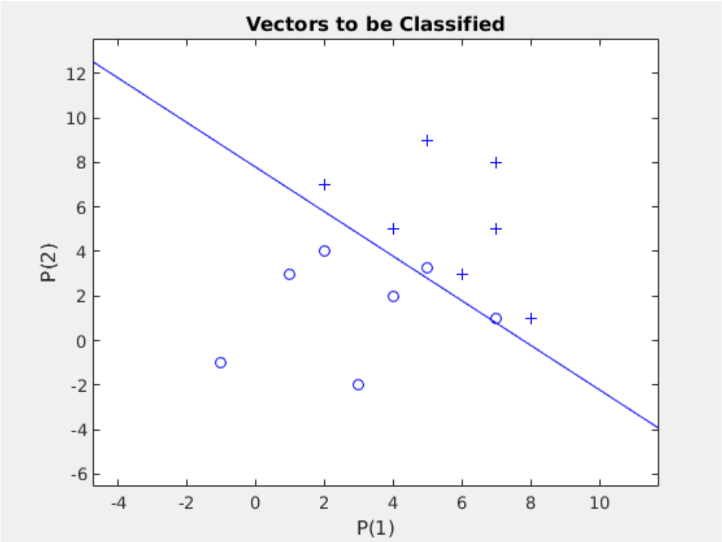
PROBLEM 1:

inputs in class 1 and class 2;
w1= [(2, 7); (8, 1); (7, 5); (6, 3); (7, 8); (5, 9); (4, 5)]
w2 = [(4, 2); (1, 1); (1, 3; (3, 2); (5, 3.25); (2, 4); (7, 1)]

Initially

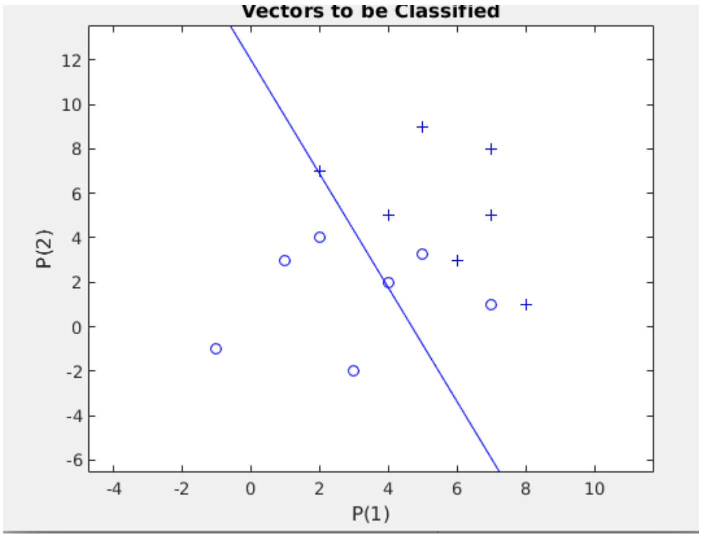
```
w= [
   2 8 7 6 7 5 4 4 -1 1 3 5 2 7
   7 1 5 3 8 9 5  2 -1  3 -2 3.25 4 1];
t1 = [1 1 1 1 1 1 1 0 0 0 0 0 0 0];

plotpv(w, t1);
w0 = -6;
a = [ 0.77 0.77];
plotpc(a, w0);
```



SINGLE SAMPLE PERCEPTRON

Algorithm:In every iteration, we add every misclassified sample(yk) to the weight vector for learning;

```
w0 = -6;
a = [ 0.77 0.77];

[a,w0] = sample_perceptron(w, w0, t1);
plotpc(a, w0)
```



