

## Numerical Example: Method of Least Squares<sup>1</sup>

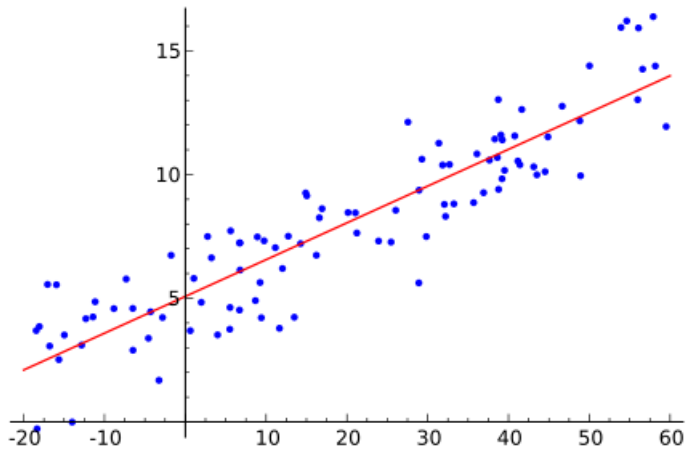
- The method of least squares is a standard approach to the approximate solution of **overdetermined** systems ( $m > n$ ).
- Let  $\{\hat{y}_i\}_{i=1}^n$  be the observed response values and  $\{y_i\}_{i=1}^n$  be the fitted response values.
- Let  $\epsilon_i = \hat{y}_i - y_i$  be the residual for  $i = 1, \dots, n$ .
- Then the sum of square residuals estimates associated with the data is given by

$$S = \sum_{i=1}^n \epsilon_i^2.$$

- The best fit in the least-squares sense **minimizes the sum of squared residuals**.

---

<sup>1</sup>See [https://en.wikipedia.org/wiki/Least\\_squares](https://en.wikipedia.org/wiki/Least_squares).



[https://commons.wikimedia.org/wiki/File:Linear\\_regression.svg](https://commons.wikimedia.org/wiki/File:Linear_regression.svg)

# Linear Least Squares

- The approach is called **linear** least squares since **the assumed function is linear in the parameters to be estimated**.
- For example, consider

$$y = ax + b,$$

where  $a$  and  $b$  are to be determined.

- Then we have  $\epsilon_i = (ax_i + b) - \hat{y}_i$  so that

$$S = \sum_{i=1}^n ((ax_i + b) - \hat{y}_i)^2.$$

- Now consider the partial derivatives of  $S$  with respect to  $a$  and  $b$ :

$$\frac{\partial S}{\partial a} = -2 \sum_{i=1}^n x_i (y_i - (ax_i + b)) = 0,$$

$$\frac{\partial S}{\partial b} = -2 \sum_{i=1}^n (y_i - (ax_i + b)) = 0.$$

- We reorganize the above equations as follows:

$$a \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i = \sum_{i=1}^n x_i y_i,$$

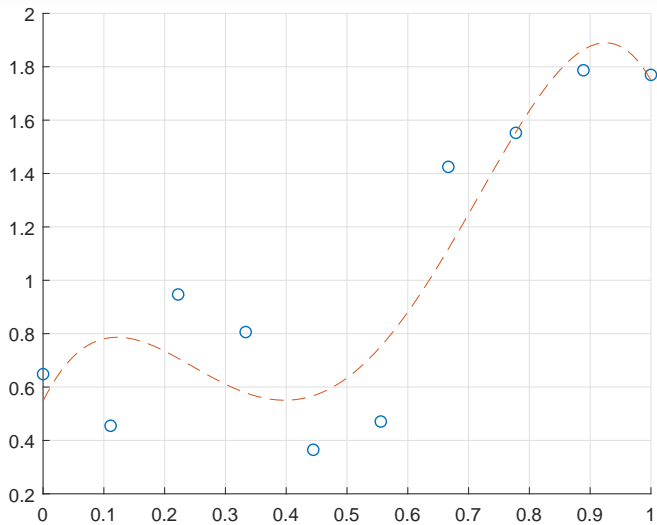
$$a \sum_{i=1}^n x_i + nb = \sum_{i=1}^n y_i.$$

- It could be done by using **normal equations**.<sup>2</sup>

<sup>2</sup>See [https://en.wikipedia.org/wiki/Linear\\_least\\_squares\\_\(mathematics\)#Derivation\\_of\\_the\\_normal\\_equations](https://en.wikipedia.org/wiki/Linear_least_squares_(mathematics)#Derivation_of_the_normal_equations).

