ONLINE CLASSIFICATION OF DYNAMIC MULTILAYER-NETWORK TIME SERIES IN RIEMANNIAN MANIFOLDS

Cong Ye^{*} Konstantinos Slavakis^{*} Johan Nakuci[†] Sarah F. Muldoon[‡] John Medaglia[§]

^{*}Department of Electrical Engineering, University at Buffalo (UB), SUNY, USA

[†]Neuroscience Program, UB, SUNY, USA

[‡]Department of Mathematics and CDS&E Program, UB, SUNY, USA

[§]Department of Psychology, Drexel University and Perelman School of Medicine, University of Pennsylvania, USA

ABSTRACT

This work exploits Riemannian manifolds to introduce a geometric framework for online state and community classification in dynamic multilayer networks where nodes are annotated with time series. A bottom-up approach is followed, starting from the extraction of Riemannian features from nodal time series, and reaching up to online/sequential classification of features via geodesic distances and angular information in the tangent spaces of a Riemannian manifold. As a case study, features in the Grassmann manifold are generated by fitting a kernel autoregressive-moving-average model to the nodal time series of the multilayer network. The paper highlights also numerical tests on synthetic and real brain-network data, where it is shown that the proposed geometric framework outperforms state-of-the-art deep-learning models in classification accuracy, especially in cases where the number of training data is small with respect to the number of the testing ones.

Index Terms— Network, multilayer, classification, online, Riemannian manifold.

1. INTRODUCTION

Classifying network states and communities is useful to describe the dynamics within numerous real systems, such as brains, social groups, wireless communications, material, and gene networks [1–4]. With a typical network consisting of nodes and edges, this work considers the case where nodes are annotated with time series, as in functional magnetic resonance imaging (fMRI), which monitors blood oxygen-level dependent (BOLD) time series [5], and electroencephalography (EEG), where time series are collected by electrodes placed on the scalp of a subject. Recent developments point towards the direction of multilayer (a.k.a. multiscale, multiview, or multiplex) networks [6–9], where layers differentiate modes of connectivity among nodes; e.g., subjects in a medical study, functional covariance in BOLD fMRI data, or frequency bands in EEG [10].

Two pivotal *supervised* learning tasks in network-data analytics, with a plethora of applications in diverse areas of networks, are state, e.g., [11, 12], and community, e.g., [13, 14], classification. In the present context, “state” refers to a “*global*” connectivity pattern among *all* nodes in a network that stays fixed over some time interval. In multilayer networks, “state” includes also connections among nodes from different layers. “State classification” is defined as the task of assigning a label to a global connectivity pattern of the network taken from a set of pre-defined labels that have been extracted beforehand from training data and are available to the user (supervised learning). Moreover, a network “community” is defined as a group of nodes that share common characteristics/features among their time series. Nodes that belong to the same community are considered to be “more connected” to each other than to the rest of the nodes in the network [15]. “Community classification” is defined as the task of assigning nodes to pre-defined communities that have been learned from training data and are available to the user. In state

and community classification, label assignment is based on the computation of the “proximity” of features, which are extracted from the nodal time series of the network, to features that have been extracted beforehand from the available training data.

It is crucial in applications with a massive volume of data, e.g., long time series, to devise algorithms that run sequentially on features and achieve a low computational footprint by classifying features on the fly as soon as a new feature is observed, without the need to process and keep in memory all of the observed data. Most of the online classification frameworks for network time series use popular machine-learning schemes as off-the-shelf tools, e.g., hidden Markov models for state classification [11, 12], and sequential support vector machines [13] as well as the wavelet transform [14] for community classification. In contrast to the previously mentioned frameworks, which cannot address both state- and community-classification tasks in a unifying way, the recently introduced temporal deep-learning (DL) models [16–19], although designed for single-layer networks, can accommodate both state and community classification sequentially. Notwithstanding, state-classification tasks in [16–19] require the knowledge of the starting and ending points of each of the states in the testing data, which is hardly the case in real-life scenarios.

