Matlab code to read the dataset

1) fid = fopen('20 Percent Training Set.csv','rt');

>> data = textscan(fid,['%d %s %s %s ', repmat('%f ',1,37), '%s'], 'delimiter',',');

>> fclose(fid);

2) to remove all variables

>>clear

>>clc

3) your working file shold be in workspace

4) **NSL - Network Socket Layer**

**5)**To rename all column header

data.Properties.VariableNames([1:42]) = {'Duration’,‘Protocol\_type’,‘Service’,‘Flag’,‘src\_bytes’,‘dst\_bytes’,‘Land’,‘wrong\_fragment’,‘urgent’,‘Hot’,‘num\_failed\_logins’,‘logged\_in’,‘num\_compromised’,‘root\_shell’,‘su\_attempted’,‘num\_root’,‘num\_file\_creations’,‘num\_shells’,‘num\_access\_files’,‘num\_outbound\_cmds’,‘is\_host\_login’,‘is\_guest\_login’,‘count’,‘srv\_count’,‘serror\_rate’,‘srv\_serror\_rate’,‘rerror\_rate’,‘srv\_rerror\_rate’,‘same\_srv\_rate’,‘diff\_srv\_rate’,‘srv\_diff\_host\_rate’,‘dst\_host\_count’,‘dst\_host\_srv\_count’,‘dst\_host\_same\_srv\_rate’,‘dst\_host\_diff\_srv\_rate’,‘dst\_host\_same\_src\_port\_rate’,‘dst\_host\_srv\_diff\_host\_rate’,‘dst\_host\_serror\_rate’,‘dst\_host\_srv\_serror\_rate’,‘dst\_host\_rerror\_rate’,‘dst\_host\_srv\_rerror\_rate’,'label'};

6) code to convert nominal values to binary

for i=1:size(sdata,1)

% trainig

if((sdata{i,42}=='normal'))

sdata.class(i) = '0'; %dataframe.variablename(row)=value%

else

sdata.class(i)= '1';

end

end

7) To extract particular vector in dataset

Vactorname=df(:,[column num]); % src=numeric(:,[2]);

8)To calculate min and max

n=min(src.src\_bytes);

>> [max,maxi]=max(src.src\_bytes); maxi have the index of it.

9) who command is used to display all variables.

10)whos command is used to display all variables values.

11) Categorical carefull with ‘value’;

12)Code to convert nominal to numeric

function sample = preprocess2(sample)

for i=1:size(sample)

if(sample{i,2}=='tcp')

sample{i,2}='2';

elseif(sample{i,2}=='udp')

sample{i,2}='3';

end

if(sample{i,2}=='icmp')

sample{i,2}='4';

end

13) to delete any column

sample(:,'protocol\_type') = [];

14) Be carefull with () and {}

15) nominal to numeric in Flag feature

for i=1:m

if((sample{i,4}=='OTH'))

sample.Flag(i) = '5';

elseif((sample{i,4}=='REJ'))

sample.Flag(i)= '6';

elseif((sample{i,4}=='RSTO'))

sample.Flag(i)= '7';

elseif((sample{i,4}=='RSTOS0'))

sample.Flag(i)= '8';

elseif((sample{i,4}=='RSTR'))

sample.Flag(i)= '9';

elseif((sample{i,4}=='S0'))

sample.Flag(i)= '10';

elseif((sample{i,4}=='S1'))

sample.Flag(i)= '11';

elseif((sample{i,4}=='S2'))

sample.Flag(i)= '12';

elseif((sample{i,4}=='S3'))

sample.Flag(i)= '13';

elseif((sample{i,4}=='SF'))

sample.Flag(i)= '14';

elseif((sample{i,4}=='SH'))

sample.Flag(i)= '15';

end

end

16)

Normalization

s=size(sdata,1);

Max\_range = 1;

Min\_range = 0;

src\_min=min(sdata.src\_bytes);

src\_max=max(sdata.src\_bytes);

for i=1:s

p=sdata.src\_bytes(i);

p=(src\_max\*((p - src\_min)\*(Max\_range-Min\_range))/((src\_max-src\_min)))+Min\_range;

end

17) Coefficint matrix of src\_bytes,dst\_bytes;

coeff = pca(ingredients)

coeff =

1.0000 -0.0001

0.0001 1.0000

18)column of sdata 25,26

coeff =

0.7069 0.7073

0.7073 -0.7069

19)PCA plot of column 26 and 27

biplot(coeff(:,1:2),'scores',heat(:,1:2),'varlabels',{'v\_1','v\_2',});

