Deep-learning based linear average consensus for faster convergence over temporal network

Masako Kishida†, Masaki Ogura†, and Tadashi Wadayama

Abstract—In this paper, we study the problem of accelerating the linear average consensus algorithm over complex networks. We specifically present a data-driven methodology for tuning the weights of temporal (i.e., time-varying) networks by using deep learning techniques. We first unfold the linear average consensus protocol to obtain a feedforward signal flow graph, which we regard as a neural network. We then train the neural network by using standard deep learning technique to minimize the consensus error over a given finite time-horizon. As a result of the training, we obtain a set of optimized time-varying weights for faster consensus in the network. Numerical simulations are presented to show that our methodology can achieve a significantly smaller consensus error than the static optimal strategy.

I. INTRODUCTION

The distributed agreement problem on networks, often referred to as a consensus problem [1], is an important problem in the network science and engineering, with applications in multi-agent coordination [2], distributed computing [3], distributed sensor networks [4], wireless communication systems [5], and power systems [6]. In the average consensus problem, nodes in the network seek to converge their state variables to the average of their initial states in a distributed manner. The standard solution to the average consensus problem is to use the linear average consensus algorithm [7], in which each node updates its state taking a weighted linear average of its own state and the state of its neighbors. This algorithm results in a linear dynamical system whose state transition matrix involves the Laplacian matrix of the underlying communication network.

Designing consensus algorithms with fast convergence speed is of significant practical interest because such algorithms allow the multi-agent systems to reach an agreement with fewer iterations and, therefore, by consuming less communication resource. In the context of the linear average consensus algorithm, the problem of finding the optimal weights of edges for maximizing the asymptotic consensus speed can be reduced to a convex optimization problem [3], under the assumption that the communication network is static and undirected. It was recently shown by the authors in [8] that the optimal weights can be computed in a distributed manner by an iterative computation. Zelazo et al. [9] clarified the role of cycles in the linear average consensus algorithm and presented a methodology for accelerating the consensus by adding edges to a network. On the other hand, for the case of directed networks, Hao and Barooah [10] presented a method to accelerate the convergence rate of a linear (but not necessarily an average) consensus algorithm by tuning the weights of edges in the network.

A natural consequence of seeking for further acceleration of consensus algorithms is the emergence of finite-time consensus algorithms [11], in which edge-weights are typically assumed to be time-varying and the designer exploits the additional flexibility to realize consensus in a finite-time. The finite-time consensus algorithm proposed in [12] achieves consensus by stochastic (but possibly asymmetric) matrices in $N(N − 1)/2$ iterations, where $N$ denotes the number of nodes in the network. The authors in [13], [14] used tools from graph signal processing (see, e.g., [15]) to show that, by allowing non-stochasticity for the state-update matrices, one can realize a finite-time consensus in at most $N$ steps. The theoretical aspects of these works have been further investigated in [16]. Recently, Falsone et al. [17] showed that the number of steps required for consensus can be further improved to $N/2$ in the specific case of ring networks having an even number of nodes.

Despite the aforementioned advances for consensus acceleration, there is still a lack of an effective methodology for answering the following basic question: Given a finite time-window as well as an underlying network structure, how should we dynamically tune the edge weights in the network for achieving as accurate consensus as possible at the end of the time-window? If the length of the time-window is not long enough to run the aforementioned finite-time consensus algorithms, currently available options are effectively limited to using the static optimal strategies (e.g., [3]), which does not allow us to dynamically tune the weights of the network. To fill in this gap, in this paper we present a data-driven approach for tuning the weights of an undirected temporal (i.e., time-varying) networks by using deep learning techniques. We first unfold the consensus algorithm and obtain a feedforward signal flow graph [18], which we regard as a neural network. We then use the standard stochastic gradient descent algorithm to train the parameters in each layer of the neural network (i.e., the weights of each snapshot of the temporal network) to minimize the consensus error over a finite time-horizon, which results in an optimized temporal network for faster consensus. We numerically confirm...
that our approach can drastically accelerate the convergence speed in the linear average consensus algorithm.

This paper is organized as follows. In Section II we state the problem of dynamically tuning the edge weights to accelerate the linear average consensus algorithm, and then we propose our methodology for solving the problem using standard techniques in the field of deep learning. In Section III we evaluate the performance of the proposed method with various numerical simulations. We finally conclude the paper in Section IV.

