A Study of Online Learning with Multiple Kernels and Variable Metric

February 2017

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering

Keio University
Graduate School of Science and Technology
School of Integrated Design Engineering

TODA, Osamu
Abstract

In this thesis entitled “A Study of Online Learning with Multiple Kernels and Variable Metric”, online learning schemes are considered. One of the recent topics in signal processing is how to deal with big data. Batch processing of big data involves high computational complexity. When data arrive sequentially, the batch learning needs to recompute the solution at each time. The batch learning is thus inappropriate to process such sequential data. Online learning, of which the complexity is linear order, adaptively processes sequential data and reflects recently observed information to learning. The existing online learning schemes need a priori information to design a proper metric and/or kernel. This thesis presents effective online learning schemes based on a variable metric and multiple kernels. The proposed schemes estimate a proper metric and/or kernel, and, at the same time, learn an unknown system to be estimated.

This thesis consists of six chapters.

Chapter 1 introduces the background and the outline of this thesis.

Chapter 2 introduces mathematical basics that will be used throughout this thesis, including proximity operator, variable-metric projection, and reproducing kernel.

Chapter 3 introduces a novel adaptive filtering scheme named the metric-combining normalized least mean square (MC-NLMS) algorithm. The proposed scheme is based on iterative metric projections with a metric designed by combining multiple metric matrices convexly in an adaptive manner, thereby taking advantages of the metrics that rely on multiple pieces of information. This chapter compares the natural proportionate NLMS (NPNLMS) algorithm, which is a special case of MC-NLMS, with the improved PNLS (IPNLMS) algorithm and it is shown that the performance of NPNLMS is controllable with the convex combination coefficient as opposed to IPNLMS. This chapter also presents another example of MC-NLMS with its application to an acoustic echo cancellation problem and shows the efficacy of the proposed scheme.

Chapter 4 investigates the use of compactly supported kernels (CSKs) for the kernel NLMS (KNLMS) algorithm. The use of CSKs yields sparse kernelized input vectors, offering an opportunity for complexity reduction. A simple two-step method to compute the kernelized input vectors efficiently is presented. In the first step, it computes an over-estimation of the support of the kernelized input vector based on a certain l1-ball. In the second step, it identifies the exact support by detailed inspections based on an l2-ball. Also, the proposed method employs the identified support given by the second step for dictionary construction, and then reduces the amount of l2-distance evaluations, leading to the complexity reduction. The nu-
merical examples show that the proposed algorithm achieves significant complexity reduction.

Chapter 5 investigates the online model selection and learning scheme, which is based on the multikernel adaptive filtering framework, for nonlinear-function estimation tasks. Designing a proper reproducing kernel is one of the major issues that arise frequently when one applies kernel methods particularly to online nonlinear estimation problems. The multikernel adaptive filtering framework gives a convex analytic recipe with multiple kernels employed simultaneously, offering a practical remedy to the challenging design issue. Here, a typical choice of multiple kernels is a set of Gaussian kernels with different scales (variance parameters); another choice is a set of linear/polynomial and Gaussian kernels. The present work takes one step further. This thesis presents a fully-adaptive all-in-one learning scheme that jointly makes the model-selection and learning (i.e., nonlinear estimation under the selected model).

Chapter 6 summarizes the results of this thesis and gives an outlook on future research.
Acknowledgments

This dissertation completes my studies in this fascinating research group of Yukawa Laboratory in Keio University since 2013. Having in mind that words cannot fully describe feeling, I would like to express my deepest gratitude to my adviser Assoc. Prof. Masahiro Yukawa for his guidance, support, and friendship through all these years. I believe that all discussion and thinking about signal processing with Prof. Yukawa become an infrastructure of informational tools in heavy electrical industry.

I would also like to express my gratitude for the referees who examined this dissertation, Prof. M. Ikehara, Prof. Y. Sanada and Prof. S. Adachi. Their invaluable reviews and advises made clear relation between this dissertation and other research areas.

Furthermore, I would like to thank Prof. I. Yamada from Tokyo Institute of Technology, for giving me helpful-advises about powerful strategies of a fix-point theory in signal processing, and Dr. A. Sugiyama, Media and Information Research Laboratories, NEC Corporation, for his coaching that is about how to utilize signal processing in engineering.

A part of this dissertation was conducted under a contract of research and development for radio resource enhancement, organized by the Ministry of Internal Affairs and Communications, Japan.

I would also like to express my appreciation Prof. H. Kikuchi, Prof. S. Sasaki, Assoc. Prof. S. Muramatsu in Niigata University, and all students of Yukawa Laboratory, Yamada Laboratory and KLab for providing with a fun-to-work environment.

Finally, I would also like to express my appreciation my bosses and a senior Mr. M. Kawamura, H. Ikuji, T. Toyama and K. Nakao in Meidensha Corporation. By using the dissertation and my experience as a foundation for product development, I will construct next-generation of heavy electrical techniques with their group from now on.
Contents

Abstract 1
Acknowledgments 3
List of Acronyms 11

1 General Introduction 13
   1.1 Adaptive Filter and System Model 13
   1.2 Adaptive Filtering Algorithms 15
   1.3 This Study 22

2 Preliminaries 25
   2.1 Proximity Operator 25
   2.2 Metric Projection and Convex Projection 27
   2.3 Variable-Metric Projection Framework 27
   2.4 Reproducing Kernel 29

3 MC-NLMS 31
   3.1 Introduction 31
   3.2 Metric-Combining Adaptive Filter 32
      3.2.1 The Proposed Adaptive Algorithm 33
      3.2.2 Examples of Metric Design 33
      3.2.3 Advantage of NPNLMS over IPNLMS 34
   3.3 Application to Echo Cancellation 40
      3.3.1 A Straightforward Approach 40
      3.3.2 The Proposed Approach Using MC-NLMS 42
      3.3.3 Numerical Examples 43
   3.4 Conclusion 45

4 CSK-NLMS 47
   4.1 Introduction 47
   4.2 Compactly Supported Kernels 48
   4.3 Proposed Algorithm 50
      4.3.1 Efficient Support Identification through Over-estimation 50
      4.3.2 Complexity of the Proposed Algorithms with CSK 51
   4.4 Numerical Examples 52
   4.5 Conclusion 54
5 Online Model Selection and Learning Method
  5.1 Introduction ...................................... 57
  5.2 Adaptive Generalized Forward-backward Splitting Method .......... 59
  5.3 System and MKAF .................................. 59
  5.4 Proposed Scheme .................................. 61
    5.4.1 Dictionary Refinement ............................ 61
    5.4.2 Cost Function with Double Regularizers .................. 63
    5.4.3 Coefficient Updating Rule ........................... 64
    5.4.4 Remarks .......................................... 65
  5.5 Numerical Examples .................................. 69
    5.5.1 Experiment A ...................................... 70
    5.5.2 Experiment B ...................................... 75
    5.5.3 Experiment C: Real Data ............................ 77
  5.6 Conclusion .......................................... 82

6 General Conclusion ..................................... 89

Publications Related to the Dissertation .......................... 91

Bibliography .................................................. 93
# List of Figures

1.1 An illustration of an adaptive filter. ............................................. 14
1.2 An illustration of a geometric interpretation of the projection and NLMS in noiseless case. .................................................. 15
1.3 An overview of adaptive algorithms in the 20th century. ................. 18
1.4 An overview of adaptive algorithms based on convex optimization methods. ................................................................. 19
1.5 An overview of splitting methods in convex optimization. ............... 20
1.6 A history of nonlinear adaptive filtering approaches. ....................... 21
1.7 The overall structure of this dissertation. ..................................... 24
2.1 A geometric interpretation of the orthogonal projections onto the hyperplane $\Pi_n$ in terms of three different metrics. ..................... 28
2.2 An illustration of the case that an adequate metric changes over time. 29
3.1 Impulse responses with different sparsity levels and transform-domain expression. ......................................................... 35
3.2 MSE learning curves for the high-sparse impulse response $h_{1}^*$. $\alpha = 0$ corresponds to PNLMS, and $\alpha = 1$ corresponds to NLMS. .................. 36
3.3 MSE learning curves for the semi-sparse impulse response $h_{2}^*$. $\alpha = 0$ corresponds to PNLMS, and $\alpha = 1$ corresponds to NLMS. .................. 37
3.4 The diagonal elements of the metric matrix of IPNLMS (or NPNLMS) as a function of $\alpha$ for $h_{1}^*$. ............................................ 38
3.5 The diagonal elements of the metric matrix of IPNLMS (or NPNLMS) as a function of $\alpha$ for $h_{2}^*$. ............................................ 39
3.6 The WTP serial approach. ......................................................... 41
3.7 The MC-NLMS parallel approach. .............................................. 41
3.8 The curve of the combination coefficient $\alpha_n$. .......................... 43
3.9 NMSE learning curves for an AEC problem in the case of the white input. ................................................................. 43
3.10 NMSE learning curves for an AEC problem in the case of a colored input (the eigenvalue spread: $2.5 \times 10^{13}$). ......................... 44
4.1 Compactly supported kernels and a Gaussian kernel. ...................... 49
4.2 An illustration of the proposed algorithm. .................................... 52
4.3 Dictionary size $\tilde{r}_n$, the over-estimated support size $\tilde{s}_n$ and the exact support size $\hat{s}_n$. ........................................... 55
4.4 NMSE learning curves. ............................................................. 55
LIST OF FIGURES

4.5 The complexity of the algorithms at each iteration. .......................... 56

5.1 Block diagram of the online model-selection and learning scheme. . . 58
5.2 The arithmetic and geometric means of the $\ell_2$ norms of $h_{\gamma(n)}$ for $i = 1, 2, \cdots, r_n = 50$. The linear and logarithmic scales are used for the vertical axis for clarity. .................................................. 62
5.3 Results for Experiment A. .................................................. 71
5.4 The estimandum $\psi_1$ (red) and its estimates (blue or green). ......... 73
5.5 The $\ell_2$-norms of the column vectors of $H_n$. ......................... 73
5.6 A comparison with SimpleMKL. ........................................... 74
5.7 Results for Experiment B. .................................................. 76
5.8 Results for Experiment C. .................................................. 78
5.9 The contours of $\psi_3$ and its estimates. The dictionary data $u_j$, $j \in J_n$, are marked by diamonds. ................................................. 79
5.10 The estimandum $\psi_3$ and its estimates in Experiment C. ............ 80
5.11 The estimation errors $|\psi_3(u) - \varphi_n(u)|$. ............................. 81
List of Tables

4.1 Gaussian and compactly supported functions for $L = 2$, $\rho := \|x - y\|_2$, $\alpha := 1/2\sigma^2$, and $\tilde{\rho} := \rho^2/c^2$; $(x)_+ := \max\{x, 0\}$ for $x \in \mathbb{R}$.

4.2 Computational complexity of the algorithms.

5.1 Proposed scheme.

5.2 Parameter settings for Experiment A.

5.3 Parameter settings for Experiment B.

5.4 Parameter settings for Experiment C.

5.5 A list of optimization methods for nonsmooth convex functions over a Hilbert space $\mathcal{H}$ and their adaptive extensions. Smooth and nonsmooth functions are denoted by $f$ and $g$, respectively. The $\iota_C$ is the indicator function for a convex subset $C \subset \mathcal{H}$. $A$ is a bounded linear operator.
List of Acronyms

LMS  Least Mean Square algorithm
NLMS  Normalized LMS
MC-NLMS  Metric-Combining Normalized LMS
PNLMS  Proportionate Normalized LMS
IPNLMS  Improved Proportionate NLMS
MPNLMS  $\mu$-law Proportionate NLMS
NPNLMS  Natural PNLMS
TDAF  Transform-domain Adaptive Filter
SNR  Signal Noise Ration
APSM  Adaptive Projected Subgradient Method
MSE  Mean Square Error
ARMA  Auto Regressive Moving Average
KNLMS  Kernel Normalized LMS
CSK  Compactly Supported Kernel
GFBS  Generalized Forward-Backward Splitting Method
APFBS  Adaptive Proximal Forward-backward Splitting
AGFBS  Adaptive GFBS
RKHS  Reproducing Kernel Hilbert Space
Chapter 1

General Introduction

This dissertation presents effective online learning schemes, which are based on the (time-)variable metric and multiple kernels. We introduce some techniques of adaptive filtering: e.g., a variable-metric projection framework, compactly supported kernels, a multikernel adaptive filtering algorithm and an adaptive generalized forward-backward splitting method. Each technique is developed in different fields of study. The present schemes are attained only by the developed techniques. This chapter provides the overall relationship among the study fields as well as the relations among the chapters in this dissertation.

1.1 Adaptive Filter and System Model

Let $\mathbb{R}$ and $\mathbb{N}$ denote the sets of all real numbers and nonnegative integers, respectively. We denote by $\mathcal{U} \in \mathbb{R}^L$ the input space. $\mathcal{H}$ denotes a real Hilbert space of functions. We consider, in this dissertation, the following system model:

$$d_n := \psi_n(u_n) + v_n, \quad n \in \mathbb{N}, \quad (1.1)$$

where $\psi_n \in \mathcal{H} : \mathcal{U} \to \mathbb{R}$ is the unknown (generally nonlinear) system at time $n \in \mathbb{N}$, $u_n \in \mathcal{U}$ is an input vector of length $L$, and $v_n \subset \mathbb{R}$ is the noise process. The adaptive filter models the system into the filter parameters of finite impulse response (FIR) filter. To adapt to changing system characteristics, an adaptive filtering algorithm keeps updating the coefficients over time. A number of adaptive algorithms have been developed since the middle of 20th century [1, 2] (see Section 1.2). Figure 1.1 illustrates an adaptive filter. As usual in online scenarios, it is assumed that the input vector $u_n \in \mathcal{U}$ and the desired response $d_n \in \mathbb{R}$ arrive sequentially. The task of adaptive filtering is to estimate the estimandum (a function to be estimated) $\psi_n$ by the adaptive filter $\varphi_n \in \mathcal{H}$ with the sequentially arriving input-output dataset $(u_n, d_n)_{n \in \mathbb{N}}$ in an online fashion. $\varphi_n$ is updated in accordance with an adaptive filtering algorithm. The signal to noise ratio (SNR) is defined as

$$\text{SNR} := 10 \log_{10} \frac{\mathbb{E}\{\psi_n^2(u_n)\}}{\mathbb{E}\{v_n^2\}} \text{[dB]}, \quad (1.2)$$
CHAPTER 1. GENERAL INTRODUCTION

\[ \epsilon_n := \varphi_n(u_n) - d_n \in \mathbb{R}. \]  

(1.3)

The goal is then to build an effective adaptive filtering algorithm that diminishes the error \( \epsilon_n \) in a smallest possible number of iterations. To compare adaptive algorithms, a normalized mean squared error (NMSE) at time \( n \) is defined as

\[ \text{NMSE}_n := 10 \log_{10} \frac{E\{\epsilon_n^2\}}{E\{\psi_n^2(u_n)\}} \text{[dB]}. \]  

(1.4)

To simplify the introduction, we consider the case of \( \psi_n(u_n) := \langle h^*, u_n \rangle \) with \( h^* \in \mathcal{H} := \mathbb{R}^L \) and \( \langle a, b \rangle := a^Tb \) for \( a, b \in \mathbb{R}^L \). Here, \( (\cdot)^T \) stands for transpose. The normalized least mean square (NLMS) algorithm has been proposed [3,4]:

\[ h_{n+1} := h_n - \eta_n \frac{\epsilon_n}{\|u_n\|_2^2} u_n, n \in \mathbb{N}, \]  

(1.5)

where \( h_n \) is the coefficient vector, \( \eta_n \in [0,2] \) is step size, and \( \| \cdot \|_2 \) is \( \ell_2 \) norm. The NLMS algorithm has the following geometric property. Define the hyperplane

\[ \Pi_n := \{ h_n \in \mathbb{R}^L \mid \langle h_n, u_n \rangle = d_n \}, n \in \mathbb{N}. \]  

(1.6)

The orthogonal projection of \( y \in \mathbb{R}^L \) onto \( \Pi_n \) is then given as

\[ P_{\Pi_n}(y) = y - \frac{\epsilon_n}{\|u_n\|_2^2} u_n. \]  

(1.7)

The NLMS algorithm in (1.5) is rewritten as

\[ h_{n+1} := h_n + \eta_n (P_{\Pi_n}(h_n) - h_n), n \in \mathbb{N}. \]  

(1.8)

Figure 1.2 illustrates a geometric interpretation of the orthogonal projection and the NLMS algorithm in noiseless case.
1.2. ADAPTIVE FILTERING ALGORITHMS

\[
\Pi_n
\]

\[h_n\]

\[h_{n+1}(\eta_n = 0.5)\]

\[h_{n+1}(\eta_n = 1)\]

\[h_{n+1}(\eta_n = 1.5)\]

\[h_{n+1}(\eta_n = 2)\]

(a) An illustration of the points of \(h_n\) updated by NLMS in noiseless case.

\[
\Pi_n+1
\]

\[h_n\]

\[h_{n+1}(\eta_n = 1)\]

\[h_{n+2}\]

(b) An illustration of NLMS for \(\eta_n = 1\).

Figure 1.2: An illustration of a geometric interpretation of the projection and NLMS in noiseless case.


1.2 Adaptive Filtering Algorithms

An overview of the adaptive algorithms is given in Fig. 1.3. A.N. Kolmogorov and N. Wiener have independently established theory of linear optimal filter, which is widely referred to as “Wiener filter”, based on the stationarity in 1930 [39–41]. Wiener and Hopf have derived the Wiener-Hopf equation by minimizing the mean squared error (MSE) function (see [1]). In the discrete case, the equation is reduced to the system of linear equations \(R\mathbf{h} = \mathbf{p}\), where \(\mathbf{h} \in \mathbb{R}^L\) is the filter of \(L\) taps, \(R \in \mathbb{R}^{L \times L}\) is the autocorrelation matrix of input, and \(\mathbf{p} \in \mathbb{R}^L\) is the cross-correlation vector between the input and output. A solution of the Wiener-Hopf equation, which called a Wiener solution, is difficult to compute in a real-time system. The recursive least squares (RLS) algorithm [42–44] is a method to solve approximate Wiener-Hopf equations by a recursive formula based on the matrix inversion lemma. The least mean square (LMS) algorithm is an instantaneous approximation of the gradient algorithm. LMS and NLMS are the most basic adaptive algorithms. NLMS is an algorithm based on the orthogonal projection onto a zero instantaneous-error hyperplane. The adaptive filtering algorithms have been improved based on the LMS or NLMS. The affine projection algorithm (APA) [46, 47] improves the convergence properties. Transform-domain adaptive filter (TDAF) [48, 49] updates the coefficient in the transform domain (e.g., DFT or DCT) and has a constant convergence speed. The proportionate NLMS [17, 57] improves the convergence speed. The sliding-window type adaptive filtering algorithm [50, 51] has a convergence speed like
RLS and a computational complexity like LMS. The filterd-x LMS (Fx-LMS) algorithm [55, 56] is used on single-channel feedforward adaptive noise control (ANC) systems. Fx-LMS identifies the *estimandum* (e.g., a desired ANC controller) in the presence of a secondary-path. The adaptive algorithms developed in the 20th century are based on the gradient descent method solely for a squared error loss function.

