

A NOVEL KERNEL LEAST MEAN SQUARE ALGORITHM BASED ON WEIGHTED GAUSSIAN KERNEL

BY
SALMAN RAUF IQBAL

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In
ELECTRICAL ENGINEERING

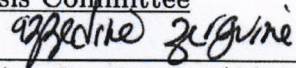
APRIL 2021

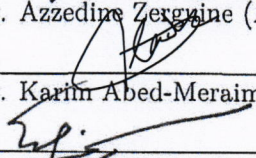
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

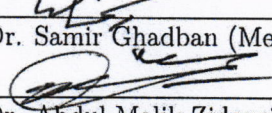
DEANSHIP OF GRADUATE STUDIES

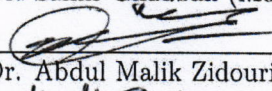
This thesis, written by **SALMAN RAUF IQBAL** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN ELECTRICAL ENGINEERING**.

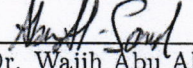
Thesis Committee


Dr. Azzedine Zerguine (Adviser)

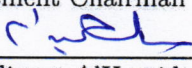

Dr. Karim Abed-Meraim (Co-Adviser)


Dr. Samir Ghadban (Member)


Dr. Abdul Malik Zidouri (Member)


Dr. Wajih Abu AlSaud (Member)


Dr. Abdallah AlAhmari
Department Chairman


Dr. Suliman AlHomidan
Dean of Graduate Studies

Date



©Salman Rauf Iqbal
2021

To my parents, wife and kids

ACKNOWLEDGMENTS

I am thankful to my advisor Dr. Azzedine for giving me the right direction and guidance to reach up to this level. I would also like to thank my Co-advisor and the committee members for their support. Moreover, I would like to thank the University, KFUPM, for providing me unlimited resources (access to research papers, software tools, utilities etc.), and giving motivation through technical sessions, seminars, and programs. I am in never-ending debt to the faculty staff, be it my advisor, professors with whom I interacted, or my course professors, all of them played a huge role in encouraging me, and keeping me motivation throughout the challenges and hardships associated with my journey of Masters.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	vi
LIST OF TABLES	ix
LIST OF FIGURES	x
ABSTRACT (ENGLISH)	xiii
ABSTRACT (ARABIC)	xiv
CHAPTER 1 INTRODUCTION	1
1.1 LMS Algorithm	2
1.2 Kernel Based Learning	3
1.3 Kernel Least Mean Square Algorithm	5
1.3.1 Advantages	7
1.3.2 Challenges	9
CHAPTER 2 ANALYSIS AND VARIANTS OF KLMS	12
2.1 Mean Square Convergence Analysis	13
2.1.1 Upper bound on step size	14
2.1.2 Steady state Mean Square Error	14
2.1.3 Transient Analysis	15
2.1.4 Effect of Kernel size on Convergence behavior	16
2.2 Quantized KLMS	19
2.3 KLMS based on Conjugate Gradient	24

2.4	KLMS with Constrained Growth	25
2.5	Information Theoretic KLMS	28
CHAPTER 3 WEIGHTED GAUSSIAN KLMS		31
3.1	Construction of WKLMS	32
3.1.1	Effect of Weights on Gram Matrix	36
3.2	Convergence Analysis	39
3.2.1	System Model	39
3.2.2	Mean Behavior	40
3.2.3	Mean-Square Behavior	42
3.3	Performance analysis of WKLMS	44
3.3.1	Schur Convex EMSE of WKLMS	46
3.3.2	Some Empirical Results	49
3.3.3	Simulation Results	53
CHAPTER 4 CONCLUSION		61
4.1	Future research	62
APPENDIX A PROOFS		63
APPENDIX B SUPPLEMENTARY MATERIAL		66
REFERENCES		67
VITAE		71

LIST OF TABLES

3.1	Summary of analytical results for various performance measures of the WKLMS algorithm.	44
3.2	Eigenvalues and eigenvalue spread of Gram Matrix having size $N = 50$	56
3.3	Comparison of steady-state EMSE of the WKLMS algorithm with $\sigma_1 \succ$ $\sigma_2 \succ \sigma_3$	58

LIST OF FIGURES

1.1	Superior MSE performance of KLMS compared to LMS. [1].	8
2.1	Evolution of excess MSE (EMSE) for different step sizes [2]. The upper bound is selected as per Equation (18), and its excess MSE is the highest. For the case when step size is chosen larger than the upper bound, algorithm becomes unstable.	16
2.2	EMSE showing monotonic increase due to increase in step size [2]. This behavior is inline with Equation (2.5), as shown by blue dots (simulation).	17
2.3	Constant increase in EMSE due to noise variance [2].	17
2.4	Effect of kernel size on EMSE. The EMSE remains invariant to kernel size. [2]	18
2.5	Change in EMSE as a function of kernel size during 1st iteration of the algorithm.[2]	18
2.6	Evolution of EMSE during 1st 50 iterations of the algorithm [2]. Above figure, Fig. 2.5 and Fig. 2.6 show that kernel size has an impact on initial convergence (Transient behavior) of the algorithm, but the steady state EMSE remains same.	19
2.7	Dependence of EMSE on quantization factor γ . Upper and Lower bound in blue are the theoretical estimates as given by Equation 2.12. [3]	22
2.8	Variation of network size (kernel centers in the memory) depending on the quantization factor. [3]	22
2.9	Comparison of network size generated by KLMS using different sparsification techniques. [3]	23

2.10	EMSE performance of KLMS and QKLMS [3].	23
2.11	MSE comparison of KLMS-CG with KLMS, Quantized KLMS and Quantized KLMS-CG	25
2.12	Network growth in terms of number of kernel centers as a function of threshold δ while predicting Mackey Glass Time Series [3]	27
2.13	Network growth as the algorithm evolves over time for different thresh- old values δ . [4]	27
2.14	Network size comparison of KLMS having Surprise Criterion (SC- KLMS) and KLMS having Coherence Criterion (KLMS-CC) [5], [6] . .	30
3.1	Eigenvalue distribution of Gram matrix $\mathbf{G}_w$. Input data is Gaussian distributed with zero mean and unit variance. $N = 50$, $d = 5$ and weights = [16,16,16,16,16].	37
3.2	Eigenvalue distribution of Gram matrix $\mathbf{G}_w$. Input data is Gaussian distributed with zero mean and unit variance. $N = 50$, $d = 5$ and weights = [0.01,0.01,0.01,0.01,0.01].	38
3.3	Eigenvalue distribution of Gram matrix $\mathbf{G}_w$. Input data is Gaussian distributed with zero mean and unit variance. $N = 50$, $d = 5$ and weights = [1,1,1,1,1].	38
3.4	Eigenvalue distribution of Gram matrix $\mathbf{G}_w$. Input data is Gaussian distributed with zero mean and unit variance. $N = 50$, $d = 5$ and weights = [0.1,0.1,1,2,2].	38
3.5	Eigenvalue distribution of Gram matrix $\mathbf{G}_w$. Input data is Gaussian distributed with zero mean and unit variance. $N = 50$, $d = 5$ and weights = [2,2,1,0.1,0.1].	38
3.6	MSE performance comparison of LMS, KLMS and WKLMS. Selected weights for WKLMS are [0.1,0.1,1,2,2].	55
3.7	MSE performance comparison of WKLMS with different weight se- lection. Selected weights for WKLMS-1 are [0.1,0.1,1,2,2] and for WKLMS-2 are [2,2,1,0.1,0.1].	55

3.8	MSE performance comparison of WKLMS with very large and small weights. Weights in WKLMS-1 are [0.01,0.01,0.01,0.01] and in WKLMS-2 are [16,16,16,16].	55
3.9	MSE behavior of WKLMS while dynamically changing the weights. WKLMS-1 has best weight assignment as [0.1, 0.1, 1, 2, 2] and WKLMS-2 has weights [2, 2, 1, 0.1, 0.1] from start till iteration 350. From 351 onwards, WKLMS-2 is assigned same weights as WKLMS-1.	56
3.10	Testing MSE for the non-linear channel equalization.	59
3.11	MSE performance comparison of WKLMS with different weight selections. Selected weights for WKLMS-1 are [0.1,0.1,1,2,2] and for WKLMS-2 are [2,2,1,0.1,0.1].	59
3.12	Comparison of analytical and simulation result for the MSE learning curves.	60

THESIS ABSTRACT

NAME: Salman Rauf Iqbal

TITLE OF STUDY: A novel Kernel Least Mean Square Algorithm based on weighted Gaussian Kernel

MAJOR FIELD: Electrical Engineering

DATE OF DEGREE: April 2021

In this thesis I have provided a comprehensive overview of Kernel Least Mean Square Algorithm (KLMS), the theory behind its success, and its applications. Moreover, a novel Kernel Least Mean Square (KLMS) algorithm is proposed using a weighted Gaussian Kernel to yield the WKLMS algorithm. The challenge of defining the weights is addressed by proposing two weight selection criteria which improve the Mean Square Error (MSE) performance. The first criterion is derived by using empirical results. Second criterion shows that the rows of weighting matrix should be similar to eigenvectors orthogonal to the input auto-correlation matrix formed with respect to the Radial Basis Function (RBF) Centers. Results obtained from both empirical and analytical techniques prove that significant performance improvement can be achieved by using the WKLMS. Lastly, some future research directions are proposed.

ملخص الرسالة

الاسم: سلمان رؤوف اقبال

عنوان الدراسة: عننرواية خوارزمية Square Mean Least Kernel تعتمد على نواة Gaussian الموزونة

التخصص: الهندسة الكهربائية

تاريخ الدرجة العلمية: أبريل 2021

عند كتابة ملخص ، ضع في اعتبارك مجرد ملخص هو ملخص وصفي قصير لأطروحتك. قد يختلف عددلقد قدمت في هذه الأطروحة نظرة عامة شاملة عن خوارزمية SquareAlgorithm Mean Least Kernel (KLMS) والنظرية الكامنة وراء نجاحها وتطبيقاتها. علاوة على ذلك ، تم اقتراح خوارزمية Square Mean Least Kernel Anovel (KLMS) باستخدام نواة Gaus-sian الموزونة للحصول على خوارزمية WKLMS. تتم معالجة التحدي المتمثل في تحديد الأوزان من خلال اقتراح معيارين لاختيار الوزن يعملان على تحسين أداء متوسط الخطأ المربع (MSE) يتم اشتقاق المعيار الأول باستخدام النتائج التجريبية. يوضح المعيار الثاني أن صفوف مصفوفة الترجيح يجب أن تكون مماثلة لمصفوفة الارتباط التلقائي للمدخلات المتشكلة فيما يتعلق بمراكز دالة RadialBasis (RBF). تثبت النتائج التي تم الحصول عليها من كل من الأساليب التجريبية والتحليلية أنه يمكن تحقيق تحسين كبير في الأداء باستخدام WKLMS. أخيرًا ، تم اقتراح بعض اتجاهات البحث المستقبلية

CHAPTER 1

INTRODUCTION

Least Mean Square (LMS) algorithm introduced in [7] made a remarkable breakthrough due to its simplicity and usefulness in solving adaptive learning problems. The major change in LMS was the replacement of deterministic gradient with its stochastic evaluation, allowing the user to bypass the calculation of Input and Input-output correlation statistic. However, LMS relies on inner products, and therefore, is efficient in solving linear adaptive learning problems.

Kernel Least Mean Square (KLMS) algorithm was introduced in [1] by utilizing the fact that LMS is relying on inner products in input space, and if these inner products are carried out in high dimensional non-linear vector space, one can attempt to resolve nonlinear adaptive learning problems. Kernel Methods [8] are an efficient tool for such non-linear transformation, where inner products in high dimensional space, also known as feature space (F) are carried out by simply evaluating the Kernel functions in the input space, without even knowing explicitly the input space to feature space nonlinear mapping.

Results shown by KLMS in solving nonlinear learning problems (Time Series fore-

casting, nonlinear channel equalization and estimation, nonlinear static function estimation etc.) were promising. However, KLMS results in a growing Radial Basis Function (RBF) network, in which every iteration of training data causes an increase in network size [1]. Moreover, there are many Kernel functions (Gaussian, Polynomial, Multi-quadrics etc.), and therefore, choosing an appropriate Kernel function for the specific problem is another challenge. In order to understand KLMS algorithms, or in general, any algorithm using kernel methods, it is necessary to introduce the concept of Kernel based learning [8]. Since KLMS is using LMS algorithm at its core, we first present necessary background of LMS, followed by some fundamental concepts related to Kernel Methods.

1.1 LMS Algorithm

The LMS algorithm introduced by Widrow [7] is a very simple, yet powerful adaptive learning algorithm to minimize mean square error. It uses quadratic cost function $J[n]$ which depends on the square of the difference between desired and actual output. Using gradient descent, the adaptive filter tap weights are updated iteratively [9]. Let $\mathbf{x}(n)$ denote the input data vector, where n is the time index, $e(n)$ the error in the filter output compared to the desired value $d[n]$, and $\mathbf{w}(n)$ the tap weight vector. The weight update equation for LMS can be written as:

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \eta e(n) \mathbf{x}(n) \quad (1.1)$$

Where η is the step size of LMS. The scalar error term $e(n)$ is defined as:

$$e(n+1) = d(n) - y(n) \quad (1.2)$$

where $y(n)$ is the output of the LMS algorithm at instant n . Equation (1.1) is derived by applying the gradient on the quadratic cost function $J(n)$. The output of LMS algorithm at instant n can be written as:

$$y(n) = \mathbf{w}(n)^T \mathbf{x}(n) \quad (1.3)$$

Equation (1.3) shows that the output of LMS algorithm is equal to the inner product between weight vector $\mathbf{w}(\mathbf{n})$, and input vector $\mathbf{x}(\mathbf{n})$. [9] and [10] provide an in-depth review and analysis of LMS algorithm.

1.2 Kernel Based Learning

Kernel based learning [8] is a technique where the learning algorithm operates in a high dimensional space, rather than the original input space. This high dimensional space is also known as feature space, denoted by F . The input data is transformed, into the feature space, and after necessary computations, the results are transformed back into the original input space. The input-to-feature space mapping is nonlinear, and the dimension of feature space is much larger than the input space, and in some cases can be even infinite.

A kernel function, satisfying Mercer's conditions (symmetric, and positive defi-

nitiness) [11] can be defined as:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \quad (1.4)$$

where $\mathbf{x}_i$ is the input vector and $\phi(\mathbf{x}_i)$ is the transformed feature space vector. Equation (1.4) shows that Kernel functions compute inner product in feature space, while their input argument is relying only on input vectors which also implies that by using Kernels, one can compute inner products in the feature space without transforming the original input vectors. Therefore, if an algorithm is relying on inner products, it can be operated in feature space by using Kernel functions. This technique, which is also known as the kernel trick in machine learning community, has been applied to various algorithms, namely, Kernel Principal Component Analysis (K-PCA), kernel Support Vector Machine (K-SVM) and Kernel Independent Component Analysis (K-ICA).

An interesting property of this high dimensional feature space is that it is a Reproducing Kernel Hilbert Space (RKHS), which means that in this space the following reproducing property holds:

$$f(\mathbf{x}) = \langle \kappa(\mathbf{x}, \cdot), f \rangle, \quad \forall f \in F \quad (1.5)$$

Where $f(\mathbf{x})$ is an arbitrary function evaluated at $\mathbf{x}$ in the RKHS, and $\kappa(\mathbf{x}, \cdot)$ is the kernel function evaluated at fixed $\mathbf{x}$ and any arbitrary input (as second argument).

