T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ROBUST ADAPTIVE LEARNING APPROACH OF ARTIFICIAL NEURAL NETWORKS

Alaa Ali Hameed HAMEED

DOKTORA TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Ocak-2017

KONYA

TEZ KABUL VE ONAYI

Alaa Ali Hameed tarafından hazırlanan "Robust Adaptive Learning Approach of Artificial Neural Networks" adlı tez çalışması 09.01.2017 tarihinde aşağıdaki jüri tarafından oy birliği ile Selçuk Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda DOKTORA TEZİ olarak Kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Doç. Dr. Sabri KOÇER	.....
Danışman Yrd. Doç. Dr. Barış KOÇER	.....
Üye Doç. Dr. Gülay TEZEL	.....
Üye Doç. Dr. Seral ÖZŞEN	.....
Üye Yrd. Doç. Dr. Onur İNAN	.....

Yukarıdaki sonucu onaylarım.

Prof. Dr. Mustafa YILMAZ
FBE Müdürü

TEZ BİLDİRİMİ

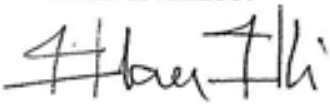
Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Alaa Ali Hameed HAMEED

Tarih: 09.01.2017



ÖZET

DOKTORA TEZİ

YAPAY SİNİR AĞLARI İÇİN SAĞLAM ADAPTİF ÖĞRENME YAKLAŞIMI

Alaa Ali Hameed HAMEED

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Barış KOÇER

2016, 152 Sayfa

Jüri
Yrd. Doç. Dr. Barış KOÇER
Doç. Dr. Gülay TEZEL
Doç. Dr. Sabri KOÇER
Doç. Dr. Seral ÖZŞEN
Yrd. Doç. Dr. Onur İNAN

Adaptif filtre tekniklerin sinyal işlemede sıklıkla kullanılmaktadır. Genellikle adaptif filtrenin kararlı durum hataların-karelerinin ortalaması (HKO) ve yakınsama hızı arasında bir seçim yapmak gerekir. Bu seçim genelde adım boyutu parametresi ile ayarlanır. Küçük adım sayısı yavaş yakınsama ve düşük kararlı durum hatasına sebep olurken tersi durum ise hızlı yakınsama ve yüksek kararlı durum hatasına sebep olur. Bu sorunu aşabilmek için rekürsif invers (RI) ve ikinci seviye rekürsif inverse RI algoritmalarının konveks kombinasyonları kullanılmıştır. Geliştirilen bu yeni metot sistem tanımlama ve gürültü engelleme uygulamalarında kullanılmıştır. Önerilen metot, hataların karelerinin ortalaması (HKO) ve yakınsama hızı bakımından "normalize en küçük ortalama kareler" (NEKOK)'nın konveks kombinasyonu ile karşılaştırılmıştır. Deneysel sonuçlar, toplanır beyaz Gaussian gürültüsü (TBGG) ve toplanır ilişkili Gaussian gürültüsü (TIGG) eklenmiş ortamlarda çalıştırıldığında, önerilen algoritmanın daha hızlı yakınsadığını ve daha küçük HKO değerleri ortaya koyduğunu göstermiştir.

Manyetik rezonans görüntülerindeki (MRI) gürültüleri azaltmak tıbbi teşhiş alanında ilgi çekici bir alan olmaya başlamış ve bu konuyla ilgili birçok metot önerilmiştir. Fakat bu algoritmaların çoğu düşük kalite veya yavaş çalışmaktan muzdariptirler. Bu sorunu çözmek için önerilen tek boyutlu konveks kombinasyon, iki boyutlu konveks kombinasyona dönüştürülmüştür. İki boyutlu konveks kombinasyon, gürültü azaltma konusunda yüksek performans sunmaktadır. Algoritmanın performansını ölçmek için bir MR görüntüsünün toplanır beyaz Gaussian gürültüsü (TBGG) ile bozulduğu varsayılmış ve bu bozulma önerilen algoritma ile düzeltilmeye çalışılmıştır. Simülasyonlar algoritmanın görüntüyü başarılı bir şekilde düzelttiğini göstermektedir.

Bu tezde bir öğretici ve bir öğreticisiz olmak üzere iki yeni yapay sinir ağı metodu önerilmiştir. Öğreticili öğrenmede "değişken adaptif momentumlu geri yayılım algoritması" (DAMGY) adında yeni bir sınıflandırıcı önerilmiştir. Bu algoritma geri yayılım algoritmasının modifiye edilmiş bir halidir ve kararlı durum hataya yakınsama hızını arttırırken hata oranınıda düşürerek desen tanıma performansını arttırmayı amaçlar. Bu algoritma girişin oto korelasyon matrisinin katsayılarını kontrol eden öğrenme katsayısı tarafından kontrol edilir. Bu katsayı sayesinde ağırlıklar güncellenirken düşük hata oranı yakalanmaktadır. Algoritmanın performansını ölçmek için k-en yakın komşu (k-EYK), Naive Bayes (NB), doğrusal ayırteden analizi (DAA), Destek Vektör Makineleri (DVM), geri yayılım ve adaptif momentumlu geri yayılım algoritması (AMGY) kullanılmış ve performans, yakınsama hızı, hataların karelerinin ortalaması ve doğruluk bakımından değerlendirilmiştir.

Öğreticisiz öğrenmede, birçok yapay zekâ uygulamasında kullanılan kendi kendini düzenleyen harita (KDH) algoritması birçok araştırmacının ilgisini çekmektedir. Bu tezde klasik KDH algoritmasına adaptif bir öğrenme becerisi kazandıran bir algoritma önerilmiştir. Önerilen KDH algoritması değişken öğrenme katsayısı ile optimal ağırlıkları ve kazanan nöronları kısa sürede bularak klasik KDH'un dezavantajlarını yok etmektedir. Önerilen KDH algoritmasının optimum ağ ağırlıklarını bulma hızı diğer öğreticisiz algoritmalarla karşılaştırılmıştır. Ayrıca önerilen KDH algoritması klasik KDH, kendi kendini düzenleyen harita ile Gauss fonksiyonu (KDHGF) ve parametre-az kendi kendini düzenleyen harita (PAKDH) algoritmalarıyla da karşılaştırılmıştır. Önerilen KDH algoritması yakınsama hız, niceleme hızı, bulunan ağın topoloji hatası ve doğruluk kriterlerinde üstün performans gösterdiği gösterilmiştir. DAMGY ve önerilen KDH algoritmasının performansı UCI ve KEEL veritabanlarından alınan veri setleri ile de test edilmiştir.

Anahtar Kelimeler: Adaptif filtreler, iki boyutlu konveks kombinasyon, MRI, öğreticili öğrenme, yapay sinir ağı, geri yayılım, adaptif momentum, öğreticisiz öğrenme, kendi kendini düzenleyen harita (KDH), öğrenme katsayısı.

ABSTRACT

Ph.D THESIS

ROBUST ADAPTIVE LEARNING APPROACH OF ARTIFICIAL NEURAL NETWORKS

Alaa Ali Hameed HAMEED

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING

Advisor: Assist. Prof. Dr. Barış KOÇER

2016, 152 Pages

Jury
Assist. Prof. Dr. Barış KOÇER
Assoc. Prof. Dr. Gülay TEZEL
Assoc. Prof. Dr. Sabri KOÇER
Assoc. Prof. Dr. Seral ÖZŞEN
Assist. Prof. Dr. Onur İNAN

Adaptive filtering techniques are frequently used in signal processing applications. In adaptive filters, usually there is a trade-off between the steady-state mean-square error (MSE), and the initial convergence rate of the filter. This trade-off is usually controlled by the step-size. A small step-size leads to a relatively slow convergence rate with low MSE and vice versa. A new convex combination of recursive inverse (RI) and second-order RI algorithms is developed to overcome this trade-off. The new method used in system identification and noise cancellation applications. Proposed method is compared to convex combination of the normalized least-mean-square (NLMS) algorithms in terms of mean-square error (MSE) and rate of convergence. Simulations show that the proposed algorithm provides faster convergence rate with lower MSE than combined NLMS algorithms in both additive white Gaussian noise (AWGN) and additive correlated Gaussian noise (ACGN) environments.

De-noising magnetic resonance images (MRI) has recently become an interesting topic in medical diagnosis applications. Many algorithms have been proposed for this purpose. However, these algorithms usually suffer from poor performance or time consumption. In order to improve the MRI images, the proposed 1-D convex combination method extended to 2-D convex combination. The 2-D convex combination provides high performance in terms of noise removal. A de-noising experiment has been conducted on MR image that is assumed to be corrupted by an additive white Gaussian noise (AWGN) for testing purposes. Simulations show that the proposed algorithm successfully recovers the image.

In this thesis we present two modified neural network algorithms. One of them is supervised and the other is unsupervised learning. In the supervised learning, a novel machine learning classifier of back-propagation algorithm with variable adaptive momentum (BPVAM) is proposed. The proposed algorithm is a modified version of the BP algorithm to improve its convergence behavior in both sides, accelerate the convergence process for accessing the optimum steady-state and minimizing the error misadjustment to improve the recognized patterns superiorly. This algorithm is controlled by the adaptive momentum parameter which is dependent on the eigenvalues of the autocorrelation matrix of the input. It provides low error performance for the weights update. To discuss the performance measures of the BPVAM algorithm and the other supervised learning algorithms such as K-nearest neighbours (K-NN), Naive Bayes (NB), linear discriminant analysis (LDA), support vector machines (SVM), BP, and BP with

adaptive momentum (BPAM) have been compared in term of speed of convergence, sum of squared error (SSE), and accuracy.

In the unsupervised learning, the self-organizing map (SOM) has attracted attention of many researchers, where it has successfully applied to a wide range of artificial intelligence applications. In this thesis, new intelligent adaptive learning of the conventional SOM algorithm is proposed. The proposed SOM overcomes the disadvantages of the conventional SOM by deriving a new variable learning rate that can adaptively achieve the optimal weights and obtain the winner neurons in a short time. Performance of the proposed SOM was compared with other unsupervised algorithms by examining the speed of finding optimum network weight update. The proposed SOM algorithm was also compared with conventional SOM, Gaussian-function with self-organizing map (GF-SOM), parameter-less self-organizing map (PLSOM) algorithms. The proposed SOM algorithm showed superiority in terms of convergence rate, quantization error (QE), topology error (TE) of preserving map and accuracy during the recognition process. The BPVAM, and proposed SOM algorithms experiments were conducted using different databases from UCI and KEEL repository.

Keywords: Adaptive filters, convex adaptive filtering, 2-D convex combination, MRI, supervised learning, neural network, back-propagation, adaptive momentum, unsupervised learning, self-organizing map (SOM), learning rate.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisors Assist. Prof. Dr. Barış Koçer and Assist. Prof. Dr. Mohammad Shukri Salman for their continuous support, great guidance, endless help, knowledge and huge confidence they gave me.

I am also very grateful to Prof. Dr. Bekir Karlık for the advice and support, who has shown a large interest in my work. His stimulating motivation and valuable ideas helped me to complete this thesis. Our numerous discussions, either face-to-face or through the email, have greatly improved this work. I thank him for constructive criticism and careful revision of the text.

I would like to thank my thesis monitoring committee members Prof. Dr. Yüksel Özbay, Assoc. Prof. Dr. Gülay Tezel, Assoc. Prof. Dr. Seral Özşen for all of their guidance through this process; your discussion, ideas, and feedback have been absolutely invaluable.

I would also like to thank my colleagues and friends for all their help and collaboration. I also wish to thank my dear friend Murat Karakuş for his constant support and encouragement throughout my graduate career.

My warm regards to my amazing family, father, mother, brothers (Mohammed Ali Hameed, Ahmed Ali Hameed, Hussein Ali Hameed and Hassan Ali Hameed) and sisters for their love, support, and constant encouragement.

Alaa Ali Hameed HAMEED
KONYA-2016

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

1-D	1-Dimensional
2-D	2-Dimensional
ACGN	Additive Correlated Gaussian Noise
ANN	Artificial Neural Networks
AR	Autoregressive model
AWGN	Additive White Gaussian Noise
BMU	Best Matching Unit
BP	Backpropagation
BPAM	BP with Adaptive Momentum
BPFM	BP with Fixed Momentum
BPNN	Backpropagation Neural Network
BPVAM	Backpropagation Algorithm with Variable Adaptive Momentum
FIR	Finite Impulse Response
GF-SOM	Gaussian-Function with Self-Organizing Map
IIR	Infinite Impulse Response
K-NN	K-Nearest Neighbours
LDA	Linear Discriminant Analysis
LMS	Least Mean Square
LS	Least-Squares
MRI	Magnetic Resonance Images
MSE	Mean-Square-Error
NB	Naive Bayes
NLMS	Normalized Least-Mean-Square
PLSOM	Parameter-Less Self-Organizing Map
QE	Quantization Error
RI	Recursive Inverse
RLS	Recursive-Least-Squares
RZA-LLMS	Reweighted Zero-Attracting Leaky-Least-Mean Square
SNR	Signal-to-Noise Ratio
SOM	Self-Organizing Map
SSE	Sum-of-Squared-Error

SVM	Support Vector Machines
TDVSS	Transform Domain LMS with Variable Step-Size
TE	Topology Error
ZA-LLMS	Zero-Attracting Leaky Least-Mean-Square
I	Identity Matrix
R	Autocorrelation matrix
p	Cross-correlation vector
w	Weight coefficients
x	Input data
D	Training set
E	Expectation operator
Eu	Euler number
FN	False Negatives
FP	False Positives
N	Filter length
Q	Covariance matrix
T	Transposition operator
TN	True Negatives
TP	True Positives
d	Desired response
e	Estimation error
f	Activation function
$ln(.)$	Natural logarithm
q	Stochastic i.i.d.
t	Target value
v	Measurement noise
y	Output
$\boldsymbol{h}$	Unknown system
α	Adaptive momentum
β	Forgetting factor
η	Learning rate
θ	Threshold
λ	Convex combination parameter

μ	Step-size
σ	Width of Gaussian
ϵ	Scaling variable
σ^2	Variance

1. INTRODUCTION

Digital filters are classified such as linear or non-linear, continuous-time or discrete-time, and recursive or non-recursive (Ozbay et al., 2003). Adaptive filtering techniques are frequently used in signal processing applications (Sayed, 2003; Sayed, 2008). The performance of adaptive filter algorithms is usually measured in terms of convergence rate and/or the minimum mean-square-error (MSE). The optimum performance of an adaptive filter is usually achieved by the Wiener filter (Haykin, 2002). However, Wiener filter requires a prior knowledge about the input signal.

While adaptive filter algorithms that does not require this prior knowledge. The performance of this algorithms is controlled by a step-size parameter that generates a trade-off between the convergence rate of the adaptive filter and its SS-MSE (Haykin, 2002; Sayed, 2003). A constant step-size parameter is used to update the filter coefficients. This step-size parameter has a critical effect on the performance of the algorithm. A relatively large step-size value provides a fast convergence but high MSE. On the other hand, a small step-size causes slow convergence with low MSE.

To overcome such a trade-off, convex combination of adaptive filters started appearing in the last decade (Arenas-Garcia et al., 2006; Mandic et al., 2007; Kosat and Singer, 2009; Azpicueta-Ruiz et al., 2008; Trump, 2009a). Convex combination of adaptive filter has provided an improved performance in both stationary and non-stationary environments (Zhang and Chambers, 2006). The convex combination is usually constructed by two adaptive filters; one of them provides fast convergence with high output steady-state error and the other has slow convergence with low MSE (Martinez-Ramon et al., 2002). The schematic diagram of the convex combination of adaptive filters is shown in Fig. 2.1.

Therefore, a combination of such two systems would provide us, in a way, a solution of such trade-off. This solution is done by extracting out the faster converging part and the lower MSE part from the two filtering algorithms. By this the filter will provide a performance with fast convergence and low error.

A 2-Dimensional (2-D) adaptive filter is an extension of 1-D adaptive filters which deals with two dimensional signals (i.e., images) (Hadhoud and Thomas, 1998). One of the most attracting applications of such algorithms is image de-noising.

Artificial neural networks is an interesting research that is well-known in the area of machine learning and has provided the best solutions to many problems.

Currently it attracts more and more attendance in artificial intelligence. In general, the Artificial neural networks try to model useful high-level abstractions, learning hierarchies of concepts, and multiple-layer architectures for better data representations. Artificial neural networks is capable of learning complex abstractions and it is known to be especially useful for problems with high-dimensional input, such as speech and image recognition.

Backpropagation neural network (BPNN) is a supervised machine learning that has attracted the attention of many researchers in a wide range of applications. BPNN has high capability to solve complex problems that cannot be solved using traditional machine learning techniques (Karlik, 2014). There are many BPNN applications on marketing (Iseri and Karlik, 2009; Chiang et al., 2006), bioinformatics (Karlik, 2013), medicine (Behram et al., 2007; Karlik and Cemel, 2012), engineering (Dvir et al., 2006; Karlik, 2000), and others (Karlik et al., 1998; Lee et al., 2005). Unfortunately, it has a couple of obstacles that usually restrict its algorithm performance; the slow convergence and the high steady-state error. To overcome these obstacles, the momentum technique have been proposed, where it can speed up the convergence rate and decrease the steepest descent error (Yu et al., 1993; Yu and Liu, 2002; Shao and Zheng, 2009), efficiently.

Self-organizing map (SOM) as an unsupervised learning algorithm attracts many researchers. It can be applied in various applications such as clustering, image recognition, and sound recognition (Kohonen, 1990; Kohonen et al., 1996; Vesanto and Alhoniemi, 2000). SOM projects the input samples from high dimensional space to the low dimensional space (Kohonen, 1982; Kohonen, 2001). The architecture of SOM is simply consists of an input layer and an output layer with highly interconnection between each other where each connection is associated with a weight. The output is organized as a two dimensional grid of neurons, where the data visualization of the output is implemented during the learning process.

SOM algorithm does not require the knowledge of the target output in the recognition process (Mukherjee, 1997). The conceptual idea is that each input vector is multiplied by the weight vectors to calculate the distances. Furthermore the map nodes output must be the same weight vectors (Kohonen, 1989; Song and Hopke, 1996; Kim et al., 2002; Hoffmann, 2005; Marini et al., 2005; Bianchi et al., 2007). The winner output node is determined by the shortest distance between the input vector and output nodes. The winner weights node are updates for each input (Rojas, 1996; Astel et al.,

2007). The clustering inputs depending on the similarity of the input features (Fonseca et al., 2006). The time consumption of training depends on dataset size and whether it can reach the optimum weights.

1.1 Our Contributions

In this work we have proposed different algorithms; we have proposed a new convex combination RI adaptive filter and we derive its tracking analysis; for artificial neural networks we have proposed a new backpropagation algorithm called BPVAM, and a new SOM algorithm.

1.1.1 Adaptive filters

In order to overcome the trade-off in the field of adaptive filtering, various variable step-size LMS-type algorithms have been developed. Recently, Ahmad et al. (2011) have proposed a new recursive inverse (RI) adaptive algorithm. This algorithm has shown great performance compared to different adaptive algorithms with less or comparable computational complexity (Ahmad et al., 2010b). It has shown robust performance in impulsive noise and non-stationary environments (Ahmad et al., 2012; Ahmad et al., 2013b). However, the trade-off between the convergence rate and the MSE is not radically solved.

In this work, we propose a new RI convexly combined adaptive filtering algorithms which provides a very high performance, in terms of convergence rate and SS-MSE, compared to the other proposals. The new convex combination of RI algorithms is to extract the fast convergence and low MSE properties of two adaptive filters and combine their performances. The tracking performances of the convexly combined RI algorithms have been discussed theoretically and experimentally. It shows that the derived theoretical SS-MSE of the convexly combined RI algorithm is in match with the experimental one.

The performance of proposed convex combinations has compared to various adaptive algorithms and others convex combinations that shown in the literature (see Chapter 3):

1. Normalized least-mean-square (NLMS) algorithm.
2. Recursive inverse (RI) algorithm.

3. Second order RI Algorithm.
4. Convex of Two NLMS Algorithms.
5. Convex of Two RI Algorithms.
6. Convex RI and second order RI Algorithms.

In additive white Gaussian noise (AWGN) and additive correlated Gaussian noise (ACGN) environments in the settings of system identification, and noise cancellation.

Simulation results show that the new combinations highly outperforms other algorithm in terms of both MSE and convergence rate. Furthermore, the 2-D version of the algorithm has shown high performance in image de-noising.

1.1.2 Artificial neural networks

The proposed neural networks algorithms are BPVAM which is for supervised learning, and new SOM algorithm for unsupervised learning.

1.1.2.1 Supervised learning algorithm

The conventional BP algorithm has a couple of obstacles that usually limit its performance; the slow convergence and the high steady-state error. To overcome these obstacles, the new variable adaptive momentum technique has been proposed, where it can speed up the convergence rate and decrease the steepest descent error SSE, efficiently.

This momentum has been proposed as a learning rate in (Ahmad et al., 2011; Ahmad et al., 2010a; Ahmad et al., 2013a; Hameed et al., 2014) and has been used simply in various adaptive filtering applications successfully. This proposed adaptive momentum is characterized by two parameters (λ, β) providing lower computational cost and more robustness in various applications. It also shows better performance deservedly over the conventional BP and BPAM algorithms in terms of speed of convergence, SSE and accuracy.

To prove our claim, the performance of the proposed BP algorithm based on a variable adaptive momentum, shortly called BPVAM, are compared to different supervised machine learning techniques that take place in the literature (see Chapter 4):

1. K-nearest neighbours (K-NN).

2. Naive Bayes (NB).
3. Linear discriminant analysis (LDA).
4. Support vector machines (SVM).
5. Back-propagation (BP).
6. Back-propagation with adaptive momentum (BPAM).

For this purpose, the BPVAM and the other algorithms were applied on a benchmark-XOR problem and various datasets such as breast cancer, heart, heart-statlog, iris, lung-cancer, MAGIC gamma telescope, and wine datasets recorded at UCI repository in order to compare their performance. The results show that the performance of the BPVAM algorithm is better than the others.

1.1.2.2 Unsupervised learning algorithm

The major problem with SOMs is that they are very computationally expensive which is a major drawback since as the dimensions of the data increases, dimension reduction visualization techniques become more important, but unfortunately then time to compute them also increases. For calculating the input features similarity map, the more neighbors you use to calculate the distance the better similarity map you will get, but the number of distances the algorithm needs to compute increases exponentially.

In this thesis a new adaptive learning rate of decreasing functions. The proposed SOM algorithm overcomes many disadvantages of conventional SOM algorithm. It needs less implementation time (fast convergence), obtains low Quantization Error (QE) and better Topology Error (TE) preserving map during the recognition process.

Experiments also showed that the proposed algorithm is able to recognize more features and getting a higher accuracy compared to different unsupervised machine learning techniques that shown in the literature (see Chapter 4):

1. Conventional self-organizing map (SOM) algorithm.
2. Gaussian-function with self-organizing map (GF-SOM) algorithm.
3. Parameter-less self-organizing map (PLSOM) algorithm.

In this thesis, the new unsupervised learning algorithm is compared with conventional SOM, GF-SOM and PLSOM algorithms using different criteria such as QE, TE, CPU time and accuracy. Extensive experiments are conducted to evaluate the performance of proposed algorithm using different well-known datasets such as Appendicitis, Balance, Wisconsin breast, Dermatology, Ionosphere, Iris, Sonar and

Wine datasets from UCI and KEEL repositories. The new SOM outperforms the other algorithms in terms of the used criteria as shown in the results.

1.2 Overview Of This Thesis

This thesis is organized as follows. Introduction and our contributions are outlined in Chapter 1. Chapter 2 provides a literature review about the well-known adaptive filters and their significance, where some adaptive filtering algorithms have been presented by many researchers for better learning and adapting rates. Also, machine learning in supervised and unsupervised learning have presented. BP neural network, and SOM algorithms have presented with some of their variants.

Chapter 3 introduces the adaptive filters and their mechanism, system identification and noise cancellation configurations, some adaptive filtering algorithms, the proposed convex combinations including the one-dimensional (1-D), and two-dimensional (2-D) convexly combined RI algorithms.

In Chapter 4, machine learning in supervised and unsupervised learning are introduced. The supervised BP neural network and the unsupervised SOM learning algorithms are briefly described. In addition the proposed BPVAM, and proposed SOM algorithms derivations are proved.

In Chapter 5, in section one, the proposed convex combinations of RI algorithms compared to the performance of convex NLMS algorithms are discussed. In simulation results the combinations performance were investigated in terms of the MSE and convergence rate for system identification, and noise cancellation settings, in both additive white Gaussian noise (AWGN) and additive correlated Gaussian noise (ACGN) environments. In section two, the performance measures of proposed BPVAM algorithm and the other supervised learning algorithms such as K-nearest neighbours (K-NN), Naive Bayes (NB), linear discriminant analysis (LDA), support vector machines (SVM), BP, and BP with adaptive momentum (BPAM) are compared in terms of speed of convergence, sum-of-squared-error (SSE), and accuracy by implementing benchmark XOR-problem and seven datasets from UCI repository. In the last section, the proposed SOM algorithm was also compared to the conventional SOM, GF-SOM and PLSOM algorithmsin terms of convergence rate, quantization error (QE), topology error (TE) preserving map and accuracy during the recognition process. Extensive experiments were conducted using eight different datasets from UCI and KEEL

repository.

Finally, the thesis is concluded in Chapter 6 with suggested future work.

2. LITERATURE REVIEW

This chapter will provide overviews about the adaptive filters, their problems and development stages in general. Also, the artificial neural networks (ANN), its architecture, and their advantages and disadvantages will present.

2.1 Adaptive Filters

Performing common processing operations on a sequence of data can be considered as filtering the raw data (Canan et al., 1998). A digital filter with fixed coefficients can be designed by using well defined proporties. However, in some situations, where the specifications are not available or time-varying, a filter that updating the coefficients with time is required which is called as an adaptive filter. Adaptive filtering has shown a great interest by researchers during the last decades. This interest is due to those vast application areas of adaptive filters (Muneyasu et al., 1998; Haykin, 2002; Sayed, 2003; Salman, 2014; Gwadabe and Salman, 2015; Ma et al., 2015).

LMS algorithm is one of the well-known adaptive filtering techniques which has been used to solve many problems. However, the performance of an adaptive filter is usually controlled by some parameters that usually generate a trade-off between the convergence rate and SS-MSE. Since the LMS algorithm is a gradient descent based algorithm, a constant step-size parameter is used to update the filter coefficients. This step-size parameter has a critical effect on the performance of the algorithm. A large step-size value provides a fast convergence but a high MSE where a small step-size causes slow convergence with low MSE. This trade-off can be set in the favor of both increase in the convergence rate and decrease in misadjustment for the best performance by using a variable step-size (Aboulnasr and Mayyas, 1997; Turan and Salman, 2014).

The NLMS algorithm has been proposed to improve the performance of the traditional LMS algorithm (Haykin, 2002). The NLMS algorithm provides a fast convergence rate, and achieves a minimal steady-state error by normalizing the step-size by the power of the input. Shin et al., (2004) have developed a variable step-size affine projection algorithm to improve the convergence rate with lower misadjustment at early stages of adaptation.

The recursive-least-squares (RLS) algorithm has been offering a superior

performance in adaptive filtering, which outperforms the LMS algorithm and its versions (Maouche and Slock, 2000; Wang, 2009; Eksioglu and Tanc, 2011). The RLS algorithm can keep the output performance in the steady-state on improving over time. In the case of the least-squares (LS) algorithms, the performance is controlled by the forgetting factor with a value close to unity to obtain the convergence and stability of the algorithm at the same time.. A major drawback of the RLS algorithm is its high computational complexity (Haykin, 2002).

Ahmad et al., (2011) has proposed a new RI adaptive filtering algorithm that showed better performance compared to the transform domain LMS with variable step-size (TDVSS) in terms of the convergence rate, and it is very comparable to the RLS algorithm, with lower computational complexity than RLS algorithm by avoiding the inversion of the autocorrelation matrix. Ahmad et al., (2010b) used the second-order estimates of the correlations, the second-order RI algorithm has provided MSE performance that cannot be attained using the TDVSS algorithm. It has shown robust performance in impulsive noise and non-stationary environments (Ahmad et al., 2012; Ahmad et al., 2013b). However, the trade-off between the convergence rate and the MSE is not radically solved.

In the last decade, a convex combination of adaptive filtering algorithms has been frequently used to overcome this trade-off (Kozat and Singer, 2000; Arenas-Garcia et al., 2003; Arenas-Garcia et al., 2005; Arenas-Garcia et al., 2006; Azpicueta-Ruiz et al., 2008; Silva and Nascimento, 2008; Trump, 2009a; Trump, 2009b; Azpicueta-Ruiz et al., 2010). The main idea behind these proposals is to extract the fast convergence and low MSE properties of two adaptive filters and combine their performances. However, most of these proposals still converge to a relatively high MSE. The convex scheme consists of combining two adaptive filters. One possibility is depicted in Fig. 2.1. The output and error estimates of the adaptive filters are combined using the parameter $\lambda(n)$.

Mandic et al., (2007) proposed a collaborative adaptive learning algorithm using hybrid filters. They combined two different adaptive filters in order to attain lower MSE with high convergence rate. However, the combined performance of their proposed algorithm cannot exactly track the learning curves of both filters. Trump (2009b) discussed the tracking performance of a combination of two NLMS adaptive filters. In that paper, the combiner can track the learning curves of the combined filters.

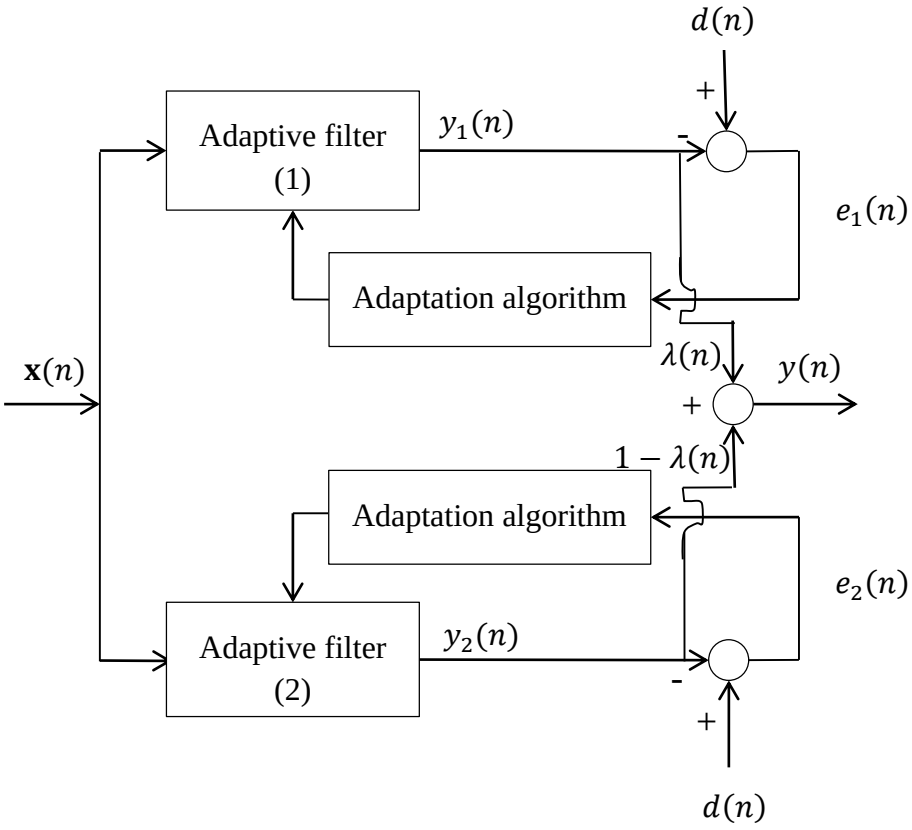


Figure 2.1. The block diagram of the convex combination of adaptive filters

Salman et al., (2015) proposed a new convex combination of two different algorithms as zero-attracting leaky least-mean-square (ZA-LLMS) and reweighted zero-attracting leaky-least-mean square (RZA-LLMS) algorithms in sparse system identification setting. The performances of the aforementioned algorithms has been tested and compared to the result of the new combination. It showed that the proposed algorithm has a good ability to track the MSD curves of the other algorithms under a various noise environments.

A 2-Dimensional (2-D) adaptive filters is an extension of 1-D adaptive filters which deals with two dimensional signals (i.e., images) (Hadhoud and Thomas, 1998). One of the most attracting applications of such algorithms is image de-noising. There are many 2-D adaptive algorithms such as: OBA, OBAI, TDBLMS, TDLMS, and TDBLMS (Mikhael and Wu, 1989; Mikhael and Ghosh, 1992; Hadhoud and Thomas, 1998; Wang and Wang, 1998). Two-dimensional least-mean-square (TDLMS) updates its coefficients in horizontal and vertical directions on a 2-D plane. Even though the 2-D LMS algorithm has shown good performance in image de-noising systems, its

performance still poor when the signal-to-noise ratio (SNR) is relatively low.

Magnetic resonance image (MRI) de-noising is one of the most interesting applications in adaptive filters techniques, especially, if the SNR is relatively low (Phatak et al., 2011) or the noise model is complicated. For instance, if the MRI data is corrupted by Rician noise from the measurement process, this will reduce the accuracy of any automatic analysis (Mustafa et al., 2012). The challenge, here, is to remove such noise by segmentation, classification and registration (Chang et al., 2011). There are many applications to denoising in the MRI such as, adaptive multi-scale data condensation (Ray et al., 2012) total variation and local noise estimation (Varghees et al., 2012) and adaptive non-local means filtering (Kang et al., 2013).

However, these algorithms usually suffer from poor performance or time consumption. Hameed et al., (2014) proposed a 2-D version convex combination of recursive inverse algorithms (RI) algorithm that provides fast convergence at the beginning to save time and then provides high performance in terms of noise removal. The de-noising experiment has been conducted on MR image that is assumed to be corrupted by an additive white Gaussian noise (AWGN). Simulations showed that the proposed algorithm successfully recovers the image.

2.2 Artificial Neural Network

Backpropagation (BP) neural networks, and Self-organizing map (SOM) are common neural network algorithms, they provide high performances in solve many different problems in a wide range of applications as shown in this literature.

2.2.1 Backpropagation (BP) algorithm

BP artificial neural networks algorithm is an interesting topic which has attracted many researchers in various domains (Karlik et al., 1998; Karlik, 2000; Lee et al., 2005; Chiang et al., 2006; Dvir et al., 2006; Behrman et al., 2007; Iseri and Karlik, 2009; Karlik and Cemel, 2012; Karlik, 2013). However, it has a couple of obstacles that usually restrict its algorithm performance; the slow convergence and the high steady-state error. In order to speed up the convergence rate and decrease the steepest descent error, the momentum technique has been proposed (Yu et al., 1993; Yu and Liu, 2002; Shao and Zheng, 2009). Many versions of momentum have been proposed and trained

in different applications (Qiu et al., 1992; Istook and Martinez, 2002). The BP with fixed momentum (BPFM) has been proposed by adding a momentum term to the conventional BP weight update equation (Rumelhart and McClelland, 1986; Rumelhart et al., 1988) which can speed up learning by reaching optimal steady-state error in short time. Whereas, the fixed momentum is supposed to be constant within the range of [0,1]. Unfortunately, when the fixed momentum is in the direction of the negative gradient which is in an opposing direction to the previous update, it may cause the weight to be arranged the slope of the error surface instead of down the slope as target. The fixed momentum has also been extended to adaptive momentum by superiorly adapting itself along with iterations, achieving optimal convergence speed in the BP implementation. The adaptive momentum can update itself iteratively and automatically during the iteration process and depending on the prediction of the output error (Wu et al., 2002; Wu and Xu, 2002).

Yu and Liu (2002) proposed a new acceleration technique of BP algorithm by deriving adaptive learning rate and momentum coefficient. The proposed adaptive parameters are adjusted iteratively to access the optimum weight vectors in a short time of process. Where, the new BP proved superior convergence compared to other competing methods.

Shao and Zheng (2009,2011) have proposed a back-propagation with adaptive momentum (BPAM) which provides better performance than the conventional BP algorithm in terms of both convergence rate (decreasing time of convergence) and SSE. In their study, the momentum is updating itself iteratively by multiplying the current weights with the previous weights. If the momentum coecients are less than zero, they are denied as positive value to accelerate learning by updating momentum. Otherwise the momentum is considered as zero to maintain the error downhill.

2.2.2 Self-organizing map (SOM) algorithm

SOM which is most common unsupervised algorithm used in artificial neural networks has been proposed by (Kohonen, 1990). SOM used to solve many problems of different applications (Kaski et al., 1998; Tamukoh et al., 2004). It maps the input data from high dimensional input vector to low dimensional maps according to the features relations between the input data. The algorithm competitive layer used to learn clustering the input vectors. SOM algorithm is suffers from the time consumption of

training. Training time depends on dataset size and whether the algorithm can reach the optimum weights and the topology preservation.

Many versions of the SOM have been proposed to improve the vector quantization and the topology preservation performances (Chi and Yang, 2006; Wong et al., 2006; Brugger et al., 2008; Chi and Yang, 2008; Gorgonio and Costa, 2008; Yen and Wu, 2008; Cottrell et al., 2009; Arous and Ellouze, 2010; Tasdemir et al., 2011; Appiah et al., 2012; Ayadi et al., 2012; Yang et al., 2012). In (Brugger et al., 2008), authors proposed a new way that can detect the cluster automatically by applying the cluster algorithm (Bogdan and Rosenstiel, 2001) to the SOM.

Chi and Yang (2006) proposed a hybrid clustering by combining SOM, and the ant K-means algorithm which is a meta-heuristic method that has been used to investigate the optimum solutions of many problems. So, the ant K-means algorithm used to guide the K-means algorithm to the place the objects with high probabilities depending on the characteristic of pheromone updating. The SOM and ant K-means algorithm has high performance compared to the conventional SOM algorithm, and some clustering methods.

Another improvement proposed by (Tasdemir et al., 2011), which is demonstrated that the neurons of SOM can be clustered hierarchically depending on the density without needing the distance dissimilarity. In addition, the Gaussian-function with self-organizing map (GF-SOM) algorithm has been widely used in many applications as a neighborhood function.

To overcome the limitation of conventional SOM, Berglund & Sitte proposed a new SOM called the Parameter-Less self-organizing map (PLSOM) algorithm (Berglund and Sitte, 2006). Their algorithm calculates the local quadratic fitting error instead of using the well-known neighbourhood size and learning parameters as in the conventional SOM algorithm. The problem with PLSOM algorithm is in the initial weight distribution which suffers from over-reliance and over-sensitivity to outliers. This suffering continues even after a period of processing time (Berglund, 2010).

Yamaguchi et al., (2010) proposed an adaptive hierarchical competitive network layer of SOM algorithm based on Tree-Structured. The proposed algorithm adapts automatically to detect the optimum number of neurons, by adding or deleting the map neurons using means error and frequency techniques.

3. ADAPTIVE FILTERING

Adaptive filtering has been implemented to overcome the limitations of the conventional static filters, where the adaptive filter can deal with unknown or time varying input signals in a various noise environments (Diniz, 1997). They have been used in different applications such as: system identification (Glentis et al., 1999), channel equalization/identification and interference suppression in communications systems (Madhow and Honig, 1994; Gesbert and Duhamel, 2000), and acoustic echo cancellation (Gay and Benesty, 2000). The main idea of filter adaptation is the output signals of the filter depends on the weight coefficient vectors, that adjust itself iteratively with time processing to minimize the error of estimation between filter output and desired output. Fig. 3.1 illustrates the main diagram of an adaptive filter. It is shown in the figure how to remove noise using adaptive filters, where $y(n)$ is the filter output, $d(n)$ is the desired response and $e(n)$ is the estimation error of the adaptive filter for time iteration n.

Adaptive filters classified as; finite impulse response (FIR), and infinite impulse response (IIR). IIR filters are beyond the scope of this thesis. The output of adaptive FIR filter is obtained linearly by combining the present and past input signal samples $N-1$, where N is the number of the filter coefficients. Adaptive FIR filter is preferred over adaptive IIR filter because of its robustness and simplicity. Moreover, FIR filters have been used in many practical applications such as: channel estimation in communications systems (Breining et al., 1999).

