Leacture-6

Least-Squares application

★ Least-Square data fitting

⇒ We are given:

It can be anything

- functions $f_1, \dots f_n : S \to R$, called regressors or basic function.
- data or measurements or samples (s_i, g_i) $i = 1, \dots m$ where $s_i \in S$, $g_i \in R$ (usually $m \gg n$)

⇒ Problem: find coefficients $x_1, \dots x_n \in R$ So that

$$x_1 f(s_i) + \dots + x_n f(s_i) \approx g_i, \quad i = 1, \dots m$$

⇒ least-squares fit: choose x to minimize total square fitting error:

$$\sum_{i=1}^{m} \left(x_1 f_1(s_i) + \dots + x_n f_n(s_i) - g_i \right)^2$$

⇒ Using matrix notation, total least square fitting error is $\|Ax - g\|^2$ where $\boxed{A_{ij} = f_j(s_i)}$

⇒ hence, least-square fit is given by

$$\boxed{x = (A^T A)^{-1} A^T y}$$

(assuming A is skinny, full rank)

⇒ Corresponding function is

$$\boxed{f_{lsfit} = x_1 f_1(s) + \cdots + x_n f_n(s)}$$

⇒ Application
- → Interpolation, Extrapolation
- → Smoothing of data
- → developing simple, approximate model of data.

★ Least-squares polynomial fitting

⇒ Problem: fit polynomial of degree $< n$

$$p(t) = a_0 + a_1 t + \cdots + a_{n-1} t^{n-1}$$

to data (t_i, y_i), $i = 1, \ldots, m$

⇒ basic functions are $f_j(t) = t^{j-1}$, $j = 1, \ldots n$

⇒ matrix A has form $A_{ij} = t_i^{j-1}$

$$A = \begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \cdots & t_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & t_m^2 & \cdots & t_m^{n-1} \end{bmatrix}$$

{Called a Vandermonde matrix}

★ Growing sets of regressors

⇒ Consider family of least square problems

$$\text{minimize} \left\| \sum_{i=1}^{p} x_i a_i - y \right\|$$

$\forall \, p = 1, \ldots n$

($a_1, \ldots a_p$ are called regressors)

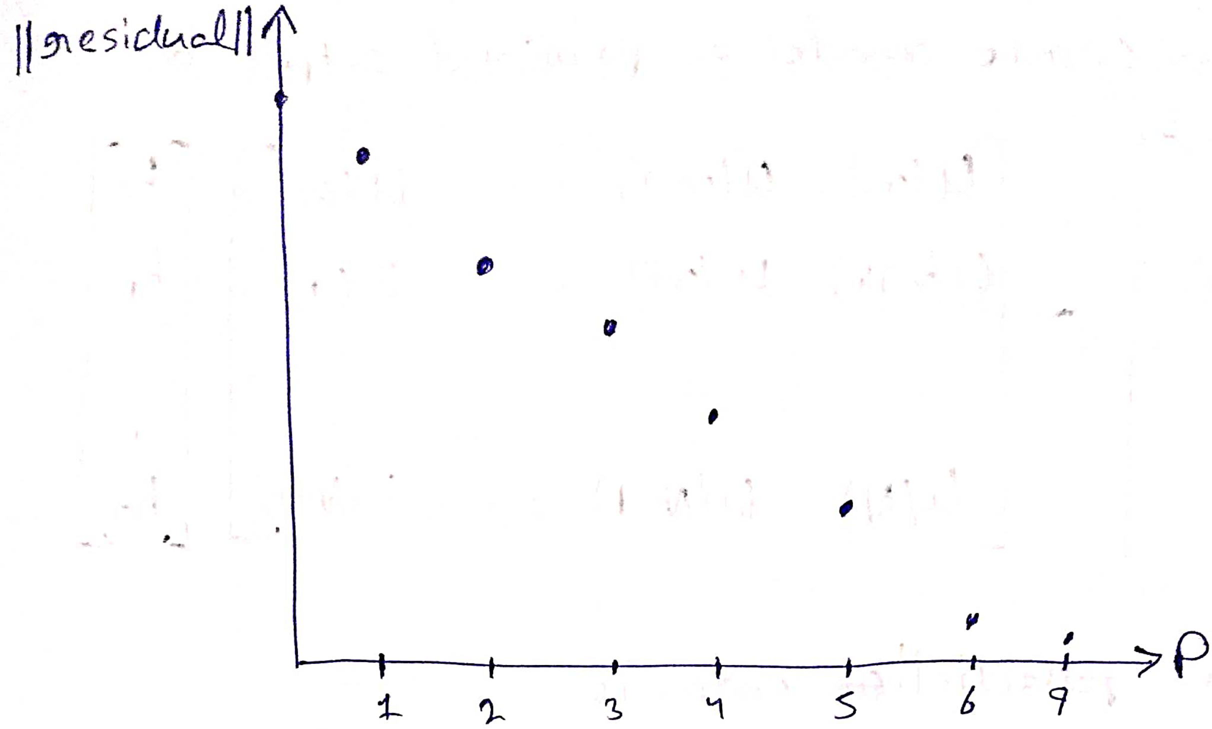
- approximate y by linear combination of $a_1, \ldots a_p$.
- project y onto span $\{a_1, \ldots a_p\}$
- regress y on $a_1, \ldots a_p$
- as p increases, get better fit, so optimal residual decreases.

