A Study of Online Nonlinear Estimation Using Reproducing Kernels: ℓ_p-regularized Least Squares and Kernel Weight Design

February 2021

Kwangjin Jeong

A Thesis for the Degree of Ph.D. in Engineering

A Study of Online Nonlinear Estimation Using Reproducing Kernels: ℓ_p-regularized Least Squares and Kernel Weight Design

February 2021

Graduate School of Science and Technology
Keio University

Kwangjin Jeong

Acknowledgments

This thesis completes my studies in the doctoral course of Keio University since April 2018.

I would like to express my deepest gratitude to my advisor Associate Professor Masahiro Yukawa for his significant support and guidance to my entire study in the graduate school.

I would like also to express my deep gratitude to Professor Masaaki Ikehara of Keio university for his support in both aspects of my study and daily life, especially in my first year in the graduate school.

Furthermore, it is my pleasure to thank to all the current and formal students of Yukawa laboratory for the friendship. In particular, I feel thankful deeply for having meaningful time including profound discussions with my roommates Mr. Masaaki Takizawa and Ms. Ran Iwamoto.

My special thanks go to Ms. Ishii for her careful support on my overall activities related to my research.

I would like also deeply thank to Professors Masaaki Ikehara and Yukitoshi Sanada and Associate Professor Kei Kobayashi for serving as members of the examining committee of this thesis.

My works related to this thesis were supported by JSPS Grants-in-Aid (JP18J22032, JP18H01446).

Contents

List of Figures

List of Tables

Abstract

Reproducing kernels have been extensively exploited for online nonlinear estimation tasks such as time-series data prediction which are of significant importance in many fields of science and engineering. However, two major challenges for online learning with kernels have also been recognized. First, the size of the dictionary grows linearly with the amount of observed data. Second, it is practically impossible to choose an appropriate kernel in an online situation where data are observed sequentially. Multikernel adaptive filtering has been proposed to solve the second issue, although a proper kernel weight design is essential to exploit its full potential.

In this thesis, the insights related to the above two issues are presented. The thesis consists of the following five chapters. Chapter 1 introduces the background and motivation of the thesis. Chapter 2 provides mathematical preliminaries on sparse optimization and online learning with kernels. The related works are introduced as well. In Chapter 3, sparse optimization by ℓ_p-norm $(0 < p < 1)$ regularization is considered as a fundamental study for sparsification of the dictionary. Given an ℓ_p-regularized least squares problem, the behavior of the critical points with a variation of the regularization parameter is investigated. In addition, it is proved that there exists a continuous, piecewise smooth path of critical points connecting the origin and the sparsest least square solution. Chapter 4 proposes a kernel weight design for multikernel adaptive filtering and shows that the proposed design is optimal in a certain sense. The proposed weight design is derived by equalizing the convergence speeds of kernel-wise coefficient errors. The derived weights can be calculated recursively using the observed data and the recursive kernel weighting is implemented to a multikernel adaptive filtering algorithm using metric projection. Numerical results show the superiority of the proposed algorithm over both of the carefully tuned preset weights and the machine learning algorithm based on multiple kernel learning, in the senses of the accuracy and the dictionary size. Chapter 5 summarizes the previous chapters and gives concluding remarks of this thesis as well as the future prospects for this study.

Chapter 1

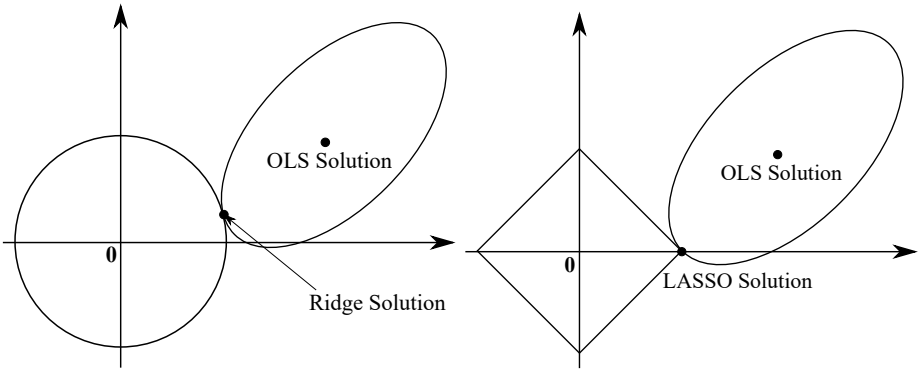
General Introduction

The study of my Ph. D. course has been devoted for development of an online learning algorithm exploiting reproducing kernels, with a appropriate technique for dictionary sparsification. This thesis presents (i) a theoretical study on nonconvex regularizaiton for sparse optimization and (ii) a multikernel adaptive filtering algorithm with an automatic tuning of kernel weights. This chapter briefly introduces the background of sparse optimization and kernel adaptive filtering.

1.1 Sparse Optimization

In signal processing, the way how to transform a signal, also called as *encoding–decoding* or *analysis–synthesis*, has always been a fundamental topic of study. Mallat and Zhang [1] have focused on the use of a large dictionary which consists of atoms with various waveforms including Fourier and wavelets. As an approach for flexible signal decomposition, a greedy algorithm called the matching pursuit (MP) has been proposed in [1]. Donoho and Johnstone [2] have proposed a wavelet selection technique for reconstruction of a noise-corrupted signal. The soft thresholding operator, which has been proposed in [2], has been also used for signal denoising in [3]. Both studies of [1] and [2] have consequently led to a signal representation by a small number of atoms from a redundant dictionary. Such tasks can be generalized to finding a *sparse* solution of an underdetermined least squares problem, which has more variables than equations, and hence there exist infinitely many solutions. From a practical viewpoint, sparse decomposition of a signal leads to small computational costs for processing tasks of the decomposed signal. Therefore, finding a sparse representation eventually improves efficiency of signal processing even though using a redundant dictionary may be memory-inefficient.

The ℓ_1-norm is the absolute sum of the coefficients associated with the selected atoms. Chen *et al.* [4] have proposed linear programming approaches which solve a least squares problem regularized by the ℓ_1-norm to obtain a sparse solution of signal decomposition. It has also been shown in [4] that the use of soft thresholding corresponds to solving a particular case of an ℓ_1-regularized least squares problem. In the statistics community, a similar concept has been considered in an overdetermined system with a unique ordinary



(a) Ridge regression (b) LASSO

Figure 1.1: The solutions of ridge regression and LASSO.

least squares (OLS) solution. Tibshirani [5] has proposed a variable selection technique named least absolute shrinkage and selection operator (LASSO), to achieve a fine bias-variance tradeoff in regression. It is also shown that LASSO coincides with solving a least squares problem with an ℓ_1-norm constraint, and LASSO achieves a sparser model than the ridge regression, which is constrained by the squared ℓ_2-norm. See Figure 1.1 for comparison between solutions of ridge regression and LASSO.

Sparsity of a vector is measured by the ℓ_0-norm, which counts the number of its nonzero entries. Since the ℓ_0-norm is discontinuous, sparse optimization based on the ℓ_0-norm yields a combinatorial optimization problem, which is NP-hard. The use of greedy algorithms to deal with the problem in affordable computational costs. The orthogonal matching pursuit (OMP) [6], an improved version of the MP, has been a celebrated example of such algorithms, since its potential ability to find a sparse solution has been reported in [7]. The OMP sets the initial coefficient vector at the origin, and then updates its step-by-step solution such that a residual error becomes zero at the solution. Thanks to simplicity and theoretical performance guarantee of the OMP, many improvements have been proposed. See [8, 9] for the ameliorated algorithms.

The ℓ_1-norm can be regarded as a convex and continuous relaxation of an ℓ_0-norm, which is nonconvex and discontinuous. See Figure 1.2 to compare the ℓ_0-norm and the ℓ_1-norm. Thanks to the convexity of the ℓ_1-norm, an ℓ_1-norm minimization problem with a zero-error constraint can be solved by linear programming. In addition, an ℓ_1-norm minimization problem has been compared with an ℓ_0-norm minimization problem profoundly. Donoho and Huo [10] have studied conditions for both problems to have unique solution. Candès and Tao [11] have proposed the restricted isometry property (RIP), which provides sufficient conditions that a solution of an ℓ_1-norm minimization problem is the same with that of an ℓ_0-norm minimization problem. It has been also shown that some random matrices satisfies a good RIP and this leads to the use of sparse optimization with using the ℓ_1-norm not only in denoising but also in general signal recovery tasks and compressive sensing [12–14].

Sparse optimization using the ℓ_1-norm has been also considered from a geometrical viewpoint. It has been reported in [5] that the LASSO solution moves

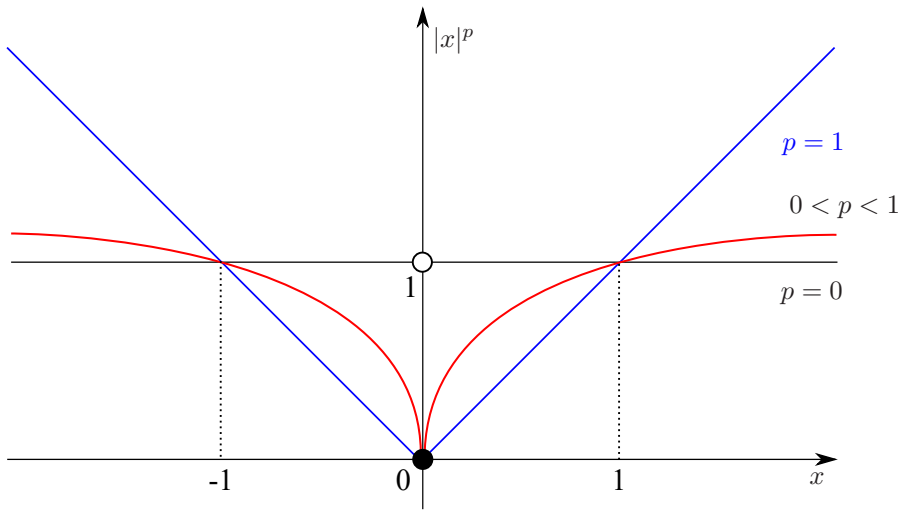


Figure 1.2: The ℓ_0-norm, the ℓ_1-norm, and the ℓ_p-norm ($0 < p < 1$).

when a hyperparameter for an ℓ_1-norm constraint changes, and the solution draws a trajectory called a solution path. Osborne *et al.* [15] have shown that the solution path is piecewise straight and proposed the homotopy algorithm which obtains the entire path. Efron *et al.* [16] have proposed another algorithm called least angle regression (LARS). Although LARS is a greedy algorithm which starts from the origin and increases the number of nonzero coefficients, a modified version which allows nullifying a nonzero coefficient is equivalent to the homotopy algorithm. Donoho and Tsaig [17] have reported that the LARS and the homotopy algorithm, which consider an overdetermined system as LASSO does, can be used to solve an ℓ_1-regularized least squares problem which considers an underdetermined system. In [17], the OMP has also been compared with the LARS, focused on the similarity in their greedy processes.

While the ℓ_1-norm has been popular, the ℓ_p-(quasi-)norm, where $0 < p < 1$, has also been considered as an attractive substitute of the ℓ_0-norm for sparse optimization because it is a closer continuous approximation of the ℓ_0-norm than the ℓ_1-norm. Chartrand and Steneva [18] have presented an RIP for the ℓ_p-norm and Foucart and Lai [19] have shown another sufficient condition for achieving the sparsest solution of an underdetermined system by ℓ_p-regularization. Xu *et al.* [20] have focused on the case of $p = 0.5$ and solved an $\ell_{0.5}$-regularized least squares problem using a thresholding operator derived from cubic equations. See the related algorithms in [21, 22]. Yukawa and Amari [23] have extended LARS to the ℓ_p case. They have observed that the solution path under ℓ_p-regularization is discontinuous, while the path consisting of the critical points is continuous and piecewise smooth. (See Figure 1.3.) It is shown that the breakpoints of the piecewise smooth path correspond to the step-by-step solution of the OMP, and this is a more direct link than that shown in [17].

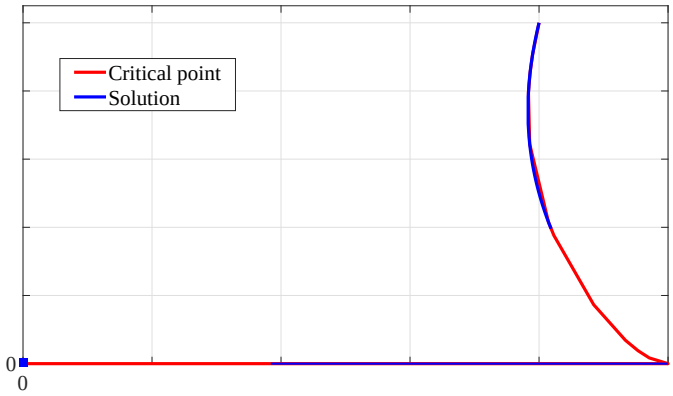


Figure 1.3: The solution path and the critical-point path under an ℓ_p-regularization.

1.2 Kernel Adaptive Filtering

The core idea of kernel methods is to replace an inner product by a positive definite kernel function. This is eventually equivalent to map the data in a parametric space onto the higher-dimensional reproducing kernel Hilbert space (RKHS) associated with the positive definite kernel function. In addition, the representer theorem guarantees that the minimizer of a regularized loss function can be represented by a linear combination of a finite number of the basis functions of the RKHS. By the above properties, calculation and representation of an estimator in a possibly infinite-dimensional RKHS can be done by a finite number of calculations in a parametric space.

Enjoying such an advantage, kernel methods have been studied not only in batch settings, but also in online settings. Adaptive filtering [24, 25] is a representative online framework for linear estimation in the signal processing community. *Kernel Adaptive Filtering* [26] has emerged from direct application of an RKHS to the existing adaptive filtering algorithms. Engel *et al.* [27] have proposed an extension of the recursive least squares (RLS) algorithm. Kernelized versions of the affine projection algorithm (APA) and the least mean square (LMS) algorithm have been proposed by Liu *et al.* [28, 29]. Richard *et al.* [30] have also proposed the kernelized APA and LMS algorithms independently from [28, 29]. The main differences between the algorithms in [28, 29] and [30] are that (i) the formers explicitly consider calculations in an RKHS while the latters only use kernelized input in the parametric space and (ii) the latters are normalized algorithms. For simplicity, hereafter the two kernelized LMS algorithms in [29] and [30] are called the KLMS algorithm and the KNLMS algorithm, respectively.

Not only for kernel adaptive filtering, but most online kernel methods face two major challenges. One is undesirable growth of the dictionary, which contains the basis functions representing the estimator. Exploiting an infinite-dimensional RKHS inherently allows an excessively large basis. For online learning of large datasets, in particular, this implies the dictionary size and

the computational costs also increase linearly to the number of observed data. Wang *et al.* [31] have called such an undesired growth of dictionary as *the curse of kernelization*. The other is difficulty of kernel selection, which significantly affects on the performance of kernel methods including kernel adaptive filtering. It is practically impossible to choose a proper kernel before a sufficient number of observations. Kernel methods have been studied from various viewpoints to tackle the above challenges.

For kernel adaptive filtering, *sparsification* of the dictionary using certain criteria has been considered even in earlier works as [27]. The sparsification criteria are based on one or several measures such as a novelty [32], the approximate linear dependence (ALD) [27], the coherence [30], and the surprise [33]. The ALD measures a distance in an RKHS between the newly observed data and its projection onto the subspace spanned by all the basis functions in the dictionary. Van Vaerenbergh *et al.* [34] have found that, from the Bayesian viewpoint, the ALD can be regarded as the prediction variance which is used in the online Gaussian process regression [35]. The coherence has been already considered in greedy algorithms for sparse optimization [1, 4, 6, 36]. The use of the coherence for kernel adaptive filtering has been proposed in [30] and the quantized KLMS (QKLMS) [37] employs essentially the same criterion. The surprise is an information-theoretical measure which generalizes the novelty proposed by Platt [32] and the above other measures. Liu *et al.* [33] has shown that both of the ALD and the coherence are particular cases of the surprise which can be derived by assuming the Gaussian input distribution and unknown desired signal. Soft thresholding operator has also been used for sparse kernel adaptive filtering in [38].

To circumvent the difficulty of kernel selection, a simultaneous use of multiple kernels has been studied. Inspired by the multiple kernel learning (MKL) [39–44] which has emerged in the machine learning community, many kernel adaptive filtering using multiple kernels have been proposed [45, 46]. Yukawa has proposed another way of simultaneously using multiple kernels, called *multikernel adaptive filtering* [47, 48]. While the other methods consider a convex combination of kernels or an ensemble of a number of kernel adaptive filters, multikernel adaptive filtering takes different convex combinations so that different kinds of basis functions are included in the dictionary. Such a use of multiple kernel allows multikernel adaptive filtering a higher degree of freedom in estimation than the other methods [49]. Effectiveness of multikernel adaptive filtering has been shown in several applications including communications [50–52] and sensor network [53].

1.3 This Study

The main topics of this thesis are twofold.

- One is on critical-point paths of an ℓ_p-regularized least squares problem in an underdetermined system. Although the study by Yukawa and Amari [23] have presented a LARS extension in an overdetermined system and observed a continuous path of critical points, the existence of a path connecting the origin and the ordinary least squares (OLS) solution has not been proved. In Chapter 3, it is proved that (i) there exists a continuous critical-point path connecting the origin and the (unique) OLS

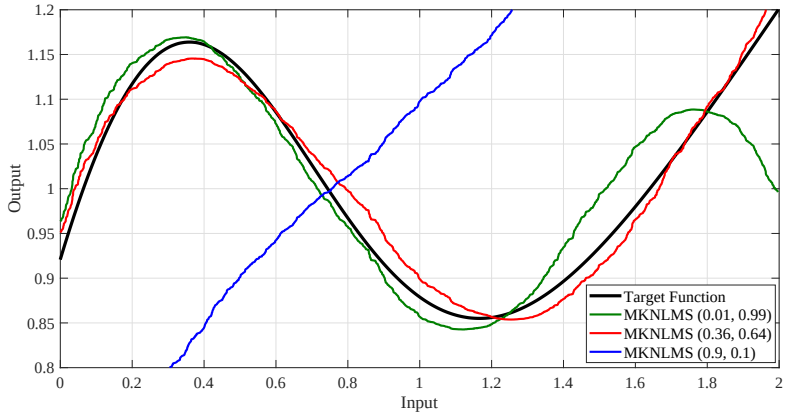


Figure 1.4: Effect of kernel weights on multikernel adaptive filtering.

solution in an overdetermined system, and (ii) there exists a continuous critical-point path connecting the origin and the sparsest OLS solution in an underdetermined system.

- The other is on a kernel weight design for multikernel adaptive filtering. The performnace of multikernel adaptive filtering heavily depends on the choice of kernel weights, as shown by Figure 1.3, whereas the weights are learned in MKL. In Figure 1.3, the target function is estimated using two kernels: one is linear and the other is Gaussian. In the legend, MKNLMS (a, b) means that the kernel weights for the linear kernel and the Gaussian kernel are a and $b = 1 - a$, respectively. The proposed weight design in Chapter 4 is derived by equalizing the convergence speeds of kernel-wise coefficient errors. The derived weights can be calculated recursively using the observed data and the recursive kernel weighting is implemented to a multikernel adaptive filtering algorithm using metric projection. Numerical results show the superiority of the proposed algorithm over both of the carefully tuned preset weights and an online regression algorithm based on MKL, in the senses of the accuracy and the dictionary size.

The remainder of this thesis is organized as follows. Chapter 2 provides mathematical preliminaries on sparse optimization and online learning with kernels. The related works are introduced as well. In Chapter 3, sparse optimization by ℓ_p-regularization is considered as a fundamental study for sparsification of the dictionary. Chapter 4 proposes a kernel weight design for multikernel adaptive filtering and shows that the proposed design is optimal in a certain sense. Chapter 5 summarizes the previous chapters and gives concluding remarks of this thesis as well as the future prospects for this study.

Chapter 2

Preliminaries

The main results of this thesis are derived from discussions dealing with various mathematical concepts. This chapter presents preliminary materials for the discussions in the main chapters.

2.1 Notations

Let $\mathbb{R}$ and $\mathbb{N}$ denote the sets of all real numbers and all nonnegative integers, respectively. $\mathbb{R}^L$ and $\mathbb{R}^{L_1 \times L_2}$ are the L-dimensional Euclidean space and the set of all $L_1 \times L_2$ matrices, respectively. In addition, define $\mathbb{R}^L_{++} := \{\boldsymbol{x} \in \mathbb{R}^L : x_i > 0,\ \forall i\}$. The superscript T stands for the transposition of a vector or a matrix. $\boldsymbol{I}_r$ and $\boldsymbol{O}$ are the $r \times r$ identity matrix and the zero matrix of appropriate size, respectively. $\mathbf{1}_r$ and $\mathbf{0}$ are the vector of r ones and the zero vector of appropriate size, respectively. For a scalar a, $|a|$ stands for the absolute value of a. For a set $\mathcal{I}$, $|\mathcal{I}|$ is the cardinality, or size, of $\mathcal{I}$. For a vector $\boldsymbol{x} \in \mathbb{L}$, the ℓ_p-norm of $\boldsymbol{x}$ for $0 < p < \infty$ is defined as follows:

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{L} |x_i|^p \right)^{1/p}, \tag{2.1}$$

while the ℓ_0-norm $\|\boldsymbol{x}\|_0$ is the number of nonzero entries of $\boldsymbol{x}$. The range space and the null space of a matrix are denoted by $\mathcal{R}(\cdot)$ and $\mathcal{K}(\cdot)$, respectively.

2.2 ℓ_p-regularized Least Squares Problem

Consider the linear system model

$$\boldsymbol{d} := \boldsymbol{U}^{\mathsf{T}} \boldsymbol{h}^* + \boldsymbol{\epsilon} \in \mathbb{R}^l, \tag{2.2}$$

where the sensing matrix $\boldsymbol{U} := [\boldsymbol{u}_1 \cdots \boldsymbol{u}_l] \in \mathbb{R}^{m \times l}$ and the measurement vector $\boldsymbol{d}$ are given, and $\boldsymbol{h}^* \in \mathbb{R}^m$ and the noise vector $\boldsymbol{\epsilon} \in \mathbb{R}^l$ are unknown. Assume that $\boldsymbol{U}$ has full rank; *i.e.*, $\mathrm{Rank}(\boldsymbol{U}) = \min\{m, l\}$. The linear system $\boldsymbol{U}^{\mathsf{T}} \boldsymbol{h} = \boldsymbol{d}$, $\boldsymbol{d} \in \mathbb{R}^l$, has infinitely many OLS solutions if the system is underdetermined ($m > l$), whereas it has a unique one if the system is overdetermined ($m \leq l$).