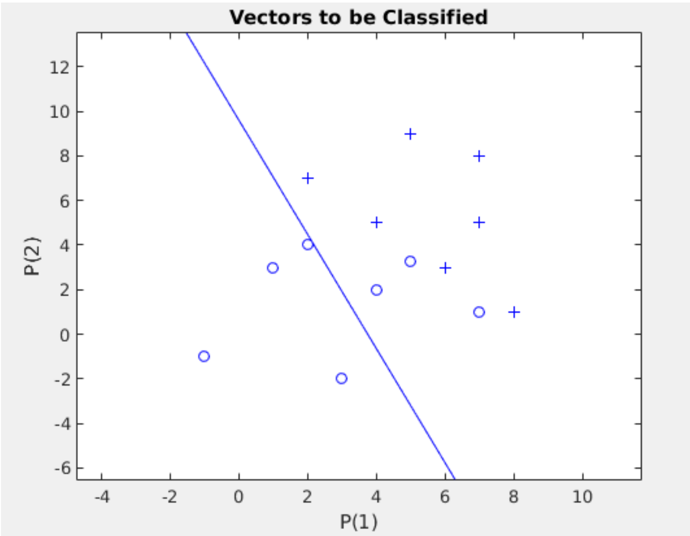
#CODE

```
function [a,b] = sample_perceptron( w1 , b, t1)

a = [0.77, 0.77];
n=14;
cnt = 0;
while(cnt<100000)

    c=0;
    for k = 1:n

        if( misclassified( w1(:,k), a, b,0, t1(k) ) )
            c= c+1;
            a = a + w1(:,k)';
            b = b - 25;
        end
    end
    if( c == 0)
        break;
    end
    cnt = cnt + 1;
end

end
```

Elapsed time is 7.321561 seconds.

Single Sample Perceptron with Margin

```
b = 0;
 w2 = -7;
 [a2, w2] = single_margin(w,w2, b , t1);
 plotpc(a2, w2);
```



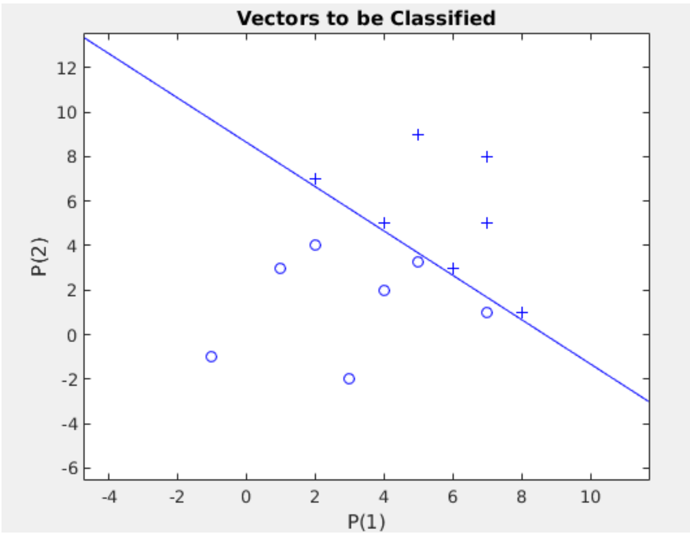

```
function [a,w0] = single_margin( w , w0, b, t )

a = [0.77, 0.77];
n=14;
cnt = 0;
lr = 1;
```

```
while(cnt<10000)

    c=0;
    for k = 1:n

        if( misclassified( w(:,k), a, w0,b, t(k) ) )
            c= c+1;
            a = a + lr*w(:,k)';
           % disp(w1(:,k) );
            w0 = w0 - 20;
        end

    end
    disp(c);
    if( c == 0)
        break;
    end
    cnt = cnt + 1;
end

end
```

Elapsed time is 1.684404 seconds.

BATCH RELAXATION WITH MARGIN

```
b3 = 0.8;
w3 = -1;
lr = 0.001;
a = [0.7 0.7];
[a3, w3] = batch_relaxation(w, w3, b3, t1, lr ,a);
plotpc(a3, w3);
```



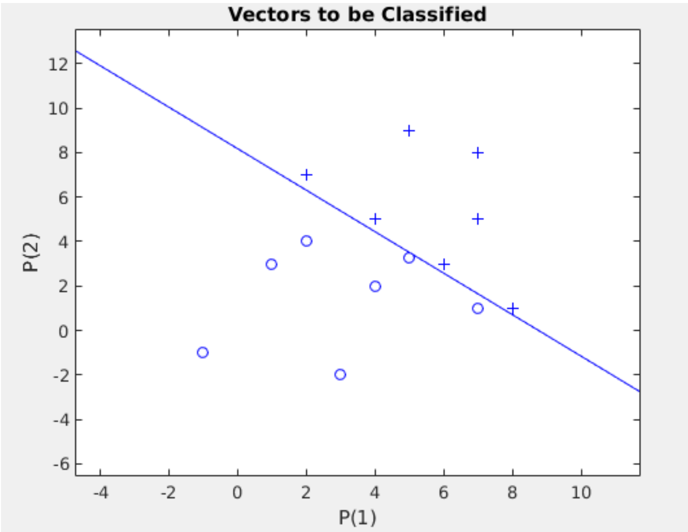

```
function [a,w0] = batch_relaxation( w , w0, b, t , lr, a)



    n=14;
    cnt = 0;
    while(cnt<10)

        c=0;
        y_sum = [
              0
```

```
            0
            ];
        for k = 1:n

            if( misclassified( w(:,k), a, w0, b, t(k) ) )
                c= c+1;
                y = w(:,k);
                factor =  ((b - a*y ) ) / (norm(y) * norm(y) );
                y_sum = y_sum + factor*y;

            end
        end

          % disp(factor);
          a = a +  lr*y_sum';
                % disp(w1(:,k) );
          w0 = w0 - 1;
        if( c == 0)
            break;
        end
        cnt = cnt + 1;
    end

end
```

