Least Squares Fitting

MATLAB Implementation

Tamas Kis | kis@stanford.edu

TAMAS KIS
https://github.com/tamaskis

LICENSE

Contents

least_squares_fit

Fits a linear, polynomial, power, exponential, or logarithmic model to a set of data using the method of least squares.

Syntax

```
[c0,c1] = least_squares_fit(x,y)
[c0,c1] = least_squares_fit(x,y,'linear')
c = least_squares_fit(x,y,'poly',n)
[a,b] = least_squares_fit(x,y,'power')
[a,b] = least_squares_fit(x,y,'exp')
[a,b] = least_squares_fit(x,y,'log')
```

Description

`[c0,c1] = least_squares_fit(x,y)` returns the coefficients `c0` and `c1` for the linear least squares fit $y = c_0 + c_1 x$ to a set of data defined by the vectors `x` (independent variable) and `y` (dependent variable).

`[c0,c1] = least_squares_fit(x,y,'linear')` returns the coefficients `c0` and `c1` for the linear least squares fit $y = c_0 + c_1 x$ to a set of data defined by the vectors `x` (independent variable) and `y` (dependent variable).

`c = least_squares_fit(x,y,'poly',n)` returns the coefficient vector $\mathbf{c} = (c_0, ..., c_n)^T$ for the n^{th} degree polynomial least squares fit $y = c_0 + c_1 x + \cdots + c_n x^n$ to a set of data defined by the vectors `x` (independent variable) and `y` (dependent variable).

`[a,b] = least_squares_fit(x,y,'power')` returns the coefficients `a` and `b` for the power least squares fit $y = ax^b$ to a set of data defined by the vectors `x` (independent variable) and `y` (dependent variable).

`[a,b] = least_squares_fit(x,y,'exp')` returns the coefficients `a` and `b` for the exponential least squares fit $y = ae^{bx}$ to a set of data defined by the vectors `x` (independent variable) and `y` (dependent variable).

`[a,b] = least_squares_fit(x,y,'log')` returns the coefficients `a` and `b` for the logarithmic least squares fit $y = a + b \ln x$ to a set of data defined by the vectors `x` (independent variable) and `y` (dependent variable).

Examples

Example 1

Fit a linear curve to the following data [5]:

x	y
2	4
3	5
5	7
7	10
9	15

■ *SOLUTION*

Defining the $\mathbf{x}$ and $\mathbf{y}$ vectors,

```
x = [2,3,5,7,9];
y = [4,5,7,10,15];
```

To find the coefficients for the linear least squares fit, we can use either of the following:

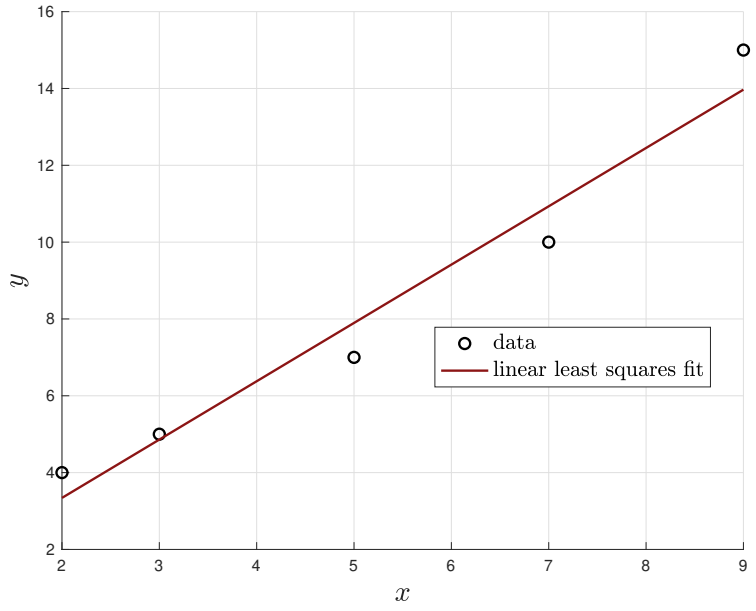
```
[c0,c1] = least_squares_fit(x,y);
[c0,c1] = least_squares_fit(x,y,'linear');
```

