Applicable Artificial Intelligence module

course work

Submitted by

Mrityunjay Rajesh Rai
MSc Automation Control and Robotics
Student ID:c0042585

Index

Introduction

The objective of this task is to train a Neural net, when the target and input dataset is provided to the neural net, the neural net learns from that data by adjusting the weights of individual neuron weights to fit the target dataset. This process is repeated until the error is reduced to below a specific range or until the specified epochs is reached. By using this method the neural net learns from the data and classifies the target output. Since the neural network works on the same principle as the neuron in the human brain works, so if it is hard for humans to classify a classification problem, it will be harder for a neural net also to classify the classification problem. The task aims to train a neural net to determine whether the person has no heart disease, mild heart disease or severe heart disease. The dataset that is going to be used in this task is displayed below.

297x13 double

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	63	1	1	145	233	1	2	150	0	2.3000	3	0	6
2	67	1	4	160	286	0	2	108	1	1.5000	2	3	3
3	37	1	3	130	250	0	0	187	0	3.5000	3	0	3
4	41	0	2	130	204	0	2	172	0	1.4000	1	0	3
5	56	1	2	120	236	0	0	178	0	0.8000	1	0	3
6	62	0	4	140	268	0	2	160	0	3.6000	3	2	3
7	57	0	4	120	354	0	0	163	1	0.6000	1	0	3
8	63	1	4	130	254	0	2	147	0	1.4000	2	1	7
9	53	1	4	140	203	1	2	155	1	3.1000	3	0	7
10	57	1	4	140	192	0	0	148	0	0.4000	2	0	6
11	56	0	2	140	294	0	2	153	0	1.3000	2	0	3
12	56	1	3	130	256	1	2	142	1	0.6000	2	1	6
13	44	1	2	120	263	0	0	173	0	0	1	0	7
14	52	1	3	172	199	1	0	162	0	0.5000	1	0	7
15	48	1	2	110	229	0	0	168	0	1	3	0	7
16	54	1	4	140	239	0	0	160	0	1.2000	1	0	3
17	48	0	3	130	275	0	0	139	0	0.2000	1	0	3
18	49	1	2	130	266	0	0	171	0	0.6000	1	0	3

297x1 double

	1
1	0
2	1
3	0
4	0
5	0
6	2
7	0
8	1
9	1
10	0
11	0
12	1
13	0
14	0
15	1
16	0
17	0
18	0

There are 13 columns in the input dataset, each column has a different scale of the data range. The size of this data is 297(rows)*13(columns). The target dataset is displayed below

Here the class 0 represents no heart condition, class 1 represents mild heart condition and class 2 represents severe heart condition. The data size of the target dataset is 297(rows)*1(columns). The first row of the input dataset corresponds to the first row of the target dataset. Based on the row of the input dataset the algorithm will predict the target dataset.

Requirement Analysis

- The input and the target dataset is combined, this enables easy manipulation of the dataset.

```
load('cleveland_heart_disease_dataset_labelled.mat')
xn = normalize(x)
c=[xn t];
```

- Since the input dataset has a different range of dataset in the columns, to deal with this issue, the data is either normalized or standardized. The standardization converts the data range approximately between [-3, 3]. The normalization converts the data in the range [0,1].
- To check the accuracy of the trained neural network on unseen data, the dataset is split into two-part, test data and train data. The Neural net learns from the training data and once the training is done, based on the test data input the neural net classifies the target output. Initially, the data is split according to the classes. Here there are 160 datasets of class 0, 89 datasets of class 1 and 48 datasets of class 2. In the code, 20 samples from each class are used for testing purpose. Here i, j, k contains the number of rows of class 0,1 and 2 respectively.

```
%seperating the data according to class.
for i=1:s
    if c(i,14)==0
        a0=[a0;c(i,1:14)]
    elseif c(i,14)==1
        a1=[a1;c(i,1:14)]
    else c(i,14)==2
        a2=[a2;c(i,1:14)]
    end
end
[i,~]=size(a0)
[j,~]=size(a1)
[k,~]=size(a2)
% Test sample per class
td=20
setdemorandstream(672880951)
% taking td samples from each class for testing
a0_test=a0(i-td+1:i,:);
a1_test=a1(j-td+1:j,:);
a2_test=a2(k-td+1:k,:);
```

