Deep Unfolding Aided Parameter Optimization for Multi-task Diffusion LMS Algorithm

Xiaoqing Tong* and Kazunori Hayashi†
* Kyoto University, Kyoto
E-mail: tong.xiaoqing.75d@st.kyoto-u.ac.jp
† Kyoto University, Kyoto
E-mail: hayashi.kazunori.4w@kyoto-u.ac.jp

***Abstract*—The paper proposes a novel algorithm termed TMDLMS (Trainable Multi-task Diffusion Least Mean Squares) to collaboratively estimate multiple vectors, which are assumed to have certain correlations, from observations obtained at multiple nodes in the network. Adopting the concept of deep unfolding, the proposed algorithm is obtained by unfolding iterative process of the conventional MDLMS (Multi-task Diffusion Least Mean Squares) algorithm, resulting in a multi-layered signal flow graph reminiscent of neural networks. In this structure, the parameters in the algorithm such as the step size of each layer are considered as trainable parameters, allowing for optimization via machine learning techniques, such as stochastic gradient descent (SGD). Numerical experimental results indicate that, compared to the conventional MDLMS using fixed parameters, the proposed algorithm can boast the convergence rates and achieve lower steady-state errors at the same time. The results demonstrate the validity of the proposed approach under various conditions.**

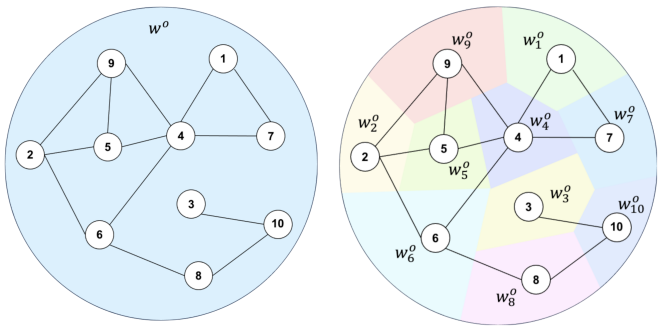
I. Introduction

In the fields of adaptive signal processing, Least Mean Squares (LMS) algorithm [1] would be the most prominent adaptive algorithm because of its simplicity and robustness. While LMS has been widely used in various applications, it utilizes its own measurement only in the update equation to obtain the estimate of the unknown vector. This leads to the development of diffusion LMS (DLMS) [2] for collaborative estimation of a common unknown vector in distributed environment. Moreover, DLMS has been extended to multi-task diffusion LMS (MDLMS) [3], which can estimate different unknown vectors but with certain correlations in collaborative manner to cope with more intricate networks. However, LMS based algorithms are known to suffer from slow rate of convergence in general.

Model-driven machine learning (ML) [4] has become a popular tool in communication systems in recent years. The concept of model-driven ML is to merge principled algorithms that have performance guarantees with tools derived from ML techniques, aiming to combine the strengths of both approaches. Deep unfolding [5] is one of powerful model-driven ML methods that fuses iterative optimization algorithms with tools from neural networks to efficiently solve various tasks in adaptive signal processing. As stated in [6], deep unfolding takes an existing iterative algorithm with a predefined number of iterations, unfolds it in the direction of time axis with the multi-layer structure, and introduces a set of trainable parameters. These parameters are then learned using deep learning techniques, utilizing suitable loss functions, stochastic gradient descent (SGD), and backpropagation. With learned parameters, the unfolded algorithm can achieve better convergence performance than that of original iterative algorithm in range of tasks. Currently, deep unfolding has been applied to various applications. For instance, hyperparameters of the iterative shrinkage thresholding algorithm (ISTA) [7] has been determined by using deep unfolding leading to the trainable iterative shrinkage thresholding algorithm (TISTA) [8]. It has been demonstrated that TISTA achieves a significantly faster convergence rate compared to the conventional ISTA.

Previous research has integrated deep unfolding with LMS based algorithms as well. Sasaki et al. combined deep unfolding with both LMS and normalized LMS (NLMS) algorithms, named TLMS and TNLMS, respectively [9]. Grounded in the concept of deep unfolding, they treated the step size as a learnable parameter within the iterative process. Numerical results have indicated that this approach could enhance convergence rates and reduce steady-state error with the TNLMS algorithm showing particularly notable performance. Similarly, NISHIHATA et al. [10] applied deep unfolding to the DLMS algorithm, which has demonstrated an improved convergence rate compared to the original DLMS by optimizing the step size and edge weights.