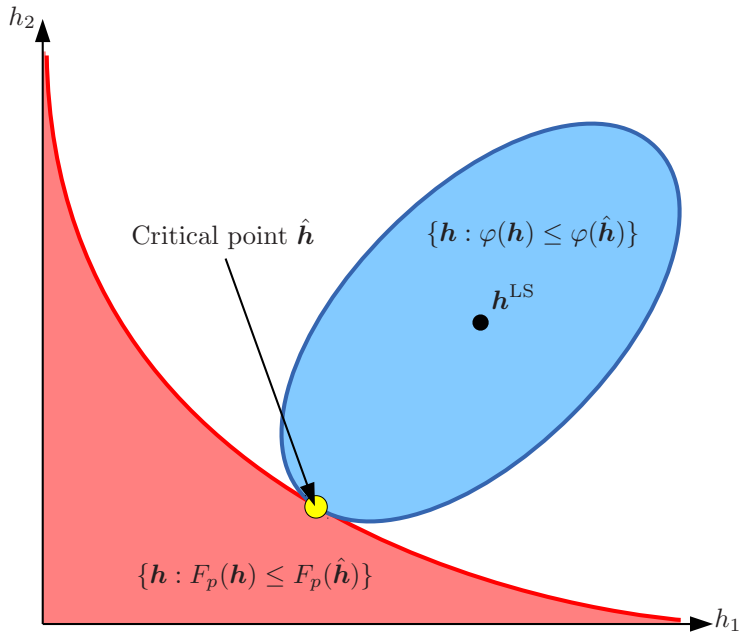


Figure 2.1: A geometric interpretation of critical point.

A problem of finding a sparsest OLS solution is formulated as follows:

$$\text{find } \boldsymbol{h}_{\text{sparsest}} \in \underset{\boldsymbol{h}\in\mathbb{R}^m}{\operatorname{argmin}} \|\boldsymbol{h}\|_0 \text{ s.t. } \boldsymbol{d} = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{h}. \tag{2.3}$$

The ℓ_0-norm involves combinatorial problem and hence solving (2.3) requires unaffordable computational costs for large systems. Consider the following ℓ_p-regularized least squares problem for $p > 0$:

$$(\mathcal{L}_\lambda^p)\text{: find } \hat{\boldsymbol{h}}_\lambda \in \underset{\boldsymbol{h}\in\mathbb{R}^m}{\operatorname{argmin}} f_\lambda(\boldsymbol{h}) := \varphi(\boldsymbol{h}) + \lambda F_p(\boldsymbol{h}), \tag{2.4}$$

where $\varphi(\boldsymbol{h}) := \frac{1}{2}\|\boldsymbol{U}^{\mathsf{T}}\boldsymbol{h} - \boldsymbol{d}\|_2^2$, $F_p(\boldsymbol{h}) := \frac{1}{p}\|\boldsymbol{h}\|_p^p = \frac{1}{p}\sum_{i=1}^m |h_i|^p$, and $\lambda \geq 0$ is the regularization parameter.

It has been shown in [23] that the set of global minima (or even local minima) for every possible $\lambda \geq 0$ is discontinuous. In this thesis, the critical point is considered as a generalized concept of the stationary points. Indeed, F_p is differentiable only at those points which have no zero component. One cannot therefore define a stationary point of f_λ in the entire space, but instead can define a critical point by considering the partial derivatives of f_λ in terms of only nonzero components. Prior to the definition of a critical point, define the *support* of a vector as follows.

Definition 1 (Support)**.** *The support of a vector $\boldsymbol{h}$ is defined as supp$(\boldsymbol{h}) := \{i \in \{1, 2, \cdots, n\} : h_i \neq 0\}$, i.e., it is the set of the active indices of $\boldsymbol{h}$.*

For $\mathcal{I} = \{i_1, i_2, \cdots, i_m\} :=$ supp$(\boldsymbol{h})$ with its cardinality $|\mathcal{I}| = m$, denote by $\nabla_{\mathcal{I}}$ the gradient with respect to the nonzero components; *i.e.*, for any function

$f(\boldsymbol{h}) : \mathbb{R}^m \mapsto \mathbb{R}$,

$$\nabla_{\mathcal{I}} f(\boldsymbol{h}) := \left[\frac{\partial f}{\partial h_{i_1}} \ \frac{\partial f}{\partial h_{i_2}} \ \cdots \ \frac{\partial f}{\partial h_{i_m}} \right]^{\mathsf{T}}. \tag{2.5}$$

Definition 2 (Critical point)**.** *A vector $\boldsymbol{h}$ is said to be a critical point of f_λ if it satisfies the following condition:*

$$\nabla_{\mathcal{I}} \varphi(\boldsymbol{h}) + \lambda \nabla_{\mathcal{I}} F_p(\boldsymbol{h}) = \boldsymbol{0}. \tag{2.6}$$

Definition 2 implies that the functions φ and F_p share the same tangent plane at a critical point. See Figure 2.1. Hereafter, the first order condition (2.6) is referred to as the critical point condition.

2.3 Multikernel Adaptive Filtering with Weighted Kernels

2.3.1 Multikernel Adaptive Filtering

Kernel adaptive filtering considers the following nonlinear system model:

$$d_n = \psi(\boldsymbol{u}_n) + \nu_n \in \mathbb{R}, \ n \in \mathbb{N}, \tag{2.7}$$

where $\boldsymbol{u}_n \in \mathbb{R}^L$ is the input vector, d_n is the corresponding output generated by the unknown function ψ of interest, and ν_n is the zero-mean additive noise which is assumed independent of $\boldsymbol{u}_n$. Multikernel adaptive filtering simultaneously exploits multiple kernel functions to estimate the target function ψ given the observation $\boldsymbol{u}_n$ and d_n up to each time instant n. Suppose that Q kernel functions $\kappa_q(\cdot,\cdot), \ q = 1, 2, \cdots, Q$, are used. The dictionary for the qth kernel is defined as $\mathcal{D}_{q,n} := \{\kappa_q(\boldsymbol{x}_1^{(q)}, \cdot), \cdots, \kappa_q(\boldsymbol{x}_{r_{q,n}}^{(q)}, \cdot)\}$, where $r_{q,n}$ is the cardinality of $\mathcal{D}_{q,n}$ and the vectors $\boldsymbol{x}_j^{(q)} \in \mathbb{R}^L, \ j = 1, 2, \cdots, r_{q,n}$, determine the dictionary elements. The instantaneous estimate is then obtained as

$$\hat{\varphi}_n(\boldsymbol{u}_n) = \sum_{q=1}^{Q} \sum_{j=1}^{r_{q,n}} h_{n,j}^{(q)} \kappa_q(\boldsymbol{x}_j^{(q)}, \boldsymbol{u}_n) = \sum_{q=1}^{Q} \boldsymbol{k}_n^{(q)\mathsf{T}} \boldsymbol{h}_n^{(q)}, \tag{2.8}$$

where

$$\begin{aligned} \boldsymbol{k}_n^{(q)} &= \left[\kappa_q\big(\boldsymbol{x}_1^{(q)}, \boldsymbol{u}_n\big), \cdots, \kappa_q\big(\boldsymbol{x}_{r_{q,n}}^{(q)}, \boldsymbol{u}_n\big)\right]^{\mathsf{T}} \in \mathbb{R}^{r_{q,n}}, \\ \boldsymbol{h}_n^{(q)} &= \left[h_{n,1}^{(q)}, \cdots, h_{n,r_{q,n}}^{(q)}\right]^{\mathsf{T}} \in \mathbb{R}^{r_{q,n}}, \end{aligned}$$

for all $q = 1, 2, \cdots, Q$. Concatenating the vectors as

$$\begin{aligned} \boldsymbol{k}_n &= \left[\boldsymbol{k}_n^{(1)\mathsf{T}}, \cdots, \boldsymbol{k}_n^{(Q)\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{R}^{r_n}, \\ \boldsymbol{h}_n &= \left[\boldsymbol{h}_n^{(1)\mathsf{T}}, \cdots, \boldsymbol{h}_n^{(Q)\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{R}^{r_n}, \end{aligned}$$

where $r_n = \sum_{q=1}^{Q} r_{q,n}$, reduces (2.8) to

$$\hat{\varphi}_n(\boldsymbol{u}_n) = \boldsymbol{k}_n^{\mathsf{T}} \boldsymbol{h}_n. \tag{2.9}$$

The nonlinear estimation of ψ then reduces to linear estimation of the Euclidean vector $\boldsymbol{h}^*$, which is a minimizer of some error function, given the *kernelized* input vector $\boldsymbol{k}_n,\ n \in \mathbb{N}$. For $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^L$, the celebrated examples of positive definite reproducing kernel are given as below.

Normalized Gaussian kernel

$$\kappa_{\mathrm{G}}(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{(\sqrt{2\pi}\sigma)^L} \exp\left(- \frac{\|\boldsymbol{u} - \boldsymbol{v}\|_2^2}{2\sigma^2} \right),\ \sigma > 0. \tag{2.10}$$

Linear kernel

$$\kappa_{\mathrm{L}}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^{\mathsf{T}} \boldsymbol{v} + c, \quad c \in \mathbb{R}. \tag{2.11}$$

Polynomial kernel

$$\kappa_{\mathrm{P}}(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^{\mathsf{T}} \boldsymbol{v} + c)^a, \quad c \in \mathbb{R},\ a \in \mathbb{N}. \tag{2.12}$$

In the multikernel adaptive filtering algorithms, the dictionaries $\mathcal{D}_{q,n}$ for the Gaussian kernel are constructed in online fashion, as in the case of the other kernel adaptive filtering algorithms, whereas the dictionary for the linear kernel is fixed. This is because the RKHS for the linear kernel has finite dimension and its basis is known a priori whereas that for the Gaussian kernel has infinite dimension and an appropriate dictionary is unknown in practice. Some criteria for dictionary construction can be found in the literature [27, 30, 32, 33, 35]. A popular criterion for the Gaussian kernel is the coherence criterion [30], which tests whether the maximal entry of the kernelized input vector $\boldsymbol{k}_n^{(q)}$ is below a prespecified threshold, say $\delta > 0$. If the maximal entry is below δ, then the input is considered to be novel (since it is incoherent to any existing elements of the dictionary) and is thus added to the dictionary. The resulting dictionary consists of incoherent elements, meaning that the centers of the Gaussian functions in the dictionary are apart from each other at the prespecified level.

The MKNLMS algorithm [47,48] is a multikernel adaptive filtering algorithm of which the error function is the mean squared error $\mathrm{MSE}(\boldsymbol{h}) := E\big[(\boldsymbol{k}_n^{\mathsf{T}} \boldsymbol{h} - d_n)^2\big]$, where $\boldsymbol{h} \in \mathbb{R}^{r_n}$ and $E[\cdot]$ stands for the expectation with respect to the input $\boldsymbol{u}_n$ and the noise ν_n. Without a manipulation of dictionaries, each update of the MKNLMS algorithm can be written as follows:

$$\boldsymbol{h}_{n+1} = \boldsymbol{h}_n - \mu \frac{\boldsymbol{k}_n^{\mathsf{T}} \boldsymbol{h}_n - d_n}{\|\boldsymbol{k}_n\|_2^2} \boldsymbol{k}_n, \tag{2.13}$$

where $\mu \in (0, 2)$ is the step size. When $Q = 1$, the MKNLMS algorithm reduces to the KNLMS algorithm [30].

2.3.2 Relation to MKL

MKL has extensively been studied in machine learning community. In a similar form to (2.8), the instantaneous estimate by MKL can be written as follows:

$$\begin{aligned} \hat{\varphi}_n(\boldsymbol{u}_n) &= \sum_{j=1}^{r_n} h_{n,j} \left(\sum_{q=1}^{Q} \theta_{q,n} \kappa_q(\boldsymbol{x}_j, \boldsymbol{u}_n) \right) \\ &= \sum_{j=1}^{r_n} h_{n,j} \kappa_n(\boldsymbol{x}_j, \boldsymbol{u}_n), \end{aligned} \tag{2.14}$$

Algorithm 1 OMKR (hedge)

0) $\mathcal{D}_0 = \emptyset,\ \boldsymbol{h}_0^{(q)} = [\,],\ \theta_{q,1} = \dfrac{1}{Q},\ q = 1, 2, \cdots, Q,\ \mu > 0$: step size, $\beta \in (0, 1)$: discounting parameter.

1) Prediction: $\hat{\varphi}_n(\boldsymbol{u}_n) = \sum_{q=1}^{Q} \theta_{q,n} \boldsymbol{k}_n^{(q)\mathsf{T}} \boldsymbol{h}_n^{(q)}$.

2) $\epsilon_{q,n} = d_n - \boldsymbol{k}_n^{(q)\mathsf{T}} \boldsymbol{h}_n^{(q)}$.

3) $\boldsymbol{h}_{n+1}^{(q)} = \begin{bmatrix} \boldsymbol{h}_n^{(q)} \\ \mu\epsilon_{q,n} \end{bmatrix},\ \tilde{\theta}_{q,n+1} = \theta_{q,n} \beta^{\epsilon_{q,n}^2}$.

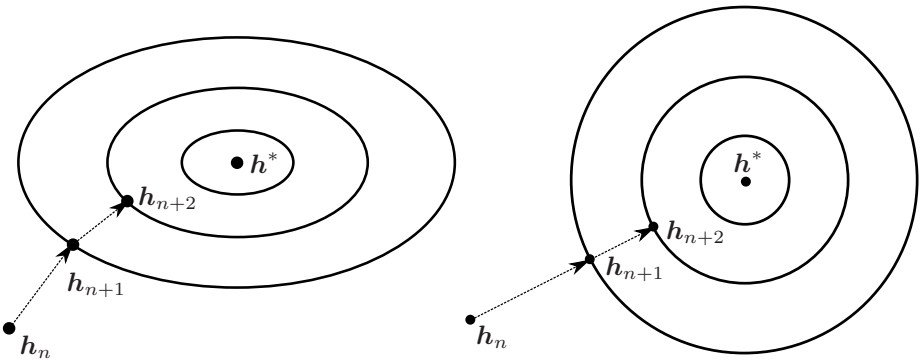
4) $\theta_{q,n+1} = \dfrac{\tilde{\theta}_{q,n+1}}{\sum_{p=1}^{Q} \tilde{\theta}_{p,n+1}}$.

5) $\mathcal{D}_{q,n+1} = \mathcal{D}_{q,n} \cup \{\kappa_q(\boldsymbol{u}_n, \cdot)\}$.

where $\theta_{q,n} \geq 0,\ q = 1, 2, \cdots, Q$, are the kernel weights such that $\sum_{q=1}^{Q} \theta_{q,n} = 1$. Most MKL algorithms, such as those in [42–44], first update $\theta_{q,n}$ to construct a convex combination of multiple kernels, and then update the coefficients $h_{n,j}$. One can see from (2.14) that MKL uses the appropriately designed single kernel κ_n, and the degree of freedom in MKL is $Q + r_n$. On the other hand, as seen from (2.8), the degree of freedom in multikernel adaptive filtering is $Q \times r_n$, which is much larger than $Q + r_n$ in practical situations. Thanks to the higher degree of freedom, multikernel adaptive filtering enjoys a more compact representation than MKL by excluding those kernels which are unnecessary for estimation. Moreover, multikernel adaptive filtering can potentially extract the local structures of the target containing multiple components, since different kernels can be used to express different components. In Section 5, the proposed method is experimentally compared with the OMKR algorithm [43, 44]. The OMKR algorithm has been developed as an MKL algorithm for online regression, while most MKL algorithms deal with classification tasks under a batch setting. It is therefore a particular example of MKL that can be directly compared with multikernel adaptive filtering in numerical examples. Algorithm 1 shows the OMKR algorithm with a discounting parameter, which controls the kernel weights according to the corresponding kernel-wise prediction errors.

2.3.3 Imposing Kernel Weights

A severe degradation in performance of multikernel adaptive filtering may occur when there exists some imbalance among the kernels. Such imbalance can be alleviated by introducing kernel weights. Given positive definite kernels $\kappa_1, \kappa_2, \cdots, \kappa_Q$, imposing weights $w_q > 0,\ q = 1, 2, \cdots, Q$, on the kernels changes the geometry of the original RKHS [48]. Geometry of the space is important for multikernel adaptive filtering, of which the update is based on the projection onto a zero-instantaneous-error hyperplane. The use of unweighted kernels, as in Figure 2.2(a), leaves the possible imbalance among the kernels and may result in slow convergence of the errors which are associated with some of the kernels. Figure 2.2(b) shows the change of geometry by the appropriate kernel weights such that the imbalance is alleviated. Applying the kernel weights to the update



(a) Without weights (b) With appropriate weights

Figure 2.2: Change of geometry by kernel weights.

(2.13) yields

$$\boldsymbol{g}_{n+1} = \boldsymbol{g}_n - \mu \frac{\boldsymbol{k}_n^{\mathsf{T}} \boldsymbol{W} \boldsymbol{g}_n - d_n}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2} \boldsymbol{W}\boldsymbol{k}_n, \tag{2.15}$$

where $\boldsymbol{g}_n := \boldsymbol{W}^{-1}\boldsymbol{h}_n$ and the weight matrix $\boldsymbol{W}$ is given as

$$\boldsymbol{W} := \begin{bmatrix} w_1 \boldsymbol{I}_{r_{1,n}} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & w_2 \boldsymbol{I}_{r_{2,n}} & \cdots & \boldsymbol{O} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & w_Q \boldsymbol{I}_{r_{Q,n}} \end{bmatrix}.$$

The choice of $\boldsymbol{W}$ governs the transient behavior of the coefficient error $\boldsymbol{z}_n := \boldsymbol{g}_n - \boldsymbol{g}^*$, where $\boldsymbol{g}^* := \boldsymbol{W}^{-1}\boldsymbol{h}^*$. Specifically, relative convergence rates among the kernel-wise coefficient errors, which coincide with the subvectors of $\boldsymbol{z}_n$, are determined by the ratios among the weights $w_1, w_2, \cdots, w_Q$, as well as the input autocorrelation matrix. An appropriate choice of kernel weights leads to uniform convergence rates of the kernel-wise coefficient errors. The kernel weights, however, need to be chosen carefully because a wrong choice may result in a worse performance than the use of unweighted kernels and even than the use of a single kernel.

Chapter 3

Critical-Point Paths under ℓ_p-regularization $(0 < p < 1)$

3.1 Introduction

Regularization by the ℓ_p-norm $(0 < p < 1)$ has been regarded as an attractive approach of sparse optimization, since (i) the ℓ_p-norm is a better approximation of the ℓ_0-norm (the number of nonzero entries); and (ii) it has been reported that solving an ℓ_p-norm minimization problem yields a sparser solution than minimizing the ℓ_1-norm [18, 19, 22, 54, 55]. By extending the idea of the LARS [16], the solution path of a least squares problem regularized by the ℓ_p-norm has been investigated [23, 56, 57]. The solution path consists of all the solutions, each of which corresponds to a value of the regularization parameter. In the ℓ_1 case, the solution moves along a gradual change of the regularization parameter, and hence the solution path is continuous. On the other hand, in the ℓ_p case, it has been reported that the solution suddenly jumps, whereas the corresponding regularization parameter changes gradually [23, 57]. Because of nonconvexity of the ℓ_p-norm, the ℓ_p-regularized least squares problem can have multiple critical points, where the gradient is zero, including the solutions. (See also Definition 2.) By tracking the critical points with changing the regularization parameter, one can obtain a continuous and piecewise smooth paths.

In this chapter, it is shown that every sparsest OLS solution is connected to the origin by a continuous path of critical points (Theorem 3.2). The path of critical points is shortly referred to as a *critical-point path*. The critical-point path is investigated first in an overdetermined system, where the OLS solution is unique. It is proved that (i) there exists a unique path emerging from the OLS solution and (ii) it does not stop (the velocity vector of the path does not become zero) as long as the support does not change (see Lemma 3.1). This implies that the path eventually falls out of the region corresponding to the current support unless the path ends up with some loop(s). It is ensured automatically that a path does not bifurcate excluding those critical points at which the Hessian matrix of the ℓ_p-regularized squared error function is singular. Such a singular point always exists on a path indeed (see Proposition 3.2), and therefore it is assumed that the path does not bifurcate at any singular points in order to make the analysis feasible. Although the assumption excluding bifurcation is

required for the main theorem, the theorem holds even when the assumption is violated (see Remark 3.1).

The above argument tells us that the path goes continuously in the region corresponding to the current support and finally reaches a point out of the region. The support of this end point is a proper subset of that of the OLS solution (which means that the end point has at least one zero component), and it is the OLS solution of a modified squared error function forcing those components associated with the off-support to be zero. (See Proposition 3.1.) Repetitions of this lead the path to the origin, which verifies the existence of a piecewise smooth path (with each end point being its breakpoint) from the OLS solution to the origin in the overdetermined case. (See Theorem 3.1.) The basic idea of the proof for an underdetermined case is reducing the problem to an overdetermined one in terms of a small number of components associated with the support of an OLS solution. The reduction is only possible for such a solution whose support has its associated submatrix of the fat sensing matrix be full column rank. (See Theorem 3.3.) It is proved that being full-rank is a necessary condition for a reduced overdetermined system to include the sparsest OLS solution of the entire underdetermined system. (See Lemma 3.2) There is also a possibility that a path exists even though the problem cannot be reduced to an overdetermined one. For such an issue, a necessary condition for an OLS solution to be connected with the origin by a continuous path is presented. (See Proposition 3.3.)

3.2 A Path to the OLS Solution

An overdetermined system, where the OLS solution is unique, is considered in this section. Figure 3.1 illustrates critical-point paths, which are obtained by gradually increasing the ℓ_p-norm of critical points. One can see that the blue path connects the origin and the OLS solution $\boldsymbol{h}^{\mathrm{LS}}$, while the red and green paths reach some other point. The existence of a critical-point path connecting the origin and the OLS solution is proved under a mild condition. It is also shown that the path passes through a point at which the Hessian matrix of $f_\lambda := \varphi(\boldsymbol{h}) + \lambda F_p(\boldsymbol{h})$ is singular.

3.2.1 Existence of a Continuous Critical-point Path

Let us arbitrarily choose a vector $\hat{\boldsymbol{h}} \in \mathbb{R}^m$ with its support $\mathcal{I} := \mathrm{supp}(\hat{\boldsymbol{h}})$ satisfying $\nabla_{\mathcal{I}}\varphi(\hat{\boldsymbol{h}}) = \boldsymbol{0}$, and define

$$S_{\mathcal{I}} := \{\boldsymbol{h} : \mathrm{supp}(\boldsymbol{h}) = \mathcal{I}\} \ (\ni \hat{\boldsymbol{h}}). \tag{3.1}$$

The matrix $\boldsymbol{G} := \boldsymbol{U}\boldsymbol{U}^{\mathsf{T}} \in \mathbb{R}^{m\times m}$ is positive definite in an overdetermined case, from which it can be verified that $\hat{\boldsymbol{h}}$ is a unique critical point in $S_{\mathcal{I}}$ for $\lambda = 0$.