By using (1.5), the right hand side of (1.4) can be expressed as:

$$\begin{aligned}
\langle (\mathbf{x}_i), (\mathbf{x}_j) \rangle &= \langle \kappa(\mathbf{x}_i, \cdot), \kappa(\mathbf{x}_j, \cdot) \rangle \\
&= \kappa(\mathbf{x}_i, \mathbf{x}_j)
\end{aligned} \tag{1.6}$$

Equation (1.6) shows that a kernel function with an input defines a Reproducing Kernel Hilbert Space. Since the mapping is nonlinear, one can take advantage of it by using the linear algorithm to learn the nonlinear problem.

There are many types of Kernels, but the most important and frequently used Kernel is the Gaussian Kernel due to its universal approximation capability [12]. Gaussian kernel, also known as Radial Basis Function (RBF) Kernel, is defined as:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2) \tag{1.7}$$

where σ is the kernel size (or width). Gaussian Kernel belongs to the class of translation invariant kernels, and they induce feature space of infinite dimension (since a Gaussian Kernel is exponential function, which has an infinite Taylor series expansion). Throughout this thesis we will use Gaussian kernel.

1.3 Kernel Least Mean Square Algorithm

Following the details presented in [1] and [12], Equation (1.1) implies that LMS algorithm relies on the inner product between weight and input vectors. It implies that can one perform this inner product in the feature space by using an appropriate kernel.

Let us denote the transformed input vector $\mathbf{x}(n)$ in the feature space as $\boldsymbol{\phi}(n)$, and the tap weight vector $\mathbf{w}(n)$ as $\boldsymbol{\Omega}(n)$. Then, Equation (1.1) of LMS algorithm can be rewritten in the feature space as:

$$\boldsymbol{\Omega}(n) = \boldsymbol{\Omega}(n-1) + \eta e(n) \boldsymbol{\phi}(n) \quad (1.8)$$

Since we are using Gaussian Kernel which induces an infinite dimensional feature space, the weight vector $\boldsymbol{\Omega}(n)$, being a vector in infinite dimensional feature space will be of infinite dimension. Let us assume $\boldsymbol{\Omega}(0) = 0$, then using (1.8) we get:

$$\boldsymbol{\Omega}(1) = \eta e(0) \boldsymbol{\phi}(0)$$

Continuing in similar way, we can write the weight update equation in terms of error and input parameters as below:

$$\boldsymbol{\Omega}(n) = \eta \sum_{i=1}^n e(i) \boldsymbol{\phi}(i) \quad (1.9)$$

We can rewrite LMS output Equation (1.3) in the feature space as below:

$$y(n) = \boldsymbol{\Omega}(n)^T \boldsymbol{\phi}(n) \quad (1.10)$$

$$\begin{aligned} &= \eta \sum_{i=1}^n e(i) \langle \boldsymbol{\phi}(i)^T \boldsymbol{\phi}(n) \rangle \\ &= \eta \sum_{i=1}^n e(i) \kappa(\mathbf{x}(i), \mathbf{x}(n)) \end{aligned} \quad (1.11)$$

Equation (1.11) defines the KLMS algorithm, which implements the LMS algorithm

in the feature space. Note that the left hand side of (1.10) is output vector in input space, while the right hand side is in feature space. This implicit mapping was possible because of the kernel, which will compute the inner product in feature space based on vectors from the input space, as shown in (1.11).

1.3.1 Advantages

There are three main advantages of KLMS that are important to be discussed and are presented below.

Online Learning

There are primarily two types of learning algorithms: batch-mode and online. While most of the kernel based learning algorithms are batch-mode, KLMS algorithm can operate online. As we see from (1.11), the output of the KLMS algorithm at time instant n depends on the $n-1$ training samples. It implies that KLMS can produce output even while it is under training, contrary to batch-mode algorithms which can only operate once testing set has been completed.

Nonlinear learning

Since the mapping between input and feature space (RKHS) is nonlinear, with suitable choice of Kernel, complexity of the learning problem in the feature space is significantly reduced. Both LMS and KLMS rely on inner products, which is a linear operation. When nonlinear problem is mapped in the feature space, the same linear operation, being inner products, solves it more efficiently, provided that the choice of feature

space is appropriate. In the case of Gaussian kernel, the feature space is of infinite dimension, which means that the space spanned by feature vectors will cause the error term defined in (1.2) to be very small. Figure 1.1 shows KLMS performing better than LMS in terms of mean square error while equalizing a nonlinear communication channel.

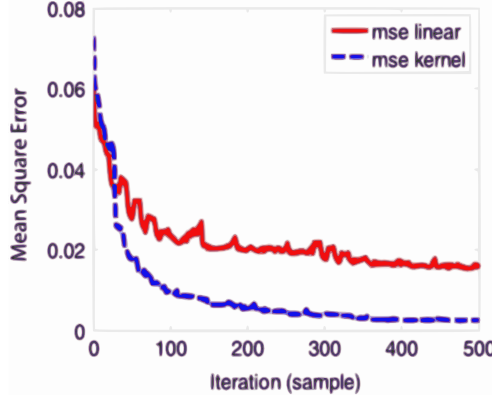


Figure 1.1: Superior MSE performance of KLMS compared to LMS. [1].

Well-Posedness

In [1] it is shown that KLMS algorithm does not require explicit regularization [13], as is usually required in most of the Kernel based learning algorithms, implying that it is well-posed. There are two approaches adopted in [1] to reach this conclusion, one is by applying small step size theory and convergence analysis of LMS in the mean sense, the second is by analyzing the solution norm of the empirical risk minimized by the algorithm. Here we briefly summarize the second approach. The objective of KLMS is to minimize the below unconstrained empirical risk:

$$\min_{\Omega} J(\Omega) = \frac{1}{N} \sum_{i=1}^N (y(i) - \Omega \phi(i))^2 \quad (1.12)$$

In [1], the authors show that instead of solving the unconstrained optimization problem shown in (1.12), the KLMS algorithm attempts to solve the below constrained optimization equation:

$$\begin{aligned} \min_{\Omega} J(\Omega) &= \frac{1}{N} \sum_{i=1}^N (y(i) - \Omega \phi(i))^2 \\ \text{subject to } \|\Omega_N\|^2 &< \sigma_1 \eta (\|\Omega^o\|^2 + 2\eta \|\mathbf{v}\|^2) \end{aligned} \quad (1.13)$$

where $\mathbf{v}$ is the noise term and σ_1 is the largest eigenvalue of gram matrix, and η is the step size of the algorithm. Equation (1.13) tells us that KLMS attempts to minimize the constrained empirical risk. From regularization network theory [14], we know that the constrained optimization problem, when the solution norm is bounded, is equivalent to solving an unconstrained regularized optimization problem. Thus, (1.13) can be rewritten as:

$$\min_{\Omega} J[\Omega] = \frac{1}{N} \sum_{i=1}^N (y[i] - \Omega \phi[i])^2 + \lambda \|\Omega\|^2 \quad (1.14)$$

Therefore, KLMS algorithm doesn't require explicit regularization as it is inherently well-posed in minimizing the empirical risk shown in (1.12).

1.3.2 Challenges

Kernel LMS brings many advantages as summarized above, but it also introduces some major challenges in its design and implementation.

Choice of Kernel

There are many type of Kernels used in the literature which satisfy the Mercer's conditions [5] and (1.4) (for example Polynomial kernels and RBF Kernels). But the most widely used Kernel is the Gaussian Kernel, defined in (1.7) mainly due to its universal approximation capability [15]. However, depending upon the problem, Gaussian kernel might not be the optimum choice. In most of the real world problems, a-priori information about the input data and noise is not available, and therefore, the choice of Kernel has to be made during the training of algorithm. It is still an open problem to decide the most optimum Kernel of choice for the problem in hand. Several studies have been conducted in this regard. Nevertheless, in most of the cases when sufficient information about the problem is not available, Gaussian Kernel is a safe and suitable choice. However, choosing Gaussian kernel also requires to optimally select the Kernel width parameter σ as shown in (1.7). Choosing an optimum value of h is not straight forward and requires cross validation techniques [16], [17] . Choosing a very low value of h will cause all the inputs to appear almost orthogonal to each other, and, a very high value of h will cause all the inputs to look similar. The Gaussian kernel is very sensitive to the width parameter, and therefore, it should be carefully chosen for the problem under consideration.

Growing Network

From Equation (1.10) or (1.11), the output of KLMS depends upon the kernel evaluation of the current input with all past inputs, and weighted by the respective errors. It creates a growing network, where at every next iteration, a new kernel evaluation is

introduced along with its respective error weight. With Gaussian Kernel as our choice, (1.11) implies a growing RBF network, where the size of the network is equal to length of training data. Correspondingly, KLMS implementation needs high memory.

Iterative Kernel Evaluations

Equation (1.11) implies that not only KLMS requires kernel evaluation at every training instance, but these evaluations are also unique. It is therefore required that they must be computed at every iteration. These computations are unique in the sense that at every iteration, the input data vector is compared with the updated network, which is always larger than the previous iteration. By network here, we mean the collection of Kernels in the memory which are going to be evaluated against the new input vector. Moreover, the new kernel which is added in the network is weighted by the error term, $e(i)$. All these factors lead to calculating the newly updated network response at that specific iteration.

Therefore, at every training iteration, KLMS requires complete RBF network to be updated, and this leads to the fact that KLMS is computationally more expensive than its linear counterpart, LMS algorithm.

CHAPTER 2

ANALYSIS AND VARIANTS OF KLMS

In this Chapter, we will dive into the convergence analysis of KLMS. By convergence analysis, it is implied to study the transient, as well as steady state response of the algorithm. In the case of transient response, our interest is to see how fast the algorithm converges to minimum mean square error, and in steady state analysis, the interest is to find our factors determining the minimum mean square error achievable by the algorithm.

Moreover, we will also discuss in detail the different variants of KLMS, i.e., the different variations of KLMS that have been proposed in literature to improve the performance of the algorithm, or to optimize its time and space complexities.

2.1 Mean Square Convergence Analysis

In [10], using the fundamental Energy Conservation relation, analytical expressions defining the mean square convergence behavior of KLMS have been derived. These analytical expressions give useful insight into the transient as well as steady state behavior of the algorithm. The fundamental energy conservation relation of the KLMS in feature space can be defined as:

$$E \left[\left\| \tilde{\mathbf{\Omega}}(i) \right\|_{\mathbb{F}}^2 \right] + E \left[\frac{e_a^2(i)}{\kappa(\mathbf{x}(\mathbf{i}), \mathbf{x}(\mathbf{i}))} \right] = E \left[\left\| \tilde{\mathbf{\Omega}}(i-1) \right\|_{\mathbb{F}}^2 \right] + E \left[\frac{e_p^2(i)}{\kappa(\mathbf{x}(\mathbf{i}), \mathbf{x}(\mathbf{i}))} \right] \quad (2.1)$$

Here in above Equation, $\tilde{\mathbf{\Omega}}(i) = \mathbf{\Omega}^o - \mathbf{\Omega}(i)$ is the weight error vector, $\mathbf{\Omega}^o$ being the optimum weight vector, and both vectors are in feature space $\mathbb{F}$. $e_a(i) = \mathbf{\Omega}^T(i-1)\phi(i)$ and $e_p(i) = \mathbf{\Omega}^T(i)\phi(i)$ are the a priori and a posteriori errors. Considering the case of Gaussian kernel, where $\kappa(\mathbf{x}(i), \mathbf{x}(i)) = 1$, above Equation (2.1) reduces to:

$$E \left[\left\| \tilde{\mathbf{\Omega}}(i) \right\|_{\mathbb{F}}^2 \right] + E [e_a^2(i)] = E \left[\left\| \tilde{\mathbf{\Omega}}(i-1) \right\|_{\mathbb{F}}^2 \right] + E [e_p^2(i)] \quad (2.2)$$

Above equation shows how the weight error vector evolves with each iteration of the algorithm. Assuming additive noise having zero mean, independent and identically distributed (i.i.d), and independent of $e_a(i)$, and considering that $e_p(i) = e_a(i) - \eta e(i)$, where η being the step size of algorithm, and $e(i) = e_a(i) + v(i)$, $v(i)$ being the noise term, Equation (2.2) can be written as:

$$E \left[\left\| \tilde{\mathbf{\Omega}}(i) \right\|_{\mathbb{F}}^2 \right] = E \left[\left\| \tilde{\mathbf{\Omega}}(i-1) \right\|_{\mathbb{F}}^2 \right] - 2\eta E [e_a^2(i)] + \eta^2 (E [e_a^2(i)] + \zeta_v^2) \quad (2.3)$$

Here ζ_v^2 is the noise variance. Using (2.3), the convergence behavior of KLMS can be analyzed.

2.1.1 Upper bound on step size

For stable convergence of the algorithm, the weight error vector $\tilde{\mathbf{\Omega}}(i)$ has to decrease with each iteration i . In other words, $\|\tilde{\mathbf{\Omega}}(i)\| < \|\tilde{\mathbf{\Omega}}(i-1)\|$. Applying this condition to Equation (2.3), the authors in [2] showed upper bound on step size η to be:

$$\eta \leq \frac{2E[e_a^2(i)]}{E[e_a^2(i)] + \zeta_v^2} \quad (2.4)$$

When the noise power ζ_v^2 is very small, the bound on step size is approximately equal to 2. The plot of excess MSE (EMSE) using different step sizes is shown in Fig. 2.1. As is evident from the plot, choosing a step size greater than the upper bound causes an indefinite increase in EMSE. An optimum step size guarantees convergence, but the EMSE is relatively greater than smaller step sizes.

2.1.2 Steady state Mean Square Error

Steady state Mean Square Error (MSE) can be calculated from Equation (2.3) by considering that when $i \rightarrow \infty$, we have $\tilde{\mathbf{\Omega}}(i) = \tilde{\mathbf{\Omega}}(i-1)$. Therefore, (2.3) becomes:

$$\lim_{i \rightarrow \infty} E[e_a^2(i)] = \frac{\eta \zeta_v^2}{2 - \eta} \quad (2.5)$$

It is clear from Equation (2.5) above that at steady state, the excess MSE of KLMS depends only on step size and noise variance. Further, in the case of Gaussian Kernel, the excess MSE is independent of kernel width σ . Also, if step size is chosen higher than its upper bound as per Equation (2.4), the algorithm doesn't converge in the mean square sense, and its excess MSE increases without bound. Figure 2.2 and Figure 2.3 show the response of algorithm by varying these parameters, and it is matching with analytical result in Equation 2.6 above.

2.1.3 Transient Analysis

For convergence speed of KLMS, as per Equation (2.3), the only parameter that we can change manually is the step size η . Using (2.3), the step size η that will cause maximum change in weight error vector for each iteration i is given by below equation:

$$\eta = \frac{2E[e_a^2(i)]}{E[e_a^2(i)] + \zeta_v^2} \quad (2.6)$$

From above Equation, it is clear that the fastest convergence is obtained when η is assigned the maximum value (Upper bound, as per Equation (2.4)). In other words, KLMS convergence is limited by the upper bound on its step size. As shown in Fig. 2.1, if the step size is chosen for fastest convergence, the excess MSE becomes high, and vice versa. Therefore, in practical cases, one has to find an optimum trade off between convergence speed and excess MSE.

2.1.4 Effect of Kernel size on Convergence behavior

Depending on the type of inputs, the kernel size h will have corresponding effect on the convergence behavior of KLMS. A larger kernel size is suited for the case when the algorithm is learning a smooth mapping between desired response and inputs. On the other hand, a smaller kernel size is better for learning sharp mappings. As shown in Fig. 2.4, Fig 2.5 and Fig 2.6, the kernel size has significant impact on the transient response of the algorithm, but no effect on the EMSE. Fig 2.6 shows the change in EMSE during the 1st iteration of the algorithm using different values for kernel size.

