

Matrix Algebra

Ken Nyholm

Outline

- Matrix operations
- OLS example
 - Matrix algebra
 - Maximum likelihood estimation
- Lagged variables
- Simulation example

Matrix operations

- Matrix transpose
- Dimension of matrices

while c would be a vector. The superscript T is used to signify the transpose of a matrix or a vector, for example, C^T and c^T would be the transpose of the before mentioned matrix and vector. The transposition operator reorganises the rows and columns vectors and matrices: what was before a column becomes a row after the transposition is invoked. Subscripts, in connection to matrices and vectors are used to signify the dimension of the object at hand, e.g:

$$C = C_{(n,k)} = \begin{bmatrix} c_{1,1} & \cdots & c_{1,k} \\ \vdots & & \vdots \\ c_{n,1} & \cdots & c_{n,k} \end{bmatrix}$$

in the case of a matrix, where n gives the number of rows and k the number of columns. Hence a column vector will have $k = 1$ and a row vector will have $n = 1$. Effectively, C^T , is:

$$C^T = C_{(k,n)}^T = \begin{bmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & & \vdots \\ c_{k,1} & \cdots & c_{k,n} \end{bmatrix}.$$

Matrix operations

- In Matlab it looks like this:

```
Command Window
>> C = randn(4,2)

C =

    0.71432    1.254
    1.6236   -1.5937
   -0.69178   -1.441
    0.858     0.57115

>> C'

ans =

    0.71432    1.6236   -0.69178    0.858
    1.254   -1.5937   -1.441    0.57115
```

Matrix operations

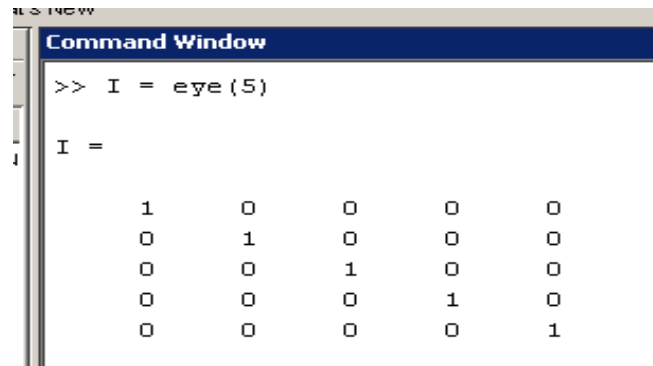
- The identity matrix

A special matrix is denoted the "identity" matrix and defined by

$$I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & 1 & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Similarly, the symbol ι is used to denote a vector of ones, so $\iota = [1, 1, \dots, 1]^T$.

- In Matlab



```
Command Window
>> I = eye(5)

I =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

Matrix operations

- Matrix product

of Matlab, a "." in front of an operator signifies an element-by-element operation. The example below illustrates the difference between element-by-element multiplication and conventional multiplication.

```
.....  
[1] Q=[1 2 3; 4 5 6; 7 8 9];  
[2] q =[1;1;1];  
[3] z1=Q*q    % regular matrix multiplication  
[4] z2=Q(:,1).*q  % element-by-element multiplication  
.....
```

The result stored in z1 is equal to

$$z1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} (1*1) + (2*1) + (3*1) \\ (4*1) + (5*1) + (6*1) \\ (7*1) + (8*1) + (9*1) \end{bmatrix} = \begin{bmatrix} 6 \\ 15 \\ 24 \end{bmatrix},$$

- Generally:

$$Q_{(m,n)} * W_{(n,k)} = Z_{(m,k)}.$$

Matrix operations

- Element-by-element operations in Matlab uses the `.*` operator
- Here, however, dimensions must match (see footnote 5 on page 10)

$$z2 = Q(:,1) .* q = \begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix} .* \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1*1 \\ 4*1 \\ 7*1 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix}.$$

- `Q(:,1)` takes out the first column of the matrix `Q`
- Naturally, `Q(:,k)` takes out the `k`'th column

Matrix operations

- Symmetric matrix:

5.2.5 Symmetric matrix

When $A = A^T$ the matrix A is said to be symmetric.

- Rank of a matrix:

5.2.8 Rank

A matrix is said to have full rank if all columns (rows) are linearly independent. A matrix having rank $\rho < c$, where c is the number of columns in the matrix is said not to have full rank. Let c be the number of columns and r be the number of rows of a given matrix, then:

$$\begin{aligned}\rho [A_{(r,c)}] &\leq \min(r, c), \\ \rho(A) &= \rho(A^T) \\ \rho(A^T * A) &= \rho(A * A^T) = \rho(A)\end{aligned}$$

When $\rho(A) = r$ the matrix is said to have full row rank and when $\rho(A) = c$ it is said to have full column rank. The rank concept is important because it relates to the invertability of matrices. For a square matrix $S_{(c,c)}$ with rank c , S is said to be non-singular and a unique S^{-1} , the inverse of S , exists. When the rank of S is less than c , S is said to be singular and its inverse does not exist.

Matrix operations

- Matrix inverse:

5.2.9 Inverse

If the determinant of a matrix $A_{(n \times n)}$ is different from zero then the inverse of A exists and is denoted by A^{-1} .

$$(A * B * C)^{-1} = C^{-1} * B^{-1} * A^{-1}$$

$$(A^T)^{-1} = (A^{-1})^T$$

$$A * A^{-1} = I$$

$$[\alpha A]^{-1} = \alpha^{-1} A^{-1}, \text{ for } \alpha \neq 0$$

A non-singular matrix is one for which the inverse exists. A singular matrix is one which has a determinant of zero. Thus, the inverse of a singular matrix does not exist.

Matrix operations

- Matrix inverse in Matlab

