SIViP (2017) 11:235–242
DOI 10.1007/s11760-016-0925-2

ORIGINAL PAPER

Computationally highly efficient mixture of adaptive filters

O. Fatih Kilic[1] · **M. Omer Sayin**[2] · **Ibrahim Delibalta**[3] · **Suleyman S. Kozat**[1]

Received: 31 December 2015 / Revised: 23 April 2016 / Accepted: 14 June 2016 / Published online: 4 July 2016

Abstract We introduce a new combination approach for the mixture of adaptive filters based on the set-membership filtering (SMF) framework. We perform SMF to combine the outputs of several parallel running adaptive algorithms and propose unconstrained, affinely constrained and convexly constrained combination weight configurations. Here, we achieve better trade-off in terms of the transient and steady-state convergence performance while providing significant computational reduction. Hence, through the introduced approaches, we can greatly enhance the convergence performance of the constituent filters with a slight increase in the computational load. In this sense, our approaches are suitable for big data applications where the data should be processed in streams with highly efficient algorithms. In the numerical examples, we demonstrate the superior performance of the proposed approaches over the state of the art using the well-known datasets in the machine learning literature.

✉ O. Fatih Kilic
kilic@ee.bilkent.edu.tr

M. Omer Sayin
sayin2@illinois.edu

Ibrahim Delibalta
ibrahim.delibalta@turktelekom.com.tr

Suleyman S. Kozat
kozat@ee.bilkent.edu.tr

[1] The Department of Electrical and Electronics Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey

[2] The Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Champaign, IL, USA

[3] Turk Telekom Communications Services Inc., Istanbul, Turkey

Keywords Big data · Computational reduction · Mixture approach · Set-membership filtering · Affine combination · Convex combination

1 Introduction

For certain adaptive filtering scenarios, we can select an appropriate adaptation algorithm with its parameters, e.g., the length of the filter or the learning rate, based on the a priori knowledge about the structure and statistics of the data model [1,2]. However, the performance of the algorithm might degrade severely due to the improper design in the lack of a priori information. As an example, conventional adaptive filtering algorithms, e.g., the least mean square (LMS) algorithm, in general demonstrate degraded performance in the impulsive noise environment, while the algorithms robust against impulsive interferences, e.g., the sign algorithm (SA), achieve inferior performance over the conventional algorithms in the impulse-free noise environments [3].

Recently, the mixture approaches have been proposed to combine various adaptive filters with different configurations to achieve better performance than any of the individual algorithm [1,4–11]. Particularly, through the mixture approach we can achieve enhanced performance in a wider range of adaptive filtering applications. The mixture model outputs a weighted linear combination of the output of various adaptive filtering algorithms such that the final output signal estimates better the desired signal. As those weights could be fixed with hindsight about the temporal data, we can also adapt those combination weights sequentially based on the observed data. However, we emphasize that the mixture approaches multiplicatively increase the combination load due to the need to run several adaptive algorithms in paral-

lel. Hence, these approaches cannot be used for applications involving big data due to their impractical computational need. To this end, in this paper, we introduce a mixture approach using the SMF in order to reduce computational load and achieve improved performance. In the conventional least squares algorithms, e.g., the LMS algorithm (or the stochastic gradient descent algorithm), we minimize a cost function of the error term defined as the difference between the desired and the estimated signals. On the contrary, the set-membership filtering approach seeks to find any parameter yielding smaller error terms than a predefined bound. SMF approach achieves relatively fast convergence performance in addition to the reduced computational load since we do not update the parameter unless we obtain larger error than the bound [12–14].

The organization of the paper as follows. In Sect. 2, first we present the main framework for mixture combination of adaptive filters. We describe the structure of set-membership filters and explain its algorithm in Sect. 3. In Sect. 4, we present unconstrained, affine and convex constrained combination methods for the set-membership filters. We demonstrate the performance of the presented method in Sect. 5 and later we conclude the paper with final remarks in Sect. 6.

2 Problem description

Considering an online setting where only the current feature vector[1] $\mathbf{x}(t)$ at time $t \geq 1$ is available for corresponding data $d(t)$. Our aim is to sequentially estimate $d(t)$ such that $\hat{d}(t) = f(\mathbf{x}(t))$, and for the estimation, in this work we use linear mixture of parallel adaptive filters.

In this structure, our system consists of two parts. In the first part, we have m adaptive filter algorithm running in parallel to estimate desired signal $d(t)$ as in Fig. 1. Each filter with their parameter vector $\mathbf{w}_i(t)$, $i = 1, \ldots, m$ and input vector $\mathbf{x}(t)$ produce an estimate $\hat{d}_i(t) = \mathbf{x}^T(t)\mathbf{w}_i(t)$, and in next step we update their parameter vector according to estimation error $e_i(t) \triangleq d(t) - \hat{d}_i(t)$

In second part of the system, we have the mixture stage. At this point, we obtain the final estimate of the system by linearly combining the estimates of parallel adaptive filters as $\hat{d}(t) = \mathbf{w}^T(t)\mathbf{y}(t)$ where $\mathbf{y}(t) = col\{\hat{d}_1(t), \ldots, \hat{d}_m(t)\}$ and $\mathbf{w}(t) = col\{w^{(1)}(t), \ldots, w^{(m)}(t)\}$ is mixture weights

[1] Through this paper, bold lower case letters denote column vectors and bold upper case letter denote matrices. For a vector $\mathbf{a}$ (or matrix $\mathbf{A}$), $\mathbf{a}^T$ (or $\mathbf{A}^T$) is its ordinary transpose. The operator col$\{\cdot\}$ produces a column vector or a matrix in which the arguments of col$\{\cdot\}$ are stacked one under the other. For a given vector $\mathbf{w}$, $\mathbf{w}^{(i)}$ denotes the ith individual entry of $\mathbf{w}$. Similarly for a given matrix $\mathbf{G}$, $\mathbf{G}^{(i)}$ is the ith row of $\mathbf{G}$. For a vector argument, diag$\{\cdot\}$ creates a diagonal matrix whose diagonal entries are elements of the associated vector.

