**Training algorithm**

% BACKPROPAGATION ALGORITHM TRAINING: ONLY FOR SINGLE HIDDEN LAYER

% Data Set

pattern=[0.992272174 0.993241816 0.986543631 0.988741475

0.992386796 0.994537051 0.990059282 0.991602228

0.998833263 0.999673724 0.991979024 0.992297678

0.997790137 0.99963024 0.997607016 0.997869092

0.996046481 0.997324322 0.995568744 0.997307319

0.996750432 0.997308296 0.994724667 0.996560373

0.999066415 0.999946451 0.997715091 0.997715091

0.996645875 1 0.996637373 0.999349011

0.992446794 0.996505261 0.992261132 0.994768249

0.996066513 0.996774958 0.991663301 0.992784503

0.991188007 0.998181786 0.991188007 0.993226279

0.99442155 0.998939285 0.988044559 0.990238885

0.993224814 0.998304909 0.99293655 0.996176639

0.98912169 0.991842904 0.987998046 0.990434025

0.982174913 0.988498355 0.981766653 0.988453797

0.978602196 0.983857008 0.976995734 0.981347449

0.962694897 0.978526172 0.962086513 0.978085861

0.96159021 0.963390347 0.959830625 0.963184653

0.962812939 0.964338883 0.959930199 0.9614066

0.958655387 0.960400998 0.957651251 0.959114167

0.958052475 0.961309568 0.957531645 0.960190321

0.957374126 0.961859322 0.956760758 0.956770726

0.953793396 0.957812385 0.953700858 0.956671641

0.956698709 0.957528225 0.954202633 0.955485946

0.955276833 0.958199637 0.955276833 0.958199637

0.951364354 0.954778478 0.950606953 0.954749456

0.949106513 0.952784276 0.948727763 0.951782093

0.945518747 0.948624672 0.944739261 0.94842103

0.945851472 0.946372302 0.943274879 0.943980296

0.943894793 0.945786393 0.942846587 0.945775937

0.946077589 0.946077589 0.941877042 0.943990849

0.943494058 0.946848574 0.940884924 0.946738545

0.944422561 0.947288396 0.941369697 0.944018307

0.939663661 0.9433724 0.937427806 0.942966192

0.93073265 0.944266997 0.93073265 0.940975996

0.916429761 0.933080097 0.913192309 0.930059674

0.913133777 0.920594348 0.910600667 0.917204263

0.900288821 0.913730142 0.900288821 0.913544969

0.896972219 0.899894535 0.89473138 0.894967464

0.899493798 0.900905218 0.895757503 0.897088307

0.901443539 0.902765841 0.89714938 0.898537738

0.908473864 0.909412921 0.897597117 0.902413084

0.909392401 0.910246933 0.90709 0.907682359

0.910252405 0.913417937 0.905308429 0.908621416

0.912304749 0.913224751 0.908021533 0.90904619

0.905750207 0.912366212 0.903673922 0.910529627

0.910889322 0.912648907 0.908146611 0.90902567

0.910420087 0.914317419 0.910121856 0.911715322

0.908193221 0.909000166 0.903046972 0.907847011

0.908655421 0.913017201 0.907014465 0.90867946

0.909473017 0.91317472 0.909024693 0.910694182

0.907814472 0.911853981 0.907033422 0.908658939

0.90735364 0.908675942 0.903709882 0.904879649

0.909429924 0.913620504 0.907480281 0.907480281

0.904975704 0.907446177 0.898553178 0.90550699

0.907192603 0.910261004 0.904663988 0.907771379

0.915987984 0.916183124 0.90365692 0.906993945

0.914713172 0.920567378 0.914713172 0.91701909

0.915333575 0.916553372 0.908029057 0.912588908

0.914586629 0.914971828 0.908710533 0.913990264

0.910838314 0.916357743 0.910838314 0.914617116

0.913949223 0.916255141 0.906373542 0.908984629

0.914545588 0.914555652 0.91075076 0.913257291

0.90965262 0.91904935 0.90965262 0.91597401

0.916710989 0.918879811 0.905141823 0.907733855

0.912575424 0.91806075 0.909529497 0.917529464

0.905520573 0.912469304 0.903166578 0.911979059

0.911451682 0.911451682 0.904725549 0.905301394

0.919436601 0.919756819 0.913306833 0.913637507

0.917755581 0.923064919 0.917755581 0.920192146

0.908810107 0.915937953 0.906639233 0.915251004

0.909330351 0.911924533 0.90700098 0.907059024

0.908302762 0.913308885 0.905212374 0.90656917

0.911424224 0.911424224 0.904069675 0.906750825

0.901803235 0.913070164 0.901331947 0.911187554

0.904228758 0.908736038 0.900167262 0.902296508

0.913714605 0.913714605 0.901960852 0.903896521

0.902005312 0.918502527 0.900298788 0.91682053

0.920777958 0.920777958 0.904832061 0.904832061

0.924906487 0.925884631 0.92290525 0.924567313

0.926958831 0.9274181 0.924310123 0.926880267

0.925242242 0.928150974 0.923086905 0.927479563

0.923871864 0.9278118 0.922525523 0.925170225

