Table of Contents

Homework 5, ECE283, Morten Lie.

Collaboration with Sondre Kongsgård, Brage Sæther, Anders Vagle

```
clear all;
close all;
clc;
addpath('functions');
```

Parameters

```
n_comps = 3;
n_u = 6;
N = 200;
d = 100;
u = zeros(d, 6);
sigma2 = 0.01;
K_max = 5;
n_rand_inits = 5;
```

Problem 1:

All eigenvalues that were greater than the mean were given the property as a dominant eigenvalue. By trying different values for N, it seems like the number of dominant eigenvalues increase with increasing N, but plateaus at 6.

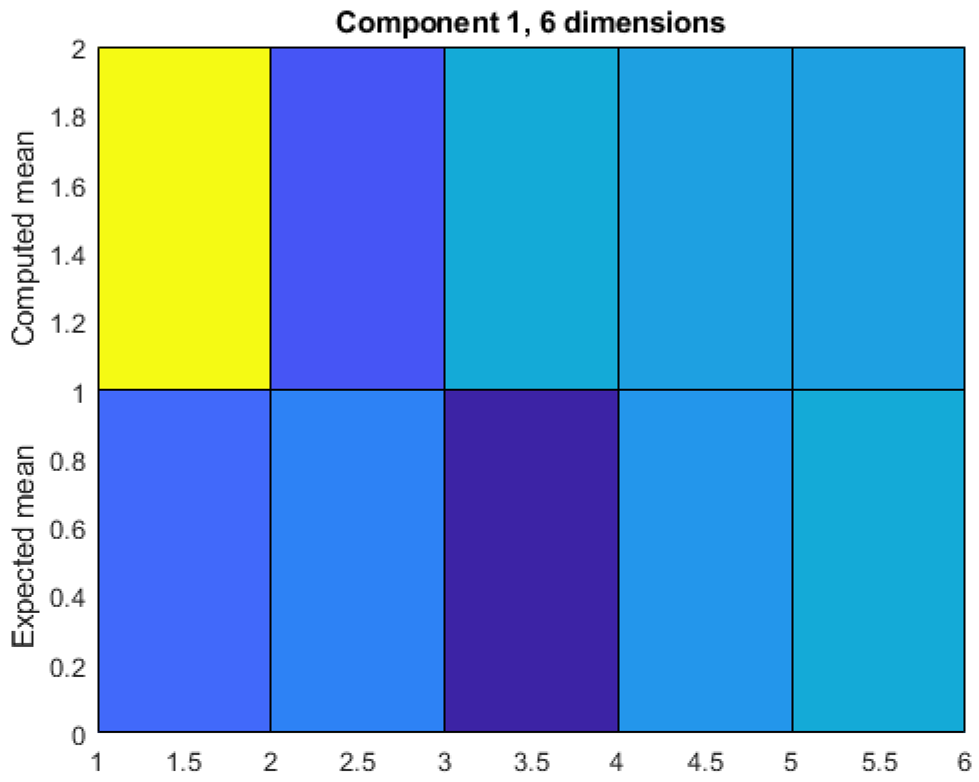
```
% Generate data
for j = 1:n_u
    u(:,j) = generate_random_vector(d);
    while check_orthogonality(u,j) == 0
        u(:,j) = generate_random_vector(d);
    end
end
[X_d, Z_d] = generate_sample_data(u,sigma2,N);

% Single value decomposision, finding dominant singular values
[U,S,V] = svd(X_d);
eig_mean = eigenvalue_mean(S);
for d0 = 0:d
```