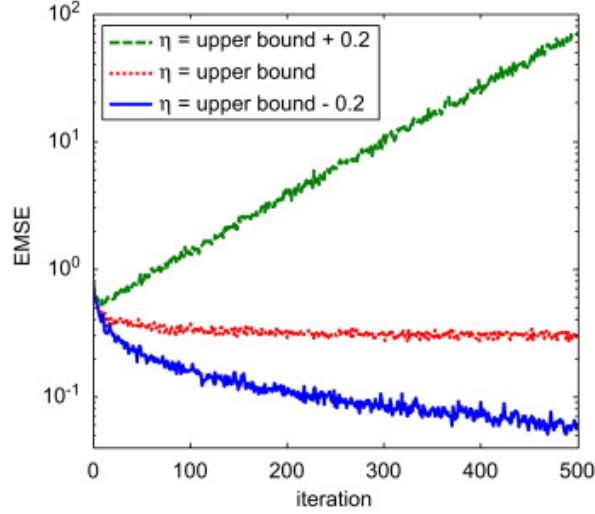


Figure 2.1: Evolution of excess MSE (EMSE) for different step sizes [2]. The upper bound is selected as per Equation (18), and its excess MSE is the highest. For the case when step size is chosen larger than the upper bound, algorithm becomes unstable.

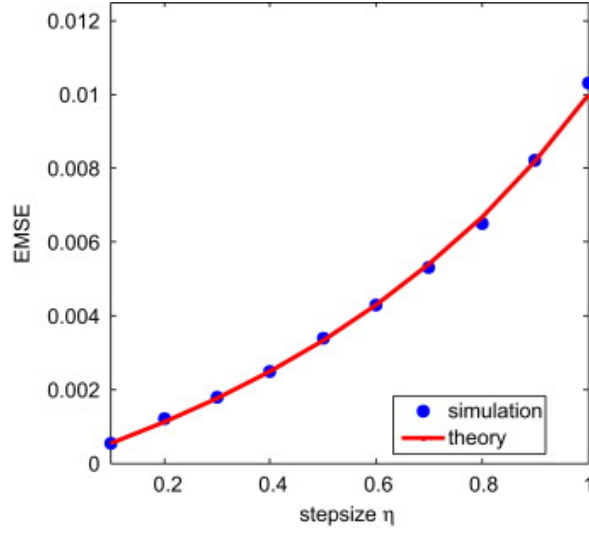


Figure 2.2: EMSE showing monotonic increase due to increase in step size [2]. This behavior is inline with Equation (2.5), as shown by blue dots (simulation).

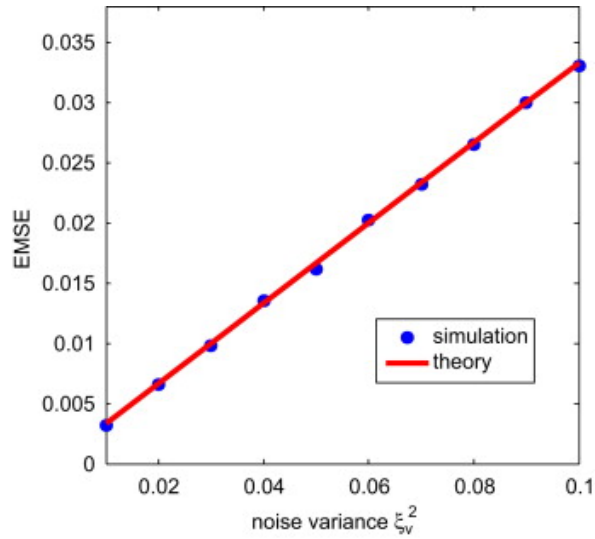


Figure 2.3: Constant increase in EMSE due to noise variance [2].

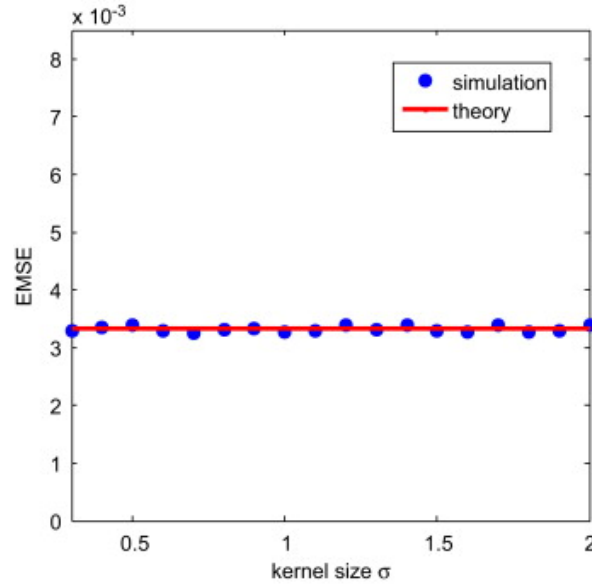


Figure 2.4: Effect of kernel size on EMSE. The EMSE remains invariant to kernel size.
[2]

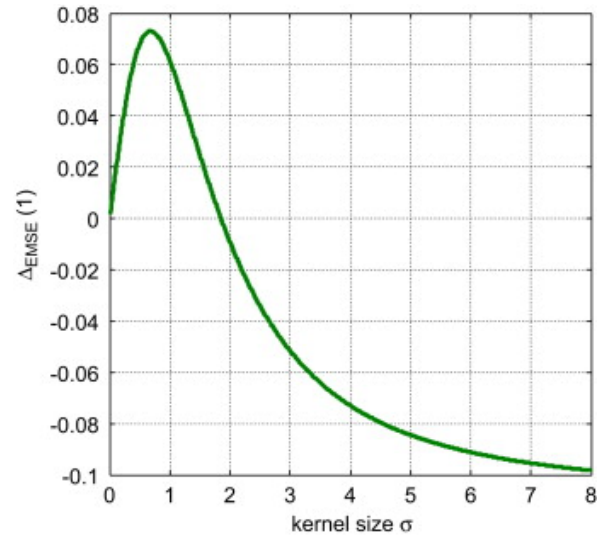


Figure 2.5: Change in EMSE as a function of kernel size during 1st iteration of the algorithm.[2]

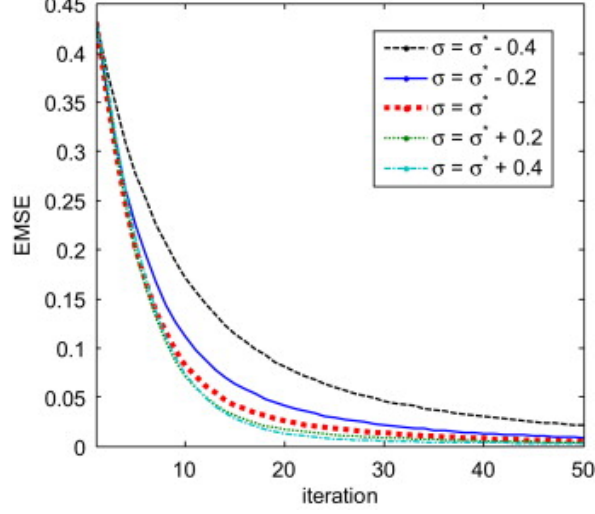


Figure 2.6: Evolution of EMSE during 1st 50 iterations of the algorithm [2]. Above figure, Fig. 2.5 and Fig. 2.6 show that kernel size has an impact on initial convergence (Transient behavior) of the algorithm, but the steady state EMSE remains same.

2.2 Quantized KLMS

In order to limit the growing computational requirements of KLMS, several techniques have been applied to curb the network growth of the algorithm. Much of these techniques have focused on defining rules to accept only useful inputs as the new centers of the network. Recall the KLMS creates a growing network after each iteration i , which in case of Gaussian Kernel, becomes a growing RBF network. Every new input is added as a new center to the existing network, therefore, increasing the network size with each iteration. In Quantized KLMS (QKLMS) algorithm, the inputs space U is quantized by using a vector quantization operator $Q[.]$. The motive behind quantization is to limit the growth of network by only accepting those inputs as new centers which are away from existing quantized inputs. By doing so, inputs which are within the limits of quantization precision will be rounded off by the respective quantized input. Let $\mathcal{Q}$ denote the quantization operator in feature space F , then we can define

the weight update and output Equation of quantized KLMS as:

$$\mathbf{\Omega}(i) = \mathbf{\Omega}(i-1) + \eta e(i) \mathcal{Q}[\phi(i)] \quad (2.7)$$

$$y(n) = \eta \sum_{i=1}^n e(i) \kappa(\mathbf{Q}[\mathbf{x}(i)], \mathbf{x}(n)) \quad (2.8)$$

Equation (2.7) corresponds to quantization done in the feature space, while (2.8) refers to quantization in input space. Let u_q denote the quantized input, then the energy conservation relation for quantized KLMS can be written as:

$$\begin{aligned} \|\tilde{\mathbf{\Omega}}(i)\|_{\mathbb{F}}^2 + \frac{e_a^2(i)}{\kappa(u_q(i), u(i))^2} &= \|\tilde{\mathbf{\Omega}}(i-1)\|_{\mathbb{F}}^2 \\ + \frac{e_p^2(i)}{\kappa(u_q(i), u(i))^2} + \beta_q & \end{aligned} \quad (2.9)$$

It should be noted that the expectation operator is not present in above Equation. Equation 2.9 is simply derived from applying energy conservation relation to the basic apriori and a-posteriori error equations of the algorithm. If we ignore β_q , rest of the above Equation is similar to the one for KLMS. The term β_q has a complicated relation with a priori and a posteriori errors of the algorithm, and interested readers are encouraged to have a look at [3] for more details. The point of interest is to see how quantization of input space affects the mean square behavior of the algorithm. Following the same steps as were done for KLMS, substituting a posteriori error and then applying expectation operator, we get:

$$\begin{aligned} E \left[\|\tilde{\mathbf{\Omega}}(i)\|_{\mathbb{F}}^2 \right] &= E \left[\|\tilde{\mathbf{\Omega}}(i-1)\|_{\mathbb{F}}^2 \right] + \eta^2 E \left[e^2(i) \right] \\ &\quad - 2\eta E \left[e(i) \tilde{\mathbf{\Omega}}(i-1)^T \boldsymbol{\varphi}_q(i) \right] \end{aligned} \quad (2.10)$$

Using above equation i.i.d assumption on the input and additive noise ζ_v^2 , the conditions for mean square convergence can be shown to be:

$$\forall i, \quad \begin{cases} E \left[e_a(i) \tilde{\Omega}(i-1)^T \varphi_q(i) \right] > 0 & (C1) \\ 0 < \eta \leq \frac{2E[e_a(i) \tilde{\Omega}(i-1)^T \varphi_q(i)]}{E[e_a^2(i)] + \sigma_v^2} & (C2) \end{cases} \quad (2.11)$$

Note that the mean square convergence depends on two conditions, (C1) and (C2), unlike KLMS where step size bound is enough. Similarly, the Excess MSE can be shown to be bounded as:

$$\max \left\{ \frac{\eta \sigma_v^2 - 2\xi_\gamma}{2 - \eta}, 0 \right\} \leq \lim_{i \rightarrow \infty} E[e_a^2(i)] \leq \frac{\eta \sigma_v^2 + 2\xi_\gamma}{2 - \eta} \quad (2.12)$$

where $\xi_\gamma = \sqrt{2 - 2 \exp(-(1/2)\gamma^2)} E[\|\mathbf{\Omega}^*\|_{\mathbb{R}}^2]$, γ being the quantization factor which is equal to ratio of quantization size and kernel width σ . Here, by quantization size we mean the euclidean norm of the difference between two input vectors. In the case when quantization size is zero, the resulting quantization factor is zero and Equation (2.12) reduces to Equation (2.5), which is the Excess MSE for KLMS. Fig. 2.7 shows the variation of excess MSE with quantization factor γ when QKLMS is used for static function estimation shown in [3]. Fig. 2.8 shows that larger the quantization factor, smaller the RBF network created by the algorithm. In Fig. 2.9, network size generated by KLMS are shown by using different sparsification techniques. SC-KLMS is referring to the case where input space is compressed using Surprise criterion [5], and NC-KLMS where Novelty Detection technique has been applied [18]. Lastly, in Fig. 2.10, we see the performance comparison of KLMS and QKLMS in terms of excess

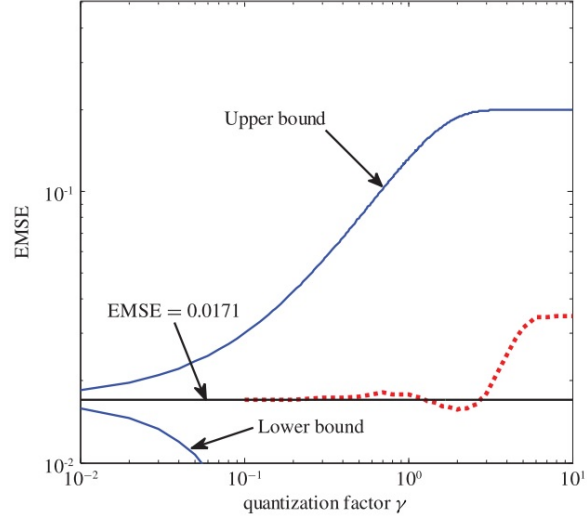


Figure 2.7: Dependence of EMSE on quantization factor γ . Upper and Lower bound in blue are the theoretical estimates as given by Equation 2.12. [3]

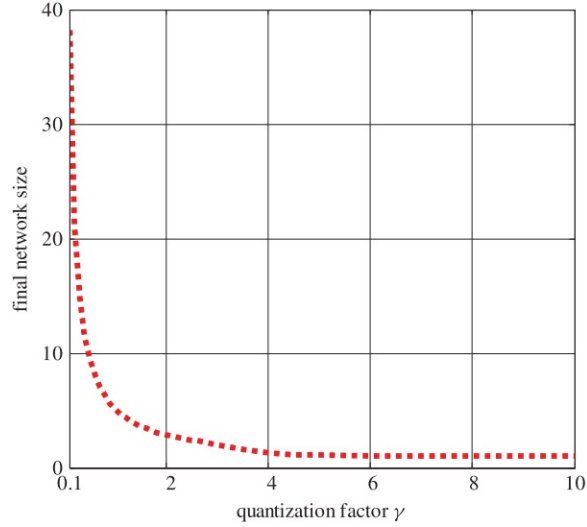


Figure 2.8: Variation of network size (kernel centers in the memory) depending on the quantization factor. [3]

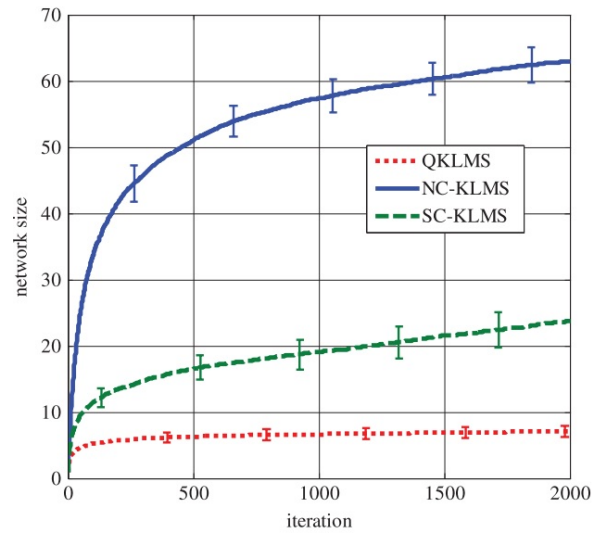


Figure 2.9: Comparison of network size generated by KLMS using different sparsification techniques. [3]

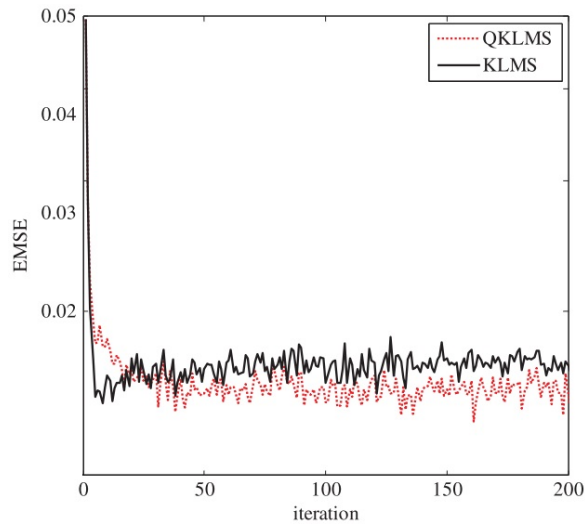


Figure 2.10: EMSE performance of KLMS and QKLMS [3].

