9. Least squares

- least squares problem
- solution and normal equations
- multi-objective least squares
- control
- estimation and inversion

Least squares problem

- let A be $m \times n$ and consider $Ax = b$ where b is an m-vector
- in most applications, $m > n$ and there is no x that satisfies $Ax = b$

Least squares problem: choose x that minimizes the residual norm $r = Ax - b$:

$$\text{minimize} \quad \|Ax - b\|^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} A_{ij} x_j - b_i \right)^2$$

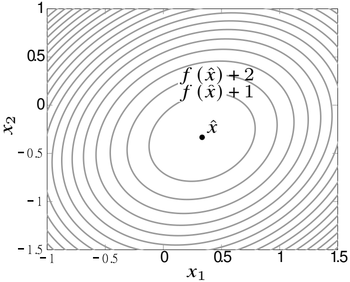
- x is *variable*, A, b are called *data*, $\|Ax - b\|^2$ is the *objective* function
- also called *regression* (in data fitting context)
- $\hat{x}$ is a *solution* of the least squares problem if

$$\|A\hat{x} - b\|^2 \leq \|Ax - b\|^2 \quad \text{for any } n\text{-vector } x$$

  - $\hat{x}$ also called *least-squares approximate solution* of $Ax = b$
  - if $\hat{r} = A\hat{x} - b = 0$, then $\hat{x}$ solves linear equation $Ax = b$

Example

$$A = \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$



- $Ax = b$ has no solution
- least squares problem:

$$\text{minimize} \quad \|Ax - b\|^2 = (2x_1 - 1)^2 + (-x_1 + x_2)^2 + (2x_2 + 1)^2$$

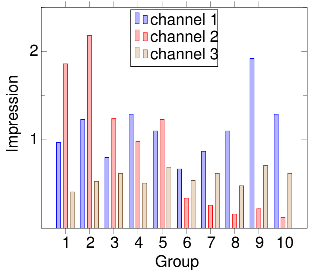
- least squares solution is $\hat{x} = (1/3, -1/3)$
- $\|A\hat{x} - b\|^2 = 2/3$ is smallest posible value of $\|Ax - b\|^2$

Example: Advertising purchases

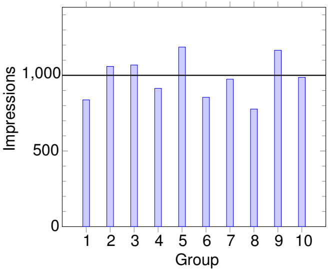
- m demographics groups (audiences), n advertising channels
- v_i^{des} is target number of views or impressions for group i
- R_{ij} is # views in group i per dollar spent on ads in channel j
- s_j is amount of advertising purchased in channel j
- $(Rs)_i$ is total number of views in group i
- least squares problem: minimize $\|Rs - v^{\text{des}}\|^2$ (ignoring $s \geq 0$ and budget)

Example: $m = 10$, $n = 3$, $v^{\text{des}} = 10^3 \times \mathbf{1}$, $\hat{s} = (62, 100, 1443)$

columns R (1000 views per dollar)



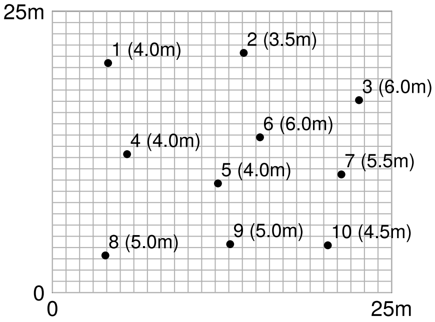
v^{des} and achieved views $R\hat{s}$



Example: Illumination

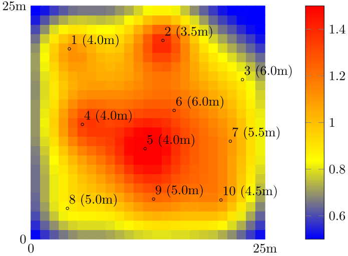
- n lamps illuminate an area divided in m regions
- b_i is target illumination level at region i
- x_j is power of lamp j
- A_{ij} is illumination in region i if lamp j is on with power 1, other lamps are off
- $(Ax)_i$ is illumination level at region i

