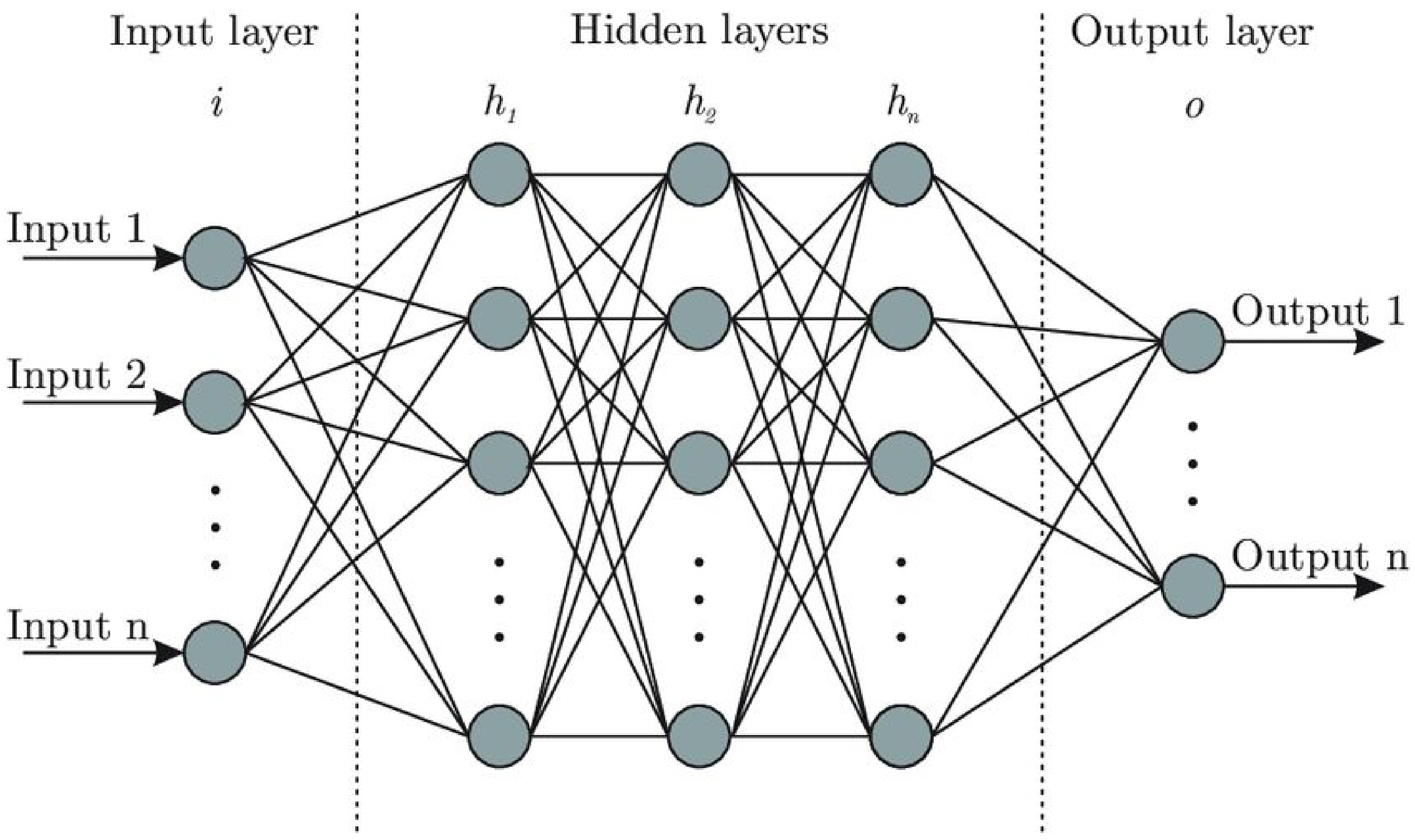
FEED FORWARD NEURAL NETWORK



WITH BACKPROPAGATION

MACHINE LEARNING

INTRODUCTION

A feed forward neural network has been implemented . The weights in the network have been updated using a backpropagation algorithm. The code allows the user to set several configurable parameters regarding the structure of the network as well as the division of data(training and testing). In order to avoid overfitting, weight decay and cross validation strategies have been used. It is possible to run the network on any given data (after preprocessing to correct the format).

THEORY

A feedforward neural network is an artificial neural network where the nodes never form a cycle. This kind of neural network has an input layer, hidden layers, and an output layer.

