ESTIMATION AND LEARNING OF DYNAMIC NONLINEAR NETWORKS (DYNNETS)

Mojtaba Sahraee-Ardakan and Alyson K. Fletcher

University of California at Los Angeles

ABSTRACT

Learning high-dimensional systems from data is often computationally challenging in the presence of nonlinearities and dynamics. This paper proposes a novel approach for identification of high-dimensional systems based on decomposing systems into networks of low-dimensional linear dynamical subsystems with memoryless, scalar nonlinear feedback elements and memoryless, linear interactions. The proposed model, called Dynamic Nonlinear Networks (DyNNets), can encompass a wide range of complex phenomena and is particularly well-suited for modeling neuronal systems. It is shown that the posterior density of the hidden states given the unknown parameters of a DyNNet admits a factorable structure that separates the linear dynamics, memoryless nonlinearities, and linear interactions. This factorization enables efficient implementation of maximum a posteriori (MAP) state estimation and system identification via the alternating direction method of multipliers (ADMM). The methodology is illustrated on estimation of neural mass models.

***Index Terms*—** ADMM, neural modeling, nonlinear systems

1. INTRODUCTION

Many estimation and identification problems for biological neural systems are made challenging by the combination of three key features: large numbers of unknown or hidden variables, nonlinearities, and dynamics. Learning such high-dimensional nonlinear dynamical systems typically suffers from the *curse of dimensionality*: The computational complexities of basic tasks such as optimal state estimation grow exponentially in the dimension of the state. To address this challenge, this paper presents a novel framework for high-dimensional system identification based on decomposing large systems into networks of low-dimensional linear dynamical subsystems in feedback with scalar, nonlinear memoryless elements and linear, memoryless interconnections between the subsystems. We call these networks Dynamic Nonlinear Networks (DyNNets). This representation is extremely general: it can model arbitrary network topologies, nonlinearities, and non-Gaussian elements in feedback and feedforward connections.

We show that in any network represented as a DyNNet, the posterior density of the hidden signals given the parameters, inputs, and outputs is factorable. This factorization enables efficient joint ML-MAP estimation based on the classic Alternating Direction Method of Multipliers (ADMM) [1]. The ADMM-based algorithm reduces the nonlinear estimation problem to a sequence of low-dimensional problems that can be solved efficiently. The ADMM method is non-convex, and thus convergence of the method cannot be guaranteed. However, we illustrate fast convergence and high accuracy in a simulation of the method for connectivity detection in a neural mass model.

1.1. Related Work

The feedback representation of a network we propose here is inspired, in part, by the linear fractional transform (LFT) that is used in robust control [2, 3]. The LFT was used for worst-case analysis and identification of linear dynamical systems alongside unstructured, norm-bounded uncertainties. Here, the norm-bounded uncertainty is replaced by a set of low-dimensional nonlinear systems.

The DyNNet model proposed here can be viewed as a generalization of the standard Recurrent Neural Network (RNN) [4], with the DyNNet being able to incorporate a richer class of dynamics, nonlinearities and stochastic elements within each hidden unit. However, the DyNNet also imposes a particular factorable structure that greatly simplifies the learning. Specifically, when a neural network has stochastic dynamics in the hidden states, parameter learning typically requires estimation of the posterior density of the hidden states, which is usually the computational bottleneck.

The proposed learning algorithm for the DyNNet uses two modifications to reduce the complexity: First, to avoid approximate inference of the posterior, we compute only a maximum a posteriori (MAP) point estimate for each parameter choice. Second, we exploit the factorable structure of the DyNNet to efficiently compute this MAP point estimate via ADMM. We show that these modifications significantly reduce the complexity of learning and estimation.

Finally, an ADMM approach for learning weights in an RNN is also presented in [5], but it requires that the nonlin-

This work was supported in part by the U.S. National Science Foundation under Grant 1254204 and in part by the Office of Naval Research Grant N00014-15-1-677.

earities are essentially invertible so that they can be ignored. ADMM and related techniques such as elastic averaging [6,7] have also been use to parallelize learning of deep neural nets over multiple sets of training data.