Inspired by the prior works, this paper applies the deep unfolding technique to the MDLMS algorithm, and proposes a new algorithm termed "Trainable Multi-task Diffusion Least Mean Squares" (TMDLMS). The algorithm is to collaboratively estimate multiple unknown vectors, which are presumed to have certain correlations, from observation vectors collected by multiple nodes in a network. We have unfolded the iterative process of the MDLMS algorithm, resulting in a multi-layered signal flow graph reminiscent of neural networks. Within this structure, parameters in the algorithm, such as the step size, are considered trainable. For optimizing these parameters, we have employed optimization algorithms such as Adam [11]. To assess the practical performance of the TMDLMS algorithm, numerical experiments have been conducted under various correlation conditions. The results have indicated that, compared to MDLMS, the proposed algorithm boasts convergence rates and achieves lower steady-state errors at the same time.



(a) (b)

Fig. 1. Examples of two types of learning problems. (a) Single-task learning problem. (b) Multi-task learning problem. [3]

II. Review of LMS, DLMS and MDLMS algorithms

Here, we briefly review LMS, DLMS and MDLMS algorithms implemented for the estimation of an unknown vector (or unknown vectors) in the network as the basis of the discussion in this paper.

A. Problem Setting

We consider a sensor network represented by an undirected graph with N nodes. The presence of an edge between any two nodes signifies the existence of a communication link between them.

Assuming that the unknown vector to be estimated by node i is $\boldsymbol{w}_i^{\mathrm{o}} \in R^{L\times 1}$, the linear measurement of the unknown vector $\boldsymbol{w}_i^{\mathrm{o}}$ observed at node i at time $k \geq 0$ is given by

$$d_i^{(k)} = \boldsymbol{u}_i^{(k)\mathsf{T}} \boldsymbol{w}_i^{\mathrm{o}} + v_i^{(k)}, \tag{1}$$

where $\boldsymbol{u}_i^{(k)} \in R^{L\times 1}$ is a vector used for the linear measurement by node i, and $v_i^{(k)}$ is an additive white Gaussian noise (AWGN) with mean of 0 and variance of σ_v^2.

Depending on the relation among $\boldsymbol{w}_i^{\mathrm{o}}$, the estimation problem of $\boldsymbol{w}_i^{\mathrm{o}}$ can be classified into a single-task learning problem and a multi-task learning problem as shown in Fig. 1:

- **Single-task learning problem:** All nodes have a common estimation target vector $\boldsymbol{w}^{\mathrm{o}}$. That is, in this case we have

$$\boldsymbol{w}_i^{\mathrm{o}} = \boldsymbol{w}^{\mathrm{o}}, \quad \forall i \in \{1, \ldots, N\}$$

- **Multi-task learning problem:** Each node i tries to estimate its own target vector $\boldsymbol{w}_i^{\mathrm{o}}$, which is different from the target vectors of other nodes. However, a certain correlation among $\boldsymbol{w}_i^{\mathrm{o}}$ is often assumed, which enables us to achieve collaborative estimation.

B. LMS Algorithm

Although the LMS algorithm is not designed for the estimation in the network environment, it can be applied for the problem by implementing the LMS algorithm at each node and simply ignoring the existence of communication links among the nodes.

In the LMS algorithm, the estimated vector $\boldsymbol{\phi}_i^{(k)}$ at time k and node i is updated according to the following equation:

$$\boldsymbol{\phi}_i^{(k+1)} = \boldsymbol{\phi}_i^{(k)} + \mu \boldsymbol{u}_i^{(k)} \left(d_i^{(k)} - \boldsymbol{u}_i^{(k)\mathsf{T}} \boldsymbol{\phi}_i^{(k)} \right), \tag{2}$$

where the initial estimate for the vector $\boldsymbol{\phi}_i^{(0)}$ is often set to $\mathbf{0}$.

C. DLMS Algorithm

The DLMS algorithm [2] can achieve a collaborative estimation for the single-task learning problem. The DLMS algorithm consists of the **LMS step** responsible for signal estimation and the **averaging step** that facilitates information exchange between nodes.