The critical point condition (2.6) can be rewritten as

$$\boldsymbol{U}_{\mathcal{I}}\boldsymbol{U}^{\mathsf{T}}(\boldsymbol{h} - \boldsymbol{h}^{\mathrm{LS}}) + \lambda\nabla_{\mathcal{I}}F_p(\boldsymbol{h}) = \boldsymbol{0}, \tag{3.2}$$

where $\boldsymbol{U}_{\mathcal{I}} \in \mathbb{R}^{|\mathcal{I}|\times l}$ is a submatrix of $\boldsymbol{U}$ consisting of those row vectors associated with the support $\mathcal{I}$. Since

$$\nabla_{\mathcal{I}}\varphi(\hat{\boldsymbol{h}}) = \boldsymbol{U}_{\mathcal{I}}\boldsymbol{U}^{\mathsf{T}}(\hat{\boldsymbol{h}} - \boldsymbol{h}^{\mathrm{LS}}) = \boldsymbol{0}, \tag{3.3}$$

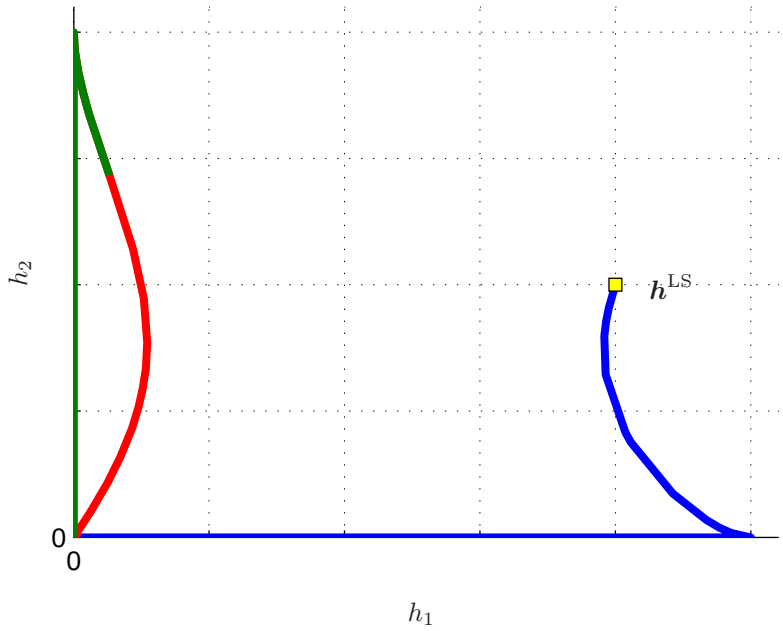


Figure 3.1: Critical-point paths in a two-dimensional overdetermined case.

one can obtain

$$\begin{aligned}\boldsymbol{U}_{\mathcal{I}}\boldsymbol{U}^{\mathsf{T}}(\boldsymbol{h}-\boldsymbol{h}^{\mathrm{LS}})-\nabla_{\mathcal{I}}\varphi(\hat{\boldsymbol{h}})+\lambda\nabla_{\mathcal{I}}F_p(\boldsymbol{h}) &= \boldsymbol{U}_{\mathcal{I}}\boldsymbol{U}^{\mathsf{T}}(\boldsymbol{h}-\hat{\boldsymbol{h}})+\lambda\nabla_{\mathcal{I}}F_p(\boldsymbol{h})\\ &= \boldsymbol{G}_{\mathcal{I}}(\boldsymbol{h}_{\mathcal{I}}-\hat{\boldsymbol{h}}_{\mathcal{I}})+\lambda\nabla_{\mathcal{I}}F_p(\boldsymbol{h})\\ &= \boldsymbol{0} \qquad (3.4)\end{aligned}$$

for any critical point $\boldsymbol{h}\in S_{\mathcal{I}}$, where $\boldsymbol{G}_{\mathcal{I}}:=\boldsymbol{U}_{\mathcal{I}}\boldsymbol{U}_{\mathcal{I}}^{\mathsf{T}}$ and $\boldsymbol{h}_{\mathcal{I}}\in\mathbb{R}^{|\mathcal{I}|}$ is a subvector of $\boldsymbol{h}$ consisting of those components associated with $\mathcal{I}$. The second equality holds because $\boldsymbol{U}^{\mathsf{T}}\boldsymbol{h}=\boldsymbol{U}_{\mathcal{I}}^{\mathsf{T}}\boldsymbol{h}_{\mathcal{I}}$. A total differential of (3.4) leads to

$$\boldsymbol{K}_{\mathcal{I}}d\boldsymbol{h}_{\mathcal{I}}=-d\lambda\nabla_{\mathcal{I}}F_p(\boldsymbol{h}). \qquad (3.5)$$

where $\boldsymbol{K}_{\mathcal{I}}:=\nabla_{\mathcal{I}}\nabla_{\mathcal{I}}f_\lambda(\boldsymbol{h})=\boldsymbol{G}_{\mathcal{I}}+\lambda\nabla_{\mathcal{I}}\nabla_{\mathcal{I}}F_p(\boldsymbol{h})$ is the Hessian matrix of f_λ.

As shown below, the matrix $\boldsymbol{K}_{\mathcal{I}}$ becomes singular at some point on a critical-point path. Assume that the following statements hold to exclude the case that a path bifurcates at singular points, guaranteeing that the path has no loop.

Assumption 3.1. *For any critical point $\tilde{\boldsymbol{h}}$ such that the Hessian matrix $\boldsymbol{K}_{\mathcal{I}}$ is singular, the following hold:*

(i) *the zero eigenvalue of $\boldsymbol{K}_{\mathcal{I}}$ is simple; and*

(ii) $\nabla_{\mathcal{I}}F_p(\tilde{\boldsymbol{h}})\notin\mathcal{R}(\boldsymbol{K}_{\mathcal{I}})$.

The main theorem of this section is stated below.

Theorem 3.1. *In an overdetermined case, there exists a continuous path of critical points connecting the origin and the OLS solution.*

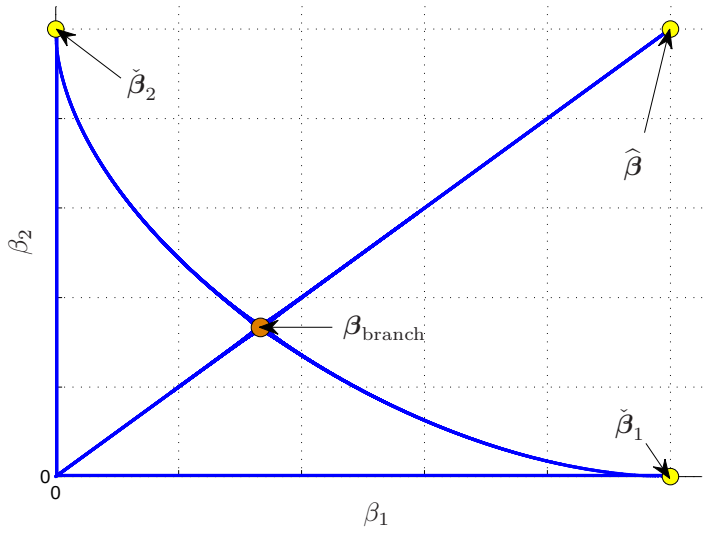


Figure 3.2: A case that Assumption 3.1 is violated but Theorem 3.1 still holds.

Remark 3.1 (On Assumption 3.1)**.**

(a) Assumption 3.1 is violated in the following cases:

(i) two eigenvalues become zero coincidentally:

(ii) the vector $\nabla_{\mathcal{I}} F_p(\tilde{\boldsymbol{h}})$ happens to lie in the subspace $\mathcal{R}(\boldsymbol{K}_{\mathcal{I}})$ which has no volume in $\mathbb{R}^{|\mathcal{I}|}$.

(b) There exists a case where Assumption 3.1 is violated but Theorem 3.1 still holds.

Remark 3.1(b) implies that Assumption 3.1 is a sufficient, but not necessary, condition for Theorem 3.1. Figure 3.2 illustrates the case of $\boldsymbol{G} = \boldsymbol{I}$, $\widehat{\boldsymbol{\beta}} = [1\ 1]^{\mathsf{T}}$ and $p = 0.5$; the Hessian matrix $\boldsymbol{K}_{\mathcal{I}}$ becomes a zero matrix at $\boldsymbol{\beta}_{\text{branch}}$ and has the zero eigenvalues with multiplicity two. As Assumption 3.1 is violated, bifurcation happens on the path emerging from $\widehat{\boldsymbol{\beta}}$ at the branch point $\boldsymbol{\beta}_{\text{branch}}$ into three branches. However, one can find that all the branches reach the origin.

Theorem 3.1 can be verified by applying the following proposition recursively, as shown later on.

Proposition 3.1. *In the region $S_{\mathcal{I}}$, there exists a path connecting the $\hat{\boldsymbol{h}}$ and some $\check{\boldsymbol{h}}$ such that (i) $supp(\check{\boldsymbol{h}}) = \check{\mathcal{I}} \subsetneq \mathcal{I}$ and (ii) $\nabla_{\check{\mathcal{I}}} \varphi(\check{\boldsymbol{h}}) = \boldsymbol{0}$ ($\lambda = 0$). The edge point $\check{\boldsymbol{h}}$ lies on the boundary of the open set $S_{\mathcal{I}}$.*

Proposition 3.1 will be proved with the following lemma.

Lemma 3.1.

(a) The $\hat{\boldsymbol{h}}$ has a neighborhood in which there is a unique path that emerges from $\hat{\boldsymbol{h}}$ and lies in $S_{\mathcal{I}}$.

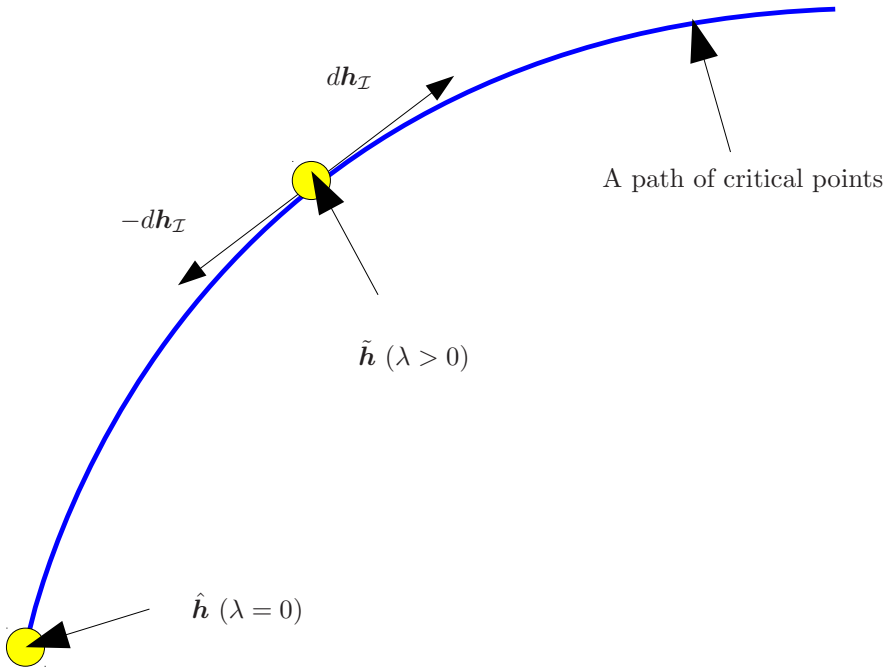


Figure 3.3: An illustration of Lemma 3.1(b).

(b) For any critical point $\tilde{\boldsymbol{h}} \in S_{\mathcal{I}}$ for $\lambda > 0$, there exists a unique path in $S_{\mathcal{I}}$ that passes through $\tilde{\boldsymbol{h}}$; i.e., the path never stops at any critical point for $\lambda > 0$.

(c) The path emerging from $\hat{\boldsymbol{h}}$ does not form any loop on $S_{\mathcal{I}}$.

Proof of Lemma 3.1(a): In a sufficiently small vicinity of $\hat{\boldsymbol{h}}$, $\boldsymbol{K}_{\mathcal{I}}$ is nonsingular (because $\boldsymbol{G}_{\mathcal{I}}$ is). Left-multiplying $\boldsymbol{K}_{\mathcal{I}}^{-1}$ to (3.5) yields the following diffenrential.

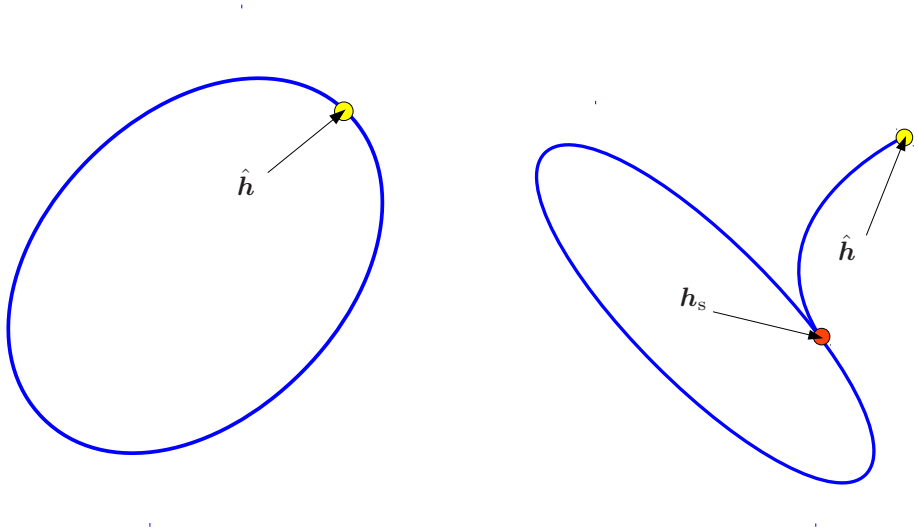
$$\frac{d\boldsymbol{h}_{\mathcal{I}}}{d\lambda} = -\boldsymbol{K}_{\mathcal{I}}^{-1} \nabla_{\mathcal{I}} F_p(\hat{\boldsymbol{h}}). \tag{3.6}$$

This verifies the claim.

Proof of Lemma 3.1(b): The case that $\boldsymbol{K}_{\mathcal{I}}$ is nonsingular has already been verified in the proof of Lemma 3.1(a). Suppose now that $\boldsymbol{K}_{\mathcal{I}}$ is singular. Then, an eigenvector $\boldsymbol{\varepsilon}_0$ of $\boldsymbol{K}_{\mathcal{I}}$ corresponding to the zero eigenvalue satisfies $\boldsymbol{K}_{\mathcal{I}}\boldsymbol{\varepsilon}_0 = \boldsymbol{0}$, meaning that the pair $(d\boldsymbol{h}_{\mathcal{I}}, d\lambda) = (\boldsymbol{\varepsilon}_0, 0)$ satisfies (3.5). With Assumption 3.1, this verifies the existence and uniqueness of a path passing through $\tilde{\boldsymbol{h}}$.

Proof of Lemma 3.1(c): Lemma 3.1(a) ensures that the path never returns to the starting point $\hat{\boldsymbol{h}}$. Hence, a loop can exist only if the path bifurcates. However, bifurcation happens neither at nonsingular points (Lemma 3.1(b)) nor at singular points (Assumption 3.1). Figure 3.4 illustrates the cases denied in this work. The cases in Figure 3.4(a) and Figure 3.4(b) are denied by Lemma 3.1(a) and Assumption 3.1, respectively. □

Proof of Proposition 3.1: Lemma 3.1(c) verifies that the path emerging from $\hat{\boldsymbol{h}}$ never stops at any critical point $\tilde{\boldsymbol{h}} \in S_{\mathcal{I}}$ for $\lambda > 0$. By Lemma 3.1(b), the path does not form a loop and finally falls out of $S_{\mathcal{I}}$. The support of $\boldsymbol{h}$ diminishes



(a) A case denied by Lemma 3.1. (b) A case denied by Assumption 3.1.

Figure 3.4: The cases denied in Lemma 3.1 and Assumption 3.1.

and this verifies $\check{\mathcal{I}} \subsetneq \mathcal{I}$. Let h_{i^*} be a vanishing component, where $i^* \in \mathcal{I} \setminus \check{\mathcal{I}}$. By the critical point condition (2.6), at any point $\tilde{\boldsymbol{h}} \in S_{\mathcal{I}}$ on the path excluding the edge point $\check{\boldsymbol{h}} \notin S_{\mathcal{I}}$, it holds that $\dfrac{\partial \varphi}{\partial h_{i^*}}(\tilde{\boldsymbol{h}}) = -\lambda \dfrac{\partial F_p}{\partial h_{i^*}}(\tilde{\boldsymbol{h}})$. As $\tilde{\boldsymbol{h}} \to \check{\boldsymbol{h}}$, $\lambda \to 0$ since $\left| \dfrac{\partial F_p}{\partial h_{i^*}}(\tilde{\boldsymbol{h}}) \right| \to \infty$, while $\left| \dfrac{\partial \varphi}{\partial h_{i^*}}(\tilde{\boldsymbol{h}}) \right| < \infty$. It is hence verified by (2.6) again that $\nabla_{\check{\mathcal{I}}} \varphi(\check{\boldsymbol{h}}) = \boldsymbol{0}$. □

Proof of Theorem 3.1: Let $\hat{\boldsymbol{h}} = \boldsymbol{h}^{\rm LS}$. Then, by Proposition 3.1, the $\hat{\boldsymbol{h}}$ is connected with another critical point $\check{\boldsymbol{h}} \in S_{\check{\mathcal{I}}}$ for some $S_{\check{\mathcal{I}}} \subsetneq S_{\mathcal{I}}$ and $\lambda = 0$. Since $\check{\boldsymbol{h}}$ is also a critical point for $\lambda = 0$, it is also possible to consider a path in $S_{\check{\mathcal{I}}}$ emerging from the $\check{\boldsymbol{h}}$ by just the same way with the first stage except replacing $\hat{\boldsymbol{h}}$ by $\check{\boldsymbol{h}}$. Repeating such stages recursively, the path eventually reaches to the origin corresponding to the empty support, and the existence of a path connecting the origin and $\boldsymbol{h}^{\rm LS}$ is verified. □

3.2.2 Characteristics of Critical-point Paths

Letting both of $\hat{\boldsymbol{h}} \in S_{\mathcal{I}}$ and $\check{\boldsymbol{h}} \in S_{\check{\mathcal{I}}}$ be critical points for $\lambda = 0$, the following proposition on a critical point at which $\boldsymbol{K}_{\mathcal{I}}$ is singular can be proved.

Proposition 3.2. *A path connecting the $\hat{\boldsymbol{h}}$ and $\check{\boldsymbol{h}}$ passes through a point at which $d\lambda = 0$ (see (3.5),) and the Hessian matrix $\boldsymbol{K}_{\mathcal{I}}$ of f_λ is singular.*

Proof: Both $\hat{\boldsymbol{h}}$ and $\check{\boldsymbol{h}}$ are critical points for $\lambda = 0$ and the other points on the path are for $\lambda > 0$. Therefore, λ along the path emerging from $\hat{\boldsymbol{h}}$ increases up to some point and then starts to decrease. At the changing point, it holds that $d\lambda = 0$. Substituting $d\lambda = 0$ into (3.5) yields $\boldsymbol{K}_{\mathcal{I}} d\boldsymbol{h}_{\mathcal{I}} = \boldsymbol{0}$. Since $d\boldsymbol{h}_{\mathcal{I}} \neq \boldsymbol{0}$ on the path, it follows that $\boldsymbol{K}_{\mathcal{I}}$ is singular at the changing point and $d\boldsymbol{h}_{\mathcal{I}}$ is an eigenvector corresponding to the zero eigenvalue. □

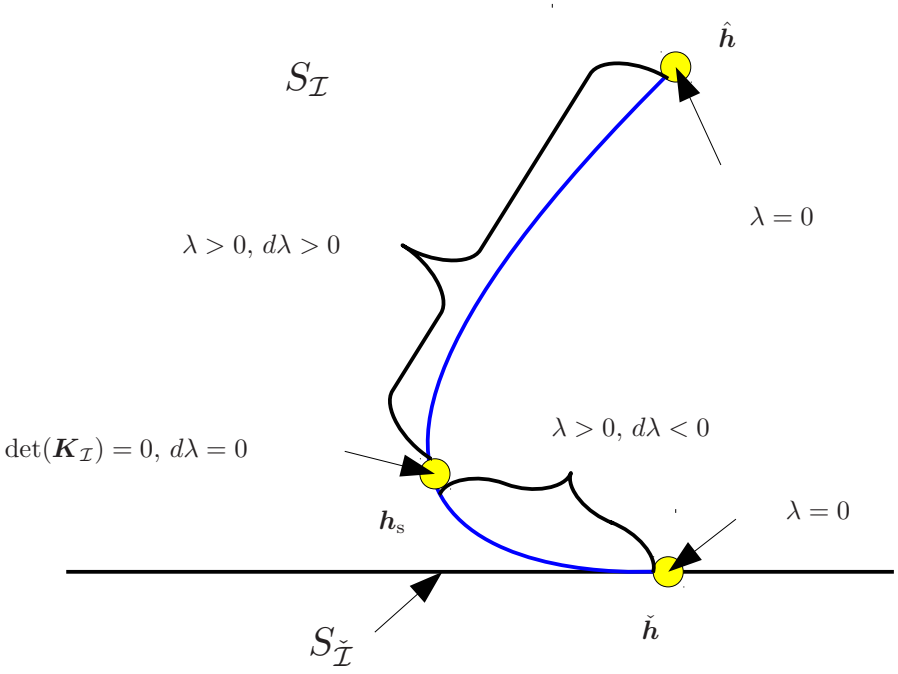


Figure 3.5: An illustration of Remark 3.2.

Below are remarks on the behavior of λ under the light of Proposition 3.2. (See Figure 3.5.)

Remark 3.2 (Behavior of λ along a path emerging from $\hat{\boldsymbol{h}}$)**.**

(a) At $\hat{\boldsymbol{h}}$ ($\lambda = 0$) : The nonsingularity of $\boldsymbol{G}_{\mathcal{I}}$ ensures the uniqueness and existence of $d\boldsymbol{h}_{\mathcal{I}}$.

(b) From $\hat{\boldsymbol{h}}$ to the singular point $\boldsymbol{h}_{\mathrm{s}}$ ($d\lambda = 0$) : By Lemma 3.1(c) and Assumption 3.1, if $\boldsymbol{K}_{\mathcal{I}}$ is nonsingular at a critical point, a single path passes the point.

(c) At $\boldsymbol{h}_{\mathrm{s}}$: By Proposition 3.2, there is a point at which $d\lambda = 0$ and $\boldsymbol{K}_{\mathcal{I}}$ is singular.

(d) From $\boldsymbol{h}_{\mathrm{s}}$ to $\check{\boldsymbol{h}}$: The path goes inside $S_{\mathcal{I}}$ with decreasing λ and falls out of $S_{\mathcal{I}}$ to reach $\check{\boldsymbol{h}}$.

3.3 Paths to Sparsest OLS Solutions

The primal interest of this chapter concerns the paths that connect the origin and the set V^* of OLS solutions in the underdetermined case. More specifically, only the paths containing a single OLS solution are considered. Although many OLS solutions exist, only some of them are connected with the origin by paths

of critical points. The existence of the paths, each of which leads to a sparsest OLS solution, are proved.

The statement below is the main result of this chapter.

Theorem 3.2. *Every sparsest OLS solution is connected with the origin by a continuous path of critical points.*

The following lemma is useful to prove Theorem 3.2.

Lemma 3.2. *Given a fixed OLS solution $\hat{\boldsymbol{h}} \in V^*$, let $\mathcal{I} = supp(\hat{\boldsymbol{h}})$. Consider the following three conditions.*

(C1) *$\boldsymbol{G}_{\mathcal{I}}$ is nonsingular.*

(C2) *$\{\boldsymbol{h} \in V^* : supp(\boldsymbol{h}) \subseteq \mathcal{I}\} = \{\hat{\boldsymbol{h}}\}$; i.e., $\hat{\boldsymbol{h}}$ is the unique OLS solution over the subspace in which $h_i = 0, \forall i \in \bar{\mathcal{I}} := \{1, 2, \cdots, n\} \setminus \mathcal{I}$.*

(C3) *$\hat{\boldsymbol{h}} \in \mathrm{argmin}_{\boldsymbol{h} \in V^*} \|\boldsymbol{h}\|_0$; i.e., $\hat{\boldsymbol{h}}$ is sparsest among the OLS solutions.*

Then, the following statements hold.