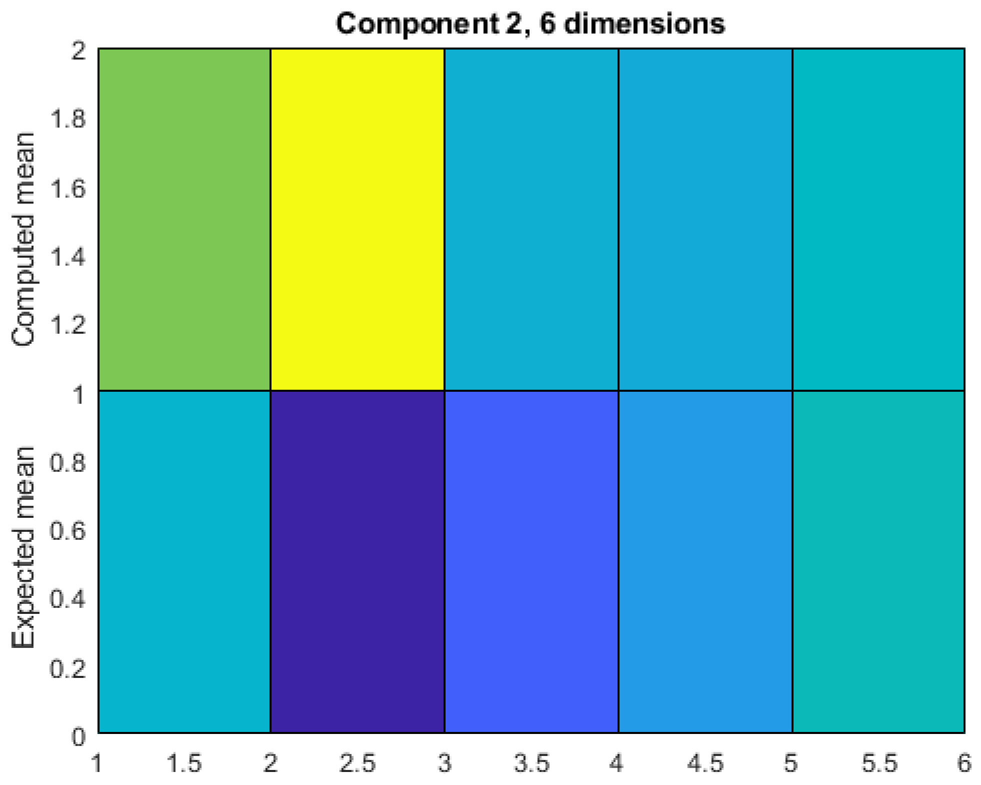
```
    if S(d0+1,d0+1) < eig_mean
        break
    end
end
fprintf(['Number of dominant singular values: ' num2str(d0) '\n']);

% PCA
S_r = zeros(d0,d0);
for i = 1:d0
    S_r(i,i) = S(i,i);
end
U_r = U(:,1:d0);
V_r = V(:,1:d0);
X_r = U_r*S_r*V_r';
X_d0 = X_r*V_r;

% d-dimensional k-means
m_opt_d0 = zeros(K_max,size(X_d0,2),K_max);
C_opt_d0 = zeros(N,K_max);
for K = 2:K_max
    min_sme = inf;
    for i = 1:n_rand_inits
        C_d0 = randi(K,N,1);
        [m_d0,C_d0] = k_means(N,K,C_d0,X_d0);
        sme = SME(m_d0,X_d0,C_d0);
        if sme < min_sme
            m_opt_d0(1:K,:,K) = m_d0;
            C_opt_d0(:,K) = C_d0;
            min_sme = sme;
        end
    end
end

% One-hot encoding
a = zeros(N,K_max,K_max);
for K = 2:K_max
    for i = 1:N
        a(i,C_opt_d0(i,K),K) = 1;
    end
end

% Generate empirical probability table for d-dimensional data
pk_kmeans_d = zeros(3,K_max,K_max);
for K = 2:K_max
    for l = 1:3
        for k = 1:K
            num_k_l = 0;
            num_l = 0;
            for i = 1:N
                if Z_d(i,l) == 1
                    num_l = num_l + 1;
                    if a(i,k,K) == 1
                        num_k_l = num_k_l + 1;
                    end
                end
            end
            pk_kmeans_d(l,k,K) = num_k_l/num_l;
        end
    end
end
plot_table(pk_kmeans_d,K_max,'d-dimensional K-means');

Number of dominant singular values: 6
Plotting table of P(k|i) generated from d-dimensional K-means with K =
 2

ans =

    0.0161    0.9839
    0.5694    0.4306
    0.6970    0.3030

Plotting table of P(k|i) generated from d-dimensional K-means with K =
 3

ans =

    0.9839         0    0.0161
    0.3194    0.6806         0
    0.1667    0.4545    0.3788

Plotting table of P(k|i) generated from d-dimensional K-means with K =
 4

ans =

         0    1.0000         0         0
         0    0.2083    0.5139    0.2778
    0.3636         0    0.2576    0.3788

Plotting table of P(k|i) generated from d-dimensional K-means with K =
 5

ans =

    0.4839         0    0.5161         0         0
         0    0.4583    0.2083         0    0.3333
         0    0.5455    0.0152    0.4394         0
```