The general framework of adaptive filtering algorithms, which is based on the convex optimization methods, gives widespread influence on the signal processing. Figure 1.4 summarizes the adaptive filtering algorithms which are adaptive extensions of the convex optimization methods to minimize nonsmooth functions. In 2004, the adaptive projected subgradient method (APSM) has been presented by Yamada [58]. APSM [58–60] is motivated by the fixed point theory of nonexpansive mapping. The subgradient method has been used as a computationally efficient replacement of the metric projection (see Section 2.2) onto a given closed convex set. The applications of APSM are expanding in the field of signal processing, including acoustic echo canceller [61], MIMO wireless communication system [62], online machine learning [63] and adaptive learning in diffusion networks [64]. Inspired by the successful adaptive extension of Polyak’s projected subgradient method, other convex optimization methods have also been extended, e.g., the adaptive proximal forward-backward splitting algorithm (APFBS) [65], the adaptive Douglas-Rachford splitting method (ADRS) [66, 67], the adaptive alternating direction method [68] and the adaptive primal-dual splitting algorithm (APDS) [69]. The key of the adaptive extensions is nonsmooth (but proximable) convex functions. Note here, a function $f$ is said to be proximable if the proximity operator $\text{prox}_f$ is computable. The proximity operator is useful for instance, for sparse optimization problem. APFBS, ADRS and APDS can minimize the sum of a smooth-function and a (or a few) nonsmooth and proximable function (or functions). An appropriate adaptive algorithm needs to be chosen according to the number of nonsmooth (proximable) functions. Figure 1.5 summarizes several types of optimization problems and appropriate methods to solve each of those problems.

Figure 1.6 shows the history of nonlinear adaptive filters. The framework of the nonlinear adaptive filtering algorithms is divided into

- adaptive polynomial filters (adaptive Volterra filter),
- neural networks with radial-basis function (nonconvex optimization),
- single kernel adaptive filtering, and
- multikernel adaptive filtering (i.e., an adaptive filter with multiple kernels).

Furthermore, the convex optimization methods in Fig. 1.6 have been utilized to develop kernel adaptive filtering algorithms in order to accelerate the convergence speed and/or improve estimation accuracy. In the kernel adaptive filter, the unknown system is modeled as a function in reproducing kernel Hilbert space (RKHS). Kernel methods have become an appealing tool in many practical fields, including biomedical engineering [70], remote sensing [71, 72], control [73, 74] and communication systems [75]. The kernel adaptive filtering is a new framework for nonlinear estimation tasks.
The framework for the nonlinear system estimation consists of mapping the input $\mathbf{u}_n$ into RKHS, and then several existing linear adaptive filtering techniques [1, 2] were reconsidered to derive nonlinear generalizations operating in RKHS. The kernel recursive least-square (KRLS) algorithm, which is a kernel-based counterpart of RLS, was introduced in [76]. The sliding-window scheme of KRLS was derived in [77], and the KRLS tracker algorithm, which has a process to forget past information, was presented in [78]. On the other hand, the kernel LMS algorithm and the kernel APA method were introduced in [79–81]. Although the kernel adaptive filter represents the nonlinear system by a function in RKHS, the KNLMS algorithm estimates the system by updating coefficients in a parameter space. It is therefore called “linear-in-parameter method”. The hyperplane projection along affine subspace (HYPASS) algorithm [82–85] improves the convergence speed by exploiting the geometry of RKHS.
CHAPTER 1. GENERAL INTRODUCTION

Adaptive filtering algorithm

A.N. Kolomogorov (1939, 1941) [39, 40]
N. Wiener (1949) [41]

C.F. Gauss (1795) [42]
A.M. Legendre (1805) [43]
R.L. Plackett (1950) [44]

B. Widrow, M.E. Hoff (1960) [45]

J. Nagumo (1967) [3]

Improvements of LMS or NLMS

affine projection

APA

T. Hinamoto, S. Mackawa (1975) [46]
K. Ozeki, T. Umeda (1984) [47]

transform domain type

TDADF

S.S. Narayan et al. (1983) [48]

sliding-window type

SW-LMS

B.F. Farden (1981) [50]
W.A. Gardner (1984) [51]

nonlinear type (polynomial)

Adaptive Polynomial Filter
(Adaptive Volterra Filter)

T. Koh, E. Powers (1985) [33]
J. Lee, V.J. Mathews (1993) [52, 53]

variable step size type

Variable-step AF

R. Harris et al. (1986) [54]
R.H. Kwong, E.W. Johnston (1992) [35]

single-path type

Fx-LMS

C. Elliott, I. Stothers et al. (1987) [55]
E. Bjarnason (1995) [56]

proportionate type

PNLMS

S.L. Gay (1998) [17]
D.L. Duttweiler (2000) [57]

Figure 1.3: An overview of adaptive algorithms in the 20th century.
1.2. ADAPTIVE FILTERING ALGORITHMS

Figure 1.4: An overview of adaptive algorithms based on convex optimization methods.

Convex optimization

- Projected Gradient Method
  A.A. Goldstein (1964) [86]

- Proximal Forward-Backward Splitting
  B. Mercier (1979) [87]
  G.B. Passty (1979) [88]

- Douglas-Rachford Splitting
  P.L. Lions, B. Mercier (1979) [90]

- Alternating Direction Method
  R. Glowinski, A. Marrocco (1975) [91]
  D. Gabay, B. Mercier (1976) [92]

- Generalized Forward-backward Splitting
  H. Raguet, J. Fadili, G. Peyré (2013) [93]

- Primal-Dual Splitting
  A. Chambolle, T. Pock (2011) [94]
  L. Condat (2012) [96]

Adaptive filtering algorithm

- Adaptive Projected Subgradient Method (APSM)

- FOBOS
  J. Duchi, Y. Singer (2009) [89]

- Adaptive Proximal Forward-Backward Splitting (APFBS)

- Adaptive Douglas-Rachford Splitting (ADRS)
  I. Yamada, S. Gandy, M. Yamagishi (2011) [66]
  M. Yamagishi, I. Yamada (2013) [67]

- Adaptive Alternating Direction Method
  S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein (2011) [68]

- Adaptive Generalized Forward-backward Splitting
  O. Toda, M. Yukawa (this dissertation)

- Adaptive Primal-Dual Splitting
  P.L. Combettes, J.C. Pesquet (2016) [97]
  S. Oto, M. Yamagishi, I. Yamada (2016) [69]
Figure 1.5: An overview of splitting methods in convex optimization.

\[ \min_{h \in \mathcal{H}} (f(h) + g_1(h) + g_2(h)) \]

**Generalized Forward-backward Splitting**

_Baguet, Fedki, Pegré (2013) [93]_

\[ \min_{h \in \mathcal{H}} (f(h) + g(h)) \]

**Proximal Forward-backward Splitting**

_Mercier (1979), [87], Passty (1979) [88]_

\[ \min_{h \in \mathcal{H}} g_1(h) + g_2(h) \]

**Douglas-Rachford Splitting**

___Lions, Mercier (1979) [90]___

\[ \min_{h \in \mathcal{H}} g_1(h) + g_2(Ah) \]

**Alternating Direction Method**

_Glowinski, Marrocco (1975) [91]_

_Gabay, Mercier (1976) [92]_

\[ \min_{h \in \mathcal{H}} f(h) + g_1(h) + g_2(Ah) \]

**Primal-Dual Splitting**

_Condat (2012) [96]_

\[ f: \text{ a differentiable convex function} \]

\[ g, g_1, g_2: \text{nonsmooth convex functions} \]

\[ A: \text{given matrix} \]

\[ I: \text{identity matrix} \]

\[ h \in \mathcal{H}: \text{vector} \]
1.2. ADAPTIVE FILTERING ALGORITHMS

Nonlinear AF

- Adaptive Polynomial Filter (Volterra Filter)

- Neural Networks with Radial-basis Function
  K.J. Lang, G.E. Hinton (1988) [100],
  P.M. Grant et al. (1991) [101], J.C. Príncipe et al. (1993) [102]

Linear AF

- RLS [42–44]
- LMS [45]
- NLMS [3, 4]
- APFBS [65]
- APSM [58]

Kernel AF

- Kernel RLS
  - Y. Engel et al. (2004) [76]
- SW-KRLS
  - S.V. Vaerenbergh (2006) [77]
- Kernel LMS
- Kernel-DCT
  - C. Richard et al. (2008) [80]
- KAPA
  - C. Richard et al. (2008) [80]
- KAPA-DCT
  - K. Nishikawa et al. (2013) [81]
- KNLS
  - C. Richard et al. (2009) [80]
- HYPASS
  - M. Yukawa, R. Ishii (2012) [82]
- KNLMS
  - M. Yukawa (2012) [107]

Convex combination of KAFs

- Convex combination of KAFs
  - W. Gao, C. Richard et al. (2014) [106]

- Convex optimization methods

- CHYPASS
  - M. Yukawa (2015) [83]

- Kernel APSM
  - K. Slavakis et al. (2009) [75]
  - Φ-PASS
  - M. Tubizawa, M. Yukawa (2015) [110]

- Kernel AGFBS
  - O. Toda, M. Yukawa (This dissertation)

Figure 1.6: A history of nonlinear adaptive filtering approaches.
1.3 This Study

This dissertation addresses the following questions:

1. Given multiple metrics designed to improve the convergence behaviors, how can we leverage all of them and maximize the benefits form them?

2. How can we reduce the evaluation costs of kernel functions?

3. How can we jointly optimize a kernel and an estimator using the “optimized” kernel?

Figure 1.7 shows the overall structure of this dissertation. The main body is divided into three parts, each of which is devoted to each of the above questions.

Chapter 2 provides notation employed throughout this dissertation.

Chapter 3 presents a novel linear adaptive filtering scheme named the metric-combining normalized least mean square algorithm. A metric, which is given by a positive definite matrix, measures the distance between two points in the Euclidean space of filters. The metric projection is defined as the “closest point”, in the sense of minimizing the metric distance, in a closed convex set from the current adaptive filter. The performance of an adaptive filtering algorithm depends on the metric design. The presented scheme is based on iterative metric projections with a metric designed by combining multiple metric-matrices convexly in an adaptive manner, thereby taking advantages of the metrics which rely on multiple pieces of information. We compare the improved PNLMS (IPNLMS) algorithm with the natural proportionate NLMS (NPNLMS) algorithm, which is a special case of MC-NLMS, and it is shown that the performance of NPNLMS is controllable with the combination coefficient, whereas IPNLMS shows sensitivity to the choice of the combination coefficient. We also present an application to an acoustic echo cancellation problem and show the efficacy of the proposed scheme.

Chapter 4 presents a kernel adaptive filtering scheme with compactly supported kernels (CSKs). The use of CSKs yields sparse kernelized input vectors, offering an opportunity for complexity reduction. We propose a simple two-step method to compute the kernelized input vectors efficiently. In the first step, it computes an over-estimation of the support of the kernelized input vector based on a certain \( \ell_1 \)-ball. In the second step, it identifies the exact support by detailed examinations based on an \( \ell_2 \)-ball. Also, we employ the identified support given by the second step for coherence construction. The proposed method reduces the amount of \( \ell_2 \)-distance evaluations, leading to the complexity reduction. The numerical examples show that the proposed algorithm achieves significant complexity reduction.

Chapter 5 presents a method to jointly optimize the kernel and estimator. Designing a proper reproducing kernel is one of the major issues that arise frequently when one applies kernel methods particularly to online nonlinear estimation problems. See [75, 80, 111, 112] for online kernel learning or kernel adaptive filtering approaches. The multikernel adaptive filtering framework [83, 107, 113] gives a convex analytic recipe with multiple kernels employed simultaneously, offering a practical remedy to the challenging design issue. Here, a typical choice of multiple kernels
is a set of Gaussian kernels with different scales; another choice is a set of linear/polynomial and Gaussian kernels [83]. We take one step further. Our focus is on the selection of proper models. In addition to model-selection, one needs to find an *estimandum* under the selected model for accomplishing a nonlinear estimation task. The aim is develop a fully-adaptive all-in-one learning scheme that jointly makes the model-selection and learning (nonlinear estimation under the selected model).
Figure 1.7: The overall structure of this dissertation.
Chapter 2

Preliminaries

This chapter introduces basic notation and mathematical tools that are used throughout this dissertation. Let \( \mathbb{N}^* \) denote the set of all positive integers. This dissertation considers the schemes for solving asymptotic minimization problems of a sequence \((\Theta_n)_{n \in \mathbb{N}}\) of convex cost functions given in the following form:

\[
\Theta_n(h) := \phi_n(h) + \sum_{s=1}^{S} \Omega_n^{(s)}(h), \quad h \in \mathcal{H}, \quad S \in \mathbb{N}^*, \tag{2.1}
\]

where \( \phi_n : \mathcal{H} \to \mathbb{R} \) is convex and smooth, and \( \Omega_n^{(s)} : \mathcal{H} \to (-\infty, \infty] = \mathbb{R} \cup \{\infty\} \) for all \( S \in \mathbb{N}^* \) are convex and proximable (but are possibly nondifferentiable). The adaptive extension of GFBS, which uses the gradient of \( \phi_n \) and the proximity operators of \( \Omega_n^{(s)} \)'s, is applied to minimize the cost function in (2.1). A real Hilbert space equipped with an inner product is denoted by \((\mathcal{H}, \langle \cdot, \cdot \rangle_\mathcal{H})\). Also \(|\cdot|_1\) and \(|\cdot|_\mathcal{H}\) denote the \( \ell_1 \) norm and the norm in \( \mathcal{H} \), respectively.

2.1 Proximity Operator

A point \( x \in C \) is called an interior point of a subset \( C \subset \mathcal{H} \) if there exists \( \epsilon > 0 \) satisfying

\[
\{ y \in \mathcal{H} \mid |y - x|_\mathcal{H} < \epsilon \} \subset C. \tag{2.2}
\]

The set of all interior points is called an interior of \( C \) and is denoted by \( \text{int } C \). A set \( C \) is open if \( \text{int } C = C \) (i.e. every point in \( C \) is an interior point). A set \( C \) is closed if its complement \( \mathcal{H}\setminus C = \{ x \in \mathcal{H} \mid x \notin C \} \) is open. A set \( C \) is called a convex set if \((1 - \eta)x + \eta y \in C, \forall x, y \in C, \forall \eta \in [0,1] \). If a set \( C \) is closed as well as convex, it is called a closed convex set.

Definition 1 (convex function, proper function [58]). A function \( f : \mathcal{H} \to (-\infty, \infty] := \mathbb{R} \cup \{\infty\} \) is called convex if

\[
f((1-\eta)x + \eta y) \leq (1-\eta)f(x) + \eta f(y), \forall x, y \in \mathcal{H}, \forall \eta \in (0,1). \tag{2.3}
\]
In particular, a convex function $f : \mathcal{H} \to (-\infty, \infty]$ is called proper if

$$\text{dom}(f) := \{x \in \mathcal{H} \mid f(x) < \infty\} \neq \emptyset.$$  

(2.4)

A function $f : \mathcal{H} \to \mathbb{R}$ is continuous at $x \in \mathcal{H}$ if, for all $\epsilon > 0$, there exists $\delta > 0$ such that

$$\|x - y\|_\mathcal{H} \leq \delta, y \in \mathcal{H} \Rightarrow \|f(x) - f(y)\|_\mathcal{H} \leq \epsilon.$$  

(2.5)

Continuity can be described in terms of limits: whenever a sequence $(x_1, x_2, \cdots)$ in $\mathcal{H}$ converges to a point $x^* \in \mathcal{H}$, the sequence $(f(x_1), f(x_2), \cdots)$ converges to $f(x^*)$, i.e., $\lim_{n \to \infty} f(x_i) = f(\lim_{n \to \infty} x_i)$.

**Definition 2** (lower semicontinuous function [58]). A function $f : \mathcal{H} \to (-\infty, \infty]$ is called lower semicontinuous if a set $\text{lev}_{\leq \delta}(f) := \{x \in \mathcal{H} \mid f(x) \leq \delta\}$ is closed for every $\delta \in \mathbb{R}$.

Note here that $f$ is lower semicontinuous if $f$ is continuous over $\mathcal{H}$. The set of all proper lower semicontinuous convex functions is denoted by $\Gamma_0(\mathcal{H})$.

**Definition 3** (strictly convex [58]). A function $f \in \Gamma_0(\mathcal{H})$ is called strictly convex if

$$(x \neq y, \eta \in (0,1)) \Rightarrow f((1-\eta)x + \eta y) < (1-\eta)f(x) + \eta f(y).$$  

(2.6)

**Definition 4** (coercivity [58]). A function $f \in \Gamma_0(\mathcal{H})$ is called coercive if

$$\|x\|_\mathcal{H} \to \infty \Rightarrow f(x) \to \infty.$$  

(2.7)

The existence of minimizer of $f$ is guaranteed by its coercivity.

A point $x \in \mathcal{H}$ is called a fixed point of a mapping $T : \mathcal{H} \to \mathcal{H}$ if $T(x) = x$. The set of all fixed points of $T$ is denoted by $\text{Fix}(T) := \{x \in \mathcal{H} \mid T(x) = x\}$. A mapping $T : \mathcal{H} \to \mathcal{H}$ is called $\beta$-Lipschitz continuous with a Lipschitz constant $\beta > 0$ over $\mathcal{H}$ if

$$\|T(x) - T(y)\|_\mathcal{H} \leq \beta \|x - y\|_\mathcal{H}, \forall x, y \in \mathcal{H}.$$  

(2.8)

In particular, $T$ is called contractive if $\beta < 1$ and is called nonexpansive if $\beta = 1$.

