APPLICATIONS OF KERNEL ALGORITHMS FOR NONLINEAR CHANNEL EQUALIZATION AND MACKEY-GLASS TIME-SERIES PREDICTION

KHALID IBRAHIM

312-FET/MSEE/S13

Thesis Supervisor

DR. IJAZ MANSOOR QURESHI

DEPARTMENT OF ELECTRONIC ENGINEERING,

FACULTY OF ENGINEERING AND TECHNOLOGY

INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD

Nonlinear systems.

Adaptive filters.

Mercer Theorem.

Thesis entitled

APPLICATION OF KERNEL ALGORITHMS FOR ONLINEAR CHANNEL EQUALIZATION AND MACKEY-GLASS TIME-SERIES PREDICTION

Submitted to International Islamic University Islamabad in partial fulfillment of the requirements for the award of degree of

MS ELECTRONICS ENGINEERING

BY

KHALID IBRAHIM

312-FET/MSEE/S13

SESSION 2013-2015

DEPARTMENT OF ELECTRONICS ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD

Certificate of Approval

This thesis titled

APPLICATION OF KERNEL ALGORITHMS FOR NONLINEAR CHANNEL EQUALIZATION AND MACKEY-GLASS TIME-SERIES PREDICTION

By

Khalid Ibrahim [Reg #. 312-FET/MSEE/S13]

Has been approved for the International Islamic University, Islamabad

External Examiner: ____________________

Dr. Tanveer Ahmad Cheema
Department of Electronic Engineering, ISRA University, Islamabad

Internal Examiner: ____________________

Dr. Syed Zubair
Department of Electronic Engineering, IIU, Islamabad

Supervisor: ____________________

Dr. Ijaz Mansoor Qureshi
Department of Electrical Engineering, IIU, Islamabad

Chairman: ____________________
Dr. Muhammad Amir
Department of Electronic Engineering, IIU, Islamabad

Dean: ____________________
Dr. Aqdas Naveed Malik
Faculty of Engineering and Technology, IIU, Islamabad

DECLARATION

I, **Khalid Ibrahim s/o Muhammad Ibrahim**, Reg. No. **312-FET/MSEE/s13** student of MS electronics engineering in Session 2013-2015, hereby declare that the matter printed in the thesis titled "**APPLICATIONS OF KERNEL ALGORITHMS FOR NONLINEAR CHANNEL EQUALIZATION AND MACKEY-GLASS TIME-SERIES PREDICTION**" is my own work and has not been printed, published and submitted as research work, thesis or publication in any form in any University, Research Institution etc. in Pakistan or abroad.

Khalid Ibrahim

Dated: 09-03-2015

DEDICATION

I dedicate all my efforts to the **Holy Prophet Mohammad (S.A.W.),** who has been sent as a mercy to all mankind, my parents who have always supported me morally and financially, and have always prayed to Almighty ALLAH for my success, to my entire family members and to my respected teachers.

ACKNOWLEDGEMENTS

Praise to Almighty ALLAH, the most Benevolent, the Omnipotent and Merciful, the Creator of the Universe, who enabled me to accomplish this declaration. Blessing on MOHAMMAD (S.A.W), the seal of prophets and his pious progeny. I am grateful to International Islamic University, Islamabad for providing me a chance to enhance my skills and knowledge through this thesis and I wish to express my deepest gratitude to my worthy and honorable supervisor **Dr. Ijaz Mansoor Qureshi** for his kind supervision with keen interest and valuable guidance. His professional understanding and profound knowledge and kind attitude helped me in compiling this dissertation. His constructive criticism and directional guidance always motivated me to work hard.

I am especially grateful to my family members for their support. This thesis would not have been possible unless the cooperation and encouragement of my parents, brother and kind sisters.

And in the end I would like to pay my special thanks to my friends Ahmed Saleem, Sheraz Shafiq, Fahad Munir, Bilal Shoib, Naved Ishtiaq, Muhammad Waseem Khan, Shahid Ikram and Sir Jawad Ali Shah who helped me in preparing my thesis.

I ask ALLAH to help all those who have helped me and guided me from beginning to the end of my thesis.

ABSTRACT

This thesis describes the two applications Mackey-Glass time series prediction and nonlinear channel equalization using kernel algorithms which are based on reproducing kernel Hilbert spaces (RKHS). The mathematical theory of reproducing kernel Hilbert space provides the powerful basis for the nonlinear adaptive filters in high dimensional feature space such as KLMS and KAPA. After nonlinear transformation from input space to high dimensional feature space the kernel trick is exploited to express inner product with kernel evaluation. Due to their high dimensionality, kernel adaptive filters are universal approximators. Moreover kernel algorithms do not stuck in local minima. Simulation results are the evidence of the better performance of the said approach.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES:

List Of Algorithms:

Chapter 1

1 INTRODUCTION

Channel equalization is a practice used to improve link quality in unfriendly environment. In communication systems a transmitted signal when passed through communication channel is corrupted by the channel impairments. The basic idea of equalization is to reverse the distortion incurred by the communication channel. In the other way round, an inverse model of the channel is constructed so that channel effects could be nullified [1]. Historically, in digital communication systems, channels equalization has been studied using linear FIR models [2]. But in real world communication, channels are highly nonlinear due to channel impairments like multipath fading etc. These kind of channels are out of the scope of linear filters [3]. Therefore some nonlinear models were required to cope with the nonlinearities of the system.

Most of the recurrently encountered data in real world has chaos in nature and thus constitutes chaotic time series. Mackey-Glass time series is frequently used prominent chaotic time series. Mackey-Glass time series is chaotic time series to model the nonlinear systems. Mackey-Glass time series is generated by ordinary differential equations [4]. This series become a benchmark to test the nonlinear modeling capability of many algorithms.

Linear adaptive filters are widely used for linear channel equalization and linear system identification but LMS results are not good while dealing with nonlinear data. Kernel adaptive filters are adopted with Reproducing kernel Hilbert spaces. In kernel adaptive filters the inputs are transformed to high dimensional feature space with the help of some nonlinear mapping Φ. Then successive linear operation are applied on this transformed data.

1.1 Problem statement

Nonlinear channel equalization and time series forecasting are hot topics of the research in this modern era. Many physical channels exhibit nonlinear characteristics [5]. To equalize the channel corrupted by noise sources especially when the channel is nonlinear is a challenging task. Nonlinear Channel equalization and time series prediction of Mackey-Glass Time-series using kernel methods, the kernel least mean square(KLMS) and the kernel Affine projection Algorithm(KAPA), that can provide excellent results at minimal cost is a challenging assignment that has to be addressed seriously. This will promises optimal results that can guarantee efficiency and accuracy.

1.2 Proposed work

Kernel adaptive filters are adopted with reproducing kernel Hilbert spaces. The inputs are transformed to high dimensional feature space with the help of some nonlinear mapping Φ. Then successive linear operation are applied on this transformed data [6]. In this work successful application of kernel methods (KLMS and KAPA) are studied on nonlinear channel equalization and Mackey-Glass time series prediction.

Nonlinear modeling capability and universal approximations are some exclusive and attractive features of kernel based algorithms. These important features lead their importance in nonlinear problems [6]. In this work the Kernel algorithms are applied to nonlinear problems. The aim is to construct a nonlinear filter with the ability to map the nonlinearity of the system accurately. The iterative nature of adaptive filter is supposed to follow this iterative mechanism:

$$f_i = f_{i-1} + Gain\,(i)\,e(i)$$

where f_i is current value of updating function while f_{i-1} shows its values at previous iteration. Reproducing kernel Hilbert space, a high dimensional space, is used to achieve this objective. Moreover, this feature space has linear structure. In this work I will use kernel least mean square (KLMS) and kernel affine projection Algorithms (KAPA) to equalize the nonlinear channel and to predict Mackey-Glass Time series.

1.3 Thesis organization

In this introductory chapter, background of the problems is stated. Since both the problems are nonlinear so the need for algorithm with nonlinear modeling capability is stressed. After presenting brief introduction in chapter 1, Chapter 2 is dedicated to present basics of adaptive filters, its importance and explanation with block diagram is shown. Then nonlinearity is defined relative to the linearity condition. The two properties of a linear systems are discussed. Nonlinear systems are mathematically represented by nonlinear system of equation like nonlinear recurrent relationships and nonlinear differential equations. After that chapter 3explains reproducing kernel Hilbert spaces. Starting from inner product spaces, its definition and properties and then gradually reaching to RKHS. Inner product space has to satisfy three properties the symmetry, the bi-linearity and positive definite norm. An inner product space is called

Hilbert space when it is complete and a Hilbert space with kernel satisfying reproducing property is called reproducing kernel Hilbert space. Chapter 4 explains kernel algorithms in detail. This chapter is the heart of the thesis. Least mean square and affine projection algorithms are derived and then kernel least mean square and kernel affine projection are derived with mathematical details. All of the algorithms are summarized in algorithm tables.

Chapter 5 is there to elaborate in details the applications; the nonlinear channel equalization and Mackey-Glass time series. Channel equalization, its importance and types are discussed. Then nonlinear channel equalization with block diagram is given. The second application is Mackey-Glass time series. Its equation and the way how it is generated then brief introduction to its parameters are given. Chapter 6 is dedicated to the simulation and results of my work. Applications are simulated in Matlab r2012a. The results are shown in terms of mean square error. The effects of the change in step size parameters and different noise levels to get optimum solution is also tabulated, subsequently. Finally, chapter 7 concludes the thesis with numbered conclusion remarks. Scope of the work is written at the end for interested scholars to dig out the world of kernel filters. References are given at the end of thesis for the cited literature.

1.4 Review of the literature

In communication systems inverse modeling is an important part of the receiving systems. Channel equalization is hot topic in modern research. Many well-known algorithms are applied on this problem. In past, linear adaptive algorithms were used for equalizing channel impairments. LMS is the widely used algorithm used for channel equalization [2]. The reason of its wide spread use is its simplicity and least complexity. This is very simple mechanism but is confined to linear domain. It does not help when the relation between input and output is highly nonlinear. Nevertheless, it is famous that the LMS can only reduce the error estimate to some degree [7]. While Recursive Least square (RLS), On the other hand, converges considerably quicker than the LMS algorithm. The tradeoff is between numerical instability and computational complexity [8]. Number of affine projection based adaptive filter algorithms like standard version of affine projection algorithm and regularized affine projection algorithm were used. Using affine projection, a step size parameter is carefully chosen to get better convergence and MSE. During the adaptation process the optimal selection of the step size parameter, least steady state MSE and quick convergence can be obtained. These algorithms include Variable Step Size versions of affine projection and normalized LMS [9].