Due to space considerations some of the detailed derivations and simulation results are provided in a full paper [8].

2. DYNAMIC NONLINEAR NETWORKS

Abstractly, a DyNNet is an interconnected system with three types of components: (i) N low dimensional linear dynamical subsystems; (ii) memoryless nonlinear feedback elements within each subsystem; and (iii) linear interactions between the subsystems. The network can be precisely described as follows: Dynamics of the ith subsystem, $i = 1, \ldots, N$, in state-space form [9] is given as

$$\begin{bmatrix} \mathbf{x}_i^{k+1} \\ \mathbf{z}_i^k \\ \mathbf{v}_i^k \\ \mathbf{y}_i^k \end{bmatrix} = \left[\begin{array}{c|cccc} \mathbf{A}_i & \mathbf{B}_{ir} & \mathbf{B}_{is} & \mathbf{B}_{id} & \mathbf{B}_{iu} \\ \hline \mathbf{C}_{iz} & \mathbf{D}_{izr} & \mathbf{D}_{izs} & \mathbf{D}_{izd} & \mathbf{D}_{izu} \\ \mathbf{C}_{iv} & \mathbf{D}_{iyr} & \mathbf{D}_{ivs} & \mathbf{D}_{ivd} & \mathbf{D}_{ivu} \\ \mathbf{C}_{iy} & \mathbf{D}_{iyr} & \mathbf{D}_{iys} & \mathbf{D}_{iyd} & \mathbf{D}_{iyu} \end{array}\right] \begin{bmatrix} \mathbf{x}_i^k \\ \mathbf{r}_i^k \\ \mathbf{s}_i^k \\ \mathbf{d}_i^k \\ \mathbf{u}_i^k \end{bmatrix}, \tag{1}$$

where $k \in \{0, \ldots, T-1\}$ is the time index, $\mathbf{x}_i^k$ is the internal state, $\mathbf{d}_i^k$ is a noise term, $\mathbf{u}_i^k$ is a known input, $\mathbf{y}_i^k$ is a measured output, and $\mathbf{z}_i^k$, $\mathbf{v}_i^k$, $\mathbf{r}_i^k$, and $\mathbf{s}_i^k$ are additional hidden signals; these all may be scalars or column vectors. The matrices $\mathbf{A}_i, \mathbf{B}_{ir}, \ldots$ are system matrices and $\mathbf{d}_i^k$ is i.i.d. with $\mathbf{d}_i^k \sim \mathcal{N}(0,1)$. The signals $\mathbf{v}_i^k$ and $\mathbf{s}_i^k$ are related by a general probabilistic transition function

$$p(\mathbf{s}_i^{k+1} | \mathbf{v}_i^k, \lambda_i), \tag{2}$$

with parameters λ_i that may need to be learned. This transition function can introduce nonlinear and non-Gaussian stochastic components into the network. The signals $\mathbf{z}_i^k$ and $\mathbf{r}_i^k$ represent the linear interconnections between subsystems

$$\mathbf{r}_i^{k+1} = \sum_{j=1}^{N} \mathbf{W}_{ij} \mathbf{z}_i^k, \tag{3}$$

where $\mathbf{W}_{ij}$ is linear interaction factor between subsystems i and j. In the sequel, for signals such as $\mathbf{x}$, we will use the notation $\mathbf{x}^k$ to be column vector with values of $\mathbf{x}_i^k$ and let $\mathbf{x}$ denote the T-column matrix of $\mathbf{x}^k$s over all times k.

Below, it is shown how to fit a DyNNet to the data, and an application for modeling neural data is presented in Section 4.