- In the training dataset, there are 140 datasets for class 0, 69 datasets of class 1 and 28 data set for class 2, if this data is feed to the neural net for training, the neural net will be biased. The neural net will be predicting more classes 0 in comparison class1 and class 2. For this reason, an additional dataset has been added to other class until all classes have the same number of the dataset. The additional dataset refers to adding the same class dataset repeatedly. Here in the code td is the no of samples per class. To change the train and test samples just change the td, train and test samples will be split accordingly and the data imbalance in the training is done

automatically by the code provided below. For class 2, since there is a small number of samples for training, while loop is used to add additional datasets.

```
%assigning the rest of the samples to training class and dealing with
%imbalanced class
a0_train=a0(1:i-td,:);%i-j
a1_train=[a1(1:j-td,:);a1(1:i-j,:)];

t=1
a3=a2(1:k-td,:)
while t<fix((i-td)/(k-td))
    a3=[a3;a2(1:k-td,:)]
    t=t+1
end
[t,~]=size(a3)
a2_train=[a3;a2(1:i-td-t,:)]
```

- Now there is an equal number of datasets for class 0, class 1 and class 2 in the training dataset, adding the different classes of test and train in a single variable.

```
%merging all train test classes as a sigle
Test=[a0_test;a1_test;a2_test]
Train=[a0_train;a1_train;a2_train]
```

- While adding the dataset the classes are organized sequentially, if this data is feed to the neural net the results will be biased and this could lead to a poor result. The dataset is randomly is distributed such that all classes are mixed randomly. The size of the test sample is 60*14 and the train sample size is 420*14

```
%randomizing the classes of train test samples
test_random=Test(randperm(size(Test,1)),:)
train_random=Train(randperm(size(Train,1)),:)
%random_x = x(randperm(size(x, 1)), :)
```

- Here the target dataset is changed, Now 1 represents no heart condition, 2 represents mild heart condition and 3 represents a severe heart condition.

```
%Neural Net
train_random(:,14)=train_random(:,14)+1
test_random(:,14)=test_random(:,14)+1
```

Design considerations:

For this classification problem, a patternnet neural network has been used in Matlab. The parameter that can be manipulated for the neural net is **activation function, training function** and **performance function**. The role of the transfer function is to update the weight and bias while backpropagation. The role of the activation function is to convert the sum of weighted input to output using a certain function. The role of the performance function is the calculation of error between the actual value and the predicted value, based on that error the weights are updated accordingly.

The predefined activation function available in Matlab is

- `"compet","elliotsig","hardlim","hardlims","logsig","netinv","poslin","purelin","radbas","radbasn","satlin","satlins","softmax","tansig","tribas"`

The predefined training function provide by Matlab are

- `"trainbfg","traincgb","traincgf","traincgp","traingd","traingda","traingdm","traingdx","trainoss","trainrp","trainscg","trainb","trainc","trainr","trains","trainbu","trainru"`

The predefined performance function provided by Matlab are

- `"mae","mse","sae","sse","crossentropy","msesparse"`

For design purpose, there is no fixed rule on how to select the number of neurons in the hidden layer so, for best results, the model is trained for different number of neurons.
For this classification problem, only a single hidden layer is used, using more hidden layers could lead to overfitting of data and this will lead to poor test classification.
To determine which parameter works better in combination with the other parameters, the best way is to create a neural net with all parameter combination and see which parameter gives a good result and then save the model that performs the best on the test dataset. Additional parameters such as learning rate, number of the hidden layer can be added to the program. The more the parameter, the longer time the program takes to find the best neural net parameters. After training the model1 with all these parameters, the best model1 parameter is

au5	"trainbfg"
au6	"poslin"
au7	"mse"

- The best parameters are trainbfg(training function), poslin(activation function), mse(performance function), when these functions are used together the accuracy of the model is 65%

- **trainbfg(BFGS Quasi-Newton Backpropagation):** Newton's method is an alternative to the conjugate gradient methods for fast optimization. The basic step of Newton's method is $\mathbf{x}_{k+1}=\mathbf{x}_k-\mathbf{A}_{-1k}\mathbf{g}_k$ (Matlab, 2021).

- **poslin(activation function):** poslin is a neural transfer function. Transfer functions calculate a layer's output from its net input. The transfer function poslin returns the output n if n is greater than or equal to zero and 0 if n is less than or equal to zero.
  poslin(n) = n, if n >= 0
  = 0, if n <= 0 (Matlab, 2021)
- **mse(performance function):** The mse performance function represents the error between the actual and predicted dataset. Mathematically it is represented by