Elapsed time is 0.000680 seconds.

SINGLE RELAXATION WITH MARGIN

```
b = 0.1;
w01 = -1;
a1 = [0.77 0.77];
[a1, w01 ] = relaxation_margin(w, w01, b, t1);
plotpc(a1, w01);
```



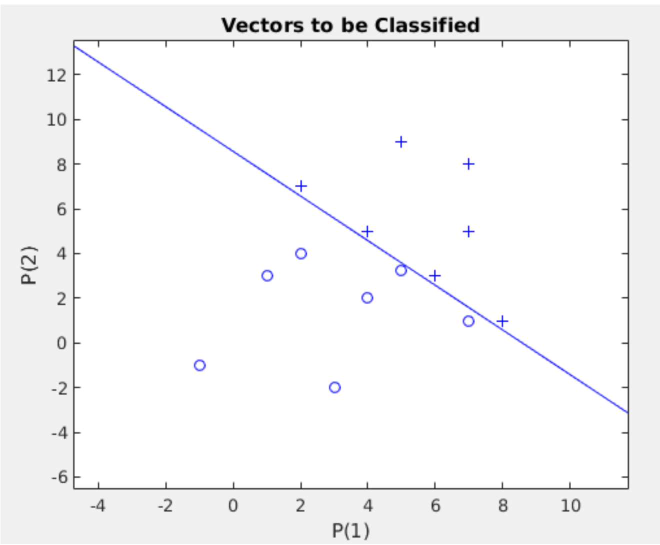

```
function [a,w0] = relaxation_margin( w , w0, b, t )

    lr = 0.001;
    a = [0.77, 0.77];
```

```
    n=14;
    cnt = 0;
    while(cnt<1000)

        c=0;
        for k = 1:n

            if( misclassified( w(:,k), a, w0, b, t(k) ) )
                c= c+1;
                y = w(:,k);
                factor = lr * ( (b - a*y )*(b - a*y ) )/ (norm(y) * norm(y) );
               % disp(factor);
                a = a +  factor*y';
                % disp(w1(:,k) );
                w0 = w0 - 1;
            end
        end
        disp(c);
        if( c == 0)
            break;
        end
        cnt = cnt + 1;
    end

end
```

Elapsed time is 0.004942 seconds.

LMS

```
theta = 0;
lr = 0.0001;
w4 = -6;
b4 = [0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1  0.1 0.1 0.1 0.1 0.1 ];
%b4 = [1 1 1 1 1 1 1 1 1 1 1 1 1 1];
a4 = [0.7 0.7];
[a4, w4] = lms_hoff(w, w4, b4, t1, a4, lr, theta);
plotpc(a4, w4);
```

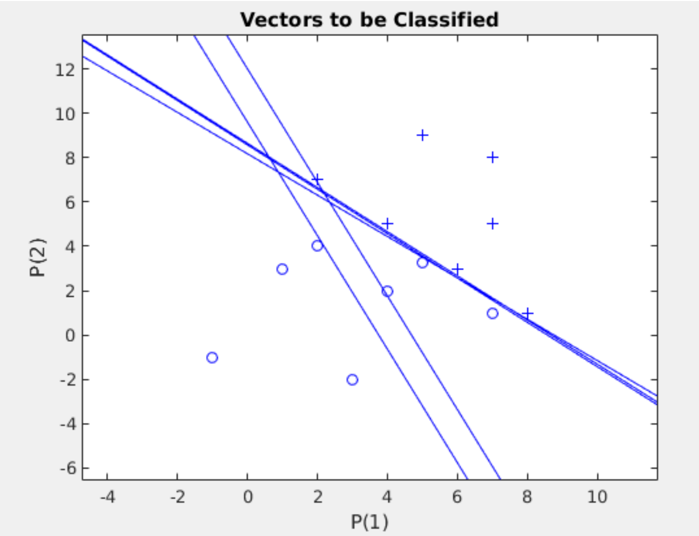



```
function [a, w0] = lms_hoff(w, w0, b, t, a, lr, theta)
```

```
n = 14;
factor = 100000000;
while(1)

    c=0;
    for k = 1:n

        y = w(:,k);
        if(misclassified(y ,a , w0, b(:,k), t(:,k) ) )
            factor = lr* (b(:,k) - a*y) ;
            a = a + factor*y';
             c = c + 1;
        end
        if(factor < theta)
            disp(factor);
            c=0;
            break;
        end
    end
    disp(c);
    if( c==0)
        break;
    end
end
```

