**Classification Implementation**

1. Artificial Neural Network

The excel sheet created during feature extraction is given as input to the neural network. Therefore, the input would be a 250\*21 matrix where 250 are the number of samples and 21 are the number of features. Our target would be given as a 250\*4 matrix where the representation will be combination of 0 and 1. For eg:- Pituitary Adenoma will be 1 0 0 0 ; Meningioma will be 0 1 0 0 ; HGG will be 0 0 1 0 ; and GBM will be 0 0 0 1.

The training method used is ‘trainbr’

trainbr is a network training function that updates the weight and bias values according to Levenberg-Marquardt optimization. It minimizes a combination of squared errors and weights, and then determines the correct combination to produce a network that generalizes well. The process is called Bayesian regularization.

net.trainFcn = 'trainbr' sets the network trainFcn property.

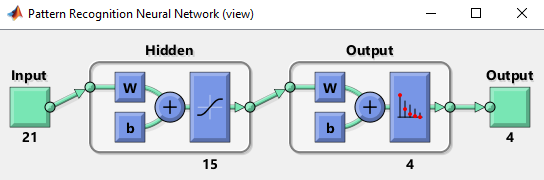
Training occurs according to trainbr training parameters, shown here with their default values:

|  |  |  |
| --- | --- | --- |
| net.trainParam.epochs | 1000 | Maximum number of epochs to train |
| net.trainParam.goal | 0 | Performance goal |
| net.trainParam.mu | 0.005 | Marquardt adjustment parameter |
| net.trainParam.mu\_dec | 0.1 | Decrease factor for mu |
| net.trainParam.mu\_inc | 10 | Increase factor for mu |
| net.trainParam.mu\_max | 1e10 | Maximum value for mu |
| net.trainParam.max\_fail | inf | Maximum validation failures |
| net.trainParam.min\_grad | 1e-7 | Minimum performance gradient |
| net.trainParam.show | 25 | Epochs between displays (NaN for no displays) |
| net.trainParam.showCommandLine | false | Generate command-line output |
| net.trainParam.showWindow | true | Show training GUI |
| net.trainParam.time | inf | Maximum time to train in seconds |
|  |  |  |

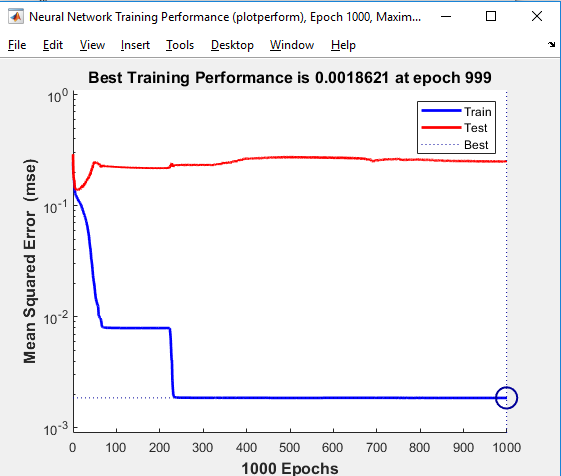
The training and testing data are broken in the ratio 80:20

Validation stops are disabled by default (max\_fail = inf) so that training can continue until an optimal combination of errors and weights is found.

The network architecture is:



The performance plot is:



The confusion matrix is:



1. Convolution Neural Network

The image datastore stores all the images of all the 4 classes. The images in the datastore are resized so that the input layer has the same size for all the images. The image datastore is then split in the ratio 75:25 for training and validation image datastore. The layers used in CNN are given as:

1 '' Image Input 100x100x1 images with 'zerocenter' normalization

2 '' Convolution 8 3x3 convolutions with stride [1 1] and padding [1 1 1 1]

3 '' Batch Normalization Batch normalization

4 '' ReLU ReLU

5 '' Max Pooling 2x2 max pooling with stride [2 2] and padding [0 0 0 0]

6 '' Convolution 16 3x3 convolutions with stride [1 1] and padding [1 1 1 1]

7 '' Batch Normalization Batch normalization

8 '' ReLU ReLU

9 '' Max Pooling 2x2 max pooling with stride [2 2] and padding [0 0 0 0]

10 '' Convolution 32 3x3 convolutions with stride [1 1] and padding [1 1 1 1]

11 '' Batch Normalization Batch normalization

12 '' ReLU ReLU

13 '' Max Pooling 2x2 max pooling with stride [2 2] and padding [0 0 0 0]

14 '' Convolution 32 3x3 convolutions with stride [1 1] and padding [1 1 1 1]

15 '' Batch Normalization Batch normalization

16 '' ReLU ReLU

17 '' Max Pooling 2x2 max pooling with stride [2 2] and padding [0 0 0 0]

18 '' Convolution 64 3x3 convolutions with stride [1 1] and padding [1 1 1 1]

19 '' Batch Normalization Batch normalization

20 '' ReLU ReLU

21 '' Max Pooling 2x2 max pooling with stride [2 2] and padding [0 0 0 0]

22 '' Convolution 128 3x3 convolutions with stride [1 1] and padding [1 1 1 1]

23 '' Batch Normalization Batch normalization

24 '' ReLU ReLU

25 '' Max Pooling 2x2 max pooling with stride [2 2] and padding [0 0 0 0]

26 '' Fully Connected 4 fully connected layer

27 '' Softmax softmax

28 '' Classification Output crossentropyex