

۱۳۹۲/۱/۱۴



تمرین سری ۲

شبکه های عصبی ۱

استاد درس:

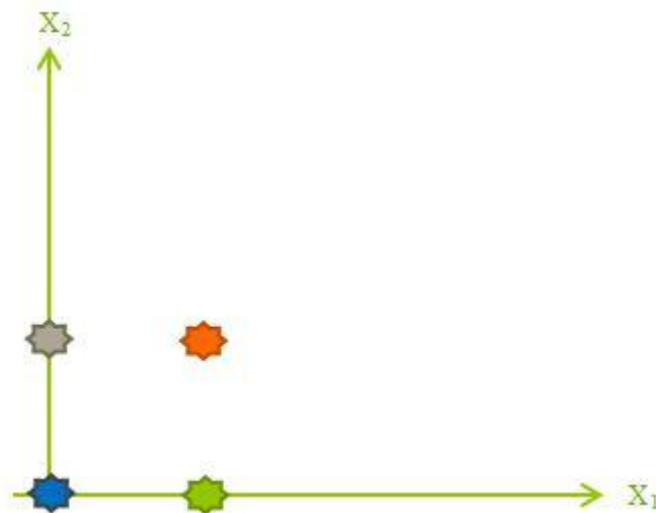
جناب آقای دکتر سید صاکی

دانشجو: بهنام عادلی | ۹۰۱۳۳۰۲۲

پاسخ پرسش ۱:

الف) طراحی دستی

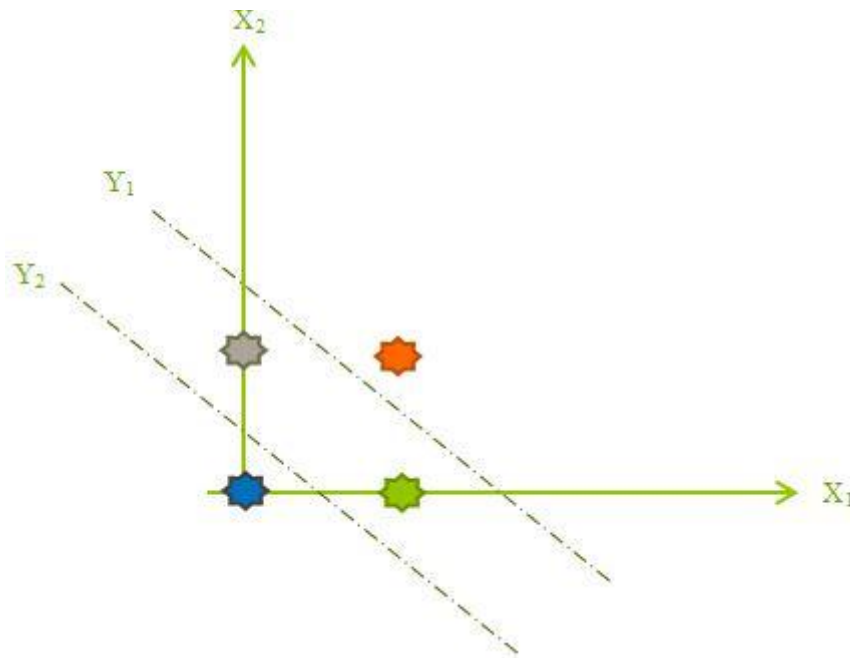
ابتدا باید مساله را به صورت گرافیکی رسم نماییم تا بتوان توصلر بهتری از مساله داشت و همچنین بتوان تعداد نرون‌های لایه پنهان را بدست آورد. ورودی‌های مساله به صورت زیر هستند:



شکل ۱-۳ الف: حالات متفاوت ورودی که با رنگ‌های متفاوت نشان داده شده‌اند.

