Tutorial 5§:
Advanced Statistics with MATLAB

Daniela Raicu
draicu@cs.depaul.edu
School of Computer Science, Telecommunications, and Information Systems
DePaul University, Chicago, IL 60604

The purpose of this tutorial is to present several advanced statistics techniques using Matlab Statistics toolbox. For this tutorial, we assume that you know the basics of Matlab (covered in Tutorial 1) and the basic statistics functions implemented in Matlab. If you need to refresh your Matlab skills, then a good starting point is to read and solve the exercises from Tutorial #1 (Introduction to Matlab) and Tutorial #3 (Statistics with matlab). The tutorial purpose is to teach you how to use several Matlab built-in functions to calculate advanced statistics for different data sets in different applications; the tutorial is intended for users running a professional version of MATLAB 6.5, Release 13.

Topics discussed in this tutorial include:

1. Covariance matrices and Eigenvalues
2. Principal component analysis
3. Canonical Correlation
4. Polynomial fit for a set of points

1. Covariance matrices and Eigenvalues

Covariance matrix from raw data "cov.m"

```
» X = [1 1 2 3;7 8 9 7;1 3 2 1;10 9 11 9;2 2 1 1; 3 4 1 2]

X =

     1     1     2     3
     7     8     9     7
     1     3     2     1
    10     9    11     9
     2     2     1     1
     3     4     1     2

» cov(X)

ans =

   13.6000   11.6000   15.6000   11.8000
   11.6000   10.7000   13.6000    9.9000
   15.6000   13.6000   19.8667   14.6667
   11.8000    9.9000   14.6667   11.3667
```

♣ Event Sponsor: Visual Computing Area Curriculum Quality of Instruction Council (QIC) grant

```
» cov(X(:,1))

ans =

   13.6000

» cov(X(:,2))

ans =

   10.7000
```

Eigenvalues from raw data "eig.m"

```
» eig(cov(X))

ans =

    0.5731
    0.1590
    1.4018 %second largest eigenvalue that captures the second largest variance
   53.3994 %largest eigenvalue that captures the largest variance in the data
```

2. Principal Component Analysis

Principal components analysis from raw data: "princomp.m": takes a data matrix X and returns the principal components in PC, the so-called Z-scores in SCORES, the eigenvalues of the covariance matrix of X in LATENT, and Hotelling's T-squared statistic for each data point in TSQUARE.

```
» [pc, score, latent, tsquare] = princomp(X)

pc =

    0.4961   -0.3697    0.6397   -0.4561
    0.4316   -0.6706   -0.4003    0.4515
    0.6032    0.4179   -0.5296   -0.4254
    0.4514    0.4889    0.3874    0.6381


score =

   -4.7824    2.0736    0.3947    0.2489
    7.2433    0.0423   -0.7265    0.2473
   -4.8221   -0.2454   -1.1806   -0.1243
   11.2723    0.0763    0.5080   -0.2441
   -5.3608   -0.3624    0.3889   -0.6064
   -3.5502   -1.5843    0.6155    0.4786


latent =
```

```
   53.3994
    1.4018
    0.5731
    0.1590


tsquare =

    4.1571
    2.2893
    3.0077
    3.2088
    3.2088
    4.1284
```

3. Canonical Correlation

Canonical correlation analysis for two types of attributes: “Canoncorr.m”

[A,B,R,U,V] = CANONCORR(X,Y) computes the sample canonical coefficients for the N-by-D1 and N-by-D2 data matrices X and Y. X and Y must have the same number of observations (rows) but can have different numbers of variables (cols). The jth columns of A and B contain the canonical coefficients, i.e. the linear combination of variables making up the jth canonical variable for X and Y, respectively. Columns of A and B are scaled to make COV(U) and COV(V) (see below) the identity matrix. R containing the sample canonical correlations; the jth element of R is the correlation between the jth columns of U and V (see below), the canonical variables computed as