3. LEARNING DyNNets

Let θ be the set of all parameters in the system

$$\theta = \{\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i, \mathbf{D}_i, \mathbf{W}, \lambda_i\}, \tag{4}$$

where $(\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i, \mathbf{D}_i)$ are the matrices in the linear system (1), $\mathbf{W}$ is the interconnection matrix in (3), and λ_i are parameters in the nonlinear feedback (2). Also, let $\mathbf{q}$ be the set of all hidden signals in the model (1),

$$\mathbf{q} = \{\mathbf{x}, \mathbf{z}, \mathbf{v}, \mathbf{r}, \mathbf{s}\}. \tag{5}$$

Our goal is to estimate the hidden variables $\mathbf{q}$ and learn the parameters θ from the input–output data $(\mathbf{u}, \mathbf{y})$. We consider *joint MAP-ML* estimation for $\mathbf{q}$ and θ:

$$\begin{aligned} (\widehat{\mathbf{q}}, \widehat{\theta}) = \arg\max_{\mathbf{q},\theta} p(\mathbf{q}, \mathbf{y} | \mathbf{u}, \theta) = \arg\min_{\mathbf{q},\theta} F(\mathbf{q}, \theta), \\ F(\mathbf{q}, \theta) := -\ln p(\mathbf{q} | \mathbf{u}, \mathbf{y}). \end{aligned} \tag{6}$$

We propose to approximately perform the optimization (6) via alternating between two steps: (i) *State estimation:* For a given parameter estimate $\widehat{\theta}$, we find the MAP estimate of $\mathbf{q}$:

$$\widehat{\mathbf{q}} = \arg\max_{\mathbf{q}} p(\mathbf{q} | \mathbf{u}, \mathbf{y}, \widehat{\theta}); \tag{7}$$

and (ii) *Parameter update:* Given the estimate $\widehat{\mathbf{q}}$ of the hidden variables $\mathbf{q}$, we update the parameters θ via a gradient step.

3.1. State Estimation via ADMM

We first consider the state estimation problem (7). Since the parameters are known, we drop the dependence on the terms in θ. Under the i.i.d. Gaussian assumption of $\mathbf{d}_i^k$ and the transition probabilities $p(\mathbf{s}_i^{k+1} | \mathbf{v}_i^k)$, the negative log posterior in (6) can be written as a sum of three energy functions,

$$F(\mathbf{q}) = F_{\text{lin}}(\mathbf{q}) + F_{\text{nl}}(\mathbf{q}) + F_{\text{mix}}(\mathbf{q}), \tag{8}$$

where

$$F_{\text{lin}}(\mathbf{q}) := -\sum_{k=0}^{T-1} \sum_{i=1}^{N} \ln p(\mathbf{x}_i^{k+1}, \mathbf{z}_i^k, \mathbf{v}_i^k, \mathbf{y}_i^k | \mathbf{x}_i^k, \mathbf{r}_i^k, \mathbf{s}_i^k), \tag{9a}$$

$$F_{\text{nl}}(\mathbf{q}) := F_{\text{nl}}(\mathbf{s}, \mathbf{v}) = -\sum_{k=0}^{T-1} \sum_{i=1}^{N} \ln p(\mathbf{s}_i^{k+1} | \mathbf{v}_i^k), \tag{9b}$$

$$F_{\text{mix}}(\mathbf{q}) := F_{\text{mix}}(\mathbf{r}, \mathbf{z}) = \sum_{k=0}^{T-1} \delta(\mathbf{r}^{k+1} - \mathbf{W}\mathbf{z}^k), \tag{9c}$$

where the delta function denotes

$$\delta(\mathbf{r}^{k+1} - \mathbf{W}\mathbf{z}^k) = \begin{cases} 0, & \text{if } \mathbf{r}^{k+1} = \mathbf{W}\mathbf{z}^k; \\ \infty, & \text{if } \mathbf{r}^{k+1} \neq \mathbf{W}\mathbf{z}^k. \end{cases}$$

