Applied Mathematics 205

Unit I: Data Fitting

Lecturer: Dr. David Knezevic

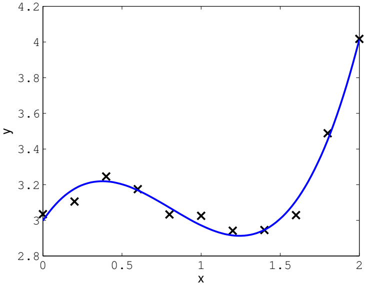
Unit I: Data Fitting

Chapter $\mathrm{I}.3$: Linear Least Squares

The Problem Formulation

Recall that it can be advantageous to not fit data points exactly (e.g. due to experimental error), we don't want to "overfit"

Suppose we want to fit a cubic polynomial to 11 data points



Question: How do we do this?

The Problem Formulation

Suppose we have m constraints and n parameters with $m > n$ (e.g. $m = 11$, $n = 4$ on previous slide)

In terms of linear algebra, this is an overdetermined system $Ab = y$, where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$ (parameters), $y \in \mathbb{R}^m$ (data)

$$\begin{bmatrix} & & \\ & A & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} b \end{bmatrix} = \begin{bmatrix} \\ y \\ \\ \\ \end{bmatrix}$$

i.e. we have a "tall, thin" matrix A

The Problem Formulation

In general, cannot be solved exactly (hence we will write $Ab \simeq y$); instead our goal is to minimize the residual, $r(b) \in \mathbb{R}^m$

$$r(b) \equiv y - Ab$$

A highly effective approach for this is the method of least squares:[1]
Find parameter vector $b \in \mathbb{R}^n$ that minimizes $\|r(b)\|_2$

As we shall see, we use the 2-norm since it gives us a differentiable function to minimize (can then use calculus)

[1] Developed by Gauss and Legendre for fitting astronomical observations with experimental error

The Normal Equations

Goal is to minimize $\|r(b)\|_2$, recall that $\|r(b)\|_2 = \sqrt{\sum_{i=1}^{n} r_i(b)^2}$

The minimizing b is the same for $\|r(b)\|_2$ and $\|r(b)\|_2^2$, hence we consider the differentiable "objective function" $\phi(b) = \|r(b)\|_2^2$

$$
\begin{aligned}
\phi(b) &= \|r\|_2^2 = r^T r = (y - Ab)^T (y - Ab) \\
&= y^T y - y^T Ab - b^T A^T y + b^T A^T Ab \\
&= y^T y - 2b^T A^T y + b^T A^T Ab
\end{aligned}
$$

where last line follows from $y^T Ab = (y^T Ab)^T$, since $y^T Ab \in \mathbb{R}$

ϕ is a quadratic function of b, and is non-negative, hence a minimum must exist, (but not nec. unique, e.g. $f(b_1, b_2) = b_1^2$)

The Normal Equations

To find minimum of $\phi(b) = y^T y - 2b^T A^T y + b^T A^T A b$,
differentiate wrt b and set to zero

Differentiate $b^T A^T y$ wrt b: Let $c \equiv A^T y \in \mathbb{R}^n$

$$b^T c = \sum_{i=1}^{n} b_i c_i \implies \nabla(b^T c) = c \implies \nabla(b^T A^T y) = A^T y$$

The Normal Equations

Note that $A^T A$ is symmetric

Consider $b^T M b$ for symmetric matrix $M \in \mathbb{R}^{n \times n}$

$$b^T M b = b^T \left(\sum_{j=1}^{n} m_{(:,j)} b_j \right)$$

From the product rule

$$\begin{aligned}
\frac{\partial}{\partial b_k}(b^T M b) &= e_k^T \sum_{j=1}^{n} m_{(:,j)} b_j + b^T m_{(:,k)} \\
&= \sum_{j=1}^{n} m_{(k,j)} b_j + b^T m_{(:,k)} \\
&= m_{(k,:)} b + b^T m_{(:,k)} \\
&= 2 m_{(k,:)} b,
\end{aligned}$$

where the last line follows from symmetry of M, and therefore

$$\nabla(b^T M b) = 2Mb, \qquad \text{so that} \qquad \nabla(b^T A^T A b) = 2A^T A b$$

The Normal Equations

Putting it all together, we obtain

$$\nabla\phi(b) = -2A^T y + 2A^T Ab$$

We set $\nabla\phi(b) = 0$ to obtain

$$-2A^T y + 2A^T Ab = 0 \Longrightarrow A^T Ab = A^T y$$

The square system $A^T Ab = A^T y$ is known as the normal equations

The Normal Equations

For $A \in \mathbb{R}^{m \times n}$ with $m > n$, $A^T A$ is singular if and only if A is rank-deficient.[2]

Proof:

($\Rightarrow$) Suppose $A^T A$ is singular. $\exists z \neq 0$ such that $A^T A z = 0$. Hence $z^T A^T A z = \|Az\|_2^2 = 0$, so that $Az = 0$. Therefore A is rank-deficient.