In the LMS step, the estimated vector $\boldsymbol{\psi}_i^{(k)}$ at time k and node i is updated by using its own measurement as

$$\boldsymbol{\psi}_i^{(k+1)} = \boldsymbol{\phi}_i^{(k)} + \mu \boldsymbol{u}_i^{(k)} \left(d_i^{(k)} - \boldsymbol{u}_i^{(k)\mathsf{T}} \boldsymbol{\phi}_i^{(k)} \right), \tag{3}$$

which is basically the same as the update equation of the LMS algorithm.

Each node exchanges $\boldsymbol{\psi}_i^{(k)}$ obtained in the LMS step with its neighbor nodes $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is a set of neighbors of node i including i itself. Then, the updated estimate of the vector in the averaging step is obtained at each node i by the weighted average of $\boldsymbol{\psi}_i^{(k)}$ based on the idea of the averaging consensus technique. Specifically, the update equation in the averaging step is given by

$$\boldsymbol{\phi}_i^{(k)} = \sum_{j\in\mathcal{N}_i} c_{ij} \boldsymbol{\psi}_j^{(k)}, \tag{4}$$

where c_{ij} is the weight given for $\boldsymbol{\psi}_j^{(k)}$ obtained from node j in the averaging update at node i. The choice of the weight c_{ij} has an impact on the convergence performance of DLMS. In this paper, we employ an established approach named Metropolis rule [12] given by

$$c_{ij} = \begin{cases} \frac{1}{\max(|\mathcal{N}_i|, |\mathcal{N}_j|)} & \text{if } j \in \mathcal{N}_i \text{ and } i \neq j \\ 1 - \sum_{j\in\mathcal{N}_i^-} c_{ij} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where $|\mathcal{N}_i|$ denotes the cardinality of the set $\mathcal{N}_i$, and $\mathcal{N}_i^-$ denotes the set $\mathcal{N}_i$ but without node i.

D. MDLMS Algorithm

The MDLMS algorithm has been proposed by Chen et al. in [3] for the multi-task learning problem. To be more precise, they have considered more general problem of the clustered multi-task learning problem than the multi-task learning problem, where nodes in the same cluster share a common target vector. Thus, the multi-task learning problem considered in this paper can be regarded as a special case of the clustered multi-task learning problem, where the number of nodes in each cluster is fixed to one.

The update equation of the MDLMS algorithm for the multi-task learning problem is given by

$$\boldsymbol{\phi}_i^{(k+1)} = \boldsymbol{\phi}_i^{(k)} + \mu \boldsymbol{u}_i^{(k)} \left[d_i^{(k)} - \boldsymbol{u}_i^{(k)\mathsf{T}} \boldsymbol{\phi}_j^{(k)} \right] + \eta\mu \sum_{j \in \mathcal{N}_i^-} \rho_{ij} \left(\boldsymbol{\phi}_j^{(k)} - \boldsymbol{\phi}_i^{(k)} \right), \tag{6}$$

where a regularization term to delineate the interrelation of estimated vectors across neighbor nodes is incorporated.The non-negative coefficient ρ_{ij} modulate the regularization intensity, which is achieved by applying the Metropolis rule [12], as specified in Equation (5). Recognizing inter-task relationships is crucial, offering potential enhancements in estimation precision.

III. Proposed TMDLMS Algorithms

Here, we propose TMDLMS algorithm by treating parameters of the conventional MDLMS algorithm in each iteration to be independent learnable parameters, and optimize them by supervised learning using the pair of the measurement vector and the corresponding desired signal as training data with some SGD based algorithm.

We introduce the proposed TMDLMS algorithm by treating the parameters μ and η of the MDLMS algorithm to be the independent learnable parameters. Namely, in the proposed algorithm, μ and η in different iteration become distinct learnable parameters denoted as $\mu^{(k)}$ and $\eta^{(k)}$. Thus, the update equation of the TMDLMS algorithm is given by

$$\boldsymbol{\phi}_i^{(k+1)} = \boldsymbol{\phi}_i^{(k)} + \mu^{(k)} \boldsymbol{u}_i^{(k)} \left[d_i^{(k)} - \boldsymbol{u}_i^{(k)\mathsf{T}} \boldsymbol{\phi}_j^{(k)} \right] + \eta^{(k)}\mu^{(k)} \sum_{j \in \mathcal{N}_i^-} \rho_{ij} \left(\boldsymbol{\phi}_j^{(k)} - \boldsymbol{\phi}_i^{(k)} \right). \tag{7}$$

