MAT 343 Laboratory 5
Least Squares

In this laboratory session we will learn how to

1. Factor a symmetric matrix using the Cholesky decomposition.
2. Solve Least Squares problems using MATLAB
3. Plot the data points together with the least squares approximation.

Introduction

Let S be a symmetric matrix. Since S is a square matrix (why?), it has an LU decomposition. In fact, the symmetry makes it possible to write S in the form $S = LL^T$, where L is a lower-triangular matrix. This is called the *Cholesky decomposition* of S, and it is defined only for symmetric matrices. Given a symmetric matrix, the Cholesky decomposition is always preferable in computations because it can be computed in half the time of the LU decomposition.
There is nothing special about writing the Cholesky decomposition in terms of a lower-triangular matrix. We can just as well write $S = U^TU$, where $U = L^T$. In MATLAB, the command

```
U = chol(S)
```

returns the Cholesky decomposition of the symmetric matrix S in the upper-triangular matrix U.
Once the Cholesky decomposition $S = U^TU$ is found, solving a system of the form

$$S\mathbf{c} = \mathbf{z}$$

proceeds in the same 2-step manner as solving $A\mathbf{x} = \mathbf{b}$ with the LU decomposition:

1. Solve $U^T\mathbf{w} = \mathbf{z}$.
2. Solve $U\mathbf{c} = \mathbf{w}$.

The normal equations for the least squares

Suppose we are given a collection of data points of the form $\{(x_i, y_i)\}_{i=1}^n$, which are observed values of some quantities x and y (for example, some physical, or any other quantities). The simplest possible relation between x and y is linear: $y = c_1 + c_2x$, where c_1 and c_2 are some unknown constant coefficients (yet to be determined). However, due to error in measurement (imperfect device, imperfect conditions of measuring, causing inevitable noise), the above theoretical relation is hardly ever achieved. Instead, the true relation between observed quantities x and y is of the form

$$y = c_1 + c_2\,x + \varepsilon \tag{1}$$

where ε is the random error, or "noise" - that is, the amount by which $c_1 + c_2\,x$ fails to coincide with y. Applying our model represented by the equation (1) to the data points, we have

$$y_i = c_1 + c_2x_i + \varepsilon_i. \tag{2}$$

Our model just states that the relation is linear (plus the error term), but does not say anything about the values of the constants c_1 and c_2. For the same set of data $\{(x_i, y_i)\}_{i=1}^n$, different choice of c_1 and c_2 gives different errors (residuals) ε_i. It is our task to find most plausible estimates of c_1 and c_2 based

on the observed data. Although MATLAB provides an entire toolbox for statistical analysis, we will use only a few simple commands to get started; the pedagogical purpose is to make sure that you understand how the problem is set up mathematically.
Equation (2) can be extended to any polynomial model. For example, if we suppose that

$$y_i = c_1 + c_2 x_i + c_3 x_i^2 + \ldots + c_p x_i^{p-1} + \varepsilon_i \tag{3}$$

then we have to estimate the p unknowns $c_1, c_2, \ldots, c_p$ so that ε_i is small for all i. Statisticians write the estimation problem in the following form:

$$\mathbf{y} = X\mathbf{c} + \varepsilon \tag{4}$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$ is an n-vector of observations (i.e., the dependent variable), $\mathbf{c} = (c_1, c_2, \ldots, c_p)^T$ is the p-vector of unknown parameters, X is the $n \times p$ matrix

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{p-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{p-1} \\ \vdots & \ddots & \vdots & & \\ 1 & x_n & x_n^2 & \ldots & x_n^{p-1} \end{bmatrix}$$

and $\varepsilon = (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)^T$ is the vector of residuals corresponding to the n observations.
It is straightforward to verify that the ith row of the matrix-vector product, plus the ith component of ε, yields the right-hand side of (3).
The least squares solution of (4) is the solution $\widehat{\mathbf{c}}$ of the *Normal Equations:*

$$X^T X\mathbf{c} = X^T\mathbf{y} \tag{5}$$

The vector $\widehat{\mathbf{c}}$ minimizes the quantity $\parallel \varepsilon \parallel = \parallel \mathbf{y} - X\mathbf{c} \parallel$, that is, minimizes the difference between the sum of squares of the differences between the "predicted" value of $\mathbf{y}$ from the "observed" value of $\mathbf{y}$, where the sum is taken over every point in the dataset.
Notice that $X^T X$ is a symmetric matrix.
The system (5) can be solved as follows:

1. Define $\mathbf{z} = X^T\mathbf{y}$.
2. Define $S = X^T X$.
3. Solve $S\mathbf{c} = \mathbf{z}$.

The last step must be performed using the Cholesky decomposition of S and solving the two triangular systems.