20)column 40 and 41

coeff =

0.6929 0.7210

0.7210 -0.6929

21) column 29 and 30

coeff =

0.9849 0.1731

-0.1731 0.984

22) Service field nominal to numeric

for i=1:m

if((service{i,1}=='IRC'))

service.Service(i) = '16';

elseif((service{i,1}=='X11'))

service.Service(i)= '17';

elseif((service{i,1}=='Z39\_50'))

service.Service(i)= '18';

elseif((service{i,1}=='auth'))

service.Service(i)= '19';

elseif((service{i,1}=='bgp'))

service.Service(i)= '20';

elseif((service{i,1}=='courier'))

service.Service(i)= '21';

elseif((service{i,1}=='csnet\_ns'))

service.Service(i)= '22';

elseif((service{i,1}=='ctf'))

service.Service(i)= '23';

elseif((service{i,1}=='daytime'))

service.Service(i)= '24';

elseif((service{i,1}=='discard'))

service.Service(i)= '25';

elseif((service{i,1}=='domain'))

service.Service(i)= '26';

elseif((service{i,1}=='domain\_u'))

service.Service(i)= '27';

elseif((service{i,1}=='echo'))

service.Service(i)= '28';

elseif((service{i,1}=='eco\_i'))

service.Service(i)= '29';

elseif((service{i,1}=='ecr\_i'))

service.Service(i)= '30';

elseif((service{i,1}=='efs'))

service.Service(i)= '31';

elseif((service{i,1}=='exec'))

service.Service(i)= '32';

elseif((service{i,1}=='finger'))

service.Service(i)= '33';

elseif((service{i,1}=='ftp'))

service.Service(i)= '34';

elseif((service{i,1}=='ftp\_data'))

service.Service(i)= '35';

elseif((service{i,1}=='gopher'))

service.Service(i)= '36';

elseif((service{i,1}=='hostnames'))

service.Service(i)= '37';

elseif((service{i,1}=='http'))

service.Service(i)= '38';

elseif((service{i,1}=='http\_443'))

service.Service(i)= '39';

elseif((service{i,1}=='imap4'))

service.Service(i)= '40';

elseif((service{i,1}=='iso\_tsap'))

service.Service(i)= '41';

elseif((service{i,1}=='klogin'))

service.Service(i)= '42';

elseif((service{i,1}=='kshell'))

service.Service(i)= '43';

elseif((service{i,1}=='ldap'))

service.Service(i)= '44';

elseif((service{i,1}=='link'))

service.Service(i)= '45';

elseif((service{i,1}=='login'))

service.Service(i)= '46';

elseif((service{i,1}=='mtp'))

service.Service(i)= '47';

elseif((service{i,1}=='name'))

service.Service(i)= '48';

elseif((service{i,1}=='netbios\_dgm'))

service.Service(i)= '49';

elseif((service{i,1}=='netbios\_ns'))

service.Service(i)= '50';

elseif((service{i,1}=='netbios\_ssn'))

service.Service(i)= '51';

elseif((service{i,1}=='netstat'))

service.Service(i)= '52';

elseif((service{i,1}=='nnsp'))

service.Service(i)= '53';

elseif((service{i,1}=='nntp'))

service.Service(i)= '54';

elseif((service{i,1}=='ntp\_u'))

service.Service(i)= '55';

elseif((service{i,1}=='other'))

service.Service(i)= '56';

elseif((service{i,1}=='pm\_dump'))

service.Service(i)= '57';

elseif((service{i,1}=='pop\_2'))

service.Service(i)= '58';

elseif((service{i,1}=='pop\_3'))

service.Service(i)= '59';

elseif((service{i,1}=='printer'))

service.Service(i)= '60';

elseif((service{i,1}=='private'))

service.Service(i)= '61';

elseif((service{i,1}=='red\_i'))

service.Service(i)= '62';

elseif((service{i,1}=='remote\_job'))

service.Service(i)= '63';

elseif((service{i,1}=='rje'))

service.Service(i)= '64';

elseif((service{i,1}=='shell'))

service.Service(i)= '65';

elseif((service{i,1}=='smtp'))

service.Service(i)= '66';

elseif((service{i,1}=='sql\_net'))

service.Service(i)= '67';

elseif((service{i,1}=='ssh'))

service.Service(i)= '68';

elseif((service{i,1}=='sunrpc'))

service.Service(i)= '69';

elseif((service{i,1}=='supdup'))

service.Service(i)= '70';

elseif((service{i,1}=='systat'))

service.Service(i)= '71';

elseif((service{i,1}=='telnet'))

service.Service(i)= '72';

elseif((service{i,1}=='tim\_i'))

service.Service(i)= '73';

elseif((service{i,1}=='time'))

service.Service(i)= '74';

elseif((service{i,1}=='urp\_i'))

service.Service(i)= '75';

elseif((service{i,1}=='uucp'))

service.Service(i)= '76';

elseif((service{i,1}=='uucp\_path'))

service.Service(i)= '77';

elseif((service{i,1}=='vmnet'))

service.Service(i)= '78';

elseif((service{i,1}=='whois'))

service.Service(i)= '79';

elseif((service{i,1}=='http\_8001'))

service.Service(i)= '80';

elseif((service{i,1}=='urh\_i'))

service.Service(i)= '81';

end

end

25) To start neural network simulator in matlab

Nnstart;