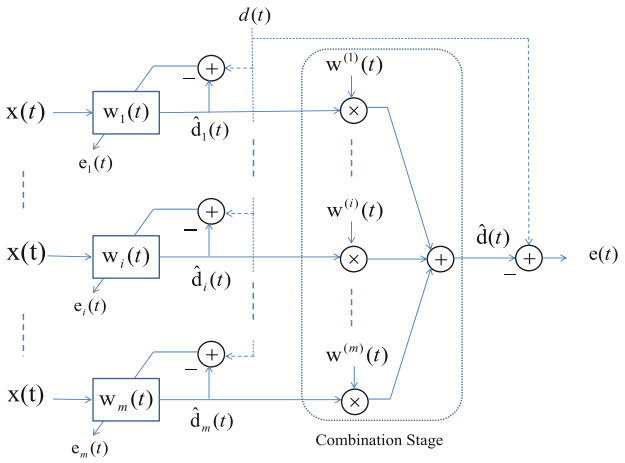

Fig. 1 Mixture combination of parallel filters

vector. Linear combination parameters of this stage are updated adaptively according to the final estimation error $e(t) \triangleq d(t) - \hat{d}(t)$.

Usage of conventional least squares algorithms such as least mean square algorithm in these mixture combination systems results in an update of parameter vectors at each step. This notion is not advantageous for most big data applications due to high computational load that this feature will create. Therefore, as a solution, we employ set-membership filters and their mixture combination for this structure.

In subsequent sections, we first introduce the structure of the set-membership filters (SMF), then we introduce methods for linear mixture combination of these set-membership filters.

3 Structure of set-membership filters

For the general linear-in-parameter filters whose input is $\mathbf{x} \in \mathbb{R}^n$, the desired output is real scalar d and the output of the filter is $\hat{d} = \mathbf{x}^T\mathbf{w}$ where $\mathbf{w} \in \mathbb{R}^n$ is the parameter vector for the filter, and the filter error is defined as $e(\mathbf{w}) = d - \hat{d}$. In the general setting, filter estimates the parameter vector to minimize the cost which is a function of the filter error [2]. However, in the set-membership filtering scheme, we update the parameter vector to satisfy a predefined upper bound γ on the filter error for all data pairs $(d, \mathbf{x})$ in a model space $\mathcal{S}$ such that

$$|e(\mathbf{w})|^2 \leq \gamma, \quad \forall (d, x) \in \mathcal{S}. \tag{1}$$

Therefore, any parameter vector satisfying (1) is an acceptable solution and the set of these solutions forms the feasibility set which is defined as

$$\Gamma \triangleq \bigcap_{(d,\mathbf{x})\in\mathcal{S}} \{\mathbf{w} \in \mathbb{R}^n : |d - \mathbf{x}^T\mathbf{w}|^2 \leq \gamma^2\}. \tag{2}$$

If the model space $\mathcal{S}$ is known priorly, then it is possible to estimate the feasibility set or a parameter vector in it. However, there is no closed-form solution for an arbitrary $\mathcal{S}$ and in practice the model space is not known completely or it is time-varying [12]. Therefore, we estimate the feasibility set or one of its members using set-membership adaptive recursive techniques (SMART).

Considering a practical case, where only measured data pair $(d_t, \mathbf{x}_t) \in \mathcal{S}$ is available, the constraint set $\mathcal{H}_t$ containing all parameter vectors satisfying (1) is defined as

$$\mathcal{H}(t) \triangleq \{\mathbf{w} \in \mathbb{R}^n : |d(t) - \mathbf{w}^T\mathbf{x}(t)| \leq \gamma\}. \tag{3}$$

Here, the constraint set is a region enclosed by the parallel hyperplanes defined with $|d(t) - \mathbf{x}(t)^T\mathbf{w}| = \gamma$ and an estimate for the feasibility set at time t is membership set $\phi_t \triangleq \bigcap_{\tau=1}^{t} \mathcal{H}(\tau)$. We approximate the membership set for tractable and computable results by projecting current parameter vector $\mathbf{w}(t)$ onto constraint set $\mathcal{H}(t+1)$ if it is not contained in it and assure an error upper bound of γ [12]. We express the problem defined above as

$$\mathbf{w}(t+1) = \arg \min_{\mathbf{w} \in \mathcal{H}(t+1)} \|\mathbf{w} - \mathbf{w}(t)\|^2. \tag{4}$$

We solved the optimization problem with constraint in (4) with the method of Lagrange multipliers. The Lagrangian to the optimization problem in (4) is

$$\mathcal{L}(\mathbf{w}, \tau) = \|\mathbf{w} - \mathbf{w}(t)\|^2 + \tau(|e(t)| - \gamma). \tag{5}$$

Solution to the Lagrangian in (5) is

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t)\frac{\mathbf{x}(t)e(t)}{\mathbf{x}^T(t)\mathbf{x}(t)} \tag{6}$$

where

$$\mu(t) = \begin{cases} 1 - \frac{\gamma}{|e(t)|} & \text{if } |e(t)| > \gamma, \\ 0 & \text{otherwise.} \end{cases}$$

The resulting algorithm in (6) is named as set-membership normalized least mean square algorithm (SM-NLMS) and achieves better convergence speed and steady-state MSE with reduced computational load than NLMS algorithm [12]. In next section, we use this SMF structure in constituent and combination filters of mixture combination approach to create computationally efficient and fast converging estimation system.