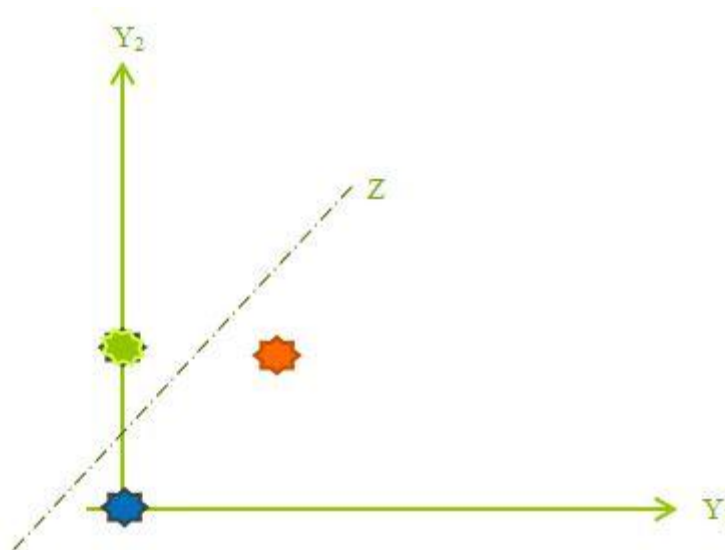
ورودی چهار حالت متفاوت دارد که با چهار ستاره به رنگ‌های متفاوت نشان داده شده‌اند. پس می‌توانیم با استفاده از دو مرز به شکل زیر ابتدا فضا را از چهار به سه کاهش دهیم و سپس با استفاده از یک مرز دیگر این سه حالت را به دو حالت که جواب نهایی مساله است تقلیل دهیم.

با توجه به شکل ۳.۲ برای داشتن دو مرز نیاز به دو نرون خواهیم داشت. جواب‌های این لایه در شکل ۳.۳ نمایش داده شده است.



شکل ۳-۲ دو مرز برای اینکه دو حالت (۰۱) و (۱۰) را با هم متحد نماییم.

حال با استفاده از یک نرون همان‌طور که در بالا توضیح داده شد و در شکل ۳.۳ دیده می‌شود می‌توان فضا را به دو بخش یا دو حالت تقسیم نمود که همان جواب نهایی مساله است. برای این مساله بالای خط چین ابر صفحه Z که در اینجا یک خط است صفر و زیر آن یک در نظر گرفته می‌شود.

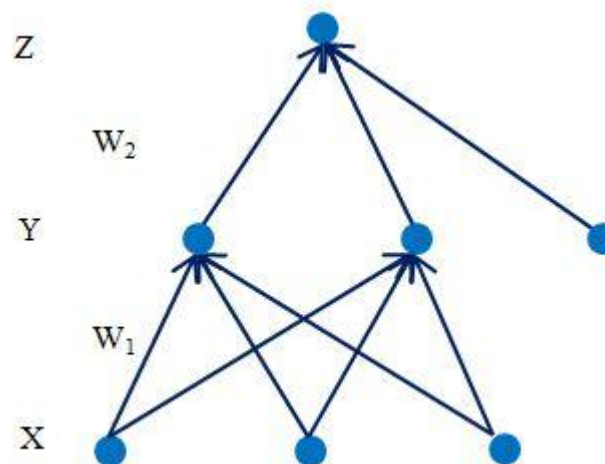


شکل ۳-۳ نمودار خروجی لایه پنهان Y و مرز تصمیم لایه Z در آن

در نهایت جواب به شکل ۴-۳ در خواهد آمد. با توجه به شکل های ۱-۳ تا ۴-۳ نمودار گرافیکی شبکه عصبی طراحی شده در شکل ۵-۳ نشان داده شده است.



شکل ۴-۳ خروجی نهایی.



شکل ۵-۳ نمودار گرافیکی شبکه عصبی یک لایه پنهان برای این مساله.

وزن های بین لایه ها را می توان با استفاده از تئوری بازی ها^۱ و یا با آزمون و خطا بدست آورد. در این تمرین ما با استفاده از تئوری بازی ها تمایز بین داده های ورودی و خروجی ایجاد کردیم.

جدول ۱-۳ مقادیر وزن ها برای لایه اول و دوم

W_1	X_1	X_2	$\theta_1 = -1$
Y_1	$1/2$	$1/2$	$3/4$
Y_2	1	1	$3/4$

جدول ۲-۳ وزن ها را برای ارتباط لایه دوم و لایه خروجی نشان می دهد

¹ Game Theory

W_2	Y_1	Y_2	$\theta_2 = -1$
Z	$1/2$	$-3/4$	$-1/2$

(ب)