To find the MAP estimate of the states, $F(\mathbf{q}, \theta)$ in (6) should be minimized over $\mathbf{q}$. Using ADMM [1], the partitioning in (8) can be exploited to turn the optimization problem into multiple low-dimensional optimizations. Specifically, we

use a technique known as *variable splitting* where we introduce another set of variables $\mathbf{q}'$ and rewrite the MAP estimation problem as

$$\min_{\mathbf{q},\mathbf{q}'} [F_{\text{lin}}(\mathbf{q}) + F_{\text{nl}}(\mathbf{q}') + F_{\text{mix}}(\mathbf{q}')] \quad \text{s.t.} \quad \mathbf{q} = \mathbf{q}', \quad (10)$$

where $\mathbf{q} = \mathbf{q}'$ means $\mathbf{x} = \mathbf{x}'$, $\mathbf{z} = \mathbf{z}'$, ... Corresponding to these constraints, let $\boldsymbol{\mu}$ be a collection of dual parameters $\boldsymbol{\mu}_x$, $\boldsymbol{\mu}_z$, ... of identical size to their corresponding variable, and $\boldsymbol{\gamma}$ be a set of positive weights γ_x, γ_z, ... Then the weighted inner product between $\boldsymbol{\mu}$ and $\mathbf{q}$ can be defined as

$$\langle \boldsymbol{\mu}, \mathbf{q} \rangle_{\boldsymbol{\gamma}} := \gamma_x \operatorname{Tr}(\boldsymbol{\mu}_x^{\mathsf{T}} \mathbf{x}) + \cdots + \gamma_s \operatorname{Tr}(\boldsymbol{\mu}_s^{\mathsf{T}} \mathbf{s}), \quad (11)$$

where the summation is over all the components in $\mathbf{q}$ and $\operatorname{Tr}(\cdot)$ is the trace of a matrix. Also define the weighted norm of $\mathbf{q}$ as

$$\|\mathbf{q}\|_{\boldsymbol{\gamma}}^2 := \langle \mathbf{q}, \mathbf{q} \rangle_{\boldsymbol{\gamma}} = \gamma_x \|\mathbf{x}\|_F^2 + \cdots + \gamma_s \|\mathbf{s}\|_F^2. \quad (12)$$

With these definitions, the augmented Lagrangian corresponding to (10) is defined as

$$\begin{aligned} L(\mathbf{q}, \mathbf{q}', \boldsymbol{\mu}) :=& F_{\text{lin}}(\mathbf{q}) + F_{\text{nl}}(\mathbf{q}') + F_{\text{mix}}(\mathbf{q}') + \langle \boldsymbol{\mu}, \mathbf{q} - \mathbf{q}' \rangle_{\boldsymbol{\gamma}} \\ &+ \frac{1}{2} \|\mathbf{q} - \mathbf{q}'\|_{\boldsymbol{\gamma}}^2. \end{aligned}$$

The ADMM iterations [1] are then given by

$$\widehat{\mathbf{q}} \leftarrow \arg\min_{\mathbf{q}} L(\mathbf{q}, \widehat{\mathbf{q}}', \boldsymbol{\mu}), \quad (13a)$$

$$\widehat{\mathbf{q}}' \leftarrow \arg\min_{\mathbf{q}} L(\widehat{\mathbf{q}}, \mathbf{q}, \boldsymbol{\mu}), \quad (13b)$$

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \widehat{\mathbf{q}} - \widehat{\mathbf{q}}'. \quad (13c)$$

These minimizations can be simplified as follows: First, let $\bar{\mathbf{q}} = \widehat{\mathbf{q}} + \boldsymbol{\mu}$, then we see that $\widehat{\mathbf{q}}$ in (13a) can be obtained by the minimization

$$\begin{aligned} \widehat{\mathbf{q}} &\leftarrow \arg\min_{\mathbf{q}} H_{\text{lin}}(\mathbf{q} | \bar{\mathbf{q}}, \boldsymbol{\gamma}) \\ &= \arg\min_{\mathbf{q}} \left[F_{\text{lin}}(\mathbf{q}) + \frac{1}{2} \|\mathbf{q} - \bar{\mathbf{q}}\|_{\boldsymbol{\gamma}}^2 \right]. \end{aligned} \quad (14)$$