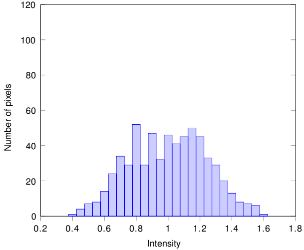
Example: lamp positions and heights with $m = 25 \times 25$, $n = 10$



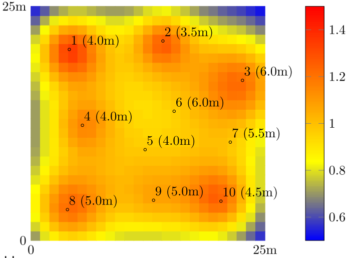
Illumination

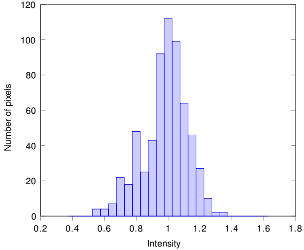
equal lamp powers $(x = \mathbf{1})$





least squares solution $\hat{x}$, with $b = \mathbf{1}$





Outline

- least squares problem
- **solution and normal equations**
- multi-objective least squares
- control
- estimation and inversion

Least squares solution

$$\text{minimize} \quad \|Ax - b\|^2$$

Normal equations: a solution $\hat{x}$ must satisfy the *normal equations:*

$$A^T A \hat{x} = A^T b$$

and if A has *linearly independent columns*, then the solution is unique

$$\hat{x} = (A^T A)^{-1} A^T b = A^\dagger b$$

- $A^\dagger = (A^T A)^{-1} A^T$ is the psuedo-inverse of A, which is also a left inverse
- $\hat{x} = A^\dagger b$ solves the linear equation $Ax = b$ if it has a solution
- if $Ax = b$ does not have a solution, then $A\hat{x} \neq b$

Proof using algebra

suppose $\hat{x}$ satisfies the normal equations $A^T(A\hat{x} - b) = 0$, then for any n-vector x

$$\begin{aligned}
\|Ax - b\|^2 &= \|(Ax - A\hat{x}) + (A\hat{x} - b)\|^2 \\
&= \|A(x - \hat{x})\|^2 + \|A\hat{x} - b\|^2 + 2(A(x - \hat{x}))^T(A\hat{x} - b) \\
&= \|A(x - \hat{x})\|^2 + \|A\hat{x} - b\|^2 + 2(x - \hat{x})^T A^T(A\hat{x} - b) \\
&= \|A(x - \hat{x})\|^2 + \|A\hat{x} - b\|^2
\end{aligned}$$

- hence for any x, $\|Ax - b\|^2 \geq \|A\hat{x} - b\|^2$
- if A has linearly independent columns, then

$$\|Ax - b\|^2 > \|A\hat{x} - b\|^2 \quad \text{(unique solution)}$$

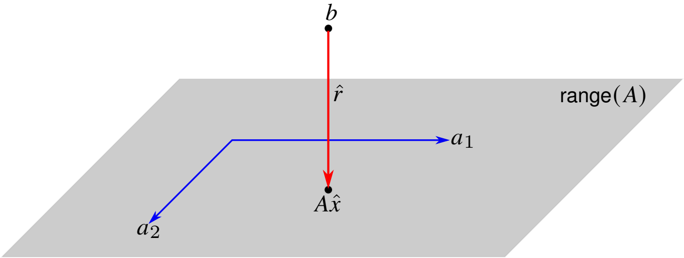
this is because $\|A(x - \hat{x})\|^2 = 0 \Rightarrow A(x - \hat{x}) = 0 \Rightarrow x = \hat{x}$

Geometric interpretation

let $a_1, \ldots, a_n$ denote columns of A, then

$$\|Ax - b\|^2 = \|(x_1 a_1 + \cdots + x_n a_n) - b\|^2$$

- $A\hat{x}$ is the vector in $\text{range}(A) = \text{span}(a_1, \ldots, a_n)$ closest to b
- $\hat{r} = A\hat{x} - b$ is orthogonal to $\text{range}(A)$: $\hat{r} \perp Aw$ for any w



- $A\hat{x} = AA^\dagger b$ is projection on $\text{range}(A)$

Example

given two different types of concrete:

- 1st contains 30% cement, 40% gravel, and 30% sand (percentages of weight)
- 2nd contains 10% cement, 20% gravel, and 70% sand

how many pounds of each type of concrete should you mix together so that you get a concrete mixture that has as close as possible to a total of 5 pounds of cement, 3 pounds of gravel, and 4 pounds of sand?