The time variant parameters $\mu^{(k)}$ and $\eta^{(k)}$ are optimized by using SGD, where the loss function is defined as

$$J_i^{(k)} = \frac{1}{N} \sum_{i=0}^{N-1} \left\| d_i^{(k)} - y_i^{(k)} \right\|_2^2, \tag{8}$$

where

$$y_i^{(k)} = \boldsymbol{u}_i^{(k)\mathsf{T}} \boldsymbol{\phi}_i^{(k-1)}. \tag{9}$$

Then, we describe the gradient descent-based approach in the TMDLMS algorithm, assuming the size of the batch to be $B = 1$ for simplicity. The partial differentiation of the loss function $J_i^{(t)}$ at the t-th round of the incremental training with respect to the step size $\mu_i^{(k-\tau)}$ and the regularization coefficient $\eta_i^{(k-\tau)}$ are given by

$$\begin{aligned} \frac{\partial J_i^{(t)}}{\partial \mu_i^{(t-\tau)}} &= \frac{\partial J_i^{(t)}}{\partial y_i^{(t)}} \frac{\partial y_i^{(t)}}{\partial y_i^{(t-1)}} \frac{\partial y_i^{(t-1)}}{\partial y_i^{(t-2)}} \cdots \\ &\quad \cdots \frac{\partial y_i^{(t-\tau+2)}}{\partial y_i^{(t-\tau+1)}} \frac{\partial y_i^{(t-\tau+1)}}{\partial \mu_i^{(t-\tau)}} \\ &= (-2)^\tau \cdot e_i^{(t)} {e_i^{(t-\tau)}}^2 \boldsymbol{u}_i^{(t-\tau+1)\mathsf{T}} \boldsymbol{u}_i^{(t-\tau)} \\ &\quad \cdot \prod_{k=t-\tau+1}^{t-1} \left[\mu_i^{(k)} \boldsymbol{u}_i^{(k+1)\mathsf{T}} \boldsymbol{u}_i^{(k)\mathsf{T}} \right], \end{aligned} \tag{10}$$

$$\begin{aligned} \frac{\partial J_i^{(t)}}{\partial \eta_i^{(t-\tau)}} &= \frac{\partial J_i^{(t)}}{\partial y_i^{(t)}} \frac{\partial y_i^{(t)}}{\partial y_i^{(t-1)}} \frac{\partial y_i^{(t-1)}}{\partial y_i^{(t-2)}} \cdots \\ &\quad \cdots \frac{\partial y_i^{(t-\tau+2)}}{\partial y_i^{(t-\tau+1)}} \frac{\partial y_i^{(t-\tau+1)}}{\partial \eta_i^{(t-\tau)}} \\ &= (-2)^\tau \cdot e_i^{(t)} \left(y_j^{(t-1)} - y_i^{(t-1)} \right) \\ &\quad \cdot \prod_{k=t-\tau+1}^{t-1} \left[\mu_i^{(k)} \boldsymbol{u}_i^{(k+1)\mathsf{T}} \boldsymbol{u}_i^{(k)\mathsf{T}} \right]. \end{aligned} \tag{11}$$

IV. Numerical Results

We have conducted numerical experiments of the system identification problem as an example of the multi-task learning problem in order to evaluate the performance of the proposed TMDLMS algorithm in comparison with that of conventional algorithms such as LMS, DLMS and MDLMS.

A. Experimental Setup

In the numerical evaluations, we consider to identify the system characterized by the target vectors $\boldsymbol{w}_i^{\mathrm{o}} \in R^{L\times 1}$ $(i = 1, \ldots, N)$, which corresponds to node i in the network of $N = 10$ nodes, from the linear observations of

$$d_i^{(k)} = \boldsymbol{u}_i^{(k)\mathsf{T}} \boldsymbol{w}_i^{\mathrm{o}} + v_i^{(k)}, \tag{12}$$

where the measurement noise $v_i^{(k)}$ is assumed to follow $\mathcal{N}(0, 0.1)$, and the length of the target vector is set to $L = 10$. The vector $\boldsymbol{u}_i^{(k)}$ is used to obtain the linear measurement $d_i^{(k)}$ at node i and time k, whose elements are generated from independent Gaussian distribution with mean of 0 and variance of 1.