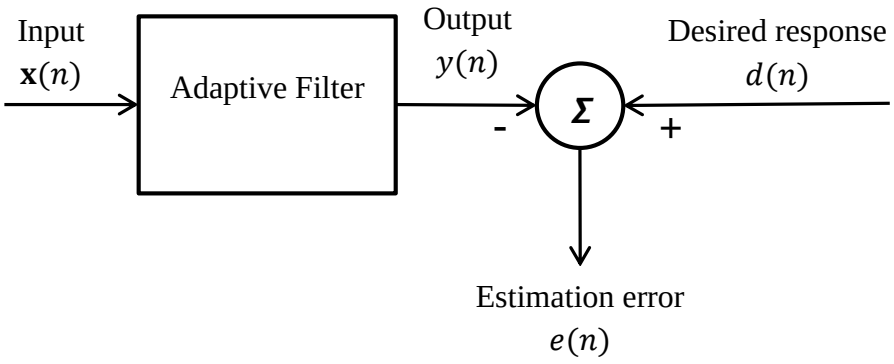


Figure 3.1. Block diagram of the statistical filtering problem

3.1 Adaptive Filtering Configurations

Two adaptive filtering configurations are presented in this thesis. One application is system identification and the other is noise cancellation. Different adaptive algorithms will be tested using this applications in additive white Gaussian noise (AWGN) and additive correlated Gaussian noise (ACGN) environments.

3.1.1 System identification

This configuration has been used in several areas. The system identification is necessary for many applications such as: identification of the acoustic echo path in acoustic echo cancellation (Dogancay and Tanrikulu, 2001), channel identification in communications systems (Gesbert and Duhamel, 2000). The adaptive filter is capable to obtain a best fit of a linear model for an unknown system with a time varying model. The unknown system and adaptive filter are fed simultaneously by the same input signal, and the output of the unknown system added to measurement noise $v(n)$ to provide the desired response signal of the adaptive filter $d(n)$, where $d(n) = \boldsymbol{h}^T\mathbf{x}(n) + v(n), \boldsymbol{h}$ is the optimal filter coefficient vector (unknown system). Fig. 3.2 depicts the system identification configuration.

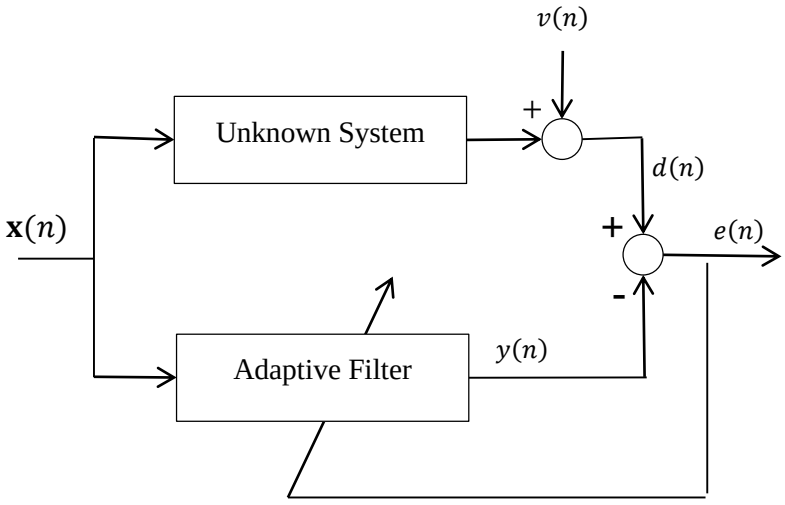


Figure 3.2. System identification configuration

where $\mathbf{x}(n)$, $v(n)$, $y(n)$, and $e(n)$, are the input signal, the measurement noise, adaptive filter output signal and the irreducible error, respectively.

3.1.2 Noise cancellation

This configuration is used to eliminate noise by passing the received signal through the configuration using adaptive algorithm. Static filters need to have prior knowledge about the characteristics of the input signal or noise to estimate the desired information. While, the adaptive filters do not require prior knowledge about input signal. Noise cancellation is applied in radio communications and mobile phones, because those are affected from high-noise signals. The adaptive noise cancellation is depicted in Fig. 3.3.

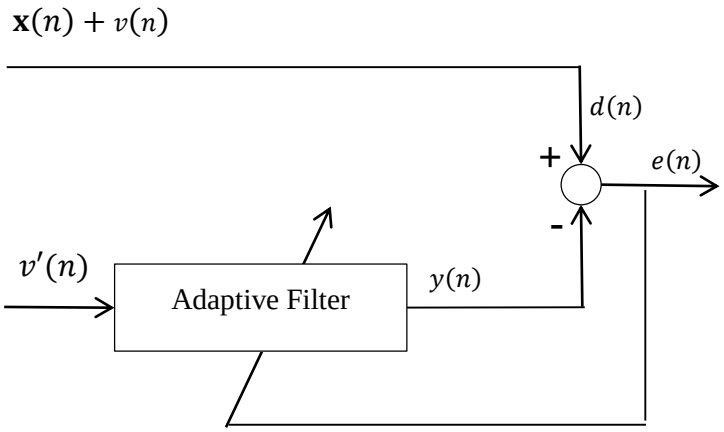


Figure 3.3. Noise cancellation configuration

The desired response $d(n)$ includes the input $\mathbf{x}(n)$ and an uncorrelated noise $v(n)$. A second input $v'(n)$ used as a noise to feed the adaptive filter which is uncorrelated with $v(n)$ and independent of $\mathbf{x}(n)$ so that it can extract the desired information. The filter coefficients of adaptive algorithm $\mathbf{w}(n)$ adjust themselves automatically to reduce the error $e(n)$ between $y(n)$ and $d(n)$, and obtain the desired signal.

3.2 Adaptation Algorithms

In this section, some adaptive filters are reviewed. The first algorithm is the normalized least-mean-square (NLMS) proposed in (Haykin, 2002), the second is the recursive inverse (RI) by (Ahmad et al., 2011), and the last one is the second-order recursive inverse (second-order RI) by (Ahmad et al., 2013b). These algorithms have been used in diverse adaptive filtering applications.

3.2.1 Normalized least-mean-square (NLMS) algorithm

The least-mean-square (LMS) algorithm is a widely used adaptive algorithm (Bouboulis and Theodoridis, 2010). It is characterized by its simplicity, robustness and low cost (Haykin, 2002). This adaptive algorithm is based on the gradient descent method of the cost function $\left(J(n) = e^2(n)\right)$. The weight update equation of LMS algorithm is derived as,

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu\mathbf{x}(n)e(n) \tag{3.1}$$

Where μ is the step-size, that controls the convergence rate and the stability of the algorithm.

The LMS algorithm adjusts the tap weights vector in a recursive manner until obtaining the optimum weights vector to access minimum error on the required signal using (3.1). The step size is constant in the range of,

$$0 < \mu < \frac{2}{\lambda_{max}} \tag{3.2}$$

Where λ_{max} is the input autocorrelation matrix $\mathbf{R}$ with largest eigenvalue, used to guarantee the stability. The trace of $\mathbf{R}$ (sum of the eigenvalues) is used instead of λ_{max}. Therefore, the value of step-size is within $0 < \mu < \frac{2}{trace\,(\mathbf{R})}$. The trace $(\mathbf{R}) = \|\mathbf{x}(n)\|^2$ is related to the power of the $\mathbf{x}(n)$. The well-known step size is obtained as;

$$0 < \mu < \frac{2}{\|\mathbf{x}(n)\|^2} \tag{3.3}$$

The step-size μ is inversely proportional to the power of the input signal. Accordingly, when the power of the input is high, the step size is imposed to a small value, on the other hand, when the power of the input is low the step size becomes large. This relationship enables normalizing the step-size of the LMS algorithm according to the input signal power. The normalized step-size provides a useful LMS-type algorithm, commonly known as normalized LMS (NLMS) algorithm (Haykin, 2002).

The NLMS algorithm with normalized step-size term updates the weights vector as,

$$\mathbf{w}(n) = \mathbf{w}(\mathbf{n}-\mathbf{1}) + \frac{\mu}{\mathbf{x}^T(n)\mathbf{x}(n) + \epsilon}\mathbf{x}(n)e(n) \tag{3.4}$$

Where the step-size is in the range [0,2]. The importance of normalizing the step size is improving the convergence behavior in the NLMS algorithm. So the algorithm becomes powerful in non-stationary applications like speech recognition. In addition, the speed of convergence is improved to achieve the minimum steady-state MSE quickly (Azpicueta-Ruiz et al., 2010).

3.2.2 Recursive inverse (RI) algorithm

Any stationary discrete-time stochastic process can be expressed as:

$$\mathbf{x}(n) = u(n) + v(n) \tag{3.5}$$

Where $u(n)$ is the desired signal and $v(n)$ is the noise process. Removing noise from $\mathbf{x}(n)$ is a challenge in many signal-processing applications.

Many adaptive algorithms have been used to update the coefficients of the filter shown in Fig. 3.1. In the recently proposed adaptive RI algorithm, the autocorrelation matrix is recursively estimated and not its inverse. The weight-updated equation of the RI is:

$$\mathbf{w}(n) = [\mathbf{I} - \mu(n)\mathbf{R}(n)]\mathbf{w}(n-1) + \mu(n)\mathbf{p}(n) \tag{3.6}$$

Where n is the time parameter $(n = 1,2, \dots)$, $\mathbf{w}(n)$ is the filter weight vector at time n with length N , $\mathbf{I}$ is an $N \times N$ identity matrix, $\mu(n)$ is the variable step size, $\mathbf{R}(n)$ is the autocorrelation matrix of the tap-input vector, and $\mathbf{p}(n)$ is the cross-correlation vector between the tap-input vector and desired response of the adaptive filter. The correlations of the tap-input vector and the desired response are recursively estimated as:

$$\mathbf{R}(n) = \beta\mathbf{R}(n-1) + \mathbf{x}(n)\mathbf{x}^T(n) \tag{3.7}$$

$$\mathbf{p}(n) = \beta\mathbf{p}(n-1) + d(n)\mathbf{x}(n) \tag{3.8}$$

Where β is the forgetting factor, which is usually selected very close to unity, and the step size μ(n) is given as:

$$\mu(n) = \frac{\mu_0}{1-\beta^n} \; where \; \mu_0 < \mu_{max} \tag{3.9}$$

Where $\mu_{max} < \frac{2}{\lambda_{max}(\mathbf{R}(n))}$ and $\lambda_{max}\big(\mathbf{R}(n)\big)$ is the maximum eigenvalue of $\mathbf{R}(n)$.

3.2.3 Second order recursive inverse (second-order RI) algorithm

In order to improve the performance of the RI algorithm, a second-order RI estimate of the correlations with the same updated equation as in (3.6) can be used:

$$\mathbf{R}(n) = \beta_1\mathbf{R}(n-1) + \beta_2\mathbf{R}(n-2) + \mathbf{x}(n)\mathbf{x}^{\mathbf{T}}(n) \tag{3.10}$$

$$\mathbf{p}(n) = \beta_1\mathbf{p}(n-1) + \beta_2\mathbf{p}(n-2) + d(n)\mathbf{x}(n) \tag{3.11}$$

By selecting $\beta_1 = \beta_2 = \frac{1}{2}\beta$, the computational complexity of the second-order RI will be comparable to that of the first-order RI algorithm. Taking the expectation of Eq. (3.10) gives:

$$\bar{\mathbf{R}}(n) = \frac{1}{2}\beta_1 \bar{\mathbf{R}}(n-1) + \frac{1}{2}\beta_2 \bar{\mathbf{R}}(n-2) + \mathbf{R_{xx}} \tag{3.12}$$

Where $\mathbf{R_{xx}} = E\{\mathbf{x}(n)\mathbf{x}^T(n)\}$ and $\bar{\mathbf{R}}(n) = E\{\mathbf{R}(n)\}$. After calculating the transfer function of Eq. (3.12), its poles are:

$$\begin{aligned} z_1 &= \frac{1}{4}\left(\beta - \sqrt{\beta^2 + 8\beta}\right) \\ z_2 &= \frac{1}{4}\left(\beta + \sqrt{\beta^2 + 8\beta}\right) \end{aligned} \tag{3.13}$$

Where z_1 and z_2 have magnitudes of less than unity if $\beta < 1$. Solving Eq. (3.12) with the initial conditions $\bar{\mathbf{R}}(-2) = \bar{\mathbf{R}}(-1) = \bar{\mathbf{R}}(0) = 0$ yields:

$$\bar{\mathbf{R}}(n) = \left(\frac{1}{\beta - 1} + \alpha_1 z_1^k + \alpha_2 z_2^k\right)\mathbf{R_{xx}} \tag{3.14}$$

Where

$$\begin{aligned} \alpha_1 &= \frac{\beta - z_2}{(1-\beta)(z_2 - z_1)} \\ \alpha_2 &= \frac{\beta - z_1}{(1-\beta)(z_2 - z_1)} \end{aligned} \tag{3.15}$$

Letting

$$\gamma(n) = \frac{1}{\beta - 1} + \alpha_1 z_1^n + \alpha_2 z_2^n \tag{3.16}$$

The variable step size of the second-order RI algorithm is then:

$$\mu(n) = \frac{\mu_0}{\gamma(n)} \tag{3.17}$$

Where μ_0 and $\gamma(n)$ are defined in Eqs. (3.9) and (3.16), respectively. This variable step size enables us to reach a low MSE that cannot to be attained by the NLMS or the first-order RI algorithm.

3.3 Convex Combination

In this section, different RI convex combinations such as convex combination of RI algorithms, convex combination of RI and second-order RI algorithms, and 2-D convex of RI and second-order RI algorithms have been proposed to improve the performance of the adaptive filters.

3.3.1 Convex combination of RI algorithms

Consider the combination of two adaptive filters in noise cancelation setting which has been recently proposed in (Ma et al., 2015), as shown in Fig.3.4. Starting by the update equation of the RI.

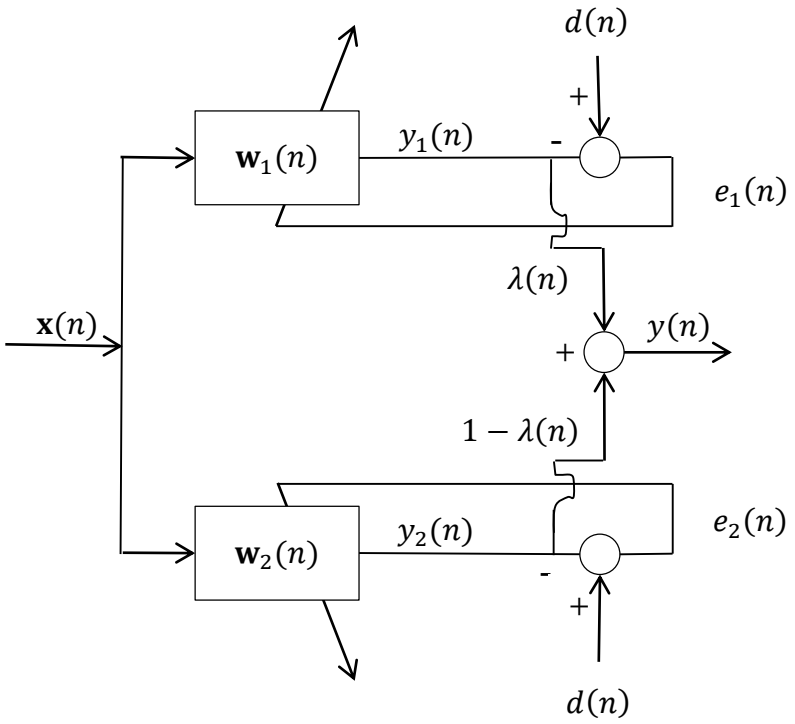


Figure 3.4. Convex combination of two adaptive filters for a noise cancelation setting

$$\mathbf{w}_i(n) = [\mathbf{I} - \mu_i(n)\mathbf{R}_i(n)]\mathbf{w}_i(n-1) + \mu_i(n)\mathbf{p}_i(n) \tag{3.18}$$

Where n is the time index ($n = 1,2,\dots$), $\mathbf{w}_i(n)$ is the i^{th} filter weight vector at time n with length N ($i = 1,2$), $\mathbf{I}$ is an $N \times N$ identity matrix, $\mu_i(n)$ is the i^{th} variable step-size, $\mathbf{R}_i(n)$ is the i^{th} autocorrelation matrix of the tap-input vector and $\mathbf{p}_i(n)$ is the i^{th} cross-correlation vector between the tap-input vector $\mathbf{x}(n)$ and desired response $d(n)$ of the adaptive filter. The correlations are recursively estimated as the following,

$$\mathbf{R}_i(n) = \beta\mathbf{R}_i(n-1) + \mathbf{x}(n)\mathbf{x}^T(n) \tag{3.19}$$

$$\mathbf{p}_i(n) = \beta\mathbf{p}_i(n-1) + d(n)\mathbf{x}(n) \tag{3.20}$$

Where β is the forgetting factor with a value close to unity.

$$\mu_i(n) = \frac{\mu_0}{1-\beta^n} \; where \; \mu_0 < \mu_{max} \tag{3.21}$$

Where $\mu_{max} < \frac{2}{\lambda_{max}(\mathbf{R}_i(n))}$ and $\lambda_{max}\big(\mathbf{R}_i(n)\big)$ is the maximum eigenvalue of $\mathbf{R}_i(n)$.

The error of each individual filter is formulated as

$$e_i(n) = d(n) - \mathbf{w}_i^T(n-1)\mathbf{x}(n) \tag{3.22}$$

And the desired response is;

$$d(n) = \mathbf{x}(n) + v(n) \tag{3.23}$$

Where $v(n)$ is the measurement noise. The outputs of the two adaptive filters can be combined according to (Arenas-Garcia et al., 2006; Azpicueta-Ruiz et al., 2008), by the following equation,

$$y(n) = \lambda(n)y_1(n) + [1-\lambda(n)]y_2(n) \tag{3.24}$$

where $y_i(n) = \mathbf{w}_i^T(n-1)\mathbf{x}(n)$ and convex combination parameter, $\lambda(n)$ is given by,

$$\lambda(n) = \frac{E[(d(n) - y_2(n))(y_1(n) - y_2(n))]}{E[(y_1(n) - y_2(n))^2]} \tag{3.25}$$

The error signal of the above mentioned combination is given as,

$$\begin{aligned} e(n) &= d(n) - y(n) \\ &= d(n) - \lambda(n)y_1(n) - \big(1 - \lambda(n)\big)y_2(n) \end{aligned} \tag{3.26}$$

3.3.1.1 Tracking analysis of convexly combined RI algorithms

In this section, the tracking analysis of the proposed algorithm is presented and SS-MSE criterion is derived.

Let us start by the random walk model.

$$\mathbf{w}_0(n) = \mathbf{w}_0(n-1) + q(n) \tag{3.27}$$

Where $q(n)$ is a stochastic i.i.d. with zero mean and covariance matrix $Q = E\{q(n)q(n)\}$. The weight error vector of i^{th} filter is defined as:

$$\widetilde{\mathbf{w}}_i(n) = \mathbf{w}_0(n) - \mathbf{w}_i(n) \tag{3.28}$$

The a priori error is defined as,

$$e_{i,a}(n) = \mathbf{x}^T(n)[\mathbf{w}_0(n) - \mathbf{w}_i(n-1)] \tag{3.29}$$

And the a posteriori error,

$$e_{i,p}(n) = \mathbf{x}^T(n)[\mathbf{w}_0(n) - \mathbf{w}_i(n)] \tag{3.30}$$

Now, we calculate the overall output error by subtracting the overall output of

filters in (3.24) form the desired response and by using (3.29) and (3.30),

$$\begin{aligned} e_a(n) &= d(n) - \lambda(n)y_1(n) - \big(1-\lambda(n)\big)y_2(n) \\ &= d(n) - \lambda(n)y_1(n) - y_2(n) - \lambda(n)y_2(n) \\ &= e_{2,a}(n) - \lambda(n)[-d(n) + y_1(n) + d(n) - y_2(n)] \\ &= e_{2,a}(n) - \lambda(n)[-e_{1,a}(n) + e_{2,a}(n)] \\ &= e_{2,a}(n) + \lambda(n)e_{1,a}(n) - \lambda(n)e_{2,a}(n) \\ &= \big(1-\lambda(n)\big)e_{2,a}(n) + \lambda(n)e_{1,a}(n) \end{aligned} \tag{3.31}$$

Evaluate $E[e_a^2(n)]$ using (3.31),

$$\begin{aligned} E[e_a^2(n)] &= E\left[\Big(\big(1-\lambda(n)\big)e_{2,a}(n) + \lambda(n)e_{1,a}(n)\Big)\Big(\big(1-\lambda(n)\big)e_{2,a}(n) + \lambda(n)e_{1,a}(n)\Big)\right] \\ &= \big(1-\lambda(n)\big)^2 E\big[e_{2,a}^2(n)\big] + 2\lambda(n)\big(1-\lambda(n)\big)E\big[e_{1,a}(n)e_{2,a}(n)\big] \\ &\quad +\lambda^2(n)E\big[e_{2,a}^2(n)\big] \end{aligned} \tag{3.32}$$

To evaluate $E[e_a^2(n)]$,we first need to evaluate the cross terms in (3.32),

$$\begin{aligned} &E\big[e_{1,a}(n)e_{2,a}(n)\big] \\ &\quad = E[(\mathbf{w}_0(n) - \mathbf{w}_1(n-1))^T\mathbf{x}(n)\mathbf{x}^T(n)(\mathbf{w}_0(n) - \mathbf{w}_2(n-1))] \end{aligned} \tag{3.33}$$

Subtracting both sides of (3.18) from $\mathbf{w}_0(n)$and by using (3.29) and (3.30) we get,

$$\widetilde{\mathbf{w}}_i(n) = \widetilde{\mathbf{w}}_i(n-1) - \mu_i(n)\mathbf{x}(n)e_i(n) + \mu_i(n)\beta_i\xi_i(n-1) \tag{3.34}$$

Where $\xi_i(n-1) = \mathbf{R}(n)\mathbf{w}_i(n-1) - \mathbf{p}(n)$. Multiplying both sides of (3.34) by $\mathbf{x}^T(n)$ from the left side gives,

$$e_{i,p}(n) = e_{i,a}(n) - \mu_i(n)\mathbf{x}^T(n)\mathbf{x}(n)e_i(n) + \mu_i(n)\beta_i\mathbf{x}^T(n)\xi_i(n-1) \tag{3.35}$$

Substituting (3.35) in (3.34) yields:

$$\widetilde{\mathbf{w}}_i(n) = \widetilde{\mathbf{w}}_i(n-1) - \mu_i(n)\beta_i\xi_i(n-1) + \frac{\mu_i(n)\mathbf{x}(n)}{\mu_i(n)\mathbf{x}^T(n)\mathbf{x}(n)}\left[e_{i,a}(n) - e_{i,p}(n) + \mu_i(n)\beta_i\mathbf{x}^T(n)\xi_i(n-1)\right] \quad (3.36)$$

Note that $tr\{\mathbf{x}(n)\mathbf{x}^T(n)\xi_i(n-1)\} = tr\{\xi_i(n-1)\mathbf{x}^T(n)\mathbf{x}(n)\}$ (Haykin, 2002) and hence $\mathbf{x}(n)\mathbf{x}^T(n)\xi_i(n-1) \approx \xi_i(n-1)\mathbf{x}^T(n)\mathbf{x}(n)$. Rearranging (3.36) provides;

$$\widetilde{\mathbf{w}}_i(n) - \frac{\mathbf{x}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)}e_{i,a}(n) = \widetilde{\mathbf{w}}_i(n-1) - \frac{\mathbf{x}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)}e_{i,p}(n) \quad (3.37)$$

Multiplying both side of the first filter in (3.37) by their counterpart of the second filter yields:

$$\widetilde{\mathbf{w}}_1^T(n)\widetilde{\mathbf{w}}_2(n) + \frac{e_{1,a}(n)e_{2,a}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)} = \widetilde{\mathbf{w}}_1^T(n-1)\widetilde{\mathbf{w}}_2(n-1) + \frac{e_{1,p}(n)e_{2,p}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)} \quad (3.38)$$

Applying the random walk model is to derive the following expression for the expectation of the inner product of the weight error vectors of the two individual filters at the time instant n gives,

$$\begin{aligned}
&E[(\mathbf{w}_0(n) - \mathbf{w}_1(n-1))^T(\mathbf{w}_0(n) - \mathbf{w}_2(n-1)) \\
&\quad = E\big[\mathbf{w}_0(n-1) + q(n) - \mathbf{w}_1(n-1))^T\big(\mathbf{w}_0(n-1) + q(n) \\
&\quad - \mathbf{w}_2(n-1)\big)\big] = E\big[\widetilde{\mathbf{w}}_1(n-1) + q(n))^T\big(\widetilde{\mathbf{w}}_2(n-1) + q(n)\big)\big] \\
&\quad = E[\widetilde{\mathbf{w}}_1^T(n-1)\widetilde{\mathbf{w}}_2(n-1)] + E[\widetilde{\mathbf{w}}_1^T(n-1)q(n)] \\
&\quad + E[q^T(n)\widetilde{\mathbf{w}}_2(n-1)] + E[q(n)q(n)] \\
&\quad = E[\widetilde{\mathbf{w}}_1^T(n-1)\widetilde{\mathbf{w}}_2(n-1)] + Tr\{Q\} \quad (3.39)
\end{aligned}$$

Substituting (3.39) into (3.38) and simplifying gives,

$$\begin{aligned}
&E[\widetilde{\mathbf{w}}_1^T(n)\widetilde{\mathbf{w}}_2(n)] + E\left[\frac{e_{1,a}(n)e_{2,a}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)}\right] \\
&\quad = E[\widetilde{\mathbf{w}}_1^T(n-1)\widetilde{\mathbf{w}}_2(n-1)] + E\left[\frac{e_{1,p}(n)e_{2,p}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)}\right] + Tr\{Q\} \quad (3.40)
\end{aligned}$$

In the steady state assume,

$$E[\tilde{\mathbf{w}}_1^T(n)\tilde{\mathbf{w}}_2(n)] \approx E[\tilde{\mathbf{w}}_1^T(n-1)\tilde{\mathbf{w}}_2(n-1)] \tag{3.41}$$

And hence,

$$E\left[\frac{e_{1,a}(n)e_{2,a}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)}\right] = E\left[\frac{e_{1,p}(n)e_{2,p}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)}\right] + Tr\{Q\} \tag{3.42}$$

Now, if we substitute (3.35) in (3.42) and rearrange we obtain,

$$E\left[e_{1,a}(n)\mu_2(n)e_2(n)\right] + E\left[e_{2,a}(n)\mu_1(n)e_1(n)\right] = E[\mu_1(n)\mu_2(n)\mathbf{x}^T(n)\mathbf{x}(n)e_1(n)e_2(n)] + Tr\{Q\} \tag{3.43}$$

Now substitute $e_i(n) = e_{i,a}(n) + \nu(n)$ in (3.43), having in mind $\mathbf{x}^T(n)\mathbf{x}(n) \approx E\{\mathbf{x}^T(n)\mathbf{x}(n)\} = \sigma_{\mathbf{x}}^2$ and $E[e_{i,a}(n)\,\nu(n)] = 0$ and simplifying,

$$\mu_2(n)E\left[e_{1,a}(n)\left(e_{2,a}(n)+\nu(n)\right)\right] + \mu_1(n)E\left[e_{2,a}(n)\left(e_{1,a}(n)+\nu(n)\right)\right] = \mu_1(n)\mu_2(n)E\left[\mathbf{x}^T(n)\mathbf{x}(n)\left(e_{1,a}(n)+\nu(n)\right)\left(e_{2,a}(n)+\nu(n)\right)\right] + Tr\{Q\} \tag{3.44}$$

Substituting (3.44) in (3.33) and taking $\lim_{n\to\infty}$ gives,

$$lim_{n\to\infty}E\left[e_{1,a}(n)e_{2,a}(n)\right] = z(n)[\mu_1(n)\mu_2(n)\sigma_v^2E[\mathbf{x}^T(n)\mathbf{x}(n)] + Tr\{Q\}] \tag{3.45}$$

Where $z(n) = \frac{1}{\mu_1(n)+\mu_2(n)-\mu_1(n)\mu_2(n)E[\mathbf{x}^T(n)\mathbf{x}(n)]}$

For a single filter case, we have,

$$lim_{n\to\infty}E\left[e_{i,a}(n)e_{i,a}(n)\right] = \frac{1}{2\mu_i(n) - \mu_i^2(n)E[\mathbf{x}^T(n)\mathbf{x}(n)]}[\mu_i^2(n)\sigma_v^2E[\mathbf{x}^T(n)\mathbf{x}(n)] + Tr\{Q\}] \tag{3.46}$$

Substituting (3.45) and (3.46) in (3.32) gives:

$$
\begin{aligned}
&lim_{n\to\infty}E[e_a^2(n)] \\
&= \frac{2\lambda(\infty)(1-\lambda(\infty))}{\mu_1(n)+\mu_2(n)-\mu_1(n)\mu_2(n)E[\mathbf{x}^T(n)\mathbf{x}(n)]}[\mu_1(n)\mu_2(n)\sigma_v^2E[\mathbf{x}^T(n)\mathbf{x}(n)]+Tr\{Q\}] \\
&\quad+\frac{\lambda^2(\infty)}{2\mu_1(n)-\mu_1^2(n)E[\mathbf{x}^T(n)\mathbf{x}(n)]}[\mu_1^2(n)\sigma_v^2E[\mathbf{x}^T(n)\mathbf{x}(n)]+Tr\{Q\}] \\
&\quad+\frac{(1-\lambda(\infty))^2}{2\mu_2(n)-\mu_2^2(n)E[\mathbf{x}^T(n)\mathbf{x}(n)]}[\mu_2^2(n)\sigma_v^2E[\mathbf{x}^T(n)\mathbf{x}(n)] \\
&\quad+Tr\{Q\}] \qquad (3.47)
\end{aligned}
$$

3.3.2 Convex combination of RI and second-order RI algorithms

To obtain better combination than in section (3.3.1), we combine RI and second-order RI algorithms, where the update equation same as in (3.18). The $\mu_i(n)$ is the i^{th} variable step-size of (3.9) and (3.17). $\mathbf{R}_i(n)$ is the i^{th} autocorrelation matrix and $\mathbf{p}_i(n)$ is the i^{th} cross-correlation of RI and second order RI algorithms. The correlations of the tap-input vector and the desired response are recursively estimated by the following,

$$\mathbf{R}_1(n) = \beta\mathbf{R}_1(n-1) + \mathbf{x}(n)\mathbf{x}^T(n) \qquad (3.48)$$

$$\mathbf{p}_1(n) = \beta\mathbf{p}_1(n-1) + d(n)\mathbf{x}(n) \qquad (3.49)$$

$$\mathbf{R}_2(n) = \beta_1\mathbf{R}_{2,}(n-1) + \beta_2\mathbf{R}_2(n-2) + \mathbf{x}(n)\mathbf{x}^T(n) \qquad (3.50)$$

$$\mathbf{p}_2(n) = \beta_1\mathbf{p}_2(n-1) + \beta_2\mathbf{p}_2(n-2) + d(n)\mathbf{x}(n) \qquad (3.51)$$

$$\mu_1(n) = \frac{\mu_0}{1-\beta^n} \text{ where } \mu_0 < \mu_{max} \qquad (3.52)$$

Where $\mu_{max} < \frac{2}{\lambda_{max}(\mathbf{R}_1(n))}$ and $\lambda_{max}(\mathbf{R}_1(n))$ is the maximum eigenvalue of $\mathbf{R}_1(n)$.

On the other hand, the variable step-size $\mu_2(n)$ of the second-order RI is:

$$\mu_2(n) = \frac{\mu_0}{\gamma(n)} \tag{3.53}$$

Where μ_0 and $\gamma(n)$ are defined in section (3.2).

The RI and second-order RI Algorithms are combined in the noise cancellation setting, as shown in Fig. 3.5.

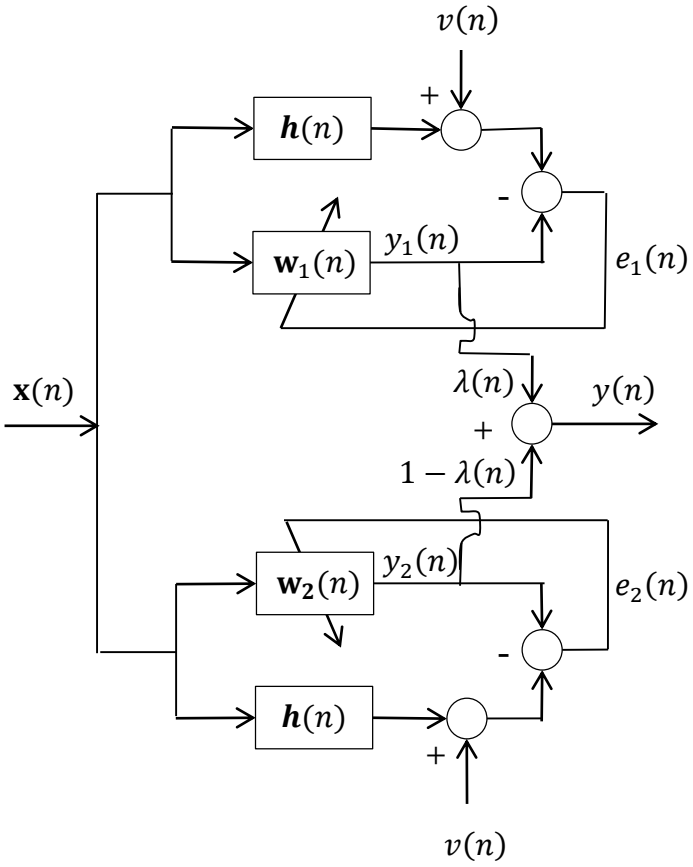


Figure 3.5. Convex combination of two adaptive filters for system identification setting

The output estimated error of the i^{th} adaptive filter of each combined algorithms shows as,

$$e_i(n) = d(n) - \mathbf{w}_i^T(n-1)\mathbf{x}(n) \tag{3.54}$$

$$d(n) = \boldsymbol{h}^T\mathbf{x}(n) + v(n) \tag{3.55}$$

And

$$y_i(n) = \mathbf{w}_i^T(n-1)\mathbf{x}(n) \tag{3.56}$$

Where $\mathbf{w}_i^T(n)$ is the tap weight vector, $y_i(n)$ is the output of the i^{th} adaptive filter with $i = 1,2$, $\boldsymbol{h}$ is the optimal filter coefficient, and $v(n)$ is the measurement noise.

The overall output filter combination can be computed similar to (3.24),

$$y(n) = \lambda(n)\big(\mathbf{w}_1^T(n-1)\mathbf{x}(n)\big) + [1-\lambda(n)]\big(\mathbf{w}_2^T(n-1)\mathbf{x}(n)\big) \tag{3.57}$$

Where the $\mathbf{w}_1(n)$, and $\mathbf{w}_2(n)$ are the weight vectors of combined algorithms. $\lambda(n)$ is given in (3.25).

Replacing (3.55) and (3.24) in (3.26) the error signal of the combination yields;

$$e(n) = \boldsymbol{h}^T\mathbf{x}(k) + v(k) - \lambda(n)y_1(n) - \big(1-\lambda(n)\big)y_2(n) \tag{3.58}$$

The $e(n)$ is the total output error combination in the system identification setting.

3.3.3 2-D Convex of RI and second-order RI algorithms

The update equation in (3.18) can be extended to 2-D form in a convex combination of two RI algorithms such below:

$$\mathbf{w}_{i,n}(k_1,k_2) = \big[\mathbf{I} - \mu_i(n)\mathbf{R}_{i,n}\big]\mathbf{w}_{i,n-1}(k_1,k_2) + \mu_i(n)\mathbf{p}_{i,n} \tag{3.59}$$

Where $\mathbf{w}_{i,n}(k_1,k_2)$ is the extended weight vector of 2-D form with size $N \times N$ and at time n, $i = 1,2$, $k_1 = 0,1,\dots,N-1$ and $k_2 = 0,1,\cdots,N-1$. $\mathbf{R}_{1,n}$,$\mathbf{p}_{1,n}$,$\mathbf{R}_{2,n}$, and $\mathbf{p}_{2,n}$ are respectively given by,

$$\mathbf{R}_{1,n} = \beta\mathbf{R}_{1,n-1} + \mathbf{x}(m_1,m_2)\mathbf{x}^T(m_1,m_2) \tag{3.60}$$

$$\mathbf{p}_{1,n} = \beta \mathbf{p}_{1,n-1} + d(m_1, m_2)\mathbf{x}(m_1, m_2) \tag{3.61}$$

$$\mathbf{R}_{2,n} = \beta_1 \mathbf{R}_{2,n-1} + \beta_2 \mathbf{R}_{2,n-2} + \mathbf{x}(m_1, m_2)\mathbf{x}^T(m_1, m_2) \tag{3.62}$$

$$\mathbf{p}_{2,n} = \beta_1 \mathbf{p}_{2,n-1} + \beta_2 \mathbf{p}_{2,n-2} + d(m_1, m_2)\mathbf{x}(m_1, m_2) \tag{3.63}$$

Where $\mathbf{x}(m_1, m_2)$ is the filter input and $d(m_1, m_2)$is the desired output. The filter input $\mathbf{x}(m_1, m_2)$ and tap-weight vector $\mathbf{w}_{i,n}(k_1, k_2)$ can be defined using the following column-ordered vectors such as,

$$\mathbf{x}(m_1, m_2) = \begin{bmatrix} \mathbf{x}(m_1, m_2) \\ \vdots \\ \mathbf{x}(m_1, m_2 - N + 1) \\ \vdots \\ \mathbf{x}(m_1 - N + 1, m_2) \\ \vdots \\ \mathbf{x}(m_1 - N + 1, m_2 - N + 1) \end{bmatrix} \tag{3.64}$$

And

$$\mathbf{w}_{i,n}(k_1, k_2) = \begin{bmatrix} \mathbf{w}_{i,n}(0,0) \\ \vdots \\ \mathbf{w}_{i,n}(0, N-1) \\ \vdots \\ \mathbf{w}_{i,n}(N-1,0) \\ \vdots \\ \mathbf{w}_{i,n}(N-1, N-1) \end{bmatrix} \tag{3.65}$$

A possible way of data reuse is shown in Fig. 3.6. In this scheme, as shown in Fig. 3.6 (a), we can consider a mask, i.e. 3×3 pixels, which moves horizontally to the right by one column at a time until the end of each row. Afterward, the same process is repeated with the next row below until the last 9 pixels of the image are reached. At the end of each process of the mask, the data can be reshaped as shown in Fig. 3.6 (b), starting from the last pixel in the lower right corner going left and up.

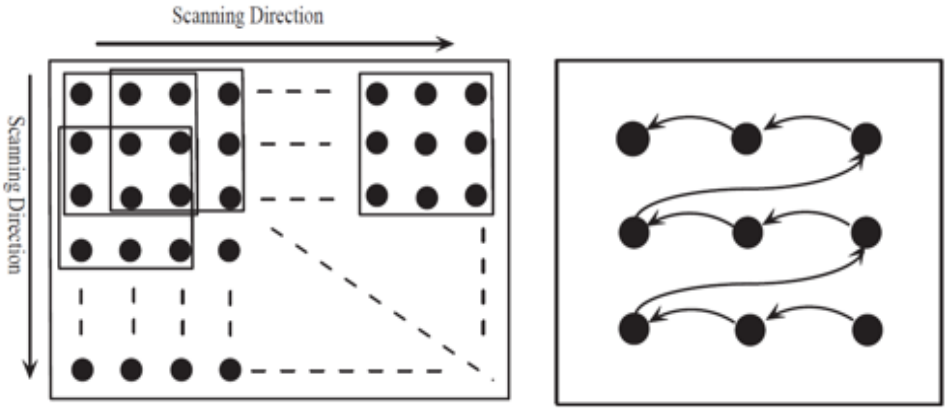


Figure 3.6. Rectangular configuration of data-reusing in 2-D plane

The i^{th} filter output is given by the following 2-D convolution:

$$y_{i,n}(m_1, m_2) = \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \mathbf{w}_{i,n}(k_1, k_2)\, \mathbf{x}(m_1 - k_1, m_2 - k_2) \tag{3.66}$$

Fig. 3.7 shows the convex combination of two RI adaptive filters in 2-D form.