Elapsed time is 0.001206 seconds.

Plot of all the algorithms



Problem 2:

DataSet link

Preprocessing: The dataset had 32x32 bitmaps for each digit. It has been down-sampled to 8x8 by counting number of 1s in each 4x4 grid.

Constructing the neural network:

1. number of input units = 64, output units = 2;
2. The hidden units are chosen such that total number of weights is atleast (no. Of data

points)/10;

3. eta (learning rate) : 0.001
4. The digits chosen are 0,1, 7;
5. The target values are [0 0] -> 0 ; [0 1] ->1 ; [1 0] -> 7;
6. The target values are set based on the 65th number in the training data set;
7. Initializing the weights: The weights are chosen randomly in the range -1 to 1. (they should be chosen in the range (-1/sqrt(d), -1/sqrt(d)) but that leads to very small values);

```
function w = initw(inp , out)

w = [];
for k = 1:out
    tmp = [];
    for j = 1:inp
     % tmp = [tmp, -1/sqrt(inp) ];
     r = -1 + (1 +1).*rand(1,1);
     tmp = [tmp, r ];
    end
    w = [w; tmp];
end
end
```

8. TRAINING: STOCHASTIC TRAINING
   1. feedforward

```
function [out]  = feedforward( inp , w, out_sz )
    out = [];

    for k = 1:out_sz
        netk = net(w,inp, k);
        out = [out; sigmoid(netk)];
    end
end
```

   2. backpropagation

```
function [w, bl] = backpropagation(w, kj, inp, theta)
    global eta;
    bl = 1;
    %hidden to output
    [r, c] = size(w);
    if( kj == 1)
        for k = 1:r
            netk = net(w, inp, k);
            for j = 1:c
                cond = deltak(k, netk)*inp(j,:) ;
                %disp(deltak(k, netk) );
                if( cond < theta)
                    bl = 0;
                end

                w(k,j) = w(k,j) + eta*cond;
            end
        end
    else
        for j = 1:r
            netj = net(w, inp, j);
            for i = 1:c
                cond = deltaj(j, netj)*inp(j,:);
               % if( cond < theta)
               %     bl = 0;
               % end
```

```
                w(j,i) = w(j,i) + eta*cond;
            end
        end
    end
end
```

3. Deltak

```
function dk = deltak(k, netk)
    global t;
    global z;
    global m;
    dk = (t(k,m) - z(k,:))* sigmoid_derivative(netk);

end
```

4. Deltaj

```
function dj = deltaj(j, netj)
    global wkj;
    global y;

    [r, c] = size(wkj);
    sum = 0;
    for k = 1:r
        netk = net(wkj, y, k);
        sum = sum + wkj(k,j)*deltak(k, netk);
    end

    dj = sum* sigmoid_derivative(netj);
end
```

5. finding net

```
function net_value = net(w, inp, k)
    [r,c] = size(w);
    net_value = 0;
    for j=1:c
        net_value = net_value + w(k,j)*inp(j,:);
    end

end
```

6. Stochastic training

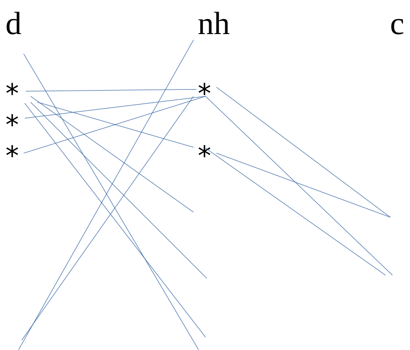
```
while(cnt <10000)
    cnt = cnt + 1;
    m = 1 + (COL - 1).*randi(1,1);
    [y] = feedforward( x(:,m), wji, nh);
    [z] = feedforward( y, wkj, c);
    [wkj, bl] = backpropagation(wkj, 1, y, theta);
    [wji, bl] = backpropagation(wji, 0, x(:,m), theta);
end
```

