Destiny McClain

Professor Michael Spivey

COGS 104

7 May 2020

Assignment 13: Make a two-layer backpropagation network that solves the XOR problem

clear

%set epoch counter to zero

epoch=0;

%set mse to a high number

mse=999;

%set learning rate

lrate=0.2;

%input patterns

Inputs=[ 0 0; 1 0; 0 1; 1 1];

%target patterns

Targets=[ 0; 1; 1; 0];

%randon weight matriz number 1 and its momentum

W1=rand(2,2)\*2-1;

momentum\_W1=zeros(2,2);

%randon weight matrix number 2 and its momentum

W2=4\*rand(2,1)\*2-2;

momentum\_W2=zeros(2,1);

% hidden bias and its momentum

bias\_h=rand(1,2)\*2-1;

momentum\_h=zeros(1,2);

%output bias and its momentum

bias\_o=rand\*2-1;

momentum\_o=0;

yeet=[];

while mse>0.01 %&& epoch<5000

%increment epoch counter

epoch=epoch+1;

if epoch>5000

break

end

for n=1:4 %four I/O patterns

%set "in" to the nth row of your "Inputs" matrix

in=Inputs(n,:);

%set "targ" to the nth row of your "Targets" Matrix

targ=Targets(n);

%compute the hidden layer's net activation

hid=in\*W1+bias\_h; %input, passing thourgh the matrix, and adding the bias that each hidden node has

hid=1./(1+exp(-hid));

%compute the output layer's net activation

out=hid\*W2+bias\_o;

out=1./(1+exp(-out));

%record the output node's squared error for that I/O pattern

err=targ-out; %raw error

sqerr(n)=mean(err.^2);

%compute deltas for both W1 and W2

delta\_out=err.\*(out.\*(1-out)); %pg 329

delta\_hid=hid.\*(1-hid).\*(delta\_out\*W2');

%compute changes in all weights...

W1ch=(in'\*delta\_hid)\*lrate; %presynaptic activation

W2ch=(hid'\*delta\_out)\*lrate; %presynaptic activation

bias\_hch=delta\_hid\*lrate;

bias\_och=delta\_out\*lrate;

%update weight matrices

W1=W1+W1ch+(0.9.\*momentum\_W1);

W2=W2+W2ch+(0.9.\*momentum\_W2);

%update hidden bias

bias\_h\_change=bias\_hch+(0.9.\*momentum\_h);

bias\_h=bias\_h+bias\_h\_change;

%update output bias

bias\_o\_change=bias\_och+(0.9.\*momentum\_o);

bias\_o=bias\_o+bias\_o\_change;

%capture weight changes in both matrices

momenutm\_W1=W1ch;

momentum\_W2=W2ch;

%capture current output bias weight changes

momentum\_bias\_o=bias\_o\_change;

%capture current input bias weight changes

momentum\_bias\_h=bias\_h\_change;

end

%compute the mse for all four I/O patterns and print out eahc time

mse=mean(sqerr);

mse;

%Keep track of the mse for each training epoch

yeet(epoch)=mse;

