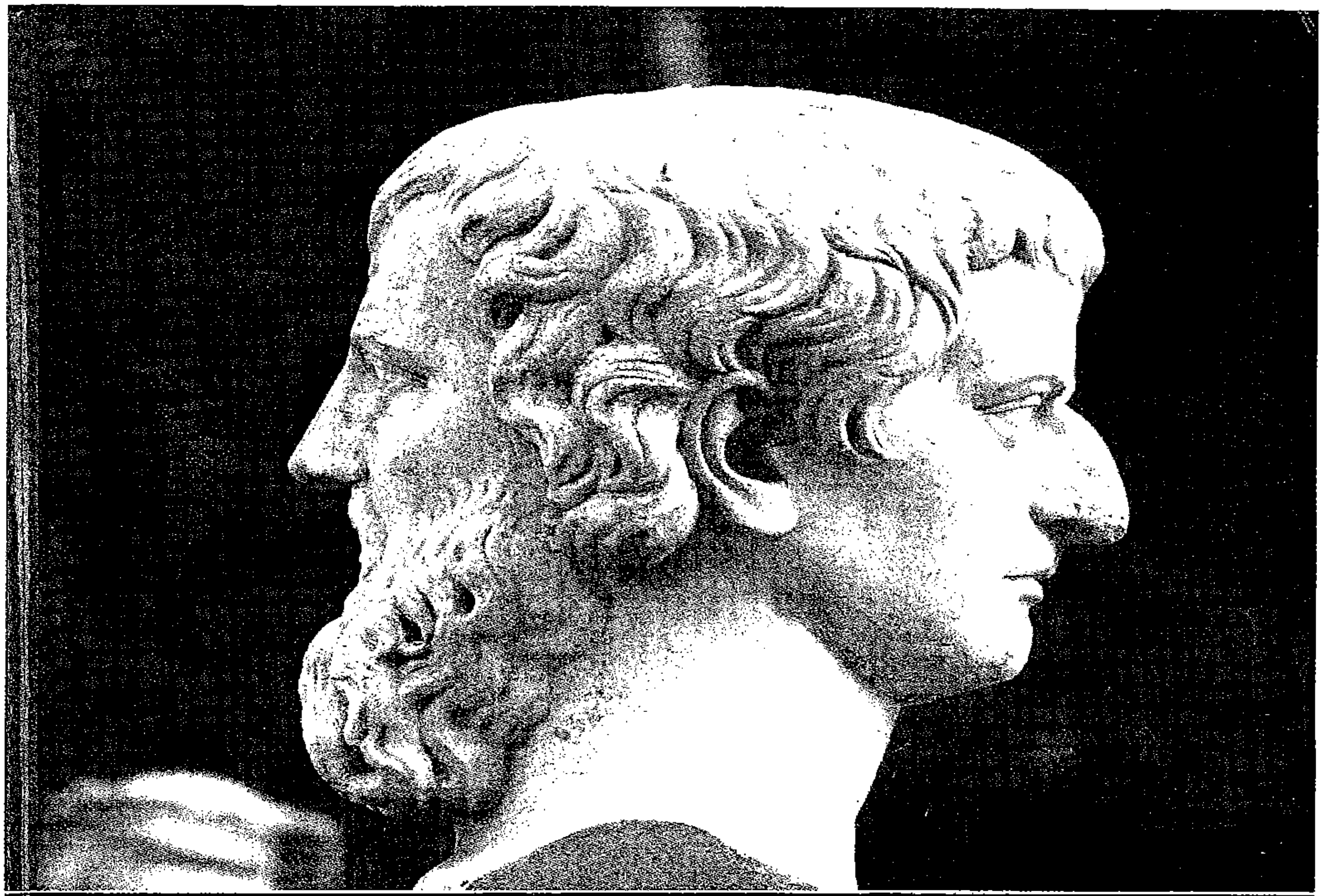
EXHIBIT G

THIS EXHIBIT HAS BEEN
REDACTED IN ITS ENTIRETY

EXHIBIT H



LEARNING FROM DATA

Concepts, Theory, and Methods

SECOND EDITION

VLADIMIR CHERKASSKY • FILIP MULIER

Library of Congress Cataloging-in-Publication Data:

Cherkassky, Vladimir S.
Learning from data : concepts, theory, and methods / by Vladimir Cherkassky, Filip Mulier. – 2nd ed.
p. cm.
ISBN 978-0-471-68182-3 (cloth)
1. Adaptive signal processing. 2. Machine learning. 3. Neural networks (Computer science) 4. Fuzzy systems. I. Mulier, Filip. II. Title.
TK5102.9.C475 2007
006.3′1–dc22 2006038736

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

PREFACE

There are two problems in modern science:

- too many people use different terminology to solve the same problems;
- even more people use the same terminology to address completely different issues.