⇒ Solution for each $p \leq n$ is given by

$$x_{ls}^{(p)} = (A_p^T A_p)^{-1} A_p^T y = R_p^{-1} Q_p^T y$$

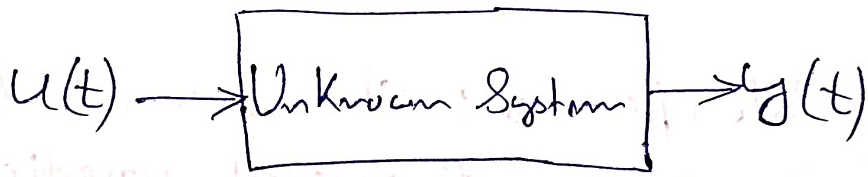
★ Norm of optimal residual versus p

⇒ Plot of optimal residual vs p shows how well y can be matched by linear combination of $a_1 \ldots a_p$, as function of p.



★ Least-square system Identification {

⇒ We measure input $u(t)$ and output $y(t)$
$\forall\, t = 0, \dots N$ of unknown system.



System Identification Problem: Find reasonable model for system based on measured I/O data.

Moving-average (MA) model with m delays. (Modelling dynamic System)

$$\tilde{y}(t) = h_0 u(t) + h_1 u(t-1) + \dots + h_m u(t-m)$$

where $h_0, \dots h_m \in \mathbb{R}$

⇒ We can write model or predicted output as,

$$\begin{bmatrix} \tilde{y}(n) \\ \tilde{y}(n+1) \\ \vdots \\ \tilde{y}(N) \end{bmatrix} = \begin{bmatrix} u(n) & u(n-1) & \cdots & u(0) \\ u(n+1) & u(n) & \cdots & u(1) \\ \vdots & \vdots & \ddots & \vdots \\ u(N) & u(N-1) & \cdots & u(N-n) \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_n \end{bmatrix}$$

⇒ model prediction error is

$$e = \begin{bmatrix} y(n) - \tilde{y}(n) \\ \vdots \\ y(N) - \tilde{y}(N) \end{bmatrix}$$

⇒ least-square identification: Choose model (i.e. h) that minimizes norm of model prediction error $\|e\|$.

★ Model order selection

(how large should be n?)

→ Obviously the larger n, the smaller the prediction error on the data used to form the model.

→ Suggests using largest possible model order for smallest prediction errors

<u>difficulty</u>: for n too large the predictive ability of the model on other I/O data (from the same system) become worse.

★ <u>Cross-validation</u>

"Evaluating model predictive performance on another I/O data set not used to develop model"

⇒ Plot suggests $n=10$ is a good choice.

★ <u>Growing sets of measurements</u>

least-squares problem in row form:

$$\text{minimize } \|Ax-y\|^2 = \sum_{i=1}^{m} (a_i^T x - y_i)^2$$

→ where a_i^T are the rows of A $(a_i \in \mathbb{R}^n)$

$\Rightarrow$ Solution is

$$x_{ls} = (A_m^T A_m)^{-1} A_m^T y = \left(\sum_{i=1}^{m} a_i a_i^T\right)^{-1} \sum_{i=1}^{m} y_i a_i$$

$\Rightarrow$ Suppose that a_i and y_i become available sequentially. (i.e. m increases with time)

★ Recursive least-square

$\Rightarrow$ We can compute $x_{ls}(m) = \left(\sum_{i=1}^{m} a_i a_i^T\right)^{-1} \sum_{i=1}^{m} y_i a_i$

~~respectiv~~ recursively.

$\Rightarrow$ initialize $P(0) = 0 \in R^{n \times n}$, $q(0) = 0 \in R^n$

$\Rightarrow$ for $m = 0, 1 \cdots$

$$P(m+1) = P(m) + a_{m+1} a_{m+1}^T$$

$$q(m+1) = q(m) + y_{m+1} a_{m+1}$$

$\Rightarrow$ If $P(m)$ is invertible, we have

$$x_{ls}(m) = P(m)^{-1} q(m)$$

★ Fast update for recursive least-square

$\Rightarrow$ We can calculate

$$P(m+1)^{-1} = \left(P(m) + a_{m+1} a_{m+1}^T\right)^{-1}$$

⇒ efficiently from $P(m)^{-1}$ using the
rank one update formula:

$$\boxed{(P + aa^T)^{-1} = P^{-1} - \frac{1}{1 + a^T P^{-1} a} (P^{-1}a)(P^{-1}a)^T}$$

⇒ gives an $O(n^2)$ method for computing $P(m+1)^{-1}$ from $P(m)^{-1}$.

⇒ Standard methods for computing $P(m+1)^{-1}$ from $(m+1)$ is $O(n^3)$.