Problem 2:

In order to achieve some geometric insight, we look at the expected mean value versus the calculated mean value given my the k-means algorithm. When calculating the expected mean value, we take the mean value of the component expression, which turnes out to only be the first expression as the other expressions are parts of a zero mean distribution. The plot you see is a colorvalue comparison between the two. As one may see in the figure, the expected mean values and calculated mean values are very similar.

```
% Project u vectors down to d0 dimensions
[U,S,V] = svd(u');
for j = 1:d0
    S_r(j,j) = S(j,j);
end
U_r = U(:,1:d0);
V_r = V(:,1:d0);
X_r = U_r*S_r*V_r';
u_d0 = X_r*V_r;

plot_mean_comp(pk_kmeans_d,2,u_d0,m_opt_d0(1:2,:,2));
```





Problem 3 & Problem 4

After trying different values for m, I got that m=10 gave the best reconstruction.

```
% Random Projections and Compressed Sensing
m = 10; %Based on experiments
phi = zeros(m,d);
for i = 1:m
    for j = 1:d
        if rand < 0.5
            phi(i,j) = -1;
        else
            phi(i,j) = 1;
        end
    end
end

n_draws = 10;
lambdas = 0:0.0005:0.01;
noise = zeros(1,d);
norm_MSE = zeros(n_comps,length(lambdas));
for draw = 1:n_draws
    [S, Z] = generate_sample_data(u,0,N); %Sigma = 0 to remove noise
 component implemented in function
    noise_matrix = normrnd(0,sigma2*eye(d));
    for j = 1:d
```