Feedforward

hx -

1. $Z\_in = b + \Sigma xw$
2. Z = Activation_fun(Z_in)

Backpropagation

1.	Delta for output layer	$\boldsymbol{\delta}$ = (O - Z)*Activation_fun_derivative(Z_in)
2.	Delta for hidden layers	$= (\sum_k w)$ *Activation_fun_derivative(Z_in)
3.	Delta_w	Δw = $\boldsymbol{\alpha\delta}$Z
4.	Delta_bias	Δb = $\boldsymbol{\alpha\delta}$
5.	Weight updation	w = w + Δw
6.	Bias updation	b = b + Δb

Weight Decay When training neural networks, it is common to use **"weight decay,"** where after each update, the **weights** are multiplied by a factor slightly less than 1. This prevents the **weights** from growing too large, and can be seen as gradient descent on a quadratic **regularization** term.

Hence the weight updation now becomes

Weight = Weight * (1 - 2*$\alpha\gamma$) + Δ w

As now error is changed to $E = (1/2)(\Sigma\Sigma(target - output)^2) + gamma * \Sigma W^2$

w_new = w _old * learning_rate * dLoss / dw

d(wd * w^2) / dw = 2 * wd * w (similar to d(x^2)/dx = 2x)

About Dataset

Data Set Characteristics:	Multivariate	**Number of Instances:**	310
Attribute Characteristics:	Real	**Number of Attributes:**	6
Associated Tasks:	Classification	**Missing Values?**	N/A

Data Set Information:

Biomedical data set built by Dr. Henrique da Mota during a medical residency period in the Group of Applied Research in Orthopaedics (GARO). The data have been organized in classification tasks. The task consists in classifying patients as belonging to one out of three categories:

Normal (100 patients)

Disk Hernia (60 patients)

Spondylolisthesis (150 patients).

Attribute Information:

Each patient is represented in the data set by six biomechanical attributes derived from the shape and orientation of the pelvis and lumbar spine (in this order): **pelvic incidence, pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius and grade of spondylolisthesis.** The following convention is used for the class labels: 0 (Disk Hernia), 2(Spondylolisthesis) , 3(Normal).

Variables

Alpha= 0.05 -learning rate

Gamma 0.01 -weight decay

Attributes- number of attributes in oe data instance (here= 6)

classification -number of classes (here=3)

Instances- number of data instances (here=310)

Dataset- dataset read from file

dataset_output - output for data

Validation- validation set

Target - store encoded classification

Val_target -encoded classification of validation set

Layer_info - Number of nodes and activation of each layer

Layers - Layer[i] contains values of nodes in layer i where layer[0] is input

Weight -weight[i] contains 2d matrix for weight values between layer i and i+1 , eg.weight [0] has 2d matrix for weight values between input and first hidden layer

delta_w - change in weight

bias_w - bias weight for each node

best_w - weight when error of validation lowest

best_bias_w - bias weight when error of validation lowest

Best_epoch - epoch number when error of validation lowest

Min_val_error - minimum validation error

User_defined_adjecency- connection of weights when user defines network

train_data_size-number of data instances for training val_data_size-number of data instances for testing

Functions Used

chosen activation functions are :

1.sigmoid and its derivative dSigmoid

double sigmoid(double x);
double dSigmoid(double x);

2.tanh and its derivative tanh

double tanh(double x);
double dtanh(double x);

3.ReLU and its derivative dReLU

double ReLU(double x);
double dReLU(double x);

void print_out(const vector<double> &v)
Prints the output generated by the neural net by applying a max operator on all the values .
Eg [0.2 0.3 0.5] -----> [0 0 1]

void classify(const vector<double>& data_output,vector<vector<double>> &target);
to convert output given in the dataset into vector form.
Eg. if we have 3 classes then output 2 will be converted as ----> [0 1 0]

double convert_to_num(string str);

double fRand(double fMin, double fMax)
This function will randomly generate a number between fMin and fMax.
We have set fMin = -0.05 and fMax = 0.05.

double Activation(int type,double value)
This function has a switch case which calls the activation functions based on the type that we are passing as an argument.

double Activation_Derivative(int type,double value)
This function has a switch case which calls the derivatives of activation functions based on the type that we are passing as an argument.

void file_read(vector<vector<double> >& dataset)
This function will read the data from file and populate the dataset 2d vector whose rows represent the dataset instances while columns represent the attributes values.

void split_data(vector<vector<double>> & dataset,vector<vector<double>> & validation,vector<vector<double>> & target,vector<vector<double>> &validation_target)
This function is used to split the data into testing and validation based on the split ratio that we are taking from the user.

