KERNEL ADAPTIVE FILTERING

A Comprehensive Introduction

Weifeng Liu, José C. Príncipe, and
Simon Haykin

WILEY

JOHN WILEY & SONS, INC., PUBLICATION

KERNEL ADAPTIVE FILTERING

Adaptive and Learning Systems for Signal Processing, Communication, and Control

Editor: Simon Haykin

A complete list of titles in this series appears at the end of this volume.

KERNEL ADAPTIVE FILTERING

A Comprehensive Introduction

Weifeng Liu, José C. Príncipe, and
Simon Haykin

WILEY

JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

Library of Congress Cataloging-in-Publication Data:

Liu, Weifeng
Kernel adaptive filtering : a comprehensive introduction / Jose C. Principle, Weifeng Liu, Simon Haykin.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-470-44753-6
1. Adaptive filters. 2. Kernel functions. I. Príncipe, José C. II. Haykin, Simon S., 1931– III. Title.
TK7872.F5P745 2010
621.382′23–dc22

2009042654

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

To our families

CONTENTS

PREFACE

For the first time, this book presents a comprehensive and unifying introduction to kernel adaptive filtering. Adaptive signal processing theory has been built on three pillars: the linear model, the mean square cost, and the adaptive least-square learning algorithm. When nonlinear models are required, the simplicity of linear adaptive filters evaporates and a designer has to deal with function approximation, neural networks, local minima, regularization, and so on. Is this the only way to go beyond the linear solution? Perhaps there is an alternative, which is the focus of this book. The basic concept is to perform adaptive filtering in a linear space that is related nonlinearly to the original input space. If this is possible, then all three pillars and our intuition about linear models can still be of use, and we end up implementing nonlinear filters in the input space.

This book will draw on the theory of reproducing kernel Hilbert spaces (RKHS) to implement the nonlinear transformation of the input to a high-dimensional feature space induced by a positive-definite function called *reproducing kernel*. If the filtering and adaptation operations to be performed in RKHS can be expressed by inner products of projected samples, then they can be directly calculated by kernel evaluations in the input space. We use this approach to introduce a family of adaptive filtering algorithms in RKHS:

- The kernel least-mean-square algorithm
- The kernel affine projection algorithms
- The kernel recursive least-squares algorithm
- The extended kernel recursive least-squares algorithm

These kernel-learning algorithms bridge closely two important areas of adaptive filtering and neural networks, and they embody beautifully two important methodologies of error-correction learning and memory-based learning. The bottlenecks of the RKHS approach to nonlinear filter design are the need for regularization, the need to select the kernel function, and the need to curtail the growth of the filter structure. This book will present in a mathematically rigorous manner the issues and the solutions to all these

problems, and it will illustrate with examples the performance gains of kernel adaptive filtering.

Chapter 1 starts with an introduction to general concepts in machine learning, linear adaptive filters, and conventional nonlinear methods. Then, the theory of reproducing kernel Hilbert spaces is presented as the mathematical foundation of kernel adaptive filters. We stress that kernel adaptive filters are universal function approximators, have no local minima during adaptation, and require reasonable computational resources.

Chapter 2 studies the kernel least-mean-square algorithm, which is the simplest among the family of kernel adaptive filters. We develop the algorithm in a step-by-step manner and delve into all the practical aspects of selecting the kernel function, picking the step-size parameter, sparsification, and regularization. Two computer experiments, one with Mackey–Glass chaotic time-series prediction and the other with nonlinear channel equalization, are presented.

Chapter 3 covers the kernel affine projection algorithms, which is a family of four similar algorithms. The mathematical equations of filtering and adaptation are thoroughly derived from first principles, and useful implementation techniques are discussed fully. Many well-known methods can be derived as special cases of the kernel affine projection algorithms. Three detailed applications are included to show their wide applicability and design flexibility.

Chapter 4 presents the kernel recursive least-squares algorithm and the theory of Gaussian process regression. A sparsification approach called approximate linear dependency is discussed. And with the aid of the Bayesian interpretation, we also present a powerful model selection method called "maximum marginal likelihood". Two computer experiments are conducted to study the performance of different sparsification schemes and the effectiveness of maximum marginal likelihood to determine the kernel parameters.