**Definition 5** (proximity operator [114,115]). Given any point $x \in \mathcal{H}$, the proximity operator of index $\gamma \in (0, \infty)$ of $f \in \Gamma_0$ is defined by

$$\text{prox}_{\gamma f} : \mathcal{H} \to \mathcal{H} : x \mapsto \arg\min_{y \in \mathcal{H}} \left\{f(y) + \frac{1}{2\gamma} \|x - y\|_\mathcal{H}^2\right\},$$  

(2.9)

where the existence and the uniqueness of the minimizer are guaranteed respectively by the coercivity and the strict convexity of $f(\cdot) + \frac{1}{2\gamma} \|\cdot\|_\mathcal{H}^2$. Equivalently, for every $x \in \mathcal{H}$, $\text{prox}_{\gamma f}(x)$ is characterized as a unique point satisfying

$$\{\text{prox}_{\gamma f}(x)\} = \{z \in \mathcal{H} \mid z + \gamma \partial f(z) \ni x\},$$  

(2.10)

i.e.,

$$\text{prox}_{\gamma f}(x) = (I + \gamma \partial f)^{-1}(x),$$  

(2.11)
which is again equivalent to
\[
\left\langle y - \text{prox}_{\gamma f}(x), \frac{x - \text{prox}_{\gamma f}(x)}{\gamma} \right\rangle \leq f(\text{prox}_{\gamma f}(x)) - f(y), \forall y \in \mathcal{H}.
\] (2.12)

The proximity operator is firmly nonexpansive, i.e.,
\[
\rho_{\text{prox}_{\gamma f}} := 2\text{prox}_{\gamma f} - I : \mathcal{H} \to \mathcal{H},
\] is nonexpansive:
\[
\|2\text{prox}_{\gamma f} - I\| \leq \|x - y\|, \quad \forall x, y \in \mathcal{H}.
\] (2.13)

If \(\arg\min_{x \in \mathcal{H}} f(x) \neq \emptyset\), the set of all minimizers of \(f\) is equal to the fixed point set of \(\text{prox}_{\gamma f} : \mathcal{H} \to \mathcal{H}\); i.e.,
\[
\arg\min_{x \in \mathcal{H}} f(x) = \text{Fix}(\text{prox}_{\gamma f}).
\] (2.14)

### 2.2 Metric Projection and Convex Projection

**Definition 6 (metric projection).** Given a nonempty closed convex set \(C \subset \mathcal{H}\) and any point \(x \in \mathcal{H}\), there exists a unique point \(P_C(x) \in C\) satisfying
\[
d_C := \min_{z \in C} \|x - z\| = \|x - P_C(x)\|.
\] (2.15)

The mapping \(\mathcal{H} \ni x \mapsto P_C(x) \in C\) is called the metric projection, or the convex projection, onto \(C\) and it holds that \(P_C = \text{prox}_{\gamma C}, \forall \gamma \in (0, \infty)\) [114, 115].

### 2.3 Variable-Metric Projection Framework

We show that the projection depends on the metric design. Given any positive definite matrix \((\mathbb{R}^{L \times L})^+ A \succ 0\), we define an inner product and its induced norm, respectively, as
\[
\langle x, y \rangle_A := x^T Ay, \forall x, y \in \mathbb{R}^L, \text{ and } \|x\|_A := \sqrt{\langle x, x \rangle_A}, \forall x \in \mathbb{R}^L.
\]
Hereafter, we regard \(A\) as a metric which determines the metric-distance between \(x\) and \(y\) by \(\|x - y\|_A\).

**Definition 7.**

(a) **A-orthogonal:** Vectors \(x\) and \(y\) are said to be \(A\)-orthogonal if \(\langle x, y \rangle_A = 0\).

(b) **A-projection:** Given a closed convex set \(C \subset \mathbb{R}^L\), the metric projection of \(x\) onto \(C\) in terms of the metric \(A\) is defined as
\[
P_C^A(x) := \arg\min_{y \in C} \|y - x\|_A,
\] (2.16)

which is referred to as the \(A\)-projection of \(x\) onto \(C\).

The hyperplane, which is the set of filters \(h\) such that \(\varepsilon_n = 0\), is rewritten as follows:
\[
\Pi_n := \{ h \in \mathbb{R}^L \mid h^T u_n = \langle h, A^{-1} u_n \rangle_A = d_n \}, n \in \mathbb{N},
\] (2.17)
CHAPTER 2. PRELIMINARIES

where the normal vector of $\Pi_n$ is $A^{-1}u_n$ in $(\mathbb{R}^L, \langle \cdot, \cdot \rangle_A)$. The $A$-projection of $z$ onto the hyperplane $\Pi_n$ can be computed as

$$P^A_{\Pi_n}(z) = z - \frac{\hat{\varepsilon}_n(z)}{u_n^TA^{-1}u_n}A^{-1}u_n,$$

where $\hat{\varepsilon}_n(z) := z^Tu_n - d_n$. Figure 2.1 illustrates the projections onto $\Pi_n$ in the case of $d_n = 0$ based on three different metrics: the Euclidean metric (i.e., the identity matrix $I$) and two other metrics $A \succ 0$ and $B \succ 0$. It is clear that the input vector $u_n$ is $I$-orthogonal to any $h \in \Pi_n$ since $\langle h, u_n \rangle_I = h^Tu_n = d_n = 0$. Moreover, the vector $A^{-1}u_n$ is $A$-orthogonal to any $h \in \Pi_n$ since $\langle h, A^{-1}u_n \rangle_A = h^TA^TA^{-1}u_n = 0$. The same applies to the metric $B$. For any $h \not\in \Pi_n$, the projections $P^I_{\Pi_n}(z)$ and $P^A_{\Pi_n}(z)$ are different from each other (unless $A$ is proportional to $I$), implying that the orthogonal projection depends on the metric design. In fact, the metric design affects the performance of adaptive algorithms [61,116–118], as will be shown in Section 3.2.

Let us explain now that the use of variable metric is beneficial for improving the performance of an adaptive algorithm. Indeed,

(a) an adequate metric may change in time due to the nonstationarity of data or may depend on the phase of adaptation process;

(b) an adequate metric is unavailable in practice especially during the initial phase due to the lack of information even if the adequate metric is time invariant.

Figure 2.2 illustrates case (a). In the figure, the $A$-projection is more suitable than the $B$-projection at time $n$, since $P^A_{\Pi_n}(h_n)$ is closer to $h^*$ than $P^B_{\Pi_n}(h_n)$. In contrast, at time $j$, the $B$-projection is more suitable than the $A$-projection. It is therefore better to control the metric adaptively in this case. A concrete example will be given in Section 3.3 for acoustic echo cancellation applications. In the latter case, on the other hand, one needs to approximate the (possibly time-invariant) suitable metric by using data acquired up to each time instant. In both cases, the metric to be used in the algorithm changes in time and the variable-metric projection framework [61,117] encompasses those cases. An adequate design of the metric under the new concept of metric combining presented in Chapter 3 yields significant improvements in performance.
2.4. Reproducing Kernel

Definition 8 (Reproducing kernel). Let $\mathcal{H}$ be a real Hilbert space of functions $f : U \to \mathbb{R}$. A function $\kappa : U \times U \to \mathbb{R}$ is called a reproducing kernel for $\mathcal{H}$ if

1. $\kappa (\cdot, u) \in \mathcal{H}, \forall u \in U$,
2. $f (u) = \langle f, \kappa (\cdot, u) \rangle_{\mathcal{H}}, \forall f \in \mathcal{H}, \forall u \in U$. (reproducing property)

The name reproducing kernel comes from property 2 in Definition 8. The reproducing kernel of a Hilbert space is unique [119]. Suppose that $\kappa_1, \kappa_2$ are reproducing kernels for $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$, and then the property 2 of Definition 8 gives

$$\langle f, \kappa_1 (\cdot, u) - \kappa_2 (\cdot, u) \rangle_{\mathcal{H}} = 0, \forall f \in \mathcal{H}, \forall u \in U. \quad (2.19)$$

Setting $f = \kappa_1 (\cdot, u) - \kappa_2 (\cdot, u)$ for a fixed $u$ shows the uniqueness. A Hilbert space which has a reproducing kernel is called “a reproducing kernel Hilbert space” (RKHS). RKHS has the following property.
Theorem 1. ([120]) Suppose $H$ is a Hilbert space of functions $f : U \to \mathbb{R}$ with reproducing kernel $\kappa$. Then,

1. $\kappa(u_1, u_2) = \langle \kappa(\cdot, u_1), \kappa(\cdot, u_2) \rangle_H$ for $u_1, u_2 \in U$.

2. $\kappa(u_1, u_2) = \kappa(u_2, u_1)$ for $u_1, u_2 \in U$.

3. Convergence in the Hilbert space norm implies pointwise convergence, i.e.,

$$\lim_{n \to \infty} \| f - f_n \|_H = 0 \text{ implies } \lim_{n \to \infty} | f(u) - f_n(u) | = 0, \forall u \in U.$$

For $x, y \in U = \mathbb{R}^L$, we present typical reproducing kernels below.

Euclid inner product (linear kernel):

$$\kappa(x, y) = x^T y.$$

Polynomial kernel:

$$\kappa(x, y) = (x^T y + c)^p, \quad p \in \mathbb{N}, c \geq 0.$$

Laplacian kernel:

$$\kappa(x, y) = \exp \left( -\beta \sum_{l=1}^{L} |x_l - y_l| \right), \beta > 0,$$

where $x_l = [x]_l$ and $y_l = [y]_l$.

Gaussian kernel:

$$\kappa(x, y) = \exp \left( -\frac{\| x - y \|_2^2}{2\sigma^2} \right), \sigma > 0.$$

Matérn’s Compactly supported kernel:

$$\kappa(x, y) = \max \left\{ \left(1 - \frac{\| x - y \|_2^2}{\sigma^2} \right)^L, 0 \right\}.$$
Chapter 3

Metric Combining-NLMS Based on Time-Variable Metric Projection

3.1 Introduction

The goal of this chapter is to provide a simple but effective adaptive filtering algorithm based on a simultaneous use of multiple pieces of information. Indeed, a significant amount of researches has been done to improve the convergence behavior of the classical NLMS algorithm by using available information about input signals and unknown systems to be estimated [17, 38, 48, 49, 57, 65, 121–126]. For instance, Narayan et al. have proposed TDAF which improves the convergence stability for colored inputs, and this approach can be regarded as using information of the second-order statistics of input signals [48, 49]. Meanwhile, Duttweiler has proposed the PNLMS algorithm which accelerates the convergence speed for sparse impulse responses, and this approach exploits the a priori information that unknown systems tend to be sparse [57]. Here, sparse means that most of the coefficients are (close to) zero. An efficient adaptive filtering algorithm with high convergence speed is demanded for advanced communication systems [1]. Acoustic echo cancellation (AEC), for instance, is still an important application of adaptive filtering due to the rapid spread of Internet conference. In this application, the unknown systems are typically sparse and the speech input signals have high autocorrelation. This is a case in which multiple pieces of information are available for adaptation. As yet however, no techniques have been developed to exploit multiple pieces of information simultaneously so that better performance can be attained than using only one of them.

In this chapter, we propose an efficient adaptive filtering scheme based on iterative use of metric projection with the metric changing in time according to incoming data. The variable-metric projection technique has been proposed by Yukawa and Yamada in 2009 [117]. From the variable-metric projection viewpoint, the TDAF and PNLMS algorithms can be regarded as ones using time-varying metrics designed with available information about the input statistics and the system sparsity, respectively. Here, the metric measures the distance between two points in the
Euclidean space of adaptive filters and is given by a positive definite matrix. The projection has been used extensively as a backbone in adaptive filtering algorithms [108, 127], and is defined as the ‘closest’ point, in the sense of minimizing the metric distance, in a closed convex set from the current adaptive filter. The proposed scheme combines multiple metric-matrices convexly in an adaptive manner for taking advantages of the metrics, dubbed the Metric-Combining Normalized Least Mean Square (MC-NLMS) algorithm. In terms of metric, it turns out that the improved PNLMS (IPNLMS) algorithm (Benesty and Gay, 2002) [121] combines two metrics as a convex combination of the inverses of the metric-matrices. To see that metric is an intrinsic factor that governs the performance of iterative projection methods, this chapter shows the performance of the proposed scheme and the IPNLMS algorithm using the same metric-matrices. Numerical examples show that the performance of the proposed scheme changes gradually as the convex-combination coefficients slide from zero to one, whereas that of IPNLMS changes drastically by a slight change of the coefficients. From a practical point of view, the proposed scheme is user-friendly compared to IPNLMS since an assignment of the coefficient leads to expectable performance, or in other words, the performance of the scheme is quite controllable. This supports the significance of considering the metric used for the computation of the projection.

The above example exploits two metrics: the Euclidean metric (which is used, for example, in NLMS) and the metric based on sparsity information. As an example to exploit multiple pieces of information, we apply the proposed scheme to the AEC problem. As already mentioned, it is possible to use the information about the system sparsity and the coloredness of input signals. More specifically, the sparsity is exploited to design a metric-matrix by following the way of the $\mu$-law proportionate NLMS (MPNLMS) algorithm (Deng and Doroslovački, 2005). Roughly speaking, the aim of using these two metrics is taking the advantage of MPNLMS (which is the fast initial convergence) and the advantage of TDAF (which is the uniform rate of convergence through the adaptation). Numerical examples show that the proposed scheme achieves better overall performances, than using one of the metric-matrices, due to a dynamic control of the combination coefficients.

### 3.2 Metric-Combining Adaptive Filtering Algorithm

In this section, the proposed algorithm is first presented which is referred to as the MC-NLMS algorithm. Several specific metric designs are then presented. Finally, a particular case of MC-NLMS (which is closely related to IPNLMS) is studied and its advantage over IPNLMS is shown.
3.2. METRIC-COMBINING ADAPTIVE FILTER

3.2.1 The Proposed Adaptive Algorithm

A time-variable metric is defined with \( M \) metrics \( A_{n,m} \geq 0, \ n \in \mathbb{N}, \ m = 1, 2, \cdots, M \), as follows:\(^1\)

\[
A_n := \sum_{m=1}^{M} \alpha_{n,m} A_{n,m},
\]

where \( \alpha_{n,m} \in [0,1] \) is the convex combination coefficients satisfying \( \sum_{m=1}^{M} \alpha_{n,m} = 1, \ \forall n \in \mathbb{N} \). We let the maximal eigenvalue of \( A_{n,m} \) to be normalized, or equivalently \( \| A_{n,m} \|_2 = 1 \). Although there are other ways for normalizing the metrics \( A_{n,m} \), it will become clear in Section 3.2.3 that the normalization by \( \| A_{n,m} \|_2 \) is reasonable. The proposed adaptive algorithm is given as follows (see (2.18)):

\[
h_{n+1} := h_n + \eta_n \left[ P_{A_n} \left( \hat{\epsilon}_n \right) - h_n \right] = h_n - \eta_n \frac{\hat{\epsilon}_n (h_n)}{u_n^T A_n^{-1} u_n} A_n^{-1} u_n, \ n \in \mathbb{N},
\]

where \( \eta_n \in [0,2] \) is the step size at time \( n \). We call (3.2) together with (3.1) the MC-NLMS algorithm. The performance of MC-NLMS is governed by the design of the metrics \( A_{n,m} \) and the combination coefficients \( \alpha_{n,m} \). Each \( A_{n,m} \) is designed based on each piece of available information (such as the sparsity information), and the proposed algorithm exploits the whole available information by designing the metric \( A_n \) in the form of (3.1). Furthermore, \( \alpha_{n,m} \) is controlled dynamically to increase the gain coming from the combined metric \( A_n \) at each iteration.

3.2.2 Examples of Metric Design

We present some special cases of the MC-NLMS algorithm with different metric designs.

1) Case of \( M = 1 \):

a) Letting \( A_n = I \) yields the NLMS algorithm. In this case, the metric and the norm become Euclidean.

b) Letting

\[
A_n := \left( \sum_{i=1}^{L} \gamma_n^{(i)} \right) \text{diag}^{-1} \left( \gamma_n^{(1)}, \gamma_n^{(2)}, \cdots, \gamma_n^{(L)} \right)
\]

yields the PNLMS algorithm [57], where diag\(^{-1}\) is the inverse of diagonal

\(^1\)Given \( M \) different metrics \( A_m \ (m = 1, 2, \cdots, M) \), we can consider the \( M \) different Hilbert spaces \( (\mathbb{R}^L, \langle \cdot, \cdot \rangle_{A_m}), \ m = 1, 2, \cdots, M \). What if we do the learning in the Cartesian product of those spaces? This fundamental question has been investigated in [128]. The metric-combining idea presented in (3.1) comes from the algorithm that is obtained as a particular example of APSM [59] in the product space.
matrix and

\[
\gamma_n^{(i)} := \max \left\{ \gamma_n^{\text{max}}, |h_n^{(i)}| \right\} > 0, \tag{3.4}
\]

\[
\gamma_n^{\text{max}} := \rho \max \left\{ \sigma, |h_n^{(1)}|, |h_n^{(2)}|, \ldots, |h_n^{(L)}| \right\} > 0 \tag{3.5}
\]

Here, \( \rho > 0 \) and \( \sigma > 0 \) are positive constants. Note that the inverse \( A_n^{-1} \)

of the metric-matrix, rather than \( A_n \), is used in (3.2).

c) Other examples of MC-NLMS for \( M = 1 \) include KPNLMS [129], TDAF [48, 49] and QNAF [130, 131].

2) Case of \( M = 2 \):

a) Letting

\[
A_n := \left[ \alpha A_{n,1}^{-1} + (1 - \alpha) A_{n,2}^{-1} \right]^{-1}, \tag{3.6}
\]

\[
A_{n,1}^{-1} := \frac{1}{L} I, \tag{3.7}
\]

\[
A_{n,2}^{-1} := \frac{1}{\|h_n\|_1 + \delta \text{diag}(|h_n|)}, \tag{3.8}
\]

yields the IPNLMS algorithm [121], where \( \alpha \in [0, 1] \), \( \delta \geq 0 \) is the regularization parameter, and \( |\cdot| \) is the elementwise absolute-value operation. Note here that this is not a special case of MC-NLMS because \( A_n \) is not given in the form of (3.1). (Here, \( A_{n,1} \) is employed by NLMS and \( A_{n,2} \)

for \( \delta = 0 \) is identical to \( A_n \) in (3.3) with \( \gamma_n^{\text{max}} := 0 \)).

b) The Natural-PNLMS (NPNLMS) algorithm: Let

\[
A_n := \alpha A_{n,1} + (1 - \alpha) A_{n,2}, \tag{3.9}
\]

\[
A_{n,1} := I, \tag{3.10}
\]

\[
A_{n,2} := \frac{1}{\|G_n\|_2} \text{diag}(G_n), \tag{3.11}
\]

where \( G_n := \text{diag}\left( g_n^{(1)}, g_n^{(2)}, \ldots, g_n^{(L)} \right) \), \( g_n^{(i)} := \left( \max \left\{ \tau, |h_n^{(i)}| \right\} \right)^{-1} \), \( \tau > 0 \)

is the regularization parameter, \( \alpha_{n,1} := \alpha \in [0, 1] \), and \( \alpha_{n,2} := 1 - \alpha \). We refer to this special case of MC-NLMS as NPNLMS.

NPNLMS exploits basically the same information as IPNLMS but in a natural way. The following subsection will show the significant advantages of NPNLMS over IPNLMS.

### 3.2.3 Advantage of NPNLMS over IPNLMS

Computer simulations are conducted to show how much impact the change of the convex combination coefficient \( \alpha \) gives on the performance of IPNLMS and NPNLMS. We use white input signals, additive white Gaussian noise with SNR
3.2. METRIC-COMBINING ADAPTIVE FILTER

Figure 3.1: Impulse responses with different sparsity levels and transform-domain expression.