Defining the vectors to calculate/plot the linear fit $y = c_0 + c_1 x$,

```
x_fit = min(x):((max(x)-min(x))/1000):max(x);
y_fit = c0+c1*x_fit;
```

To plot,

```
figure;
hold on;
plot(x,y,'ko','markersize',7,'linewidth',1.5);
plot(x_fit,y_fit,'color',[140,21,21]/255,'linewidth',1.5);
hold off;
grid on;
xlabel('$x$','interpreter','latex','fontsize',18);
ylabel('$y$','interpreter','latex','fontsize',18);
legend('data','linear least squares fit','interpreter','latex',...
    'fontsize',14,'location','best');
```



Example 2

Fit a 2nd-degree polynomial (i.e. a quadratic) curve to the following data [3]:

x	y
0	8.3
1	11
2	14.7
3	19.7
4	26.7
5	35.2
6	44.4
7	55.9
8	68.9
9	83.2
10	98.8
11	114.2
12	127.1
13	140.1
14	164
15	190.9
16	214.3

■ *SOLUTION*

Defining the $\mathbf{x}$ and $\mathbf{y}$ vectors,

```
x = 0:1:16;
y = [8.3,11,14.7,19.7,26.7,35.2,44.4,55.9,68.9,83.2,98.8,114.2,127.1,...
    140.1,164,190.9,214.3];
```

To find the coefficients for the 2nd-degree polynomial least squares fit,

```
c = least_squares_fit(x,y,'poly',2);
```

Extracting the coefficients c_0, c_1, and c_2 from the coefficient vector $\mathbf{c} = (c_0, c_1, c_2)^T$,

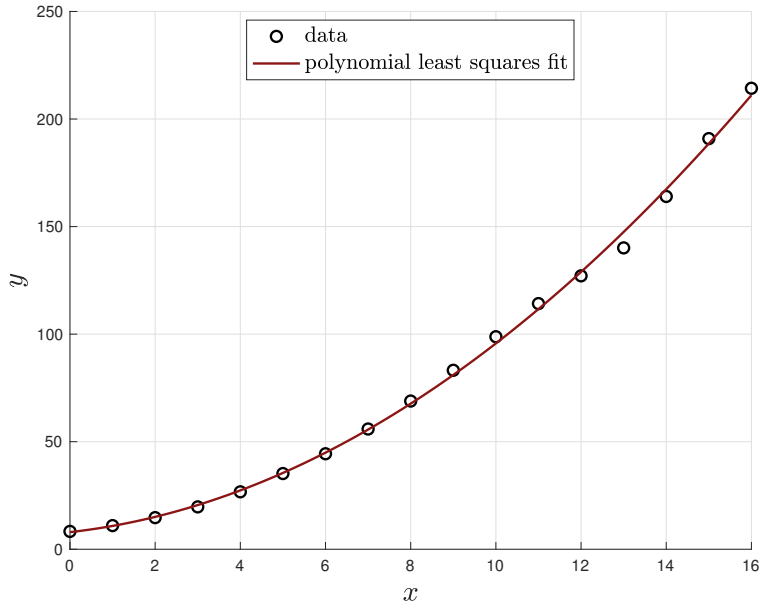
```
c0 = c(1);
c1 = c(2);
c2 = c(3);
```

Defining the vectors to calculate/plot the polynomial fit $y = c_0 + c_1 x + c_2 x^2$,

```
x_fit = min(x):((max(x)-min(x))/1000):max(x);
y_fit = c0+c1*x_fit+c2*x_fit.^2;
```

