ADAPTIVE MIXTURE METHODS USING BREGMAN DIVERGENCES

Huseyin A. Inan, Mehmet A. Donmez, Suleyman S. Kozat, Senior Member, IEEE

Koc University
Istanbul, Turkey
Email: huseyin.inan@boun.edu.tr, {mdonmez,skozat}@ku.edu.tr

ABSTRACT

We investigate affinely constrained mixture methods adaptively combining outputs of m constituent filters running in parallel to model a desired signal. We use Bregman divergences and obtain multiplicative updates to train these linear combination weights under the affine constraints. We use the unnormalized relative entropy and the relative entropy that produce the exponentiated gradient update with unnormalized weights (EGU) and the exponentiated gradient update with positive and negative weights (EG), respectively. We carry out the mean and the mean-square transient analysis of the affinely constrained mixtures of m filters using the EGU or EG algorithms. We compare performances of different algorithms through our simulations and illustrate the accuracy of our results.

I. INTRODUCTION

We study mixture methods based on Bregman divergences that adaptively combine outputs of adaptive filters running in parallel to model a desired signal. The overall system has two stages. The first stage contains adaptive filters running in parallel to model a desired signal. In the second stage, the outputs of these adaptive filters are linearly combined to produce the final output of the overall system. We use Bregman divergences and obtain multiplicative updates [1] to train these linear combination weights under affine constraints [2]. As Bregman divergences, we use the unnormalized relative entropy and the relative entropy [1] that produce the exponentiated gradient update with unnormalized weights (EGU) and the exponentiated gradient update with positive and negative weights (EG), respectively. We emphasize that to the best of our knowledge, this is the first mean and mean-square transient analysis of the affinely constrained of m filters using the EGU or EG algorithms in the mixture framework (which also naturally covers unconstrained mixtures and the classical approach [3]). We illustrate the accuracy of our analysis in our simulations and demonstrate advantages of these algorithms for sparse mixture systems.

Adaptive combination methods are used in order to improve the steady-state and/or convergence performance over the constituent algorithms [2], [4], [5]. Affine combination of adaptive filters is studied in [2], where it is shown that the mean-square deviation of the affine combination can be made less than the mean-square deviation of the constituent filters in certain situations. The transient analysis of the affine combinations is carried out in [6].

Unlike the previous approaches to train the mixture weights, we use the family of EG algorithms here which are shown to converge faster than the LMS algorithm when the system impulse response is sparse, e.g., in network and acoustic echo cancellation problems [7]. Similarly, in our simulations, we observe that using the EG algorithm to train the mixture weights yields better performance compared to using the LMS algorithm to train the mixture weights for the combinations having more than 2 filters and when the combination favors only a few of the constituent filters.

The organization of the paper is as follows. In Section II, we first describe the mixture framework for the combination of adaptive filters. In Section III, we study the affinely constrained mixture methods with the EGU and EG algorithms. In Section IV, we perform the transient analysis of the affinely constrained mixtures using the EGU and EG algorithm. Finally, in Section V, we perform simulations to show the accuracy of our results and to compare the performances of the different algorithms

II. SYSTEM DESCRIPTION

The framework that we study has two stages[1]. In the first stage, we have m adaptive filters producing outputs $\hat{y}_i(t)$, $i = 1, \ldots, m$, running in parallel to model a desired signal $y(t)$ as seen in Fig. 1. The second stage is the mixture stage where the outputs of the first stage filters are combined to improve either steady-state or transient and/or tracking performance over the constituent filters. We linearly combine

[1]In this paper, all vectors are column vectors and represented by boldface lowercase letters. Matrices are represented by boldface capital letters. Given a vector $\boldsymbol{w}$, $w^{(i)}$ denotes the ith individual entry of $\boldsymbol{w}$, $\|\boldsymbol{w}\| \triangleq \sqrt{\boldsymbol{w}^T\boldsymbol{w}}$ is the l_2 norm. For a vector $\boldsymbol{w}$, $\mathrm{diag}(\boldsymbol{w})$ represents a diagonal matrix formed using the entries of $\boldsymbol{w}$. For a matrix $\boldsymbol{W}$, $\mathrm{diag}(\boldsymbol{W})$ represents a column vector that contains the diagonal entries of $\boldsymbol{W}$. For two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, we define the concatenation $[\boldsymbol{v}_1; \boldsymbol{v}_2] \triangleq [\boldsymbol{v}_1^T \ \boldsymbol{v}_2^T]^T$. For a random variable v, $\bar{v}$ is the expected value. Vectors (or matrices) $\mathbf{1}$ and $\mathbf{0}$, with an abuse of notation, denote vectors (or matrices) of all ones or zeros, respectively, where the size of the vector (or the matrix) is understood from the context.

