Exercise 3: The Hebb Rule

[10 points, handed out on 16.04.2008, submit until 07.05.2008]

Apply Hebb's rule for a linear neuron. The supplied datasets have dimension 2. The linear neuron computes the output

\[ y = \sum_{i=1}^{2} w_i x_i. \]

The given Matlab code shall be completed by you. Those parts of the code which should be changed are marked by %------------. Make the following changes in the code:

- choose a learning rate \( \gamma \).
- choose the number of simulation steps.
- insert code to compute the correlation matrix \( C \)
- insert code to compute the weight updates (Hebb's rule).

Apart from this, the code is complete. The dataset and the normalized weight vector is plotted. The eigenvectors and eigenvalues of the correlation matrix \( C \) are computed and printed on the screen.

The program loads the dataset \texttt{ue3_dp1.mat} (first dataset). To analyse the second dataset, exchange the line \texttt{load ue3_dp1.mat} with \texttt{load ue3_dp2.mat}.

- Submit a printout of the completed program.
- Does the weight vector converge to the eigenvector with the largest eigenvalue for dataset 1, 2? If not, why?
- Does the weight vector maximize (approximately) the variance of the output on datasets 1, 2? Make plots to show that. If not, why?

Matlab Programm

Use the dataset in the ZIP directory \texttt{http://www.igi.tugraz.at/lehre/NNB/SS08/aufgaben/ue3dp.zip}.
Use the following Matlab code from \texttt{http://www.igi.tugraz.at/lehre/NNB/SS08/aufgaben/nnb_hebb.m}.

```matlab
clear all;
close all;

% Learning rate
% Choose a good learning rate
%------------
gamma = 0;
%------------
% Number of training steps
% choose it
%------------
```
numEpochs = 5000;
%-------------------

% dataset X (p times 2 matrix)
% X(i,:) is the i-th Input-Vector
% the dataset is loaded
% the commented line loads dataset 2
%-------------------
load ue3_dp1.mat;
% load ue3_dp2.mat;
%-------------------

% number of datapoints
p = size(X,1);

% plot data
figure(1);
clf;
plot(X(:,1),X(:,2),'.');

% Compute correlation matrix
% use matrix multiplication
%-------------------
C = rand(2,2);
%-------------------

% eigenvalues and eigenvectors
% eigV(:,i)...i-th eigenvector of C
% lambda(i,i)...i-th eigenvalue of C
[eigV, lambda] = eig(C);

% initialize weight vector
% random initialization
w = rand(2,1);

% number of graphics updates
inter_epoch = 5;
for i=1:numEpochs
  % one learn step

    % choose datapoint
    mu = ceil(50*rand(1,1));
    X_mu = X(mu,:); % momentary data point

    % compute output y
    y = X_mu*w;

    % Apply learning rule
    % code Hebb's rule here
    %-------------------
    w = w;
    %-------------------
% Plot new weight vector
if(mod(i,inter_epoch)==0)
    fprintf('Epoch %g
',i);
    figure(1);
    clf;
    plot(X(:,1),X(:,2),'.');
    hold on;
    w_normed = w/norm(w);
    plot([0,w_normed(1)],[0,w_normed(2)],'r-');
end
end

% Plot final weight vector
figure(1);
clf;
plot(X(:,1),X(:,2),'.');
hold on;
w_normed = w/norm(w);
plot([0,w_normed(1)],[0,w_normed(2)],'r-');

% Print eigenvalues and eigenvectors
for i=1:2
    fprintf('Eigenwert %g: %g
',i,lambda(i,i));
    fprintf('Eigenvektor %g: (%g, %g)
',i,eigV(1,i),eigV(2,i));
end