In solving filtering and prediction problems it is well known to apply linear adaptive algorithms like LMS, KLMS, APA and RLS etc. [8]. Lately, as an extension of the linear counterparts, kernel adaptive filters have been proposed that adaptively learn the non-linear systems. Kernel adaptive filters are derived by applying the kernel method to linear adaptive filters, and several algorithms were proposed, i.e. KLMS, KRLS and KAPA etc. [10].

Chapter 2

2 ADAPTIVE FILTERS

2.1 Linear adaptive filters

Adaptive Filter theory consist of three main parts: the linear filter, weight update mechanism and the cost function which is the mean square error, normally. Linear adaptive filters build a linear combination of input and weights of the system. Linear filters learn the adaptation sequentially. By adaptive we mean that the system is self-designing in which the adaptive filter depends on a recursive algorithm for its operation. Adaptive filters have incorporated weight update mechanism that allows these filters to adjust their free parameters repeatedly in response to the changes in the surrounding environment [7]. In all this process the learning process is online. Their learning mechanism is very simple. That is the reason adaptive filters are widely used in system identification, in adaptive noise cancellation, in communication receivers as adaptive equalizers, and in radar and sonar as adaptive beam forming etc. This is very simple mechanism but is confined to linear domain only. It does not help when the relation between desired response and input is highly nonlinear [6].

Typical adaptive filters comprises of two core portions; one is transversal filter and the other is adaptive weight control mechanism as shown in Figure 1. Transversal filter is linear FIR filter which convolves input with the weights of the filter. The other part is responsible for adaptive learning depending on the error calculations. This adaptive mechanism is the heart of algorithms.

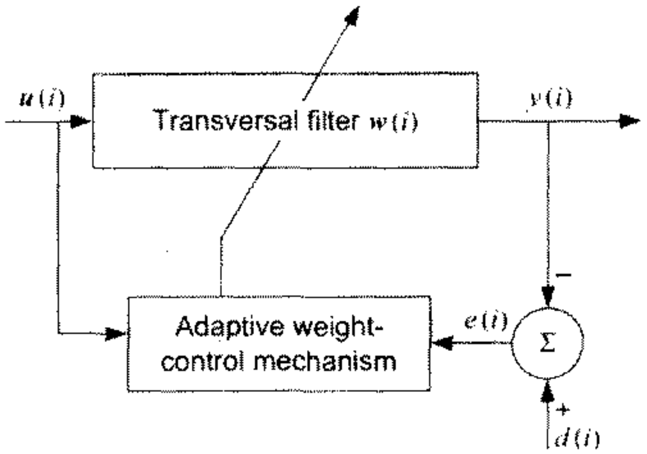

Figure 1: Linear Adaptive Filter; A block diagram

2.2 What is nonlinearity?

"It is the property of chaotic system which cannot be decomposed into the parts and reassembled into the same thing, and do not change in proportion of the change in input" [11]. The relation between input and output of the system is not linear rather input output relationship is curved. Or the input output relation which does not satisfy properties of a linear system i.e. superposition principle.

2.2.1 Properties of a linear system

The input output relationship $f(.)$ that satisfy following properties is called linear system.

1. Addition

$$f(x+y) = f(x) + f(y)$$

2. Scalar multiplication

$$f(\alpha x) = \alpha f(x)$$

These two conditions can be combined to form superposition principle.

The system which does not satisfy superposition property is simply nonlinear system [12].

2.3 Modeling nonlinear systems

Nonlinearity lead to randomness and unpredictable nature termed as chaos. Nonlinear events must be modeled with the system capable of nonlinear approximations [13]. Most of the real world systems exhibit nonlinear behavior. Nonlinear systems are characterized by nonlinear equations. Nonlinear equations include Nonlinear Algebraic equations, nonlinear recurrent relationships and nonlinear differential equations. Discrete nonlinear systems are represented by nonlinear recurrent relationships. This kind of relationship is used in sequences or train of samples related by nonlinear functions of previous terms. In nonlinear channel equalization and system identification problems the nonlinearity is modeled by nonlinear recurrent relationships [14]. The famous Mackey-Glass time series is generated by using nonlinear ordinary differential equations [15].

2.4 Nonlinear Adaptive Filters

Nonlinear filtering can be accomplished in so many ways. But here nonlinear filtering in Reproducing Kernel Hilbert Space is discussed. The theme is to transform the input data into high dimensional feature space then adaptive filtering is performed on this transformed data in feature space. This high dimensional space is associated to input space by some nonlinear function. Theory of Reproducing Kernel Hilbert Space (RKHS) is used to transform input into high dimensional feature space by some nonlinear transformation function. After transformation three basic parts of linear adaptive filtering i.e. linear filter, weight update mechanism and the cost function, can be exploited. While finding the output of the system, instead of taking inner product in feature space we find kernel evaluation using the vectors of input low dimensional space. This is called Kernel trick. Kernel trick enable us to compute output without finding inner product. Rather kernel evaluation is used in input space. Kernel evaluation in input space is equivalent to inner product in feature space.

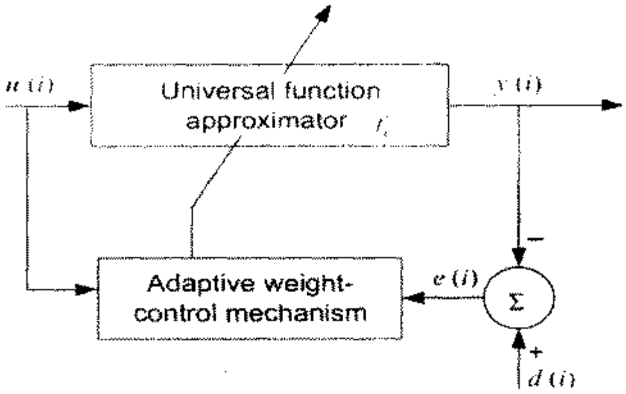
u (i)
Universal function approximator
f_i
y (i)
Adaptive weight-control mechanism
e (i)
Σ
−
+
d (i)

Figure 2: Nonlinear Filter structure

Chapter 3

3 REPRODUCING KERNEL HILBERT SPACES (RKHS)

Reproducing kernel Hilbert space is high dimensional feature space. This high dimensional space has linear structure to be exploited in modeling nonlinear filters in feature space. Using RKHS approach, nonlinear filters are developed in the linear space. Before going into details of Reproducing Hilbert Space it seems better to revise theory of Inner Product Space.

3.1 Inner product space

"In linear algebra, an *inner product space* is a vector space with the addition of an inner product. The inner product is a generalization of the 'dot product' regularly used in vector algebra" [16]. An inner product is also termed as normed vector space because a norm is associated with the inner product, naturally. [17]

3.1.1 Elementary properties of an inner product space

Let we have two vector **v** and **u**. To be in Inner product space they must satisfy the properties given bellow:

1. Symmetry

$$< \boldsymbol{u}, \boldsymbol{v} > = < \boldsymbol{v}, \boldsymbol{u} >$$

2. Bi-linearity

$$< a\,\boldsymbol{u} + \boldsymbol{v}, \boldsymbol{w} > = a < \boldsymbol{u}, \boldsymbol{w} > + < \boldsymbol{v}, \boldsymbol{w} >$$

3. Positive Definiteness:

$$< \boldsymbol{u}, \boldsymbol{u} > \geq 0 \quad \text{With equality if and only if } \mathbf{u} = \mathbf{0}$$

"A complete inner product space is called Hilbert space. Historically, inner product spaces are sometimes referred to as pre-Hilbert spaces" [18]. Completeness is defined by: "If every Cauchy sequence of vector convergence to a limit in H then inner product space H is complete" [6]. A special Hilbert space with a kernel is Reproducing Kernel Hilbert space (RKHS).

Let a vector space H is spanned by real valued kernel generated functions of **u** κ (u, ·). Assume h (·) and g (·) are two functions chosen from that space H that are correspondingly denoted by.

$$h = \sum_{i=1}^{l} a_i \, \kappa (c_i \, , .)$$

and

$$g = \sum_{j=1}^{m} b_j \, \kappa (\tilde{c}_j , .)$$

where the a_i , b_j belongs to **U** for all i and j and are called expansion coefficients. The inner product of the functions h and g

$$< h \, , \; g > = \sum_{i=1}^{l} \sum_{j=1}^{m} a_i \, \kappa(c_i \, , \; \tilde{c}_j) \, b_j$$

satisfies the properties of Symmetry, Scaling and distributive property and finally the squared norm or positive definiteness. For these evidences, the bilinear form $< h, g >$ is definitely an inner product.

3.2 Reproducing property

To be a Reproducing Kernel Hilbert space Kernel Hilbert space must hold reproducing property. Let $g(.)$ is real valued function of $\mathbf{u}$ i.e. $g(.) = \kappa(\boldsymbol{u},.)$. Taking inner product with another real valued function h, we get

$$< h,g > = < h,\kappa(\boldsymbol{u},\cdot) > = \sum_{i=1}^{l} a_i \; \kappa(c_i, \boldsymbol{u})$$

This is recognized as reproducing property. The kernel $\kappa(\boldsymbol{u},\boldsymbol{u}')$ is the function of input vectors while input vector belongs to $\mathbf{U}$. Subsequent two conditions to be satisfied for a kernel to be a reproducing kernel $\kappa(\boldsymbol{u},\boldsymbol{u}')$of vector space H.

1. For every input vector $\mathbf{u}$ belongs to U, kernel must be in feature space H.
2. Reproducing property must be satisfied by the kernel $\kappa(\boldsymbol{u},\boldsymbol{u}')$

Mercer Kernel also hold these conditions, therefore mercer kernel is also called reproducing kernel. The space spanned by mercer kernel is called reproducing kernel space H. If the reproducing kernel space H is complete, it is called reproducing kernel Hilbert space (RKHS).