- letting x_1 and x_2 to be the amounts of concrete of the first and second types

- the above problem can be formulated as the least squares problem:

$$\text{minimize} \quad \left\| \begin{bmatrix} 0.3 & 0.1 \\ 0.4 & 0.2 \\ 0.3 & 0.7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \\ 4 \end{bmatrix} \right\|^2 = \|Ax - b\|^2,$$

  where $x = (x_1, x_2)$

- since the columns of A are linearly independent, the solution is

$$\hat{x} = (A^T A)^{-1} A^T b = \begin{bmatrix} 10.6 \\ 0.961 \end{bmatrix}$$

QR factorization method

using QR factorization $A = QR$, we have

$$\begin{aligned} \hat{x} = (A^T A)^{-1} A^T b &= \left((QR)^T (QR)\right)^{-1} (QR)^T b \\ &= (R^T Q^T Q R)^{-1} R^T Q^T b \\ &= R^{-1} Q^T b \end{aligned}$$

- identical formula for solving $Ax = b$ for square invertible A
- here $\hat{x}$ gives least squares approximate solution to $Ax = b$

Algorithm

1. compute QR factorization $A = QR$ ($2mn^2$ flops if A is $m \times n$)
2. matrix-vector product $Q^T b$ ($2mn$ flops)
3. solve $Rx = Q^T b$ by back substitution (n^2 flops)

Complexity: $2mn^2$ flops

Example

$$A = \begin{bmatrix} 3 & -6 \\ 4 & -8 \\ 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} -1 \\ 7 \\ 2 \end{bmatrix}$$

1. QR factorization: $A = QR$ with

$$Q = \begin{bmatrix} 3/5 & 0 \\ 4/5 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 5 & -10 \\ 0 & 1 \end{bmatrix}$$

2. calculate $d = Q^T b = (5, 2)$

3. solve $Rx = d$

$$\begin{bmatrix} 5 & -10 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

solution is $x_1 = 5, x_2 = 2$

Solving normal equations directly

---

given $m \times n$ matrix A with linearly independent columns and n-vector b

1. form $B = A^T A$ and $y = A^T b$
2. compute the Cholesky factorization $B = R^T R$ (R is lower triangular)
3. solve $R^T z = y$ for z using forward substitution
4. solve $Rx = z$ for x using back substitution

Complexity: approximately $mn^2 + n^3/3$ (flops)

- step 1 costs mn^2
- step 2 is approximately $n^3/3$ flops
- steps 3 and 4 cost order n^2 flops
- when $m \gg n$, the main cost becomes in forming the matrix $B = A^T A$

Comparison of the two methods

Complexity

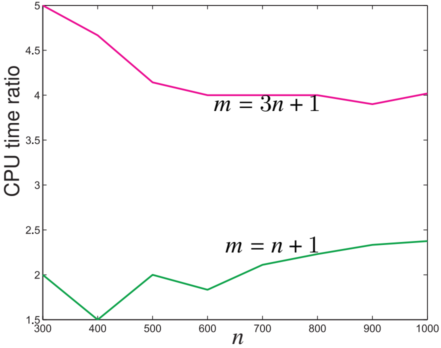
- Cholesky method: $mn^2 + (1/3)n^3$ flops
- QR method: $2mn^2$ flops
- Cholesky method is faster by a factor of at most two (if $m \gg n$)

Numerical stability: QR factorization method is more stable

- QR method computes R without "squaring" A (*i.e.*, forming A^TA)
- this is important when the columns of A are "almost" linearly dependent

Example

- randomly create A and a vector b
- plot **ratio** of CPU times for using QR fact. over normal equations options



- normal equations method is more efficient

Code

```
for n = 300:100:1000
% fill a rectangular matrix A and a vector b with random numbers
m = n+1; % or m= 3*n+1
A = randn(m,n); b = randn(m,1);
% solve and find execution times; first, Matlab way using QR
t0 = cputime;
xqr = A \ b;
temp = cputime;
tqr(n/100-2) = temp - t0;
% next use normal equations
t0 = temp;
B = A'*
A; y = A'*
b;
xne = B \ y;
temp = cputime;
tne(n/100-2) = temp - t0;
end
ratio = tqr./tne;
plot(300:100:1000,ratio)
```

Solving the normal equations

- last example shows direct method is faster
- however, QR method is more stable as illustrated next

