**Advanced Computational Techniques**

SSE 635

Dr. MacNeil

Spring 2017

**Project III**

Brent Bitler

Matthew Robison

Table of Contents

[Table of Contents 2](#_Toc477108855)

[Table of Figures 3](#_Toc477108856)

[Table of Tables 3](#_Toc477108857)

[No table of figures entries found. 3](#_Toc477108858)

[Introduction to Neural Networks and Back Propagation 4](#_Toc477108859)

[Background 4](#_Toc477108860)

[Neural Network Implementation 7](#_Toc477108861)

[Input 7](#_Toc477108862)

[Create the Neural Network 8](#_Toc477108863)

[Training 10](#_Toc477108864)

[Output 10](#_Toc477108865)

[Conclusion 11](#_Toc477108866)

[Overview 11](#_Toc477108867)

[Appendix 12](#_Toc477108868)

[Driver.cs 12](#_Toc477108869)

[Neural Network.cs 14](#_Toc477108870)

[NetworkTrainer.cs 27](#_Toc477108871)

[SimpleData.xml 39](#_Toc477108872)

[Output.xml 45](#_Toc477108873)

[Error.txt 68](#_Toc477108874)

[Bibliography 80](#_Toc477108875)

[Activity Log – Bitler 81](#_Toc477108876)

[Activity Log – Robison 83](#_Toc477108877)

Table of Figures

[Figure 1. Artificial Neuron example 4](#_Toc477108878)

[Figure 2. Calculation of the net input signal 4](#_Toc477108879)

[Figure 3. Calculation of the signmoid function 4](#_Toc477108880)

[Figure 4. Graph of sigmoid function 5](#_Toc477108881)

[Figure 5. Derivative of the sigmoid function 5](#_Toc477108882)

[Figure 6. Feedforward neural network example 6](#_Toc477108883)

[Figure 7. Output node error calculation 6](#_Toc477108884)

[Figure 8. Hidden layer node error calculation 6](#_Toc477108885)

[Figure 9. Retrieve Training Data 7](#_Toc477108886)

[Figure 10. XML DataSet 8](#_Toc477108887)

[Figure 11. Create Neural Network 8](#_Toc477108888)

[Figure 12. Visual Representation of Neural Network 9](#_Toc477108889)

Table of Tables

**No table of figures entries found.**

Introduction to Neural Networks and Back Propagation

## Background

SSE 635’s purpose is to educate us on the use of advanced computational techniques and their applications. Project III’s focus is on neural networks because neither of us have taken a computational intelligence class prior to this semester. The text begins by discussing the building block of a neural network, the artificial neuron, as shown in figure 1 [1].

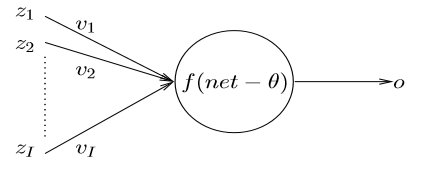


Figure 1. Artificial Neuron example

It receives an input vector of I input signals denoted by **z** = (z1, z2, …, zI) whereupon each input signal has a weight, *vi,* to either build up or work towards tearing down the signal. The net input signal is used in computing the output signal, *o*, from the activation function *f*AN [1]. The next input signal can then be calculated as the weighted sum all input signals as shown in figure 2 [1]. The text contains several activation functions to determine the output of the neuron.

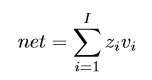


Figure 2. Calculation of the net input signal

For our purposes the sigmoid function was used with its mapping illustrated in figures 3 and 4, where θ = 0.

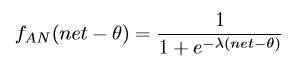


Figure 3. Calculation of the signmoid function

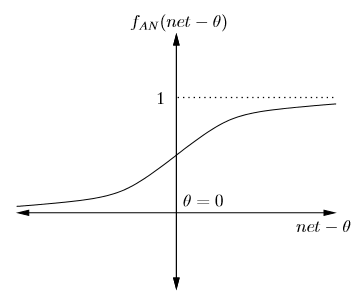


Figure 4. Graph of sigmoid function

The next step in this process is to facilitate the learning of the network. There are three main types of learning, whereupon supervised learning was used for this report [1]:

* Supervised learning – dataset is provided to help adjust the weight values to minimize error of the neuron and target output
* Unsupervised learning – no external sources are provided to discover patterns
* Reinforcement learning – the neuron is rewarded for positive performance and penalized for negative performance

The derivate of the sigmoid transfer function utilizes the quotient rule from differential calculus. The derivation of the derivative is seen in figure 5, thanks to Wolfram Alpha [2]. This will be essential when discussing the output layer later in the report.



Figure 5. Derivative of the sigmoid function

Extending figure 1 to multiple neurons, an input layer, hidden layer, and output layer are utilized in a feedforward neural network for our purposes with a backpropagation training method as illustrated in figure 6.

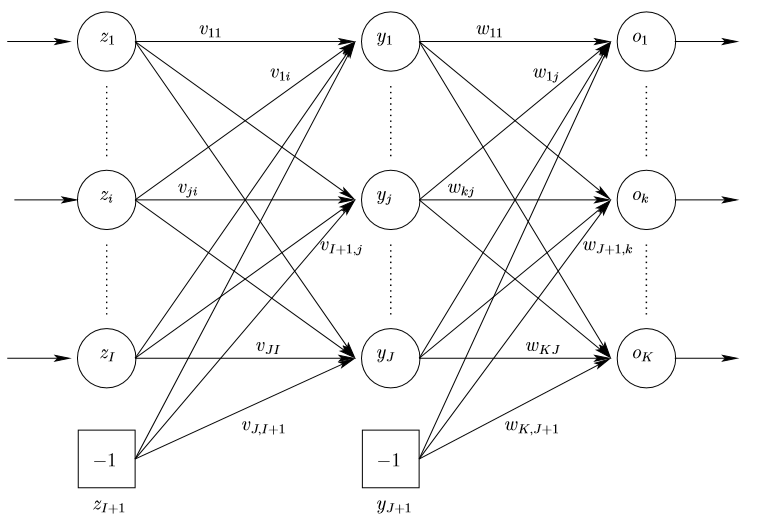


Figure 6. Feedforward neural network example

The text continues by deriving essential equations for the various layers of the neural network. These values are provided below for the sake of completeness with the strategy as follows:

1. Allow the network to run one iteration forward with given input data to achieve an output
2. For every output node, calculate the error given by figure 7, where the highlighted portion is the derivative of the sigmoid function
3. For every hidden layer node, calculate the error given by figure 8, where the highlighted portion is the derivative of the sigmoid function
4. Update the weights and biases by adding the new change in weight to the current weight in the network and repeat this process for the biases in the network such that error is minimized



Figure 7. Output node error calculation



Figure 8. Hidden layer node error calculation

Neural Network Implementation

The purpose of our console application is to take a data set defined by an xml file, create a neural network, and then train the neural network with back propagation algorithm using the xml input file for supervised training of the neural network.

## Input

The first step of our program is to obtain the xml file for the that will be used for supervised training of the neural network. Below in figure 9 we can see the starting point of our console application. In this snippet of code, we are creating a new XmlDocument object and loading our simpleData.xml file into the object. Then we create a new DataSet object and populate the object with the data in our XmlDocument Object.

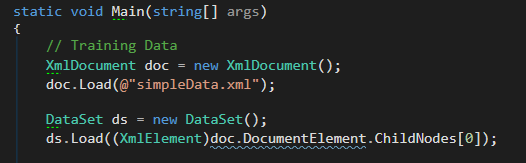


Figure 9. Retrieve Training Data

The input data simpleData is a dataset that used to represent numbers. This is done by having 25 Data Index that can have a value of 1 to indicate that the corresponding square has writing in it or a 0 to indicate a blank square. Below in figure 10 we can see a collage of the data set viewed in xml data viewer. The corresponding xml file can be found in the appendix of the paper.

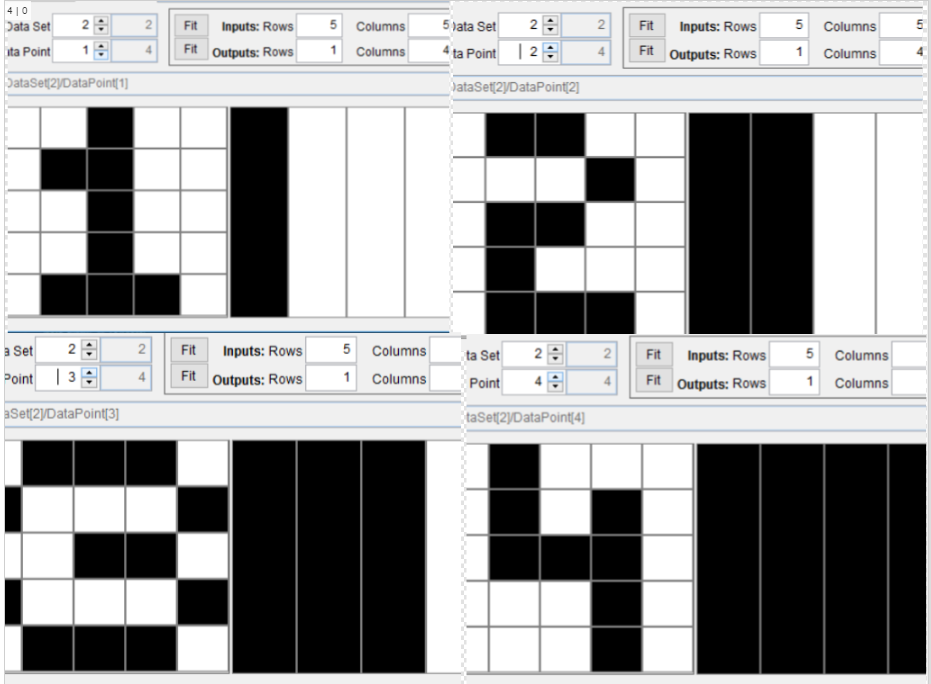


Figure 10. XML DataSet

## Create the Neural Network

The Next Step in the program is to create the default neural network below in figure 11 we can see the code snippet that creates the neural network.

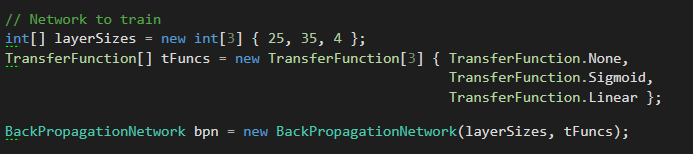


Figure 11. Create Neural Network

The first thing the code snippet does is set the int array for layerSize that we will need to pass into the BackPropagationNetwork Constructor. As we can see this code sets the first element of the array to 25 which mean we will have 25 input neurons in the neural network. The second element of the array is set to 35 meaning we will have 35 hidden layer neurons, and finally the third element of the array is set to 4 meaning there will be 4 output nodes in the neural network. The next line of code creates an array of TransferFunction objects we populate the array with the values none, sigmoid, and linear. We will need this object to pass into the BackPropagationNetwork so later we will know which transfer functions to use during training. The third line of code creates a new BackPropagationNetwork bpn and passes in the layerSize array and tFuncs Array to the Constructor. Below in Figure 12 we can see a visual representation of the neural network created.

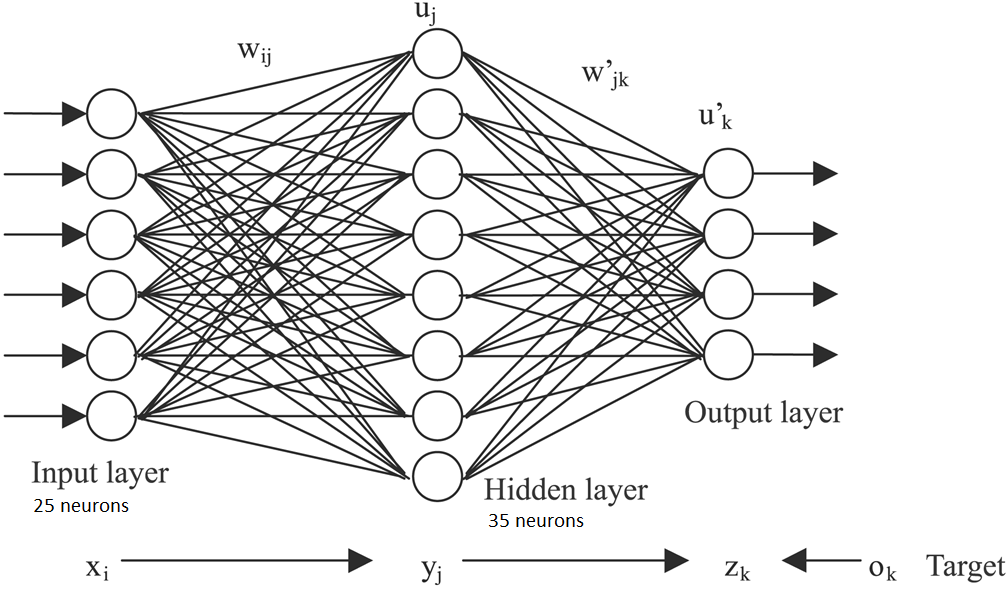


Figure 12. Visual Representation of Neural Network

## Training

The next part of our console application is to train the neural network, this is where most the work of the program takes place. Below in figure 13 we can see where the driver program starts the training process. The first thing that needs to be done is we need to create a NetworkTrainer object. The constructor for the NetworkTrainer object takes in a BackPropagationNetwork object this is the network that we are going to train. Next the constructor takes in a DataSet object which remember is initialized with our input xml file. This DataSet Object is used for supervised training of the network. Next thing we need to set some variables of the NetworkTrainer Object the next three lines set maxError limit that is used for giving the NetworkTrainer a halt condition. Then we set maxIterations variable this is also used for halt condition. Finally, we set the nudge window variable, the purpose of the nudge window is to keep our training algorithm from becoming stale and not converging. The last three lines of the entry point write to the console that we are training, calls the TrainDataSet method and finally write done to the console when training is complete.

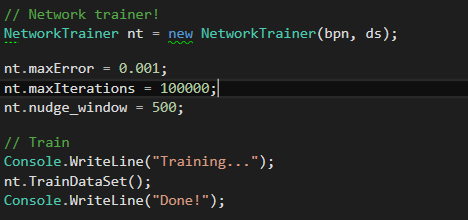


Figure 13. Training Entry Point

Inside of the TrainDataSet method all the necessary initializations are taken care of and the structure for continuous training are in place to run the algorithm until one of the two halt conditions are hit. Below in figure 14 we can see the train method from the BackPropagationNetwork class this is where the network starts to train itself.

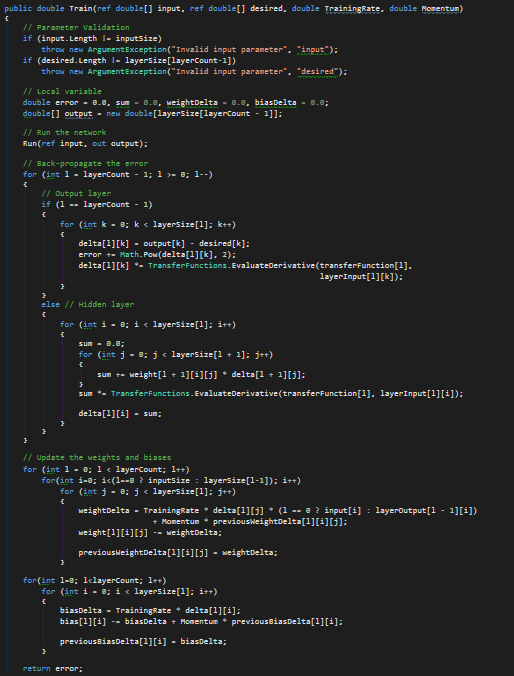


Figure 14. Train Method

Inside of the train method we first check to make sure that the parameters are valid if they are not then we throw and exception. Then we need to set a local variables error, sum, weightDelta, biasDelta and output. We run the network forward so we can get the output of the network to fill our output variable. From the input received by the train method. Then we need to back propagate thru the network from the output layer to the input layer to find the error in correlation to the training data. Then we can calculate the delta we calculate the delta first by getting the difference between our input and our desired. Then we multiple that by the derivative of the transfer function that we set when creating the neural network. The next step in the train method is to update the biases and weights of the network using the deltas that we calculated from back propagation. Finally, we return the error calculate between the current input and the desired.

## Output

The last step of our console application is to write a xml file that describes the neural network and write our error file that documents the error history from each iteration of our train method. Below in figure 15 we can see the were the driver.cs file implements this.