3.3 Mercer Theorem

Mercer theorem states that a continuous, symmetric, non-negative definite reproducing kernel $\kappa(\boldsymbol{u},\boldsymbol{u}')$, the functions of the input vector $\mathbf{u}$ and $\mathbf{u}$', can be extended as follow

$$\kappa(\boldsymbol{u},\boldsymbol{u}') = \sum_{i=1}^{\infty} \lambda_i \ \varphi_i(\boldsymbol{u})\varphi_i(\boldsymbol{u}')$$

where λ_i and φ_i are Eigen values and corresponding Eigen functions correspondingly. A nonlinear mapping $\boldsymbol{\varphi}$ from input space U to high dimensional feature space can be shown as follow

$$\boldsymbol{\varphi} = \left[\sqrt{\lambda_1}\varphi_1(\boldsymbol{u})\sqrt{\lambda_2}\varphi_2(\boldsymbol{u})\sqrt{\lambda_3}\varphi_3(\boldsymbol{u})\right]$$

In vector form

$$\varphi(\boldsymbol{u})^T\varphi(\boldsymbol{u}) = \kappa(\boldsymbol{u},\boldsymbol{u}') \qquad 3.1$$

This equation is also called kernel trick and is the essence of kernel methods. Calculating inner product in high dimensional space is cumbersome task due to high dimensionality of the data. So, Instead of finding inner product in high dimensional space F equivalent kernel evaluation is used. Kernel evaluation enable us to do calculations in relatively low dimensional input space U. Hence computational complexity is reduced remarkably.

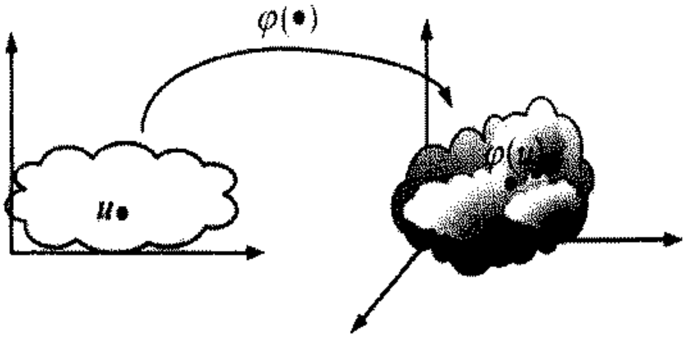

Figure 3: Nonlinear mapping: φ: U→F

The well-known commonly used continuous, symmetric, positive – definite kernels from input space U to high dimensional space F are

1. Gaussian Kernel

$$\kappa(\boldsymbol{u},\boldsymbol{u}') = exp(-a||\ \boldsymbol{u}-\boldsymbol{u}'\ ||)$$

And the polynomial kernel

$$\kappa(u,u') = (u^T u' + 1)^p$$

3.4 Kernel Adaptive Filters

Kernel adaptive filtering is a filtering technique used to solve nonlinear problems. Linear adaptive filters in rkhs is termed as kernel adaptive filters. Modeling nonlinear data is done with ease using kernel adaptive filters.

The summery of the scheme is given bellow:

1. *Transformation:* Using reproducing kernel input vector is transformed into high dimensional feature space. In such a way that kernel evaluation can be efficiently used to find inner product in high dimensional space.
2. *Proper linear operations:* Transformed data is subjected to some linear operations.

If the algorithm is formulated in terms of inner product (on in terms of kernel evaluation) computation in high dimensional feature space in not necessary. [6] Rather computations are carried out in input space using "kernel trick". Reproducing Kernel

Hilbert Space (RKHS) provides convexity, linearity and universal approximation capability as is the major requirement in modeling nonlinear filter.

Here is an example to show why transformation to high dimensional space is helpful in learning. Let we have two dimensional input $\mathbf{u} = [u_1 , u_2]^T$ and let the nonlinear mapping be

$$f(u_1, u_2) = a_1u_1 + a_2u_2 + a_3{u_1}^2 + a_4{u_2}^2$$

where a_i $(for\ i = 1,2,3,4)$ are some constants. $f(u_1, u_2)$ is nonlinear combination of u_1 and u_2. Clearly a linear combiner trying to model the function $f(u_1, u_2)$ by linear combinations of u_1 and u_2 can't model it accurately because of the square terms present in the function $f(u_1, u_2)$. On the other hand by using kernel method the transformation of the input is as follow

$$(u_1, u_2) \xrightarrow{\varphi} (x_1, x_2, x_3, x_4, x_5, x_6) = \left(1, {u_1}^2, \sqrt{2}u_1u_2, {u_2}^2, \sqrt{2}u_1, \sqrt{2}u_2\right)$$

Using nonlinear transformation function $\boldsymbol{\varphi}$ two dimensional input $\mathbf{u} = [u_1, u_2]$ is transformed to nonlinear data. Dimensions of this transformed input is clearly greater than that of untransformed input. Moreover the relation between input and transformed input is nonlinear (due to square terms). At this stage, it is easy to formulate linear combination of the high dimensional transformed input.

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = 0\ .\ x_1 + a_3x_2 + 0\ .x_3 + a_4x_4 + \frac{a_1x_5}{\sqrt{2}} + \frac{a_2x_6}{\sqrt{2}}$$

It is obvious that mapping into feature space has made the job easy. Voltera and Weiner series can be seen as the special cases of kernel methodology [19]. The complexity of these series is no more dependent on the order of nonlinearity and the input dimensionality when Voltera and Weiner series is formulated as linear regression problem in Reproducing Kernel Hilbert Space.

Based on these evidences the goal is modeling the linear adaptive filters in Reproducing Kernel Hilbert Space for optimizing in least square sense. As classic adaptive filters are formulated in terms of inner products we get nonlinear adaptive filters in feature space in such a way that kernel evaluation in input space is equivalent to inner product in feature space. Hence achieving universal approximation and avoiding algorithm to stick in local minima.

Linear adaptive filters are special case of kernel adaptive filters when these filters are expressed in feature space. So kernel adaptive filters can be termed as generalization of the linear adaptive filters. The weights and hence memory of kernel adaptive filters shows growing structure. Kernel adaptive filters form growing radial basis function network. By this way they learn the network topology and hence adaptively adjust its free parameters. Learning is good combination of previous data and error correction. Memory based learning causes an increase in computation time. To avoid this there is need to choose some informative data samples instead of all training data. Dealing with large and redundant data set kernel algorithms reduce training time and result in a relatively squeezed network with equivalent accuracy. Novelty Criterion is widely used procedure used to select informative data samples among large data set. [6] In kernel adaptive filters the linear structure of underlying reproducing Hilbert Space, where the algorithm exists, is exploited.

3.5 Bottleneck

There are few limitations in kernel adaptive filters. First, regularization is required. Selection of kernel function is another big issue to deal with carefully. The recursive mechanism in kernel adaptive filters has embedded structure with growing memory that is why kernel operation is memory intensive operation. So the need of reducing the growing network size of kernel adaptive filters is handled such that performance of the filter is unchanged.

Chapter 4

4 KERNEL ALGORITHMS

In solving filtering and prediction problems it is well known to apply linear adaptive algorithms like LMS, KLMS, APA and RLS etc. [8]. Lately, as an extension of the linear counterparts, kernel adaptive filters have been proposed that adaptively learn the non-linear systems. Kernel adaptive filters are derived by applying the kernel method to linear adaptive filters, and several algorithms were proposed, i.e. KLMS, KRLS and KAPA etc. [10]. This chapter is dedicated to the study of two Kernel Algorithms, the Kernel Least Mean Square (KLMS) Algorithm and Kernel Affine Projection Algorithm (KAPA). The simplest algorithm among the family of kernel algorithms is Kernel Least Mean Square (KLMS) Algorithm. Following the same methodology explained in chapter 2 (Reproducing Kernel Hilbert Spaces) the least mean square (LMS) algorithm is transformed directly into high dimensional feature space (RKHS). Over-all procedure for linear LMS is followed to derive kernel LMS. Inner product is found by kernel evaluation using reproducing property of RKHS. Calculating inner product in feature space by kernel evaluation of the input space is indeed the key benefit of RKHS approach.

4.1 Least Mean Square Algorithm (LMS)

In linear LMS linear FIR model is assumed for filtering. That is, the relation between input $\boldsymbol{u}(i)$ and desired response $d(i)$ is linear. With the assumption that input vector $\boldsymbol{u}(i)$ and desired response $d(i)$ are known LMS follow these steps.

To initialize the algorithm, if prior knowledge of tap weight vector is known use it. Otherwise assume it to be zero i.e.

$$\boldsymbol{w}(0) = 0$$

$\boldsymbol{w}(i)$ is Estimate of the tap weight at iteration i is to be computed. For $i = 0, 1,2, 3.....$

Compute

$$\boldsymbol{e}(i) = \boldsymbol{d}(i) - \boldsymbol{w}(i-1)^T \boldsymbol{u}(i)$$

$$\boldsymbol{w}(i) = \boldsymbol{w}(i-1) + \mu\, \boldsymbol{e}(i)\, \boldsymbol{u}(i)$$

where $\boldsymbol{e}(i)$ is called error, μ is step size parameter, chosen precisely, and $\boldsymbol{w}(i)$ is estimate of weight update vector at i time. To estimate the tap weights the algorithm is optimized in least square sense. The cost function is mean of the squared error i.e.

$$\mathrm{J}(\mathrm{w}) = \sum_{i=1}^{N} e(i)^2$$

$$= \sum_{i=1}^{N} (d(i) - w^T u(i))^2$$

$$= \sum_{i=1}^{N} \left(d(i)^2 - w^T u(i)\, u(i)^T \mathrm{w} - 2\, d(i)\, w^T u(i)\right)$$

Standard LMS can be derived by taking gradient of the cost function $J(w)$ with respect to tap weight vector **w**.

$$\nabla \mathrm{w}\, \mathrm{J} = \sum_{i=1}^{N} \left(0 + 2\, \boldsymbol{w}^T \boldsymbol{u}(i)\, u(i)^T - 2\, d(i)\, \boldsymbol{u}(i)\right)$$

$$= -2\sum_{i=1}^{N}\boldsymbol{u}(i)\,[d(i) - \boldsymbol{w}^T\boldsymbol{u}(i)] \qquad (as\, e(i) = d(i) - \boldsymbol{w}^T\boldsymbol{u}(i))$$

$$= -2\sum_{i=1}^{N}\boldsymbol{u}(i)\,e(i)$$

at time i

$$= -2\boldsymbol{u}(i)\,e(i)$$

Now from method of steepest descent

$$\mathbf{w}(i) = \mathbf{w}(i-1) - \frac{1}{2}\mu\,\nabla_{\mathrm{w}} J$$

so by putting the value of cost function gradient, we get

$$\mathbf{w}(i) = \mathbf{w}(i-1) - \frac{1}{2}\mu\,(-2\;\boldsymbol{u}(i)\,e(i))$$

Hence, we obtain LMS algorithm as bellow

$$\mathbf{w}(i) = \mathbf{w}(i-1) + \mu\,\mathbf{e}(i)\,\mathbf{u}(i)$$

Summary of LMS algorithm is given bellow. Weight vector is initialized as zero, usually.