end

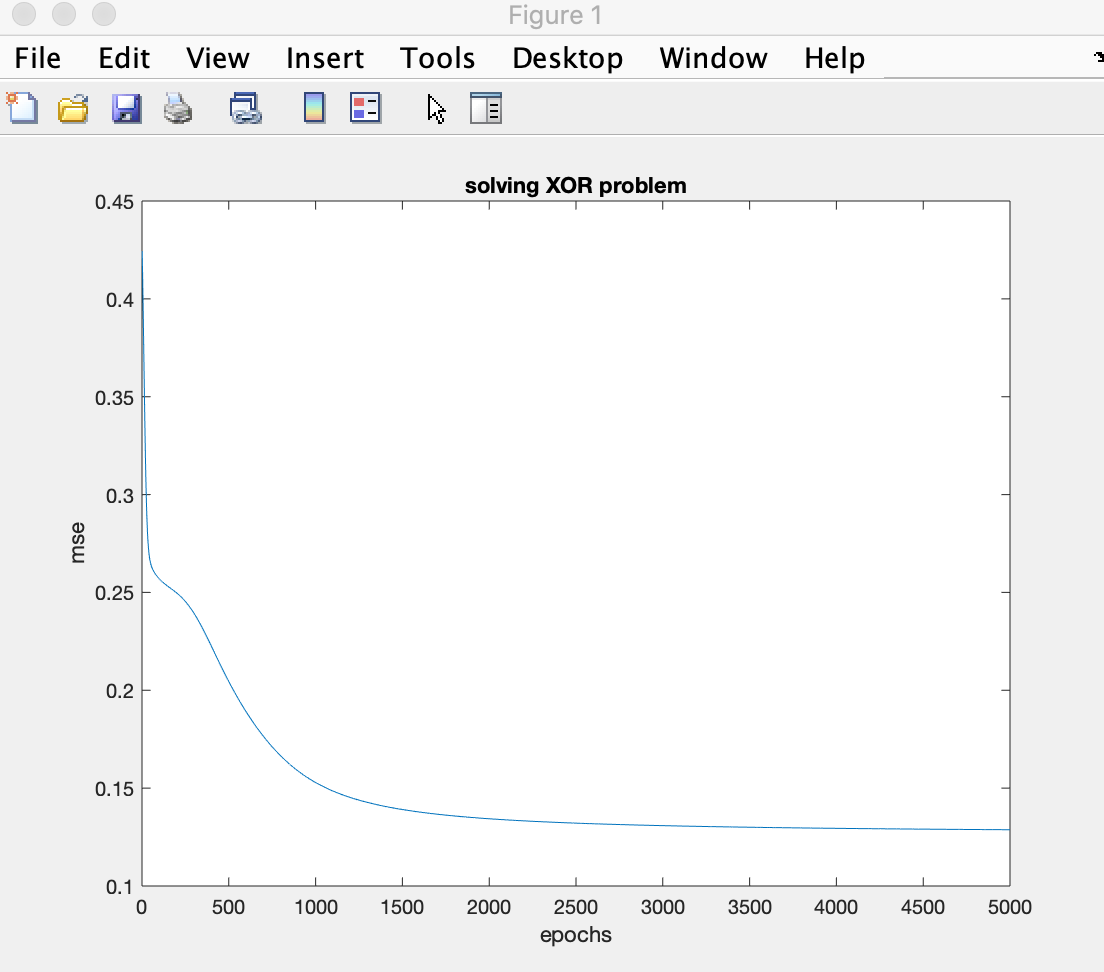
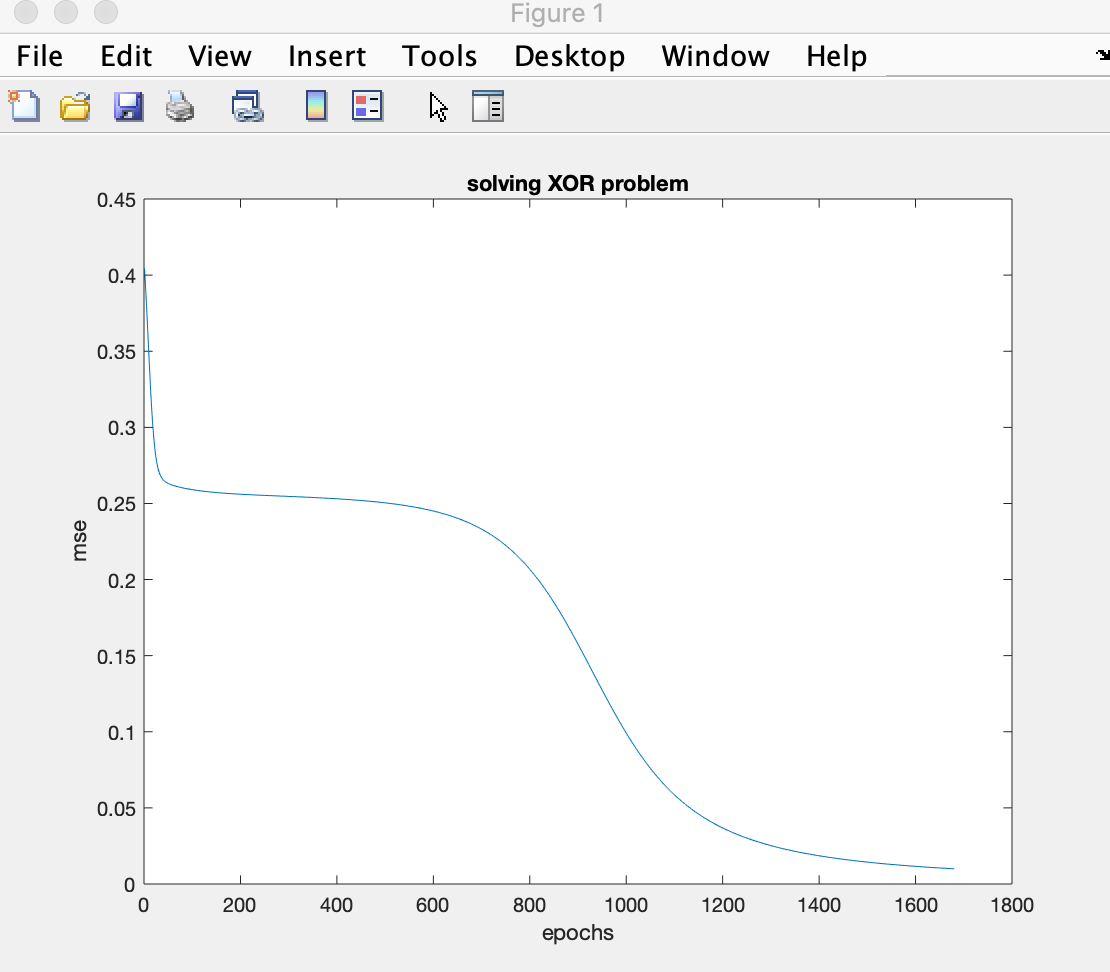
plot(yeet)

xlabel('epochs')

ylabel('mse')

title('solving XOR problem')

**Trial 1 Trial 2**



**Trial 3**

