INDEPENDENT COMPONENT ANALYSIS LAB 10 VS265 - SOLUTION

MAYUR MUDIGONDA

1. Code

1.1. **Question 1.**

```
% ica.m - runs the ICA algorithm
%
% you must have previously defined a data matrix X and a
% basis function matrix A
load test_data;
A = rand(64);
A = A*diag(1./sqrt(sum(A.*A)));
[N K]=size(X);
M=size(A,2);

% learning rate - you will need to adjust this
eta=0.01;

distrib =2; %0 for Laplacian, 1 for Cauchy, 2 for Gaussian

h=showbfs(A);

t=0;
while (1)

    % compute S and Z
    S=inv(A)*X;
    if distrib==0
        Z = sign(S); %Laplacian Prior
    elseif distrib==1
        Z = 2*S./(1+ S.*S); %Cauchy Prior
    else
        Z = S; %Gaussian
    end
    % collect stats
    ZS_ave=Z*S'/K;
```

```
    % update basis functions
    % dA=...
    dA = eta*(A*ZS_ave - A);
    A=A+dA;

    showbfs(A,'black',h)
    t=t+1;
    fprintf('\rtrial %3d',t)

end
```

1.2. **Problem 2.**

```
%% 2: ICA on natural images

load IMAGES.mat
num_images=size(IMAGES,3);
A=randn(64);
A=A*diag(1./sqrt(sum(A.*A)));
[N,M]=size(A);
% patch size
sz=sqrt(N);
% batch size
K=10000;
% learning rate - you will need to adjust this
eta=0.01;
distrib=2;
h=showbfs(A);
t=0;
while (1)

    % choose and image at random and extract image patches
    imi=ceil(num_images*rand);
    X=extract_patches(IMAGES(:,:,imi),sz,K);

    for i=1:1000
        % compute S and Z
        S=inv(A)*X;
        if distrib==0
            Z=sign(S);
        elseif distrib==1
            Z=2*S./(1+S.*S);
        else
            Z=S;
```

```
        end

        % collect stats
        ZS_ave=Z*S'/K;

        % update basis functions
        dA=eta*(A*ZS_ave-A);
        A=A+dA;

        t=t+1;
    end;
    showbfs(A,'black',h)
    fprintf('\rtrial %3d',t)
end
```

2. Test data

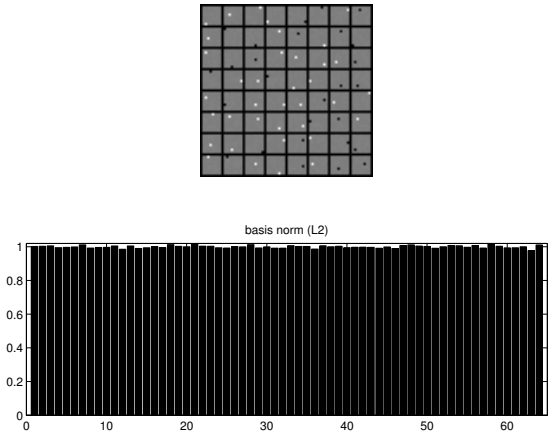

FIGURE 1. Identity matrices unto a permutation learnt on the test data provided using a Laplacian prior

3. Natural Scenes

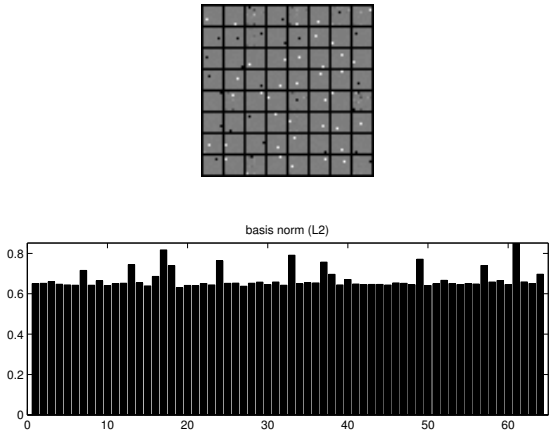

FIGURE 2. Identity matrices unto a permutation learnt on the test data provided using a Cauchy prior. Note how this prior also learns effectively the same codebook that the Laplacian prior does