Algorithm 1: Least Mean Square Algorithm

```
Initialization:
w(0) = 0 , choose μ
Computation:
While { u (i) , d(i) } are available do
e(i) = d(i) - $\mathbf{w}^T(i-1)\,\mathbf{u}(i)$
w( i ) = w ( i -1 ) + μ e(i) u( i )
```

The simplicity of the LMS algorithm is evident from Algorithm 1 [6].

4.2 Kernel Least Mean Square Algorithm (KLMS)

It is well-known that linear filtering model is used is LMS. If the mapping between input **u** and desired response d is highly nonlinear then this kind of situation is out of the scope of standard LMS i.e. reduced performance is observed for LMS. To cope with the nonlinearity of the input data another useful algorithm of the same kind is formulated called kernel LMS. Kernel LMS is based on Reproducing Kernel Hilbert Space (RKHS) approach. This is in turn LMS in RKHS. This algorithm is clever enough to map the nonlinearity, efficiently. To start with, the input **u**(i) is transformed into high dimensional feature space F as $\varphi(u(i))$ and $\mathbf{w}(i)$ to $\boldsymbol{\omega}(i)$, the weight vector in feature space F. Now following the similar stochastic gradient procedure as followed for LMS the kernel LMS can also be derived easily.

For simplicity $\boldsymbol{\varphi}(u(i))$ will be treated as$\boldsymbol{\varphi}(i)$.

Initially $\omega(0) = 0$

and $e(i) = d(i) - \boldsymbol{\omega}^T(i-1)\,\boldsymbol{\varphi}\,(i)$

weigh update mechanism: $\boldsymbol{\omega}(i) = \boldsymbol{\omega}\,(i-1) + \mu\, e(i)\,\boldsymbol{\varphi}\,(i)$

High order of similarity can be observed in the two algorithms. $\boldsymbol{\omega}^T\ \boldsymbol{\varphi}\,(i)$ is much better model compared with $\boldsymbol{wT}\ \boldsymbol{u}\,(i)$ due to high dimensionality of former. Yet, $\boldsymbol{\varphi}$ is not known directly so carrying out the computation in another way.

$$\boldsymbol{\omega}(i) = \boldsymbol{\omega}(i-1) + \mu\, \boldsymbol{e}(i)\,\boldsymbol{\varphi}(i)$$

Repeatedly putting the values of weight vector, we get

$$\boldsymbol{\omega}(i) = \boldsymbol{\omega}(i-1) + \mu\, \boldsymbol{e}(i)\,\boldsymbol{\varphi}(i)$$

$$= [\boldsymbol{\omega}(i-2) + \mu\, \boldsymbol{e}(i-1)\,\boldsymbol{\varphi}(i-1)] + \mu\, \boldsymbol{e}(i)\,\boldsymbol{\varphi}(i)$$

$= \omega(i-2) + \mu\,[e(i-1)\,\boldsymbol{\varphi}(i-1) + e(i)\,\boldsymbol{\varphi}(i)]$

$= \omega(i-3) + \mu\, e(i-2)\,\boldsymbol{\varphi}(i-2) + \mu\,[e(i-1)\,\boldsymbol{\varphi}(i-1) + e(i)\,\boldsymbol{\varphi}(i)]$

$= \omega (i -3) + \mu [e(i-2) \boldsymbol{\varphi} (i-2) + e(i -1) \boldsymbol{\varphi} (i -1) + e(i) \boldsymbol{\varphi} (i)]$

. . .

$= \omega(0) + \mu \sum_{j=1}^{i} e(j)\boldsymbol{\varphi}(j)$

$\omega(i) = \mu \sum_{j=1}^{i} e(j)\boldsymbol{\varphi}(j)$ ($\omega(0)$ is assumed to be zero; initial condition)

This alternative approach lead to the conclusion that transformed $\boldsymbol{\omega}(i)$ can be found by present and previous i-step transformed inputs weighted multiplied by error and scaled by step size parameter μ. Now, for new input **u'** the output of the system is found by taking inner product of the transformed version if this new input **u**′ and the transformed weight vector.

$$\boldsymbol{\omega}(i)^T \boldsymbol{\varphi}(\mathbf{u}') = \left[\mu \sum_{j=1}^{i} e(j)\boldsymbol{\varphi}(j)^T \right] \boldsymbol{\varphi}(\mathbf{u}')$$

$$\boldsymbol{\omega}(i)^T \boldsymbol{\varphi}(\mathbf{u}') = \mu \sum_{j=1}^{i} e(j) \; [\boldsymbol{\varphi}(j)^T \boldsymbol{\varphi}(\mathbf{u}')]$$

Inner product of transformed input and the transformed weight serve as the output of the system. Now by using kernel trick the output of the system (or equivalently the inner product) can easily be computed in the input space.

$$\boldsymbol{\varphi}(\boldsymbol{u})^T \boldsymbol{\varphi}(\boldsymbol{u}') = \kappa(\boldsymbol{u}, \boldsymbol{u}')$$

Using this equation, output of the system can be found as

$$\boldsymbol{\omega}(i)^T \boldsymbol{\varphi}(\mathbf{u}') = \mu \sum_{j=1}^{i} e(j) \; \kappa(\boldsymbol{u}, \boldsymbol{u}')$$

It is surprising to see that output of the filter is independent of the weight rather output is sum of the previous error multiplied with kernel evaluation on previously received data. This is indeed LMS in RKHS. Output is calculated by kernel evaluation. This new algorithm is called kernel least mean square (KLMS) algorithm. Generalization of the said algorithm is as follow:

If f_i is nonlinear mapping between input and output it time i we will get the following learning rule:

$$f_{i-1} = \mu \sum_{j=1}^{i-1} e(i)\, \kappa(\boldsymbol{u}(j),.)$$

Output: $$f_{i-1}(\boldsymbol{u}(i)) = \mu \sum_{j=1}^{i-1} e(i)\, \kappa(\boldsymbol{u}(j), \boldsymbol{u}(i))$$

Error: $$e(i) = d(i) - f_{i-1}(\boldsymbol{u}(i))$$

Weight update: $$f_i = f_{i-1} + \mu e(i) \kappa(\boldsymbol{u}(j),.)$$

In kernel least mean square algorithm new kernel unit is assigned to the upcoming training data. The input $\mathbf{u}(i)$ is considered as the center and $\boldsymbol{a}_i = \mu\, e(i)$ as expansion coefficients. During training period these centers and expansion coefficients are stored in the memory called dictionary. That is the reason kernel operation is memory intensive operation. Kernel least mean square algorithm is summarized in Algorithm 2 [6].

Algorithm 2: Kernel least mean square algorithm

```
Initialization:
Chose step size parameter $\mu$ and kernel type $\kappa$
$a_i(1) = \mu\, d(1)$ , $C(1) = \{\mathbf{u}(1)\}$ , $f_1 = \boldsymbol{a}_i(1)\, \kappa(\boldsymbol{u}(1), \ .)$
Computation:
While { u(i) , d(i) } available do
    % compute the output
    $f_{i-1}(\mathbf{u}(i)) = \mu \sum_{j=1}^{i-1} e(j)\ \ \kappa(\boldsymbol{u}(j),\ \mathbf{u}(i))$
    % compute the error
    $e(i) = d(i) - f_{i-1}(\mathbf{u}(i))$
    % store the new center
    $C(i) = \{C(i-1) , \mathbf{u}(i)\}$
    % compute and store the coefficients
    $a_i = \mu\, e(i)$
end while
```

Practically, in feature space, the access to transformed weights and the transformed input is not available directly. They are known implicitly, therefore updating process is done through expansion coefficients. The relation between weights $\boldsymbol{\omega}$ and expansion coefficients a_j is given bellow:

$$\boldsymbol{\omega}(i) = \sum_{j=1}^{i} \boldsymbol{a}_j(i) \boldsymbol{\varphi}(j)$$

4.3 Difficulties

There are few things to be specified yet. First is kernel selection, that is, it is a serious concern to choose suitable kernel capable of modeling the problem satisfactorily. The other is to choose suitable step size parameter; μ and the last is to deal with the increasing network size. The network size of the kernel algorithm keep on increasing with iterations. Therefore kernel methods are memory exhaustive methods. So suitable measures to be taken to reduce the network size while maintaining good performance. It is recognized that the Gaussian kernel has universal approximation capability in RKHS. If any specific kernel is not given in the problem then Gaussian kernel is chosen by default [6]. The remarkable features of Gaussian kernel is that it is stable numerically and do not stuck in local minima. For the said reasons Gaussian kernel give very good results. Gaussian kernel is given by the following equation

$$\kappa(u, u') = \exp(-a \; ||u - u'||)$$

and the polynomial kernel

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = (\boldsymbol{u}^T \boldsymbol{u}' + 1)^p$$

where in Gaussian kernel a is the kernel parameter. The kernel parameter is also termed as smoothing parameter or kernel size. While p is order of the polynomial function in polynomial kernel. To cope with the increasing network size of kernel function a method called 'Novelty Criterion' is proposed.

4.4 Affine Projection Algorithm (APA)

One of the most capable algorithms for filtering problems is the affine projection (AP) algorithm. Numerous effective approximations and implementations of AP algorithm are used in a diverse applications. Applying kernel trick to affine projection algorithm to derive an algorithm working best in nonlinear domain called kernel affine projection algorithm (KAPA). APA inherits the simplicity of LMS while decreasing gradient noise thus increasing LMS performance.

A least square regression data model can be constructed as

$$e(i) = d(i) - \boldsymbol{w}^T \boldsymbol{u}(i)$$

where $d(i)$ and $e(i)$ are desired response and error respectively, while $\boldsymbol{u}(i)$ is L x 1 data vector with covariance matrix $\boldsymbol{R}_u = E\ [\boldsymbol{uu}T]$, a positive definite matrix. $\boldsymbol{R}_{du} = E\ [d\ \boldsymbol{u}]$ is cross-covariance vector of $\boldsymbol{u}$ and d. The error is minimized in least square sense, which will give

$$min\, j(\boldsymbol{w}) = E\ [d - \boldsymbol{w}^T\ \boldsymbol{u}]$$

The optimum solution (Weiner solution) is given by $w_o = \mathbf{R}^{-1}\boldsymbol{r}_{du}$.

To approximate $\boldsymbol{w}_o$ several methods are adopted. Examples are

1. Gradient Descent method

Estimation of **w** is found iteratively using the following weight update equation

$$\boldsymbol{w}(i) = w(i-1) + \mu[\boldsymbol{r}_{du} - \boldsymbol{R}_u \boldsymbol{w}(i-1)] \qquad 4.1$$

while $\boldsymbol{w}(0)$ is some initial guess, assumed zero normally.