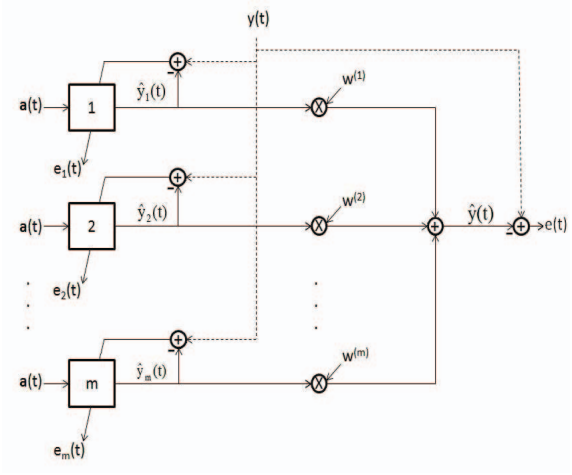

Fig. 1. A linear mixture of outputs of m adaptive filters.

the outputs of the first stage filters to produce the final output $\hat{y}(t) = \boldsymbol{w}^T(t)\boldsymbol{x}(t)$, where $\boldsymbol{x}(t) \stackrel{\triangle}{=} [\hat{y}_1(t), \ldots, \hat{y}_m(t)]^T$ and train the mixture weights using multiplicative updates (or exponentiated gradient updates). We point out that in order to satisfy the constraints and derive the multiplicative updates, we use reparametrization of the mixture weights as $\boldsymbol{w}(t) = \boldsymbol{f}(\boldsymbol{z}(t))$ and perform the update on $\boldsymbol{z}(t)$ as

$$\boldsymbol{z}(t+1) = \arg\min_{\boldsymbol{z}} \left\{ d(\boldsymbol{z}, \boldsymbol{z}(t)) + \mu\, l\big(y(t), \boldsymbol{f}^T(\boldsymbol{z})\boldsymbol{x}(t)\big) \right\}, \tag{1}$$

where μ is the learning rate of the update, $d(\cdot,\cdot)$ is an appropriate distance measure and $l(\cdot,\cdot)$ is the instantenous loss. We emphasize that in (1), the updated vector $\boldsymbol{z}$ is forced to be close to the present vector $\boldsymbol{z}(t)$ by $d(\boldsymbol{z}(t+1), \boldsymbol{z}(t))$, while trying to accurately model the current data by $l\big(y(t), \boldsymbol{f}^T(\boldsymbol{z})\boldsymbol{x}(t)\big)$. However, instead of directly minimizing (1), a linearized version of (1)

$$\boldsymbol{z}(t+1) = \arg\min_{\boldsymbol{z}} \left\{ d(\boldsymbol{z}, \boldsymbol{z}(t)) + l\left(y(t), \boldsymbol{f}^T(\boldsymbol{z}(t))\boldsymbol{x}(t)\right) \right. \tag{2}$$

$$\left. + \mu \nabla_{\boldsymbol{z}} l\left(y(t), \boldsymbol{f}^T(\boldsymbol{z})\boldsymbol{x}(t)\right)^T \Big|_{\boldsymbol{z}=\boldsymbol{z}(t)} (\boldsymbol{z} - \boldsymbol{z}(t)) \right\}$$

is minimized to get the desired update.