II. WEIGHT OPTIMIZATION BY DEEP LEARNING TECHNIQUES

In this section, we describe our methodology for tuning the edge weights of the networks for accelerating the linear average consensus algorithm within a given finite-time horizon. We first give a brief review of the linear average consensus algorithm and state its basic properties. We then describe our data-driven methodology for tuning the weights of the network, in which we apply the techniques in the deep-learning to the signal flow graph obtained by unfolding the consensus algorithm.

A. Linear average consensus algorithm

Let $G$ be an undirected and unweighted network having the node set $V = \{1, \ldots, N\}$ and the edge set $E$ consisting of unordered pairs of nodes in $V$. Each node in $G$ represents an agent, which is supposed to communicate with its neighbors at each time. In this paper, we focus on the discrete-time case. Let $x_i(k) \in \mathbb{R}$ denote the state of the $i$th node at time $k \geq 0$, and $N_i$ denote the set of neighbors of node $i$. In the standard linear average consensus protocol [1], each node $i$ updates its own state according to the following difference equation:

$$x_i(k+1) = x_i(k) + \sum_{j \in N_i} w_{ij}(k)(x_j(k) - x_i(k)), \quad x_i(0) = x_{0,i},$$

where $w_{ij}(k) = w_{ji}(k) \geq 0$ represents the weight of the (undirected) edge $\{i, j\}$ at time $k$ and $x_{0,i}$ is the initial state of node $i$. For each time $k \geq 0$, we define the adjacency matrix of the network $W(k) \in \mathbb{R}^{N \times N}$ by

$$W_{ij}(k) = \begin{cases} w_{ij}(k), & \text{if } j \in N_i, \\ 0, & \text{otherwise}. \end{cases}$$

Define the degree matrix of the network at time $k$ by

$$D(k) = \text{diag}(d_1(k), \ldots, d_N(k)), \quad d_i(k) = \sum_{j \in N_i} w_{ij}.$$ 

Then, the evolution of the state vector $x(k) = [x_1(k) \ldots x_N(k)]^\top$ in the linear average consensus protocol (1) is written as

$$x(k+1) = (I - L(k))x(k), \quad x(0) = x_0,$$

where $L(k) = D(k) - W(k)$ is the Laplacian matrix of the network and $x_0 = [x_{1,0} \ldots x_{N,0}]^\top$ denotes the initial state vector.

The objective of this paper is to present a framework for tuning the weights $\{w_{ij}(k)\}_{k \geq 0, (i,j) \in E}$ for the faster average consensus in a given finite time window. Let us denote the average of the initial states of the nodes by

$$c = \frac{1}{N} \sum_{i=1}^{N} x_{i,0}.$$ 

Define the consensus error vector

$$e(k) = x(k) - c1$$

where $1$ denotes the all-one $N$-dimensional column vector. We are now ready to state the problem studied in this paper.

Problem 2.1 (Consensus acceleration problem): Let $G$ be an undirected and unweighted network having $N$ nodes. Let $T$ be a positive integer. Assume that the set of initial states follow a probability distribution $\mathcal{X}_0$, i.e.,

$$\{x_{0,1}, \ldots, x_{0,N}\} \sim \mathcal{X}_0.$$ 

Find the set of nonnegative weights

$$\{w_{ij}(k)\}_{k \in [0, \ldots, T-1], (i,j) \in E}$$

that minimizes the average consensus error defined by

$$\gamma_T = E[\|e(T)\|]$$

where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^N$, and $E[\cdot]$ denotes the expected value.

Problem 2.1 is a non-convex problem and it is difficult to compute a set of $\{w_{ij}(k)\}_{k \geq 0, (i,j) \in E}$ that minimizes $\gamma_T$. This motivates us to tackle this problem using a data-driven approach to find a suboptimal solution. In the next subsection, we describe our data-driven approach for tuning the edge weights by using deep leaning techniques. We remark that, although we assume our knowledge of the initial probability distribution $\mathcal{X}_0$ in the process of optimization, the optimized edge-weights can drastically accelerate the consensus protocol even if the initial states do not follow the given distribution. We numerically illustrate this universality property of our approach in Subsection III-D.