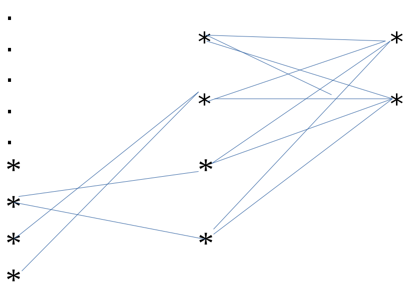
6. Testing

For testing, the ouput z matrix was rounded off to 0 or 1. Hence z matrix is compared with target matrix to validate;

Report

Case 1:
d = 64, nh = 6 , c = 2;





Every hidden layer unit is connected to every output unit and every input unit to every hidden unit.

CLASS 0: (91 images for digit 0)

Input to hidden weights

Columns 1 through 13

-0.9397	0.5219	0.7924	0.2377	0.4415	-0.0723	0.5178	-0.9857	-0.6130	-0.8171	-0.6641	0.8845	0.6989
-0.2013	0.9601	0.0867	0.8259	-0.3505	-0.4178	0.7591	0.5594	-0.2100	0.7658	-0.1475	-0.0661	-0.1490
0.6605	0.4304	0.7643	-0.6853	-0.6085	0.7530	-0.0870	-0.6369	0.4388	0.7561	0.5851	0.6750	0.0651
-0.1838	0.1952	-0.1486	-0.7564	-0.3001	0.0934	-0.7459	-0.8371	-0.4211	0.3536	-0.0860	-0.0016	-0.8831
0.4136	0.8605	0.6424	-0.9188	0.9339	0.7774	0.2791	-0.9766	0.9560	-0.2274	-0.1992	0.0221	0.1048
-0.5319	-0.7267	-0.3475	-0.3111	0.8535	0.7727	0.5761	-0.8550	-0.1644	-0.2105	-0.1781	0.9439	-0.5282

Columns 14 through 26

-0.7254	0.3215	-0.9230	-0.7037	-0.6305	0.2907	0.7131	0.7068	-0.0119	0.5618	-0.4507	0.9348	0.9811
-0.0884	0.3920	-0.8077	0.4681	-0.7388	-0.0058	-0.4795	0.1845	-0.5436	0.9114	-0.8728	-0.6323	0.2496
-0.5918	0.0499	0.6756	-0.5803	0.4225	0.7144	-0.0684	-0.0900	0.8741	0.2457	0.6018	-0.2079	-0.3682
-0.8528	-0.3243	-0.3279	0.9699	0.5033	0.2930	0.7657	0.0237	-0.2477	0.7675	-0.5085	-0.6720	0.3172
0.9193	0.7123	-0.0451	0.7571	-0.2187	-0.9002	-0.3836	-0.3581	-0.0759	-0.0222	0.4150	-0.2639	0.8844
-0.7868	0.5961	-0.9143	0.9159	0.5993	0.9172	0.5275	0.2125	0.9741	0.1211	-0.0122	-0.4979	-0.6120

Columns 27 through 39

-0.8311	-0.3404	-0.4634	-0.5682	0.5340	-0.2699	-0.6920	-0.6700	0.7867	0.8916	0.0349	-0.9633	-0.1614
-0.4160	-0.7245	-0.6210	0.9144	-0.4056	-0.5244	0.3215	-0.9903	-0.8425	-0.3797	-0.5509	0.3917	0.4016
-0.2534	0.4270	-0.6248	-0.0310	-0.7694	-0.7311	0.8834	0.2509	-0.1337	0.4046	0.0483	0.2939	0.5454
-0.4888	-0.6848	-0.4468	-0.4026	-0.9882	-0.5428	0.3402	0.9633	-0.4878	0.2383	0.9436	-0.0157	-0.3980
-0.4621	0.4923	0.8197	0.1706	0.8145	-0.8851	0.1150	-0.9926	-0.6922	-0.1835	-0.5316	0.0721	-0.3753
-0.0185	0.4178	0.9713	-0.3347	-0.5636	-0.3134	-0.8208	0.2592	-0.2476	0.7708	-0.1509	0.7776	-0.5555