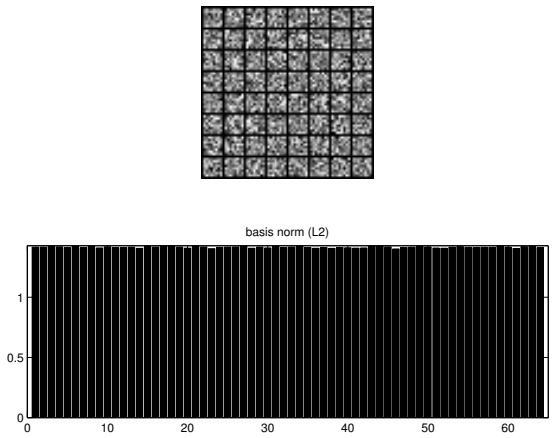

FIGURE 3. Code books learnt on the test data provided using a Gaussian prior. This prior does not learn a meaningful codebook and ends up with fuzzy codebook elements because the prior is too smooth over too large a probability space to find the components (codebooks) that we care about

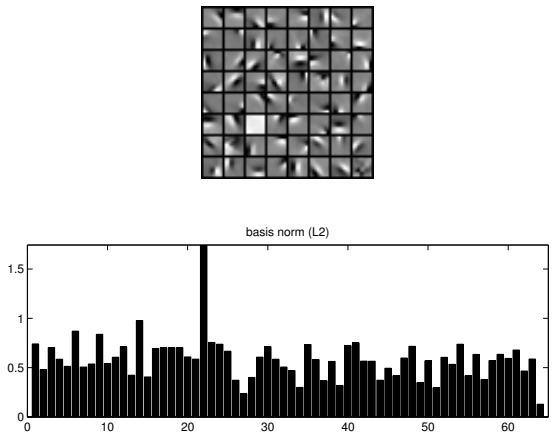

FIGURE 4. Codebooks learnt on natural scene patches using a Laplacian prior. Note how the codebooks resemble Gabor like wavelets.

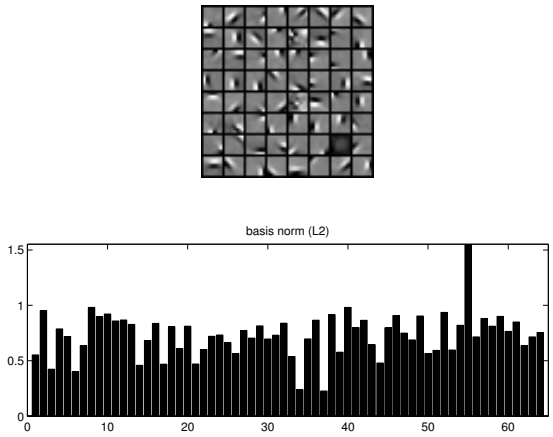

FIGURE 5. Codebooks learnt on natural scene patches using a Cauchy prior. Note how this prior also learns Gabor like wavelets.

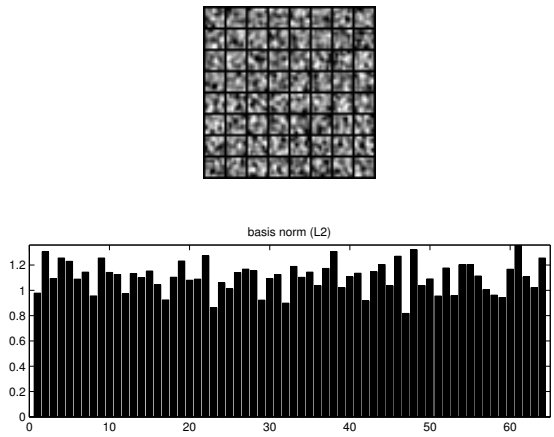

FIGURE 6. Code books learnt on natural scene patches using a Gaussian prior. This prior again fails to learn meaningful codebook elements. Even after 20000 iterations did not converge to stable dictionary elements