In the next section, we use the unnormalized relative entropy

$$d(\boldsymbol{z}, \boldsymbol{z}(t)) = \left\{ \sum_{i=1}^{m} \left[z^{(i)} \ln\left(\frac{z^{(i)}}{z^{(i)}(t)}\right) + z^{(i)}(t) - z^{(i)} \right] \right\} \tag{3}$$

for positively constrained $\boldsymbol{z}$ and $\boldsymbol{z}(t)$, $\boldsymbol{z} \in \mathbb{R}_+^m$, $\boldsymbol{z}(t) \in \mathbb{R}_+^m$ and the relative entropy

$$d_2(\boldsymbol{z}, \boldsymbol{z}(t)) = \left\{ \sum_{i=1}^{m} \left[z^{(i)} \ln\left(\frac{z^{(i)}}{z^{(i)}(t)}\right) \right] \right\} \tag{4}$$

where $\boldsymbol{z}$ is constraint to be in an extended simplex such that $z^{(i)} \geq 0$, $\sum_{k=1}^{m} z^{(i)} = u$ for some $u \geq 1$ as the distance measures, with appropriately selected $\boldsymbol{f}(\cdot)$ to derive updates on mixture weights under different constraints. We now investigate affinely constrained mixture of m adaptive filters using (3) and (4) as the distance measures.

III. AFFINELY CONSTRAINED MIXTURE

In this section, we investigate the affinely constrained mixture updated with the EGU and EG algorithms.

When an affine constraint is imposed on the mixture such that $\boldsymbol{w}^T(t)\mathbf{1} = 1$, we get

$$\hat{y}(t) = \boldsymbol{w}(t)^T\boldsymbol{x}(t), \;\; e(t) = y(t) - \hat{y}(t),$$
$$w^{(i)}(t) = \lambda^{(i)}(t), \;\; i = 1, \ldots, m-1,$$
$$w^{(m)}(t) = 1 - \sum_{i=1}^{m-1} \lambda^{(i)}(t),$$

where the $m-1$ dimensional vector $\boldsymbol{\lambda}(t) \stackrel{\triangle}{=} [\lambda^{(1)}(t), \ldots, \lambda^{(m-1)}(t)]^T$ is the unconstrained weight vector, i.e., $\boldsymbol{\lambda}(t) \in \mathbb{R}^{m-1}$. Using $\boldsymbol{\lambda}(t)$ as the unconstrained weight vector, the error can be written as $e(t) = \big[y(t) - \hat{y}_m(t)\big] - \boldsymbol{\lambda}^T(t)\boldsymbol{\delta}(t)$, where $\boldsymbol{\delta}(t) \stackrel{\triangle}{=} [\hat{y}_1(t) - \hat{y}_m(t), \ldots, \hat{y}_{m-1}(t) - \hat{y}_m(t)]^T$. To be able to derive a multiplicative update on $\boldsymbol{\lambda}(t)$, we use $\boldsymbol{\lambda}(t) = \boldsymbol{\lambda}_1(t) - \boldsymbol{\lambda}_2(t)$, where $\boldsymbol{\lambda}_1(t)$ and $\boldsymbol{\lambda}_2(t)$ are constrained to be nonnegative, i.e., $\boldsymbol{\lambda}_i(t) \in \mathbb{R}_+^{m-1}$, $i = 1, 2$. After we collect unconstrained weights in $\boldsymbol{\lambda}_a(t) = [\boldsymbol{\lambda}_1(t); \boldsymbol{\lambda}_2(t)]$, we define a function of loss $e(t)$ as $l_{\mathrm{a}}\left(\boldsymbol{\lambda}_a(t)\right) \stackrel{\triangle}{=} e^2(t)$ and update positively constrained $\lambda_a(t)$ as follows.

Unnormalized Relative Entropy: Using the unconstrained relative entropy as the distance measure, we obtain

$$\lambda_a^{(i)}(t+1) = \lambda_a^{(i)}(t) \exp\left\{\mu e(t)(\hat{y}_i(t) - \hat{y}_m(t))\right\}, \tag{5}$$
$$i = 1, \ldots, m-1,$$
$$\lambda_a^{(i)}(t+1) = \lambda_a^{(i)}(t) \exp\left\{-\mu e(t)(\hat{y}_i(t) - \hat{y}_m(t))\right\}, \tag{6}$$
$$i = m, \ldots, 2(m-1),$$

providing the multiplicative updates on $\boldsymbol{\lambda}_1(t)$ and $\boldsymbol{\lambda}_2(t)$.