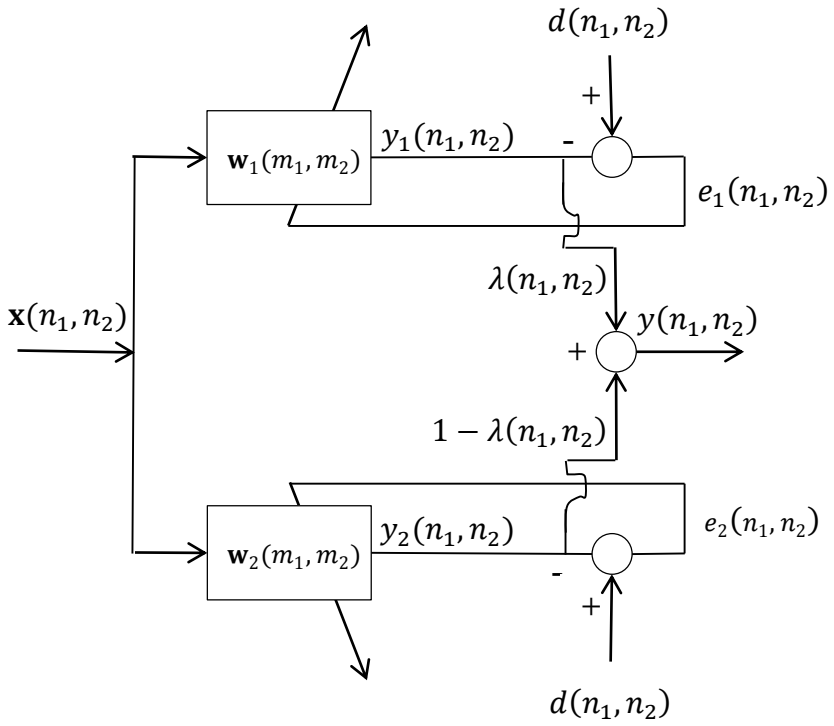


Figure 3.7. The block diagram of a 2-D convex RI algorithm

The output of the 2-D convexly combined algorithms can be computed as (Andersso, 1985),

$$y_n(m_1, m_2) = \lambda(m_1, m_2) y_{1,n}(m_1, m_2) + [1 - \lambda(m_1, m_2)] y_{2,n}(m_1, m_2) \qquad (3.67)$$

4. MACHINE LEARNING

Because of the modifications in technology huge amounts of data is recorded every day and the analyzing for these data cannot be done using classical statistics methods. So the best way to deal with this amount of information is using machine learning (Langley and Simon, 1995; Duffy, 1997).

Machine learning consists of intelligent methods which can teach programs how to make decisions. Those methods attracted attentions of researhers and have applied in all areas of computer science and beyond. Machine learning methods have used in many applications such as credit scoring, drug design, fraud detection, spam filters, stock trading.

Machine learning is about building systems that take data and make a correct prediction of solutions, without interference of human. The problems can be solved by machine learning methods presented by:

- Classification: which is a process of selecting incoming data and decide to which category these data belong to. A well-known example for it is the spam e-mail detection. A classifier can determine an e-mail is normal or spam.
- Regression: which is the process that can analyze the changing of output according to the change in one input while holding the other inputs at fixed values.
- Clustering: which is a process of partitioning the data into groups based on similarity, where the methods do not have prior knowledge about output.
- Associations: which is the process of discovering the interesting relations between variables in big databases.

Machine learning methods are divided into supervised and unsupervised methods. Supervised algorithms are trained from labeled training data (source) which consists of known observations to predict the label of unseen data (target). Unsupervised machine learning algorithms deals with registering unlabeled data instances to clusters depending on similarities.

4.1 Supervised Learning

Supervised learning is the most studied and famous field in machine learning

(Kotsiantis et al., 2007; Dollar et al., 2006), it is predicting the class of each instance in a dataset using training set. The training set is used to build a model which is used to classify testing set instances. Let the training set $D = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \dots, (\mathbf{x}_N, c_N)\}$, with N data instances which can be described as a vector of features, $\mathbf{x} = \{\mathbf{x}(1), \mathbf{x}(2), \dots, \mathbf{x}(p)\}$, where p is input feature count, and output of the system can take a value from $c = \{c(1), c(2), \dots, c(k)\}$, where the k is class label count. The supervised algorithm builds a model from training data D to assign the actual output as a class label to new or unseen input vectors of $\mathbf{x}_{N+1}, \mathbf{x}_{N+2}, \dots$

4.1.1 Supervised learning approaches

Many methods have been introduced in the field of supervised learning (Kotsiantis et al., 2006), they can be categorized into different approaches such as decision trees, instance-based learning, learning regressions, linear models, kernel methods, Bayesian classifiers, and artificial neural networks.

Decision trees (Myles et al., 2004) is widely used approach in the machine learning. The method has hierarchical models and it divides the data variable gradually the each branch descending from that node corresponds to one of the possible values for this variable. The leaf nodes of a decision tree contain the class label assigned to each sub region of the problem.

Instance-based learning methods do not provide a certain model like other approaches. They works by store the values of training inputs until the new instance examine be to classify. To assign the class label of new instances the new instances should be related to the stored instances. Finally classification approach will be related to the nearest category similar to it (De Mantaras and Armengol, 1998).

Linear models are simple methods for classification and prediction (Kutner et al., 2005). They have been widely used in statistical applications for a long time. The linear models are easy to understand, where the output is usually a weighted sum of the data inputs. The weight's magnitude shows the significance of each variable and its sign indicates if the effect is positive or negative on the system. The disadvantage in these methods can not be sufficient with nonlinear problems, where the resulting solution may not fit the data.

Kernel methods provide a simple connection from non-linearity to linearity problems (Scholkopf and Smola, 2002). These methods use to solve the nonlinear

problems by mapping them to linear separable problems with low dimentions. Then these problems can be modeling easier. The goal of this mapping will make the data more easily to separate with these new dimensions. The first approach was used with these methods by solving the binary classification problems using support vector machine.

Support vector machine is a common classifier can map the training data into a feature space with the help of a kernel function. The mapping is computing the similarity between two given observations. This transformation makes the problem linearly separable and then can use the decision hyperplanes between classes (Burges, 1998). However the training time will be very high to achieved very accurate results, but it is efficiency in high-dimensional problems, and good classifier for treatment the nonlinear problems.

Bayesian classifiers, which is depending on Bayes rule that used to classify the new input **x** depending on its variables, the class of the system can be chosen according to its maximum probability (Rosenblatt, 1962).

Artificial neural networks (Ripley, 2007; Haykin, 2009) are a model that tries to simulate the structure of functional aspects of biological neural networks. These networks consist of an interconnected group of processing units called neurons, and processes information using a connection is approach to computation. They are composed by one or more layers of processing units connected with each other. Each processing unit aggregates the inputs that it could receive from the environment or could be the outputs of other processing units, and sends the result, which is a weighted sum of the inputs in the simplest case, to other processing units. The connection between processing units are modeled with weights.

The simplest networks are called perceptrons (Rosenblatt, 1962) which have a single layer of processing units. Although these classifiers are able to distinguish labels in a binary classification problem by linear discrimination function, they present some limitations. If a set of instances is not linearly separable, perceptrons cannot make model where all instances are classified properly. Instead of them Multi layered perceptrons have been created to try to solve this problem (Williams and Hinton, 1986). These networks are usually used to model inputs and outputs by means of nonlinear discrimination functions. There are several ways with which a network can be trained for solving the problems. However, the well-known and widely used learning algorithm to estimate the values of the weights is the back propagation algorithm (Williams and

Hinton, 1986), which uses gradient descent to tune network parameters to get best fit a training set of input-output pairs. Artificial neural networks usually provide higher accuracies than other methods.

4.1.2 Predictive model validation

The performance of classification methods need to evaluate to know how well the classifiers work, where many criteria used to test the algorithm performances. The validation technique is a simple procedure to estimate the correct classes. The correct classes can be count it as a right class, because it is similar to actual class. On the other hand, wrong class considering as error. In the supervised learning there are several measures to evaluate performance, one of these measures depend on the error rate, also the percentage of the correct predicted classes over the total actual data as

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{4.1}$$

Also, sensitivity and specificity used to evaluate the algorithm performances.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.2}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{4.3}$$

Where, $TP, TN, FP,$ and FN are the number of true positives, true negatives, false positives, and false negatives, respectively. The maximum number of correct classified instances is the score of obtained result from algorithm.

Many evaluation methods have used such as k-fold cross validation (Stone, 1974). This method works by dividing all instances of dataset into k of approximately equal size. Each partition with using as a testing, using trained model of k-1 subsets. The process will repeated for all k subsets. Finally, the k accuracy values of tested partitions are sums and averaged by k to obtain final accuracy.

Another common technique using to compute the accuracy is called leave-one-out method (Mosteller and Tukey, 1968). Where this method is a special case of k-fold

cross validation. The accuracy computation similar to k-fold cross validation, but the number of k-fold is equals to the number of instance of dataset.

4.1.3 Artificial neural network (ANN)

This section we will see the biological background of ANN, and where the idea come from, in addition the basic structure of an ANN with neurodynamics, architecture and its parameters will describe in details.

4.1.3.1 The biological paradigm

ANN is inspired from human brain structure particularly the neurological part. ANN tries to simulate the working of biological networks (Ripley, 2007). However they are much far from their operation capacities because of complexity of the biological networks (Destexhe and Sejnowski, 1995). The cells found in human brain are called neurons. The information or signals are transferred between the neurons throught connections called as axons. The information receives to a neuron through its dendrites. Human brain contains about 100 billion neurons and 1014 synapses, the simple structure of a neuron shown in Fig. 4.1.

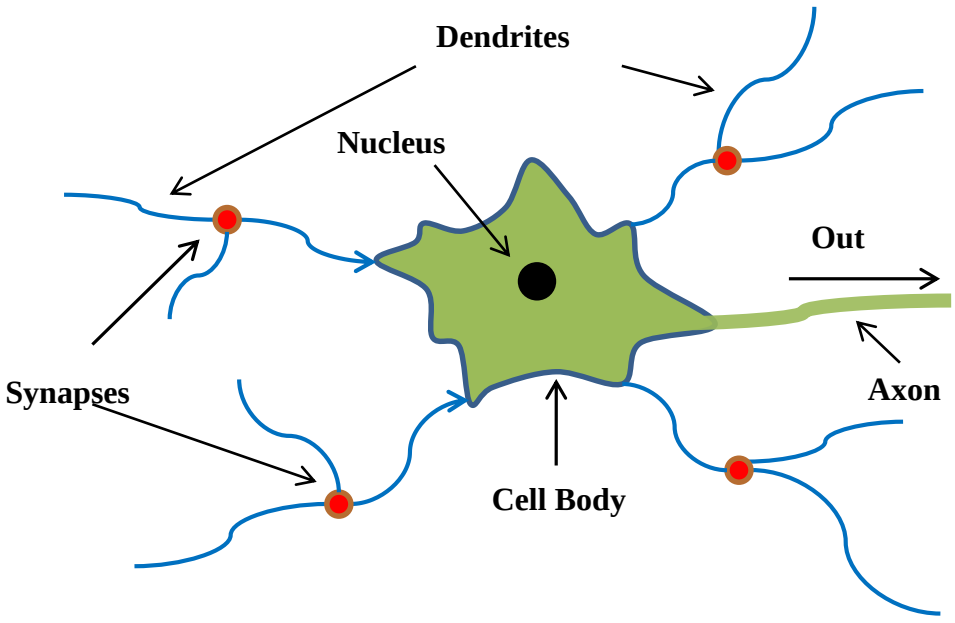


Figure 4.1. Schematic structure of a nerve cell

The processing in the biological brain is very powerful because it acts in a parallel manner and also it is very tolerant to faults. The fault tolerance came from the neural multi pathways which makes the information spreads in many directions and allow it to deal with noise in perfect ways (Carnevale and Rosenthal, 1992).

Biological neuron is so complicated so that even the most powerful supercomputers cannot simulate one neuron. The researchers are forced to simplify the structure of neurons while developing ANN.

ANN is different from other conventional computational techniques. The system designers of ANN do not need to know the rules and models that are required to perform the desired task. Instead it needs to train the system using the training samples. Training is similar to the process of the teaching children to recognize certain shapes letters and colors.

While ANN deals with the information in parallel way, computers deals with it in a serial fashion. While human brain sends information in milliseconds (10^{-3} s), computers sends it in much faster about nanosecond (10^{-9} s) range but brain still much better than computers in pattern recognition because it works in parallel way and every neuron works as a processor.

ANN is easy to build and it can deals with large noisy data. They are typically suitable to deal with nonlinear problems in which there are unknown rules.

They are very good in dealing with noisy or uncompleted data, because the spread in parallel minimizes the effect of loss data.

While training for ANN is relatively simple; a pre-processing is very important. But the process still easier comparing to modeling problems with conventional statistical methods.

4.1.3.2 Basic structure of an ANN

Artificial neural network consist of neurons which are the main items of the structure. Basically, ANN structure consist of three layers input layer, hidden layer, and output layer as shown in Fig. 4.2 and each layer consist of neurons. Some ANN may have multiple hidden layers, depending on the problem.

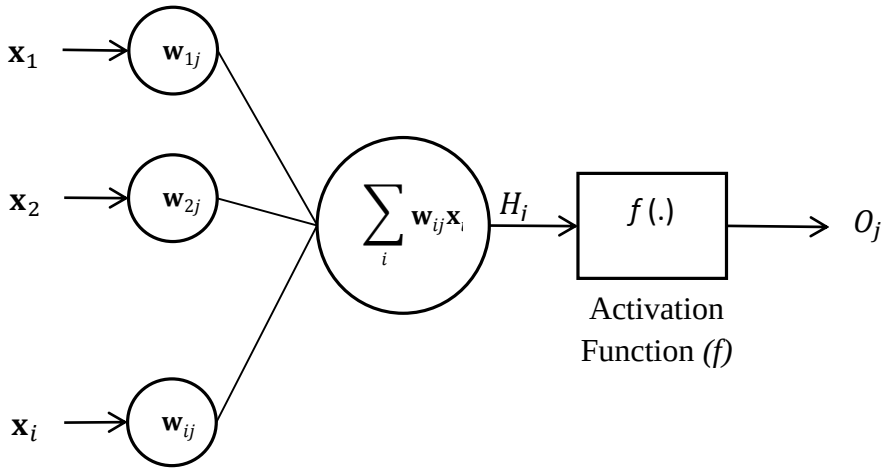


Figure 4.2. A basic schema of artificial neural network (ANN)

Neurons in brain communicate with each other by sending information using complex connection so ANN tries to simulate the same action by communicating each other using weighted connections. The weights could be negative or positive. Positive weights means the neuron is in excitation, while negative weights the neuron is in inhibition. In Fig. 4.2 shows the inputs $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i)$ are connected to hidden layer neuron j with weights $(\mathbf{w}_{1j}, \mathbf{w}_{2j}, \dots, \mathbf{w}_{ij})$. Each neuron receives the sum of multiplied inputs by its associated weight coefficients. At that moment the output should be transferred to an activation function $f(H_j)$, that is non-linear to give the final output (o_j). A commonly used transfer function is sigmoid function because of its differentiable properties is a simple, especially when backpropagation algorithm used.

The backpropagation ANN computes the output similar to feed forward ANN where it takes the input multiplied by its weights and all inputs are summed in each neuron then transferred to threshold function. The goal of BP algorithm is minimizing the error between output and the target. It propagates the error back to the network and the weights will be rearranged proportionally to the size of their error. This process is done forward and backward until the error drops under a specified minimum value.

To construct an ANN, labeled train data must presented to the system. The input data should be categorized as training or validation, where the training is the process of making ANN understands the pattern of data, and the validation is the process of testing the classification ability of ANN. The structure of an ANN must be defined exactly (number of neurons, number of hidden layers, types of activation functions used… etc.). The process of training is selecting the optimum weights that triggers a desired output

when a certain data enters the ANN and then validation process should be applied to whether the network working fine in selecting the desired output for certain input data.

Changing the number of neurons or number of hidden layers or activation function are options for reaching the good classification results in an ANN. The error estimation can prove the best ANN structure, for example if an ANN is over trained (not halted in the right time) the problem of curve fitting will happen and the system will try to fit itself instead of creating a generalized model. This will result bad prediction for validation of test sets. Another problem may happen if the system is not trained for long time it may settle at local minima and also will give bad results because the system generates a sub optimal model.

4.1.3.3 Brief description of the ANN parameters

Learning rate determines how much correction will be done in each iteration to the weights to reduce the error in the output. Small learning rate will increase the time of training, but it will make a better chance to reach the optimal weights by achieving minimum error, but will increase the chance of dropping the system into local minima. Large learning rate values will reduce the time of training, but it will increase the chance to overshoots the optimum values and in some cases the system will not be able to learn anything. To get rid of this dilemma the learning rate values should be tested for most appropriate value for certain application.

Momentum is a coefficent which determines the ratio of current weight to add updated weight as an extra term. If we assume system convergences to the minimum error step by step, momentum helps increasing the size of the steps. It also helps to avoid stuck in local minima. A big momentum will speed up the system convergence. On the other hand, a very small momentum cannot avoid local minima, and slow down the convergence of learning.

Training tolerance is the criteria that prove the network performance in the manner of accuracy obtained in the learning stage from training dataset. The testing tolerance is the accuracy that achieved on the test dataset using the optimal training model.

Error function should decrease constantly and converge to a minimum during training. System can decide if it should stop or not according to error function value. So it is an indication whether the ANN model is reached the optimal or suboptimal value.

4.1.3.4 Neural networks neurodynamics

The input layer of ANN acts as a buffer between the input data and ANN system, some of the input data should be scaled, converted, encoded, or normalized to be suitable for ANN.

The output layer for ANN works as the same of the input layer, but it transfers information to the outside of ANN. In some cases a post-processing is needed for the output, scaling may be made to it and it depends on the type of the system related to the ANN system.

To determine the actual output, the sum of multiplication between input data and weight coefficients are connected to transfer function f of the ANN. The function will decide the output, in general the activation functions used in hidden layer are in nonlinear characteristic. This mechanizm is shown clearly by,

$$o_j = f_j\left(\sum_i \mathbf{w}_{ij}\mathbf{x}_i\right) \tag{4.4}$$

The popular functions are used Logistic, and Hyperbolic functions.

$$\text{Logistic:} \quad f(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}} \tag{4.5}$$

$$\text{Hyperbolic tangent:} \quad f(\mathbf{x}) = \frac{e^{\mathbf{x}}-e^{-\mathbf{x}}}{e^{\mathbf{x}}+e^{-\mathbf{x}}} \tag{4.6}$$

The logistic functions are commonly used due to the ease of computing, it is in boundaries between 0 to 1. The derivative is given by;

$$\grave{f}(\mathbf{x}) = f(\mathbf{x})\big(1 - f(\mathbf{x})\big) \tag{4.7}$$

Applying the output of neuron o_j in logistic function f_j becomes,

$$f_j = \frac{1}{1 + e^{-\sum_i \mathbf{w}_{ij}\mathbf{x}_i - \theta_j}} \tag{4.8}$$

Where θ_j is the threshold.

The output decided in logostic function depending on this condition as,

$$f_j = \begin{cases} 1, & if \sum_i \mathbf{w}_{ij}\mathbf{x}_i > \theta_j \\ 0, & else \end{cases} \quad (4.9)$$

The weights are initially selected randomly or using an algorithm. These weights are adjusted during training to minimize the output error, iteratively.

4.1.3.5 Neural networks architecture

4.1.3.5.1 Types of interconnections between neurons

Each layer is a fully connected to other through neurons. Every neuron connects to all neurons of next layer and each connection between these neurons represented with weight coefficients. Neurons receives stimulus from input and processes the information to produce the output. The input processing method of neurons and connections between neurons can be structured in different ways. The commonly used neural network structure is called multi-layer perceptron (MLPs), which process the data from input to output in one direction.

4.1.3.5.2 The number of hidden neurons

Choosing the optimum number of hidden neurons have been discussed by many researchers. Generally, it can be smaller to the number of inputs. But there is no rule can give a fixed number of hidden neurons in the hidden layer. The suitable number of hidden neurons depends on the problem. However many rules are suggested by researchers that are related to select the number of hidden neurons.

4.1.3.5.3 The number of hidden layers

Some researcher demonstrated that, linear separable data can be divided without

hidden layers. So that means, with linear problems the simpler techniques can also work. On the other hand, in the nonlinear problems a wide variety of applications two hidden layers are needed to find solution. Two hidden layers make it skillful to solve different complicated problems in the MLP. Commonly used transfer functions are sigmoid, step, and threshold functions.

4.1.3.5.4 The perceptron

The perceptron model is a mathematical expression of a biological neuron. Biologically, the neurons receives signals via dendrite from neighboring neurons to process it, then transmit it to next neurons through the axons. The perceptron received signals as numerical values from its neighboors, and each signal is multipled by a weight value to obtain the output. Then output pass through threshold function to calculate the actual output. The perceptron is widely used with classification problems, which is about classifying input **x** into class c using neuron j as the following,

$$\sum \mathbf{w}_{ij}\mathbf{x}_i > \theta \tag{4.10}$$

As $\mathbf{w}_{ij}$ is the weight links from input i to neuron j, $\mathbf{x}_i$ is the received signal vector, and θ is the threshold on neuron j.

The weight $\mathbf{w}_{ij}$ of perceptron is shown by,

$$\mathbf{w}_{ij}(n) = \mathbf{w}_{ij}(n-1) + \eta\delta_j\mathbf{x}_i(n) \tag{4.11}$$

Where $\mathbf{w}_{ij}(n)$ is the weight coefficients vector which is updated in each iteration n, η is the learning rate in the range [0,1] and δ_j is the estimated error of neuron j is given by;

$$\delta_j = t_j - o_j \tag{4.12}$$

Where t_j and o_j are the target output and the desired output values from neuron j, respectively.

The δ_j process will minimize iteratively until obtain the optimum $\mathbf{w}_{ij}$. A single perceptron can solve binary Boolean (logic) functions such as AND, OR and NOT. Unfortunately a single perceptron seems only can solve linear problems that show limitations of its capability to solve many of nonlinear problems. (Minsky and Paperi, 1969) conclude that perceptron cannot solve nonlinear problems such as (XOR) problem.

4.1.4 XOR problem with multilayer perceptron

Linear separability is defined as a linear hyperplane that can separate the instances into two separate regions, if there were n inputs and $n > 2$, see the equation,

$$\sum_{i=1}^{n} \mathbf{w}_{ij}\mathbf{x}_j = \theta_j \tag{4.13}$$

As we know the hyperplane can divide the solution space into two partitions. In most cases this plane seperates the data into classes and the finding suitable plane problem is called classification problem. In some cases when the classes are more than two ($n > 2$) it is very difficult to find suitable separation planes but particular ANNs (e.g. MLPs) can learn to separate these kind of classes.

The perceptron can solve most cases of Boolean linear and separable functions but not the XOR problem, because XOR function is the simplest nonseparable function, where the output function is 0 if its two inputs are the same; on the other hand, output is 1 if the inputs are different. The XOR truth table is Table 4.1.

Table 4.1. Truth table of XOR function

$\mathbf{x}_1$	$\mathbf{x}_2$	Output (O_j)
0	0	0
0	1	1
1	0	1
1	1	0

The multilayer perceptron model (MLP) model can solve the XOR problem using two neurons in the hidden layer easily. The inputs are linked to hidden neurons with appropriate weights as shown in Fig. 4.3.

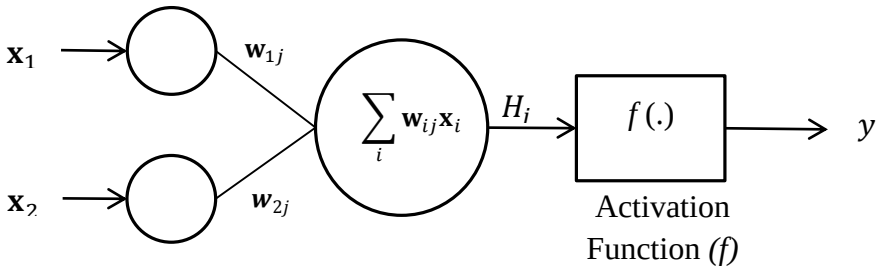


Figure 4.3. Diagram of perceptron model with activation function

The perceptron classifies the input data depending on the following equation,

$$\begin{aligned} y &= f(H_j) \\ &= f(\mathbf{w}_{1j}\mathbf{x}_1 + \mathbf{w}_{2j}\mathbf{x}_2, \theta) \\ &= \begin{cases} 1, & if\ \mathbf{w}_{1j}\mathbf{x}_1 + \mathbf{w}_{2j}\mathbf{x}_2 \geq \theta \\ 0, & else\ \mathbf{w}_{1j}\mathbf{x}_1 + \mathbf{w}_{2j}\mathbf{x}_2 < \theta \end{cases} \end{aligned} \tag{4.14}$$

Where H_j is the neuron of the overall output that will pass through activation function f, θ is the threshold value, $\mathbf{w}_{1j}$ and $\mathbf{w}_{2j}$ are the weights on the connection between inputs ($\mathbf{x}_1$,$\mathbf{x}_2$) and hidden neurons j.

A set of weight values will be calculated to achieve the required output, the two lines on the $\mathbf{x}_1$ and $\mathbf{x}_2$ plane are obtained by,

$$\theta = \mathbf{w}_{1j}\mathbf{x}_1 + \mathbf{w}_{2j}\mathbf{x}_2 \tag{4.15}$$

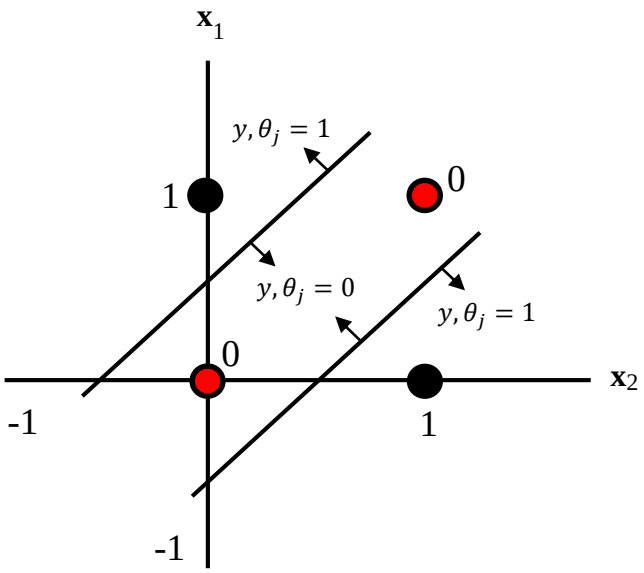


Figure 4.4. The separation of nonlinear XOR data points

Fig. 4.4 shows outputs of the XOR Boolean function. As it seen in the figure XOR has nonlinear separable output, where it is not possible to separate the 0s and 1s using one hyperplane. MLP with two hyperplanes is a great technique to solve XOR function.

4.1.5 Learning algorithms

The learning concept in the ANN corresponds to weight coefficient adjustment procedure. ANNs is famous for using "learning idea" exhaustively. Learning process uses training dataset that contains inputs and corresponding target values. The goal of network is minimizing the error between the target values $t(n)$ and the desired output $o(n)$ until reach the correct output with minimum error estimation $E(n)$. The weights are adjusted gradually and iteratively during the learning time. The common learning algorithm is called backpropagation algorithm.

Many learning algorithms have been proposed, but in this chapter only mostly famous algorithms which are Delta rule, and backpropagation (BP) algorithms are refered where the learning in these algorithms are adjusting the weights $\mathbf{w}_{ij}(n)$ and the biases $b_j(n)$ to minimize the overall squared error $E(n)$ by,

$$E(n) = \frac{1}{2}\sum_{i} \|t(n) - o(n)\|^2 \tag{4.16}$$

Where i, $t(n)$, and $o(n)$ are the the input patterns, the target, and the estimated output, respectively. Weight coefficients are adjusted until minimized error obtained in a suitable time period. The error surface will moves down gradually untill fall to ravine.

4.1.5.1 The delta rule

Widrow and Hoff were proposed the Delta rule in (Widrow and Hoff, 1960). It is also called least mean square (LMS) algorithm and most commonly applied learning rule. Delta rule learning is simply comparing input and the output vectors to predict the correct answer. If the difference is big, the algorithm will adjust the weight vector to minimize the difference, otherwise the convergence will reach the steady-state with minimum error. There isn't hidden neuron in Delta rule so it is a simple linear function.

The weight change $\mathbf{w}_{ij}(n)$ is given by,

$$\Delta\mathbf{w}_{ij}(n) = \eta\delta_j(n)\mathbf{x}_i(n) \tag{4.17}$$

Where η is the learning rate betwen [0,1], and updated weight on neuron j.

The Delta rule error calculated by,

$$\delta_j = t(n) - o(n) \tag{4.18}$$

Where $o(n)$ given by,

$$o(n) = \sum_{i=1}^{N} \mathbf{x}_i(n)\mathbf{w}_{ij}\,(n) \tag{4.19}$$

4.1.5.2 Back-propagation (BP) algorithm

BP algorithm is a supervised learning one; that is applied with a known dataset of input-target samples and has been used in prediction and classification applications.

Fig. 4.5 illustrates the BP architecture in details. It consists of an input layer, one or more hidden layer, and an output layer. Layers are connected sequentially starting from the input layer through the hidden layers to the output layer. Where the connections between layers contain weights and each layer includes one or more neurons.

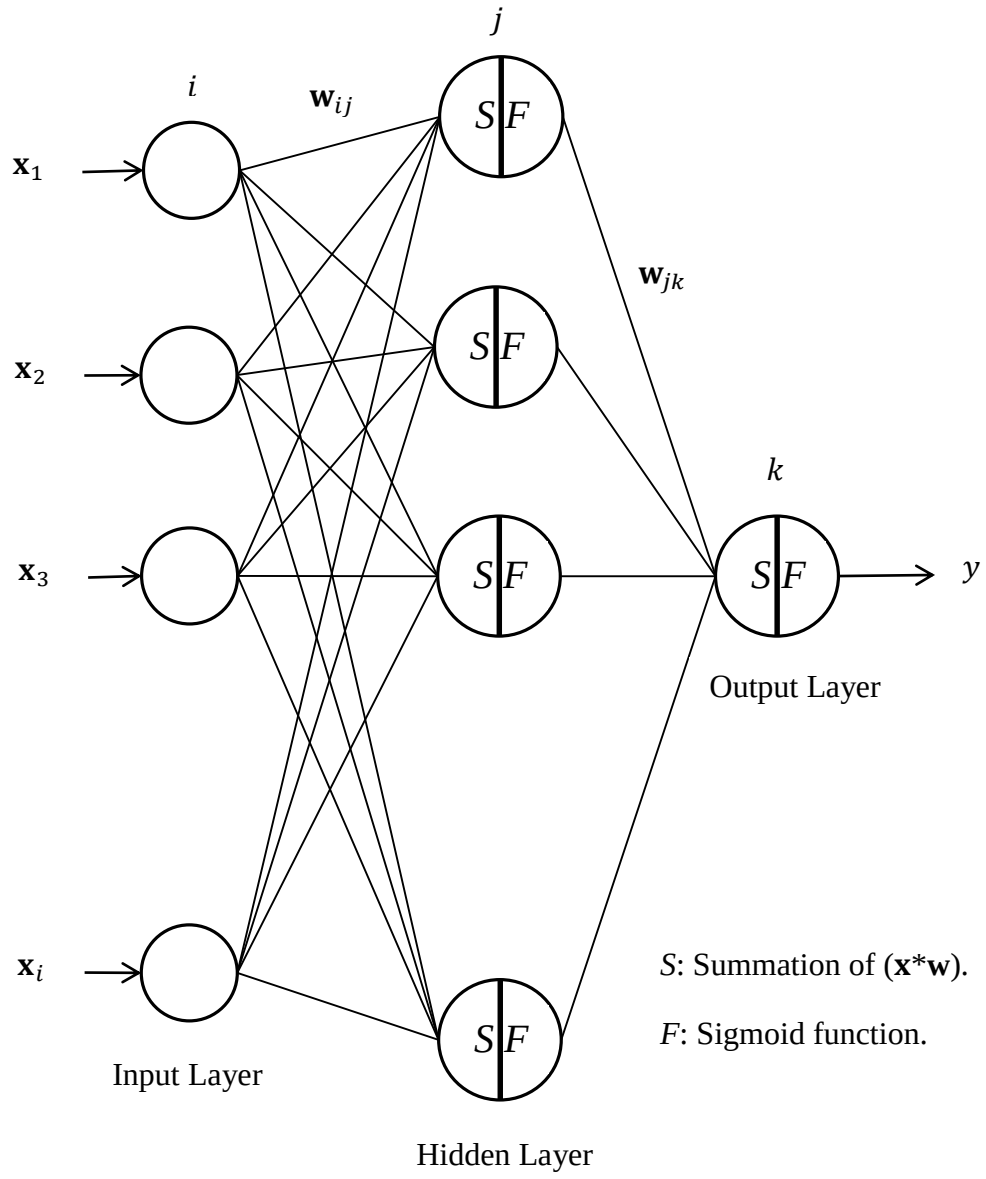


Figure 4.5. Schematic representation of back-propagation architecture

The basic idea behind BP is to minimize the overall output error gradually during the learning process. Whereas the training sets are estimated iteratively through the input layer to predict the correct output. The BP process is divided in two stages: forward and backward process. In the forward process, the BP architecture is described

as: $\mathbf{x}$ are the inputs to the neural network with i neurons, $\mathbf{w}_{ij}$ are the weights of interconnections between inputs and hidden layers, and j neurons for the hidden layer. Hidden layer is defined by,

$$H_j = b_{in} + \sum_{i=1}^{N} \mathbf{x}_i \mathbf{w}_{ij} \tag{4.20}$$

Where b_{in} is a bias input layer, hidden layer will pass through activation function (f). In this study, sigmoid function has been used as a good activation function (Karlik and Olgac, 2011),

$$f(H_j) = \frac{1}{1 + e^{-(H_j)}}, \tag{4.21}$$

After calculating the overall output by multiplying the output of the hidden layer neurons with the hidden layer weights $\mathbf{w}_{jk}$, the results, then, pass through a sigmoid function (called threshold) as shown below;

$$y_k = b_n + \sum_{j=1}^{m} \mathbf{w}_{jk} f(H_j) \tag{4.22}$$

Where b_n is the bias of hidden layer and k output neurons.

To predict the correct output by obtaining minimum error E for each pattern (p) by subtracting the overall output o from the target t;

$$E = \frac{1}{2} \sum_{j=1}^{p} (t_j - o_j)^2 \tag{4.23}$$

We can minimize E by using an iterative process of gradient descent for each weight which is updated using the increment;

$$\Delta \mathbf{w}_{kj} = -\frac{\eta \partial E}{\partial \mathbf{w}_{kj}} \qquad for \;\; j = 0,1, \dots, n \tag{4.24}$$

Where η defines learning coefficient which represent the step size of each iteration in the negative gradient direction. One all partial derivatives are computed as,

$$\frac{\partial E}{\partial \mathbf{w}_{kj}} = \frac{\partial E}{\partial\, net_k}\frac{\partial\, net_k}{\partial\, \mathbf{w}_{kj}} \tag{4.25}$$

$$\frac{\partial\, net_k}{\partial\, \mathbf{w}_{kj}} = \frac{\partial}{\partial\, \mathbf{w}_{kj}}\sum_j \mathbf{w}_{kj} y_j = \sum_j \frac{\partial\, \mathbf{w}_{kj}.y_j}{\partial\, \mathbf{w}_{kj}} = y_j \tag{4.26}$$

In the backward process, weights on the connections between all layers will be updated to minimize the error between target (or desired) and output until finding the optimum weights with minimum E (Ozbay and kalik, 2002). The error of belonging to neutons of layer k called "delata" (shortly ∂) is computed by,

$$\delta_o = -\frac{\partial E}{\partial\, net_k} \tag{4.27}$$

(4.25) is placed instead of both (4.26) and (4.27);

$$\frac{\partial E}{\partial \mathbf{w}_{kj}} = -\delta_o.y_j \tag{4.28}$$

The backpropagation step computes the gradient of E with respect to the input. We can perform gradient descent by adding to each weight $\mathbf{w}_{kj}$the increment $\Delta\mathbf{w}_{kj}$,

$$\Delta\mathbf{w}_{kj} = \eta\delta_o.y_j \tag{4.29}$$

If we denote the backpropagation error at the j^{th} node by δ_j, we can then express the partial derivative of e with respect to $\mathbf{w}_{ij}$ as,

$$\delta_o = -\frac{\partial E}{\partial net_k} = -\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k} \tag{4.30}$$

$$\frac{\partial E}{\partial o_k} = -(t_k - o_k) \tag{4.31}$$

(4.31) is defined local error for k neurons.

$$\frac{\partial o_k}{\partial net_k} = \grave{f}_k(net_k) \tag{4.32}$$

The last two equations are placed instead of (4.30),

$$\delta_o = (t_k - o_k)\grave{f}_k(net_k) \tag{4.33}$$

For k neurons, (4.29) is placed instead of the last one;

$$\Delta \mathbf{w}_{kj} = \eta(t_k - o_k)\grave{f}_k(net_k)y_j \tag{4.34}$$

If we denote the backpropagation error at the j^{th} (hidden layer) by δ_j, we can then express the partial derivative of E with respect to $\mathbf{w}_{ji}$ as,

$$\begin{aligned}\Delta \mathbf{w}_{ji} &= -\eta \frac{\partial E}{\partial \mathbf{w}_{ji}} = -\eta \frac{\partial E}{\partial\, net_j}\frac{\partial\, net_j}{\partial\, \mathbf{w}_{ji}} = -\eta \frac{\partial E}{\partial\, net_j}\mathbf{x}_i \\ &= -\eta\left(-\frac{\partial E}{\partial y_j} - \eta \frac{\partial y_j}{\partial\, net_j}\right)\mathbf{x}_i = \eta\left(\frac{\partial E}{\partial y_j}\right)\grave{f}_j(net_j)\mathbf{x}_i \end{aligned} \tag{4.35}$$

We can perform grdient decent by adding to each weight $\mathbf{w}_{ji}$ the increment $\mathbf{w}_{ji}$ as,

$$\Delta \mathbf{w}_{ji} = \eta \delta_y x_i \tag{4.36}$$

However, factor not directly promoted. In particular, the effect of the output layed,

$$-\frac{\partial E}{\partial y_j} = -\sum_k \frac{\partial E}{\partial net_k}\frac{\partial net_k}{\partial y_j} = \sum_k \left(-\frac{\partial E}{\partial net_k}\right)\frac{\partial}{\partial y_j}\sum_k \mathbf{w}_{kj}\, y_j$$

$$= \sum_k \left(-\frac{\partial E}{\partial net_k}\right) \mathbf{w}_{kj} = \sum_k \delta_o \mathbf{w}_{kj} \tag{4.37}$$

In this case, δ_y can be found;

$$\delta_y = f_j(net_j) \sum_k \delta_o \mathbf{w}_{kj} \tag{4.38}$$

Transfer (threshold) is used as a function of the sigmoid function,

$$f_j(net_j) = y_j = \frac{1}{1 + e^{-net_j}} \tag{4.39}$$

Whether the derivative of this expression, when necessary simplification, the following equations are obtained;

$$f_j(net_j) = \frac{1}{1+e^{-net_j}} \frac{1+e^{-net_j}-1}{1+e^{-net_j}} \tag{4.40}$$

$$\frac{\partial y_j}{\partial net_k} = y_j(1 - y_j) \tag{4.41}$$

When similar works are made for k layer, the following equations are obtained;

$$\frac{\partial o_k}{\partial net_k} = \grave{f}_k(net_k) = o_k(1 - o_k) \tag{4.42}$$

$$\delta_o = (t_k - o_k) o_k (1 - o_k) \tag{4.43}$$

$$\delta_y = y_j(1 - y_j) \sum_k \delta_o \mathbf{w}_{kj} \tag{4.44}$$

After adding learning and momentum coefficients, updating the weights for both hidden ($\mathbf{w}_{ji}$) and output ($\mathbf{w}_{kj}$) layer are obtained such as,

$$\Delta \mathbf{w}_{ji}(n+1) = \eta \delta_y \mathbf{x}_i(n) + \alpha \Delta \mathbf{w}_{ji}(n) \tag{4.45}$$

$$\Delta \mathbf{w}_{kj}(n+1) = \eta \delta_o y_i(n) + \alpha \Delta \mathbf{w}_{kj}(n) \qquad n = 0,1,\ldots \tag{4.46}$$

Where n represents number of iteration and η is the learning rate (Karlik, 2000). The biases (bias weights) are updated by,

$$b_j(n) = b_j(n) + \eta . f\big(H_j\big)\Big(1 - f\big(H_j\big)\Big)\mathbf{x}_i(n)\mathbf{w}_j(n)e(n) \tag{4.47}$$

4.1.5.3 Back-propagation with adaptive filtering momentum (BPAM) algorithm

Adaptive momentum is an efficient way to speed-up BPNN process and improve the algorithm's accuracy. Shao and Zheng (2009) proposed an adaptive momentum algorithm called BPAM. The weight update equation has been derived as;

$$\Delta \mathbf{w}(n) = -\eta E\mathbf{w}\big(\mathbf{w}(n)\big) + \alpha(n)\Delta \mathbf{w}(n-1) \qquad n = 0,1,\ldots \tag{4.48}$$

Where α is the adaptive momentum and updated by the following:

$$\alpha_i(n) = \begin{cases} \alpha \dfrac{-\eta E\mathbf{w}\big(\mathbf{w}(n)\big).\Delta \mathbf{w}(n-1)}{\|\Delta \mathbf{w}(n-1)\|^2}, if E\mathbf{w}\big(\mathbf{w}(n)\big).\Delta \mathbf{w}(n-1) < 0 \\ 0, \qquad\qquad otherwise \end{cases} \tag{4.49}$$

Where $\alpha_i(n) = (\alpha_0(n), \alpha_1(n), \ldots, \alpha_j(n))$ are changing corresponding to the equation above, $i = 0,1,\ldots j$.

