```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            NEUROGENESIS RBM
% -------------------------------------------------------------------------
% Execute this file second.
%
% NOTE:
% epsilon (learning rate) = 0.3
% sparsity cost = 0
% weight decay = 0.005
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% VARIABLES:---------------------------------------------------------------

trialNum = 10;
visible_data = zeros(trialNum,10);
hidden_data = zeros(trialNum,50);
visible_recon = zeros(trialNum,10);
hidden_recon = zeros(trialNum,50);
delta_weight = zeros(10,50);
weight = zeros(10,50);
Pmat_h = zeros(1,50);
Pmat_v = zeros(1,10);
Pmat_h_r = zeros(1,50);

test_visible_data = zeros(trialNum,10);
test_hidden_data = zeros(trialNum,50);
test_visible_recon = zeros(trialNum,10);
test_hidden_recon = zeros(trialNum,50);

p = 0.05; % target activation
q = zeros(1,trialNum);

weightdecay = 0.005;

% INITIALIZE PATTERN INPUT % WEIGHTS **************************************

% Randomize visible unit pattern:------------------------------------------
% length = 10;
% percent = 0.1;
% [orig_data v1 v2 v3 v4 v5 v6 v7 v8 v9 v10] = patternGen(length, percent);
%
% visible_data = [v1; v2; v3; v4; v5; v6; v7; v8; v9; v10];

% Randomize values into weight matrix:-------------------------------------

% !!! Computations if NOT using normal distribution.

% for i = 1:size(weight,1)
%     for j = 1:size(weight,2)
%         weight(i,j) = rand(1);
%     end
% end
```

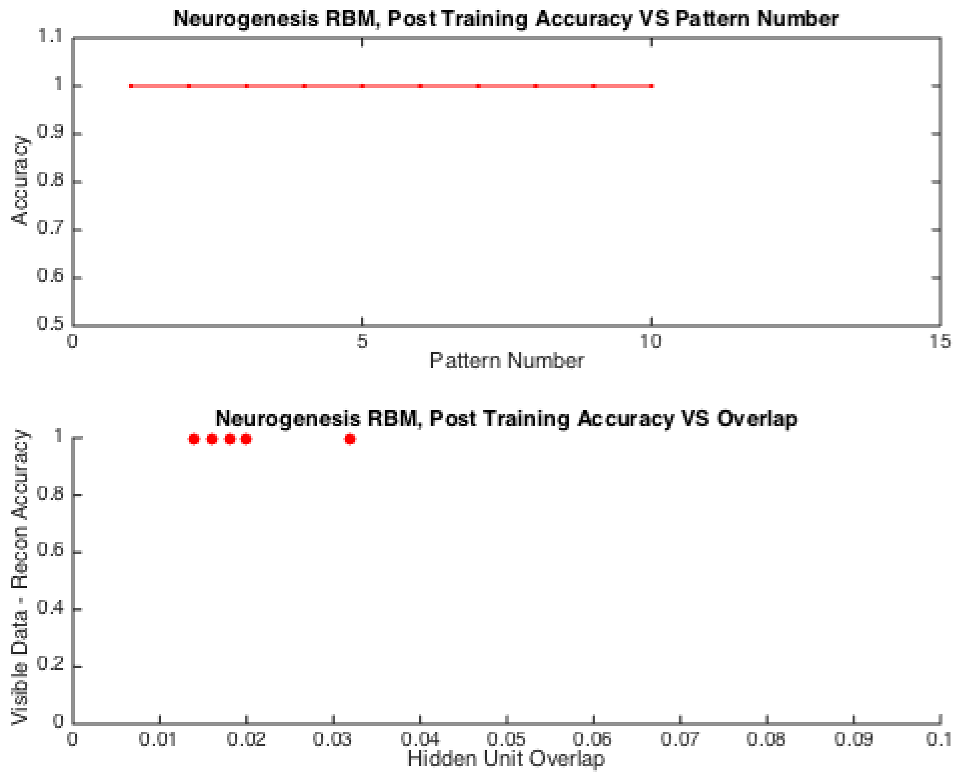
```
%
% weight;

% !!! Computations if randomizing weight matrix using a normal
% distribution.