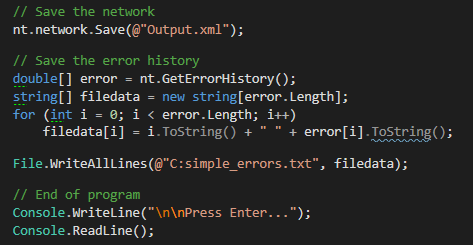


Figure 15. Output implementation

First, we call the save method on the network trainer object which writes the neural network that we trained to a xml file. Below in Figure 16 we can see an example of what our neural network xml file looks like. The complete xml file can be found in the appendix.



Figure 16. Output xml snippet

Finally, the program gets the error history from the NetworkTrainer object and writes it to a text file below in figure 17. We can see a snippet of the error.txt file the whole file can be found in the appendix.

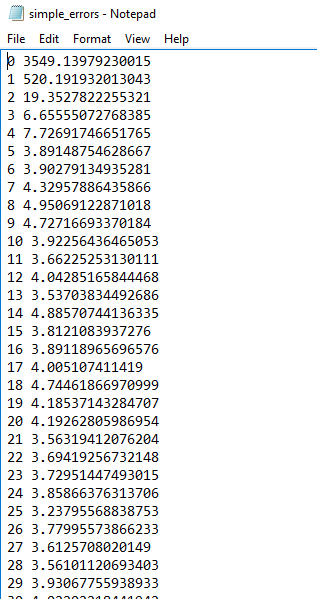


Figure 17. simple errors snippet

Conclusion

## Overview

In conclusion, much was learned about neural networks.

Appendix

## Driver.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using NeuralNetwork;

using System.Xml;

using System.IO;

namespace NeuralNetworkTutorialApp

{

class Program

{

static void Main(string[] args)

{

// Training Data

XmlDocument doc = new XmlDocument();

doc.Load(@"simpleData.xml");

DataSet ds = new DataSet();

ds.Load((XmlElement)doc.DocumentElement.ChildNodes[0]);

// Network to train

int[] layerSizes = new int[3] { 25, 35, 4 };

TransferFunction[] tFuncs = new TransferFunction[3] { TransferFunction.None,

TransferFunction.Sigmoid,

TransferFunction.Linear };

BackPropagationNetwork bpn = new BackPropagationNetwork(layerSizes, tFuncs);

// Network trainer

NetworkTrainer nt = new NetworkTrainer(bpn, ds);

nt.maxError = 0.001; nt.maxIterations = 100000;

nt.nudge\_window = 500;

// Train

Console.WriteLine("Training...");

nt.TrainDataSet();

Console.WriteLine("Done!");

// Save the network

nt.network.Save(@"Output.xml");

// Save the error history

double[] error = nt.GetErrorHistory();

string[] filedata = new string[error.Length];

for (int i = 0; i < error.Length; i++)

filedata[i] = i.ToString() + " " + error[i].ToString();

File.WriteAllLines(@"simple\_errors.txt", filedata);

// End of program

Console.WriteLine("\n\nPress Enter...");

Console.ReadLine();

}

}

}

## Neural Network.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Xml;

namespace NeuralNetwork

{

#region Transfer functions and their derivatives

public enum TransferFunction

{

None,

Sigmoid,

Linear,

Gaussian,

RationalSigmoid

}

static class TransferFunctions

{

public static double Evaluate(TransferFunction tFunc, double input)

{

switch (tFunc)

{

case TransferFunction.Sigmoid:

return sigmoid(input);

case TransferFunction.Linear:

return linear(input);

case TransferFunction.Gaussian:

return gaussian(input);

case TransferFunction.RationalSigmoid:

return rationalsigmoid(input);

case TransferFunction.None:

default:

return 0.0;

}

}

public static double EvaluateDerivative(TransferFunction tFunc, double input)

{

switch (tFunc)

{

case TransferFunction.Sigmoid:

return sigmoid\_derivative(input);

case TransferFunction.Linear:

return linear\_derivative(input);

case TransferFunction.Gaussian:

return gaussian\_derivative(input);

case TransferFunction.RationalSigmoid:

return rationalsigmoid\_derivative(input);

case TransferFunction.None:

default:

return 0.0;

}

}

/\* Transfer function definitions \*/

private static double sigmoid(double x)

{

return 1.0 / (1.0 + Math.Exp(-x));

}

private static double sigmoid\_derivative(double x)

{

return sigmoid(x) \* (1 - sigmoid(x));

}

private static double linear(double x)

{

return x;

}

private static double linear\_derivative(double x)

{

return 1.0;

}

private static double gaussian(double x)

{

return Math.Exp(-Math.Pow(x, 2));

}

private static double gaussian\_derivative(double x)

{

return -2.0 \* x \* gaussian(x);

}

private static double rationalsigmoid(double x)

{

return x / (1.0 + Math.Sqrt(1.0 + x \* x));

}

private static double rationalsigmoid\_derivative(double x)

{

double val = Math.Sqrt(1.0 + x \* x);

return 1.0 / (val \* (1 + val));

}

}

#endregion

public class BackPropagationNetwork

{

#region Constructors

public BackPropagationNetwork(int[] layerSizes, TransferFunction[] transferFunctions)

{

// Validate the input data

if (transferFunctions.Length != layerSizes.Length || transferFunctions[0] != TransferFunction.None)

throw new ArgumentException("Cannot construct a network with these parameters.");

// Initialize network layers

layerCount = layerSizes.Length - 1;

inputSize = layerSizes[0];

layerSize = new int[layerCount];

for (int i = 0; i < layerCount; i++)

layerSize[i] = layerSizes[i + 1];

transferFunction = new TransferFunction[layerCount];

for (int i = 0; i < layerCount; i++)

transferFunction[i] = transferFunctions[i + 1];

// Start dimensioning arrays

bias = new double[layerCount][];

previousBiasDelta = new double[layerCount][];

delta = new double[layerCount][];

layerOutput = new double[layerCount][];

layerInput = new double[layerCount][];

weight = new double[layerCount][][];

previousWeightDelta = new double[layerCount][][];

// Fill 2 dimensional arrays

for (int l = 0; l < layerCount; l++)

{

bias[l] = new double[layerSize[l]];

previousBiasDelta[l] = new double[layerSize[l]];

delta[l] = new double[layerSize[l]];

layerOutput[l] = new double[layerSize[l]];

layerInput[l] = new double[layerSize[l]];

weight[l] = new double[l == 0 ? inputSize : layerSize[l - 1]][];

previousWeightDelta[l] = new double[l == 0 ? inputSize : layerSize[l - 1]][];

for (int i = 0; i < (l == 0 ? inputSize : layerSize[l - 1]); i++)

{

weight[l][i] = new double[layerSize[l]];

previousWeightDelta[l][i] = new double[layerSize[l]];

}

}

// Initialize the weights

for (int l = 0; l < layerCount; l++)

{

for (int j = 0; j < layerSize[l]; j++)

{

bias[l][j] = Gaussian.GetRandomGaussian();

previousBiasDelta[l][j] = 0.0;

layerOutput[l][j] = 0.0;

layerInput[l][j] = 0.0;

delta[l][j] = 0.0;

}

for (int i = 0; i < (l == 0 ? inputSize : layerSize[l - 1]); i++)

{

for (int j = 0; j < layerSize[l]; j++)

{

weight[l][i][j] = Gaussian.GetRandomGaussian();

previousWeightDelta[l][i][j] = 0.0;

}

}

}

}

public BackPropagationNetwork(string FilePath)

{

loaded = false;

Load(FilePath);

loaded = true;

}

#endregion

#region Methods

// Public methods

public void Run(ref double[] input, out double[] output)

{

// Make sure we have enough data

if (input.Length != inputSize)

throw new ArgumentException("Input data is not of the correct dimension.");

// Dimension

output = new double[layerSize[layerCount - 1]];

/\* Run the network! \*/

for (int l = 0; l < layerCount; l++)

{

for (int j = 0; j < layerSize[l]; j++)

{

double sum = 0.0;

for (int i = 0; i < (l == 0 ? inputSize : layerSize[l - 1]); i++)

sum += weight[l][i][j] \* (l == 0 ? input[i] : layerOutput[l - 1][i]);

sum += bias[l][j];

layerInput[l][j] = sum;

layerOutput[l][j] = TransferFunctions.Evaluate(transferFunction[l], sum);

}

}

// Copy the output to the output array

for (int i = 0; i < layerSize[layerCount - 1]; i++)

output[i] = layerOutput[layerCount - 1][i];

}

public double Train(ref double[] input, ref double[] desired, double TrainingRate, double Momentum)

{

// Parameter Validation

if (input.Length != inputSize)

throw new ArgumentException("Invalid input parameter", "input");

if (desired.Length != layerSize[layerCount-1])

throw new ArgumentException("Invalid input parameter", "desired");

// Local variable

double error = 0.0, sum = 0.0, weightDelta = 0.0, biasDelta = 0.0;

double[] output = new double[layerSize[layerCount - 1]];

// Run the network

Run(ref input, out output);

// Back-propagate the error

for (int l = layerCount - 1; l >= 0; l--)

{

// Output layer

if (l == layerCount - 1)

{

for (int k = 0; k < layerSize[l]; k++)

{

delta[l][k] = output[k] - desired[k];

error += Math.Pow(delta[l][k], 2);

delta[l][k] \*= TransferFunctions.EvaluateDerivative(transferFunction[l],

layerInput[l][k]);

}

}

else // Hidden layer

{

for (int i = 0; i < layerSize[l]; i++)

{

sum = 0.0;

for (int j = 0; j < layerSize[l + 1]; j++)

{

sum += weight[l + 1][i][j] \* delta[l + 1][j];

}

sum \*= TransferFunctions.EvaluateDerivative(transferFunction[l], layerInput[l][i]);

delta[l][i] = sum;

}

}

}

// Update the weights and biases

for (int l = 0; l < layerCount; l++)

for(int i=0; i<(l==0 ? inputSize : layerSize[l-1]); i++)

for (int j = 0; j < layerSize[l]; j++)

{

weightDelta = TrainingRate \* delta[l][j] \* (l == 0 ? input[i] : layerOutput[l - 1][i])

+ Momentum \* previousWeightDelta[l][i][j];

weight[l][i][j] -= weightDelta;

previousWeightDelta[l][i][j] = weightDelta;

}

for(int l=0; l<layerCount; l++)

for (int i = 0; i < layerSize[l]; i++)

{

biasDelta = TrainingRate \* delta[l][i];

bias[l][i] -= biasDelta + Momentum \* previousBiasDelta[l][i];

previousBiasDelta[l][i] = biasDelta;

}

return error;

}

public void Save(string FilePath)

{

if (FilePath == null)

return;

XmlWriterSettings settings = new XmlWriterSettings();

settings.Indent = true;

settings.IndentChars = "\t";

XmlWriter writer = XmlWriter.Create(FilePath,settings);

// Begin document

writer.WriteStartElement("NeuralNetwork");

writer.WriteAttributeString("Type", "BackPropagation");

// Parameters element

writer.WriteStartElement("Parameters");

writer.WriteElementString("Name", Name);

writer.WriteElementString("inputSize", inputSize.ToString());

writer.WriteElementString("layerCount", layerCount.ToString());

// Layer sizes

writer.WriteStartElement("Layers");

for (int l = 0; l < layerCount; l++)

{

writer.WriteStartElement("Layer");

writer.WriteAttributeString("Index", l.ToString());

writer.WriteAttributeString("Size", layerSize[l].ToString());

writer.WriteAttributeString("Type", transferFunction[l].ToString());

writer.WriteEndElement(); // Layer

}

writer.WriteEndElement(); // Layers

writer.WriteEndElement(); // Parameters

// Weights and biases

writer.WriteStartElement("Weights");

for (int l = 0; l < layerCount; l++)

{

writer.WriteStartElement("Layer");

writer.WriteAttributeString("Index", l.ToString());

for (int j = 0; j < layerSize[l]; j++)

{

writer.WriteStartElement("Node");

writer.WriteAttributeString("Index", j.ToString());

writer.WriteAttributeString("Bias", bias[l][j].ToString());

for (int i = 0; i < (l == 0 ? inputSize : layerSize[l - 1]); i++)

{

writer.WriteStartElement("Axon");

writer.WriteAttributeString("Index", i.ToString());

writer.WriteString(weight[l][i][j].ToString());

writer.WriteEndElement(); // Axon

}

writer.WriteEndElement(); // Node

}

writer.WriteEndElement(); // Layer

}

writer.WriteEndElement(); // Weights

writer.WriteEndElement(); // NeuralNetwork

writer.Flush();

writer.Close();

}

public void Load(string FilePath)

{

if (FilePath == null)

return;

doc = new XmlDocument();

doc.Load(FilePath);

string BasePath = "", NodePath = "";

double value;

// Load from xml

if (xPathValue("NeuralNetwork/@Type") != "BackPropagation")

return;

BasePath = "NeuralNetwork/Parameters/";

Name = xPathValue(BasePath + "Name");

int.TryParse(xPathValue(BasePath + "inputSize"), out inputSize);

int.TryParse(xPathValue(BasePath + "layerCount"), out layerCount);

layerSize = new int[layerCount];

transferFunction = new TransferFunction[layerCount];

BasePath = "NeuralNetwork/Parameters/Layers/Layer";

for (int l = 0; l < layerCount; l++)

{

int.TryParse(xPathValue(BasePath + "[@Index='" + l.ToString() + "']/@Size"), out layerSize[l]);

Enum.TryParse<TransferFunction>(xPathValue(BasePath + "[@Index='" + l.ToString() + "']/@Type"), out transferFunction[l]);

}

// Parse the Weights element

// Start dimensioning arrays

bias = new double[layerCount][];

previousBiasDelta = new double[layerCount][];

delta = new double[layerCount][];

layerOutput = new double[layerCount][];

layerInput = new double[layerCount][];

weight = new double[layerCount][][];

previousWeightDelta = new double[layerCount][][];

// Fill 2 dimensional arrays

for (int l = 0; l < layerCount; l++)

{

bias[l] = new double[layerSize[l]];

previousBiasDelta[l] = new double[layerSize[l]];

delta[l] = new double[layerSize[l]];

layerOutput[l] = new double[layerSize[l]];

layerInput[l] = new double[layerSize[l]];

weight[l] = new double[l == 0 ? inputSize : layerSize[l - 1]][];

previousWeightDelta[l] = new double[l == 0 ? inputSize : layerSize[l - 1]][];

for (int i = 0; i < (l == 0 ? inputSize : layerSize[l - 1]); i++)

{

weight[l][i] = new double[layerSize[l]];

previousWeightDelta[l][i] = new double[layerSize[l]];

}

}

// Initialize the weights

for (int l = 0; l < layerCount; l++)

{

BasePath = "NeuralNetwork/Weights/Layer[@Index='" + l.ToString() + "']/";

for (int j = 0; j < layerSize[l]; j++)

{

NodePath = "Node[@Index='" + j.ToString() + "']/@Bias";

double.TryParse(xPathValue(BasePath + NodePath), out value);

bias[l][j] = value;

previousBiasDelta[l][j] = 0.0;

layerOutput[l][j] = 0.0;

layerInput[l][j] = 0.0;

delta[l][j] = 0.0;

}

for (int i = 0; i < (l == 0 ? inputSize : layerSize[l - 1]); i++)

{

for (int j = 0; j < layerSize[l]; j++)

{

NodePath = "Node[@Index='" + j.ToString() + "']/Axon[@Index='" + i.ToString() + "']";

double.TryParse(xPathValue(BasePath + NodePath), out value);

weight[l][i][j] = value;

previousWeightDelta[l][i][j] = 0.0;

}

}

}

// "release"

doc = null;

}

public void Nudge(double scalar)

{

// Go through all of the weights and biases and augment them

for (int l = 0; l < layerCount; l++)

{

for (int j = 0; j < layerSize[l]; j++)

{

// Nudge the weights

for (int i = 0; i < (l == 0 ? inputSize : layerSize[l - 1]); i++)

{

double w = weight[l][i][j];

double u = Gaussian.GetRandomGaussian(0f, w \* scalar);

weight[l][i][j] += u;

previousWeightDelta[l][i][j] = 0f;

}

// Nudge the bias

double b = bias[l][j];

double v = Gaussian.GetRandomGaussian(0f, b \* scalar);

bias[l][j] += v;

previousBiasDelta[l][j] = 0f;

}

}

}

// Private methods

private string xPathValue(string xPath)

{

XmlNode node = doc.SelectSingleNode(xPath);

if (node == null)

throw new ArgumentException("Cannot find specified node", xPath);

return node.InnerText;

}

#endregion

#region Public data

public string Name = "Default";

#endregion

#region Private data

private int layerCount;

private int inputSize;

private int[] layerSize;

private TransferFunction[] transferFunction;

private double[][] layerOutput;

private double[][] layerInput;

private double[][] bias;

private double[][] delta;

private double[][] previousBiasDelta;

private double[][][] weight;

private double[][][] previousWeightDelta;

private XmlDocument doc = null;

private bool loaded = true;