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

In the code, to make the neural net output more robust instead of training a single model, multiple neural networks will be trained. To build multiple models, just change the number of neurons and save the model with a different variable name. When there are multiple models, the mode option will be used to filter the most common output. This process ensures that the model is more robust and provides greater accuracy. After running the code 5 times with the different number of neurons and saving the model each time, there is 5 trained neural net with the best parameters and mode is used to select most common output.

```
% model1 with 80 hidden layer neurons and best parameters
load gregnet1
y=gregnet1(test_random(:,1:13)')
classes1=vec2ind(y)';


% model2 with 120 hidden layer neurons and best parameters
load gregnet2
y=gregnet2(test_random(:,1:13)')
classes2=vec2ind(y)';


% model3 with 50 hidden layer neurons and best parameters
load gregnet3
y=gregnet3(test_random(:,1:13)')
classes3=vec2ind(y)';

% model4 with 150 hidden layer neurons and best parameters
load gregnet4
y=gregnet4(test_random(:,1:13)')
classes4=vec2ind(y)';

% model5 with 200 hidden layer neurons and best parameters
load gregnet5
y=gregnet5(test_random(:,1:13)')
classes5=vec2ind(y)';

classes=[classes1,classes2,classes3,classes4,classes5]
class=mode(classes')'
confusionchart(test_random(:,14),class);
```

Model 1 is trained with 1 hidden layer and there are 80 neurons.
Model 2 is trained with 1 hidden layer and there are 120 neurons.
Model 3 is trained with 1 hidden layer and there are 50 neurons.
Model 4 is trained with 1 hidden layer and there are 150 neurons.
Model 5 is trained with 1 hidden layer and there are 200 neurons.

Implementation and Testing.

To check all these parameters one by one, a looping system is used. pa1 contains all the training function, pa2 contains all the activation function and pa3 contain the performance functions. , when the loop is active, the first loop(au) ensures changing of the training function, the second loop(au1) changes the activation function and finally the third loop ensures that the performances function changes.
The target variable contains the one-hot encoding of the target dataset. The output of the neural net is in one hot encoding format and is changed into the target dataset back via the **vec2ind** command.

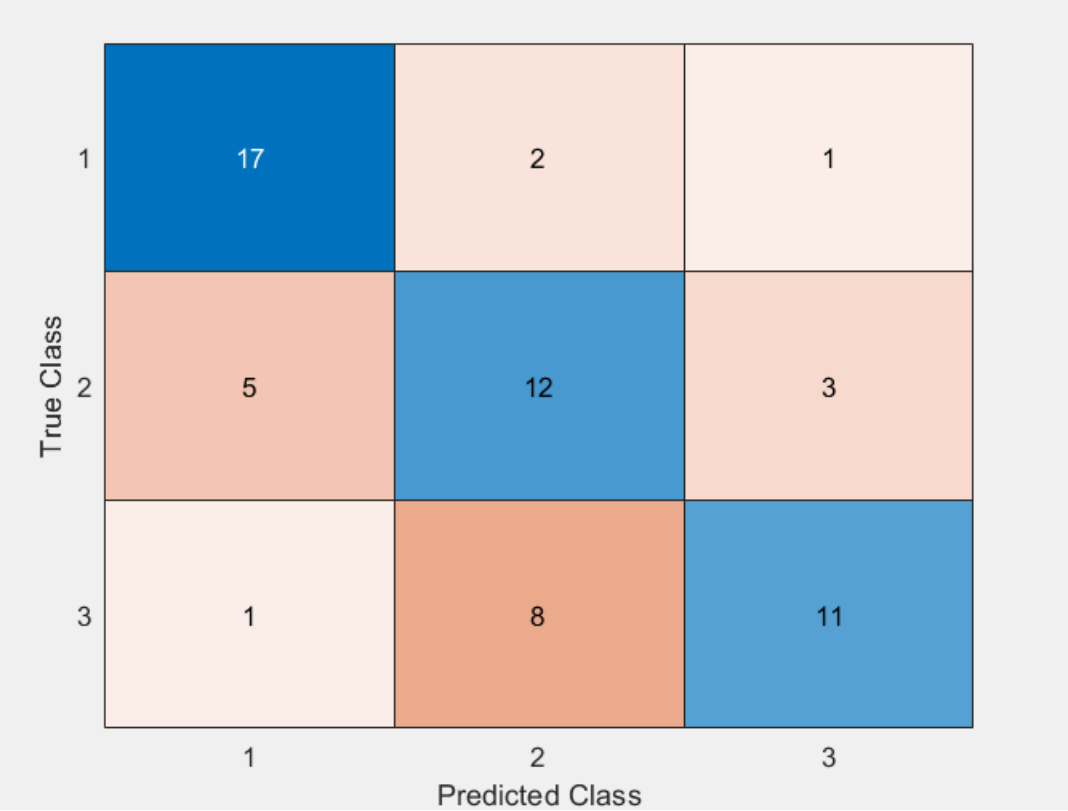
```
pa1=["trainbr","trainlm","trainbfg","traincgb","traincgf","traincgp","traingd","traingda","traingdm","traingdx","trainoss","trainrp","trainscg","trainb","trainc","trainr","trains"
pa2=["elliotsig","compet","hardlim","hardlims","logsig","netinv","poslin","purelin","radbas","radbasn","satlin","satlins","softmax","tansig","tribas"]
pa3=["mse","mae","sae","sse","crossentropy","msesparse"]
[~,pa4]=size(pa1)
[~,pa5]=size(pa2)
[~,pa6]=size(pa3)
au=1
m=0
% finding best parameter for neural net based on how well the model performs on test data
while au<=pa4
    target=ind2vec(train_random(:,14)')
    net=patternnet(26);
    net.trainFcn=pa1(au)
    au1=1
    while au1<=pa5
        net.layers{1}.transferFcn=pa2(au1)
        au2=1
        while au2<=pa6
            try
                net.performFcn=pa3(au2)
                %net.divideFcn='dividetrain';
                %net.performParam.normalization='Standard'
                net.trainParam.epochs=5000;
                net=train(net,train_random(:,1:13)',target)
                y=net(test_random(:,1:13)')
                classes=vec2ind(y)';
```