2. Newton's recursion:

This method is used normally to avoid slowness or to increase convergence speed. Weight update of Newton recursion is

$$\boldsymbol{w}(i) = \boldsymbol{w}(i-1) + \mu\,(\boldsymbol{R_u} + \varepsilon\,\boldsymbol{I})^{-1}\,[\boldsymbol{r_{du}} - \boldsymbol{R_u}\,\boldsymbol{w}(i-1)] \qquad 4.2$$

while $\boldsymbol{w}(0)$ is some initial guess, assumed zero normally. Division by zero is avoided by taking a small positive number, the smoothing factor ε. And μ is step size parameter to be chosen initially.

3. Stochastic gradient algorithm

Stochastic gradient algorithm replace $\boldsymbol{R_u}$ and $\boldsymbol{r_{du}}$ by local data approximation. LMS and Affine projection algorithms are two famous members of stochastic gradient algorithm family. There are many methods to get these kind of approximation. It is assumed that desired response $\{d(1), d(2), d(3), \ldots.\}$ and input vector **u** $\{\,\boldsymbol{u}\,(1), \boldsymbol{u}\,(2), \boldsymbol{u}\,(3), \ldots\}$ are known.

To approximate covariance matrix $\boldsymbol{R_u}$ and cross-covariance vector $\boldsymbol{r_{du}}$ LMS uses instantaneous values of $\widehat{\boldsymbol{R_u}} = \boldsymbol{u}(i)\boldsymbol{u}^T(i)$ and $\widehat{\boldsymbol{r_{du}}} = d(i)\boldsymbol{u}(i)$. Putting these values in steepest descent (equation 4.1) and Newton recursion algorithms (equation 4.2), we get

$$\boldsymbol{w}(i) = \boldsymbol{w}(\,i-1) + \mu\,\boldsymbol{u}(i)\,[d(i) - \boldsymbol{u}(i)^T\,\boldsymbol{w}(i-1)]$$

and

$$\boldsymbol{w}(i) = \boldsymbol{w}(\,i-1) + \mu\,\boldsymbol{u}(i)\,[\boldsymbol{u}(i)\boldsymbol{T}\,\boldsymbol{u}(i) + \varepsilon\,\boldsymbol{I}]^{-1}[d(i) - \boldsymbol{u}(i)^T\boldsymbol{w}(i-1)]$$

On the other hand, affine projection algorithm uses superior approximations. $\boldsymbol{R_u}$ and $\boldsymbol{r}_{\text{du}}$ are approximated by K most new inputs and observations.

Let $\quad U(i) = [\boldsymbol{u}(i-K+1), \ldots., \boldsymbol{u}(i)\,]_{L\,x\,K}$

and

$$d(i) = [d(i-K+1), \ldots, d(i)]$$

Then

$$\widehat{R_u} = \frac{1}{K} \boldsymbol{U}(i)\boldsymbol{U}^T(i) \qquad 4.3$$

and

$$\widehat{r_{du}} = \frac{1}{K} \boldsymbol{U}(i)\boldsymbol{d}(i) \qquad 4.4$$

Substituting equation 4.3 and equation 4.4 into weight update of gradient descent and Newton recursion, we obtain the following results called affine projection algorithm.

$$\boldsymbol{w}(i) = \boldsymbol{w}(i-1) + \mu\, \boldsymbol{U}(i)\, [\boldsymbol{d}(i) - \boldsymbol{U}(i)^T \boldsymbol{w}(i-1)]$$

and

$$\boldsymbol{w}(i) = \boldsymbol{w}(i-1) + \mu\, \boldsymbol{U}(i)\, [\boldsymbol{U}(i)^T\, \boldsymbol{U}(i) + \varepsilon\, \boldsymbol{I}]^{-1} [\boldsymbol{d}(i) - \boldsymbol{U}(i)^T \boldsymbol{w}(i-1)]$$

where ***d*** is desired vector and ***U*** is input data matrix.

The last two equations are called APA-1 and APA-2 respectively.

4.5 Kernel Affine projection Algorithm (KAPA)

Adopting the similar approach as followed to derive kernel LMS, the kernel affine projection algorithm is derived next. Transformed input vector is represented by $\boldsymbol{\varphi}(\mathbf{u}(\mathrm{i}))$. For the sack of simplicity $\boldsymbol{\varphi}(\mathbf{u}(\mathrm{i}))$ will be used as $\boldsymbol{\varphi}(\mathrm{i})$, further. It is assumed that desired vector $\{d(1), d(2), d(3), \dots\}$ and input data vector $\{\boldsymbol{\varphi}(1), \boldsymbol{\varphi}(2), \boldsymbol{\varphi}(3), \dots\}$ are known.

The error is minimized in least square sense, which will give

$$min\, J(\boldsymbol{\omega}) = E\,[d - \boldsymbol{\omega}^T\, \varphi(\boldsymbol{u})]$$

To estimate weight vector through stochastic gradient descent method (equation 4.1), we have

$$\boldsymbol{\omega}(i) \;=\; \boldsymbol{\omega}(\,i-1) \;+\; \mu\,\boldsymbol{\Phi}(i)\,[\boldsymbol{d}(i) - \boldsymbol{\Phi}(i)^T\,\boldsymbol{\omega}(i-1)] \qquad 4.5)$$

and by stochastic Newton method (equation 4.2), we have

$$\boldsymbol{\omega}(i) \;=\; \boldsymbol{\omega}(\,i-1) \;+\; \mu\,\boldsymbol{\Phi}(i)\,[\boldsymbol{\Phi}(i)^T\,\boldsymbol{\Phi}(i) \;+\; \varepsilon\,\boldsymbol{I}]^{-1}[\boldsymbol{d}(i) - \boldsymbol{\Phi}(i)^T\,\boldsymbol{\omega}(i-1)] \qquad 4.6)$$

where $\boldsymbol{\Phi}(i) = [\boldsymbol{\varphi}(i-K+1), \ldots, \boldsymbol{\varphi}(i)]$. The last two equations are the kernel versions of affine projection and known as KAPA-1 and KAPA-2 respectively.

4.5.1 KAPA-1 (Simple KAPA)

The simplest algorithm among the family of kernel affine algorithms is KAPA-1 and therefore it is celled simple-KAPA. Weight vector in high dimensional feature space is not known explicitly. The same procedure is repeated here to derive weight vector in terms of linear combination of transformed input vector and expansion coefficients.

$\omega(0) \;=\; 0$

$\omega(1) = \;\mu\, d(1)\, \varphi(1) \;=\; \boldsymbol{a}1\,(1)\,\varphi(1)$ $\qquad\qquad$ [As $\mathbf{a}_1 = \mu$ d (1)]

$\omega(2) \;=\; \boldsymbol{a}1\,(2)\,\varphi(1) \;+\; \boldsymbol{a}2\,(2)\,\varphi(2)$

$\omega(3) \;=\; \boldsymbol{a}1\,(3)\,\varphi(1) \;+\; \boldsymbol{a}2\,(3)\,\varphi(2) \;+\; \boldsymbol{a}3\,(3)\,\varphi(3)$

...

$$\begin{aligned} \boldsymbol{\omega}\,(i-1) &= \sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\,\boldsymbol{\varphi}(j) \\ \boldsymbol{\omega}\,(i) &= \sum_{j=1}^{i} \boldsymbol{a}_j\,(i)\,\boldsymbol{\varphi}(j) \end{aligned} \qquad 4.7)$$

Practically the access to the transformed weights is not feasible that is why updating of weight vector is indirectly accomplished by expansion coefficients. Expansion coefficients are defined by

$$a_k(i) = \begin{cases} \mu\left(d(i) - \sum_{j=1}^{i-1} a_j(i-1)\,\kappa_{i,j}\right), & k = i \\ a_k(i-1) + \mu\left(d(k) - \sum_{j=1}^{i-1} a_j(i-1)\,\kappa_{k,j}\right), & i-K+1 \le k \le i-1 \\ a_k(i-1), & 1 \le k < i-K+1 \end{cases} \qquad 4.8)$$

Filter output is given by inner product of transformed input data matrix and transformed weight vector

$$\boldsymbol{\Phi}^T(i)\boldsymbol{\omega}\,(i-1) = \left[\boldsymbol{\Phi}^T(i)\left\{\sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\,\boldsymbol{\varphi}(j)\right\}\right]$$

$$= \left[[\boldsymbol{\varphi}(i-K+1),\dots,\boldsymbol{\varphi}(i)]^T\left\{\sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\,\boldsymbol{\varphi}(j)\right\}\right]$$

$$= [\sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\,\boldsymbol{\varphi}^T(i-K+1)\boldsymbol{\varphi}(j),\dots,$$

$$\sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\boldsymbol{\varphi}^T(-1)\boldsymbol{\varphi}(j), \sum_{j=1}^{i-1} \boldsymbol{a}_j\,(-1)\,\boldsymbol{\varphi}^T(i)\boldsymbol{\varphi}(j)]$$

Now using kernel trick, the output is written as

$$= \left[\sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\kappa_{i-K+1,j},\dots,\sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\kappa_{i-1,j}, \sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\kappa_{i,j}\right]$$

Error is computed by

$$e(i) = \boldsymbol{d}(i) - \boldsymbol{\Phi}(i)^T\boldsymbol{\omega}(i-1)$$

$$e(i) = \boldsymbol{d}(i) - \left[\sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\kappa_{i-K+1,j},\dots,\sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\kappa_{i-1,j}, \sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\kappa_{i,j}\right]$$

Putting equation 4.7 and $\boldsymbol{\Phi}(\mathrm{i}) = [\boldsymbol{\varphi}(i-K+1),\dots,\boldsymbol{\varphi}(i)]$ in weight update equation of KAPA-1 (equation 4.5), we get

$$\boldsymbol{\omega}(i) \;=\; \boldsymbol{\omega}(\,i-1) \;+\; \mu\,\boldsymbol{\Phi}(i)\,[\boldsymbol{d}(i) - \boldsymbol{\Phi}(i)^{\boldsymbol{T}}\,\boldsymbol{\omega}(i-1)]$$

$$\boldsymbol{\omega}(i) \;=\; \boldsymbol{\omega}(i-1) \;+\; \mu\,\boldsymbol{\Phi}(i)\,e(i)$$

$$= \sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\,\boldsymbol{\varphi}(j) + \mu[\boldsymbol{\varphi}(i-K+1),\dots,\boldsymbol{\varphi}(i)]^T e(i)$$

$$= \sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\,\boldsymbol{\varphi}(j) + \sum_{j=1}^{K} \mu\,e_j(i)\,\boldsymbol{\varphi}(i-j+1)$$

Where kernel evaluation notation $\kappa_{i,j} = \kappa\big(\boldsymbol{u}(i), \boldsymbol{u}(j)\big)$ is used for simplicity.