#endregion

}

public static class Gaussian

{

private static Random gen = new Random();

public static double GetRandomGaussian()

{

return GetRandomGaussian(0.0, 1.0);

}

public static double GetRandomGaussian(double mean, double stddev)

{

double rVal1, rVal2;

GetRandomGaussian(mean, stddev, out rVal1, out rVal2);

return rVal1;

}

public static void GetRandomGaussian(double mean, double stddev, out double val1, out double val2)

{

double u, v, s, t;

do

{

u = 2 \* gen.NextDouble() - 1;

v = 2 \* gen.NextDouble() - 1;

} while (u \* u + v \* v > 1 || (u == 0 && v == 0));

s = u \* u + v \* v;

t = Math.Sqrt((-2.0 \* Math.Log(s)) / s);

val1 = stddev \* u \* t + mean;

val2 = stddev \* v \* t + mean;

}

}

}

## 

## NetworkTrainer.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Xml;

namespace NeuralNetwork

{

#region Support classes

public class DataPoint

{

public DataPoint() { }

public DataPoint(double[] Input, double[] Output) { Load(Input, Output); }

public DataPoint(XmlElement Elem) { Load(Elem); }

public void Load(double[] Input, double[] Output)

{

input = new double[Input.Length]; output = new double[Output.Length];

Array.Copy(Input, input, Input.Length);

Array.Copy(Output, output, Output.Length);

}

public void Load(XmlElement elem)

{

XmlNode nType;

int lIn, lOut, i;

nType = elem.SelectSingleNode("Input");

lIn = nType.ChildNodes.Count;

input = new double[lIn];

foreach (XmlNode node in nType.ChildNodes)

{

XmlElement Node = (XmlElement)node;

double val;

int.TryParse(Node.GetAttribute("Index"), out i);

double.TryParse(Node.InnerText, out val);

input[i] = val;

}

nType = elem.SelectSingleNode("Output");

lOut = nType.ChildNodes.Count;

output = new double[lOut];

foreach (XmlNode node in nType.ChildNodes)

{

XmlElement Node = (XmlElement)node;

double val;

int.TryParse(Node.GetAttribute("Index"), out i);

double.TryParse(Node.InnerText, out val);

output[i] = val;

}

}

public XmlElement ToXml(XmlDocument doc)

{

XmlElement nDataPoint, nType, node;

int lIn = input.Length, lOut = output.Length;

nDataPoint = doc.CreateElement("DataPoint");

nType = doc.CreateElement("Input");

for (int i = 0; i < lIn; i++)

{

node = doc.CreateElement("Data");

node.SetAttribute("Index", i.ToString());

node.AppendChild(doc.CreateTextNode(input[i].ToString()));

nType.AppendChild(node);

}

nDataPoint.AppendChild(nType);

nType = doc.CreateElement("Output");

for (int i = 0; i < lOut; i++)

{

node = doc.CreateElement("Data");

node.SetAttribute("Index", i.ToString());

node.AppendChild(doc.CreateTextNode(output[i].ToString()));

nType.AppendChild(node);

}

nDataPoint.AppendChild(nType);

return nDataPoint;

}

public double[] input, output;

public int inputSize { get { return input.Length; } }

public int outputSize { get { return output.Length; } }

}

public class DataSet

{

public DataSet() { Data = new List<DataPoint>(); }

public XmlElement ToXml(XmlDocument doc)

{

XmlElement nDataSet;

nDataSet = doc.CreateElement("DataSet");

foreach (DataPoint d in Data)

nDataSet.AppendChild(d.ToXml(doc));

return nDataSet;

}

public void Load(XmlElement nDataSet)

{

foreach (XmlNode node in nDataSet.ChildNodes)

{

DataPoint d = new DataPoint((XmlElement)node);

Data.Add(d);

}

}

public List<DataPoint> Data;

public int Size

{

get { return Data.Count; }

}

}

public class Permutator

{

public Permutator(int Size)

{

index = new int[Size];

for (int i = 0; i < Size; i++)

index[i] = i;

Permute(Size);

}

public void Permute(int nTimes)

{

int i, j, t;

for (int n = 0; n < nTimes; n++)

{

i = gen.Next(index.Length);

j = gen.Next(index.Length);

if (i != j)

{

t = index[i];

index[i] = index[j];

index[j] = t;

}

}

}

public int this[int i]

{

get

{

return index[i];

}

}

private Random gen = new Random();

private int[] index;

}

#endregion

#region Network training classes

public class SimpleNetworkTrainer

{

// Constructors

public SimpleNetworkTrainer(BackPropagationNetwork BPN, DataSet DS)

{

network = BPN; dataSet = DS;

idx = new Permutator(dataSet.Size);

iterations = 0;

errorHistory = new List<double>();

}

// Training method

public void TrainDataSet()

{

do

{

// Prepare to train epoch

iterations++; error = 0.0;

idx.Permute(dataSet.Size);

// Train this epoch

for (int i = 0; i < dataSet.Size; i++)

{

error += network.Train( ref dataSet.Data[idx[i]].input,

ref dataSet.Data[idx[i]].output,

trainingRate, momentum);

}

// Track this error history

errorHistory.Add(error);

// Check whether to Nudge

if (iterations % nudge\_window == 0)

CheckNudge();

} while (error > maxError && iterations < maxIterations);

}

// Accessor method

public double[] GetErrorHistory()

{

return errorHistory.ToArray();

}

// Private method

private void CheckNudge()

{

double oldAvg = 0f, newAvg = 0f;

int l = errorHistory.Count;

// Do i enough data?

if (iterations < 2 \* nudge\_window) return;

// Compute our averages and compare

for (int i = 0; i < nudge\_window; i++)

{

oldAvg += errorHistory[l - 2 \* nudge\_window + i];

newAvg += errorHistory[l - nudge\_window + i];

}

oldAvg /= nudge\_window; newAvg /= nudge\_window;

Console.Write("Iter {0} oldAvg {1:0.0000} newAvg {2:0.0000}", iterations, oldAvg, newAvg);

if (((double)Math.Abs(newAvg - oldAvg)) / nudge\_window < nudge\_tolerance)

{

network.Nudge(nudge\_scale);

Console.Write(" Nudged.");

}

Console.Write("\n");

}

// Public fields

public double maxError = 0.1, maxIterations = 100000;

public double trainingRate = 0.25, momentum = 0.15;

public int nudge\_window = 50;

public double nudge\_scale = 0.25, nudge\_tolerance = 0.0001;

// Private fields

private double error;

private int iterations;

private Permutator idx;

private List<double> errorHistory;

// Training materials

public BackPropagationNetwork network;

public DataSet dataSet;

}

public class NetworkTrainer

{

// Constructor

public NetworkTrainer(BackPropagationNetwork BPN, DataSet DS)

{

network = BPN;

dataSet = DS;

Initialize();

}

public void Initialize()

{

iterations = 0;

if (idx == null)

idx = new Permutator(dataSet.Size);

else

idx.Permute(dataSet.Size);

if (errorHistory == null)

errorHistory = new List<double>();

else

errorHistory.Clear();

}

// Public train method

public bool TrainDataSet()

{

bool success = true;

if (success)

success = \_BeforeTrainDataSet();

if (success)

success = \_TrainDataSetAction();

if (success)

success = \_AfterTrainDataSet();

return success;

}

// Protected hook methods

protected virtual bool BeforeTrainDataSet() { return true; }

protected virtual bool AfterTrainDataSet() { return true; }

protected virtual bool BeforeTrainEpoch() { return true; }

protected virtual bool AfterTrainEpoch() { return true; }

protected virtual bool BeforeTrainDataPoint(ref double[] Input, ref double[] Output, int Index) { return true; }

protected virtual bool AfterTrainDataPoint(ref double[] Input, ref double[] Output, int Index) { return true; }

// Private training methods

private bool \_BeforeTrainDataSet()

{

Initialize();

return BeforeTrainDataSet();

}

private bool \_TrainDataSetAction()

{

bool success = true;

do

{

if (success)

success = \_BeforeTrainEpoch();

if (success)

success = \_TrainEpochAction();

if (success)

success = \_AfterTrainEpoch();

} while (error > maxError && iterations < maxIterations && success);

return success;

}

private bool \_AfterTrainDataSet()

{

return AfterTrainDataSet();

}

private bool \_BeforeTrainEpoch()

{

// Prepare to train epoch

iterations++;

error = 0.0;

idx.Permute(dataSet.Size);

return BeforeTrainEpoch();

}

private bool \_TrainEpochAction()

{

bool success = true;

// Train this epoch

for (int i = 0; i < dataSet.Size && success; i++)

{

// Make a local copy of the data point's data

double[] input = new double[dataSet.Data[idx[i]].inputSize];

double[] output = new double[dataSet.Data[idx[i]].outputSize];

Array.Copy(dataSet.Data[idx[i]].input, input, input.Length);

Array.Copy(dataSet.Data[idx[i]].output, output, output.Length);

if (success)

success = BeforeTrainDataPoint(ref input, ref output, idx[i]);

if (success)

error += network.Train(ref input, ref output, trainingRate, momentum);

if (success)

success = AfterTrainDataPoint(ref input, ref output, idx[i]);

}

return success;

}

private bool \_AfterTrainEpoch()

{

// Track this error history

errorHistory.Add(error);

// Check whether to Nudge

if (iterations % nudge\_window == 0 && nudge)

CheckNudge();

return AfterTrainEpoch();

}

// Accessor method

public double[] GetErrorHistory()

{

return errorHistory.ToArray();

}

// Private method

private void CheckNudge()

{

double oldAvg = 0f, newAvg = 0f;

int l = errorHistory.Count;

// Do i enough data?

if (iterations < 2 \* nudge\_window) return;

// Compute our averages and compare

for (int i = 0; i < nudge\_window; i++)

{

oldAvg += errorHistory[l - 2 \* nudge\_window + i];

newAvg += errorHistory[l - nudge\_window + i];

}

oldAvg /= nudge\_window; newAvg /= nudge\_window;

//Console.Write("Iter {0} oldAvg {1:0.0000} newAvg {2:0.0000}", iterations, oldAvg, newAvg);

if (((double)Math.Abs(newAvg - oldAvg)) / nudge\_window < nudge\_tolerance)

{

network.Nudge(nudge\_scale);

//Console.Write(" Nudged.");

}

//Console.Write("\n");

}

// Public fields

public double maxError = 0.1, maxIterations = 100000;

public double trainingRate = 0.25, momentum = 0.15;

public int nudge\_window = 50;

public bool nudge = true;

public double nudge\_scale = 0.25, nudge\_tolerance = 0.0001;

// Private fields

private double error;

private int iterations;

private Permutator idx;

private List<double> errorHistory;

// Training materials

public BackPropagationNetwork network;

public DataSet dataSet;

}

public class BinaryNoiseTrainer : NetworkTrainer

{

// Constructor

public BinaryNoiseTrainer(BackPropagationNetwork BPN, DataSet DS)

: base(BPN, DS)

{

// Additional initialization stuff here

rnd = new Random();

}

// Overrides

protected override bool BeforeTrainEpoch()

{

// Reset the NoisyData

NoisyData = new DataSet();

return true;

}

protected override bool BeforeTrainDataPoint(ref double[] Input, ref double[] Output, int Index)

{

// Add some noise to the input data

// Add noise to the input

for(int i=0; i<Input.Length; i++)

if(rnd.NextDouble() < \_noise\_density)

Input[i] = (Input[i] == 0.0 ? 1.0 : 0.0);

// Add this "dirty" data point to the data set

DataPoint dp = new DataPoint(Input, Output);

NoisyData.Data.Add(dp);

return true;

}

// Private data

private double \_noise\_density = 0.10;

public double noise\_density {

get { return \_noise\_density;}

set{

\_noise\_density = Math.Min(1.0, Math.Max(0.0, value));

}

}

public DataSet NoisyData;

private Random rnd;