B. Data-driven weight optimization

To adjust the weights by using deep leaning techniques, we first unfold the recursive state-update formula (1) and obtain a signal-flow graph shown in Fig. 1a. Unlike a standard deep neural network, the resulting neural network has a structure and contains no activation function. The structure of the neural network corresponds to the structure of the graph $G$, and the same between all layers. The neurons of $k$th layer corresponds to the nodes at time $k$ as shown in Fig. 1b.

We then apply a standard technique in the field of deep learning to adjust the weights; namely, we train the network with a stochastic gradient descent algorithm such as SGD.
RMSprop, or Adam. As in [18], we use the technique of the incremental training for adjusting the weights. In the incremental training, we first consider only the first layer (i.e., we set $k = 1$ in Fig. 1a) and attempt to minimize the loss function of the average consensus error $\gamma_1$ using a number of randomly generated initial state $x_0$ as the training data, which we call the 1st generation. After training the first set of weights $w_{ij}(0)$, we proceed to training the first two sets of edge weights by appending the second layer to the neural network and replacing the loss function by $\gamma_2$. In this training, we use the result from the 1st generation as the initial value of the first layer and train the entire neural network. We repeat this process to finally optimize the weights $w_{ij}(T−1)$ between $T−1$st and $T$th layers.

III. PERFORMANCE EVALUATION

In this section, we illustrate the effectiveness of the proposed method by various numerical simulations. The simulations were performed in PyTorch [19]. The loss function is the mean squared error, $\gamma^2_T$, and the size of minibatch is one. We used Adam for the optimization with learning rate 0.01. The number of data-set per learning is 10000. For evaluations, 100 samples were used.

A. Baseline strategy

Throughout this section, we compare the performance of the proposed method with that of the static optimal strategy presented in [3]. Assume that the initial state $x_0$ is a deterministic vector. Let us further suppose that the edge weights $w_{ij}(k)$ does not depend on time $k$. Under these assumptions, the authors in [3] have shown that the problem of finding the edge weights minimizing the asymptotic convergence factor

$$r_{\text{asym}} = \sup_{x_0} \limsup_{k \to \infty} \left( \frac{\|e(k)\|}{\|e(0)\|} \right)^{1/k}$$

reduces to solving a linear matrix inequality, which can be globally and efficiently solved [20]. As the baseline strategy, we use the following time-invariant consensus protocol

$$x_i(k+1) = x_i(k) + \sum_{j \in N_i} w_{ij}^{\text{stat}} (x_j(k) - x_i(k)),$$

$$0 \leq k \leq T - 1$$

where the weights $w_{ij}^{\text{stat}}$ are obtained from solving the linear matrix inequality.

B. Deterministic networks

In this subsection, we use the following two empirical and synthetic deterministic networks; Karate network [21] ($N = 34$ nodes) and the square lattice network ($N = 6^2 = 36$ nodes). We assume that the initial state of each node independently follows a uniform distribution on the interval $[-1, 1]$.

For Karate network, we set $T = 5$ and numerically optimize the edge weights of the networks at the times $k = 0, \ldots, 4$. In Fig. 2, we present the optimized weights of edges in the network. We then empirically evaluate the average consensus error $E[\|e(k)\|]$ for $k = 0, \ldots, 5$. The results are shown in Fig. 3. The accuracy of the consensus archived by the proposed method is 10 times better than the static optimal policy. We observe that the weights of the network are changing in a non-trivial manner. It is worth noting that the weights at time $k = 1$ are relatively large at various edges in the network. This sudden increase in edge weights in fact drives the nodes away from the consensus state but only temporarily. In Fig. 4, we show the sample trajectories of the average consensus protocol with the static optimal and proposed edge weights. We confirm that the proposed edge weights achieves a more precise average consensus at time $k = 5$ compared with the static optimal strategy.

