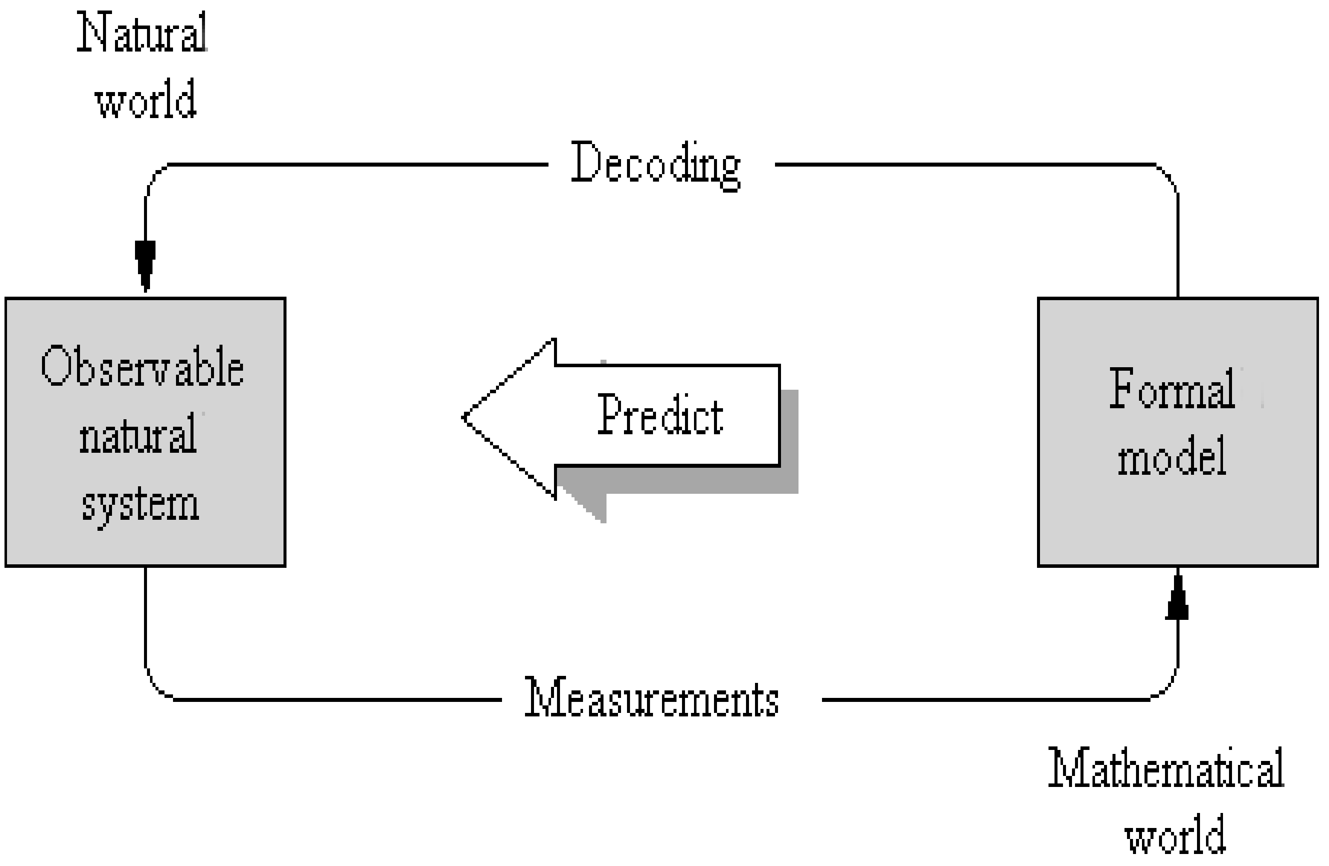
III. REGRESSION and OPTIMIZATION

- This part provides an overview of system identification and optimization techniques:
  - to predict a system’s behaviour,
  - to explain the interactions and relationships between inputs and outputs,
  - to design a controller based on the model of the system
- Least Square Methods
- Derivative Based Nonlinear Optimization

The goal of this chapter is to introduce the following concepts:

- Data fitting and the derivation of the best linear (regression) model
- Iterative solution of the regression model
- Steepest descent methods
- The least mean square (LMS) estimator for the gradient
- The trade-off between speed of adaptation and solution accuracy

3.1.System Identification and Adaptive Systems



Natural systems and formal models

- Neural and adaptive systems are used in many important engineering applications, such as signal enhancement, noise cancellation, classification of input patterns, system identification, prediction, and control.
- Instead of being built a priori from specification, neural and adaptive systems use external data to automatically set their parameters. This means that neural systems are parametric.
- The system output improves with respect to the desired goal (i.e., the error decreases through training).

- We now know some powerful topologies that are able to create universal input-output mappings.
- We also know how to design general adaptive algorithms to extract information from data and adapt the parameters of the mappers.
- We are also starting to understand the pre-requisites for generalization, that is, how to guarantee that the performance in the training set extends to the data found during system operation.

Therefore we are in a position to design effective adaptive solutions to moderately difficult real-world problems.

System Identification: Introduction

- Goal
  - Determine a mathematical model for an unknown system (or target system) by observing its input-output data pairs
- Purposes
  - To predict a system’s behavior, as in time series prediction & weather forecasting
  - To explain the interactions & relationships between inputs & outputs of a system
  - To design a controller based on the model of a system, as an aircraft or ship control
  - Simulate the system under control once the model is known

Structure identification

Apply a-priori knowledge about the target system to determine a class of models within which the search for the most suitable model is to be conducted; this class of model is denoted by a function $y = f(u,\theta)$ where:

y is the model output

u is the input vector

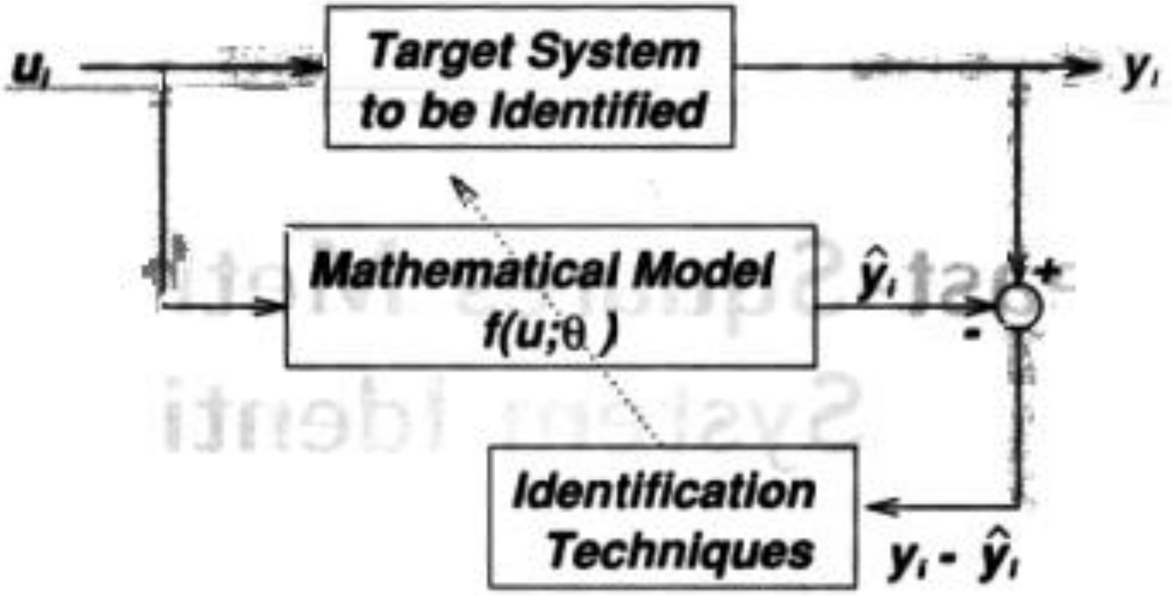
θ Is the parameter vector

f depends on the problem at hand & on the designer’s experience & the laws of nature governing the target system