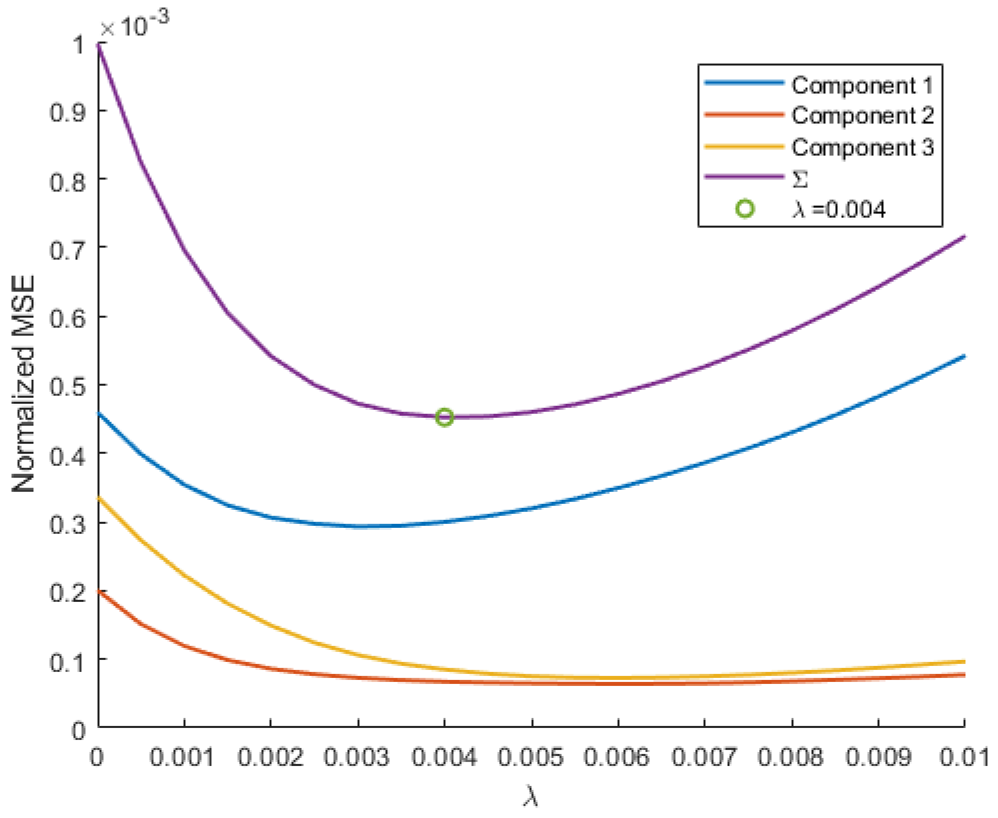
```
        noise(j) = noise_matrix(j,j);
    end
    X = S + noise;
    y = (1/sqrt(m)*phi*X')';
    B = [u(:,1) u(:,2) u(:,3) u(:,4) u(:,5) u(:,6)];
    lasso_mat = 1/sqrt(m)*phi*B;
    a_hat = lasso_func(lasso_mat,y',lambdas);

    for l = 1:length(lambdas)
        S_hat = (B*a_hat(:,:,l)')';
        for i = 1:N
            for comp = 1:n_comps
                if Z(i,comp) == 1
                    norm_MSE(comp,l) = norm_MSE(comp,l) +
 immse(S(i,:),S_hat(i,:))/sum(sum(S(i,:).^2));
                end
            end
        end
    end
end
norm_MSE = norm_MSE./n_draws;
[~,I] = min(sum(norm_MSE));
lambda_opt = lambdas(I);

fprintf(['Best lambda found was lambda = ' num2str(lambda_opt) '\n']);
```

```
Best lambda found was lambda = 0.004
```

Problem 5:

When trying different lambdas, I first tried with values from 0,001 to 10 with increments of multiples of 10, and then for further inspection, I tried values between 0.005 and 0.01 and took the argmin of the sum of the normalized MSE, summed over the classes. See plot for more details

```
% Plotting the MSE curves
figure()
hold on
for comp = 1:n_comps
    plot(lambdas,norm_MSE(comp,:),'Linewidth',1.5);
end
plot(lambdas, sum(norm_MSE),'Linewidth',1.5);
scatter(lambda_opt,sum(norm_MSE(:,I)),'Linewidth',1.5);
hold off
legend('Component 1', 'Component 2', 'Component 3', '\Sigma',
 ['\lambda =' num2str(lambda_opt)]);
xlabel('\lambda')
ylabel('Normalized MSE');
```



Problem 6:

Comparing the eulidean distances squared of the u vectors and the u vectors projected, we see that the values are very similar. Look below for implementation and difference value.

```
% Compare projected data and original data with euclidean distances
v = (1/sqrt(m)*phi*u);
D_orig = 0;
D_proj = 0;
for i = 1:n_u
    for j = 1:n_u
        D_orig = D_orig + sqrt(sum((u(:,i) - u(:,j)).^2));
        D_proj = D_proj + sqrt(sum((v(:,i) - v(:,j)).^2));
    end
end
fprintf(['Euclidean distance of original u: ' num2str(D_orig) '\n']);
fprintf(['Euclidean distance in projected space: '
 num2str(D_proj) '\n']);
fprintf(['Difference of euclidean distances: ' num2str(abs(D_orig-
D_proj)) '\n']);
```

```
Euclidean distance of original u: 232.6954
Euclidean distance in projected space: 197.9174
Difference of euclidean distances: 34.778
```