Similarly, let $\bar{\mathbf{q}} = \widehat{\mathbf{q}} - \boldsymbol{\mu}$, then the minimization over $\widehat{\mathbf{q}}'$ in (13b) is equivalent to minimizing two separate functions

$$\begin{aligned} (\widehat{\mathbf{s}}', \widehat{\mathbf{v}}') &\leftarrow \arg\min_{\mathbf{s},\mathbf{v}} H_{\text{nl}}(\mathbf{s}, \mathbf{v} | \bar{\mathbf{s}}, \bar{\mathbf{v}}, \gamma_s, \gamma_v) \\ &= \arg\min_{\mathbf{s},\mathbf{v}} \left[F_{\text{nl}}(\mathbf{s}, \mathbf{v}) + \frac{\gamma_s}{2} \|\mathbf{s} - \bar{\mathbf{s}}\|_F^2 + \frac{\gamma_v}{2} \|\mathbf{v} - \bar{\mathbf{v}}\|_F^2 \right], \end{aligned} \quad (15)$$

$$\begin{aligned} (\widehat{\mathbf{r}}', \widehat{\mathbf{z}}') &\leftarrow \arg\min_{\mathbf{r},\mathbf{z}} H_{\text{mix}}(\mathbf{r}, \mathbf{z} | \bar{\mathbf{r}}, \bar{\mathbf{z}}, \gamma_r, \gamma_z) \\ &= \arg\min_{\mathbf{r},\mathbf{z}} \left[F_{\text{mix}}(\mathbf{r}, \mathbf{z}) + \frac{\gamma_r}{2} \|\mathbf{r} - \bar{\mathbf{r}}\|_F^2 + \frac{\gamma_z}{2} \|\mathbf{z} - \bar{\mathbf{z}}\|_F^2 \right]. \end{aligned} \quad (16)$$

We assume that there are N linear time-invariant systems with d states per system, the nonlinear functions are discretized to L outputs, there are T time steps, and that $T \gg N \gg d$. It is shown in the full paper [8] that the minimizations can be performed via the following:

- (14) can be performed via N linear Kalman smoothers with T time steps each, with complexity $O(NTd)$.
- (15) can be performed via NT nonlinear minimizations, each minimization of the dimension $(\mathbf{s}_i^{k+1}, \mathbf{v}_i^k)$ and total complexity of $O(NTL)$.
- (16) can be performed via T least-squares minimizations with N variables each, with a complexity of $O(N^2 T d)$.

The total complexity is $O(N^2 T)$, which is identical to simply simulating the system. Convergence of ADMM can be guaranteed for convex penalties [1]. Unfortunately, since the nonlinear penalty (9b) is generally non-convex, the convergence is not guaranteed here. Nevertheless, ADMM often works well in practice, as we will show in Section 4.

3.2. Parameter Learning via Gradient Descent

We now turn to the problem of updating the estimates of the parameters θ. Given the negative log posterior energy function $F(\mathbf{q}, \theta)$ in (6), define

$$\bar{F}(\theta) := \min_{\mathbf{q}} F(\mathbf{q}, \theta). \quad (17)$$

Thus, the optimization problem (6) is equivalent to finding $\widehat{\theta} = \arg\min_\theta \bar{F}(\theta)$. We will show how we can compute a gradient step for this optimization using the state estimation procedure described in the previous section. First, we partition the parameters θ in (4) into three components,

$$\theta_{\text{lin}} = \{\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i, \mathbf{D}_i\}, \quad \theta_{\text{mix}} = \mathbf{W}, \quad \theta_{\text{nl}} = \{\lambda_i\}. \quad (18)$$