(a) (C1) *and* (C2) *are equivalent.*

(b) (C3) *implies* (C1) *and* (C2).

(c) Assume that

$$\min_{\boldsymbol{h} \in V^*} \|\boldsymbol{h}\|_0 = r := Rank(\boldsymbol{U}). \tag{3.7}$$

In this case, all the conditions (C1), (C2) *and* (C3) *are equivalent.*

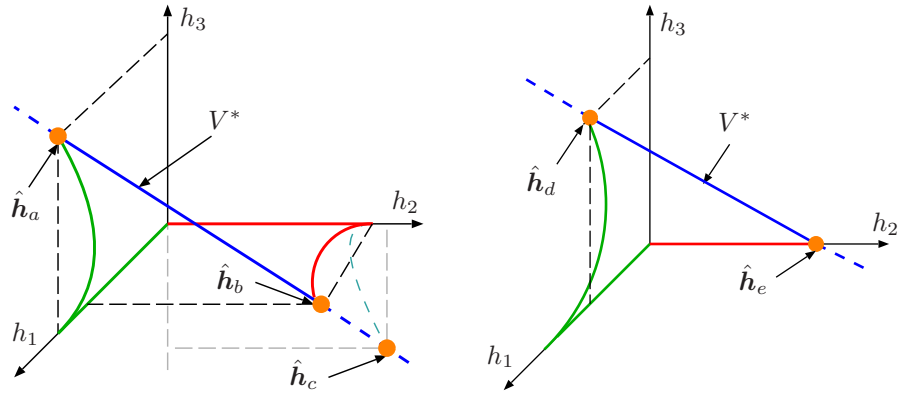
Proof of Lemma 3.2(a):
(C2) $\Rightarrow$ (C1): Assume that $\boldsymbol{G}_{\mathcal{I}}$ is singular, *i.e.*, $\mathcal{K}(\boldsymbol{U}_{\mathcal{I}}^{\mathsf{T}}) \setminus \{\boldsymbol{0}\}$ is nonempty. It is shown that $\hat{\boldsymbol{h}}$ is only an element in $\{\boldsymbol{h} \in V^* : \mathrm{supp}(\boldsymbol{h}) \subseteq \mathcal{I}\}$, or equivalently that there exists some $\tilde{\boldsymbol{h}} \neq \hat{\boldsymbol{h}}$ such that $\tilde{\boldsymbol{h}} \in V^*$ and $\mathrm{supp}(\tilde{\boldsymbol{h}}) \subseteq \mathcal{I}$. Then,

$$\begin{aligned} \tilde{\boldsymbol{h}} \in V^* &\Leftrightarrow \nabla\varphi(\tilde{\boldsymbol{h}}) = \boldsymbol{0} \\ &\Leftrightarrow \boldsymbol{U}(\boldsymbol{U}_{\mathcal{I}}^T \tilde{\boldsymbol{h}}_{\mathcal{I}} - \boldsymbol{d}) = \boldsymbol{0} \\ &\Leftrightarrow \boldsymbol{U}\boldsymbol{U}_{\mathcal{I}}^{\mathsf{T}} \tilde{\boldsymbol{h}}_{\mathcal{I}} = \boldsymbol{U}\boldsymbol{d}. \end{aligned} \tag{3.8}$$

The vector $\hat{\boldsymbol{h}}_{\mathcal{I}}$ is clearly a solution of (3.8) and the singularity of $\boldsymbol{G}_{\mathcal{I}}$ suggests that $\tilde{\boldsymbol{h}}_{\mathcal{I}} := \hat{\boldsymbol{h}}_{\mathcal{I}} + \boldsymbol{a}$ for any $\boldsymbol{a} \in \mathcal{K}(\boldsymbol{U}_{\mathcal{I}}^{\mathsf{T}}) \neq \{\boldsymbol{0}\}$ satisfies (3.8).
(C1) $\Rightarrow$ (C2): Assume that $\boldsymbol{G}_{\mathcal{I}}$ is nonsingular. It is clear that $\{\hat{\boldsymbol{h}}\} \subset \{\boldsymbol{h} \in V^* : \mathrm{supp}(\boldsymbol{h}) \subseteq \mathcal{I}\}$. The inclusion $\{\hat{\boldsymbol{h}}\} \supset \{\boldsymbol{h} \in V^* : \mathrm{supp}(\boldsymbol{h}) \subseteq \mathcal{I}\}$ can be verified by showing that $\tilde{\boldsymbol{h}} := \hat{\boldsymbol{h}} + \boldsymbol{a}$ satisfies (3.8) only for $\boldsymbol{a} = \boldsymbol{0}$. Since $\hat{\boldsymbol{h}}_{\mathcal{I}}$ is a solution of (3.8), substituting $\tilde{\boldsymbol{h}} := \hat{\boldsymbol{h}} + \boldsymbol{a}$ into (3.8) yields

$$\begin{aligned} \boldsymbol{U}\boldsymbol{U}_{\mathcal{I}}^{\mathsf{T}}\boldsymbol{a} = \boldsymbol{0} &\Leftrightarrow \boldsymbol{U}_{\mathcal{I}}^{\mathsf{T}}\boldsymbol{a} \in \mathcal{K}(\boldsymbol{U}) \subset \mathcal{K}(\boldsymbol{U}_{\mathcal{I}}) \\ &\Leftrightarrow \boldsymbol{U}_{\mathcal{I}}^{\mathsf{T}}\boldsymbol{a} = \boldsymbol{0}. \end{aligned} \tag{3.9}$$

The nonsingularity of $\boldsymbol{G}_{\mathcal{I}}$ verifies that (3.8) holds for $\boldsymbol{a} = \boldsymbol{0}$.
Proof of Lemma 3.2(b): It is shown that (C3) $\Rightarrow$ (C2). Assume that $\hat{\boldsymbol{h}}$ is sparsest over V^*. Suppose that there exists another vector $(\hat{\boldsymbol{h}} \neq)\tilde{\boldsymbol{h}} \in V^*$



(a) A case that (3.7) is satisfied. (b) A case that (3.7) is not satisfied.

Figure 3.6: Paths of critical points connect the origin and those OLS solutions satisfying (C2).

satisfying $\text{supp}(\tilde{\boldsymbol{h}}) = \text{supp}(\hat{\boldsymbol{h}})$. Then, there exists a sparser vector $\alpha\hat{\boldsymbol{h}} + (1-\alpha)\tilde{\boldsymbol{h}} \in V^*$ for some $\alpha \in \mathbb{R}$ than $\hat{\boldsymbol{h}}$. This contradicts the assumption and hence (C2) follows.

Proof of Lemma 3.2(c): (C3) $\Rightarrow$ (C2) has already been proved in the proof of Lemma 3.2(b) and the equivalence between (C1) and (C2) has also verified by Lemma 3.2(a). It is thus sufficient to show here that (C1) $\Rightarrow$ (C3) for completion of the proof. Assume that $\hat{\boldsymbol{h}} \in V^*$ does not satisfy (C3); *i.e.*, $\hat{\boldsymbol{h}} \notin \text{argmin}_{\boldsymbol{h}\in V^*} \|\boldsymbol{h}\|_0$. Then, $\|\hat{\boldsymbol{h}}\|_0 > r = \min_{\boldsymbol{h}\in V^*} \|\boldsymbol{h}\|_0$ by the assumption (3.7). This implies that $\boldsymbol{U}_{\mathcal{I}}$ is row rank deficient and hence $\boldsymbol{G}_{\mathcal{I}} = \boldsymbol{U}_{\mathcal{I}}\boldsymbol{U}_{\mathcal{I}}^{\mathsf{T}}$ is singular. This completes the proof. □

It must be pointed out that the assumption (3.7) is sufficient (but not necessary) to have the equivalence among (C1) – (C3). To elaborate the role of the assumption better, let us consider the three-dimensional examples described in Figure 3.6. Figure 3.6(a) illustrates the case that V^* is a one-dimensional linear manifold which intersects each of the coordinate planes h_1-h_2, h_2-h_3, and h_3-h_1 at a single point out of the h_1, h_2, and h_3 coordinates. In this case, one can readily verify that (3.7) is satisfied; note that $2 = \min_{\boldsymbol{h}\in V^*} \|\boldsymbol{h}\|_0 = \text{Rank}(\boldsymbol{U}) = 2$. Looking at the points $\hat{\boldsymbol{h}}_{\text{a}}$, $\hat{\boldsymbol{h}}_{\text{b}}$, and $\hat{\boldsymbol{h}}_{\text{c}}$ in the figure, one can find that the equivalence between (C2) and (C3) holds in this case. Now, turn to Figure 3.6(b), which illustrates the case that V^* intersects each of the coordinate planes h_1-h_2, h_2-h_3, and h_3-h_1 at a single point as the previous case but the intersection point $\hat{\boldsymbol{h}}_{\text{e}}$ is on the h_2 coordinate. In this case, (3.7) does not hold since $1 = \min_{\boldsymbol{h}\in V^*} \|\boldsymbol{h}\|_0 \neq \text{Rank}(\boldsymbol{U}) = 2$. The point $\hat{\boldsymbol{h}}_{\text{e}}$ indicates that (C3) implies (C2), while the point $\hat{\boldsymbol{h}}_{\text{d}}$ indicates that (C2) does not implies (C3).

The following theorem is more general than Theorem 3.2 in the sense that it guarantees the existence of a critical-point path not only for sparsest OLS solutions but also for some other ones.

Theorem 3.3. *For any $\hat{\boldsymbol{h}} \in V^*$ satisfying* (C1)*, there exists a continuous path of critical points connecting the origin and $\hat{\boldsymbol{h}}$.*

Proof: Consider critical points in the subspace in which $h_i = 0, \forall i \in \bar{\mathcal{I}}$. Define

$\varphi_{\mathcal{I}}(\boldsymbol{h}_{\mathcal{I}}) := \frac{1}{2}\|\boldsymbol{U}_{\mathcal{I}}^{\mathsf{T}}\boldsymbol{h}_{\mathcal{I}} - \boldsymbol{d}\|_2^2$ and $F_{p,\mathcal{I}}(\boldsymbol{h}_{\mathcal{I}}) := \frac{1}{p}\|\boldsymbol{h}_{\mathcal{I}}\|_p^p$. Then, it holds that

$$\varphi(\boldsymbol{h}) = \varphi_{\mathcal{I}}(\boldsymbol{h}_{\mathcal{I}}), \quad F_p(\boldsymbol{h}) = F_{p,\mathcal{I}}(\boldsymbol{h}_{\mathcal{I}})$$
$$\nabla_{\mathcal{I}}\varphi(\boldsymbol{h}) = \nabla\varphi_{\mathcal{I}}(\boldsymbol{h}_{\mathcal{I}}), \quad \nabla_{\mathcal{I}}F_p(\boldsymbol{h}) = \nabla F_{p,\mathcal{I}}(\boldsymbol{h}_{\mathcal{I}})$$

in the subspace. The critical point condition is therefore reduced to

$$\nabla\varphi_{\mathcal{I}}(\boldsymbol{h}_{\mathcal{I}}) + \lambda\nabla F_{p,\mathcal{I}}(\boldsymbol{h}_{\mathcal{I}}) = \boldsymbol{0}. \tag{3.10}$$

The critical points in the subspace can be expressed as those for the following problem:

$$\text{find } \boldsymbol{h}^*_{\lambda,\mathcal{I}} \in \operatorname*{argmin}_{\boldsymbol{h}_{\mathcal{I}}\in\mathbb{R}^{|\mathcal{I}|}} f_{\lambda,\mathcal{I}}(\boldsymbol{h}) := \varphi_{\mathcal{I}}(\boldsymbol{h}_{\mathcal{I}}) + \lambda F_{p,\mathcal{I}}(\boldsymbol{h}_{\mathcal{I}}). \tag{3.11}$$

Due to the nonsingularity of $\boldsymbol{G}_{\mathcal{I}}$, Theorem 3.1 can be applied to the problem in (3.11) to verify the existence of a continuous path of critical points connecting $\hat{\boldsymbol{h}}$ and the origin. □

Lemma 3.2(b) ensures that every sparsest solution satisfies (C2), and hence Theorem 3.3 directly implies Theorem 3.2. In Figure 3.6, one can see that all the points $\hat{\boldsymbol{h}}_{\mathrm{a}}$, $\hat{\boldsymbol{h}}_{\mathrm{b}}$, $\hat{\boldsymbol{h}}_{\mathrm{c}}$, $\hat{\boldsymbol{h}}_{\mathrm{d}}$, and $\hat{\boldsymbol{h}}_{\mathrm{e}}$ that satisfy (C2) are connected with the origin by paths of critical points. This illustrates the validity of Theorem 3.3 under Lemma 3.2.

3.4 Paths from other OLS Solutions

In this section, the possibilities of critical-point paths emerging from some of the OLS solutions, which have not been considered in the previous section, are discussed. In fact, being sparsest (or having a sufficiently small support) is a sufficient, but not necessary, condition for an OLS solution to be connected with the origin by a path. The following proposition gives a necessary condition for an OLS solution to have a path emerging from the solution itself.

Proposition 3.3. *Let $\hat{\boldsymbol{h}} \in V^*$ with its support $\mathcal{I}$. Then, in a vicinity of $\hat{\boldsymbol{h}}$, there exists a path of critical points with support $\mathcal{I}$ emerging from $\hat{\boldsymbol{h}}$ only if*

$$\nabla_{\mathcal{I}}F_p(\hat{\boldsymbol{h}}) \in \mathcal{R}(\boldsymbol{U}_{\mathcal{I}}). \tag{3.12}$$

Here, a path such that λ is continuously zero (in part) is neglected.

Proof: Suppose that there exists a path $\mathcal{C}$ of critical points with support $\mathcal{I}$ leading to $\hat{\boldsymbol{h}}$. Then, define a sequence $(\boldsymbol{h}_n)_{n\in\mathbb{N}} \subset \mathcal{C}$ converging to $\hat{\boldsymbol{h}}$. Since $\lambda = 0$ at $\hat{\boldsymbol{h}} \in V^*$, $\lambda \neq 0$ in the vicinity of $\hat{\boldsymbol{h}}$ and hence there exists a sufficiently large $N_0 \in \mathbb{N}$ such that $\nabla_{\mathcal{I}}F_p(\boldsymbol{h}_n)$ is aligned to $\nabla_{\mathcal{I}}\varphi(\boldsymbol{h}_n)$ for all $n \geq N_0$. This implies that $(\nabla_{\mathcal{I}}F_p(\boldsymbol{h}_n))_{n\geq N_0} \subset \mathcal{R}(\boldsymbol{U}_{\mathcal{I}})$. The closedness of the subspace $\mathcal{R}(\boldsymbol{U}_{\mathcal{I}})$ and the continuity of the operator $\nabla_{\mathcal{I}}F_p$ ensure $\nabla_{\mathcal{I}}F_p(\hat{\boldsymbol{h}}) \in \mathcal{R}(\boldsymbol{U}_{\mathcal{I}})$. □

The existence of an OLS solution $\hat{\boldsymbol{h}} \in V^*$ with full support which satisfies the condition in Proposition 3.3 can be considered as follows. Without loss of generality, suppose that $V^* \cap \mathcal{R}(\boldsymbol{U}) \subset \mathbb{R}^n_{++} := \{\boldsymbol{h} \in \mathbb{R}^m : h_i > 0, \forall i\}$; $V^* \cap \mathcal{R}(\boldsymbol{U})$ contains a sole vector which is the minimum Euclidean-norm OLS solution. Define dual coordinates by $\boldsymbol{\eta} := \nabla F_p(\boldsymbol{h})$ for $\boldsymbol{h} \in \mathbb{R}^n_{++}$. It is then not difficult to see that the correspondence between $\boldsymbol{h}$ and $\boldsymbol{\eta}$ is one to one over $\mathbb{R}^n_{++}$. Consider

the special case of Rank($\boldsymbol{U}$) = 1. In this case, $\mathcal{R}(\boldsymbol{U})$ is a one-dimensional subspace and V^* is a hyperplane in $\mathbb{R}^m$. One can see that the hyperplane V^* separates the two convex sets in $\mathbb{R}^n_{++}$: $S_1 := (0, \epsilon)^n$ and $S_2 := (\delta, \infty)^n$ for some small $\epsilon > 0$ and large $\delta > 0$. The subspace $\mathcal{R}(\boldsymbol{U})$ contains some $\boldsymbol{\eta}_1 \in (1/\sqrt{\epsilon}, \infty)^n$ and $\boldsymbol{\eta}_2 \in (0, 1/\sqrt{\delta})^n$. Then the inverse images of $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$ under the nonlinear mapping ∇F_p satisfy $\boldsymbol{h}_1 \in (0, \epsilon)^n$ and $\boldsymbol{h}_2 \in (\delta, \infty)^n$, respectively. This implies that the inverse image of the ray $\mathcal{R}(\boldsymbol{U}) \cap \mathbb{R}^n_{++}$ contains a continuous curve connecting the two points $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ which are separated by the hyperplane V^*. Hence, the curve crosses V^*, meaning that there exists an OLS solution with full support satisfying (3.12).

The same discussion can be also applied to the case that Rank($\boldsymbol{U}$) = $n - 1$. However, the above discussion gives us doubts that, in the case that $1 <$ Rank($\boldsymbol{U}$) $< n-1$, there exists in general no full-support OLS solution satisfying (3.12). In this case, one can extend the discussion for Rank($\boldsymbol{U}$) = $n - 1$ by restricting oneself to an $r + 1$ subspace

$$M_{\mathcal{I}} := \{\boldsymbol{h} : \mathrm{supp}(\boldsymbol{h}) \subseteq \mathcal{I}\} \tag{3.13}$$

for any $\mathcal{I} \subset \{1, 2, \cdots, n\}$ with $|\mathcal{I}| = r + 1$ so that $V^*_{\mathcal{I}} := \boldsymbol{h}_{\mathcal{I}} \in \mathbb{R}^{r+1} : [\boldsymbol{h}^T_{\mathcal{I}}\ \boldsymbol{0}^T] \in V^*\}$ is a one-dimensional line and $\mathcal{R}(\boldsymbol{U}_{\mathcal{I}}) \subset \mathbb{R}^{r+1}$ is a hyperplane in $\mathbb{R}^{r+1}$.

3.5 Concluding Remarks

In this chapter, critical points of a least squares problem under ℓ_p-regularization ($0 < p < 1$) have been studied. It has been proved under a mild condition that there exists a critical-point path for every sparsest OLS soluion which connects it with the origin. The major contributions of this work are summarized as follows.

- In an overdetermined case, there is a path connecting the unique OLS solution and the origin (Theorem 3.1).
- The path connecting the OLS solution and the origin consists of curves each of which connects critical points for $\lambda = 0$, hence includes a point where the Hessian matrix $\boldsymbol{K}_{\mathcal{I}}$ is singular (Proposition 3.2, Remark 3.2).
- Every sparsest OLS solution in an underdetermined case is connected with the origin by a continuous path of the critical points (Theorem 3.2).
- In an underdetermined case, there is a path connecting an OLS solution and the origin if the solution can be regarded as that of an overdetermined case (Theorem 3.3).
- An OLS solution has a path of critical points emerging from itself only if $\nabla_{\mathcal{I}} F_p(\boldsymbol{h})$ lies in the range of $\boldsymbol{G}_{\mathcal{I}}$ (Proposition 3.3).

It has turned out that ℓ_p-regularization ($0 < p < 1$) does yield sparsest solutions of a least squares problem. The existence theorem presented in this paper will become a driving force for developing an efficient method to compute a sparsest solution by using ℓ_p-regularization.

Chapter 4

Kernel Weight Design for Multikernel Adaptive Filtering

4.1 Introduction

How can multiple kernel functions cooperate well for estimating an unknown nonlinear function of interest in online fashion? This is a highly important question for the signal processing community because many tasks in signal processing and machine learning can be cast as online nonlinear estimation problems. Reproducing kernel has widely been studied due to a number of reasons including the so-called kernel trick to reduce an inner product computation in a possibly infinite dimensional space to finite-dimensional arithmetics, as well as the solid theoretical basis. Kernel adaptive filtering [26–28, 30, 33, 37, 38, 58–63] has become a popular approach to online nonlinear estimation during the last decade. The success of kernel adaptive filtering, however, relies heavily on the choice of kernel, and it is difficult in practice to design a kernel fitting a given specific task well.

In the machine learning community, MKL [39–44] has extensively been studied mainly under batch settings motivated by the difficulty of kernel design, and online algorithms have also been proposed recently [43, 44]. Most MKL algorithms consist of two steps. In the first step, an appropriate kernel which fits the observed data well is constructed by a convex combination of multiple kernels, which is then used to estimate the unknown nonlinear function. The second step of MKL is essentially a coefficient update with a single kernel, and hence MKL has a severe limitation in estimating a target function which consists of multiple components such as linear and nonlinear functions. In the signal processing community, on the other hand, multikernel adaptive filtering [47–49] has been studied as an effective method utilizing multiple reproducing kernels under online settings. See [45, 46, 64–69] for other approaches using multiple kernels. Multikernel adaptive filtering has a higher degree of freedom in estimation than MKL.Thanks to the higher degree of freedom, multikernel adaptive filtering enjoys (a) a more compact representation than MKL, and (b) a potential ability

to extract the local structures of the target containing multiple components.

Appropriate treatments of kernel weights are necessary for multikernel adaptive filtering since a naive use of unweighted kernels never guarantees desirable performance when there is some imbalance among the kernels. To use an analogy, suppose the following situation in linear adaptive filtering. When the power of the input vector is concentrated on a small subset of entries, the input autocorrelation matrix has large eigenvalue spread. It has been known that large eigenvalue spread of the input autocorrelation matrix causes unbalanced convergence rates among the eigenvectors, and hence deteriorates the performance of adaptive filtering [24, 25]. The eigenvalue spread can be reduced by imposing weights on the entries so that the powers of weighted entries become equal to each other. The only difference in the case of multikernel adaptive filtering from the linear case is that each kernel weight is not imposed on a single entry of the input vector, but on all the entries associated with the corresponding kernel. The power balance and convergence rates are then considered in a *kernel-wise* sense. A principled way of tuning the kernel weights, which lead to uniform convergence rates of the kernel-wise coefficient errors, for multikernel adaptive filtering has not been investigated so far. Such hyper-parameter tuning problems have been empirically dealt with under some pretraining or validation process, which is undesirable in online settings. To exploit the full potential of multikernel adaptive filtering, it is therefore essential to develop a technique which yields the appropriate kernel weights for an online manner.

The headline question presented at the beginning can be subdivided as follows: (i) Is there a set of kernel weights which equalizes the convergence rates of the kernel-wise coefficient errors? (ii) If such a set of weights exists, how can it be implemented in online fashion? In this chapter, an answer to the above questions is presented by deriving a recursive kernel weighting technique which updates the kernel weights at each iteration, instead of presetting the weights.