We then consider the average consensus on the square lattice network. We set $T = 10$ and numerically optimize the edge weights of the network. The results are shown in Figs. 5 and 6. As in the case of Karate network, the accuracy of the consensus by the proposed at the final time $k = 10$ is about 10 times better than the static optimal methodology. The optimized weights show a trend similar to the one for

\[ x(0) \]
\[ L_0 \]
\[ x(1) \]
\[ L_1 \]
\[ x(2) \]
\[ \cdots \]
\[ x(T-1) \]
\[ x(T) \]
the Karate network. The weights at times $k = 1$ and $k = 6$ are relatively larger than the weights at other time instants, which leads to a temporal increase in the consensus error at times $k = 2$ and $k = 7$, respectively. Despite this phenomena, the proposed approach allows the nodes to achieve a better average consensus at the final time $k = 10$.

**C. Random synthetic networks**

We use the following three random and synthetic network models: the Erdős-Rényi (ER) network ($N = 100$ nodes and $M = 252$ edges, where the probability for edge creation is 0.05) the Barabási-Albert (BA) model [22] ($N = 100$ and
For each of the three random graph models, we create 10 realizations of networks, for which we run the proposed algorithm to obtain the optimized edge weights. We then empirically compute the mean consensus errors \( E[∥e(k)∥] \) for \( k = 0, \ldots, 10 \) for each of the 3 \( \times \) 10 cases. We show the results in Fig. 9. From the figure, we confirm that only the case of WS network model presents a temporal increase in the mean consensus errors, while the errors from the other two cases decrease monotonically.

### D. Periodic continuation

In the previous subsections, we have confirmed that the proposed method can drastically accelerate the average consensus algorithm under the assumption that we are given a prespecified finite time-window and that we know the distribution of the initial states. In this subsection, we further show that a periodic continuation of our algorithm yields an consensus algorithm that effectively accelerates the consensus for any initial state vector and over an infinite time-window.

For given \( T \), let \( G^*(0), \ldots, G^*(T - 1) \) denote the networks whose weights are optimized by the proposed method. Let \( L^*(0), \ldots, L^*(T - 1) \) denote the Laplacian matrices of the networks. Then, the proposed consensus algorithm is written as

\[
x(k + 1) = (I - L^*(k))x(k), \quad 0 \leq k \leq T - 1.
\]

By periodically extending the state transition matrices \( I - L^*(0), \ldots, I - L^*(T - 1) \), we obtain the following average consensus protocol over an infinite time horizon:

\[
x(sT + \tau + 1) = \prod_{j=0}^{\tau}(I - L^*(\tau - t))x(sT), \quad 0 \leq \tau \leq T - 1, \quad s \geq 0.
\] (4)

Define the asymptotic convergence factor of this algorithm by equation (2). The next lemma gives an explicit representation of the asymptotic convergence factor of the consensus algorithm (4).
Lemma 3.1: For given $T$, suppose that $L^*(0)$, $L^*(T-1)$ denote the optimized weighted Laplacian matrices of the networks. The asymptotic convergence factor of the consensus algorithm (4) equals

$$r_{\text{asym}} := \sup_{x_0 \neq 0} \limsup_{k \to \infty} \left( \frac{\|e(k)\|}{\|e(0)\|} \right)^{1/k} \leq \left( \prod_{i=0}^{T-1} (1 - L^*(T - 1 - t)) \right)^{1/T}.$$  \hspace{1cm} (5)

Proof: First note that

$$e(k + 1) = \prod_{t=0}^{k} (I - L^*(k - t)) e(0).$$

By expressing $k$ using $s$ and $\tau$ as

$$k = sT + \tau, \quad \tau \in \{0, \ldots, T-1\},$$

we have

$$\limsup_{k \to \infty} = \limsup_{s \to \infty} \max_{\tau \in \{0, \ldots, T-1\}} \left( \prod_{t=0}^{s} (I - L^*(sT + \tau - t)) \right).$$

This completes the proof.

Using Lemma 3.1, we compute the asymptotic convergence factor of the consensus algorithm (4) for each of the five networks (i.e., Karate, lattice, ER, BA, and WS networks). We also compute the asymptotic convergence factor of the baseline strategy (3) for each of the five networks. We show the asymptotic convergence factors in Table I. The proposed method achieves less convergence factors, which shows the effectiveness of the proposed approach even in the case of infinite time-horizon problems.