MSE. It should be noted that although QKLMS addresses the requirement of growing network size, the resulting EMSE are almost same for both KLMS and QKLMS.

2.3 KLMS based on Conjugate Gradient

In [19], a new variant of KLMS has been proposed based on conjugate gradient. In this approach, the estimation error is minimized by following the conjugate search directions. In this way, the algorithm converges to the optimal solution faster than the KLMS which is based on method of steepest descent. Due to quick convergence, the network generated by this method is smaller compared to KLMS. In other words by using conjugate gradient based KLMS (KLMS-CG), the growing network problem of KLMS is addressed to some extent by achieving faster convergence to steady state MSE. The KLMS-CG can be defined as follows:

$$\left\{ \begin{array}{l} \boldsymbol{\omega}(0) = 0 \\ e(i) = d(i) - \boldsymbol{\omega}(i-1)^T \boldsymbol{\varphi}(i) \\ \boldsymbol{\omega}(i) = \boldsymbol{\omega}(i-1) + \eta \boldsymbol{\rho}(i) \\ \boldsymbol{\rho}(i) = -\boldsymbol{g}(i) + \beta(i) \boldsymbol{\rho}(i-1) \\ \boldsymbol{g}(i) = -e(i) \boldsymbol{\varphi}(i) \\ \beta(i) = \frac{\kappa(\boldsymbol{g}(i), \boldsymbol{g}(i))}{\kappa(\boldsymbol{g}(i-1), \boldsymbol{g}(i-1))} \end{array} \right. \quad (2.13)$$

Unlike KLMS, here $\boldsymbol{\rho}(i)$ is the conjugate search direction, which can be defined as:

$$\boldsymbol{\rho}(i) = -\boldsymbol{g}(i) + \beta(i) \boldsymbol{\rho}(i-1) \quad (2.14)$$

The term $\mathbf{g}(i)$ is equal to $-e(i)\phi(i)$, same as we have in KLMS. The term $\beta(i)$ is known as the conjugate coefficient and is defined as:

$$\beta(i) = \frac{\kappa(\mathbf{g}(i), \mathbf{g}(i))}{\kappa(\mathbf{g}(i-1), \mathbf{g}(i-1))} \quad (2.15)$$

It is evident from above equations that computation cost of KLMS-CG is higher than the original KLMS, but the benefit is faster convergence, as shown in below in Fig. 2.12. In Fig. 2.12, shows that there is a significant improvement in transient

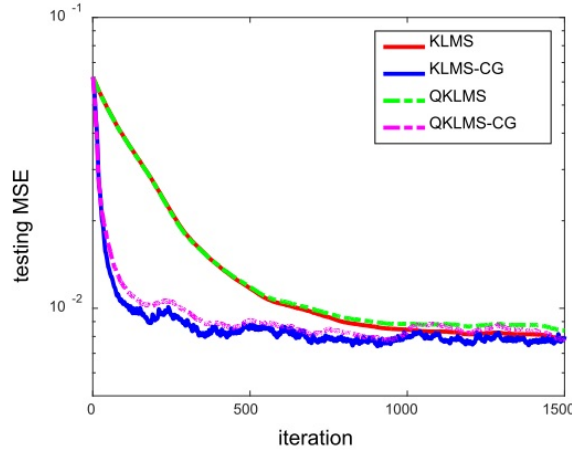


Figure 2.11: MSE comparison of KLMS-CG with KLMS, Quantized KLMS and Quantized KLMS-CG

response of KLMS by using conjugate gradient. Both KLMS-CG and QKLMS-CG have superior convergence speed compared to KLMS and QKLMS.

2.4 KLMS with Constrained Growth

In order to curb the growth of network in KLMS, authors in [4] proposed that a new center should only be added to existing network if it fulfils the linear independence condition. The Linear independence condition is determined by the least squares

solution to below cost function:

$$J(\bar{\beta}) = \left\| \Phi(\mathbf{x}(n)) - \sum_{k=1}^{M_n} \beta_k \Phi(\mathbf{x}(r_{n-1,k})) \right\|^2 \quad (2.16)$$

Here β_k is a real number, $\bar{\beta} = [\beta_1, \dots, \beta_k]^T$, r_{n-1} is the list of indices corresponding to linearly independent vectors identified so far up to iteration $n - 1$, $M(n)$ being the length of r_n and $r_{n-1,k}$ is the k th element of r_{n-1} . If the cost function defined above is close to zero, the incoming input vector $\Phi(\mathbf{x}(n))$ is dependent. Larger the value of $J(\bar{\beta})$, more linearly independent the input vector becomes from the dictionary of existing linearly independent vectors, r_{n-1} . It should be noted that one can set a threshold to define whether the new input vector should be considered linearly dependent or independent. The working methodology of constrained growth KLMS is like this: If an input vector is sufficiently independent of vectors indexed by r_{n-1} , then it is added in the dictionary as a new kernel center. Otherwise, the input is discarded. In this way, the growth of KLMS network can be constrained. In order to see this happening, the output equation of KLMS, Eq. 1.11 can now be written as:

$$y(n+1) = \eta \sum_{i=0}^{n-1} e_{\alpha,n}(i) \kappa(\mathbf{x}(i), \mathbf{x}(n+1)) \quad (2.17)$$

Above Equation shows that it requires one kernel evaluation less than original KLMS at iteration $(n+1)$. Important point to note here is that the benefit of linear dependency of input $x(n)$ at iteration n is taken at the next iteration. In this way, if more inputs are linearly dependent on previous inputs, the network growth will be

limited further. Let δ denote the cost function in Eq. 2.16, which is the threshold for declaring an input sample linearly independent from previous inputs. Fig. 2.13 shows the number of centers accumulated as a function of δ [4]. The algorithm was used to predict Mackey Glass Time Series [20], and the output was tested after 600 training samples. In the case of $\delta = 0$, the network has 600 centers, and the centers decrease as the threshold is increased. Fig. 2.14 shows the evolution of network growth over

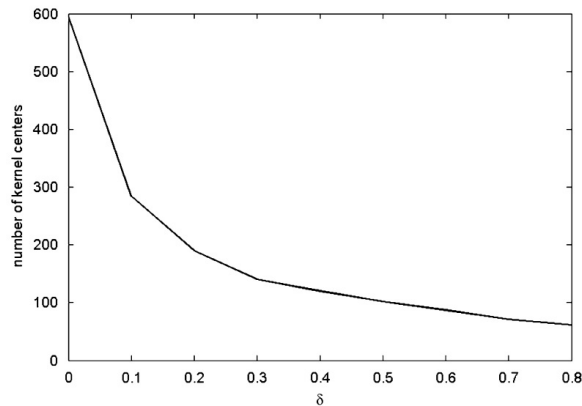


Figure 2.12: Network growth in terms of number of kernel centers as a function of threshold δ while predicting Mackey Glass Time Series [3]

time index for three different threshold values. When $\delta = 0.5$, the network growth is constrained close to 100 centers.

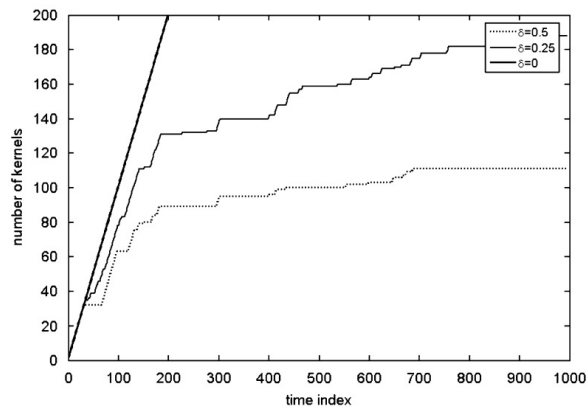


Figure 2.13: Network growth as the algorithm evolves over time for different threshold values δ . [4]

2.5 Information Theoretic KLMS

In order to address the growing network size of KLMS, an information theoretic approach to design Kernel Adaptive Filters has been presented in [5], known as Surprise Criterion (SC). SC is an input sparsification technique, like Novelty Criterion [18], Coherence Criterion [6] and approximate Linear Dependency (ALD) [21]. The main idea is to select only those inputs which have enough useful information for the learning algorithm. If an input doesn't contain much information compared to the learning algorithm, it is considered redundant. Given an input $\mathbf{u}$, and desired response d from a learning machine $\mathcal{T}$, the surprise criterion can be defined as [5]:

$$S_{\mathcal{T}}(\mathbf{u}, d) = -\ln p(\mathbf{u}, d \mid \mathcal{T}) \quad (2.18)$$

The surprise criterion is the negative log likelihood of the joint probability realized by the learning machine $\mathcal{T}$. Notice that it is different from the Shannon Information measure where true joint probability of the pair $(\mathbf{u}, d \mid \mathcal{T})$ is required [22]. The Surprise Criterion gives a high measure if the learning algorithm is least expecting the desired response d corresponding to input $\mathbf{u}$. In other words, if the desired response by the algorithm is highly likely, there is not much to learn and the algorithm can discard adding the new input to the existing network of centers. And if it is otherwise, the network will expand as usual. If the outputs of algorithm are assumed to be jointly Gaussian, then by using Gaussian Process Regression [23], following expression for

Surprise measure is obtained when applied to KLMS [5]:

$$S(i+1) = \frac{1}{2} \ln r(i+1) + \frac{e^2(i+1)}{2r(i+1)} - \ln[p(\mathbf{u}(i+1) | \mathcal{T}(i))] \quad (2.19)$$

Here $S(i+1)$ is the surprise measure calculated for new input $u(i+1)$. $e(i+1)$ is the prediction error, and $r(i+1)$ is the metric calculated as:

$$r(i+1) = \lambda + \kappa(\mathbf{u}(i+1), \mathbf{u}(i+1)) - \max_{\forall \mathbf{c}_j \in \mathcal{C}(i)} \frac{\kappa^2(\mathbf{u}(i+1), \mathbf{c}_j)}{\kappa(\mathbf{c}_j, \mathbf{c}_j)} \quad (2.20)$$

λ is a regularisation parameter, and $\mathbf{c}_j$ are the centers accumulated in the memory after i iterations.

- If no a priori information is known about the inputs, the value $\ln[p(\mathbf{u}(i+1) | \mathcal{T}(i))]$ can be assumed to be a constant.
- A new input is first evaluated by using expressions above. If the Surprise measure is significantly high, the input is considered useful and is added as a new center in the set $\mathcal{C}(i)$. If surprise measure is low, it is assumed that the current learning algorithm with existing centers in the memory is capable enough to predict the desired value, and hence, the input is not added in the memory and is processed as usual by the algorithm. This evaluation results in limiting the growing memory requirements of the original KLMS algorithm.
- Both prediction error $e(i+1)$ and the term $r(i+1)$ contribute in measuring the surprise in the new input sample. The prediction error $e(i+1)$ is the ability of the learning system to correctly predict future value. And the term $r(i+1)$ is

incorporating the distance metric between new input and existing centers in the set $\mathcal{C}(i)$.

- By analyzing Eq. (2.20), we can conclude that the surprise measure is very sensitive to the maximum distance between new input and existing centers. If an input is in close proximity of the existing centers, the value of $r(i+1)$ will be high, resulting in relatively lower value of surprise. The prediction error will decide how much lower the value can go, but the measure is dominated by the maximum distance.

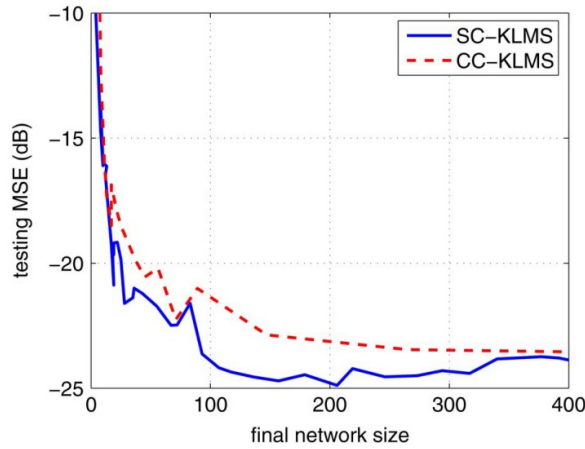


Figure 2.14: Network size comparison of KLMS having Suprise Criterion (SC-KLMS) and KLMS having Coherence Criterion (KLMS-CC) [5], [6]

CHAPTER 3

WEIGHTED GAUSSIAN KLMS

In this chapter, a novel Weighted Kernel Least Mean Square (WKLMS) algorithm is proposed by introducing a weighted Gaussian Kernel. The learning behavior of the WKLMS algorithm is studied. Mean Square Error (MSE) analysis shows that the WKLMS algorithm outperforms both the Least Mean Square (LMS) and KLMS algorithms in terms of transient state as well as steady state responses. We study the effect of weighted Gaussian Kernel on the associated Kernel matrix, its eigenvalue spread and distribution, and show how these parameters affect the convergence behavior of the algorithm. Both the transient and steady-state mean-square-error (MSE) behaviors of the WKLMS algorithm are studied and stability bound is derived. For a non-stationary environment, tracking analysis for a correlated random walk channel is presented. We also prove that the steady state excess MSE (EMSE) of the WKLMS is Schur convex function of the weight elements in its kernel weight matrix and hence it follows the majorization of the kernel weight elements. This helps to decide which kernel weight matrix can provide better MSE performance. Simulations results are provided to contrast the performance of the proposed WKLMS with that of its coun-

terparts KLMS and LMS algorithms. In addition, the derived analytical results of the WKLMS are validated by simulation results for various values step-size.

3.1 Construction of WKLMS

A Gaussian Kernel with $\mathbf{u}_i$ and $\mathbf{u}_j$ as input data vectors of dimension m is defined as [8]

$$\kappa(\mathbf{u}_i, \mathbf{u}_j) = \exp(-\|\mathbf{u}_i - \mathbf{u}_j\|^2/2h), \quad (3.1)$$

where h is the Kernel bandwidth. We proposed the following weighted Gaussian Kernel¹

$$\kappa_{\Sigma}(\mathbf{u}_i, \mathbf{u}_j) = \exp(-\|\mathbf{u}_i - \mathbf{u}_j\|_{\Sigma}^2/2h), \quad (3.2)$$

where Σ is an $m \times m$ weighting matrix. This can be a full matrix with all non-zero entries. However, a full matrix Σ can be diagonalized using eigenvalue decomposition $\Sigma = U\Sigma'U^T$ which results in transformed input vectors $\mathbf{u}_i' = U^T\mathbf{u}_i$ and $\mathbf{u}_j' = U^T\mathbf{u}_j$. Thus, we consider Σ as a diagonal weighting matrix in the rest of this chapter.