Example: a 3×2 matrix with "almost linearly dependent" columns

$$A = \begin{bmatrix} 1 & -1 \\ 0 & 10^{-5} \\ 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 10^{-5} \\ 1 \end{bmatrix}$$

we round intermediate results to 8 significant decimal digits

Method 1: form Gram matrix A^TA and solve normal equations

$$A^TA = \begin{bmatrix} 1 & -1 \\ -1 & 1+10^{-10} \end{bmatrix} \rightsquigarrow \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad A^Tb = \begin{bmatrix} 0 \\ 10^{-10} \end{bmatrix}$$

after rounding, the Gram matrix is singular; hence method fails

Method 2: QR factorization of A is

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & -1 \\ 0 & 10^{-5} \end{bmatrix}$$

rounding does not change any values (in this example)

- problem with method 1 occurs when forming Gram matrix A^TA
- QR factorization method is more stable because it avoids forming A^TA

Standard methods for solving the linear least squares

Normal equations (Cholesky)

- fast, simple, intuitive
- can be unstable when columns of A are "almost" linearly dependent

QR factorization

- this is the "standard" approach (*e.g.*, in MATLAB)
- more robust than the normal equations approach
- more computationally expensive than the normal equations approach if $m \gg n$

Singular value decomposition (SVD) (more on this later in course)

- used mostly when columns of A are (almost) dependent
- very robust but more expensive than QR approach

Matrix least squares

$$\text{minimize} \quad \|AX - B\|_F^2$$

- variable is the $n \times k$ matrix $X = [x_1 \ \cdots \ x_k]$
- A is an $m \times n$ matrix and B is an $m \times k$ matrix
- decouples into a set of k ordinary least squares since

$$\|AX - B\|_F^2 = \|Ax_1 - b_1\|^2 + \cdots + \|Ax_k - b_k\|^2$$

  where x_j is the jth column of X and b_j is the jth column of B
- can choose the columns x_j independently, by minimizing $\|Ax_j - b_j\|^2$
- assuming A has linearly independent columns, the solution is $\hat{x}_j = A^\dagger b_j$ or

$$\hat{X} = A^\dagger B$$

Outline

- least squares problem
- solution and normal equations
- **multi-objective least squares**
- control
- estimation and inversion

Multi-objective least squares

choose n-vector x so that the following objectives are all small

$$J_1 = \|A_1 x - b_1\|^2, \ldots, J_k = \|A_k x - b_k\|^2$$

- A_i is an $m_i \times n$ matrix, b_i is an m_i-vector, $i = 1, \ldots, k$
- J_i are the objectives in a multi-objective (multi-criterion) optimization problem

Weighted sum objective: choose positive *weights* λ_i and find x that minimizes

$$J = \lambda_1 J_1 + \cdots + \lambda_k J_k = \lambda_1 \|A_1 x - b_1\|^2 + \cdots + \lambda_k \|A_k x - b_k\|^2$$

- we set $\lambda_1 = 1$, and call J_1 the *primary objective*
- λ_i gives how much we care about J_i being small, relative to J_1
- terms $\lambda_2 J_2, \ldots, \lambda_k J_k$ are called *regularization terms*

Weighted sum solution

write weighted-sum objective as

$$J = \left\| \begin{bmatrix} \sqrt{\lambda_1}\,(A_1 x - b_1) \\ \vdots \\ \sqrt{\lambda_k}\,(A_k x - b_k) \end{bmatrix} \right\|^2$$

so we have $J = \|\tilde{A}x - \tilde{b}\|^2$, with

$$\tilde{A} = \begin{bmatrix} \sqrt{\lambda_1} A_1 \\ \vdots \\ \sqrt{\lambda_k} A_k \end{bmatrix}, \qquad \tilde{b} = \begin{bmatrix} \sqrt{\lambda_1} b_1 \\ \vdots \\ \sqrt{\lambda_k} b_k \end{bmatrix}$$

Weighted sum solution: assuming columns of $\tilde{A}$ are linearly independent,

$$\begin{aligned} \hat{x} &= (\tilde{A}^T \tilde{A})^{-1} \tilde{A}^T \tilde{b} \\ &= (\lambda_1 A_1^T A_1 + \cdots + \lambda_k A_k^T A_k)^{-1} (\lambda_1 A_1^T b_1 + \cdots + \lambda_k A_k^T b_k) \end{aligned}$$