To plot,

```
figure;
hold on;
plot(x,y,'ko','markersize',7,'linewidth',1.5);
plot(x_fit,y_fit,'color',[140,21,21]/255,'linewidth',1.5);
hold off;
grid on;
xlabel('$x$','interpreter','latex','fontsize',18);
ylabel('$y$','interpreter','latex','fontsize',18);
legend('data','polynomial least squares fit','interpreter','latex',...
    'fontsize',14,'location','best');
```



Example 3

Fit a power curve to the following data [2]:

x	y
0.5	0.8
2.4	9.3
3.2	37.9
4.9	68.2
6.5	155
7.8	198

■ *SOLUTION*

Defining the **x** and **y** vectors,

```
x = [0.5,2.4,3.2,4.9,6.5,7.8];
y = [0.8,9.3,37.9,68.2,155,198];
```

To find the coefficients for the power least squares fit,

```
[a,b] = least_squares_fit(x,y,'power');
```

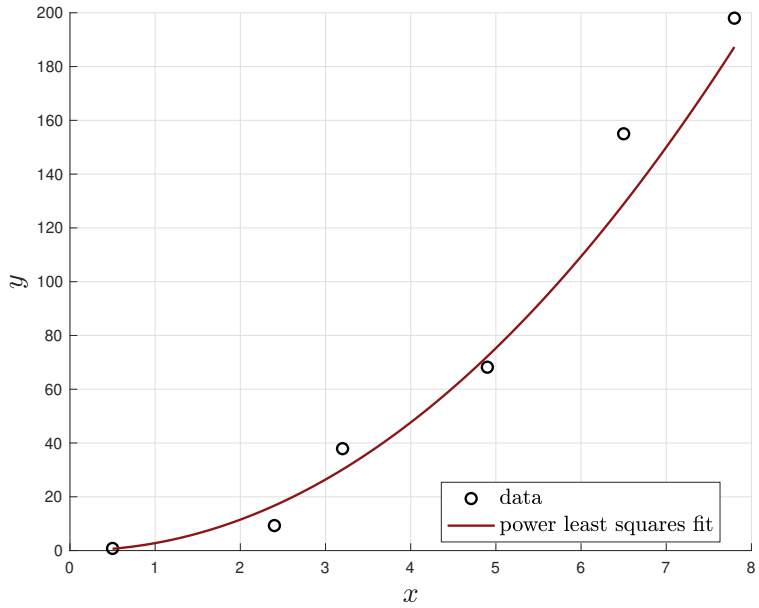
Defining the vectors to calculate/plot the power fit $y = ax^b$,

```
x_fit = min(x):((max(x)-min(x))/1000):max(x);
y_fit = a*x_fit.^b;
```

To plot,

```
figure;
```

```
hold on;
plot(x,y,'ko','markersize',7,'linewidth',1.5);
plot(x_fit,y_fit,'color',[140,21,21]/255,'linewidth',1.5);
hold off;
grid on;
xlabel('$x$','interpreter','latex','fontsize',18);
ylabel('$y$','interpreter','latex','fontsize',18);
legend('data','power least squares fit','interpreter','latex',...
    'fontsize',14,'location','best');
```



Example 4

Fit an exponential curve to the following data [3]:

x	y
0	8.3
1	11
2	14.7
3	19.7
4	26.7
5	35.2
6	44.4
7	55.9
8	68.9
9	83.2
10	98.8
11	114.2
12	127.1
13	140.1
14	164
15	190.9
16	214.3

■ *SOLUTION*

Defining the $\mathbf{x}$ and $\mathbf{y}$ vectors,

```
x = 0:1:16;
y = [8.3,11,14.7,19.7,26.7,35.2,44.4,55.9,68.9,83.2,98.8,114.2,127.1,...
    140.1,164,190.9,214.3];
```