کدهای زیر را در MATLAB® برنامه نویسی کرده و تابع نهایی را ذخیره می‌کنیم:

```
%
% Neural Network I Course, Prof. Seyed-Salehi
% Biomedical Engineering Department, Amir-Kabir University of Technology
% Homework 3, Problem 1, Part b
% Written by: ||Behtom Adeli|| MSc Student of Bioelectrical Eng.
% R.A. @ Biomedical Singla Prossecing Lab.
%
%
%
%
% 2-2-1 Back Propagation Neural Network
%
% Help:
% m: Input Layer Neuran Nubmer
% n: Hidden Layer Neuran Nubmer
% l: Output Layer Neuran Nubmer
% x: Inputs Matrix
% d: Desired Output Matrix
% Etta: Learning Coefficient be delkhah
% Emax: Max of Final Acceptable Error
% Alfa: Acceleration Coefficient for Error Correction
%
function [k,E,V,W]=BPNN(m,n,l,x,d,Etta,Alfa,Emax,Kmax)

s=size(x);
P=s(1); % Number of Inputs
f=inline('1./(1+exp(-y))');
V=(2*randn(m+1,n)-1); % Weights Between Input and Hidden Layer
W=(2*randn(n+1,l)-1); % ,, ,, Hidden and Output Layer
k=1;
E=1;
DW=zeros(n+1,1);
DV=zeros(m+1,n);
```

```

while (sqrt(E/P/1)>=Emax && k<Kmax)
    E=0;
    for p=1:P
        y=f(x(p,:)*V);
        y(n+1)=1;
        z=f(y*W);
        E=sum((d(p,:)-z)^2)+E;
        Dz=(d(p,:)-z)*z*(1-z);
        Dy=y.*(1-y).*(Dz*W');
        DW=Etta*y'*Dz+Alfa*DW;
        W=W+DW;
        DV=Etta*x(p,:)'*Dy(1:2)+Alfa*DV;
        V=V+DV;
    end
    T=sqrt(E/P/1);
    plot(k,T)
    grid on
    hold on
    xlabel('Iteration')
    ylabel('sqrt(E/(PL))')
    k=k+1;
end
end

```

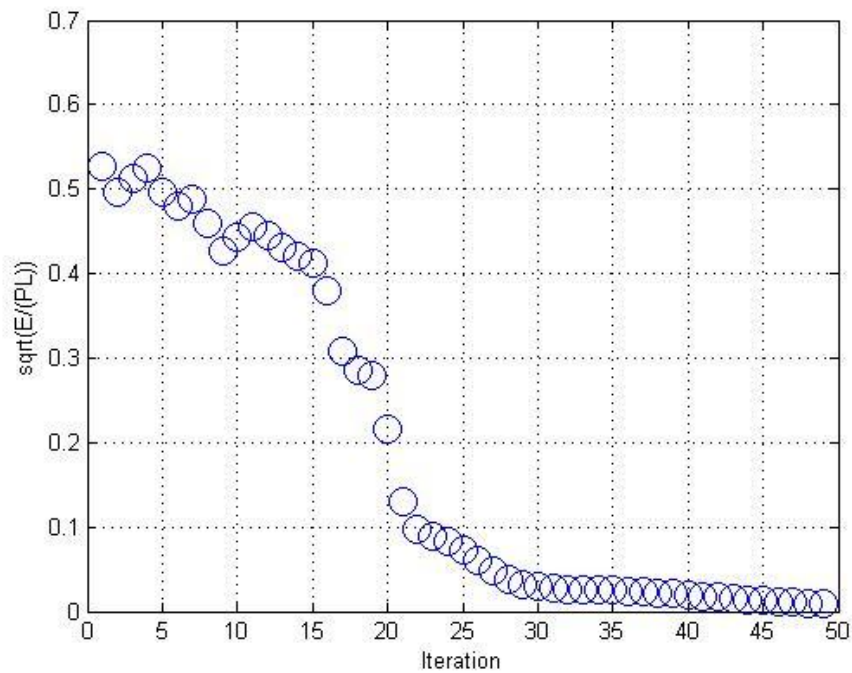
حال دستور زیر را در command window اجرا می‌کنیم:

```

>> x=[1 1 1; 1 0 1; 0 1 1; 0 0 1];
>> d=[1;0;0;1];
>> [k,E,V,W]=BPNN(2,2,1,x,d,0.7,0.99,0.01,3000)

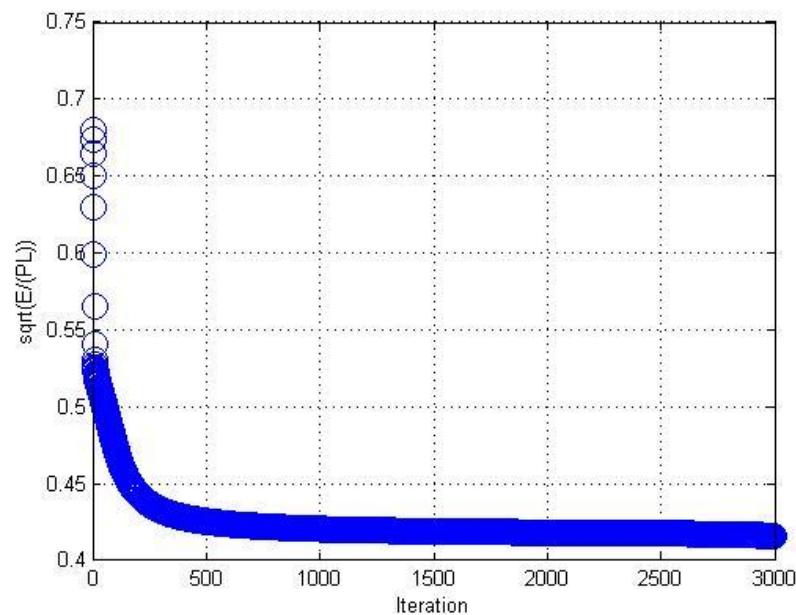
```