}

#endregion

}

## 

## SimpleData.xml

<Root>

<DataSet>

<DataPoint>

<Input>

<Data Index="0">0</Data>

<Data Index="1">0</Data>

<Data Index="2">1</Data>

<Data Index="3">0</Data>

<Data Index="4">0</Data>

<Data Index="5">0</Data>

<Data Index="6">0</Data>

<Data Index="7">1</Data>

<Data Index="8">0</Data>

<Data Index="9">0</Data>

<Data Index="10">1</Data>

<Data Index="11">1</Data>

<Data Index="12">1</Data>

<Data Index="13">1</Data>

<Data Index="14">1</Data>

<Data Index="15">0</Data>

<Data Index="16">0</Data>

<Data Index="17">1</Data>

<Data Index="18">0</Data>

<Data Index="19">0</Data>

<Data Index="20">0</Data>

<Data Index="21">0</Data>

<Data Index="22">1</Data>

<Data Index="23">0</Data>

<Data Index="24">0</Data>

</Input>

<Output>

<Data Index="0">1</Data>

<Data Index="1">0</Data>

<Data Index="2">0</Data>

<Data Index="3">0</Data>

</Output>

</DataPoint>

<DataPoint>

<Input>

<Data Index="0">1</Data>

<Data Index="1">0</Data>

<Data Index="2">0</Data>

<Data Index="3">0</Data>

<Data Index="4">1</Data>

<Data Index="5">0</Data>

<Data Index="6">1</Data>

<Data Index="7">0</Data>

<Data Index="8">1</Data>

<Data Index="9">0</Data>

<Data Index="10">0</Data>

<Data Index="11">0</Data>

<Data Index="12">1</Data>

<Data Index="13">0</Data>

<Data Index="14">0</Data>

<Data Index="15">0</Data>

<Data Index="16">1</Data>

<Data Index="17">0</Data>

<Data Index="18">1</Data>

<Data Index="19">0</Data>

<Data Index="20">1</Data>

<Data Index="21">0</Data>

<Data Index="22">0</Data>

<Data Index="23">0</Data>

<Data Index="24">1</Data>

</Input>

<Output>

<Data Index="0">0</Data>

<Data Index="1">2</Data>

<Data Index="2">0</Data>

<Data Index="3">0</Data>

</Output>

</DataPoint>

<DataPoint>

<Input>

<Data Index="0">0</Data>

<Data Index="1">0</Data>

<Data Index="2">0</Data>

<Data Index="3">0</Data>

<Data Index="4">0</Data>

<Data Index="5">0</Data>

<Data Index="6">0</Data>

<Data Index="7">0</Data>

<Data Index="8">0</Data>

<Data Index="9">0</Data>

<Data Index="10">1</Data>

<Data Index="11">1</Data>

<Data Index="12">1</Data>

<Data Index="13">1</Data>

<Data Index="14">1</Data>

<Data Index="15">0</Data>

<Data Index="16">0</Data>

<Data Index="17">0</Data>

<Data Index="18">0</Data>

<Data Index="19">0</Data>

<Data Index="20">0</Data>

<Data Index="21">0</Data>

<Data Index="22">0</Data>

<Data Index="23">0</Data>

<Data Index="24">0</Data>

</Input>

<Output>

<Data Index="0">0</Data>

<Data Index="1">0</Data>

<Data Index="2">1</Data>

<Data Index="3">0</Data>

</Output>

</DataPoint>

<DataPoint>

<Input>

<Data Index="0">0</Data>

<Data Index="1">0</Data>

<Data Index="2">1</Data>

<Data Index="3">0</Data>

<Data Index="4">0</Data>

<Data Index="5">0</Data>

<Data Index="6">0</Data>

<Data Index="7">0</Data>

<Data Index="8">0</Data>

<Data Index="9">0</Data>

<Data Index="10">1</Data>

<Data Index="11">1</Data>

<Data Index="12">1</Data>

<Data Index="13">1</Data>

<Data Index="14">1</Data>

<Data Index="15">0</Data>

<Data Index="16">0</Data>

<Data Index="17">0</Data>

<Data Index="18">0</Data>

<Data Index="19">0</Data>

<Data Index="20">0</Data>

<Data Index="21">0</Data>

<Data Index="22">1</Data>

<Data Index="23">0</Data>

<Data Index="24">0</Data>

</Input>

<Output>

<Data Index="0">0</Data>

<Data Index="1">0</Data>

<Data Index="2">0</Data>

<Data Index="3">1</Data>

</Output>

</DataPoint>

</DataSet>

<DataSet>

<DataPoint>

<Input>

<Data Index="0">0</Data>

<Data Index="1">0</Data>

<Data Index="2">1</Data>

<Data Index="3">0</Data>

<Data Index="4">0</Data>

<Data Index="5">0</Data>

<Data Index="6">1</Data>

<Data Index="7">1</Data>

<Data Index="8">0</Data>

<Data Index="9">0</Data>

<Data Index="10">0</Data>

<Data Index="11">0</Data>

<Data Index="12">1</Data>

<Data Index="13">0</Data>

<Data Index="14">0</Data>

<Data Index="15">0</Data>

<Data Index="16">0</Data>

<Data Index="17">1</Data>

<Data Index="18">0</Data>

<Data Index="19">0</Data>

<Data Index="20">0</Data>

<Data Index="21">1</Data>

<Data Index="22">1</Data>

<Data Index="23">1</Data>

<Data Index="24">0</Data>

</Input>

<Output>

<Data Index="0">1</Data>

<Data Index="1">0</Data>

<Data Index="2">0</Data>

<Data Index="3">0</Data>

</Output>

</DataPoint>

<DataPoint>

<Input>

<Data Index="0">0</Data>

<Data Index="1">1</Data>

<Data Index="2">1</Data>

<Data Index="3">0</Data>

<Data Index="4">0</Data>

<Data Index="5">0</Data>

<Data Index="6">0</Data>

<Data Index="7">0</Data>

<Data Index="8">1</Data>

<Data Index="9">0</Data>

<Data Index="10">0</Data>

<Data Index="11">1</Data>

<Data Index="12">1</Data>

<Data Index="13">0</Data>

<Data Index="14">0</Data>

<Data Index="15">0</Data>

<Data Index="16">1</Data>

<Data Index="17">0</Data>

<Data Index="18">0</Data>

<Data Index="19">0</Data>

<Data Index="20">0</Data>

<Data Index="21">1</Data>

<Data Index="22">1</Data>

<Data Index="23">1</Data>

<Data Index="24">0</Data>

</Input>

<Output>

<Data Index="0">1</Data>

<Data Index="1">1</Data>

<Data Index="2">0</Data>

<Data Index="3">0</Data>

</Output>

</DataPoint>

<DataPoint>

<Input>

<Data Index="0">0</Data>

<Data Index="1">1</Data>

<Data Index="2">1</Data>

<Data Index="3">1</Data>

<Data Index="4">0</Data>

<Data Index="5">1</Data>

<Data Index="6">0</Data>

<Data Index="7">0</Data>

<Data Index="8">0</Data>

<Data Index="9">1</Data>

<Data Index="10">0</Data>

<Data Index="11">0</Data>

<Data Index="12">1</Data>

<Data Index="13">1</Data>

<Data Index="14">0</Data>

<Data Index="15">1</Data>

<Data Index="16">0</Data>

<Data Index="17">0</Data>

<Data Index="18">0</Data>

<Data Index="19">1</Data>

<Data Index="20">0</Data>

<Data Index="21">1</Data>

<Data Index="22">1</Data>

<Data Index="23">1</Data>

<Data Index="24">0</Data>

</Input>

<Output>

<Data Index="0">1</Data>

<Data Index="1">1</Data>

<Data Index="2">1</Data>

<Data Index="3">0</Data>

</Output>

</DataPoint>

<DataPoint>

<Input>

<Data Index="0">0</Data>

<Data Index="1">1</Data>

<Data Index="2">0</Data>

<Data Index="3">0</Data>

<Data Index="4">0</Data>

<Data Index="5">0</Data>

<Data Index="6">1</Data>

<Data Index="7">0</Data>

<Data Index="8">1</Data>

<Data Index="9">0</Data>

<Data Index="10">0</Data>

<Data Index="11">1</Data>

<Data Index="12">1</Data>

<Data Index="13">1</Data>

<Data Index="14">0</Data>

<Data Index="15">0</Data>

<Data Index="16">0</Data>

<Data Index="17">0</Data>

<Data Index="18">1</Data>

<Data Index="19">0</Data>

<Data Index="20">0</Data>

<Data Index="21">0</Data>

<Data Index="22">0</Data>

<Data Index="23">1</Data>

<Data Index="24">0</Data>

</Input>

<Output>

<Data Index="0">1</Data>

<Data Index="1">1</Data>

<Data Index="2">1</Data>

<Data Index="3">1</Data>

</Output>

</DataPoint>

</DataSet>

</Root>

## 

## Output.xml

<?xml version="1.0" encoding="utf-8"?>

<NeuralNetwork Type="BackPropagation">

<Parameters>

<Name>Default</Name>

<inputSize>25</inputSize>

<layerCount>2</layerCount>

<Layers>

<Layer Index="0" Size="35" Type="Sigmoid" />

<Layer Index="1" Size="4" Type="Linear" />

</Layers>

</Parameters>

<Weights>

<Layer Index="0">

<Node Index="0" Bias="-0.637224556127275">

<Axon Index="0">-2.02349963497412</Axon>

<Axon Index="1">-0.573687994121889</Axon>

<Axon Index="2">-0.241497490283106</Axon>

<Axon Index="3">-0.299959567196153</Axon>

<Axon Index="4">1.02492941398588</Axon>

<Axon Index="5">1.65540989105929</Axon>

<Axon Index="6">-0.892682416409745</Axon>

<Axon Index="7">-0.99366333400747</Axon>

<Axon Index="8">1.00521552207078</Axon>

<Axon Index="9">-0.77100521215945</Axon>

<Axon Index="10">-3.44977218842731</Axon>

<Axon Index="11">-0.769061713303465</Axon>

<Axon Index="12">-3.68290550995133</Axon>

<Axon Index="13">-0.833441239998937</Axon>

<Axon Index="14">-2.45084648954971</Axon>

<Axon Index="15">1.8011334487758</Axon>

<Axon Index="16">-1.81177327398196</Axon>

<Axon Index="17">-0.00906512896280137</Axon>

<Axon Index="18">-0.526336674092259</Axon>

<Axon Index="19">1.97929110547026</Axon>

<Axon Index="20">-2.35230319860441</Axon>

<Axon Index="21">0.633252268034414</Axon>

<Axon Index="22">0.494188394499366</Axon>

<Axon Index="23">0.493499096009778</Axon>

<Axon Index="24">1.26432588821817</Axon>

</Node>

<Node Index="1" Bias="-7.10830811899008">

<Axon Index="0">-5.186399213983</Axon>

<Axon Index="1">-0.512998015674535</Axon>

<Axon Index="2">0.281329979465504</Axon>

<Axon Index="3">1.21411285734276</Axon>

<Axon Index="4">-5.80863814018689</Axon>

<Axon Index="5">2.48062883335753</Axon>

<Axon Index="6">-4.10901947564368</Axon>

<Axon Index="7">0.768901278461757</Axon>

<Axon Index="8">-6.7984134703234</Axon>

<Axon Index="9">-0.892431153425366</Axon>

<Axon Index="10">-1.10855101797994</Axon>

<Axon Index="11">-0.545949227017688</Axon>

<Axon Index="12">-8.06434711380877</Axon>

<Axon Index="13">-1.11168714821242</Axon>

<Axon Index="14">-1.42795510280442</Axon>

<Axon Index="15">1.51131568602191</Axon>

<Axon Index="16">-4.78788564571703</Axon>

<Axon Index="17">-1.16430776242499</Axon>

<Axon Index="18">-5.80358002311782</Axon>

<Axon Index="19">-0.857696932024281</Axon>

<Axon Index="20">-7.78459373779837</Axon>

<Axon Index="21">0.274843630995113</Axon>

<Axon Index="22">-0.691318238014446</Axon>

<Axon Index="23">0.281533821374628</Axon>

<Axon Index="24">-4.11887354428904</Axon>

</Node>

<Node Index="2" Bias="-4.32832458254733">

<Axon Index="0">-4.05995274292924</Axon>

<Axon Index="1">-0.584890290731763</Axon>

<Axon Index="2">0.947039032945949</Axon>

<Axon Index="3">-1.00124032913027</Axon>

<Axon Index="4">-4.61298646216319</Axon>

<Axon Index="5">0.251066466218862</Axon>

<Axon Index="6">-4.63805237086598</Axon>

<Axon Index="7">-0.0694081963382324</Axon>

<Axon Index="8">-4.04942702254055</Axon>

<Axon Index="9">1.08855341944647</Axon>

<Axon Index="10">0.42515943877155</Axon>

<Axon Index="11">-0.537603385577501</Axon>

<Axon Index="12">-4.83869321545542</Axon>

<Axon Index="13">-0.269005794049577</Axon>

<Axon Index="14">0.709294837359732</Axon>

<Axon Index="15">0.967287974250355</Axon>

<Axon Index="16">-4.59698885721704</Axon>

<Axon Index="17">-0.728486000616945</Axon>

<Axon Index="18">-3.58658172238236</Axon>

<Axon Index="19">0.638980463688557</Axon>

<Axon Index="20">-3.48204689036771</Axon>

<Axon Index="21">-0.309014816497717</Axon>

<Axon Index="22">0.500683802593654</Axon>

<Axon Index="23">0.362222538628012</Axon>

<Axon Index="24">-1.88803023134951</Axon>

</Node>

<Node Index="3" Bias="-0.172907449363075">

<Axon Index="0">-0.762963937862058</Axon>

<Axon Index="1">0.818400560449986</Axon>

<Axon Index="2">-0.533759902061574</Axon>

<Axon Index="3">-0.37449226680287</Axon>

<Axon Index="4">0.28726136271943</Axon>

<Axon Index="5">-1.321900437379</Axon>

<Axon Index="6">0.239831333912234</Axon>

<Axon Index="7">1.2377176092612</Axon>

<Axon Index="8">-0.601474846657102</Axon>

<Axon Index="9">0.48134369216865</Axon>

<Axon Index="10">-0.41268768152893</Axon>

<Axon Index="11">0.356022466298589</Axon>

<Axon Index="12">1.00448665916061</Axon>

<Axon Index="13">0.658710014728243</Axon>

<Axon Index="14">-0.704423384887457</Axon>

<Axon Index="15">-0.838781921075341</Axon>

<Axon Index="16">0.0225751033622524</Axon>

<Axon Index="17">1.11278918660924</Axon>

<Axon Index="18">-0.0847340872840909</Axon>

<Axon Index="19">-1.55262882106118</Axon>

<Axon Index="20">-2.37028323760636</Axon>

<Axon Index="21">0.468665274158406</Axon>

<Axon Index="22">-1.90945323337111</Axon>

<Axon Index="23">-0.971646282183778</Axon>

<Axon Index="24">-1.3927091447067</Axon>

</Node>

<Node Index="4" Bias="-1.12775193087577">

<Axon Index="0">-1.64863475836507</Axon>

<Axon Index="1">0.483697471236287</Axon>

<Axon Index="2">-0.0675808740629412</Axon>

<Axon Index="3">0.696124148678547</Axon>

<Axon Index="4">-1.74534699779578</Axon>

<Axon Index="5">-0.0194706621129436</Axon>

<Axon Index="6">-2.02667820901451</Axon>

<Axon Index="7">1.25029710890418</Axon>

<Axon Index="8">-2.46143013014763</Axon>

<Axon Index="9">-2.1014518480324</Axon>

<Axon Index="10">-1.78646273712257</Axon>

<Axon Index="11">-3.61287050506793</Axon>

<Axon Index="12">-1.81985359308028</Axon>

<Axon Index="13">-1.1655483105574</Axon>

<Axon Index="14">-1.85545492213815</Axon>

<Axon Index="15">-0.952858669595549</Axon>

<Axon Index="16">-2.13455527882487</Axon>

<Axon Index="17">-0.422494962026811</Axon>

<Axon Index="18">-3.09168104317922</Axon>

<Axon Index="19">0.84376801977233</Axon>

<Axon Index="20">-2.49605405255651</Axon>

<Axon Index="21">0.616783539049194</Axon>

<Axon Index="22">1.99787578030825</Axon>

<Axon Index="23">-1.04774348909536</Axon>

<Axon Index="24">-2.04184737251938</Axon>

</Node>

<Node Index="5" Bias="-1.98345645978351">

<Axon Index="0">-1.71387201648543</Axon>

<Axon Index="1">-1.29653288239046</Axon>

<Axon Index="2">-2.37880079127612</Axon>

<Axon Index="3">1.24322170116634</Axon>

<Axon Index="4">-0.0663542404787776</Axon>

<Axon Index="5">0.0177785293245935</Axon>

<Axon Index="6">-0.741722317445618</Axon>

<Axon Index="7">0.0207748526948639</Axon>

<Axon Index="8">-0.556773124079629</Axon>

<Axon Index="9">-0.899072035407593</Axon>

<Axon Index="10">-1.24374813563785</Axon>

<Axon Index="11">-0.0883765430476889</Axon>

<Axon Index="12">0.230934192222916</Axon>

<Axon Index="13">-1.18874008516818</Axon>

<Axon Index="14">-0.50786865913758</Axon>

<Axon Index="15">0.964369242828331</Axon>

<Axon Index="16">-1.43108085313206</Axon>

<Axon Index="17">-0.340885824664007</Axon>

<Axon Index="18">-0.268693091337557</Axon>

<Axon Index="19">-0.585810777038829</Axon>

<Axon Index="20">-0.48919012380685</Axon>

<Axon Index="21">-1.75463759465563</Axon>

<Axon Index="22">-0.238740472167221</Axon>

<Axon Index="23">-0.636617225911493</Axon>

<Axon Index="24">1.25784823998993</Axon>

</Node>

<Node Index="6" Bias="-3.64618963020909">