Relative Entropy: Using the relative entropy as the distance measure, defining $r^{(i)}(t) \stackrel{\triangle}{=} \exp\left\{\mu e(t)(\hat{y}_i(t) - \hat{y}_m(t))\right\}$ and $h^{(i)}(t) \stackrel{\triangle}{=} \exp\left\{-\mu e(t)(\hat{y}_i(t) - \hat{y}_m(t))\right\}$, we obtain

$$\lambda_a^{(i)}(t+1) = \frac{u\lambda_a^{(i)}(t) r^{(i)}(t)}{\sum_{k=1}^{m-1}\left[\lambda_a^{(k)}(t) r^{(k)}(t) + \lambda_a^{(k+m-1)}(t) h^{(k)}(t)\right]}, \tag{7}$$
$$i = 1, \ldots, m-1,$$
$$\lambda_a^{(i)}(t+1) = \frac{u\lambda_a^{(i)}(t) h^{(i)}(t)}{\sum_{k=1}^{m-1}\left[\lambda_a^{(k)}(t) r^{(k)}(t) + \lambda_a^{(k+m-1)}(t) h^{(k)}(t)\right]}, \tag{8}$$
$$i = m, \ldots, 2(m-1),$$

providing the multiplicative updates on $\boldsymbol{\lambda}_a(t)$.

IV. TRANSIENT ANALYSIS

In this section, we first perform transient analysis of the mixture weights updated using the unconstrained relative entropy. Then, we continue with the transient analysis of the mixture weights updated using the relative entropy.

Unnormalized Relative Entropy: For the affinely constrained combination updated with the EGU algorithm, we have multiplicative updates (5) and (6). If $e(t)$ and $\hat{y}_i(t) - \hat{y}_m(t)$ for each $i = 1, \ldots, m-1$ are bounded, then we can write (5) and (6) as

$$\lambda_1^{(i)}(t+1) = \lambda_1^{(i)}(t)\big(1 + \mu e(t)(\hat{y}_i(t) - \hat{y}_m(t)) + O(\mu^2)\big), \quad (9)$$

$$\lambda_2^{(i)}(t+1) = \lambda_2^{(i)}(t)\big(1 - \mu e(t)(\hat{y}_i(t) - \hat{y}_m(t)) + O(\mu^2)\big), \quad (10)$$

for $i = 1, \ldots, m-1$, and since μ is small, we approximate (9) and (10) and obtain updates on $\boldsymbol{\lambda}_1(t)$ and $\boldsymbol{\lambda}_2(t)$ as

$$\boldsymbol{\lambda}_1(t+1) = \big(I + \mu e(t)\mathrm{diag}\big(\boldsymbol{\delta}(t)\big)\big)\boldsymbol{\lambda}_1(t), \quad (11)$$

$$\boldsymbol{\lambda}_2(t+1) = \big(I - \mu e(t)\mathrm{diag}\big(\boldsymbol{\delta}(t)\big)\big)\boldsymbol{\lambda}_2(t). \quad (12)$$

Now, collecting the weights in $\boldsymbol{\lambda}_a(t) = [\boldsymbol{\lambda}_1(t); \boldsymbol{\lambda}_2(t)]$, using the updates (11) and (12), we can write update on $\boldsymbol{\lambda}_a(t)$ as

$$\boldsymbol{\lambda}_a(t+1) = \big(I + \mu e(t)\mathrm{diag}\big(\boldsymbol{u}(t)\big)\big)\boldsymbol{\lambda}_a(t) \quad (13)$$

where $\boldsymbol{u}(t)$ is defined as $\boldsymbol{u}(t) \triangleq [\boldsymbol{\delta}(t); -\boldsymbol{\delta}(t)]$.

For any desired signal $y(t)$, we can write $y(t) - \hat{y}_m(t) = \boldsymbol{\lambda}_0^T(t)\boldsymbol{\delta}(t) + e_0(t)$, where $\boldsymbol{\lambda}_0(t)$ is the optimum solution such that $\boldsymbol{\lambda}_0(t) = \boldsymbol{R}^{-1}(t)\boldsymbol{p}(t)$, $\boldsymbol{R}(t)$ and $\boldsymbol{p}(t)$ are defined as $\boldsymbol{R}(t) \triangleq E\big[\boldsymbol{\delta}(t)\boldsymbol{\delta}^T(t)\big]$ and $\boldsymbol{p}(t) = E\big\{\boldsymbol{\delta}(t)\big[y(t) - \hat{y}_m(t)\big]\big\}$ and $e_0(t)$ is zero-mean disturbance and uncorrelated to $\boldsymbol{\delta}(t)$. We next show that the mixture weights converge to the optimum solution in the steady-state such that $\lim_{t\to\infty} E\big[\boldsymbol{\lambda}(t)\big] = \lim_{t\to\infty} \boldsymbol{\lambda}_0(t)$.