k =	E =
50	3.4554e-004
V =	W =
-15.6519 14.2220	-10.5385
13.0268 -11.3804	-19.7502
-9.7300 -9.0032	6.1398

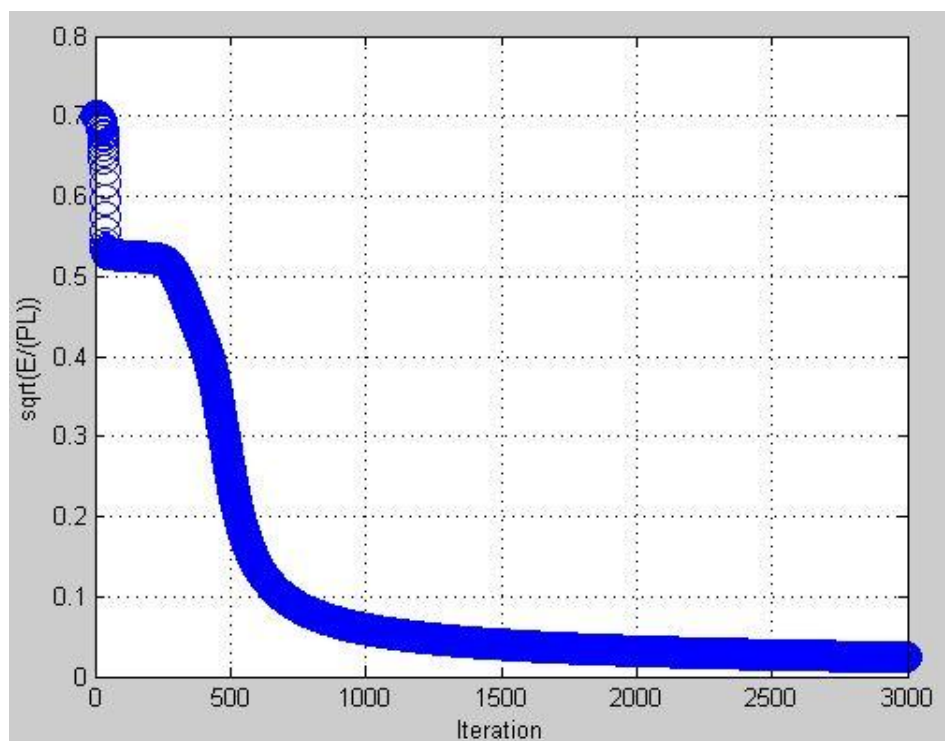


شکل ۳-۶ نمودار مقدار خطا بر حسب دفعات تعلیم در شبکه سوال ۱ با $\eta = 0.7$ & $\alpha = 0.99$