<Axon Index="0">0.101502255304969</Axon>

<Axon Index="1">-1.56324045683817</Axon>

<Axon Index="2">-0.733728272554327</Axon>

<Axon Index="3">0.94742338093482</Axon>

<Axon Index="4">0.635384701098628</Axon>

<Axon Index="5">-1.06953857071266</Axon>

<Axon Index="6">0.957137002562614</Axon>

<Axon Index="7">-0.168966138535249</Axon>

<Axon Index="8">-2.46909757157949</Axon>

<Axon Index="9">0.834577866273555</Axon>

<Axon Index="10">-3.05662028590983</Axon>

<Axon Index="11">-1.86163428605028</Axon>

<Axon Index="12">-4.03188600461327</Axon>

<Axon Index="13">-3.47109869596655</Axon>

<Axon Index="14">-2.18996009544376</Axon>

<Axon Index="15">-0.0589896355644006</Axon>

<Axon Index="16">1.34471931445288</Axon>

<Axon Index="17">0.542092989060346</Axon>

<Axon Index="18">0.162202978927795</Axon>

<Axon Index="19">-0.837429535455518</Axon>

<Axon Index="20">0.171133089721138</Axon>

<Axon Index="21">2.14859115596671</Axon>

<Axon Index="22">1.87348644496511</Axon>

<Axon Index="23">0.835095183733526</Axon>

<Axon Index="24">-1.04440583696094</Axon>

</Node>

<Node Index="7" Bias="-0.177908744702897">

<Axon Index="0">-1.93466596801392</Axon>

<Axon Index="1">-0.3659324978196</Axon>

<Axon Index="2">0.859670225962326</Axon>

<Axon Index="3">1.06858110716468</Axon>

<Axon Index="4">-1.04834806573778</Axon>

<Axon Index="5">-0.7237344626343</Axon>

<Axon Index="6">-0.425605632696411</Axon>

<Axon Index="7">0.0923426574375902</Axon>

<Axon Index="8">-0.384743458271985</Axon>

<Axon Index="9">-1.15505144298181</Axon>

<Axon Index="10">-0.423109227982318</Axon>

<Axon Index="11">0.758781766589152</Axon>

<Axon Index="12">-2.44645397552648</Axon>

<Axon Index="13">1.38864444827705</Axon>

<Axon Index="14">1.79171235597651</Axon>

<Axon Index="15">-1.04927301389602</Axon>

<Axon Index="16">0.537278462514925</Axon>

<Axon Index="17">2.10532592205687</Axon>

<Axon Index="18">-1.86282831925768</Axon>

<Axon Index="19">0.376460683181911</Axon>

<Axon Index="20">-0.595382356761426</Axon>

<Axon Index="21">1.57354170830298</Axon>

<Axon Index="22">0.161453480198126</Axon>

<Axon Index="23">-1.04717781827694</Axon>

<Axon Index="24">-1.2034294091885</Axon>

</Node>

<Node Index="8" Bias="-14.3219276968478">

<Axon Index="0">-13.6427304875588</Axon>

<Axon Index="1">-0.292770716393924</Axon>

<Axon Index="2">-0.502189122672799</Axon>

<Axon Index="3">-0.376074736121351</Axon>

<Axon Index="4">-11.6631614841643</Axon>

<Axon Index="5">1.88912401203518</Axon>

<Axon Index="6">-12.4465845846036</Axon>

<Axon Index="7">-0.962405241415205</Axon>

<Axon Index="8">-13.0866499879955</Axon>

<Axon Index="9">-0.456211307375322</Axon>

<Axon Index="10">-1.42033058202754</Axon>

<Axon Index="11">-2.69393421626944</Axon>

<Axon Index="12">-15.0329205333233</Axon>

<Axon Index="13">-1.97502094725264</Axon>

<Axon Index="14">-2.20135581329423</Axon>

<Axon Index="15">1.13843904601977</Axon>

<Axon Index="16">-12.3540004417926</Axon>

<Axon Index="17">0.501248714381978</Axon>

<Axon Index="18">-11.0693438107129</Axon>

<Axon Index="19">-0.00298497706864195</Axon>

<Axon Index="20">-13.6433328446892</Axon>

<Axon Index="21">-0.791277428880429</Axon>

<Axon Index="22">1.62557112323572</Axon>

<Axon Index="23">-1.23235589721905</Axon>

<Axon Index="24">-13.0908064634581</Axon>

</Node>

<Node Index="9" Bias="-1.58500170777776">

<Axon Index="0">-1.49128634769286</Axon>

<Axon Index="1">1.21556395480183</Axon>

<Axon Index="2">-1.49833893646103</Axon>

<Axon Index="3">-1.31180893408144</Axon>

<Axon Index="4">-0.379659072616777</Axon>

<Axon Index="5">2.0479617815693</Axon>

<Axon Index="6">0.286805812510539</Axon>

<Axon Index="7">-0.614691739658035</Axon>

<Axon Index="8">-0.981757899276666</Axon>

<Axon Index="9">0.658202961887055</Axon>

<Axon Index="10">-3.38509607119167</Axon>

<Axon Index="11">-3.54385852801067</Axon>

<Axon Index="12">-2.36099288725368</Axon>

<Axon Index="13">-2.677027624755</Axon>

<Axon Index="14">-2.10787524510255</Axon>

<Axon Index="15">-1.37067526233202</Axon>

<Axon Index="16">-1.45701444812992</Axon>

<Axon Index="17">-1.02971442727276</Axon>

<Axon Index="18">0.868508756963375</Axon>

<Axon Index="19">0.19375258612934</Axon>

<Axon Index="20">-1.47805940996695</Axon>

<Axon Index="21">0.0792219618060994</Axon>

<Axon Index="22">-2.75595817818019</Axon>

<Axon Index="23">0.578510862628572</Axon>

<Axon Index="24">-0.779218186266912</Axon>

</Node>

<Node Index="10" Bias="-2.28988440684473">

<Axon Index="0">-0.497309753664954</Axon>

<Axon Index="1">-1.66610339909584</Axon>

<Axon Index="2">0.151272721472404</Axon>

<Axon Index="3">-0.281148849449717</Axon>

<Axon Index="4">-2.01000648670387</Axon>

<Axon Index="5">0.109481068397825</Axon>

<Axon Index="6">-0.728824950460878</Axon>

<Axon Index="7">0.30077058773817</Axon>

<Axon Index="8">-0.52419910181665</Axon>

<Axon Index="9">-0.721006423394531</Axon>

<Axon Index="10">-1.46756290992436</Axon>

<Axon Index="11">0.983525326490844</Axon>

<Axon Index="12">-2.39812666419375</Axon>

<Axon Index="13">0.105212988321978</Axon>

<Axon Index="14">-1.64076284242388</Axon>

<Axon Index="15">1.07073950843733</Axon>

<Axon Index="16">-0.160053902718247</Axon>

<Axon Index="17">0.287761038332569</Axon>

<Axon Index="18">-2.23479545039722</Axon>

<Axon Index="19">0.759010969244517</Axon>

<Axon Index="20">-1.26157488413753</Axon>

<Axon Index="21">-1.45467930635672</Axon>

<Axon Index="22">-0.105418644540545</Axon>

<Axon Index="23">0.270384215715819</Axon>

<Axon Index="24">-0.387077548012446</Axon>

</Node>

<Node Index="11" Bias="-5.7461313362805">

<Axon Index="0">-4.93193242820537</Axon>

<Axon Index="1">0.0510013151195052</Axon>

<Axon Index="2">-0.170468744710128</Axon>

<Axon Index="3">0.906413673971662</Axon>

<Axon Index="4">-6.02627551784742</Axon>

<Axon Index="5">2.42052191583965</Axon>

<Axon Index="6">-2.68710169204173</Axon>

<Axon Index="7">0.572585808324372</Axon>

<Axon Index="8">-4.3699273714875</Axon>

<Axon Index="9">1.33347923475198</Axon>

<Axon Index="10">-2.34969860228592</Axon>

<Axon Index="11">-3.19688030021879</Axon>

<Axon Index="12">-9.00143177551764</Axon>

<Axon Index="13">-1.24948283032828</Axon>

<Axon Index="14">-1.20409334924835</Axon>

<Axon Index="15">0.717428397253411</Axon>

<Axon Index="16">-5.14596834980687</Axon>

<Axon Index="17">-0.0618763665262397</Axon>

<Axon Index="18">-4.8899367496906</Axon>

<Axon Index="19">-0.896226064914052</Axon>

<Axon Index="20">-4.61487172721623</Axon>

<Axon Index="21">1.04733886758864</Axon>

<Axon Index="22">0.109495403651998</Axon>

<Axon Index="23">-0.413496915140722</Axon>

<Axon Index="24">-6.02652411722537</Axon>

</Node>

<Node Index="12" Bias="-7.13248659169725">

<Axon Index="0">-6.09553485821688</Axon>

<Axon Index="1">-0.580234039603925</Axon>

<Axon Index="2">0.60002737288797</Axon>

<Axon Index="3">-0.679919873403779</Axon>

<Axon Index="4">-6.53243398534136</Axon>

<Axon Index="5">-0.46441675567458</Axon>

<Axon Index="6">-5.38867254358881</Axon>

<Axon Index="7">-0.303276774926144</Axon>

<Axon Index="8">-5.45384694743344</Axon>

<Axon Index="9">0.264259304602565</Axon>

<Axon Index="10">0.30489551177065</Axon>

<Axon Index="11">-0.543939154814332</Axon>

<Axon Index="12">-6.03758255989088</Axon>

<Axon Index="13">-0.670234568391168</Axon>

<Axon Index="14">-0.727551959338951</Axon>

<Axon Index="15">0.875231476061978</Axon>

<Axon Index="16">-4.45711345719697</Axon>

<Axon Index="17">0.56054157038654</Axon>

<Axon Index="18">-4.53738291738689</Axon>

<Axon Index="19">0.528572279064585</Axon>

<Axon Index="20">-5.35374720529715</Axon>

<Axon Index="21">-0.872512841710048</Axon>

<Axon Index="22">-1.1710733863065</Axon>

<Axon Index="23">1.0284999531022</Axon>

<Axon Index="24">-5.18005109396559</Axon>

</Node>

<Node Index="13" Bias="-2.09861731477716">

<Axon Index="0">1.1166623789982</Axon>

<Axon Index="1">0.752225864525074</Axon>

<Axon Index="2">1.2906065738237</Axon>

<Axon Index="3">-1.05357682878553</Axon>

<Axon Index="4">-1.88153720783067</Axon>

<Axon Index="5">0.789446440511218</Axon>

<Axon Index="6">-2.15698262768009</Axon>

<Axon Index="7">1.60080559225754</Axon>

<Axon Index="8">-0.855266961022009</Axon>

<Axon Index="9">1.3508999564464</Axon>

<Axon Index="10">-0.103467210115317</Axon>

<Axon Index="11">-1.8169885604031</Axon>

<Axon Index="12">-3.333173561514</Axon>

<Axon Index="13">-1.28477960621121</Axon>

<Axon Index="14">-1.20459560429061</Axon>

<Axon Index="15">1.23446774247305</Axon>

<Axon Index="16">0.259625047377527</Axon>

<Axon Index="17">-1.31435302836688</Axon>

<Axon Index="18">-1.38481828981483</Axon>

<Axon Index="19">0.338564516877023</Axon>

<Axon Index="20">-1.47019759668521</Axon>

<Axon Index="21">0.0994677001980828</Axon>

<Axon Index="22">-0.469853741669982</Axon>

<Axon Index="23">-0.488056981130454</Axon>

<Axon Index="24">-0.809127562341041</Axon>

</Node>

<Node Index="14" Bias="-4.04513458941918">

<Axon Index="0">-3.01627562807412</Axon>

<Axon Index="1">1.2724278919748</Axon>

<Axon Index="2">1.15759189853478</Axon>

<Axon Index="3">0.94154913736795</Axon>

<Axon Index="4">-3.16647408688931</Axon>

<Axon Index="5">0.714129422745947</Axon>

<Axon Index="6">-5.20683821412525</Axon>

<Axon Index="7">0.655364246293696</Axon>

<Axon Index="8">-2.49802893665249</Axon>

<Axon Index="9">-2.14401762106516</Axon>

<Axon Index="10">-1.71273520331939</Axon>

<Axon Index="11">-0.1065431273242</Axon>

<Axon Index="12">-2.85716635172937</Axon>

<Axon Index="13">-0.123414471300718</Axon>

<Axon Index="14">-0.855113130256214</Axon>

<Axon Index="15">-0.915073703920363</Axon>

<Axon Index="16">-2.95101749228471</Axon>

<Axon Index="17">0.379564225971199</Axon>

<Axon Index="18">-4.08933208118522</Axon>

<Axon Index="19">-1.52641824908283</Axon>

<Axon Index="20">-3.92420739274448</Axon>

<Axon Index="21">-0.163807667660777</Axon>

<Axon Index="22">-0.0547527156507549</Axon>

<Axon Index="23">-0.265549438464416</Axon>

<Axon Index="24">-2.84560918301331</Axon>

</Node>

<Node Index="15" Bias="-17.4060467903864">

<Axon Index="0">-18.4114604007554</Axon>

<Axon Index="1">-2.21599907483159</Axon>

<Axon Index="2">-0.885909799427943</Axon>

<Axon Index="3">-1.92152331223787</Axon>

<Axon Index="4">-20.1687420389428</Axon>

<Axon Index="5">0.153972352158727</Axon>

<Axon Index="6">-18.0911854998705</Axon>

<Axon Index="7">1.13578142265378</Axon>

<Axon Index="8">-18.080552953056</Axon>

<Axon Index="9">-1.6371494844129</Axon>

<Axon Index="10">0.199217128743355</Axon>

<Axon Index="11">0.141198864907596</Axon>

<Axon Index="12">-16.737822403651</Axon>

<Axon Index="13">1.48063335767988</Axon>

<Axon Index="14">1.10218225479444</Axon>

<Axon Index="15">-3.15115184695752</Axon>

<Axon Index="16">-15.7858751077817</Axon>

<Axon Index="17">-0.524318948773623</Axon>

<Axon Index="18">-18.3665658375146</Axon>

<Axon Index="19">0.4891871568222</Axon>

<Axon Index="20">-19.2026198478688</Axon>

<Axon Index="21">-0.84635277806716</Axon>

<Axon Index="22">0.0364671040287403</Axon>

<Axon Index="23">-2.1552770140136</Axon>

<Axon Index="24">-18.1149121947736</Axon>

</Node>

<Node Index="16" Bias="-5.29731508442973">

<Axon Index="0">-6.28014101973145</Axon>

<Axon Index="1">0.406527603537761</Axon>

<Axon Index="2">-0.974835455214333</Axon>

<Axon Index="3">0.130675503033458</Axon>

<Axon Index="4">-4.72091837610023</Axon>

<Axon Index="5">-1.43161233042865</Axon>

<Axon Index="6">-4.47044265510413</Axon>

<Axon Index="7">-1.42380253007556</Axon>

<Axon Index="8">-4.48216952263156</Axon>

<Axon Index="9">-0.72786982359384</Axon>

<Axon Index="10">0.551215784326</Axon>

<Axon Index="11">0.296138698284203</Axon>

<Axon Index="12">-5.50050291494209</Axon>

<Axon Index="13">-1.04321455965598</Axon>

<Axon Index="14">-0.210824782478519</Axon>

<Axon Index="15">1.81645634228792</Axon>

<Axon Index="16">-3.97067710183033</Axon>

<Axon Index="17">-1.2119804762366</Axon>

<Axon Index="18">-4.8174928471362</Axon>

<Axon Index="19">-0.0934121063794198</Axon>

<Axon Index="20">-5.15402190496571</Axon>

<Axon Index="21">-0.773103383913275</Axon>

<Axon Index="22">-1.7470627290294</Axon>

<Axon Index="23">-1.23295417767235</Axon>

<Axon Index="24">-4.13516963406846</Axon>

</Node>

<Node Index="17" Bias="-7.01163296884185">

<Axon Index="0">-6.39980329110232</Axon>

<Axon Index="1">0.0172723795222547</Axon>

<Axon Index="2">1.21933211788108</Axon>

<Axon Index="3">0.00403352790876059</Axon>

<Axon Index="4">-6.40928469147251</Axon>

<Axon Index="5">-0.404282850377684</Axon>

<Axon Index="6">-7.81921212978589</Axon>

<Axon Index="7">2.11158795008259</Axon>

<Axon Index="8">-7.56305978882887</Axon>

<Axon Index="9">-0.0651157743455472</Axon>

<Axon Index="10">1.82167353661037</Axon>

<Axon Index="11">1.28108727383124</Axon>

<Axon Index="12">-5.34903526934347</Axon>

<Axon Index="13">0.597057859790624</Axon>

<Axon Index="14">1.6864409806685</Axon>

<Axon Index="15">0.741940559803878</Axon>

<Axon Index="16">-8.42214286899242</Axon>

<Axon Index="17">0.511589282723161</Axon>

<Axon Index="18">-8.04750674080792</Axon>

<Axon Index="19">-0.563592669021198</Axon>

<Axon Index="20">-9.16748588375951</Axon>

<Axon Index="21">0.866340356931745</Axon>

<Axon Index="22">0.245554721542316</Axon>

<Axon Index="23">-1.03029754338171</Axon>

<Axon Index="24">-8.58801576357043</Axon>

</Node>

<Node Index="18" Bias="-0.860685936182803">

<Axon Index="0">0.438775553563825</Axon>

<Axon Index="1">0.821119329801366</Axon>

<Axon Index="2">0.994238358809474</Axon>

<Axon Index="3">-1.06559357157599</Axon>

<Axon Index="4">-1.62513630527393</Axon>

<Axon Index="5">1.18662091589702</Axon>

<Axon Index="6">-0.909279314428003</Axon>

<Axon Index="7">1.15472914987924</Axon>

<Axon Index="8">-1.5056622064607</Axon>

<Axon Index="9">0.250823655092151</Axon>

<Axon Index="10">-1.77624994140624</Axon>