INPUT

1.READ DATA FROM FILE :

The data in the file is given in a comma separated manner with the output class as the last column

Eg.

```
i1    ,i2     ,i3  ,i4        ,i5       ,i6     ,output
39.06 , 10.06 , 25.02 , 29        ,114.41 , 4.56    , 1
56.54 , 14.38 , 44.99 , 42.16  , 101.72 , 25.77  , 3
```

2.Configurable Parameters from user

A. **NUMBER OF HIDDEN LAYERS**

   (Any number of layers can be chosen starting from 0 which is a direct connection of input to output)

B. **NUMBER OF NEURONS FOR EACH HIDDEN LAYER**

   (As no. of neurons for the input layer will be attributed and for the output layer it will be the output class of that data instance).

C. **ACTIVATION FUNCTION FOR EACH LAYER**

   1 -Sigmoid,

2 -for ReLU

3- Tanh

D. FULLY CONNECTED NETWORK OR CUSTOM NETWORK

0-fully connected

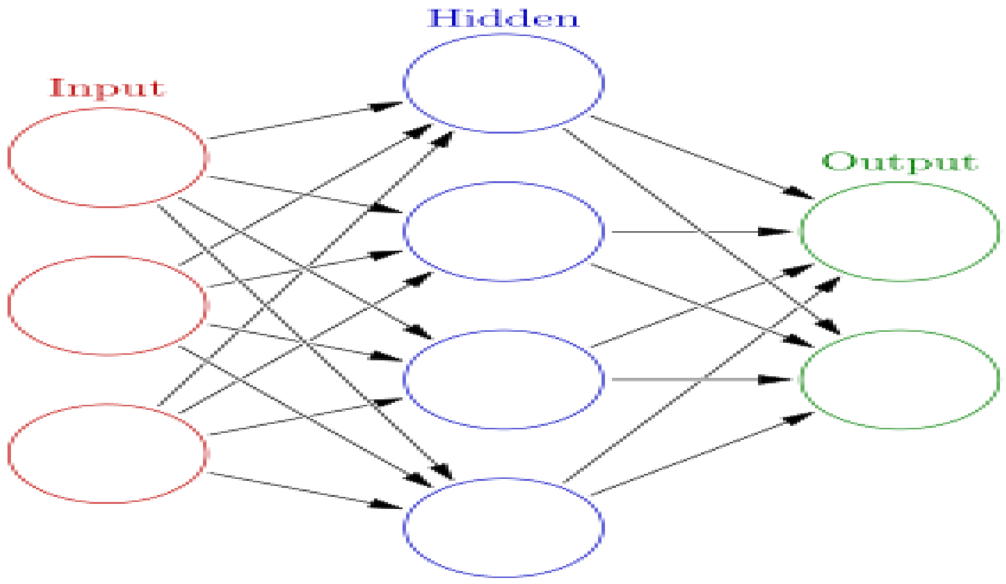
1-custom network

If the user wants to define his own network then an adjacency matrix has to be given for the weights between every 2 layers which is stored in a vector of 2d matrices.

Eg if one hidden layer two adjacency matrix need to be given (here in image)

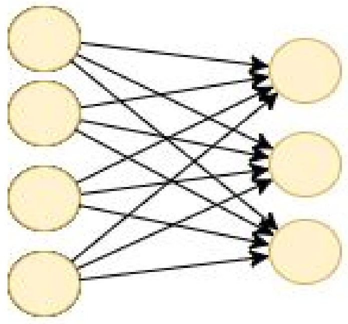
Dimensions of matrix will depend on number of nodes

(number of nodes in prev layer * number of nodes in next layer)



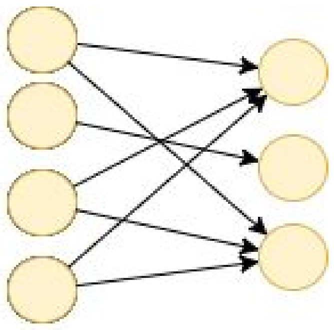
First matrix for W0,1 Second matrix for W1,2

dim(3*4) dim(4*2)