4 Proposed combination methods

We deploy SMF scheme for the mixture combination of constituent set-membership filters with different error bounds running in parallel to estimate the desired signal $d(t)$. We emphasize that using SMF scheme provides lower computational complexity which offers a comparable performance suitable for big data applications than standard LMS algorithms. Also we get benefits of fast converging and lower steady-state MSE performance obtained by using different bounds on constituent filters on our estimation. Also

We use a system where m SMF filter running in parallel as in Fig. 1, each one updates its parameter vector $\mathbf{w}_i(t) \in \mathbb{R}^n$ and produces estimate $\hat{d}_i(t) = \mathbf{x}^T(t)\mathbf{w}_i(t)$ with respect to its bound γ_i. In the combination stage of m constituent filters, we combine each filter output linearly through time variant weight vector $w(t)^{(i)} \in \mathbb{R}^m$ which is trained with combinator SMF filter with bound $\bar{\gamma}$. We denote input to the combination stage as $\mathbf{y}(t) \triangleq col\{\hat{d}_1(t), \ldots, \hat{d}_m(t)\}$, and the parameter vector of the combination stage is $\mathbf{w}(t) \triangleq col\{w^{(1)}(t), \ldots, w^m(t)\}$. The output of the combination stage is $\hat{d}(t) = \mathbf{y}^T(t)\mathbf{w}(t)$, and the final estimation error is $e(t) \triangleq d_t - \hat{d}(t)$.

In the following subsections, we seek and train parameter vectors for the combination stage weights satisfying upper bound $\bar{\gamma}$ within different parameter spaces.

4.1 Unconstrained linear mixture parameters

The first parameter space is for the unconstrained linear mixture weights and defined as $\mathcal{W}_1 \triangleq \{\mathbf{w} \in \mathbb{R}^m\}$ which is the Euclidean space. Therefore, within the SMF scheme, for finding and update of the weights we have

$$\mathbf{w}(t+1) = \arg \min_{\mathbf{w} \in \mathcal{H}_1(t)} ||\mathbf{w} - \mathbf{w}(t)||^2 \tag{7}$$

where $\mathcal{H}_1(t) \triangleq \{\mathbf{w} \in \mathcal{W}_1 : |d(t) - \mathbf{w}^T\mathbf{y}(t)| \leq \bar{\gamma}$ is the constraint set for the update and the solution for the (7) as we did in (4) yields

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t)\frac{\mathbf{y}(t)e(t)}{\mathbf{y}^T(t)\mathbf{y}(t)} \tag{8}$$

where

$$\mu(t) = \begin{cases} 1 - \frac{\bar{\gamma}}{|e(t)|} & \text{if } |e(t)| > \bar{\gamma}, \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm for the unconstrained mixture method is given in Algorithm 1.

4.2 Affine mixture parameters

Parameter space for the affine mixture weights is defined as $\mathcal{W}_2 \triangleq \{\mathbf{w} \in \mathbb{R}^m : \mathbf{1}^T\mathbf{w} = 1\}$ where $\mathbf{1} \in \mathbb{R}^m$ denotes a vector of ones such that sum of weights to be one, i.e., $\sum_{i=1}^{m} w^{(i)} = 1$. Therefore, the constraint set in this case is

$$\mathcal{H}_2(t) \triangleq \{\mathbf{w} \in \mathcal{W}_2 : |d(t) - \mathbf{w}^T\mathbf{y}(t)| \leq \bar{\gamma}\}.$$

Algorithm 1 The Set-Membership Unconstrained Mixture Algorithm

Choose $\bar{\gamma}$
$\mathbf{w}(0) \leftarrow$ *Initialize*
$\alpha \leftarrow$ *Constant*
for $i = 1$ **to** m **do**
  $\mathbf{w}_i(0) \leftarrow$ *Initialize*
  Choose γ_i
end for
for all $t \geq 0$ **do**
  for $i = 1$ **to** m **do**
    $\hat{d}_i(t) = \mathbf{x}^T(t)\mathbf{w}_i(t)$
    $e_i(t) = d(t) - \hat{d}_i(t)$
    if $|e_i(t)| > \gamma_i$ **then**
      $\mu_i(t) = 1 - \frac{\gamma_i}{|e_i(t)|}$
      $\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \mu_i(t)\frac{\mathbf{x}(t)e_i(t)}{\alpha + \mathbf{x}^T(t)\mathbf{x}(t)}$
    end if
  end for
  $\mathbf{y}(t) = [\hat{d}_1(t) \ldots \hat{d}_m(t)]^T$
  $\hat{d}(t) = \mathbf{y}^T(t)\mathbf{w}(t)$
  $e(t) = d(t) - \hat{d}(t)$
  if $|e(t)| > \bar{\gamma}$ **then**
    $\mu(t) = 1 - \frac{\bar{\gamma}}{|e(t)|}$
    $\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t)\frac{\mathbf{y}(t)e(t)}{\alpha + \mathbf{y}^T(t)\mathbf{y}(t)}$
  end if
end for