MATLAB Notes

- Transposition is denoted by prime (i.e. a single quotation mark): X^T is denoted by `X'`.
- The triangular systems obtained from the Cholesky decomposition must be solved using the MATLAB "backslash" command. For example, to solve $U^T\mathbf{w} = \mathbf{z}$, type `w =U'\z`.
- Given the vector $\mathbf{a} = [a_1, a_2, \ldots, a_n]^T$, you can form the vector $[a_1^2, a_2^2, \ldots, a_n^2]^T$ with the expression `a.^2` (component-wise exponentiation).
- When building the matrix X you should use the special matrix `ones`.

Example 1: Least Squares Fit to a Data Set by a Linear Function.

Compute the coefficients of the best linear least-squares fit to the following data.

x	2.4	3.6	3.6	4.1	4.7	5.3
y	33.8	34.7	35.5	36.0	37.5	38.1

Plot both the linear function and the data points on the same axis system.

Solution

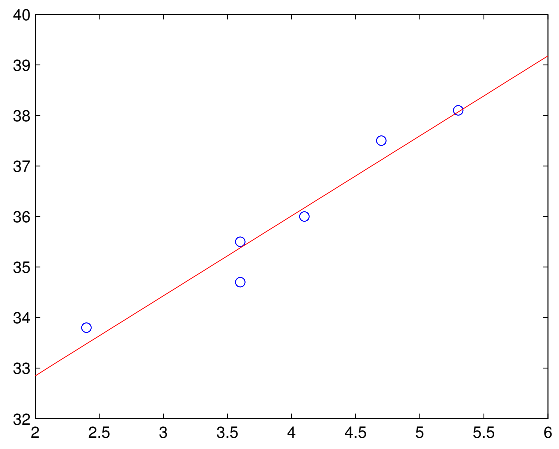
We can solve the problem with the following MATLAB commands

```
x = [2.4;3.6;3.6;4.1;4.7;5.3];          % build vector of x -values
y = [33.8;34.7;35.5;36.0;37.5;38.1];    % build vector of y -values
X = [ones(size(x)),x];    % build the matrix X for linear model
z = X'*y;                 % right hand side of the Normal Equations
S = X'*X;                 % Left hand side of the Normal Equations
U = chol(S);              % Cholesky decomposition
w = U'\z;     % solve the normal equations using Cholesky decomposition
c = U\w
plot(x,y,'o')             % plot the data points
q = 2:0.1:6;              % define a vector for plotting the linear fit
fit = c(1)+c(2)*q;        % define the linear fit
hold on
plot(q,fit,'r');      % plot the linear fit together with the data points
```

Running the script file produces the output

```
c =
   29.6821
    1.5826
```

and the following figure.



EXERCISES

Instructions: For each of the following exercises, create an M-file to store the MATLAB commands. Copy and paste the M-file into a text document. Include in the text document the pictures produced by MATLAB. Resize and crop the pictures so that they do not take up too much space. If the question requires written answers, include them in the text file in the appropriate location. Make sure you clearly label and separate all the Exercises and remove unnecessary page breaks and blank spaces.

1. Download the file `gco2.dat` from the web page. This file contains data from 31 years in the span from 1980 to 2017 and the average global annual carbon dioxide concentration in the atmosphere (in parts per million) over marine surface sites (7 years are missing, for the purpose of randomization of the lab problem). The data come from the Earth Systems Research Laboratory (Global Monitoring Division) of the National Oceanic and Atmospheric Administration (which is the government agency charged with developing weather and climate forecast models; if you are interested in additional information and data set you can visit http://www.esrl.noaa.gov/gmd/ccgg/trends). The estimated standard deviation in the observations is 0.1 ppm.
   Save the file in the MATLAB working directory. Then load it with

   ```
   dat = load('gco2.dat');
   ```

   Note: if you do not save the file in your working directory, you need to include the whole path to load it into MATLAB. For instance, if you saved your file in the `C:\tmp` directory you can load it by entering `load('C:\temp\gco2.dat')`.
   This creates the 31×2 array `dat` whose first column is the year and the second column is the CO_2 concentration.
   You may find it convenient to create two separate data vectors, as follows:

   ```
   year = dat(:,1);
   conc = dat(:,2);
   ```

   Plot the data with

   ```
   plot(year,conc,'o')
   ```

   (a) Follow Example 1 to find the line of the best fit for these data. What are the coefficients c_1 and c_2? Give the values with five digits of accuracy (use `format short e`). Plot the line in black (use `q = year;`), together with the original data. Use `'linewidth',2` and `axis tight` in your plot.

