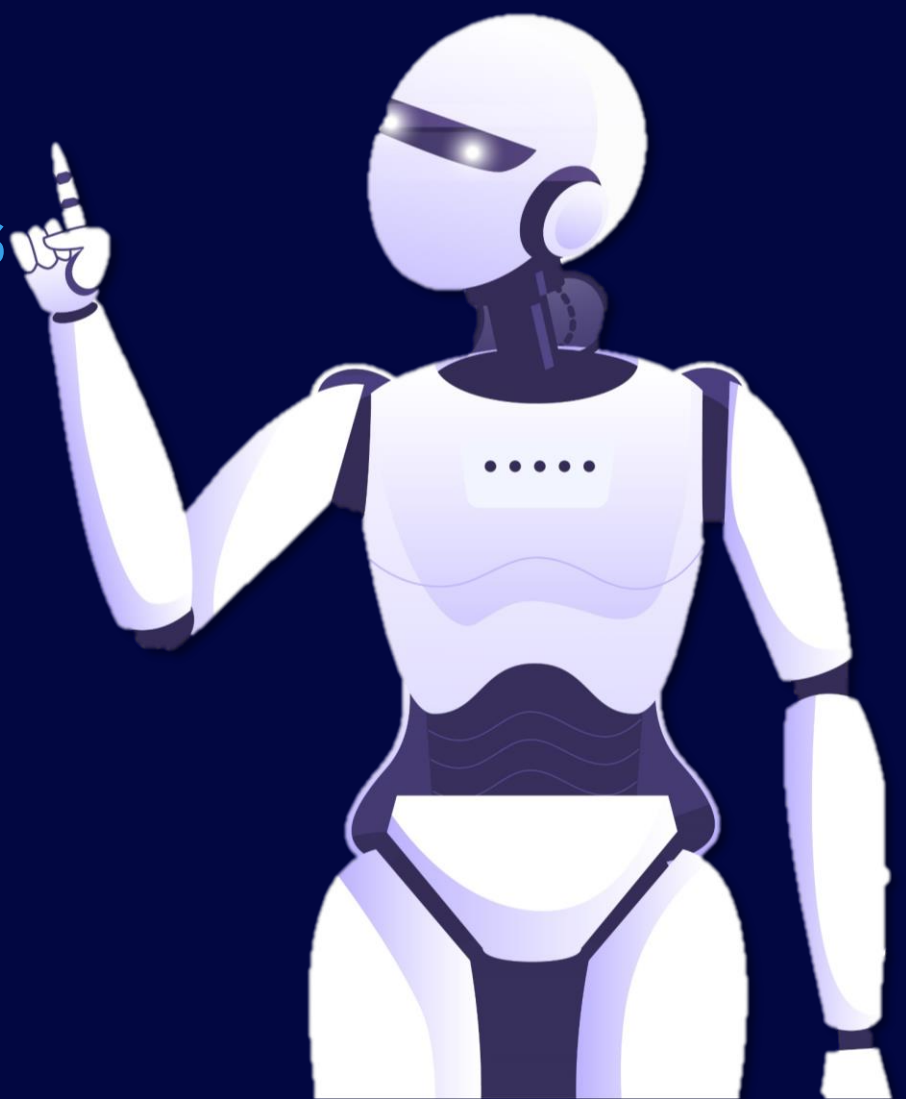
artificial
intelligence

NEURAL NETWORKS LABORATORY PROJECT

Antonio Trujillo
Alberto Trigueros
Jesús Fuentes
María Peinado
Group C

INDEX

1. Introduction
2. Structure of the application, design choices
3. Experimental results, performance slots, coments



INTRODUCTION

This project is focused on creating a function with the ability of learning. The algorithm should gain knowledge by using several training patterns and a multilayer perceptron neural network. In the following slides, we are going to explain how we got that.

STRUCTURE OF THE APPLICATION, DESIGN CHOICES

We generate random value between $-\pi$ and π

We apply the function given

We create a Matrix to store the random values (to lines for x and y)

We create a Grid were the first column is for values x, the second one is for y.

```java
public class Generate {
    public static double valorRandom() {
        return (Math.random() * 2 *Math.PI) - Math.PI;
    }
    public static double generaF(double x, double y) {
        return Math.sin(x) * Math.cos(y);
    }
    public static double[] [] generaMatrix(){
        double [] [] res = new double [1000] [2];
        for(int i = 0; i < 1000; i++) {
            for(int j = 0; j < 2; j++) {
                res[i][j] = valorRandom();
            }
        }
        return res;
    }
    public static double [][] generagrid(){
        double [][]res = new double [10000] [2];
        double k = 2 * Math.PI/99;
        for(int i = 0; i < 10000; i++) {
            res[i][0] = -Math.PI + 2 * (Math.PI)*((double)(i%100)/(double) (99));
            res[i][1] = -Math.PI + 2 * (Math.PI)*((double)(i/100)/(double) (99));
        }

        return res;
    }
}
```

STRUCTURE OF THE APPLICATION, DESIGN CHOICES

We generate the training and the validation matrix using the class Generate we have just created. And also the matrix for the results for each of them

Applying F (generaF) we obtain the values of the results for the training and the validation Matrix

We create the network

For the first layer we have un neuron for each input. After trying diferents values we choose 14 neurons for the second layer
Finally, the last layer for the output