We remove the affine constraint with the following parametrization. Define parameter vector $\mathbf{z}(t) \in \mathbb{R}^{n-1}$ where

$$\mathbf{z}^{(i)}(t) \triangleq \mathbf{w}^{(i)}(t), \quad \forall i \in \{1, 2, \ldots, m-1\}$$

and

$$\mathbf{w}^{(m)}(t) = 1 - \sum_{i=1}^{m-1} \mathbf{z}^{(i)}(t) \tag{9}$$

Therefore, the final estimation error is expressed with the use of unconstrained parameter vector as

$$\begin{aligned} e(t) &= d(t) - \underbrace{\begin{bmatrix} \mathbf{z}^{(1)}(t) \\ \vdots \\ \mathbf{z}^{(m-1)}(t) \\ 1 - \mathbf{1}^T\mathbf{z}(t) \end{bmatrix}^T}_{\mathbf{w}(t)} \underbrace{\begin{bmatrix} \hat{d}_1(t) \\ \vdots \\ \hat{d}_{m-1}(t) \\ \hat{d}_m(t) \end{bmatrix}}_{\mathbf{y}(t)}, \\ &= d(t) - \begin{bmatrix} \hat{d}_1(t) \\ \vdots \\ \hat{d}_{m-1}(t) \end{bmatrix}^T \mathbf{z}(t) - (1 - \mathbf{1}^T\mathbf{z}(t))\hat{d}_m(t), \\ &= \underbrace{d(t) - \hat{d}_m(t)}_{a(t)} - \underbrace{\begin{bmatrix} \hat{d}_1(t) - \hat{d}_m(t) \\ \vdots \\ \hat{d}_{m-1}(t) - \hat{d}_m(t) \end{bmatrix}^T}_{\mathbf{c}(t)} \mathbf{z}(t). \end{aligned} \tag{10}$$

Here in (9), we present $z(t)$ as the unconstrained parameter vector, $a(t)$ as the desired signal and $c(t)$ as the input to the unconstrained optimization problem which is given as

$$\mathbf{z}(t+1) = \arg \min_{\mathbf{z} \in \widetilde{\mathcal{H}}_2(t)} \|\mathbf{z} - \mathbf{z}(t)\|^2, \tag{11}$$

where the constraint set is defined as $\widetilde{\mathcal{H}}_2(t) \triangleq \{\mathbf{z} \in \mathbb{R}^{m-1} : |a(t) - \mathbf{z}^T\mathbf{c}(t)| \leq \gamma\}$. Since now the optimization problem is same as in unconstrained case, as in (7) the solution yields

$$\mathbf{z}(t+1) = \mathbf{z}(t) + \mu(t)\frac{\mathbf{c}(t)e(t)}{\mathbf{c}(t)^T\mathbf{c}(t)} \tag{12}$$

where

$$\mu(t) = \begin{cases} 1 - \frac{\gamma}{|e(t)|} & \text{if } |e(t)| > \gamma, \\ 0 & \text{otherwise.} \end{cases}$$

The input vector $\mathbf{c}(t)$ to the re-parameterized unconstrained version of the optimization problem can be expressed in terms of initial input vector $y(t)$ as

$$\mathbf{c}(t) = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 & -1 \\ 0 & 1 & \cdots & 0 & -1 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & -1 \end{bmatrix}}_{\widetilde{\mathbf{G}}} \mathbf{y}(t)$$

Therefore, we can express each element of unconstrained parameter vector as

$$\mathbf{z}^{(i)}(t+1) = \mathbf{z}^{(i)}(t) + \mu(t)\frac{\widetilde{\mathbf{G}}^{(i)}\mathbf{y}(t)e(t)}{\mathbf{y}(t)^T\widetilde{\mathbf{G}}^T\widetilde{\mathbf{G}}\mathbf{y}(t)} \tag{13}$$

which leads to

$$1 - \sum_{i=1}^{m-1} \mathbf{z}^{(i)}(t+1) = 1 - \sum_{i=1}^{m-1} \mathbf{z}^{(i)}(t) - \mu(t)\frac{\sum_{i=1}^{m-1} \widetilde{\mathbf{G}}^{(i)}\mathbf{y}(t)e(t)}{\mathbf{y}(t)^T\widetilde{\mathbf{G}}^T\widetilde{\mathbf{G}}\mathbf{y}(t)} \tag{14}$$

and inserting (9) leads to

$$\mathbf{w}^{(m)}(t+1) = \mathbf{w}^{(m)}(t) + \mu(t) \underbrace{\begin{bmatrix} -1 \\ \vdots \\ -1 \\ m-1 \end{bmatrix}^T}_{\mathbf{g}} \frac{\mathbf{y}(t)e(t)}{\mathbf{y}(t)^T\widetilde{\mathbf{G}}^T\widetilde{\mathbf{G}}\mathbf{y}(t)}. \tag{15}$$

Thus, by (13) and (15), we have

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t) \underbrace{\begin{bmatrix} \widetilde{\mathbf{G}} \\ \mathbf{g}^T \end{bmatrix}}_{\mathbf{G}} \frac{\mathbf{y}(t)e(t)}{\mathbf{y}(t)^T\widetilde{\mathbf{G}}^T\widetilde{\mathbf{G}}\mathbf{y}(t)}. \tag{16}$$