For the network structure, we have randomly generated the 10-node network, where all nodes are randomly positioned within a square region of side length $1/\sqrt{3}$. Edges are established between nodes if their distance was less than or equal to 0.25. We have further ensured that each node had at least one neighboring node.

The target vectors $\boldsymbol{w}_i^{\mathrm{o}}$ are systematically generated based on the network structure as follows: Initially, we generate the target vector $\boldsymbol{w}_1^{\mathrm{o}}$ of node 1 as a sample from the multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Then, for the other nodes, if

node j is adjacent to node i whose target vector $\boldsymbol{w}_i^{\mathrm{o}}$ has been already generated, we generate the target vector of node j as

$$\boldsymbol{w}_j^{\mathrm{o}} = \frac{1}{\sqrt{1+\sigma^2}}(\boldsymbol{w}_i^{\mathrm{o}} + \delta\boldsymbol{w}_j^*), \tag{13}$$

where $\delta\boldsymbol{w}_j^*$ is a sample from the multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$. Note that the value of σ^2 can control the similarity of the target vectors of the neighboring nodes. Namely, if σ^2 is smaller, the target vectors of different nodes are similar, and if σ^2 is larger, the target vectors are more different.

In the proposed TMDLMS, we have employed Adam [11] as the optimization algorithm and utilized Optuna [13] to determine the hyper-parameters, such ad the learning rate and the initial value. Optuna [13] is employed to optimize the loss function $J_i^{(k)}$, which quantifies the error between the model output and the target values. The optimization process is conducted through multiple trials, where Optuna selects a set of initial values for the hyperparameters, such as $\mu^{(k)}$ and $\eta^{(k)}$, from predefined ranges in each trial. These hyperparameters are then used to update the model, and the loss value for the current trial is computed. By iteratively exploring different hyperparameter combinations and using the loss value feedback from each trial, Optuna gradually narrows the search space.As the trials progress, Optuna dynamically adjusts its hyperparameter search strategy based on historical data, steadily converging toward the optimal solution.

B. Performance Evaluation

Figures 2-4 depict the learning curves of the proposed TMDLMS algorithms, accompanied by the profiles of the step size $\mu^{(k)}$ and $\eta^{(k)}$ obtained by the proposed deep unfolding approach under the condition of $\sigma^2 = 0.01$. In the figures, "TMDLMS ($\mu^{(k)}$)" and "TMDLMS ($\mu^{(k)}$ and $\eta^{(k)}$)" stand for the performance of the proposed TMDLMS with the learnable parameter(s) of $\mu^{(k)}$ only (η is fixed to 0.1) and of $\mu^{(k)}$ and $\eta^{(k)}$, respectively. For a comprehensive evaluation, we have also presented the performance of the LMS, the DLMS, and the MDLMS algorithm with fixed parameters in the figures. As observed in Figure 2, both proposed algorithms achieve better convergence performance compared to the conventional schemes, with the trainable η achieving the lowest steady-state error. Figure 3 reveals that the learned step size $\mu^{(k)}$ tends to decrease as iterations progress, aligning qualitatively with conventional heuristic variable step size approaches.

Figures 5-7 illustrate the learning curves and the profiles of the parameters of the proposed TMDLMS algorithm under the condition of $\sigma^2 = 0.001$, which is smaller than the previous case. As seen in Figures 5, the DLMS achieves better performance than the standalone LMS algorithm. This is because the DLMS algorithm can share information with neighboring nodes, which improves the accuracy of weight estimates. The proposed TMDLMS algorithms can outperform other existing schemes in terms of both the convergence rate and the steady-state error in this case as well.