Chapter 5 discusses the extended kernel recursive least-squares algorithm on the basis of the kernel recursive least-squares algorithm. We study systematically the problem of estimating the state of a linear dynamic system in RKHS from a sequence of noisy observations. Several important theorems are presented with proofs to outline the significance and basic approaches. This chapter contains two examples, Rayleigh channel tracking and Lorenz time-series modeling.

Chapter 6 is devoted to addressing the principal bottleneck of kernel adaptive filters, i.e., their growing structure. We introduce a subjective information measure called *surprise* and present a unifying sparsification scheme to curtail the growth effectively of kernel adaptive filters. Three interesting computer simulations are presented to illustrate the theories.

This book should appeal to engineers, computer scientists, and graduate students who are interested in adaptive filtering, neural networks, and kernel methods. A total of 12 computer-oriented experiments are distributed throughout the book that have been designed to reinforce the concepts discussed in the chapters. The computer experiments are listed in Table 1. Their MATLAB®

Table 1. A listing of all computer experiments in the book. MATLAB® programs that generate the results can be downloaded by all readers from the book's website http://www.cnel.ufl.edu/~weifeng/publication.htm.

Computer experiment	Topic
2.1	KLMS Applied to Mackey–Glass Time-Series Prediction
2.2	KLMS Applied to Nonlinear Channel Equalization
3.1	KAPA Applied to Mackey–Glass Time-Series Prediction
3.2	KAPA Applied to Noise Cancellation
3.3	KAPA Applied to Nonlinear Channel Equalization
4.1	KRLS Applied to Mackey–Glass Time-Series Prediction
4.2	Model Selection by Maximum Marginal Likelihood
5.1	EX-KRLS Applied to Rayleigh Channel Tracking
5.2	EX-KRLS Applied to Lorenz Time-Series Prediction
6.1	Surprise Criterion Applied to Nonlinear Regression
6.2	Surprise Criterion Applied to Mackey–Glass Time-Series Prediction
6.3	Surprise Criterion Applied to CO_2 Concentration Forecasting

implementations can be downloaded directly from the website http://www.cnel.ufl.edu/~weifeng/publication.htm. To keep the codes readable, we placed simplicity over performance during design and implementation. These programs are provided without any additional guarantees.

We have strived to reflect fully the latest advances of this emerging area in the book. Each chapter concludes with a summary of the state of the art and potential future directions for research. This book should be a useful guide to both those who look for nonlinear adaptive filtering methodologies to solve practical problems and those who seek inspiring research ideas in related areas.

ACKNOWLEDGMENTS

We would like to start by thanking Dr. Puskal P. Pokharel, Seagate Technology; Dr. Murali Rao, University of Florida; Dr. Jay Gopalakrishnan, University of Florida; and Il Park, University of Florida, for their help in the development of the kernel adaptive filtering theory. We are most grateful to Dr. Aysegul Gunduz, Albany Medical College, Wadsworth Center; Dr. John Harris, University of Florida; and Dr. Steven Van Vaerenbergh, University of Cantabria, Spain, for providing many useful comments and constructive feedback on an early version of the manuscript of the book.

Many others have been kind enough to read critically selected chapters/sections of the book; in alphabetical order, they are as follows:

Erion Hasanbelliu, University of Florida, Gainesville, Florida

Dr. Kyu-hwa Jeong, Intel Corporation, Santa Clara, California

Dr. Ruijiang Li, University of California, San Diego, California

Dr. Antonio Paiva, University of Utah, Salt Lake City, Utah

Alexander Singh, University of Florida, Gainesville, Florida

Dr. Yiwen Wang, Hong Kong University of Science and Technology, Hong Kong

Dr. Jianwu Xu, University of Chicago, Chicago, Illinois

We also wish to thank (in alphabetical order): Dr. Peter Bartlett, University of California, Berkeley; Dr. Andrzej Cichocki, Riken, Brain Science Institute, Japan; Dr. Corinna Cortes, Google Lab; Dr. Graham C. Goodwin, University of Newcastle, UK; Dr. Michael Jordan, University of California, Berkeley; Dr. Thomas Kailath, Stanford University; Dr. Joel S. Kvitky, Rand Corporation; Dr. Yann LeCun, New York University; Dr. Derong Liu, University of Illinois at Chicago; Dr. David J. C. MacKay, University of Cambridge, UK; Dr. Tomaso Poggio, Massachusetts Institute of Technology; Dr. Ali H. Sayed, University of California, Los Angeles; Dr. Bernhard Schölkopf, Max Planck Institute for Biological Cybernetics, Germany; Dr. Sergios Theodoridis, University of Athens, Greece; Dr. Yoram Singer, Google Lab; Dr. Alexander J. Smola, Yahoo! research; Dr. Johan Suykens, Katholieke Universiteit

Leuven, Belgium; Dr. Paul Werbos, The National Science Foundation; Dr. Bernard Widrow, Stanford University; and Dr. Chris Williams, University of Edinburgh, UK.