Similar to the case of the KLMS, the output of the proposed WKLMS with training size N and input data vector $\mathbf{u}_n$ can be computed as

$$y_n = \eta \sum_{i=1}^N e_i^a \exp(-\|\mathbf{u}_i - \mathbf{u}_n\|_{\Sigma}^2/2h), \quad (3.3)$$

where η is the step size and e_i^a is the a priori error, which at the n^{th} iteration can be

computed as

$$e_n^a = y_n - \eta \sum_{i=1}^{n-1} e_i^a \exp(-\|\mathbf{u}_i - \mathbf{u}_n\|_{\Sigma}^2/2h), \quad (3.4)$$

The Equations (3.3) and (3.4) defines the proposed Weighted KLMS (WKLMS) algorithm. To obtain recursive formulation of the algorithm, consider a finite size dictionary is considered in [24] and we utilize the same approach in this work. For a dictionary of size M , the vector $\boldsymbol{\kappa}_{\Sigma}(n)$ represents a vector of kernels for all the inputs at time n , that is

$$\boldsymbol{\kappa}_{\Sigma}(n) = [\kappa_{\Sigma}(\mathbf{u}_n, \mathbf{u}_1), \kappa_{\Sigma}(\mathbf{u}_n, \mathbf{u}_2), \dots, \kappa_{\Sigma}(\mathbf{u}_n, \mathbf{u}_M)]^T, \quad (3.5)$$

Now, by representing the weight vector as $\mathbf{w}_n$, its update recursion is given by

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \eta e_n \boldsymbol{\kappa}_{\Sigma}(n). \quad (3.6)$$

where e_n is the estimation error of the adaptive filter which can be formulated as

$$e_n = d_n - \mathbf{w}_n^T \boldsymbol{\kappa}_{\Sigma}(n). \quad (3.7)$$

The Gram matrix of the weighted Gaussian Kernel in (3.2) can be defined as:

$$\mathbf{G}_w = \begin{bmatrix} 1 & \kappa_{\Sigma}(u_1, u_2) & \cdot & \cdot & \kappa_{\Sigma}(u_1, u_M) \\ \kappa_{\Sigma}(u_2, u_1) & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \kappa_{\Sigma}(u_{M-1}, u_M) \\ \kappa_{\Sigma}(u_M, u_1) & \cdot & \cdot & \cdot & 1 \end{bmatrix}. \quad (3.8)$$

The main diagonal elements are equal to unity since

$$\begin{aligned} \kappa_{\Sigma}(\mathbf{u}_i, \mathbf{u}_i) &= \exp(-(\mathbf{u}_i - \mathbf{u}_i)^T \Sigma (\mathbf{u}_i - \mathbf{u}_i)/2h) \\ &= \exp(0) \\ &= 1. \end{aligned} \quad (3.9)$$

Moreover, for $\mathbf{u}_i \neq \mathbf{u}_j$ the following holds:

$$0 < \kappa_{\Sigma}(\mathbf{u}_i, \mathbf{u}_i) < 1. \quad (3.10)$$

Equations (3.9) and (3.10) ensure that the Gram matrix $\mathbf{G}_{\Sigma}$ is always positive definite, and hence, (3.2) qualifies as a valid Kernel.

Let $\lambda_1, \lambda_2, \dots, \lambda_M$ be the M eigenvalues of the Gram matrix in (3.8), the eigenvalue spread is defined as the ratio of the largest eigenvalue to the smallest and is given by

$$\rho_{\Sigma} = \lambda_{max}/\lambda_{min}. \quad (3.11)$$

The convergence behavior of the WKLMS algorithm is related to the Gram matrix. It can be easily shown that the mis-adjustment of the WKLMS algorithm is related to the Gram matrix as [12]

$$M = \frac{\eta}{2N} \text{tr}[\mathbf{G}_{\Sigma}], \quad (3.12)$$

where tr is the trace operator.

Using above definitions, some important results are listed below.

Theorem 2.1: The misadjustment of the WKLMS algorithm remains invariant for any choice of the weighting matrix Σ

Proof. Proof: Regardless of the weights in the weighting matrix, (3.8) tells us that the main diagonal elements of Gram matrix will always be equal to unity. Therefore,

$$\text{tr}[\mathbf{G}_{\Sigma}] = \text{tr}[\mathbf{G}], \quad (3.13)$$

which shows that for any choice of Σ , the misadjustment of the algorithm, as defined in (3.12), remains invariant. ■

3.1.1 Effect of Weights on Gram Matrix

Assuming that each input data vector $\mathbf{u}_i$ is of size d , the corresponding weighting matrix can be written as

$$\mathbf{\Sigma} = \begin{bmatrix} w_{ii} & 0 & . & . & 0 \\ 0 & w_{22} & & & . \\ . & & . & & . \\ . & & & . & . \\ 0 & . & . & . & w_{dd} \end{bmatrix}. \quad (3.14)$$

From (3.7), (3.8), (3.9) and (3.14), it can be concluded that:

- If the weights w_{ii} are chosen very large, the corresponding kernel evaluations in (3.7) will be very small in magnitude. Since Kernels are evaluating inner products in feature space, therefore, the input data vectors will become indistinguishable, resulting in poor performance by the WKLMS algorithm. Moreover, since all the off diagonal elements in (3.8) will be much smaller than unity, the resulting eigenvalues and hence the eigenvalue spread will be much lower compared to the Gram matrix of (3.1). Eigenvalue distribution for large weights is shown in Fig. 3.1, where a training set of 50 data points has been taken, which is further blocked into vector form with each vector of length 5. The data points are normally distributed with unit variance and zero mean.
- If the weights w_{ii} are chosen very small, the corresponding Gram matrix will have large values, and consequently, large eigenvalue spread. Fig. 3.2 shows the

eigenvalue distribution for very small values of weights. As a comparison, Fig. 3.3 shows eigenvalue distribution when all weights are equal to unity.

- Equation (3.12) implies that the mis-adjustment factor for both the KLMS and the WKLMS algorithms will remain same if the step size η is kept same. However, we found that the mis-adjustment factor depends on both the choice and order of the weights in the weighting matrix.
- Selecting the weights in the increasing order of magnitude gives higher precedence to novel inputs in the input data vector $\mathbf{v}_i$. Depending on the weights, the eigenvalue distribution is affected, as shown in Fig. 3.4.
- Fig. 3.5 shows that the weight selection in decreasing order of their magnitude does not significantly change the eigenvalue distribution.
- The weighting matrix in (3.14) acts as a linear transformation in the feature space, rotating and translating the mapped data points represented by the feature vectors.

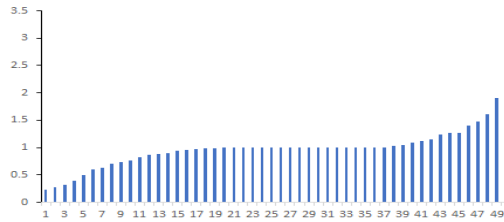


Figure 3.1: Eigenvalue distribution of Gram matrix $\mathbf{G}_w$. Input data is Gaussian distributed with zero mean and unit variance. $N = 50$, $d = 5$ and weights = $[16, 16, 16, 16, 16]$.



Figure 3.2: Eigenvalue distribution of Gram matrix $\mathbf{G}_w$. Input data is Gaussian distributed with zero mean and unit variance. $N = 50$, $d = 5$ and weights = $[0.01, 0.01, 0.01, 0.01, 0.01]$.

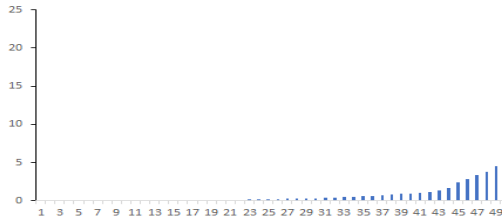


Figure 3.3: Eigenvalue distribution of Gram matrix $\mathbf{G}_w$. Input data is Gaussian distributed with zero mean and unit variance. $N = 50$, $d = 5$ and weights = $[1, 1, 1, 1, 1]$.

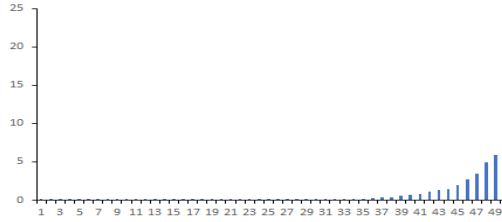


Figure 3.4: Eigenvalue distribution of Gram matrix $\mathbf{G}_w$. Input data is Gaussian distributed with zero mean and unit variance. $N = 50$, $d = 5$ and weights = $[0.1, 0.1, 1, 2, 2]$.

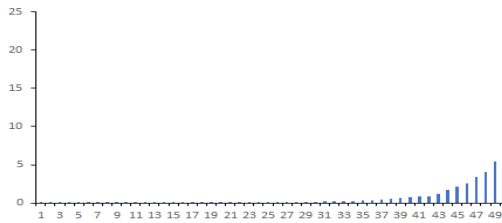


Figure 3.5: Eigenvalue distribution of Gram matrix $\mathbf{G}_w$. Input data is Gaussian distributed with zero mean and unit variance. $N = 50$, $d = 5$ and weights = $[2, 2, 1, 0.1, 0.1]$.

3.2 Convergence Analysis

In this section, we provide the convergence analysis of the proposed WKLMS algorithm both in the mean and mean-square sense. For this, we derive expressions to analyze both the transient and the steady-state behavior of the proposed algorithm. The stability bounds are thus obtained that guarantee the convergence of the WKLMS algorithm.

3.2.1 System Model

Consider the system identification scenario in which the desired response d_n is the output of a channel with impulse response $\mathbf{w}_o$, that is,

$$d_n = \mathbf{w}_o^T \boldsymbol{\kappa}_{\Sigma}(n) + v_n \quad (3.15)$$

where v_n is a zero mean i.i.d. noise sequence with variance σ_v^2 . Let $\tilde{\mathbf{w}}_n = \mathbf{w}_o - \mathbf{w}_n$ denote the weight error vector. Thus, the estimation error defined in (3.7) can be reformulated as

$$e_n = (\mathbf{w}_o - \mathbf{w}_{n-1})^T \boldsymbol{\kappa}_{\Sigma}(n) + v_n = \tilde{\mathbf{w}}_{n-1}^T \boldsymbol{\kappa}_{\Sigma}(n) + v_n, \quad (3.16)$$

By subtracting $\mathbf{w}_n$ from both sides of (3.6) and utilizing the above formulation of e_n , we can show that

$$\tilde{\mathbf{w}}_n = \tilde{\mathbf{w}}_{n-1} - \eta e_n \boldsymbol{\kappa}_{\Sigma}(n) \quad (3.17)$$

We assume that the sequences $\{v_i\}$ and $\{\mathbf{u}_i\}$ are mutually independent which is well justified in real practice. Consequently, the sequence of kernel vectors $\{\boldsymbol{\kappa}_{\Sigma}(n)\}$ will also be independent of the noise sequence $\{v_i\}$. Moreover, we also assume that the input vectors are Gaussian distributed, i.e. $\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_u)$. We also assume that the input vectors are independent but their entries are correlated.

3.2.2 Mean Behavior

Subtracting $\mathbf{w}^o$ from both sides of (3.6) and taking expectation to obtain

$$E[\tilde{\mathbf{w}}_n] = (\mathbf{I} - \eta \mathbf{R}_{\kappa\kappa}) E[\tilde{\mathbf{w}}_{n-1}] \quad (3.18)$$

where $\mathbf{R}_{\kappa\kappa} = E[\boldsymbol{\kappa}_{\Sigma}(n) \boldsymbol{\kappa}_{\Sigma}^T(n)]$ is the correlation matrix of the dictionary kernel vector $\boldsymbol{\kappa}_{\Sigma}(n)$. After employing the eigenvalue decomposition $\mathbf{R}_{\kappa\kappa} = \mathbf{U} \boldsymbol{\Lambda}_{\kappa\kappa} \mathbf{U}^T$, the above recursion transformed to

$$E[\bar{\mathbf{w}}_n] = (\mathbf{I} - \eta \boldsymbol{\Lambda}_{\kappa\kappa}) E[\bar{\mathbf{w}}_{n-1}] \quad (3.19)$$

where $\bar{\mathbf{w}}_n = \mathbf{U}^T \tilde{\mathbf{w}}_n$ is transformed weight error vector. In order to analyze the above, we need to evaluate the entries of the correlation matrix $\mathbf{R}_{\kappa\kappa}$ which are defined as

$$[\mathbf{R}_{\kappa\kappa}]_{ij} = \begin{cases} E[\kappa^2(\mathbf{u}_n, \mathbf{u}_i)], & i = j \\ E[\kappa(\mathbf{u}_n, \mathbf{u}_i) \kappa(\mathbf{u}_n, \mathbf{u}_j)], & i \neq j \end{cases} \quad (3.20)$$

We introduce the following notations:

$$\|\mathbf{x}\|_{\mathbf{Q}_2}^2 = \|\mathbf{u}_n - \mathbf{u}_i\|_{\Sigma}^2, \quad (3.21)$$

$$\text{and } \|\mathbf{z}\|_{\mathbf{Q}_3}^2 = \|\mathbf{u}_n - \mathbf{u}_i\|_{\Sigma}^2 + \|\mathbf{u}_n - \mathbf{u}_j\|_{\Sigma}^2, \quad (3.22)$$

where

$$\mathbf{x} = (\mathbf{u}_n^T \mathbf{u}_i^T)^T, \quad \mathbf{z} = (\mathbf{u}_n^T \mathbf{u}_i^T \mathbf{u}_j^T)^T \quad (3.23)$$

and

$$\mathbf{Q}_2 = \begin{bmatrix} \Sigma & -\Sigma \\ -\Sigma & \Sigma \end{bmatrix}, \quad \mathbf{Q}_3 = \begin{bmatrix} 2\Sigma & -\Sigma & -\Sigma \\ -\Sigma & \Sigma & \mathbf{0} \\ -\Sigma & \mathbf{0} & \Sigma \end{bmatrix} \quad (3.24)$$

where $\mathbf{0}$ is the null matrix. Finally, by using the result of [24], the entries of $\mathbf{R}_{\kappa}$ are given by

$$[\mathbf{R}_{\kappa\kappa}]_{ij} = \begin{cases} \det\{\mathbf{I}_{2m} + 2\mathbf{Q}_2\mathbf{R}_x/h^2\}^{-\frac{1}{2}}, & i = j \\ \det\{\mathbf{I}_{3m} + \mathbf{Q}_3\mathbf{R}_z/h^2\}^{-\frac{1}{2}}, & i \neq j \end{cases} \quad (3.25)$$

where $\mathbf{I}_{2m}$ and $\mathbf{I}_{3m}$ are the identity matrix of the size of $2m \times 2m$ and $3m \times 3m$, respectively. Since, we have assumed that the input vectors $\{\mathbf{u}_n\}$ are independent, we can show that the correlation matrices $\mathbf{R}_x$ and $\mathbf{R}_z$ can be calculated as

$$\mathbf{R}_x = \begin{bmatrix} \mathbf{R}_u & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_u \end{bmatrix}, \quad \mathbf{R}_z = \begin{bmatrix} \mathbf{R}_u & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_u & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_u \end{bmatrix} \quad (3.26)$$

Finally, by studying the recursion (3.19) with the aid of eigenvalue decomposition of $\mathbf{R}_\kappa$, the mean behavior of the WKLMS can be evaluated. Thus, it can be shown that the WKLMS will converge in the mean sense if the following bound is satisfied

$$0 < \eta < \frac{2}{\lambda_{max}} \quad (3.27)$$

where λ_{max} is the maximum eigenvalue of $\mathbf{\Lambda}_{\kappa\kappa}$.