This work spins off from [20], introduced very recently to address sequential clustering (i.e., *unsupervised* classification) in Riemannian manifolds for dynamic multilayer networks. As a case study, features are extracted from nodal time series via a kernel autoregressive-moving-average model, and mapped into the Grassmann manifold (Grassmannian) [21, 22]; a well-known Riemannian manifold. Landmark points are identified to concisely describe the geometry of the possibly massive point-cloud of features, and a novel online classification scheme is brought forth, coined online geodesic classification by tangent spaces (onlineGCT). This work highlights also the numerical tests of onlineGCT against the popular DL models GoogLeNet [23], DenseNet [24], ResNet [25], VGG-16 and VGG-19 [26], which are widely used in bio-medical image classification [27–29]. Notice that DenseNet has the largest size (201 layers deep), VGG-16 has the smallest one (16 layers deep), and that a small-size DL net requires in general less training data to tune its parameters than large-size nets do. Models [23–26] do not suffer from the previously mentioned issues of [16–19] with regards to state classification. Numerical tests on synthetic and real data for state and community classification demonstrate that the proposed framework outperforms [23–26] in classification accuracy, especially in cases where the number of training data is small with respect to the number of the testing ones, at the expense of larger computational times.

2. ONLINE CLASSIFICATION

First, a methodology for extracting Riemannian features from multilayer network time series is provided. As a case study, the Grassmann manifold (Grassmannian) is considered. Notwithstanding, the

online classification module of Section 2.2 can work in any Riemannian manifold.

2.1. Extracting Grassmannian features

Consider a multilayer network $\mathcal{G} := (\mathcal{N}, \mathcal{E}, \mathcal{L})$, with nodes $\mathcal{N}$ of cardinality $|\mathcal{N}|$, edges $\mathcal{E}$, and layers $\mathcal{L}$ of cardinality $|\mathcal{L}|$. Node $\nu \in \mathcal{N}$ in layer $l \in \mathcal{L}$ is annotated with the time-series $(y_\nu^{(l)}[t])_{t \in \mathbb{Z}_{>0}}$, where $\mathbb{Z}_{>0}$ denotes the set of all positive integer numbers. For a subset $\mathcal{V}$ of nodes and an integer $q \in \mathbb{Z}_{>0}$, the $q \times 1$ vector $\mathbf{y}_\mathcal{V}^{(l)}[t]$ is used in this manuscript to collect all signal samples from nodes $\mathcal{V}$ in the l th layer of the network at time t . To capture any latent non-linear dependencies in the network, a user-defined reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ is considered [30], with its associated reproducing kernel function $\kappa(\cdot, \cdot)$ and mapping function $\varphi : \mathbb{R}^q \rightarrow \mathcal{H} : \mathbf{y} \mapsto \varphi(\mathbf{y}) := \kappa(\mathbf{y}, \cdot)$. For example, an $\mathcal{H}$ with a Gaussian kernel $\kappa(\cdot, \cdot)$ is an infinite dimensional RKHS [30].

In the case of state classification, where a “snapshot” of all nodes of the l th layer is needed, $\mathcal{V} := \mathcal{N}$, $q := |\mathcal{N}|$, and $\mathbf{y}_\mathcal{N}^{(l)}[t] := [y_1^{(l)}[t], \dots, y_{|\mathcal{N}|}^{(l)}[t]]^\top$, where $\top$ denotes vector/matrix transposition. For a parameter $T_w \in \mathbb{Z}_{>0}$, define $\boldsymbol{\varphi}_t^{(l)} := [\varphi(\mathbf{y}_\mathcal{N}^{(l)}[t]), \varphi(\mathbf{y}_\mathcal{N}^{(l)}[t + 1]), \dots, \varphi(\mathbf{y}_\mathcal{N}^{(l)}[t + T_w - 1])]^\top \in \mathcal{H}^{T_w}$. Moreover, since “state” refers to a global connectivity pattern, to capture even inter-layer connections, let $\boldsymbol{\varphi}_t := [\boldsymbol{\varphi}_t^{(1)}, \dots, \boldsymbol{\varphi}_t^{(L)}]^\top \in \mathcal{H}^{T_w|\mathcal{L}|}$ gather information from all layers. Then, following the kernel-autoregressive-moving-average (kARMA) model of [31], it is assumed that there exist (unknown) matrices $\mathbf{C} \in \mathbb{R}^{T_w|\mathcal{L}| \times \rho}$, $\mathbf{A} \in \mathbb{R}^{\rho \times \rho}$ for some user-defined parameter $\rho \in \mathbb{Z}_{>0}$, and latent $\boldsymbol{\psi}_t \in \mathcal{H}^\rho$, $\mathbf{v}_t \in \mathcal{H}^{T_w|\mathcal{L}|}$, and $\boldsymbol{\omega}_t \in \mathcal{H}^\rho$, which model noise and approximation errors, such that $\forall t$, $\boldsymbol{\varphi}_t = \mathbf{C}\boldsymbol{\psi}_t + \mathbf{v}_t$ and $\boldsymbol{\psi}_t = \mathbf{A}\boldsymbol{\psi}_{t-1} + \boldsymbol{\omega}_t$. For a user-defined parameter $m \in \mathbb{Z}_{>0}$, the so-called *observability matrix* [32] is defined as: $\mathbf{O} := [\mathbf{C}^\top, (\mathbf{C}\mathbf{A})^\top, \dots, (\mathbf{C}\mathbf{A}^{m-1})^\top]^\top \in \mathbb{R}^{mT_w|\mathcal{L}| \times \rho}$. An estimate $\hat{\mathbf{O}}_t$ of $\mathbf{O}$ per time instance t is computed here by the same sequential way employed in the clustering frameworks of [20, 31]. Due to space limitations, the description of such a procedure is omitted.