($\Leftarrow$) Suppose A is rank-deficient. $\exists z \neq 0$ such that $Az = 0$, hence $A^T A z = 0$, so that $A^T A$ is singular.

[2] Recall $A \in \mathbb{R}^{m \times n}$, $m > n$ is rank-deficient if columns are not L.I., i.e. $\exists z \neq 0$ s.t. $Az = 0$

The Normal Equations

If A has full rank we can solve the normal equations to find b

Generally it is a bad idea to solve the normal equations directly, since $\mathrm{cond}(A^T A) = \mathrm{cond}(A)^2$

We will discuss better methods (QR, SVD) next Unit that do not square the condition number

Matlab "backslash"

"Backslash" ($\backslash$) is one of the most useful operators in Matlab

"Overloaded" to do different calculations in different contexts

Most standard situation is "solve $Ax = b$" via $x = A\backslash b$ for square system (uses LU decomposition, cf. Unit II)

If system is over-determined, "backslash" finds least squares solution (uses QR factorization, cf. Unit II)

Matlab "backslash"

Find least-squares fit for degree 11 polynomial to 50 samples of $y = \cos(4x)$ for $x \in [0, 1]$

```
format long
x = linspace(0,1,50)';
A = fliplr(vander(x));
A = A(:,1:12);
y = cos(4*x);

% solve normal equations
fprintf('cond(A'*A) = %d\n\n ', cond(A'*A))
b_normal = (A'*A) \ (A'*y)

% solve using 'backslash' (less rounding error)
b_backslash = A \ y
```

Matlab "backslash"

$\texttt{cond}(A^T A) = 1.354 \times 10^{16}$

$$b_{\rm normal} = \begin{bmatrix} 1.000000051508329 \\ -0.000015133093351 \\ -7.999431402147580 \\ -0.008391428014185 \\ 10.731053092678904 \\ -0.291236426826351 \\ -4.862157012040036 \\ -1.510203667008908 \\ 3.386344100793780 \\ -1.238285407662096 \\ 0.144069879639166 \\ -0.005390299320099 \end{bmatrix}, \quad b_{\rm backslash} = \begin{bmatrix} 1.000000000996605 \\ -0.000000422742734 \\ -7.999981235694049 \\ -0.000318763130923 \\ 10.669430795224949 \\ -0.013820285245551 \\ -5.647075634537363 \\ -0.075316011985823 \\ 1.693606949690125 \\ 0.006032118434138 \\ -0.374241707313253 \\ 0.088040576742115 \end{bmatrix}$$

Error[3] in $b_{\rm normal} = O(1)$, i.e. the map $y \to b$ using the Normal equations is ill-conditioned

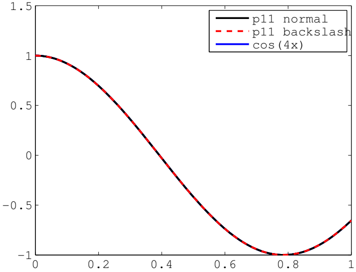
[3] With respect to $b_{\rm backslash}$

Matlab "backslash"

But solving the normal equations still yields a small residual, hence we obtain a good fit to the data

$$\|r(b_{\text{normal}})\|_2 = \|y - Ab_{\text{normal}}\|_2 = 2.24 \times 10^{-7}$$

$$\|r(b_{\text{backslash}})\|_2 = \|y - Ab_{\text{backslash}}\|_2 = 8.00 \times 10^{-9}$$



We will discuss the distinction between *small residual* and *small error* in Unit II

Non-polynomial Least-squares fitting

Note that so far we have exclusively used polynomials, for interpolation and for least-squares fitting

- Polynomials are a popular choice since they are good for approximating general functions[4]
- Also, appropriate for interpolation since we know that a unique degree n polynomial interpolates $n+1$ data points

However, we can use other functions for linear least-squares: we just need the model to depend linearly on parameters