(here, A_i can be wide, or have dependent columns)

Optimal trade-off curve

Bi-criterion problem: we let $\hat{x}(\lambda)$ be minimizer of bi-criterion objectives

$$J_1 + \lambda J_2 = \|A_1 x - b_1\|^2 + \lambda \|A_2 x - b_2\|^2$$

Pareto optimal point

- $\hat{x}(\lambda)$ is called *Pareto optimal*
- there is no point z that satisfies

$$J_1(z) < J_1(\hat{x}(\lambda)), \quad J_2(z) < J_2(\hat{x}(\lambda))$$

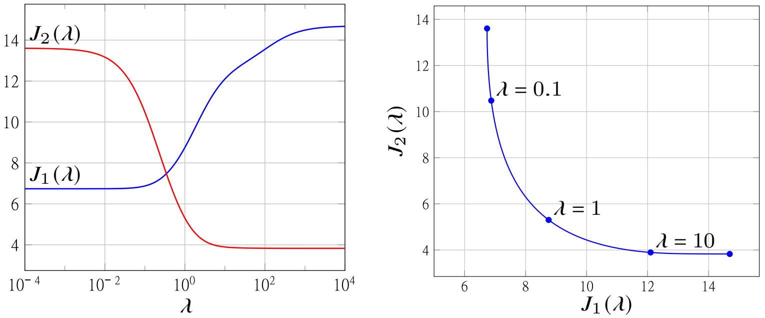
  i.e., no other point beats $\hat{x}$ on both objectives

Optimal trade-off curve

$$\big(J_1(\hat{x}(\lambda)), J_2(\hat{x}(\lambda))\big) \quad \text{for} \quad \lambda > 0$$

Example

A_1 and A_2 both 10×5



- we can achieve a substantial reduction in J_2 with only a small increase in J_1
- weights are typically logarithmically spaced; for N values of $\lambda^{\min} \leq \lambda \leq \lambda^{\max}$:

$$\lambda^{\min}, \quad \theta\lambda^{\min}, \quad \theta^2\lambda^{\min}, \ldots, \quad \theta^{N-1}\lambda^{\min} = \lambda^{\max}$$

with $\theta = (\lambda^{\max}/\lambda^{\min})^{1/(N-1)}$

Tikhonov regularization

the weighted least squares problem

$$\text{minimize} \quad \|Ax - y\|^2 + \lambda \|x\|^2$$

is known as *Tikhonov regularization*

- goal is to make $\|Ax - y\|$ small with x that is not too big
- equivalent to solving
$$(A^T A + \lambda I)x = A^T y$$
- solution is unique (if $\lambda > 0$) even when A has linearly dependent columns

Outline

- least squares problem
- solution and normal equations
- multi-objective least squares
- **control**
- estimation and inversion

Control

$$y = Ax + b$$

- n-vector x corresponds to *actions* or *inputs*
- m-vector y corresponds to *results* or *outputs*
- A and b are known (from analytical models, data fitting, ...)
- goal is to choose x, to optimize multiple objectives on x and y

Multi-objective control

- primary objective: $J_1 = \|y - y^{\text{des}}\|^2$, y^{des} is a given desired/target output
- typical secondary objectives:
  - x is small: $J_2 = \|x\|^2$
  - x is not far from a nominal input: $J_2 = \|x - x^{\text{nom}}\|^2$

Optimal input design

Linear dynamical system

$$y(t) = h_0 u(t) + h_1 u(t-1) + h_2 u(t-2) + \cdots + h_t u(0)$$

- output $y(t)$ and input $u(t)$ are scalar
- we assume input $u(t)$ is zero for $t < 0$
- coefficients $h_0, h_1, \ldots$ are the impulse response coefficients
- output is convolution of input with impulse response

Optimal input design

- optimization variable is the input sequence $x = (u(0), u(1), \ldots, u(N))$
- goal is to track a desired output using a small and slowly varying input

Input design objectives

$$\text{minimize} \quad J_{\rm t}(x) + \lambda_{\rm v} J_{\rm v}(x) + \lambda_{\rm m} J_{\rm m}(x)$$