```
Command Window
>> disp(C)
     3     2     1
     2     2     1
     1     1     1

>> disp(inv(C))
     1    -1     0
    -1     2    -1
     0    -1     2

>> disp(C^-1)
     1    -1     0
    -1     2    -1
     0    -1     2

>> disp(pinv(C))
     1    -1     0
    -1     2    -1
-3.3307e-016 -1     2
```

Matrix operations

- `pinv` can be helpful in empirical implementations

```
Command Window
>> help pinv
PINV    Pseudoinverse.
        X = PINV(A) produces a matrix X of the same dimensions
        as A' so that A*X*A = A, X*A*X = X and A*X and X*A
        are Hermitian. The computation is based on SVD(A) and any
        singular values less than a tolerance are treated as zero.
        The default tolerance is MAX(SIZE(A)) * NORM(A) * EPS(class(A)).

        PINV(A,TOL) uses the tolerance TOL instead of the default.

        Class support for input A:
          float: double, single

        See also rank.

        Reference page in Help browser
          doc\_pinv
```

OLS example

- Ordinary Least Squares:

- 1.6.1 The basic setup

A linear relationship between variables, one variable y depending on other variables x_i s, can be formulated as:

$$Y = b_0 + b_1 * X_1 + b_2 * X_2 + \dots + e.$$

or when collected in matrix form:

$$Y = X * b + e, \tag{1.1}$$

- For a proper derivation of the OLS estimates see the lecture notes page 18-19
- Where the squared residuals are calculated and minimised

OLS example

- A short-hand calculation shows the following:

A short-hand way to obtain this relationship is by pre-multiplying [1.1] by X :

$$X^T Y = X^T X \beta + X^T e,$$

since $E[X^T e] = 0$, we obtain:

$$X^T Y = X^T X \beta,$$

and

$$\beta = (X^T X)^{-1} X^T Y.$$

- Which are the parameter estimates of the OLS regression

OLS example

- When we have parameter estimates there are also uncertainty around the parameter estimates
- These are called the standard errors on the parameter estimates
- And, for the OLS estimates, they are calculated by:

The error term variance σ^2 is unknown in most practical applications and thus needs to be approximated/estimated. This is usually done in the following way:

$$\text{var}[\beta] = s^2 (X^T X)^{-1} = \frac{u^T u}{N - k} (X^T X)^{-1},$$

where, $u = Y - X\beta$, i.e. the actual residuals obtained from data, and N is the total number of observations and k the number of estimated parameters.

OLS example

- Exercise: Make a function that calculates the OLS estimates, standard errors and R-squared of a dataset provided by the user

OLS example

- OLS can also be illustrated by maximum likelihood estimation
- Can be useful if we want to impose restriction on the parameter estimates
- But here it is rather shown as an introduction to finding the arguments that minimise a given function

OLS example

- The maximum likelihood set-up

can be found. To illustrate, below we sketch how the OLS regression can be solved using the maximum likelihood (ML) estimation technique.

$$Y = X\beta + u,$$

the residuals are assumed to follow a normal distribution, hence:

$$u \sim N(0, \sigma^2 I),$$

and we therefore need the normal density:

$$L(s^2, \beta) = (2\pi s^2)^{-T/2} \exp \left[-\frac{1}{2s^2} \sum_{j=1}^T (y_j - x_j \beta)^2 \right].$$

Functions that involve *exp* are difficult to optimise. It is much easier to optimise $\ln(L)$ in stead of L , such an approach is legitimate because the parameters that optimise L also optimise $\ln(L)$. For these reasons likelihood optimisation involves the log likelihood function:

$$\log L = -\frac{T}{2} \ln(2\pi s^2) - \frac{1}{2s^2} \sum_{j=1}^T (y_j - x_j \beta)^2.$$

- See, EX_likeli_1.m

Lagged variables

$$r_t = k + ar_{t-1} + br_{t-2} + cr_{t-3} + u_t.$$

As can be seen from the regression equation the right-hand-side variables are all lagged versions of the left-hand-side variable. Such a regression is also called an autoregression which is defined by the number of lags that are included. In our case three lags are included and the regression is hence termed an autoregression of order 3, in short AR(3). Naturally this concept generalises to n lags, and hence to an AR(n), and to k variables, in which case the regression is called a vector autoregression, in short VAR.¹⁰

To construct the left- and right-hand-side variable from R the following can be invoked:

```
.....  
[1] r      = R(2:end,1); % allocates the 3m rate to variable r  
[2] rLag3 = r(1:end-3,:); % creates the 3.lag  
[3] rLag2 = r(2:end-2,:); % creates the 2.lag  
[4] rLag1 = r(3:end-1,:); % creates the 1.lag  
[5] Y = r(4:end,:); % adjusting r  
[6] X = [ ones(length(Y),1) rLag1 rLag2 rLag3 ];  
.....
```

Simulation example

- Start with the desired model, for example the following

$$\hat{r}_t = k + a * \hat{r}_{t-1} + b * \hat{r}_{t-2} + c * \hat{r}_{t-3} + \hat{u}_t,$$

using $[\hat{r}_3; \hat{r}_2; \hat{r}_1; \hat{u}_3] = [r_3; r_2; r_1; u_3] = [0.64; 0.76; 0.97; -0.097]$. This will allow us to generate simulated sample paths for r_t taking into account parameter uncertainty as well as the uncertainty that stems from the innovations to the process as represented by the error term. Often in simulation applications only a new innovation term is generated on the basis of an assumed distribution, or regenerated from the observed errors (this is also termed bootstrapping), and the parameter uncertainty is ignored. There is no particular reason as to ignore parameter uncertainty, it all depends on the problem that one sets out to solve.

The current example is fairly general in that it allows for both parameter uncertainty as well as a uncertainty stemming from the innovation term. In order to generate the simulation, the following steps need to be completed:

- 1) draw innovations from the appropriate distribution of the desired length
- 2) generate parameter-value draws, also from appropriate distributions
- 3) insert the generated parameters in the equation to the simulated
- 4) feed innovations through the equation

Simulation example

- Generating correlated random numbers (see pages 91-93 in the lecture notes):

$$C = U^T U. \quad (4.23)$$

The upper triangular matrix U is useful for generating correlated random numbers in the following way:

$$R_{(r,c)} = n_{(r,c)} * U_{(c,c)}. \quad (4.24)$$

First uncorrelated random numbers can be generated of the desired dimension, e.g. $n \sim N(0, 1)$. Then these random numbers are "run through" the Cholesky decomposition to generate the matrix R of correlated random numbers of the desired dimension, here (r, c) .

Simulation example

- A Matlab implementation of this:

```
-----  
[1] disp('The covariance matrix')  
[2] C = [ 7.9440 7.8265 8.3193 6.8492;  
[3]      7.8265 7.7473 8.2614 6.8304;  
[4]      8.3193 8.2614 8.8664 7.4661;  
[5]      6.8492 6.8304 7.4661 7.1715 ]  
[6] disp('Upper triangular from cholesky decomposition')  
[7] U = chol(C)  
[8] disp('Calculating transpose(U)*U')  
[9] disp(U'*U)  
[10] n = randn(500000,4);  
[11] R = n*U;  
[12] disp('Covariance matrix of simulated random variables')  
[13] disp(cov(R))  
-----
```