The BPAM adaptive momentum adjusts itself iteratively by multiplying the current weight coefficients by the previous updated coefficients. If the momentum coefficients are less than zero, they are given a positive value to accelerate learning by updating momentum. Otherwise the momentum is considered as zero to maintain the error downhill.

4.1.5.4 Back-propagation algorithm with variable adaptive momentum (BPVAM) algorithm

In the BPAM algorithm, α (the adaptive momentum) is controlled by the learning rate parameter η. This η is dependent on the eigenvalues of the autocorrelation matrix of the input. If these eigenvalues are relatively high, η should be selected relatively small to provide a low error performance and, hence, the convergence rate of the algorithm will be low. On the other hand, if the eigenvalues of the autocorrelation matrix are relatively small, η can be selected relatively high which, in turn, provides high convergence rate but also high error performance.

If we consider estimating the autocorrelation matrix, $\mathbf{R}(n)$, of the input recursively as:

$$\mathbf{R}(n+1) = \beta\mathbf{R}(n) + \mathbf{R_{xx}} \tag{4.50}$$

Where β is the forgetting factor ($0 \ll \beta < 1$), $\mathbf{R_{xx}} = E\{\mathbf{x}(n)\mathbf{x}^T(n)\}$, E is the expectation operator, T is the transposition operator and $\mathbf{x}(n)$ are the inputs. Taking expected value for both sides of (4.50),

$$\bar{\mathbf{R}}(n) = \frac{1-\beta^n}{1-\beta}\mathbf{R_{xx}} \tag{4.51}$$

Where $\bar{\mathbf{R}}(n) = E\{\mathbf{R}(n)\}$. Solving (4.51) in the steady-state ($n \to \infty$) yields:

$$\bar{\mathbf{R}}(\infty) = \frac{1}{1-\beta} \tag{4.52}$$

Equation (4.52) implies that the eigenvalues of the estimated autocorrelation matrix increase exponentially, and in the limit they become $\frac{1}{1-\beta}$ times the original one.

In this study, following the same procedure of (Ahmad et al., 2011), we propose a new backpropagation algorithm with variable adaptive momentum. The proposed variable momentum is expressed by (Ahmad et al., 2011):

$$\alpha(n) = \frac{\lambda}{1-\beta^n} \tag{4.53}$$

Where $\lambda < \frac{2-2\beta}{max\ eigen\ value\ of \mathbf{R_{xx}}}$

In (4.53), for enough large β, the term $1-\beta^n$ reaches unity and, at the beginning, $\alpha(n)$ is relatively large to provide a fast convergence of the weights update in (4.45) and (4.46) and after some time is becomes very close to λ (a small positive constant), hence, it provides low error performance for the weights update in (4.45) and (4.46).

$$\Delta \mathbf{w}_{ji}(n+1) = \eta \delta_y \mathbf{x}_i(n) + \left(\frac{\lambda}{1-\beta^n}\right) \Delta \mathbf{w}_{ji}(n) \tag{4.54}$$

$$\Delta \mathbf{w}_{kj}(n+1) = \eta \delta_o y_i(n) + \left(\frac{\lambda}{1-\beta^n}\right) \Delta \mathbf{w}_{kj}(n), \qquad n = 0,1,\ldots \tag{4.55}$$

Where n represents number of iteration, and $\Delta \mathbf{w}$ is defined as updating the weights.

4.1.6 Statistical classification algorithms

4.1.6.1 K-NN classifier

The k-nearest neighbours (K-NN) is a non-parametric classifier, and one of the most popular and simple methods, which has been used to solve different types of problems in data mining (Shouman et al., 2012). K-NN is able to classify instances based on majority class or closest neighbours of **x** in the training set (Derelioglu and Gugen, 2011). By partitioning the feature space into K regions are to find the closest feature space of new instance (Larose, 2005). In this study the Euclidean distance has been used to compute the distance between both of neighbouring instances **x**, and y.

$$\sqrt{\sum (\mathbf{x}_i - y_i)^2} \tag{4.56}$$

Where $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ and $y = (y_1, y_2, \ldots, y_N)$, however, N is n-dimensional space. Moreover K-NN has more robustness to noisy training data (Duda et al., 2001; EI Gayar et al, 2006).

4.1.6.2 Naïve bayes

Naïve Bayes (NB) classifier is characterized as easy and fast to implement because it requires a small training data which is to estimate the parameters (means and variances of the variable) for classification (Keramati and Yousefi, 2011). The main idea behind NB process is independently investigating attributes of data sets (Kim, 2009; Hou et al., 2010; Murphy, 2006), and it is easy to expand the model for very large datasets.

4.1.6.3 Support vector machine

Support Vector Machine (SVM) is one of the useful and well-known supervised learning algorithms which have been widely attracting many researchers due to its high classification accuracies in wide variety of fields (Vapnik, 1995; Baesens et al., 2003). SVM has proved its the ability in solving non-linear problems (Martens et al., 2009). However, SVM suffers from high computational complexity which causes an increase usage of memory and time consumption with big data implementation. The working principle in the non-linearly applications is to map the training set into high-dimensional feature. Linearly separable used to find a maximum margin by investigating the optimal hyperplane (separating hyperplane) between the classes (Huang et al., 2002). The linearly separable method has clearly depicted in Fig. 4.6.

Gaussian radial kernel function is usually used to transfer non-linear data to a high-dimensional feature space (Burbidge et al., 2001). The kernel function is given by,

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_y\|^2}{2\sigma^2}\right) \tag{4.57}$$

Where $\mathbf{x}_i$ is defined as input of the training set.

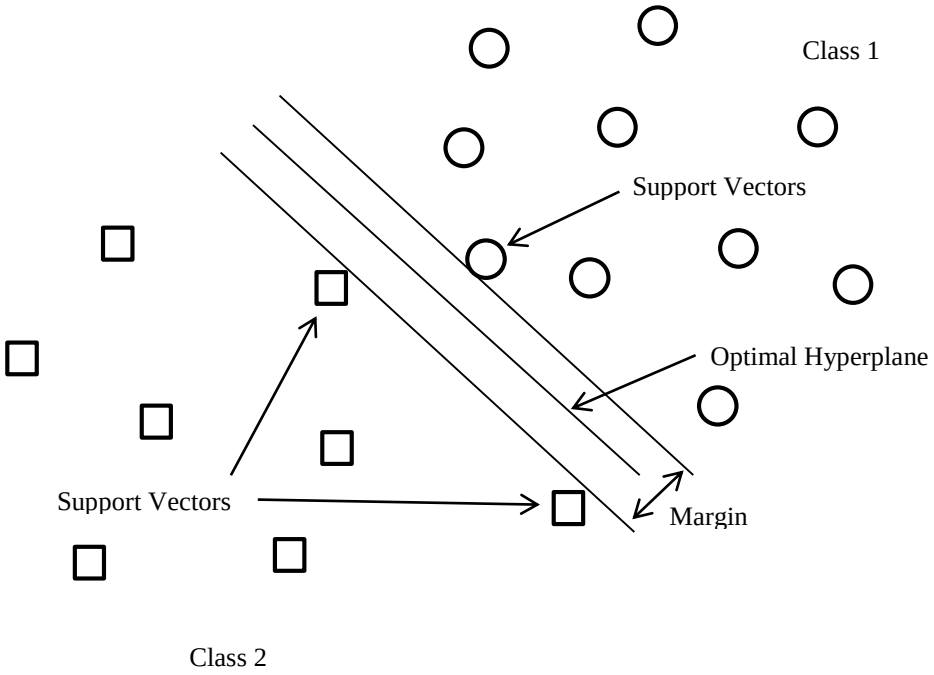


Figure 4.6. SVM estimate the hyperplane of best classes separate

The SVM separating hyperplanes can be depicted for investigating classification using the lemma by,

$$F(\mathbf{x}) = sgn\left(\sum a_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b\right) \tag{4.58}$$

Where a_i is the Lagrangian method for mapping data to high-dimensional space, y_i is the class label of $\mathbf{x}_i$ and b is the bias, sgn function which describes the instances; i.e., +1 for a positive class or -1 for the negative class,

4.1.6.4 Linear discriminant analysis

Linear Discriminant Analysis (LDA) is characterized by its simplicity and success in solving different types of problems (Yang and Yang, 2003). LDA provides a linear combination of features by creating a new variable (Karlik, 2014). LDA is based upon the concept of searching for a linear combination of variables which are predicted by the best separation of two target classes. The discrimination is to compute the Mahalanobis distance between two classes by,

$$\Delta^2 = \beta^T(\mu_1 - \mu_2) \tag{4.59}$$

Where Δ is defined Mahalanobis distance, μ_1 and μ_2 are mean vectors, and β is the coefficient vector.

4.2 Unsupervised Learning

Unsupervised learning is one of the mostly studied fields in machine learning. It aims to extract structures from data instances. Its goal is finding the groups of similar features in the dataset without prior knowledge about data, which is also called clustering. The clustering can find a suitable structure from collected data which are characterized by several values. These groups are cruited depending on the similarity of the data instances. Each group is called cluster. The conventional clustering methods have categorized into two major groups as hierarchical and partitional.

The clustering can be defined as follows. Let the data set $D = \{(\mathbf{x}(1), \dots, \mathbf{x}(N)\}$ are the set of instances described as a vector of features in a space of dimension F, that is, $\mathbf{x}(i) \in \Re^F, \forall_i \in \{1, \dots, N\}$. The process assigns a cluster label $c(i)$ to each instance, with $c(i) \in \{1, \dots, k\}$, depending on the similarity measurement between all instances. The clusters count k can not be predicted easily, so it is needed to consider all data instance. A common way is estimating similarity among input data.

The way of using similarity plays an important role in clustering approaches, such as hierarchical or partitional. Most commonly used similarity technique is the concept of distance which measures how much this object is close to that cluster. Mostly used distance measurement techniques are Euclidean, Minkowski, and Mahalanobis distances.

The Euclidean distance is calculated the distance between two instances of the same length given by,

$$\big(\mathbf{x}(n), \mathbf{x}(n+1)\big) = \sqrt{\sum_{j=1}^{F} (\mathbf{x}_j(n) - \mathbf{x}_j(n+1))^2} \tag{4.60}$$

The Minkowski distance is a generalization Euclidean distance as defined by,

$$\big(\mathbf{x}(n),\mathbf{x}(n+1)\big)=\sqrt{\sum_{j=1}^{F}\mathbf{w}_j^r(\mathbf{x}_j(n)-\mathbf{x}_j(n+1))^r} \qquad (4.61)$$

Where r is the Minkowski distance, actually it is Manhattan distance has measures with $r=1$, $\mathbf{w}_j$ is a possible weight for feature j.

The Mahalanobis distance is defined by,

$$\big(\mathbf{x}(n),\mathbf{x}(n+1)\big)=\sqrt{\big(\mathbf{x}(n)-\mathbf{x}(n+1)\big)^T\sum-1\big(\mathbf{x}(n)-\mathbf{x}(n+1)\big)} \qquad (4.62)$$

Where Σ is the covariance matrix in a space of dimension f.

4.2.1 Unsupervised learning approaches

Different starting points and criterias usually lead to different categories of clustering algorithms (Everitt et al., 2001; Xu and Wunsch, 2005). Generally clustering techniques can be classified into hierarchical clustering, partitional clustering, and probabilistic clustering, depended on the characteristics of the generated clusters. Partitional clustering groups the elements exclusively, so that any element belonging to one specific cluster cannot be a member of another cluster. On the other hand, hierarchical clustering produces a hierarchical structure of clusters. Hierarchical clustering proceeds successively by either merging smaller clusters into larger ones or by splitting larger clusters into smaller ones. Finally, probabilistic clustering provides a cluster membership probability for every element, where elements have a specific probability of being members of several clusters. Some of the clustering techniques above are used throughout this thesis.

A clustering method which used commonly is the Kohonen self-organizing map or SOM, which is a type of neural network that can perform unsupervised learning Kohonen (1990).

4.2.1.1 SOM algorithm

The architecture of SOM consists of an input and an output layer connected to each other where each connection is associated with a weight. The topologies of neuron connections used for SOM map are hexagonal and rectangular (Tasdemir et al., 2011; Wong et al., 2006). The SOM output layer consists of 2-D grid of $z \times d$ neurons in low dimension. The input vectors $\mathbf{x}_i = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, $i = 1, 2, \dots n$, where i is the number of input vectors and n is the input units. Each i is associated to the SOM map through weight vector $\mathbf{w} = (\mathbf{w}_{n1}, \mathbf{w}_{n2}, \dots, \mathbf{w}_{nd})$, where the d is the SOM dimension as shown in Fig. 4.7.

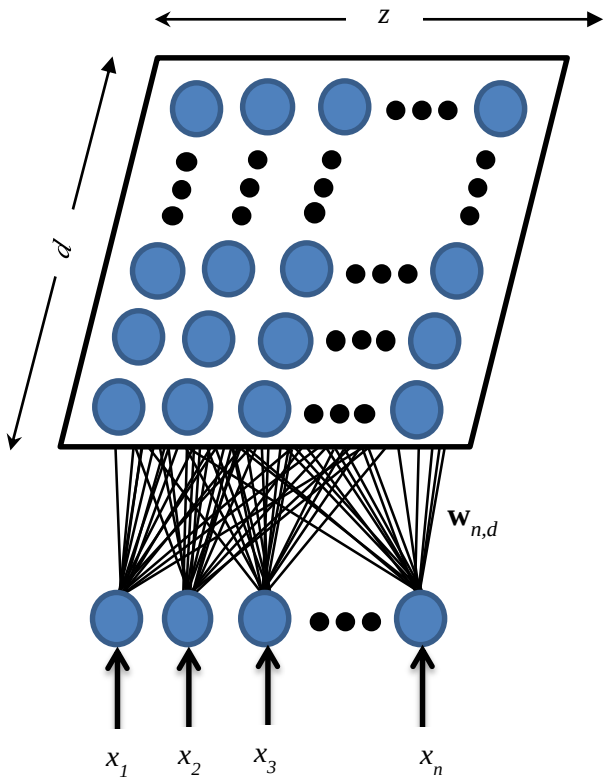


Figure 4.7. Illustration of the kohonen SOM architecture

The learning process of conventional SOM consists of the following steps: Firstly, initialize the weight vector $\mathbf{w}_{n,d}$ randomly as grid of $z \times d$ neurons. Secondly, feed the SOM network by the input vector $\mathbf{x}$ from dataset. Input vector $\mathbf{x}$ is fed to all neurons, synchronously. Thirdly, after calculating the distance between input and output neurons, find the closest neuron to input (smallest distance), called best matching unit

(BMU). The BMU of the winner neuron c is calculated using Euclidian distances,

$$c = \underset{i}{arg\ min}\ (\|\mathbf{w}_i(n) - \mathbf{x}(n)\|) \tag{4.63}$$

This process is repeated for all other input vectors in the dataset. Finally, the weight vectors of winner neuron are updated in an each iteration of learning process using,

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \alpha(n)\,.\,[\mathbf{x}(n) - \mathbf{w}_i(n)] \tag{4.64}$$

Where, $\alpha(n)$ is the learning rate.

One variant of SOM algorithm is the GF-SOM algorithm. It has been a widely used in many applications as a neighborhood function. Weight vectors update for GF-SOM is done using,

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + h_{c,i}(n)[\mathbf{x}(n) - \mathbf{w}_i(n)] \tag{4.65}$$

Where $h_{c,i}$ is the Gaussian neighborhood function defined as,

$$h_{c,i}(n) = \alpha(n)\,.\,exp\left(-\frac{\|r_c - r_i\|}{2\sigma^2(n)}\right) \tag{4.66}$$

Where $\|r_c - r\|$ is the Euclidean distance between the positions of the winning neuron c and the neuron i on the grid in each weight updating and $\sigma(n)$ is the width of Gaussian. The $\alpha(n)$ and $\sigma(n)$ are decreasing gradually during the learning process by constants factors δ_α and δ_σ, respectively, as

$$\alpha(n+1) = \delta_\alpha.\alpha(n) \tag{4.67}$$

$$\sigma(n+1) = \delta_\sigma.\sigma(n) \tag{4.68}$$

4.2.1.2 PLSOM algorithm

The main idea behind PLSOM is to overcome the limitation of conventional SOM. The PLSOM does not need the learning rate annealing scheme or the neighborhood size annealing schemes as in the conventional SOM algorithm (Berglund and Sitte, 2006). The basic process of PLSOM depends on its efficiency in fitting the input data. The fitting error is computed using the scaling variable ϵ by normalizing the distance between the input space $\mathbf{x}(n)$ and the weight vector $\mathbf{w}_c(n)$ of the winner neuron c. The variable ϵ is applies to scale the weight update equation as in (4.72) and to determine the size of the neighborhood as in (4.73). The scaling variable ϵ is calculated as,

$$\epsilon(n) = \frac{\|\mathbf{w}_c(n) - \mathbf{x}(n)\|}{\rho(n)} \tag{4.69}$$

Where

$$\rho(n) = max(\|\mathbf{x}(n) - \mathbf{w}_c(n)\|, \rho(n-1)) \tag{4.70}$$

$$\rho(0) = \|\mathbf{x}(0) - \mathbf{w}_c(0)\| \tag{4.71}$$

If ϵ is large the map will fit input space poorly. It will require large readjustments which in tern means that further iterations are required. On the other hand, if ϵ is small the map fitting will be satisfactory and no large update is required. In PLSOM, the weight vectors of the winner neuron are updated using,

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \epsilon(n).h_{c,i}(n)[\mathbf{x}(n) - \mathbf{w}_i(n)] \tag{4.72}$$

The used neighborhood function is the Gaussian neighborhood $h_{c,i}$, given as,

$$h_{c,i}(n) = exp\left(-\frac{\|r_c - r_i\|}{\Theta(\epsilon(n))^2}\right) \tag{4.73}$$

Where $\|r_c - r_i\|$ is the distance between the positions of the winning neuron c

and the neuron i on the map as mentioned before in the conventional SOM. The $\Theta\big(\epsilon(n)\big)$, is used for scaling the neighborhood function as,

$$\Theta\big(\epsilon(n+1)\big) = \beta.\epsilon(n), \qquad \Theta\big(\epsilon(n+1)\big) \geq \theta_{min} \tag{4.74}$$

In addition, there is another way to calculate $\Theta\big(\epsilon(n)\big)$ shown in (4.75) and (4.76), where $\beta = constant\ \forall n$, and θ_{min} is constant.

$$\Theta\big(\epsilon(n)\big) = (\beta - \theta_{min})\epsilon(n) + \theta_{min} \tag{4.75}$$

$$\Theta\big(\epsilon(n)\big) = (\beta - \theta_{min})ln(1 + \epsilon(n))(Eu - 1) + \theta_{min} \tag{4.76}$$

Where $ln(.)$ and Eu are the natural logarithm and Euler number, respectively.

4.2.1.3 Proposed SOM algorithm

In this thesis a new adaptive learning rate $\alpha(n)$ of SOM algorithm was derived. The eigenvalues of the autocorrelation matrix $\mathbf{R}(n)$ of the input is the key factor in selecting $\alpha(n)$. The eigenvalues and adaptive learning rate should be selected in an inversely proportional way. To guarantee a low QE performance and low convergence rate in case of relatively high eigenvalues, the $\alpha(n)$ should be selected with relatively small value. A relatively high value of $\alpha(n)$ can be selected to provide high convergence rate in case of relatively small eigenvalues of the autocorrelation matrix $\mathbf{R}(n)$. This, of course, might results in a high error performance as a side effect.

Recursively estimating the input's the autocorrelation matrix, $\mathbf{R}(n)$,

$$\mathbf{R}(n+1) = \beta.\mathbf{R}(n) + \mathbf{R_{xx}} \tag{4.77}$$

Here β is the forgetting factor ($0 << \beta < 1$), $\mathbf{R_{xx}} = E\{\mathbf{x}(n)\mathbf{x}^T(n)\}$, E is the expectation operator, T is the transposition operator and $\mathbf{x}(n)$ is the input. Taking expected value for both sides of (4.77),

$$\bar{\mathrm{R}}(n) = \frac{1-\beta^n}{1-\beta}\mathbf{R_{xx}} \tag{4.78}$$

Where $\bar{\mathbf{R}}(n) = E\{\mathbf{R}(n)\}$. By solving (4.76) in the steady-state ($n \to \infty$) we get,

$$\bar{\mathbf{R}}(\infty) = \frac{1}{1-\beta} \tag{4.79}$$

According to (4.79), the eigenvalues of the estimated autocorrelation matrix will increase exponentially, and at the limit they become $\frac{1}{1-\beta}$ times the original autocorrelation matrix.

Based on the same technique of (Berglund, 2010), we are proposing a new variable adaptive learning of SOM algorithm in this thesis. The weight vectors are derived in similar manner as in (Kohonen, 1982; Kohonen, 2001) as,

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) + \alpha(n).[\mathbf{x}(n) - \mathbf{w}_i(n)], \;\; n = 0, 1, \ldots \tag{4.80}$$

Where $\mathbf{w}_i(n+1)$ is defined as updating the weights.

The proposed variable adaptive learning is defined by (Ahmad et al., 2011).

$$\alpha(n) = \frac{\lambda}{1-\beta^n} \tag{4.81}$$

The idea behind (4.81) is simple. If β is large enough, at the beginning, where n is small, the term $(1-\beta^n)$ will be relatively small and, hence, $\alpha(n)$ will be relatively large which will guarantee faster convergence of the updated weights in (4.80). After a while, when n increases enough, the term $(1-\beta^n)$ will be very close to one and, hence, $\alpha(n)$ is approximately equal to λ (a small positive constant), hence, it will guarantee low error performance for the updated weights in (4.80).

5. SIMULATION RESULTS

This work includes three sections which are convex combination of RI algorithms, BPVAM, and proposed SOM algorithm. The proposed algorithms have been tested in different application to prove the performance.

The compared methods were coded in MATLAB R2010b. The experiments were run on a Core (TM) i7-3612QM CPU (2.10 GHZ) PC equipped with 8,00 GB of RAM with Windows 7 Ultimate operating system.

5.1 Convex Combination of Recursive Inverse (RI) Algorithms

Convex combination of RI algorithms has been presented in 1-D convex combination RI, and 2-D convex combination RI algorithms.

5.1.1 1- D Convex combination of RI algorithms

The combinations here have been tested in noise cancellation and system identification settings using different adaptive algorithms.

5.1.1.1 Convex combination of two RI algorithms

In this section, the performance of the proposed algorithm is compered to its theoretical steady-state (SS) MSE given in (3.47), and to the performance of the NLMS algorithm in additive white Gaussian noise (AWGN) and additive correlated Gaussian noise (ACGN) environments for the noise cancelation setting shown in Fig. 3.4.

The input signal is assumed to be a white Gaussian process with zero mean and unity variance. Simulations are implemented with a filter length $N = 16$ taps and 300 independent runs. For simplicity, the expectation operation in Eq. (3.25) can be replaced by,

$$P_{\mathbf{x}}(n) = (1-\gamma)P_{\mathbf{x}}(n-1) + \gamma \mathbf{x}^2(n) \tag{5.1}$$

Where $\mathbf{x}(n)$ is the signal to be averaged and $\gamma = 0.01$.

5.1.1.1.1 Additive white Gaussian noise

In the first experiment, the theoretical and experimental SS-MSE performances of the convex RI algorithm are compared to each other in AWGN environment. Also, the performance of the convex RI algorithm is compared to that of the convex NLMS algorithm in terms of the convergence rate and SS-MSE. The AWGN process is assumed with zero mean and variance $\sigma_v^2 = 2 \times 10^{-4}$. Simulations were performed with the following parameter: For the convex RI: $\beta_1 = 0.994$, $\mu_1 = 0.001$, $\beta_2 = 0.998$ and $\mu_2 = 0.00003$. For the convex NLMS: $\mu_1 = 0.45$ and $\mu_2 = 0.1$. Fig. 5.1 presents the performances of convex RI and NLMS algorithms. From the figure, it is noticed that the theoretical and experimental SS-MSEs of the convex RI algorithm are in match (-58 dB). On the other hand, under them same conditions, the convex RI algorithm converges to the SS-MSE faster than the NLMS algorithm by approximately 350 iteration with 5.6 dB lower MSE. Fig. 5.2 shows the evolution curves of λ for both algorithms. The evolution of λ in the case of the RI approaches its minimum value much faster than that of the NLMS algorithm, which confirms the results in Fig. 5.1.

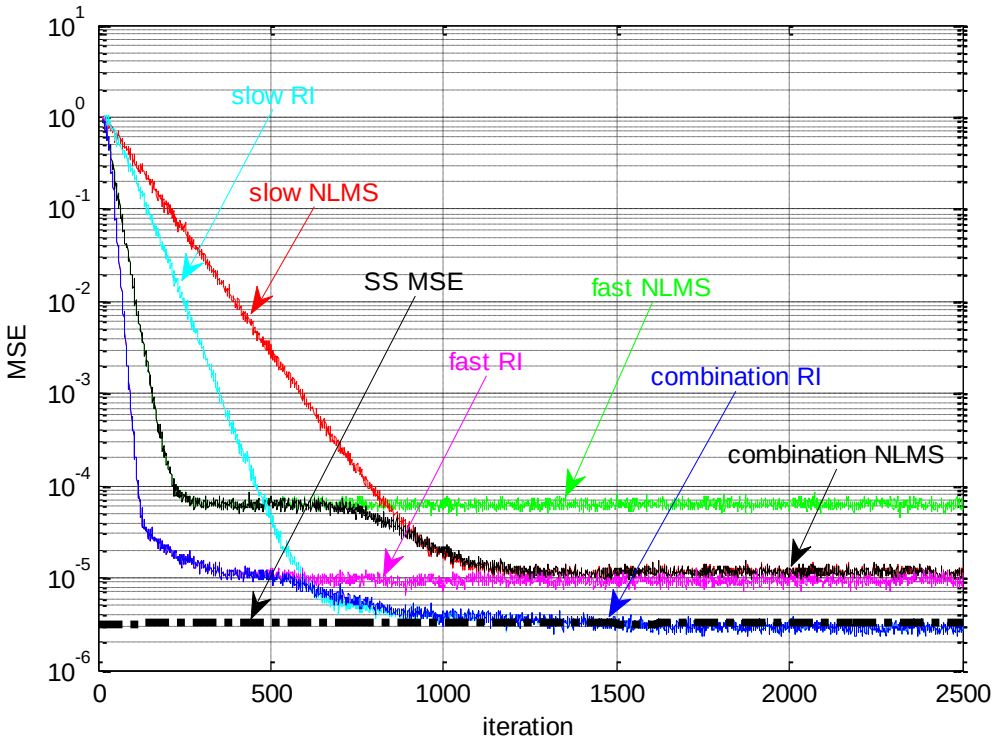


Figure 5.1. MSE combination curves of NLMS (MSE= -52.4 dB) and RI algorithms (MSE= -58 dB) in AWGN

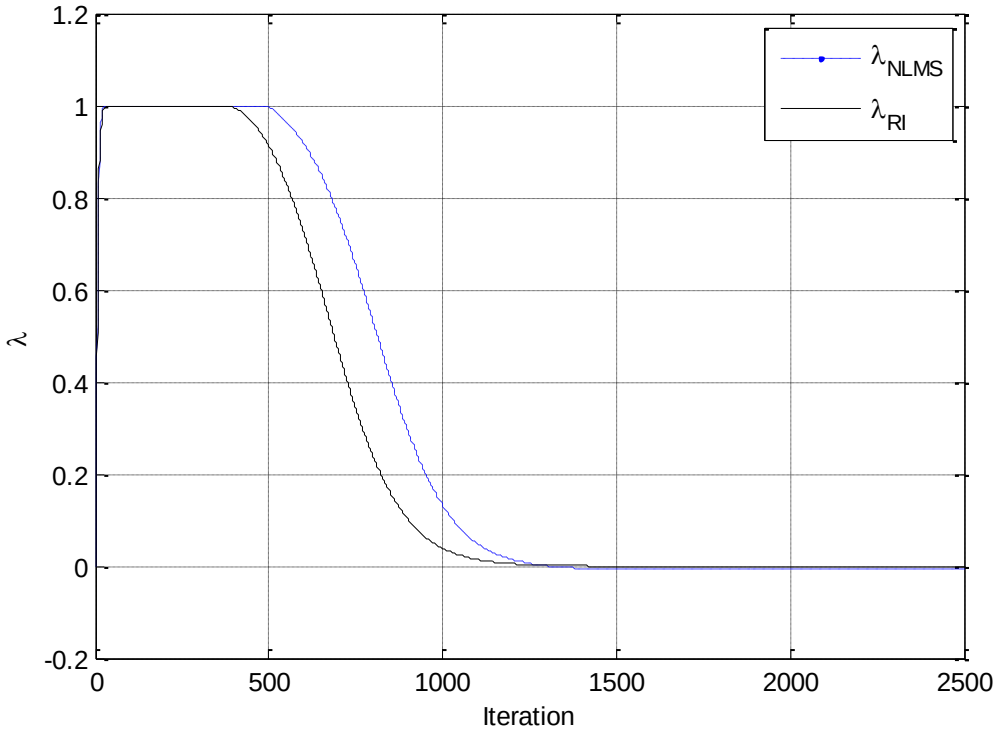


Figure 5.2. Evolution curves of λ in AWGN with 30 dB SNR

5.1.1.1.2 Additive correlated Gaussian noise

To show the effect of the noise correlation on the performances (theoretical and experimental) of the convex RI algorithm, the input signal is assumed to be corrupted by an ACGN. The ACGN is created the AR(1) process given as: $(\eta(n) = 0.9\eta(n) + \nu(n))$, where $\nu(n)$ is an AWGN process with zero mean and variance $\sigma_v^2 = 2 \times 10^{-4}$. The algorithms are simulated with the same parameters of the experiment in Section 5.1.1.1.1.

Fig. 5.3 shows that the theoretical and experimental SS errors of the convex RI are again in match. Also, the convex RI algorithm converges to 6.5 dB lower SS-MSE than the convex NLMS with almost 350 iteration faster convergence rate. Fig. 5.4 proves convex RI faster than NLMS.

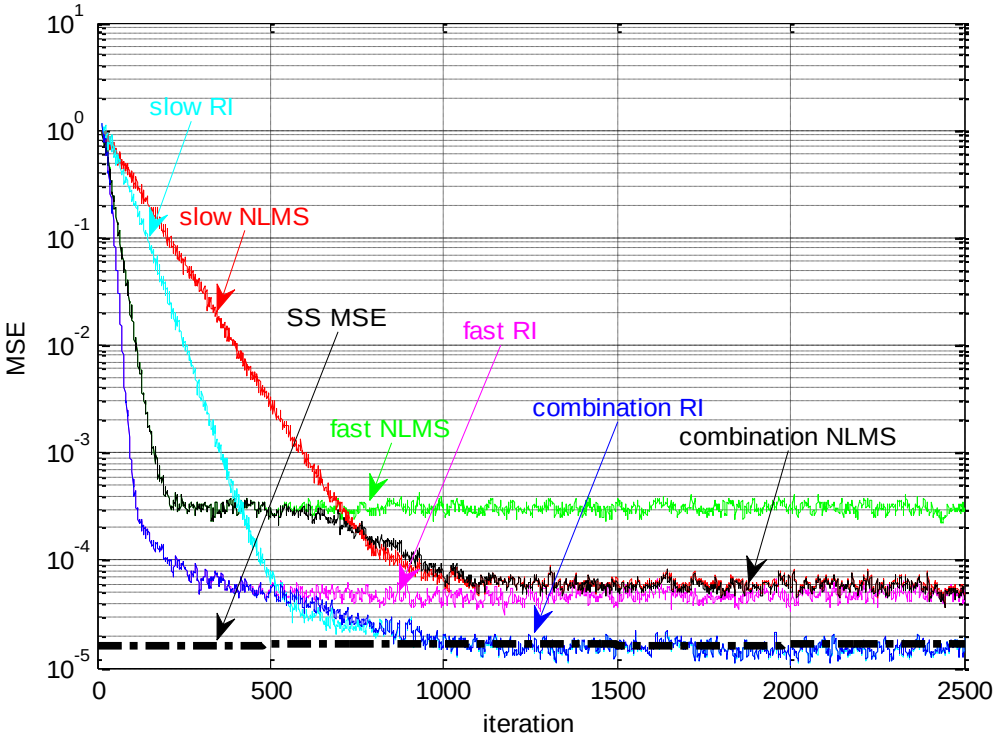


Figure 5.3. MSE combination curves of NLMS (MSE= -44.5 dB) and RI algorithms (MSE= -51 dB) in ACGN

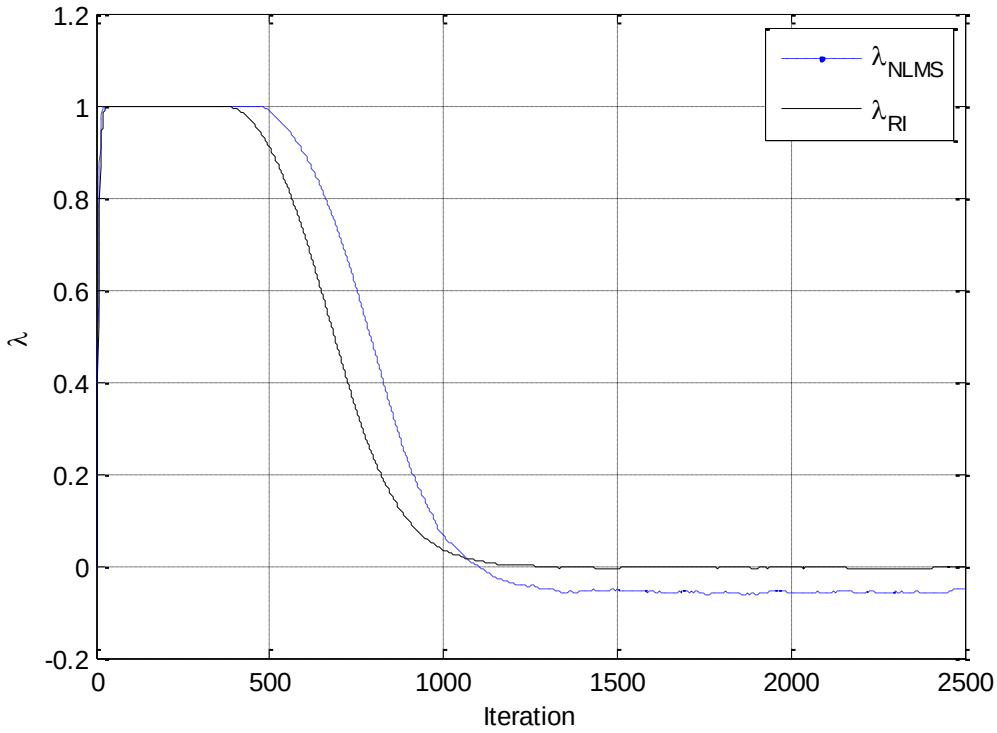


Figure 5.4. Evolution curves of λ in ACGN with 30 dB SNR

5.1.1.2 Convex combination of RI and second-order RI algorithms

Consider the combination of 2 adaptive filters in the system identification setting, as shown in Fig. 3.5.

In this thesis, in order to test the performance of the proposed algorithm under different noise environments, we compare the performances of combined RI and NLMS algorithms in a system identification setting for both AWGN and ACGN environments. The received signal was generated using a fourth-order autoregressive (AR(4)) model:

$$\mathbf{x}(n) = 1.79\mathbf{x}(n-1) - 1.85\mathbf{x}(n-2) + 1.27\mathbf{x}(n-3) - 0.41\mathbf{x}(n-4) + v_0(n), \quad (5.2)$$

Where v_0 is a white Gaussian signal with zero mean and variance $\sigma_{v_0}^2 = 0.15$. This variance value is selected in order to provide a unity power of the input signal $\mathbf{x}(n)$:

In practice, the expectation operators in (3.25) can be replaced by (5.1).

Simulations were done with the following parameters: the filter length $N = 16$ taps, and noise variance in both experiments is selected to maintain the signal-to-noise ratio (SNR) at 30 dB. All the experiments are averaged over 200 independent runs. The unknown system is assumed to be a low-pass filter with the impulse response depicted in Fig. 5.5.

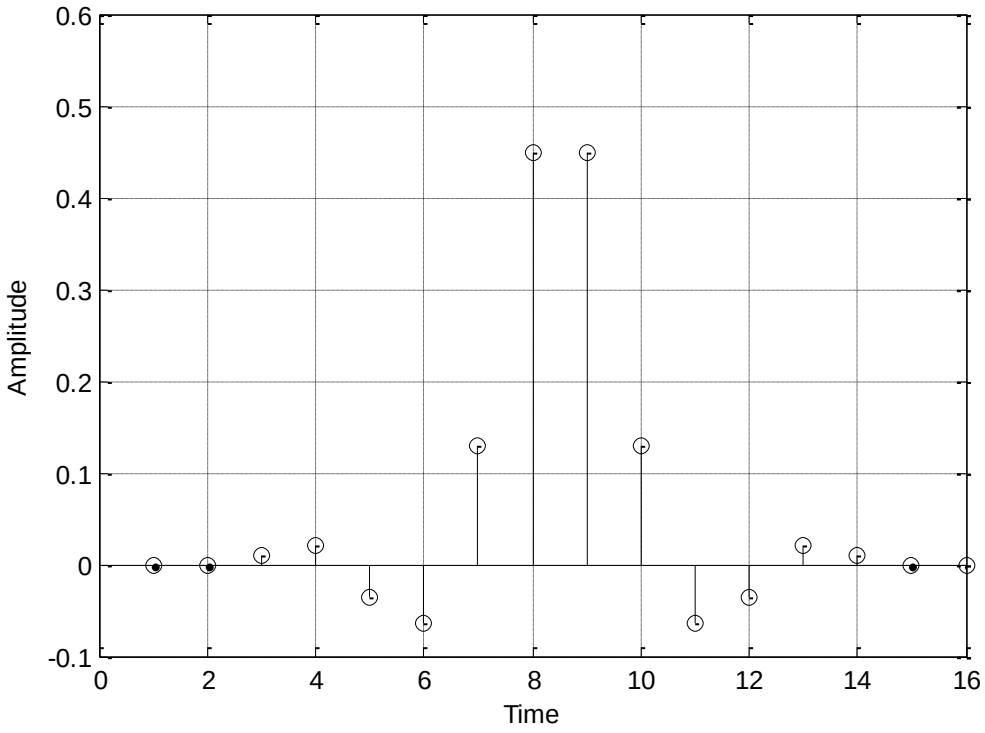


Figure 5.5. Impulse response of the unknown system

5.1.1.2.1 Additive white Gaussian noise

In this experiment, the input signal $\mathbf{x}(n)$ is assumed to be corrupted with AWGN. Simulations are done with the following parameters: for NLMS: $\mu_1 = 0.5$ and $\mu_2 = 0.1$. For RI: $\beta = 0.991$, $\mu_0 = 0.00146$. For the second-order RI: $\beta = 0.997$, $\mu_0 = 0.05$. From Fig. 5.6, we observe a fast convergence at the beginning followed by a slower second convergence with lower MSE for both algorithms. The combination curve follows the fast converging and the low MSE curves in both cases. However, the RI algorithm converges faster than the NLMS algorithm (3700 iterations faster) with 8 dB lower MSE. Fig. 5.7 shows the evolution curves of λ for both algorithms. The evolution of λ in the case of the RI approaches its minimum value much faster than that of the NLMS algorithm. This high performance of the convex RI is due to the use of the variable step size and the instantaneous estimates of the correlations which, in turn, enhance the performance of the proposed algorithm.