Algorithm 3: Kernel Affine Projection Algorithm

```
Initialization:

Chose step size parameter μ

$a_1(1) = \mu d(1)$

Computation:

while {u(i), d(i)} available do

  % allocate a new unit

  $a_i(i-1) = 0$

    for $k = \max(1, i - K + 1)$ to $i$ do

    % evaluate output of the current network

    $y(i;k) = \sum_{j=1}^{i-1} a_j(i-1)\kappa_{k,j}$

    % compute errors

    $e(i;k) = d(k) - y(i;k)$

    % update the $\min(i, K)$ most recent units

    $a_k(i) = a_k(i-1) + \mu e(i;k)$

    end for

      if $i > K$ then

      % keep the remaining

        for $k$= 1 to $i - K$ do

                $a_k(i) = a_k(i-1)$

   end for

      end if ; end while
```

4.5.2 KAPA-2 (Normalized KAPA)

In the same manner, using equation 4.3-4.4 and 4.7 in equation 4.1, the Newton Recursion for kernel algorithm is summarized as bellow.

$$\boldsymbol{\omega}(i-1) = \sum_{j=1}^{i-1} \boldsymbol{a}_j\,(i-1)\,\boldsymbol{\varphi}(j)$$

$$\boldsymbol{e}(i) = \boldsymbol{d}(i) - \boldsymbol{\Phi}(i)^T \boldsymbol{\omega}(i-1)$$

and

$$\omega(i) = \omega(i-1) + \mu\boldsymbol{\Phi}(i)\,[\boldsymbol{G}(i) + \varepsilon\boldsymbol{I}]^{-1}\boldsymbol{e}(i)$$

where $\mathrm{G}(i) = \boldsymbol{\Phi}(i)^T\boldsymbol{\Phi}(i)$ is named Gram matrix.

The rest of the algorithm is similar to that of Algorithm 3.

Chapter 5

5 APPLICATIONS

The two kernel algorithms explained earlier are applied to two problems; one from class of inverse modeling and other from prediction. What is nonlinear channel equalization and what is its importance? What is Mackey-Glass time series and what is its importance and usage? These are the two question to be answered next.

5.1 Nonlinear Channel Equalization

Equalization is a practice used to improve link quality in unfriendly environment. The basic idea of equalization is to reverse the distortion incurred by the communication channel. In typical communication system information signal is transmitted over transmission channel. Transmitted signal while passing through transmission channel get corrupted by the channel distortions.

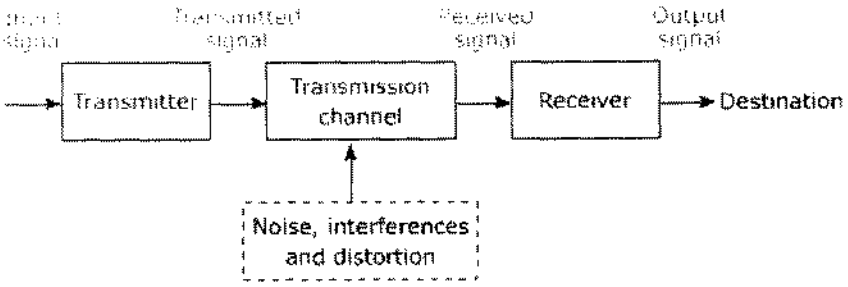

Figure 4: A typical communication system

To equalize the channel impairments and the aim to get error free communication equalizer are used. Most of the real world communication channels are nonlinear e.g. wireless communication channels and satellite communication channels etc. Nonlinear channel is modeled as cascade connection of linear filter and subsequent nonlinearity, as depicted in Figure 5.

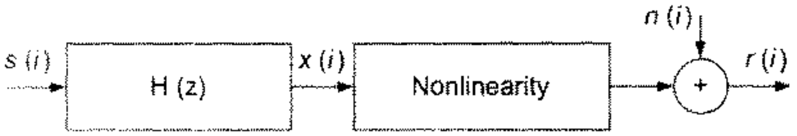

Figure 5: Nonlinear channel model

H(z) is transfer function of linear finite impulse response (FIR) filter. This FIR filter is modeled as

$$x(i) = s(i) + 0.5s(i-1)$$

It is very clear from the above equation that the current signal $s(i)$ and a delayed version of the signal scaled by 0.5 are added together to get output. The output is then needed to give as an input to nonlinearity defined by the following equation.

$$r(i) = x(i) - 0.9\, x^2(i) + n(i)$$

The square term in the above nonlinear equation is responsible for nonlinearity. Where $n(i)$ is white Gaussian noise which is added to the received signal. Hence nonlinear system is combinely represented by the following set of equations.

$$\begin{aligned} x(i) &= s(i) + 0.5s(i-1) \\ r(i) &= x(i) - 0.9\,x^2(i) + n(i) \end{aligned} \tag{5.1}$$

Linear channel impairments, that result in transmission quality degradation, like attenuation, spreading and phase jitter and nonlinear channel impairments like interference, shadowing, harmonics, additive noise, Inter Symbol Interference (ISI), produced by multipath in time dispersive channels, are compensated with equalizers.

Channel impairments causes uncertainty in the received data samples. Receiver receives a distorted signal. The samples of the received signal take any value instead of taking discreet levels that were supposed to receive. Receiver is left with no option except to estimate the signals originally transmitted. Channel transfer function is used to model the channel characteristics in which certain signal components are attenuated and delayed uniquely. Coefficients of the channel are unknown and time-varying generally. Statistical changes in the channel causes channel coefficients to vary accordingly. The job of the equalizer is to track the time-varying channel characteristics iteratively then try to reverse its effects and hence called *adaptive equalizer*. [20] A typical block representation of adaptive equalizer is depicted in the **Error! Reference source not found.**. In which $u(n)$ is input to the system of which transfer function is unknown. In telecommunication we may refer this system as channel through which a signal will be transmitted. Signal distortions are due to this channel and equalizer is there to reverse the changes incurred by this channel. Output of the channel $x(n)$ is treated as input to the adaptive equalizer. Output of the equalizer is then subtracted from desired response $d(n)$ resulting in the error. This error $\boldsymbol{d}(n)$ and output of the channel $x(n)$ is used to adaptively adjust transfer function of adjustable filter in least square sense. This process is carried out iteratively until mean square error (MSE) is minimized to its least steady state position.

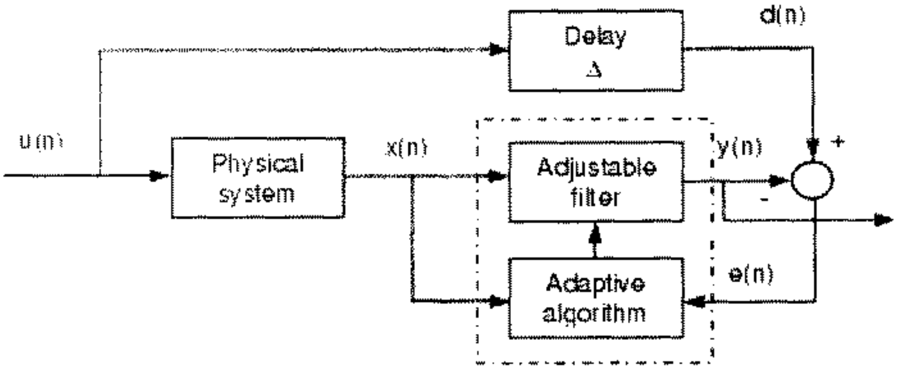

Figure 6: Block diagram of an adaptive equalizer

Adaptive equalizations are of two types, basically: one is linear and the other is nonlinear equalization. Linear equalizer is simple implemented without feedback path. The nonlinear equalizer, on the other hand, has some feedback mechanism. These are extensively used in wireless communication channels where linear equalizers are insufficient to cope with the nonlinearities introduced by the channel [21].Equalizer is trained by feeding training sequence to the equalize input. Data transmission in OFDM systems is accomplished by sending data chunks or data blocks one after other. To equalize such a channels training sequence is inserted between successive data blocks. The purpose of this training sequence is to inform the decoder to regulate its parameters for approaching data blocks [22] [23].

5.2 Mackey-Glass time series

Most of the recurrently encountered data in real world has chaos in nature and thus constitutes chaotic time series. Mackey-Glass time series is frequently used prominent chaotic time series. In 1977 Mackey and Glass model physiological system using nonlinear ordinary differential equation and hence called Mackey-Glass time series. Interesting examples of Mackey-Glass time series are weather data, stock exchange data. The Mackey-Glass time series is extensively applied in glucose metabolism and production of red blood cells [24]. So, Mackey-Glass time series is used for performance analysis of many nonlinear algorithms. The prediction of chaotic time series means to predict delayed version of $x(t)$. Predicting Mackey-Glass time series is benchmark test in the community of time series prediction. This system has no input and only one output. The Mackay-Glass equation is time delay differential equation. This nonlinear time series is expressed as follow

$$\frac{dx(t)}{dt} = -bx(t) + \frac{ax(t-\tau)}{1+x(t-\tau)^{n}}$$

In above equation the parameters a, b and n have real values while τ is real valued time delay. $x(t)$ represents the value at time t and $x(t - \tau)$ represents values after some delay of τ. It represent chaotic behavior when $\tau > 17$. This equation was used by Mackey and Glass to model physiological control systems [15]. Mackey-Glass equation represents a typical feedback system. The key benefit of its simplicity has led to make it standard model to assess the nonlinear modeling capability of nonlinear algorithms.

Chapter 6

6 RESULTS AND SIMULATIONS

Application of nonlinear filtering techniques known as kernel algorithms is studied on nonlinear channel equalization and prediction of short time Mackey-Glass time series. This thesis is confined to the application of the two algorithms, among the family of kernel algorithms, kernel LMS and kernel APA. Nonlinear channel is equalized with KLMS and KAPA and their performance comparison is shown.

6.1 Nonlinear channel equalization

6.1.1 Comparison of LMS and KLMS

Due to simplification of LMS it is extensively used for channel equalization but here KLMS is applied for equalization of nonlinear channel problem. A communication channel, which is nonlinear in nature, is exhibited as linear FIR filter with a subsequent nonlinearity, as shown in Figure 5. This nonlinear channel is characterized by equation (5.1. A signal $s(i)$, where i is from 1 to N, is transmitted through the nonlinear channel. At the receiving end of the channel additive white Gaussian noise (AWGN) corrupts the signal. The signal thus received is represented as $r(i);\ 1 \leq i \leq N$. $s(i)$ is the input signal to be transmitted through nonlinear channel, $n(i)$ is white Gaussian noise. Its mean is zero and the variance is σ^2. While $r(i)$ is received signal to be given to equalizer as its input. The job of the equalizer is to track the time-varying channel characteristics iteratively then try to reverse its effects and hence called adaptive equalizer.