Note that $\widetilde{\mathbf{G}}^T\widetilde{\mathbf{G}} = \mathbf{G}$, therefore Eq. (15) yields to parameter vector update of

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t)\frac{\mathbf{G}\mathbf{y}(t)e(t)}{\mathbf{y}(t)^T\mathbf{G}\mathbf{y}(t)} \tag{17}$$

where

$$\mathbf{G} \triangleq \begin{bmatrix} \mathbf{I}_{m-1} & -\mathbf{1} \\ -\mathbf{1}^T & m-1 \end{bmatrix}$$

and

$$\mu(t) = \begin{cases} 1 - \frac{\gamma}{|e(t)|} & \text{if } |e(t)| > \gamma, \\ 0 & \text{otherwise.} \end{cases}$$

and $-\mathbf{1} \in \mathbb{R}^{m-1}$ is a vector where all its elements are minus one. Note that, algorithm for affine combination is easily obtained by introducing matrix

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_{m-1} & -\mathbf{1} \\ -\mathbf{1}^T & m-1 \end{bmatrix}$$

and replacing the line 22 in Algorithm 1 with the update line $\mathbf{w}(t+1) = \mathbf{w}(t) + \mu(t)\frac{G\mathbf{y}(t)e(t)}{\alpha + \mathbf{y}(t)^T G\mathbf{y}(t)}$.

4.3 Convex mixture parameters

Lastly, the parameter space for the convex mixture weights is defined as $\mathcal{W}_3 = \{\mathbf{w} \in \mathbb{R}^m : \mathbf{1}^T\mathbf{w} = 1 \wedge \mathbf{w}^{(i)} \geq 0, \forall i \in \{1, \ldots, m\}\}$ In order to get unconstrained optimization problem as we did above, we re-parameterize vector $\mathbf{w}(t)$ with the parameter vector $\mathbf{z}(t) \in \mathbb{R}^m$ as in [1]

$$\mathbf{w}^{(i)}(t) = \frac{e^{-\mathbf{z}^{(i)}(t)}}{\sum_{k=1}^{m} e^{-\mathbf{z}^{(k)}(t)}}. \tag{18}$$

Note that SM-NLMS algorithm also could be constructed through gradient descent method with stochastic cost function defined as

$$F(e(t)) \triangleq \begin{cases} \left(\frac{|e(t)|-\gamma}{\|\mathbf{y}(t)\|}\right)^2 & |e(t)| > \gamma \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, for the unconstrained parameter vector update, stochastic gradient algorithm is given by

$$\mathbf{z}(t+1) = \mathbf{z}(t) - \frac{1}{2}\nabla_{\mathbf{z}} F(e(t)) \tag{19}$$

which by chain rule yields to

$$\mathbf{z}(t+1) = \mathbf{z}(t) - \frac{1}{2}[\nabla_{\mathbf{z}}\mathbf{w}(t)]^T \nabla_{\mathbf{w}} F(e(t)). \tag{20}$$

Note that $\nabla_{\mathbf{z}}\mathbf{w}(t) = \mathbf{w}(t)\mathbf{w}(t)^T - \text{diag}\{\mathbf{w}(t)\}$ [1] and by this we obtain

$$\mathbf{z}(t+1) = \mathbf{z}(t) + \mu(t)[\mathbf{w}(t)\mathbf{w}(t)^T - \text{diag}\{\mathbf{w}(t)\}]\frac{\mathbf{y}(t)e(t)}{\mathbf{y}(t)^T\mathbf{y}(t)} \tag{21}$$

where

$$\mu(t) = \begin{cases} 1 - \frac{\gamma}{|e(t)|} & \text{if } |e(t)| > \gamma, \\ 0 & \text{otherwise.} \end{cases}$$

and

$$\mathbf{w}(t) = \frac{e^{-\mathbf{z}(t)}}{\|e^{-\mathbf{z}(t)}\|_1}.$$

Finally, we easily obtain the algorithm for the convex mixture method by defining unconstrained parameter vector as in (18) and by replacing line 22 in Algorithm 1 with the update line in (20).

With the algorithms defined above, in next section we evaluate the MSE performance of the algorithms within different schemes.

5 Simulations and results

In this section, through series of simulations, we demonstrate the performance of the proposed SMF filter mixture algorithms and compare the steady-state and convergence performances with various methods, i.e., NLMS, variable step size NLMS and affine projection algorithm, as well as its superior computational efficiency [2,15]. We first considered the performance for stationary case where statistics of source data is not changing, and with stationary data, we also analyzed how predetermined error bounds of SMFs effects the performance of SMF mixture system. We also investigated the cases with non-stationary data where sudden changes happen in source statistics, and the power of the additive noise is also changing. Then, we demonstrate simulations with real and synthetic benchmark datasets such as Elevators and Kinematics data [16]. In the final part, we compare computational load of the proposed algorithms with respect to NLMS mixture algorithm and other state-of-the-art algorithms to demonstrate the computational efficiency of our solutions.

Through this section, we refer set-membership normalized least mean square algorithm as "SM-NLMS" and unconstrained, affine and convex mixture of these filters as "SM-UNC," "SM-AFF" and "SM-CONV," respectively. We also introduce variable step size NLMS algorithm as "VSS-NLMS" and affine projection algorithm as "APA" [2,15].