To find the coefficients for the exponential least squares fit,

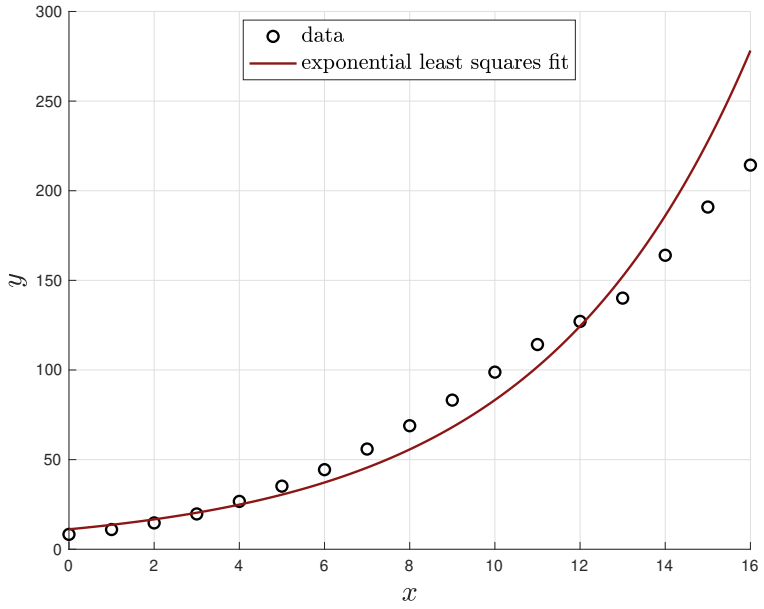
```
[a,b] = least_squares_fit(x,y,'exp');
```

Defining the vectors to calculate/plot the exponential fit $y = ae^{bx}$,

```
x_fit = min(x):((max(x)-min(x))/1000):max(x);
y_fit = a*exp(b*x_fit);
```

To plot,

```
figure;
hold on;
plot(x,y,'ko','markersize',7,'linewidth',1.5);
plot(x_fit,y_fit,'color',[140,21,21]/255,'linewidth',1.5);
hold off;
grid on;
xlabel('$x$','interpreter','latex','fontsize',18);
ylabel('$y$','interpreter','latex','fontsize',18);
legend('data','exponential least squares fit','interpreter','latex',...
    'fontsize',14,'location','best');
```



Example 5

Fit a logarithmic curve to the following data [4]:

x	y
1	10
7	19
20	30
50	35
79	51

■ *SOLUTION*

Defining the $\mathbf{x}$ and $\mathbf{y}$ vectors,

```
x = [1,7,20,50,79];
y = [10,19,30,35,51];
```

To find the coefficients for the logarithmic least squares fit,

```
[a,b] = least_squares_fit(x,y,'log');
```

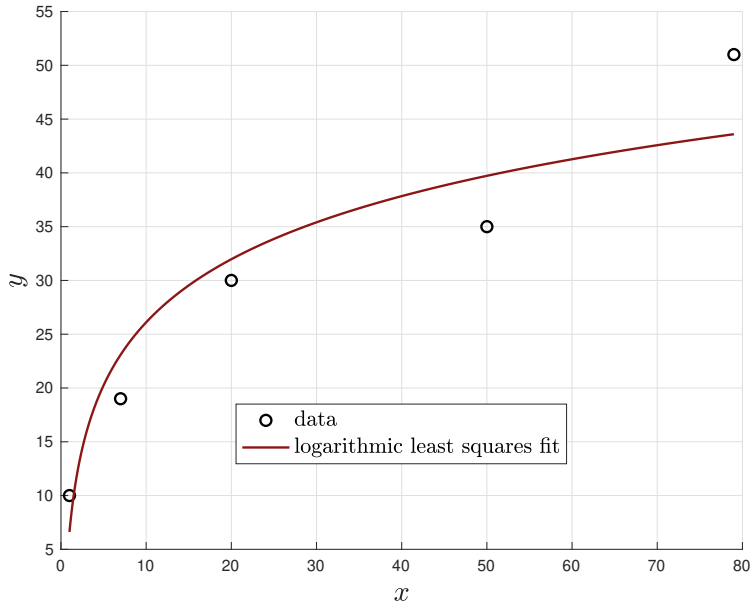
Defining the vectors to calculate/plot the logarithmic fit $y = a + b \ln x$,

```
x_fit = min(x):((max(x)-min(x))/1000):max(x);
y_fit = a+b*log(x_fit);
```