4.2 Transient Behavior and Weight Derivation

4.2.1 Preliminaries to Transient Analysis

In advance of discussions on transient analysis, some preliminary informations related to this chapter must be reminded. The nonlinear adaptive filtering model is given as follows.

$$d_n = \psi(\boldsymbol{u}_n) + \nu_n \in \mathbb{R},\ n \in \mathbb{N}, \tag{4.1}$$

where $\boldsymbol{u}_n \in \mathbb{R}^L$ is the input vector, d_n is the corresponding output generated by the unknown function ψ of interest, and ν_n is the zero-mean additive noise which is assumed independent of $\boldsymbol{u}_n$. Given Q kernel functions $\kappa_q(\cdot,\cdot),\ q = 1, 2, \cdots, Q$, and the corresponding dictionaries $\mathcal{D}_{q,n} := \{\kappa_q(\boldsymbol{x}_1^{(q)}, \cdot), \cdots, \kappa_q(\boldsymbol{x}_{r_{q,n}}^{(q)}, \cdot)\}$, whose cardinality is $r_{q,n}$, multikernel adaptive filtering achieves the following estimate of the unknown function ψ.

$$\hat{\varphi}_n(\boldsymbol{u}_n) = \sum_{q=1}^{Q} \sum_{j=1}^{r_{q,n}} h_{n,j}^{(q)} \kappa_q(\boldsymbol{x}_j^{(q)}, \boldsymbol{u}_n) = \sum_{q=1}^{Q} \boldsymbol{k}_n^{(q)\mathsf{T}} \boldsymbol{h}_n^{(q)} = \boldsymbol{k}_n^{\mathsf{T}} \boldsymbol{h}_n, \tag{4.2}$$

where

$$\boldsymbol{k}_n^{(q)} = \left[\kappa_q\left(\boldsymbol{x}_1^{(q)}, \boldsymbol{u}_n\right), \cdots, \kappa_q\left(\boldsymbol{x}_{r_{q,n}}^{(q)}, \boldsymbol{u}_n\right)\right]^{\mathsf{T}} \in \mathbb{R}^{r_{q,n}},$$
$$\boldsymbol{h}_n^{(q)} = \left[h_{n,1}^{(q)}, \cdots, h_{n,r_{q,n}}^{(q)}\right]^{\mathsf{T}} \in \mathbb{R}^{r_{q,n}},$$

for all $q = 1, 2, \cdots, Q$, and

$$\boldsymbol{k}_n = \left[\boldsymbol{k}_n^{(1)\mathsf{T}}, \cdots, \boldsymbol{k}_n^{(Q)\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{R}^{r_n},$$
$$\boldsymbol{h}_n = \left[\boldsymbol{h}_n^{(1)\mathsf{T}}, \cdots, \boldsymbol{h}_n^{(Q)\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{R}^{r_n}$$

are the concatenated vectors, where $r_n = \sum_{q=1}^{Q} r_{q,n}$. The nonlinear estimation of ψ then reduces to linear estimation of the Euclidean vector $\boldsymbol{h}^*$, which is a minimizer of some error function. The MKNLMS algorithm [47, 48] is a multikernel adaptive filtering algorithm of which the error function is the mean squared error $\mathrm{MSE}(\boldsymbol{h}) := E\left[(\boldsymbol{k}_n^{\mathsf{T}}\boldsymbol{h} - d_n)^2\right]$, where $\boldsymbol{h} \in \mathbb{R}^{r_n}$ and $E[\cdot]$ stands for the expectation with respect to the input $\boldsymbol{u}_n$ and the noise ν_n. Without a manipulation of dictionaries, each update of the MKNLMS algorithm can be written as follows:

$$\boldsymbol{h}_{n+1} = \boldsymbol{h}_n - \mu \frac{\boldsymbol{k}_n^{\mathsf{T}}\boldsymbol{h}_n - d_n}{\|\boldsymbol{k}_n\|_2^2}\boldsymbol{k}_n, \tag{4.3}$$

Applying the kernel weights to the update (4.3) yields

$$\boldsymbol{g}_{n+1} = \boldsymbol{g}_n - \mu \frac{\boldsymbol{k}_n^{\mathsf{T}}\boldsymbol{W}\boldsymbol{g}_n - d_n}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\boldsymbol{W}\boldsymbol{k}_n, \tag{4.4}$$

where $\boldsymbol{g}_n := \boldsymbol{W}^{-1}\boldsymbol{h}_n$ and the weight matrix $\boldsymbol{W}$ is given as

$$\boldsymbol{W} := \begin{bmatrix} w_1\boldsymbol{I}_{r_{1,n}} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & w_2\boldsymbol{I}_{r_{2,n}} & \cdots & \boldsymbol{O} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & w_Q\boldsymbol{I}_{r_{Q,n}} \end{bmatrix}. \tag{4.5}$$

On the other hand, the coefficient error $\boldsymbol{z}_n := \boldsymbol{g}_n - \boldsymbol{g}^*$, where $\boldsymbol{g}^* := \boldsymbol{W}^{-1}\boldsymbol{h}^*$, is updated by equation (4.4) as follows:

$$\boldsymbol{z}_{n+1} = \boldsymbol{z}_n - \mu \frac{\boldsymbol{k}_n^{\mathsf{T}}\boldsymbol{W}\boldsymbol{z}_n - \nu_n}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\boldsymbol{W}\boldsymbol{k}_n. \tag{4.6}$$

For the qth block, the blockwise update is given as

$$\boldsymbol{z}_{n+1}^{(q)} = \boldsymbol{z}_n^{(q)} - \mu w_q \frac{\boldsymbol{k}_n^{\mathsf{T}}\boldsymbol{W}\boldsymbol{z}_n - \nu_n}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\boldsymbol{k}_n^{(q)}, \tag{4.7}$$

$$\begin{aligned}\|\boldsymbol{z}_{n+1}^{(q)}\|_2^2 =& \|\boldsymbol{z}_n^{(q)}\|_2^2 - 2\mu w_q \frac{\boldsymbol{k}_n^{\mathsf{T}}\boldsymbol{W}\boldsymbol{z}_n - \nu_n}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\boldsymbol{k}_n^{(q)\mathsf{T}}\boldsymbol{z}_n^{(q)} \\ &+ (\mu w_q)^2 \frac{\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\frac{\left(\boldsymbol{k}_n^{\mathsf{T}}\boldsymbol{W}\boldsymbol{z}_n - \nu_n\right)^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}.\end{aligned} \tag{4.8}$$

It is clear that the transient behavior of the squared norm (4.8) depends on the choice of the kernel weights. It is now to focus on the convergence rates of $\|\boldsymbol{z}_{n+1}^{(q)}\|_2^2,\ q = 1, 2, \cdots, Q$, and find the kernel weights which lead to the uniform convergence rate, *i.e.*, $E\big[\|\boldsymbol{z}_{n+1}^{(q)}\|_2^2\big] = \rho E\big[\|\boldsymbol{z}_n^{(q)}\|_2^2\big]$ with the same $\rho \in (0, 1)$ for all q. For simplicity, the transient analysis of the coefficient errors is considered in the case that the dictionaries are fixed ($r_{q,n}$ is fixed as well for all q). In addition to fixing the dictionaries, the following properties are also assumed.

Assumption 4.1.

(a) At each time n, the kernelized input $\boldsymbol{k}_n$ and the additive noise ν_n are independent of each other.

(b) The fluctuations in the squared norms $\|\boldsymbol{k}_n^{(q)}\|_2^2,\ q = 1, 2, \cdots, Q$, and $\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2$ are sufficiently small to justify the approximations

$$E\left[\frac{\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right] \approx \frac{E\big[\|\boldsymbol{k}_n^{(q)}\|_2^2\big]}{E\big[\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2\big]},$$

where the matrix $\boldsymbol{W}$ is given as (4.5).

The above assumptions are commonly used for performance analysis of the NLMS algorithm [24, 25].

The goal of this chapter is to derive a set of weights that equalizes the expected convergence rates of the kernel-wise subvectors $\boldsymbol{z}_n^{(q)},\ q = 1, 2, \cdots, Q$, of the coefficient error vector $\boldsymbol{z}_n$ in the sense of the squared norm. The coefficient error vector $\boldsymbol{z}_n$ itself, however, is unknown due to its dependency on the unknown target $\boldsymbol{g}^*$. To tackle with this issue, we assume $\boldsymbol{z}_n$ as a random variable which has the following properties:

Assumption 4.2 (Assumption on $\boldsymbol{z}_n$)**.**

(a) The coefficient error vector $\boldsymbol{z}_n$ is distributed spherically with zero mean.

(b) The distribution of $\boldsymbol{z}_n$ is nearly even in the larger area than the area where the weighted input $\boldsymbol{W}\boldsymbol{k}_n$ can exist, so that $\boldsymbol{z}_n$ is approximately independent of $\boldsymbol{W}\boldsymbol{k}_n$.

The above assumption implies the following properties:

- The subvectors $\boldsymbol{z}_n^{(q)},\ q = 1, 2, \cdots, Q$ are independent or each other.
- The probability of $\boldsymbol{z}_n$ depends only on its squared norm $\|\boldsymbol{z}_n\|_2^2$.

4.2.2 Derivation of Kernel Weights Equalizing Convergence Rates

Using the expectation $E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_n^{(q)}\|_2^2\big]$ with respect to $\boldsymbol{z}_n$, the transient analysis on (4.8) becomes tractable even if $\boldsymbol{z}_n^{(q)}$ itself is unknown. The expectation of

(4.8) with respect to $\boldsymbol{z}_n$ is

$$\begin{aligned}E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_{n+1}^{(q)}\|_2^2\big] =& E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_n^{(q)}\|_2^2\big] - 2\mu w_q E_{\boldsymbol{z}_n}\left[\frac{\boldsymbol{z}_n^{(q)\mathsf{T}}\boldsymbol{k}_n^{(q)}\boldsymbol{k}_n^{\mathsf{T}}\boldsymbol{W}\boldsymbol{z}_n}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right] \\ &+ (\mu w_q)^2\frac{\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}E_{\boldsymbol{z}_n}\left[\frac{(\boldsymbol{k}_n^{\mathsf{T}}\boldsymbol{W}\boldsymbol{z}_n)^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right] \\ &+ (\mu w_q)^2\frac{\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\frac{\nu_n^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}. \end{aligned} \tag{4.9}$$

The quadratic terms in (4.9) are calculated with the following lemma.

Lemma 4.1. *Let $\boldsymbol{A} \in \mathbb{R}^{r_n\times r_n}$ be an arbitrary matrix. If $\boldsymbol{z}_n$ is spherically distributed with zero mean, it holds that*

$$E_{\boldsymbol{z}_n}\big[\boldsymbol{z}_n^{\mathsf{T}}\boldsymbol{A}\boldsymbol{z}_n\big] = \frac{E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_n\|_2^2\big]}{r_n}\mathrm{Trace}\big(\boldsymbol{A}\big). \tag{4.10}$$

Proof: It is trivial that

$$\begin{aligned}E_{\boldsymbol{z}_n}\big[\boldsymbol{z}_n^{\mathsf{T}}\boldsymbol{A}\boldsymbol{z}_n\big] =& \mathrm{Trace}\big(E_{\boldsymbol{z}_n}\big[\boldsymbol{z}_n\boldsymbol{z}_n^{\mathsf{T}}\boldsymbol{A}\big]\big) \\ =& \mathrm{Trace}\big(E_{\boldsymbol{z}_n}\big[\boldsymbol{z}_n\boldsymbol{z}_n^{\mathsf{T}}\big]\boldsymbol{A}\big). \end{aligned} \tag{4.11}$$

Since $\boldsymbol{z}_n$ is spherically distributed with zero mean, it holds that

$$E_{\boldsymbol{z}_n}\big[\boldsymbol{z}_n\boldsymbol{z}_n^{\mathsf{T}}\big] = \lambda_n\boldsymbol{I}_{r_n}, \tag{4.12}$$

for some $\lambda_n \geq 0$. From (4.12), the variance λ_n can be calculated as

$$\lambda_n = \frac{\mathrm{Trace}\big(E_{\boldsymbol{z}_n}\big[\boldsymbol{z}_n\boldsymbol{z}_n^{\mathsf{T}}\big]\big)}{\mathrm{Trace}\big(\boldsymbol{I}_{r_n}\big)} = \frac{E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_n\|_2^2\big]}{r_n}. \tag{4.13}$$

Substituting both of (4.12) and (4.13) into (4.11) leads to

$$\begin{aligned}E_{\boldsymbol{z}_n}\big[\boldsymbol{z}_n^{\mathsf{T}}\boldsymbol{A}\boldsymbol{z}_n\big] =& \mathrm{Trace}\big(E_{\boldsymbol{z}_n}\big[\boldsymbol{z}_n\boldsymbol{z}_n^{\mathsf{T}}\big]\boldsymbol{A}\big) \\ =& \mathrm{Trace}\big(\lambda_n\boldsymbol{I}_{r_n}\boldsymbol{A}\big) \\ =& \lambda_n\mathrm{Trace}\big(\boldsymbol{A}\big) \\ =& \frac{E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_n\|_2^2\big]}{r_n}\mathrm{Trace}\big(\boldsymbol{A}\big), \end{aligned} \tag{4.14}$$

which completes the proof. □

By Lemma 4.1 and Assumption 4.2(b), (4.9) leads to

$$\begin{aligned}E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_{n+1}^{(q)}\|_2^2\big] \approx& E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_n^{(q)}\|_2^2\big] - 2\frac{\mu w_q^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\|\boldsymbol{k}_n^{(q)}\|_2^2\frac{E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_n^{(q)}\|_2^2\big]}{r_{q,n}} \\ &+ \frac{(\mu w_q)^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\|\boldsymbol{k}_n^{(q)}\|_2^2\left(\frac{E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_n\|_2^2\big]}{r_n} + \frac{\nu_n^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right) \\ =& \left(1 - \frac{\mu(2-\mu)}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{r_{q,n}}\right)E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_n^{(q)}\|_2^2\big] \\ &+ (\mu w_q)^2\frac{\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\frac{\nu_n^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}, \end{aligned} \tag{4.15}$$

where the second equality holds by

$$E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_n^{(q)}\|_2^2\big] = \frac{r_{q,n}}{r_n} E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_n\|_2^2\big], \tag{4.16}$$

which is readily verified by Lemma 4.1. The expectation of (4.15) with respect to $\boldsymbol{k}_n$ and ν_n is

$$\begin{aligned} E\big[E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_{n+1}^{(q)}\|_2^2\big]\big] = E\bigg[1 - \frac{\mu(2-\mu)}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{r_{q,n}}\bigg] E_{\boldsymbol{z}_n}\big[\|\boldsymbol{z}_n^{(q)}\|_2^2\big] \\ + (\mu w_q)^2 E\bigg[\frac{\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\frac{\nu_n^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\bigg]. \end{aligned} \tag{4.17}$$

Since $\mu \in (0,2)$, it holds that

$$0 < \frac{\mu(2-\mu)}{r_{q,n}} \leq \frac{1}{r_{q,n}}. \tag{4.18}$$

In addition, by $w_q > 0,\ q = 1, 2, \cdots, Q$, it holds that

$$0 < \frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2} < 1. \tag{4.19}$$

From (4.18) and (4.19), it follows that

$$E\bigg[1 - \frac{\mu(2-\mu)}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{r_{q,n}}\bigg] = 1 - \frac{\mu(2-\mu)}{r_{q,n}} E\bigg[\frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\bigg] \in (0,1). \tag{4.20}$$

This implies that the expected squared norms $E_{\boldsymbol{z}_n}[\|\boldsymbol{z}_n^{(q)}\|_2^2]$ of the kernel-wise coefficient errors decrease monotonically after each update (4.4). Then the expected squared norms converge to zero and (4.20) can be regarded as the convergence rate. If a uniform convergence rate, say $\rho > 0$, is achieved by some kernel weight setting, then it holds that

$$E\bigg[1 - \frac{\mu(2-\mu)}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{r_{q,n}}\bigg] = \rho, \tag{4.21}$$

for all $q = 1, 2, \cdots, Q$, as shown in the following proposition.

Proposition 4.1. *Setting the kernel weights as*

$$w_q = \gamma\bigg(\frac{E\big[\|\boldsymbol{k}_n^{(q)}\|_2^2\big]}{r_{q,n}}\bigg)^{-\frac{1}{2}},\ \gamma > 0,\ q = 1, 2, \cdots, Q, \tag{4.22}$$

achieves the uniform convergence rate

$$\rho = 1 - \frac{\mu(2-\mu)}{r_n}, \tag{4.23}$$

for all $q = 1, 2, \cdots, Q$.

Proof: From Assumption 4.1(b) and (4.22),

$$\begin{aligned} E\left[\frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right] \approx & \frac{w_q^2 E\left[\|\boldsymbol{k}_n^{(q)}\|_2^2\right]}{E\left[\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2\right]} \\ = & \frac{w_q^2 E\left[\|\boldsymbol{k}_n^{(q)}\|_2^2\right]}{\sum_{p=1}^{Q} w_p^2 E\left[\|\boldsymbol{k}_n^{(p)}\|_2^2\right]} \\ = & \frac{r_{q,n}\gamma^2}{\sum_{p=1}^{Q} r_{p,n}\gamma^2} = \frac{r_{q,n}}{r_n}, \end{aligned} \tag{4.24}$$

for all q. This leads to

$$\begin{aligned} E\left[1-\frac{\mu(2-\mu)}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{r_{q,n}}\right] = & 1-\frac{\mu(2-\mu)}{r_{q,n}}E\left[\frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right] \\ = & 1-\frac{\mu(2-\mu)}{r_{q,n}}\frac{r_{q,n}}{r_n} \\ = & 1-\frac{\mu(2-\mu)}{r_n}, \end{aligned} \tag{4.25}$$

which does not depend on q. □

4.2.3 Steady-state Error

The summation (4.17) with respect to q yields the following transient behavior of the entire error.

$$\begin{aligned} E\left[E_{\boldsymbol{z}_n}\left[\|\boldsymbol{z}_{n+1}\|_2^2\right]\right] = & \sum_{q=1}^{Q} E\left[E_{\boldsymbol{z}_n}\left[\|\boldsymbol{z}_{n+1}^{(q)}\|_2^2\right]\right] \\ = & E_{\boldsymbol{z}_n}\left[\|\boldsymbol{z}_n\|_2^2\right] - \frac{\mu(2-\mu)}{r_n}E_{\boldsymbol{z}_n}\left[\|\boldsymbol{z}_n\|_2^2\right]\sum_{q=1}^{Q}E\left[\frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right] \\ & + \mu^2\sum_{q=1}^{Q}E\left[\frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\frac{\nu_n^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right] \\ = & \left(1-\frac{\mu(2-\mu)}{r_n}\right)E_{\boldsymbol{z}_n}\left[\|\boldsymbol{z}_n\|_2^2\right] + \mu^2 E\left[\frac{\nu_n^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right], \end{aligned} \tag{4.26}$$

where the second equality is due to (4.16), and the last equality holds because

$$\sum_{q=1}^{Q}E\left[\frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right] = E\left[\sum_{q=1}^{Q}\frac{w_q^2\|\boldsymbol{k}_n^{(q)}\|_2^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right] = 1. \tag{4.27}$$

From (4.26), it holds that

$$\begin{aligned} E_{\boldsymbol{z}_n}\left[\|\boldsymbol{z}_n\|_2^2\right] = & \frac{r_n}{\mu(2-\mu)}\mu^2 E\left[\frac{\nu_n^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right] \\ = & \frac{\mu r_n}{2-\mu}E\left[\frac{\nu_n^2}{\|\boldsymbol{W}\boldsymbol{k}_n\|_2^2}\right] \end{aligned} \tag{4.28}$$

at the steady state. The above steady-state error depends on the choice of $\boldsymbol{W}$. In particular, if the weights w_q, $q = 1, 2, \cdots, Q$, become larger, the steady-state error becomes smaller even if the ratios among the weights are fixed. By considering the normalized error

$$\frac{E_{\boldsymbol{z}_n}\left[\|\boldsymbol{z}_n\|_2^2\right]}{E_{\boldsymbol{z}_n}\left[\|\boldsymbol{g}^*\|_2^2\right]} = \frac{E_{\boldsymbol{z}_n}\left[\|\boldsymbol{z}_n\|_2^2\right]}{E_{\boldsymbol{z}_n}\left[\|\boldsymbol{z}_n\|_2^2\right] + \|\boldsymbol{g}_n\|_2^2}, \tag{4.29}$$

such dependency on the amplitudes is eliminated. In addition, both of (4.23) and (4.29) are independent of the value of γ if the kernel weights are chosen by (4.22). Hereafter, $\gamma = 1$ without loss of generality.

Remark 4.1 (Relation to Batch Normalization)**.** *The case where $r_{q,n} = 1$ of the kernel weight* (4.22) *corresponds to the use of entrywise kernel weights. Imposing the entrywise weights is similar to the batch normalization, which has been popular in machine learning community [70–72]. The batch normalization aims to balance the scales of the entries of the observed input vector. In the kernel weights* (4.22)*, $\frac{E[\|\boldsymbol{k}_n^{(q)}\|_2^2]}{r_{q,n}}$ can be seen as a scale of the qth kernel because it holds that*

$$\frac{E\left[\|\boldsymbol{k}_n^{(q)}\|_2^2\right]}{r_{q,n}} = E\left[\frac{\|\boldsymbol{k}_n^{(q)}\|_2^2}{r_{q,n}}\right] \approx \int_{\mathbb{R}^L} E\left[\left(\kappa_q(\boldsymbol{u}_n, \boldsymbol{x})\right)^2\right] d\boldsymbol{x}, \tag{4.30}$$

for sufficiently large $r_{q,n}$. The essential difference between imposing the kernel weight (4.22) *and the batch normalization are twofold.*

- *The kernel weight does not consider the mean of the kernelized input, where the batch normalization includes the subtraction of the mean. Imposing the kernel weights is therefore equivalent to the batch normalization when the mean is zero.*
- *The kernel weight does not use the mini-batch in order to approximate the expectation. (In practice, the mini-batch can be used to implement the kernel weights.)*

4.3 Implementation of the Kernel Weights

4.3.1 Weight Estimation from Observations

In kernel adaptive filtering, the length $r_{q,n}$ of the qth kernelized input may change in time. In addition, $E\left[\|\boldsymbol{k}_n^{(q)}\|_2^2\right]$ is unavailable because the statistical properties of $\boldsymbol{k}_n^{(q)}$ are unknown. The expectation in the kernel weight (4.22) can be approximated by the sample mean

$$E\left[\frac{\|\boldsymbol{k}_n^{(q)}\|_2^2}{r_{q,n}}\right] \approx \frac{1}{n}\sum_{l=1}^{n} \frac{\|\boldsymbol{k}_l^{(q)}\|_2^2}{r_{q,l}}. \tag{4.31}$$

By the above approximation, the kernel weights can be updated recursively. Although an exponentially weighted average with a forgetting factor can be

used instead, the simple sample mean is solely focused in the present study. Under (4.31) with $\gamma = 1$, (4.22) is approximated by

$$w_{q,n} = \left(\frac{1}{n} \sum_{l=1}^{n} \frac{\|\boldsymbol{k}_l^{(q)}\|_2^2}{r_{q,l}} \right)^{-\frac{1}{2}}. \tag{4.32}$$

4.3.2 Implementation Based on Metric Design

Replacing the fixed weight matrix $\boldsymbol{W}$ in (4.4) to $\boldsymbol{W}_n$ which is time-varying, the following update is obtained.

$$\boldsymbol{g}_{n+1} = \boldsymbol{g}_n - \mu \frac{\boldsymbol{k}_n^\mathsf{T} \boldsymbol{W}_n \boldsymbol{g}_n - d_n}{\boldsymbol{k}_n^\mathsf{T} \boldsymbol{W}_n^2 \boldsymbol{k}_n} \boldsymbol{W}_n \boldsymbol{k}_n, \tag{4.33}$$

where

$$\boldsymbol{W}_n := \begin{bmatrix} w_{1,n} \boldsymbol{I}_{r_{1,n}} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & w_{2,n} \boldsymbol{I}_{r_{2,n}} & \cdots & \boldsymbol{O} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & w_{Q,n} \boldsymbol{I}_{r_{Q,n}} \end{bmatrix}. \tag{4.34}$$

Since (4.33) can be regarded as the update equation of the NLMS algorithm for the modified input vector $\boldsymbol{W}_n \boldsymbol{k}_n$, it is clear that the algorithm tracks the vector $\boldsymbol{g}_n^* = \boldsymbol{W}_n^{-1} \boldsymbol{h}^*$, so that the output of the filter to this modified input vector satisfies $(\boldsymbol{W}_n \boldsymbol{k}_n)^\mathsf{T} \boldsymbol{g}_n^* = \boldsymbol{k}_n^\mathsf{T} \boldsymbol{h}^*$. This, however, means that the target vector $\boldsymbol{g}_n^*$ changes as $\boldsymbol{W}_n$ changes, even if $\boldsymbol{h}^*$ is fixed. This may cause slow convergence of the coefficients which comes from strong time dependency of the kernel weights during the initial phase of adaptation. This dependency issue can be fixed by left-multiplying the both sides of (4.33) by $\boldsymbol{W}_n$, which leads to

$$\boldsymbol{h}_{n+1} = \boldsymbol{h}_n - \mu \frac{\boldsymbol{k}_n^\mathsf{T} \boldsymbol{h}_n - d_n}{\boldsymbol{k}_n^\mathsf{T} \boldsymbol{W}_n^2 \boldsymbol{k}_n} \boldsymbol{W}_n^2 \boldsymbol{k}_n, \tag{4.35}$$

with $\boldsymbol{h}_n = \boldsymbol{W}_n \boldsymbol{g}_n$. The above update tracks the original target vector $\boldsymbol{h}^*$, and hence is free from the dependency of the target vector on $\boldsymbol{W}_n$. Provided that $\boldsymbol{h}^*$ stays constant, it can then be guaranteed that the coefficient vector $\boldsymbol{h}_n$ approaches monotonically to $\boldsymbol{h}^*$ after each update in terms of the metric $\boldsymbol{W}_n^{-2}$ since the update equation in (4.35) can be interpreted as a relaxed projection in terms of the metric $\boldsymbol{W}_n^{-2}$. See [73–77] for the metric projection-based adaptive filtering algorithms. The entire MKNLMS algorithm equipped with the proposed metric design is presented in Algorithm 2. As a measure for the sparsification criterion, the novelty [32], the coherence [30], or the surprise [33] can be used.