5.1 Stationary data

In this part, we study our algorithms in a stationary environment where data source statistics do not change over time. We create a sequence considering a linear-in-parameter model $d_t = \mathbf{w}_o^T \mathbf{x}_t + n_t$ where $\mathbf{w}_o \in \mathbb{R}^7$ denotes the parameter of interest, $\mathbf{x}_t \in \mathbb{R}^7$ is the input regressor vector and n_t is the additive white Gaussian noise signal with fixed variance σ_n^2. We use input vectors with eigenvalue spread of 1 and 0 dB SNR signal. Parameter of interest chosen randomly from normal distribution and normalized to $||\mathbf{w}_o|| = 1$. We use 10 constituent SM-NLMS filters with different error-bound set around $\sqrt{5\sigma_n^2}$. For comparison, we used NLMS mixture algorithm and a single NLMS algorithm with step size $\mu_{\text{NLMS}} = 0.2$, VSS-NLMS algorithm with step size range $(\mu_{\max}, \mu_{\min}) = (0.2, 0.02)$ and APA algorithm of order 5. In Fig. 2, we demonstrated the time-accumulated regression errors averaged over 100 independent trials. We observe that, SMF and NLMS mixture of set-membership filters outperform other filters (NLMS, VSS-NLMS and APA) in both convergence rate and residual error sense. Also, note that SMF mixture algorithms achieve better steady-state error than the NLMS mixture algorithm.

In addition, error-bound selection is indeed a problem for set-membership filtering (SMF), especially when the power of the noise of the environment is unknown. One of our main motivation for using the mixture approach with SMF is to resolve this problem by combining different SMFs with a wide range of representative error bounds. Hence, in the first stage we use diverse range of error bounds to cover nearly every important realistic case. However, we emphasize that the selection of the error bound in the final stage is important. The error bound of the mixture filter determines the trade-off between low residual error and low computational complexity. Therefore, it should be selected based on the application specifications. For instance, if we seek a low residual error

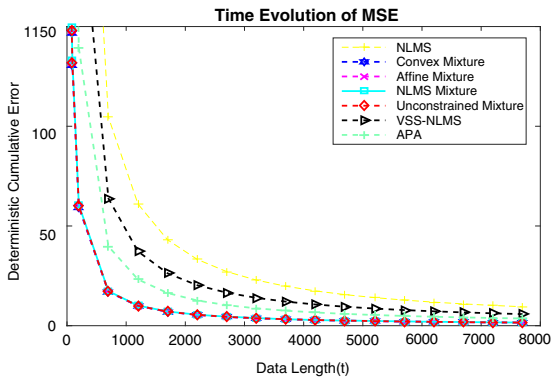

Fig. 2 Time-accumulated error performance of proposed algorithms compared with other algorithms over stationary data having 0 dB SNR and input vector eigenvalue spread of 1

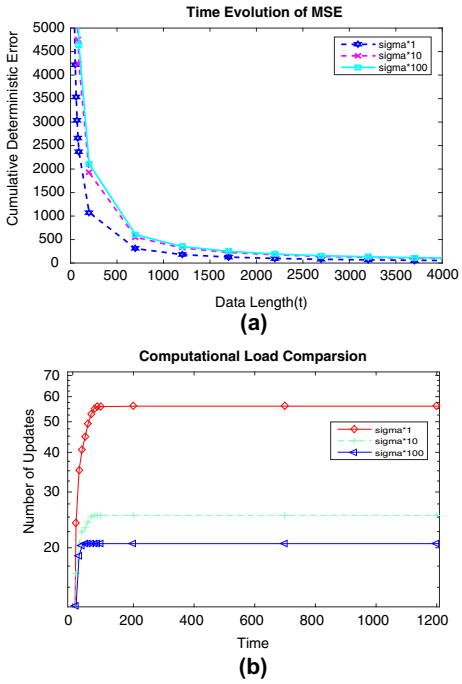

Fig. 3 Analysis for different error-bound selections. **a** Evolution of MSE for different final error-bound selections. **b** Number of update required for different final error-bound selections

and computational load is not a concern, then we set a tight bound and system updates itself until reaching the desired bound. For another case, if we seek for convergence with a low computational complexity, then we set a loose bound and system stops updating after converging to the bound. Therefore, here, we study the selection of the final stage error bound in a stationary environment. We use unconstrained mixture of constituent filters as a combination filter. For comparison, we set the error bound of the final stage as $\sqrt{5\sigma_n^2}$, $10\sqrt{5\sigma_n^2}$ and $100\sqrt{5\sigma_n^2}$ for different cases. We present the evolution of MSE for different selection of final error bound in Fig. 3 and evolution of the number of updates they require in Fig. 3.