Fully
Connected

1	1	1
1	1	1
1	1	1
1	1	1



Sparse
Connections

1	0	1
0	1	0
1	0	1
1	0	1

E. **THE PERCENTAGE OF DATA (Split Ratio)TO BE KEPT AS TRAINING AND TESTING (default training -80% testing-20%)**

PROCEDURE

2. Data read from files
3. Data rows randomly shuffled
4. Output extracted from last column
5. Output converted into classification vectors eg-class 1 =[1 0 0]
6. The input data and output is split into training and testing sets
7. Configurable parameters are taken from user
    a. Number of hidden layers
    b. Number of neurons in each layer
    c. Activation for each layer
    d. Network connections among neurons
8. Weights and biases are initialised randomly between -0.05 and 0.05.
9. Epoch_error=0
10. For each epoch
    a. For each data instance
        i. Feedforward - neuron values of all the hidden layers and output layer is calculated using hx.
        ii. Backpropagation
            1. Delta for output layer is calculated
            2. Delta for all the hidden layers is calculated and bias is updated.
            3. Delta_w is calculated.
            4. Weight decay is applied
            5. Weight is updated.
            6. Error calculation and added in Epoch_error
    b. Error is calculated for all instances of validation set
    c. When error for validation set is minimum the state of weights is stored(cross validation)
11. Output-Final minimum error for validation and training set.

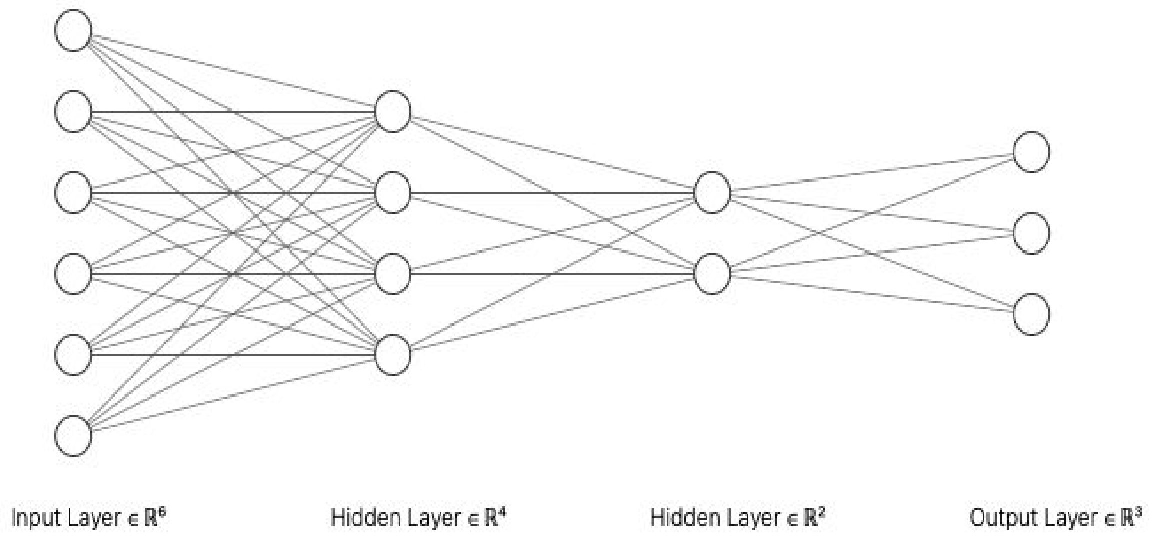
OUTPUT

The neural Network outputs the weights and the error of the state where the error of the validation set is least .