- primary objective: track desired output y_{des} over an interval $[0, N]$:

$$J_{\rm t}(x) = \sum_{t=0}^{N} \big(y(t) - y_{\rm des}(t)\big)^2$$

- secondary objectives: use a small and slowly varying input signal:

$$J_{\rm m}(x) = \sum_{t=0}^{N} u(t)^2$$

$$J_{\rm v}(x) = \sum_{t=0}^{N-1} (u(t+1) - u(t))^2$$

Tracking error

$$J_{\mathrm{t}}(x) = \sum_{t=0}^{N} (y(t) - y_{\mathsf{des}}(t))^2 = \|A_{\mathrm{t}}x - b_{\mathrm{t}}\|^2$$

with

$$A_{\mathrm{t}} = \begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 & 0 \\ h_1 & h_0 & 0 & \cdots & 0 & 0 \\ h_2 & h_1 & h_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{N-1} & h_{N-2} & h_{N-3} & \cdots & h_0 & 0 \\ h_N & h_{N-1} & h_{N-2} & \cdots & h_1 & h_0 \end{bmatrix}, \quad b_{\mathrm{t}} = \begin{bmatrix} y_{\mathrm{des}}(0) \\ y_{\mathrm{des}}(1) \\ y_{\mathrm{des}}(2) \\ \vdots \\ y_{\mathrm{des}}(N-1) \\ y_{\mathrm{des}}(N) \end{bmatrix}$$

Input variation and magnitude

Input variation

$$J_{\rm v}(x) = \sum_{t=0}^{N-1} (u(t+1) - u(t))^2 = \|Dx\|^2$$

where D the $N \times (N+1)$ difference matrix

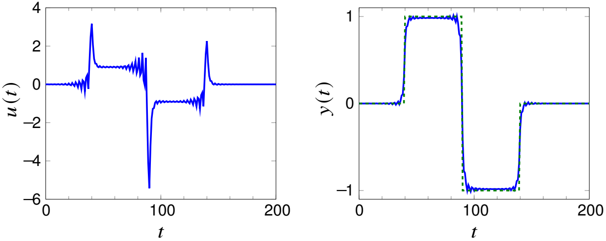
$$D = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix}$$

Input magnitude

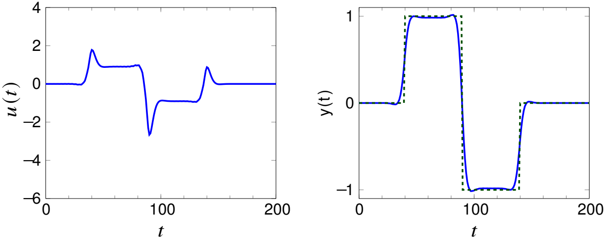
$$J_{\rm m}(x) = \sum_{t=0}^{N} u(t)^2 = \|x\|^2$$

Example

$\lambda_{\mathrm{v}} = 0$, small λ_{m}



larger λ_{v} larger λ_{m}



Outline

- least squares problem
- solution and normal equations
- multi-objective least squares
- control
- **estimation and inversion**

Estimation (inversion)

measurement model:

$$y = Ax + v$$

- n-vector x contains parameters we want to estimate
- m-vector y contains the *measurements*
- m-vector v are (unknown) *noises* or *measurement errors*
- $m \times n$ matrix A connects parameters to measurements

Least squares estimation

- we guess x by minimizing $J_1 = \|Ax - y\|^2$
- when v is nonzero or A has dependent columns, we cannot determine x exactly
- in this case, we add other objectives to encode prior information about x
  - x is small: $J_2 = \|x\|^2$
  - x is not far from a nominal input: $J_2 = \|x - x^{\text{nom}}\|^2$

Example: estimating a periodic time series

- T-vector y is a (measured) time series, of a periodic time series with period P
- P-vector x gives its values over one period, so

$$\hat{y} = (x, x, \ldots, x)$$

  where we assume here for simplicity that T is a multiple of P

- we can express $\hat{y}$ as $\hat{y} = Ax$, where A is the $T \times P$ selector matrix

$$A = \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix}$$

- we assume that the periodic time series is smooth

$$x_1 \approx x_2, \quad \ldots, \quad x_{P-1} \approx x_P, \quad x_P \approx x_1$$