Columns 40 through 52

-0.7558	0.9037	-0.4596	0.7873	0.7904	0.5511	0.8176	-0.9666	0.9186	-0.8495	-0.4297	-0.0238	0.9411
0.7048	0.1193	0.1418	0.0630	0.7189	-0.1062	-0.2321	0.1919	-0.3888	-0.7966	-0.2550	-0.5545	0.2876
0.2251	-0.0068	-0.0402	-0.7907	0.8221	-0.1457	-0.1570	-0.9524	0.0099	0.9164	-0.4603	0.4759	-0.7774
0.1941	0.0867	-0.7405	0.1044	-0.7019	-0.4830	0.0574	0.6267	0.4370	0.0016	0.7879	0.9014	-0.0753
0.4188	0.1756	-0.5201	-0.8704	-0.9918	-0.6203	-0.0161	0.2344	-0.7529	-0.7324	-0.3701	-0.8228	-0.1809
0.5899	0.7126	-0.1498	0.1737	-0.8206	-0.5028	0.1836	0.0372	0.8241	0.3226	-0.0047	0.2656	-0.8313

Columns 53 through 64

0.1362	-0.4521	-0.7418	-0.6896	-0.4624	-0.1322	-0.2082	-0.1495	0.2288	-0.1064	-0.9760	-0.9315
0.5545	-0.1388	0.2433	-0.4407	-0.0427	-0.3694	0.6555	0.4875	-0.7102	-0.8623	0.0654	0.4891
0.5277	0.5205	0.0706	-0.6723	0.8493	-0.5945	-0.9714	0.1923	0.2038	-0.0733	0.1817	0.4360
-0.4249	0.8527	-0.0074	0.7546	-0.5736	-0.7713	-0.7514	0.1481	-0.0673	-0.2520	0.5092	-0.0889
-0.4339	0.2627	-0.5590	0.3100	-0.6116	-0.8093	0.8382	0.1180	-0.1632	-0.7776	0.1070	-0.7759
-0.9224	0.4992	-0.3565	-0.8710	0.8425	-0.8953	-0.8681	0.1431	-0.1256	-0.9492	-0.9046	-0.5269

output to hidden weights

-0.5439 0.2347 -0.6173 -0.5166 0.4857 -0.7042

0.5425	-0.8532	0.8644	0.3062	-0.4345	0.6756

CLASS 1: (102 images for digit 1)

Input to hidden weights

Columns 1 through 13

0.9626	-0.6743	-0.6868	-0.6727	-0.8430	0.4368	-0.1075	0.0202	0.2103	0.7661	-0.7562	-0.4195	-0.8753
-0.6774	-0.2933	-0.8840	-0.1118	-0.1179	-0.3466	0.8288	-0.0109	0.4204	0.3292	0.6635	-0.4840	0.5384
0.2193	0.9354	0.2681	0.5544	0.7969	-0.5470	-0.7976	0.6564	-0.1512	0.4541	0.8734	-0.4071	-0.3757
0.5648	-0.6381	0.8844	0.5207	0.6134	-0.1676	-0.4193	0.3408	0.6973	0.7238	0.7744	0.6674	-0.0494
-0.5153	-0.6631	0.7222	-0.3467	-0.3497	-0.1997	0.5270	-0.6719	0.3766	0.2866	0.6427	-0.2429	-0.7762
0.3309	-0.5624	0.8349	-0.2318	-0.9223	-0.5376	0.8215	0.7457	0.1727	0.1293	-0.9901	-0.7385	-0.5630

Columns 14 through 26

-0.7129	-0.8641	-0.6298	0.5374	0.7954	0.0365	0.6893	-0.5425	-0.1433	-0.2634	0.3921	0.8888	-0.5642
-0.1789	-0.8887	0.7413	0.9446	-0.7661	0.0166	0.5111	-0.7801	-0.9323	-0.1179	-0.7405	0.9432	0.8803
0.0993	0.1050	0.2807	0.1932	-0.6778	-0.1643	-0.2620	-0.8396	-0.5275	0.9885	0.8185	0.5814	-0.7298
-0.0427	-0.8891	-0.1058	0.1714	-0.7996	-0.3322	-0.4916	0.1406	-0.1490	-0.1895	-0.9113	-0.5004	-0.5314
0.7190	0.4264	0.1500	0.4823	0.5073	0.1102	0.6018	0.3407	0.0702	0.3719	0.9104	0.2555	0.6383
0.6614	-0.2170	-0.3561	-0.4687	-0.3992	-0.1359	-0.6074	0.1427	0.3231	0.3789	-0.2848	0.9213	-0.5048