4.3.3 Use of Linear Kernel

For the linear kernel, a fixed dictionary $\mathcal{D}_\mathrm{L} := \{\boldsymbol{e}_1^\mathsf{T}(\cdot), \boldsymbol{e}_2^\mathsf{T}(\cdot), \cdots, \boldsymbol{e}_L^\mathsf{T}(\cdot), \mathbb{1}\}$, where $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_L\}$ is the standard basis of the Euclidean space $\mathbb{R}^L$ and $\mathbb{1} : \mathbb{R}^L \mapsto \mathbb{R} : \boldsymbol{x} \mapsto 1$, can be considered since the dictionary $\mathcal{D}_\mathrm{L}$ completes a basis of an $L + 1$ dimensional Euclidean space. From (2.11), such a use of the linear kernel is actually the simultaneous use of two linear kernel functions

Algorithm 2 MKNLMS with Adaptive Kernel Weighting

0) $\mathcal{D}_{q,n} = \emptyset,\ \boldsymbol{h}_0^{(q)} = [\,],\ q = 1, 2, \cdots, Q,\ \mu \in (0, 2)$: step size.

1) Based on a sparsification criterion, decide whether to add the input $\kappa_q(\boldsymbol{u}_n, \cdot)$ to the dictionary $\mathcal{D}_{q,n}$ or not.

2) Vector augmentation:

$$\tilde{\boldsymbol{k}}_n^{(q)} = \begin{cases} \begin{bmatrix} \boldsymbol{k}_n^{(q)} \\ \kappa_q(\boldsymbol{u}_n, \boldsymbol{u}_n) \end{bmatrix} & \text{if } \mathcal{D}_q \text{ grows,} \\ \boldsymbol{k}_n^{(q)} & \text{otherwise,} \end{cases}$$

$$\tilde{\boldsymbol{h}}_n^{(q)} = \begin{cases} \begin{bmatrix} \boldsymbol{h}_n^{(q)} \\ 0 \end{bmatrix} & \text{if } \mathcal{D}_q \text{ grows,} \\ \boldsymbol{h}_n^{(q)} & \text{otherwise.} \end{cases}$$

3) $\tilde{\boldsymbol{k}}_n = \begin{bmatrix} \tilde{\boldsymbol{k}}_n^{(1)} \\ \tilde{\boldsymbol{k}}_n^{(2)} \\ \vdots \\ \tilde{\boldsymbol{k}}_n^{(Q)} \end{bmatrix},\ \tilde{\boldsymbol{h}}_n = \begin{bmatrix} \tilde{\boldsymbol{h}}_n^{(1)} \\ \tilde{\boldsymbol{h}}_n^{(2)} \\ \vdots \\ \tilde{\boldsymbol{h}}_n^{(Q)} \end{bmatrix}.$

4) $w_{q,n}^2 = \left(\frac{1}{n} \sum_{l=1}^{n} \frac{\|\boldsymbol{k}_l^{(q)}\|_2^2}{r_l^{(q)}} \right)^{-1},\ \forall q = 1, 2, \cdots, Q.$

5) $\boldsymbol{h}_{n+1} = \tilde{\boldsymbol{h}}_n - \mu \dfrac{\tilde{\boldsymbol{k}}_n^{\mathsf{T}} \tilde{\boldsymbol{h}}_n - d_n}{\tilde{\boldsymbol{k}}_n^{\mathsf{T}} \boldsymbol{W}_n^2 \tilde{\boldsymbol{k}}_n} \boldsymbol{W}_n^2 \tilde{\boldsymbol{k}}_n.$

$\kappa_{\mathrm{L}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^{\mathsf{T}} \boldsymbol{y}$ and $\kappa_{\mathrm{C}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^{\mathsf{T}} \boldsymbol{y} + 1$, with the corresponding dictionaries $\mathcal{D}_{\mathrm{L}} := \{\kappa_{\mathrm{L}}(\boldsymbol{e}_1, \cdot),\ \kappa_{\mathrm{L}}(\boldsymbol{e}_2, \cdot),\ \cdots, \kappa_{\mathrm{L}}(\boldsymbol{e}_L, \cdot)\}$ and $\mathcal{D}_{\mathrm{C}} := \{\kappa_{\mathrm{C}}(\boldsymbol{0}, \cdot)\}$, respectively. An application of the above kernel functions and dictionaries to the weight (4.22) yields the following weights:

$$w_{\mathrm{L}} = \left(E\left[\frac{\|\boldsymbol{u}_n\|_2^2}{L} \right] \right)^{-\frac{1}{2}},\ w_{\mathrm{C}} = 1. \tag{4.36}$$

The above weights can be approximated at each iteration by

$$w_{\mathrm{L},n} = \left(\frac{1}{n} \sum_{l=1}^{n} \frac{\|\boldsymbol{u}_l\|_2^2}{L} \right)^{-\frac{1}{2}}, \tag{4.37}$$

and

$$w_{\mathrm{C},n} = 1. \tag{4.38}$$

4.3.4 Calculation of Kernel Weights (4.22) under Gaussian Input

Here the kernel weights for the linear kernel and a Gaussian kernel at the steady state are calculated in the case that the input vector is under the i.i.d. Gaussian

distribution $\mathcal{N}(0, \sigma_u^2)$. The dictionary size r for the Gaussian kernel is constant at the steady state and then the kernelized input is given as follows:

$$\boldsymbol{k}_n = \begin{bmatrix} \boldsymbol{u}_n \\ \boldsymbol{k}_n^G \end{bmatrix} \in \mathbb{R}^{L+r}. \tag{4.39}$$

Case 1: Linear kernel (Sec. 4.3.3)
Since $w_{\mathrm{C}} = 1$ is fixed, it is sufficient to calculate w_{L}.

$$E\left[\frac{\|\boldsymbol{u}_n\|_2^2}{L}\right] = \frac{1}{L}\sum_{i=1}^{L}(2\pi\sigma_u^2)^{-\frac{L}{2}} \int_{-\infty}^{\infty} u_i^2 \exp\left(-\frac{\|\boldsymbol{u}\|_2^2}{2\sigma_u^2}\right) d\boldsymbol{u}. \tag{4.40}$$

For all i, the integral part can be rewritten as follows:

$$\begin{aligned} &\int_{-\infty}^{\infty} u_i^2 \exp\left(-\frac{\|\boldsymbol{u}\|_2^2}{2\sigma_u^2}\right) d\boldsymbol{u} \\ =&\left[\int_{-\infty}^{\infty} u_i^2 \exp\left(-\frac{u_i^2}{2\sigma_u^2}\right) du_i\right] \prod_{j\neq i} \int_{-\infty}^{\infty} \exp\left(-\frac{u_j^2}{2\sigma_u^2}\right) du_j. \end{aligned} \tag{4.41}$$

The equality holds because the input follows the i.i.d. Gaussian distribution. Since

$$\int_{-\infty}^{\infty} \exp\left(-\frac{u_j^2}{2\sigma_u^2}\right) du_j = \sqrt{2\pi\sigma_u^2} \tag{4.42}$$

for all j, the following holds.

$$\begin{aligned} E\left[\frac{\|\boldsymbol{u}_n\|_2^2}{L}\right] &= \frac{1}{L}\sum_{i=1}^{L}(2\pi\sigma_u^2)^{-\frac{1}{2}} \int_{-\infty}^{\infty} u_i^2 \exp\left(-\frac{u_i^2}{2\sigma_u^2}\right) du_i \\ &= \frac{1}{L}\sum_{i=1}^{L} \sigma_u^2 = \sigma_u^2. \end{aligned} \tag{4.43}$$

Then, by (4.36),

$$w_{\mathrm{L}} = \frac{1}{\sigma_u}. \tag{4.44}$$

Case 2: Gaussian kernel defined in (2.10)

$$\begin{aligned} E\left[\frac{\|\boldsymbol{k}_n^G\|_2^2}{r}\right] =& \frac{1}{r}(2\pi\sigma_u^2)^{-\frac{L}{2}}(2\pi\sigma^2)^{-L} \\ &\times \sum_{i=1}^{r} \int_{-\infty}^{\infty} \exp\left(-\frac{\|\boldsymbol{u}-\boldsymbol{x}_i\|_2^2}{\sigma^2}\right) \exp\left(-\frac{\|\boldsymbol{u}\|_2^2}{2\sigma_u^2}\right) d\boldsymbol{u}, \end{aligned} \tag{4.45}$$

where $\mathcal{D}_{\mathrm{G}} = \{\kappa_{\mathrm{G}}(\boldsymbol{x}_1, \cdot), \kappa_{\mathrm{G}}(\boldsymbol{x}_2, \cdot), \cdots, \kappa_{\mathrm{G}}(\boldsymbol{x}_r, \cdot)\}$ is the dictionary. As in Case 1,

the integral part can be calculated as follows:

$$\begin{aligned}
&\int_{-\infty}^{\infty} \exp\left(-\frac{\|\boldsymbol{u}-\boldsymbol{x}_i\|_2^2}{\sigma^2}\right)\exp\left(-\frac{\|\boldsymbol{u}\|_2^2}{2\sigma_u^2}\right)d\boldsymbol{u}\\
=&\int_{-\infty}^{\infty} \exp\left(-\left(\frac{1}{\sigma^2}+\frac{1}{2\sigma_u^2}\right)\|\boldsymbol{u}\|_2^2+\frac{2\boldsymbol{u}^\mathsf{T}\boldsymbol{x}_i}{\sigma^2}-\frac{\|\boldsymbol{x}_i\|_2^2}{\sigma^2}\right)d\boldsymbol{u}\\
=&\exp\left(-\frac{\|\boldsymbol{x}_i\|_2^2}{2\sigma_u^2+\sigma^2}\right)\\
&\times\int_{-\infty}^{\infty}\exp\left(-\left(\frac{1}{\sigma^2}+\frac{1}{2\sigma_u^2}\right)\left\|\boldsymbol{u}-\frac{2\sigma_u^2}{2\sigma_u^2+\sigma^2}\boldsymbol{x}_i\right\|_2^2\right)d\boldsymbol{u}\\
=&\left(\frac{2\pi\sigma_u^2\sigma^2}{2\sigma_u^2+\sigma^2}\right)^{\frac{L}{2}}\exp\left(-\frac{\|\boldsymbol{x}_i\|_2^2}{2\sigma_u^2+\sigma^2}\right).
\end{aligned}\tag{4.46}$$

Then it holds that

$$\begin{aligned}
E\left[\frac{\|\boldsymbol{k}_n^G\|_2^2}{r}\right]=&\frac{1}{r}(2\pi\sigma^2)^{-L}\left(\frac{\sigma^2}{2\sigma_u^2+\sigma^2}\right)^{\frac{L}{2}}\\
&\times\sum_{i=1}^{r}\exp\left(-\frac{\|\boldsymbol{x}_i\|_2^2}{2\sigma_u^2+\sigma^2}\right),
\end{aligned}\tag{4.47}$$

and the kernel weight is calculated as

$$w_{\mathrm{G}}=\sqrt{\frac{r(2\pi\sigma^2)^L}{\left(\frac{\sigma^2}{2\sigma_u^2+\sigma^2}\right)^{\frac{L}{2}}\sum_{i=1}^{r}\exp\left(-\frac{\|\boldsymbol{x}_i\|_2^2}{2\sigma_u^2+\sigma^2}\right)}}.\tag{4.48}$$

For a sufficiently large r, the following approximation can be used.

$$\begin{aligned}
\frac{1}{r}\sum_{i=1}^{r}\exp\left(-\frac{\|\boldsymbol{x}_i\|_2^2}{2\sigma_u^2+\sigma^2}\right)\approx&\int_{\mathbb{R}^L}\exp\left(-\frac{\|\boldsymbol{x}\|_2^2}{2\sigma_u^2+\sigma^2}\right)d\boldsymbol{x}\\
=&\left(2\pi(2\sigma_u^2+\sigma^2)\right)^{\frac{L}{2}}.
\end{aligned}\tag{4.49}$$

Substituting (4.49) into (4.48) leads to

$$w_{\mathrm{G}}=(2\pi\sigma^2)^{\frac{L}{4}}.\tag{4.50}$$

4.4 Numerical Experiments

In this section, numerical experiments related to system estimation and time series prediction are presented. The following algorithms are studied in those experiments:

1. The OMKR algorithm [43, 44] (Algorithm 1),
2. The MKNLMS algorithm [47, 48] with unweighted kernels,*i.e.,* $w_q = 1/Q$ for all q,

Table 4.1: Computational costs for Gaussian kernels

	# Additions	# Multiplications
OMKR (hedge)	$(2Lr_n + 1)Q - 1$	$((L + 4)r_n + 3)Q + 1$
MKNLMS (manual)	$(2L + 2)r_n - 1$	$(L + 3)r_n + 1$
MKNLMS (proposed)	$(2L + 3)r_n - 1$	$(L + 4)r_n + 3Q + 1$

3. The MKNLMS algorithm with weighted kernels, with manually tuned kernel weights which achieve the smallest MSE, and

4. The MKNLMS algorithm with weighted kernels, whose weights are tuned by the proposed automatic weighting.

As a measure for the sparsification criterion of the MKNLMS algorithm, the coherence [30] is employed. The best preset kernel weights for the MKNLMS algorithm have been found by grid search. In advance of the numerical results, see the computational costs of the above algorithms in Table 4.1, where L, r_n, and Q are the input length, the dictionary size, and the number of kernels, respectively.

4.4.1 Experiments with Toy Data

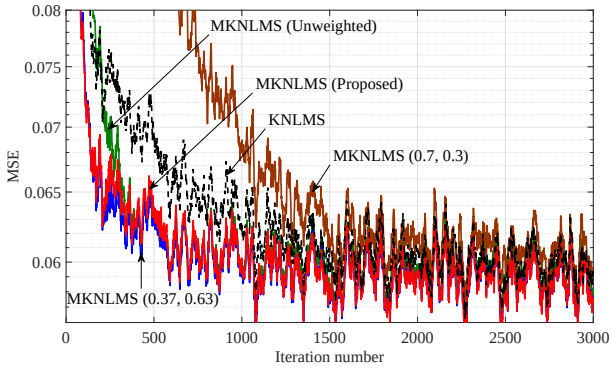
Experiment 1a: The efficacy of kernel weights is investigated by a simple regression task. Three preset weights for the MKNLMS algorithm are tested as well as the proposed automatic weighting. In addition, the NLMS algorithm [78] and the KNLMS algorithm [30] are also compared with the MKNLMS algorithms. The NLMS and KNLMS algorithms correspond to the cases using the linear kernel and the Gaussian kernel, respectively. The target system is given as follows:

$$\psi(u) = 0.6u + \exp\left(-\frac{(u - 0.2)^2}{2 \times 0.5^2}\right). \tag{4.51}$$

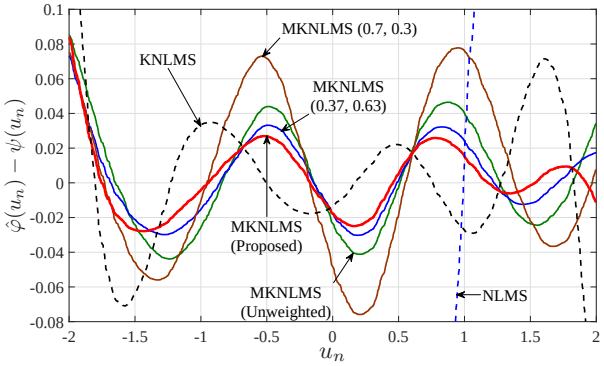
The regression task is to estimate $\psi(u)$ by observing the current input $u_n \in \mathbb{R}$ $(L = 1)$ and the noisy output $d_n = \psi(u_n) + \nu_n$, where u_n is randomly generated under the uniform distribution in the interval $(-2,\ 2)$ and ν_n is the additive Gaussian noise satisfying SNR = 10 dB. The initial settings for this experiment are given in Table 4.2. The averaged results of 200 trials are presented in Figure 4.1. Note that the NLMS algorithm is not shown in Figure 4.1(a) because its MSE is excessively large. Figure 4.1(b) shows the differences $\hat{\varphi}(u_n) - \psi(u_n)$, where each $\hat{\varphi}(u_n)$ is the estimated function by an algorithm. The kernel weights in Figure 4.1(d) are normalized so that the sum of the weights for the linear and Gaussian kernels becomes one.

Experiment 1b: This experiment shows that the proposed technique generates the weight sequence convergent to the weights derived in Section 4.2.2. The target system is given as

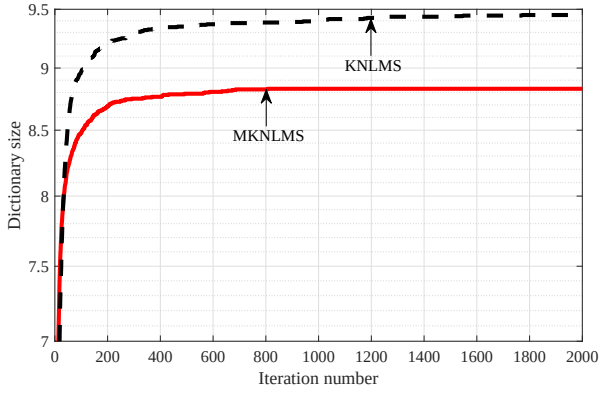
$$d_n = \boldsymbol{\beta}_1^{\mathsf{T}} \boldsymbol{u}_n + 30 \exp\left(-\frac{\|\boldsymbol{u}_n - \boldsymbol{\beta}_2\|_2^2}{2 \times 0.5^2}\right) - 10 \exp\left(-\frac{\|\boldsymbol{u}_n - \boldsymbol{\beta}_3\|_2^2}{2 \times 0.5^2}\right), \tag{4.52}$$



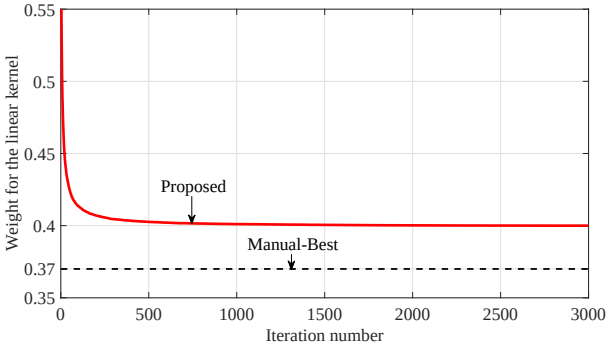
(a)



(b)



(c)

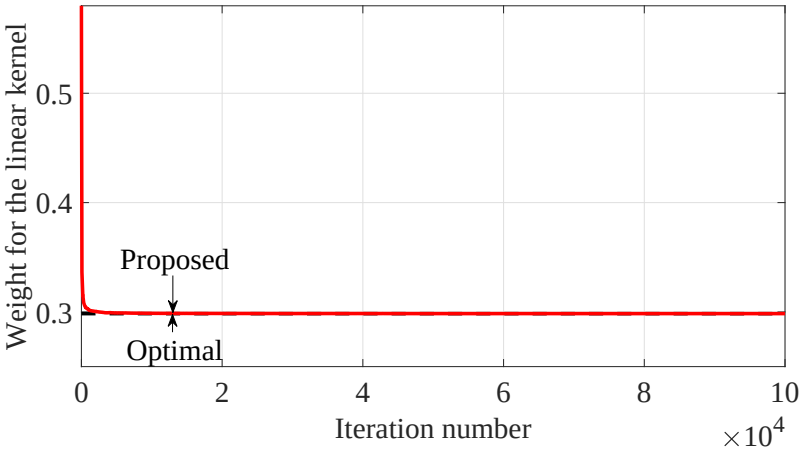


(d)

Figure 4.1: Results of Experiment 1a.

Table 4.2: Experiment 1a - Initial settings

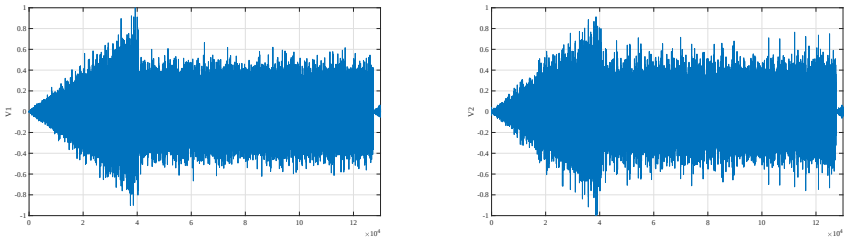
Step size	$\mu = 0.1$
Kernel parameter	$\sigma = 0.5$
Coherence threshold	$\delta = 0.8$



(a)

(b)

Figure 4.2: Weight evolution with Gaussian input(Experiment 1b.)



(a) Input (b) Output

Figure 4.3: Silverbox dataset for Experiment 2a.