Parameter identification

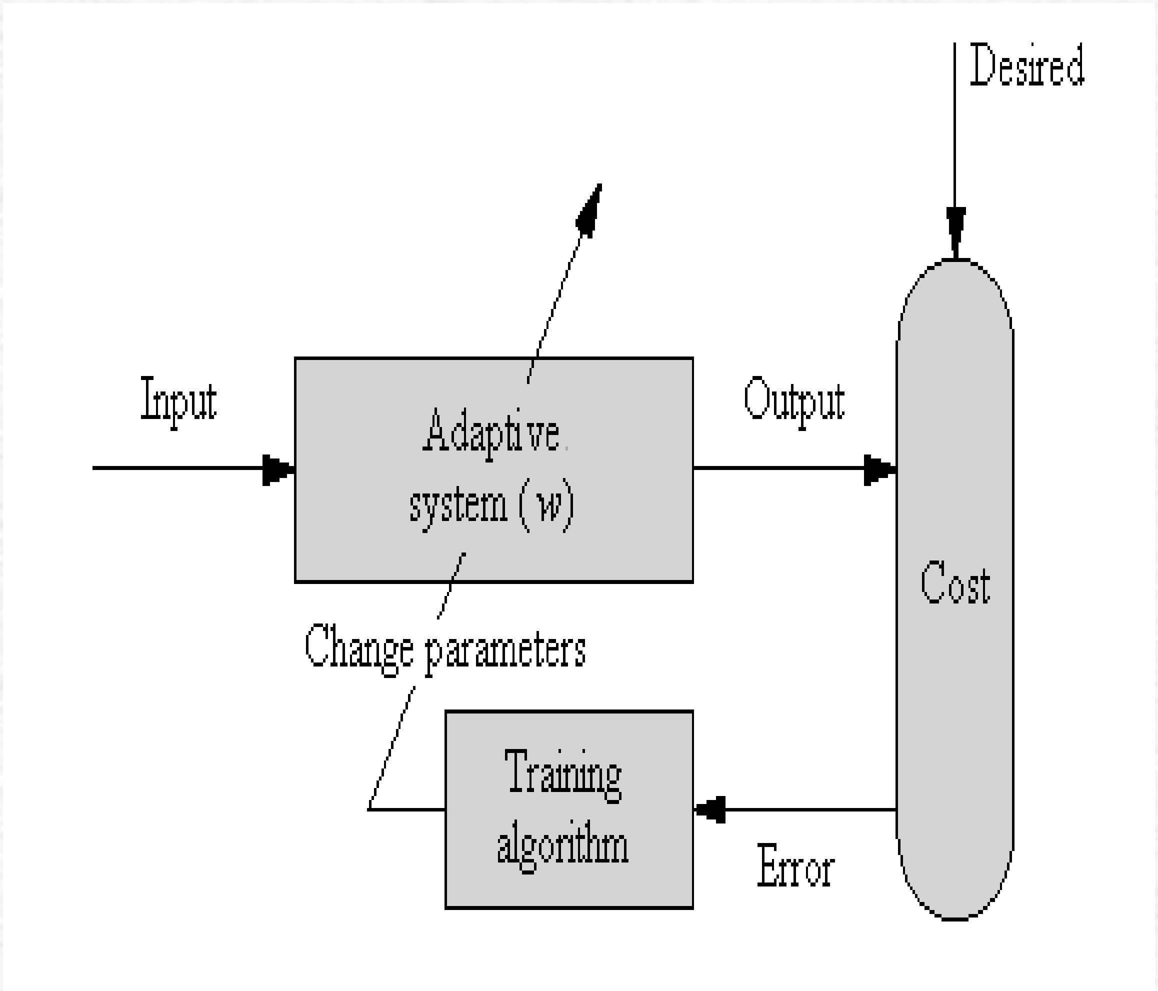
- The structure of the model is known, however we need to apply optimization techniques in order to determine the parameter vector $\theta = \hat{\theta}$ such that the resulting model $\hat{y} = f(u, \hat{\theta})$ describes the system appropriately:

$$\left| y_i - \hat{y} \right| \rightarrow 0 \ \text{ with } y_i \text{ assigned to } u_i$$


u_i
Target System to be Identified
y_i
Mathematical Model $f(u;\theta)$
$\hat{y}_i$
+
-
Identification Techniques
$y_i - \hat{y}_i$

5.1 Block diagram for parameter identification.

Adaptive Systems

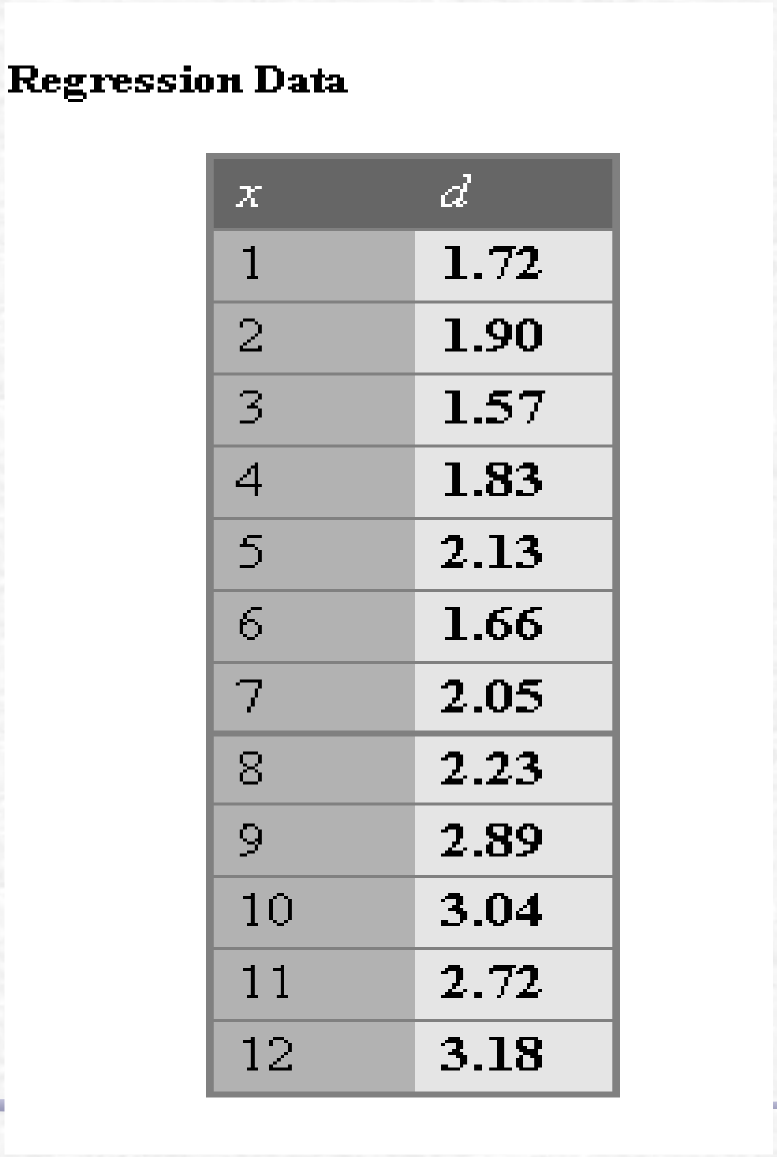


Adaptive system's design

Identification

- Structure
- Parameter
- Specify Mathematical Model
- Perform Parameter Identification (best fit) on Training Set
- Conduct Validation with unseen data set

Data Collection

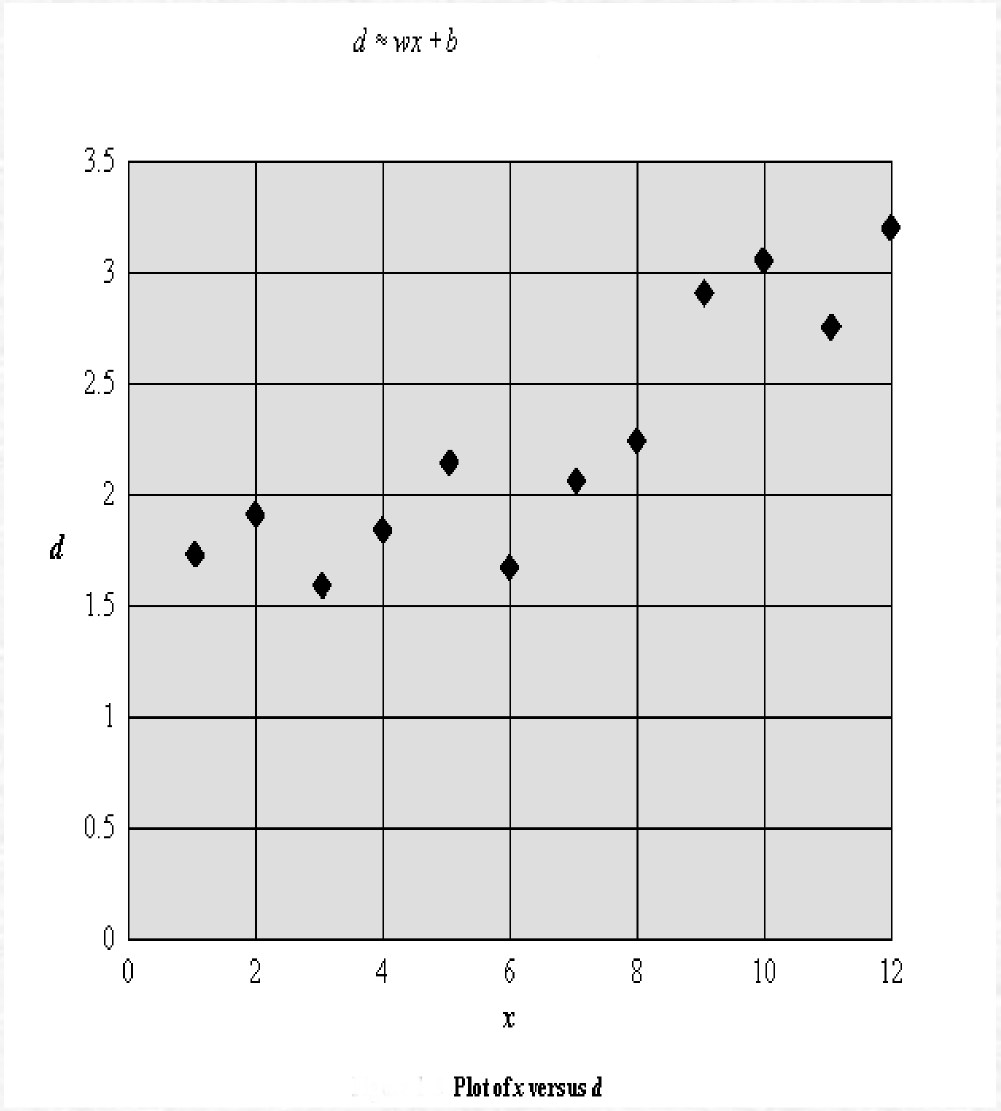


Regression Data

x	d
1	1.72
2	1.90
3	1.57
4	1.83
5	2.13
6	1.66
7	2.05
8	2.23
9	2.89
10	3.04
11	2.72
12	3.18

The data-collection phase must be carefully planned to ensure that

- Data will be sufficient
- Data will capture the fundamental principles at work
- Data is as free as possible from observation noise.