Anonymous

In recent years, there has been an explosive growth of methods for learning (or estimating dependencies) from data. This is not surprising given the proliferation of

- low-cost computers (for implementing such methods in software)
- low-cost sensors and database technology (for collecting and storing data)
- highly computer-literate application experts (who can pose "interesting" application problems)

A learning method is an algorithm (usually implemented in software) that estimates an unknown mapping (dependency) between a system's inputs and outputs from the available data, namely from known (input, output) samples. Once such a dependency has been accurately estimated, it can be used for prediction of future system outputs from the known input values. This book provides a unified description of principles and methods for learning dependencies from data.

Methods for estimating dependencies from data have been traditionally explored in diverse fields such as statistics (multivariate regression and classification), engineering (pattern recognition), and computer science (artificial intelligence, machine

2.1 FORMULATION OF THE LEARNING PROBLEM

Learning is the process of estimating an unknown (input, output) dependency or structure of a System using a limited number of observations. The general learning scenario involves three components (Fig. 2.1): a *Generator* of random input vectors, a *System* that returns an output for a given input vector, and the *Learning Machine* that estimates an unknown (input, output) mapping of the System from the observed (input, output) samples. This formulation is very general and describes many practical learning problems found in engineering and statistics, such as interpolation, regression, classification, clustering, and density estimation. Before we look at the learning machine in detail, let us clearly describe the roles of each component in mathematical terms:

Generator: The generator (or sampling distribution) produces random vectors $\mathbf{x} \in \Re^d$ drawn independently from a fixed probability density $p(\mathbf{x})$, which is unknown. In statistical terminology, this situation is called observational. It differs from the designed experiment setting, which involves creating a deterministic sampling scheme optimal for a specific analysis according to experiment design theory. In this book, the observational setting is usually assumed; that is, a modeler (learning machine) has had no control over which input values were supplied to the System.

System: The system produces an output value y for every input vector $\mathbf{x}$ according to the fixed conditional density $p(y|\mathbf{x})$, which is also unknown. Note that this description includes the specific case of a deterministic system, where $y = t(\mathbf{x})$, as well as the regression formulation of $y = t(\mathbf{x}) + \xi$, where ξ is random noise with zero mean. Real systems rarely have truly random outputs; however, they often have unmeasured inputs (Fig. 1.1). Statistically, the effect of these changing unobserved inputs on the output of the System can be characterized as random and represented as a probability distribution.

Learning Machine: In the most general case, the Learning Machine is capable of implementing a set of functions $f(\mathbf{x}, \omega)$, $\omega \in \Omega$, where Ω is a set of abstract

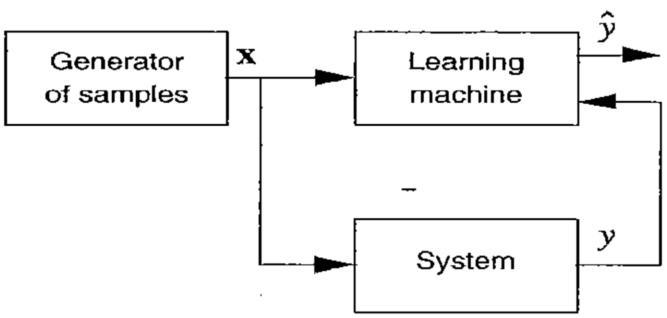


FIGURE 2.1 A Learning Machine using observations of the System to form an approximation of its output.

parameters used only to index the set of functions. In this formulation, the set of functions implemented by the Learning Machine can be any set of functions, chosen a priori, before the formal inference (learning) process has begun. Let us look at some simple examples of Learning Machines and how they fit this formal description. The examples chosen are all solutions to the regression problem, which is only one of the four most common learning tasks (Section 2.1.2). The examples illustrate the notion of a set of functions (of a Learning Machine) and not the mechanism by which the Learning Machine chooses the best approximating function from this set.

Example 2.1: Parametric regression (fixed-degree polynomial)

In this example, the set of functions is specified as a polynomial of fixed degree and the training data have a single predictor variable ($x \in \Re^1$). The set of functions implemented by the Learning Machine is

$$f(x, \mathbf{w}) = \sum_{i=0}^{M-1} w_i x^i, \tag{2.1}$$

where the set of parameters Ω takes the form of vectors $\mathbf{w} = [w_0, \ldots, w_{M-1}]$ of fixed length M.

Example 2.2: Semiparametric regression (polynomial of arbitrary degree)

One way to provide a wider class of functions for the Learning Machine is to remove the restriction of fixed polynomial degree. The degree of the polynomial now becomes another parameter that indexes the set of functions

$$f_m(x, \mathbf{w}_m) = \sum_{i=0}^{m-1} w_i x^i. \tag{2.2}$$

Here the set of parameters Ω takes the form of vectors $\mathbf{w}_m = [w_0, \ldots, w_{m-1}]$, which have an arbitrary length m.