sigma = 0.0001;
mu = 0.0;
weight = normrnd(mu, sigma, 10, 50); % Random normal distribution fxn.
%weight(weight<0) = 0; % Make negatives into zeros.
weight;

% COMPUTE LEARNING RATE 1-g(t) AND SPARSITY COST(g(t)) ********************

% !!! This code should only be used when factoring in sigmoidal aging.

epsilon = zeros(1,trialNum); % Learning rate
gt_vect = zeros(1,trialNum); % Sparsity cost
dt = 1/trialNum;

for i = 1:trialNum
    gt_vect(i) = gompertz(-1+(dt*i)); %add to the min. which is -1
    epsilon(i) = 1-gt_vect(i);
end

gt_vect;
epsilon;

gt_normalized = normalize_var(gt_vect, 0.1, 0.3);
epsilon_normalized = normalize_var(epsilon, 0, 0.9);

% !!! Make epsilon and cost constants for trials when not accounting for
% sigmoidal aging.

    % Learning rate 1-g(t) = 0.1 for non neurogen; 0.3 for neurogen.
    % Sparsity cost = 0.9 for non neurogen; 0 for neurogen.


% COMPUTE VISIBLE DATA, HIDDEN DATA, AND THE RECONS ***********************

for i = 1:size(visible_data, 1) %i is trial number

    % Getting values of Pmat_h:
    Pmat_h = 1./(1+exp(-visible_data(i,:) * weight));

    % Set hidden units to 1 depending on the values of Pmat_h:
    for j = 1:50;
        if rand(1) < Pmat_h(j);
            hidden_data(i,j) = 1;
        end
    end

    q(i) = mean(hidden_data(i,:));
```

```
    % Getting values of Pmat_v:
    Pmat_v = 1./(1+exp(-hidden_data(i,:) * weight'));


    % Set visible recon units to 1 depending on the values of Pmat_v:
    for j = 1:10;
        if rand(1) < Pmat_v(j);
            visible_recon(i,j) = 1;
        end
    end

    % Getting values of Pmat_h_r:
    Pmat_h_r = 1./(1+exp(-visible_recon(i,:) * weight));


    % Set hidden recon units to 1 depending on the vales of Pmat_H_r:
    for j = 1:50
        if rand(1) < Pmat_h_r(j);
            hidden_recon(i,j) = 1;
        end
    end


    %!!!!!! Since we are assuming neurons are all new, replaced epsilon and
    %sparsity cost as a constant across all trials:

    for k = 1:size(visible_data,2) %1:10
        for m = 1:size(hidden_data,2) %1:50
            % 0.3 replaced epsilon(i)
            delta_weight(k,m) = 0.3*((visible_data(i,k)*hidden_data(i,m))...
                - (visible_recon(i,k)*hidden_recon(i,m))) - ...
                (weightdecay * weight(k,m)) ...
                - (0 * (q(i)-p));% (0 replaced gt_normalized(i));
        end
    end

    weight = weight + delta_weight; % Update weight

end

% COMPUTE VISIBLE DATA, HIDDEN DATA, AND THE RECONS ***********************

for i = 1:size(visible_data, 1) %i is trial number

    % Getting values of Pmat_h:
    Pmat_h = 1./(1+exp(-visible_data(i,:) * weight));

    % Set hidden units to 1 depending on the values of Pmat_h:
    for j = 1:50;
        if rand(1) < Pmat_h(j);
            hidden_data(i,j) = 1;
        end
    end
```

```
    q(i) = mean(hidden_data(i,:));


    % Getting values of Pmat_v:
    Pmat_v = 1./(1+exp(-hidden_data(i,:) * weight'));


    % Set visible recon units to 1 depending on the values of Pmat_v:
    for j = 1:10;
        if rand(1) < Pmat_v(j);p
            visible_recon(i,j) = 1;
        end
    end

    % Getting values of Pmat_h_r:
    Pmat_h_r = 1./(1+exp(-visible_recon(i,:) * weight));


    % Set hidden recon units to 1 depending on the vales of Pmat_H_r:
    for j = 1:50
        if rand(1) < Pmat_h_r(j);
            hidden_recon(i,j) = 1;
        end
    end


    %!!!!!! Since we are assuming neurons are all new, replaced epsilon and
    %sparsity cost as a constant across all trials:

    for k = 1:size(visible_data,2) %1:10
        for m = 1:size(hidden_data,2) %1:50
            % 0.1 replaced epsilon(i)
            delta_weight(k,m) = 0.1*((visible_data(i,k)*hidden_data(i,m))...