- The task of fitting data using a linear model is referred to as **linear regression**. Thus the linear equation is also called the **regression function** and unknown parameters are **regression coefficients**
- The criterion called *mean square error,* ***MSE*** is used to determine which linear estimator is best

- The data set composed of m desired input-output pairs $(u_i;y_i)$ $(i = 1,\ldots,m)$ is called the training data

- System identification needs to do both structure & parameter identification repeatedly until satisfactory model is found: it does this as follows:

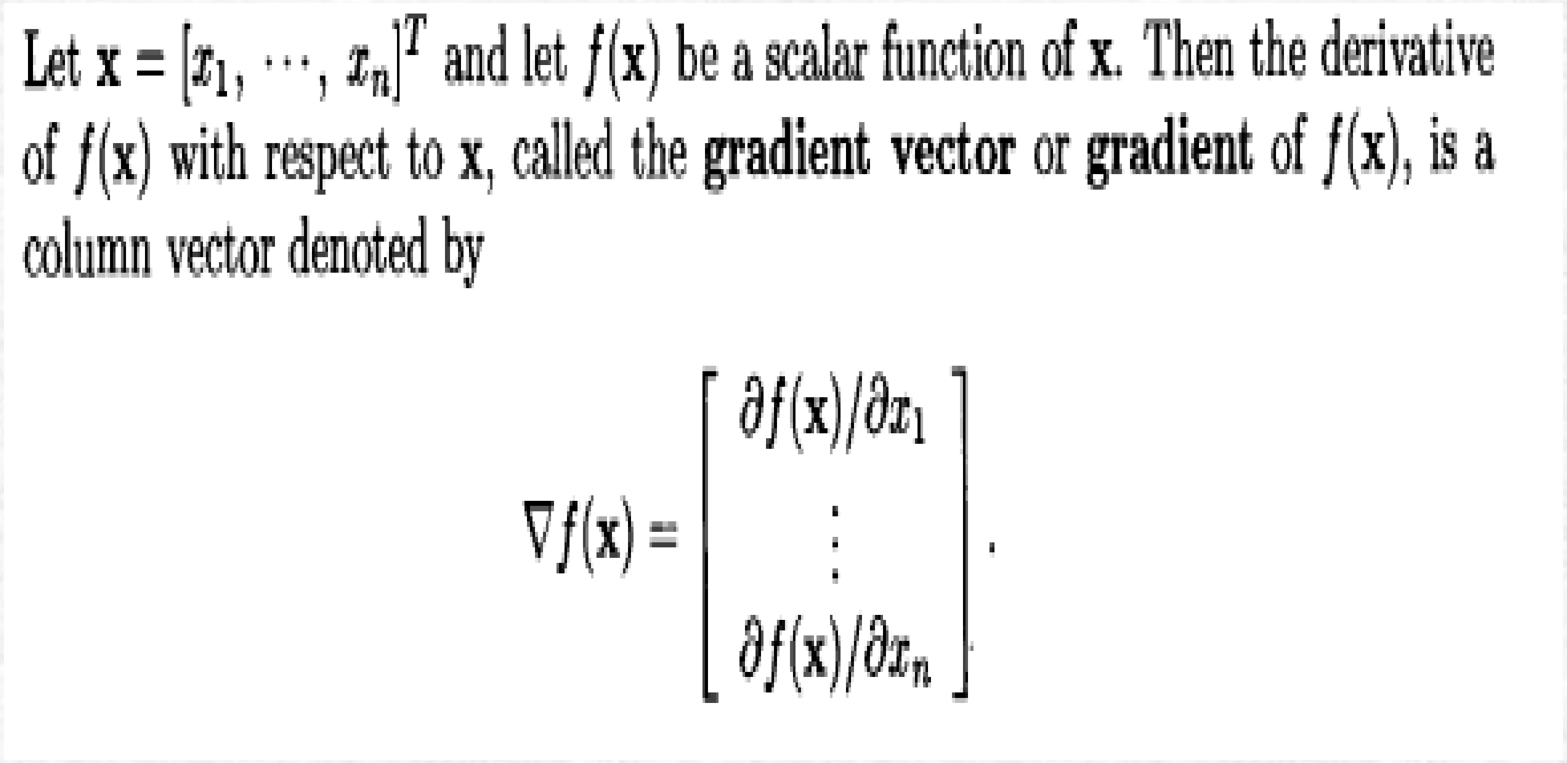
  1) Specify & parameterize a class of mathematical models representing the system to be identified

  2) Perform parameter identification to choose the parameters that best fit the training data set

  3) Conduct validation set to see if the model identified responds correctly to an unseen data set

  4) Terminate the procedure once the results of the validation test are satisfactory. Otherwise, another class of model is selected & repeat step 2 to 4

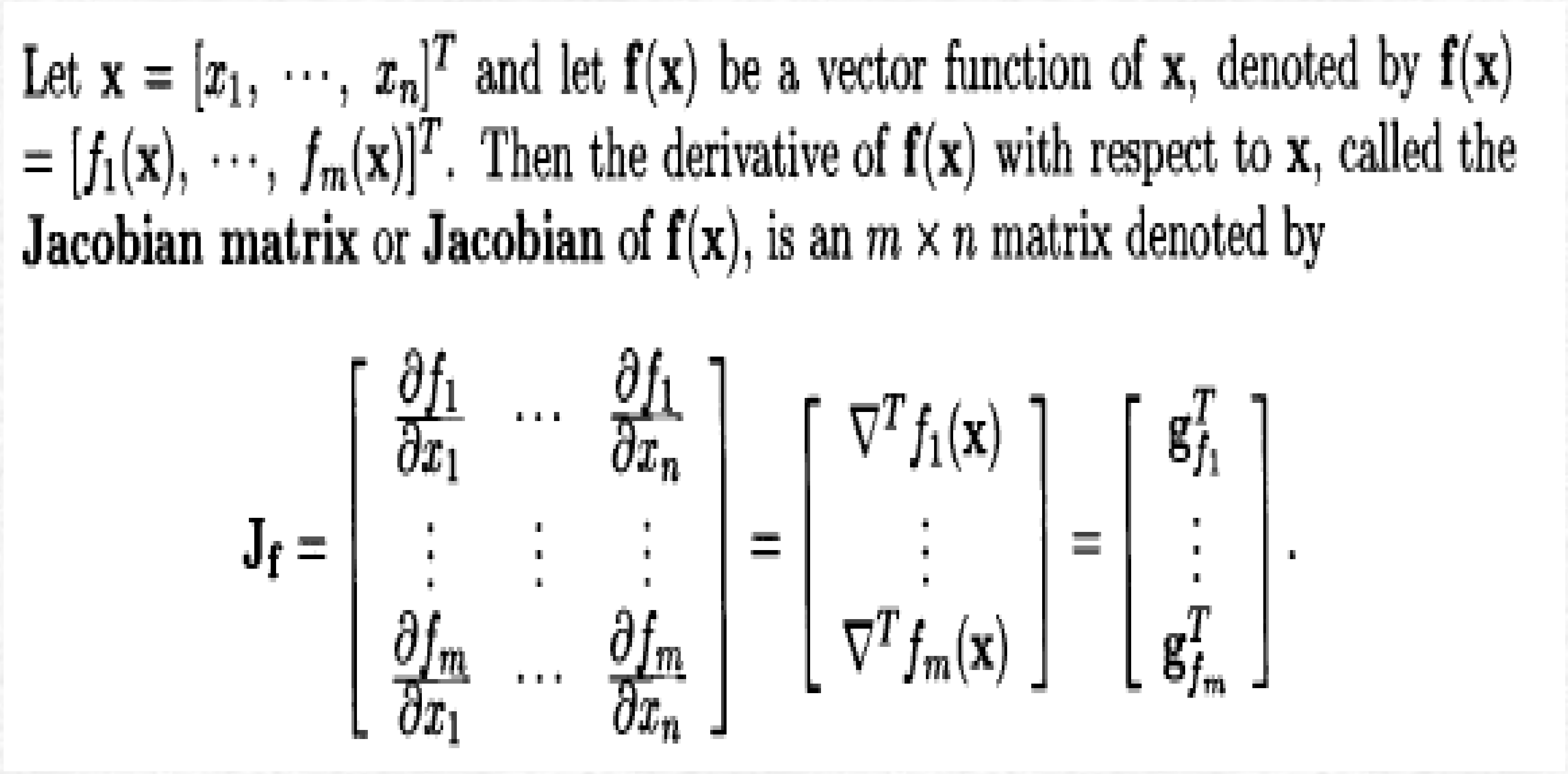
Gradient of a Scalar Function



Let $\mathbf{x} = [x_1, \cdots, x_n]^T$ and let $f(\mathbf{x})$ be a scalar function of $\mathbf{x}$. Then the derivative of $f(\mathbf{x})$ with respect to $\mathbf{x}$, called the **gradient vector** or **gradient** of $f(\mathbf{x})$, is a column vector denoted by