3.2.3 Mean-Square Behavior

The main aim of this section is to study the transient and steady-state behavior of the excess MSE (EMSE) for the WKLMS algorithm. The EMSE (denoted by ζ) is defined as

$$\zeta_n = E[e_n^2] - \sigma_v^2 \quad (3.28)$$

If we define *a priori* estimation error as

$$e_{an} = \tilde{\mathbf{w}}_{n-1}^T \boldsymbol{\kappa}_\Sigma(n), \quad (3.29)$$

Then, the EMSE of the WKLMS can be expressed as

$$\zeta_n = E[e_{an}^2] = E\|\tilde{\mathbf{w}}_{n-1}\|_{\mathbf{R}_{\kappa\kappa}}^2 \quad (3.30)$$

Next, by employing the eigenvalue decomposition $\mathbf{R}_{\kappa\kappa} = \mathbf{U}\mathbf{\Lambda}_{\kappa\kappa}\mathbf{U}^T$, the EMSE can be reformulated as

$$\zeta_n = E\|\bar{\mathbf{w}}_{n-1}\|_{\mathbf{\Lambda}_{\kappa\kappa}}^2 = E\|\bar{\mathbf{w}}_{n-1}\|_{\boldsymbol{\lambda}}^2 \quad (3.31)$$

where $\boldsymbol{\lambda}$ is the vector containing diagonal elements of $\mathbf{\Lambda}_{\kappa\kappa}$. In order to investigate the mean square behavior of the proposed WKLMS algorithm, we employ the approach of energy conservation [10]. As a result, the transient EMSE learning curve is found to be

$$\zeta_n = E\|\mathbf{w}_o\|_{\mathbf{F}^n\boldsymbol{\lambda}}^2 + \eta^2\sigma_v^2\boldsymbol{\lambda}^T(\mathbf{I} + \mathbf{F} + \cdots + \mathbf{F}^{n-1})\boldsymbol{\lambda} \quad (3.32)$$

where

$$\mathbf{F} = \mathbf{I} - 2\eta\mathbf{\Lambda}_{\kappa\kappa} + \eta^2\mathbf{\Lambda}_{\kappa}^2 + \eta^2\boldsymbol{\lambda}\boldsymbol{\lambda}^T \quad (3.33)$$

To obtain the steady-state performance, the above recursion can be analyzed at $n \rightarrow \infty$ which results in

$$\zeta_\infty = \eta^2\sigma_v^2\boldsymbol{\lambda}^T(\mathbf{I} - \mathbf{F})^{-1}\boldsymbol{\lambda} = \frac{\eta\sigma_v^2\sum_{k=1}^M\frac{\lambda_k}{2-\eta\lambda_k}}{1-\eta\sum_{k=1}^M\frac{\lambda_k}{2-\eta\lambda_k}} \quad (3.34)$$

The mean square stability can be obtained if the magnitude of the eigenvalues of matrix $\mathbf{F}$ is bounded by unity, i.e., $|\lambda_{\mathbf{F}}| < 1$ which results in

$$\sum_{k=1}^M\frac{\eta\lambda_k}{2-\eta\lambda_k} < 1, \quad (3.35)$$

where λ_k is the k^{th} eigenvalue of $\mathbf{R}_{\kappa\kappa}$. Similar to the EMSE, transient and steady-state MSD for the WKLMS can also be obtained. Complete summary of the performance

of the WKLMS is provided in Table below.

Performance Measure	Analytical Result
EMSE Learning Curve	$\zeta_n = E\ \mathbf{w}_o\ _{\mathbf{F}^n}^2 + \eta^2 \sigma_v^2 \boldsymbol{\lambda}^T (\mathbf{I} + \mathbf{F} + \dots + \mathbf{F}^{n-1}) \boldsymbol{\lambda}$
MSD Learning Curve	$\text{MSD}_n = E\ \mathbf{w}_o\ _{\mathbf{F}^n \mathbf{1}}^2 + \eta^2 \sigma_v^2 \boldsymbol{\lambda}^T (\mathbf{I} + \mathbf{F} + \dots + \mathbf{F}^{n-1}) \mathbf{1}$
Mean Stability	$0 < \mu < 2/\lambda_{max}$
Mean Square Stability	$\sum_{k=1}^M \frac{\eta \lambda_k}{2 - \eta \lambda_k} < 1$
Steady-state EMSE	$\zeta_\infty = \frac{\eta \sigma_v^2 \sum_{k=1}^M \frac{\lambda_k}{2 - \eta \lambda_k}}{1 - \eta \sum_{k=1}^M \frac{\lambda_k}{2 - \eta \lambda_k}}$
Steady-state MSD	$\text{MSD}_\infty = \frac{\eta \sigma_v^2 \sum_{k=1}^M \frac{1}{2 - \eta \lambda_k}}{1 - \eta \sum_{k=1}^M \frac{1}{2 - \eta \lambda_k}}$

Table 3.1: Summary of analytical results for various performance measures of the WKLMS algorithm.

3.3 Performance analysis of WKLMS

In this section, we provide an analysis that can help us to know which weights of the WKLMS algorithm can provide us lower steady-state EMSE. Before proceeding to the analysis, we start the discussion with the question: Given two diagonal weight matrices $\mathbf{W}_1$ and $\mathbf{W}_2$, which of the weights will result in lower steady-state EMSE for the WKLMS algorithm? To answer this, we utilize the concept of Majorization and Schur convexity as introduced in [25]. These concepts are briefly outlined next.

Majorization Property

Majorization property deals with the spread of elements in vectors [26]. Formally, the concept of majorization can be defined as follows [26]

For any two vectors $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^m$ with descending order components $x_1 \geq x_2 \geq \dots \geq x_m \geq 0$ and $y_1 \geq y_2 \geq \dots \geq y_m \geq 0$, the vector $\mathbf{x}$ majorizes the vector $\mathbf{y}$ (written as $\mathbf{x} \succ \mathbf{y}$) if

$$\sum_{i=1}^k x_i \geq \sum_{i=1}^k y_i, \quad k = 1, 2, \dots, m-1, \quad \text{and} \quad \sum_{i=1}^m x_i = \sum_{i=1}^m y_i. \quad (3.36)$$

Schur-convex Functions

Any function that preserve the ordering of majorization is called *Schur-convex* function [26]. More specifically, we can define Schur-convex functions as follows [26]:

A *Schur-convex* function, is a function $f : \mathbb{R}^M \rightarrow \mathbb{R}$, for which $\mathbf{x} \succ \mathbf{y}$ implies $f(\mathbf{x}) \geq f(\mathbf{y})$.

Thus, we want to utilize the concept of majorization to quantify the impact spread of the diagonal matrices $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ on the steady-state EMSE. By defining the vectors $\boldsymbol{\sigma}_i = \Delta \text{diag}(\mathbf{\Sigma}_i)$, we can say that $\boldsymbol{\sigma}_1$ majorizes $\boldsymbol{\sigma}_2$ as [26]

$$\sum_{i=1}^j \sigma_1(i) \geq \sum_{i=1}^j \sigma_2(i), \quad j = 1, 2, \dots, m-1 \quad (3.37)$$

$$\sum_{i=1}^m \sigma_1(i) = \sum_{i=1}^m \sigma_2(i) \quad (3.38)$$

where $\sigma_1(i)$ and $\sigma_2(i)$ represent the i^{th} elements in the vectors $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$, respectively. Our aim is to study the impact of spread of the diagonal matrix on the EMSE of the WKLMS algorithm. It should be noted that the EMSE are the scalar valued function

of vectors $\boldsymbol{\sigma}_i$. Thus, our study is related to the Schur-convexity of the EMSE also which will help us to conclude which weight matrix will result in lower EMSE of the WKLMS algorithm. The proof for the Schur-convexity of the EMSE is provided next.

3.3.1 Schur Convex EMSE of WKLMS

In this section, we provide the proof for the Schur-convexity of the EMSE for the WKLMS algorithm. For that, we utilize the approach given in [25] which states that if any scalar function of vector $\boldsymbol{\sigma}$ (i.e., $f(\boldsymbol{\sigma})$) is symmetric in its argument, that is, permutation of the elements of $\boldsymbol{\sigma}$ does not change the value of $f(\boldsymbol{\sigma})$, then the Schur-convexity can be proved if the following condition is satisfied:

$$\left(\boldsymbol{\sigma}(1) - \boldsymbol{\sigma}(2)\right) \left[\frac{\partial f(\boldsymbol{\sigma})}{\partial \boldsymbol{\sigma}(1)} - \frac{\partial f(\boldsymbol{\sigma})}{\partial \boldsymbol{\sigma}(2)} \right] \geq 0. \quad (3.39)$$

It can be easily seen from the expression for the steady-state EMSE that is symmetric w.r.t. eigenvalues $\{\lambda_i\}$. Moreover, since the weight matrices used in WKLMS are diagonal matrices, it can be easily conclude that the EMSE of the WKLMS is symmetric w.r.t. elements in $\boldsymbol{\sigma}$. Thus, in order to prove that the EMSE of the WKLMS (denoted by ζ_∞) is Schur-convex, we need to simply show that

$$\left(\boldsymbol{\sigma}(1) - \boldsymbol{\sigma}(2)\right) \left[\frac{\partial \zeta_\infty}{\partial \boldsymbol{\sigma}(1)} - \frac{\partial \zeta_\infty}{\partial \boldsymbol{\sigma}(2)} \right] \geq 0 \quad (3.40)$$

In order to test the above condition, we utilize the expression for the steady-state EMSE of the WKLMS given in (3.34). We start by defining the following variable

$$S \triangleq \sum_{k=1}^M \frac{\lambda_k}{2 - \eta\lambda_k} \quad (3.41)$$

Thus, the $\frac{\partial \zeta_\infty}{\partial \sigma(1)}$ can be obtained as

$$\frac{\partial \zeta_\infty}{\partial \sigma(1)} = \frac{\eta\sigma_v^2(1 - \eta S) \frac{\partial S}{\partial \sigma(1)} - \eta\sigma_v^2 S \frac{\partial}{\partial \sigma(1)}(1 - \eta S)}{(1 - \eta S)^2} = \frac{\eta\sigma_v^2 \frac{\partial S}{\partial \sigma(1)}}{(1 - \eta S)^2} \quad (3.42)$$

To simplify the analysis, we consider white input (i.e., $\mathbf{R}_u = \mathbf{I}_m$) which results in $\mathbf{R}_x = \mathbf{I}_{2m}$ and $\mathbf{R}_z = \mathbf{I}_{3m}$. Moreover, we consider the case of dictionary size $M = 2$ to make the analysis tractable². As a result, the correlation matrix $\mathbf{R}_\kappa$ will become

$$\mathbf{R}_{\kappa\kappa} = \begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \quad (3.43)$$

where

$$r(0) = \det\{\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2\}^{-\frac{1}{2}}, \quad \text{and} \quad r(1) = \det\{\mathbf{I}_{2m} + \mathbf{Q}_3/h^2\}^{-\frac{1}{2}} \quad (3.44)$$

Thus, the eigenvalues of the correlation matrix $\mathbf{R}_{\kappa\kappa}$ will be

$$\lambda_1 = r(0) + r(1), \quad \text{and} \quad \lambda_2 = r(0) - r(1) \quad (3.45)$$

Thus, for $M = 2$ with λ_1 and λ_2 given in (3.45), the derivative $\frac{\partial S}{\partial \sigma(1)}$ can be obtained as follows

$$\begin{aligned}\frac{\partial S}{\partial \sigma(1)} &= \left(\frac{(2 - \eta\lambda_1)\frac{\partial \lambda_1}{\partial \sigma(1)} - \lambda_1(0 - \eta\frac{\partial \lambda_1}{\partial \sigma(1)})}{(2 - \eta\lambda_1)^2} + \frac{(2 - \eta\lambda_2)\frac{\partial \lambda_2}{\partial \sigma(1)} - \lambda_2(0 - \eta\frac{\partial \lambda_2}{\partial \sigma(1)})}{(2 - \eta\lambda_2)^2} \right) \\ &= 2 \left(\frac{\frac{\partial r(0)}{\partial \sigma(1)} + \frac{\partial r(1)}{\partial \sigma(1)}}{(2 - \eta\lambda_1)^2} + \frac{\frac{\partial r(0)}{\partial \sigma(1)} - \frac{\partial r(1)}{\partial \sigma(1)}}{(2 - \eta\lambda_2)^2} \right)\end{aligned}\quad (3.46)$$

As a result, the $\frac{\partial \zeta_\infty}{\partial \sigma(1)}$ is found to be

$$\frac{\partial \zeta_\infty}{\partial \sigma(1)} = \frac{2\eta\sigma_v^2}{(1 - \eta S)^2} \left(\frac{\frac{\partial r(0)}{\partial \sigma(1)} + \frac{\partial r(1)}{\partial \sigma(1)}}{(2 - \eta\lambda_1)^2} + \frac{\frac{\partial r(0)}{\partial \sigma(1)} - \frac{\partial r(1)}{\partial \sigma(1)}}{(2 - \eta\lambda_2)^2} \right) \quad (3.47)$$

Similarly, we can show that

$$\frac{\partial \zeta_\infty}{\partial \sigma(2)} = \frac{2\eta\sigma_v^2}{(1 - \eta S)^2} \left(\frac{\frac{\partial r(0)}{\partial \sigma(2)} + \frac{\partial r(1)}{\partial \sigma(2)}}{(2 - \eta\lambda_1)^2} + \frac{\frac{\partial r(0)}{\partial \sigma(2)} - \frac{\partial r(1)}{\partial \sigma(2)}}{(2 - \eta\lambda_2)^2} \right) \quad (3.48)$$

Next, by subtracting (3.48) from (3.47) to obtain

$$\begin{aligned}\frac{\partial \zeta_\infty}{\partial \sigma(1)} - \frac{\partial \zeta_\infty}{\partial \sigma(2)} &= \frac{2\eta\sigma_v^2}{(1 - \eta S)^2} \left[\left(\frac{\partial r(0)}{\partial \sigma(1)} - \frac{\partial r(0)}{\partial \sigma(2)} \right) \left(\frac{1}{(2 - \eta\lambda_1)^2} + \frac{1}{(2 - \eta\lambda_2)^2} \right) \right. \\ &\quad \left. + \left(\frac{\partial r(1)}{\partial \sigma(1)} - \frac{\partial r(1)}{\partial \sigma(2)} \right) \left(\frac{1}{(2 - \eta\lambda_1)^2} - \frac{1}{(2 - \eta\lambda_2)^2} \right) \right]\end{aligned}\quad (3.49)$$

To proceed further, we first evaluate the derivative terms $\frac{\partial r(0)}{\partial \sigma(1)}$ and $\frac{\partial r(0)}{\partial \sigma(2)}$ in Appendix which results in

$$\left(\frac{\partial r(0)}{\partial \sigma(1)} - \frac{\partial r(0)}{\partial \sigma(2)} \right) = -2|\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2|^{-\frac{1}{2}} \left(\frac{1}{(h^2 + 4\sigma(1))} - \frac{1}{(h^2 + 4\sigma(2))} \right) \quad (3.50)$$

which is always positive for $\sigma(1) > \sigma(2)$. Hence, we conclude that the first term inside the bracket in (3.49) is always positive for $\sigma(1) > \sigma(2)$.