Columns 27 through 39

0.5060	0.0397	0.5201	0.7617	-0.5250	0.8794	-0.9457	0.1230	0.7809	-0.6660	-0.1495	-0.3462	0.1352
0.8662	0.9675	0.3860	0.4049	-0.2574	-0.8718	-0.1131	0.0859	0.0803	0.5725	0.2028	0.8940	0.2624
0.1005	-0.8925	0.3530	0.6444	0.8227	-0.7057	-0.3904	0.5596	-0.0257	0.0982	0.7563	0.9765	-0.9618
0.3984	0.2980	-0.7814	-0.7613	-0.8638	0.4507	0.0926	0.6490	-0.8075	-0.5066	-0.9115	0.2934	-0.4649
-0.6221	0.5030	-0.5568	-0.1671	0.5817	-0.2001	-0.9098	-0.5979	-0.9143	0.2783	-0.4965	-0.1602	-0.6563
0.5043	-0.2118	0.6973	-0.2483	0.4230	-0.3009	0.7621	0.5088	-0.1234	-0.1909	-0.2464	-0.1057	-0.2316

Columns 40 through 52

0.6540	-0.2166	0.2427	0.9303	0.6569	0.6268	0.9102	-0.6338	0.3409	0.4712	0.6391	0.3620	-0.3399
0.7576	-0.7547	0.6271	0.9552	-0.5376	0.8254	0.3527	0.4501	0.3715	0.5934	0.0592	0.5092	0.0071
-0.2975	0.0692	0.4955	0.9399	0.3902	-0.4536	0.9021	0.8152	-0.0853	0.4662	0.1163	0.9877	-0.2215
0.3985	-0.7592	0.0282	-0.9280	-0.8171	-0.0819	0.4015	0.1829	-0.6099	0.2178	0.3786	-0.3901	0.5898
0.7689	-0.3180	-0.5898	0.5908	-0.9511	-0.9204	0.0596	0.8667	0.2044	0.2024	0.2877	0.8439	-0.5615
-0.8468	-0.1117	0.2208	-0.3827	-0.1790	-0.7902	-0.0385	-0.4074	-0.9597	0.7766	-0.5716	0.3750	-0.6835

Columns 53 through 64

-0.9306	-0.7596	-0.4326	0.6765	0.8027	-0.1001	0.9771	-0.9324	0.7189	0.0571	0.9509	0.7544
0.8140	-0.7528	0.7044	-0.6500	0.5283	-0.0814	0.2121	-0.2720	-0.1962	0.4601	0.9102	0.7576
0.7632	0.7864	-0.2281	0.4550	0.8252	0.8081	-0.8372	-0.9334	0.1447	-0.9816	-0.6035	-0.9588
-0.6637	-0.5371	0.1697	0.7167	0.6176	0.1604	-1.0152	0.8693	0.5652	-0.2211	0.7272	0.3290
0.7698	-0.9024	0.6241	-0.9348	-0.1699	0.1893	-0.6064	0.5922	-0.5780	-0.6512	0.2277	0.8821
0.9564	0.0131	0.7906	0.4607	0.0021	0.2222	-0.2937	0.0602	0.6016	-0.4449	-0.4069	-0.2959

Output to hidden weights

0.9082	-0.0703	0.7096	0.2599	-0.6980	-0.8975
-0.9553	-0.4668	0.1747	-0.6747	-0.8487	0.1117

CLASS7: (98 images for digit 7)

hidden to input

Columns 1 through 13

-0.0625	-0.5132	0.8398	0.4317	-0.0227	-0.8439	0.0601	0.3321	0.2154	-0.1609	-0.7784	0.6332	-0.5014
0.5461	-0.2151	0.2105	-0.5052	-0.4196	-0.9614	-0.3054	-0.7164	-0.1770	-0.6939	0.6580	0.4784	-0.8023
0.5981	0.8058	-0.3750	-0.4368	-0.9864	-0.0083	0.9770	0.4759	-0.3786	0.2008	0.5633	-0.7769	0.1586
0.9916	-0.1268	-0.3911	-0.5070	0.9217	-0.5542	-0.2088	-0.5510	-0.4600	-0.1631	0.9955	0.8221	0.1009
-0.1772	-0.6432	0.9864	0.0359	0.7657	0.2916	-0.0724	-0.8143	0.9308	0.2353	-0.6929	-0.2081	0.7601
-0.1765	0.4263	-0.7981	0.6210	0.2767	0.7972	0.2447	-0.1697	0.2962	-0.0204	-0.8114	0.2757	0.9016

