# Homework 2: Simple Hopfield Net

Andy Reagan

February 10, 2015

## 1 Discussion

After coding up the Hopfield network, which is very simple once I'm done, I have to say that I had a lot of trouble getting the details right. There is a gap between the details given in the literature and the exact coding of this that I seemed to miss, but I do think that I have a working code. At least, it gets the training correctly and settles down for the other inputs that I tested.

The core of the Hopfield network being useful seems to be that it monotonically decreases the energy function given, such that any solution is gauranteed to be a local optimal solution. Techniques like annealing attempt to search the energy space more broadly. So, finding an appropriate energy (Liapunov) function which is minimized by the network and then being able to solve the gradient of that to find the weights is the hard part, and it seems like others have been able to find energy functions which are suitable for solving problems of interest.

## Full code

```
% simple hopfield net implementation
% in MATLAB here
% *uses the ideas from the original paper by hopfield
% and the output here is verbose
% because I don't quite understand what is going on here
% and have been troubleshooting it for awhile
%
% 2015-02-10
% Andy Reagan

% size of our network
% seems to have to be equal to our training data
N = 3;

% threshold for output activation
u = 0;

% training patterns
% are the column
disp('training␣patterns:')
T = [1,0;1,0;0,1];
disp(T);

% set the weights
% via Hopfield
W = zeros(N);
for i=1:length(T(1,:))
    W = W + T(:,i)*T(:,i)';
end
disp('W␣before␣setting␣diag␣to␣0');
disp(W);
% could set the diag of W to 0
% a la the stability results
% but this breaks the desired training!
% W = W - diag(diag(W));
% disp('W after setting diag to 0');
% disp(W);

%% test the training patterns

for j=1:length(T(1,:))
    % these are the values of the nodes
    % set to the training pattern
    V = T(:,j);
    % some pattern we didn't train on
    P = zeros(N,1);
    disp('initial␣state␣of␣the␣nodes');
    disp(V);

    while ~isequal(V,P)
        % for i=1:10
        % store the old
        P = V;
        % compute net sum
        % V = W*V;
        % compute component async
        for j=randperm(length(V))
            V(j) = W(j,:)*V;
        end
        disp('updated␣net')
        disp(V);
        % apply threshold to output
        V = V > u*ones(size(V));
        disp('with␣threshold');
        disp(V);
    end
end

%% test a different pattern
% NOTE: terrible code design
% but MATLAB is annoying with multiple functions in a file

disp('testing␣the␣value␣for␣0,1,0');

V = [0;1;0];
% some pattern we didn't train on
P = zeros(N,1);
disp('initial␣state␣of␣the␣nodes');
disp(V);

while ~isequal(V,P)
% for i=1:10
    % store the old
    P = V;
    % compute net sum
    % V = W*V;
    % compute component async
    for j=randperm(length(V))
        V(j) = W(j,:)*V;
    end
    disp('updated␣net')
    disp(V);
    % apply threshold to output
    V = V > u*ones(size(V));
    disp('with␣threshold');
    disp(V);
end
```