Table 4.3: Experiment 2a - Initial settings

Step size	$\mu = 0.7$
Kernel parameter	$\sigma = 0.9$
Discounting parameter (OMKR)	$\beta = 0.5$
Coherence threshold (MKNLMS)	$\delta = 0.9$

where the input $\boldsymbol{u}_n \in \mathbb{R}^3$ is randomly generated under the i.i.d. Gaussian distribution $\mathcal{N}(0, 0.125)$ and $\boldsymbol{\beta}_1 = [0.5, 0.5, 0]^\mathsf{T}$, $\boldsymbol{\beta}_2 = [0.5, 0, 0.5]^\mathsf{T}$, $\boldsymbol{\beta}_3 = [0, 0.5, 0.5]^\mathsf{T}$. The target system is estimated using the linear kernel given in Section 4.3.3 and the Gaussian kernel with the kernel parameter $\sigma = 0.5$ and with the fixed dictionary

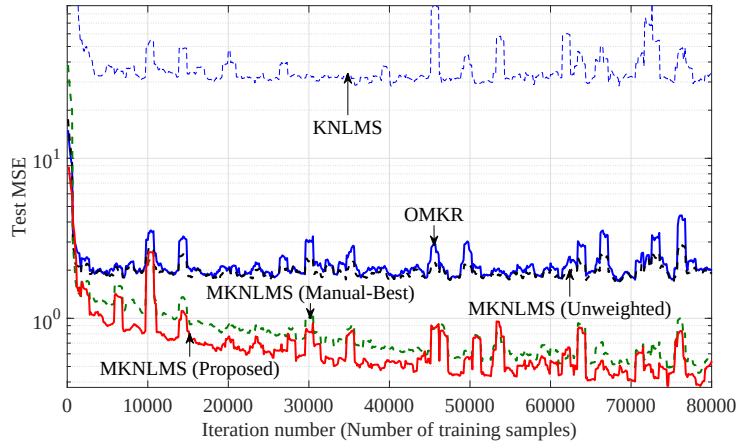
$$\mathcal{D}_G := \{\kappa_\mathrm{G}(\boldsymbol{e}_1, \cdot), \kappa_\mathrm{G}(\boldsymbol{e}_2, \cdot), \kappa_\mathrm{G}(\boldsymbol{e}_3, \cdot)\}, \tag{4.53}$$

where κ_G and $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_3$ are the Gaussian kernel defined in (2.10) and the standard basis defined in Section 4.3.3, respectively. Figure 4.2 shows the behavior of the kernel weight for the linear kernel. If the input distribution and fixed dictionaries are known, the kernel weights at the steady state can be calculated by (4.22). See Section 4.3.4 for detailed calculation for Gaussian input. Under the normalization, the calculated weight is plotted with the dashed line, labeled as *Optimal*, while the empirically obtained weight is plotted with the red solid line, labeled as *Proposed* in Figure 4.2(a).

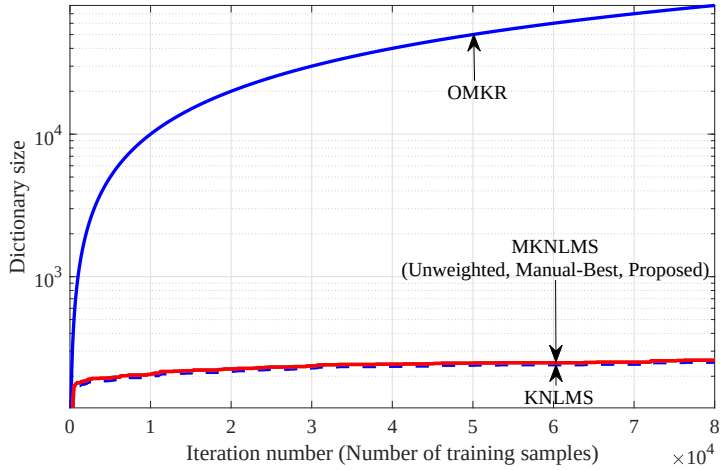
4.4.2 Experiments with Real Data - System Estimation

Experiment 2a: Silverbox dataset [79] is used in this experiment. The dataset consists of the input and output data shown in Figure 4.3. The first 40000 pairs of the input and the output, the triangular part, are used as the test data. The next 80000 pairs, the rectangular part, are used as the training data. Denoting the nth values of the input data and the output data by u_n and d_n, respectively, the observed values $u_n, u_{n-1}, u_{n-2}, u_{n-3}$ and $d_{n-1}, d_{n-2}, d_{n-3}$ as the input of the algorithms at the nth iteration, and hence $L = 7$. The initial settings for this experiment are given in Table 4.3. The results of Experiment 2a are shown in Figure 4.4.

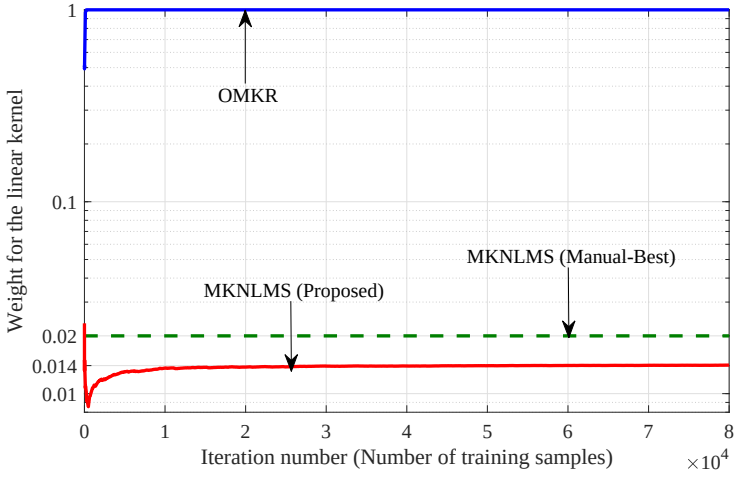
Experiment 2b: A subset of the Human Sensing Consortium challenge dataset



(a)



(b)

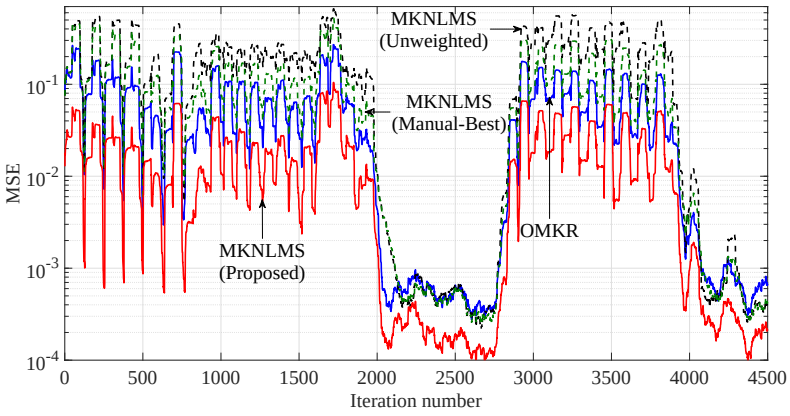


(c)

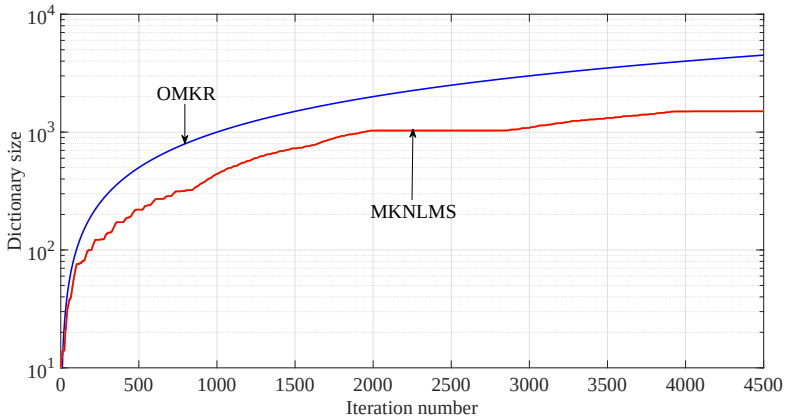
Figure 4.4: Results of Experiment 2a.

Table 4.4: Experiment 2b - Initial settings

Step size	$\mu = 0.01$
Kernel parameter	$\sigma = 0.2$
Discounting parameter (OMKR)	$\beta = 0.5$
Coherence threshold (MKNLMS)	$\delta = 0.2$



(a)



(b)

Figure 4.5: Results of Experiment 2b.

Table 4.5: Experiment 3a - Initial settings

Step size	$\mu = 0.1$
Kernel parameters	$\sigma^{(1)} = 2,\ \sigma^{(2)} = 1,$ $\sigma^{(3)} = 2^{-1},\ \sigma^{(4)} = 2^{-2},$ $\sigma^{(5)} = 2^{-3},\ \sigma^{(6)} = 2^{-4}$
Discounting parameter (OMKR)	$\beta = 0.5$
Coherence threshold (MKNLMS)	$\delta = 0.6$

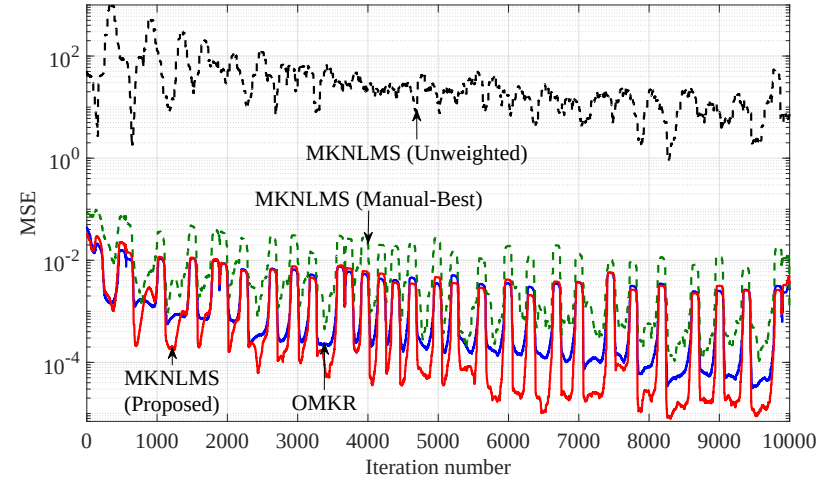
is used for this experiment. (For the detail, visit *http://hasc.jp/hc2011.*) This dataset has been obtained by three-dimensional sensor attached on a person who moves in various patterns, and hence the system to be estimated is non-stationary. The input is a vector of length $L = 8$ of which the entries are three-dimensional coordinates (x, y, z) of the recent three samples excluding the present z, which is used as the output. One linear kernel and one Gaussian kernel are used with the initial parameters given in Table 4.4.

4.4.3 Experiments with Real Data - Time Series Prediction

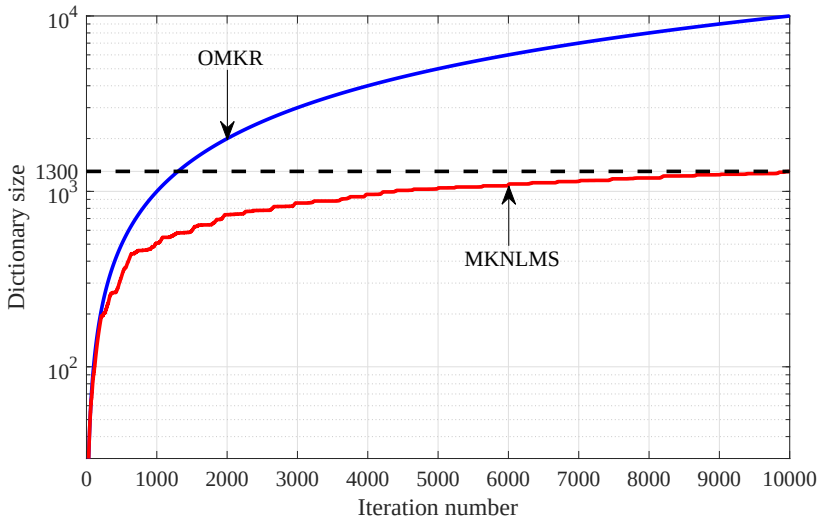
Experiment 3a: The Santa Fe laser dataset [80] is given as the time series data to be estimated. At each time, the next value is predicted by the algorithms given the recent nine values as input, *i.e.*, $L = 9$. The initial settings for this experiment are given in Table 4.5. As seen in Table 4.5, six Gaussian kernels are used for this task. Figure 4.6 shows the results of the experiment. In addition, effect of the number of kernels on performance of the proposed technique has been examined with the same dataset. See Figure 4.7 and Figure 4.8. The setting for $Q = 6$ is the same with that in Table 4.5. For $Q = 3$ and $Q = 9$, the maximal value of $\sigma^{(q)}$ (Figure 4.7) or the minimal value (Figure 4.8) is fixed as that of Table 4.5. It is seen that, when the Gaussian kernels are used, performance of the proposed technique mainly depends on the minimal kernel parameter, rather than the number of Gaussian kernels.

Experiment 3b: *Buzz prediction on social media* dataset [81] is used for this experiment and Experiment 3c. The dataset is obtained from two social media: one is Tom's hardware and the other is Twitter. As a characteristic of social media, the dataset has been considered to be non-stationary. For this experiment, the data from Tom's hardware is used. The input length is $L = 96$ and the other initial settings are given in Table 4.6. Here one linear kernel and four Gaussian kernels are used for the prediction. See Figure 4.9 for the results of the experiment.

Experiment 3c: This experiment uses the Twitter dataset, which has been considered to be less stationary than the data from Tom's hardware. The input length is $L = 77$. The linear kernel is not used for this dataset, but six Gaussian kernels are employed as shown in Table 4.7. The dictionary size of the OMKR algorithm is here limited to 2600, *i.e.*, the maximal dictionary size of the MKNLMS algorithm as Figure 4.10(b), to reduce computational costs.

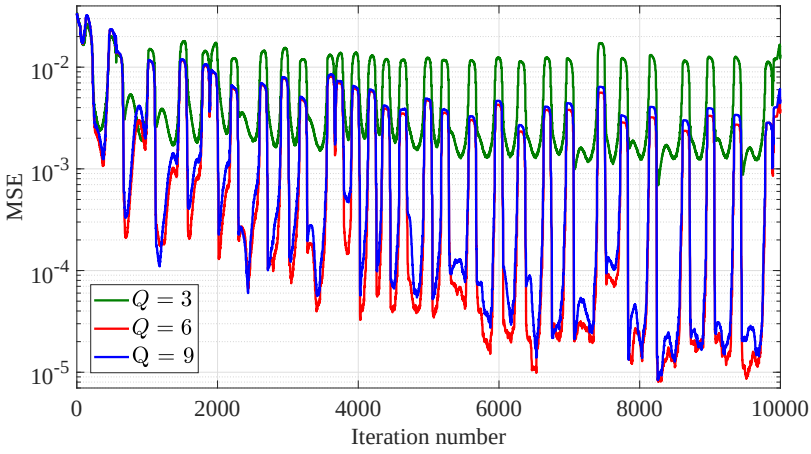


(a)

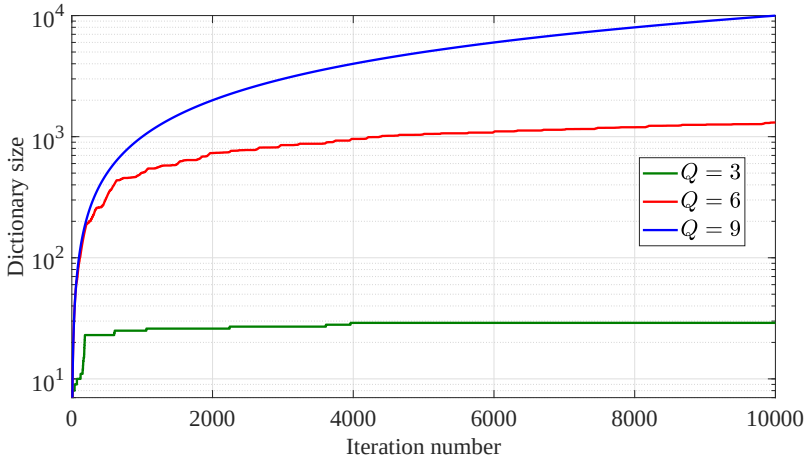


(b)

Figure 4.6: Results of Experiment 3a.

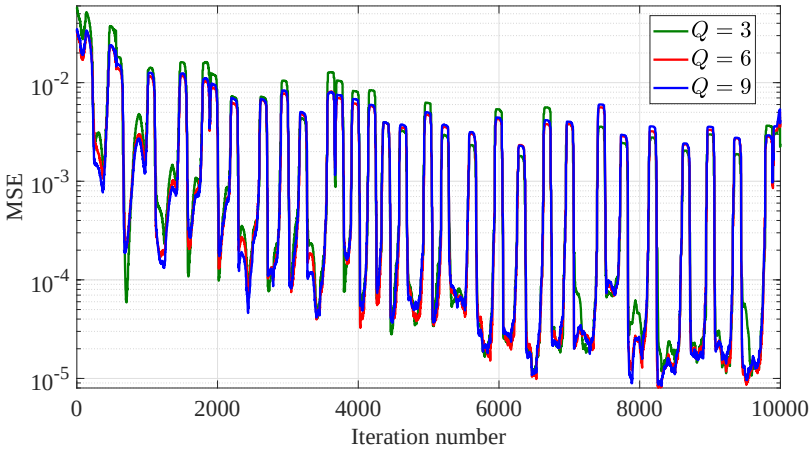


(a)

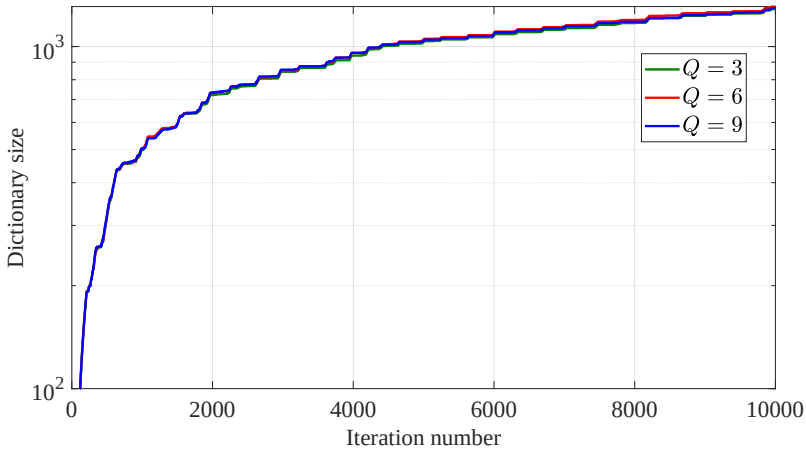


(b)

Figure 4.7: Effect of the number of kernels ($\max_q \sigma^{(q)} = 2$).



(a)

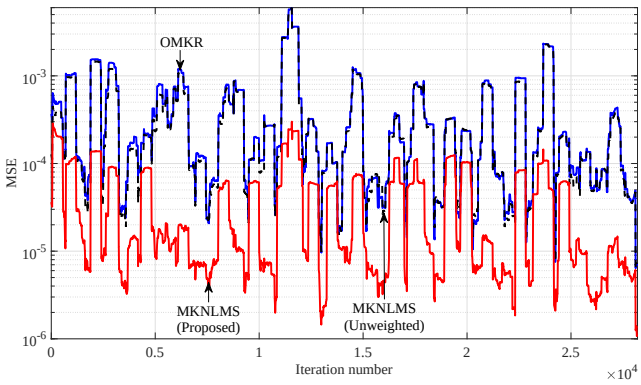


(b)

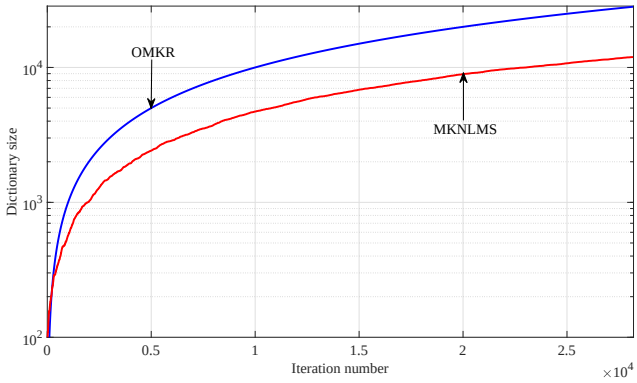
Figure 4.8: Effect of the number of kernels ($\min_q \sigma^{(q)} = 2^{-4}$).

Table 4.6: Experiment 3b - Initial settings

Step size	$\mu = 0.5$
Kernel parameters	$\sigma^{(1)} = 2^{-2},\ \sigma^{(2)} = 2^{-3},$ $\sigma^{(3)} = 2^{-4},\ \sigma^{(4)} = 2^{-5}$
Discounting parameter (OMKR)	$\beta = 0.5$
Coherence threshold (MKNLMS)	$\delta = 0.5$



(a)

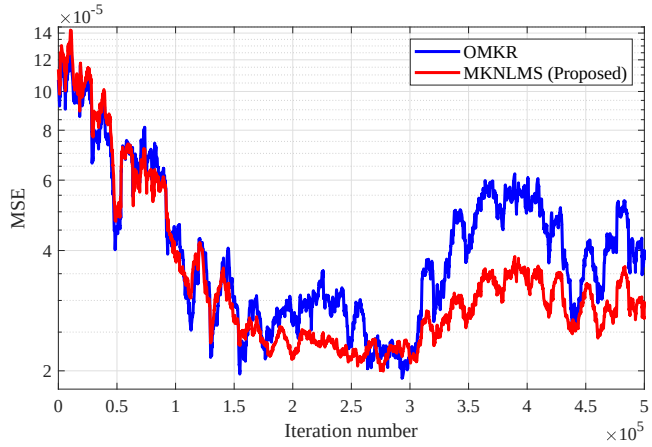


(b)

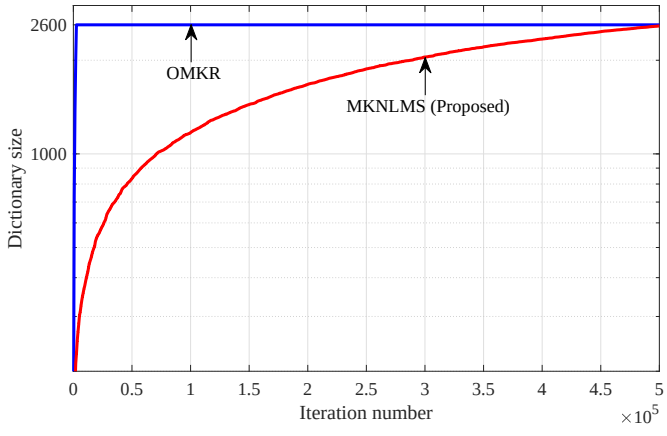
Figure 4.9: Results of Experiment 3b.

Table 4.7: Experiment 3c - Initial settings

Step size	$\mu = 0.01$
Kernel parameters	$\sigma^{(1)} = 2,\ \sigma^{(2)} = 1,$ $\sigma^{(3)} = 2^{-1},\ \sigma^{(4)} = 2^{-2},$ $\sigma^{(5)} = 2^{-3},\ \sigma^{(6)} = 2^{-4}$
Discounting parameter (OMKR)	$\beta = 0.5$
Coherence threshold (MKNLMS)	$\delta = 0.0001$



(a)



(b)

Figure 4.10: Results of Experiment 3c.