## Example

```
1 clear; clc; close all;
2
3 rng(3); % fix the random seed
4 N = 10;
5 x = linspace(0, 1, N); x = x(:);
6 y = cos(rand(size(x)) * pi / 2) + x .^ 2;
7 figure; hold on; grid on; plot(x, y, 'o');
8 degree = 4;
9
10 M = @(x, degree) repmat(x, 1, degree + 1);
11 A = @(mat) bsxfun(@(x, i) x .^ i, mat, ...
12                 size(mat, 2) - 1 : -1 : 0);
13 pp = A(M(x, degree)) \ y % show the coefficients
14 xq = linspace(min(x), max(x), 100); xq = xq(:);
15 yq = A(M(xq, degree)) * pp;
16 plot(xq, yq, '--');
```



# Polynomial Regression

- **polyfit**( $x, y, n$ ) returns the coefficients for a polynomial of degree  $n$  that is a best fit for the set of sample data  $(x, y)$  (in a least-squares sense).

## Example

```
1 clear; clc; close all;
2
3 rng(3);
4 N = 10;
5 x = linspace(0, 1, N); x = x(:);
6 y = cos(rand(size(x)) * pi / 2) + x .^ 2;
7 figure; hold on; grid on; plot(x, y, 'o');
8 degree = 4;
9
10 p = polyfit(x, y, degree)
11 xq = linspace(0, 1, 50);
12 yq = polyval(p, xq);
13 plot(xq, yq);
```

- The result is identical to the figure shown before.



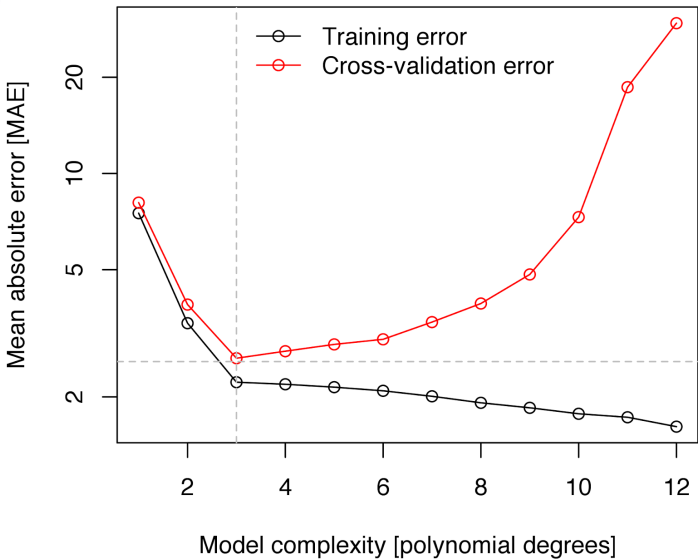
# Overfitting

- Overfitting is the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably.
  - In other words, the overfitted model is perfect to **in-sample** data but not robust in **out-of-sample** data.
  - For example, Runge's phenomenon.<sup>3</sup>
- Law of parsimony<sup>4</sup> states that **simpler solutions are more likely to be correct than complex ones.**

---

<sup>3</sup>See [https://en.wikipedia.org/wiki/Runge's\\_phenomenon](https://en.wikipedia.org/wiki/Runge's_phenomenon).

<sup>4</sup>Aka Occam's Razor.



# Polynomials

- Let  $n \in \mathbb{N} \cup \{0\}$ , and  $x, a_0, \dots, a_n \in \mathbb{R}$ .
- $f(x)$  is said to be a polynomial with degree  $n$  provided that

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0,$$

where  $a_n \neq 0$ .