30dB, and impulse responses \(h_1^*\) and \(h_2^*\) which have different sparsity levels for \(L = 256\) (see Figs. 3.1(a) and 3.1(b)). For IPNLMS, \(\eta_n = 0.4, \forall n \in \mathbb{N}\), and \(\delta = 10^{-4}\). For NPNLMS, \(\eta_n = 0.4, \forall n \in \mathbb{N}\), and \(\tau = 10^{-4}\).

Figures 3.2 and 3.3 show the results for \(h_1^*\) and \(h_2^*\), respectively. For both IPNLMS and NPNLMS, \(\alpha = 0\) and \(\alpha = 1\) correspond respectively to PNLMS and NLMS. It can be seen that the performance of the proposed scheme (NPNLMS) is well controlled by the combination coefficient. In contrast, the performance of IPNLMS is not well controlled by the combination coefficient, and \(\alpha = 0.2\), 0.4, and 0.6 yield nearly the same performance on average. As a result, the best performance of IPNLMS is inferior to that of NPNLMS for both \(h_1^*\) and \(h_2^*\). This implies that the performance of the proposed scheme is quite controllable with the combination coefficients.

To analyze the phenomena observed in Figs. 3.2 and 3.3, we plot in Figs. 3.4 and 3.5 the diagonal elements of \(A_n\) for \(h_1^*\) and \(h_2^*\), respectively, as a function of \(\alpha\). By definition, the metric projection is unchanged under a multiplication of the metric by a positive scalar (see (2.16)). Therefore, for each \(\alpha\), the metric matrix \(A_n\) for IPNLMS is divided by its 2-norm (i.e., its spectrum radius) so that its maximum eigenvalue becomes unity. It can be seen that the ratios between the maximum eigenvalue and the other eigenvalues drop linearly for NPNLMS. For IPNLMS, on the other hand, the ratio between the maximum eigenvalue and the second largest eigenvalue behaves quite differently from the ratio between the maximum eigenvalue and the smallest eigenvalue. We emphasize here again that the projection depends on metric (see Section 2.3). The above observations clarify the reason why the performance of NPNLMS is controllable by \(\alpha\) and why that of IPNLMS is not.
Figure 3.2: MSE learning curves for the high-sparse impulse response $h_1^*$. $\alpha = 0$ corresponds to PNLMS, and $\alpha = 1$ corresponds to NLMS.
3.2. **METRIC-COMBINING ADAPTIVE FILTER**

![Graph of metric-combining adaptive filter](image)

(a) I-PNLMS

(b) N-PNLMS

Figure 3.3: MSE learning curves for the semi-sparse impulse response $h^*_2$. $\alpha = 0$ corresponds to PNLMS, and $\alpha = 1$ corresponds to NLMS.
Figure 3.4: The diagonal elements of the metric matrix of IPNLMS (or NPNLMS) as a function of $\alpha$ for $h_n^*$. 
3.2. METRIC-COMBINING ADAPTIVE FILTER

Figure 3.5: The diagonal elements of the metric matrix of IPNLMS (or NPNLMS) as a function of $\alpha$ for $h_2^*$. 
3.3 Application to Echo Cancellation

The PNLMS algorithm exploits the sparsity information for accelerating the convergence speed, while the TDAF algorithm exploits the second-order statistics of input signals to decorrelate the signals and to improve the convergence behavior. It is known that the coloredness of input signals deteriorates the performance of PNLMS. This motivates us to use the information about both of the sparsity and the input statistics simultaneously for the AEC problem, as speech input signals are known to be highly colored. What would be an effective approach to exploiting the sparsity and statistics simultaneously? A straightforward approach is to whiten the signal first and then apply the proportionate adaptation to the whitened input signal. This Whitening and Then Proportionate (WTP) approach is discussed in Section 3.3.1 and will turn out to be disadvantageous. We then show in Section 3.3.2 how to apply the MC-NLMS algorithm to the AEC problem. Finally, we present the simulation results in Section 3.3.3.

3.3.1 A Straightforward Approach

The WTP approach is illustrated in Fig. 3.6. It can be regarded as a serial approach composed of two stages: the colored signal is whitened in the first stage, and then the whitened signal is processed by the PNLMS algorithm in the second stage. The WTP approach is given by:

$$\hat{w}_{n+1} := \hat{w}_n - \eta_n \frac{\hat{e}_n(\hat{w}_n)}{v_n^T A_n^{-1} v_n} A_n^{-1} \hat{v}_n,$$

(3.12)

Here, $\hat{w}_n$ is the adaptive filter in the transform domain, $\hat{e}_n(\hat{w}) := \hat{w}^T \hat{v}_n - d_n$, $\hat{w} \in \mathbb{R}^L$, $\hat{v}_n$ is the whitened input vector defined as $\hat{v}_n := \Delta_n^{-1/2} Q v_n \in \mathbb{R}^L$, where $Q \in \mathbb{R}^{L \times L}$ is an appropriate orthogonal matrix such as the discrete cosine transform (DCT) matrix, $\Delta_n \in \mathbb{R}^{L \times L}$ is a diagonal matrix that normalizes the power of each element of the transform-domain input vector, and $A_n$ is a proportionate-type metric. Unfortunately, this approach is ineffective for the following reason. To obtain the same response as the pair of unknown system and input $(h^*, u_n)$, the companion of $\hat{v}_n$ is $\hat{w}^* = \Delta_n^{1/2} Q h^*$. There is no guarantee that the sparseness of $h^*$ is preserved under the transformation; i.e., $\hat{w}^*$ is no longer sparse in general as illustrated in Fig. 3.1(c).

---

2The filter-input pairs $(\hat{w}, \hat{v}_n)$, $(w, v_n)$, and $(h, u_n)$ should have the same response, namely $\hat{w}^T \hat{v}_n = w^T v_n = h^T u_n$, $\forall u_n \Leftrightarrow \hat{w}^T \Delta_n^{-1/2} Q u_n = w^T Q u_n = h^T u_n$, $\forall u_n \Leftrightarrow Q^T \Delta_n^{-1/2} \hat{w} = Q^T w = h$. Thus, the adaptive filter for the modified inputs $\hat{v}_n$ is $\hat{w}_n := \Delta_n^{1/2} Q h^*$. 
3.3. APPLICATION TO ECHO CANCELLATION

Figure 3.6: The WTP serial approach.

Figure 3.7: The MC-NLMS parallel approach.
3.3.2 The Proposed Approach Using MC-NLMS

To exploit the sparsity and statistical information simultaneously by MC-NLMS, we let

$$A_n := \alpha_n A_{n,1} + \left(1 - \alpha_n\right) A_{n,2}, \quad (3.13)$$

$$A_{n,1} = \frac{1}{\left\|G_{n}^{MP}\right\|_2} G_{n}^{MP}, \quad (3.14)$$

$$A_{n,2} = Q^T \left( \frac{1}{\left\|\Delta_n\right\|_2} \Delta_n \right) Q. \quad (3.15)$$

Here, $G_{n}^{MP}$ is the metric of MPNLMS [132] which improves the convergence speed of PNLS by designing the metric optimally for white inputs, and $A_{n,2}$ is the metric of TDAF in the time domain since $\langle h_n, u_n \rangle_{A_{n,2}} = h_n^T A_{n,2} u_n = w^T \Delta_n^{-1} v_n = \langle w, v_n \rangle_{\Delta_n}$. The MC-NLMS algorithm can take the advantages of both algorithms by controlling $\alpha_n \in [0, 1]$ dynamically. If $\alpha_n = 1$ for all $n \in \mathbb{N}$, MC-NLMS is reduced to MPNLMS. On the other hand, if $\alpha_n = 0$ for all $n \in \mathbb{N}$, it is reduced to the transform-domain NLMS (TD-NLMS) algorithm which is a normalized version of TDAF in the time-domain.

Our controlling strategy of $\alpha_n$ is the following.

1) Initial phase: Assign a large weight to $A_{n,1}$-metric to attain fast initial convergence. (Take the advantage of MPNLMS.)

2) From middle phase to steady state: Assign a large weight to $A_{n,2}$-metric to gain the benefit of the constant rate of convergence. (Take the advantage of TD-NLMS.)

3) According to the analysis in [117], the fluctuations of the metric $A_n$ should be sufficiently small for the algorithm to converge, implying that the parameters $\alpha_n$ should be changed gradually.

Under the above ideas, we recursively define $\alpha_n$ for an initial value $\alpha_0 := 1$ as follows:³

$$\alpha_{n+1} = \min \{1, \tau_n\}, \quad (3.16)$$

$$\tau_{n+1} = (1 - \beta) \tau_n + \beta \frac{\|h_{n+1} - h_n\|_2}{\|h_{n+1}\|_2}, \quad (3.17)$$

where $\beta \in (0, 1)$ and $\tau_n$ is initialized as $\tau_0 = 1$. A typical choice is $\beta = 10^{(5 - \log_2 L)}$ for $128 \leq L \leq 512$.

³We conducted experiments by letting $\alpha_n = 1$ in the initial phase and $\alpha_n = 0$ after that. The point here is when to switch the weight $\alpha_n$ from 1 to 0. When we set the switching timing appropriately, the result was as good as the performance of the proposed algorithm presented in Section 3.3.3. However, when the timing was inappropriate, the performance was degraded seriously. In contrast to the sensitivity to the choice of the timing behind this switching strategy, our proposed technique in (3.16) and (3.17) shifts the weight $\alpha_n$ gradually, and thus it is fairly insensitive to the choice of $\beta$. 

3.3. APPLICATION TO ECHO CANCELLATION

3.3.1 Application to Echo Cancellation

If the unknown system $h^*$ changes into $\tilde{h}^* \in \mathbb{R}^L$ at a time instant $\hat{n}$ abruptly, one may detect the change, reinitialize the $\tau_n$ parameter as $\tau_{\hat{n}} = 1$, and then track the new unknown system $\tilde{h}^*$ by adapting $h_n$ with a large weight reassigned to $A_{n,1}$. Note here that the metric $A_{n,1}$ in (3.14) should reflect the sparsity of the difference vector $\tilde{h}^* - h_{\hat{n}}$ rather than that of $\tilde{h}^*$ [133]. The $A_{n,1}$ is designed based on $h_n - h_{\hat{n}}$ in practice.

3.3.3 Numerical Examples

We show the efficacy of the proposed scheme for the AEC application. The proposed scheme is compared with NLMS, MPNLMS, TD-NLMS, and WTP. The simulation settings are given as follows: the sparse unknown system is defined as $h^* := [h^T_2, 0, \cdots, 0]^T \in \mathbb{R}^L$ for $L = 512$. We use two types of input signals for $(u_n)_{n \in \mathbb{N}}$: one is the zero-mean white Gaussian signal with the unit variance and the
other is the USASI signal (which is speech-like wide sense stationary). Here, the USASI signal is modeled on the auto-regressive moving average (ARMA) process [134] and is characterized by the rational function:

\[
H(z) := \frac{1 - z^2}{1 - 1.70223z^{-1} + 0.71902z^{-2}}, \quad z \in \mathbb{C},
\]

(3.18)

where \( \mathbb{C} \) denotes the set of all complex numbers. The noise \( v_n \) is additive white Gaussian with SNR = 30dB. For all adaptive algorithms, we set \( \eta_n = 0.4 \). For MPNLMS [132], \( \rho = 0.01 \), \( \sigma = 0.01 \), and \( \mu = 1000 \). For TD-NLMS, the forgetting factor is set to \( \nu = 10^{-3} \). For MC-NLMS, \( \beta = 5.0 \times 10^{-4} \) which gives \( \alpha_n \) depicted in Fig. 3.8.

Figures 3.9 and 3.10 show the NMSE learning curves. It can be seen that the proposed scheme outperforms the other algorithms for both white and highly-colored input signals. In particular, the good overall performance in the highly-colored input case comes from taking the advantages of MPNLMS and TD-NLMS by controlling the combination coefficient \( \alpha_n \) adequately. The MPNLMS algorithm attains good performance in white input case, while its convergence slows down gradually since its metric is no longer optimal for colored inputs. The TD-NLMS algorithm keeps its convergence speed to be fairly constant. As explained in Section 3.3.1, the WTP serial approach loses the sparsity of \( h^* \) by its whitening process, resulting in poor performance.
3.4 Conclusion

This chapter has provided an attractive approach to exploiting multiple pieces of information available in potential applications of adaptive filters. We presented an efficient adaptive filtering scheme, MC-NLMS, which incorporates multiple pieces of information into the metric and improves the convergence behavior. The proposed scheme is based on the variable-metric projection concept. We compared the IPNLMS algorithm with the NPNLMS algorithm, which is a special case of MC-NLMS, and it was shown that the performance of NPNLMS was controllable with the combination coefficient as opposed to IPNLMS. We also have presented an application to an echo cancellation problem and showed the efficacy of the proposed scheme. We applied the MC-NLMS algorithm to the AEC application. The proposed scheme outperformed the other algorithms for both white and highly-colored input signals.
Chapter 4

Compactly Supported Kernel Adaptive Filter

4.1 Introduction

Gaussian kernel is one of the radial basis functions which depend on the distance between two samples and does not depend on the direction. This particular property is one of the reasons for the popularity of Gaussian kernel for approximating unknown functions. In the literature of kernel adaptive filtering (kernel online learning), the Gaussian kernel has widely been employed [75, 76, 80, 106, 111, 112, 135]. From the computational aspects, the Gaussian kernel is not necessarily perfect because many practical applications such as kernel ridge regression require the inversion of the kernel matrix which is dense in the case of Gaussian kernels. Compactly supported and positive definite functions have been studied in statistics [120, 136–139]. These functions have been applied to scattered data approximation and surface reconstruction as well as machine learning problems [140–144]. A remarkable advantage of the compactly supported and positive definite functions is that the kernel matrix becomes sparse, which reduces the computational burden of kernel methods [144].

The KNLMS algorithm [80] is one of the simplest examples of kernel adaptive filtering algorithms. The algorithm is free from the computation of the kernel matrix inversion since it projects the current estimate onto the zero-instantaneous hyperplane in the Euclidean space. However, the length of the kernelized input vector, which gives the normal vector of the hyperplane, is the size of the dictionary, which implies that the complexity for computing the kernelized input vector and the projection increases linearly with the dictionary size.

This chapter focuses on the sparseness of the kernelized input vector for compactly supported kernels (CSKs) and presents an efficient KNLMS algorithm with CSKs. The support of the kernelized input vector corresponds to those data points in the dictionary which are contained by the $\ell_2$-ball centered at the current data; the radius is determined by the support of the CSK employed. To avoid computing the $\ell_2$-distances from the current data to all the dictionary data, we propose a simple two-step method. In the first step, it computes an over-estimate of the support based on the concentric circumscribed $\ell_1$-ball of the $\ell_2$-ball. In the second step, it identifies the exact support by computing the $\ell_2$-distances from the current data to
Table 4.1: Gaussian and compactly supported functions for $L = 2$, $\rho := \|x - y\|_2$, $\alpha := 1/2\sigma^2$, and $\tilde{\rho} := \rho^2/c^2$; $(x)_+ := \max\{x, 0\}$ for $x \in \mathbb{R}$.

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td>$\varphi_G(\rho) = \exp(-\alpha \rho^2)$</td>
</tr>
<tr>
<td>Matérn’s CSK [136]</td>
<td>$\varphi_{\text{Mrt}}(\rho) = (1 - \tilde{\rho})^2_+$</td>
</tr>
<tr>
<td>Wendland’s CSK [137]</td>
<td>$\varphi_{\text{Wnd}}(\rho) = (1 - \tilde{\rho})^6_+ (35\tilde{\rho}^2 + 18\tilde{\rho} + 3)/3$</td>
</tr>
<tr>
<td>Wu’s CSK [138]</td>
<td>$\varphi_{\text{Wu}}(\rho) = (1 - \tilde{\rho})_+^4 (3\tilde{\rho}^3 + 12\tilde{\rho}^2 + 16\tilde{\rho} + 4)/4$</td>
</tr>
</tbody>
</table>

only those dictionary data which are contained by the $\ell_1$-ball. Also, we employ the identified support given by the second step for coherence construction. This proposed method reduces the amount of $\ell_2$-distance evaluations drastically, leading to a substantial reduction of the computational loads. The numerical examples show that the proposed algorithm achieves significant complexity reduction.

4.2 Compactly Supported Kernels

Let $\{\kappa(\cdot, u_j)\}_{j \in \mathcal{J}}$ be the dictionary indicated by the index set $\mathcal{J}_n := \{j_1^{(n)}, j_2^{(n)}, \cdots, j_r^{(n)}\} \subset \{0, 1, \cdots, n - 1\}$, where $r_n \in \mathbb{N}^*$ is the dictionary size. A kernel adaptive filter is then given by

$$\varphi_n(u) = \sum_{j \in \mathcal{J}_n} h_j^{(n)} \kappa(u, u_j), u \in \mathcal{U}, \quad (4.1)$$

where $h_j^{(n)} \in \mathbb{R}$. An estimate of $d_n$ can be expressed in a vector form as follows:

$$\hat{d}_n := \varphi_n(u_n) = \hat{h}_n^T k_n, \quad (4.2)$$

where the $i$th components of $h_n \in \mathbb{R}^{r_n}$ and $k_n \in \mathbb{R}^{r_n}$ are given by $[h_n]_i := h_j^{(n)}$ and $[k_n]_i := \kappa(u_n, u_j^{(n)})$, respectively. Initialize the dictionary index set $J_{-1} := \emptyset$ and the filter $h_0 := \emptyset$. KNLMS [80] designs the dictionary index set $J_n$ based on the coherence criterion as follows:

$$J_{n+1} := \begin{cases} J_n \cup \{n\}, & \text{if } \min_{j \in J_n} |\kappa(u_n, u_j)| \leq \mu_0, \\ J_n, & \text{otherwise}, \end{cases} \quad (4.3)$$

where $\mu_0 \in (0, 1)$ is the coherence threshold. If $J_{n+1} \neq J_n$, then we let $h_j^{(n)} := 0$ and define augmented vectors $\hat{h}_n := [h_n^T, 0]^T \in \mathbb{R}^{r_n+1}$ and $\hat{k}_n := [k_n^T, 1]^T \in \mathbb{R}^{r_n+1}$. Otherwise, we let $\hat{h}_n := h_n$ and $\hat{k}_n := k_n$. The update equation is given as:

$$h_{n+1} := \hat{h}_n - \eta_n \frac{\hat{h}_n^T \hat{k}_n - \hat{d}_n}{\hat{k}_n^T \hat{k}_n} \hat{k}_n, n \in \mathbb{N}, \quad (4.4)$$
4.2. COMPACTLY SUPPORTED KERNELS

where \( \eta_n \in [0, 2] \) is the step size. The Gaussian kernel defined in Section 2.4 can be rewritten as

\[ \kappa_G(x, y) := \varphi_G(\rho) := \exp\left(-\frac{\rho^2}{2\sigma^2}\right), \tag{4.5} \]

where \( \rho := \|x - y\|_2, x, y \in \mathbb{R}^L \). This means that the Gaussian kernel depends only on the distance between two samples, but not on the direction of their difference vector. Such functions are called radial, and its special cases are compactly supported radial functions, which we shall denote by \( \varphi_C : \mathbb{R} \rightarrow \mathbb{R} \). A CSK can be defined as \( \kappa_C(x, y) := \varphi_C(\rho) \). Table 4.1 presents some known CSKs studied by Matérn [136], Wendland [137], and Wu [138] for the smoothness parameter set to 2 with the cutoff parameter \( c > 0 \), which controls the support region (see Fig. 4.1(a)). Fig. 4.1(b) shows the Gaussian kernel for \( \sigma = 0.2 \) and the CSKs for \( c = 2\sigma (= 0.4) \). The functions presented in Table 4.1 are strictly positive over the range \([0, c)\) and zero over \([c, \infty)\).