To plot,

```
figure;
hold on;
```

```
plot(x,y,'ko','markersize',7,'linewidth',1.5);
plot(x_fit,y_fit,'color',[140,21,21]/255,'linewidth',1.5);
hold off;
grid on;
xlabel('$x$','interpreter','latex','fontsize',18);
ylabel('$y$','interpreter','latex','fontsize',18);
legend('data','logarithmic least squares fit','interpreter','latex',...
    'fontsize',14,'location','best');
```



Links

GitHub®:
https://github.com/tamaskis/least_squares_fit-MATLAB

Least Squares Fitting

Normal Equations and the Least Squares Solution

Let S be a subspace of $\mathbb{R}^m$. For each $\mathbf{x} \in \mathbb{R}^m$, there is a unique element $\mathbf{p}$ of S that is closest to $\mathbf{x}$, that is $\|\mathbf{x} - \mathbf{y}\| > \|\mathbf{x} - \mathbf{p}\|$ for any $\mathbf{y} \neq \mathbf{p}$ in S. Furthermore, a given vector $\mathbf{p} \in S$ will be closest to a given vector $\mathbf{x} \in \mathbb{R}^m$ if and only if $\mathbf{x} - \mathbf{p} \in S^\perp$.

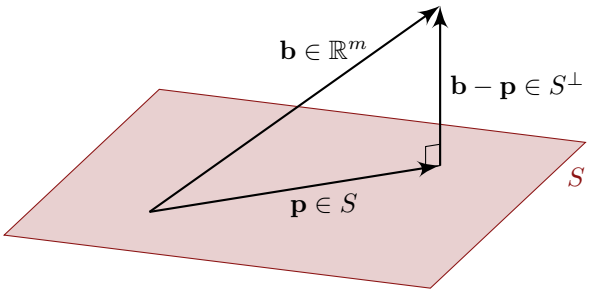


Figure 1: Closest vector $\mathbf{p} \in S$ to a vector $\mathbf{x} \in \mathbb{R}^m$.

Slightly reworded, the vector $\mathbf{p} \in S$ that best approximates (i.e. is closest to) a vector $\mathbf{x} \in \mathbb{R}^m$ is orthogonal to a vector $\mathbf{x} - \mathbf{p} \in S^\perp$. This is important for the next scenario, where we consider the linear system $\mathbf{Ax} = \mathbf{x}$. If $\mathbf{x}$ is *not* in the column space of A, that is $\mathbf{x} \notin R(\mathbf{A})$, then we know that $\mathbf{Ax} = \mathbf{x}$ is inconsistent. Nonetheless, we still wish to find a vector $\hat{\mathbf{x}}$ that best approximates the solution to $\mathbf{Ax} = \mathbf{x}$. Since $\mathbf{Ax} \in R(\mathbf{A})$ and $\mathbf{x} \notin R(\mathbf{A})$, we know that the $\mathbf{Ax}$ that best approximates $\mathbf{x}$ must satisfy $\mathbf{x} - \mathbf{Ax} \in R(\mathbf{A})^\perp$ (this is essentially the same scenario as before, where $\mathbf{Ax}$ takes the place of $\mathbf{p}$ and $R(\mathbf{A})$ takes the places of S.