5.2 Non-stationary data

In this part, we study the proposed algorithms with non-stationary data where the statistics of source data have sudden changes, i.e., concept drift, and have additive noise with a time-varying power. For this purpose, we create a sequence with the model $d_t = \mathbf{w}_t^T \mathbf{x}_t + n_t$ where $\mathbf{w}_t \in \mathbb{R}^7$ represents the time-dependent parameter of interest and n_t is white Gaussian noise with time-varying variance σ_n^2. We generated the parameter of interest $\mathbf{w}_0$ as a normalized vector from normal distribution. We changed that parameter of interest to $-\mathbf{w}_0$ at the middle of the sequence to create the non-stationary environment. At that time we also changed the power of the additive noise signal to create the time-varying

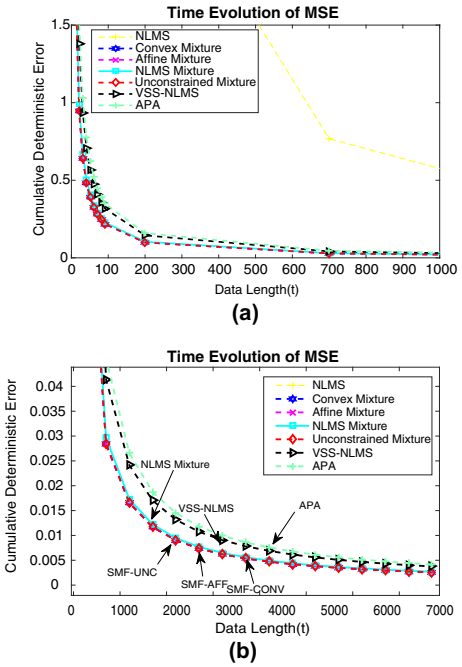

Fig. 4 Time-accumulated error performance of proposed algorithms compared with other algorithms over non-stationary data having 0 dB SNR and eigenvalue spread of 1. **a** Convergence rate behavior over first 1000 instances. **b** Residual error behavior over convergence instances

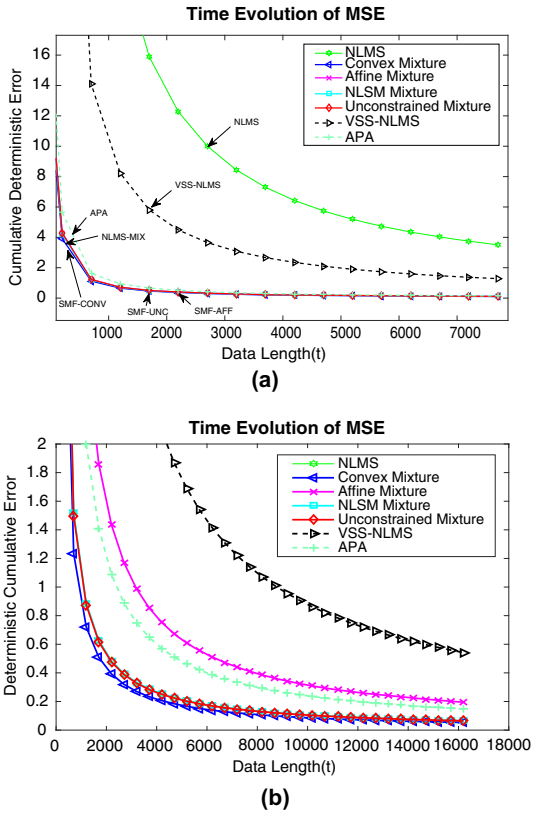

Fig. 5 Time-accumulated error performance of proposed algorithms compared with NLMS algorithms over Pumadyn and Elevator datasets. **a** Pumadyn dataset results. **b** Elevator dataset results

noise statistics. We created 8000 instances using this model configuration and set eigenvalue spread of the input vectors as 1. At the beginning, we set the SNR of signal as 0 dB, and at iteration 4000, we changed it to -10 dB. We use same filter configurations as the stationary case. We present accumulated error results averaged over 100 independent trials in Fig. 4. Here, we observe that mixture algorithms perform both in convergence rate and residual error sense better than other filters even for the non-stationary data with time-varying noise. Note that due to different error-bound coverage of the constituent filters of the mixture algorithms, we observe a robust performance under non-stationary data and time-varying noise circumstances which resulted in better performance than single use of filters.

5.3 Benchmark real data

Here, we apply our algorithms to the regression of the benchmark real-life problems [16]. In real-life dataset experiments, we use 10 constituent SMF filters, and since this time we do not know the power of the additive noise, we set the error bounds of the SMF filters in a wide range spread around 0.15 and again we choose the error bound for the combinator SMF filter as 0.15. For NLMS algorithms, we choose step size $\mu_{\text{NLMS}} = 0.2$. For VSS-NLMS algorithm, we set the step size range as $(\mu_{\max}, \mu_{\min}) = (0.2, 0.02)$ and for APA algorithm we choose its order different for each dataset according to their regressor dimension. We make 100 trials over a dataset by shuffling the data at each trial. For the first experiment, we use Pumadyn data with regressor dimension $n = 32$ which is a dataset obtained from a realistic simulations of the dynamics of Unimation Puma 560 robot arm [16]. We set the order of APA algorithm as 10 for this case. We present the accumulated error results averaged over 100 trials in Fig. 5. Note that in Fig. 5, mixture approaches show superior performance over other filters. Although APA algorithm shows a close performance to mixture filters, we emphasize that APA algorithm is computationally inefficient for big data applications compared to proposed methods since it requires memory for holding old data at its order and require more multiplication and addition operations at each update. We present detailed results for that in the computational load analysis part.

Besides Pumadyn experiment, we use Elevator data with regressor dimension 18 which is a dataset obtained from the task of controlling F16 aircraft and the desired data is related to an action taken on the elevators of the aircraft [16]. We set the order of APA algorithm as 8 for this case. We presented the results for this dataset in Fig. 5, and we should emphasize that similar behavior in results is observed.