RESULTS

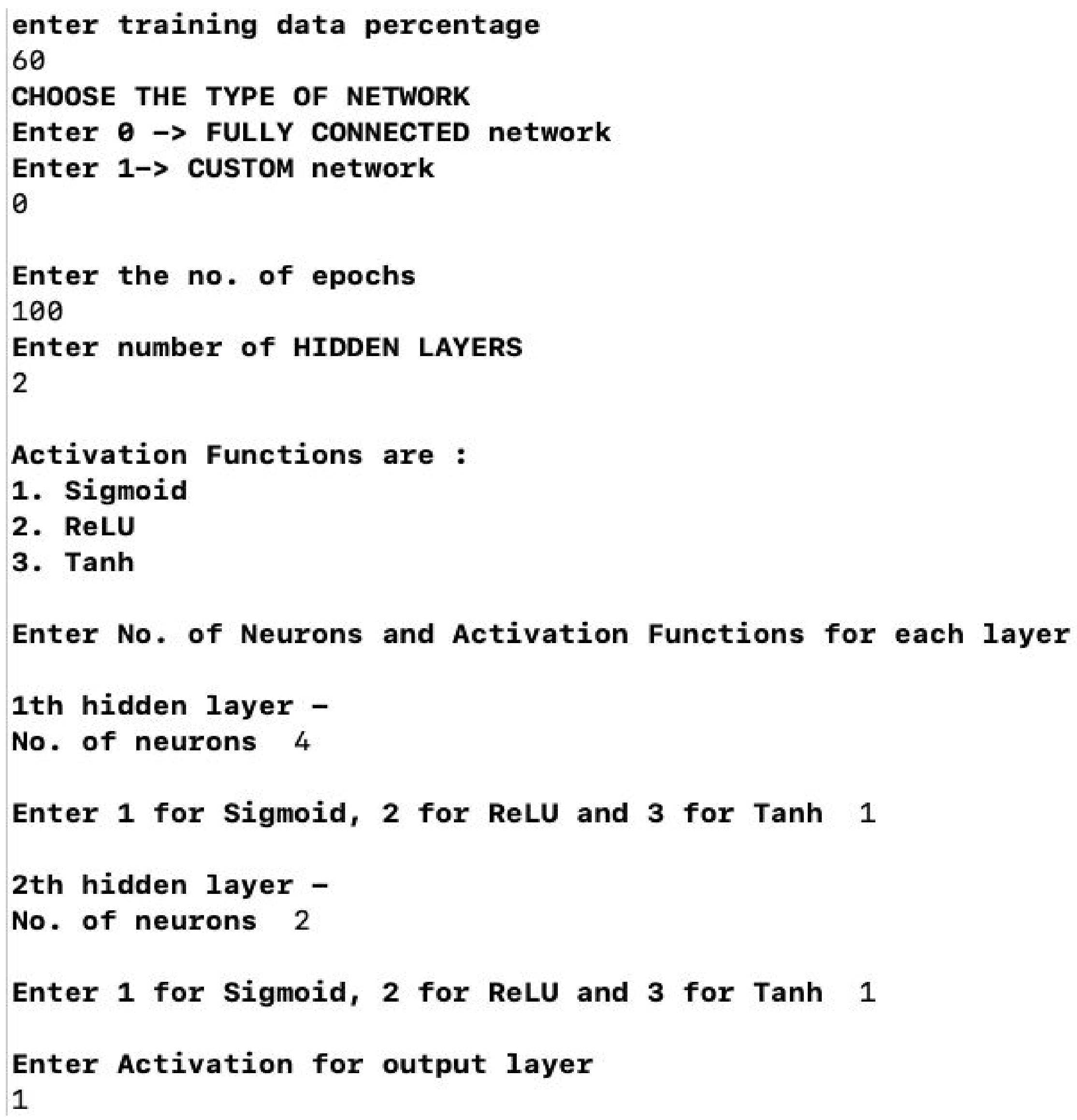
The model and the processing techniques discussed above achieves 70% classification accuracy on the chosen dataset.

One of the Configurable set of parameters for which this is achieved :

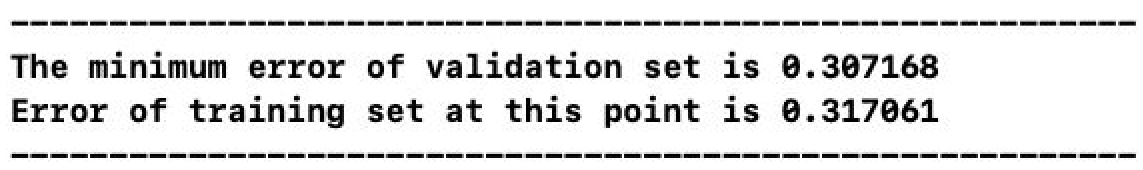


With no. of epochs = 100 and activation sigmoid. Training set percentage=60%

Screenshots of program Execution



```
enter training data percentage
60
CHOOSE THE TYPE OF NETWORK
Enter 0 -> FULLY CONNECTED network
Enter 1-> CUSTOM network
0

Enter the no. of epochs
100
Enter number of HIDDEN LAYERS
2

Activation Functions are :
1. Sigmoid
2. ReLU
3. Tanh

Enter No. of Neurons and Activation Functions for each layer

1th hidden layer -
No. of neurons  4

Enter 1 for Sigmoid, 2 for ReLU and 3 for Tanh  1

2th hidden layer -
No. of neurons  2

Enter 1 for Sigmoid, 2 for ReLU and 3 for Tanh  1

Enter Activation for output layer
1
```



```
--------------------------------------------------------------
The minimum error of validation set is 0.307168
Error of training set at this point is 0.317061
--------------------------------------------------------------
```

CONCLUSION

The accuracy of the network varies with the number of epochs, number of layers ,number of nodes in each hidden layer. and the kind of activation functions used .

It was found that the weights became constant after 100 to 150 epochs . These parameters need to be found out experimentally for maximum accuracy. Higher number of nodes in the hidden layers and more number of hidden layers did not improve accuracy too much . Also a larger dataset may lead to a higher accuracy with more optimizations .

REFERENCES

1.Machine Learning-Tom Mitchell

2. Fundamentals of neural networks- Laurene V. Fausett