Using similar approach, we can show that $\left(\frac{\partial r(1)}{\partial \sigma(1)} - \frac{\partial r(1)}{\partial \sigma(2)}\right) \geq 0$ for $\sigma(1) > \sigma(2)$. Moreover, we can easily see that $\left(\frac{1}{(2 - \eta\lambda_1)^2} - \frac{1}{(2 - \eta\lambda_2)^2}\right)$ as $\lambda_1 > \lambda_2$. Thus, we finally conclude that

$$\left(\sigma(1) - \sigma(2)\right) \left[\frac{\partial \zeta_\infty}{\partial \sigma(1)} - \frac{\partial \zeta_\infty}{\partial \sigma(2)}\right] \geq 0 \quad \text{for } \sigma(1) > \sigma(2) \quad (3.51)$$

which completes the proof that the steady-state EMSE of the WKLMS is Schur-convex function of σ . Thus, $\sigma_1 \succ \sigma_2$ will result in $\zeta_\infty(\sigma_1) \geq \zeta_\infty(\sigma_2)$.

3.3.2 Some Empirical Results

Transient Analysis

The transient behavior of the WKLMS algorithm is involved to express in closed form because it operates in an infinite dimensional feature space. In [2], the authors have derived closed form expressions for transient as well as steady state analysis of the KLMS algorithm. In [12], the authors have derived an expression for excess mean square error, but our empirical findings prove that some modifications in their results are required to justify the mean square error behavior of the WKLMS algorithm for fixed training set.

For fixed training size N , our empirical findings are that:

- The transient behavior gets affected by the natural modes (or eigenvalues) of the

Gram matrix (3.8), provided that the order of the weights in (3.14) is fixed. If the smaller modes are dominant and are uniformly distributed, WKLMS converges slowly, which is in accordance with the results derived for LMS in [9]. If the eigenvalues are exponentially distributed, the algorithm converges faster.

- With fixed order of weights, High eigenvalue spread of gram matrix guarantees faster convergence and vice versa.
- If the weights are selected in increasing order of their magnitude, the algorithm converges slower. Placing the same weights in decreasing order decreases the convergence time. In both cases, the eigenvalue distribution is same, which is exponential, and the eigenvalue spread also remains almost same. It also implies that faster convergence of the WKLMS algorithm can be achieved by keeping the eigenvalue spread almost constant, or in other words, restricted to some defined limits.
- Dynamically changing the weights during convergence changes the transient performance accordingly. The algorithm exhibits severe peak in MSE at the instant when weights are changed, and later exhibits the convergence behavior specific to the newly assigned weights.

Steady-State Analysis

For the same reasons stated in the transient analysis, we revert to empirical conclusions to learn the steady-state behavior of the WKLMS algorithm. We conclude the following from the empirical findings that for fixed training size N :

- The eigenvalue spread and distribution of Gram matrix (3.8) does not play a significant role in the steady state MSE of the WKLMS algorithm.
- The MSE is highly sensitive to the ordering of weights in (3.14). If the weights are placed in an increasing order, the resulted MSE by the WKLMS algorithm is minimum when compared to weights placed in any other order.
- If all weights are equal, then the MSE exhibits minimum value at a certain choice of weight. Decreasing or increasing the weight value further increases the MSE. We empirically found that the weight value at which MSE is minimum can be approximated as

$$w_{ii}^o \approx \sqrt{\mathbb{E}[v^2]}, \quad (3.52)$$

where v is the input data vector element, assumed to be a random variable with uniform probability distribution, w_{ii}^o is chosen to be positive, and $\mathbb{E}$ denotes the Expectation operator. Since the probability distribution of input vector elements is unknown, the expected value in (3.52) needs to be empirically calculated during the training of the algorithm.

- Smaller the weights, more oscillatory the steady state response of WKLMS becomes. If the weights are chosen close to zero, the algorithm exhibits severe oscillations throughout the convergence to minimum MSE.
- The weighting matrix in (3.14) poses no threat to the convergence of the WKLMS algorithm. Therefore, if the learning rate, or step size is chosen properly, the convergence of the algorithm might be slowed down, but is always

guaranteed with any choice of weights.

- Dynamically changing the weights during convergence changes the MSE. Regardless at which instant the weights are changed, WKLMS approaches the MSE specific to the most recently assigned weights.

Empirical Weight selection criteria

Equation (3.52) provides an approximation for selecting weight value which is assigned to all the weight variables in (3.14). We now propose an approximation for selecting unique optimum weights for which the WKLMS exhibits the fastest transient response with minimum MSE.

Consider an input data vector of dimension d . The corresponding weighting matrix (3.14) will have d number of weights. Assuming d is odd and the input vector elements are uniformly distributed, if v denotes the input vector element, we have found best convergence results if weights are estimated as

$$w_{ii}^o \approx \begin{cases} |\mathbb{E}[v]|/10, & \text{for } 0 < i < \frac{d}{2} \\ 1, & \text{for } i = \frac{d}{2} \\ \sqrt{\mathbb{E}[v^2]}, & \text{for } \frac{d}{2} < i \leq d \end{cases} \quad (3.53)$$

It should be noted that (3.53) is valid for scalar data input as it is operating on data points or, in case of vector data, on individual vector elements. The estimate gets better with an increase in the number of training input data sequence.

3.3.3 Simulation Results

In this section, the performance analysis of the WKLMS algorithm is investigated and compared with that of the linear LMS and standard KLMS results.

In simulation tasks, we investigate nonlinear channel equalization. For this task, we consider a non-linear channel whose output y_i is obtained using $y_i = x_i - 0.9x_i^2 + n_i$ where n_i is the zero mean additive white Gaussian noise with variance $\sigma_v^2 = 0.01$ and x_i is the linear output via $x_i = s_i - 0.5s_{i-1}$ where s_i is the input to the communication channel.

Task 1: Nonlinear Channel Equalization

In the first task, we consider nonlinear channel equalization. The equalizer length is set to 5, the step-sizes for the weight updates of LMS, KLMS, and WKLMS are 0.01, 0.2, and 0.2, respectively. We compare the MSE performance of the proposed WKLMS with that of the conventional linear LMS and KLMS algorithms. The results for the testing MSE are reported in Fig. 3.6. It can be depicted from the results that the performance of the proposed WKLMS is better than its counterpart in terms of both convergence speed and steady-state MSE. Fig. 3.6 shows that when weights are selected according to (3.53), WKLMS outperforms both LMS and KLMS both in the transient and steady state phase by converging faster and showing minimum MSE. Fig. 3.7 compares the performance of WKLMS with 2 sets of weights, first one chosen as given by (3.53) and the second set of weights are same but placed in decreasing order. Fig. 3.7 shows that both the magnitude and order of

weights play critical role during convergence. Placing the weights in increasing order increases the convergence time with relatively much lower MSE, while in decreasing order the algorithm converges slightly faster but with very high MSE

Fig. 3.8 shows that when selected weights are very small or large in magnitude, WKLMS performs very poorly and shows a high MSE compared to KLMS. When weights are very small, the algorithm shows high MSE and exhibits severe oscillations in the transient as well as in the steady state phase. In the case of large weights, the algorithm shows smooth convergence and steady state behavior, but still the MSE is very high compared to KLMS.

By reversing the order of weights, eigenvalue distribution of Gram matrix $\mathbf{W}_d$ and spread remains almost same. The sum of eigenvalues always remains constant, independent of weight selection, and is equal to input data training size N . When large weights are selected, the eigenvalue spread is decreased considerably. Similarly, selecting small weights increases the eigenvalue spread. These results are summarized in Table I. From Table I and Fig. 3.8, we conclude that in the case of small eigenvalue spread, the algorithm exhibits smooth convergence behavior and if it is larger, the convergence is oscillatory. But in both cases, the MSE is very high compared to WKLMS in Fig. 3.6.

Fig. 3.9 shows that if weights are dynamically changed during training iterations, the algorithm exhibits strong spikes at the instant when weights are changed, and later converges to the newly assigned weight specific MSE.

Task 2: Impact of Weighting Matrix (Majorization Study)

In this task, we investigate the impact of weighting matrix on the MSE performance of

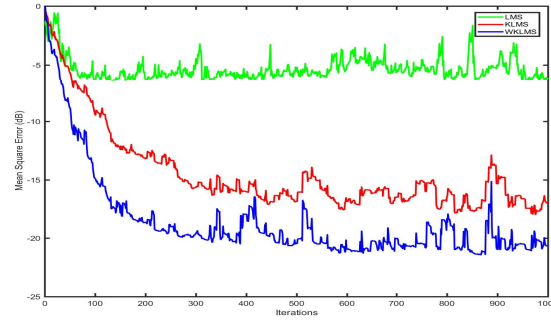


Figure 3.6: MSE performance comparison of LMS, KLMS and WKLMS. Selected weights for WKLMS are $[0.1, 0.1, 1, 2, 2]$.

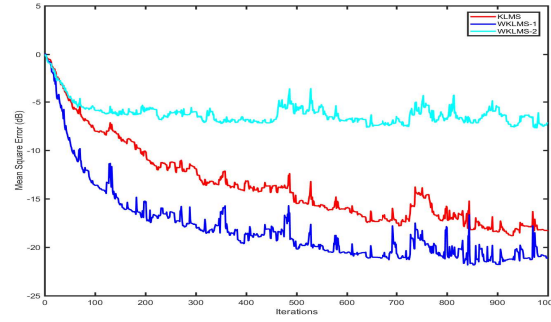


Figure 3.7: MSE performance comparison of WKLMS with different weight selection. Selected weights for WKLMS-1 are $[0.1, 0.1, 1, 2, 2]$ and for WKLMS-2 are $[2, 2, 1, 0.1, 0.1]$.

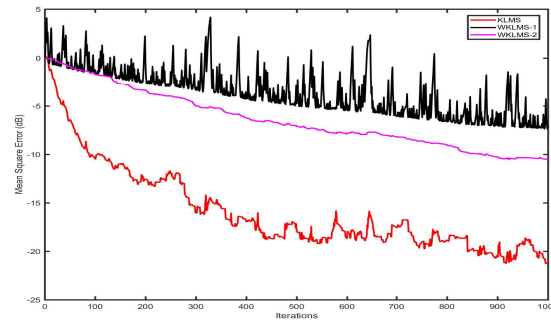


Figure 3.8: MSE performance comparison of WKLMS with very large and small weights. Weights in WKLMS-1 are $[0.01, 0.01, 0.01, 0.01]$ and in WKLMS-2 are $[16, 16, 16, 16]$.

Table 3.2: Eigenvalues and eigenvalue spread of Gram Matrix having size $N = 50$

blue!25 Weights	blue!25 λ_{min}	blue!25 λ_{max}	blue!25 ρ_w
[1, 1, 1, 1, 1]	0.002	13.3	6.6×10^3
[0.1, 0.1, 1, 2, 2]	0.00032	18.3	5.6×10^4
[2, 2, 1, 0.1, 0.1]	0.00047	16.9	3.5×10^4
[0.01, 0.01, 0.01, 0.01, 0.01]	2.4×10^{-10}	48.9	2×10^{11}
[16, 16, 16, 16, 16]	0.3	2.61	8.487

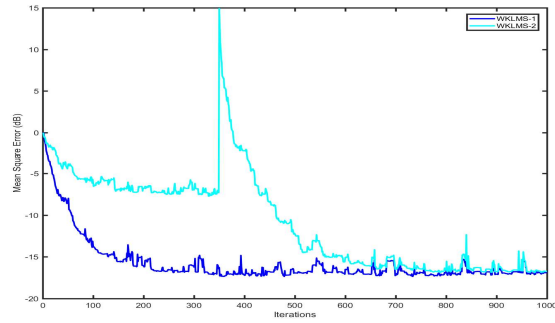


Figure 3.9: MSE behavior of WKLMS while dynamically changing the weights. WKLMS-1 has best weight assignment as [0.1, 0.1, 1, 2, 2] and WKLMS-2 has weights [2, 2, 1, 0.1, 0.1] from start till iteration 350. From 351 onwards, WKLMS-2 is assigned same weights as WKLMS-1.

the proposed WKLMS algorithm. More specifically, we want to validate the majorization and Schur-convexity properties of the WKLMS which we have proved analytically in Section V.

For this, we consider the problem of channel equalization with parameters η and σ_v^2 set to $\eta = .2$ and $\sigma_v^2 = .01$, respectively. To contrast the performance, we consider two weighting matrices Σ : one with ascending diagonal elements, i.e., $\sigma_1 = [0.1, 0.1, 1, 2, 2]$ (say WKLMS-1) and second with descending diagonal elements, i.e., $\sigma_2 = [2, 2, 1, 0.1, 0.1]$ (say WKLMS-2). Thus, we have $\sigma_2 \succ \sigma_1$. The MSE results are reported in Fig. 3.10. It can be easily seen that the MSE of the WKLMS-2 is greater than that of the WKLMS-1, that is, $\zeta_\infty(\sigma_2) > \zeta_\infty(\sigma_1)$ which validates our theoretical proof for majorization and Schur-convexity..

In order to further validate our majorization study related to MSE of the WKLMS algorithm, we compare the steady-state MSE of three weighting vectors $\sigma_1 = [4.8889, 0.1000, 0.0100, 0.0010, 0.0001]$, $\sigma_2 = [3.6000, 0.8900, 0.3000, 0.2000, 0.0100]$, and $\sigma_3 = [2.8000, 1.1000, 0.6000, 0.4500, 0.0500]$. It can be observed that these vector satisfies the majorization order as $\sigma_1 \succ \sigma_2 \succ \sigma_3$. We use the dictionary size of $M = 2$ and the step-size of the algorithm is set to 0.005 (i.e., $\eta = .005$). For each of the three vectors, we evaluate the eigenvalues of correlation matrix $\mathbf{R}_{\kappa\kappa}$ using the expressions in (3.25) which are then utilized to evaluate the steady-state EMSE given in (3.34). These results are reported in Table 3.8 which shows that the $\zeta_\infty(\sigma_3) > \zeta_\infty(\sigma_2) > \zeta_\infty(\sigma_1)$ which is in-line with the findings.

Weight vector σ	Eigenvalues of $\mathbf{R}_{\kappa\kappa}$	Steady-state EMSE
$\sigma_1 = [4.8889, 0.1000, 0.0100, 0.0010, 0.0001]$	$[0.0972, 0.2676]$	9.1329×10^{-6}
$\sigma_2 = [3.6000, 0.8900, 0.3000, 0.2000, 0.0100]$	$[0.0356, 0.0820]$	2.9415×10^{-6}
$\sigma_3 = [2.8000, 1.1000, 0.6000, 0.4500, 0.0500]$	$[0.0235, 0.0494]$	1.8231×10^{-6}

Table 3.3: Comparison of steady-state EMSE of the WKLMS algorithm with $\sigma_1 \succ \sigma_2 \succ \sigma_3$.

Validation of Analytical Results

In this task, we validate the theoretical analysis of the WKLMS algorithm. More specifically, we compare the MSE learning curves of the WKLMS algorithm obtained via simulations with the one calculated via derived expression. For this purpose, we use white input vectors of length 5. The dictionary size is also set to 5. The weighting matrix has diagonal weight vector $\sigma = [0.1, 0.1, 1, 2, 2]$. The MSE learning curves are compared for two different values of step-size, that is, 0.1 and .05. The simulated curves are obtained by averaging over 400 independent experiments, whereas the theoretical curves are obtained by plotting the MSE recursion derived in Eq. (3.32). A good match can be observed between the simulation and theory which validates our analysis.