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \partial f(\mathbf{x})/\partial x_1 \\ \vdots \\ \partial f(\mathbf{x})/\partial x_n \end{bmatrix}.$$

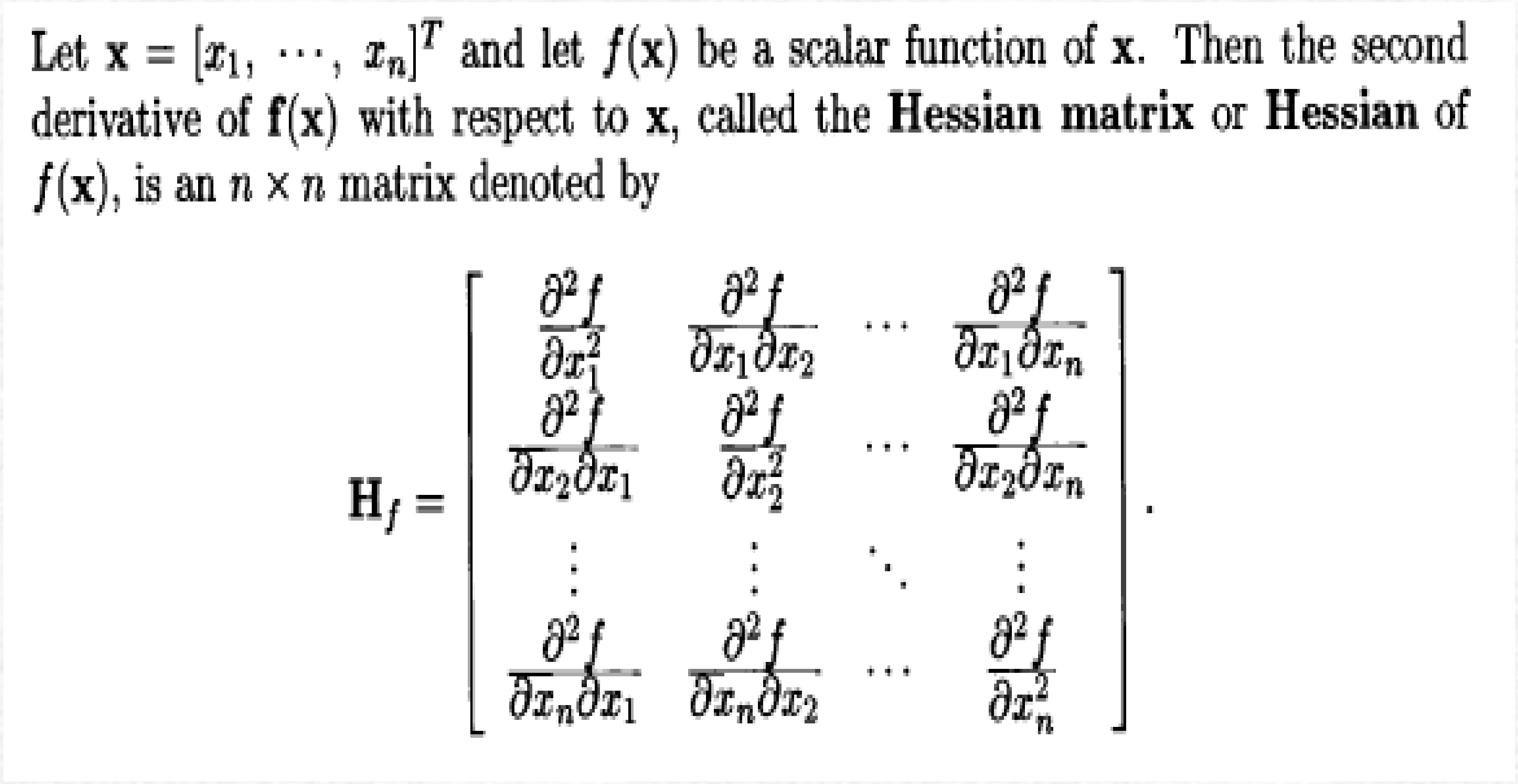
Jacobian of a Vector Function



Let $\mathbf{x} = [x_1, \cdots, x_n]^T$ and let $\mathbf{f}(\mathbf{x})$ be a vector function of $\mathbf{x}$, denoted by $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \cdots, f_m(\mathbf{x})]^T$. Then the derivative of $\mathbf{f}(\mathbf{x})$ with respect to $\mathbf{x}$, called the **Jacobian matrix** or **Jacobian** of $\mathbf{f}(\mathbf{x})$, is an $m \times n$ matrix denoted by

$$\mathbf{J_f} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^T f_1(\mathbf{x}) \\ \vdots \\ \nabla^T f_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \mathbf{g}_{f_1}^T \\ \vdots \\ \mathbf{g}_{f_m}^T \end{bmatrix}.$$

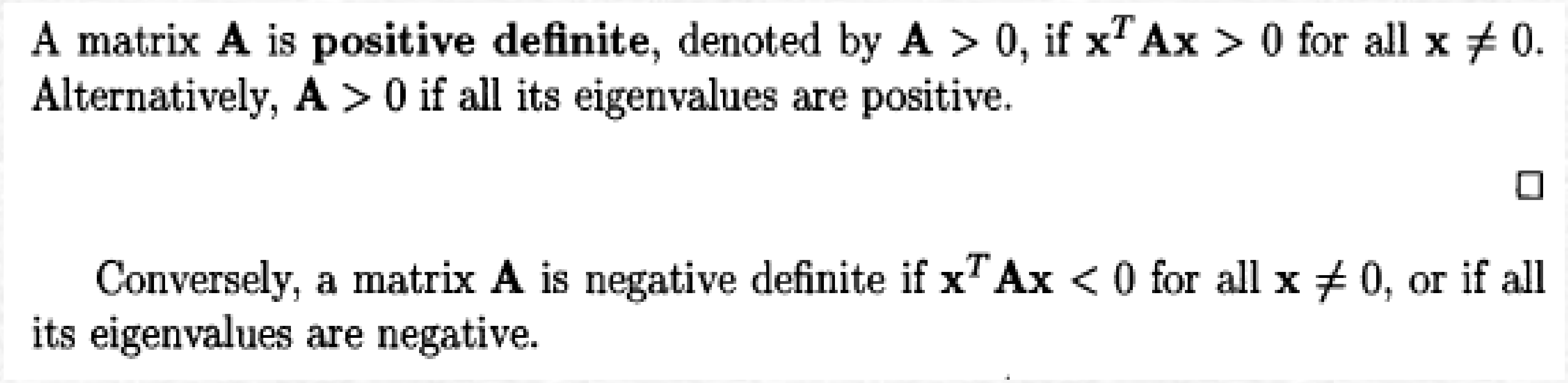
Hessian of a Scalar Function



Let $\mathbf{x} = [x_1, \cdots, x_n]^T$ and let $f(\mathbf{x})$ be a scalar function of $\mathbf{x}$. Then the second derivative of $\mathbf{f}(\mathbf{x})$ with respect to $\mathbf{x}$, called the **Hessian matrix** or **Hessian** of $f(\mathbf{x})$, is an $n \times n$ matrix denoted by

$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

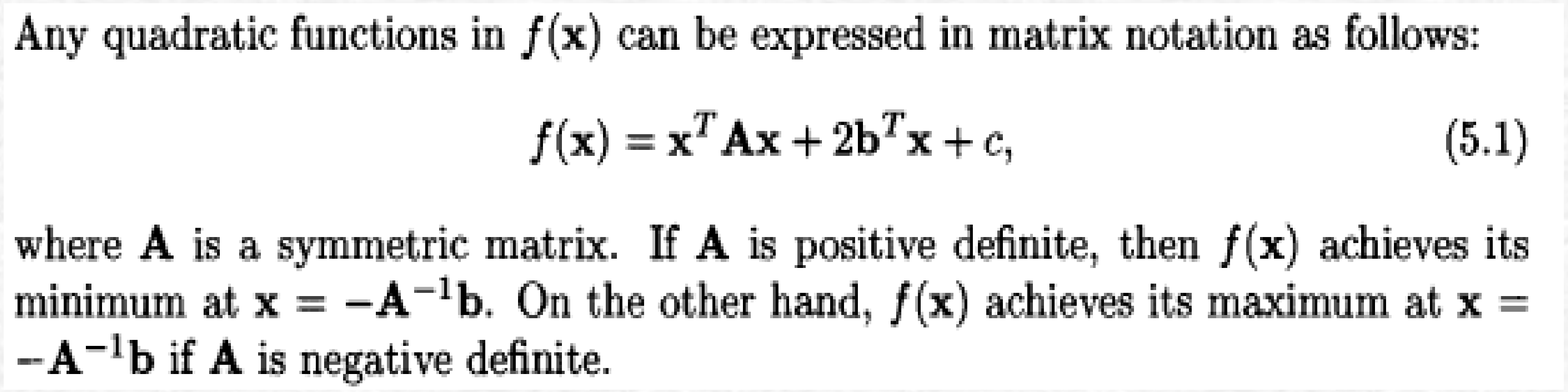
Optimum of a Quadratic Function



A matrix $\mathbf{A}$ is **positive definite**, denoted by $\mathbf{A} > 0$, if $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for all $\mathbf{x} \neq 0$. Alternatively, $\mathbf{A} > 0$ if all its eigenvalues are positive.