- We often express a polynomial by its coefficient vector  $[a_n, a_{n-1}, \dots, a_0]$ .
- In fact, the set of polynomials with coefficients in  $\mathbb{R}$  is a **vector space** over  $\mathbb{R}$ , denoted by  $\mathbb{P}_n$ .<sup>5</sup>

---

<sup>5</sup>See [https://en.wikipedia.org/wiki/Examples\\_of\\_vector\\_spaces#Polynomial\\_vector\\_spaces](https://en.wikipedia.org/wiki/Examples_of_vector_spaces#Polynomial_vector_spaces).

## Arithmetic Operations of Polynomials

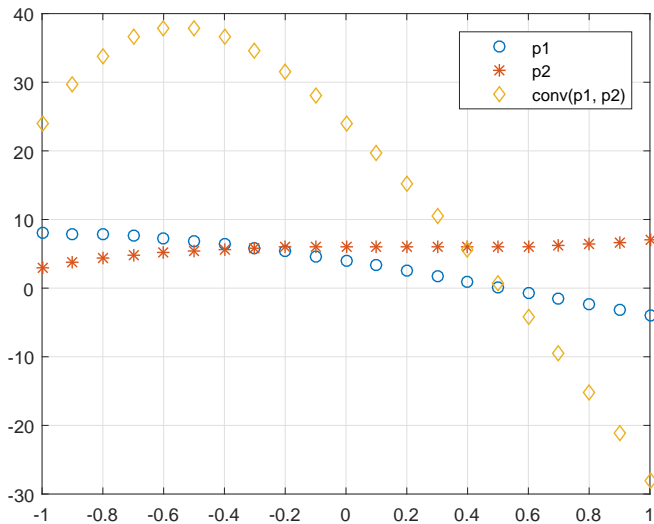
- Let  $p_1$  and  $p_2$  be two coefficient vectors of polynomials of the same degree.
- Then we have the following operations:
  - addition and subtraction:  $p_1 \pm p_2$ .
  - multiplication: **conv**( $p_1, p_2$ ).<sup>6</sup>
  - division:  $[q, r] = \mathbf{deconv}(p_1, p_2)$ .<sup>7</sup>
- Use **polyval**( $p_1, x$ ) to calculate the function values of  $p_1$  on  $x$ .

---

<sup>6</sup>See <http://en.wikipedia.org/wiki/Convolution>.

<sup>7</sup>Equivalently,  $v = \mathbf{conv}(u, q) + r$ . Also see [http://en.wikipedia.org/wiki/Euclidean\\_division](http://en.wikipedia.org/wiki/Euclidean_division).

```
1 clear; clc;
2
3 p1 = [1 -2 -7 4];
4 p2 = [2 -1 0 6];
5 p3 = p1 + p2
6 p4 = p1 - p2
7 p5 = conv(p1, p2)
8 [q, r] = deconv(p1, p2)
9
10 x = linspace(-1, 1, 20);
11 plot(x, polyval(p1, x), 'o', x, polyval(p2, x), ...
      '* ', x, polyval(p5, x), 'd');
12 grid on; legend('p1', 'p2', 'conv(p1, p2)');
```



## Finding Roots of Polynomial

- Use **roots**( $p$ ) for all roots of the polynomial  $p$ .<sup>8</sup>
- For example,

```
1 clear; clc; close all;
2
3 p = [1, 3, 1, 5, -1];
4 r = roots(p) % find all roots of p
5 polyval(p, r) % why not zeros?
```

---

<sup>8</sup>See [https://en.wikipedia.org/wiki/Jenkins-Traub\\_algorithm](https://en.wikipedia.org/wiki/Jenkins-Traub_algorithm).

## Exercise: Internal Rate of Return (IRR)<sup>9</sup>

- Consider two assets.
- For asset A, you are promised to receive the cash flows as follows:

$$C_0 = -100, C_1 = 0, C_2 = 0, C_3 = 120.$$

- For asset B, the cash flows are

$$C_0 = -100, C_1 = 6, C_2 = 6, C_3 = 108.$$

- Which asset is more desirable?

---

<sup>9</sup>See [https://en.wikipedia.org/wiki/Internal\\_rate\\_of\\_return](https://en.wikipedia.org/wiki/Internal_rate_of_return).



- Given a collection of pairs (time, cash flow), the IRR is a rate of return when the net present value is zero.
- Explicitly, the IRR can be calculated by solving

$$\sum_{i=0}^N \frac{C_i}{(1+r)^i} = 0,$$

where  $C_i$  is the cash flow at time  $i$ .