Example: estimating a periodic time series

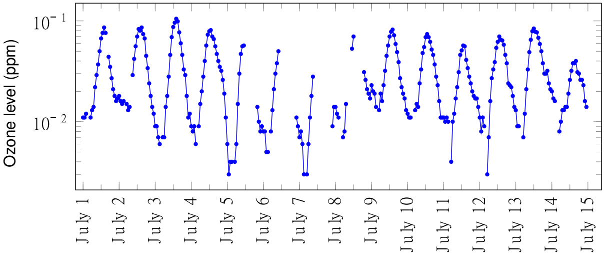
we estimate the periodic time series by minimizing

$$\|Ax - y\|^2 + \lambda \|D^{\text{circ}} x\|^2$$

where D^{circ} is the $P \times P$ circular difference matrix

$$D^{\text{circ}} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 & -1 \end{bmatrix}$$

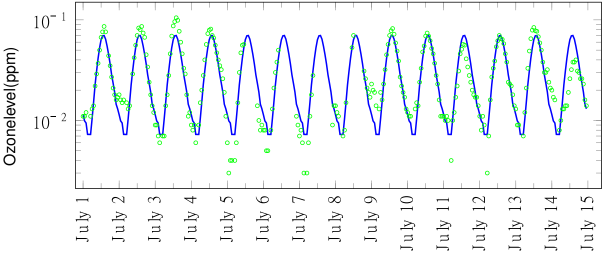
Example: hourly ozone measurements

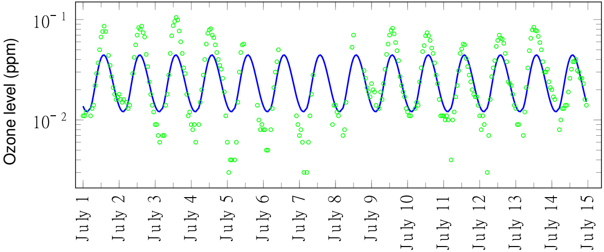


- 336-vector c of measurements with some missing values
- $c_{24(j-1)+i}$, $i = 1, \ldots, 24$, contain hourly values on day j, $j = 1, \ldots, 14$
- $M_j \subseteq \{1, 2, \ldots, 24\}$ is set with indices of available measurements on day j
- least squares objective:

$$\sum_{j=1}^{14} \sum_{i \in M_j} \left(x_i - \log\left(c_{24(j-1)+i}\right)\right)^2 + \lambda \left(\sum_{i=1}^{23} (x_{i+1} - x_i)^2 + (x_1 - x_{24})^2 \right)$$

results for $\lambda = 1$ and $\lambda = 100$





Example: least squares image deblurring

$$y = Ax + v$$

- x is unknown image, y is observed blurred noisy image
- A is (known) blurring matrix, v is (unknown) noise
- images are $M \times N$, stored as MN-vectors

Least squares deblurring

$$\text{minimize} \quad \|Ax - y\|^2 + \lambda \left(\|D_\mathrm{h}x\|^2 + \|D_\mathrm{v}x\|^2\right)$$

- 1st term is "data fidelity" term: ensures $A\hat{x} \approx y$
- 2nd term penalizes differences between values at neighboring pixels

$$\|D_\mathrm{h}x\|^2 + \|D_\mathrm{v}x\|^2 = \sum_{i=1}^{M} \sum_{j=1}^{N-1} \left(X_{i,j+1} - X_{ij}\right)^2 + \sum_{i=1}^{M-1} \sum_{j=1}^{N} \left(X_{i+1,j} - X_{ij}\right)^2$$

when X is the $M \times N$ image stored in the MN-vector x

Example: least squares image deblurring

suppose x is the $M \times N$ image X, stored column-wise as MN-vector

$$x = \left(X_{1:M,1}, X_{1:M,2}, \ldots, X_{1:M,N}\right)$$

- **horizontal differencing:** $(N-1) \times N$ *block* matrix with $M \times M$ blocks

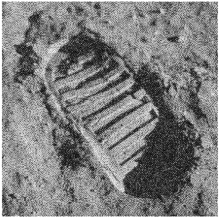
$$D_{\mathrm{h}} = \begin{bmatrix} -I & I & 0 & \cdots & 0 & 0 & 0 \\ 0 & -I & I & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -I & I \end{bmatrix}$$

- **vertical differencing:** $N \times N$ *block* matrix with $(M-1) \times M$ blocks

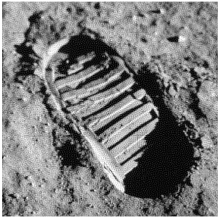
$$D_{\mathrm{v}} = \begin{bmatrix} D & 0 & \cdots & 0 \\ 0 & D & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & D \end{bmatrix}, \quad D = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix}$$