□

Conversely, a matrix $\mathbf{A}$ is negative definite if $\mathbf{x}^T\mathbf{A}\mathbf{x} < 0$ for all $\mathbf{x} \neq 0$, or if all its eigenvalues are negative.



Any quadratic functions in $f(\mathbf{x})$ can be expressed in matrix notation as follows:

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + 2\mathbf{b}^T\mathbf{x} + c, \tag{5.1}$$

where $\mathbf{A}$ is a symmetric matrix. If $\mathbf{A}$ is positive definite, then $f(\mathbf{x})$ achieves its minimum at $\mathbf{x} = -\mathbf{A}^{-1}\mathbf{b}$. On the other hand, $f(\mathbf{x})$ achieves its maximum at $\mathbf{x} = -\mathbf{A}^{-1}\mathbf{b}$ if $\mathbf{A}$ is negative definite.

Some Useful Formulas

$$\nabla(\mathbf{x}^T\mathbf{y}) = \nabla(\mathbf{y}^T\mathbf{x}) = \mathbf{y}$$

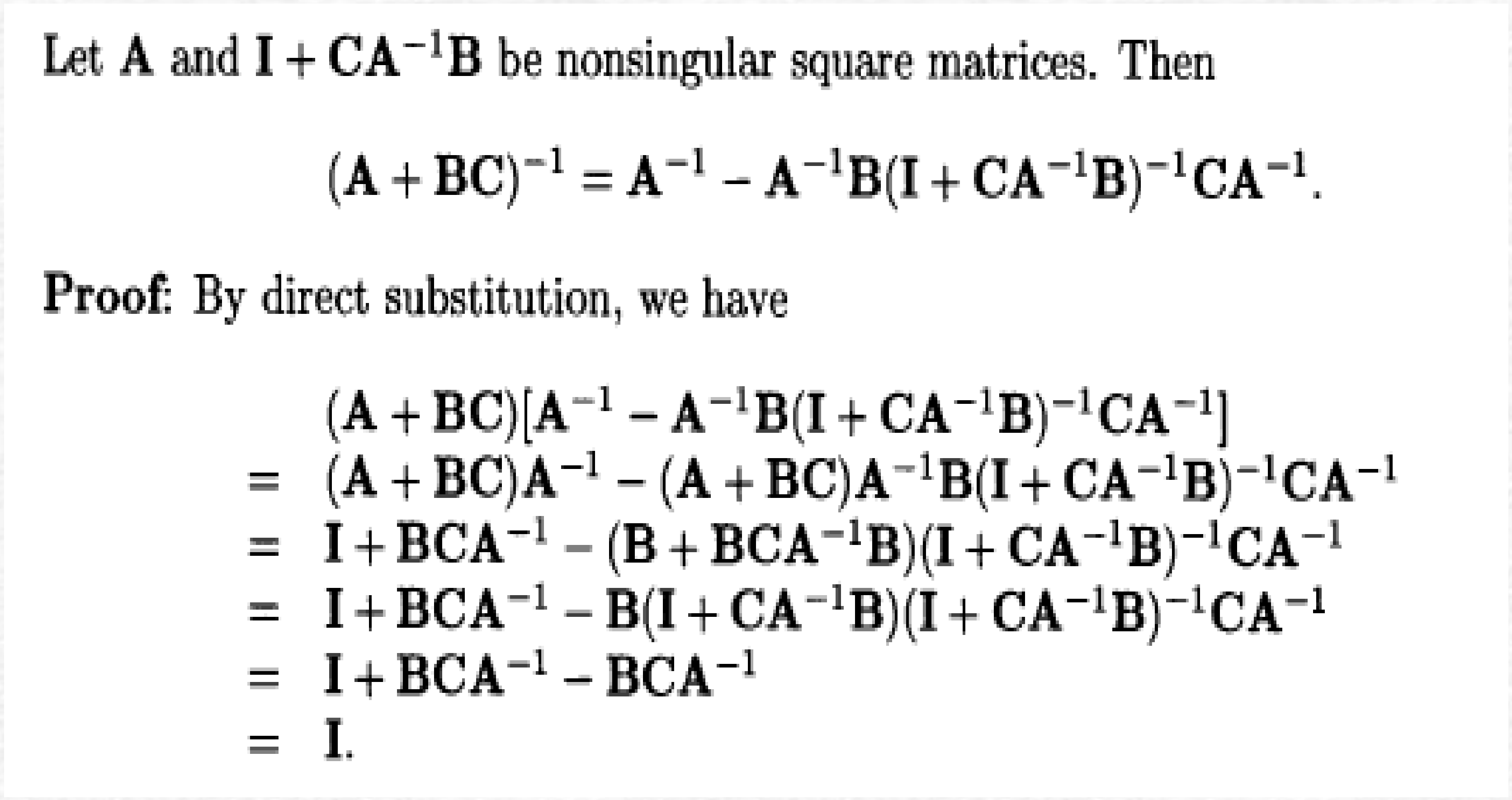
$$\nabla(\mathbf{x}^T\mathbf{x}) = 2\mathbf{x}$$

$$\nabla(\mathbf{x}^T\mathbf{A}\mathbf{y}) = \mathbf{A}\mathbf{y}$$

$$\nabla(\mathbf{y}^T\mathbf{A}\mathbf{x}) = \mathbf{A}^T\mathbf{y}$$

$$\nabla(\mathbf{x}^T\mathbf{A}\mathbf{x}) = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}$$

Matrix Inversion Lemma



Let $\mathbf{A}$ and $\mathbf{I}+\mathbf{C}\mathbf{A}^{-1}\mathbf{B}$ be nonsingular square matrices. Then

$$(\mathbf{A}+\mathbf{B}\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{I}+\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}.$$

Proof: By direct substitution, we have

$$\begin{aligned}
& (\mathbf{A}+\mathbf{B}\mathbf{C})[\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{I}+\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}] \\
= \; & (\mathbf{A}+\mathbf{B}\mathbf{C})\mathbf{A}^{-1} - (\mathbf{A}+\mathbf{B}\mathbf{C})\mathbf{A}^{-1}\mathbf{B}(\mathbf{I}+\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} \\
= \; & \mathbf{I}+\mathbf{B}\mathbf{C}\mathbf{A}^{-1} - (\mathbf{B}+\mathbf{B}\mathbf{C}\mathbf{A}^{-1}\mathbf{B})(\mathbf{I}+\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} \\
= \; & \mathbf{I}+\mathbf{B}\mathbf{C}\mathbf{A}^{-1} - \mathbf{B}(\mathbf{I}+\mathbf{C}\mathbf{A}^{-1}\mathbf{B})(\mathbf{I}+\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1} \\
= \; & \mathbf{I}+\mathbf{B}\mathbf{C}\mathbf{A}^{-1} - \mathbf{B}\mathbf{C}\mathbf{A}^{-1} \\
= \; & \mathbf{I}.
\end{aligned}$$

Taylor Series Expansion

$$f(\mathbf{x}+\mathbf{d}) = f(\mathbf{x}) + \sum_{i=1}^{n} \frac{\partial f(\mathbf{x})}{\partial x_i} d_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} d_i d_j + \text{ H.O.T.}$$

$$f(\mathbf{x}+\mathbf{d}) \approx f(\mathbf{x}) + \mathbf{g}^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \mathbf{H} \mathbf{d}.$$

General form:

$$y = \theta_1 f_1(u) + \theta_2 f_2(u) + \ldots + \theta_n f_n(u) \qquad (*)$$

where:

- $u = (u_1, \ldots, u_p)^T$ is the model input vector
- $f_1, \ldots, f_n$ are known functions of u
- $\theta_1, \ldots, \theta_n$ are unknown parameters to be estimated

- The task of fitting data using a linear model is referred to as linear regression
- We collect a training data set

  $\{(u_i;y_i), i = 1, \ldots, m\}$

  Equation (*) becomes:

$$\begin{cases} f_1(u_1)\theta_1 + f_2(u_1)\theta_2 + \ldots + f_n(u_1)\theta_n = y_1 \\ f_1(u_2)\theta_1 + f_2(u_2)\theta_2 + \ldots + f_n(u_2)\theta_n = y_2 \\ \vdots \\ f_1(u_m)\theta_1 + f_2(u_m)\theta_2 + \ldots + f_n(u_m)\theta_n = y_m \end{cases}$$

  Which is equivalent to: $A\,\theta = y$

Where: A is an m*n matrix which is: $\mathbf{A} = \begin{bmatrix} f_1(u_1)\cdots f_n(u_1) \\ \vdots \\ f_1(u_m)\cdots f_n(u_m) \end{bmatrix}$

θ is n*1 unknown parameter vector:

$$\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

and y is an m*1 output vector:

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \text{and } a_i^T = [f_1(u_i),\dots,f_n(u_i)]$$

$A\,\theta = y \Leftrightarrow \theta = A^{-1}y$ (solution)

- We have m outputs & n fitting parameters to find (or m equations & n unknown variables)

- Usually m is greater than n, since the model is just an approximation of the target system & the data observed might be corrupted, therefore an exact solution is not always possible!

