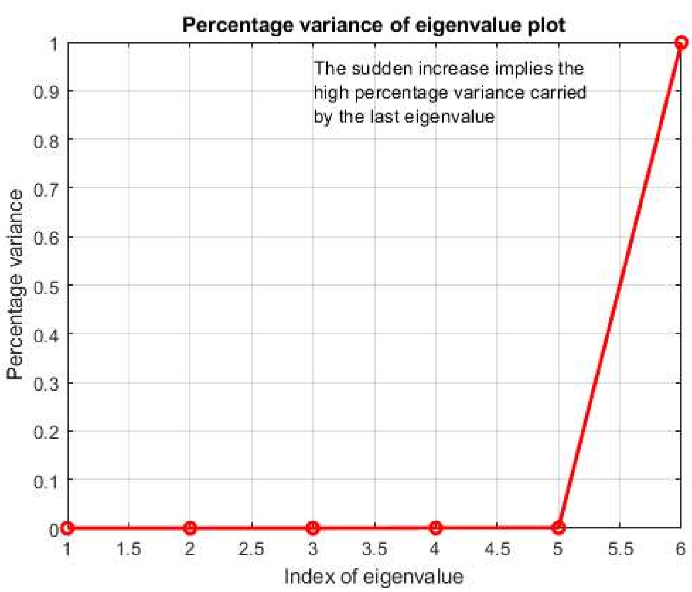
Contents


```
% Assignment 3, Multivariate Data Analysis CH5440
% Ojas Phadake CH22B007

clc;
clear;
close all;
```

A

No of independent variables = 6 No of linear constraints = 3 (volume balance on 3 nodes)

```
nsamples = 1000;
load flowdata.mat;
Atrue = [1 -1 1 0 1 0; 0 1 0 -1 -1 0; 0 0 -1 1 0 -1];

std(1) = 0.1;
std(2) = 0.1;
std(3) = 0.1;
std(4) = 0.1;
std(5) = 0.1;
std(6) = 0.1;
Ltrue = diag(std); % Assumed standard deviation variations

nvar = 6;

[~, D] = eig(Fmeas'*Fmeas/nsamples);

var = diag(D);
var = var/sum(var); % This contains the fraction of data variance contained by each eigenvalue

x_axis = 1:1:6;
plot(x_axis, var, "LineStyle","-", "Color","red", "Marker","o", "LineWidth", 2);
grid on;
title("Percentage variance of eigenvalue plot"); xlabel("Index of eigenvalue"); ylabel("Percentage variance");
str = {'The sudden increase implies the', 'high percentage variance carried', 'by the last eigenvalue'};
text(3, 0.9, str)
```



B

```
Lsinv = inv(Ltrue);
Ys = Fmeas*Lsinv/sqrt(nsamples);

[v, D] = eig(Fmeas'*Fmeas/nsamples);
lambda = diag(D);
dest = nvar-1;
nfact = nvar-dest;
Amat = v(:,1:dest)';
% Use SVD to determine eigenvectors

[u, s, v] = svd(Ys,0);
spca = diag(s);
lambda = spca.^2;
% Use hypothesis test to determine number of constraints

alpha = 0.05;
tau = zeros(nvar-2,1);
crit = zeros(nvar-2,1);
for d = nvar-1:-1:2
   nu = (d-1)*(d+2)/2;
   nprime = nsamples-nvar - (2*nvar+11)/6;
   lbar = mean(lambda(nvar-d+1:end));
   tau(d-1) = nprime*(d*log(lbar)-sum(log(lambda(nvar-d+1:end))));
   crit(d-1) = chi2inv(1-alpha,nu);
end
flag = 1;
d = nvar-2;

dest = 1;
while flag
    if ( tau(d) > crit(d) )
        d = d-1;
        if ( d < 2)
           flag = 0;
        end
    else
        dest = d + 1;
        flag = 0;
    end
    disp("Flag is:"+ flag);
    disp("Dest is: " + dest);
end
disp("-------------------------------------------------------------------------")
```

```
Flag is:1
Dest is: 1
Flag is:1
Dest is: 1
Flag is:0
Dest is: 3
-------------------------------------------------------------------------
```