Problem 7

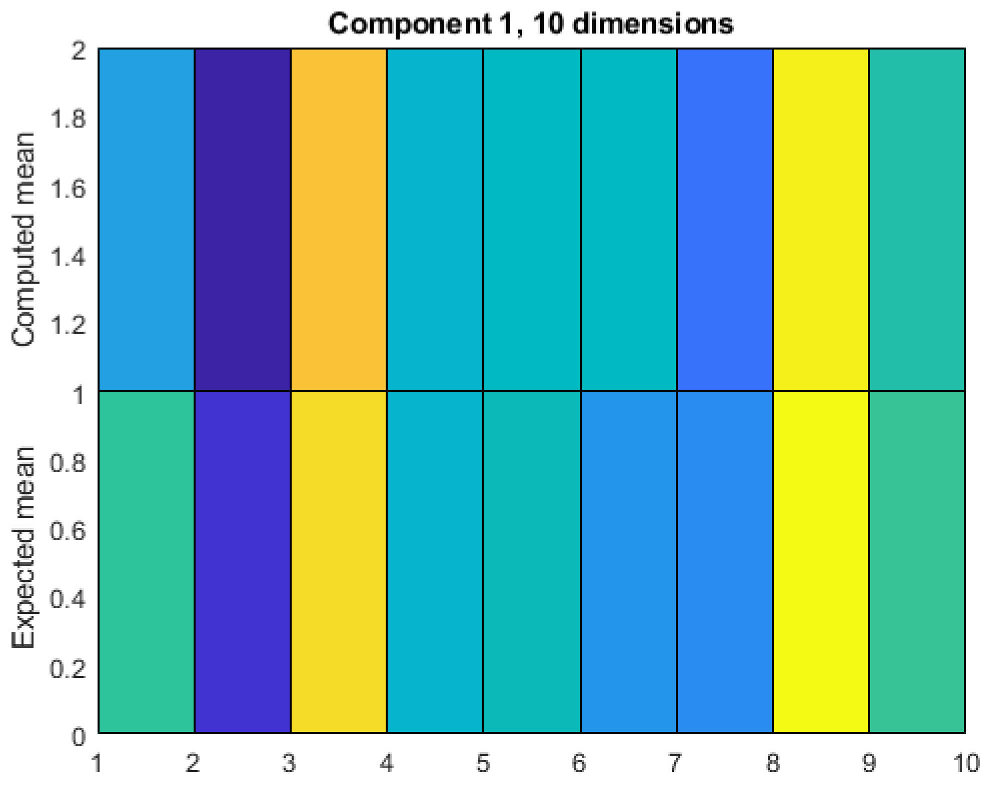
```
% m-dimensional k-means
m_opt_m = zeros(K_max,size(y,2),K_max);
C_opt_m = zeros(N,K_max);
for K = 2:K_max
    min_sme = inf;
    for i = 1:n_rand_inits
        C_m = randi(K,N,1);
        [m_m,C_m] = k_means(N,K,C_m,y);
        sme = SME(m_m,y,C_m);
        if sme < min_sme
            m_opt_m(1:K,:,K) = m_m;
            C_opt_m(:,K) = C_m;
            min_sme = sme;
        end
    end
end

% One-hot encoding
a = zeros(N,K_max,K_max);
for K = 2:K_max
    for i = 1:N
        a(i,C_opt_m(i,K),K) = 1;
    end
end

% Generate empirical probability table for m-dimensional data
pk_kmeans_m = zeros(3,K_max,K_max);
for K = 2:K_max
    for l = 1:3
        for k = 1:K
            num_k_l = 0;
            num_l = 0;
            for i = 1:N
                if Z(i,l) == 1
                    num_l = num_l + 1;
                    if a(i,k,K) == 1
                        num_k_l = num_k_l + 1;
                    end
                end
            end
            pk_kmeans_m(l,k,K) = num_k_l/num_l;
        end
    end
end
plot_table(pk_kmeans_m,K_max,'m-dimensional K-means');
```

```
Plotting table of P(k|i) generated from m-dimensional K-means with K =
 2

ans =
    0.7097    0.2903
    0.1642    0.8358
    0.9014    0.0986

Plotting table of P(k|i) generated from m-dimensional K-means with K =
 3

ans =

    0.2742    0.4194    0.3065
    0.6119         0    0.3881
    0.0563    0.0141    0.9296

Plotting table of P(k|i) generated from m-dimensional K-means with K =
 4

ans =

    0.4194         0    0.5484    0.0323
         0    0.3582    0.0448    0.5970
         0    0.7183    0.2817         0

Plotting table of P(k|i) generated from m-dimensional K-means with K =
 5

ans =

    0.1935    0.4194         0    0.3871         0
    0.1194         0    0.4328    0.1343    0.3134
    0.9155    0.0141         0    0.0704         0
```