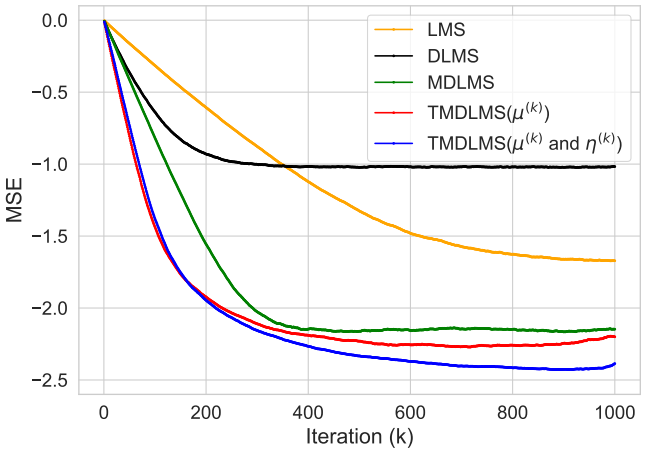


Fig. 2. Learning curve of proposed method ($\sigma^2 = 0.01$)

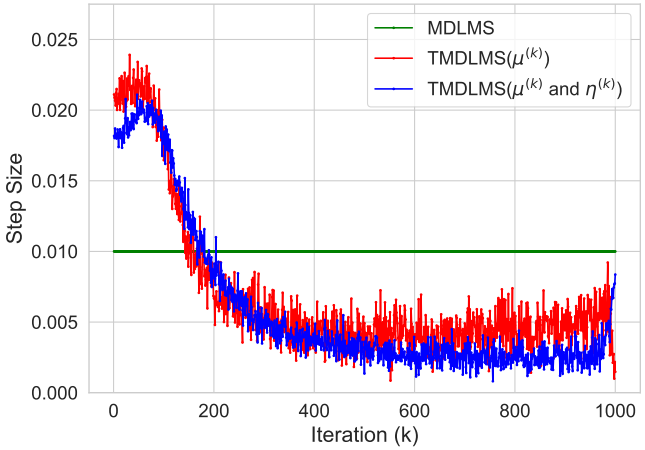


Fig. 3. Profile of step size $\mu^{(k)}$ of proposed TMDLMS ($\sigma^2 = 0.01$)

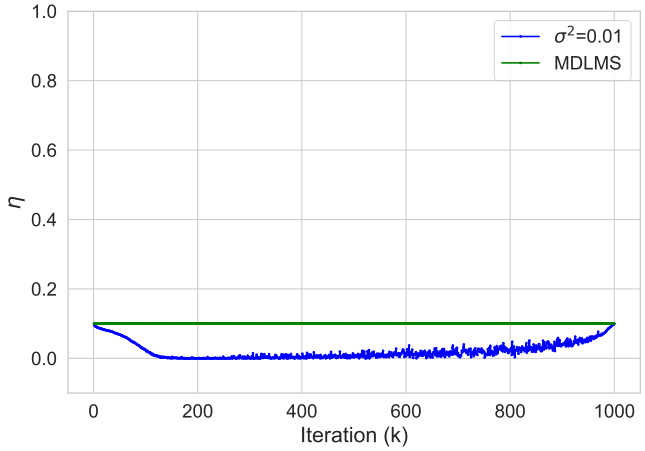


Fig. 4. Profile of η of proposed TMDLMS ($\sigma^2 = 0.01$)

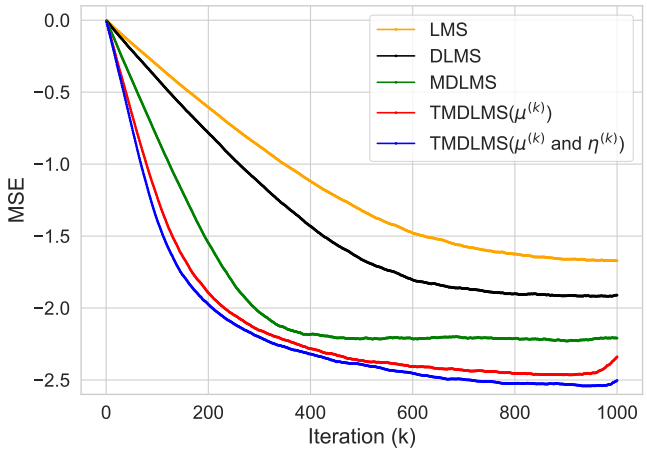


Fig. 5. Learning curve of proposed method ($\sigma^2 = 0.001$)