البته با توجه به اینکه مقدار شتاب تصحیح عدد بالایی در نظر گرفته شده بود (۰.۹۹) سیستم برخی اوقات در کمینه موضعی گیر می‌کرد به طوری که خطا از حدی کمتر نمی‌شد اما در حالتی که از کمینه موضعی رد می‌شد به سرعت و در تعداد دفعات کم (۵۰) ب هسرعت همگرا می‌شد.




```
>> [k,E,V,W]=BPNN(2,2,1,x,d,0.7,0.3,0.01,3000)
```

k =**3000****E =****0.0023****V =****-4.6741 -6.1609****W =****-9.1264****-4.6684 -6.1311****9.3673****6.9492 2.4722****4.2937**

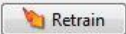
شکل ۳-۷

ب ۲) مقایسه با Toolbox شبکه عصبی MATLAB®

با استفاده از این Toolbox در MATLAB® شبکه را به صورت زیر تعلیم دادیم:

Train Network
Train the network to fit the inputs and targets.


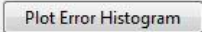

Train Network:
Train using Levenberg-Marquardt backpropagation. (trainlm)



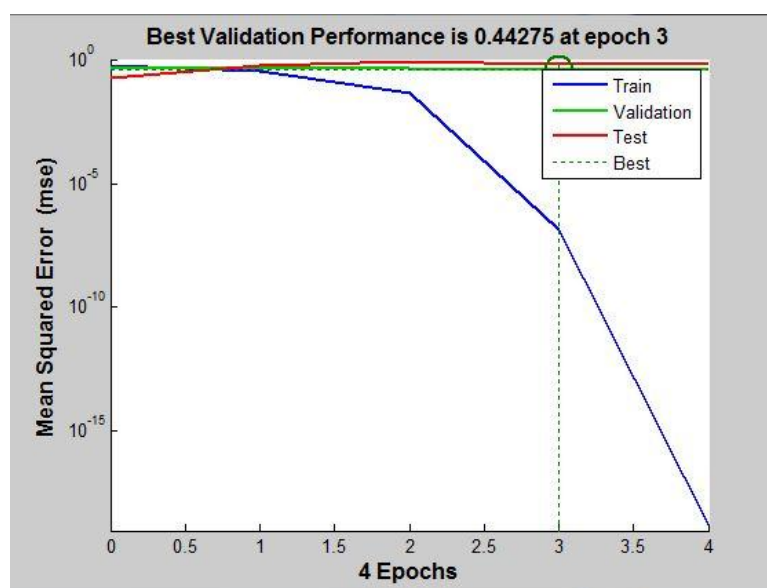
Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

Results:

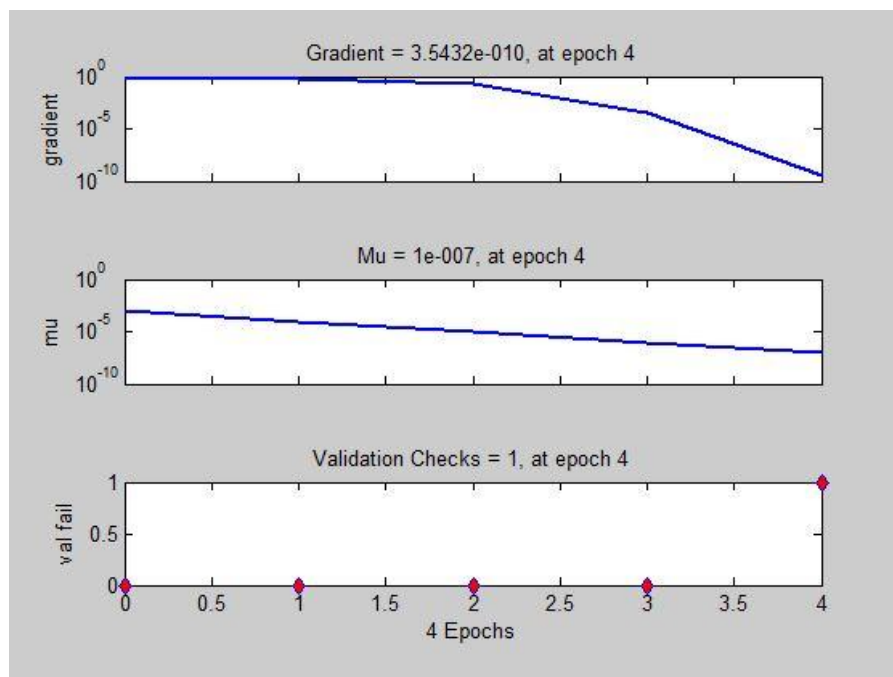
	Samples	MSE	R
Training:	2	1.48333e-7	9.99999e-1
Validation:	1	4.42749e-1	1.00000e-0
Testing:	1	6.78525e-1	9.99999e-1