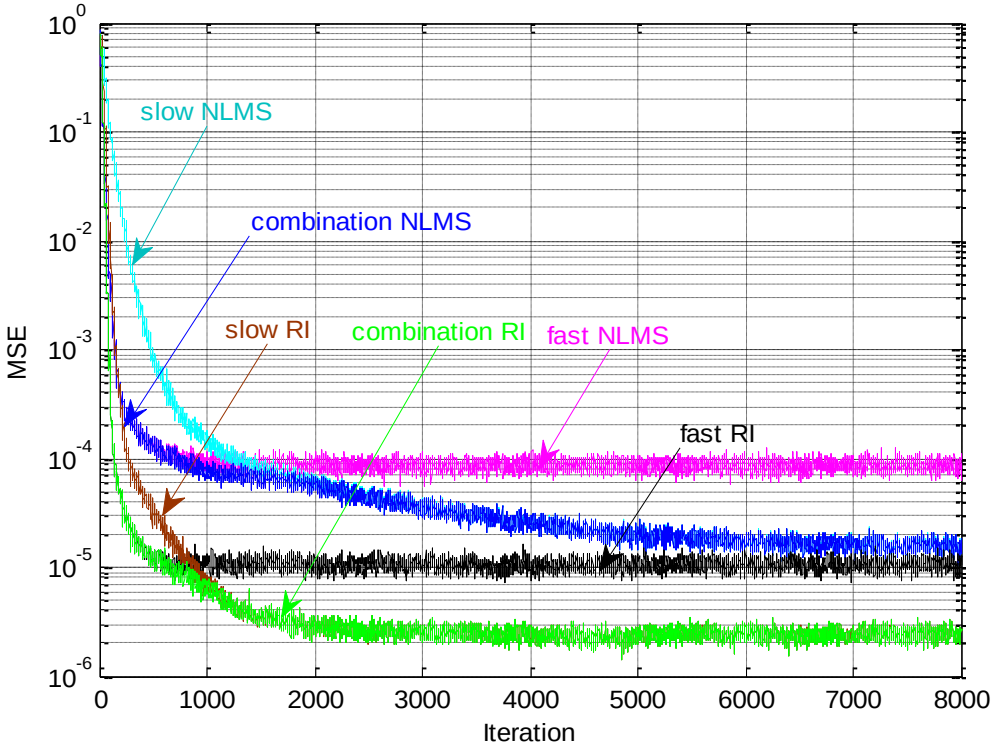


Figure 5.6. MSE combination curves of NLMS and RI algorithms in AWGN with 30 dB SNR

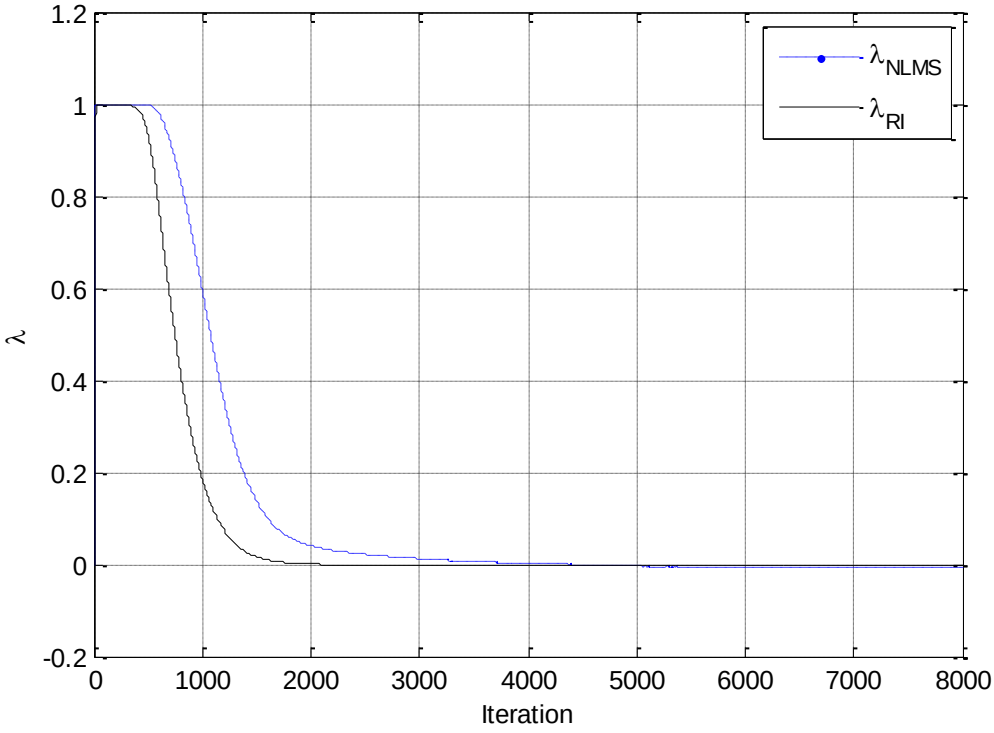


Figure 5.7. Evolution curves of λ in AWGN with 30 dB SNR

In order to check the robustness of the proposed algorithm according to the change in the SNR values, Fig. 5.8 is provided. It is easy to note that the proposed algorithm keeps a fixed difference in its MSE value with respect to the convex NLMS algorithm.

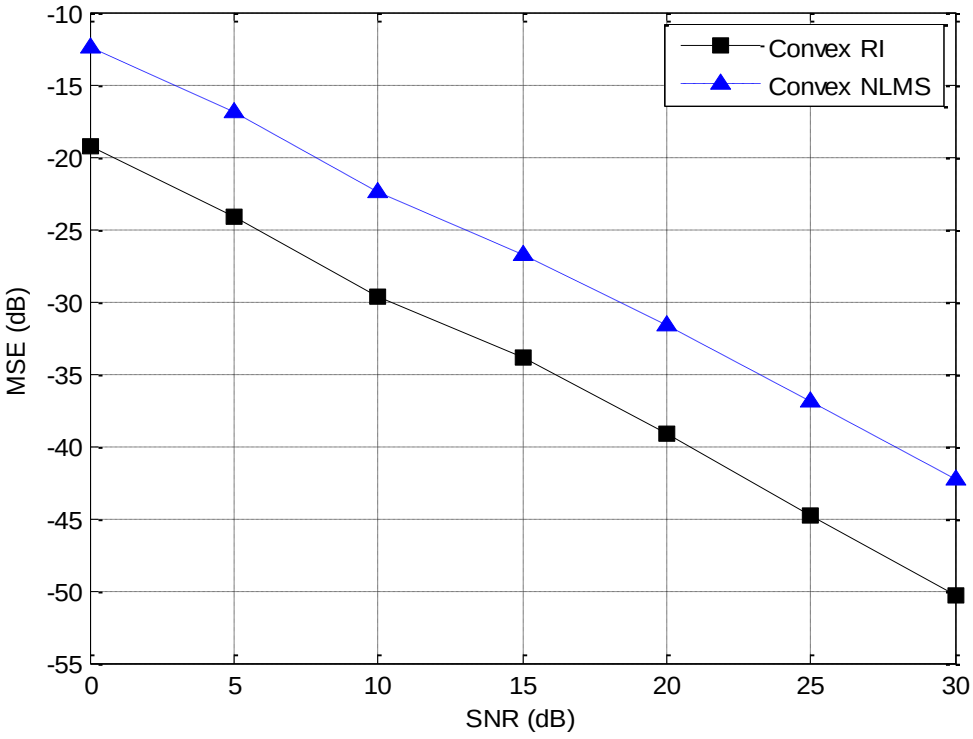


Figure 5.8. MSE for convex RI and NLMS algorithms with different SNR's in AWGN

5.1.1.2.2 Additive correlated Gaussian noise

Now, in order to investigate the performance of the proposed algorithm due to the change in noise type, the input signal $\mathbf{x}(n)$ is assumed to be corrupted with ACGN. The ACGN is created using $AR(1)\big(\eta(n) = 0.9\eta(n-1) + v(n)\big)$, where $v(n)$ is an additive white Gaussian signal with zero mean and variance that maintains 30 dB SNR. Simulations are done with the following parameters: for NLMS: $\mu_1 = 0.5$ and $\mu_2 = 0.1$. For RI: $\beta = 0.99$, $\mu_0 = 0.0015$. For the second-order RI: $\beta = 0.997$, $\mu_0 = 0.05$. From Fig. 5.9, we observe a fast convergence at the beginning followed by a slower second convergence with lower MSE for both algorithms. The combination curve follows the fast converging and the low MSE curves in both cases. However, the RI algorithm converges faster than the NLMS algorithm (850 iterations faster) with 9.5 dB lower

MSE.

From this, we note that the proposed algorithm performs better than the convex NLMS algorithm with a higher difference (9.5 dB) than the case of the AWGN process (8 dB difference). This improvement is due to the instantaneous estimate of the correlations.

Fig. 5.10 shows the evolution curves of λ for both algorithms. The evolution of λ in the case of the RI approaches its minimum value faster than that of the NLMS algorithm, which confirms the results in Fig. 5.9. On the other hand, the evolution curve of λ in the case of the NLMS algorithm fails to reach that minimum value, which means that the NLMS algorithm fails to reach the optimum MSE.

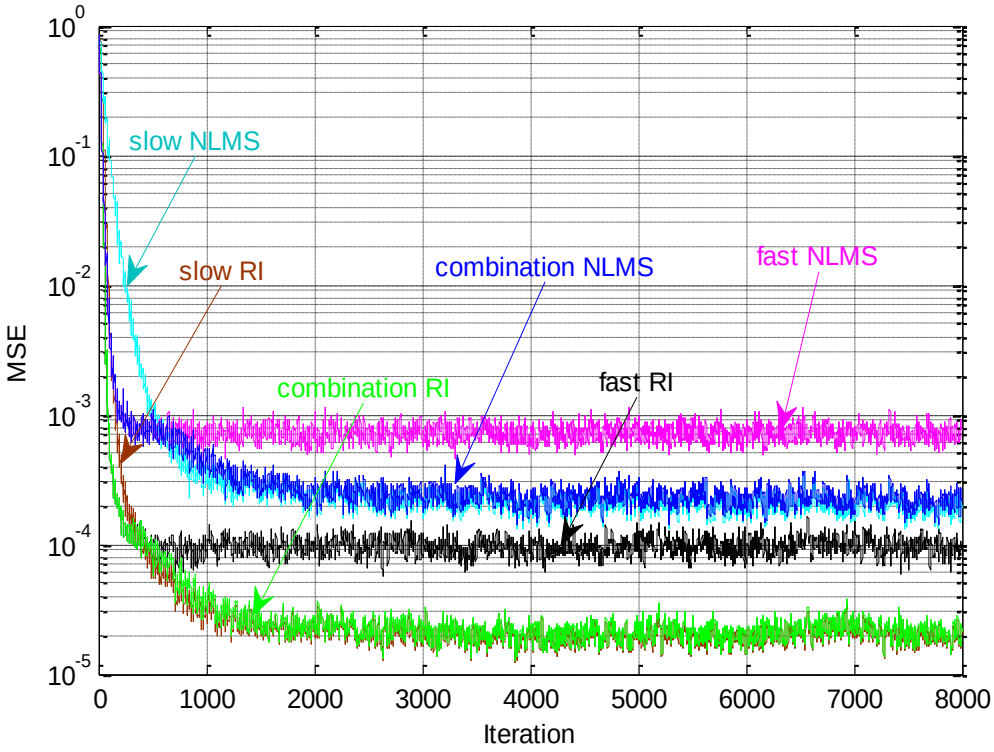


Figure 5.9. MSE combination curves of NLMS and RI algorithms in ACGN with 30 dB SNR

To investigate the performance of the proposed algorithm due to the change in the SNR values, Fig. 5.11 is provided. We note that the proposed algorithm almost keeps a fixed difference in its MSE value with respect to the convex NLMS algorithm (around 9 dB difference).

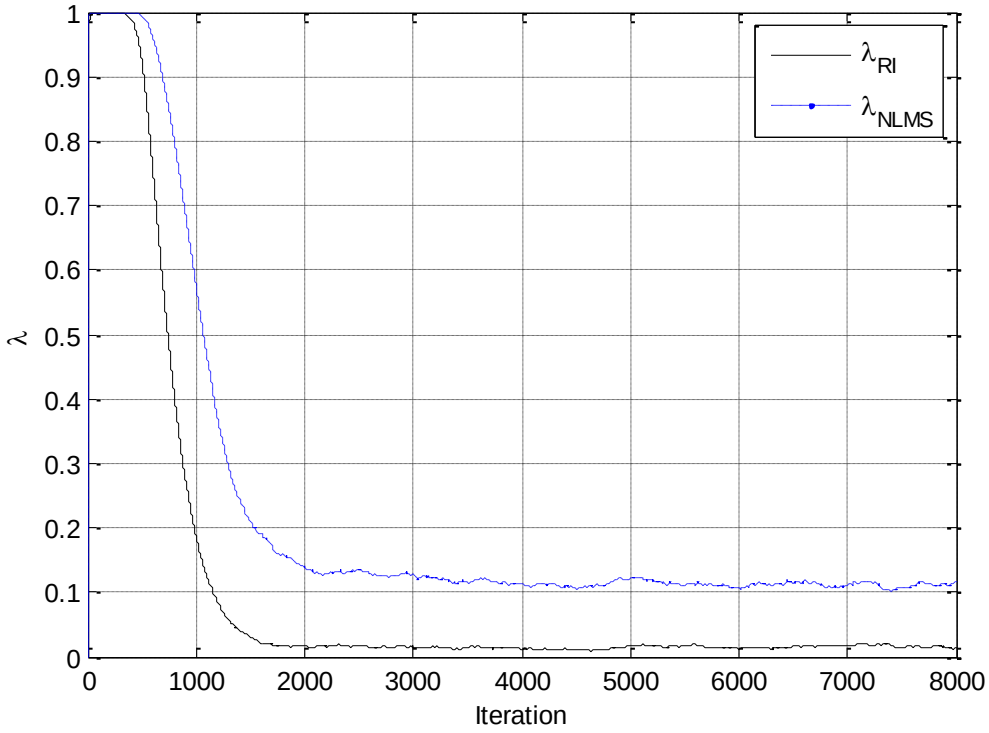


Figure 5.10. Evolution curves of λ in ACGN with 30 dB SNR

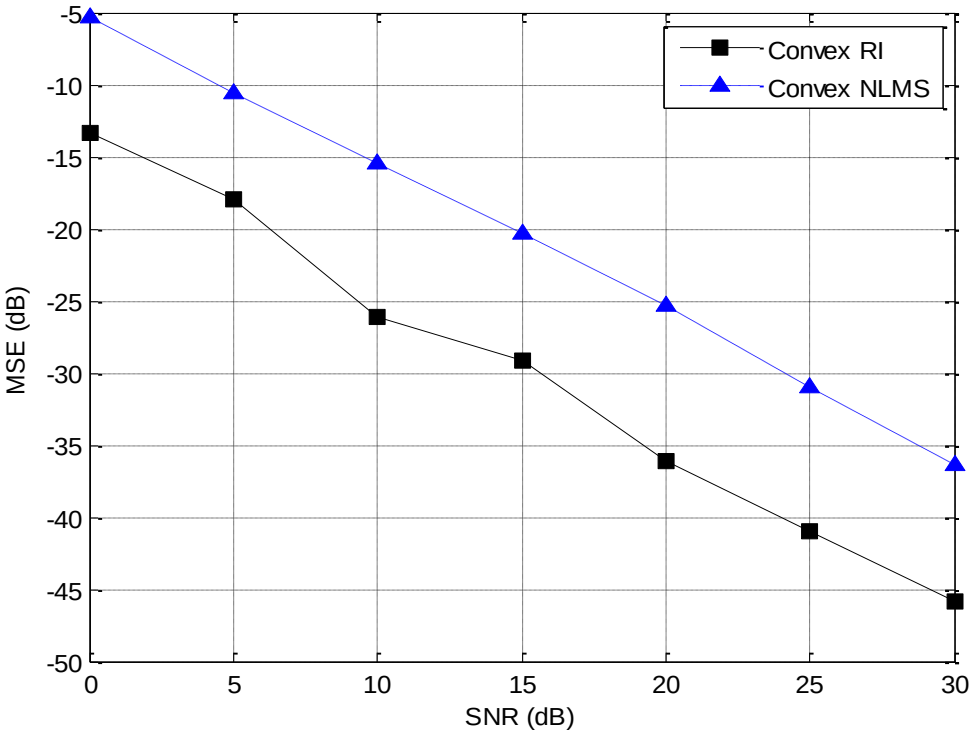


Figure 5.11. MSE for convex RI and NLMS algorithms with different SNR's in ACGN

5.1.2 2-D Convex combination of RI algorithms

5.1.2.1 Convex of two RI algorithms

In this experiment, an MRI image having breast cancer is tested. The input image is assumed to be normalized and of size 204 × 204 pixels with 150 graylevels, and corrupted by an AWGN with zero mean and variance normalized ($\sigma^2 = 0.3$). For the fast algorithm: $\beta = 0.998$ and $\mu_0 = 0.0005$. For the slow algorithm: $\beta = 0.992$ and $\mu_0 = 0.000001$.

Fig. 5.12 (a) shows the original image. Fig. 5.12 (b) shows the image corrupted by an AWGN. Fig. 5.12 (c) shows the image recovered by the fast RI algorithm. It is very easy to note the fast convergence of the algorithm (the top part of the image is relatively clear). Also, the high MSE can be figured out by the relative darkness of the image. In Fig. 5.12 (d), which shows the image recovered by the slow RI algorithm, the slow convergence and the lower MSE are figured out by the dark region at the top of the image and the relatively clearer recovered image, respectively. Fig. 5.12 (e), combines the better performances of Fig. 5.12 (c) and Fig. 5.12 (d). Hence, we can see in Fig. 5.12 (e) the clear top of the image (fast convergence) and the clearer whole image (low MSE) very easily.

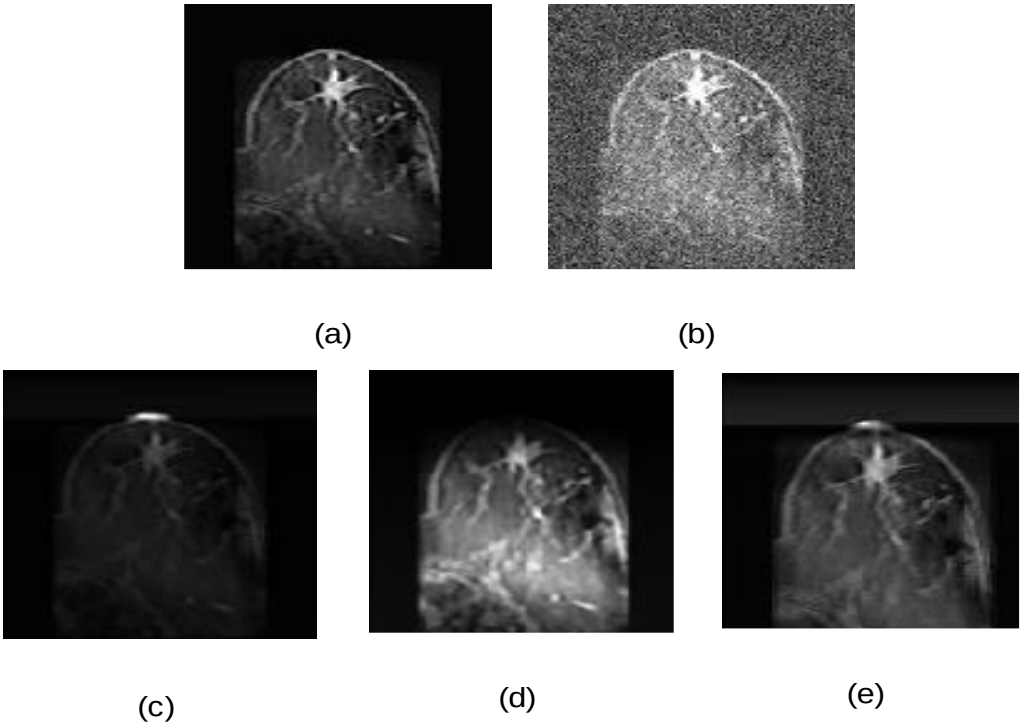


Figure 5.12. (a) Original image of breast cancer, (b) Original image corrupted by an AWGN, (c) Recovered image using the fast RI algorithm (PSNR = 30 dB), (d) Recovered image using the slow RI (PSNR= 33.5 dB), and (e) Recovered image using the convex combination of RI algorithms (PSNR= 35.3 dB)

5.1.2.2 Convex RI and second-order RI algorithms

The second experiment is done to test the performance of the proposed 2-D convex combination RI algorithm. The MRI input is of size 204×204 pixels with 150 gray levels and corrupted by AWGN in zero mean and normalized variance ($\sigma^2 = 0.3$). For the RI algorithm: $\beta = 0.991$ and $\mu_0 = 0.0015$. For the second-order RI: $\beta = 0.999$ and $\mu_0 = 0.00015$.

Fig. 5.13 (a) shows the original MRI of a brain and Fig. 5.13 (b) shows the image with AWGN. Fig. 5.13 (c) shows the recovered image using the RI algorithm. We notice that the figure is not well recovered; in other words, the MSE is relatively high. Fig. 5.13 (d) depicts the recovered image using the second-order RI algorithm. Even though the figure looks very clear, the top side of the figure is relatively dark due to the slow convergence behavior of the algorithm. Fig. 5.13 (e) combines the performance of both algorithms; i.e., the figure is as clear as that in Fig. 5.13 (d) with no dark region at the top side as of Fig. 5.13 (c).

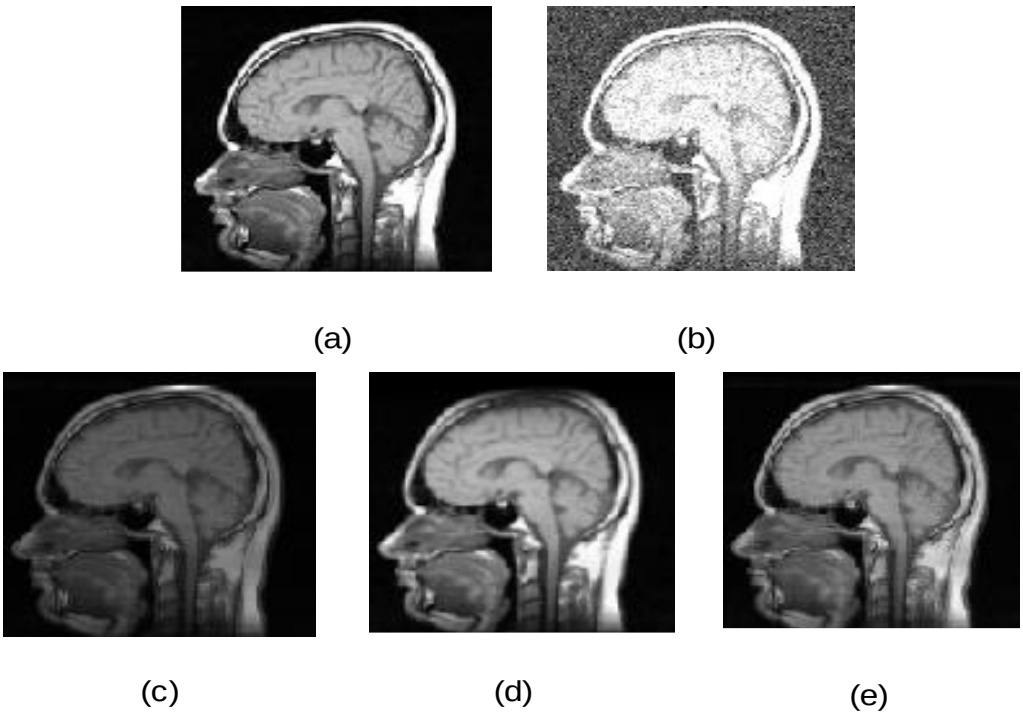


Figure 5.13. (a) Original image of brain, (b) noisy image corrupted by an AWGN, (c) recovered image by 2-D RI, (d) recovered image by 2-D 2nd order RI, and (e) recovered image by 2-D convex combination of RI algorithms

5.2 Back-propagation Algorithm with Variable Adaptive Momentum (BPVAM) Algorithm

5.2.1 XOR problem

XOR problem is one of the non-linear problems that can be perfectly solved using the BP algorithm. To minimize the cost of solving this problem and to improve the quality of the predicted output by the BP algorithm, several types of adaptive momentum techniques have been suggested in (Yu et al., 1993; Yu and Liu, 2002; Shao and Zheng, 2009). In this experiment, we demonstrate the robustness of our proposed algorithm in investigating the speed of convergence and achieving a lower misadjustment (steady-state) error compared to the conventional BP and PBPAM algorithms. The experiment has been performed using the following parameters: for neural network structure: one hidden layer with two neurons. For the BP algorithm: $\eta = 0.8$, and $\alpha = 0.0001$. For the BPAM algorithm: $\eta = 0.8$, and $\alpha = 0.000007$. For the proposed algorithm: $\eta = 0.8, \lambda = 0.0001, \beta = 0.999$.

From Fig. 5.14, we can see the effectiveness of the proposed algorithm outperforming the other algorithms by giving a higher prediction, and the output is closer to the real output than the other algorithms with a minimum misadjustment error. In Fig. 5.14, we can see the convergence curves behavior. The proposed algorithm outperforms the other algorithms in a faster convergence rate and a lower SSE.

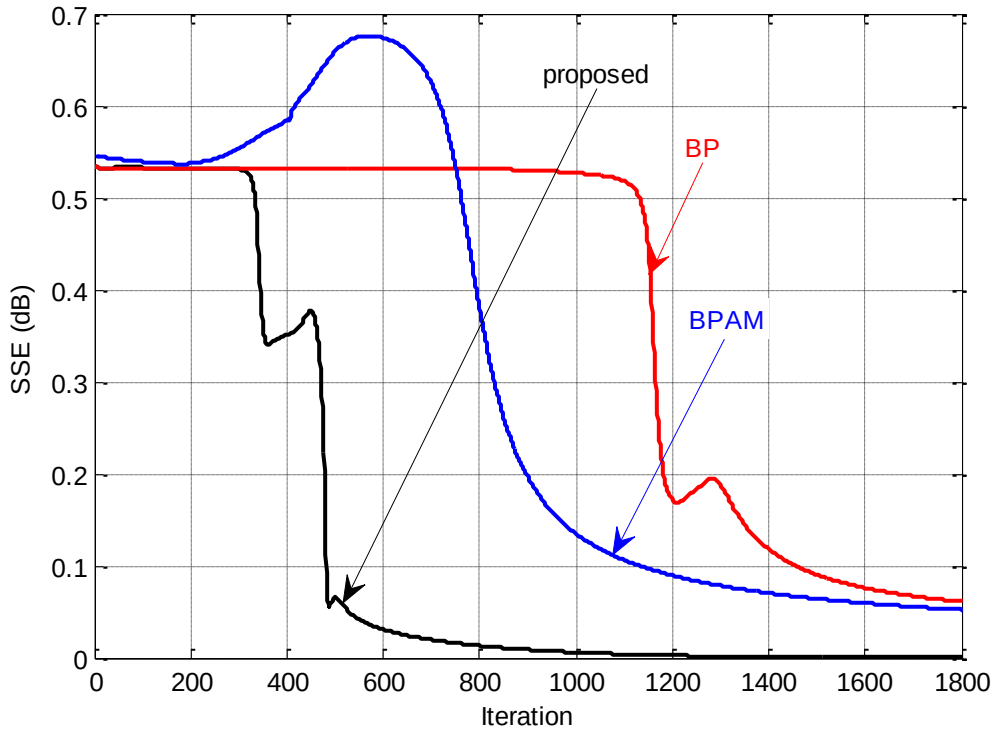


Figure 5.14. SSE results of BP, BPAM, and proposed algorithms for XOR problem

The gradient errors of the proposed method are decrease quickly to optimize the weight coefficients that guess the correct class in a shorter time (around 1000 iterations). The BPAM and BP algorithms need 1500 and 1800 iterations, respectively, to reach the optimum SSE. According to Table 5.1, the proposed algorithm needs less time to reach minimum error compared to the other algorithms.

Table 5.1. Error convergence behavior in iterations for XOR problem

Algorithm	SSE for 1000 iterations	SSE for 2000 iterations
BP	0.527	0.05391
BPAM	0.1351	0.04697
Proposed	0.00592	3.273e-005

5.2.2 Comparison of performances

The performance of the proposed method has been compared to other machine learning methods such as NB, KNN, LDA, SVM, BP, and BPAM algorithms. Table 5.2 describes the properties of each type of datasets used in this study. These datasets are summarized according to the number of samples, attributes, and classes. All the datasets has numeric features and no missing values. Each dataset has been normalized between [0,1], and 70% of each dataset was used as a training set, and 30% as testing. Also 10 fold cross validation applied for performance measurement. Normal test results and the cross validation results sperated by coma and cross validation results marked by star. As it seen in the results, cross validation have worse classification performance than normal test when %70 of data selected as train set and the rest for test set. η, α, λ and β values are determined according to experiments that presented in Appendix 1 section.

Table 5.2. The properties of used datasets

Data set	# instances	# attributes	# classes
Breast Cancer	198	32	2
Heart	297	13	5
Heart-Statlog	270	13	2
Iris	150	4	3
Lung-Cancer	32	56	3
MAGIC Gamma Telescope	19020	10	2
Wine	178	13	3

5.2.2.1 Breast cancer dataset

This dataset consists of 198 instances, 32 attributes, and 2 classes. The structure of MLP consists of 32 neurons for the input layer, 4 neurons for one hidden layer, one neuron for the output layer. The performances of the proposed BPVAM algorithm and the other supervised learning algorithms were compared using the breast cancer dataset. In this experiment, according to the accuracies results of all algorithms as seen in Table 5.3, BPVAM, SVM, and BP algorithms are slightly better than the other supervised learning algorithms.

According to SSE curves as seen in Fig. 5.15, the BPVAM algorithm is slightly better than both BP and BPAM algorithms. For each MLP structure, the following parameters were used; for the BP algorithm: $\eta = 0.9$, and $\alpha = 0.01$; for the BPAM

algorithm: $\eta = 0.9$, and $\alpha = 0.01$; and for the proposed BPVAM algorithm: $\eta = 0.9$, $\lambda = 0.0085$, $\beta = 0.992$.

Table 5.3. Comparison of accuracies by using different algorithms for breast cancer dataset

Method	Classification Accuracy (%)
K-NN (Yang et al., 2010)	76.80
NB (Kotsiantis and Pintelas, 2004)	74.58
SVM (Yang et al., 2010)	77.80
LDA (Yang et al., 2010)	71.70
BP (Kotsiantis and Pintelas, 2004)	76.13
BP (our result)	77.59, 76.44*
BPAM	75.86
BPVAM (Proposed)	77.59, 76.44*

Table 5.4. Error convergence behavior in iterations for breast cancer dataset

Algorithm	SSE for 460 iterations	SSE for 1000 iterations	CPU time (seconds)	Sensitivity	Specificity
BP	5.721	4.707, 6.295*	19.1569	90.24	47.05
BPAM	5.708	4.718	158.3878	90.00	44.44
BPVAM (Proposed)	5.452	4.678, 5.485*	31.1534	90.24	47.05

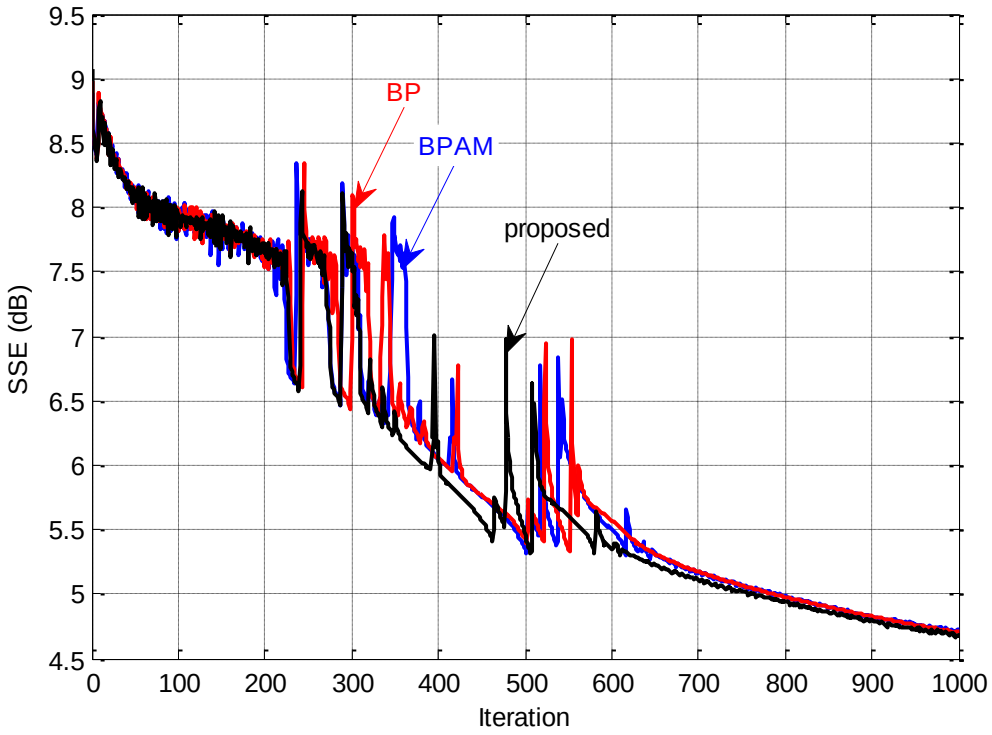


Figure 5.15. SSE results of BP, BPAM, and proposed algorithms for breast cancer dataset

The BPVAM algorithm provides the lowest SSE compared to the conventional BP and BPAM algorithms by 460 and 1000 iterations, respectively. The BPVAM does not require more processing time to be implemented than the BPAM algorithm, and gives the correct prediction within shorter time as seen Table 5.4.

5.2.2.2 Heart dataset

The heart dataset has been widely used in machine learning techniques and many methods has been modified to improve the data accuracy. This dataset consists of 297 instances, 13 attributes, and 5 classes. So the structure of the MLP consists of 13 neurons for the input layer, 13 neurons for one hidden layer, one neuron for the output layer. For each MLP structure, the following parameters were used; for the BP algorithm: $\eta = 0.03$, and $\alpha = 0.01$; for the BPAM algorithm: $\eta = 0.03$, and $\alpha = 0.9$; and for the proposed BPVAM algorithm: $\eta = 0.03, \lambda = 0.022$, $\beta = 0.995$. As seen in Table 5.5, the performances of testing results for the BPVAM, BPAM, and conventional BP algorithms are almost the same but better than the other supervised learning algorithms.

Table 5.5. Comparison of accuracies by using different algorithms for heart dataset

Method	Classification Accuracy (%)
K-NN (Yang and Kecman, 2008; Yang et al., 2010)	59.90, 58.00
NB (Saetern and Eiamkanitchat, 2014)	55.77
SVM (Yang and Kecman, 2008; Yang et al., 2010)	58.60, 56.80
LDA (Yang and Kecman, 2008; Yang et al., 2010)	53.90, 53.70
BP (our result)	61.96, 54.80*
BPAM	61.96
BPVAM (Proposed)	61.96, 54.80*

Table 5.6. Error convergence behaviourin iterations for heart dataset

Algorithm	SSE for 25 iterations	SSE for 1000 iterations	CPU time (seconds)
BP	10.3	3.969, 5.346*	40.6071
BPAM	9.93	3.958	203.0041
BPVAM (Proposed)	6.69	3.961, 5.31*	47.6271

According to SSE curves, as seen in Fig. 5.16, the BPVAM algorithm is faster than both the BP and BPAM algorithms. As seen in Table 5.6, BPVAM is 2 times better than both the conventional BP and BPAM according to SSE by 25 iterations. Regarding

the CPU processing time, the proposed BPVAM algorithm is three times faster than the BPAM algorithm, but it takes a little extra time than the conventional BP algorithm.

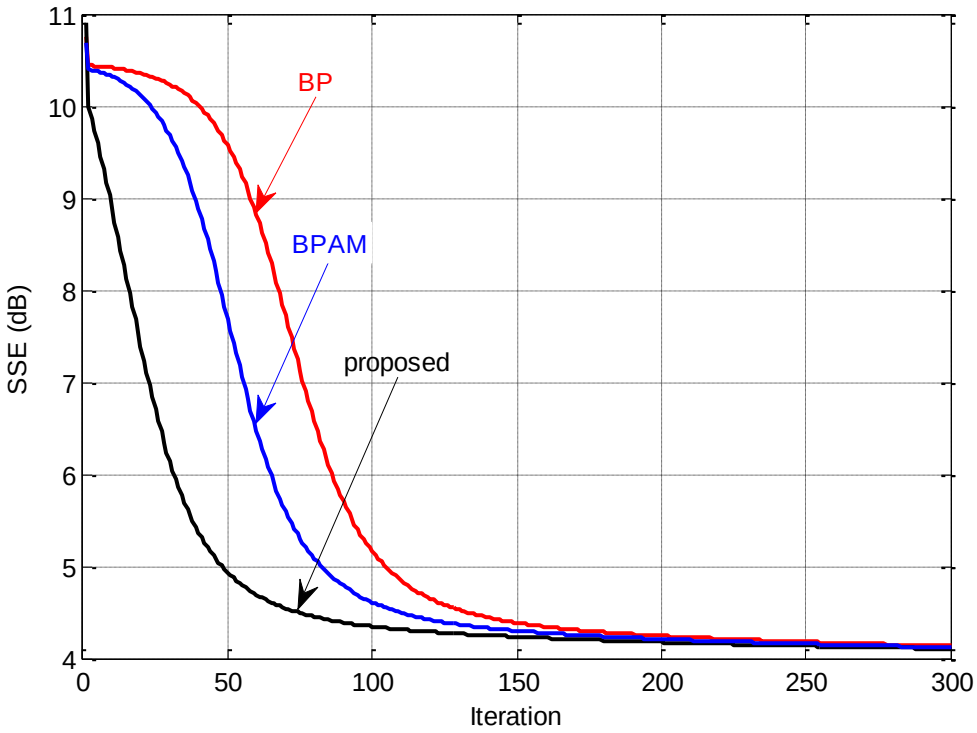


Figure 5.16. SSE results of BP, BPAM, and proposed algorithms for heart dataset

5.2.2.3 Heart-statlog dataset

This dataset consists of 270 instances, 13 attributes, and 2 classes. Thus, the structure of MLP consist 240 of 13 neurons for the input layer, 13 neurons for one hidden layer, one neuron for the output layer. For each MLP structure, the following parameters were used; for the BP algorithm: $\eta = 0.1$, and $\alpha = 0.05$; for the BPAM algorithm: $\eta = 0.1$, and $\alpha = 0.00027$; and for the proposed BPVAM algorithm: $\eta = 0.1$, $\lambda = 0.025$, $\beta = 0.996$. Table 5.7 shows that the accuracies of testing results for the conventional BP and the proposed BPVAM algorithm (88.75%) is better than those of the BPAM and the other supervised learning algorithms.

In Fig. 5.17, we can see that the BPVAM algorithm is faster than both the BP and BPAM algorithms. According to SSE curves as seen in Table 5.8, all algorithms have shown approximately the same performance. Regarding CPU processing time, the

proposed BPVAM algorithm is nearly the same as the conventional BP algorithm and three times faster than the BPAM algorithm.

