QUESTION 1:

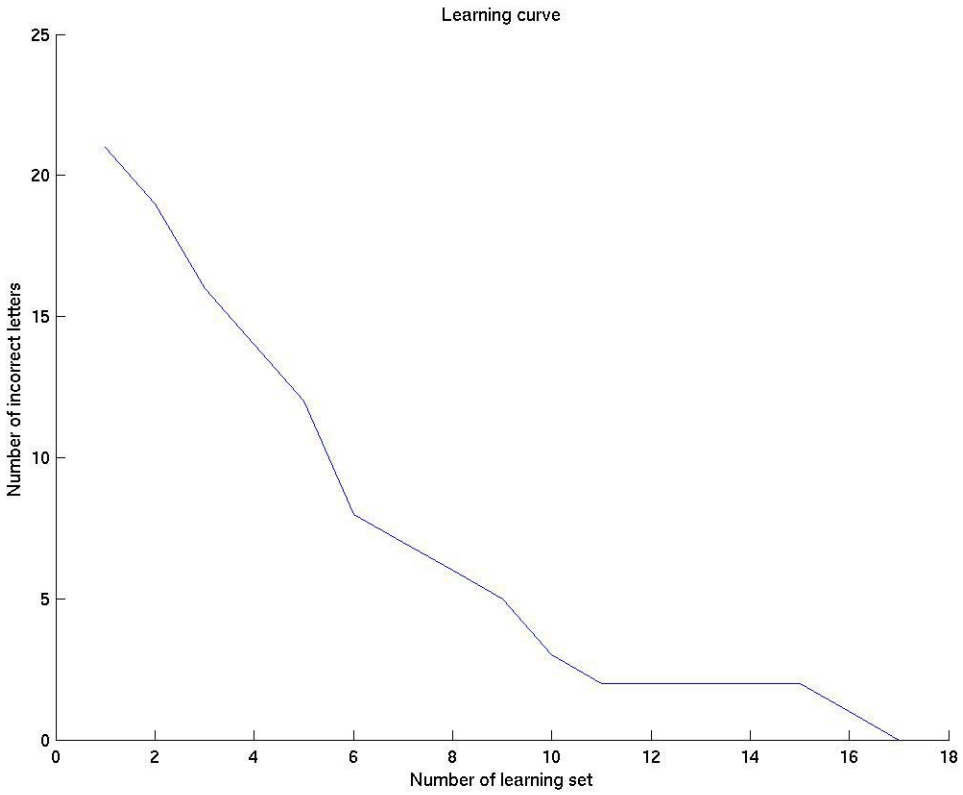


Figure 1 Learning curve of the network versus the training steps

[Program: Hebbian.m]