0.920397742 0.922072801 0.915340122 0.921534969

0.922405429 0.922554545 0.917245304 0.920613403

0.925136122 0.926287322 0.920995573 0.923286052

0.921634444 0.926727633 0.921565945 0.925722031

0.92396743 0.929179149 0.917334324 0.92033921

0.924132083 0.925456339 0.922143353 0.922991338

0.927448587 0.927531158 0.922984303 0.924645877

0.929021532 0.932156479 0.927849812 0.927938929

0.927366505 0.929120617 0.92391437 0.927044333

0.929833558 0.929833558 0.925134167 0.928202569

0.928901439 0.930616562 0.92723791 0.930460411

0.927428066 0.929692455 0.924005442 0.929271687

0.931299992 0.93131895 0.928106513 0.928175013

0.93000466 0.936709784 0.928570766 0.932379175

0.930357906 0.930878736 0.927366505 0.929398328

0.926527021 0.932493699 0.926527021 0.931251916];

fid = fopen('wih.dat','w'); % Weights stored of input-hidden layer

fid1 = fopen('who.dat','w'); % Output stored of hidden-output layer

alpha =0.9; % Momentum

%Convergence is made faster if a momentum factor is added to the weight updation process.

eta = 0.8; % Learning rate

tol = 0.001; % Error tolerance

Q = 99; % Total no. of the patterns to be input

n = 3; q = 2; p = 1; % Architecture

% Initializing the values and weights

Wih = 2 \* rand(n,q) - 1; % Input-hidden random weight matrix

Whj = 2 \* rand(q,p) - 1; % Hidden-output random weight matrix

DeltaWih = zeros(n,q); % Weight change matrices

DeltaWhj = zeros(q,p); % matrix of qxp of zeros

DeltaWihOld = zeros(n,q);

DeltaWhjOld = zeros(q,p);

Si = [pattern(:,1:3)]; % Input signals

D = pattern(:,4); % Desired values

Sh = [zeros(1,q)]; % Hidden neuron signals

Sy = zeros(1,p); % Output neuron signals

deltaO = zeros(1,p); % Error-slope product at output

deltaH = zeros(1,q); % Error-slope product at hidden

sumerror = 2\*tol; % To get in to the loop

i=0;

itt=0;

% Training BPA network

while sumerror>tol && itt<20000 % Iterate(Stops when error tolerance = 0.001 or when iteration reaches 20,000

sumerror = 0;

for k = 1:Q % for loop of input data (Q=99 times)

Zh = Si(k,:) \* Wih; % Hidden activations

Sh = [1./(1 + exp(-Zh))]; % Binary sigmoid function Hidden signals

Yj = Sh \* Whj; % Output activations

Sy = 1./(1 + exp(-Yj)); % Binary sigmoid function Output signals

Ek = D(k) - Sy; % Error vector

deltaO = Ek .\* Sy .\* (1 - Sy);% Delta output

for h = 1:q % Delta W: hidden-output

DeltaWhj(h,:) = deltaO \* Sh(h);

end

for h = 2:q % Delta hidden

deltaH(h) = (deltaO \* Whj(h,:)') \* Sh(h) \* (1 - Sh(h));

end

for i = 1:n % Delta W: input-hidden

DeltaWih(i,:) = deltaH(q:q) \* Si(k,i);

end

Wih = Wih + eta \* DeltaWih + alpha \* DeltaWihOld;

Whj = Whj + eta \* DeltaWhj + alpha \* DeltaWhjOld;

DeltaWihOld = DeltaWih; % Update weights(or)Store changes

DeltaWhjOld = DeltaWhj;

sumerror = sumerror + sum(Ek.^2); % Compute error

end

Iteration = itt

sumerror % Print epoch error

itt=itt+1;

end

Wih

Whj

fprintf(fid,'%12.8f\n',Wih);

fprintf(fid1,'%12.8f\n',Whj);

status = fclose(fid);

status = fclose(fid1);

**Test algorithm**

%(2) BACKPROPAGATION ALGORITHM TESTING: ONLY FOR SINGLE HIDDEN LAYER

pattern=[0.992272174 0.993241816 0.986543631

0.992386796 0.994537051 0.990059282

0.998833263 0.999673724 0.991979024

0.997790137 0.99963024 0.997607016

0.996046481 0.997324322 0.995568744

0.996750432 0.997308296 0.994724667

0.999066415 0.999946451 0.997715091

0.996645875 1 0.996637373

0.992446794 0.996505261 0.992261132];

Q=9;

fid1 = fopen('bpnout.dat','w'); % Store the output of test data

fid = fopen('wih.dat'); % input-hidden values

a = fscanf(fid,'%g %g %g',[3 inf]); % It has three rows now.

Wih = a;

fclose(fid);

fid = fopen('who.dat'); % hidden-output values

a = fscanf(fid,'%g %g',[2 inf]); % It has two rows now.

Whj = a;

fclose(fid);

for k = 1:Q % for loop for input test pattern

Si = [ pattern(:,1:3)]; % Input signals

Zh = Si(k,:) \* Wih; % Hidden activations

Sh = [1./(1 + exp(-Zh))]; % Binary activation function Hidden signals

Yj = Sh \* Whj; % Output activations

Sy(k) = 1./(1 + exp(-Yj)); % Output signals

fprintf(fid1,'%12.8f\n',Sy(k));

end

% status = fclose(fid);

Sy

fclose(fid1);