Figure 4.1: Compactly supported kernels and a Gaussian kernel.
4.3 Proposed Algorithm for Complexity Reduction

Due to the compactness of the support of $\varphi_C$, the kernelized input vector $\mathbf{k}_n$ becomes sparse with the use of an appropriately chosen $c$; in other words, $c$ controls the sparsity of $\mathbf{k}_n$. Without the sparse structure, the complexity for obtaining $\mathbf{k}_n$ and for updating $\mathbf{h}_n$ is linear in the dictionary size $r_n$. However, under the sparseness, once we know the support

$$\text{supp}(\mathbf{k}_n) := \{ \iota \in \mathcal{J}_n \mid [\mathbf{k}_n]_\iota := \kappa(u_n, u_\iota) \neq 0 \}$$

$$= \{ \iota \in \mathcal{J}_n \mid \| u_n - u_\iota \|_2 < c \}, \quad (4.6)$$

the complexity for the remaining computational tasks to obtain $\mathbf{k}_n$ and to update $\mathbf{h}_n$ is linear in the support size. This is the core of the complexity reduction.

4.3.1 Efficient Support Identification through Over-estimation

The support estimation is constructed by using an $\ell_1$-ball to reduce the complexity in terms of multiplications. The computational reduction is based on the fact that the evaluation of the $\ell_1$ norm involves no multiplications (only absolute value evaluations and additions). The idea is to exclude, at the first step, (not all but) most of the 'off-support' dictionary elements which locate out of the support of $\varphi_C(\| \cdot - u_n \|_2)$. Because of this, the evaluations of the Euclidean distance $\| u_n - u_\iota \|_2$ can be saved for the 'off-support' dictionary elements.

An open $\ell_1$-ball centered at $\mathbf{u} \in \mathbb{R}^L$ with radius $\gamma > 0$ is defined as follows:

$$\mathcal{B}_1(\mathbf{u}, \gamma) := \{ \mathbf{x} \in \mathbb{R}^L \mid \| \mathbf{x} - \mathbf{u} \|_1 < \gamma \}. \quad (4.7)$$

The proposed algorithm is composed of three stages.

Stage 1 : The support identification is accomplished by two steps.

Step 1 : Over-estimation based on $\ell_1$-ball

An over-estimate of $\text{supp}(\mathbf{k}_n)$ is defined by

$$\hat{\mathcal{I}}_n := \left\{ \iota \in \mathcal{J}_n \mid \mathbf{u}_\iota \in \mathcal{B}_1(\mathbf{u}_n, \sqrt{L}c) \right\} \supset \text{supp}(\mathbf{k}_n). \quad (4.8)$$

Here, the inclusion holds because $\sqrt{L} \| \mathbf{x} \|_1 \geq \| \mathbf{x} \|_2$, $\forall \mathbf{x} \in \mathbb{R}^L$.

Step 2 : Support identification based on $\ell_2$-ball

The open $\ell_2$-ball centered at $\mathbf{u}$ with radius $\gamma > 0$ is defined as

$$\mathcal{B}_2(\mathbf{u}, \gamma) := \{ \mathbf{x} \in \mathbb{R}^L \mid \| \mathbf{x} - \mathbf{u} \|_2 < \gamma \}. \quad (4.9)$$

Then, the support is identified under the inclusion in (4.8), as

$$\text{supp}(\mathbf{k}_n) = \left\{ \iota \in \hat{\mathcal{I}}_n \mid \mathbf{u}_\iota \in \mathcal{B}_2(\mathbf{u}_n, c) \right\}. \quad (4.10)$$
4.3. PROPOSED ALGORITHM

Stage 2: The dictionary is constructed on \( \text{supp} (k_n) \) as follows:

\[
J_{n+1} := \begin{cases} 
J_n \cup \{ n \}, & \text{if } u_i \notin B_2 (u_n, \mu_1 c) \ \forall i \in \text{supp} (k_n), \\
J_n, & \text{otherwise},
\end{cases}
\]  
(4.11)

where \( \mu_1 \in (0, 1) \) is the threshold. Note that the condition in (4.11) can be checked only for the subset \( \text{supp} (k_n) \) of \( J_n \).

Stage 3: The filter is updated by (4.4). Note here that the off-support components are unchanged and therefore the complexity for the update is determined by the support size, not by the dictionary size.

The parameter \( \mu_1 \) of the proposed algorithm is closely related to the \( \mu_0 \) of KNLMS in (4.3). In fact, setting

\[
\mu_1 = \frac{\sigma}{c} \sqrt{2 \ln \left( \frac{\mu_0^{-1}}{\mu_1^{-1}} \right)},
\]  
(4.12)

one can obtain the same dictionary as that of KNLMS. The proposed algorithm is insensitive to the choice of \( \mu_1 \) within a range in which the dictionary size is sufficiently large.

Example 1. To illustrate the proposed algorithm, we show a simple example in Fig. 4.2. In Fig. 4.2(a), the dictionary data are plotted by \( r_n = 200 \) dots. Note that those data are indicated by circles (rather than dots) in \( B_1 (u_n, \sqrt{2} c) \). In Fig. 4.2(b), the data in \( B_2 (u_n, c) \) are indicated by filled ones.

Stage 1:

Step 1. An over-estimate \( \hat{I}_n \) is attained by taking the 12 circles in \( B_1 (u_n, \sqrt{2} c) \); see Fig. 4.2(a).

Step 2. The \( \text{supp} (k_n) \) is indicated by the 9 filled circles, out of 12, in \( B_2 (u_n, c) \); see Fig. 4.2(b).

Stage 2:

There exist some dots in \( B_2 (u_n, \mu_1 c) \) (see Fig. 4.2(b)), and hence \( J_{n+1} = J_n \).

Stage 3:

By using the \( \text{supp} (k_n) \) obtained, the proposed algorithm finally updates the filter by (4.4).

In this case, 12 evaluations of \( \| u_n - u_i \|_2 \) need to be performed to obtain \( k_n \), and (4.3) and (4.4) are computed only for the length-9 on-support subvector of \( k_n \).

4.3.2 Complexity of the Proposed Algorithms with CSK

The total number of multiplications and comparisons of the proposed algorithm with Wendland’s CSK is \( (L+1)r_n+(L+1)s_n+14s_n+2 \) where \( r_n \) is the dictionary size, \( s_n \) is the size of the over-estimated support \( \hat{I}_n \) and \( s_n \) is the support size. The complexity
CHAPTER 4. CSK-NLMS

(a) Over-estimation of \( \text{supp}(k_n) \) by \( B_1(u_n, \sqrt{2}c) \).

(b) Support identification and the dictionary construction.

Figure 4.2: An illustration of the proposed algorithm.

of KNLMS with Gaussian kernel is \((L + 6) r_n + 3\). The proposed algorithm is more efficient than KNLMS when

\[
\hat{s}_n < \frac{5r_n + 1}{17} \quad (\approx 0.29r_n + 0.07 < 0.3r_n).
\]

(4.13)

The complexity of the algorithms is summarized in Table 4.2.

4.4 Numerical Examples

Computer simulations are conducted to verify the efficiency of the proposed algorithm. We use the Gaussian kernel and the Wendland’s CSK which is a popular family of CSK; see Table 4.1. In our experiments, the other CSKs exhibited almost the same performance as Wendland’s CSK, and thus we only show the results for Wendland’s one. The three adaptive algorithms are compared: KNLMS [80] with Gaussian (KNLMS-Gaussian), KNLMS with CSK (KNLMS-CSK), and the
Table 4.2: Computational complexity of the algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Kernel</th>
<th>Multiplication</th>
<th>Addition</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNLMS</td>
<td>Gaussian</td>
<td>$(L+5) r_n + 2$</td>
<td>$(L+1) r_n - 1$</td>
<td>$r_n + 1$</td>
</tr>
<tr>
<td></td>
<td>Matérn</td>
<td>$(L+5) r_n + 2$</td>
<td>$(L+3) r_n - 1$</td>
<td>$2r_n + 1$</td>
</tr>
<tr>
<td></td>
<td>Wendland</td>
<td>$(L+12) r_n + 2$</td>
<td>$(L+5) r_n - 1$</td>
<td>$r_n + 1$</td>
</tr>
<tr>
<td></td>
<td>Wu</td>
<td>$(L+14) r_n + 2$</td>
<td>$(L+6) r_n - 1$</td>
<td>$2r_n + 1$</td>
</tr>
<tr>
<td>Proposed</td>
<td>Matérn</td>
<td>$L\hat{s}_n + 5s_n + 2$</td>
<td>$(2L - 1) r_n + (L - 1) \hat{s}_n + 4s_n - 1$</td>
<td>$(L + 1)r_n + \hat{s}_n + 2s_n$</td>
</tr>
<tr>
<td></td>
<td>Wendland</td>
<td>$L\hat{s}_n + 12s_n + 2$</td>
<td>$(2L - 1) r_n + (L - 1) \hat{s}_n + 6s_n - 1$</td>
<td>$r_n + 1$</td>
</tr>
<tr>
<td></td>
<td>Wu</td>
<td>$L\hat{s}_n + 14s_n + 2$</td>
<td>$(2L - 1) r_n + (L - 1) \hat{s}_n + 7s_n - 1$</td>
<td>$2r_n + 1$</td>
</tr>
</tbody>
</table>

The proposed algorithm. The simulation settings are given as follows: $L = 2$, the output is generated as $d_n := \psi(u_n) + v_n, n \in \mathbb{N}$, with

$$
\psi(u) := \sum_{j=1}^{4} h^*_j \exp \left( -\frac{\|u - c_j\|^2}{2\sigma^*_j} \right),
$$

(4.14)

where $(h^*_1, h^*_2, h^*_3, h^*_4) = (1.5, -1.5, -0.5, 1.5)$, $(\sigma^*_1, \sigma^*_2, \sigma^*_3, \sigma^*_4) = ((0.4, 1, 0.4, 0.8), c_1 = [0.5, 0.5]^{\top}, c_2 = [-0.5, -0.5]^{\top}, c_3 = [0.5, -0.5]^{\top}$ and $c_4 = [-0.5, 0.5]^{\top}$. The input $u_n \in (-3, 3)^2$ obeys the i.i.d. uniform distribution. The noise $v_n$ is additive white Gaussian noise with SNR = 20dB. The step size for each algorithm is set to $\eta_n = 0.1, \forall n \in \mathbb{N}$. The parameter for KNLMS-Gaussian is set to $\sigma = 0.2$. The parameters for KNLMS-CSK and the proposed algorithm are set to $c = 0.8$. We set $\mu_0 = 0.5$ for KNLMS and KNLMS-CSK, and $\mu_1 = 0.29$ for the proposed algorithm; the $\mu_1$ is determined automatically with (4.12).

In Fig. 4.3, the average value of the final dictionary size $\tilde{r}_{3x10^4} = 439$ for all algorithms. The average over-estimated support size $\hat{s}_n$ and the average support size $s_n$ are plotted respectively by the green and red curves. The average and maximum values of $\text{supp}(k_n)$ were 29 and 31, respectively, and those for the over-estimated support were 36 and 38, respectively. The figure shows the efficiency of the $\ell_1$-based over-estimation because $\hat{s}_n - s_n \approx 3 \ll r_n \approx 200$ (see also Fig. 4.2).

Figure 4.4 shows the NMSE learning curves. It can be seen that all algorithms show comparable convergence performances. To see the complexity for each algorithm, we plot the number of multiplications and additions in Fig. 4.5. It can be seen that the number of multiplications of the proposed algorithm is smaller than the one of KNLMS; the number of multiplications plus comparisons is smaller as well. Consequently, the proposed algorithm can significantly reduce the complexity while keeping the estimation performance.
4.5 Conclusion

This chapter has presented an efficient KNLMS algorithm with the use of compactly supported kernels. The proposed algorithm was composed of three stages. In the first stage, the support of the kernelized input vector is identified by two steps through its over-estimate based on the circumscribed $\ell_1$-ball. In the second stage, the dictionary is constructed based on the knowledge about the support identified in the first stage. In the third stage, the filter is updated based on the support information acquired in the first stage and the dictionary constructed in the second stage. Overall, the complexity was reduced significantly as demonstrated by the numerical examples.
4.5. CONCLUSION

Figure 4.3: Dictionary size $\bar{r}_n$, the over-estimated support size $\bar{s}_n$ and the exact support size $\tilde{s}_n$.

Figure 4.4: NMSE learning curves.
Figure 4.5: The complexity of the algorithms at each iteration.
Chapter 5

Online Model Selection and Learning Method

5.1 Introduction

Online kernel learning has widely been studied for nonlinear estimation tasks over the past decades [75, 78, 80, 111, 112, 147]. Kernel design is one of the central issues, because the data arrive sequentially and are thus unavailable in advance for cross validation. The multikernel adaptive filtering framework [83, 107, 113] is a convex analytic recipe offering a practical remedy to the challenging issue of kernel design. Here, a typical choice is a set of Gaussian kernels with different scale-parameters (variances); the use of linear and Gaussian kernels has also been studied in [83]. In the previous studies, only a few kernels were employed. This implies that some knowledge were implicitly assumed available to design the kernels. In this chapter, as many Gaussian kernels as needed to include all possible scale-parameters are supposed to be employed. This eliminates the necessity of the knowledge for the kernel design. The key concept arising from the employment of many kernels is model selection (kernel selection). To accomplish a nonlinear estimation task, one needs to find an estimandum (a nonlinear function to be estimated) under the selected model. Namely, there are two sub-tasks for completing the nonlinear estimation: (i) model selection and (ii) learning (i.e., parameter estimation under the selected model).

The major contribution of this paper is the development of a fully-adaptive all-in-one learning scheme that jointly carries out the two sub-tasks. We propose an efficient online model-selection and learning scheme that employs those kernels with a wide range of scale-parameters within the multikernel adaptive filtering framework. The multikernel adaptive filter is formed as a linear combination of Gaussian functions (with different scales) centered at some data points. The set of those center points is referred to as the dictionary, and each center point as the dictionary data. The online model-selection and learning task is cast as a minimization problem of a sequence of data-dependent (and thus time-dependent) convex functions of a coefficient matrix. Here, the columns and rows of the matrix correspond to the dictionary data and kernels, respectively. Each of the convex functions is a superposition of three terms: (i) the instantaneous data fidelity function, (ii) the sum of
the $\ell_2$ norms of the columns (referred to as the column-block $\ell_1$ norm), and (iii) the sum of the $\ell_2$ norms of the rows (referred to as the row-block $\ell_1$ norm). The column-block $\ell_1$ norm here enhances column-wise sparsity (leading to sparsification of the dictionary), while the row-block $\ell_1$ norm enhances row-wise sparsity (leading to the selection of relevant kernels). The first fidelity term is smooth, but the other block-$\ell_1$ regularization terms are nonsmooth. Fortunately, however, the proximity operator of each nonsmooth term can be computed easily (i.e., the nonsmooth terms are proximable), and this allows to develop an efficient adaptive algorithm for the present task.

To suppress a sequence of time-dependent convex functions, the promising convex optimization methods [87, 88, 90, 93, 96, 148] have been extended successfully to adaptive algorithms [58, 60, 65–67, 97] (see Appendix A). We derive and use an adaptive extension of the GFBS method [93], because it minimizes the sum of a smooth function and multiple nonsmooth functions and is therefore most appropriate for our formulation. The derived algorithm, referred to as AGFBS, leverages the gradient of the smooth fidelity function and the proximity operators of the nonsmooth regularizers. The AGFBS algorithm is computationally efficient because the gradient and the proximity operators are all easy to compute. Reducing the time-dependent cost functions by AGFBS leads to the selection of appropriate kernels, the construction of a sparse dictionary, and the efficient parameter estimation under the selected kernels and the constructed dictionary.

The proposed scheme consists of the dictionary refinement process and the coefficient update process. The former process grows the dictionary and also discard unnecessary data from the dictionary, and the latter process is carried out by AGFBS. To be more precise, the coefficients for the unnecessary data automatically diminish by AGFBS, and such data are discarded through the dictionary refinement process (see Remark 2 in Section 5.4.4 for details). The proposed scheme has a higher degree of freedom compared to the multiple kernel learning (MKL) [149–156] (see Remark 5 in Section 5.4.4). The estimator in our framework can be viewed as an element of the sum of multiple RKHSs [145, 146], and the proposed scheme can be viewed as jointly
estimating an appropriate subspace of the sum space that efficiently approximates the *estimandum* and the coefficients to express a good estimator in the subspace (see Appendix B). Numerical examples show that the scheme can select appropriate kernels automatically and achieves excellent performance with a small-sized dictionary. The results show some striking advantages of the proposed scheme over the SimpleMKL method [154].

The remainder of the paper is organized as follows. The system model and the multikernel adaptive filtering framework are introduced in Section 5.3. The proposed online model-selection and learning scheme is presented in Section 5.4. Numerical examples to show the efficacy of the scheme are presented in Section 5.5, followed by conclusion in Section 5.6.

### 5.2 Adaptive Generalized Forward-backward Splitting Method

This chapter considers the asymptotic minimization problem of a sequence \( (\Theta_n)_{n \in \mathbb{N}} \) of convex cost functions given in (2.1). We will derive AGFBS to minimize (2.1) in an online fashion. The AGFBS algorithm basically uses the gradient for reducing the smooth term \( \phi \) and uses the proximity operators (2.9) for reducing the nonsmooth terms \( \Omega^{(s)} \). For the initial vectors \( h_0 := 0 \) and \( z_0^{(s)} := 0 \), AGFBS generates the sequences \((h_n)_{n \in \mathbb{N}} \subset \mathcal{H} \) and \((z_n^{(s)})_{n \in \mathbb{N}} \subset \mathcal{H}, s \in S \), recursively by

\[
\begin{align*}
    z_{n+1}^{(s)} &= z_n^{(s)} + \beta \left( \text{prox}_{\frac{\alpha_s}{2 \eta_n} \Omega^{(s)}} \left( 2h_n - z_n^{(s)} - \eta_n \nabla \phi (z_n) \right) - h_n \right), s \in S, \\
    h_{n+1} &= \sum_{s \in S} \alpha_s z_{n+1}^{(s)},
\end{align*}
\]

where \( \beta \in \left( 0, \min \left\{ \frac{3}{2}, \frac{\eta_n + 2}{2 \eta_n} \right\} \right) \) is the relaxation parameter, and \( \alpha_s > 0, s \in S \), are the convex combination coefficients satisfying \( \sum_{s \in S} \alpha_s = 1 \). The AGFBS algorithm can efficiently deal with a superposition of one smooth term and multiple proximable terms. It can be viewed as a nontrivial special case of the adaptive primal-dual splitting method [69]. If the cost function contains only one proximable term (i.e., in the case of \( S = 1 \)), one can apply the APFBS method [65].