- To overcome this inherent conceptual problem, an error vector e is added to compensate

$$A\,\theta + e = y$$

- Our goal consists now of finding $\hat{\theta}$ that reduces the errors between y_i & $\hat{y}_i = a_i^T\theta$
- $$\min_{\theta} \text{imize} \sum_{i-1}^{m} \left| y_i - a_i^T\theta \right|$$ (not derivable!)

rather

$$\min_{\theta} \text{imize} \sum_{i=1}^{m} \left(y_i - a_i^T\theta \right)^2$$

If $e = y - A\theta$ then:

$$E(\theta) = \sum_{i=1}^{i=m} (y_i - a_i^T \theta)^2 = e^T e = (y - A\theta)^T (y - A\theta)$$

We need to compute:

$$\min_{\theta} (y - A\theta)^T (y - A\theta)$$

Theorem [least-squares estimator]

The squared error is minimized when $\theta = \hat{\theta}$ (called the least-squares estimators LSE) satisfies the normal equation $A^TA\hat{\theta} = A^Ty$, if A^TA is nonsingular, $\hat{\theta}$ is unique & is given by

$$\hat{\theta} = (A^TA)^{-1}A^Ty$$

Proof

$$E(\boldsymbol{\theta}) = (\mathbf{y}^T - \boldsymbol{\theta}^T\mathbf{A}^T)(\mathbf{y} - \mathbf{A}\boldsymbol{\theta}) = \boldsymbol{\theta}^T\mathbf{A}^T\mathbf{A}\boldsymbol{\theta} - 2\mathbf{y}^T\mathbf{A}\boldsymbol{\theta} + \mathbf{y}^T\mathbf{y}.$$

Then the derivative of $E(\boldsymbol{\theta})$ is

$$\frac{\partial E(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 2\mathbf{A}^T\mathbf{A}\boldsymbol{\theta} - 2\mathbf{A}^T\mathbf{y}.$$

By setting $\frac{\partial E(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 0$ at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$, we obtain the normal equation

$$\mathbf{A}^T\mathbf{A}\hat{\boldsymbol{\theta}} = \mathbf{A}^T\mathbf{y}.$$

$$\hat{\boldsymbol{\theta}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}.$$

Example

The relationship is between the spring length & the force applied $L = k_1 f + k_0$ (linear model)

- Goal: find $\hat{\theta} = (k_o, k_1)^T$ that best fits the data for a given force f_o, we need to determine the corresponding spring length L_0
  - Solution 1: provide 2 pairs (L_0, f_0) and (L_1, f_1) and solve a linear system of 2 equations and 2 variables k_1 and k_0 $\Rightarrow \hat{\mathbf{k}}_1$ & $\hat{\mathbf{k}}_0$. However, because of noisy data, this solution is not reliable!
  - Solution 2: use a larger training set (L_i, f_i)

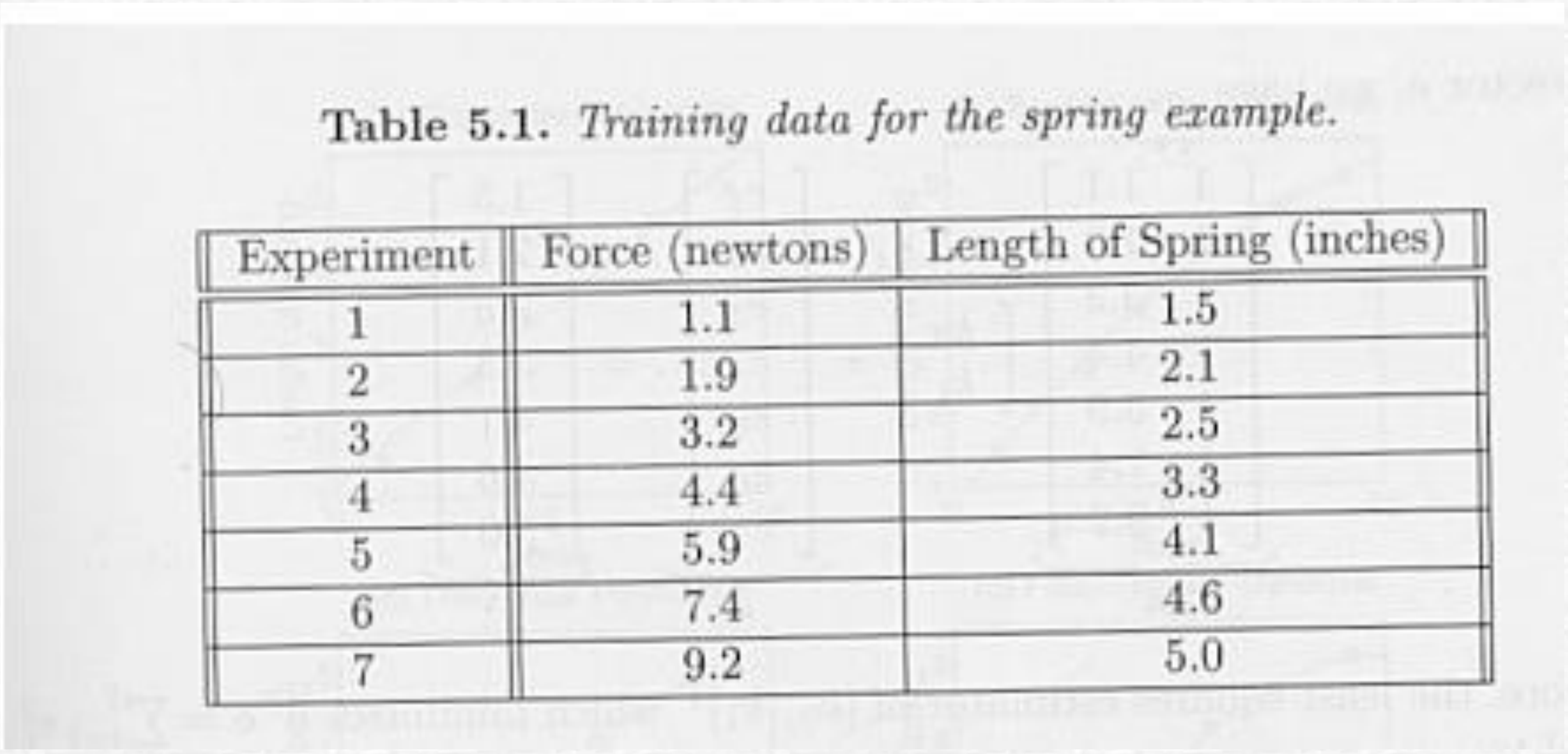


Table 5.1. *Training data for the spring example.*

Experiment	Force (newtons)	Length of Spring (inches)
1	1.1	1.5
2	1.9	2.1
3	3.2	2.5
4	4.4	3.3
5	5.9	4.1
6	7.4	4.6
7	9.2	5.0

since y = e + Aθ, we can write:

$$\underbrace{\begin{bmatrix} 1 & 1.1 \\ 1 & 1.9 \\ 1 & 3.2 \\ 1 & 4.4 \\ 1 & 5.9 \\ 1 & 7.4 \\ 1 & 9.2 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} k_0 \\ k_1 \end{bmatrix}}_{\theta} + \underbrace{\begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ \vdots \\ e_7 \end{bmatrix}}_{\mathbf{e}} = \underbrace{\begin{bmatrix} 1.5 \\ 2.1 \\ 2.5 \\ 3.3 \\ 4.1 \\ 4.6 \\ 5.0 \end{bmatrix}}_{\mathbf{y}}$$