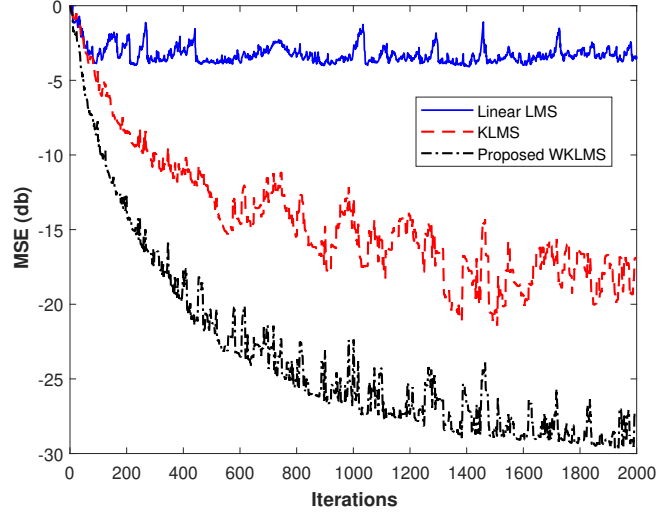


Figure 3.10: Testing MSE for the non-linear channel equalization.

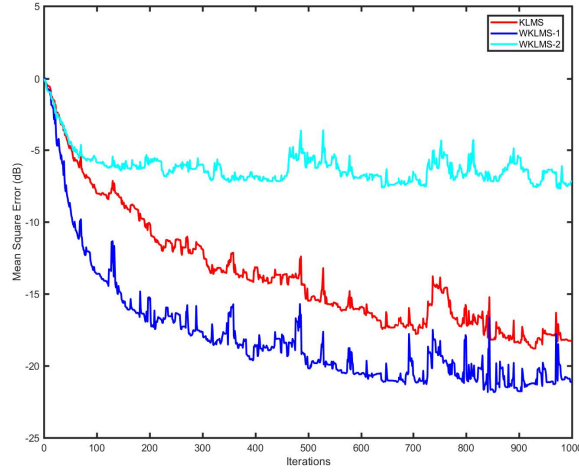


Figure 3.11: MSE performance comparison of WKLMS with different weight selections. Selected weights for WKLMS-1 are $[0.1, 0.1, 1, 2, 2]$ and for WKLMS-2 are $[2, 2, 1, 0.1, 0.1]$.

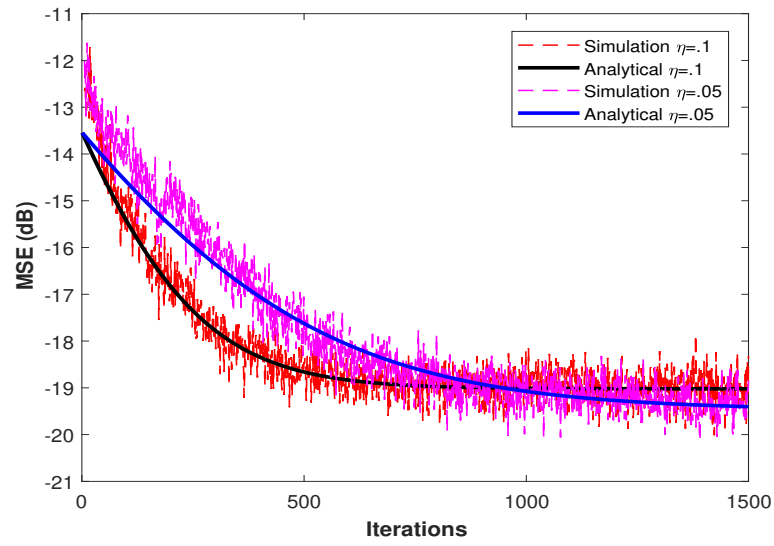


Figure 3.12: Comparison of analytical and simulation result for the MSE learning curves.

CHAPTER 4

CONCLUSION

In this work, we have first presented the construction of KLMS algorithm, followed by its different variants proposed in the literature. Each variant of KLMS is targeted towards increasing the performance or optimizing the time and space complexity of the algorithm. Most of the variations in KLMS presented in the literature have targeted its linearly growing structure, which in turn means growing memory requirement. In this regard, many techniques have been used, like approximate linear dependency, information theoretic criteria, and input space compression techniques like vector quantization, and multiple kernel learning.

Lastly, we have presented a novel KLMS algorithm based on Weighted Gaussian Kernel, labeled as WKLMS. We have presented its construction, Transient as well as steady state analysis, and experimental results to confirm the superiority of its performance over regular KLMS algorithm.

4.1 Future research

Although many variants of KLMS exists, however, they overcome the growing network size by compromising either the simplicity of the algorithm, or loss of information from input space (sparsification and compression techniques). Unlike LMS, no variant of KLMS has yet successfully overcome the general problem of growing network size by keeping the construction of the algorithm simple, and without compromising any information belonging to input space.

Moreover, selection of optimum kernel, along with its parameters is still an open, and challenging problem. Heuristics in the literature do help, but still do not guarantee to be always optimum for a specific problem. Therefore, the choice of Kernel, and whether to choose multiple kernels for a given problem, is still an open research problem. From Fig 3.9, we can observe that dynamically changing the weights has a tremendous impact on the algorithm. Noting that WKLMS, like KLMS creates a growing RBF neural network, where the weights are replaced by error coefficients scaled by the step size, it requires further investigation that why the error coefficients accumulated in the memory play literally no role in deciding the MSE, but rather, the weights in the weight matrix 3.14 play dominant role.

APPENDIX A

PROOFS

Evaluating the derivatives $\frac{\partial r(0)}{\partial \boldsymbol{\sigma}(1)}$ and $\frac{\partial r(0)}{\partial \boldsymbol{\sigma}(2)}$

We first evaluate the derivative $\frac{\partial r(0)}{\partial \boldsymbol{\sigma}(1)}$ by utilizing the expression for $r(0)$ given in (3.44).

Thus, we have

$$\frac{\partial r(0)}{\partial \boldsymbol{\sigma}(1)} = \frac{\partial}{\partial \boldsymbol{\sigma}(1)} |\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2|^{-\frac{1}{2}} = -\frac{1}{2} |\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2|^{-\frac{3}{2}} \frac{\partial}{\partial \boldsymbol{\sigma}(1)} |\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2| \quad (\text{A.1})$$

Next, using the following derivative solution for any arbitrary matrix $\mathbf{A}$ ¹

$$\frac{\partial |\mathbf{A}|}{\partial x} = |\mathbf{A}| \text{Tr} \left(\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial x} \right) \quad (\text{A.2})$$

we can evaluate $\frac{\partial r(0)}{\partial \boldsymbol{\sigma}(1)}$ as

$$\begin{aligned} \frac{\partial r(0)}{\partial \boldsymbol{\sigma}(1)} &= -\frac{1}{2} |\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2|^{-\frac{1}{2}} \text{Tr} \left((\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2)^{-1} \frac{\partial}{\partial \boldsymbol{\sigma}(1)} (\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2) \right) \\ &= -\frac{1}{h^2} |\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2|^{-\frac{1}{2}} \text{Tr} \left((\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2)^{-1} \mathbf{Q}_2' \right) \end{aligned} \quad (\text{A.3})$$

where

$$\mathbf{Q}_2' = \begin{bmatrix} \mathbf{1}\mathbf{1}_1 & -\mathbf{1}\mathbf{1}_1 \\ -\mathbf{1}\mathbf{1}_1 & \mathbf{1}\mathbf{1}_1 \end{bmatrix} \quad (\text{A.4})$$

where $\mathbf{1}\mathbf{1}_1$ is an $m \times m$ matrix with all zeros except 1 at position (1,1). In order to proceed further, we evaluate the term $(\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2)^{-1}$ which can be expressed as follows

$$(\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2)^{-1} = \begin{bmatrix} \mathbf{I}_m + \frac{2}{h^2}\mathbf{\Sigma} & -\frac{2}{h^2}\mathbf{\Sigma} \\ -\frac{2}{h^2}\mathbf{\Sigma} & \mathbf{I}_m + \frac{2}{h^2}\mathbf{\Sigma} \end{bmatrix}^{-1} \quad (\text{A.5})$$

Next, we employ the following block matrix inversion property

$$\begin{bmatrix} \mathbf{A} & \mathbf{U} \\ \mathbf{V} & \mathbf{C} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{U}(\mathbf{C} - \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{U}(\mathbf{C} - \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1} \\ -(\mathbf{C} - \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1} & (\mathbf{C} - \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1} \end{bmatrix}^{-1} \quad (\text{A.6})$$

to obtain the following result

$$(\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2)^{-1} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \quad (\text{A.7})$$

where

$$\begin{aligned} \text{diag}(\mathbf{A}) &= \left[\frac{h^4 + 4h^2\sigma(1) + 4\sigma^2(1)}{(h^2 + 4\sigma(1))(h^2 + 2\sigma(1))}, \frac{h^4 + 4h^2\sigma(2) + 4\sigma^2(2)}{(h^2 + 4\sigma(2))(h^2 + 2\sigma(2))}, \dots, \frac{h^4 + 4h^2\sigma(m) + 4\sigma^2(m)}{(h^2 + 4\sigma(m))(h^2 + 2\sigma(m))} \right] \\ \text{diag}(\mathbf{B}) &= \left[\frac{2\sigma(1)}{(h^2 + 4\sigma(1))}, \frac{2\sigma(2)}{(h^2 + 4\sigma(2))}, \dots, \frac{2\sigma(m)}{(h^2 + 4\sigma(m))} \right] \\ \text{and } \text{diag}(\mathbf{C}) &= \left[\frac{(h^2 + 2\sigma(1))}{(h^2 + 4\sigma(1))}, \frac{(h^2 + 2\sigma(2))}{(h^2 + 4\sigma(2))}, \dots, \frac{(h^2 + 2\sigma(m))}{(h^2 + 4\sigma(m))} \right] \end{aligned} \quad (\text{A.8})$$

Thus, the derivative $\frac{\partial r(0)}{\partial \sigma(1)}$ will result in

$$\begin{aligned}\frac{\partial r(0)}{\partial \sigma(1)} &= -\frac{1}{h^2} |\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2|^{-\frac{1}{2}} \left(\frac{h^4 + 2h^2\sigma(1)}{(h^2 + 4\sigma(1))(h^2 + 2\sigma(1))} + \frac{h^2}{(h^2 + 4\sigma(1))} \right) \\ &= -2|\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2|^{-\frac{1}{2}} \left(\frac{1}{(h^2 + 4\sigma(1))} \right)\end{aligned}\tag{A.9}$$

Similarly, the derivative $\frac{\partial r(0)}{\partial \sigma(2)}$ can be shown to be

$$\frac{\partial r(0)}{\partial \sigma(2)} = -2|\mathbf{I}_{2m} + 2\mathbf{Q}_2/h^2|^{-\frac{1}{2}} \left(\frac{1}{(h^2 + 4\sigma(2))} \right)\tag{A.10}$$

APPENDIX B

SUPPLEMENTARY MATERIAL

REFERENCES

- [1] W. Liu, P. Pokharel, and J. Principe, “The kernel least-mean-square algorithm,” *IEEE Transactions on Signal Processing*, vol. 56, pp. 543–554, 2008.
- [2] B. Chen, S. Zhao, P. Zhu, and J. C. Principe, “Mean square convergence analysis for kernel least mean square algorithm,” *Signal Processing*, vol. 92, pp. 2624–2632, 2012.
- [3] B. Chen, S. Zhao, P. Zhu, and J. C. Principe, “Quantized kernel least mean square algorithm,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, pp. 22–32, 2012.
- [4] P. P. Pokharel, W. Liu, and J. C. Principe, “Kernel least mean square algorithm with constrained growth,” *Signal Processing*, vol. 89, pp. 257–265, 2009.
- [5] W. Liu, I. Park, and J. C. Principe, “An Information Theoretic Approach of Designing Sparse Kernel Adaptive Filters,” *IEEE Transactions on Neural Networks*, vol. 20, pp. 1950–1961, 2009.
- [6] C. Richard, J. C. M. Bermudez, and P. Honeine, “Online prediction of time series data with kernels,” *IEEE Transactions on Signal Processing*, vol. 57, pp. 1058–

1066, 2009.

- [7] B. Widrow, “Adaptive Filter I: Fundamentals (TR 6764-6),” *Technical Report*, pp. 2624–2632, 1966.
- [8] B. Scholkopf and A. Smola, *Learning with Kernels, Support Vector Machines, Regularization, Optimization and Beyond*, 2002.
- [9] S. Haykin, *Adaptive Filter theory, 4th Ed.* Prentice Hall, 2002.
- [10] A. Syed, *Fundamentals of Adaptive Filtering.* New York: Wiley, 2003.
- [11] N. Aronszajn, “Theory of reproducing kernels,” *Transactions of the American Mathematical Society* 68 (1950), vol. 68, p. 337–404, 1950.
- [12] W. Liu, J. Principe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*, 2010.
- [13] A. Tikhonov and V. Arsenin, *Solution of Ill-Posed Problems.* Washington DC: Winston & Sons, 1977.
- [14] F. Girosi, M. Jones, and T. Poggio, “Regularization theory and neural networks architectures,” *Neural Computation*, vol. 7, pp. 219–269, 1995.
- [15] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.
- [16] S. An, W. Liu, and S. Venkatesh, “Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression,” *Pattern Recognition*, vol. 40, pp. 2154–2162, 2009.

- [17] G. C. Cawley and N. L. C. Talbot, “Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers,” *Pattern Recognition*, vol. 36, pp. 2585–2592, 2003.
- [18] J. Platt, “A resource-allocating network for function interpolation,” *Neural Computation*, vol. 3, pp. 213–225, 1991.
- [19] S. Peng, “Kernel least mean square based on conjugate gradient,” in *Acoustics Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017.
- [20] L. Glass and M. Mackey, *From Clocks to Chaos: The Rhythms of Life*. Princeton University Press, Princeton, NJ, 1988.
- [21] Y. Engel, S. Mannor, and R. Meir, “The kernel recursive least-squares algorithm,” *IEEE Transactions on Signal Processing*, vol. 52, pp. 2275–2285, 2004.
- [22] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley-Interscience,, 1991.
- [23] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.
- [24] W. Parreira, J. Bermudez, C. Richard, and J. Tournet, “Stochastic behavior analysis of the Gaussian kernel least-mean-square algorithm,” *IEEE Transactions on Signal Processing*, vol. 60, pp. 2208–2222, 2012.

- [25] K. A. Al-Hujaili, T. Y. Al-Naffouri, and M. Moinuddin, “The steady-state of the (Normalized) LMS is schur convex,” in *Acoustics Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on.* IEEE, 2016.
- [26] A. W. Marshall, I. Olkin, and B. C. Arnold, *Inequalities: Theory of Majorization and Its Applications.* Springer, Science & Business Media, 2010.

VITAE

- Name: Salman Rauf Iqbal
- Nationality: Pakistani
- Date of Birth: Jan 23, 1986
- Email: *eng.rauf86@gmail.com*
- Permanent Address: 594A, Block E-1, Postal Code: 54782, Lahore, Pakistan
- Research Interests: Kernel based learning, Causal Inference
- Hobbies: Running, Weight Training, Swimming, Cycling, Traveling