Example 2.3: Nonparametric regression (kernel smoothing)

Additional flexibility can also be achieved by using a nonparametric approach like kernel averaging to define the set of functions supported by the Learning Machine. Here the set of functions is

$$f_\alpha(x, \mathbf{w}_n | \mathbf{x}_n) = \frac{\sum_{i=1}^{n} w_i K_\alpha(x, x_i)}{\sum_{i=1}^{n} K_\alpha(x, x_i)}, \tag{2.3}$$

where n is the number of samples and $K_\alpha(x, x')$ is called the *kernel* function with bandwidth α. For the general case $\mathbf{x} \in \Re^d$, the kernel function $K(\mathbf{x}, \mathbf{x}')$ obeys the following properties:

1. $K(\mathbf{x}, \mathbf{x}')$ takes on its maximum value when $\mathbf{x}' = \mathbf{x}$
2. $|K(\mathbf{x}, \mathbf{x}')|$ decreases with $|\mathbf{x} - \mathbf{x}'|$
3. $K(\mathbf{x}, \mathbf{x}')$ is in general a symmetric function of $2d$ variables

Usually, the kernel function is chosen to be radially symmetric, making it a function of one variable $K(\eta)$, where η is the scaled distance between $\mathbf{x}$ and $\mathbf{x}'$:

$$\eta = \frac{|\mathbf{x} - \mathbf{x}'|}{s(\mathbf{x})}.$$

The scale factor $s(\mathbf{x})$ defines the size (or width) of the region around $\mathbf{x}$ for which K is large. It is common to set the scale factor to a constant value $s(\mathbf{x}) = \alpha$, which is the form of the kernel used in our example equation (2.3). An example of a typical kernel function is the Gaussian

$$K_\alpha(x, x') = \exp\left(-\frac{(x - x')^2}{2\alpha^2}\right). \tag{2.4}$$

In this Learning Machine, the set of parameters Ω takes the form of vectors $[\alpha, w_1, \ldots, w_n]$ of a fixed length that depends on the number of samples n. In this example, it is assumed that the input samples $\mathbf{x}_n = [x_1, \ldots, x_n]$ are used in the specification of the set of approximating functions of the Learning Machine. This is formally stated in (2.3) by having the set of approximating functions conditioned on the given vector of predictor sample values. The previous two examples did not use input samples for specifying the set of functions.

Choice of approximating functions: Ideally, the choice of a set of approximating functions reflects a priori knowledge about the System (unknown dependency). However, in practice, due to complex and often informal nature of a priori knowledge, such specification of approximating functions may be difficult or impossible. Hence, there may be a need to incorporate a priori knowledge into the learning method with an already given set of approximating functions. These issues are discussed in more detail in Section 2.3. There is also an important distinction between two types of approximating functions: linear in parameters or nonlinear in parameters. Throughout this book, learning (estimation) procedures using the former are also referred to as *linear*, whereas those using the latter are called *nonlinear*. We point out that the notion of linearity is with respect to parameters rather than input variables. For example, polynomial regression (2.2) is a linear method. Another example of a linear class of approximating functions (for regression) is the trigonometric expansion

$$f_m(x, \mathbf{v}_m, \mathbf{w}_m) = \sum_{j=1}^{m-1} (v_j \sin(jx) + w_j \cos(jx)) + w_0.$$

On the contrary, multilayer networks of the form

$$f_m(\mathbf{x}, \mathbf{w}, V) = w_0 + \sum_{j=1}^{m} w_j g\left(v_{0j} + \sum_{i=1}^{d} x_i v_{ij}\right)$$

provide an example of nonlinear parameterization because it depends nonlinearly on parameters V via nonlinear basis function g (usually taken as the so-called sigmoid activation function).

The distinction between linear and nonlinear methods is important in practice because learning (estimation) of model parameters amounts to solving a linear or nonlinear optimization problem, respectively.

2.1.1 Objective of Learning

As noted in Section 1.5, there may be two distinct interpretations of the goal of learning for generic system shown in Fig. 2.1. Under statistical model estimation framework, the goal of learning is accurate *identification* of the unknown system, whereas under *predictive learning* the goal is accurate *imitation* (of a system's output). It should be clear that the goal of system identification is more demanding than the goal of system imitation. For instance, accurate system identification does not depend on the distribution of input samples, whereas good predictive model is usually conditional upon this (unknown) distribution. Hence, an accurate model (in the sense of system's identification) would certainly provide good generalization (in the predictive sense), but the opposite may not be true. The mathematical treatment of system identification leads to the function approximation framework and to fundamental problems of estimating multivariate functions known as the curse of dimensionality (see Chapter 3). On the contrary, the goal of predictive learning leads to Vapnik–Chervonenkis (VC) learning theory described later in Chapter 4. This book advocates the setting of predictive learning, which formally defines the notion of accurate system imitation (via minimization of prediction risk) as described in this section. We contrast the function approximation approach versus predictive learning throughout the book, in particular, using empirical comparisons in Section 3.4.5.

The problem encountered by the Learning Machine is to select a function (from the set of functions it supports) that best approximates the System's response. The Learning Machine is limited to observing a finite number (n) of examples in order to make this selection. These training data as produced by the Generator and System will be independent and identically distributed (iid) according to the joint probability density function (pdf)

$$p(\mathbf{x}, y) = p(\mathbf{x})p(y|\mathbf{x}). \tag{2.5}$$

The finite sample (training data) from this distribution is denoted by

$$(\mathbf{x}_i, y_i), \qquad (i = 1, \ldots, n). \tag{2.6}$$