26) To count number of ones

y=sum(class.class=='1')

27)Algorithm

Scaled Conjugate Gradient Back propagation Learning Algorithm

step 1: Normalize the inputs and outputs with respect to their maximum values. Neural networks work better if input and outputs lies between 0 to 1 for each training pair is in a normalized form.

step 2: User can specify the number of neurons at the hidden layer based on features and dataset you can increase the number of neurons.

step 3: weights of synapses connecting between input neurons to hidden neurons represents as Vi and weights of synapses connecting hidden neurons to output neurons represents as Wi.

step 4: Compute the inputs to the hidden layer by multiplying corresponding weights of synapses.

step 5: Let the hidden layer units evaluate the output using the sigmoid function as{O}H = {- - -1/(1+e(-IHi))- - -}.

step 6: Let the output layer units evaluate the output using softmax function as {O}O = {- - - ((e(i)/ Σk=1k ei k---}this is the network output.

step 7: output error is back propagating by updating the weights and threshold values.

28) Implementation of Matlab code.

|  |
| --- |
| function [Y,Xf,Af] = NeuralNetworkFunction(X,~,~) |
|  |
| % ===== NEURAL NETWORK CONSTANTS ===== |
|  |
| % Input 1 |
| x1\_step1.xoffset = 0; |
| x1\_step1.gain = 2; |
| x1\_step1.ymin = -1; |
|  |
| % Layer 1 |
| b1 = [-13.324760466907521916;-10.900504512945476066;7.7777516391799013107;4.6666666664439002687;1.5555555556795430672;1.5555555556026188224;4.6666666539768257849;7.7777984863246585334;-10.898343380655127532;15.165303780350770424]; |
| IW1\_1 = [14.675239533092478084;13.988384375943413218;-14.000026138597876368;-14.000000000222767582;-13.999999999875473833;13.999999999952766672;14.00000001268984029;13.999979291453120922;-13.990545508233759975;12.834696219649229576]; |
|  |
| % Layer 2 |
| b2 = -0.12315652314979817772; |
| LW2\_1 = [1.9579826377084112998 1.6603031876655682808 -4.0253905328637697991 -2.7792172905021814699 -1.2636267177841775666 2.2495251525014055005 3.0539400757714054713 1.8665795616648681587 -1.9427155124482371473 0.52085338496705602118]; |
|  |
| % ===== SIMULATION ======== |
|  |
| % Format Input Arguments |
| isCellX = iscell(X); |
| if ~isCellX |
| X = {X}; |
| end |
|  |
| % Dimensions |
| TS = size(X,2); % timesteps |
| if ~isempty(X) |
| Q = size(X{1},1); % samples/series |
| else |
| Q = 0; |
| end |
|  |
| % Allocate Outputs |
| Y = cell(1,TS); |
|  |
| % Time loop |
| for ts=1:TS |
|  |
| % Input 1 |
| X{1,ts} = X{1,ts}'; |
| Xp1 = mapminmax\_apply(X{1,ts},x1\_step1); |
|  |
| % Layer 1 |
| a1 = tansig\_apply(repmat(b1,1,Q) + IW1\_1\*Xp1); |
|  |
| % Layer 2 |
| a2 = logsig\_apply(repmat(b2,1,Q) + LW2\_1\*a1); |
|  |
| % Output 1 |
| Y{1,ts} = a2; |
| Y{1,ts} = Y{1,ts}'; |
| end |
|  |
| % Final Delay States |
| Xf = cell(1,0); |
| Af = cell(2,0); |
|  |
| % Format Output Arguments |
| if ~isCellX |
| Y = cell2mat(Y); |
| end |
| end |
|  |
| % ===== MODULE FUNCTIONS ======== |
|  |
| % Map Minimum and Maximum Input Processing Function |
| function y = mapminmax\_apply(x,settings) |
| y = bsxfun(@minus,x,settings.xoffset); |
| y = bsxfun(@times,y,settings.gain); |
| y = bsxfun(@plus,y,settings.ymin); |
| end |
|  |
| % Sigmoid Positive Transfer Function |
| function a = logsig\_apply(n,~) |
| a = 1 ./ (1 + exp(-n)); |
| end |
|  |
| % Sigmoid Symmetric Transfer Function |
| function a = tansig\_apply(n,~) |
| a = 2 ./ (1 + exp(-2\*n)) - 1; |
| end |