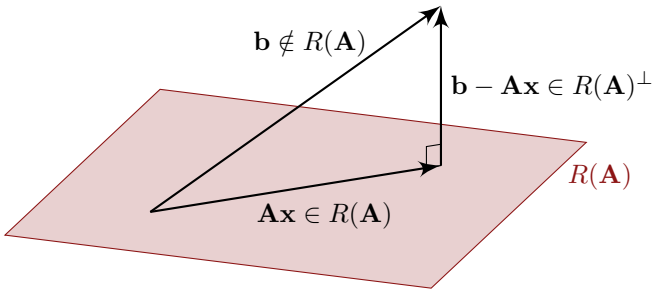


Figure 2: Closest vector $\mathbf{Ax} \in R(\mathbf{A})$ to a vector $\mathbf{x} \notin R(\mathbf{A})$.

Since $\mathbf{x} - \mathbf{Ax} \in R(\mathbf{A})^\perp$, we know $\mathbf{x} - \mathbf{Ax} \in N(\mathbf{A}^T)$, where $N(\mathbf{A}^T)$ is the null space of $\mathbf{A}^T$. Recall that $N(\mathbf{A}^T) = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}^T\mathbf{x} = \mathbf{0}\}$. Therefore, if $\mathbf{x} - \mathbf{Ax} \in N(\mathbf{A}^T)$, then

$$\mathbf{A}^T(\mathbf{x} - \mathbf{Ax}) = \mathbf{0}$$

$$\mathbf{A}^T\mathbf{x} - \mathbf{A}^T\mathbf{Ax} = \mathbf{0}$$

$$\boxed{\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{x}} \tag{1}$$

Eq. (1) represents a system of equations which are collectively known as the **normal equations**. To solve for $\mathbf{x}$, we can note that $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{A} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Therefore, multiplying both sides of Eq. (1) from the left by $(\mathbf{A}^T\mathbf{A})^{-1}$,

$$(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Ax} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{x}$$

$$\mathbf{I}\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{x}$$

$$\boxed{\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{x}} \tag{2}$$

$\hat{\mathbf{x}}$ is referred to as the **least squares** solution [7].

Polynomial Data Fitting

A polynomial p_n of degree n is defined as

$$p_n(x) = c_0 + c_1x + c_2x^2 + \ldots + c_nx^n$$

Let's say we have a set of data $\{(x_1, y_1), ..., (x_m, y_m)\}$. We wish to find a polynomial of degree n that best approximates the data set. For an arbitrary x, we have $p_n(x) = c_0 + c_1x + c_2x^2 + \ldots + c_nx^n$. Therefore, we can form the linear system

$$\begin{aligned}
c_0 + c_1x_1 + c_2x_1^2 + \ldots + c_nx_1^n &= y_1 \\
c_0 + c_1x_2 + c_2x_2^2 + \ldots + c_nx_2^n &= y_2 \\
&\vdots \\
c_0 + c_1x_m + c_2x_m^2 + \ldots + c_nx_m^n &= y_m
\end{aligned}$$

Writing this linear system in matrix form,

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^n \\ 1 & x_2 & x_2^2 & \ldots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \ldots & x_m^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Let

$$\boxed{\mathbf{A} = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^n \\ 1 & x_2 & x_2^2 & \ldots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \ldots & x_m^n \end{bmatrix}} \tag{3}$$

$$\boxed{\mathbf{c} = (c_0, ..., c_n)^T} \tag{4}$$

$$\boxed{\mathbf{y} = (y_1, ..., y_m)^T} \tag{5}$$

Then the linear system can be written simply as $\mathbf{A}\mathbf{c} = \mathbf{y}$ where $\mathbf{A} \in \mathbb{R}^{m\times(n+1)}$, $\mathbf{c} \in \mathbb{R}^{n+1}$, and $\mathbf{y} \in \mathbb{R}^m$. In general, the number of data points (m) is far greater than the number of coefficients corresponding to an n^{th}-degree polynomial ($n + 1$). Since $m > n + 1$, the linear system $\mathbf{A}\mathbf{c} = \mathbf{y}$ is overdetermined. From the previous section, we know that the best approximation of the solution to an overdetermined linear system is the least squares solution. From that section, we also know that the least squares solution $\hat{\mathbf{x}}$ to an overdetermined linear system can be found by solving the corresponding normal equations for that linear system. To form the normal equations corresponding to $\mathbf{A}\mathbf{c} = \mathbf{y}$, we just need to multiply both sides by $\mathbf{A}^T$ from the left.