Table 5.7. Comparison of accuracies by using different algorithms for heart-statlog dataset

Method	Classification Accuracy (%)
K-NN (Mendialdua et al., 2012)	86.66
NB (Mendialdua et al., 2012)	83.71
SVM (Wang et al., 2014; Wang et al., 2015)	81.48, 81.48
LDA (Mendialdua et al., 2012)	85.19
BP (Wang et al., 2014; Wang et al., 2015)	79.18, 77.41
BP (our result)	88.75, 83.36*
BPAM	88.75
BPVAM (Proposed)	88.75, 83.36*

Table 5.8. Error convergence behaviour in iterations for heart-statlog dataset

Algorithm	SSE for 20 iterations	SSE for 400 iterations	CPU time (seconds)	Sensitivity	Specificity
BP	0.141	0.111, 0.106*	19.1257	89.36	87.87
BPAM	0.139	0.11	94.0686	89.36	87.87
BPVAM (Proposed)	0.122	0.11, 0.105*	22.7605	89.36	87.87

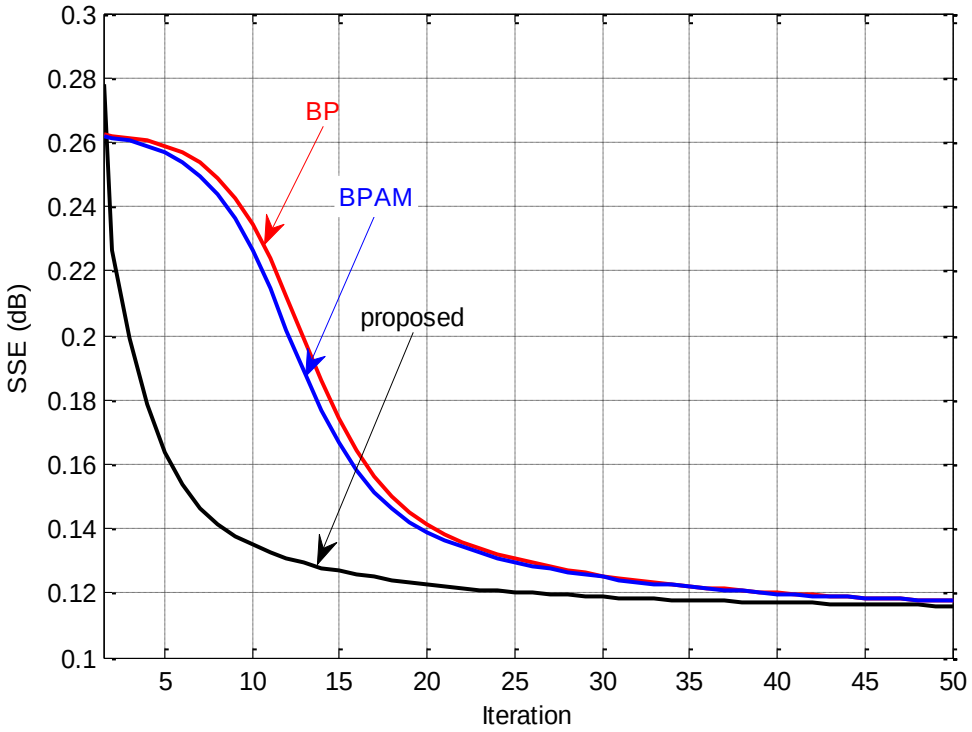


Figure 5.17. SSE results of BP, BPAM, and proposed algorithms for heart-statlog dataset

5.2.2.4 Iris dataset

This dataset consists of 150 instances, 4 attributes, and 3 classes. Thus, the structure of MLP consists of 4 neurons for the input layer, 4 neurons for one hidden layer, 3 neurons for the output layer. For each MLP structure, the following parameters were used; for the BP algorithm:$\eta = 0.05$, and $\alpha = 0.001$; for the BPAM algorithm: $\eta = 0.05$, and $\alpha = 0.0003$; and for the proposed BPVAM algorithm: $\eta = 0.05$, $\lambda = 0.02, \beta = 0.998$. As seen in Table 5.9, the accuracies of testing results for the conventional BP, BPAM, and the proposed BPVAM is 100 % which are better than the other supervised learning algorithms used previously in literature.

Table 5.9. Comparison of accuracies by using different algorithms for iris dataset

Method	Classification Accuracy (%)
K-NN (Yang and Kecman, 2008; Yang et al., 2010)	96.70, 94.90
NB (Wickramasinghe et al., 2005; Huang et al., 2011)	93.33, 96.40
SVM (Wang et al., 2014; Wang et al., 2015; Yang and Kecman, 2008; Wickramasinghe et al., 2005)	96.00, 96.00, 98.00, 93.33
LDA (Yang and Kecman, 2008; Yang et al., 2010)	98.00, 97.90
BP (Wang et al., 2014; Wang et al., 2015)	96.60, 97.33
BP (our result)	100, 93.33*
BPAM	100
BPVAM (Proposed)	100, 94.00*

Table 5.10. Error convergence behaviour in iterations for iris dataset

Algorithm	SSE for 80 iterations	SSE for 2000 iterations	CPU time (seconds)
BP	2.24	0.184, 0.188*	28.4858
BPAM	1.99	0.18	68.8120
BPVAM (Proposed)	0.2534	0.18, 0.185*	37.8458

According to the SSE curves see Fig. 5.18, the BPVAM algorithm has better excess steady-state curve and shorter time (about 80 iterations) than the other algorithms. Regarding the CPU processing time, the conventional BP algorithm is faster than both the BPAM and the proposed BPVAM algorithms. But the BPVAM algorithm is faster than the BPAM algorithm as shown in Table 5.10.

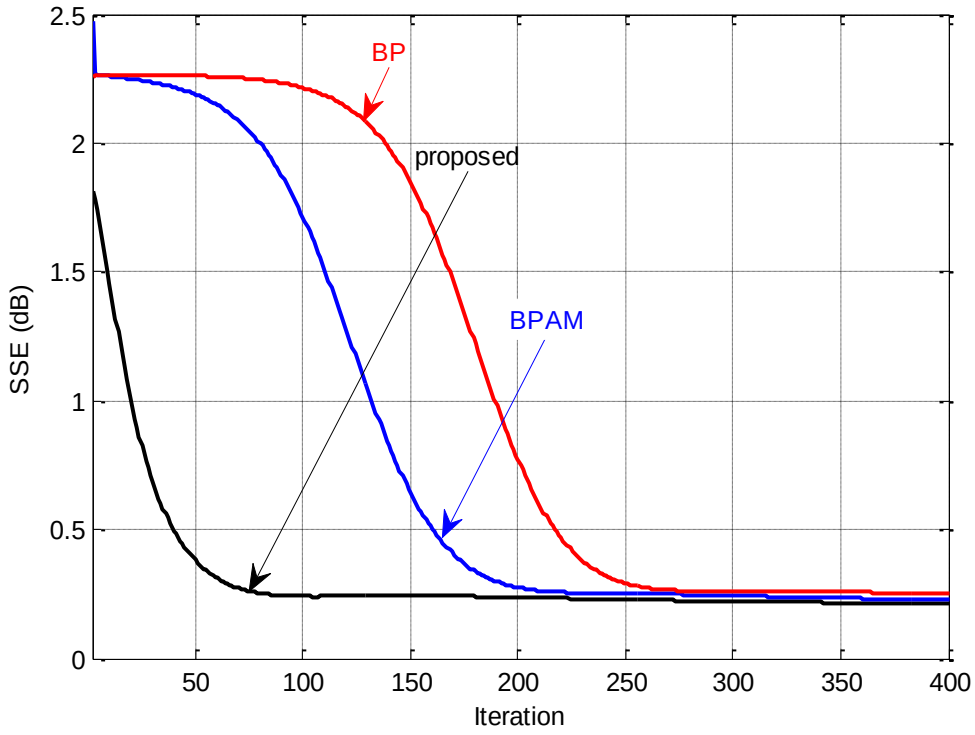


Figure 5.18. SSE results of BP, BPAM, and proposed algorithms for iris dataset

5.2.2.5 Lung-cancer dataset

In literature, there are many studies on the lung-cancer dataset. But researchers have not obtained high accuracy yet by using this dataset. This dataset consists of 32 instances, 56 attributes, and 3 classes. Thus, the structure of MLP consists of 56 neurons for the input layer, 20 neurons for one hidden layer, and one neuron for the output layer. For each MLP structure, the following parameters were used; for the BP algorithm: $\eta = 0.1$, and $\alpha = 0.05$; for the BPAM algorithm: $\eta = 0.1$, and $\alpha = 0.00001$; and for the proposed BPVAM algorithm: $\eta = 0.1$, $\lambda = 0.005$, $\beta = 0.998$.

The proposed BPVAM algorithm is compared to the BP and BPAM algorithms, and other supervised learning algorithms in literature. The testing results show that the proposed algorithm with the BP and BPAM algorithms have the highest accuracy rate compared to the other methods, as illustrated in Table 5.11.

The convergence speed of the proposed algorithm clearly shows the efficiency of the proposed momentum in enabling the proposed algorithm to outperform the conventional BP and BPAM algorithms as seen in Fig. 5.19. The proposed algorithm converges to the lowest SSE in almost 100 iterations. On the other hand, the other

algorithms need more than 500 iterations to reach the optimum SSE. Moreover, as seen in Table 5.12, the CPU processing time for the conventional BP algorithm is less than both the BPAM and the proposed BPVAM algorithms. But the BPVAM algorithm requires approximately 8 times less processing time than the BPAM algorithm.

Table 5.11. Comparison of accuracies by using different algorithms for lung-cancer dataset

Method	Classification Accuracy (%)
NB (Saetern and Eiamkanitchat, 2014)	59.37
SVM (Wang et al., 2014; Wang et al., 2015)	50.00, 50.00
BP (Wang et al., 2014; Wang et al., 2015)	44.37, 43.75
BP (our result)	60.00, 65.00*
BPAM	60.00
BPVAM (Proposed)	60.00, 65.00*

Table 5.12. Error convergence behaviour in iterations for lung-cancer dataset

Algorithm	SSE for 170 iterations	SSE for 1000 iterations	CPU time (seconds)
BP	0.561	0.0063, 0.0072*	13.3849
BPAM	0.624	0.006	132.4760
BPVAM (Proposed)	0.183	0.0054, 0.0065*	18.3145

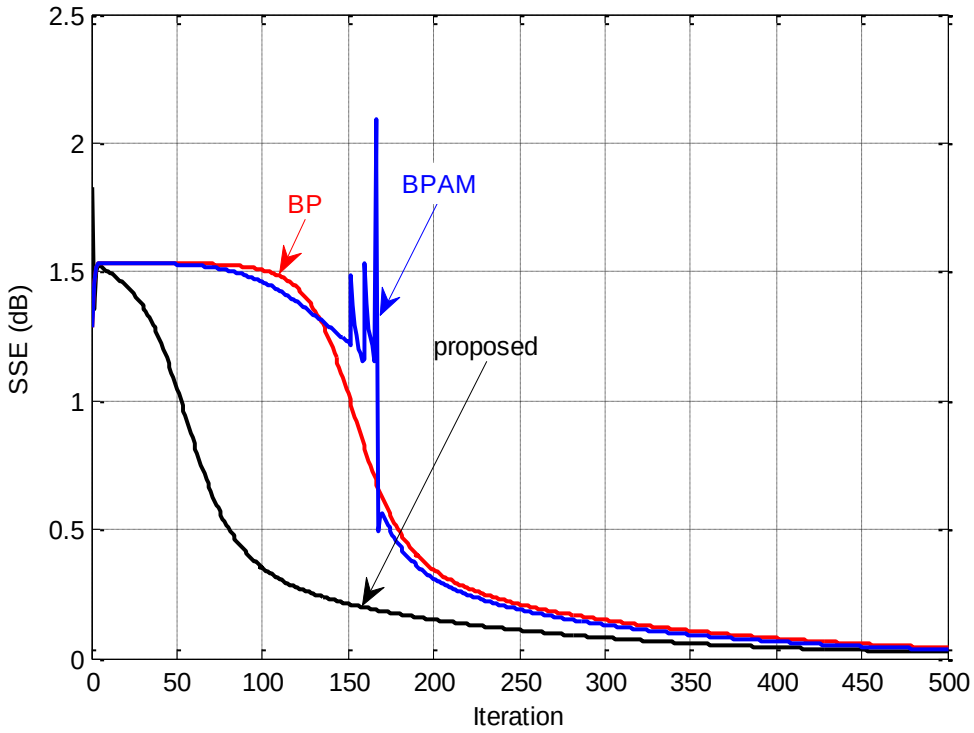


Figure 5.19. SSE results of BP, BPAM, and proposed algorithms for lung-cancer dataset

5.2.2.6 MAGIC Gamma telescope dataset

Our above simulations ensured that the proposed BPVAM has high improvement compared to the BP and BPAM algorithms, and other supervised learning algorithms. To compare its performance to the other algorithms in big size datasets, we selected the MAGIC Gamma Telescope dataset. This dataset consists of 19020 instances, 10 attributes, and 2 classes. Thus, the structure of MLP consists of 10 neurons for the input layer, 5 neurons for one hidden layer, and one neuron for the output layer. For each MLP structure, the following parameters were used; for the BP algorithm:$\eta = 0.7$, and $\alpha = 0.001$; for the BPAM algorithm: $\eta = 0.7$, and $\alpha = 0.06$; and for the proposed BPVAM algorithm: $\eta = 0.7$, $\lambda = 0.025$, $\beta = 0.99$.

Table 5.13 shows that the accuracy of the testing results for the proposed BPVAM algorithm (85.80%) is better than those of the conventional BP, BPAM and the other supervised learning algorithms.

Table 5.13. Comparison of accuracies by using different algorithms for MAGIC gamma telescope dataset

Method	Classification Accuracy (%)
K-NN (Huang et al., 2011; Mendialdua et al., 2012; Barnabe-Lortie et al., 2015)	76.58, 82.57, 77.89
NB (Huang et al., 2011)	72.71
SVM (Huang et al., 2011; Barnabe-Lortie et al., 2015)	78.91, 78.98
LDA (Huang et al., 2011)	78.15
BP (our result)	85.00
BPAM	85.50
BPVAM (Proposed)	85.80

Table 5.14. Error Convergence Behaviour in Iterations for MAGIC gamma telescope dataset

Algorithm	SSE for 200 iterations	SSE for 1000 iterations	CPU time (seconds)	Sensitivity	Specificity
BP	1.282	1.127	1.5907e+003	88.68	78.32
BPAM	1.292	1.125	5.2759e+003	87.99	80.62
BPVAM (Proposed)	1.207	1.113	1.7688e+003	88.12	80.92

In Fig. 5.20, we can see that the BPVAM algorithm is faster than the conventional BP and BPAM algorithms. According to SSE as seen in Table 5.14, the proposed BPVAM algorithm outperforms the other algorithms, where it keeps lower SSE until the end of the process. Regarding CPU processing time, the proposed BPVAM algorithm is nearly the same as the conventional BP algorithm and three times faster than the BPAM algorithm.

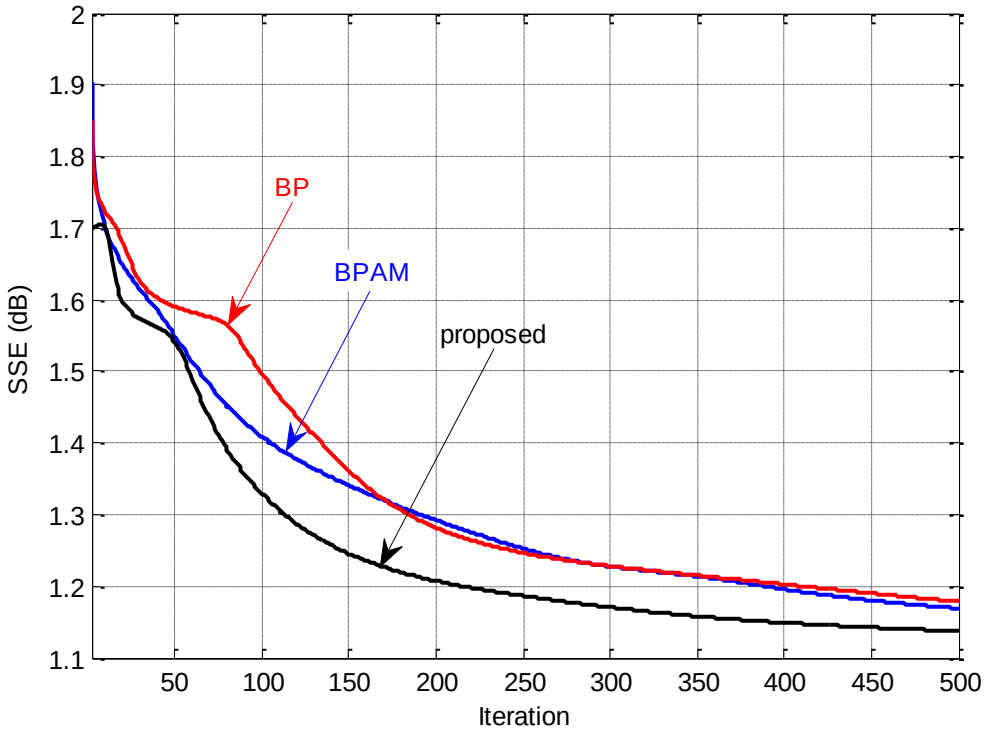


Figure 5.20. SSE results of BP, BPAM, and proposed algorithms for MAGIC gamma telescope dataset

5.2.2.7 Wine dataset

This well-known dataset recorded by UCI repository consists of 178 instances, 13 attributes, and 3 classes. Thus, the structure of MLP consists of 13 neurons for the input layer, 7 neurons for one hidden layer, and one neuron for the output layer. For each MLP structure, the following parameters are used; for the BP algorithm: $\eta = 0.9$, and $\alpha = 0.01$; for the BPAM algorithm: $\eta = 0.9$, and $\alpha = 0.00001$; and for the proposed BPVAM algorithm: $\eta = 0.9$, $\lambda = 0.0002$, $\beta = 0.9998$. As seen in Table 5.15, the proposed algorithm provides the highest accuracy (100 %) among the other algorithms.

According to SSE results as seen in Fig. 5.21, the proposed BPVAM algorithm provides better performance than the conventional BP and the BPAM algorithms. As seen in Table 5.16, the convergence behavior of the proposed algorithm is close to that of conventional BP algorithm, but approximately it is 3 times faster than the BPAM algorithm.

Table 5.15. Comparison of accuracies by using different algorithms for wine dataset

Method	Classification Accuracy (%)
K-NN (Yang and Kecman, 2008; Yang et al., 2010)	97.80, 96.10
NB (Wickramasinghe et al., 2005; Macskassy, 2011)	90.57, 96.50
SVM (Wang et al., 2014; Wang et al., 2015; Yang and Kecman, 2008; Wickramasinghe et al., 2005)	95.31, 95.51, 98.90, 92.45
LDA (Yang and Kecman, 2008; Yang et al., 2010)	98.90, 98.50
BP (Wang et al., 2014; Wang et al., 2015; Sarangi et al., 2013)	26.29, 26.97, 95.39
BP (our result)	98.11, 97.71*
BPAM	98.11
BPVAM (Proposed)	100, 98.33*

Table 5.16. Error Convergence Behaviour in Iterations for wine dataset

Algorithm	SSE for 200 iterations	SSE for 500 iterations	CPU time (seconds)
BP	0.81	0.186, 0.144*	11.1229
BPAM	0.73	0.135	40.9503
BPVAM (Proposed)	0.54	0.133, 0.131*	12.7453

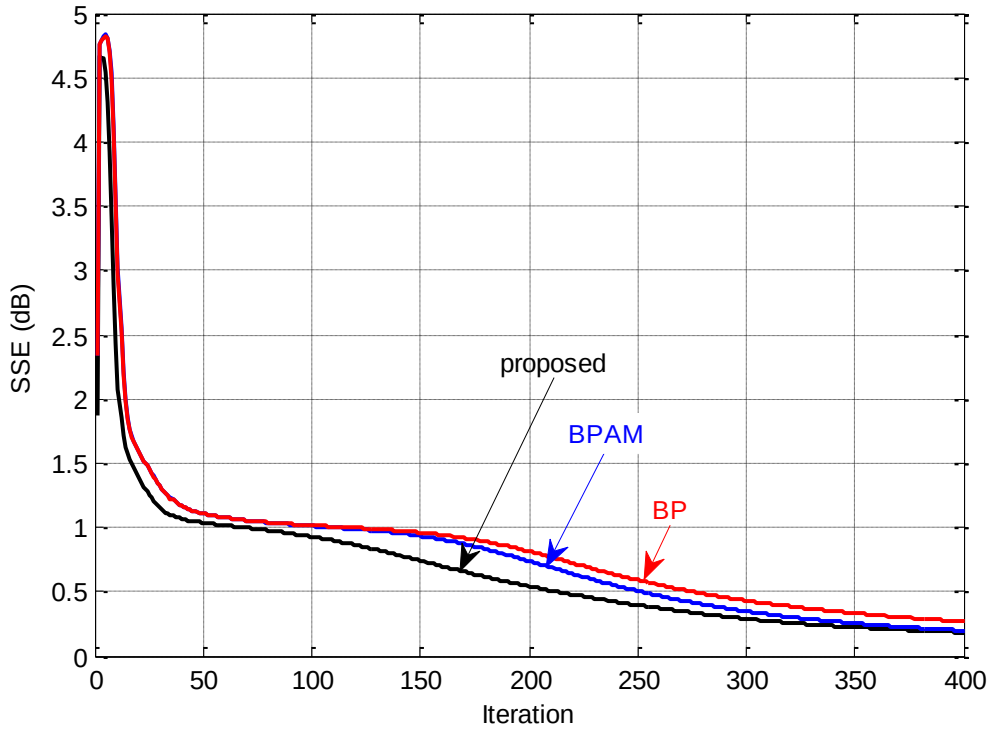


Figure 5.21. SSE results of BP, BPAM, and proposed algorithms for wine dataset

5.3 Proposed SOM Algorithm

Extensive experiments were conducted in this section to investigate the performance of the proposed SOM using a different datasets and scenarios. It was compared against the conventional SOM, GF-SOM, and PLSOM algorithms. Comparisons were done using accuracy, number of iteration, quantization error (QE), topology error (TE), and CPU time consumption. QE values were recorded over 50 iterations in all experiments for all algorithms and all datasets. Table 5.17, shows the detailed description of the used datasets in this study such as the number of samples, attributes, and classes. These datasets were collected from UCI and KEEL repository. All datasets values were normalized using Min-Max normalization between 0 and 1. All the datasets divided in to 70% training set and 30% testing set. Moreover, for all algorithms, experiments were conducted separately to select their optimal parameters. The GF-SOM parameters of (Wang et al., 2015), are lr: 0.3, map height: 6, map width: 8, map neighborhood function: Gaussian, and initial neighborhood size: 8.

$$QE = \frac{\sum_{i=1}^{n} d(\|\mathbf{w}_c(n) - \mathbf{x}(n)\|)}{n} \tag{5.3}$$

where QE is the average distance between the inputs and winner neuron (BMU).

$$TE = \frac{\sum_{i=1}^{n} u(\mathbf{x}_i)}{n} \tag{5.4}$$

where $u(\mathbf{x}_i)$ is 1 if the current and next best matching units (BMUs) are not adjacent neurons, otherwise $u(\mathbf{x}_i)$ is 0.

Values of δ_α, δ_σ, β and λ are determined according to experiments that presented in Appendix 2 section.

Table 5.17. Summary of used datasets from UCI repository

Data set	# instances	# attributes	# classes
Appendicitis	106	7	2
Balance	625	4	3
Wisconsin Breast	699	9	2
Dermatology	366	33	6
Ionosphere	351	34	2
Iris	150	4	3
Sonar	208	60	2
Wine	178	13	3

5.3.1 Appendicitis dataset

To evaluate the performance of the proposed SOM against other algorithms, we used Appendicitis dataset. The dataset consist of 106 instances, each with 7 attributes and 2 classes (patient has appendicitis (class label 1) or not (class label 0)). The structure of the used Kohonen maps consists of 7 neurons for the input layer with 2-D grid of 7×2 neurons in competitive layer. In this experiment, the used algorithms were implemented under the following parameters: for the conventional SOM algorithm, $\delta_\alpha = 0.55$, for GF-SOM, $\delta_\alpha = 0.5$, and $\delta_\sigma = 0.001$, for the PLSOM algorithm, $\beta = 1.14$, and for the proposed SOM, $\lambda = 0.00062$, and $\beta = 0.994$.

Table 5.18 shows that the classification accuracy of the PLSOM and proposed SOM algorithms are much better than other algorithms including algorithms in (Shao and Zheng, 2009; Shao and Zheng, 2011). However, the proposed SOM obtained that accuracy with only 20 iterations while PLSOM needed 50 iteration to reach that accuracy.

Furthermore, comparing the QE of all algorithms shows that the QE of proposed SOM is much less than the others. It also converges faster than the other algorithms as shown in Fig. 5.22. Regarding topology preserving, Fig. 5.23, shows that the proposed SOM is efficient in preserving the topology of the map. The preservation of the map started early in iterations 3, while PLSOM started at 7^{th} iteration. Moreover, the proposed SOM algorithm has CPU processing time close to the conventional SOM (~1.5 msec/itr), which is lower than that of the PLSOM algorithm (~20 msec/itr).

Table 5.18. Performances comparison of the conventional SOM, GF-SOM, PLSOM and proposed algorithms for appendicitis dataset

Appendicitis Dataset	Accuracy (%)	# iteration	QE	CPU time (msec/itr)
GF-SOM (Wang et al., 2014; Wang et al., 2015)	73.87, 75.47	1000	–	–
GF-SOM (our)	81.25	50	0.1443	56.16
conventional SOM	81.25	50	0.1441	52.1
PLSOM	90.63	50	0.1714	73.01
Proposed SOM	90.63	20	0.1408	53.67

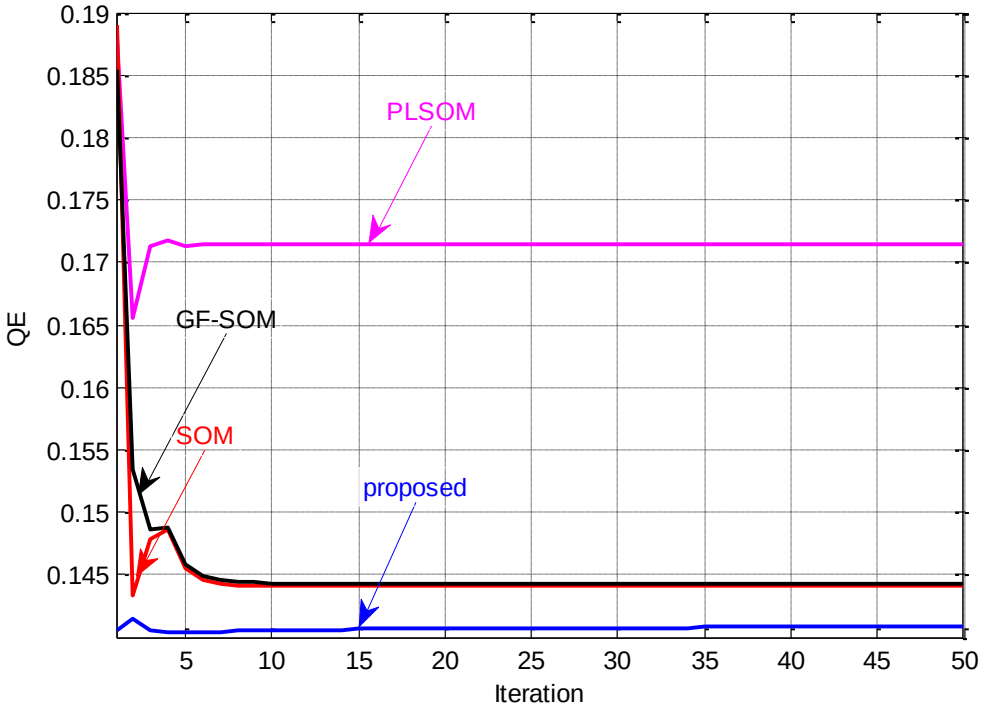


Figure 5.22. QE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Appendicitis dataset

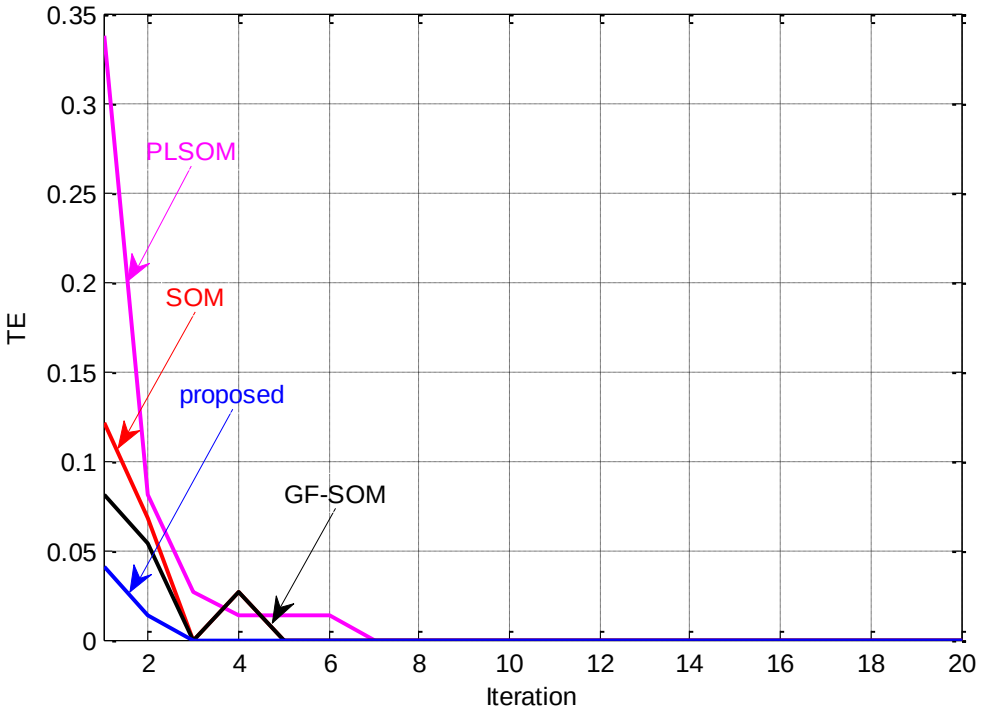


Figure 5.23. TE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Appendicitis dataset

5.3.2 Balance dataset

In the 2nd experiment the performances of all algorithms are compared using Balance dataset. The Balance dataset consists of 625 instances with 4 attributes and 3 classes (left, balanced and right). The structure of Kohonen maps of this dataset consists of 4 neurons for the input layer with 2-D grid of 4×3 neurons in competitive layer. In this experiment the used algorithms were implemented under the following parameters: for the conventional SOM algorithm, $\delta_\alpha = 0.15$, for GF-SOM, $\delta_\alpha = 0.8$, and $\delta_\sigma = 0.0085$, for the PLSOM algorithm, $\beta = 2$, and for the proposed SOM, $\lambda = 0.003$ and $\beta = 0.99$.

Table 5.19 shows that the classification accuracy of the proposed SOM algorithm is much better than other conventional SOM and PLSOM algorithms and comparable to performances of algorithms in (Wang et al., 2014; Wang et al., 2015). Also, algorithms needed from 45 to 50 iterations reach their best accuracies.

Furthermore, comparing the QE of all algorithms shows that the proposed SOM reaches better QE (9^{th} iteration) but it fails in maintaining this low error and starts to get higher QE with more iterations. The other algorithms got higher QE but converged after nearly 10 iterations as shown in Fig. 5.24. Regarding topology preserving, Fig. 5.25 shows that the proposed SOM same as other algorithms suffered from fluctuations before it reached its steady state after 14 iterations. PLSOM, on the other hand, reached it at his 8^{th} iteration. This can be justified by the way the algorithms are clustering the given data. While some algorithms keep searching for new clusters at each iteration, other algorithms are fixing the clusters at first iterations. Algorithms which keep searching for more clusters will show some fluctuations on TE before reaching zero TE as shown in Fig. 5.25. This on the other hand helps to improve accuracy of that algorithm. Moreover, the proposed SOM algorithm has CPU processing time very close to the conventional SOM (~0.30 msec/itr), which is much lower than that of the PLSOM algorithm (~100 msec/itr).

Table 5.19. Performance comparison of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Balance dataset

Balance Dataset	Accuracy (%)	# iteration	QE	CPU time (msec/itr)
GF-SOM (Wang et al., 2014; Wang et al., 2015)	73.04, 73.44	1000	–	–
GF-SOM (our)	63.10	50	0.239	59.952
conventional SOM	61.50	50	0.242	57.72
PLSOM	63.10	50	0.227	158.5
Proposed SOM	75.94	45	0.233	58.03

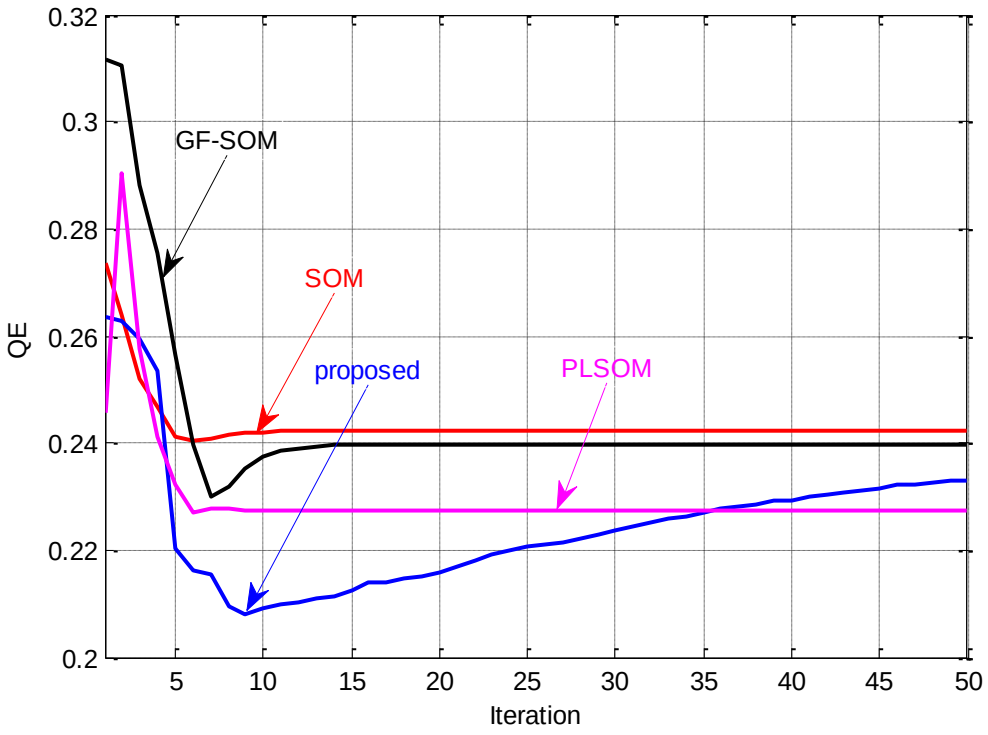


Figure 5.24. QE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Balance dataset

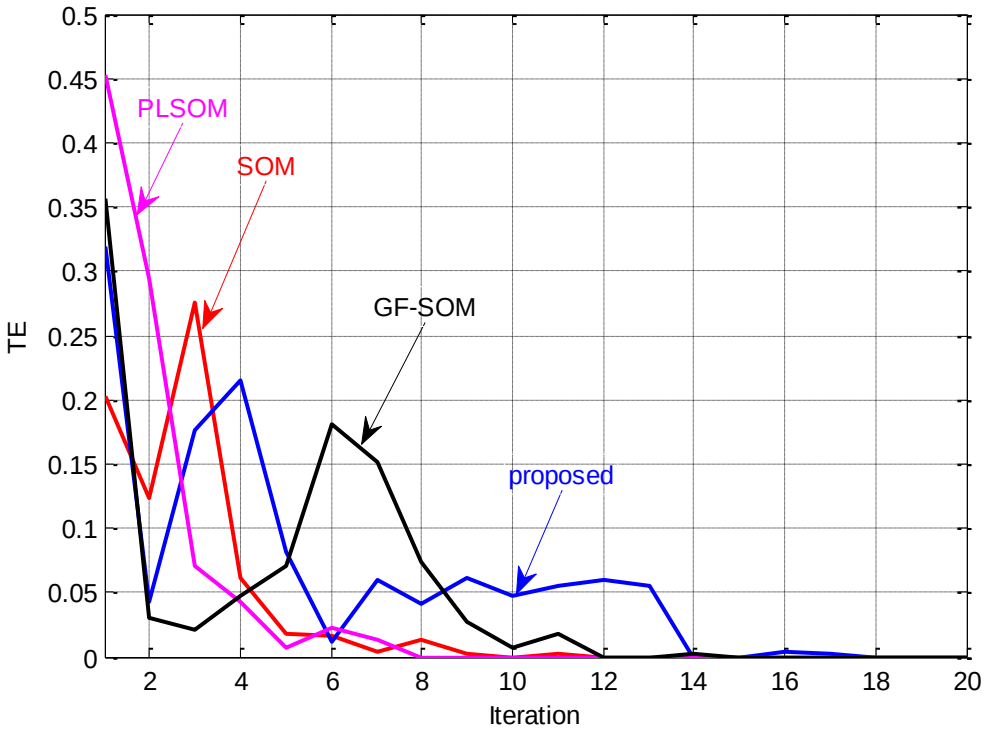


Figure 5.25. TE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Balance dataset

5.3.3 Wisconsin breast dataset

This experiment uses one of the most popular datasets in the UCI repository called Wisconsin Breast dataset. The dataset consists of 699 instances, 9 attributes and 2 classes (2 for benign, 4 for malignant). The structure of the used Kohonen maps consists of 8 neurons for the input layer with 2-D grid of 8×2 neurons in competitive layer.

The basic parameters used in this experiment are: for the conventional SOM algorithm: $\delta_\alpha = 0.3$, for GF-SOM: $\delta_\alpha = 0.1$, and $\delta_\sigma = 1.5$, for the PLSOM algorithm: $\beta = 0.6$, and for the proposed SOM: $\lambda = 0.0015$, and $\beta = 0.95$. According to the classification accuracy, the proposed SOM obtains the highest accuracy (100%) as given in Table 5.20. Proposed SOM needed only 20 iterations to reach is accuracy. The PLSOM accuracy is the lowest among compared algorithms (~3% lower). Even though all algorithms converged nearly after same number of iterations, the proposed SOM had the minimum QE among those algorithms as shown in Fig 5.26.

Fig. 5.27 shows that when other algorithms had fluctuations before reaching steady state, proposed SOM reaches the steady state earlier for topology error (TE=0)

after 3 iterations. For CPU processing time, algorithms have close processing times with PLSOM recording the highest processing time per iteration (~ 5 msec/itr higher than the proposed SOM).

Table 5.20. Performance comparison of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Wisconsin Breast dataset

Wisconsin Breast Dataset	Accuracy (%)	# iteration	QE	CPU time (msec/itr)
GF-SOM (Wang et al., 2014; Wang et al., 2015)	95.49, 95.99	1000	–	–
GF-SOM (our)	99.04	50	0.148	42.744
conventional SOM	99.04	50	0.148	39.312
PLSOM	96.65	50	0.18	44.93
Proposed SOM	100	20	0.147	39,624

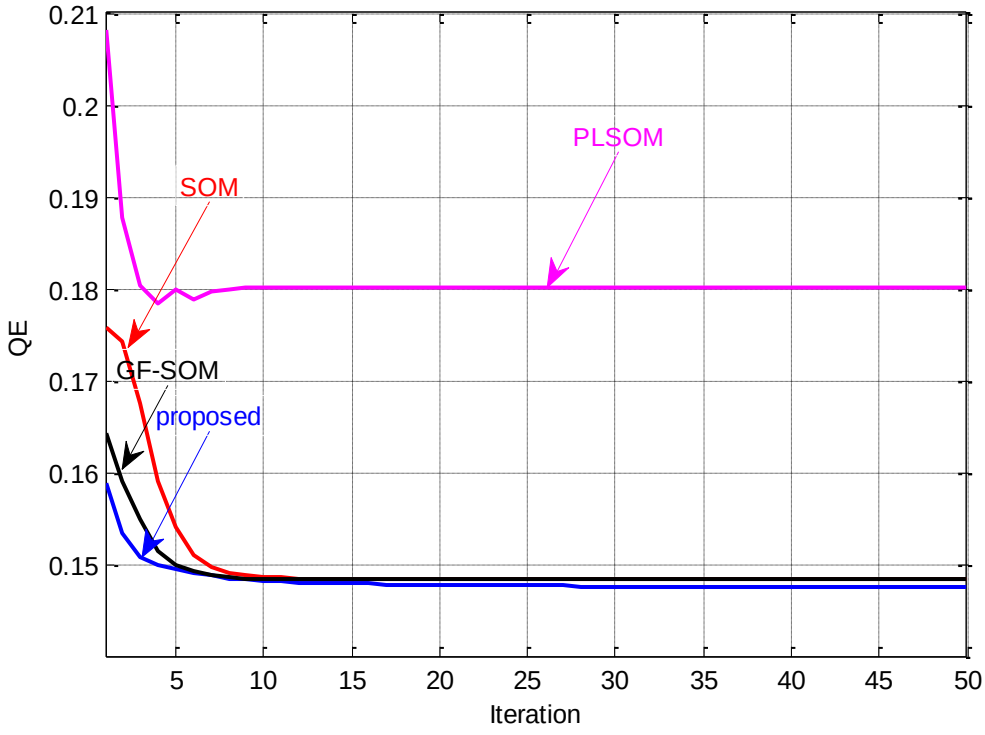


Figure 5.26. QE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Wisconsin Breast dataset

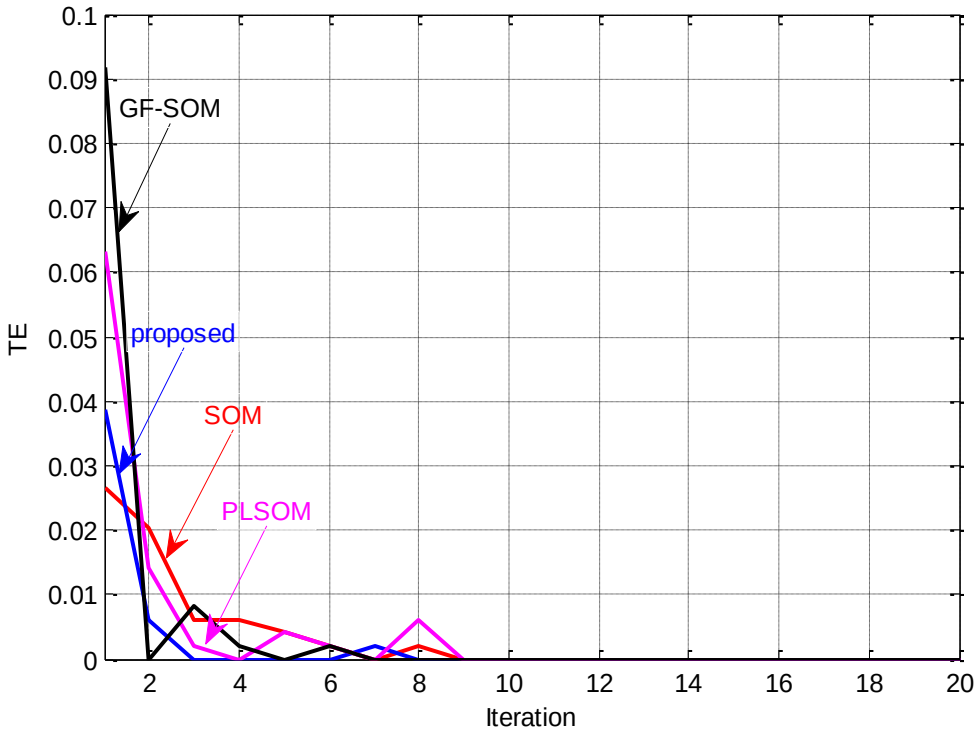


Figure 5.27. TE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Wisconsin Breast dataset

5.3.4 Dermatology dataset

The fourth experiment uses Dermatology dataset from UCI repository. Dataset consists of 366 instances, 33 attributes and 6 classes (psoriasis, seboreic dermatitis, lichen planus, pityriasisrosea, cronic dermatitis and pityriasisrubrapilaris). The structure of all the used Kohonen maps consists of 33 neurons for the input layer and 2-D grid size is 33×6 neurons in competitive layer. In this experiment the used algorithms were implemented under the following parameters: for the conventional SOM algorithm, $\delta_\alpha = 0.4$, for GF-SOM, $\delta_\alpha = 0.4$, and $\delta_\sigma = 0.02$, for the PLSOM algorithm, $\beta = 13.5$, and for the proposed SOM, $\lambda = 0.005$, and $\beta = 0.99$.