### 5.3 Multikernel Adaptive Filtering

Multikernel adaptive filtering [83, 107, 157] employs multiple reproducing kernels. The key assumption in this chapter is that no prior information is available for designing an appropriate kernel to represent \( \psi \). This is a remarkable difference from the ordinary kernel adaptive filtering approaches. Due to the assumption, many possible kernels are employed: \( \kappa_q : U \times U \rightarrow \mathbb{R}, q \in Q := \{1, 2, \cdots, Q\} \). A typical

\[ (\eta_n)_{n \in \mathbb{N}} \] is the step size sequence.
choice is a set of Gaussian kernels

\[ \kappa_q(x, y) := \exp \left( -\frac{\|x - y\|_2^2}{2\sigma_q^2} \right), \quad x, y \in \mathcal{U}, q \in \mathcal{Q}, \]

with a wide range of scale-parameters \( \sigma_q > 0 \), where \( \| \cdot \| \) is the \( \ell_2 \) norm. Let \( \{ u_j \}_{j \in \mathcal{J}_n} \) be the dictionary indicated by the dictionary index set

\[ \mathcal{J}_n := \left\{ j_1^{(n)}, j_2^{(n)}, \ldots, j_{r_n}^{(n)} \right\} \subset \{0, 1, \ldots, n - 1\}, \]

where \( r_n \in \mathbb{N}^* \) is the size of the dictionary. A multikernel adaptive filter is given in the following form:

\[ \varphi_n(u) := \sum_{q \in \mathcal{Q}} \sum_{j \in \mathcal{J}_n} h^{(q)}_{j,n} \kappa_q(u, u_j), \quad u \in \mathcal{U}, \tag{5.3} \]

where \( h^{(q)}_{j,n} \in \mathbb{R} \) are the filter coefficients.
5.4 Proposed Scheme for Online Model-Selection and Learning

A block diagram of the proposed scheme is illustrated in Fig. 5.1, in which

\[
H_n := \begin{bmatrix}
h_{j_1^{(n+1)}, n} & h_{j_2^{(n+1)}, n} & \cdots & h_{j_{r_{n+1}}^{(n+1)}, n}
\end{bmatrix} \in \mathbb{R}^{Q \times r_{n+1}}
\]

is the coefficient matrix with \( h_{j,n} := \begin{bmatrix} h_{j,1}^{(1)}, h_{j,1}^{(2)}, \cdots, h_{j,1}^{(Q)} \end{bmatrix}^T \in \mathbb{R}^Q, j \in J_{n+1} \). The proposed scheme consists of two processes: (i) the dictionary refinement and (ii) the coefficient update (carried out by AGFBS). The two processes cannot completely be separated from each other. Once a new measurement pair \((u_n, d_n)\) arrives, the proposed scheme first refines the dictionary (i.e., \( J_n \) is updated to \( J_{n+1} \) and \( H_n \) is resized accordingly), and then updates the coefficients for the updated dictionary indicated by \( J_{n+1} \). Here, the dictionary refinement is an important process to keep the dictionary updated, and it includes the shrink-and-pruning step as well as the growing step. To be strict, the coefficient-updating process is in charge of the shrinking part, and therefore the pruning step actually involves both processes. This is the reason why the two processes cannot be separated from each other. Alternating the two processes lead to online model-selection and learning; i.e., the proposed scheme simultaneously accomplishes, in an online fashion, (i) kernel selection, (ii) dictionary construction, and (iii) parameter estimation.

We describe the details of the dictionary refinement process in Section 5.4.1. We then present our cost function and AGFBS that carries out the coefficient updating process in Sections 5.4.2 and 5.4.3, respectively. We finally present some remarks on the proposed scheme in Section 5.4.4.

5.4.1 Dictionary Refinement

The dictionary refinement process is described below.

1. **Shrink-and-pruning step.**

   (a) Shrink each column vector \( h_{j,n} \) of the coefficient matrix \( H_n \) at each iteration of AGFBS (see the coefficient updating process described in Section 5.4.3).

   (b) Discard the nearly-zero columns (corresponding to the unnecessary elements of the dictionary). See (5.4) below.

2. **Growing step.** Add the new datum to the pruned dictionary based on some error criterion. See (5.6) below.

Specifically, the pruned dictionary is defined with the following index set:

\[
J_{n+1}^- := \left\{ j \in J_n \mid \| h_{j,n} \|_2 \geq \epsilon_J \left( \prod_{j \in J_n} \| h_{j,n} \|_2 \right)^{1/r_n} \right\}, \tag{5.4}
\]
where $\varepsilon_{J} \geq 0$. It should be mentioned that $\left(\prod_{j \in J} \| h_{j,n} \|_2\right)^{1/r_n}$ is the geometric mean of the norms of the column coefficient-vectors $h_{j,n}$. Figure 5.2 illustrates the geometric and arithmetic means of the norms of $h_{j,n}$. The arithmetic mean tends to be governed by those samples with large amplitudes in general, and it actually misses some significant components in the figure (e.g., the components of $i = 14, \cdots, 20$). On the other hand, the geometric mean well distinguishes between necessary and unnecessary dictionary elements.

We define the estimation error of the current kernel adaptive filter with the pruned dictionary as follows:

$$
\varepsilon_n^{(J^-)} := \hat{\varphi}_n(u_n) - d_n,
$$

where

$$
\hat{\varphi}_n(u) := \sum_{q \in U} \sum_{j \in J_{n+1}} h_{j,n}^{(q)} \kappa_q(u, u_j).
$$
5.4. PROPOSED SCHEME

Note here that \( \hat{\varphi}_n(u) \) is slightly different from \( \varphi_n(u) \) due to the absence of the unnecessary dictionary elements. If \( \left| \varepsilon_n^{(J^-)} \right| \) is sufficiently large relative to the desired response \( |d_n| \), the new datum is added to the pruned dictionary. Specifically, the dictionary index set is updated as

\[
\mathcal{J}_{n+1} := \begin{cases} 
\mathcal{J}_n^- \cup \{n\}, & \text{if } \left| \varepsilon_n^{(J^-)} / d_n \right| > \delta, \quad n \in \mathbb{N}, \\
\mathcal{J}_n^-, & \text{otherwise},
\end{cases}
\]

(5.6)

where \( \delta \geq 0 \) is the threshold. Whenever the dictionary grows, the associated coefficients are initialized to zero; i.e., \( h_{n,n}^{(1)} := h_{n,n}^{(2)} := \cdots := h_{n,n}^{(Q)} := 0 \).

5.4.2 Cost Function with Double Regularizers

Let us define an inner product between two matrices \( A, B \in \mathbb{R}^{Q \times r_{n+1}} \) by \( \langle A, B \rangle := \text{trace}(A^T B) \), and its induced norm by \( \|A\|_F := \sqrt{\langle A, A \rangle} \) (which is the Frobenius norm).

After the dictionary refinement described in Section 5.4.1, the multikernel adaptive filter is given by

\[
\hat{\varphi}_n(u) = \sum_{q \in Q} \sum_{j \in \mathcal{J}_{n+1}} h_{j,n}^{(q)} \kappa_q(u, u_j), \quad u \in \mathcal{U}.
\]

The output \( \hat{\varphi}_n(u_n) \) to the current input vector \( u_n \) can be expressed in a matrix form as follows:

\[
\hat{\varphi}_n(u_n) := \langle H_n, K_n \rangle, \quad n \in \mathbb{N},
\]

where

\[
K_n := \begin{bmatrix}
k_{j_{1},n} & k_{j_{2},n} & \cdots & k_{j_{r_{n+1}},n}
k_{j_{1},n} & k_{j_{2},n} & \cdots & k_{j_{r_{n+1}},n}
\end{bmatrix} \in \mathbb{R}^{Q \times r_{n+1}},
\]

\[
k_{j,n} := \begin{bmatrix}
\kappa_1(u_n, u_j), \kappa_2(u_n, u_j), \cdots, \kappa_Q(u_n, u_j)
\end{bmatrix}^T \in \mathbb{R}^Q, \quad j \in \mathcal{J}_{n+1}.
\]

For a variable matrix

\[
H := \begin{bmatrix}
h_1 & h_2 & \cdots & h_{r_{n+1}}
\end{bmatrix} \\
:= \begin{bmatrix}
\chi_1 & \chi_2 & \cdots & \chi_Q
\end{bmatrix}^T \in \mathbb{R}^{Q \times r_{n+1}},
\]

(5.7)

our cost function to minimize is given as follows:

\[
\Theta_n(H) := \phi_n(H) + \Omega_n^{(1)}(H) + \Omega_n^{(2)}(H), \quad n \in \mathbb{N},
\]

(5.8)

where

\[
\phi_n(H) := \frac{1}{2} q^2 \left( H, \Pi_n^{(F)} \right) \quad \text{(fidelity)},
\]

\[
\Omega_n^{(1)}(H) := \lambda_1 \sum_{i=1}^{r_{n+1}} w_{i,n}^{(1)} \|h_i\|_2 \quad \text{(data selection)},
\]

\[
\Omega_n^{(2)}(H) := \lambda_2 \sum_{i=1}^{r_{n+1}} w_{i,n}^{(2)} \|h_i\|_2 \quad \text{(data selection)},
\]

\[
\Omega_n^{(3)}(H) := \lambda_3 \sum_{i=1}^{r_{n+1}} w_{i,n}^{(3)} \|h_i\|_2 \quad \text{(data selection)},
\]

\[
\Omega_n^{(4)}(H) := \lambda_4 \sum_{i=1}^{r_{n+1}} w_{i,n}^{(4)} \|h_i\|_2 \quad \text{(data selection)},
\]

\[
\Omega_n^{(5)}(H) := \lambda_5 \sum_{i=1}^{r_{n+1}} w_{i,n}^{(5)} \|h_i\|_2 \quad \text{(data selection)}.
\]
\[
\Omega_n^{(2)}(H) := \lambda_2 \sum_{q \in Q} w_{q,n}^{(2)} \|X_q\|_2 \text{ (kernel selection)}.
\]

Here, \(\lambda_1, \lambda_2 \geq 0\) are the regularization parameters,

\[
w_{i,n}^{(1)} := r_{n+1} \nu_{i,n}^{(1)} \left( \sum_{i=1}^{r_{n+1}} \nu_{i,n}^{(1)} \right)^{-1}, \quad i = 1, 2, \ldots, r_{n+1},
\]

\[
w_{q,n}^{(2)} := Q \nu_{q,n}^{(2)} \left( \sum_{p \in Q} \nu_{p,n}^{(2)} \right)^{-1}, \quad q \in Q,
\]

are the weights with \(\nu_{i,n}^{(1)} := \|h_{j_{i+1,n}}\|_2^{-1}, \quad i = 1, 2, \ldots, r_{n+1}\), and \(\nu_{q,n}^{(2)} := \|X_{q,n}\|_2^{-1}, \quad q \in Q\) (where \(X_{q,n} \in \mathbb{R}^{r_{n+1}}\) denotes the \(q\)th column of \(H_n^\top\)), and \(d(H, \Pi_n^{(F)}) := \min_{X \in \Pi_n^{(F)}} \|X - H\|_F\) is the metric distance from a point \(H\) to the hyperplane \(\Pi_n^{(F)} = \{H \in \mathbb{R}^{Q \times r_{n+1}} \mid \langle H, K_n \rangle = d_n\}\).

Note here that large weights are allocated to such data groups (or kernel groups) that hardly contribute to the estimation. As a result, those unnecessary data groups shrink and diminish quickly. The role of each term in (5.8) is summarized below.

- \(\phi_n(H)\) measures the instantaneous data fidelity, contributing to the reduction of estimation errors.
- \(\Omega_n^{(1)}(H)\) is the column-block \(\ell_1\) norm to promote the column-wise sparsity of \(H\), contributing to the data selection.
- \(\Omega_n^{(2)}(H)\) is the row-block \(\ell_1\) norm to promote the row-wise sparsity of \(H\), contributing to the kernel selection.

### 5.4.3 Coefficient Updating Rule

Due to the nonsmoothness of \(\Omega_n^{(1)}\) and \(\Omega_n^{(2)}\), their gradients are unavailable. Fortunately, the proximity operators of \(\Omega_n^{(1)}\) and \(\Omega_n^{(2)}\) can be computed easily, and therefore \(\Theta_n\) can be treated by extending the GFBS algorithm [93]. Let \(Z_n^{(1)}\) and \(Z_n^{(2)}\) be the auxiliary matrices corresponding respectively to the two regularizers \(\Omega_n^{(1)}\) and \(\Omega_n^{(2)}\). According to the dictionary refinement, all the matrices \(H_n, Z_n^{(1)},\) and \(Z_n^{(2)}\) are resized from \(Q \times r_n\) matrices (associated with \(J_n\)) into \(Q \times r_{n+1}\) ones (associated with \(J_{n+1}\)) prior to each AGFBS update. When \(n \in J_{n+1}\) (i.e., when the new datum is added into the dictionary), the new column of the resized matrix \(H_n\) is defined as \(h_{n,n} := 0\); the same applies to \(Z_n^{(1)}\) and \(Z_n^{(2)}\). For the initial vectors \(H_0 := 0 \in \mathbb{R}^{Q \times 1}\) and \(Z_0^{(1)} := Z_0^{(2)} := 0 \in \mathbb{R}^{Q \times 1}\), AGFBS generates the sequences \((H_n)_{n \in \mathbb{N}} \subset \mathbb{R}^{Q \times r_{n+1}}\) and \((Z_n^{(s)})_{n \in \mathbb{N}} \subset \mathbb{R}^{Q \times r_{n+1}}, s = 1, 2,\) recursively by
5.4. PROPOSED SCHEME

\[ Z_{n+1}^{(s)} := Z_n^{(s)} + \beta \left( \text{prox}_{\alpha_1 \Omega_n^{(1)}} \left( 2H_n - Z_n^{(s)} - \eta_n \nabla \phi_n(H_n) \right) - H_n \right), \quad s = 1, 2, \quad (5.12) \]

\[ H_{n+1} := \alpha_1 Z_{n+1}^{(1)} + \alpha_2 Z_{n+1}^{(2)}, \quad (5.13) \]

where \( \alpha_1 := \alpha \) and \( \alpha_2 := 1 - \alpha \) for \( \alpha \in (0, 1) \). The gradient \( \nabla \phi_n(H_n) \) in (5.12) can be computed as

\[ \nabla \phi_n(H_n) = H_n - P_{\Pi_n^{(p)}}(H_n). \quad (5.14) \]

Here,

\[ P_{\Pi_n^{(p)}}(H_n) := \arg\min_{X \in \Pi_n^{(p)}} \| X - H_n \|_F = H_n - \frac{\varepsilon_n}{\| K_n \|_F^2} K_n \quad (5.15) \]

is the orthogonal projection of \( H_n \) onto the hyperplane \( \Pi_n^{(p)} \) in (5.11). For the matrix \( \mathbf{H} \) in (5.7), the proximity operators of \( \Omega_n^{(1)} \) and \( \Omega_n^{(2)} \) have the following closed-form expressions:

\[ \text{prox}_{\alpha_1 \Omega_n^{(1)}}(H) = \sum_{i=1}^{r_{n+1}} \max \left\{ 1 - \eta_n \lambda_{1,w_{i,n}}^{(1)}, 0 \right\} h_i e_{i, r_{n+1}}^T, \]

\[ \text{prox}_{\alpha_2 \Omega_n^{(2)}}(H) = \sum_{q=1}^{Q} \max \left\{ 1 - \eta_n \lambda_{2,w_{q,n}}^{(2)}, 0 \right\} e_{q, Q} X_q^T, \quad (5.16) \]

where \( e_{m,t} \in \mathbb{R}^t, m, t \in \mathbb{N}^* \), denotes the length-\( t \) unit vector that has one at the \( m \)th entry and zeros elsewhere. Table 5.1 summarizes the proposed scheme. A geometric interpretation of the proposed scheme is given in Appendix B.

5.4.4 Remarks

Some remarks on the proposed approach are presented below.

**Remark 1** (On the choice of \( \varepsilon_J \) and \( \delta \)). The parameters \( \varepsilon_J \) and \( \delta \) appearing in (5.4) and (5.6) of Section 5.4.1, respectively, are the key parameters for the dictionary construction. If the dictionary size is allowed to be large, one can set \( \delta \) to a relatively small value (such as 0.1 – 0.5) so that a new datum is easily added into the dictionary. In this case, we recommend that \( \varepsilon_J \) is set to a large value (such as 0.2). If, on the other hand, the dictionary size should be kept small, one should set \( \delta \) to a large value (such as 0.7 – 0.95). In this case, \( \varepsilon_J \) should be set to a small value (such as 0.05) in order to let the infrequently entering elements stay in the dictionary unless their contributions become negligible. In our extensive simulations, the proposed scheme is insensitive to small fluctuations of those parameters.

**Remark 2** (Mechanism of data/kernel selection). As described already at the beginning of Section 5.4, the proposed scheme consists of the dictionary refinement process and the coefficient update process (see also Fig. 5.1). In the coefficient update process, AGFBS (5.12) adjusts the coefficients to suppress the instantaneous errors by means of the gradient step and then shrinks the coefficients by means of the pair of proximity operators each of which is in charge of data selection or kernel
selection. The coefficient column- (or row-) vectors corresponding to unnecessary
data (or irrelevant kernels) are enforced to be zero vectors because of the shrinking
effects of the proximity operators. This is the mechanism of data/kernel selection.
In the growing and pruning process, the dictionary elements corresponding to the in-
active column vectors are discarded to sparsify the dictionary for memory efficiency.
Experiment B in Section 5.5 shows that the proposed scheme selects the relevant
kernels automatically.