While using a large number of the parameters, few parameters cannot be used in combination with other parameters, as soon as this happens the Matlab command window shows an error and the program stop. To deal with the issue **try-catch** is used, if there is an error so instead of stopping the whole program, the error is ignored and the program continues with further commands execution.

Testing: The best way to determine the accuracy of the neural net is by determining how well the model does perform on the training dataset. In the code below, the total number of classification that the model got correct is used as a parameter to save the best models and parameters. The While loop compares predicted data to the test data and counts the correct classification. Here l(variable) is the total number of correct classification by the model. For storing the best parameter m(Initial value 0) is used, if the new model has greater l(variable) in comparison to m(variable) then that model is saved and the value of m is modified. This process goes on and when the program is executed completely, the best model and parameters are stored in the variables. The best model is saved in gregnet1 variable and the best parameters are stored in au5, au6 and au7 variable.

```
while j<=i
    if classes(j)==1 & test_random(j,14)==1
        l=l+1
    elseif classes(j)==2 & test_random(j,14)==2
        l=l+1
    elseif classes(j)==3 & test_random(j,14)==3
        l=l+1
    end
    j=j+1
end
if m<l
    m=l
    gregnet1 = net;
    save gregnet1
    % The best parameters are store in the variables below
    au5=pa1(au)
    au6=pa2(au1)
    au7=pa3(au2)
end
```

To finally check the accuracy of the model1, a confusion matrix is used. Here class 1 represents no heart condition, class 2 represents mild heart condition and class 3 represents severe heart condition.



The confusion matrix is plotted between the classes and test data variables.

For class 1, 17 predictions are correct, 2 are misclassified as class2 and 1 is misclassified as class3. For class 2, 12 predictions are correct, 5 are misclassified as class 1 and 3 are misclassified as class3. For class 3, 11 predictions are correct, 8 are misclassified as class 2 and 1 is misclassified as class 1.

Additionally, there will 4 more confusion matrix for 4 different models. The final model will take all the neural nets model into account and the confusion matrix of all the models is displayed below.

Evaluation/conclusion:

The features that make the code better and efficient are

- Using the looping system to find the best parameters for the model, helps to fine-tune the model.
- Instead of using train data accuracy to find the best model, using test data to select a model, ensures that the model that performs better on test data is selected.
- Dealing with class imbalance, adding more data to the training dataset, ensures that the neural net is not biased.
- Automatic train test data split by just specifying the test data per class variable (td). For the current application, td is 20, which means that there are 20 samples of class 0, 20 samples of class 1 and 20 samples of class 2. Overall the test data has 60 samples and the test data is split accordingly.
- The model is made more robust by building multiple models instead of a single model and the results of all the models are taken into account to get the overall result.
- Using only a single hidden layer to avoid overfitting of the dataset.

For finding the best parameters, the computation power required is high and it could take days to just find the best parameter.

By using more parameters such as learning rate, the number of neurons and the number of the hidden layer the accuracy of the model can be increased but this also means more computation power will be required and the time it will take to find the best parameters will increase drastically.

References

Matlab. (2021). Matalab.