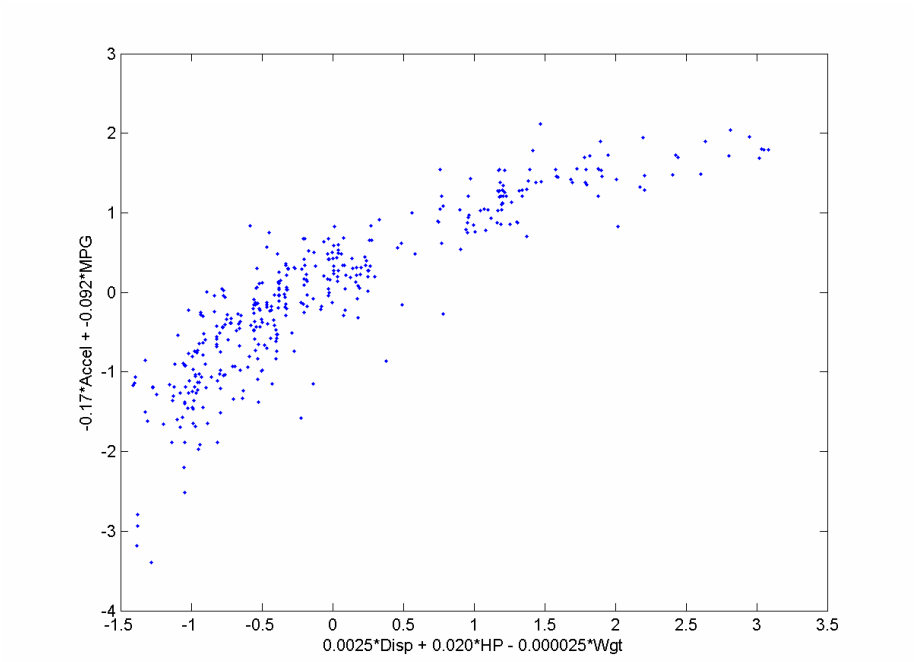
```
       U = (X - repmat(mean(X),N,1))*A and
       V = (Y - repmat(mean(Y),N,1))*B.
```

Example:

```
       load carbig;
       X = [Displacement Horsepower Weight Acceleration MPG];
       nans = sum(isnan(X),2) > 0;
       [A B r U V] = canoncorr(X(~nans,1:3), X(~nans,4:5));

       plot(U(:,1),V(:,1),'.');
       xlabel('0.0025*Disp + 0.020*HP - 0.000025*Wgt');
       ylabel('-0.17*Accel + -0.092*MPG')
```

```
>> A

A =

    0.0025    0.0048
    0.0202    0.0409
   -0.0000   -0.0027

>> size(X)

ans =
```

```
   406     5

>> B

B =

   -0.1666   -0.3637
   -0.0916    0.1078

>> r

r =

    0.8782    0.6328
```



4. Polynomial fitting

```
>> Y =

     1     1
     7     8
     1     3
    10     9
     2     2
     3     4
     2     3
```

```
        6     6
```

```
   » plot(Y(:,2),Y(:,1),'+')
```

Fit polynomial to data: "polyfit.m":

[P, S] = POLYFIT(X, Y, N) returns the polynomial coefficients P and a structure S for use with POLYVAL to obtain error estimates on predictions.

```
>>[P,S]=polyfit(Y(:,1),Y(:,2),2)

P =

   -0.0309    1.1623    0.6383

S =

        R: [3x3 double]
       df: 5
    normr: 1.9199
```

Evaluate polynomial: "polyval.m"

Y = POLYVAL(P,X), when P is a vector of length N+1 whose elements are the coefficients of a polynomial, is the value of the polynomial evaluated at X.

```
        Y = P(1)*X^N + P(2)*X^(N-1) + ... + P(N)*X + P(N+1)
```

If X is a matrix or vector, the polynomial is evaluated at all points in X. See also POLYVALM for evaluation in a matrix sense.

[Y, DELTA] = POLYVAL (P,X,S) uses the optional output structure generated by POLYFIT to generate error estimates, Y +/- delta.

```
>> Y_new=polyval(P,Y(:,1))


Y_new =


    1.7697
    7.2612
    1.7697
    9.1731
    2.8394
    3.8473
    2.8394
    6.5004
>> plot(Y(:,2),Y(:,1),'*',Y_new(:,1),Y(:,1),'+')
```