IV. CONCLUSION

In this paper, we have presented a data-driven approach for accelerating the linear average consensus algorithm over undirected temporal networks. We have first unfolded the consensus algorithm to obtain an equivalent feedforward signal flow graph, which we have regarded as a neural network. We have then showed that we can apply standard deep learning techniques to train the obtained neural network and obtain a temporal network having optimized edge-weights. We have numerically confirmed that our methodology can outperform the average consensus algorithm with the static optimal edge-weights.

REFERENCES

[1] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” Proceedings of the IEEE, vol. 95, pp. 215–233, 2007.
[2] W. Ren and R. W. Beard, “Consensus seeking in multiagent systems under dynamically changing interaction topologies,” IEEE Transactions on Automatic Control, vol. 50, pp. 655–661, 2005.
[3] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” Systems & Control Letters, vol. 53, pp. 65–78, 2004.
[4] J. Cortés and F. Bullo, “Coordination and geometric optimization via distributed dynamical systems,” SIAM Journal on Control and Optimization, vol. 44, pp. 1543–1574, 2005.
[5] K. Senel and M. Akar, “A distributed coverage adjustment algorithm for femtocell networks,” IEEE Transactions on Vehicular Technology, vol. 66, pp. 1739–1747, 2017.
[6] F. Dörfler and F. Bullo, “Synchronization and transient stability in power networks and nonuniform kuramoto oscillators,” SIAM Journal on Control and Optimization, vol. 50, pp. 1616–1642, 2012.
[7] R. Olfati-Saber and R. Murray, “Consensus problems in networks of agents with switching topology and time-delays,” IEEE Transactions on Automatic Control, vol. 49, pp. 1520–1533, 2004.
[8] L. Kempton, G. Herrmann, and M. Di Bernardo, “Self-organization of weighted networks for optimal synchronizability,” IEEE Transactions on Control of Network Systems, vol. 5, pp. 1541–1550, 2018.
[9] D. Zelazo, S. Schuler, and F. Allgöwer, “Performance and design of cycles in consensus networks,” Systems and Control Letters, vol. 62, pp. 85–96, 2013.
[10] H. Hao and P. Barooah, “Improving convergence rate of distributed consensus through asymmetric weights,” in 2012 American Control Conference, 2012, pp. 787–792.
[11] S. Sundaram and C. N. Hadjicostis, “Finite-time distributed consensus in graphs with time-invariant topologies,” in 2007 American Control Conference, 2007, pp. 711–716.
[12] J. M. Hendrickx, G. Shi, and K. H. Johansson, “Finite-time consensus using stochastic matrices with positive diagonals,” IEEE Transactions on Automatic Control, vol. 60, pp. 1070–1073, 2015.
[13] S. Safavi and U. A. Khan, “Revisiting finite-time distributed algorithms via successive nulling of eigenvalues,” IEEE Signal Processing Letters, vol. 22, pp. 54–57, 2015.
[14] Y. Shang, “Finite-time weighted average consensus and generalized consensus over a subset,” IEEE Access, vol. 4, pp. 2615–2620, 2016.
[15] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains,” IEEE Signal Processing Magazine, vol. 30, pp. 83–98, 2013.

[16] S. Apers and A. Sarlette, “Accelerating consensus by spectral clustering and polynomial filters,” IEEE Transactions on Control of Network Systems, vol. 4, pp. 544–554, 2017.

[17] A. Falsone, K. Margellos, S. Garatti, and M. Prandini, “Finite-time distributed averaging over gossip-constrained ring networks,” IEEE Transactions on Control of Network Systems, vol. 5, pp. 879–887, 2018.

[18] D. Ito, S. Takabe, and T. Wadayama, “Trainable ISTA for sparse signal recovery,” IEEE Transactions on Signal Processing, vol. 67, pp. 3113–3125, 2019.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” in 31st Conference on Neural Information Processing Systems, 2017.

[20] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, Linear Matrix Inequalities in System and Control Theory. Society for Industrial Mathematics, 1994.

[21] W. W. Zachary, “An information flow model for conflict and fission in small groups,” Journal of Anthropological Research, vol. 33, pp. 452–473, 1977.

[22] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” Science, vol. 286, pp. 509–512, 1999.

[23] D. J. Watts and S. H. Strogatz, “Collective dynamics of ’small-world’ networks,” Nature, vol. 393, pp. 440–442, 1998.