6.1.2 MSE performance

Mean square error is measured for each algorithm and then mean square error for LMS and KLMS is compared. The mechanism stated in Algorithm 2 is followed to implement KLMS. 1000 data points are used for training the nonlinear filter then 5000 data samples are tested subsequently. Training data points updates the weights of the filter. Then this updated weight is used for testing data. Again Gaussian kernel is selected with kernel parameter 0.07 and μ 0.1. For LMS μ is 0.009. The results, hence achieved, are portrayed in the figure 5.1.

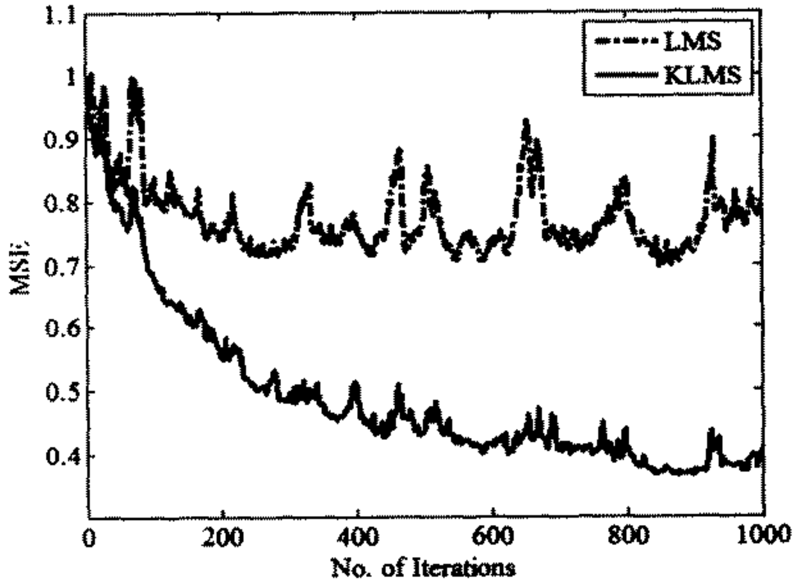

Figure 7: MSE comparison for LMS and KLMS for channel equalization problem

It is obvious from the above learning curves that KLMS outperforms LMS in terms of MSE.

6.1.3 BER performance

Another useful comparison is tabulated bellow. This table compares LMS and KLMS for changed noise levels. All of the results produced in subsequent table is generated for 10 Monte Carlo simulations. Kernel parameter is 0.1 while step size for LMS and KLMS is 0.006 and 0.2 respectively. The results are summarized in the form of "mean ± standard deviation". The comparison is made for three different noise levels.

Table 1: Mean error comparison for LMS and KLMS for different noise levels in nonlinear channel equalization problem

Algorithm	**BER($\sigma = 0.2$)**	**BER($\sigma = 0.5$)**	**BER($\sigma = 0.7$)**
LMS(μ=0.006)	0.16687±0.01946	0.18548±0.00937	0.20465±0.011508
KLMS(μ=0.2)	0.010693±0.0036213	0.074813±0.0084	0.12073±0.022388

Table 1 shows better results for KLMS. For each noise level or certain BER mean error is less for KLMS compared to LMS.

6.1.4 MSE Comparison of KAPA-I and KAPA-II with the previous algorithms

Mean square error performance is compared for LMS, affine projection, Kernel LMS, Kernel APA-1 and Kernel APA-2. Training data is 1000 samples and testing data contain 5000 samples. The kernel parameter for Gaussian kernel is 0.1. The σ is set to 0.1. Training sequence is used to update weights of the filter. This updated weight is further used for testing data. The mean square error for the test data set is calculated and is plotted against number of iterations.

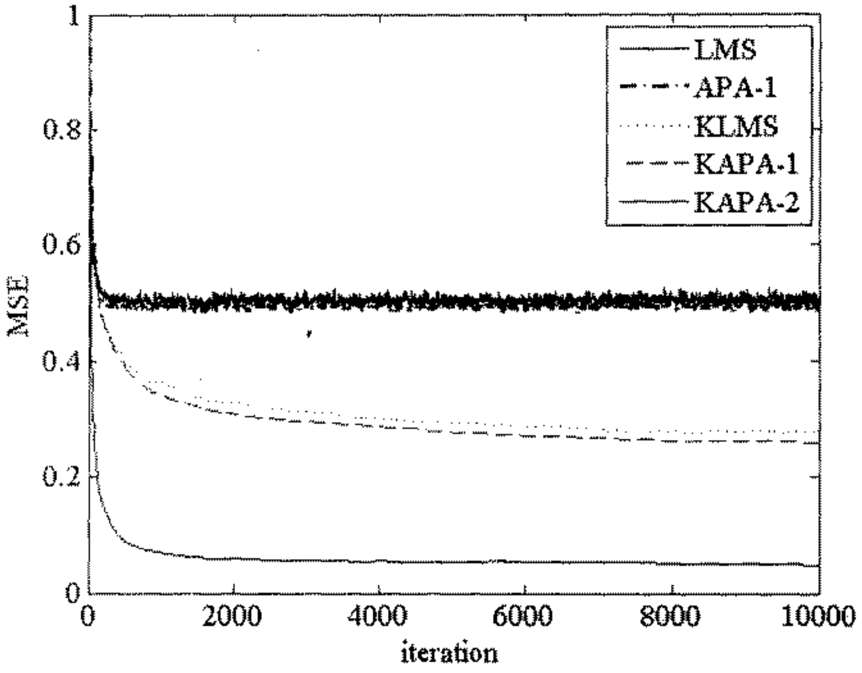

Figure 8: MSE graphs for LMS, KLMS, APA-1, KAPA-1 and KAPA-2 compared

Results for 50 Monte Carlo simulations are plotted in Figure 8. Mean square error is calculated from the output and the desired signal. Graph shows that the performance of linear filtering algorithms are comparable i.e. LMS and APA - 1 curves are almost overlapping. Similarly KLMS and KAPA-1 are alike but KAPA-2 outperforms all of the above.

6.2 Mackey-Glass Time Series Prediction

Mackey-Glass time series is a time series, chaotic in nature, and is generated by time delay nonlinear ordinary differential equation as bellow

$$\frac{dx(t)}{dt} = -bx(t) + \frac{ax(t-\tau)}{1 + x(t-\tau)^c}$$

where a, b, c and τ are real numbers. It is observed that Mackey-Glass time series shows chaotic behavior when τ is greater than 16.8 ≅ 17 [25]. It is customary to take $a = 0.1$, $b = 0.1$ and $\tau = 30$. while using these parameters in above equation 5000 samples of the time series data are obtained and are stored in the mat file mg.mat. Samples of the time series data are obtained by sampling $x(t)$, continuous curve, at the sampling rate of 6 seconds. The chaotic behavior of Mackey-Glass series, thus formed, is shown in Figure 8.

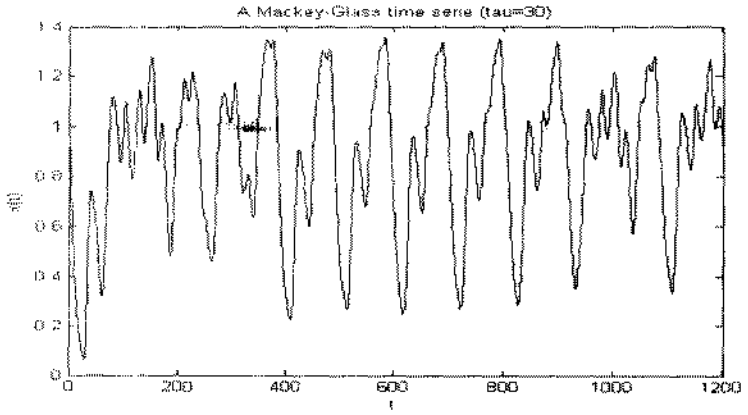

Figure 9: Chaotic behavior of Mackey-Glass Time Series (τ=30)

6.2.1 Performance of LMS and KLMS in terms of MSE

The order of the filter i.e. M is 10. It implies that most recent 10 values from the past input samples are taken to predict the current value, our desired response. Aim is to compare performance of LMS against KLMS. A portion of 500 data points is taken to train the filter and other 100 points from the mat file mg.mat for testing purpose. Additive white Gaussian noise which corrupts the data samples is zero mean and with the variance σ = 0.4. Step size parameter μ is 0.2 for LMS. In KLMS case Gaussian kernel is default choice. For Gaussian kernel, the kernel parameter i.e. 'a' is taken 1. Here the step size is also chosen 0.2. Training data is used to update the weights and then these weights are used to find error on testing data. The mean of the squared error is our performance measure. MSE is shown against number of iteration in Fig.

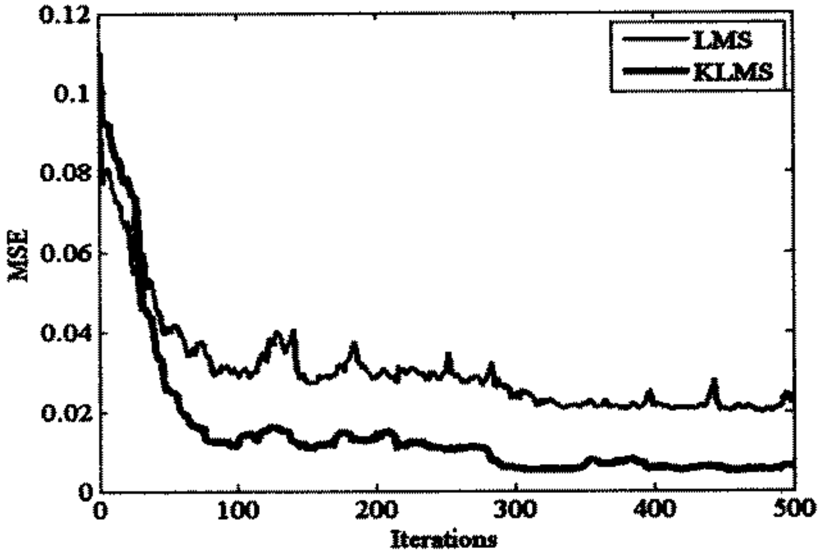

Figure 10: Mean Square Error graph for LMS and KLMS

This plot shows that increase in number of iteration reduces the mean square error. Same behavior is observed for LMS and KLMS. But KLMS outperforms LMS and steady state MSE for KLMS is lesser than that of LMS. It is worth noting that convergence of the two algorithms are same which shows that eigen-value spread in reproducing kernel Hilbert space is similar to that of input space.