Next, observe that the minimization over $\mathbf{q}$ is actually a constrained optimization. Specifically, there are two constraints: First, the variables in $\mathbf{q}$ must satisfy the linear dynamical equations (1) for each system i which we write as $\mathbf{G}(\theta_{\text{lin}}, \mathbf{q}) = 0$, for some operator $\mathbf{G}$ that is (separately) linear in both θ_{lin} and $\mathbf{q}$. Secondly, for the mixing terms we have $\mathbf{z} = \mathbf{W}\mathbf{r}$. Now, the MAP estimation algorithm in the previous section approximately solves

$$\widehat{\mathbf{q}} = \arg\min_{\mathbf{q}} F(\mathbf{q}, \theta) \quad \text{s.t.} \quad G(\theta_{\text{lin}}, \mathbf{q}) = 0, \ \mathbf{z} = \mathbf{W}\mathbf{r}. \quad (19)$$

Corresponding to this optimization, define the Lagrangian

$$L(\mathbf{q}, \theta, \nu) := F(\mathbf{q}, \theta) + \langle \nu_{\text{lin}}, G(\theta_{\text{lin}}, \mathbf{q}) \rangle + \langle \nu_{\text{mix}}, \mathbf{z} - \mathbf{W}\mathbf{r} \rangle$$

for dual parameters $\nu = (\nu_{\text{lin}}, \nu_{\text{mix}})$. Since $\widehat{\mathbf{q}}$ is a critical point of the constrained optimization (19), it must be a critical

point of this Lagrangian for some set of dual parameters ν. Therefore (see [10])

$$\frac{\partial \bar{F}(\theta)}{\partial \theta} = \frac{\partial}{\partial \theta} L(\widehat{\mathbf{q}}, \theta, \nu). \tag{20}$$

Thus, we can compute the gradient if we can compute the dual parameters. It is shown in [8] that these dual parameters can be computed with minimal additional computational cost relative to minimizations performed for computing $\widehat{\mathbf{q}}$.

Interestingly, in the case when there is no noise in the dynamics of the linear systems (so that the states are exactly known given the parameters), the gradient for the parameters reduces to the standard back-propagation in time of [4].

4. NEURAL MASS MODELING EXAMPLE

Electrocorticography (ECoG) is the practice of using electrodes placed directly on the exposed surface of the brain to record activity from the cerebral cortex. One possible way to describe the ECoG responses is via *neural mass models* [11].

Suppose there are N electrodes measured over T time steps at some sampling period Δ. Let y_i^k be the electrical local field potential (LFP) measured on electrode i at time k. Associated with each electrode, we assume there is a neural mass with a scalar internal state x_i^k. Based partially on the "nearly linearized" model [12–14], we assume that the state of the neural mass and the electrode measurement evolve as

$$\begin{aligned} x_i^{k+1} &= (1-\alpha)x_i^k + \xi_i^k + f\left[\sum_{j=1}^{N} W_{ij} x_j^{k-\delta}\right] + d_{i1}^k, \\ y_i^k &= x_i^k + d_{i2}^k, \end{aligned} \tag{21}$$

where α is a decay time constant and

$$f(v) = \frac{1}{1+\exp(-(v-\mu)/a)} \tag{22}$$

is a sigmoidal activation function. The signals $d_i = (d_{i1}; d_{i2})$ are i.i.d. zero-mean Gaussian noise with variance $(\sigma_1^2; \sigma_2^2)$ and W_{ij} represents a connectivity strength from location j to location i with a connectivity delay of δ time steps. The input ξ_i^k to the model represents aggregate signals from other brain regions such as earlier stages of perceptual processing, which are usually sparse [15]. As a simple model for the sparse input, we assume that the signals ξ_i^k are i.i.d. with density

$$p(\xi_i^k) = (1-\rho)\delta(\xi_i^k) + \frac{\rho}{S}\exp(-\xi_i^k/S). \tag{23}$$

The problem is to both estimate the sparse inputs ξ_i^k and learn the connectivity matrix $\mathbf{W}$. For simplicity, in this example we assume all other parameters are known.