Columns 14 through 26

0.0184	0.6083	0.0197	-0.3946	0.7239	-0.8852	0.4615	0.8018	-0.0719	-0.0613	-0.6924	0.9236	0.7525
0.6412	-0.5458	-0.7862	0.3255	0.9091	0.6291	0.2465	-0.3435	-0.4453	-0.1313	-0.2994	0.7555	-0.9877
0.7407	0.3795	-0.5141	-0.3146	0.0909	-0.8649	-0.1791	-0.5250	-0.0221	0.6121	-0.2443	0.0359	-0.8108
0.1927	-0.8417	0.1533	0.7963	-0.0734	-0.2032	-0.7911	0.3045	0.9834	0.3561	-0.1430	0.3096	0.1775
0.0731	0.0825	-0.1200	-0.6374	-0.5064	-0.4347	0.0541	0.1148	-0.1746	0.8065	-0.3946	0.2947	-0.4273
-0.0462	0.2067	0.1840	-0.5484	0.3379	-0.6858	0.5496	-0.5728	-0.6607	0.4526	-0.4866	-0.6734	0.2526

Columns 27 through 39

-0.0227	-0.1858	-0.7468	0.8509	-0.9888	-0.6272	-0.3519	-0.8996	-0.7109	0.4587	-0.0354	-0.3239	-0.5265
0.3929	-0.3239	-0.3901	0.2962	0.8423	0.7861	0.9943	-0.8543	-0.7408	0.9631	-0.8196	0.3723	0.8579
0.8182	-0.5848	-0.2359	0.3205	0.5167	-0.6539	0.0347	0.9907	0.4152	-0.8389	-0.9134	-0.0177	-0.1068
0.4901	0.2817	0.0074	0.8761	0.2107	0.2778	0.4053	0.7219	-0.2407	0.4242	0.0470	-0.2730	-0.1306
-0.4944	-0.2885	0.6070	0.0730	0.1752	-0.5426	-0.8010	0.2844	0.9554	-0.8014	-0.6291	-0.9861	0.6717
-0.4974	-0.4730	0.6889	-0.2041	-0.7898	-0.6112	-0.2728	0.7501	0.2006	-0.4827	-0.2821	0.7761	0.8021

Columns 40 through 52

-0.0983	-0.6291	-0.3515	-0.4721	0.6602	0.3927	-0.3329	0.1605	-0.4243	-0.4721	-0.4802	0.3542	0.0397
-0.7164	0.7688	-0.9605	-0.3147	-0.5234	0.9691	0.6932	0.5890	0.8005	0.5602	0.6730	-0.3578	0.4855
-0.0264	-0.6682	-0.2787	0.7614	0.4887	-0.1665	0.8147	-0.8114	-0.6374	0.8932	-0.7983	-0.2239	-0.4216
0.3752	-0.5463	0.9580	0.9513	-0.4210	-0.3231	0.9929	0.5779	0.5898	0.2647	0.6229	-0.1038	0.6612
-0.0422	-0.0678	0.8122	-0.1706	0.0834	0.3931	-0.8468	0.0140	-0.0311	0.7069	0.2239	-0.7701	0.2089
-0.1030	-0.4612	0.1087	-0.6412	0.7205	-0.5350	-0.6627	-0.9455	-0.3541	0.1114	0.6501	0.6096	-0.9501

Columns 53 through 64

-0.8465	-0.8883	-0.4825	-0.1201	-0.4314	0.3575	0.8992	0.5479	0.2723	0.5072	0.4936	0.1721
0.3290	-0.4217	-0.3253	0.8172	-0.9355	0.3927	-0.5825	0.6483	-0.5636	-0.8008	0.2390	-0.7924
-0.8538	-0.6108	-0.1650	-0.4142	0.4043	-0.5206	0.9190	-0.3891	-0.6902	0.1110	0.5811	-0.1123
-0.7467	0.0265	0.4319	-0.5037	0.0638	-0.2356	0.6035	0.3417	0.9657	0.8736	0.1525	-0.8396
-0.6765	-0.2068	0.0598	0.7455	0.3132	0.5704	-0.7528	0.0451	0.7941	-0.7867	0.2643	-0.8432
-0.2559	-0.0152	-0.0668	-0.9155	0.2351	0.1571	-0.4013	-0.1276	-0.7257	-0.3995	0.5238	-0.9283

output to hidden

0.8127	-0.7464	-0.5866	-0.2651	-0.2519	0.4581
0.7450	-0.9081	-0.8429	-1.0123	-0.6277	-0.9494

Accuracy:

On testing, the network correctly classifies 60% of the data.