- So the IRR is 6.27% for A and 6.62% for B.

*“Time is free, but its priceless.  
You can't own it, but you can use it.  
You can't keep it, but you can spend it.  
Once you've lost it, you can never get it back.”*  
– Harvey MacKay

*“No man can achieve success  
if he didn't first know the value of time.”*  
– Sunday Adelaja

## Integral and Derivative of Polynomials

- **polyder**( $p$ ) returns the derivative of the polynomial  $p$ .
- **polyint**( $p, k$ ) returns a polynomial representing the integral of polynomial  $p$ , using a scalar constant of integration  $k$ .

```
1 clear; clc;
2
3 p = [4 3 2 1];
4 p_der = polyder(p)
5 p_int = polyint(p, 0) % k = 0
```

## Exercise

- Let  $p$  be the coefficient vector for any polynomial with degree 3.
- Write a program to calculate the coefficients of its derivative and integration.
- Also, you may write down the matrix representation of differentiation and integration of  $p$ .
- Do not use the built-in functions.

```

1 clear; clc;
2
3 p = randi(100, 1, 4)
4 q1 = [0, p(1 : end - 1) .* [length(p) - 1 : -1 : 1]]
5 q2 = [p ./ [length(p) : -1 : 1], 0]
6
7 T1 = [0 0 0 0;
8       3 0 0 0;
9       0 2 0 0;
10      0 0 1 0];
11 T1 * p'
12 T2 = [0 1/4 0 0 0;
13       0 0 1/3 0 0;
14       0 0 0 1/2 0;
15       0 0 0 0 1;
16       0 0 0 0 0];
17 T2 * [0 p]'

```

# Eigenvalues and Eigenvectors

- Let  $A \in \mathbf{M}_{n \times n}(\mathbb{R})$ ,  $I$  be the identity, and  $v \in \mathbb{R}^n$  be nontrivial.
- An eigenvalue problem<sup>10</sup> is a system which follows

$$Av = \lambda v.$$

- Then  $u$  is an eigenvector associated with the eigenvalue  $\lambda$  by solving  $\det(A - \lambda I) = 0$ , aka the **characteristic polynomial**.
  - Use  $[V, D] = \mathbf{eig}(A)$  produces a diagonal matrix  $D$  of eigenvalues and a full matrix  $V$  whose columns are the corresponding eigenvectors so that  $AV = VD$ .

---

<sup>10</sup>See [https://en.wikipedia.org/wiki/Eigenvalues\\_and\\_eigenvectors#Applications](https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors#Applications).

## Example: Google PageRank Algorithm<sup>11</sup>

- PageRank (PR) is an algorithm used by Google Search to rank web pages in their search engine results.
- PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is.
- The underlying assumption is that **more important websites are likely to receive more links from other websites.**

---

<sup>11</sup>Larry Page and Sergey Brin (1998).

## Singular Value Decomposition (SVD)

- Let  $A \in \mathbf{M}_{m \times n}(\mathbb{R})$ ,  $u \in \mathbb{R}^m$ ,  $v \in \mathbb{R}^n$ , and  $\sigma \in \mathbb{R}$ .
- $\sigma$  is called a singular value associated with the left singular vector  $u$  and the right singular vector  $v$  for  $A$  provided that

$$A = \sum_{i=1}^n \sigma_i u_i v_i^T.$$

- In matrix form,

$$A = U \Sigma V^T,$$

where  $U$  and  $V$  consist of the left and right singular vectors, respectively, and  $\Sigma$  is a diagonal matrix whose diagonal entries are the singular values of  $A$ .

- Use  $[U, S, V] = \mathbf{svd}(A)$  for SVD.<sup>12</sup>

---

<sup>12</sup>See



## Example: Image Compression by Low-Rank Approximation

- We can have an image extremely similar to the original one, but with a smaller image size by keeping the vectors associated with a few number of first large principal components, aka **Principal Component Analysis** (PCA).<sup>13</sup>
- PCA can be done by **svd**.

---

<sup>13</sup>See <http://setosa.io/ev/principal-component-analysis/>.