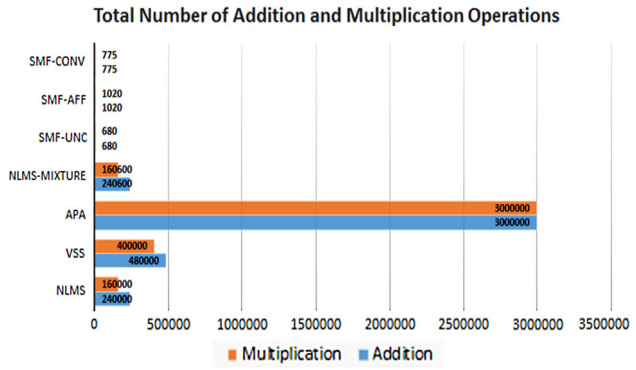

Fig. 6 Number of updates that each algorithm requires over 8000 instance stationary data

5.4 Computational load

One of the critical aspects of the proposed algorithms is the reduced computational load regarding lessened update of weights compared to the standard NLMS algorithm and mixture methods. To present that, we calculated the total number of addition and multiplication operation that each algorithm made during the simulation. In Fig. 6, we demonstrate results for addition and multiplication operation that each algorithm made in 100 independent experiment over stationary data and show that proposed algorithms are computationally more efficient than other algorithms. Although the computational cost among the proposed algorithms do not differ much, we emphasize that the unconstrained mixture is the most computationally efficient one. We note that SMF mixture algorithms provide computational efficiency up to order of magnitude of 3.

6 Conclusion

In this paper, we introduce a novel mixture of expert algorithm in order to reduce the computational demand of the mixture approaches. Since the ordinary mixture approaches are required to run several adaptive filters in parallel, they are impractical in applications involving big data due to complexity issues. To this end, by using the SMF, we significantly reduce the computational complexity of these approaches while providing superior performance. We provide unconstrained, affine and convex mixture weight configurations using set-membership filtering framework. Through numerical experiments in stationary and non-stationary environments and through regression of a benchmark real-life problem, we investigate the steady-state mean square error and convergence rate performance of these algorithms compared with other algorithms and mixture methods. In these experiments, we demonstrate that proposed algorithms reach faster convergence rate and lower steady-state error. Finally, we show that our set-membership filtering-based approaches requires less addition and multiplication operations hence less computational load than the compared algorithms.

References

1. Kozat, S.S., Erdogan, A.T., Singer, A.C., Sayed, A.H.: Steady-state MSE performance analysis of mixture approaches to adaptive filtering. IEEE Trans. Signal Process. **58**, 4050–4063 (2010)
2. Sayed, A.H.: Fundamentals of Adaptive Filtering. Wiley, NJ (2003)
3. Sayin, M.O., Denizcan Vanli, N., Kozat, S.S.: A transient analysis of affine mixtures. IEEE Trans. Signal Process. **59**(5), 6227–6232 (2011)
4. Kozat, S.S., Erdogan, A.T., Singer, A.C., Sayed, A.H.: A transient analysis of affine mixtures. IEEE Trans. Signal Process. **59**(5), 6227–6232 (2011)
5. Donmez, M.A., Kozat, S.S.: Steady-state MSE analysis of convexly constrained mixture methods. IEEE Trans. Signal Process. **60**(5), 3314–3321 (2012)
6. Arenas-Garcia, J., Figueiras-Vidal, A.R., Sayed, A.H.: Mean-square performance of a convex combination of two adaptive filters. IEEE Trans. Signal Process. **54**(3), 1078–1090 (2006)
7. Arenas-Garcia, J., Gomez-Verdejo, V., Figueiras-Vidal, A.R.: New algorithms for improved adaptive convex combination of LMS transversal filters. IEEE Trans. Instrum. Meas. **54**(6), 2239–2249 (2005)
8. Arenas-Garcia, J., Martinez-Ramon, M., Gomez-Verdejo, V., Figueiras-Vidal, A.R.: Multiple plant identifier via adaptive LMS convex combination. In: 2003 IEEE International Symposium on Intelligent Signal Processing, pp. 137–142 (2003)
9. Nascimento, V.H., Silva, M.T.M., Arenas-Garcia, J.: A low-cost implementation strategy for combinations of adaptive filters. In: Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5671–5675 (2013)
10. Lu, J., Hoi, S., Zhao, P.: Second order online collaborative filtering. In: Proceedings of Asian Conference on Machine Learning (ACML), pp. 40–55 (2013)
11. Ling, G., Yang, H.: Online learning for collaborative filtering. In: The International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2012)
12. Gollamudi, S., Nagaraj, S., Kapoor, S., Huang, Y.F.: Set-membership filtering and a set-membership normalized LMS algorithm with an adaptive step size. IEEE Signal Process. Lett. **5**(5), 111–114 (1998)
13. Diniz, P.S.R., Werner, S.: Set-membership binormalized data-reusing LMS algorithms. IEEE Trans. Signal Process. **51**(1), 124–134 (2003)
14. Koby, C., Ofer, D., Joseph, K., Shai, S.-S., Yoram, S.: Online passive-aggressive algorithms. J. Mach. Learn. Res. **7**, 551–585 (2006)
15. Zhao, H., Yu, Y.: Novel adaptive vss-nlms algorithm for system identification. In: 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP), June 2013, pp. 760–764 (2013)
16. Alcala-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., Garca, S., Snchez, L., Herrera, F.: Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J. Mult. Valued Logic Soft Comput. **17**(2–3), 255–287 (2011)