Table 5.21 shows performance comparison among the different algorithms. Conventional SOM, PLSOM and proposed SOM got the same accuracy (70.91%) with proposed SOM reaching this value using only 10 iterations. Again PLSOM recorded the highest CPU time per iteration among other algorithms. Even though the other algorithms converged in the first 6 iterations while proposed SOM converged after 18 iterations, the proposed SOM recorded better QE than all other algorithms as shown in

Fig 5.28. Fig 5.29 shows that the proposed SOM reached zero TE after 3 iteration which is faster than conventional SOM and GF-SOM (5 iterations).

Table 5.21. Performance comparison of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Dermatology dataset

Dermatology Dataset	Accuracy (%)	# iteration	QE	CPU time (msec/itr)
GF-SOM (Wang et al., 2014; Wang et al., 2015)	31.42, 31.42	1000	–	–
GF-SOM (our)	68.18	50	0.177	48.05
conventional SOM	70.91	50	0.178	43.99
PLSOM	70.91	50	0.175	56.47
Proposed SOM	70.91	10	0.173	44.62

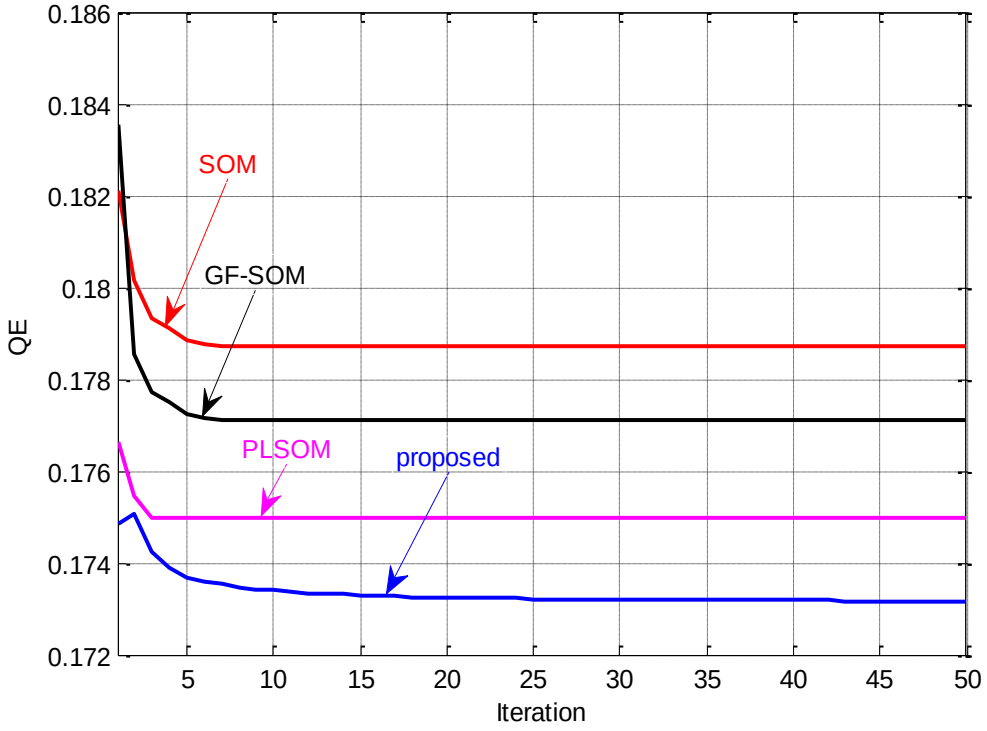


Figure 5.28. QE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Dermatology dataset

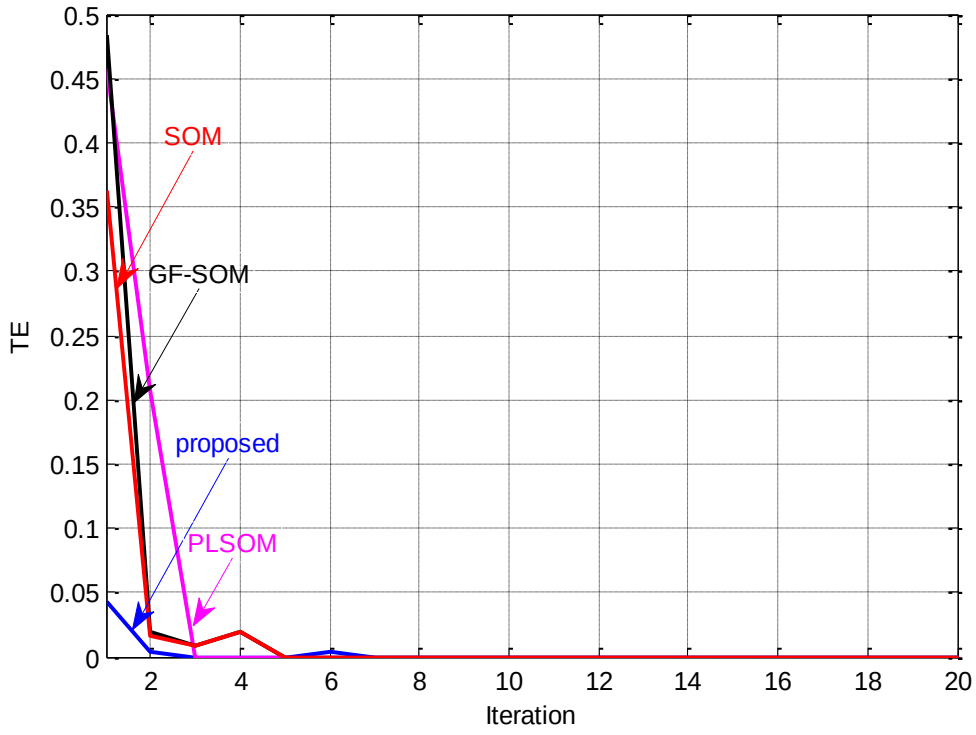


Figure 5.29. TE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Dermatology dataset

5.3.5 Ionosphere dataset

Ionosphere dataset UCI repository has been widely used by many researchers for classification purposes. Dataset had 2 classes (good and bad) with 351 instances and 34 attributes. In this experiment, the proposed SOM performance is compared with the other algorithms based on accuracy, number of iterations, QE, TE and CPU time. The structure of Kohonen maps of this dataset consists of 34 neurons for the input layer with 2-D grid of 34×2 neurons in competitive layer. The used algorithms were implemented under the following parameters, for the conventional SOM algorithm: $\delta_\alpha = 0.4$, for GF-SOM: $\delta_\alpha = 0.2$, and $\delta_\sigma = 5$, for the PLSOM algorithm: $\beta = 1.9$, and for the proposed SOM: $\lambda = 0.0025$, and $\beta = 0.98$.

In Table 5.22, the proposed SOM is providing the best classification result over other algorithms with 80.56%. The proposed learning rate is clearly showing that the weight vector update equation has achieved the optimum weights and obtained highest accuracy with lowest quantization error (QE = 0.099).

Even though the proposed algorithm has converged slower than the other algorithms, it still recorded the best QE as in shown Fig. 5.30. In Fig. 5.31, the proposed SOM outperforms conventional SOM, GF-SOM and PLSOM by reaching zero TE in one less iteration. For CPU processing time, PLSOM algorithm consumed almost double of the time required per iteration compared to other algorithms.

Table 5.22. Performance comparison of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Ionosphere dataset

Ionosphere Dataset	Accuracy (%)	# iteration	QE	CPU time (msec/itr)
GF-SOM (Wang et al., 2014; Wang et al., 2015)	79.37, 80.06	1000	–	–
GF-SOM (our)	68.52	10	0.1	57.41
conventional SOM	68.52	20	0.1	54.29
PLSOM	68.52	10	0.116	109.2
Proposed SOM	80.56	10	0.099	56.16

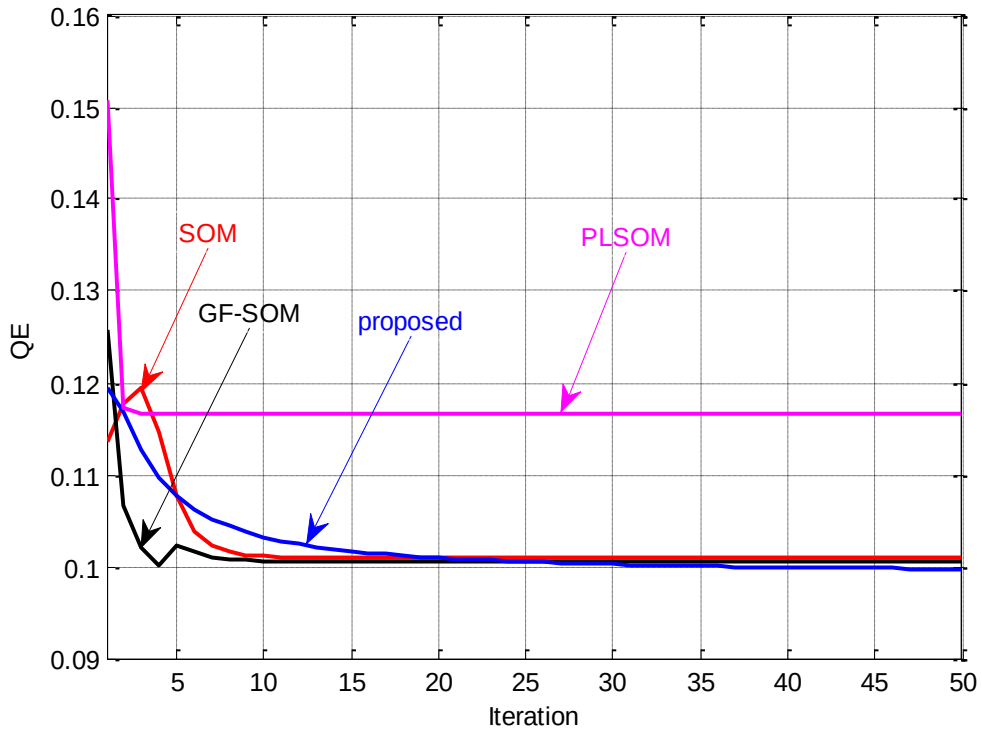


Figure 5.30. QE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Ionosphere dataset

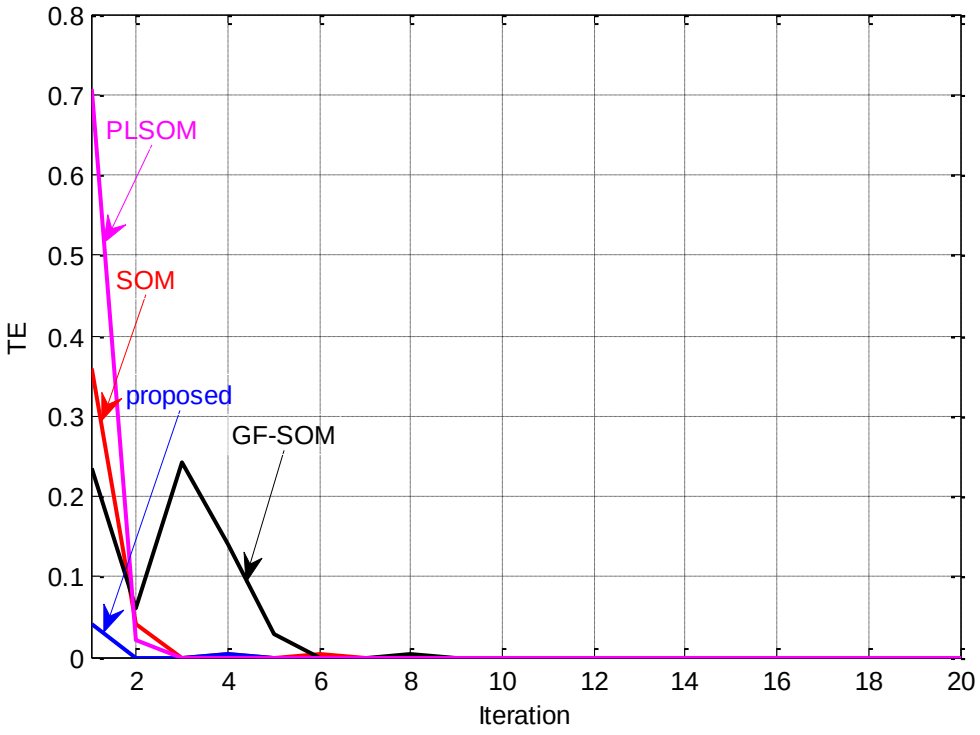


Figure 5.31. TE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Ionosphere dataset

5.3.6 Iris dataset

Another dataset we used to examine the efficiency and investigate the performance of the proposed SOM algorithm against other algorithms is Iris dataset. Iris dataset consists of 150 instances, 4 attributes and 3 classes (iris setosa, iris versicolour and iris virginica). The structure of the used Kohonen maps consists of 4 neurons for the input layer with 2-D grid of 4×3 neurons in competitive layer.

The basic parameters used in this experiment are: parameters settings for the conventional SOM algorithm: $\delta_\alpha = 0.9$, for GF-SOM: $\delta_\alpha = 0.4$, and $\delta_\sigma = 0.01$, for the PLSOM algorithm: $\beta = 3.5$, and for the proposed SOM: $\lambda = 0.005$, and $\beta = 0.98$. The comparative results are summarized in Table 5.23, where the GF-SOM, PLSOM and proposed SOM algorithms has obtained highest accuracy with 100%. Algorithms had very close QE values (see Fig. 5.32) with less CPU time requirement for the proposed SOM (40.25 msec/itr).

The topology preservation of the proposed SOM is better than other algorithms by obtained the optimum map early in the iterations 3. The PLSOM it suffered from instability and needed more time to maintain the topology as shown in Fig. 5.33.

Table 5.23. Performance comparison of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Iris dataset

Iris Dataset	Accuracy (%)	# iteration	QE	CPU time (msec/itr)
GF-SOM (Wang et al., 2014; Wang et al., 2015)	90.66, 90.00	1000	–	–
GF-SOM (our)	100	7	0.265	46.49
conventional SOM	93.33	10	0.265	39.31
PLSOM	100	5	0.265	74.88
Proposed SOM	100	5	0.264	40.25

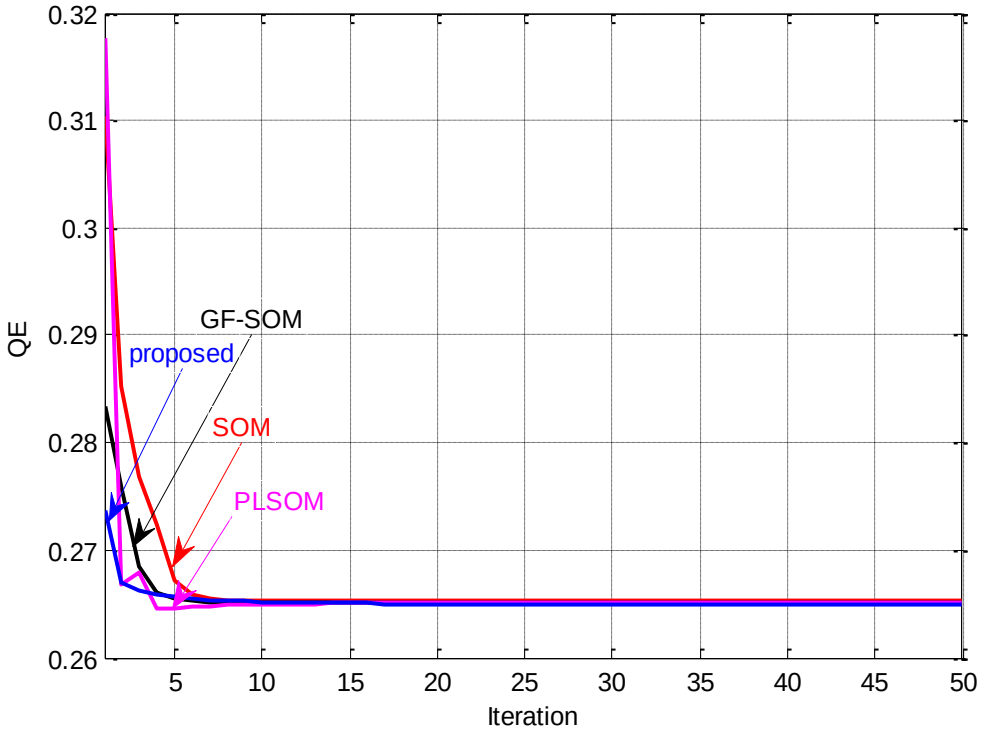


Figure 5.32. QE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Iris dataset

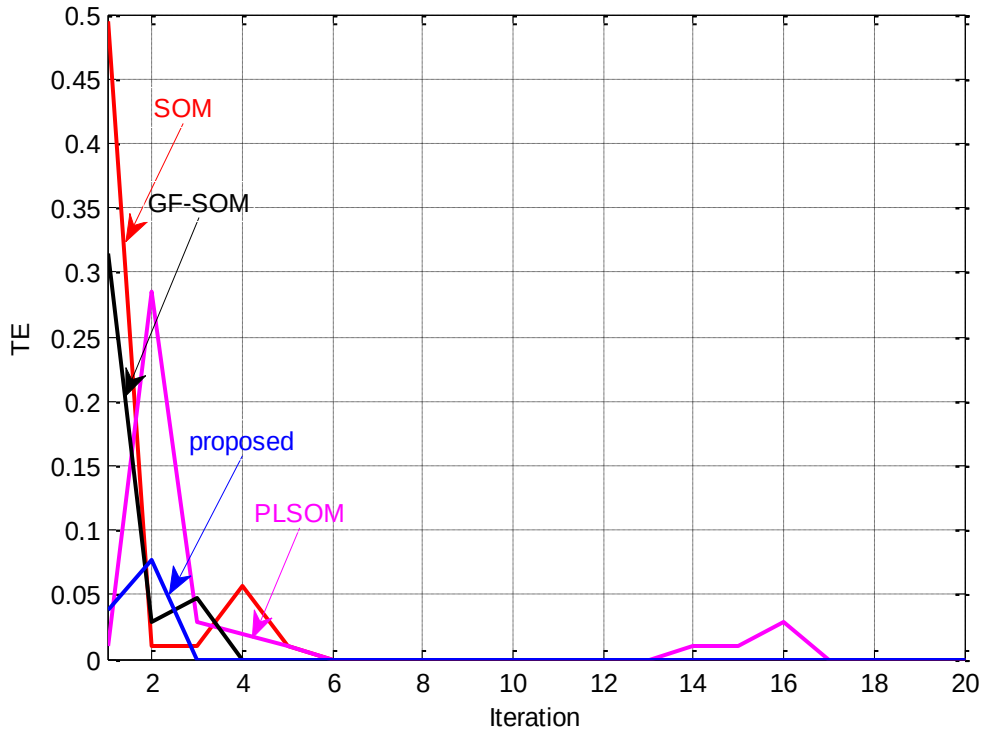


Figure 5.33. TE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Iris dataset

5.3.7 Sonar dataset

The seventh experiment for testing the proposed SOM algorithm was conducted using Sonar dataset. Dataset consists of 208 instances, 60 attributes and 2 classes (rock and mine). The structure of Kohonen maps of this dataset consists of 60 neurons for the input layer with 2-D grid of 60×2 neurons in competitive layer. In this experiment the used algorithms were implemented under the following parameters: for the conventional SOM algorithm, $\delta_\alpha = 0.7$, for GF-SOM, $\delta_\alpha = 0.28$, and $\delta_\sigma = 0.01$, for the PLSOM algorithm, $\beta = 0.4$, and for the proposed SOM, $\lambda = 0.0000095$, and $\beta = 0.999$.

Table 5.24, shows the proposed SOM algorithm higher performance than all other with accuracy (max 71.43%). The proposed SOM has minimum QE value (QE = 220.9) which is better than other algorithms and faster (see Fig. 5.34). Fig. 5.35 shows that all algorithms reached zero TE after only 2 iterations except for PLSOM which reached it after 5 iterations. For CPU processing time, PLSOM algorithm consumed almost double of the time required per iteration compared to other algorithms.

Table 5.24. Performances comparison of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Sonar dataset

Sonar Dataset	Accuracy (%)	# iteration	QE	CPU time (msec/itr)
GF-SOM (Wang et al., 2014; Wang et al., 2015)	55.77, 54.33	1000	–	–
GF-SOM (our)	57.14	50	283.3	47.42
conventional SOM	57.14	50	284.2	43.7
PLSOM	68.25	50	230.9	86.11
Proposed SOM	71.43	40	220.9	44.3

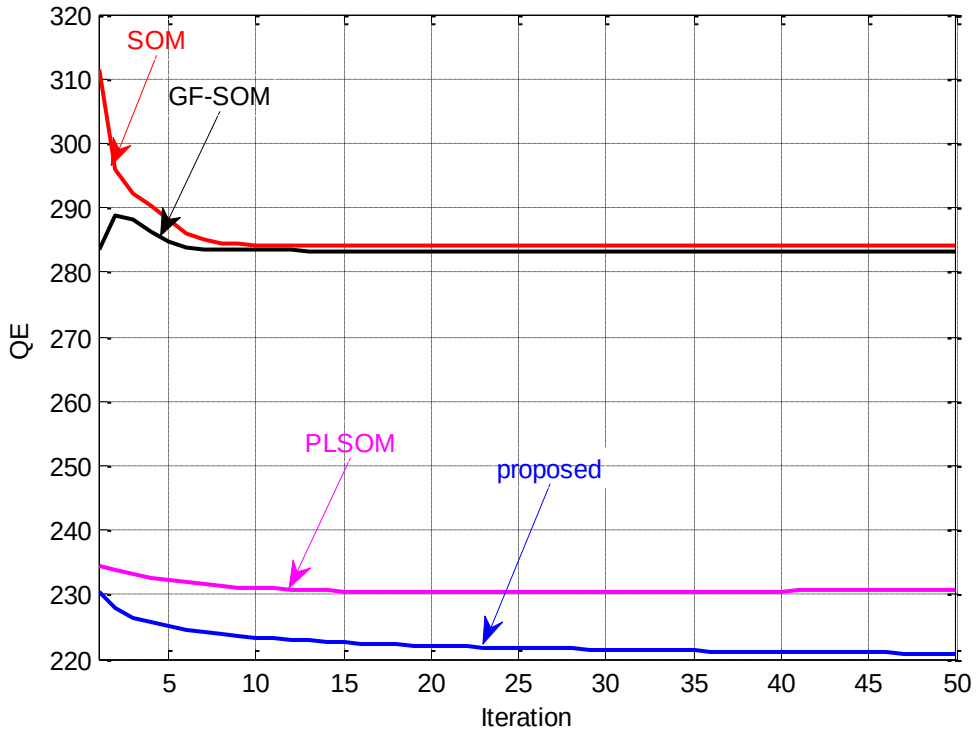


Figure 5.34. QE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Sonar dataset

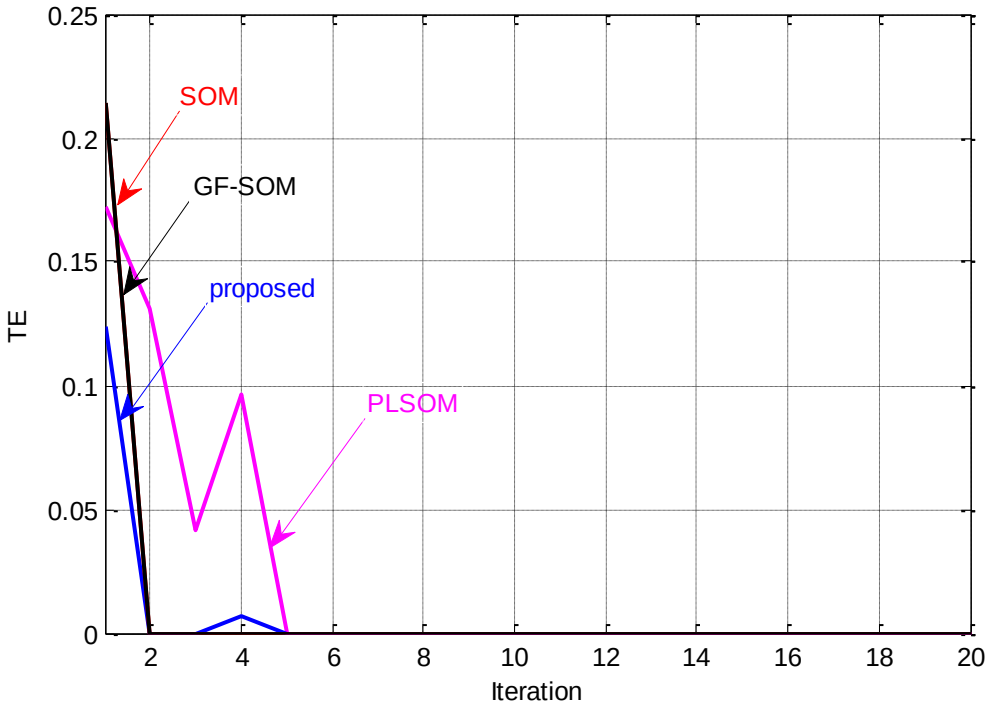


Figure 5.35. TE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Sonar dataset

5.3.8 Wine dataset

The last experiment for evaluating the proposed SOM algorithm was conducted using Wine dataset from UCI repository. Wine dataset contains 3 classes (1, 2 and 3) and 178 instances with 13 attributes. The structure of the used Kohonen maps consists of 13 neurons for the input layer with 2-D grid of 13×3 neurons in competitive layer.

The basic parameters used in this experiment for the conventional SOM algorithm: $\delta_\alpha = 0.9$, for GF-SOM: $\delta_\alpha = 0.6$, and $\delta_\sigma = 2.4$, for the PLSOM algorithm: $\beta = 1.4$, and for the proposed SOM: $\lambda = 0.009$, and $\beta = 0.98$. Table 5.25. shows that the proposed SOM algorithm recorded the best accuracy of 94.34 among other algorithms followed by PLSOM with 86.79%.

Fig. 5.36 shows that all algorithms had a fast convergence rates. The proposed SOM algorithm recorded the lowest QE with QE = 0.097 as shown in Table 5.25. The proposed SOM algorithm keep trying to improve the accuracy and map at each iteration shows some fluctuations on TE as in Fig 5.37. While all algorithms required almost

same CPU processing time per iteration, PLSOM required almost 1.5 times the CPU time.

Table 5.25. Performance comparison of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Wine dataset

Wine Dataset	Accuracy (%)	# iteration	QE	CPU time (msec/itr)
GF-SOM (Wang et al., 2014; Wang et al., 2015)	61.79, 61.80	1000	–	–
GF-SOM (our)	81.13	50	0.117	53.35
conventional SOM	71.70	50	0.127	50.544
PLSOM	86.79	15	0.113	77.06
Proposed SOM	94.34	15	0.097	52.1

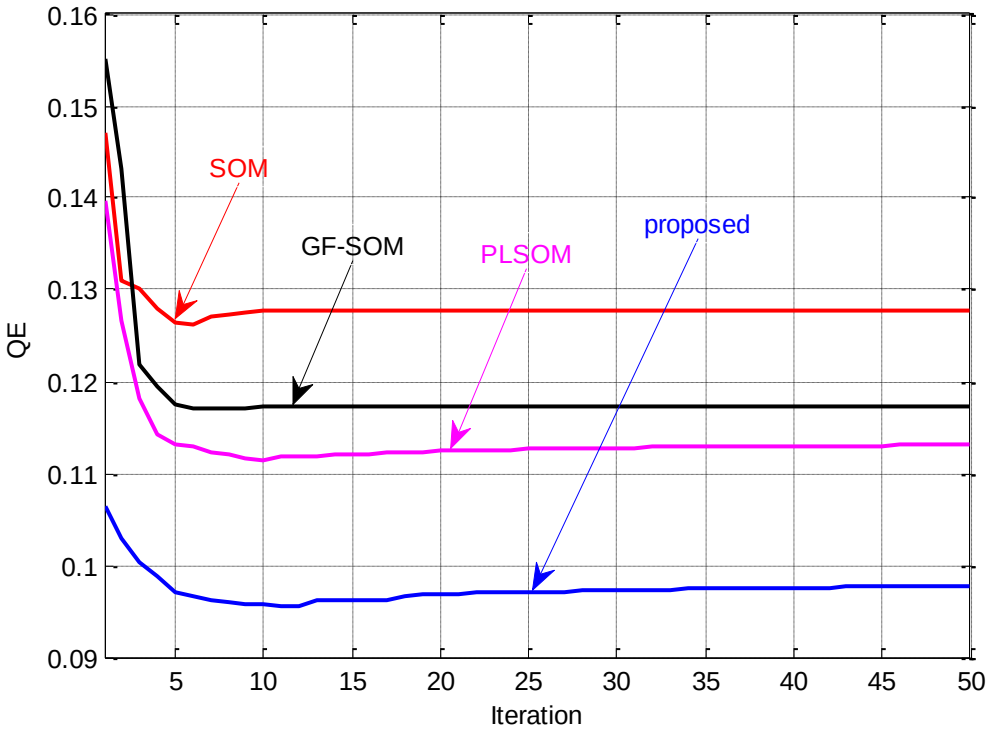


Figure 5.36. QE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Wine dataset

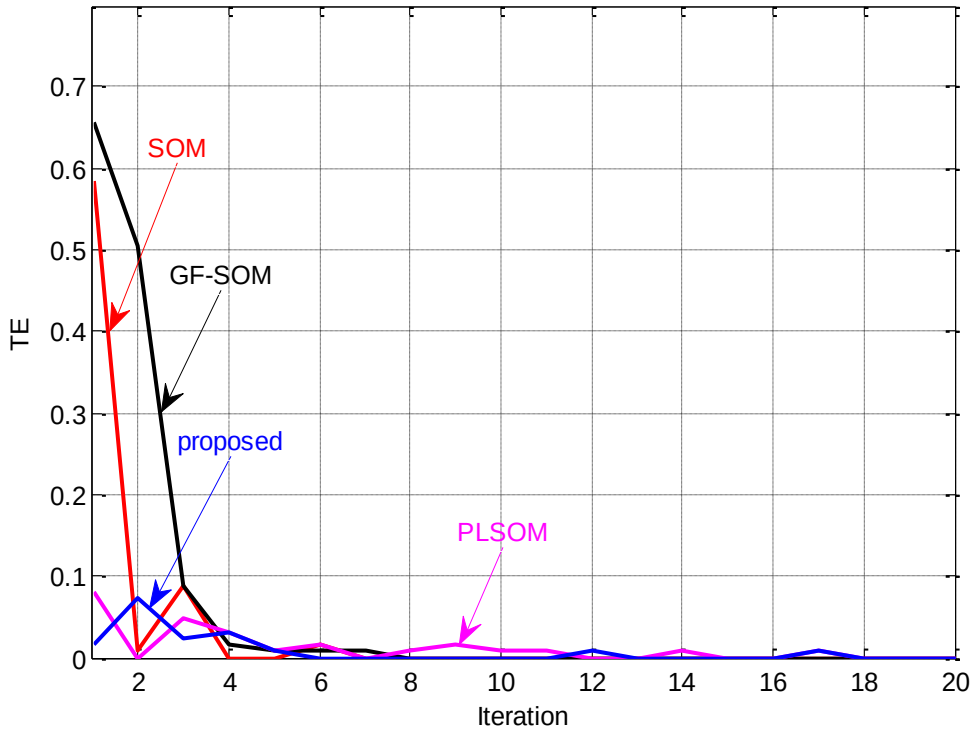


Figure 5.37. TE results of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms for Wine dataset

6. CONCLUSION AND FUTURE WORK

In this thesis, we presented a new convex combination of adaptive filtering algorithms. This combination has been proposed to overcome some of the difficulties experienced with the above-mentioned adaptive filters. We have presented the convex combination of the recently proposed RI and second-order RI algorithms. Their combined performances are compared to the combined performances of the NLMS algorithms in system identification, and noise cancellation settings. Simulation results have been showing that the proposed combination of the RI algorithms provides much better performance, in terms of MSE and rate of convergence, than that of the combined NLMS algorithms in both AWGN and ACGN environments. This gain in performance is due to the use of the instantaneous estimates of the correlations and the variable step-size in the update equation of the proposed algorithm.

The 2-D version of the recently proposed convex combination of RI algorithms has been presented. The proposed algorithm uses instantaneous estimates of the correlations which, in turn, enables the algorithm to provide high performance. The performance of the proposed algorithm has been investigated by de-noising MR images which are buried in an AWGN. The proposed algorithm is always capable of de-noising images successfully.

The main problems of artificial neural networks for both clustering and classifier algorithms to solve complexity applications are; scalability, nonlinear, problem domain of input parameters, noise, new objects, high dimensional data, constraints, and interpretation and usability (Lopes et al., 2016). Recently, new neural network algorithms are demonstrated better performance for solving complexity applications than the conventional neural network algorithms. These techniques compared with the other machine learning algorithms and handcraft features need extra more time for training and testing, and then haven't been widely used for complexity applications yet.

In addition, a new artificial neural network algorithms shows high performance in different aspects:

- For the supervised learning, the improved back-propagation algorithm with variable adaptive momentum to update the weights vectors according to input vector has been developed. This algorithm is controlled by the learning rate parameter which is dependent on the eigenvalues of the autocorrelation matrix of the input. It provides low error performance for the weights update.

This improved version of the BP algorithm by using a new variable adaptive momentum technique has been compared to well-known supervised learning algorithms such as SVM, NB, LD, K-NN, BP, and BPAM methods. The proposed BPVAM algorithm provides a faster convergence behavior than the BP and BPAM algorithms and minimizes the misadjustment error in steady-state optimum solution. The performance of the proposed algorithm outperformed those of the SVM, NB, LD, and K-NN algorithms and capable to predict the target classes, superiorly. In addition to that, it is robust at different training data sets. It shows the highest accuracy among the available machine learning methods in literature. On the other hand, it requires lower number of computation than the other well-known algorithms.

- For the unsupervised learning, we proposed a new adaptive learning rate for the conventional SOM algorithm. In this new unsupervised learning algorithm the weights updating is controlled adaptively, and the optimal weights are obtained in a short time compared to the conventional SOM and PLSOM algorithms. Moreover, the optimum weight vectors are also improved and provided lower quantization error. The proposed SOM algorithm showed higher performance compared to the conventional SOM, GF-SOM and PLSOM algorithms. Experimental results showed that the proposed SOM algorithm generally outperformed the other algorithms in terms of convergence rate, the misadjustment quantization error, preserving the topology of the map, the CPU processing time and accuracy.

In addition, the convergence phenomenon of BPVAM and proposed SOM algorithms were compared to those mentioned algorithms using the well-known datasets taken from the UCI and KEEL machine learning repository.

As a future work; the convergence analysis of 2-D version of “convex adaptive filters algorithms” may be derived. For the BPVAM algorithm; the other activation functions such as logistic sigmoid, hyperbolic tangent, and radial basis functions may be applied in the proposed BPVAM algorithm. For the proposed SOM algorithm, it is being planned to compare the performance of the proposed algorithm with well-known supervised learning algorithms using the same data sets. Moreover, it can be applied on bioinformatics data which needs to be invested in collecting samples of cases and controls and obtaining genotypes. Also, many different big data applications can be implemented.

REFERENCES

Aboulnasr, T. and Mayyas, K., 1997, A robust variable step-size LMS-type algorithm: Analysis and simulations, *IEEE Transactions on Signal Processing*, 45, 631-639.

Ahmad, M. S., Kukrer, O. and Hocanin, A., 2010a, An efficient recursive inverse adaptive filtering algorithm for channel equalization, *European Wireless Conference (EW)*, 88-92.

Ahmad, M. S., Kukrer, O. and Hocanin, A., 2010b, Recursive Inverse Adaptive Filter with Second Order Estimation of Autocorrelation Matrix, *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 482-484.

Ahmad, M. S., Kukrer, O. and Hocanin, A., 2011, Recursive inverse adaptive filtering algorithm, *Digital Signal Processing*, 21(4), 491-496.

Ahmad, M. S., Kukrer, O. and Hocanin, A., 2012, Robust Recursive Inverse Adaptive Algorithm In Impulsive Noise, *Circuits, Systems and Signal Processing*, 31(2), 703-710.

Ahmad, M. S., Kukrer, O. and Hocanin, A., 2013a, A 2-D recursive inverse adaptive algorithm, *Signal, Image and Video Processing*, 7, 221-226.

Ahmad, M. S., Kukrer, O. and Hocanin, A., 2013b, Recursive Inverse Adaptive Algorithm: A Second-Order Version, A Fast Implementation Technique, and Further Results, *Signal, Image and Video Processing*, 9(3), 665-673.

Andersso, P., 1985, Adaptive Forgetting in Recursive Identification Through Multiple Models, *International Journal of Control*, 42(5), 1175-1193.

Appiah, K., Hunter, A., Dickinson, P. and Meng, H., 2012, Implementation and applications of tri-state self-organizing maps on FPGA, *IEEE Transactions on Circuits and Systems for Video Technology*, 22(8), 1150–1160.

Arenas-Garcia, J., Figueiras-Vidal, A. R. and Sayed, A. H., 2005, Tracking properties of a convex combination of two adaptive filters, *IEEE/SP 13th Workshop on Statistical Signal Processing (SSP2005)*, doi:10.1109/SSP.2005.1628574, 109-114.

Arenas-Garcia, J., Figueiras-Vidal, A. R. and Sayed, A. H., 2006, Mean-square performance of convex combination of two adaptive filters, *IEEE Transactions on Signal Processing*, 54, 1078-1090.

Arenas-Garcia, J., Martinez-Ramon, M., Gomez-Verdejo, V. and Figueiras-Vidal, A., R., 2003, Multiple plant identifier via adaptive lms convex combination, *IEEE International Symposium on Intelligent Signal Processing*, 137-142.

Arous, N. and Ellouze, N., 2010, On the search of organization measures for a Kohonen map case study: Speech signal recognition, *International Journal of Digital Content Technology and its Applications*, 4(3), 75-84

Astel, A., Tsakovski, S., Barbieri, P. and Simeonov, V., 2007, Comparison of self-organizing maps classification approach with cluster and principal components analysis for large environmental data sets, *Water Research*, 41(19), 4566-4578.

Ayadi, T., Hamdani, T. M. and Alimi, A. M., 2012, MIGSOM: Multilevel interior growing self-organizing maps for high dimensional data clustering, *Neural Processing Letters*, 36(3), 235-256.