$$\mathbf{A}^T\mathbf{A}\mathbf{c} = \mathbf{A}^T\mathbf{y}$$

Solving this linear system for $\hat{\mathbf{c}}$,

$$\boxed{\hat{\mathbf{c}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}} \tag{6}$$

We can then extract the **least squares coefficients** $c_0, ..., c_n$ from the vector $\hat{\mathbf{c}}$ [7].

Algorithm 1 below outlines how to obtain a polynomial least squares fit to a set of data.

Algorithm 1: Fitting a polynomial to a data set.

Given: $\mathbf{x}$, $\mathbf{y}$, n (where $\mathbf{x}$ and $\mathbf{y}$ store the data set $\{(x_i, y_i)\}_{i=1}^m$)
Preallocate the matrix $\mathbf{A} \in \mathbb{R}^{m\times(n+1)}$.

// populate the matrix $\mathbf{A}$
for $i = 1$ **to** m **do**
  for $j = 1$ **to** $n + 1$ **do**
    $A_{i,j} = x_i^{j-1}$
  end
end

// obtains the least squares solution to the normal equations
$\hat{\mathbf{c}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$

// calculate least squares fitted polynomial (of degree n*), where* $\hat{\mathbf{c}} = (c_0, ..., c_n)^T$
$y = c_0 + c_1x + c_2x^2 + \cdots + c_nx^n$

Linearization [1, 6]

Oftentimes, we have data sets that are not well approximated by polynomial functions. In these cases, we may try to linearize the data set. There are three types of relations we wish to linearize (they are the most common):

1. Power Relation: $y = ax^b$
2. Exponential Relation: $y = ae^{bx}$
3. Logarithmic Relation: $y = a + b\ln x$

Power Relation

Lets begin with the power relation. Taking the natural logarithm of both sides,

$$\ln y = \ln ax^b \quad \rightarrow \quad \ln y = \ln a + \ln x^b \quad \rightarrow \quad \ln y = \ln a + b\ln x$$

$$\boxed{y = ax^b \quad \Longleftrightarrow \quad \ln y = \ln a + b\ln x} \tag{7}$$

Eq. (7) represents a 1st-order polynomial (i.e. a linear function) and can thus be written as

$$y = c_0 + c_1x$$

if we perform the change of variables

$$y = \ln y, \quad x = \ln x$$

It follows that

$$\ln a = c_0 \quad \rightarrow \quad \boxed{a = e^{c_0}} \tag{8}$$

$$\boxed{b = c_1} \tag{9}$$

Below, we outline the procedure for applying a power fit to a data set $\{(x_i, y_i)\}_{i=1}^m$.

Algorithm 2: Fitting a power relation to a data set.

Given: $\mathbf{x}$, $\mathbf{y}$ (storing the data set $\{(x_i, y_i)\}_{i=1}^m$)

// calculate the natural logarithm of all the x_i's and y_i's

$\mathbf{x} = \ln \mathbf{x}$

$\mathbf{y} = \ln \mathbf{y}$

Find the coefficient vector $\hat{\mathbf{c}}$ for the linear least squares fit to the linearized data using Algorithm 1 with $n = 1$.