therefore the LSE of $[k_0,k_1]^T$ which minimizes

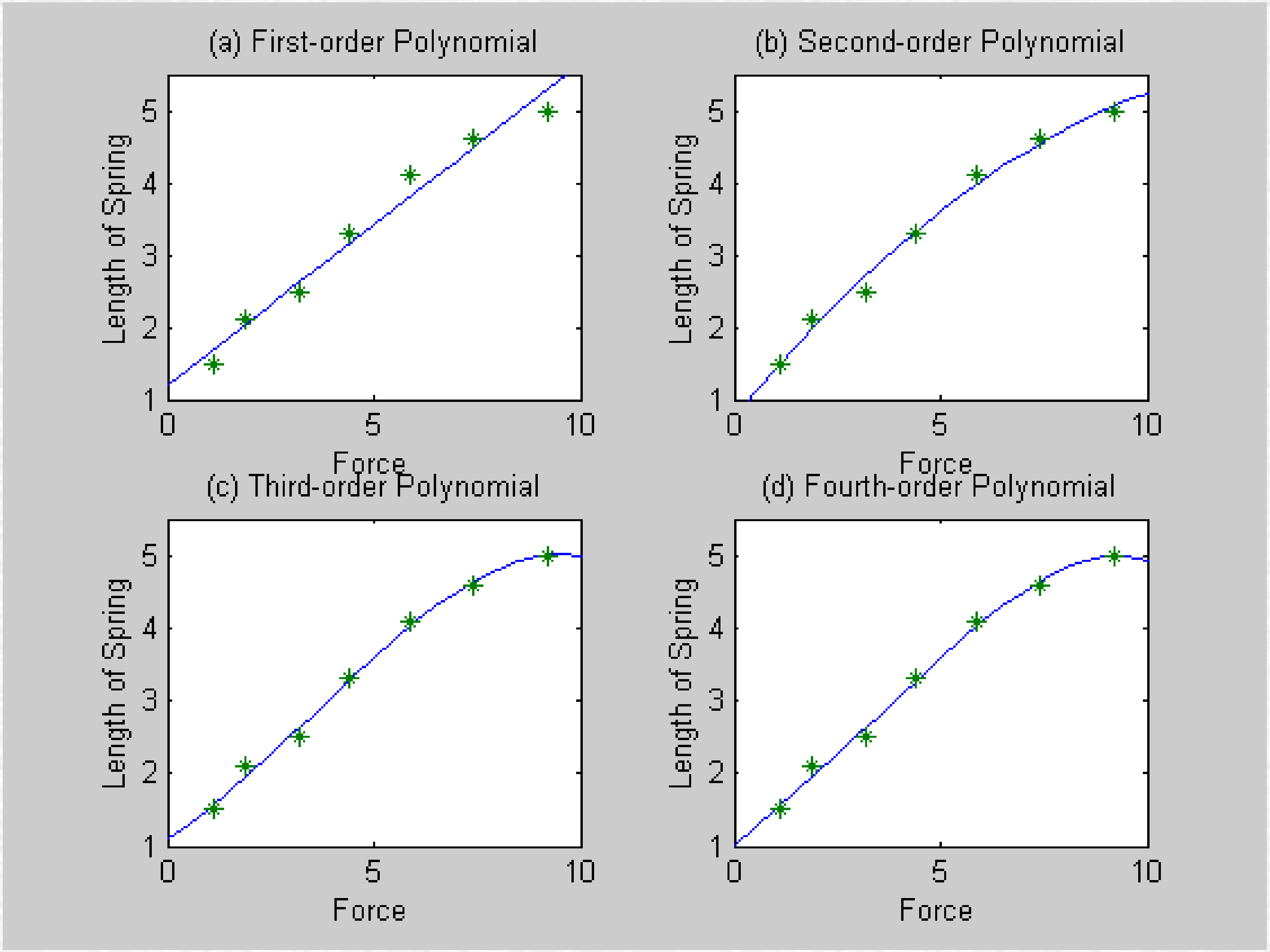
$\mathbf{e}^T\mathbf{e} = \sum_{i=1}^{7} e_i^2$ is equal to : $\begin{bmatrix} \hat{k}_0 \\ \hat{k}_1 \end{bmatrix} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y} = \begin{bmatrix} 1.20 \\ 0.44 \end{bmatrix}$

- We rely on this estimation because we have more data
- If we are not happy with the LSE estimators then we can increase the model's degree of freedom such that:

$$L = k_0 + k_1 f + k_2 f^2 + \dots + k_n f^n$$

(least square polynomial!)

- Higher order models fit better the data but they do not always reflect the inner law that governs the system
- For example, when f is increasing toward 10N, the length is decreasing!


(a) First-order Polynomial
(b) Second-order Polynomial
(c) Third-order Polynomial
(d) Fourth-order Polynomial
Length of Spring
Force
0
5
10
1
2
3
4
5

Derivarion of Least Square

$$\begin{cases} \dfrac{\partial J}{\partial b} = 0 \\ \dfrac{\partial J}{\partial w} = 0 \end{cases}$$

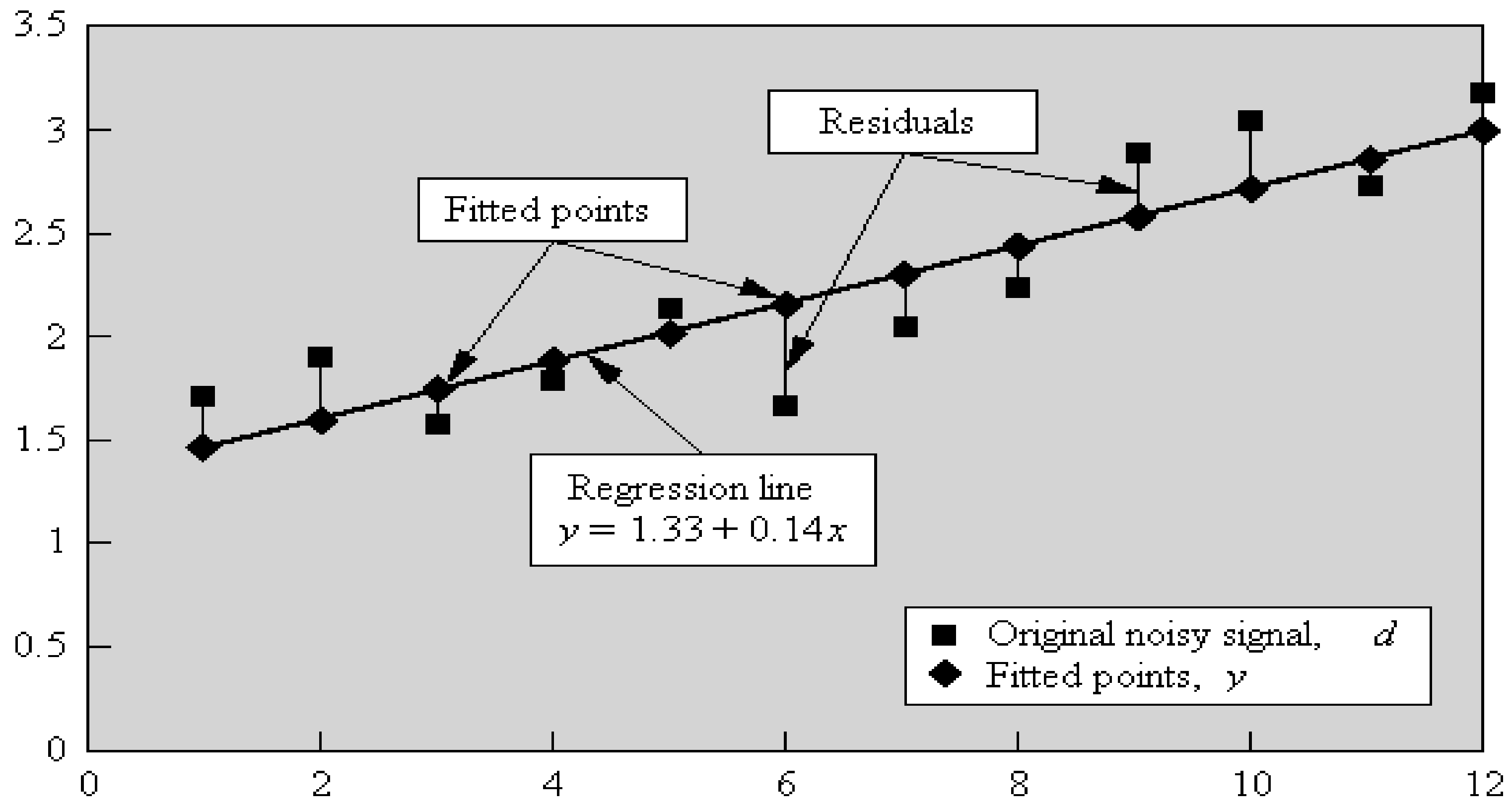
after some manipulation, <u>least square derivation</u>

$$b = \frac{\sum_i x_i^2 \sum_i d_i - \sum_i x_i \sum_i x_i d_i}{N[\sum_i (x_i - \bar{x})^2]} \qquad w = \frac{\sum_i (x_i - \bar{x})(d_i - \bar{d})}{\sum_i (x_i - \bar{x})^2}$$

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

.

$$d_i - (b + wx_i) = d_i - \tilde{d}_i = \varepsilon_i$$

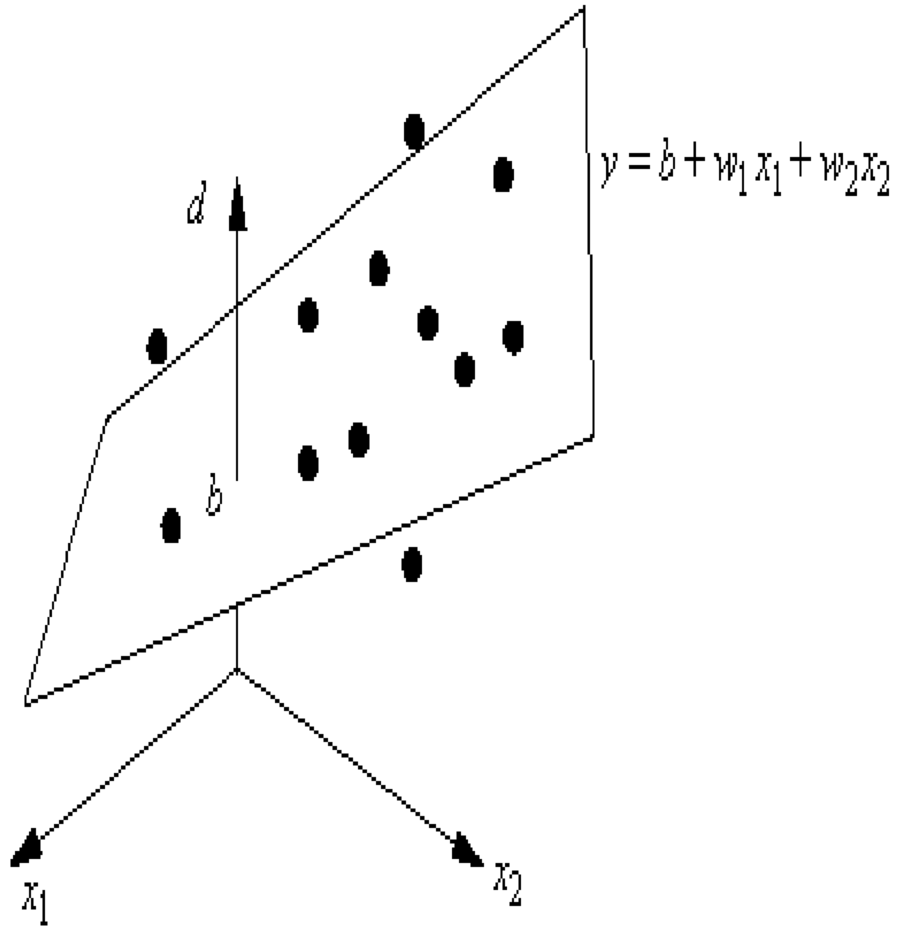


Regression line showing the deviations

Regression for Multiple Variables

Table 1-2 Multiple Regression Data

x1	x2	d
1	2	2
2	5	1
2	3	2
2	2	2
3	4	1
3	5	3
4	6	2
5	5	3
5	6	4
5	7	3
6	8	4
7	6	2
8	4	4
8	9	3
9	8	4



Fitting a regression plane to a set of samples in 2D space.