We start with the training and the validation

```java
public class main {
    public static void main (String[] args) throws FileNotFoundException {
        PrintWriter pw = new PrintWriter("output.txt");


        double[][] t = Generate.generaMatrix();
        double[][] val = Generate.generaMatrix();
        double[][] tRes = new double[1000][1];
        double[][] valRes = new double[1000][1];

        for (int i=0; i<1000; i++) {
            tRes[i][0] = Generate.generaF(t[i][0], t[i][1]);
            valRes[i][0] = Generate.generaF(val[i][0], val[i][1]);
        }

        BasicNetwork net = new BasicNetwork();

        net.addLayer(new BasicLayer(new ActivationTANH(), true, 2));
        net.addLayer(new BasicLayer(new ActivationTANH(), true, 14));
        net.addLayer(new BasicLayer(new ActivationTANH(), true, 1));
        net.getStructure().finalizeStructure();
        net.reset();

        MLDataSet trainSet = new BasicMLDataSet(t, tRes);
        MLDataSet validationSet = new BasicMLDataSet(val, valRes);

        final ResilientPropagation train = new ResilientPropagation(net, trainSet

        int epoch = 1;
```

We have choosen activationTANH because it was the one which best fitted with our algorithm

STRUCTURE OF THE APPLICATION, DESIGN CHOICES

```
    do {
        train.iteration();
        pw.append("Epoch #" + epoch + " TrainingError:" + train.getError() + "\n");
        System.out.println("Epoch #" + epoch + " TrainingError:" + train.getError());

        pw.append("Epoch #" + epoch + " ValidationError:" +  MSE(net, validationSet) + "\n");
        System.out.println("Epoch #" + epoch + " ValidationError:" +  MSE(net, validationSet));

        epoch++;
    } while (train.getError() > 0.01);

    train.finishTraining();

    double [][] tsamp = Generate.generagrid();
    double [][] tsampRes = new double[10000][1];
    for (int i=0; i<10000; i++) {
        tsampRes[i][0] = Generate.generaF(tsamp[i][0], tsamp[i][1]);
    }
    MLDataSet tsampSet = new BasicMLDataSet(tsamp, tsampRes);
    System.out.println("The mean squared error for the test samples is: " + MSE(net, tsampSet)/10);
    pw.append("The mean squared error for the test samples is: " + MSE(net, tsampSet)/10);
    pw.close();

    Encog.getInstance().shutdown();

}
```

We set up the error in 0,01 for obtaining a better results, so it started training until it get an error lower than the established.

Now we are using the grid. *tsamp* contain the coordinates, whereas the *tsampRes* contain the F result.
Finally, we compute the mean cuadratic error of the test simples with the same algorithm used before but now dividing the result by 10 because we have 10000 simples now.

For calcuting the validation error = MCE, we have added consecutively the square of the difference between the ideal result and the obtained for all values gotten.

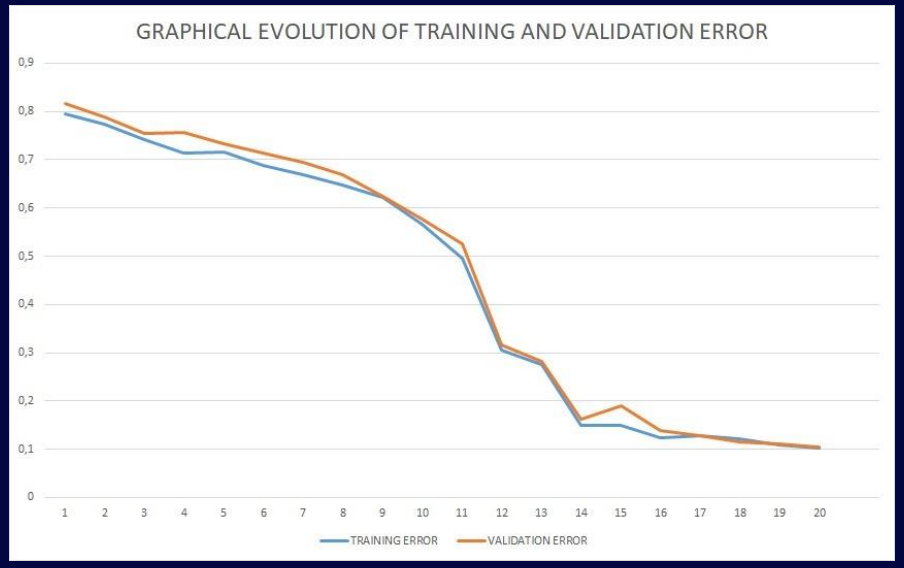
```
public static double MSE(BasicNetwork net, MLDataSet Set) {
    double error = 0;
    for (MLDataPair pair: Set) {
        final MLData output = net.compute(pair.getInput());
        error += Math.pow(output.getData(0) - pair.getIdeal().getData(0), 2);
    }

    return error/1000;
}
```

EXPERIMENTAL RESULTS, PERFORMANCE SLOTS, COMENTS

As we can see, while the epoch number increases, the error decreases, in training as in validation, which means that the network is training successfully.

This is before reaching a threshold established, that is when we consider that it is enough trained (demostrated by calculating the MCE).

EPOCH NUMBER	TRAINING ERROR	VALIDATION ERROR
1	0,794471604	0,816464717
2	0,773280287	0,787483212
3	0,74206746	0,755303046
4	0,714371285	0,75602438
5	0,715802712	0,732290959
6	0,687940252	0,713195004
7	0,669494554	0,695430415
8	0,647977834	0,668981776
9	0,621089314	0,624400418
10	0,567267471	0,576912699
11	0,495391366	0,52541455
12	0,305945917	0,315945917
13	0,274666303	0,282462389
14	0,150368353	0,161478227
15	0,14857845	0,189304708
16	0,12302745	0,138637075
17	0,128567573	0,128785948
18	0,120618762	0,1150914
19	0,108785033	0,109933473
20	0,102416767	0,105224682



The training error y lower than the validation one, and they are finally getting clores