<Axon Index="11">-1.47025005128385</Axon>

<Axon Index="12">-3.04934001682321</Axon>

<Axon Index="13">-0.617985787671791</Axon>

<Axon Index="14">-0.737893863343888</Axon>

<Axon Index="15">1.11132718598605</Axon>

<Axon Index="16">1.39540913162612</Axon>

<Axon Index="17">-1.09648296051105</Axon>

<Axon Index="18">1.27300746272372</Axon>

<Axon Index="19">0.0797850179894187</Axon>

<Axon Index="20">-0.526479679990743</Axon>

<Axon Index="21">0.544190531980361</Axon>

<Axon Index="22">0.768559344717307</Axon>

<Axon Index="23">-0.770265521185819</Axon>

<Axon Index="24">-1.02587593611962</Axon>

</Node>

<Node Index="19" Bias="-2.59914340987532">

<Axon Index="0">-0.913221352474641</Axon>

<Axon Index="1">0.60830143590283</Axon>

<Axon Index="2">0.535402669798838</Axon>

<Axon Index="3">-1.25148725494967</Axon>

<Axon Index="4">-1.20150286397743</Axon>

<Axon Index="5">0.419843591338135</Axon>

<Axon Index="6">1.22703820317256</Axon>

<Axon Index="7">0.0815177874236648</Axon>

<Axon Index="8">-0.122559977064874</Axon>

<Axon Index="9">0.0869857061003504</Axon>

<Axon Index="10">-2.87749419838305</Axon>

<Axon Index="11">-0.753492551688048</Axon>

<Axon Index="12">-0.440053074436429</Axon>

<Axon Index="13">-0.730941417418959</Axon>

<Axon Index="14">0.921262145673638</Axon>

<Axon Index="15">-1.05279446382147</Axon>

<Axon Index="16">-1.31249265208943</Axon>

<Axon Index="17">-0.894406819564453</Axon>

<Axon Index="18">0.113089327794534</Axon>

<Axon Index="19">-1.2763603292107</Axon>

<Axon Index="20">0.731701403079821</Axon>

<Axon Index="21">-0.240967485530374</Axon>

<Axon Index="22">-0.0681784410630867</Axon>

<Axon Index="23">0.722677995008549</Axon>

<Axon Index="24">-0.889715762489039</Axon>

</Node>

<Node Index="20" Bias="-1.4088545066943">

<Axon Index="0">-1.30694956694364</Axon>

<Axon Index="1">-1.40247534369603</Axon>

<Axon Index="2">0.34535223504549</Axon>

<Axon Index="3">-0.125802478434963</Axon>

<Axon Index="4">-1.6318039494989</Axon>

<Axon Index="5">-1.02418311421855</Axon>

<Axon Index="6">0.307336203486577</Axon>

<Axon Index="7">-1.52365696035262</Axon>

<Axon Index="8">1.49514516414516</Axon>

<Axon Index="9">1.43548966050188</Axon>

<Axon Index="10">-1.48307225368779</Axon>

<Axon Index="11">-0.490440412615182</Axon>

<Axon Index="12">-1.56707561643399</Axon>

<Axon Index="13">-2.42937919917</Axon>

<Axon Index="14">-1.93631285369687</Axon>

<Axon Index="15">0.67890181637364</Axon>

<Axon Index="16">-2.49531316058175</Axon>

<Axon Index="17">0.252974780894212</Axon>

<Axon Index="18">-1.43295824107566</Axon>

<Axon Index="19">-0.766986748860128</Axon>

<Axon Index="20">-1.52506663607253</Axon>

<Axon Index="21">-0.00561823529527318</Axon>

<Axon Index="22">0.725547321585455</Axon>

<Axon Index="23">1.12173022108979</Axon>

<Axon Index="24">-0.0969931566059694</Axon>

</Node>

<Node Index="21" Bias="-9.94919420322447">

<Axon Index="0">-8.64968046782465</Axon>

<Axon Index="1">-0.484808398505554</Axon>

<Axon Index="2">-0.727359241575801</Axon>

<Axon Index="3">-1.33658856743852</Axon>

<Axon Index="4">-9.62007366747621</Axon>

<Axon Index="5">0.928835450886131</Axon>

<Axon Index="6">-10.4927161930262</Axon>

<Axon Index="7">-0.679033173097553</Axon>

<Axon Index="8">-6.9460906512551</Axon>

<Axon Index="9">-0.0523768790020642</Axon>

<Axon Index="10">-1.95211770778217</Axon>

<Axon Index="11">-3.25860117230145</Axon>

<Axon Index="12">-9.51664008592694</Axon>

<Axon Index="13">-2.00491605657125</Axon>

<Axon Index="14">-0.795794755391777</Axon>

<Axon Index="15">0.246450105232215</Axon>

<Axon Index="16">-8.60568795676572</Axon>

<Axon Index="17">-0.999133443855226</Axon>

<Axon Index="18">-9.05985814217411</Axon>

<Axon Index="19">1.014856190109</Axon>

<Axon Index="20">-7.97721128643036</Axon>

<Axon Index="21">0.948491599680659</Axon>

<Axon Index="22">2.32115277619282</Axon>

<Axon Index="23">-0.13966865771329</Axon>

<Axon Index="24">-9.05335697257517</Axon>

</Node>

<Node Index="22" Bias="-2.70701674700152">

<Axon Index="0">-0.70093792640685</Axon>

<Axon Index="1">-0.878743333094617</Axon>

<Axon Index="2">0.220827078370195</Axon>

<Axon Index="3">0.643728131023444</Axon>

<Axon Index="4">0.31564132062122</Axon>

<Axon Index="5">0.176463058362924</Axon>

<Axon Index="6">1.4770009702165</Axon>

<Axon Index="7">-0.00155889700378025</Axon>

<Axon Index="8">0.27426198099973</Axon>

<Axon Index="9">0.770038978638174</Axon>

<Axon Index="10">-3.26430914745899</Axon>

<Axon Index="11">-4.46509530513714</Axon>

<Axon Index="12">-3.24544530490158</Axon>

<Axon Index="13">-2.2883792933999</Axon>

<Axon Index="14">-2.15122156768342</Axon>

<Axon Index="15">0.0982934219095892</Axon>

<Axon Index="16">-1.16407266765785</Axon>

<Axon Index="17">-0.588658825766337</Axon>

<Axon Index="18">-0.0996515648300181</Axon>

<Axon Index="19">0.215972078711717</Axon>

<Axon Index="20">0.815498348721015</Axon>

<Axon Index="21">-1.52914355715368</Axon>

<Axon Index="22">0.963363719191583</Axon>

<Axon Index="23">-0.196904293570965</Axon>

<Axon Index="24">-1.75649781693654</Axon>

</Node>

<Node Index="23" Bias="-2.21991527491737">

<Axon Index="0">-1.74861201224999</Axon>

<Axon Index="1">0.0305635966859615</Axon>

<Axon Index="2">0.477742540402027</Axon>

<Axon Index="3">1.21417114148775</Axon>

<Axon Index="4">-0.728587292425253</Axon>

<Axon Index="5">0.0220874347923832</Axon>

<Axon Index="6">-0.426258797873244</Axon>

<Axon Index="7">-0.839859344893708</Axon>

<Axon Index="8">1.36153059000611</Axon>

<Axon Index="9">-1.17726878194981</Axon>

<Axon Index="10">-2.89252430038937</Axon>

<Axon Index="11">-1.49741987849581</Axon>

<Axon Index="12">-1.09565050244502</Axon>

<Axon Index="13">-2.3274903409853</Axon>

<Axon Index="14">-3.67716547920713</Axon>

<Axon Index="15">-0.48986441590908</Axon>

<Axon Index="16">-0.273930951524854</Axon>

<Axon Index="17">0.12906479965328</Axon>

<Axon Index="18">-0.352331294831807</Axon>

<Axon Index="19">1.20801255614565</Axon>

<Axon Index="20">-0.602924276297846</Axon>

<Axon Index="21">-1.18769939663212</Axon>

<Axon Index="22">1.12352076522759</Axon>

<Axon Index="23">-2.19745434316195</Axon>

<Axon Index="24">-1.07378201295571</Axon>

</Node>

<Node Index="24" Bias="-2.58696355558737">

<Axon Index="0">-0.740753727352748</Axon>

<Axon Index="1">-0.571038543691505</Axon>

<Axon Index="2">-1.44437727274449</Axon>

<Axon Index="3">0.133665795859953</Axon>

<Axon Index="4">-2.12907122129708</Axon>

<Axon Index="5">-0.0856508959375375</Axon>

<Axon Index="6">-1.14658715952061</Axon>

<Axon Index="7">1.2172207165859</Axon>

<Axon Index="8">-0.685685527975857</Axon>

<Axon Index="9">-0.768132984421801</Axon>

<Axon Index="10">-0.379782748085692</Axon>

<Axon Index="11">-0.98055373006903</Axon>

<Axon Index="12">-2.75555586661517</Axon>

<Axon Index="13">-1.02531317912848</Axon>

<Axon Index="14">-0.791834372375099</Axon>

<Axon Index="15">-1.60108638786842</Axon>

<Axon Index="16">-1.31209967311475</Axon>

<Axon Index="17">-0.533827073939777</Axon>

<Axon Index="18">-0.187776252119472</Axon>

<Axon Index="19">0.785510076888384</Axon>

<Axon Index="20">-0.215019512820494</Axon>

<Axon Index="21">1.33787165893313</Axon>

<Axon Index="22">-0.289177697709255</Axon>

<Axon Index="23">-0.321640412631962</Axon>

<Axon Index="24">0.0856639704870689</Axon>

</Node>

<Node Index="25" Bias="-8.38751306974343">

<Axon Index="0">0.662857887733751</Axon>

<Axon Index="1">1.06459374085627</Axon>

<Axon Index="2">-10.2950872280115</Axon>

<Axon Index="3">-1.09154359416605</Axon>

<Axon Index="4">0.270064964555085</Axon>

<Axon Index="5">-0.13938809189913</Axon>

<Axon Index="6">-1.26643165426691</Axon>

<Axon Index="7">-0.311848153579093</Axon>

<Axon Index="8">0.415686732124824</Axon>

<Axon Index="9">-0.152741696310159</Axon>

<Axon Index="10">-10.4623868225108</Axon>

<Axon Index="11">-9.89175832073866</Axon>

<Axon Index="12">-9.99264402920465</Axon>

<Axon Index="13">-9.71750997757567</Axon>

<Axon Index="14">-10.5928620208249</Axon>

<Axon Index="15">0.394839024834272</Axon>

<Axon Index="16">-0.276702287872518</Axon>

<Axon Index="17">0.0733918558073011</Axon>

<Axon Index="18">0.881930664734501</Axon>

<Axon Index="19">-0.558168885288053</Axon>

<Axon Index="20">0.333043199189405</Axon>

<Axon Index="21">-0.949729806007287</Axon>

<Axon Index="22">-11.2463481655846</Axon>

<Axon Index="23">-0.924765558724602</Axon>

<Axon Index="24">0.494951090557024</Axon>

</Node>

<Node Index="26" Bias="-1.3025683540226">

<Axon Index="0">-0.351838508344887</Axon>

<Axon Index="1">0.430232326322502</Axon>

<Axon Index="2">-1.82322252371092</Axon>

<Axon Index="3">-1.10118593139124</Axon>

<Axon Index="4">-0.880457340115022</Axon>

<Axon Index="5">-0.190251774930843</Axon>

<Axon Index="6">-0.502973933216623</Axon>

<Axon Index="7">-0.870812410026793</Axon>

<Axon Index="8">-0.981311620085512</Axon>

<Axon Index="9">-0.606911851735993</Axon>

<Axon Index="10">-0.673877102273537</Axon>

<Axon Index="11">-1.87465410834229</Axon>

<Axon Index="12">-1.96092047155703</Axon>

<Axon Index="13">-0.566788734492332</Axon>

<Axon Index="14">0.392372966652934</Axon>

<Axon Index="15">0.203693723193286</Axon>

<Axon Index="16">-0.806779953089356</Axon>

<Axon Index="17">0.120949828800499</Axon>

<Axon Index="18">-0.430290114625257</Axon>

<Axon Index="19">-0.486226940478862</Axon>

<Axon Index="20">-1.57305012083525</Axon>

<Axon Index="21">0.992488926616584</Axon>

<Axon Index="22">0.664546793767858</Axon>

<Axon Index="23">1.89439439528927</Axon>

<Axon Index="24">-3.41338476704493</Axon>

</Node>

<Node Index="27" Bias="-0.690763088383429">

<Axon Index="0">2.39753332566118</Axon>

<Axon Index="1">0.631898341053712</Axon>

<Axon Index="2">-0.190038896482219</Axon>

<Axon Index="3">1.19858720095832</Axon>

<Axon Index="4">3.14774245839859</Axon>

<Axon Index="5">-0.197110368069232</Axon>

<Axon Index="6">0.193719548943555</Axon>

<Axon Index="7">-1.34280328584402</Axon>

<Axon Index="8">1.23238350841259</Axon>

<Axon Index="9">0.177277821912371</Axon>

<Axon Index="10">-2.05908754305229</Axon>

<Axon Index="11">-2.59702402239331</Axon>

<Axon Index="12">-1.31593902594136</Axon>

<Axon Index="13">-2.84674831767539</Axon>

<Axon Index="14">-2.6294092818382</Axon>

<Axon Index="15">-0.163028728384066</Axon>

<Axon Index="16">1.94340867626344</Axon>

<Axon Index="17">-0.909853264922661</Axon>

<Axon Index="18">0.378503722217735</Axon>

<Axon Index="19">-0.743305822177375</Axon>

<Axon Index="20">1.50922748442442</Axon>

<Axon Index="21">-0.271327535934182</Axon>

<Axon Index="22">-1.49571446006625</Axon>

<Axon Index="23">-0.702227147621883</Axon>

<Axon Index="24">2.01010820669622</Axon>

</Node>

<Node Index="28" Bias="-4.65677280105047">

<Axon Index="0">-1.2852857447258</Axon>

<Axon Index="1">0.712682950442201</Axon>

<Axon Index="2">0.410129724206949</Axon>

<Axon Index="3">-1.09079097053059</Axon>

<Axon Index="4">-1.22555825924206</Axon>

<Axon Index="5">0.51287349656806</Axon>

<Axon Index="6">0.290334587896786</Axon>

<Axon Index="7">-1.33245494142261</Axon>

<Axon Index="8">0.0244500275361658</Axon>

<Axon Index="9">1.01764287186463</Axon>

<Axon Index="10">-3.44407275265849</Axon>

<Axon Index="11">-3.55389197321192</Axon>

<Axon Index="12">-4.06468974007915</Axon>

<Axon Index="13">-2.02350898917322</Axon>

<Axon Index="14">-3.00562745902735</Axon>

<Axon Index="15">-0.147511272699567</Axon>

<Axon Index="16">0.596364952646273</Axon>

<Axon Index="17">0.239167926943643</Axon>

<Axon Index="18">1.44729138259507</Axon>

<Axon Index="19">0.673600216394922</Axon>

<Axon Index="20">-1.08528800230052</Axon>

<Axon Index="21">-0.287976068782095</Axon>

<Axon Index="22">0.862438129523227</Axon>

<Axon Index="23">0.795171889464587</Axon>

<Axon Index="24">-0.572779068875805</Axon>

</Node>

<Node Index="29" Bias="-1.93186130333546">

<Axon Index="0">-0.53394870482091</Axon>

<Axon Index="1">1.06448864600559</Axon>

<Axon Index="2">0.118178873635158</Axon>

<Axon Index="3">-0.73540888871995</Axon>

<Axon Index="4">0.823969909416874</Axon>

<Axon Index="5">-0.895132696252306</Axon>

<Axon Index="6">-0.900531865379596</Axon>

<Axon Index="7">1.6480066225876</Axon>

<Axon Index="8">-1.39844735515334</Axon>

<Axon Index="9">-0.479264521596081</Axon>

<Axon Index="10">-2.90411300711768</Axon>

<Axon Index="11">-1.00012944475043</Axon>

<Axon Index="12">-1.95569706612358</Axon>

<Axon Index="13">-2.37099726149284</Axon>

<Axon Index="14">-1.94446858196102</Axon>

<Axon Index="15">-0.76651278290125</Axon>

<Axon Index="16">-1.71601439305909</Axon>

<Axon Index="17">-0.0208233915700861</Axon>

<Axon Index="18">0.696223604111053</Axon>

<Axon Index="19">0.0224086252224825</Axon>

<Axon Index="20">-2.54844449688565</Axon>

<Axon Index="21">-0.349774671978048</Axon>

<Axon Index="22">-0.273839496619256</Axon>

<Axon Index="23">-0.162732562131324</Axon>

<Axon Index="24">-0.901198427346438</Axon>

</Node>

<Node Index="30" Bias="-4.60956390740102">

<Axon Index="0">-3.57890990502855</Axon>

<Axon Index="1">1.55959099646575</Axon>

<Axon Index="2">0.145499720359083</Axon>

<Axon Index="3">-0.469998411996888</Axon>

<Axon Index="4">-5.08704371157047</Axon>

<Axon Index="5">-1.51473751720672</Axon>

<Axon Index="6">-5.16112793539708</Axon>

<Axon Index="7">1.0018365043551</Axon>

<Axon Index="8">-4.26685448232712</Axon>

<Axon Index="9">1.86770446919408</Axon>

<Axon Index="10">-1.38946543637859</Axon>

<Axon Index="11">-0.512428682584751</Axon>

<Axon Index="12">-3.68639460978857</Axon>

<Axon Index="13">-0.956960883529063</Axon>

<Axon Index="14">-1.13055733812617</Axon>

<Axon Index="15">0.71823847585132</Axon>

<Axon Index="16">-4.87609698056957</Axon>

<Axon Index="17">-2.02864232344841</Axon>

<Axon Index="18">-2.20099676085341</Axon>

<Axon Index="19">-0.303795047603355</Axon>

<Axon Index="20">-5.53817246823983</Axon>