Azpicueta-Ruiz, L. A., Figueiras-Vidal, A. R. and Arenas-Garcia, J., 2008, A normalized adaptation scheme for the convex combination of two adaptive filters, *In IEEE International Conference Acoustics, Speech and Signal Processing*, 3137-3149.

Azpicueta-Ruiz, L. A., Zeller, M., Arenas-Garca, J. and Kellermann, W., 2010, Novel schemes for nonlinear acoustic echo cancellation based on filter combinations, *IEEE International Conference on Acoustics, Speech and Signal Processing*, 193-196.

Azpicueta-Ruiz, L. A., Zeller, M., Figueiras-Vidal, A. R. and Arenas-Garcia, J., 2010, Least-squares adaptation of affine combinations of multiple adaptive filters, *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2976-2979.

Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., Van thienen, J., 2003, Benchmarking state-of-the-art classisication algorithms for credit scoring, *The Journal of the Operational Research Society*, 54(6), 627-635.

Barnabe-Lortie, V., Bellinger, C. and Japkowicz, N., 2015, Active learning for one-class classification, *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 390-395.

Behrman, M., Linder, R., Assadi, A. H., Stacey, B. R. and Backonja, M. M., 2007, Classification of patients with pain based on neuropathic pain symptoms: comparison of an artificial neural network against an established scoring system, *European Journal of Pain*, 11(4), 370-376.

Berglund, E. and Sitte, J., 2006, The parameterless self-organizing map algorithm, *IEEE Transactions on Neural Networks*, 17(2), 305-316.

Berglund, E., 2010, Improved PLSOM algorithm, *Applied Intelligence*, 32(1), 122-130.

Bianchi, D., Calogero, R. and Tirozzi B., 2007, Kohonen neural networks and genetic classification, *Mathematical Computer Modelling*, 45(1), 34-60.

Bielza, C., and Larranaga, P., 2014, Discrete Bayesian network classifiers: a survey. *ACM Computing Surveys (CSUR)*, 47(1), 5.

Bogdan, M. and Rosenstiel, W., 2001, Detection of cluster in self-organizing maps for controlling a prostheses using nerve signals, *In Proceedings of 9th European Symposium on Artificial Neural Networks (ESANN)*, 131-136.

Bouboulis, P. and Theodoridis, S., 2010, The complex Gaussian kernel LMS algortihm, *In International Conference on Artificial Neural Networks*, 11-20.

Breining, C., Dreiscitel, P., Hansler, E., Mader, A., Nitsch, B., Puder, H., Schertler, T., Schmidt, G. and Tilp, J., 1999, Acoustic echo control. An application of very-high-order adaptive filters, *IEEE Signal Processing Magazine*, 16(4), 42-69.

Brugger, D., Bogdan, M. and Rosenstiel, W., 2008, Automatic cluster detection in Kohonen's SOM, *IEEE Transactions on Neural Networks*, 19(3), 442-459.

Burbidge, R., Trotter, M., Buxton, B. and Holden, S., 2001, Drug design by machine learning: support vector machines for pharmaceutical data analysis, *Computers and Chemistry*, 26(1), 5-14.

Burges, C. J., 1998, A tutorial on support vector machines for pattern recognition, *Data mining and knowledge discovery*, 2(2), 121-167.

Carnevale, N. T. and Rosenthal, S., 1992, Kinetics of diffusion in a spherical cell. i. no solute buffering, *Journal of Neuroscience Methods*, 41(3), 205-216.

Canan, S., Ozbay, Y. and Karlik, B., 1998, A method for removing low varying frequency trend from ECG signal, *Proceedings of the IEEE Biomedical Engineering Days*, 144–146, 1998.

Chang, H. H., Hsieh, T. J., Ting, Y. N. and Chu, W. C., 2011, Rician noise removal in mr images using an adaptive trilateral filter, *International Conference on Biomedical Engineering and Informatics (BMEI)*, 467-471.

Chi, S. C. and Yang, C. C., 2006, Integration of ant colony som and k-means for clustering analysis, *International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, 1–8.

Chi, S. C. and Yang, C. C., 2008, A two-stage clustering method combining ant colony SOM and k-means, *Journal of Information Science and Engineering*, 24(5), 1445-1460.

Chiang, W. Y. K., Zhang, D. and Zhou, L., 2006, Predicting and explaining patronage behavior toward web and traditional stores using neural networks: a comparative analysis with logistic regression, *Decision Support Systems*, 41(2), 514-531.

Cottrell, M., Gaubert, P., Eloy, C., Francois, D., Hallaux, G., Lacaille, J. and Verleysen, M., 2009, Fault prediction in aircraft engines using self-organizing maps, *International Workshop on Self-Organizing Maps*. Springer Berlin Heidelberg, 5629, 37-44.

De Mantaras, R. L. and Armengol, E., 1998, Machine learning from examples: Inductive and Lazy methods, *Data and Knowledge Engineering*, 25(1), 99-123.

Derelioglu, G. and Gurgen, F., 2011, Knowledge discovery using neural approach for SME's credit risk analysis problem in Turkey, *Expert Systems with Applications*, 38(8), 9313-9318.

Destexhe, A, and Sejnowski, T. J., 1995, G protein activation kinetics and spillover of gamma-aminobutyric acid may account for differences between inhibitory responses in the hippocampus and thalamus, *Proceedings of the National Academy of Sciences*, 92(21), 9515-9519.

Diniz, P. S. R., 1997, Adaptive filtering: algorithms and practical implementations, *Kluwer Academic Publishers*, Boston, USA.

Dogancay, K. and Tanrıkulu, O., 2001, Adaptive filtering algorithms with selective partial update, *IEEE Transaction Circuits and Systems-II*, 48(8), 762-769.

Dollar, P., Tu, Z. and Belongie, S., 2006, Supervised learning of edges and object boundaries. *In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, 2, 1964-1971.

Duda, R. O., Hart, P. E. and Stork, D. G., 2001, Pattern classification, *John Wiley and Sons*, NJ, USA.

Duffy, A. H. B., 1997, The'what'and'how'of learning in design, invited paper, *IEEE Intelligent Systems*, 12(3), 71-76.

Dvir, D., David, A. B., Sadeh, A. and Shenhar, A. J., 2006, Critical managerial factors affecting defense projects success: a comparison between neural network and regression analysis, *Engineering Applications of Artificial Intelligence*, 19(5), 535-543.

EIGayar, N., Schwenker, F. and Palm, G., 2006, A study of the robustness of KNN classifiers trained using soft labels, *Artificial Neural Networks in Pattern Recognition*, 67-80.

Eksioglu, E. M. and Tanc, A. K., 2011, RLS algorithm with convex regularization, *IEEE Signal Processing Letters*, 18(8), 470-473.

Everitt, B. S., Landau, S. and Leese, M., 2001, Cluster Analysis, *Arnold*, London, UK.

Fonseca, A. M., Biscaya, J. L., Aires-de-Sousa, J. and Lobo, A. M., 2006, Geographical classification of crude oils by Kohonen self-organizing maps, *Analytica Chimica Acta*, 556(2), 374-382.

Gay, S. L. and Benesty, J., 2000, Acoustic signal processing for telecommunication, *Kluwer Academic Publishers*, Boston, USA.

Gesbert, D. and Duhamel, P., 2000, Unbiased blind adaptive channel identification, *IEEE Transactions on Signal Processing*, 48(1), 148–158.

Glentis, G. O., Berberidis, K. and Theodoridis, S., 1999, Efficient least squares adaptive algorithms for FIR transversal filtering, *IEEE Signal Processing Magazine*, 16(4), 13–41.

Gorgonio, F. L. and Costa, J. A. F., 2008, Combining parallel self-organizing maps and k-means to cluster distributed data, *11th IEEE International Conference on Computational Science and Engineering Workshops*, 53-58.

Gwadabe, T. R. and Salman, M. S., 2015, A new leaky-LMS algorithm with analysis, *International Arab Journal of Information Technology*.

Hadhoud, M. M. and Thomas, D. W., 1998, The Two-Dimensional Adaptive LMS (TDLMS) Algorithm, *IEEE Transactions on Circuits and Systems*, 35(5), 485-494.

Hameed, A. A., Salman, M. S. and Karlik, B., 2014, A new 2-D convex combination of recursive inverse algorithms, *IEEE 34th International Conference on Electronics and Nanotechnology (ELNANO)*, 273-276.

Haykin, S., 2002, Adaptive filter theory, *Prentice-Hall Information and System Sciences Series*, NJ, 2002.

Haykin, S., 2009, Neural networks and learning machines, *Upper Saddle River*, NJ, USA.

Hoffmann, M., 2005, Numerical control of Kohonen neural network for scattered data approximation, *Numerical Algorithms*, 39(1), 175-186.

Hou, Y. T., Chang, Y., Chen, T., Laih, C. S. and Chen, C.M., 2010, Malicious web content detection by machine learning, *Expert Systems with Applications*, 37(1), 55-60.

Huang, C., Davis, L. S. and Townshend, J. R. G., 2002, An assessment of support vector machines for land cover classification, *International Journal of Remote Sensing*, 23(4), 725-749.

Huang, H., Pasquier, M. and Quek, C., 2011, Decision support system based on hierarchical co-evolutionary fuzzy approach: A case study in detecting gamma ray signals, *Expert Systems with Applications*, 38(9), 10719-10729.

Iseri, A. and Karlik, B., 2009, An artificial neural networks approach on automobile pricing, *Expert Systems with Applications*, 36(2), 2155-2160.

Istook, E. and Martinez, T., 2002, Improved backpropagation learning in neural networks with windowed momentum, *International Journal of Neural Systems*, 12(3-4), 303-318.

Kang, B., Choi, O., Kim, J. D. and Hwang, D., 2013, Noise reduction in magnetic resonance images using adaptive non-local means filtering, *Electronics Letters*, 49(5), 324-326.

Karlik, B. and Cemel, S. S., 2012, Diagnosing diabetes from breath odor using artificial neural networks, *Turkiye Klinikleri Journal of Medical Sciences*, 32(2), 331-336.

Karlik, B. and Olgac, A. V., 2011, Performance analysis of various activation functions in generalized MLP architectures of neural networks, *International Journal of Artificial Intelligence and Expert Systems*, 1(4), 111-122.

Karlik, B., 2000, Differentiating type of muscle movement via AR modeling and neural networks classification, *Turkish Journal of Electrical Engineering and Computer Science*, 7(1-3), 45-52.

Karlik, B., 2013, Soft computing methods in bioinformatics: A Comprehensive Review, *Mathematical and Computational Applications*, 18(3), 176-197.

Karlik, B., 2014, Machine learning algorithms for characterization of EMG signals, *International Journal of Information and Electronics Engineering*, 4(3), 189-194.

Karlik, B., Ozkaya, E., Aydin, S. and Pakdemirli, M., 1998, Vibration of a beam-mass system using artificial neural networks, *Computer and Structures*, 69, 339-347.

Kaski, S., Honkela, T., Lagus, K., and Kohonen, T. 1998. WEBSOM–self-organizing maps of document collections. *Neurocomputing*, 21(1), 101-117.

Keramati, A. and Yousefi, N., 2011, A proposed classification of data mining techniques in credit scoring, *Proceedings 2nd International Conference on Industrial Engineering and Operations Management (IEOM 2011)*, 416-424.

Kim, C. I., Yu, I. K. and Song, Y. H., 2002, Kohonen neural network and wavelet transform based approach to short-term load forecasting, *Electric Power Systems Research*, 63(3), 169-176.

Kim, Y., 2009, Boosting and measuring the performance of ensembles for a successful database marketing, *Expert Systems with Applications*, 36(2), 2161-2176.

Kohonen, T., 2001, Self-organizing maps, *Springer*, Berlin.

Kohonen, T., 1982, Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, 43(1), 59-69.

Kohonen, T., 1989, Self-Organization and Associative Memory Process, *Springer-Verlag*, Berlin.

Kohonen, T., 1990, The self-organizing maps, *Proceedings of the IEEE*, 78(9), 1464-1480.

Kohonen, T., Oja, E., Simula, O., Visa, A. and Kangas, J., 1996, Engineering applications of the self-organizing map, *Proceedings of the IEEE*, 84(9), 1358-1384.

Kosat, S. S. and Singer, A. C., 2009, A performance-weighted mixture of LMS filters, *In IEEE International Conference on Acoustics, Speech, and Signal Processing*, 3101-3104.

Kotsiantis, S. B. and Pintelas, P. E., 2004, Hybrid feature selection instead of ensembles of classifiers in medical decision support, *Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 4-9.

Kotsiantis, S. B., Zaharakis, I. D. and Pintelas, P. E., 2006, Machine learning: a review of classification and combining techniques, *Artificial Intelligence Review*, 26(3), 159-190.

Kotsiantis, S. B., Zaharakis, I. D. and Pintelas, P. E., 2007, Supervised machine learning: A review of classification techniques, *Artificial Intelligence Review*, 26(3), 159-190.

Kozat, S. S. and Singer, A. C., 2000, Multi-stage adaptive signal processing algorithms, *Proceedings of the IEEE Sensor Array and Multichannel Signal Processing Workshop*, 380-384.

Kutner, M. H., Nachtsheim, C. J., Neter, J. and Li, W., 2005, Applied linear statistical models, *McGraw-Hill Irwin*, New York, USA.

Langley, P. and Simon, H. A., 1995, Applications of machine learning and rule induction. *Communications of the ACM*, 38(11), 54-64.

Larose, D. T., 2005, K-nearest neighbor algorithm, *Discovering Knowledge in Data an Introduction to Data Mining, John Wiley and Sons*,90-106, NJ, USA.

Lee, K., Booth, D. and Alam, P., 2005, A comparison of supervised and unsupervised neural networks in predicting bankruptcy of Korean rms, *Expert Systems with Applications*, 29(1), 1-16.

Lopes, L. A., Machado, V. P., Rabelo, R. A. L., Fernandes, R. A. S. and Lima, B. V. A., 2016, Automatic labelling of clusters of discrete and continuous data with supervised machine learning, *Knowledge-Based Systems*, 106, 231-241.

Ma, W., Qu, H., Gui, G., Xu, L., Zhao, J. and Chen, B., 2015, Maximum correntropy criterion based sparse adaptive filtering algorithms for robust channel estimation under non-gaussian environments, *Journal of the Franklin Institute*, 352(7), 2708-2727.

Macskassy, S. A., 2011, Relational classifiers in a non-relational world: using homophily to create relations, *The Tenth International Conference on Machine Learning and Applications*, 1, 406-411.

Madhow, U. and Honig, M. L., 1994, MMSE interference suppression for direct-sequence spread-spectrum CDMA, *IEEE Transactions on Communications*, 42(12), 3178–3188.

Maouche, K. and Slock, D. T. M., 2000, Fast subsampled-updating stabilized fast transversal filter (FSU SFTF) RLS algorithm for adaptive filtering, *IEEE Transactions on Signal Processing*, 48(8), 2248-2257.

Mandic, D., Vayanos, P., Boukis, C., Jelfs, B., Goh, S. L., Gautama, T. and Rutkowski, T., 2007, Collaborative adaptive learning using hybrid filters, *In IEEE International Conference on Acoustics, Speech, and Signal Processing-ICASSP'07*, 3, 921-924.

Marini, F., Zupan, J. and Magri, A. L., 2005, Class-modeling using Kohonen artificial neural networks, *AnalyticaChimicaActa*, 544(1), 306-314.

Martens, D., Baesens, B. B. and Van Gestel, T., 2009, Decompositional rule extraction from support vector machines by active learning, *IEEE Transactions on Knowledge and Data Engineering*, 21(2), 178-191.

Martinez-Ramon, M., Arenas-Garcia, J., Navia-Vazquez, A. and Figueiras-Vidal, A. R., 2002, An Adaptive Combination of Adaptive Filters for Plant Identification, *In the Proceedings of 14th International Conference on Digital Signal Processing*, 1195-1198.

Mendialdua, I., Fernandez, N. O., Sierra, B. and Lazkano, E., 2012, Positive predictive value based dynamic K-nearest neighbor, *Advances in Knowledge-based and Intelligent Information and Engineering Systems*, 59-68.

Mikhael, W. B. and Ghosh, S. M., 1992, Two-dimensional block adaptive filtering algorithms, *In Proceedings of IEEE International Symposium on Circuits and Systems*, 1219-1222.

Mikhael, W. B. and Wu, F. H., 1989, A fast block fir adaptive digital filtering algorithm with individual adaptation of parameters, *IEEE Transactions on Circuits and Systems*, 36, 1-10.

Minsky, M. and Papert, S. A., 1969, Perceptrons, *MIT Press*, Cambridge, MA, USA.

Mosteller, F. and Tukey, J. W., 1968, Data analysis, including statistics, *Addison-Wesley*, 2.

Mukherjee, A., 1997, Self-organizing neural network for identification of natural modes, *Journal of Computing in Civil Engineering*, 11(1), 74-77.

Muneyasu, M., Hinamoto, T. and Yagi, H., 1998, A realization of 2-d adaptive filters using affine projection algorithm, *Journal of the Franklin Institute*, 335(7), 1185-1193.

Murphy, K. P., 2006, Naive Bayes classifier, *University of British Columbia*, Vancouver, Canada.

Mustafa, Z. A., Abrahim, B. A. and Kadah, Y. M., 2012, K11. Modified hybrid median filter for image denoising, *National Radio Science Conference (NRSC)*, 705-712.

Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A. and Brown, S. D., 2004, An introduction to decision tree modeling, *Journal of Chemometrics*, 18(6), 275-285.

Ozbay, Y. and Karlik, B., 2002, A fast training back-propagation algorithm on windows, *In Proceedings of 3rd International Symposium on Mathematical and Computational Applications*, 204-210.

Ozbay, Y., Karlik, B. and Kavsaoglu A. R., 2003, A Windows-based digital filter design, *Mathematical and Computational Applications*, 8(3), 287-294.

Phatak, K., Jakhade, S., Nene, A., Kamathe, R. S. and Joshi, K. R., 2011, Wavelet domain filtering of mr image sequences with appropriate filtering approach based on mr image type, *International Conference on Signal Processing, Communications, Computing and Networking Technologies (ICSCCN)*, 9(2), 520-525.

Qiu, G., Varley, M. R. and Terrell, T. J., 1992, Accelerated training of backpropagation networks by using adaptive momentum step, *Electronics Letters*, 28(4), 377-379.

Ray, D., Majumder, D. D. and Das, A., 2012, Noise reduction and image enhancement of mri using adaptive multiscale data condensation, *International Conference on Recent Advances in Information Technology (RAIT)*, 107-113.

Ripley, B. D., 2007, Pattern recognition and neural networks, *Cambridge University Press*, UK.

Rojas, R., 1996, Neural Networks-A Systematic Introduction, *Springer-Verlag*, Berlin.

Rosenblatt, F., 1962, Principles of neurodynamics. perceptrons and the theory of brain mechanisms, *Spartan Books*, Washingotn D.C., USA

Rumelhart, D. E. and McClelland, J. L., 1986, Parallel distributed processing: explorations in the microstructure of cognition, *MIT Press Cambridge*, 2, MA, USA.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J., 1988, Learning representations by backpropagating errors, *In Neurocomputing: Foundations of Research*, 3, 696-699.

Saetern, K. and Eiamkanitchat, N., 2014, An ensemble K-nearest neighbor with neuro-fuzzy method for classification, *Recent Advances in Information and Communication Technology*, 43-51.

Salman, M. S., 2014, Sparse leaky-LMS algorithm for system identification and its convergence analysis, *International Journal of Adaptive Control and Signal Processing*, 28(10), 1065-1072.

Salman, M. S., Hameed, A. A., Turan, C. and Karlik, B., 2015, A new sparse convex combination of ZA-LLMS and RZA-LLMS algorithms. *In 2015 23nd Signal Processing and Communications Applications Conference (SIU)*, 711-714.

Sarangi, P. P., Mishra, B. S. P., Majhi, B., Dehuri, S. and Mohan, F., 2013, Hybrid supervised learning in MLP using real-coded GA and back-propagation, *International Journal of Computer Applications*, 62(21), 32-39.

Sayed, A. H., 2003, Fundamentals of adaptive filtering, *John Wiley and Sons*, NJ, USA.

Sayed, A. H., 2008, Adaptive filters, *John Wiley and Sons*, NJ, USA.

Scholkopf, B. and Smola, A. J., 2002, Learning with kernels: support vector machines, regularization, optimization, and beyond, *MIT Press Cambridge*, MA, USA..

Seber, G. A. and Lee, A. J., 2012, Linear regression analysis, *John Wiley and Sons*, 936, NJ, USA.

Shao, H. and Zheng, G., 2009, A new BP algorithm with adaptive momentum for FNNs training, *WRI Global Congress on Intelligent Systems*, 4, 16-20.

Shao, H. and Zheng, G., 2011, Convergence analysis of a back-propagation algorithm with adaptive momentum, *Neurocomputing*, 74(5), 749-752.

Shin, H. C., Sayed, A. H. and Song, W. J., 2004, Variable step-size NLMS and affine projection algorithms, *IEEE Signal Processing Letters*, 11(2), 132-135.

Shouman, M., Turner, T. And Stocker, R., 2012, Applying K-nearest neighbor in diagnosing heart disease patients, *International Journal of Information and Education Technology*, 2(3), 220-223.

Silva, M. T. M. and Nascimento, V. H., 2008, Improving the tracking capability of adaptive filters via convex combination, *IEEE Transactions on Signal Processing*, 56(7), 3137-3149.

Song, X. H. and Hopke, P. K., 1996, Kohonen neural network as a pattern recognition method based on the weight interpretation, *Analytical Chimica Acta*, 334(1), 57-66.

Stone, M., 1974, Cross-validatory choice and assessment of statistical predictions, *Journal of the Royal Statistical Society*, 111-147.

Tamukoh, H., Aso, T., Horio, K. and Yamakawa, T. 2004, Self-organizing map hardware accelerator system and its application to realtime image enlargement, *Proceedings. 2004 IEEE International Joint Conference on Neural Networks*, 4, 2683-2687.

Tasdemir, K., Milenov, P. and Tapsall, B., 2011, Topology-based hierarchical clustering of self-organizing maps, *IEEE Transactions on Neural Networks*, 22(3), 474-485.

Trump, T., 2009a, Steady state analysis of an output signal based combination of two nlms adaptive filters, *Proceedings of the 17th European Signal Processing Conference*, 1720-1724.

Trump, T., 2009b, Tracking performance of a combination of two NLMS adaptive filters, *IEEE 15th Workshop on Statistical Signal Processing*, 29, 181-184.

Turan, C. and Salman, M. S., 2014, A sparse function controlled variable step-size LMS algorithm for system identification, *Proceedings of the Signal Processing and Communications Applications Conference*, 329-332.

Vapnik, V., 1995, The nature of statistical learning theory, *Springer-Verlag*, Berlin.

Varghees, V. N., Manikandan, M. S. and Gini, R., 2012, Adaptive mri image denoising using total-variation and local noise estimation, *International Conference on Advances in Engineering, Science and Management (ICAESM)*, 506-511.

Vesanto J. and Alhoniemi, E., 2000, Clustering of the self-organizing map, *IEEE Transactions on Neural Networks*, 11(3), 586-600.

Wang, J., 2009, A variable forgetting factor RLS adaptive filtering algorithm, *3rd IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*, 1127-1130.

Wang, K. J., Adrian, A. M., Chen, K. H. and Wang, K. M., 2015, An improved electromagnetism-like mechanism algorithm and its application to the prediction of diabetes mellitus, *Journal of Biomedical Informatics*, 54, 220-229.

Wang, K. J., Chen, K. H. and Angelia, M. A., 2014, An improved artificial immune recognition system with the opposite sign test for feature selection, *Knowledge-Based Systems*, 71, 126-145.

Wang, T. and Wang, C. L., 1998, A new two-dimensional block adaptive fir filtering algorithm and its application to image restoration, *IEEE Transactions on Image Processing*, 7, 238-246.

Wickramasinghe, L. K., Alahakoon, L. D. and Smith, K. A., 2005, Computation of meta-learning classifiers in distributed data mining using a novel cognitive memory model, *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 180-186.

Widrow, B. and Hoff, M. D., 1960, Adaptive Switching Circuits, *1960 IRE WESCON Convention Record*, Part 4, 96-104.

Williams, D. R. G. H. R. and Hinton, G. E., 1986, Learning representations by back-propagating errors, *Nature*, 323, 533-536.

Witten, I. H. and Frank, E., 2005, Data Mining: Practical machine learning tools and techniques, *Morgan Kaufmann*, San Francisco, USA .

Wong, M. L. D., Jack, L. B., and Nandi, A. K., 2006, Modified self-organising map for automated novelty detection applied to vibration signal monitoring, *Mechanical Systems and Signal Processing*, 20(3), 593-610.

Wu, W. and Xu, Y., 2002, Deterministic convergence of an online gradient method for neural networks, *Journal of Computational and Applied Mathematics*, 144(1-2), 335-347.

Wu, W., Feng, G. and Li, X., 2002, Training multilayer perceptrons via minimization of sum of ridge functions, *Advances in Computational Mathematics*, 17(4), 331-347.

Xu, R. and Wunsch, D., 2005, Survey of clustering algorithms, *IEEE Transactions on Neural Networks*, 16(3), 645-678.

Yamaguchi, T., Ichimura, T. and Mackin, K. J., 2010, Adaptive learning algorithm in tree-structured self-organizing feature map, *Proceedings of Joint 5th International Conference on Soft Computing and Intelligent Systems and 11th International Symposium on Advanced Intelligent Systems(SCIS & ISIS 2010)*, 1429-1434.

Yang, J. and Yang, J.y., 2003, Why can LDA be performed in PCA transformed space?, *Pattern Recognition*, 36(2), 563-566.

Yang, L., Ouyang, Z. and Shi, Y., 2012, A modified clustering method based on self-organizing maps and its applications, *Procedia Computer Science*, 9, 1371–1379.

Yang, T. and Kecman, V., 2008, Adaptive local hyperplane classification, *Neurocomputing*, 71(13), 3001-3004.

Yang, T., Vojislav, K., Longbing, C. and Chengqi, Z., 2010, Testing adaptive local hyperplane form multi-class classification by double cross-validation, *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp.

Yen, G. G. and Wu, Z., 2008, Ranked centroid projection: A data visualization approach with self-organizing maps, *IEEE Transactions on Neural Networks*, 19(2), 245-259.

Yu, C. C. and Liu, B. D., 2002, A backpropagation algorithm with adaptive learning rate and momentum coefficient, *Proceedings of the 2002 International Joint Conference on Neural Networks*, 2, 1218-1223.

Yu, X., Loh, N. K. and Miller, W. C., 1993, A new acceleration technique for the backpropagation algorithm, *In IEEE International Conference on Neural Networks*, 3, 1157-1161.

Zhang, Y. and Chambers, J. A., 2006, Convex combination of adaptive filters for a variable tap-length LMS algorithm, *IEEE Signal Processing Letters*, 13(10), 628-631.

APPENDICES

APPENDIX-1 Error performance (SSE) of BP, BPAM, and BPVAM algorithms are compared using different η and α parameters.

1. Breast Cancer Dataset

Table 1. Error convergence behavior of BP algorithm in 400 iterations for breast cancer dataset

η / α	0.7	0.8	0.9
0.03	6.42	6.175	6.19
0.02	6.38	6.17	6.034
0.01	6.34	6.08	6.033

Table 2. Error convergence behavior of BPAM algorithm in 400 iterations for breast cancer dataset

η / α	0.7	0.8	0.9
0.03	6.4	5.75	5.6
0.02	6.34	6.65	5.6
0.01	8.08	6.3	5.97

Table 3. Error convergence behavior of BPVAM algorithm in 400 iterations for breast cancer dataset

η / λ/β	0.7	0.8	0.9
0.0087 0.994	6.4	6.07	5.9
0.0086 0.993	6.32	6.04	5.9
0.0085 0.992	6.6	6.06	5.88

2. Heart Dataset

Table 4. Error convergence behavior of BP algorithm in 100 iterations for heart dataset

η / α	0.005	0.004	0.003
0.03	4.32	4.5	5.12
0.02	4.32	4.5	5.14
0.01	4.33	4.5	5.161

Table 5. Error convergence behavior of BPAM algorithm in 100 iterations for heart dataset

η / α	0.05	0.04	0.03
0.11	4.26	4.34	4.6
0.1	4.27	4.36	4.6
0.09	4.28	4.37	4.6

Table 6. Error convergence behavior of BPVAM algorithm in 100 iterations for heart dataset

η / λ/β	0.05	0.04	0.03
0.0087 0.994	4.24	4.266	4.3
0.0086 0.993	4.21	4.25	4.3
0.0085 0.992	4.22	4.263	4.34

3. Heart-Statlog Dataset

Table 7. Error convergence behavior of BP algorithm in 20 iterations for heart-statlog dataset

η / α	0.3	0.2	0.1
0.07	0.12	0.1228	0.14
0.06	0.12	0.123	0.141
0.05	0.12	0.123	0.141

Table 8. Error convergence behavior of BPAM algorithm in 20 iterations for heart-statlog dataset

η / α	0.3	0.2	0.1
0.00029	0.12	0.12	0.133
0.00028	0.12	0.12	0.138
0.00027	0.12	0.124	0.136

Table 9. Error convergence behavior of BPVAM algorithm in 20 iterations for heart-statlog dataset

η / λ/β	0.3	0.2	0.1
0.027 0.998	0.453	0.453	0.45
0.026 0.997	0.12	0.127	0.172
0.025 0.996	0.12	0.12	0.123

4. Iris Dataset

Table 10. Error convergence behavior of BP algorithm in 150 iterations for iris dataset

η / α	0.07	0.06	0.05
0.003	0.801	1.34	1.84
0.002	0.8	1.35	1.84
0.001	0.8	1.34	1.84

Table 11. Error convergence behavior of BPAM algorithm in 150 iterations for iris dataset

η / α	0.07	0.06	0.05
0.0005	0.99	0.25	0.45
0.0004	0.53	0.23	0.44
0.0003	0.28	0.82	0.92

Table 12. Error convergence behavior of BPVAM algorithm in 150 iterations for iris dataset

η / λ/β	0.07	0.06	0.05
0.04 0.9991	22.05	22.05	22.05
0.03 0.999	22.05	22.05	22.05
0.02 0.998	0.242	0.244	0.244

5. Lung-cancer Dataset

Table 13. Error convergence behavior of BP algorithm in 150 iterations for lung-cancer dataset

η / α	0.3	0.2	0.1
0.07	0.83	0.154	0.97
0.06	0.83	0.154	1.02
0.05	0.84	0.154	1.01

Table 14. Error convergence behavior of BPAM algorithm in 150 iterations for lung-cancer dataset

η / α	0.3	0.2	0.1
0.00003	0.04	0.09	1.53
0.00002	0.09	0.1	0.87
0.00001	0.05	0.21	1.12

Table 15. Error convergence behavior of BPVAM algorithm in 150 iterations for lung-cancer dataset

η / λ/β	0.7	0.8	0.9
0.007 0.9991	4.73	4.75	4.75
0.006 0.999	0.023	0.05	0.108
0.005 0.998	0.077	0.12	0.21

6. Wine Dataset

Table 16. Error convergence behavior of BP algorithm in 150 iterations for wine dataset

η / α	0.7	0.8	0.9
0.03	0.98	0.97	0.94
0.02	0.98	0.964	0.95
0.01	0.99	0.98	0.96

Table 17. Error convergence behavior of BPAM algorithm in 150 iterations for wine dataset

η / α	0.7	0.8	0.9
0.00003	0.37	0.33	0.964
0.00002	0.7	0.37	0.96
0.00001	0.98	0.9	0.95

Table 18. Error convergence behavior of BPVAM algorithm in 150 iterations for wine dataset

η / λ/β	0.7	0.8	0.9
0.0004 0.99991	0.99	0.986	0.95
0.0003 0.9999	0.99	0.93	0.81
0.0002 0.9998	0.97	0.87	0.73

APPENDIX-2 Error performance (QE) of the conventional SOM, GF-SOM, PLSOM and proposed SOM algorithms are compared using different δ_α, δ_σ, β and λ parameters in 20 iterations.

1. Appendicitis dataset

Table 1. QE results of the conventional SOM algorithm for appendicitis dataset

δ_α	0.57	0.56	0.55
QE	0.142	0.142	0.1416

Table 2. QE results of the GF-SOM algorithm for appendicitis dataset

δ_α / δ_σ	0.7	0.6	0.5
0.003	0.145	0.142	0.1442
0.002	0.145	0.142	0.1442
0.001	0.145	0.142	0.144

Table 3. QE results of the PLSOM algorithm for appendicitis dataset

β	1.16	1.15	1.14
QE	0.173	0.172	0.171

Table 4. QE results of the proposed SOM algorithm for appendicitis dataset

λ / β	0.00064	0.00063	0.00062
0.996	0.14	0.125	0.143
0.995	0.13	0.132	0.142
0.994	0.123	0.141	0.132

2. Balance Dataset

Table 5. QE results of the conventional SOM algorithm for balance dataset

δ_α	0.17	0.16	0.15
QE	0.24	0.25	0.24

Table 6. QE results of the GF-SOM algorithm for balance dataset

δ_α / δ_σ	1	0.9	0.8
0.0087	0.242	0.239	0.24
0.0086	0.242	0.24	0.24
0.0085	0.242	0.238	0.24

Table 7. QE results of the PLSOM algorithm for balance dataset

β	4	3	2
QE	0.22	0.22	0.23

Table 8. QE results of the proposed SOM algorithm for balance dataset

λ / β	0.005	0.004	0.003
0.992	0.2	0.21	0.2
0.991	0.2	0.21	0.216
0.99	0.22	0.22	0.2

3. Wisconsin Breast Dataset

Table 9. QE results of the conventional SOM algorithm for wisconsin dataset

δ_α	0.5	0.4	0.3
QE	0.1485	0.1485	0.1484

Table 10. QE results of the GF-SOM algorithm for wisconsin dataset

δ_α / δ_σ	0.3	0.2	0.1
1.7	0.1484	0.1484	0.1484
1.6	0.1484	0.1484	0.1484
1.5	0.1484	0.1484	0.1484

Table 11. QE results of the PLSOM algorithm for wisconsin dataset

β	0.8	0.7	0.6
QE	0.188	0.183	0.18

Table 12. QE results of the proposed SOM algorithm for wisconsin dataset

λ / β	0.0017	0.0016	0.0015
0.97	0.1483	0.1483	0.1482
0.96	0.1481	0.1481	0.148
0.95	0.148	0.148	0.1479

4. Dermatology Dataset

Table 13. QE results of the conventional SOM algorithm for dermatology dataset

δ_α	0.6	0.5	0.4
QE	0.181	0.181	0.18

Table 14. QE results of the GF-SOM algorithm for dermatology dataset

δ_α / δ_σ	0.6	0.5	0.4
0.04	0.179	0.177	0.177
0.03	0.181	0.181	0.182
0.02	0. 18	0.18	0.177

Table 15. QE results of the PLSOM algorithm for dermatology dataset

β	13.7	13.6	13.5
QE	0.188	0.177	0.176

Table 16. QE results of the proposed SOM algorithm for dermatology dataset

λ / β	0.007	0.006	0.005
0.992	0.174	0.179	0.184
0.991	0.176	0.176	0.179
0.99	0.18	0.1732	0.173

5. Ionosphere Dataset

Table 17. QE results of the conventional SOM algorithm for ionosphere dataset

δ_α	0.6	0.5	0.4
QE	0.1	0.1	0.1

Table 18. QE results of the GF-SOM algorithm for ionosphere dataset

δ_α / δ_σ	0.4	0.3	0.2
7	0.1	0.1	0.09
6	0.1	0.1	0.09
5	0.1	0.1	0.1

Table 19. QE results of the PLSOM algorithm for ionosphere dataset

β	2.1	2	1.9
QE	0.118	0.117	0.117

Table 20. QE results of the proposed SOM algorithm for ionosphere dataset

λ / β	0.0027	0.0026	0.0025
0.991	0.104	0.104	0.104
0.99	0.103	0.103	0.1
0.98	0.1	0.1	0.1

6. Iris Dataset

Table 21. QE results of the conventional SOM algorithm for iris dataset

δ_α	1.1	1	0.9
QE	0.2653	0.2655	0.265

Table 22. QE results of the GF-SOM algorithm for iris dataset

δ_α / δ_σ	0.6	0.5	0.4
0.03	0.3	0.2655	0.2881
0.02	0.2653	0.2655	0.2651
0.01	0.2653	0.2655	0.265

Table 23. QE results of the PLSOM algorithm for iris dataset

β	3.7	3.6	3.5
QE	0.267	0.2656	0.264

Table 24. QE results of the proposed SOM algorithm for iris dataset

λ / β	0.007	0.006	0.005
0.991	0.2654	0.2653	0.2652
0.99	0.2653	0.2652	0.2652
0.98	0.2651	0.265	0.265

7. Sonar Dataset

Table 25. QE results of the conventional SOM algorithm for sonar dataset

δ_α	0.9	0.8	0.7
QE	286.8	286.8	284.2

Table 26. QE results of the GF-SOM algorithm for sonar dataset

δ_α / δ_σ	0.3	0.29	0.28
0.03	283.6	283.5	283.4
0.02	283.6	283.5	283.4
0.01	283.6	283.5	283.4

Table 27. QE results of the PLSOM algorithm for sonar dataset

β	0.6	0.5	0.4
QE	241.5	233.3	230.3

Table 28. QE results of the proposed SOM algorithm for sonar dataset

λ / β	0.0000097	0.0000096	0.0000095
0.9992	223.5	224	223.5
0.9991	224.9	224.6	225.1
0.999	224.8	224.6	224.3

8. Wine Dataset

Table 29. QE results of the conventional SOM algorithm for wine dataset

δ_α	1.1	1	0.9
QE	0.137	0.138	0.133

Table 30. QE results of the GF-SOM algorithm for wine dataset

δ_α / δ_σ	0.8	0.7	0.6
2.6	0.134	0.132	0.153
2.5	0.133	0.13	0.158
2.4	0.152	0.13	0.13

Table 31. QE results of the PLSOM algorithm for wine dataset

β	1.6	1.5	1.4
QE	0.155	0.138	0.124

Table 32. QE results of the proposed SOM algorithm for wine dataset

λ / β	0.011	0.01	0.009
0.991	0.095	0.095	0.15
0.99	0.095	0.095	0.16
0.98	0.134	0.13	0.095

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Alaa Ali Hameed HAMEED
Uyruğu	**:**	Irak
Doğum Yeri ve Tarihi	**:**	Bağdat, 21 Ekim 1985
Telefon	**:**	+90 534 571 11 74
Faks	**:**	
e-mail	**:**	dr.alaa85@yahoo.com & alaa.hameed@selcuk.edu.tr

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	:	Al-Muthanna Lisesi, Bağdat	2005
Üniversite	:	Al-Mamon Üniversitesi, Bilgisayar Mühendisliği Teknikleri, Bağdat	2009
Yüksek Lisans	:	Doğu Akdeniz Üniversi, Bilgisayar Mühendisliği, Gazimağusa	2012
Doktora	:	Selçuk Üniversitesi, Bilgisayar Mühendisliği, Konya	2017

UZMANLIK ALANI

Dijital Sinyal ve Görüntü İşleme, Uyarlamalı Filtreler, Yapay Sinir Ağları ve Makine öğrenmesi.

YABANCI DİLLER

Arapça (anadili), Türkçe, İngilizce

YAYINLAR

Hameed, A. A., Salman, M. S. and Karlik, B., 2014, A new 2-D convex combination of recursive inverse algorithms, *IEEE 34th International Conference on Electronics and Nanotechnology (ELNANO),* 273-276.

Salman, M. S., Hameed, A. A., Turan, C. and Karlik, B., 2015, A new sparse convex combination of ZA-LLMS and RZA-LLMS algorithms. *In 2015 23nd Signal Processing and Communications Applications Conference (SIU),* 711-714 (Doktora tezinden).

Salman, M. S., Hameed, A. A. and Karlik, B., 2013, Convex combination ofrecursive inverse algorithms, *Turkish Journal of Electrical Engineering and Computer Sciences,* DOI: 10.3906/elk-1306-232 (Doktora tezinden).

Hameed, A. A., Karlik, B. and Salman, M. S., 2016, Back-propagation algorithm with variable adaptive momentum, *Knowledge-Based Systems*, 114, 79-87 (Doktora tezinden).

Hameed, A. A., Salman, M. S. and Karlik, B., Analysis of convexly combined recursive inverse algorithms, (under review), (Doktora tezinden).

Hameed, A. A., Karlik, B. and Salman, M. S, Robust adaptive learning approach of self-organizing maps, (under review), (Doktora tezinden).