Subtracting (12) from (11), we obtain

$$\boldsymbol{\lambda}(t+1) = \boldsymbol{\lambda}(t) - \mu e(t)\mathrm{diag}\big(\boldsymbol{\delta}(t)\big)\boldsymbol{\lambda}(t) + 2\mu e(t)\mathrm{diag}\big(\boldsymbol{\delta}(t)\big)\boldsymbol{\lambda}_1(t). \quad (14)$$

Defining $\boldsymbol{\varepsilon}(t) \triangleq \boldsymbol{\lambda}_0(t) - \boldsymbol{\lambda}(t)$, using $e(t) = \boldsymbol{\delta}^T(t)\boldsymbol{\varepsilon}(t) + e_0(t)$ and subtracting both sides from $\boldsymbol{\lambda}_0(t+1)$ in (14) yields

$$\begin{aligned}\boldsymbol{\varepsilon}(t+1) &= \boldsymbol{\varepsilon}(t) + \mu\mathrm{diag}\big(\boldsymbol{\delta}(t)\big)\boldsymbol{\lambda}(t)\boldsymbol{\delta}^T(t)\boldsymbol{\varepsilon}(t) \\ &+ \mu\mathrm{diag}\big(\boldsymbol{\delta}(t)\big)\boldsymbol{\lambda}(t)e_0(t) - 2\mu\mathrm{diag}\big(\boldsymbol{\delta}(t)\big)\boldsymbol{\lambda}_1(t)\boldsymbol{\delta}^T(t)\boldsymbol{\varepsilon}(t) \\ &- 2\mu\mathrm{diag}\big(\boldsymbol{\delta}(t)\big)\boldsymbol{\lambda}_1(t)e_0(t) + \big[\boldsymbol{\lambda}_0(t+1) - \boldsymbol{\lambda}_0(t)\big]. \quad (15)\end{aligned}$$

Taking expectations of both sides, we get $E\big[\mu\mathrm{diag}\big(\boldsymbol{\delta}(t)\big)\boldsymbol{\lambda}(t)e_0(t)\big] = 0$, $E\big[\mu\mathrm{diag}\big(\boldsymbol{\delta}(t)\big)\boldsymbol{\lambda}_1(t)e_0(t)\big] = 0$. Assuming that $\boldsymbol{\lambda}_1(t)$ and $\boldsymbol{\lambda}_2(t)$ are independent of $\boldsymbol{\varepsilon}(t)$ yields

$$\begin{aligned}E\big[\boldsymbol{\varepsilon}(t+1)\big] &= E\big[I - \mu\mathrm{diag}\big(\boldsymbol{\lambda}_1(t) + \boldsymbol{\lambda}_2(t)\big)\boldsymbol{\delta}(t)\boldsymbol{\delta}^T(t)\big]E\big[\boldsymbol{\varepsilon}(t)\big] \\ &+ E\big[\boldsymbol{\lambda}_0(t+1) - \boldsymbol{\lambda}_0(t)\big]. \quad (16)\end{aligned}$$

Since for a wide range of adaptive methods used in the first stage $\boldsymbol{R}(t)$ and $\boldsymbol{p}(t)$ are convergent [4],[3], we obtain $\lim_{t\to\infty} E\big[\boldsymbol{\lambda}_0(t+1) - \boldsymbol{\lambda}_0(t)\big] = 0$. If μ is chosen such that the eigenvalues of $E\big[I - \mu\mathrm{diag}\big(\boldsymbol{\lambda}_1(t) + \boldsymbol{\lambda}_2(t)\big)\boldsymbol{\delta}(t)\boldsymbol{\delta}^T(t)\big]$ have strictly less than unit magnitude for every t, then $\lim_{t\to\infty} E\big[\boldsymbol{\lambda}(t)\big] = \lim_{t\to\infty} \boldsymbol{\lambda}_0(t)$.