   (b) Find the best-fit *quadratic* polynomial to the CO_2 data (make sure you change appropriately the matrix X). What are the coefficients c_1, c_2 and c_3? Give the values with five digits of accuracy (use `format short e`). Plot the fitted curve in red (use `'linewidth' ,2`) together with the data points and the linear fit from part (a).
   Make sure you are plotting both lines and data points together on the same figure.
   Add a legend to the graph:
   `legend('data points', 'linear fit', 'quadratic fit', 'location' ,'northwest')`.

2. Han invests \$15,000 into a fund that combines stocks and bonds. The return varies from year to year. The balance at 5-year intervals is given in the table below

t	Balance ($\times$ \$1,000)
0	15
5	15.7
10	18.7
15	20
20	21.9
25	24.8

The goal of this problem is to find constants a and b such that the model $y = ae^{bt}$ best fits the data. In order to do that, we apply the natural logarithm to both sides of the model. This yields

$$\ln y = \ln \left(ae^{bt}\right)$$

and using properties of logarithms

$$\ln y = \ln a + \ln e^{bt}$$

$$\ln y = \ln a + bt$$

If we let $Y = \ln y$, $c_1 = \ln a$ and $c_2 = b$, the problem now reduces to finding the linear fit $Y = c_1 + c_2 t$ to the set of data points

t	Y
0	ln(15)
5	ln(15.7)
10	ln(18.7)
15	ln(20)
20	ln(21.9)
25	ln(24.8)

(a) Follow Example 1 to find the best linear fit to the set of data points (t, Y) in the table above using MATLAB. Note that the natural logarithm is entered as `log` in MATLAB.
What values do you obtain for c_1 and c_2?
Plot the linear fit together with the points (use `q = t;`)

(b) Recalling that $y = e^Y$, $a = e^{c_1}$ and $b = c_2$, plot the original data points t and y together with the exponential fit $y = ae^{bt}$. Make sure you use an appropriate vector `q` so that the graph of the exponential is nice and smooth.

(c) Use your model to predict when the balance will reach \$30,000 dollars. Note: this question can be answered by hand or using MATLAB (by extending the `q` range and zooming in the graph of the fit), by using either the linear fit or the exponential model. Make sure you explain in detail which method you used.
Caveat: Regression models are not very reliable for predicting response variable (in this case "balance") for values of predictor (in this case "time") outside of the collected data. While it certainly makes sense to ask a question about future balance, one should keep this caveat in mind when extrapolating the regression model, i.e. when extending it outside the scope of the data.

3. Consider the following set of data, obtained from http://www.weatherbase.com, which shows the average daily reflected solar radiation (in kWh/m^2) for each month measured in the last two decades at ASU, Tempe campus. The average daily reflected solar radiation, denoted by Y, can be considered as a function of month m. The months range from 1 to 12, where 1 corresponds to the month of January.

m	1	2	3	4	5	6	7	8	9	10	11	12
Y	4.9	5.5	6.5	7.1	7	6.8	6.2	6	6.4	6	5.3	4.8

Enter the months as a column vector `m` (use the colon operator `:`), and enter the average daily reflected solar radiation values as a column vector `Y`.
The goal of this problem is to find the polynomial $y = c_1 + c_2\,m + c_3\,m^2 + c_4\,m^3 + c_5\,m^4 + c_6\,m^5$, of degree 5, which gives the best least square fit to the data.

(a) Follow Example 1 to find the coefficients of the best polynomial fit of degree 5 (make sure you appropriately modify the matrix X). Print the `c` values as you will need these to compare to

part (b).
Plot the data together with the polynomial fit. Make sure you define your vector `q` so that the graph of the polynomial is nice and smooth.
Hint: In order to define an appropriate vector `q`, think about the start and end values of the `m` vector and what stepsize you may want to use to obtain a smooth graph.

(b) Let X be the matrix you used in part (a). Here is an easier way to evaluate and plot the polynomial of degree 5:

```
c = X\Y
c = c([6:-1:1]);
q = 1:0.1:12;
z = polyval(c,q);
figure
plot(q,z,m,Y,'o');
axis tight
```

How do the values of `c` compare to the ones you found in part (a)?
How does the plot compare to the one you found in part (a)?

Note that the system $X\mathbf{c} = \mathbf{Y}$ is overdetermined. In this case, the "\" command finds the least-squares solution to the system. In fact, for overdetermined systems, the "\" command is equivalent to solving the Normal Equations using the Cholesky decomposition, just like you did in the previous problems. Thus the vector `c = X\Y` is the same one you found in part (a) by solving the normal equations.
The command `polyval(c,q)` evaluates the polynomial with coefficients given by the vector `c` in *decreasing* powers of `q`. Because the powers are decreasing we had to rearrange the entries in `c` using the command `c = c([6:-1:1]);`. Type `help polyval` to learn more.