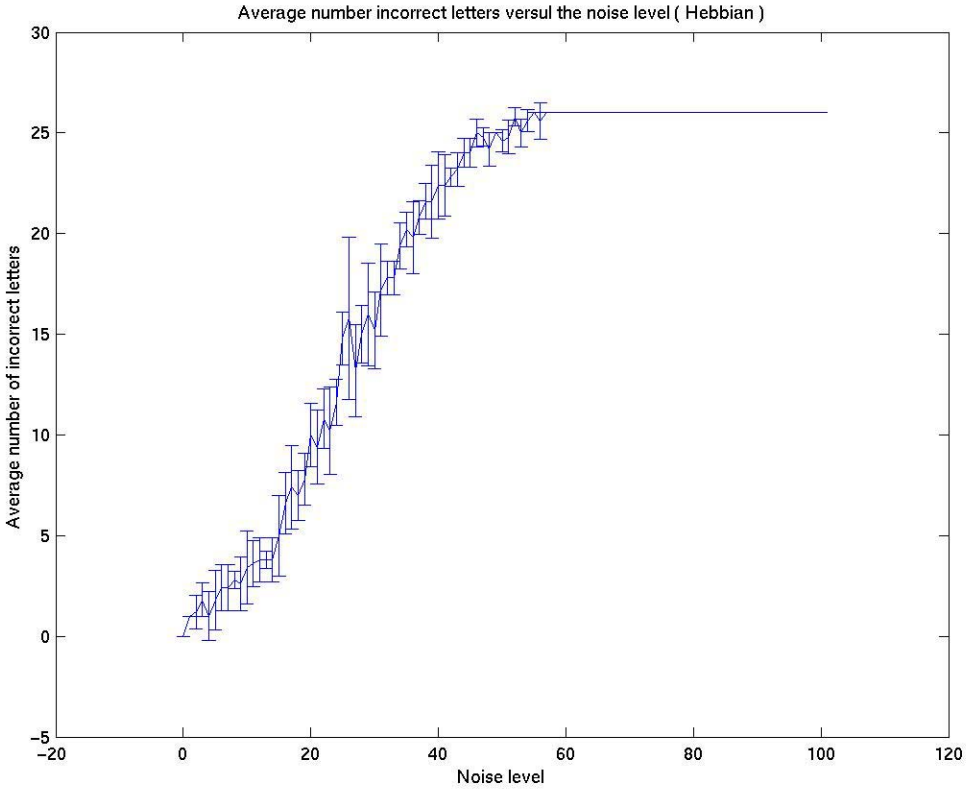


Figure 2 Average number of incorrectly recognized letters versus the noise level error bars for 5 repeat testing

[Program evaluationHebbian.m]

In our experiment, the training steps change for each training time from 10 to hundreds. We can train the network with one step to recognize 25 letter patterns. By turning the learning rate and weights, it is possible to train to network with only one step to recognize all letter patterns.

QUESTION 2:

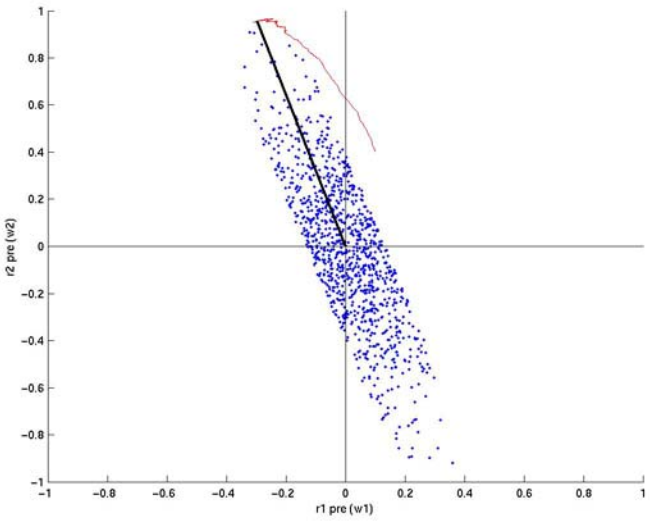
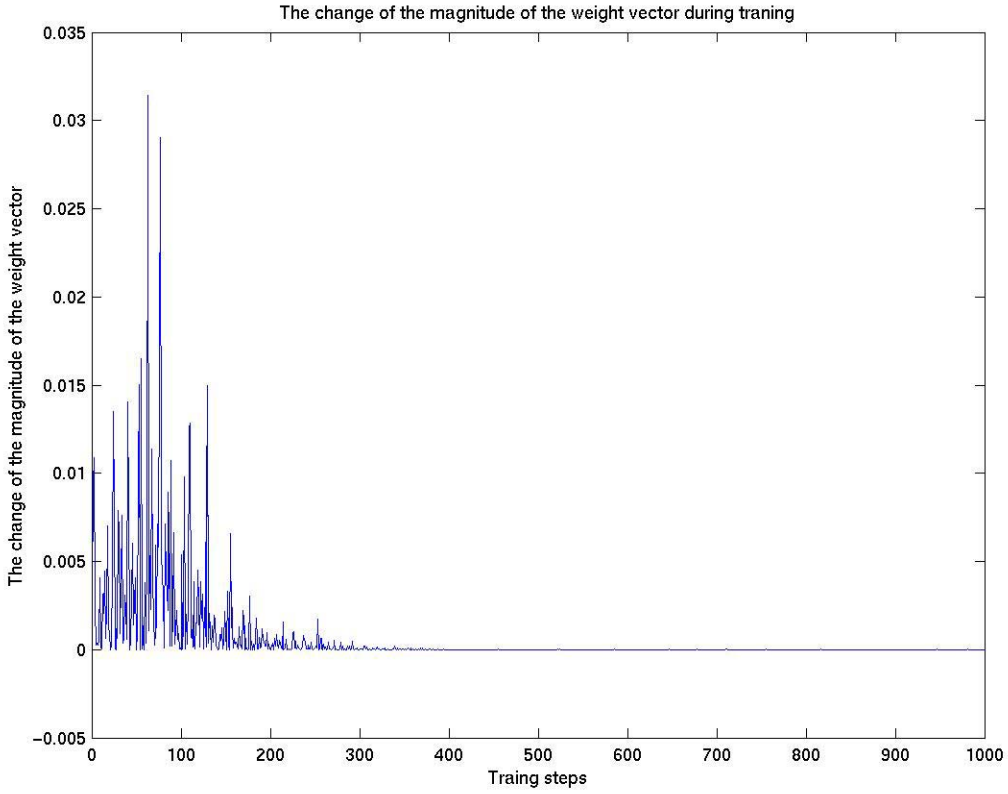