The model (21) fits easily into the DyNNet framework by defining $z_i^k = x_i^k$, $r_i^{k+1} = \sum_j W_{ij} x_i^{k-\delta+1}$, $v_i^k = r_i^k$ and $s_i^k = (s_{i1}^k; s_{i2}^k)$, $s_{i1}^k = \xi_i^k$, $s_{i2}^k = f(v_i^{k-1})$.

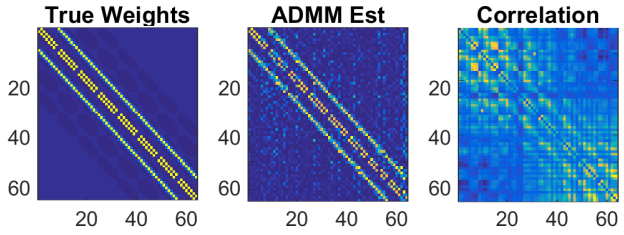

Fig. 1. Connectivity estimation for the neural mass model. Left: True connectivity strength; Middle: Estimated connectivity from the proposed algorithm; Right: Correlation between measured activities.

To obtain the ground truth data, we simulate the above network (21) with the parameters described below. We assumed an 8×8 ECoG array ($N = 64$) similar to that used in [16]. The electrode spacing is $\lambda = 250$ μm, $\Delta = 1$ ms, and $\delta = 2$. The internal states x_i^k in (21) would represent the LFP measurements, and we set α to 0.005. In the saturation function (22), we set $\mu = 1$ and $a = 0.3$. We also assume σ_1 and σ_2 are 5 dB and 20 dB below the averages of $|x_i^k|^2$ and $|y_i^k|^2$, respectively, $S = 4$, and $\rho N/\Delta = 10$. The network is simulated for 5 s. For the true connectivity matrix, we assumed a difference of Gaussians (or "Mexican Hat") model [12],

$$W_{ij} = \frac{c_1}{a_1} e^{-d_{ij}^2/(2a_1)} - \frac{c_2}{a_2} e^{-d_{ij}^2/(2a_2)}, \tag{24}$$

where d_{ij} is the distance between the two masses, and c_1, c_2, a_1 and a_2 are parameters. Here we set $c_1 = 0.25$, $c_2 = 0.005$, $a_1 = 0.5\lambda$, and $a_2 = 1.5\lambda$. We take $a_1 < a_2$ so that the connectivity is positive (excitatory) for close neural masses and negative (inhibitory) for further away masses.

The results are shown in Fig. 1. The left panel shows a color plot of the true connectivity matrix $\mathbf{W}$. Since there are $N = 64$ nodes, this is a 64×64 matrix. The banded structure arises since each node lies on an 8×8 grid and has the strongest connectivity (the yellow color) to its four neighbors. The connectivity estimate from the proposed algorithm is shown in the middle panel and can be seen to recover the connectivity well. As a comparison, one could estimate the connectivity by the simple correlation of the measured y_i^k with y_j^{k-D} for some delay D. This is shown in the right panel. Clearly it recovers the weights poorly.

5. CONCLUSIONS

We have presented a general framework for modeling nonlinear dynamical networks that partitions the dynamics and nonlinearities and divides high-dimensional networks into lower-dimensional ones. This partitioning enables efficient estimation and learning and may have immediate use for neural inference problems such as [17, 18]. Another line of work would be to study convergence along the lines of [19–21].

6. REFERENCES

[1] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[2] Kemin Zhou, John Comstock Doyle, Keith Glover, et al., *Robust and Optimal Control*, vol. 40, Prentice Hall, 1996.

[3] Greg Wolodkin, Sundeep Rangan, and Kameshwar Poolla, "An LFT approach to parameter estimation," in *Proc. IEEE Amer. Contr. Conf.*, 1997, vol. 3, pp. 2088–2092.