<Axon Index="21">0.0574137754860725</Axon>

<Axon Index="22">-0.194260912784753</Axon>

<Axon Index="23">1.66063799586861</Axon>

<Axon Index="24">-5.16567495779162</Axon>

</Node>

<Node Index="31" Bias="1.0133681940743">

<Axon Index="0">-0.375287861728916</Axon>

<Axon Index="1">0.73876782675236</Axon>

<Axon Index="2">0.0781205281716474</Axon>

<Axon Index="3">-0.253324932764015</Axon>

<Axon Index="4">-0.299722722557251</Axon>

<Axon Index="5">-1.6189312163378</Axon>

<Axon Index="6">-0.0473016982011103</Axon>

<Axon Index="7">0.0683807404639437</Axon>

<Axon Index="8">-1.07592249577769</Axon>

<Axon Index="9">0.528656267628959</Axon>

<Axon Index="10">1.08561818632768</Axon>

<Axon Index="11">0.401958319531009</Axon>

<Axon Index="12">1.78091525211659</Axon>

<Axon Index="13">-0.953797264653171</Axon>

<Axon Index="14">1.98652784035415</Axon>

<Axon Index="15">-0.324631922458155</Axon>

<Axon Index="16">0.498883781474549</Axon>

<Axon Index="17">0.840262447174389</Axon>

<Axon Index="18">-2.09178886908909</Axon>

<Axon Index="19">-0.485360588531488</Axon>

<Axon Index="20">-0.619945868087892</Axon>

<Axon Index="21">-0.169595538513482</Axon>

<Axon Index="22">2.0404950840278</Axon>

<Axon Index="23">0.603498228098262</Axon>

<Axon Index="24">-2.00181612185059</Axon>

</Node>

<Node Index="32" Bias="-7.37175051839544">

<Axon Index="0">-8.19901462461553</Axon>

<Axon Index="1">0.0599749580039845</Axon>

<Axon Index="2">-0.260069359233309</Axon>

<Axon Index="3">0.364675792856313</Axon>

<Axon Index="4">-9.62232892389952</Axon>

<Axon Index="5">0.152825475674399</Axon>

<Axon Index="6">-8.37992248036225</Axon>

<Axon Index="7">-0.102332886020999</Axon>

<Axon Index="8">-12.0149667753216</Axon>

<Axon Index="9">-0.050055177120753</Axon>

<Axon Index="10">0.651005927827655</Axon>

<Axon Index="11">0.261033192342668</Axon>

<Axon Index="12">-6.11999949622523</Axon>

<Axon Index="13">-1.75230262637358</Axon>

<Axon Index="14">-0.197527885222422</Axon>

<Axon Index="15">-0.807274200958168</Axon>

<Axon Index="16">-8.79877726209056</Axon>

<Axon Index="17">-1.96966817224077</Axon>

<Axon Index="18">-8.44485090375832</Axon>

<Axon Index="19">0.0960457291001737</Axon>

<Axon Index="20">-9.58069669667607</Axon>

<Axon Index="21">-2.01657229241406</Axon>

<Axon Index="22">-0.480144811715975</Axon>

<Axon Index="23">-0.108499293787181</Axon>

<Axon Index="24">-8.44749431986203</Axon>

</Node>

<Node Index="33" Bias="-3.17935561869814">

<Axon Index="0">-3.69289115686713</Axon>

<Axon Index="1">-0.235152913485513</Axon>

<Axon Index="2">-1.17683071450062</Axon>

<Axon Index="3">-0.651175343504287</Axon>

<Axon Index="4">-4.2412870354677</Axon>

<Axon Index="5">0.0919347798361085</Axon>

<Axon Index="6">-3.30038246211189</Axon>

<Axon Index="7">-1.30054133954228</Axon>

<Axon Index="8">-3.79169098599103</Axon>

<Axon Index="9">0.809264867987649</Axon>

<Axon Index="10">-1.12821830452484</Axon>

<Axon Index="11">-0.511068187451703</Axon>

<Axon Index="12">-5.0412122909633</Axon>

<Axon Index="13">1.15236534320013</Axon>

<Axon Index="14">1.31500424197348</Axon>

<Axon Index="15">-0.0375184174727438</Axon>

<Axon Index="16">-5.22921837170987</Axon>

<Axon Index="17">-1.12371120274009</Axon>

<Axon Index="18">-3.43954152226907</Axon>

<Axon Index="19">0.648829256409355</Axon>

<Axon Index="20">-4.86832139398938</Axon>

<Axon Index="21">-0.29122801046413</Axon>

<Axon Index="22">-1.07763544382211</Axon>

<Axon Index="23">0.889398979216386</Axon>

<Axon Index="24">-5.11818970771051</Axon>

</Node>

<Node Index="34" Bias="-0.310555750055795">

<Axon Index="0">3.10377541667846</Axon>

<Axon Index="1">-0.478884142055411</Axon>

<Axon Index="2">-1.13631886506112</Axon>

<Axon Index="3">0.106966079915831</Axon>

<Axon Index="4">2.396934531651</Axon>

<Axon Index="5">-0.344199700671821</Axon>

<Axon Index="6">0.607753830457342</Axon>

<Axon Index="7">1.30063049725425</Axon>

<Axon Index="8">0.332860904312326</Axon>

<Axon Index="9">-1.93356528852784</Axon>

<Axon Index="10">-1.47256305820266</Axon>

<Axon Index="11">-1.31558420212306</Axon>

<Axon Index="12">-2.06712972922523</Axon>

<Axon Index="13">-2.05626072316888</Axon>

<Axon Index="14">-0.190689612066529</Axon>

<Axon Index="15">-0.376250254240895</Axon>

<Axon Index="16">0.0626819992938328</Axon>

<Axon Index="17">-0.58436871868401</Axon>

<Axon Index="18">0.093851540289089</Axon>

<Axon Index="19">0.0171962344011889</Axon>

<Axon Index="20">0.643361682258399</Axon>

<Axon Index="21">0.818010954267341</Axon>

<Axon Index="22">-0.172783510935112</Axon>

<Axon Index="23">-1.51464404430078</Axon>

<Axon Index="24">-0.058663783311394</Axon>

</Node>

</Layer>

<Layer Index="1">

<Node Index="0" Bias="-1.06841932030418">

<Axon Index="0">-1.34831064533676</Axon>

<Axon Index="1">0.611735750898982</Axon>

<Axon Index="2">-0.173086111183722</Axon>

<Axon Index="3">1.02301446174971</Axon>

<Axon Index="4">0.290415867249723</Axon>

<Axon Index="5">-0.347748796866847</Axon>

<Axon Index="6">-0.0159709185398506</Axon>

<Axon Index="7">3.26959647100453</Axon>

<Axon Index="8">1.15582962527197</Axon>

<Axon Index="9">-2.85248492382027</Axon>

<Axon Index="10">2.44427833692091</Axon>

<Axon Index="11">-0.457594959908802</Axon>

<Axon Index="12">-3.01069251224585</Axon>

<Axon Index="13">-2.33050425870271</Axon>

<Axon Index="14">-1.72782873857762</Axon>

<Axon Index="15">1.85465444970771</Axon>

<Axon Index="16">1.12702772089446</Axon>

<Axon Index="17">2.39177898653198</Axon>

<Axon Index="18">-1.75955496563017</Axon>

<Axon Index="19">0.583671784581639</Axon>

<Axon Index="20">-2.17870811753403</Axon>

<Axon Index="21">0.478089594226144</Axon>

<Axon Index="22">-0.621634797125174</Axon>

<Axon Index="23">-1.00775963746657</Axon>

<Axon Index="24">1.16103985632719</Axon>

<Axon Index="25">-4.16115024845402</Axon>

<Axon Index="26">1.34450287280533</Axon>

<Axon Index="27">-0.377242149505362</Axon>

<Axon Index="28">0.487425005146716</Axon>

<Axon Index="29">0.990142903473152</Axon>

<Axon Index="30">1.55339014579445</Axon>

<Axon Index="31">-1.96306848424832</Axon>

<Axon Index="32">-0.449984597953527</Axon>

<Axon Index="33">-2.69293737080783</Axon>

<Axon Index="34">1.51260989740014</Axon>

</Node>

<Node Index="1" Bias="-0.627487368160925">

<Axon Index="0">-3.48676489325426</Axon>

<Axon Index="1">-3.51383955231528</Axon>

<Axon Index="2">5.24383510679005</Axon>

<Axon Index="3">0.285887287856482</Axon>

<Axon Index="4">-1.9397029413245</Axon>

<Axon Index="5">-0.191205505315413</Axon>

<Axon Index="6">-1.18694628524739</Axon>

<Axon Index="7">1.00469773817483</Axon>

<Axon Index="8">-0.421753964620727</Axon>

<Axon Index="9">6.38583458597078</Axon>

<Axon Index="10">-2.82199473442636</Axon>

<Axon Index="11">3.74951665979905</Axon>

<Axon Index="12">3.33690381101439</Axon>

<Axon Index="13">6.96403085287739</Axon>

<Axon Index="14">5.91825320704613</Axon>

<Axon Index="15">0.336704105850016</Axon>

<Axon Index="16">5.13550182064327</Axon>

<Axon Index="17">-5.22677462601172</Axon>

<Axon Index="18">-0.858776655863144</Axon>

<Axon Index="19">-0.721455951163624</Axon>

<Axon Index="20">5.45279068313639</Axon>

<Axon Index="21">-0.922310814332511</Axon>

<Axon Index="22">-1.92252255119594</Axon>

<Axon Index="23">-0.567904455386724</Axon>

<Axon Index="24">-2.27381786772077</Axon>

<Axon Index="25">7.96713678549391</Axon>

<Axon Index="26">0.140434937727123</Axon>

<Axon Index="27">2.9659546693208</Axon>

<Axon Index="28">-0.664891423062856</Axon>

<Axon Index="29">-4.63314510302613</Axon>

<Axon Index="30">-3.77036987200149</Axon>

<Axon Index="31">-0.275063796911492</Axon>

<Axon Index="32">2.35218591212191</Axon>

<Axon Index="33">5.33258592133823</Axon>

<Axon Index="34">-0.330812153273954</Axon>

</Node>

<Node Index="2" Bias="1.52082566921152">

<Axon Index="0">1.56227940057929</Axon>

<Axon Index="1">2.29119775311735</Axon>

<Axon Index="2">-5.42681107757538</Axon>

<Axon Index="3">0.533907467124282</Axon>

<Axon Index="4">1.3126765235812</Axon>

<Axon Index="5">-0.102051385205353</Axon>

<Axon Index="6">2.51789077424805</Axon>

<Axon Index="7">-4.61617427874826</Axon>

<Axon Index="8">-0.348664478100975</Axon>

<Axon Index="9">-5.8254491030869</Axon>

<Axon Index="10">2.74119183632476</Axon>

<Axon Index="11">-2.50096240350264</Axon>

<Axon Index="12">-2.50175924742642</Axon>

<Axon Index="13">-3.39157013443227</Axon>

<Axon Index="14">-3.98375018907066</Axon>

<Axon Index="15">0.357253086458266</Axon>

<Axon Index="16">-2.48244541809822</Axon>

<Axon Index="17">4.82695775623579</Axon>

<Axon Index="18">-1.7422829607998</Axon>

<Axon Index="19">-0.363148002329975</Axon>

<Axon Index="20">-3.13229661483337</Axon>

<Axon Index="21">1.89459799819654</Axon>

<Axon Index="22">-0.562431189157589</Axon>

<Axon Index="23">-0.0265558755882198</Axon>

<Axon Index="24">4.11749895684488</Axon>

<Axon Index="25">-4.70291314789835</Axon>

<Axon Index="26">-0.362929762154676</Axon>

<Axon Index="27">-1.26192739179757</Axon>

<Axon Index="28">1.40338481090707</Axon>

<Axon Index="29">1.49756950932038</Axon>

<Axon Index="30">1.75601249794181</Axon>

<Axon Index="31">2.42105020612505</Axon>

<Axon Index="32">-0.816813943043076</Axon>

<Axon Index="33">-5.86506143500152</Axon>

<Axon Index="34">-0.361177450989594</Axon>

</Node>

<Node Index="3" Bias="0.541598424662086">

<Axon Index="0">1.09204371196237</Axon>

<Axon Index="1">0.229286201577638</Axon>

<Axon Index="2">-0.488001471161533</Axon>

<Axon Index="3">-2.06933902164873</Axon>

<Axon Index="4">1.9326983451857</Axon>

<Axon Index="5">0.0177715252569342</Axon>

<Axon Index="6">-0.229068070456402</Axon>

<Axon Index="7">-0.588890376123351</Axon>

<Axon Index="8">0.0499688559398374</Axon>

<Axon Index="9">-2.92180735780086</Axon>

<Axon Index="10">3.07755909454531</Axon>

<Axon Index="11">-2.15148927900814</Axon>

<Axon Index="12">-4.72029675914149</Axon>

<Axon Index="13">-4.15149109855629</Axon>

<Axon Index="14">-4.80119118420495</Axon>

<Axon Index="15">-1.15387475117772</Axon>

<Axon Index="16">-3.68842243881587</Axon>

<Axon Index="17">1.92728517904372</Axon>

<Axon Index="18">-0.0786290503710588</Axon>

<Axon Index="19">0.303514447726238</Axon>

<Axon Index="20">-2.810864086725</Axon>

<Axon Index="21">2.00673439193728</Axon>

<Axon Index="22">0.348151363805142</Axon>

<Axon Index="23">-0.116923325065879</Axon>

<Axon Index="24">3.08872097365984</Axon>

<Axon Index="25">-4.26506817702801</Axon>

<Axon Index="26">-0.218472906472264</Axon>

<Axon Index="27">-0.753287437408014</Axon>

<Axon Index="28">-0.650036464112209</Axon>

<Axon Index="29">1.29971281438167</Axon>

<Axon Index="30">2.76789859957061</Axon>

<Axon Index="31">1.27994723873625</Axon>

<Axon Index="32">0.191599996712169</Axon>

<Axon Index="33">-3.32147648396575</Axon>

<Axon Index="34">0.206938228201721</Axon>

</Node>

</Layer>

</Weights>

</NeuralNetwork>

## 

## Error.txt

0 3549.13979230015

1 520.191932013043

2 19.3527822255321

3 6.65555072768385

4 7.72691746651765

5 3.89148754628667

6 3.90279134935281

7 4.32957886435866

8 4.95069122871018

9 4.72716693370184

10 3.92256436465053

11 3.66225253130111

12 4.04285165844468

13 3.53703834492686

14 4.88570744136335

15 3.8121083937276

16 3.89118965696576

17 4.005107411419

18 4.74461866970999

19 4.18537143284707

20 4.19262805986954

21 3.56319412076204

22 3.69419256732148

23 3.72951447493015

24 3.85866376313706

25 3.23795568838753

26 3.77995573866233

27 3.6125708020149

28 3.56101120693403

29 3.93067755938933

30 4.02202218441942

31 4.62538791611593

32 3.49909135077535

33 4.65137668467452

34 4.45451247865359

35 3.76269770718376

36 3.98755115439126

37 3.39586386290459

38 3.53732726109422

39 4.25899499184716

40 4.30811017484635

41 4.06499237312814

42 4.6335459468162

43 4.19552956192977

44 4.11566754859124

45 3.70966689182424

46 4.41502275878106

47 3.32765195143242

48 3.87981109231649

49 4.12429799382391

50 3.42254492001952

51 3.51164051366452

52 3.5257186994421

53 4.02237424169595

54 3.78868063393815

55 4.54603007766138

56 3.66625391418939

57 4.48186721743822

58 3.92907612834574

59 4.23647395044326

60 4.22721015577015

61 3.43165413364944

62 3.97658134985699

63 4.12314372503731

64 3.6602477522007

65 3.23992574770595

66 4.42949509342522

67 3.1478213446233

68 3.38967057085416

69 3.83010535365156

70 4.31555300198197

71 3.60961702238902

72 3.89296897222185

73 4.27886066084399

74 4.22032757004395

75 4.24446456343754

76 4.06972357043647

77 3.47760421256437

78 4.34984017256556

79 3.02037953077964

80 3.87408611959658

81 3.35499905004837

82 4.28324465668209

83 3.70394836798815

84 3.30204749320774

85 3.44757276459306

86 3.77845174162402

87 3.66633222759592

88 4.12411640209734

89 3.47528460157518

90 3.30000255885682

91 3.42688914072448

92 3.44188634557551

93 3.21370507557702

94 3.12730972446668

95 3.40311999295199

96 4.05033725072126

97 3.19533188858342

98 2.95152584184133

99 4.0123914752084

100 3.96528234719783

101 3.48640111170484

102 3.994481013234

103 3.94712537252707

104 2.99831896627913

105 3.45647786036578

106 3.1882186972536

107 4.05552574776078

108 3.16290487953691

109 3.0477805519877

110 3.83519550260839

111 3.98819433022677

112 3.74741133080667

113 3.80431837673206

114 4.02353957550241

115 3.27596586165904

116 3.70955488513985

117 3.06452167679286

118 3.26382208130326

119 3.31997097435979

120 2.98644121602235

121 3.21291039523789

122 3.23435066581141

123 3.38856757034265

124 3.67956121496497

125 3.42948781457789

126 3.43434681589405

127 3.00694811253501

128 3.41210408010054

129 3.25356958999071

130 3.61955975851984

131 3.59084476606165

132 2.93254353942906

133 3.73631588177776

134 3.80023848018316

135 3.18435079756275

136 3.34331341047144

137 2.92827999019805

138 3.42407604797475

139 2.65976035963208