Remark 3 (Adaptivity). One may need to consider “adaptivity” due to the fol-
lowing two reasons. First, the optimization space (a subspace of the RKHS under
consideration) changes over time in general because the dictionary is constructed in
an online fashion as described in Section 5.4.1. In this case, it is clear that the solu-
tion changes accordingly. Second, the unknown system is often time-varying. In this
case, in addition to the change of the optimal coefficients, the appropriate kernels
and dictionary may also change in time. For instance, the earth’s global relief, which
is used in Experiment C of Section 5.5, is gradually moving in practice. Referring to
(5.12), before performing the proximity operator, AGFBS updates the current coeffi-
cient matrix as \[ H_n - \eta \nabla \phi_n (H_n) + [H_n - Z^{(s)}_n]. \] Here, the terms in the first bracket
are nothing but the update equation of the classical NLMS algorithm [3,4] and the
terms in the second bracket can be regarded as adjustments. The AGFBS algorithm
thus inherits from NLMS the adaptivity to the change of the solution. The adaptivity
of AGFBS in the first case mentioned above will be verified by simulations in Sec-
tion 5.5. Regarding the adaptivity in the second case, it has been shown already in
[158] that our online model selection and learning scheme can adapt to the changes
of the appropriate kernels and dictionary. Indeed, a different adaptive algorithm
based on some approximation described in Remark 6 below was used in [158], and
our proposed “exact” scheme without any approximation is expected to enjoy better
adaptivity than the approximate scheme.

Remark 4 (Avoidance of overfitting). Despite its high degree of freedom, the pro-
posed scheme does not suffer from overfitting in principle. This is because the spar-
sification due to the two regularizers automatically excludes those irrelevant kernels
which may cause overfitting [159].

Remark 5 (Related works). MKL has been studied well in the machine learning
community [149–156]. It performs learning with a ‘single’ kernel that is designed
appropriately as a convex combination of multiple kernels; this is based on the well-
known fact that any nonnegative combination of positive definite kernels is again
positive definite. Hence, the degree of freedom in MKL is \( Q + r_n \). Meanwhile, the
degree of freedom in the proposed scheme is \( Q r_n \), which is much higher than \( Q + r_n \) in
practical situations. The higher degree of freedom enables to express local structures
of \( \psi \) individually by means of suitable kernels, leading to accurate approximations of
high- and low- frequency components with a reasonably-small-sized dictionary (see
[83]). The term ‘frequency’ here means the frequency (defined with the Fourier
transform as usual) of a function. It will be shown in Section 5.5.1 that the proposed
scheme requires a smaller number of dictionary elements than MKL to attain the minimum (normalized) mean squared errors due to the higher degree of freedom.

Multiscale kernel learning [160–164] is also relevant to the proposed scheme. The major advantage over the multiscale kernel learning is the full adaptivity; i.e., the model selection (the kernel selection and the dictionary construction) and the parameter estimation are jointly accomplished by a single scheme. This is clearly beneficial specifically when the frequencies of the estimandum $\psi$ change over time.

**Remark 6** (Another possible scheme). In [159], an approximate scheme based on APFBS [65] has been presented to suppress the cost functions in (5.8). Since APFBS cannot be applied directly to $\Theta_n$ in (5.8) due to the presence of two nonsmooth functions, it is applied to an approximation of $\Theta_n$ that involves the so-called Moreau envelope of $\Omega_1^{(1)}$. Here, the Moreau envelope is smooth, and its gradient can be computed easily with its proximity operator. Comparisons of this approximate scheme with the proposed scheme would be left as an interesting future work. Another possible research direction in the future includes the use of a non-Euclidean metric, since it has been shown that a Hilbertian metric yields better convergence behaviors than the Euclidean metric [84, 85, 110, 165, 166].
Table 5.1: Proposed scheme

1: Requirements and Initialization:
   scale-parameters $\sigma_q, q \in \mathbb{Q} = \{1, 2, \ldots, Q\}$
   regularization parameters $\lambda_1 > 0, \lambda_2 > 0$
   step size $\eta_n \in (0, 2)$
   $\beta \in \left(0, \min\{3/2, \frac{n_0+1}{2n_0}\}\right), \epsilon_{\mathcal{F}} \geq 0, \delta \geq 0, \alpha \in (0, 1)$
   $H_0 = Z_0^{(1)} = Z_0^{(2)} = 0 \in \mathbb{R}^{Q \times 1}$, $\mathcal{J}_0 = \{0\}$, and $r_0 = 1$

2: for $n = 0, 1, 2, \ldots, n$ do
   3: Filter output: $\varphi_n(u_n)$ ▷ see (5.3)
   4: Dictionary refinement (Section 5.4.1)
   5: Dictionary pruning: $\mathcal{J}_{n+1}^- \leftarrow \mathcal{J}_n$ ▷ see (5.4)
   6: Evaluate: $\varepsilon_{\mathcal{F}}(\mathcal{J}^-)$ ▷ see (5.5)
   7: Dictionary growing: $\mathcal{J}_{n+1} \leftarrow \mathcal{J}_{n+1}^-$ ▷ see (5.6)
   8: Resize the matrices $H_n, Z_n^{(1)}, Z_n^{(2)}$
      If $n \in \mathcal{J}_{n+1}$, the new columns are set to the zero vectors.
   9: Coefficient-update (Section 5.4.3)
   10: Compute $\nabla \varphi_n(H_n)$ ▷ see (5.14) and (5.15)
   11: Compute ▷ see (5.12)
      $\tilde{Z}_n^{(1)} = 2H_n - Z_n^{(1)} - \eta_n \nabla \varphi_n(H_n)$
      $\tilde{Z}_n^{(2)} = 2H_n - Z_n^{(2)} - \eta_n \nabla \varphi_n(H_n)$
   12: Update the auxiliary matrices ▷ see (5.12) and (5.16)
      $Z_{n+1}^{(1)} = Z_n^{(1)} + \beta \left(\text{prox}_{\frac{\alpha}{\eta_0} \Omega_n^{(1)}} \left(\tilde{Z}_n^{(1)}\right) - H_n\right)$
      $Z_{n+1}^{(2)} = Z_n^{(2)} + \beta \left(\text{prox}_{\frac{\alpha}{\eta_0} \Omega_n^{(2)}} \left(\tilde{Z}_n^{(2)}\right) - H_n\right)$
   13: Update the coefficient matrix ▷ see (5.13)
      $H_{n+1} = \alpha \tilde{Z}_{n+1}^{(1)} + (1 - \alpha) \tilde{Z}_{n+1}^{(2)}$
   14: end for
5.5 Numerical Examples

We show the efficacy of the proposed scheme through three experiments. Since there exists no other online model-selection and learning scheme, the performance of the scheme is compared to KNLMS [80] with its scale-parameter $\sigma > 0$ tuned manually. A comparison with SimpleMKL [154] is also presented in Section 5.5.1. Note here that GFBS is a batch method, and cannot be applied directly to online tasks. In all experiments, the NMSEs and dictionary sizes are computed by taking the arithmetic averages over 300 independent trials. The noise process $(v_n)_{n \in \mathbb{N}}$ is white Gaussian with SNR 20 dB. The estimandum $\psi$ in each experiment is described below.

Experiment A: We aim to evaluate the basic performance of the proposed scheme and to show that the kernel selection and the dictionary construction are realized appropriately. For this purpose, the following simple model is considered:

$$d_n = \psi_1(u_n) + v_n,$$

$$\psi_1(u) := \exp \left( \frac{- (u + 1)^2}{2 \times 0.1^2} \right) + \exp \left( \frac{- (u - 1)^2}{2 \times 1^2} \right)$$

for $u \in \mathcal{U} := \mathbb{R}$. The input data is randomly generated as i.i.d. white Gaussian noise $(u_n)_{n \in \mathbb{N}} \sim \mathcal{N}(0, 1)$.

Experiment B: We aim to show that the proposed scheme can also estimate non-Gaussian multivariate functions. For this purpose, the following multivariate function is considered [167]:

$$d_n = \psi_2(u_n) + v_n,$$

$$\psi_2(u) := 4 \left( u_1 - 2 + 8u_2 - 8u_2^3 \right)^2 + (3 - 4u_2)^2 + 16\sqrt{u_3 + 1}(2u_3 - 1)^2$$

for $u = [u_1, u_2, u_3]^T \in \mathcal{U} := [0, 1]^3$. The input vector $u_n$ obeys the i.i.d. uniform distribution between 0 and 1.

Experiment C: We aim to show that the proposed scheme works for real data. We consider the following scenario: a single sensor on a flying (or underwater) object measures altitude data at each position of a land in a sequential manner, and it estimates the global altitude map from the partial information in an online fashion. This scenario may arise when one has to employ a low-cost unmanned probe that is loaded with a single sensor for monitoring some global environments; e.g., an unmanned aerial vehicle (UAV), an autonomous underwater vehicle (AUV), etc. Such an unmanned probe is desired to learn from sequential data with low power consumption, and hence it requires efficient online learning. The ETOPO1 global relief model is used; this is a 1 arc-minute global relief model of earth’s surface provided by National Oceanic and Atmospheric Administration (NOAA) [168]. This relief dataset gives the points $[u_1, u_2, z]$ of the earth’s surface with longitude

\[http://www.ngdc.noaa.gov/mgg/global/\]
Table 5.2: Parameter settings for Experiment A.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Proposed</th>
<th>KNLMS\textsubscript{1}</th>
<th>KNLMS\textsubscript{2}</th>
<th>KNLMS\textsubscript{3}</th>
<th>KNLMS\textsubscript{4}</th>
<th>SimpleMKL</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma_1)</td>
<td>(0.01)</td>
<td>(0.01)</td>
<td>(0.01)</td>
<td>(0.01)</td>
<td>(0.1)</td>
<td>(0.1)</td>
</tr>
<tr>
<td>(\sigma_2)</td>
<td>(0.1)</td>
<td>(0.1)</td>
<td>(0.1)</td>
<td>(0.1)</td>
<td>(1.0)</td>
<td>(1.0)</td>
</tr>
<tr>
<td>(\sigma_3)</td>
<td>(1.0)</td>
<td>(1.0)</td>
<td>(1.0)</td>
<td>(1.0)</td>
<td>(10)</td>
<td>(10)</td>
</tr>
<tr>
<td>(\eta_n)</td>
<td>(0.1)</td>
<td>(0.1)</td>
<td>(0.1)</td>
<td>(0.1)</td>
<td>(1.0)</td>
<td>(1.0)</td>
</tr>
<tr>
<td>(\beta)</td>
<td>(1)</td>
<td>(1)</td>
<td>(1)</td>
<td>(1)</td>
<td>(10^{-4})</td>
<td>(10^{-4})</td>
</tr>
</tbody>
</table>

\(u_1 \in \{-180, -180 + \frac{1}{60}, \ldots, 180\}\), latitude \(u_2 \in \{-90, -90 + \frac{1}{60}, \ldots, 90\}\), and altitude \(z \in \mathbb{R}\) at each point \((u_1, u_2)\). Here, for instance, \(u_1 = -180\) stands for 180° west-longitude, and \(u_2 = 90\) stands for 90° north-latitude. From the relief model\(^3\), 31 × 31 points are picked up: \(u_1 \in \{138.5, 138.5 + \frac{1}{60}, \ldots, 139\}\) and \(u_2 \in \{34.5, 34.5 + \frac{1}{60}, \ldots, 35\}\). We regard the altitude \(z\) as a function \(\psi_3(u)\) of \(u = [u_1, u_2]^T \in \mathcal{U} := [138.5, 139.0] \times [34.5, 35.0]\), and seek to estimate the nonlinear function \(\psi_3\) from the sequentially arriving measurements of the longitude and the latitude. The considered model is given as follows:

\[
d_n = \psi_3(u_n) + v_n
\]

for \((u_n)_{n \in \mathbb{N}} \in \mathcal{U}\).\(^4\) The input data \(u_n\) are generated randomly from the i.i.d. uniform distribution over the region \([138.5, 139] \times [34.5, 35]\). Figures 5.9(a) and 5.10(a) show the contours and the three-dimensional model of \(\psi_3\), respectively. From Fig. 5.10(a), it is seen that \(\psi_3\) contains a high frequency component (the blue part corresponding to the ocean floor) and a low frequency component (the red part corresponding to the land).

In all experiments, we employ the variable-metric projection technique [117] to enhance the performance (see Appendix C). We mention that the Euclidean-metric projection has been used in [158, 159].

### 5.5.1 Experiment A

Table 5.2 summarizes the parameter settings for the proposed scheme, KNLMS and SimpleMKL. Note that we intentionally employ the irrelevant kernels (with \(\sigma_1\) and \(\sigma_4\)) to show that the scheme can select the relevant kernels automatically. Figure 5.3

\(^3\)The picked area is Izu Peninsula of Japan, covering 138.5° – 139° east-longitudes and 34.5° – 35° north-latitudes.

\(^4\)\(\psi_3\) is generated by the cubic spline interpolation using not-a-knot end conditions in MATLAB.
shows the NMSE learning curves and the evolutions of the dictionary sizes. It can be seen that the scheme attains comparable performance with smaller dictionary sizes compared to KNLMS$_2$, which uses the manually-tuned best scale-parameter $\sigma^2 = 0.1$. (Inspecting the initial adaptation phase up close, we observed that the scheme actually attains faster initial convergence than KNLMS$_2$.) KNLMS$_1$ uses a kernel overfitting to the estimandum, and thus causes slow convergence with a large size of dictionary. In contrast, KNLMS$_3$ and KNLMS$_4$ use such kernels that underfit to the estimandum, and thus suffer from high steady-state NMSEs. Figure 5.4 illustrates how the estimandum $\psi_1$ (depicted in red) is estimated by the proposed scheme and KNLMS$_2$. Each gray curve corresponds to each term of the expansion in (5.3). One can see that the estimate curve (depicted in blue color) of the proposed scheme is composed of only two components, whereas that (depicted in green color) of KNLMS$_2$ is composed of 126 components. This remarkable reduction is realized by the use of multiple kernels. In addition, the two peaks of $\psi_1$ are perfectly identified by the proposed scheme automatically. Figure 5.5(a) illustrates the $\ell_2$ norms of
the columns of the coefficient matrix $H_n$. It can be seen that the relevant kernels (corresponding to $\sigma_2$ and $\sigma_3$) are identified. The results depicted in Figs. 5.4(a) and 5.5(a) show that the proposed scheme successfully achieves the model selection and learning.

We also make a comparison with SimpleMKL by using the toolbox.\(^5\) SimpleMKL employs the same kernels as the proposed scheme. The values of the parameters $C$, $\epsilon_{\text{reg}}$, and $\lambda_{\text{reg}}$ are given in Table 5.2. Since SimpleMKL is a batch method, we compare the CPU running time to attain each level of NMSE. The values of NMSE are computed with 300 test samples and with 300 independent trials. The simulations are performed on a PC powered by Core i7-6700K (4.0GHz) processor with maximum 2GB of RAM running MATLAB R2016a on Windows 10 64bit.

Figure 5.6(a) plots NMSE against the CPU running time. Each mark of each curve corresponds to the sample size (or the iteration number) 5, 10, 100, 200, 400, 600, 800, 1000, 2000, and 3000. For the proposed scheme, each mark shows the NMSE at each iteration and the accumulated time consumed up to each iteration. For SimpleMKL, for each sample size, an optimization problem is solved by an iterative method, and each mark shows the NMSE for the obtained solution and the time consumed to compute the solution. The figure thus gives the actual required time and the obtained NMSE for each method given each set of samples. Figure 5.6(b) plots the dictionary size against the number of training samples. The results show the significant advantages of the proposed scheme over SimpleMKL from two aspects. First, the proposed scheme dramatically reduces the time to achieve a nearly optimal NMSE performance. Specifically, the proposed scheme requires 0.6 seconds, while SimpleMKL requires 200 seconds at least. Second, the dictionary size of the proposed scheme is no more than 18\(^2\) to attain the nearly optimal NMSE performance, while that of SimpleMKL is 2000 at least. This significant reduction of the dictionary size is due to the higher degrees of freedom (see Remark 5 in Section 5.4.4).

\(^5\)http://asi.insa-rouen.fr/enseignants/~arakoto/code/mklindex.html
5.5. NUMERICAL EXAMPLES

Figure 5.4: The estimandum $\psi_1$ (red) and its estimates (blue or green).

Figure 5.5: The $\ell_2$-norms of the column vectors of $H_n$. 
Figure 5.6: A comparison with SimpleMKL.
### Table 5.3: Parameter settings for Experiment B.

<table>
<thead>
<tr>
<th>Proposed</th>
<th>( \sigma_1 := 10^{-4} )</th>
<th>( \eta_n = 0.4, \beta = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma_2 := 2 \times 10^{-4} )</td>
<td>( \epsilon_J = 0.05, \delta = 0.95 )</td>
<td></td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \alpha = 0.5 )</td>
<td></td>
</tr>
<tr>
<td>( \sigma_{63} := 9 \times 10^2 )</td>
<td>( \lambda_1 = 10^{-3} )</td>
<td></td>
</tr>
<tr>
<td>KNLMS$_{Best}$</td>
<td>( \sigma := 0.45 )</td>
<td>( \eta_n = 0.4, \mu_0 = 0.9 )</td>
</tr>
</tbody>
</table>

#### 5.5.2 Experiment B

The proposed scheme employs 63 Gaussian kernels with the scale-parameters \( \sigma_q = a \times 10^b \) for \( a \in \{1, 2, \cdots, 9\} \) and \( b \in \{-4, -3, \cdots, 2\} \). The manually-tuned best scale-parameter is used for the KNLMS algorithm. Table 5.3 summarizes the parameter settings. Viewing Fig. 5.5(b), one can observe that the kernels associated with the scale-parameters \( \sigma_{21} \) to \( \sigma_{31} \) are mainly selected. The mechanism of kernel selection is explained in Remark 3 in Section 5.4.4. The best scale-parameter used for KNLMS satisfies \( \sigma_{31} < 0.45 < \sigma_{32} \). In general, the use of a smaller scale-parameter for KNLMS indeed yields better NMSE but with an increased size of dictionary. The kernel with the smaller scale-parameter would thus give a better model than that with \( \sigma = 0.45 \). Nevertheless, KNLMS cannot leverage this better model because it cannot avoid the increase of the dictionary size due to the use of a single kernel. On the other hand, owing to the use of multiple kernels, the proposed scheme achieves better NMSE performance by exploiting the kernels with multiple scales without increasing the dictionary size. Note that those kernels with short-length coefficient vectors (including those with \( \sigma_{20} \) and \( \sigma_{32} \)) play a supporting role in the estimation to express the low frequency components. Figure 5.7 shows the NMSE curves and the average dictionary sizes. The proposed scheme achieves the low steady-state NMSE with the small dictionary size because of the use of a wide range of scale-parameters.
(a) NMSE learning curves.

(b) Average dictionary sizes.

Figure 5.7: Results for Experiment B.
Table 5.4: Parameter settings for Experiment C.