Least Squares Estimator

$$J = \frac{1}{2N}\sum_i \left(d_i - \sum_{k=0}^{D} w_k x_{ik} \right)^2$$

$$\sum_i x_{ij} d_i = \sum_{k=0}^{D} w_k \sum_i x_{ik} x_{ij} \qquad j = 0,1,\ldots D$$

$$\mathbf{p} = \mathbf{R}\mathbf{w}^* \quad \text{or} \quad \mathbf{w}^* = \mathbf{R}^{-1}\mathbf{p}$$

- The solution to the extreme (minimum) of this equation can be found in exactly the same way as before, that is, by taking the derivatives of J with respect to the unknowns (wk), and equating the result to zero

Recursive Least Square

$$\boldsymbol{\theta}_{k+1} = \left(\begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix}^T \begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix}^T \begin{bmatrix} \mathbf{y} \\ y \end{bmatrix}.$$

$$\mathbf{P}_k = (\mathbf{A}^T\mathbf{A})^{-1}$$

$$\begin{aligned} \mathbf{P}_{k+1} &= \left(\begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix}^T \begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix} \right)^{-1} \\ &= \left(\begin{bmatrix} \mathbf{A}^T & \mathbf{a} \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix} \right)^{-1} \\ &= (\mathbf{A}^T\mathbf{A} + \mathbf{a}\mathbf{a}^T)^{-1}. \end{aligned}$$

These two matrices are related by

$$\mathbf{P}_k^{-1} = \mathbf{P}_{k+1}^{-1} - \mathbf{a}\mathbf{a}^T.$$

Using $\mathbf{P}_k$ and $\mathbf{P}_{k+1}$, we have

$$\begin{cases} \boldsymbol{\theta}_k & = & \mathbf{P}_k \mathbf{A}^T \mathbf{y}, \\ \boldsymbol{\theta}_{k+1} & = & \mathbf{P}_{k+1}(\mathbf{A}^T \mathbf{y} + \mathbf{a} y). \end{cases}$$

$$\mathbf{A}^T \mathbf{y} = \mathbf{P}_k^{-1} \boldsymbol{\theta}_k.$$

$$\begin{aligned} \boldsymbol{\theta}_{k+1} &= \mathbf{P}_{k+1}(\mathbf{P}_k^{-1} \boldsymbol{\theta}_n + \mathbf{a} y) \\ &= \mathbf{P}_{k+1}[(\mathbf{P}_{k+1}^{-1} - \mathbf{a}\mathbf{a}^T)\boldsymbol{\theta}_k + \mathbf{a} y] \\ &= \boldsymbol{\theta}_k + \mathbf{P}_{k+1}\mathbf{a}(y - \mathbf{a}^T \boldsymbol{\theta}_k). \end{aligned}$$

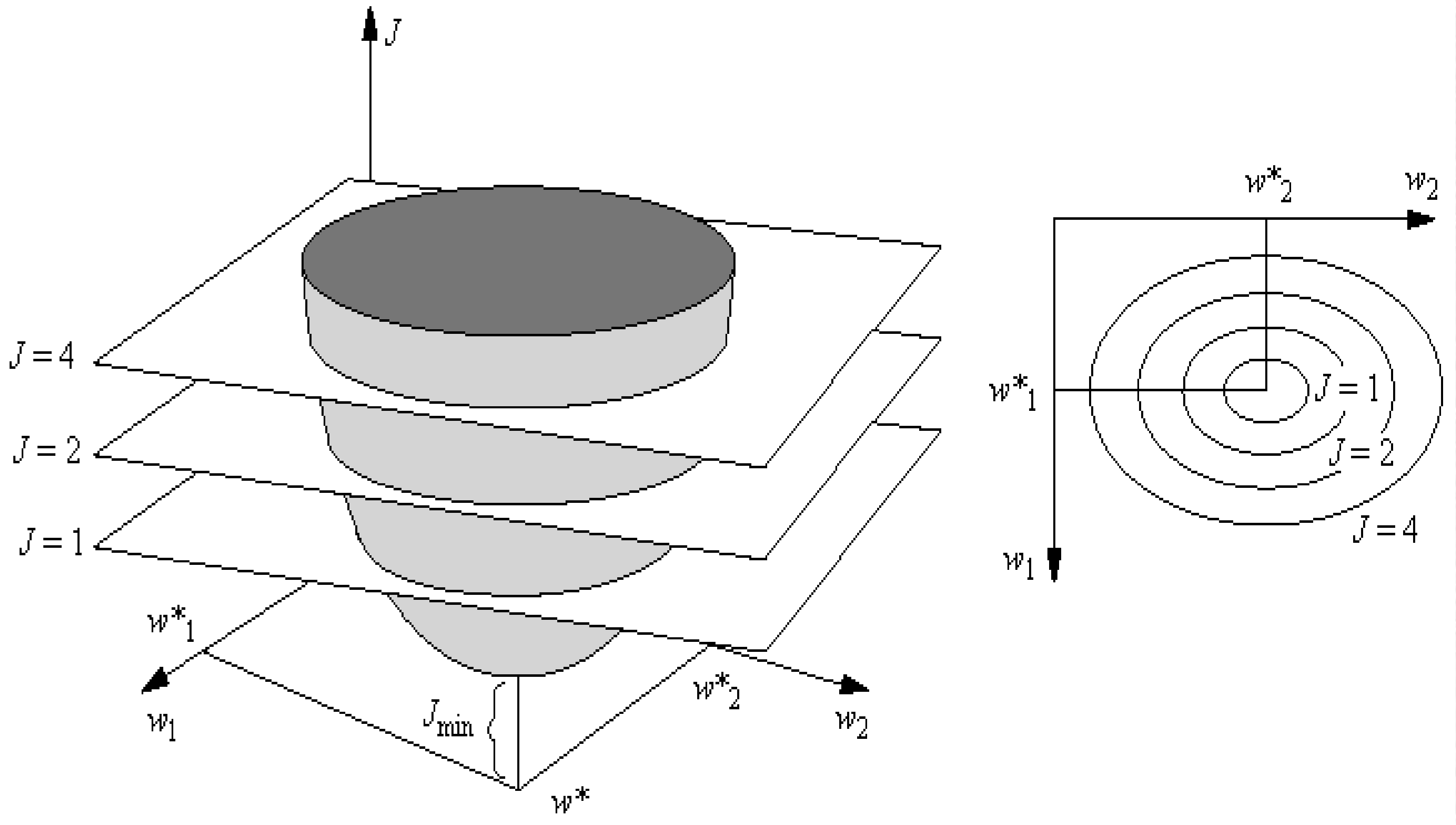
$$\mathbf{P}_{k+1} = (\mathbf{P}_k^{-1} + \mathbf{a}\mathbf{a}^T)^{-1}.$$

$$\begin{aligned} \mathbf{P}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k a (\mathbf{I} + \mathbf{a}^T \mathbf{P}_k \mathbf{a})^{-1} \mathbf{a}^T \mathbf{P}_k \\ &= \mathbf{P}_k - \frac{\mathbf{P}_k \mathbf{a}\mathbf{a}^T \mathbf{P}_k}{1 + \mathbf{a}^T \mathbf{P}_k \mathbf{a}}. \end{aligned}$$

Summary

$$\left\{\begin{array}{lcl} \mathbf{P}_{k+1} & = & \mathbf{P}_k - \dfrac{\mathbf{P}_k \mathbf{a}_{k+1} \mathbf{a}_{k+1}^T \mathbf{P}_k}{1 + \mathbf{a}_{k+1}^T \mathbf{P}_k \mathbf{a}_{k+1}}, \\ \theta_{k+1} & = & \theta_k + \mathbf{P}_{k+1} \mathbf{a}_{k+1} \left(y_{k+1} - \mathbf{a}_{k+1}^T \theta_k \right), \end{array}\right.$$

Search Procedure



The performance surface for two dimensions and its contour plot

Weight Adjustments

- Weights are initialized to arbitrary values
- Weights are continuously adjusted until the next sampling period
- Weight Adjustment Process
- A basic error correction mechanism
- $y(i) = v(i) = X^T(i)W(i)$
- $e(i) = d(i) - y(i)$
- $W(i) = f\ [e(i)]$