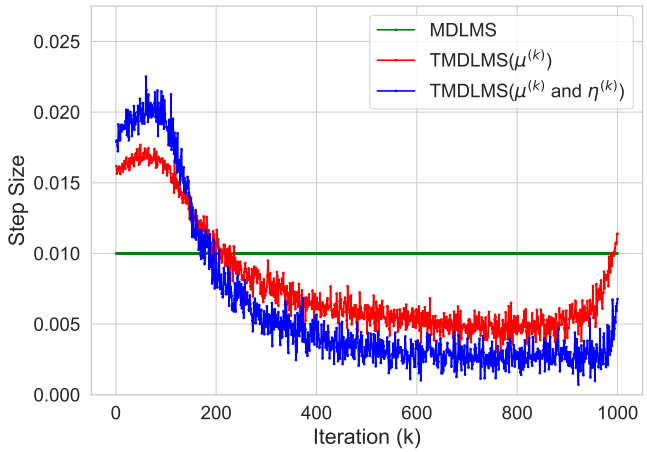


Fig. 6. Profile of step size $\mu^{(k)}$ of proposed TMDLMS ($\sigma^2 = 0.001$)

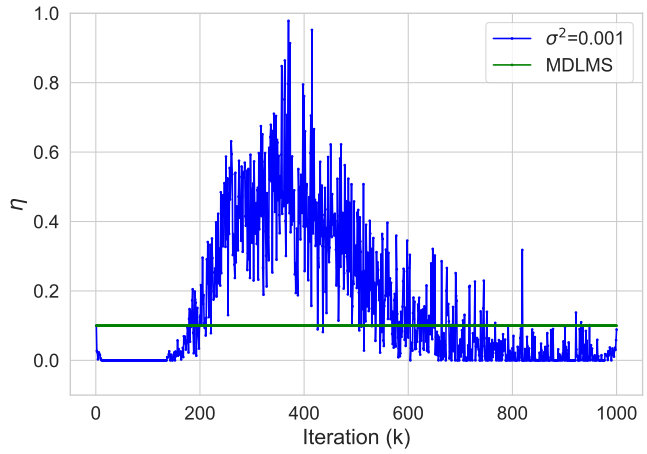


Fig. 7. Profile of η of proposed TMDLMS ($\sigma^2 = 0.001$)

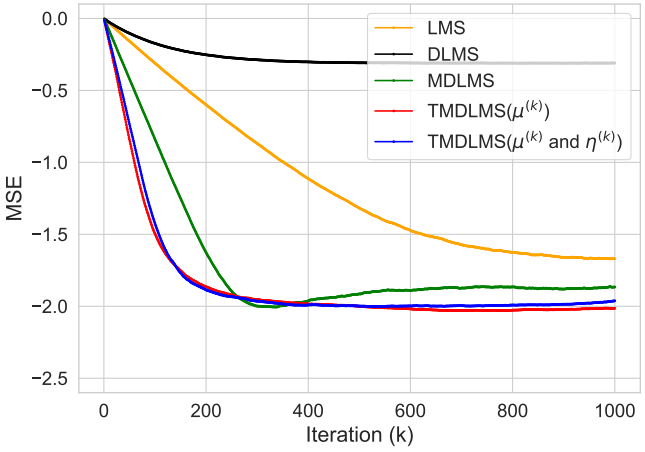


Fig. 8. Learning curve of proposed method ($\sigma^2 = 0.1$)

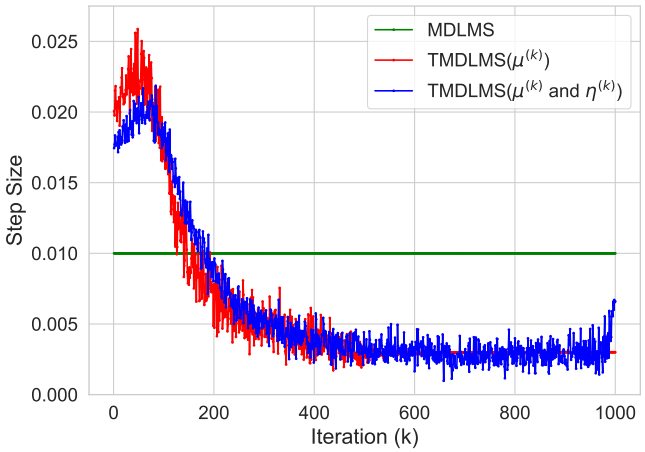


Fig. 9. Profile of step size $\mu^{(k)}$ of proposed TMDLMS ($\sigma^2 = 0.1$)

Figures 8-10 present the learning curves and the profiles of the parameters of the proposed TMDLMS algorithm under the condition of $\sigma^2 = 0.1$.