                - (visible_recon(i,k)*hidden_recon(i,m))) - ...
                (weightdecay * weight(k,m)) ...
                - (0.9 * (q(i)-p));% (0.9 replaced gt_normalized(i));
        end
    end

    weight = weight + delta_weight; % Update weight

end

% COMPUTE VISIBLE DATA, HIDDEN DATA, AND THE RECONS FOR POST TRAINING *****

for i = 1:size(visible_data, 1) %i is trial number

    % Getting values of Pmat_h:
    Pmat_h = 1./(1+exp(-test_visible_data(i,:) * weight));

    % Set hidden units to 1 depending on the values of Pmat_h:
    for j = 1:50;
        if rand(1) < Pmat_h(j);
```

```
            test_hidden_data(i,j) = 1;
        end
    end

    q(i) = mean(test_hidden_data(i,:));


    % Getting values of Pmat_v:
    Pmat_v = 1./(1+exp(-test_hidden_data(i,:) * weight'));


    % Set visible recon units to 1 depending on the values of Pmat_v:
    for j = 1:10;
        if rand(1) < Pmat_v(j);p
            test_visible_recon(i,j) = 1;
        end
    end

    % Getting values of Pmat_h_r:
    Pmat_h_r = 1./(1+exp(-test_visible_recon(i,:) * weight));


    % Set hidden recon units to 1 depending on the vales of Pmat_H_r:
    for j = 1:50
        if rand(1) < Pmat_h_r(j);
            test_hidden_recon(i,j) = 1;
        end
    end

end

% VDATA - VRECON / ACCURACY FOR POST TRAINING *****************************

vDiff_PT_neurogen = zeros(1, trialNum);

for i = 1:trialNum
        vDiff_PT_neurogen(i) = 1-(abs(sum(test_visible_data(i,:) - test_visible_re
end


% VDATA - VRECON / ACCURACY FOR DURING TRAINING ***************************

vDiff_DT_neurogen = zeros(1, trialNum);

for i = 1:trialNum
    vDiff_DT_neurogen(i) = 1-(abs(sum(test_visible_data(i,:) - test_visible_recon(
end

% HIDDEN UNIT OVERLAP *****************************************************

hiddenOverlap = zeros(10, trialNum-1);
sizeV = size(visible_data,1) * size(visible_data,2);

for i = 1:trialNum
```

```
    for j = 1:trialNum
        if (j ~= i)
            hiddenOverlap(i,j) = abs(sum(test_hidden_data(i,:)) - sum(test_hidden_
        end
    end
end

hiddenOverlapsum_neurogen = zeros(1,trialNum);

for i = 1:trialNum
    hiddenOverlapsum_neurogen(i) = sum(hiddenOverlap(i,:))/trialNum;
end

hiddenOverlapsum_neurogen;

% PLOTTING ****************************************************************

figure(2)
subplot(2,1,1)
plot(1:10, vDiff_DT_neurogen, '.-r')
axis([0 15 0.5 1.1])
title('Neurogenesis RBM, Post Training Accuracy VS Pattern Number')
xlabel('Pattern Number')
ylabel('Accuracy')

subplot(2,1,2)
scatter(hiddenOverlapsum_neurogen, vDiff_PT_neurogen, 'filled', 'r')
axis([0 0.1 0 1]);
title('Neurogenesis RBM, Post Training Accuracy VS Overlap')
xlabel('Hidden Unit Overlap')
ylabel('Visible Data - Recon Accuracy')
```



Published with MATLAB® R2014b