140 3.31798934178887

141 3.25687140014682

142 3.16150109637591

143 3.66602498635387

144 2.77184639038406

145 3.63742570419253

146 2.50658246979528

147 2.62625824691152

148 3.65914647954585

149 3.07193863536667

150 3.07091916380086

151 2.76489471399608

152 3.61507200532787

153 3.36246754506476

154 3.38032975859614

155 2.6200421882436

156 3.02078233502183

157 3.58933218793626

158 3.00904528258933

159 2.32618907780413

160 2.77782442460205

161 3.10570915984573

162 3.33338319606156

163 3.07259304580066

164 3.44556756313105

165 3.40098786984234

166 3.02677090804752

167 3.30400092146287

168 3.14827897167611

169 2.58506078301012

170 3.38043899042347

171 3.29928542031502

172 2.92061727800398

173 2.49874782971583

174 3.28094592214437

175 2.84749140247293

176 2.68969238193455

177 3.15610493276542

178 2.90747847517932

179 3.12355931506467

180 2.8099483087

181 3.1659115152748

182 2.30210710777596

183 3.27101680739952

184 2.40338507769474

185 3.35443058892169

186 2.32189771816457

187 3.13145080351978

188 2.50096167292919

189 2.70204814057222

190 2.42578503524203

191 3.11405942899693

192 2.50322584281121

193 2.65146350314669

194 3.04481343762796

195 2.58486969075182

196 2.97472161049225

197 2.25102484299791

198 2.44891316811895

199 2.7266146267456

200 2.74513744393609

201 2.26779640103306

202 2.95930320753579

203 2.46279544783946

204 3.16109330388756

205 2.63365875444937

206 1.9913743525195

207 3.32773434897825

208 2.41829553569054

209 2.88705238976347

210 2.31558699131624

211 2.88763580055075

212 2.68213740668694

213 2.18916659648107

214 3.00593312940697

215 2.68996466332139

216 2.91941745661304

217 2.42304483300396

218 2.87091933916499

219 1.93344000534341

220 2.74666250976301

221 1.76842950228421

222 3.06937758821912

223 2.47270450671485

224 2.73855046457403

225 2.36663924375538

226 2.08228017583257

227 2.14848719374017

228 2.63623872652713

229 2.97746168821821

230 2.85176909090071

231 2.23882148568757

232 2.5362067643232

233 1.87310509381977

234 3.0019614826691

235 2.37703806632459

236 2.35565579389589

237 2.33563286240618

238 2.61897610480993

239 2.36592261578357

240 2.10270486587681

241 2.59695859264486

242 2.4994415089003

243 2.0916730354173

244 2.64255178894937

245 2.50757258358836

246 2.06514738688191

247 1.8054328791002

248 2.84690908699887

249 2.10313250482874

250 1.53048990251619

251 2.70666925977875

252 2.56685520624161

253 1.58478470536205

254 2.29011369398775

255 2.73681045162014

256 1.99882233253153

257 1.37724117891046

258 2.12622016768005

259 1.86257275313156

260 1.42932056856825

261 1.88695265542241

262 1.99782361614241

263 2.37861692109105

264 1.25317717542292

265 1.8256828017848

266 1.90601488392524

267 1.22055440508182

268 2.38711728097156

269 1.95229824005298

270 2.28201403859747

271 1.80564087746052

272 1.92267180924639

273 2.19739640539476

274 1.72041936036469

275 2.11628417271672

276 1.98787821771496

277 1.79372947868633

278 1.78300519798068

279 2.14175668677205

280 2.01380162400961

281 1.23124610988081

282 1.47554451823886

283 2.5319698973777

284 1.96432962977556

285 1.27751852669407

286 2.1782234047045

287 2.3236795415293

288 1.38990441706042

289 1.77367497840032

290 2.20654694495451

291 1.76985405900663

292 2.01079538118717

293 1.67828410724848

294 1.89250434838035

295 1.96338432735944

296 1.96394214492642

297 1.9485422074386

298 1.18224342561426

299 1.62891101267584

300 1.64082174151731

301 1.70281735813306

302 2.0086319807573

303 2.03104588318938

304 2.14645663977696

305 1.76518933258969

306 1.44538322595388

307 1.0642092942956

308 1.26863890504944

309 2.33733944499843

310 1.44132682213256

311 2.27674172777974

312 1.624529540903

313 1.78399547953475

314 1.54926515187784

315 1.99222793539724

316 1.42996906053652

317 1.31702564755479

318 1.76091532478722

319 0.970203974324402

320 1.86074715445327

321 1.25080257012799

322 1.53473210742078

323 0.829170095995298

324 1.70360873807059

325 1.51194200458951

326 1.3494919937481

327 0.948860643090477

328 0.917491377639866

329 0.813391066490756

330 0.780637286721961

331 0.801432473363553

332 0.871739225763488

333 0.722169738553013

334 0.362834863184208

335 0.688059822508664

336 0.616289894757648

337 0.563343658162117

338 0.44779763813331

339 0.656522117138948

340 0.592725798936463

341 0.614177470762029

342 0.474899237984381

343 0.656722033876355

344 0.68876577948208

345 0.288919066579813

346 0.448434537707914

347 0.449843967993392

348 0.388122361445449

349 0.364412388516948

350 0.326343991920834

351 0.392198315853699

352 0.439846869116713

353 0.340430854397437

354 0.342986741523331

355 0.853986321601133

356 0.39171832241804

357 0.383222820418205

358 0.491951481125075

359 0.691391930462897

360 0.378153092797202

361 0.281661052602557

362 0.162316202144942

363 0.213655215754097

364 0.212906272982478

365 0.257421176427343

366 0.215447421835637

367 0.202235655389415

368 0.196872746451126

369 0.243985493097235

370 0.224893668466633

371 0.138772795564861

372 0.112866412423941

373 0.0881871692781294

374 0.0776923474371817

375 0.0741654478585492

376 0.136940660048245

377 0.0733292777295603

378 0.18058960669693

379 0.238444804553324

380 0.0600432853559064

381 0.0627329788725544

382 0.545984978901442

383 0.870384665787326

384 0.270270411181734

385 0.160454074087749

386 0.250570765074913

387 0.0721317941177802

388 0.0483574917136141

389 0.0986070942839473

390 0.154026410443311

391 0.126394489358471

392 0.299961768895649

393 0.0679931201409965

394 0.280326878469923

395 0.331081664719022

396 0.177201064355584

397 0.223175158974873

398 0.153442858449937

399 0.0209720773693228

400 0.0225762386729401

401 0.101113269124311

402 0.115335866450537

403 0.055624563698669

404 0.0368698403474356

405 0.0363769884406184

406 0.0201683636808075

407 0.00516815605826736

408 0.0124056069446286

409 0.0384375158426182

410 0.0554296635383375

411 0.0617859441982531

412 0.0711764744150292

413 0.023548605809176

414 0.0435310103264125

415 0.206888188651197

416 0.00693055453862696

417 0.0352632157225373

418 0.0543791704472331

419 0.0529458631759563

420 0.59224397110437

421 0.380472631386205

422 0.131905736172465

423 0.320831904321923

424 0.237814356716168

425 0.185352639362693

426 0.446234231053841

427 0.203300712635657

428 0.335972740875952

429 0.0833396868786224

430 0.0286803736776134

431 0.0658584311033604

432 0.0998855949673034

433 0.0213326758064837

434 0.085447490195853

435 0.0807096043021143

436 0.180438538307759

437 0.126592829553149

438 0.418461548956308

439 0.225824298262396

440 0.483855978669927

441 0.144185753789401

442 0.397213296020298

443 0.0101771351565012

444 0.020915378204462

445 0.0162658483203088

446 0.00495335537673791

447 0.00747566489646259

448 0.00465811924583269

449 0.00272434720255883

450 0.00592723917798545

451 0.0054393560017879

452 0.00154565110561966

453 0.00256448127804116

454 0.00793967676449315

455 0.00152620587090887

456 0.0012942509508474

457 0.0015319207500831

458 0.00538276140638311

459 0.00467090226860421

460 0.00446856594328246

461 0.00489997213818092

462 0.00145229460670362

463 0.00150220383096688

464 0.00912472629896602

465 0.0116358996972692

466 0.0430377115211808

467 0.00873145164923541

468 0.031175011678346

469 0.041827790106239

470 0.0309077569741469

471 0.0213240327432795

472 0.0219141039678868

473 0.0119752715010678

474 0.0375742626473944

475 0.0260152459220368

476 0.0224703311125032

477 0.0416939377367037

478 0.0667292720610423

479 0.148060084935075

480 0.00940065637950762

481 0.00509274691019054

482 0.00668212146481637

483 0.0058074504306006

484 0.00473239246695589

485 0.00255894155001568

486 0.00154431662017367

487 0.00341406995754135

488 0.0021919786146931

489 0.000853640986163044

Bibliography

[1] Engelbrecht, Andries P. Computational Intelligence: An Introduction, (2nd ed.). Wiley, 2007.

[2] "Sigmoid Function -- From Wolfram Mathworld". *Mathworld.wolfram.com*. N.p., 2017. Web. 1 Mar. 2017.

Activity Log – Bitler

|  |  |  |
| --- | --- | --- |
| Date | Time (mins) | Description |
| *1/9/17* | *60* | *Looked through Naïve GA* |
| *1/10/17* | *60* | *Looked through and worked Naïve GA* |
| *1/11/17* |  |  |
| *1/12/17* | *120* | *Research GA’s online for topic ideas* |
| *1/13/17* | *180* | *Read Ch 9 - GA* |
| *1/14/17* | *90* | *Read Ch 9 - GA* |
| *1/15/17* |  |  |
| *1/16/17* | *120* | *Begin constructing class structure of string unscramble program* |
| *1/17/17* | *120* | *Begin chromosome class* |
| *1/18/17* | *120* | *Continue working chromosome class, begin developing population class* |
| *1/19/17* |  |  |
| *1/20/17* |  |  |
| *1/21/17* | *60* | *Finish chromosome class* |
| *1/22/17* | *240* | *Continue working population class* |
| *1/23/17* | *120* | *Connector class* |
| *1/24/17* | *30* | *Come up with initial variable settings* |
| *1/25/17* |  |  |
| *1/26/17* | *240* | *Begin generating report* |
| *1/27/17* | *180* | *Put together test data* |
| *1/28/17* | *360* | *Conclude paper, graphs, etc* |
| *1/29/17* | *180* | *Final review of paper* |
| *1/30/17* | *15* | *Submit paper* |
| *1/31/17* |  |  |
| *2/1/17* | *90* | *Begin reading Ch 20 – Fuzzy sets* |
| *2/2/17* | *120* | *Continue reading Ch 20 – Fuzzy sets* |
| *2/3/17* | *120* | *Begin Ch 21 – Fuzzy Logic and Reasoning* |
| *2/4/17* |  |  |
| *2/5/17* | *120* | *Continue Ch 21 – Fuzzy Logic and Reasoning* |
| *2/6/17* | *90* | *Read Ch 22, 23 – Fuzzy controllers and Rough sets* |
| *2/7/17* | *120* | *Watch tutorials on fuzzy logic* |
| *2/8/17* | *120* | *Develop class structure for Heating/Cooling app* |
| *2/9/17* | *120* | *Work on Point class* |
| *2/10/17* | *180* | *Work on FuzzySet, MembershipFunction, and Region classes* |
| *2/11/17* | *120* | *Continue class development* |
| *2/12/17* |  |  |
| *2/13/17* | *300* | *Begin generating report* |
| *2/14/17* |  |  |
| *2/15/17* | *360* | *Continue report* |
| *2/16/17* | *240* | *Continue report* |
| *2/17/17* | *240* | *Conclude paper, graphs, etc* |
| *2/18/17* | *45* | *Final review of paper* |
| *2/19/17* |  |  |
| *2/20/17* | *15* | *Submit paper* |
| 2/21/17 |  |  |
| 2/22/17 | 60 | Read about Artifical Neuron Learning |
| 2/23/17 | 120 | Begin reading about Supervised Learning Neural Networks |
| 2/24/17 | 60 | Finish reading about Supervised Learning Neural Networks |
| 2/25/17 | 100 | Read about back propagation learning |
| 2/26/17 | 120 | Read about Unsupervised Learning Neural Networks and Reinforcement Learning |
| 2/27/17 | 100 | Work through code on network trainer |
| 2/28/17 |  |  |
| 3/1/17 | 360 | Network trainer section |
| 3/2/17 | 180 | Sample dataset |
| 3/3/17 |  |  |
| 3/4/17 |  |  |
| 3/5/17 |  |  |
| 3/6/17 |  |  |
| 3/7/17 | 60 | Console app/logs |
| 3/8/17 |  |  |
| 3/9/17 | 180 | Begin paper |
| 3/10/17 | 120 | Compile pictures |
| 3/11/17 | 180 | Continue paper |
| 3/12/17 | 120 | Conclude paper |
| 3/13/17 |  |  |
|  |  |  |
| *Project 1* | *2295* |  |
| *Project 2* | 2400 |  |
| Project 3 | 1760 |  |
| TOTAL | 6455 |  |

Activity Log – Robison

|  |  |  |
| --- | --- | --- |
| Date | Time (mins) | Description |
| *1/9/17* |  |  |
| *1/10/17* |  |  |
| *1/11/17* | *120* | *Reading Chapter 8* |
| *1/12/17* | *120* | *Reading Chapter 9* |
| *1/13/17* | *90* | *Working on Project 0* |
| *1/14/17* |  |  |
| *1/15/17* | *60* | *Working on Project 0* |
| *1/16/17* |  |  |
| *1/17/17* | *120* | *Started Development of GA* |
| *1/18/17* |  |  |
| *1/19/17* |  |  |
| *1/20/17* | *240* | *Continued Development on GA* |
| *1/21/17* | *480* | *Continued Development on GA* |
| *1/22/17* | *120* | *Started Developing Report* |
| *1/23/17* |  |  |
| *1/24/17* |  |  |
| *1/25/17* |  |  |
| *1/26/17* |  |  |
| *1/27/17* | *120* | *Continue Developing Report* |
| *1/28/17* |  |  |
| *1/29/17* | *180* | *Continue Developing Report* |
| *1/30/17* | *90* | *Formatting Report*  *Upload Report* |
| *1/31/17* |  |  |
| *2/1/17* | *120* | *Reading Chapter 20* |
| *2/2/17* | *120* | *Reading Chapter 21* |
| *2/3/17* |  |  |
| *2/4/17* | *180* | *Reading Chapter 22* |
| *2/5/17* | *90* | *Reading Chapter 23* |
| *2/6/17* |  |  |
| *2/7/17* |  |  |
| *2/8/17* | *120* | *Develop class structure for Heating/Cooling app* |
| *2/9/17* | *240* | *Development of MembershipFunction Class* |
| *2/10/17* |  |  |
| *2/11/17* | *240* | *Development of FuzzySet Class* |
| *2/12/17* | *240* | *Testing and Debugging Heating/Cooling app* |
| *2/13/17* | *180* | *Begin generating report* |
| *2/14/17* |  |  |
| *2/15/17* | *240* | *Continue report* |
| *2/16/17* | *180* | *Continue report* |
| *2/17/17* | *120* | *Continue report* |
| *2/18/17* | *120* | *Final review of paper* |
| *2/19/17* |  |  |
| *2/20/17* | *15* | *Submit paper* |
| 2/21/17 | 60 | Read about artifical neuron learning |
| 2/22/17 | 150 | Read about Supervised Learning Neural Networks |
| 2/23/17 | 100 | Read supervised learning neural networks |
| 2/24/17 |  |  |
| 2/25/17 | 120 | Back propagation learning |
| 2/26/17 |  |  |
| 2/27/17 | 100 | Read about unsupervised learning neural networks and reinforcement learning |
| 2/28/17 |  |  |
| 3/1/17 | 360 | Network trainer section |
| 3/2/17 | 200 | Sample dataset |
| 3/3/17 | 120 | Review class structure layout |
| 3/4/17 |  |  |
| 3/5/17 | 180 | Network driver code |
| 3/6/17 |  |  |
| 3/7/17 |  |  |
| 3/8/17 | 30 | Review console app code and logging feature |
| 3/9/17 | 180 | Work on paper |
| 3/10/17 | 120 | Continue paper editing and submissions |
| 3/11/17 | 250 | Continue working on paper |
| 3/12/17 | 300 | Help conclude paper |
| 3/13/17 |  |  |
|  |  |  |
| *Project 1* | *1740* |  |
| *Project 2* | *2625* |  |
| Project 3 | 2270 |  |
| TOTAL | 6635 |  |