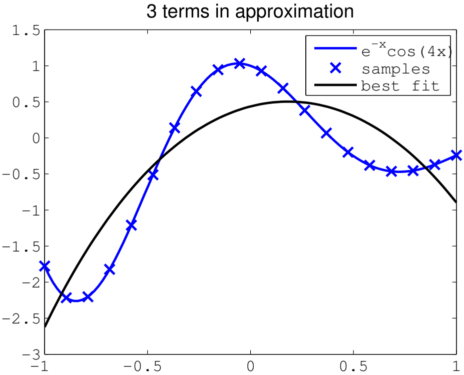
e.g. let us approximate $e^{-x}\cos(4x)$ using $f_n(x; b) \equiv \sum_{k=-n}^{n} b_k e^{kx}$

(Note that f_n is linear in b: $f_n(x; \gamma a + \sigma b) = \gamma f_n(x; a) + \sigma f_n(x; b)$)

---

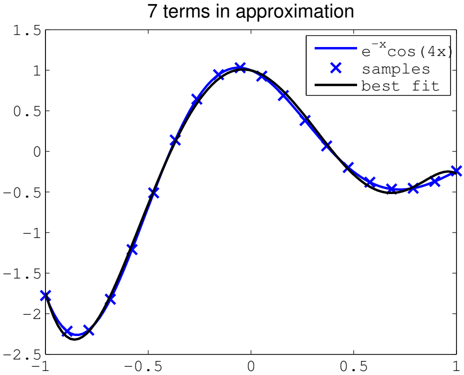
[4]Weierstrass Approximation Theorem: for any $f \in C[a, b]$, $\|f - p_n^*\|_\infty \to 0$ as $n \to \infty$, where p_n^* is best polynomial approximation in $\mathbb{P}_n[a, b]$

Non-polynomial Least-squares fitting



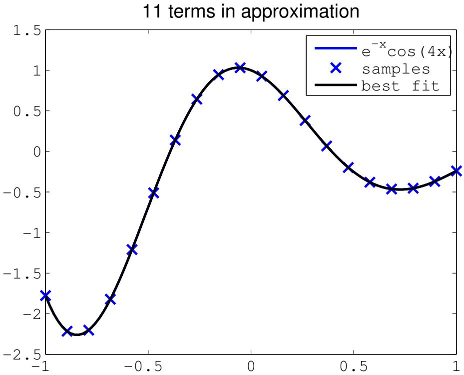
$$n = 1, \quad \frac{\|r(b)\|_2}{\|b\|_2} = 4.16 \times 10^{-1}$$

Non-polynomial Least-squares fitting



$$n = 3, \quad \frac{\|r(b)\|_2}{\|b\|_2} = 1.44 \times 10^{-3}$$

Non-polynomial Least-squares fitting



$$n = 5, \quad \frac{\|r(b)\|_2}{\|b\|_2} = 7.46 \times 10^{-6}$$

Pseudoinverse

The normal equations also motivate the idea of the "pseudoinverse"[5] A^+, (`pinv(A)` in Matlab)

The pseudoinverse can be defined differently in different contexts, for overdetermined least-squares we have (cf. Normal equations)[6]

$$A^+ \equiv (A^T A)^{-1} A^T$$

- If A is invertible, then $A^+ = A^{-1}$ i.e. "generalized inverse"

Proof: $A^+ = (A^T A)^{-1} A^T = A^{-1} (A^T)^{-1} A^T = A^{-1}$

- $A^+ A = I$, but $AA^+ \neq I$ in general (this is a "left inverse")

Least-squares solution is given by $b = A^+ y$, $A^+ \in \mathbb{R}^{n \times m}$

[5] Often called Moore-Penrose pseudoinverse

[6] Recall that if A has full rank, then $A^T A$ is invertible

Underdetermined Least Squares

So far we have focused on overconstrained systems (more constraints than parameters)

But least-squares also applies to underconstrained systems: $Ab = y$ with $A \in \mathbb{R}^{m \times n}$, $m < n$

$$\left[\quad A \quad \right] \left[\; b \; \right] = \left[\; y \; \right]$$

i.e. we have a "short, fat" matrix A

Underdetermined Least Squares

For $\phi(b) = \|r(b)\|_2^2 = \|y - Ab\|_2^2$, from $\nabla \phi = 0$ we again obtain

$$A^T A b = A^T y$$

But now $A^T A \in \mathbb{R}^{n \times n}$, but has rank m (where $m < n$), hence $A^T A$ must be singular

There are infinitely many solutions, need to be able to select one of them

Underdetermined Least Squares

First idea, pose as a constrained optimization problem to find the feasible b with minimum norm:

$$\begin{array}{ll} \text{minimize} & b^T b \\ \text{subject to} & Ab = y \end{array}$$

This can be treated using Lagrange multipliers (we will not discuss this now, see Unit IV)