As illustrated in Figure 8, when σ^2 is high, the performance of the DLMS is degraded significantly, because it assumes that the target vectors of all nodes are in common.

The MDLMS algorithm, on the other hand, can achieve better performance than the standalone LMS algorithm even in such a case. While they get information from other nodes, they can cope with the difference of target vector of each node. The proposed TMDLMS algorithms demonstrate a faster convergence and a lower steady-state error again compared to the existing algorithm. However, in this case, it's not apparent which of the two scenarios of the proposed TMDLMS, trainable η or non-trainable η, performs better. It might be said that, under the condition of $\sigma^2 = 0.1$, the trainable η has small impact on the convergence performance.

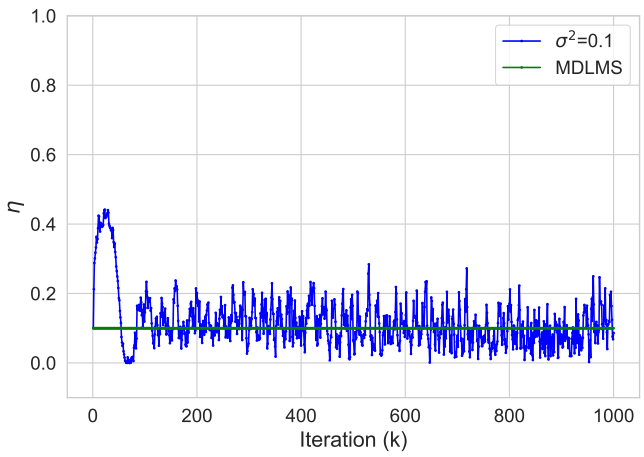


Fig. 10. Profile of η of proposed TMDLMS ($\sigma^2 = 0.1$)

V. Conclusion

We have proposed the TMDLMS algorithms by introducing learnable variable parameters into the MDLMS algorithm and optimizing those parameters with the machine learning approach using the idea of deep unfolding. Through numerical experiments of the system identification problem, we have verified the validity of the proposed approach, demonstrating that they can achieve a faster convergence rate and a reduced steady-state error concurrently. Additionally, the trainable parameter η plays a role in influencing the convergence speed and steady-state error. However, the exact nature of this influence warrants further in-depth experimentation.

Acknowledgment

This work has been supported in part by JSPS KAKENHI Grant Number JP22H00514.

References

[1] B. Widrow, M. E. Hoff, *et al.*, "Adaptive switching circuits," in *IRE WESCON convention record*, New York, vol. 4, 1960, pp. 96–104.

[2] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, 2008.

[3] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4129–4144, 2014.

[4] H. He, S. Jin, C.-K. Wen, F. Gao, G. Y. Li, and Z. Xu, "Model-driven deep learning for physical layer communications," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 77–83, 2019.

[5] J. R. Hershey, J. L. Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," *arXiv preprint arXiv:1409.2574*, 2014.

[6] A. Balatsoukas-Stimming and C. Studer, "Deep unfolding for communications systems: A survey and some new directions," in *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, IEEE, 2019, pp. 266–271.

[7] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[8] D. Ito, S. Takabe, and T. Wadayama, "Trainable ista for sparse signal recovery," *IEEE Transactions on Signal Processing*, vol. 67, no. 12, pp. 3113–3125, 2019.

[9] K. Hayashi, K. Shiohara, and T. Sasaki, "Differentiable programming based step size optimization for lms and nlms algorithms," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2019, pp. 1721–1727. DOI: 10.1109/APSIPAASC47483.2019.9023175.

[10] Y. Nishihata and K. Ishii, "Deep-unfolding aided optimization of edge weights and step sizes for diffusion lms algorithm," *IEICE Technical Report; IEICE Tech. Rep.*, vol. 121, no. 101, pp. 1–6, 2021 (in Japanese).

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[12] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[13] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.