Figure 3 Use Oja's learning rule to find the first principal component x=0.1*[x1+x2 ; x1-3*x2], where x1=2*(rand-0.5) and x2=randn



The first principal component is (-0.2719, 0.9624)

Program 1 Hebbian.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Hebbian learning rule pattern recognition %
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear;
intCount=0;
% Reading pattern from pattern file

load pattern1;

% Reading desired output matrix file
load desiredOutputMatrix;

% Reshaping pattern to 156*26
letterVectors=reshape(pattern1', 12*13, 26);

% Set learning constant and number of training
learningConstant=0.01;
trainingNumber=1;

disp('Begin training, please wait!');

% Initialize weights to random values, 26*156
weights=rand(26,156).*0.2

% repeat until the error number is zero
errorNumber=26
intPlay=0
while  ((errorNumber>0) & (intPlay<200))
    intPlay=intPlay+1;
    for letterNumber=1:26

        % Apply sample patterns to the input nodes
        inputLetterVector=letterVectors(:,letterNumber); % 156*1

        % Calculate the delta weight
        deltaWeights=
desiredOutputMatrix(:,letterNumber)*inputLetterVector';
        deltaWeights=deltaWeights*learningConstant;
        % Update the weights matrix by adding the delta term
        weights=weights+deltaWeights;
    end


    % Test output with trained weight matrix
    errorNumber=0;
    for letterNumber=1:26

        % Apply sample patterns to the input nodes
        inputLetterVector=letterVectors(:,letterNumber);

        % Cauculate the rate of output nodes
        outputVector=weights*inputLetterVector;
```

```
        % Apply game function to the output nodes
        outputVector= gainFunction(outputVector);

        % Compare the output with the desired output, update error
number variable
        if ~isequal(desiredOutputMatrix(:,letterNumber),outputVector)
            % error output
            errorNumber=errorNumber+1;
        end
    end
    intCount=intCount+1;
    % Update the learning curve X Y set
    X(trainingNumber)=trainingNumber;
    Y(trainingNumber)=errorNumber;

    % Add the number of training set
    trainingNumber=trainingNumber+1;
end

% Plot the learning curve
hold on;
plot(X,Y);
%bar(X,Y,0.4);
hold off;
title('Learning curve');
xlabel('Number of learning set');
ylabel('Number of incorrect letters');
disp('End of training, enjoy it.');