Appendix

```
% considering imbalance in the data and solving the
imbalance issue..
load('cleveland_heart_disease_dataset_labelled.mat')
xn = normalize(x)
c=[xn t];
[s,~]=size(c);
a0=[];
a1=[];
a2=[];
%seperating the data according to class.
for i=1:s
    if c(i,14)==0
        a0=[a0;c(i,1:14)]
    elseif c(i,14)==1
        a1=[a1;c(i,1:14)]
    else c(i,14)==2
        a2=[a2;c(i,1:14)]
    end
end
[i,~]=size(a0)
[j,~]=size(a1)
[k,~]=size(a2)
% Test sample per class
td=20
setdemorandstream(672880951)
% taking td samples from each class for testing
a0_test=a0(i-td+1:i,:);
a1_test=a1(j-td+1:j,:);
a2_test=a2(k-td+1:k,:);

%assigning the rest of the samples to training class and
dealing with
%imbalanced class
a0_train=a0(1:i-td,:);%i-j
a1_train=[a1(1:j-td,:);a1(1:i-j,:)];

t=1
a3=a2(1:k-td,:)
while t<fix((i-td)/(k-td))
    a3=[a3;a2(1:k-td,:)]
```

```
    t=t+1
end
[t,~]=size(a3)
a2_train=[a3;a2(1:i-td-t,:)]

%merging all train test classes as a sigle
Test=[a0_test;a1_test;a2_test]
Train=[a0_train;a1_train;a2_train]

%randomizing the classes of train test samples
test_random=Test(randperm(size(Test,1)),:)
train_random=Train(randperm(size(Train,1)),:)
%random_x = x(randperm(size(x, 1)), :)


%Neural Net
train_random(:,14)=train_random(:,14)+1
test_random(:,14)=test_random(:,14)+1


%parameters
%
pa1=["trainbr","trainlm","trainbfg","traincgb","traincgf","
traincgp","traingd","traingda","traingdm","traingdx","train
oss","trainrp","trainscg","trainb","trainr","trains","train
bu","trainru","trainc"]
pa2=["elliotsig","compet","hardlim","hardlims","logsig","ne
tinv","poslin","purelin","radbas","radbasn","satlin","satli
ns","softmax","tansig","tribas"]
pa3=["mse","mae","sae","sse","crossentropy","msesparse"]
[~,pa4]=size(pa1)
[~,pa5]=size(pa2)
[~,pa6]=size(pa3)
au=1
m=0
% finding best parameter for neural net based on how well
the model performs on test data
while au<=pa4
    target=ind2vec(train_random(:,14)')
    net=patternnet(150);
    net.trainFcn=pa1(au)
    au1=1
    while au1<=pa5
        net.layers{1}.transferFcn=pa2(au1)
        au2=1
```

```
        while au2<=pa6
            try
                net.performFcn=pa3(au2)
                %net.divideFcn='dividetrain';
                %net.performParam.normalization='Standard'
                net.trainParam.epochs=20000;
                net=train(net,train_random(:,1:13)',target)
                y=net(test_random(:,1:13)')
                classes=vec2ind(y)';
                [i,~]=size(classes)
                j=1
                l=0
                while j<=i
                    if classes(j)==1 & test_random(j,14)==1
                        l=l+1
                    elseif classes(j)==2 &
test_random(j,14)==2
                        l=l+1
                    elseif classes(j)==3 &
test_random(j,14)==3
                        l=l+1
                    end
                    j=j+1
                end
                if m<l
                    m=l
                    gregnet4 = net;
                    save gregnet4
                    % The best parameters are store in the
variables below
                    au5=pa1(au)
                    au6=pa2(au1)
                    au7=pa3(au2)
                end
            catch
            end
            au2=au2+1
        end
        au1=au1+1
    end
    au=au+1
end
hold off
```

```
% model1 with 80 hidden layer neurons and best parameters
load gregnet1
y=gregnet1(test_random(:,1:13)')
classes1=vec2ind(y)';


% model2 with 120 hidden layer neurons and best parameters
load gregnet2
y=gregnet2(test_random(:,1:13)')
classes2=vec2ind(y)';


% model3 with 50 hidden layer neurons and best parameters
load gregnet3
y=gregnet3(test_random(:,1:13)')
classes3=vec2ind(y)';

% model4 with 150 hidden layer neurons and best parameters
load gregnet4
y=gregnet4(test_random(:,1:13)')
classes4=vec2ind(y)';

% model5 with 300 hidden layer neurons and best parameters
load gregnet5
y=gregnet5(test_random(:,1:13)')
classes5=vec2ind(y)';

classes=[classes1,classes2,classes3,classes4,classes5]
class=mode(classes')'
confusionchart(test_random(:,14),class);
```