We thank the staff at Wiley, publisher George Telecki, editorial assistant Lucy Hitz, and production editor Kris Parrish, as well as the project manager, Stephanie Sakson from Toppan Best-Set Premedia, for their full support and encouragement in preparing the manuscript of the book and for all their behind-the-scenes effort in the selection of the book cover and the production of the book.

Last, but by no means least, we are grateful to Lola Brooks, McMaster University, for typing many sections of the manuscript.

NOTATION

The book discusses many algorithms involving various mathematical equations. A convenient and uniform notation is a necessity to convey clearly the basic ideas of the kernel adaptive filtering theory. We think it is helpful to summarize and explain at the beginning of the text our notational guidelines for ease of reference.

There are mainly *three* types of variables we need to distinguish:

scalar, vector, and matrix variables

The following is a list of the notational conventions used in the book:

1. We use *small italic* letters to denote *scalar variables*. For example, the output of a filter is a scalar variable, which is denoted by y.
2. We use *CAPITAL ITALIC* letters to denote *SCALAR CONSTANTS*. For example, the order of a filter is a scalar constant, which is denoted by L.
3. We use **small bold** letters for **vectors**.
4. We use **CAPITAL BOLD** letters to denote **MATRICES**.
5. We use *parentheses* to denote the *time dependency* of any variables (either scalar, vector, or matrix). For example, $d(i)$ means the value of a scalar d at time (or iteration) i. $\mathbf{u}(i)$ means the value of a vector $\mathbf{u}$ at time (or iteration) i. Similarly $\mathbf{G}(i)$ means the value of a matrix $\mathbf{G}$ at time (or iteration) i. There is no rule without an exception. f_i is used to denote the estimate of an input–output mapping f at time (or iteration) i since parenthesis is preserved for input argument like $f_i(\mathbf{u})$.
6. We use the superscript T to denote *transposition*. For example, if

$$\mathbf{d} = \begin{bmatrix} d(1) \\ d(2) \\ \cdots \\ d(N) \end{bmatrix}$$

then

$$\mathbf{d}^T = [d(1), d(2), \ldots, d(N)].$$

7. All variables in our presentation are *real*. We do not discuss complex numbers in this book.
8. All vectors in our presentation are *column vectors* without exception.
9. We use subscript indices to denote 1) a component of a vector (or a matrix), 2) a general vector that the index is not related to time (or iteration). For example, $\mathbf{c}_i$ could mean the ith vector in some set or the ith component of the vector $\mathbf{c}$ according to the context.

We have made every effort to make the notation consistent and coherent for the benefit of the reader. The following Table 2 summarizes and lists some typical examples.

Table 2. Notation.

	Description	Examples
Scalars	Small *italic* letters	d
Vectors	Small **bold** letters	$\mathbf{w}$, $\boldsymbol{\omega}$, $\mathbf{c}_i$
Matrices	Capital **BOLD** letters	$\mathbf{U}$, $\boldsymbol{\Phi}$
Time or iteration	Indices in parentheses	$\mathbf{u}(i)$, $d(i)$
Component of vectors/matrices	Subscript indices	$\mathbf{a}_j$, $\mathbf{G}_{i,j}$
Linear spaces	Capital mathbb letters	$\mathbb{F}$, $\mathbb{H}$
Scalar constants	Capital *ITALIC* letters	L, N

ABBREVIATIONS AND SYMBOLS

We collect here a list of the main abbreviations and symbols used throughout the text for ease of reference.