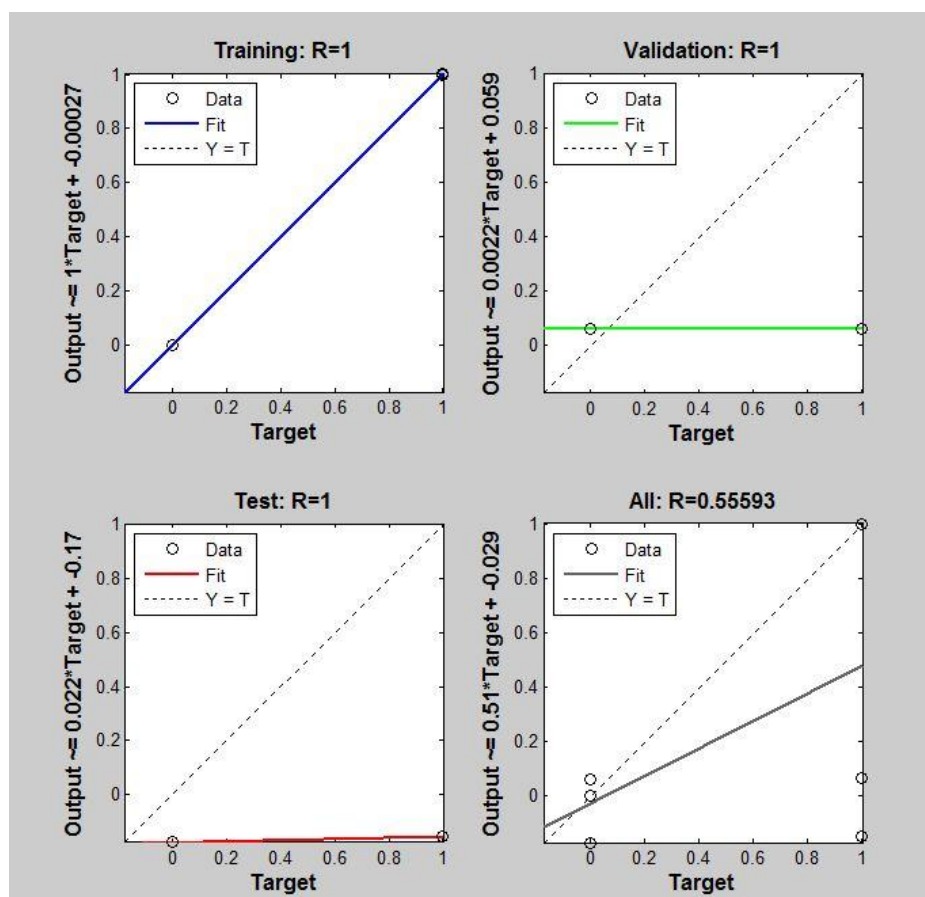
شکل ۳-۸ مرحله تعلیم و تست



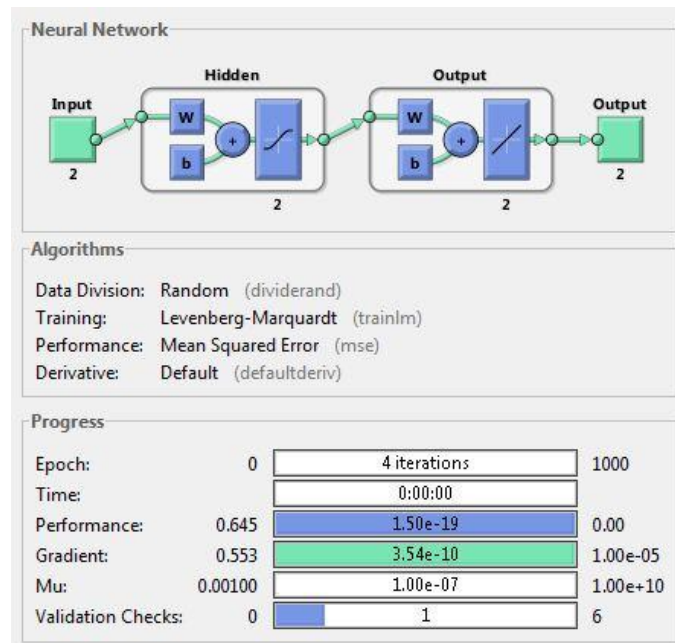
شکل ۳-۹ نمودار میانگین مربعات خطا برای هر ورودی