// calculates the power relation constants a and b from the coefficient vector $\hat{\mathbf{c}} = (c_0, c_1)^T$

$a = e^{c_0}$

$b = c_1$

// calculates the power relation least squares fit

$y = ax^b$

Exponential Relation

We can take the same approach to linearize the exponential relation.

$$\ln y = \ln ae^{bx} \quad \rightarrow \quad \ln y = \ln a + \ln e^{bx} \quad \rightarrow \quad \ln y = \ln a + bx$$

$$\boxed{y = ae^{bx} \quad \Longleftrightarrow \quad \ln y = \ln a + bx} \tag{10}$$

Eq. (10) represents a 1st-order polynomial (i.e. a linear function) and can thus be written as

$$y = c_0 + c_1 x$$

if we perform the change of variables

$$y = \ln y$$

It follows that

$$\ln a = c_0 \quad \rightarrow \quad \boxed{a = e^{c_0}} \tag{11}$$

$$\boxed{b = c_1} \tag{12}$$

Below, we outline the procedure for applying an exponential fit to a data set $\{(x_i, y_i)\}_{i=1}^m$.

Algorithm 3: Fitting an exponential relation to a data set.

Given: $\mathbf{x}$, $\mathbf{y}$ (storing the data set $\{(x_i, y_i)\}_{i=1}^m$)

// calculate the natural logarithm of all the y_i's

$\mathbf{y} = \ln \mathbf{y}$

Find the coefficient vector $\hat{\mathbf{c}}$ for the linear least squares fit to the linearized data using Algorithm 1 with $n = 1$.

// calculates the exponential relation constants a and b from the coefficient vector $\hat{\mathbf{c}} = (c_0, c_1)^T$

$a = e^{c_0}$

$b = c_1$

// calculates the exponential relation least squares fit

$y = ae^{bx}$

Logarithmic Relation

Finally, we examine the logarithmic relation. We can note that unlike the other two fits, we do not need to linearize the logarithmic relation; it is already linear in $\ln x$:

$$y = a + b \ln x$$

Thus, it can be written as

$$y = c_0 + c_1 x$$

if we perform the change of variables

$$x = \ln x$$

It follows that

$$\boxed{a = c_0} \tag{13}$$

$$\boxed{b = c_1} \tag{14}$$

Below, we outline the procedure for applying a logarithmic fit to a data set $\{(x_i, y_i)\}_{i=1}^m$.

Algorithm 4: Fitting a logarithmic relation to a data set.

Given: $\mathbf{x}$, $\mathbf{y}$ (storing the data set $\{(x_i, y_i)\}_{i=1}^m$)

// calculate the natural logarithm of all the x_i's

$\mathbf{x} = \ln \mathbf{x}$

Find the coefficient vector $\hat{\mathbf{c}}$ for the linear least squares fit to the linearized data using Algorithm 1 with $n = 1$.

// calculates the logarithmic relation constants a and b from the coefficient vector $\hat{\mathbf{c}} = (c_0, c_1)^T$

$a = c_0$

$b = c_1$

// calculates the logarithmic relation least squares fit

$y = a + b \ln x$

References

[1] Henry Chan. *Checking Assumptions and Transforming Data*. MATH 2810 Lecture Notes (Vanderbilt University). (notes taken by Tamas Kis on April 17, 2020).

[2] *curve fitting a power function*. https://www.mathworks.com/matlabcentral/answers/65624-curve-fitting-a-power-function. (accessed: March 26, 2021).

[3] *Exponential Fit*. https://mste.illinois.edu/malcz/ExpFit/FIRST.html. (accessed: March 26, 2021).

[4] *How to do exponential and logarithmic curve fitting in Python? I found only polynomial fitting*. https://stackoverflow.com/questions/3433486/how-to-do-exponential-and-logarithmic-curve-fitting-in-python-i-found-only-poly. (accessed: March 26, 2021).

[5] *Least Squares Regression*. mathsisfun.com/data/least-squares-regression.html. (accessed: March 26, 2021).

[6] *Linearizing the Equation*. http://academic.macewan.ca/physlabs/Linearization.pdf. (accessed: June 14, 2020).

[7] Gieri Simonett. *5.3 Least Squares Problems*. MATH 2410 Lecture Notes (Vanderbilt University). (notes taken by Tamas Kis on May 27, 2018).