$(\cdot)^T$	vector or matrix transposition
$\mathbf{A}^{-1}$	inverse of matrix $\mathbf{A}$
$\mathbf{E}[\cdot]$	expected value of a random variable
$m(\cdot)$	the mean of a random variance
$\sigma^2(\cdot)$	the variance of a random variable
$<\cdot,\cdot>$	inner product
$\|\cdot\|$	norm of a vector; square root of the inner product with itself
$\|\cdot\|$	absolute value of a real number or determinant of a matrix
$\propto$	proportional to
$\sim$	distributed according to
∇	gradient
$\mathbf{0}$	zero vector or matrix
β	forgetting factor
$\mathcal{C}(i)$	dictionary or center set at iteration i
$d(i)$	desired output at time or iteration i (a real scalar)
diag$\{a, b\}$	a diagonal matrix with diagonal entries a and b
δ_1	distance threshold in novelty criterion
δ_2	prediction error threshold in novelty criterion
δ_3	threshold in approximate linear dependency test
δ_{ij}	Kronecker delta
$\Delta\mathbf{w}(i)$	weight adjustment at time or iteration i (a column vector in an Euclidean space)
$\mathcal{D}$	data set
$e(i)$	output estimation error at time or iteration i
$\mathbb{F}$	feature space induced by the kernel mapping
$\mathbf{G}$	Gram matrix of (transformed) input data
$\mathbb{H}$	reproducing kernel Hilbert space

$\mathbf{I}$	identity matrix
$J(i)$	error cost at time or iteration i
$\mathbf{k}(i)$	Kalman gain (or gain vector) at time or iteration i
$K(\mathbf{A})$	condition number of a matrix $\mathbf{A}$
$\kappa(\mathbf{u}, \mathbf{u}')$	kernel (or covariance) function evaluated at $\mathbf{u}$ and $\mathbf{u}'$
L	dimensionality of the input space
λ	regularization parameter
M	dimensionality of the feature space
$\mathcal{M}$	misadjustment of the least-mean-square algorithm
$\mathbf{n}(i)$	additive noise in the state space at time or iteration i
N	number of training data
η	step-size parameter
$O(\cdot)$	of the order of a number
$\mathbf{P}$	state-error correlation matrix
$\boldsymbol{\varphi}(\cdot)$	a mapping induced by a reproducing kernel
$\boldsymbol{\varphi}(i)$	transformed filter input at time or iteration i (a column vector in a feature space)
$\boldsymbol{\Phi}$	transformed input data matrix
$\mathbf{R}$	covariance matrix of (transformed) input data
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^L$	L-dimensional real Euclidean space
$\varsigma_{\max}$	the maximum eigenvalue
$\text{tr}(\mathbf{A})$	trace of matrix $\mathbf{A}$
T_1	abnormality threshold in surprise criterion
T_2	redundancy threshold in surprise criterion
$\mathbf{u}(i)$	filter input at time or iteration i (a column vector in an Euclidean space)
$\mathbb{U}$	input domain
$\mathbf{U}$	input data matrix
$v(i)$	additive noise in the output at time or iteration i
$\mathbf{w}(i)$	weight estimate at time or iteration i (a column vector in an Euclidean space)
$\boldsymbol{\omega}(i)$	weight estimate at time or iteration i (a column vector in a feature space)
z^{-1}	unit delay operator

AIC	Akaike information criterion
ALD	approximate linear dependency
APA	affine projection algorithm
BIC	Bayesian information criterion
CC	coherence criterion
CV	cross-validation
ENC	enhanced novelty criterion
EX-RLS	extended recursive least squares algorithm
EX-KRLS	extended kernel recursive least squares algorithm

GPR	Gaussian process regression
LMS	least-mean-square algorithm
LOOCV	leave-one-out cross-validation
LS	least squares
MAP	maximum a posterior
MDL	minimum description length
MSE	mean square error
MML	maximum marginal likelihood
NC	novelty criterion
NLMS	normalized least-mean-square algorithm
KA	kernel ADALINE
KAPA	kernel affine projection algorithm
KLMS	kernel least-mean-square algorithm
KRLS	kernel recursive least-squares algorithm
PCA	principal components analysis
PDF	probability density function
RAN	resource allocating network
RBF	radial-basis function
RKHS	reproducing kernel Hilbert space
RLS	recursive least-squares algorithm
RN	regularization network
RNN	recurrent neural network
SC	surprise criterion
SNR	signal-to-noise ratio
SVD	singular value decomposition
SVM	support vector machine
SW-KRLS	sliding window kernel recursive least-squares algorithm