Edward Chen

Psych 186B

703909636

Assignment 2

**Problem 1:**

Using the rand function, I created a vector with 100 random elements and normalized the vectors using the norm function. With the two vectors: g and f, we can find the linear associator by multiplying g with f transposed. After the linear associator is computed, I tested it with g prime which is calculated by multiplying the linear associator with vector f. If calculated correctly, g prime and g should be the same in terms of direction although magnitude could be different.

Code Snippet:

% Generating vectors f and g

f = rand(1,100)-0.5; %-0.5 causes the mean to be zero

g = rand(1,100)-0.5;

% Normalizing the vectors

f = f/norm(f);

g = g/norm(g);

% Computing linear associator, A

% Assuming the learning constant is 1

A = g \* f'; % this is only one association?

% Testing to see if G' == G

g\_prime = A\*f;

g\_prime = g\_prime/norm(g\_prime)

% Finding the cosine theta

theta = dot(g,g\_prime);

theta = acos(theta)\*180/pi; %putting it in degrees

length\_gprime = norm(g\_prime);

**Problem 2:**

This problem tested to see if the linear associator is able to distinguish the difference between input vectors. I also tested to see if the f vector and f prime vector were orthogonal by seeing if the cosine is 90 degrees and if the length is of the same direction.

Code Snippet:

% Generating a new normalized random vector, f'

f\_prime = rand(1,100)-0.5;

f\_prime = f\_prime/norm(f\_prime);

% Testing to see if orthogonal

theta2 = dot(f,f\_prime);

theta2 = acos(theta2)\*180/pi;

% Compute Af' and look at the length of the vector

% it should be g which is in the same direction as g'?

test = A\*f\_prime;

length\_test = norm(test);

**Problem 3:**

I talked about this problem with Ichha so the way we approach the solution may be similar. I created two vectors and a matrix. The matrix will store each vector line by line as it goes through the loop. Each vector is randomized and normalized. After the matrix is filled, the matrix is tested to see if it is orthogonal to the vectors that were created. Again, the testing is to see if the direction is the same by looking at the cosine theta.

Code Snippet:

% Generates a matrix containing normalized vectors

for i=1:50

% Generating random vectors

F\_i = (rand(50,1)-0.5);

G\_i = (rand(50,1)-0.5);

F\_i = F\_i/norm(F\_i);

G\_i =G\_i/norm(G\_i);

% Placing vectors in matrix

Fmatrix(:,i) = F\_i;

Gmatrix(:,i) = G\_i;

Ai= G\_i\*F\_i'; % Uses outer product to find the linear association

A\_sum(i,:) = sum(Ai);

end

**Problem 4:**

I would expect the response of the system to damage to be more responsive if the representation of data is small and to have less of a response as the dimensionalities get bigger. I tested this by putting in a random function in a loop that accessed elements in the matrix and replaced it with 0. The accessing of elements was random in order to simulate noise. I believe it is less responsive because the larger the system, the less likely a change will affect the data. Furthermore, this goes along with what was mentioned in lecture previously in that there is only n vectors that are orthogonal in n dimensionality, meaning if there is a lot of data, then it would not affect it as much since the extra vectors are redundant.