Another comparison between LMS and KLMS is shown in Table 2. For KLMS the kernel parameter is one in Gaussian kernel i.e. a=1. The results are summarized after taking 150 Monte Carlo simulations. Noise power for this simulation is 0.04.

Table 2: MSE Performance comparison for Kernel LMS and LMS for different μ

Algorithms	**Training MSE**	**Testing MSE**
LMS	0.018904±0.00063824	0.020606±0.0011907
KLMS(μ = 0.1)	0.0075952±0.00030297	0.0080318±0.00078386
KLMS(μ = 0.2)	0.0055855±0.0003477	0.0055852±0.00075515
KLMS(μ = 0.6)	0.0055259±0.0011532	0.0061523±0.0018801

Results in the above table is tabulated in "average ± standard deviation" manner. It is clear from the table that KLMS outperforms LMS. This is mainly due to nonlinear modeling capability of KLMS (as time series being tested is nonlinear).

6.2.2 Performance of KAPA-I and KAPA-II

Kernel affine projection algorithm is applied to Mackey-Glass time series prediction. Order of the filter is chosen 7. It implies that 7 most recent values from the past input samples are taken to estimate the current sample, or the desired response. The aim is to compare performance of KAPA-1, KAPA-2 with LMS and KLMS. A portion of 500 data points is used to train the filter and another 100 data samples for testing mean square error. Training samples are used to update the weights of the filter and then this updated weights are used to test mean square error. Additive white Gaussian noise which corrupts the samples has zero mean and with the variance $\sigma = 0.001$. Again for LMS the step size is 0.2. Gaussian kernel is taken as kernel for KLMS. For Gaussian kernel the kernel parameter is chosen 1. Here the step size is also chosen 0.2. Training data is used to update the weights and then these weights are used to find error on testing data. The mean of the squared error is our performance measure. MSE is shown against number of iteration in Figure 10.

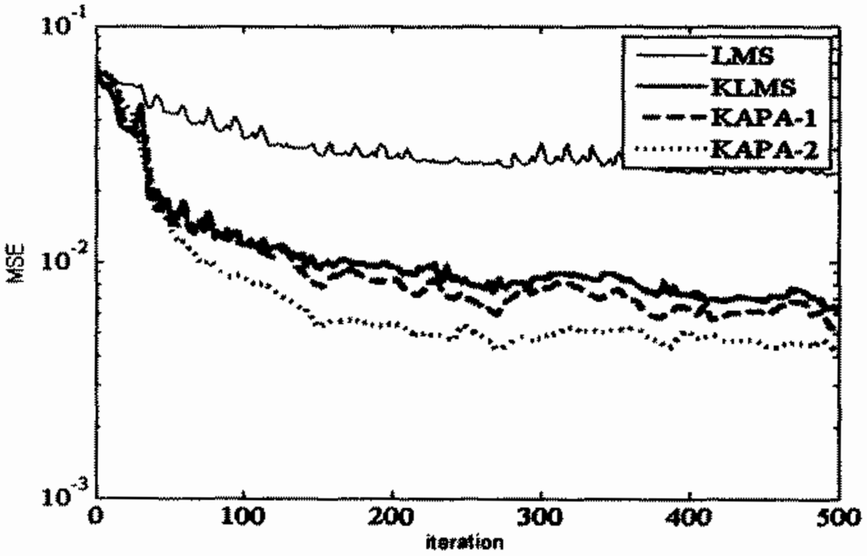

Figure 11: MSE comparison for LMS, KLMS, KAPA-1 and KAPA-2(Mackey-Glass)

[illegible] curves are compared for different step sizes.

Table 3: MSE performance comparison for LMS, KLMS, KAPA-1 and KAPA-2

Algorithms	Parameters	Test mean square errors
LMS	μ = 0.04	0.024608±0.00078585
KLMS	μ = 0.2	0.0070493±0.00031619
KAPA-1	μ = 0.04 , K =10	0.0061397±0.00038324
KAPA-2	μ = 0.04, K = 10 , ε = 0.1	0.0046028±0.00017596

[illegible] results are tabulated in the form of "average ± standard deviation". [illegible] from the results that kernel algorithms shows much better results compared to their linear counter parts. Moreover, KAPA outperforms KLMS in terms of MSE.

Chapter 7

7 CONCLUSION AND FUTURE WORK

Nonlinear channel equalization and prediction of Mackey-Glass time series are studied thoroughly. Two algorithms, kernel least mean square algorithm and kernel affine projection algorithm, among the family of kernel algorithms are applied on these problems. Kernel least mean square is the simplest among all. Kernel affine projection algorithms give relatively better results.

7.1 Conclusions

Results and simulation presented in the last section, the following conclusions are drawn.

i. Nonlinear kernel adaptive filters i.e. KLMS and KAPA are efficiently applied on the two nonlinear problems i.e. nonlinear channel equalization and Mackey-Glass time series prediction.

ii. In both channel equalization and Mackey-Glass prediction problems learning curves for mean square error shows that KLMS and KAPA are convergent and give less mean square error compared to linear adaptive algorithms. KAPA is even better than KLMS in terms of MSE curves.

iii. For different noise levels the results are consistent. Tabulated results shows that increasing bit error rate decreases the performance. Still, for KLMS mean square error is lesser than that of LMS.

iv. The variations in step sizes for KLMS show the improvement in performance, hence MSE is decreased.

v. The comparative studies of linear and kernel adaptive algorithms, in all cases, reveals the fact that for nonlinear problems kernel algorithm, due to their high dimensionality and universal approximation capability, give better results.

7.2 Future Directions

In future one may go for other kernel algorithms like kernel recursive lease square and extended kernel least square algorithm for nonlinear problems. Heuristic computation techniques like genetic algorithm, particle swarm optimization, differential evolution, ant colony optimization and bee colony optimization etc. can also be studied for nonlinear channel equalization and Mackey-Glass time series prediction.

References

[1] A. Khalaf, M. Ashraf, "Improvements in the channel Equalizer Performance Using Modified LMS and BP algorithm," *IJCSI International Journal of Computer Science Issues,* vol. Vol. 9, no. Issue 2, March 2012.

[2] F. Tong, B. Benson, Y. Li, R. Kastner, "Channel equalization based on data reuse LMS algorithem for shallow water achosic communication".

[3] S. Abrar, "Stop-and-Go Algorithms for Blind Channel Equalization in QAM Data Communication system," *COMSATS-IIT (CIIT),* pp. 35-40.

[4] W.C. Mead, R. D. Jonesalb, Y. C. Lee, C. W. Barnesa, "Using CNLS-Net to Predict the Mackey-Glass Chaotic Time Series," *IEEE,* pp. 485-490, 1991.

[5] H. Samueli, Y. Lee, "Adaptive antenna arrays and equalization techniques for high bit-rate qam receivers".

[6] J. Principe, W. Liu , S Haykin, Kernel Adaptive filtering: A comprehensive introduction, Wiley, 2010.

[7] S. Haykin, Adaptive Filtering Theory, Prentice Hall, 2001.

[8] P. Sharma, P. Gupta, P. K. Singh, "Performance Comparison of ZF, LMS and RLS Algorithms for Linear Adaptive Equalizer," *Advance in Electronic and Electric Engineering,* vol. 4, pp. pp. 587-592, 2014.

[9] H. C. Shin, A. H. Sayed, "Variable Step-Size NLMS and Affine Projection Algorithms," *IEEE SIGNAL PROCESSING LETTERS,* vol. 11, no. 2, pp. p. 132-135, 2014.

[10] F. Albu, D. Coltuc, M. Rotaru, A. Nishikawa, "An efficient implementation of the kernel affine projection algorithm," in *8th International Symp osium on Image and Signal Pro cessing and Analysis (ISPA 2013)*, Trieste, 2013.

[11] D. Atherton, An Introduction to nonlinearity in controll system, Ventus Publishing ApS, 2011.

[12] W. S. Steven , "chapter 5: Linear Systems," in *The Scientist and Engineer's Guide to Digital Signal Processing*, pp. 87-106.

[13] V. Lakshmicantham, S Leela, A.A. Martynyuk, Stability analysis of nonlinear systems, New York: Marcell Decker Inc., 1989.

[14] F. L. Lewis, S. Jagannathan, A. Yesildirek, Nueral Network controll of Robot Manipulator and Nonlinear systems, London: Taylor and Fransis Ltd, 1999.

[15] R. Schwaiger, H. A. Mayer, "Evolutionary and Coevolutionary Approaches to Time Series Prediction Using Generalized Multi-Layer Perceptrons," *IEEE*, 1999.

[16] G. Emch, Algebraic methods in statistical mechanics and quantum field theory., New York: Wiley-Interscience, 1972.

[17] P. K. Jain, K. Ahmad, "5.1 Definitions and basic properties of inner product spaces and Hilbert spaces," in *Functional analysis (2nd ed.)*, New Age International, 1995, p. p. 203.

[18] N. Young, An introduction to Hilbert space, Cambridge University Press, 1988.

[19] B. lkopf, M. O. Franz, B. Sch ö. "A unifying view of Wiener and Volterra theory and polynomial kernel regression .," *Neural Computation*, p. 3097 – 3118, 2006.

[20] L. Atapattu, G. M. Arachchige, K. Ziri-Castro, H. Suzuki, D. Jayalath, "Linear Adaptive Channel Equalization for Multiuser MIMO-OFDM systems," *IEEE*, 2012.

[21] R. Rappaport, Wireless Communications Principles and Practice, Prentice Hall, 1996.

[22] A. Tchamkerten ,I. E. Telatar, "On the use of training sequence for Channel estimation," *IEEE transactions of information theory.*

[23] J. H. Manton, "Optimal Training Sequences and Pilot Tones for OFDM systems," *IEEE*, 2001.

[24] S. Bhardwaj, S. Srivastava, J.R.P Gupta, "Chaotic Time Series Prediction Using Combination of Hidden Markov Model and Neural Nets," *IEEE*, 2010.

[25] M. F. T. H. William, "Plastic Network for Predicting the Mackey-Glass Time Series," *IEEE*, pp. 941-946, 1992.

[26] P. P. Pokharel, W. Liu, J. C. Principe. "Kernel LMS," *IEEE*, 2008.