The Lagrange multiplier approach for the constrained optimization problem yields the "minimum norm" least-squares solution

$$b = A^T(AA^T)^{-1}y$$

Hence in the underdetermined case, the pseudoinverse is defined as $A^+ = A^T(AA^T)^{-1} \in \mathbb{R}^{n\times m}$

- $AA^+ = I$, but $A^+A \neq I$ in general (this is a "right inverse")

Underdetermined Least Squares

Alternative approach that does not require Lagrange multipliers:
Recall that we solve for b by minimizing ϕ

Let's modify ϕ so that there is a unique minimum

For example, let

$$\phi(b) \equiv \|r(b)\|_2^2 + \|Sb\|_2^2$$

where $S \in \mathbb{R}^{n \times n}$ is a scaling matrix

This is called regularization: we make the problem well-posed ("more regular") by modifying the objective function

Underdetermined Least Squares

Calculating $\nabla\phi = 0$ in the same way as before leads to the system

$$(A^T A + S^T S)b = A^T y$$

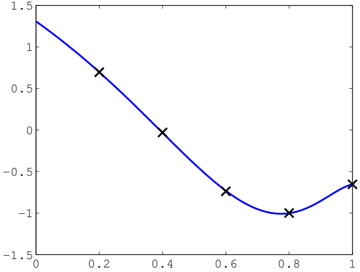
We need to choose S in some way to ensure $(A^T A + S^T S)$ is invertible

Simplest option: $S = \mu \mathrm{I} \in \mathbb{R}^{n\times n}$ for $\mu \in \mathbb{R}$

Underdetermined Least Squares

Find least-squares fit for degree 11 polynomial to 5 samples of $y = \cos(4x)$ for $x \in [0, 1]$, $\text{cond}(A^T A) = 4.78 \times 10^{17}$

Try $S = 0.001\mathrm{I}$ (i.e. $\mu = 0.001$)



$$\|r(b)\|_2 = 1.07 \times 10^{-4}$$

$$\|b\|_2 = 4.40$$

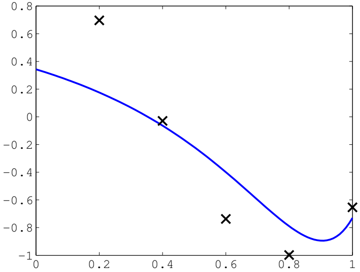
$$\text{cond}(A^T A + S^T S) = 1.54 \times 10^7$$

Fit is good since regularization term is small (but big enough to guarantee well-posedness)

Underdetermined Least Squares

Find least-squares fit for degree 11 polynomial to 5 samples of $y = \cos(4x)$ for $x \in [0, 1]$

Try $S = 0.5\mathrm{I}$ (i.e. $\mu = 0.5$)



$$\|r(b)\|_2 = 6.60 \times 10^{-1}$$

$$\|b\|_2 = 1.15$$

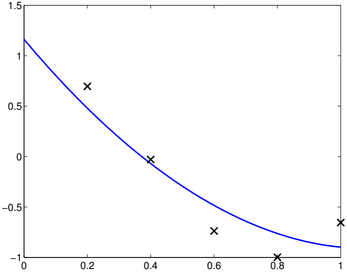
$$\mathrm{cond}(A^T A + S^T S) = 62.3$$

Reg. term is too big, not enough incentive to fit the data well!
(But we reduce $\|b\|_2$ further)

Underdetermined Least Squares

Find least-squares fit for degree 11 polynomial to 5 samples of $y = \cos(4x)$ for $x \in [0, 1]$

Try $S = \texttt{diag}(0.1, 0.1, 0.1, 10, 10\ldots, 10)$



$$\|r(b)\|_2 = 4.78 \times 10^{-1}$$

$$\|b\|_2 = 4.27$$

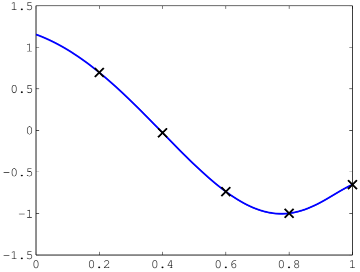
$$\mathrm{cond}(A^T A + S^T S) = 5.90 \times 10^3$$

We strongly penalize $b_3, b_4, \ldots, b_{11}$, hence the fit is close to parabolic

Underdetermined Least Squares

Find least-squares fit for degree 11 polynomial to 5 samples of $y = \cos(4x)$ for $x \in [0, 1]$

Try using Matlab's "backslash"



$$\|r(b)\|_2 = 1.03 \times 10^{-15}$$

$$\|b\|_2 = 7.18$$

"Backslash" employs Lagrange multiplier based pseudoinverse, hence satisfies the constraints to machine precision