Example

$\lambda = 10^{-6}$



$\lambda = 10^{-4}$



$\lambda = 10^{-2}$

$\lambda = 1$

Example: tomography

goal: reconstruct a (density) function $d : \mathbb{R}^2 \to \mathbb{R}$ from line integral measurements

- measurements obtained by passing a beam of radiation through region of interest, and measuring the intensity of the beam after it exits region
- used in medicine, manufacturing, networking, geology
  - common application: CAT (computer-aided tomography) scan

Line integral: parametrize line ℓ in 2-D as

$$p(t) = (x_0, y_0) + t(\cos\theta, \sin\theta)$$

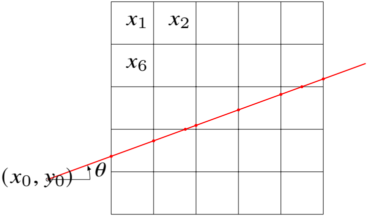
- (x_0, y_0) is any point on the line
- θ is angle measured from horizontal; t is length along line
- line integral of d on ℓ is $\int_\ell d = \int_{-\infty}^{\infty} d(p(t))dt$
- can be extended to 3-D

Line integral measurements

- assume d is constant on pixel (or voxel) i with value x_i
- measurement of integral along line i through region is

$$y_i = \int_{-\infty}^{\infty} d(p(t))dt + v_i = \sum_{j=1}^{n} A_{ij}x_j + v_i \quad \text{where } v_i \text{ is small noise}$$

- A_{ij} is the length of measurement line i in pixel j
- in matrix-vector form: $y = Ax + v$



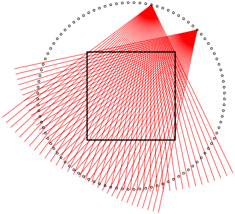
$$y = 1.06x_{16} + 0.8x_{17} + 0.27x_{12} + 1.06x_{13}$$
$$+ 1.06x_{14} + 0.53x_{15} + 0.54x_{10} + v$$

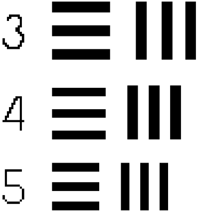
Least squares tomographic reconstruction

$$\text{minimize} \quad \|Ax - y\|^2 + \lambda \left(\|D_{\rm v}x\|^2 + \|D_{\rm h}x\|^2\right)$$

$D_{\rm v}$ and $D_{\rm h}$ are defined as in image deblurring example

Example

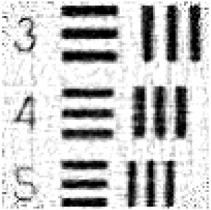




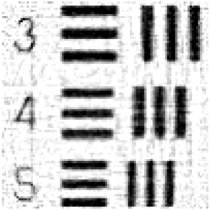
- left: 4000 lines (100 points, 40 lines per point)
- right: object placed in the square region on the left
- region of interest is divided in 10000 pixels

Regularized least squares reconstruction

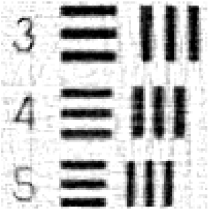
$\lambda = 10^{-2}$



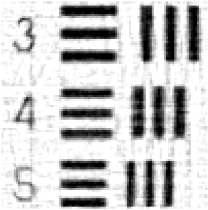
$\lambda = 10^{-1}$



$\lambda = 1$

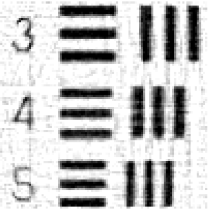


$\lambda = 5$

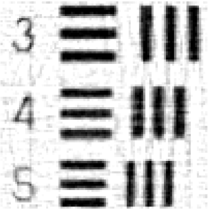


Regularized least squares reconstruction

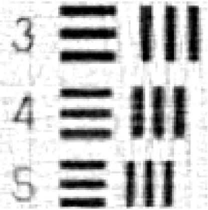
$\lambda = 1$



$\lambda = 5$



$\lambda = 10$



$\lambda = 100$

References and further readings

- S. Boyd and L. Vandenberghe. *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares,* Cambridge University Press, 2018.

- L. Vandenberghe. *EE133A lecture notes,* Univ. of California, Los Angeles. (`http://www.seas.ucla.edu/~vandenbe/ee133a.html`)