شکل ۳-۱۰ نمودارهای گرادیان، میو و ولیدیشن چک برای شبکه



شکل ۳-۱۱



شکل ۳-۱۲

```
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by NFTOOL
% Created Sun Apr 28 13:40:50 IRDT 2013
%
% This script assumes these variables are defined:
%
%   xd - input data.
%   xd - target data.

inputs = xd';
targets = xd';

% Create a Fitting Network
hiddenLayerSize = 2;
net = fitnet(hiddenLayerSize);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

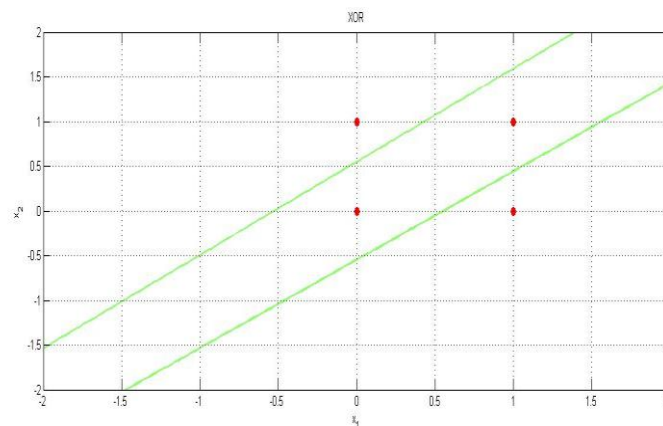
% View the Network
view(net)
```

شکل ۳-۱۳: MATLAB برای تولبک شبکه عصبی

(ج)

با رسم رویه‌ها درمی‌یابیم که در در حالت الف و ب رویه‌ها بسیار به هم شبیه هستند. تنها تفاوتی که میان آن‌ها وجود دارد این است که در حالت دوم مرزها از یک وضعیت رندم شروع شده و در نهایت به مکان نسبتاً مناسبی در فضا بین نقطه‌های ورودی می‌رسند در حالی که در حالت الف مرزها را ما خود به صورت دستی انتخاب می‌کنیم.

مرز تصمیم در حالت تعلیم یافته در شکل زیر نشان داده شده است:



شکل ۳-۱۴ مرزهای تصمیم توسط تعلیم شبکه به صورت کد دستی در MATLAB

جواب ۲:

(الف)

در چه صورت هر دو شبکه در هنگام تعلیم همگرا خواهند شد؟

شرط لازم برای وجود شبکه وارون یک به یک بودن شبکه است. یک شبکه در صورتی یک به یک است که به ازای هر ورودی منحصر به فرد x یک خروجی منحصر به فرد z تحویل دهد. این واقعیت معادل این است که برای یک خروجی خاص از شبکه تنها یک ورودی را می‌توان متناظرش یافت.

البته برای همگرایی شبکه شرایط دیگری همچون مناسب بودن تعداد نورون‌ها، الگوریتم مناسب و ... نیز لازم می‌باشد.

در چه صورت فقط یکی همگرا خواهد شد؟

واضح است که در صورتی که شبکه یک به یک نباشد در بهترین حالت تنها یکی از شبکه‌ها توانایی همگرا شدن را خواهد شد. شرط یک به یک بودن برای یک شبکه تنها در صورتی برآورده می‌گردد که در یک شبکه چند لایه تعداد نورون‌های لایه ورودی و خروجی با هم برابر باشند، بنابراین در حالت تفاوت در تعداد نورون‌های ورودی و خروجی، تنها یکی از شبکه‌ها همگرا خواهند شد. که شبکه همگرا در این حالت شبکه ایست که نورونهای لایه ورودی اش از نورون های خروجی اش بیشتر باشد.