<table>
<thead>
<tr>
<th></th>
<th>Proposed</th>
<th>KNLMS_{Best}</th>
<th>KNLMS_{GM}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\sigma_1 := 10^{-4}$</td>
<td>$\sigma := 1.0$</td>
<td>$\sigma := \sigma_{GM} \approx 0.414$</td>
</tr>
<tr>
<td></td>
<td>$\sigma_2 := 2 \times 10^{-4}$</td>
<td>$\eta_n = 0.2, \beta = 1$</td>
<td>$\eta_n = 0.2, \mu_0 = 0.8$</td>
</tr>
<tr>
<td></td>
<td>$\vdots$</td>
<td>$\epsilon_J = 0.2$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\sigma_{63} := 9 \times 10^2$</td>
<td>$\delta = 0.3, \alpha = 0.5$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\lambda_1 = 4 \times 10^{-4}$</td>
<td>$\lambda_2 = 8 \times 10^{-6}$</td>
<td></td>
</tr>
</tbody>
</table>

5.5.3 Experiment C: Real Data

We finally show the performance of the proposed scheme for the real data, which is composed of several frequency components. Table 5.4 summarizes the parameter settings. The best scale-parameter $\sigma_{\text{Best}} := \sigma_28 = 1.0$ is used for KNLMS. For reference, KNLMS with the geometric mean $\sigma_{\text{GM}} \approx 0.414$ of $\sigma_1, \sigma_2, \ldots, \sigma_{63}$, which is a central one among $\sigma_1, \sigma_2, \ldots, \sigma_{63}$, is also tested.

Figure 5.8 shows the learning curves and the average dictionary sizes. The proposed scheme attains comparable performance to that of KNLMS with the naturally-tuned best scale-parameter that is tuned manually. The dictionary size of the scheme is smaller than that of KNLMS at the steady state.

This is further clarified by Fig. 5.9, which shows the contours of the estimandum $\psi_3$ (the original relief model) and its estimates in the end of the adaptation. One can see that the scheme builds an efficient dictionary by taking data points densely around the ocean floor (the blue part) and sparsely around the land (the red part). This is because the ocean floor area contains high frequency components, and can hence be well represented by means of small-scale kernels with a locally-dense dictionary. We emphasize here that the kernel selection and the dictionary construction are automatically achieved by the scheme due to the use of the two regularizers. In contrast, KNLMS builds a globally-dense dictionary with many data points even around the land although the land area only contains low frequency components. The results suggest that the scheme can represent local structures of the estimandum efficiently. Figure 5.10 depicts $\psi_3$ and its estimates graphically.

To visualize the difference among the estimates obtained by the algorithms, Fig. 5.11 plots the estimation errors $|\psi_3(u) - \varphi_n(u)|$ at each location $u \in [138.5, 139] \times [34.5, 35]$. It can be seen that the proposed scheme achieves low estimation errors over the entire region, whereas KNLMS suffers from high estimation errors around the ocean floor. This is because the scheme expresses the low frequency components (the land area) by means of large-scale kernels with a locally-sparse dictionary and can thus form a locally-dense dictionary to express the high frequency components (the ocean floor) by means of small-scale kernels.
Figure 5.8: Results for Experiment C.
Figure 5.9: The contours of $\psi_3$ and its estimates. The dictionary data $u_j, j \in \mathcal{J}_n$, are marked by diamonds.
Figure 5.10: The *estimandum* $\psi_3$ and its estimates in Experiment C.
5.5. NUMERICAL EXAMPLES

Figure 5.11: The estimation errors $|\psi_3(u) - \varphi_n(u)|$. 
Table 5.5: A list of optimization methods for nonsmooth convex functions over a Hilbert space $H$ and their adaptive extensions. Smooth and nonsmooth functions are denoted by $f$ and $g$, respectively. The $\iota_C$ is the indicator function for a convex subset $C \subset H$. $A$ is a bounded linear operator.

<table>
<thead>
<tr>
<th>Cost Function $(f$: smooth, $g$: nonsmooth)</th>
<th>Nonadaptive $\min_h \Theta(h)$</th>
<th>Adaptive $\min_h \Theta_n(h)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Theta(h) := g(h) + \iota_C$</td>
<td>PSM [148]</td>
<td>APSM [58, 60]</td>
</tr>
<tr>
<td>$\Theta(h) := f(h) + g(h)$</td>
<td>PFBS [87, 88]</td>
<td>APFBS [65]</td>
</tr>
<tr>
<td>$\Theta(h) := g_1(h) + g_2(h)$</td>
<td>DRS [90]</td>
<td>ADRS [66, 67]</td>
</tr>
<tr>
<td>$\Theta(h) := f(h) + g_1(h) + g_2(Ah)$</td>
<td>PDS [96]</td>
<td>APDS [97]</td>
</tr>
<tr>
<td>$\Theta(h) := f(h) + \sum_{i=1}^m g_i(h)$</td>
<td>GFBS [93]</td>
<td>AGFBS</td>
</tr>
</tbody>
</table>

5.6 Conclusion

In this chapter, we have presented an efficient online model-selection and learning scheme based on the multikernel adaptive filtering framework. The proposed scheme is a fully-adaptive and all-in-one method, and it alternately updates the dictionary and the coefficients with the AGFBS algorithm. The coefficient updating rule is derived with two block-sparsity promoting regularizers, which contributed to the model-selection and the dictionary refinement, respectively. The numerical examples have demonstrated that the proposed scheme achieved the appropriate model-selection and learning simultaneously without any prior knowledge for the kernel design. The results showed that, compared to SimpleMKL, the proposed scheme achieved a nearly optimal performance in dramatically shorter CPU-running-time with significantly smaller dictionary size. Remarkably, the scheme well expressed the local structures of the estimandum by allocating suitable models to each local part of the input space, and this yielded a considerable reduction of the dictionary size. The proposed scheme makes the frequency components contained in the estimandum visible, and it is therefore expected to be useful also for data analysis.

Appendix A: Convex Optimization Methods and Adaptive Extensions

Table 5.5 summarizes the convex optimization methods to minimize nonsmooth functions and their adaptive extensions: APSM, APFBS, ADRS, APDS, and AGFBS. In particular, APSM [58, 60, 127] is an adaptive extension of Polyak’s projected subgradient method [148], and it generates a sequence of vectors converging strongly to an asymptotically optimal point under mild (deterministic) conditions. It covers a variety of adaptive filtering algorithms such as the (projected/constrained) NLMS algorithm [3, 4], the APA [46, 47], and the adaptive parallel subgradient projection algorithms [108, 109], to name a few. APSM has been applied already to a variety of
signal processing applications [61, 62, 108, 117, 129, 169–175]. The success of APSM has encouraged adaptive extensions of other optimization methods for nonsmooth convex functions, which plays an important role in modern signal processing due, for instance, to the usefulness of the \( \ell_1 \) norm for obtaining sparse solutions and the total variation for obtaining better quality in image processing. Among the existing convex optimization methods, GFBS perfectly fits to the current context since it minimizes the sum of a smooth function and multiple nonsmooth (but proximable) functions. We therefore derive and use an adaptive extension of GFBS in Section 5.4.3.

Appendix B: Joint Estimation of Subspace and Parameter

For further clarification, we present a geometric interpretation of the proposed scheme in the sum space (see Fig. 5.12(a))

\[
\mathcal{H}_{\text{sum}} := \mathcal{H}_1 + \mathcal{H}_2 + \cdots + \mathcal{H}_Q := \left\{ \sum_{q \in Q} f_q \mid f_q \in \mathcal{H}_q \right\}
\]

of RKHSs \( \mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_Q \) that are associated respectively with \( \kappa_1, \kappa_2, \ldots, \kappa_Q \). An estimator in the sum space \( \mathcal{H}_{\text{sum}} \) is given in the following form (we drop the time index \( n \) for simplicity):

\[
\varphi(u) := \sum_{q \in Q} \varphi_q(u), \quad u \in \mathcal{U},
\]

where \( \varphi_q(\cdot) := \sum_{i=1}^r h_i^{q_1} \kappa_{q_1}(\cdot, u_i) \in \mathcal{M}_q := \text{span} \left\{ \kappa_{q_1}(\cdot, u_i) \right\}_{i=1}^r \subset \mathcal{H}_q \) with \( h_i^{q_1} \in \mathbb{R} \), \( u_i \in \mathcal{U} \), and \( r \in \mathbb{N}^* \) (see Fig. 5.12(b)). Owing to the sparsification, \( \varphi \) may have a shorter expansion:

\[
\varphi(u) := \sum_{q \in Q'} \varphi_q(u),
\]

where \( Q' := \{ q_1', q_2', \ldots, q_A' \} \subset Q \) is the index set of active kernels \( (A < Q) \) that are relevant to the estimation of \( \psi \). In this case, the estimator \( \varphi \) can be characterized as an element of the subspace \( \mathcal{M} := \mathcal{M}_q + \mathcal{M}_{q_2} + \cdots + \mathcal{M}_{q_A} \) of \( \mathcal{H}_{\text{sum}} \) (see Fig. 5.12(c)). The nonlinear estimation is thus accomplished through the joint estimation of the subspace \( \mathcal{M} \) and the parameters \( h_i^{q_1} \).

From the viewpoint of subspace-parameter joint estimation, the roles of \( \phi_n, \Omega_n^{(1)}, \) and \( \Omega_n^{(2)} \) in (5.8) are given as below.

1. Reducing the fidelity function \( \phi_n \) leads to the parameter estimation.

2. Reducing the sparsity-promoting regularizers \( \Omega_n^{(1)} \) and \( \Omega_n^{(2)} \) leads to the subspace estimation (as a result of joint sparsification of kernels and dictionary).

Indeed, reducing the function \( \phi_n \) grows only those coefficients for relevant kernels or relevant dictionary elements. Those for irrelevant ones will therefore vanish by suppressing the total cost \( \Theta_n \left( := \phi_n + \Omega_n^{(1)} + \Omega_n^{(2)} \right) \). Hence, the proposed scheme
can be viewed as selecting an appropriate subspace of $H_{\text{sum}}$ and, at the same time, finding good coefficients of the basis vectors of the subspace.
5.6. CONCLUSION

(a) The sum space $\mathcal{H}_{sum}$.

(b) The subspace $\mathcal{M}_q := \text{span} \{ \kappa_q(\cdot, \mathbf{u}_1), \kappa_q(\cdot, \mathbf{u}_2) \}$.

(c) The subspace $(\varphi \in \mathcal{M} := \mathcal{M}_3 + \mathcal{M}_4 \subset \mathcal{H}_{sum}$ for the active index set $Q' := \{3, 4\}$.

\[ \mathcal{H}_{sum} := \mathcal{H}_1 + \mathcal{H}_2 \]

\[ f = f_1 + f_2 \]

\[ \mathcal{M}_{q} \]

\[ \varphi_q(\cdot) = \sum_{i=1}^{n} h_i^{(q)}(\kappa_q(\cdot, \mathbf{u}_i)) \]

\[ 0 \rightarrow \mathbf{u}_1 \rightarrow \kappa_q(\cdot, \mathbf{u}_1) \rightarrow \mathcal{H}_{sum} \]

\[ \varphi = \varphi_3 + \varphi_4 \]
Appendix C: Metric Design

This appendix introduces the idea of the variable metric projection approach [117] to attain better convergence behaviors. When the ordinary Euclidean metric is used, the coefficients associated with large-scale kernels tend to be dominant since all (most) components of the kernelized input vectors corresponding to such kernels are nearly one. To guarantee equal opportunities for all the coefficients to grow, the metric needs to be designed so as to enhance the growth of the coefficients associated with small-scale kernels.

Our weight design for the $q$th kernel at time $n \in \mathbb{N}$ is given as

$$g_{q,n} := \begin{cases} 1, & \text{if } r_n = 1, \\ \sum_{\iota \in I_n, \ell \in L} \prod_{\ell \in L} \sqrt{1 - \exp\left(\frac{-\tau_{\ell,\iota,n}}{2\sigma_q^2}\right)}, & \text{otherwise}, \end{cases}$$

which is based on the integral of the associated Gaussian function over the interval $\tau_{\ell,\iota,n} > 0$. Here, $\tau_{\ell,\iota,n}$ is empirically derived and is given as follows:

$$\tau_{\ell,\iota,n} := \max_{j \in J_{n+1}^\ell} (u_{\ell,j}) - \min_{j \in J_{n+1}^\ell} (u_{\ell,j}),$$

$$\ell \in L := \{1, 2, \ldots, L\}, \; \iota \in \mathbb{N}^*, \; n \in \mathbb{N},$$

$$I_n := \{\iota \in \mathbb{N}^* | \min_{\ell \in L} \tau_{\ell,\iota,n} > \sigma_{\min} := \min_{q \in \mathbb{Q}} \sigma_q (> 0)\}.$$

The numerator of $\tau_{\ell,\iota,n}$ is the range of the dictionary data (for the $\ell$th axis). The denominator is introduced for emphasizing the small-scale kernels. Without the denominator, nearly equal weights could be assigned to the smallest-scale kernel and a medium-scale kernel that has $\sigma_q \approx \tau_{\ell,\iota,n}$, discouraging the smallest-scale kernel. Basically, for a kernel of large scale $\sigma_q$, it holds that $\exp\left(\frac{-\tau_{\ell,\iota,n}}{2\sigma_q^2}\right) \approx 1$, which makes $g_{q,n} \approx 0$. For a kernel of small scale $\sigma_q$, conversely, $\exp\left(\frac{-\tau_{\ell,\iota,n}}{2\sigma_q^2}\right)$ could be nearly zero for some $\iota \in I_n$, which makes $g_{q,n}$ be relatively large. As such, the weights for the small-scale kernels become fairly large, encouraging the growth of the coefficients for those kernels.

We now define the matrix

$$G_n := [g_n \; g_n \; \cdots \; g_n] \in \mathbb{R}^{Q \times r_{n+1}}$$

with $g_n := \frac{1}{r_{n+1} \sum_{q \in \mathbb{Q}} g_{q,n}} [g_{1,n}, \; g_{2,n}, \; \cdots, \; g_{Q,n}]^T \in \mathbb{R}^Q$. The variable-metric projection (the so-called $G_n$-projection)

$$P^{G_n}_{\Pi_{\mathcal{H}}}(H_n) := H_n - \frac{\varepsilon_n}{\text{trace} \left( K_n^T (G_n \otimes K_n) \right)} (G_n \otimes K_n)$$  \hspace{1cm} (5.17)

can be used instead of $P_{\Pi_{\mathcal{H}}}(H_n)$ defined in (5.15), where $\otimes$ is the Kronecker prod-
uct. The weights \( \nu_{q,n}^{(2)} \) for the row-block \( \ell_1 \) norm in (5.10) are modified into

\[
\nu_{q,n}^{(2)} := \left( g_{q,n} \| \chi_{q,n} \| \right)^{-1}, \quad q \in Q.
\] (5.18)

It should be remarked that \( G_n \) can be regarded as a metric, although it is a non-square matrix. Indeed, by vectorizing \( H_n \) and \( K_n \) by stacking the column vectors on top of each other and denoting a vectorized version of \( K_n \) by \( \text{vec}(K_n) \), the corresponding operation to \( G_n \otimes K_n \) is given by \( G_n \text{vec}(K_n) \). Here, the diagonal matrix \( \tilde{G}_n := \text{diag}(g_{1,n}^T, g_{2,n}^T, \ldots, g_{n,n}^T) \) with its diagonal entries given by the vector \([g_{1,n}^T, g_{2,n}^T, \ldots, g_{n,n}^T] \) is positive definite, and hence it defines a metric.
Chapter 6

General Conclusion

In this dissertation, we have introduced the three online learning scheme; e.g., the MC-NLMS algorithm, the KNLMS algorithm with CSKs, and the online model selection and learning scheme based on the multikernel adaptive filtering framework with the AGFBS method. The proposed schemes are based on the frameworks of the variable metric and the multiple kernels. We proved that the proposed schemes are effective algorithms to the online learning without a necessary and a priori information to design a metric and/or a kernel. Chapter 2 introduced the mathematical matters; i.e., proximity operator, Moreau envelope, variable-metric projection, and reproducing kernel. The main body of this dissertation has been constructed by three chapters: Chapters 3–5.

Chapters 3 has presented metric-combining normalized least mean square (MC-NLMS) which is based on iterative metric projections with a metric designed by combining multiple metric-matrices convexly in an adaptive manner. We showed the example of MC-NLMS, e.g., NPNLMS, for IPNLMS, and it was shown that the performance of NPNLMS is controllable with the combination coefficient as opposed to IPNLMS. We also have presented an application to an acoustic echo cancellation problem and showed the efficacy of the proposed scheme.

Chapters 4 has presented KNLMS with compactly supported kernel (CSK-NLMS) which is composed by three stages. In the first stage, the support of the input kernels was identified by two steps through its over-estimate based on the circumscribed $\ell_1$-ball. In the second stage, the dictionary was constructed based on the knowledge about the support identified in the first stage. In the last stage, the filter was updated based on the supported kernels in the first stage and dictionary in the second stage. The numerical examples showed that the proposed algorithm achieves significant complexity reduction.

Chapter 5 has presented the online model selection and learning scheme which is based on a multikernel adaptive filtering framework. The proposed scheme repeats the dictionary update and AGFBS-based coefficient update alternately. The coefficient updating rule was derived with two block-sparsity promoting regularizers, of which one is the model selection and learning the model without a priori information about the proper kernels. To improve the convergence speed, the filter updating equation applied the variable-metric projection framework. The propped scheme has expressed the local structures of the estimandum by means of suitable models and
this reduce the dictionary size reasonably. The outcomes of this chapter have proved the efficacy of the online nonlinear adaptive filter in real data model, since the online model selection and learning scheme have aimed to develop a fully-adaptive all-in-one learning scheme that jointly makes the model-selection and learning. The proposed scheme also has offered an insight into the frequency components of the \textit{estimandum}, and the proposed scheme can therefore be applied to data analysis. The fruit provided by this thesis will contribute to developments in a wide range of engineering applications including Internet of Things, in which we estimate and analyze the \textit{estimandum} from large-scale of data.
Publications Related to the Dissertation

Some ideas and figures have appeared previously in the following publications.

Journal Articles:


Conference Publications:


Bibliography


nlms adaptive filter based on the statistics of a room impulse response,” *IEEE

[134] B. Porat, *Digital processing of random signals: theory and methods*, Prentice-


1986.

[137] H. Wendland, “Piecewise polynomial, positive definite and compactly sup-
ported radial functions of minimal degree,” *Advances in Computational Math-


[139] M.D. Buhmann, “Radial functions on compact support,” *Proceedings of the

[140] M. S. Floater and A. Iske, “Multistep scattered data interpolation using
compactly supported radial basis functions,” *Journal of Computational and

[141] G.E. Fasshauer, “Solving differential equations with radial basis functions:
multilevel methods and smoothing,” *Advances in Computational Mathematics*,


[143] M.G. Genton, “Classes of kernels for machine learning: A statistics perspec-
2002.

[144] H.H. Zhang, M. Genton, and P. Liu, “Compactly supported radial basis func-

[145] B. Schölkopf and A.J. Smola, *Learning with Kernels: Support Vector Ma-
chines, Regularization, Optimization, and Beyond*, The MIT Press, 1st edition,


[147] M. Yukawa, “Adaptive learning with reproducing kernels,” in *RIMS
Kokyuroku 1980 (General Topics on applications of reproducing kernels)*. Res-
earch Institute for Mathematical Sciences, Kyoto University, Jan. 2016, pp.
1–15.