[4] Simon Haykin, Ed., *Kalman Filtering and Neural Networks*, vol. 47 of *Adaptive and Learning Systems for Signal Processing, Communications, and Control*, John Wiley & Sons, 2004.

[5] Simone Scardapane, Dianhui Wang, and Massimo Panella, "A decentralized training algorithm for echo state networks in distributed big data applications," *Neural Networks*, vol. 78, pp. 65–74, June 2016.

[6] Mu Li, Li Zhou, Zichao Yang, Aaron Li, Fei Xia, David G Andersen, and Alexander Smola, "Parameter server for distributed machine learning," in *Big Learning NIPS Workshop*, 2013, vol. 1.

[7] Sixin Zhang, Anna E Choromanska, and Yann LeCun, "Deep learning with elastic averaging SGD," in *Proc. NIPS*, 2015, pp. 685–693.

[8] Mojtaba Sahraee-Ardakan and Alyson K. Fletcher, "Estimation and learning of dynamic nonlinear networks (DyNNets)," arXiv e-Print, 2017.

[9] T. Kailath, *Linear Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[10] Jorge Nocedal and Stephen J. Wright, *Numerical Optimization*, Springer Verlag, 2006.

[11] Rosalyn J. Moran, Stefan J. Kiebel, K. E. Stephan, R. B. Reilly, Jean Daunizeau, and Karl J. Friston, "A neural mass model of spectral responses in electrophysiology," *NeuroImage*, vol. 37, no. 3, pp. 706–720, 2007.

[12] Romain Veltz, *Nonlinear Analysis Methods in Neural Field Models*, Ph.D. thesis, Paris Est, 2011.

[13] Byron M. Yu, John P. Cunningham, Gopal Santhanam, Stephen I. Ryu, Krishna V. Shenoy, and Maneesh Sahani, "Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity," in *Adv. Neural Inform. Process. Syst.*, 2009, pp. 1881–1888.

[14] Jakob H. Macke, Lars Buesing, John P. Cunningham, Byron M. Yu, Krishna V. Shenoy, and Maneesh Sahani, "Empirical models of spiking in neural populations," in *Proc. NIPS*, 2011, pp. 1350–1358.

[15] Bruno A. Olshausen and David J. Field, "Sparse coding of sensory inputs," *Curr. Op. Neurobiology*, vol. 14, no. 4, pp. 481–487, 2004.

[16] JuiChih Wang, Michael Trumpis, Michele Insanally, Robert Froemke, and Jonathan Viventi, "A low-cost, multiplexed electrophysiology system for chronic μecog recordings in rodents," in *Proc. 36th Ann. IEEE Int. Conf. Engineering in Medicine and Biology Soc. (EMBC)*, 2014, pp. 5256–5259.

[17] Alyson K. Fletcher, Sundeep Rangan, Lav Varshney, and Aniruddha Bhargava, "Neural reconstruction with approximate message passing (NeuRAMP)," in *Proc. Neural Information Process. Syst.*, Granada, Spain, Dec. 2011.

[18] Alyson K. Fletcher and Sundeep Rangan, "Scalable inference for neuronal connectivity from calcium imaging," in *Proc. NIPS*, 2014, pp. 2843–2851.

[19] Ulugbek S. Kamilov, Sundeep Rangan, Alyson K. Fletcher, and Michael Unser, "Approximate message passing with consistent parameter estimation and applications to sparse learning," *IEEE Trans. Inform. Theory*, vol. 5, no. 60, pp. 2969–2985, 2014.

[20] Sundeep Rangan, Philip Schniter, Erwin Riegler, Alyson Fletcher, and Volkan Cevher, "Fixed points of generalized approximate message passing with arbitrary matrices," in *Proc. IEEE Int. Symp. Information Theory*, 2013, pp. 664–668.

[21] Alyson K. Fletcher, Mojtaba Sahraee-Ardakan, Sundeep Rangan, and Philip Schniter, "Expectation consistent approximate inference: Generalizations and convergence," arXiv e-Print arXiv:1602.07795, 2016.