4.4.4 Discussions on the Numerical Results

Proposed technique versus preset kernel weights:

In Figures 4.1(a) and 4.1(b), it is shown that (i) the MKNLMS algorithm with inappropriate kernel weights can be worse than the KNLMS algorithm with a single kernel by the case of $(0.7, 0.3)$, and (ii) the proposed kernel weighting technique achieves a similar performance with the carefully tuned kernel weights $(0.37, 0.63)$. The superior performance of the proposed technique to the manual weighting is also shown in other experiments for system estimation (Experiments 2a and 2b) and time series prediction (Experiments 3a), which use real data.

According to Table 4.1, the proposed automatic weighting requires r_n times and $r_n + 3Q$ times of additional addition and multiplication, respectively. Compared with the computational costs of the MKNLMS algorithm itself, the additional calculations for improving the performance are affordable.

MKNLMS with the proposed technique versus OMKR:

In Figure 4.4(a), the MKNLMS algorithm with the proposed technique achieves both the fastest convergence and the smallest error, while the error of the OMKR algorithm decreases fast at early iterations at the price of higher steady-state errors. Figure 4.4(c) shows that the kernel weight for the linear kernel used in the OMKR algorithm converges to one approximately. This implies the Gaussian kernel is almost inactive and the OMKR algorithm fails to achieve a good balance between the linear and Gaussian kernels.

The MKNLMS algorithm with the proposed technique achieves a better MSE than the OMKR algorithm also in the other experiments, using smaller dictionaries. In Figure 4.5(a), the MKNLMS algorithm with the proposed technique shows the best tracking ability in estimating the non-stationary system (See the 2000–2500th iterations). In the experiments of time series prediction, the OMKR algorithm achieves similar MSEs with those of the MKNLMS algorithm with preset weights. (See Figures 4.10(a) and 4.9(a).)

From the viewpoint of the computational costs, the MKNLMS algorithms are more efficient than the OMKR algorithm. Focusing on the dictionary size r_n, the computational costs of the OMKR algorithm are nearly Q times larger than those of the MKNLMS algorithm even if the same dictionaries are used at every iteration.

Behaviors of kernel weights:

As seen in Figure 4.2, in which the dictionaries are fixed, the kernel weights set by the proposed technique converge to the values given in (4.22). In Figures 4.1(d) and 4.4(c), on the other hand, the kernel weight updated by the proposed technique converges to a different value from the manually tuned one. This is because the kernel weights (4.22) depend on the sizes of the dictionaries, which change in time. The proposed technique efficiently tracks the time-varying weights, thereby achieving an even better performance than the manual tuning which is optimized for the entire behaviors. Note that the proposed weights are also different from those of the OMKR algorithm, because the former depends on the kernelized input, whereas the latter depends on the kernel-wise errors.

4.5 Concluding Remarks

In this chapter, a recursive kernel weighting technique for multikernel adaptive filtering has been proposed. To deal with the coefficient error, which is unknown in practice, the case when it is a spherically distributed random variable has been considered. The transient behavior of the expected partial coefficient error then depends on the corresponding kernel weight. From the transient behavior, a kernel weight setting which achieves a uniform convergence rate for all the kernel-wise coefficient errors has been derived. A recursive formula to implement the derived kernel weights has also been presented, by the approximation using the simple sample mean. As the time-varying kernel weights make the adaptive algorithm track the corresponding time-varying solution, the derived weights have been implemented through the corresponding metric design, which makes the solution time-invariant. Finally, the numerical results have shown the efficacy of the proposed technique. The proposed technique leads to the better performance than the manually tuned kernel weights in both senses of the steady-state MSE and the convergence rate. Compared with the OMKR algorithm, the proposed approach has achieved a lower MSE, with smaller dictionaries. For the fixed dictionaries, the proposed technique has been shown to yield eventually the kernel weights derived experimentally based on the transient behavior. The kernel weights tuned by the proposed technique depend on the dictionaries, and are thus different from the manually tuned ones in general. The proposed technique will serve as a baseline for solving the kernel weighting problem, and exploiting the full potential of multikernel adaptive filtering.

Chapter 5

General Conclusion

This thesis has presented a nonconvex regularization for sparse optimization and a kernel weight design for multikernel adaptive filtering. In Chapter 3, both in overdetermined and underdetermined systems, the existence of a critical-point path which connects the origin and the sparsest OLS solution has been proved. This implies that ℓ_p-regularization ($0 < p < 1$) does yield sparsest solutions of a least squares problem. In addition, in an underdetermined system, a necessary condition for the existence of critical-point path emerging from an OLS solution has been shown. This implies that only few nonsparse OLS solution can be obtained by solving an ℓ_p-regularized least squares problem. The theoretical results given in Chapter 3 will provide useful insights for developing an efficient sparsity-aware learning algorithm using the ℓ_p-regularization. In recent years, nonconvex regularizers other than the ℓ_p-norm have also been studied [82–89]. Some of them have been reported to preserve the convexity of the regularized cost function [82,86,88,89], and the convexity ensures that one can achieve global minima. Although the ℓ_p-norm does not enjoy such a property, considering a critical-point path can be helpful for the other nonconvex regularizers. In Chapter 4, a kernel weight design equalizing the convergence rates of kernel-wise coefficient errors has been proposed. The convergence rates have been considered in the case that the coefficient error vector is a spherically distributed random variable. The kernel weights derived by equalizing the convergence rates have been implemented into the MKNLMS algorithm using the time-varying metric design corresponding to the weights. The numerical results have shown the efficacy of the proposed technique, compared to both of the manually tuned kernel weights and the OMKR algorithm. To derive the above results, we have focused on regression, which is only a major task related to online learning. For the other tasks, the results of Chapter 4 can be a baseline when one tunes kernel weights of multiple kernels. Both results will become sound bases for developing an online learning algorithm which utilizes multiple kernels appropriately, while maintaining its dictionary in an affordable size.

There are many topics beyond the sight of this thesis. Development of a sparsity-aware online learning algorithm exploiting multiple kernels may be only a small part of them. The below can be just the next step which are based on the insights from this thesis.

- Since the existence of a critical-point path connecting the origin and the

sparsest OLS solution has been proven, now a method for achieving the sparsest OLS solution using the ℓ_p-regularization must be developed. Although a greedy approach has been presented in [23], it is not guaranteed to obtain a path connecting the origin and the sparsest OLS solution.

- The soft thresholding has already been used for dictionary sparsification [38]. When a weighted soft thresholding, which virtually realizes the ℓ_p-minimization, is employed instead of the plain soft thresholding, the results in Chapter 4 will lead to an appropriate tuning of the threshold which coincides with the regularization parameter λ.

- The kernel weights can be applied to classification tasks. Since the loss function is not the MSE in this case, the derivation of appropriate weights will be also different from that given in Chapter 4.

Bibliography

[1] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

[2] D. L. Donoho, and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, Aug. 1994.

[3] D. L. Donoho, "De-noising by soft thresholding," *IEEE Trans. Inform. Theory*, vol. 41, no. 3, pp. 713–627, May 1995.

[4] S. S. Chen, D. L. Donoho and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.

[5] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Royal Statistical Society. Series B*, vol. 58, no. 1, pp. 267–288, 1996.

[6] Y. C. Pati, R. Rezaiifar and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. Asilomar*, 1993.

[7] J. Tropp and A. C. Gilbert, "Signal recovery from partial information via orthogonal matching pursuit," *IEEE Trans. Inform. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.

[8] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, vol. 26, pp. 301–321, 2009.

[9] D. Needell and R. Vershynin, "Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit," *IEEE J. Selected Topics in Signal Process.*, vol. 4, no. 2, pp. 310–316, 2010.

[10] D. L. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition," *IEEE Trans. Inform. Theory*, vol. 47, no. 7, pp. 2845–2862, Nov. 2001.

[11] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Inform. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[12] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[13] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, Mar. 2008.

[14] M. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*, Cambridge, 2012.

[15] M. R. Osborne, B. Presnell and B. A. Turlach, "A new approach to variable selection in least squares problems," *IMA Journal of Numer. Anal.*, vol. 20, pp. 389–404, 2000.

[16] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[17] D. L. Donoho and Y. Tsaig, "Fast solution of ℓ_1-norm minimization problems when the solution may be sparse," *IEEE Trans. Inform. Theory*, vol. 54, no. 11, pp. 4789–4812, Nov. 2008.

[18] R. Chartrand and V. Staneva, "Restricted isometry properties and nonconvex compressive sensing," *Inverse Problem*, vol. 24, pp. 1–14, 2008.

[19] S. Foucart and M.-J. Lai, "Sparsest solutions of underdetermined linear systems via ℓ_q-minimization for $0 < q \leq 1$," *Appl. Comput. Harmon. Anal.*, vol. 26, pp. 395–407, 2009.

[20] Z. Xu, H. Zhang, Y. Wang, X. Chang and Y. Liang, "L1/2 regularization," *Science China*, vol. 53, no. 6, pp. 1159–1169, June 2010.

[21] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *Proc. 33rd Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2008, pp. 3869–3872.

[22] Z. Xu, X. Chang, F. Xu and H. Zhang, "$L_{1/2}$ regularization: A thresholding representation theory and a fast solver," *IEEE Trans. Neural Networks and Learning Systems*, vol. 23, no. 7, pp. 1013–1027, July 2012.

[23] M. Yukawa and S. Amari, "ℓ_p-regularized least squares $(0 < p < 1)$ and critical path," *IEEE Trans. Inform. Theory*, vol. 62, no. 1, pp. 488–502, Jan. 2016.

[24] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, 2002.

[25] A. Sayed, *Adaptive Filters*, Wiley-IEEE Press, 2008.

[26] W. Liu, J. C. Príncipe and S. Haykin, *Kernel Adaptive Filtering*, Wiley, Hoboken, NJ, USA, 2010.

[27] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least squares algorithm," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[28] W. Liu and J. Principe, "Kernel affine projection algorithms," *EURASIP Jour. Adv. Signal Process.*, vol. 2008, pp. 1–12, 2008.

[29] W. Liu, P. P. Pokharel and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 543–554, Feb. 2008.

[30] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. signal Process.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[31] Z. Wang, K. Crammer and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training," *Journal of Machine Learning Research*, vol. 13, no. 100, pp. 3103–3131, 2012.

[32] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, 1991.

[33] W. Liu, I. Park and J. Principe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Trans. Neural Network*, vol. 20, no. 12, pp. 1950–1960, Dec. 2009.

[34] S. V. Vaerenbergh, J. Fernandez-Bes and V. Elvira, "On the relationship between online gaussian process regression and kernel least mean squares algorithms," in *IEEE Int. Workshop MLSP*,, 2016.

[35] L. Csató and M. Opper, "Sparse online gaussian processes," *Neural Comput.*, vol. 14, no. 3, pp. 641–668, 2002.

[36] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inform. Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[37] B. Chen, S. Zhao, P. Zhu and J. C. Príncipe, "Quantized kernel least mean square algorithm," *IEEE Trans. Neural Netw. Learn. Sys.*, vol. 23, no. 1, pp. 22–32, Jan. 2012.

[38] W. Gao, J. Chen, C. Richard, and J. Huang, "Online dictionary learning for kernel LMS," *IEEE Trans. signal Process.*, vol. 62, no. 11, pp. 2765–2777, Jun. 2014.

[39] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc. Int. Conf. Mach. Learn. ACM*, 2004.

[40] G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semi-definite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, 2004.

[41] M. Varma and B. R. Babu, "More generality in efficient multiple kernel learning," in *Proc. Int. Conf. Mach. Learn. ACM*, 2009.

[42] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "Simple MKL," *J. Mach. Learn. Res.*, vol. 9, pp. 2491–2521, 2008.

[43] D. Sahoo, S. C. H. Hoi, and B. Li, "Online multiple kernel regression," in *Proc. the 20th ACM SIGKDD*, 2014, pp. 293–302.

[44] D. Sahoo, S. C. H. Hoi, and B. Li, "Large scale online multiple kernel regression with application to time-series prediction," *ACM Trans. Knowledge Discovery from Data*, vol. 13, no. 1, pp. 9:1–9:33, Jan. 2019.

[45] R. Pokharel, S. Seth and J. C. Principe, "Mixture kernel least mean square," in *Proc. 2013 Int. Joint Conf. on Neural Networks (IJCNN)*, 2013, pp. 1–7.

[46] W. Gao, C. Richard, J.-C. M. Bermudez and J. Huang, "Convex combinations of kernel adaptive filters," in *IEEE Int. Workshop MLSP,*, 2014.

[47] M. Yukawa, "Multikernel adaptive filtering," *IEEE Trans. Signal Process.*, vol. 60, no. 9, pp. 4672–4682, Sep. 2012.

[48] M. Yukawa, "Adaptive learning in Cartesian product of reproducing kernel Hilbert spaces," *IEEE Trans. Signal Process.*, vol. 63, no. 22, pp. 6037–6048, Nov. 2015.

[49] O. Toda and M. Yukawa, "Online model-selection and learning for nonlinear estimation based on multikernel adaptive filtering," *IEICE transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E100.A, no. 1, pp. 236–250, 2017.

[50] M. Kasparick, R. L. G. Cavalcante, S. Valentin, S. Stánczak and M. Yukawa, "Kernel-based adaptive online reconstruction of coverage maps with side information," *IEEE Trans. Vehicular Tech.*, vol. 65, no. 7, pp. 5461–5473, Jul. 2016.

[51] D. A. Awan, R. L. G. Cavalcante, M. Yukawa and S. Stanczak, "Detection for 5G-NOMA: An online adaptive machine learning approach," in *2018 IEEE International Conference on Communications (ICC)*, 2018.

[52] D. A. Awan, R. L. G. Cavalcante, M. Yukawa and S. Stanczak, "Adaptive learning for symbol detection: A reproducing kernel hilbert space approach," in *Machine Learning for Future Wireless Communications*, F. Luo, Ed., chapter 11, pp. 197–211. Wiley, 2019.

[53] B.-S. Shin, M. Yukawa, R. L. G. Cavalcante and A. Dekorsy, "Distributed adaptive learning with multiple kernels in diffusion networks," *IEEE Trans. Signal Process.*, vol. 66, no. 21, pp. 5505–5519, Nov. 2018.

[54] X. Chen, F. Xu and Y. Ye, "Lower bound theory of nonzero entries in solutions of ℓ_2-ℓ_p minimization," *SIAM J. Scientific Computing*, vol. 32, no. 5, pp. 2832–2852, 2010.

[55] M. Wang, W. Xu and A. Tang, "On the performance of sparse recovery via ℓ_p-minimization ($0 \leq p \leq 1$)," *IEEE Trans. Inform. Theory*, vol. 57, no. 11, pp. 7255–7278, Nov. 2011.

[56] M. Yukawa and S. Amari, "On extensions of LARS by Information Geometry : Convex objectives and ℓ_p-norm," in *APSIPA Annual Summit and Conference: Special Session on Recent Advances in Adaptive/Sparse Signal Processing*, Oct. 2011.

[57] M. Yukawa and S. Amari, "ℓ_p-constrained least squares ($0 < p < 1$) and its critical path," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2012, pp. 2231–2235.

[58] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[59] M. Yukawa and R. Ishii, "Online model selection and learning by multikernel adaptive filtering," in *Proc. 21th Eur. Signal Process. Conf. (EUSIPCO)*, 2013, pp. 1–5.

[60] F. Albu and K. Nishikawa, "The kernel proportionate NLMS algorithm," in *Proc. 21th Eur. Signal Process. Conf. (EUSIPCO)*, 2013, pp. 1–5.

[61] M. Takizawa and M. Yukawa, "Adaptive nonlinear estimation based on parallel projection along affine subspaces in reproducing kernel Hilbert space," *IEEE Trans. signal Process.*, vol. 63, no. 16, pp. 4257–4269, Aug. 2015.

[62] M. Yukawa and K. R. Müller, "Why does a Hilbertian metric work efficiently in online learning with kernels?," *IEEE Signal Process. Letter*, vol. 23, no. 10, pp. 1424–1428, Oct. 2016.

[63] S. Wang, L. Dang, B. Chen, C. Ling, L. Wang and S. Duan, "Kernel online learning algorithm with scale adaptation," *IEEE Trans. Circuits and Systems*, vol. 65, no. 11, pp. 1788–1792, Nov. 2018.

[64] F. A. Tobar, S.-Y. Kung, and D. P. Mandic, "Multikernel least mean square algorithm," *IEEE Trans. Neural Net. Learn. Sys.*, vol. 25, no. 2, pp. 265–277, Feb. 2014.

[65] K. Nishikawa and F. Albu, "Implementation method of kernel adaptive filter as an add-on for a linear adaptive filter," in *Proc. 23rd Eur. Signal Process. Conf. (EUSIPCO)*, 2015, pp. 2736–2740.

[66] S. V. Vaerenbergh, S. Scardapane and I. Santamaria, "Recursive multikernel filters exploiting nonlinear temporal structure," in *Proc. 25th Eur. Signal Process. Conf. (EUSIPCO)*, 2017, pp. 2674–2678.

[67] M. Ohnishi and M. Yukawa, "Online nonlinear estimation via iterative L^2-space projections: Reproducing kernel of subspace," *IEEE Trans. Signal Process.*, vol. 66, no. 15, pp. 4050–4064, Aug. 2018.

[68] T. Wada, K. Fukumori and T. Tanaka, "Dictionary learning for Gaussian kernel adaptive filtering with variable kernel center and width," in *Proc. 43th Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2018, pp. 2766–2770.

[69] M. Takizawa and M. Yukawa, "Online learning with self-tuned Gaussian kernels: Good kernel-initialization by multiscale screening," in *Proc. 44th Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2019, pp. 4863–4867.

[70] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proc. the 32th International Conference on Machine Learning (ICML)*, 2015, pp. 448–456.

[71] S. Santurkar, D. Tsipras, A. Ilyas and A. Madry, "How does batch normalization help optimization?," in *Proc. Neural Information Processing Systems (NIPS) 2018*, 2018, pp. 2483–2493.

[72] N. Bjorck, C. P. Gomes, B. Selman and K. Q. Weinberger, "Understanding batch normalization," in *Proc. Neural Information Processing Systems (NIPS) 2018*, 2018, pp. 7694–7705.

[73] S. S. Narayan, A. M. Peterson and M. J. Narasimha, "Transform domain LMS algorithm," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-31, pp. 609–615, Jun. 1983.

[74] D. F. Marshall, W. K. Jenkins and J. J. Murphy, "The use of orthogonal transforms for improving performance of adaptive filters," *IEEE Trans. Circuits Syst.*, vol. 36, no. 4, pp. 474–484, 1989.

[75] M. Yukawa, K. Slavakis, and I. Yamada, "Adaptive parallel quadratic-metric projection algorithms," *IEEE Trans. Audio, Speech and Language Process.*, vol. 15, no. 5, pp. 1665–1680, Jul. 2007.

[76] M. Yukawa and I. Yamada, "Krylov-proportionate adaptive filtering techniques not limited to sparse systems," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 927–943, 2009.

[77] M. Yukawa and I. Yamada, "A unified view of adaptive variable-metric projection algorithms," *EURASIP Jour. Adv. Signal Process.*, vol. 2009, pp. 1–13, 2009.

[78] J. Nagumo and A. Noda, "A learning method for system identification," *IEEE Trans. Automatic Control*, vol. 12, no. 3, pp. 282–287, 1967.

[79] T. Wigren and J. Schoukens, "Three free data sets for development and benchmarking in nonlinear system identification," in *European Control Conference (ECC)*, 2013, pp. 2933–2938.

[80] A.S. Weigend and N.A. Gershenfeld, *Time series prediction: Forecasting the future and understanding the past*, Addison-Wesley, Reading, MA, 1993.

[81] F. Kawala, A. Douzal-Chouakria, E. Gaussier, and E. Dimert, "Prédictions d'activité dans les réseaux sociaux en ligne," in *In Actes de la Conférence sur les Modèles et l'Analyse des Réseaux : Approches Mathématiques et Informatique (MARAMI)*, 2013, p. 16.

[82] H. Mohimani, M. Babaie-Zadeh and C. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed ℓ^0 norm," *IEEE Trans. Signal Process.*, vol. 57, no. 1, pp. 289–301, Jan. 2009.

[83] D. Malioutov and A. Aravkin, "Iterative log thresholding," in *Proc. 39th Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2014, pp. 7198–7202.

[84] Z. Li, S. Ding and Y. Li, "Dictionary learning with log-regularizer for sparse representation," *IEEE Int'l conf. DSP*, pp. 609–613, 2015.

[85] I. Bayram, "Penalty functions derived from monotone mappings," *IEEE Signal Process. Letter*, vol. 22, no. 3, pp. 265–269, Mar. 2015.

[86] E. Soubies, L. Blanc-Féraud and G. Aubert, "A continuous exact ℓ_0 penalty (cel0) for least squares regularized problem," *SIAM Journal on Imaging Sciences*, vol. 8, no. 3, pp. 1607–1639, 2015.

[87] M. Sadeghi and M. Babaie-Zadeh, "Iterative sparsification-projection: Fast and robust sparse signal approximation," *IEEE Trans. Signal Process.*, vol. 64, no. 21, pp. 5536–5548, Nov. 2016.

[88] I. W. Selesnick and I. Bayram, "Enhanced sparsity by non-separable regularization," *IEEE Trans. Signal Process.*, vol. 64, no. 9, pp. 2298–2313, May 2016.

[89] I. Selesnick, "Sparse regularization via convex analysis," *IEEE Trans. Signal Process.*, vol. 65, no. 17, pp. 4481–4494, Sep. 2017.

Publications Related to the Dissertation

Journal Article

1. K. Jeong, M. Yukawa and S. -i. Amari, "Can critical-point paths under ℓ_p-regularization ($0 < p < 1$) reach the sparsest least squares solutions?", IEEE Transactions on Information Theory, Vol. 60, No. 5, pp. 2960–2968, May 2014.

2. K. Jeong and M. Yukawa, "Kernel weights for equalizing kernel-wise convergence rates of multikernel adaptive filtering", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences Vol. E104-A, No. 6, Jun. 2021.

International Conference Paper

1. K. Jeong and M. Yukawa, "Automatic kernel weighting for multikernel adaptive filtering: Multiscale aspects", Proceedings of the 44th International Conference on Acoustic, Speech and Signal Processing, pp. 3152–3156, 2019.

Other Publications

International Conference Paper

1. K. Jeong, M. Yukawa, M. Yamagishi and I. Yamada, "Automatic shrinkage tuning robust to input correlation for sparsity-aware adaptive filtering", Proceedings of the 43rd International Conference on Acoustic, Speech and Signal Processing, pp. 4314–4318, 2018.

Domestic Conference Papers

1. K. Jeong, M. Yukawa and S. -i. Amari, "On relation between ℓ_p-regularization $(0 < p < 1)$ and sparsest least squares solution," in Proceedings of Symposium on Information Theory and Its Applications (SITA), pp. 196–199, Nov. 2013.

2. K. Jeong, M. Yukawa, M. Yamagishi and I. Yamada, "A simple self-tuning technique for adaptive proximal forward-backward splitting method", Technical Report, IEICE-SIP, pp. 71–76, 2017.

3. K. Jeong, M. Yukawa, M. Yamagishi and I. Yamada, "A time-averaging approach to automatic shrinkage tuning under colored signals for sparsity-aware adaptive filtering", Proceedings of SIP symposium, Nov. 2017.