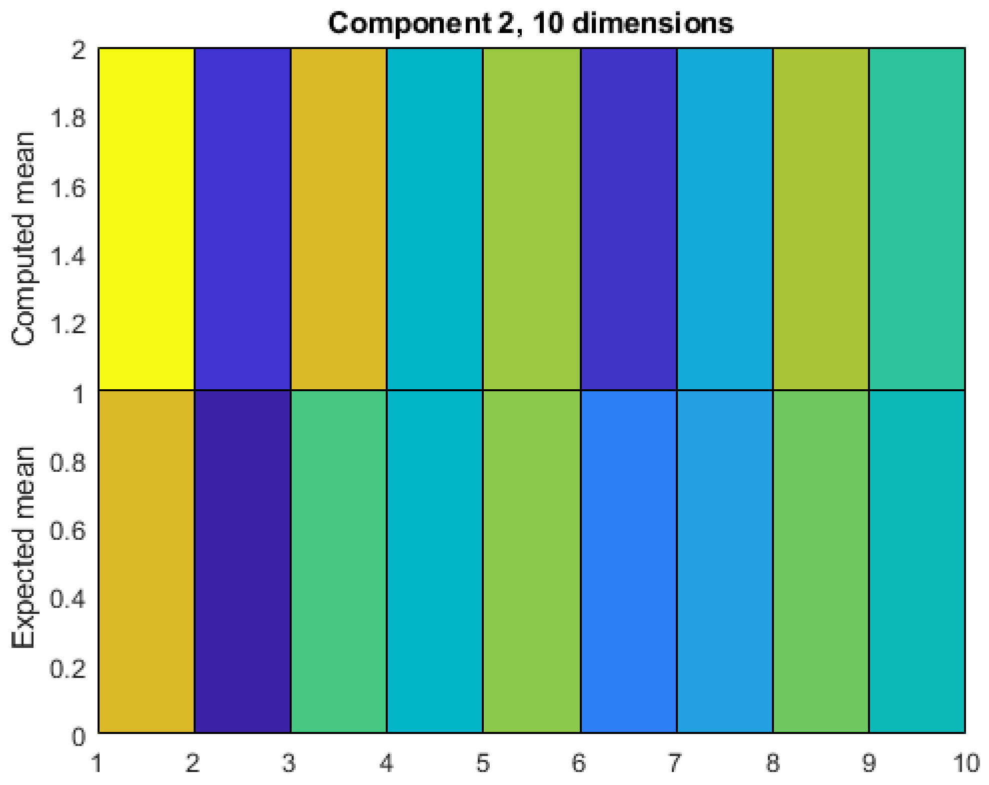
Problem 8:

Done in the same way as Problem 2. As one may see in the figure, the expected mean values and calculated mean values are very similar.

```
% Geometric insight with 2 clusters
plot_mean_comp(pk_kmeans_m,2,v',m_opt_m(1:2,:,2));
```



Component 1, 10 dimensions
Computed mean
Expected mean
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
2
3
4
5
6
7
8
9
10



Component 2, 10 dimensions
Computed mean
Expected mean
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
2
3
4
5
6
7
8
9
10

Published with MATLAB® R2018a