For the transient analysis of mean-square error, we have

$$\begin{aligned}E[e^2(t)] &= E\Big\{\big[y(t) - \hat{y}_m(t)\big]^2\Big\} - 2\bar{\boldsymbol{\lambda}}_a^T(t)\boldsymbol{\gamma}(t) \\ &+ \mathrm{tr}\Big(E\Big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\Big]\boldsymbol{\Gamma}(t)\Big), \quad (17)\end{aligned}$$

where we define $\boldsymbol{\gamma}(t) \triangleq E\big\{\boldsymbol{u}(t)\big[y(t) - \hat{y}_m(t)\big]\big\}$ and $\boldsymbol{\Gamma}(t) \triangleq E\big[\boldsymbol{u}(t)\boldsymbol{u}^T(t)\big]$.

For the recursion of $\bar{\boldsymbol{\lambda}}_a(t) = E[\boldsymbol{\lambda}_a(t)]$, using (13), we get

$$\begin{aligned}\bar{\boldsymbol{\lambda}}_a(t+1) &= \bar{\boldsymbol{\lambda}}_a(t) + \mu\mathrm{diag}\big(\boldsymbol{\gamma}(t)\big)\bar{\boldsymbol{\lambda}}_a(t) \\ &- \mu\mathrm{diag}\big(E[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)]\boldsymbol{\Gamma}(t)\big). \quad (18)\end{aligned}$$

Again using (13), assuming $\boldsymbol{\lambda}_a(t)$ is Gaussian and assuming $\lambda_a^{(i)}(t)$ and $\lambda_a^{(j)}(t)$ are independent when $i \neq j$, we have a recursion for $E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big]$ as

$$\begin{aligned}&E\big[\boldsymbol{\lambda}_a(t+1)\boldsymbol{\lambda}_a^T(t+1)\big] = \\ &\Big(I + \mu\mathrm{diag}\big(\boldsymbol{\gamma}(t)\big) - \mu\mathrm{diag}\big(\boldsymbol{\Gamma}(t)\bar{\boldsymbol{\lambda}}_a(t)\big)\Big)E[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)] \\ &- \mu E\big[\mathrm{diag}^2\big(\boldsymbol{u}(t)\big)\big]\Big(E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Big)\mathbf{1}\bar{\boldsymbol{\lambda}}_a^T(t) \\ &- \mu\mathrm{diag}\big(\bar{\boldsymbol{\lambda}}_a(t)\big)\boldsymbol{\Gamma}(t)\Big(E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Big) \\ &E[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)]\Big(\mu\mathrm{diag}\big(\boldsymbol{\gamma}(t)\big) - \mu\mathrm{diag}\big(\boldsymbol{\Gamma}(t)\bar{\boldsymbol{\lambda}}_a(t)\big)\Big) \\ &- \mu\bar{\boldsymbol{\lambda}}_a(t)\mathbf{1}^T\Big(E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Big)E\big[\mathrm{diag}^2\big(\boldsymbol{u}(t)\big)\big] \\ &- \mu\Big(E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Big)\boldsymbol{\Gamma}(t)\mathrm{diag}\big(\bar{\boldsymbol{\lambda}}_a(t)\big). \quad (19)\end{aligned}$$

Hence, we have the mean and the variance recursions in (18) and (19) respectively. We use (18) and (19) in (17) and obtain the time evolution of MSE. This completes the transient analysis for the unconstrained relative entropy.

Relative Entropy: For the affinely constrained combination updated with the EG algorithm, we have multiplicative updates (7) and (8). Using the same approximations as in (9), (10), (11) and (12), we can obtain update on $\boldsymbol{\lambda}_a(t)$ as

$$\boldsymbol{\lambda}_a(t+1) = u\frac{\big[I + \mu e(t)\mathrm{diag}\big(\boldsymbol{u}(t)\big)\big]\boldsymbol{\lambda}_a(t)}{\big[\mathbf{1}^T + \mu e(t)\boldsymbol{u}^T(t)\big]\boldsymbol{\lambda}_a(t)}. \quad (20)$$