Hypothesis testing by a separate method

```
p = size(D, 1);
r = p-1;
j = 1;

j_arr = []; r_arr = []; Q_val = []; c_val = []; stat = [];

diag_mat = sort(diag(D), 'descend');

while j<p-2
    Q = testStat(diag_mat, r, j);
    dof = (r*(r+1)/2) - 1;
    c = chi2inv(0.95, dof);
    if Q>c
        stat = [stat; "Rejected"];
    else
        stat = [stat; "Accepted"];
    end
    j_arr = [j_arr; j]; r_arr = [r_arr; r]; Q_val = [Q_val; Q]; c_val = [c_val; c];
    if j == p-r
        r = r - 1;
        j=1;
    end
    if r < 2
        break;
    end
```

```
    j = j + 1;
end

disp("What follows is the result of the hypothesis testing for normal method")
disp(table(j_arr, r_arr, Q_val, c_val, stat));
disp("The values correctly as above are j = 3, r = 3. ")
disp("That is, the fourth, fifth and sixth eigenvalues are same. ")
disp("-------------------------------------------------------------------------")
```

C

Checking if eigenvalues are same using the new method

```
% Equality of consecutive eigenvalues is checked, and this is ensured by
% taking r = 2

p = size(D, 1);
r = 2;
j = 1;

j_arr = []; Q_val = []; c_val = []; stat = [];

diag_mat = sort(diag(D), 'descend');

while j<p-1
    Q = testStat(diag_mat, r, j);
    dof = (r*(r+1)/2) - 1;
    c = chi2inv(0.95, dof);
    if Q>c
        stat = [stat; "Rejected"];
    else
        stat = [stat; "Accepted"];
    end
    j_arr = [j_arr; j]; Q_val = [Q_val; Q]; c_val = [c_val; c];
    j = j + 1;
end

disp(table(j_arr, Q_val, c_val, stat));
disp("-------------------------------------------------------------------------")
```

```
    j_arr      Q_val       c_val        stat
    _____    _________    ______    __________

      1         0.4026    5.9915    "Accepted"
      2         15.878    5.9915    "Rejected"
      3      0.0086489    5.9915    "Accepted"
      4      0.0019324    5.9915    "Accepted"

-------------------------------------------------------------------------
```

D

```
nfact = nvar - dest;
for k = nfact+1:nvar
    Amats(k-nfact,:) = v(:,k)'; % Right singular vectors corresponding to scaled data matrix
end
Amat = Amats*Lsinv;

theta_pca = 180*subspace(Atrue', Amat')/pi;
disp("The subspace angle, theta_pca is: " + theta_pca)
disp("No of constraints are: "+ dest);
disp("-------------------------------------------------------------------------")
```

```
The subspace angle, theta_pca is: 0.35852
No of constraints are: 3
-------------------------------------------------------------------------
```