(ب)

۲ الگوریتم پیشنهاد می‌شود:

روش اول) یادگیری همزمان: در این روش به اینگونه عمل می‌کنیم که دو شبکه با هم آموزش داده می‌شوند. ابتدا برای یکی از شبکه‌ها در تکرار اول یادگیری بر اساس الگوریتم پس انتشارخطا مقادیری برای ماتریس وزن‌های نورون‌های لایه اول و دوم بدست می‌آوریم. حال به سراغ شبکه دوم رفته و با توجه به این مسئله که خطا باید هم در رسیدن به خروجی مطلوب و هم در رسیدن به مقادیر نورون‌های لایه دوم در مرحله قبل کمینه شود شبکه را آموزش می‌دهیم. و این کار را

عینا برای شبکه رفت تکرار می‌کنیم. تا آنجایی که هم خروجی‌ها مطلوب شوند هم نورون‌های لایه دوم کاملاً به هم برسند. برای یک شبکه در تکرار اول یادگیری بر اساس الگوریتم پس انتشار خطا مقادیری برای ماتریس وزن‌های نورون‌های لایه اول و دوم بدست می‌آوریم که با استفاده از این ضرایب می‌توان مقدار نورون‌های لایه پنهان را در این تکرار تعیین کرد. حال با توجه به این مسئله، می‌توان شبکه برگشت را به صورت دو شبکه تک لایه مجزا فرض کرد که مقادیر خروجی مطلوب هر کدام مشخص است. پس می‌توان دستورات یادگیری شبکه‌های تک لایه را به ادامه الگوریتم اضافه کرد و در انتهای هر تکرار خطای شبکه‌های تک لایه را به خطای شبکه رفت اضافه کرد.

روش دوم) یادگیری ترتیبی: یکی از معایب این روش احتمال عدم همگرایی است ولی الگوریتم آن ساده تر است. بدین صورت که یادگیری ۲ شبکه رفت و برگشت را پشت سر هم و نه همزمان انجام داد. بدین گونه که بعد اتمام یادگیری شبکه رفت بر اساس الگوریتم یادگیری پس انتشار خطا، با داشتن ماتریس وزن‌های نهایی لایه‌ی پنهان، مقادیر قطعی خروجی نورون‌های لایه پنهان بدست می‌آید. حال می‌توان با استفاده از یکی از روش‌های یادگیری شبکه‌های تک لایه مانند روش دلتا شبکه برگشت را به صورت ۲ شبکه تک لایه پشت سر هم تعلیم داد.

جواب ۳:

(۱) ستفاده از مقادیر تصادفی برای وزن‌های نورون‌های اضافه شده و حفظ مقادیر ناشی از تعلیم‌های گذشته برای سایر نورون‌های همین لایه یکی از روشها می‌باشد.

(۲) روش دیگر این است که به وزن‌های نورون‌های اضافه شده صفر اختصاص دهیم و سایر وزن‌ها را همان‌طوری که بود بی‌تغییر باقی بگذاریم. از آنجایی که تعلیم شبکه قبل از اضافه کردن تعداد نورون‌ها باعث شده است که بردار وزن‌های شبکه به سمت خروجی مطلوب جهت گیری کند، هر چند که این جهت ممکن کاملاً صحیح نباشد اما به هر حال از جهت‌های تصادفی مطمئن‌تر است. پس اگر از روش دوم استفاده کنیم بردار نهایی در جهت تقریبی مورد نظر، خواهد بود. پس روش دوم منطقی‌تر به نظر می‌رسد.

(۳) همچنین می‌توان از وزن‌های لایه پنهان برای نورون‌های اضافه شده استفاده نمود و وزن‌های بقیه لایه‌ها که تعلیم دیده اند را دست نخورده باقی بگذاریم. با توجه با اینکه نورون‌های لایه پنهان در جهت نتیجه مطلوبی که پس از اضافه شدن لایه جدید عملاً نادرست محسوب می‌شوند، استفاده از مقادیر تصادفی نسبت به این روش به نظر بهتر می‌رسد و به جواب درست نزدیک‌تر خواهد بود.