For the recursion of $\bar{\boldsymbol{\lambda}}_a(t)$, using (20), we get

$$\begin{aligned}E\big[\boldsymbol{\lambda}_a(t+1)\big] &= E\left\{u\frac{\big[I + \mu e(t)\mathrm{diag}\big(\boldsymbol{u}(t)\big)\big]\boldsymbol{\lambda}_a(t)}{\big[\mathbf{1}^T + \mu e(t)\boldsymbol{u}^T(t)\big]\boldsymbol{\lambda}_a(t)}\right\}, \\ &\approx u\frac{\big\{I + \mu\mathrm{diag}\big(\boldsymbol{\gamma}(t)\big)\big\}E\big[\boldsymbol{\lambda}_a(t)\big] - \mu\mathrm{diag}\big(E[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)]\boldsymbol{\Gamma}(t)\big)}{\big[\mathbf{1}^T + \mu\boldsymbol{\gamma}^T(t)\big]E\big[\boldsymbol{\lambda}_a(t)\big] - \mu\mathrm{tr}\big(E[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)]\boldsymbol{\Gamma}(t)\big)}. \quad (21)\end{aligned}$$

From (20), using the same approximation in (21), assuming $\boldsymbol{\lambda}_a(t)$ is Gaussian, assuming $\lambda_a^{(i)}(t)$ and $\lambda_a^{(j)}(t)$ are independent when $i \neq j$, we have a recursion for $E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big]$ as

$$E\big[\boldsymbol{\lambda}_a(t+1)\boldsymbol{\lambda}_a^T(t+1)\big] = u^2\frac{\boldsymbol{A}(t)}{b(t)}, \quad (22)$$

where the numerator $\boldsymbol{A}(t)$ is the right hand side of (19) and the denominator

$$
\begin{aligned}
b(t) &= \mathbf{1}^T E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big]\mathbf{1} + \mu\boldsymbol{\gamma}^T(t)E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big]\mathbf{1} \\
&- \mu\bar{\boldsymbol{\lambda}}_a^T(t)\boldsymbol{\Gamma}(t)E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big]\mathbf{1} \\
&- \mu\mathbf{1}^T\Big(E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Big)\boldsymbol{\Gamma}(t)\bar{\boldsymbol{\lambda}}_a(t) \\
&- \mu\mathbf{1}^T\Big(E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Big)E\big[\mathrm{diag}^2(\boldsymbol{u}(t))\big]\mathbf{1}^T\bar{\boldsymbol{\lambda}}_a(t)\mathbf{1} \\
&+ \mu\mathbf{1}^T E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big]\boldsymbol{\gamma}(t) - \mu\mathbf{1}^T E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big]\boldsymbol{\Gamma}(t)\bar{\boldsymbol{\lambda}}_a(t) \\
&- \mu\bar{\boldsymbol{\lambda}}_a^T(t)\boldsymbol{\Gamma}(t)\Big(E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Big)\mathbf{1} \\
&- \mu\mathbf{1}^T\bar{\boldsymbol{\lambda}}_a^T(t)\mathbf{1}E\big[\mathrm{diag}^2(\boldsymbol{u}(t))\big]\Big(E\big[\boldsymbol{\lambda}_a(t)\boldsymbol{\lambda}_a^T(t)\big] - \bar{\boldsymbol{\lambda}}_a(t)\bar{\boldsymbol{\lambda}}_a^T(t)\Big)\mathbf{1}.
\end{aligned}
\tag{23}
$$

We next use the mean (21) and the variance (22), (23) recursions in (17) and obtain the time evolution of MSE.

V. SIMULATIONS

In this section, we illustrate the accuracy of our results and compare performances of different mixture methods through simulations.