E

Poor choices of independent variables identification

```
set_var = [1,2,3,4,5,6];
all_sets_three = nchoosek(set_var, 3);
s = size(all_sets_three, 1);
det_arr = zeros(s, 1);
```

```
cond_num = zeros(s,1);

for i=1:s
    Aestd = Amat(:, all_sets_three(21-i, :));
    Aesti = Amat(:, all_sets_three(i, :));
    Rest = -inv(Aestd)*Aesti;
    det_arr(i) = det(Rest);
    cond_num(i) = cond(Rest);
end

high_cond = (1000 < cond_num);
num_bad_sets = sum(high_cond)/size(high_cond, 1);
disp("The number of bad combinations are: " + sum(high_cond));

disp("The bad combinations as independent sets are: ")
bad_combinations = all_sets_three(high_cond, :); determinants = det_arr(high_cond); condition_num = cond_num(high_cond);
display_bad = table(bad_combinations, determinants, condition_num);
disp(display_bad);

fprintf("\nNow printing all the good combinations:\n")
good_combinations = all_sets_three(~high_cond, :); determinants = det_arr(~high_cond); condition_num = cond_num(~high_cond);
display_good = table(good_combinations, determinants, condition_num);
disp(display_good);

disp("The regression matrix was checked for condition number. If cond is very high, then it means that the matrix is ill conditioned or not invertible. ")
disp("I set the bounds of modulus of cond as 1000 and then classified all those as ill conditioned sets")
disp("------------------------------------------------------------------------")
```

```
The number of bad combinations are: 10
The bad combinations as independent sets are:
    bad_combinations    determinants    condition_num
    ________________    ____________    _____________

      1    2    5         0.0020632         2530.2
      1    2    6            1087.6           5773
      1    3    6         -0.038999     1.0386e+07
      1    4    6            262.01           1421
      1    5    6            207.76         1119.7
      2    3    4         0.0048132         1119.7
      2    3    5         0.0038167           1421
      2    4    5           -25.642     1.0386e+07
      3    4    5        0.00091949           5773
      3    4    6            484.67         2530.2


Now printing all the good combinations:
    good_combinations    determinants    condition_num
    _________________    ____________    _____________

       1    2    3         -0.98908         4.0494
       1    2    4         -0.99593         4.0634
       1    3    4           166.81         895.59
       1    3    5         -0.98097         4.0771
       1    4    5         -0.98396         4.0453
       2    3    6          -1.0163         4.0453
       2    4    6          -1.0194         4.0771
       2    5    6        0.0059949         895.59
       3    5    6          -1.0041         4.0634
       4    5    6           -1.011         4.0494

The regression matrix was checked for condition number. If cond is very high, then it means that the matrix is ill conditioned or not invertible.
I set the bounds of modulus of cond as 1000 and then classified all those as ill conditioned sets
------------------------------------------------------------------------
```

F

Independence terms as follows

```
Aestd = Amat(:, 1:3);
Aesti = Amat(:, 4:6);

Atrued = Amat(:, 1:3);
Atruei = Atrue(:, 4:6);

Rtrue = -inv(Atrued)*Atruei;
Rest = -inv(Aestd)*Aesti;

maxdiff = max(max(abs(Rest - Rtrue)))
disp("The final constraint/dependence matrix assuming that flow streams 4,5 and 6 are independent and 1,2 and 3 are dependent is:");
```

```
disp(Rest)
disp("------------------------------------------------------------------------")
```

```
maxdiff =

    1.0661

The final constraint/dependence matrix assuming that flow streams 4,5 and 6 are independent and 1,2 and 3 are dependent is:
   -0.0048   -0.0038    1.0141
    1.0013    0.9929   -0.0002
    1.0029   -0.0021   -1.0074

------------------------------------------------------------------------
```

Utility function for evaluating the cost function

```
function Q = testStat(D, r, j)
    N = size(D, 1);
    sum_lambda = sum(D(j+1:j+r))/r;
    sum_log_lambda = sum(log(D(j+1:j+r)));
    Q = r*(N-1)*log(sum_lambda) - (N-1)*sum_log_lambda;
end
```

```
What follows is the result of the hypothesis testing for normal method
    j_arr    r_arr      Q_val      c_val        stat
    _____    _____    _________    ______    __________

      1        5         51.791    23.685    "Rejected"
      2        4         42.002    16.919    "Rejected"
      2        3         29.647     11.07    "Rejected"
      3        3       0.019692     11.07    "Accepted"
      2        2         15.878    5.9915    "Rejected"
      3        2      0.0086489    5.9915    "Accepted"

The values correctly as above are j = 3, r = 3.
That is, the fourth, fifth and sixth eigenvalues are same.
------------------------------------------------------------------------
```

Published with MATLAB® R2023a