% Save weight matrix  to file 'weights'
save('weights','weights','-ASCII');
```

Program 2 gainFunction.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Gain Function                 %
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function g=gainFunction(x)
% Gain fuction of Hebbian Learning
% return 1 if x>=max
% return 0 else
maxNum=max(x);
g=zeros(size(x));
g(x>=threshold)=1;
```

Program 3 evaluationHebbian.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Evaluation of Hebbian pattern recognition %
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
clear;

% Reading pattern from pattern file
load pattern1;

% Reading desired output matrix file
load desiredOutputMatrix;

% Reading weight matrix
load weights;

% Reshaping pattern to 156*26
letterVectors=reshape(pattern1', 12*13, 26);

disp('Begin testing input with noise!');

% Set the repeat test number of certain noise level
totalLoopNumber=5;

timeStart=now;
for noise=0:100
noise
    X(noise+1)=noise;
    for loopIndex=1:totalLoopNumber

        % Initialize the error number
        errorNumber=0;

        % Apply sample patterns to the input nodes
        inputLetterVectors=letterVectors;

        % Make nosie version
        for i=1:26
            noiseVec=ranBinVec(156,noise*156/100);

inputLetterVectors(noiseVec==1,i)=inputLetterVectors(noiseVec==1,i)==0;
        end

        % Cauculate the rate of output nodes
        outputVectors=weights*inputLetterVectors; % 26*26
        for letterNumber=1:26

            % Apply game function to the output nodes
            for i=1:26

outputVectors(:,letterNumber)=gainFunction(outputVectors(:,letterNumber
));
            end

            % Compare the output with the desired output, update error
number variable
            if
~isequal(desiredOutputMatrix(:,letterNumber),outputVectors(:,letterNumb
er))
                % Error output
                errorNumber=errorNumber+1;
            end
```

```
        end

        % Update the recognition rate versus noisie level curve X Y set

        Y(loopIndex,noise+1)=errorNumber;
    end
end;

timeEnd=now;
disp('Total time used for recognitions: ');
disp(datestr(timeEnd-timeStart,'HH:MM:SS'));

% Compute the average error number
AverageY=mean(Y);

% Comput the standard deviation
S=std(Y);

% Plot the error bar
errorbar(X,AverageY,S);
title('Average number incorrect letters versul the noise level (
Hebbian )');
xlabel('Noise level');
ylabel('Average number of incorrect letters');

disp('End of testing input with noise.');
```

Program 4 oja.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Linear associator with Hebb and weight decay: PCA Oja  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear; clf; hold on;
y=0; w=[0.1;0.4]; a=-pi/6;
%rot=[cos(a) sin(a); -sin(a) cos(a)] % rotation matrix

pre=sqrt(w(1)^2+w(2)^2);
for i=1:1000
%    x=0.05*rand(2,1); x(1)=4*x(1); % traning examples
    x1=2*(rand-0.5);
    x2=randn;

    x=[x1+x2 ;x1-3*x2];
    x=0.1*x;

    %x=rot*x; % rotation of training examples
    y=w'*x; % network update
    plot(x(1),x(2),'.');
    w=w+0.1*y*(x-y*w); % training
    w_traj(:,i)=w; % recording of weight history
    c(i)=sqrt(w(1)^2+w(2)^2)-pre;
    pre=sqrt(w(1)^2+w(2)^2);
```

```
end


plot(w_traj(1,:), w_traj(2,:),'r');
plot([0 w(1)], [0 w(2)], 'k','linewidth',2);
axis([-1 1 -1 1]);
plot([-1 1],[0 0], 'k');
plot([0 0], [-1 1],'k');

xlabel('r1 pre (w1)');
ylabel('r2 pre (w2)');

figure(2)
plot(1:1000,c)
title('The change of the magnitude of the weight vector during
traning');
xlabel('Traing steps');
ylabel('The change of the magnitude of the weight vector');
w
```