To show accurateness of our results in (21), (22) and (23), the desired signal as well as the system parameters are selected as follows. First, a seventh-order linear filter, $\boldsymbol{w}_o = [0.25, -0.47, -0.37, 0.045, -0.18, 0.78, 0.147]^T$, is chosen [6]. The underlying signal is generated using the data model $y(t) = \tau\, \boldsymbol{w}_o^T\boldsymbol{a}(t) + n(t)$, where $\boldsymbol{a}(t)$ is an i.i.d. Gaussian vector process with zero mean and unit variance entries, i.e., $E[\boldsymbol{a}(t)\boldsymbol{a}^T(t)] = \boldsymbol{I}$, $n(t)$ is an i.i.d. Gaussian noise process with zero mean and variance $E[n^2(t)] = 0.3$, and τ is a positive scalar to control SNR. For the first experiment, we choose SNR = -10dB. We select the constituent filters such that we have 10 linear filters all using the LMS update to train their weight vectors and the learning rates for 2 constituent filters are set to $\mu_i = 0.002$, $i = 1, 6$ while the learning rates for the LMS updates for the rest of constituent filters are selected randomly $[0.1, 0.11]$. Therefore, in the steady-state, we obtain the optimum combination vector as approximately $\boldsymbol{\lambda}_o = [0.5, 0, 0, 0, 0, 0.5, 0, 0, 0, 0]^T$, i.e., the final combination vector is sparse. In the second stage, we train the combination weights with the EG and LMS algorithms and compare performances of these algorithms. We use the EG algorithm instead of the EGU algorithm because we observe that the EG algorithm performs better than the EGU algorithm in our simulations. For the second stage, the learning rates for the EG and LMS algorithms are selected as $\mu_{\mathrm{EG}} = 0.001$ and $\mu_{\mathrm{LMS}} = 0.005$ such that the MSEs of the final output produced with both algorithms converge to the same MSE, hence we can fairly compare the performances of these algorithms. In Fig. 2, we plot the MSE curves for the adaptive mixture updated with the EG algorithm, the adaptive mixture updated with the LMS algorithm, first constituent filter with $\mu_1 = 0.002$ and second constituent filter with $\mu_2 = 0.1$, where the MSE of the final output using the EG algorithm on the mixture weights converges faster than the MSE of the final output using the LMS algorithm on the mixture weights.

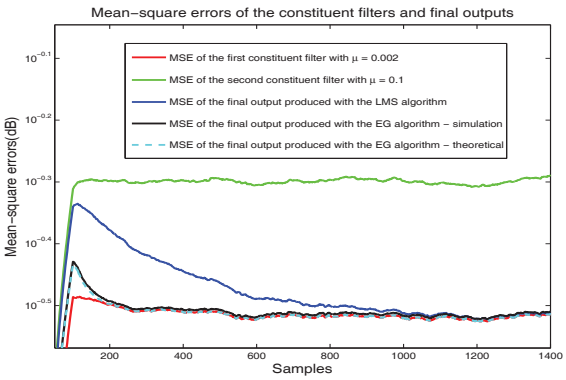

Fig. 2. MSE curves for the adaptive mixture updated with the EG algorithm, the adaptive mixture updated with the LMS algorithm, first constituent filter with $\mu_1 = 0.002$ and second constituent filter with $\mu_2 = 0.1$ are plotted.

VI. CONCLUSION

In this paper, we investigated adaptive affine mixture methods based on Bregman divergences and provided the transient analysis of these methods using the EGU and EG algorithms. In our simulations, we compared performances of the EG and LMS algorithms and observed that the EG algorithm performs better than the LMS algorithm when the combination vector in steady-state is sparse. We observed that the MSE of the final output using the EG algorithm on the mixture weights converges faster than the MSE of the final output using the LMS algorithm on the mixture weights. We also observed a close agreement between our simulations and theoretical results.

VII. REFERENCES

[1] J. Kivinen and M.K. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors," in *Inform. Comput.*, Jan. 1997, vol. 132, pp. 1–64.

[2] N. J. Bershad, J. C. M. Bermudez, and J. Tourneret, "An affine combination of two LMS adaptive filters: Transient mean-square analysis," *IEEE Tran. on Sig. Proc.*, vol. 56, no. 5, pp. 1853–1864, 2008.

[3] A. H. Sayed, *Fundamentals of Adaptive Filtering*, John Wiley and Sons, 2003.

[4] S. S. Kozat, A. T. Erdogan, A. C. Singer, and A. H. Sayed, "Steady state MSE performance analysis of mixture approaches to adaptive filtering," *IEEE Tran. on Sig. Proc.*, vol. 58, pp. 4421–4427, Aug. 2010.

[5] J. Arenas-Garcia, A. R. Figueiras-Vidal, and A. H. Sayed, "Mean-square performance of a convex combination of two adaptive filters," *IEEE Tran. on Sig. Proc.*, vol. 54, pp. 1078–1090, 2006.

[6] Suleyman S. Kozat, Alper T. Erdogan, Andrew C. Singer, and Ali H. Sayed, "Transient analysis of adaptive affine combinations," *IEEE Tran. on Sig. Proc.*, accepted, 2011.

[7] Jacob Benesty and Yiteng (Arden) Huang, "The LMS, PNLMS, and Exponentiated Gradient algorithms," *Proc. Eur. Signal Process. Conf. (EUSIPCO)*, pp. 721–724, 2004.