In community classification, where features per nodal time series need to be extracted, $\mathcal{V} := \nu$, for $\nu \in \mathcal{N}$, and for a time-window length $\tau_w \in \mathbb{Z}_{>0}$, with $q := \tau_w$, let $\mathbf{y}_\nu^{(l)}[t] := [y_\nu^{(l)}[t], \dots, y_\nu^{(l)}[t + \tau_w - 1]]^\top$. With $\boldsymbol{\varphi}_\nu^{(l)}[t] := [\varphi(\mathbf{y}_\nu^{(l)}[t]), \dots, \varphi(\mathbf{y}_\nu^{(l)}[t + T_w - 1])]^\top \in \mathcal{H}^{T_w}$, the kARMA model of [20, 31] adapts to this case as $\boldsymbol{\varphi}_\nu^{(l)}[t] = \mathbf{C}_\nu^{(l)}\boldsymbol{\psi}_\nu^{(l)}[t] + \mathbf{v}_\nu^{(l)}[t]$ and $\boldsymbol{\psi}_\nu^{(l)}[t] = \mathbf{A}^{(l)}\boldsymbol{\psi}_\nu^{(l)}[t - 1] + \boldsymbol{\omega}_\nu^{(l)}[t]$, where $\mathbf{C}_\nu^{(l)} \in \mathbb{R}^{T_w \times \rho}$, $\mathbf{A}^{(l)} \in \mathbb{R}^{\rho \times \rho}$, with the latent $\boldsymbol{\psi}_\nu^{(l)}[t] \in \mathcal{H}^\rho$, $\mathbf{v}_\nu^{(l)}[t] \in \mathcal{H}^{T_w}$, and $\boldsymbol{\omega}_\nu^{(l)}[t] \in \mathcal{H}^\rho$. The observability matrix in this case takes the form $\mathbf{O}_\nu^{(l)} := [\mathbf{C}_\nu^{(l)\top}, (\mathbf{C}_\nu^{(l)}\mathbf{A}_\nu^{(l)})^\top, \dots, (\mathbf{C}_\nu^{(l)}(\mathbf{A}_\nu^{(l)})^{m-1})^\top]^\top \in \mathbb{R}^{mT_w \times \rho}$ per node ν and layer l . As in state classification, the way by which estimates of an observability matrix are obtained sequentially in [20, 31] is also used here to compute an estimate $\hat{\mathbf{O}}_\nu^{(l)}[t]$ of $\mathbf{O}_\nu^{(l)}$ per time instance t ; details are omitted due to space limitations.

The extracted feature per time instance t , to be used in Section 2.2, is defined as the linear subspace, denoted hereafter by x_i , spanned by the columns of the estimated observability matrix: $\text{span}\{\hat{\mathbf{O}}_t\}$ in state classification and $\text{span}\{\hat{\mathbf{O}}_\nu^{(l)}[t]\}$ in community classification. Index i is used instead of t for features to avoid any confusion with the time indices of training and testing data. Feature x_i becomes a point in the Grassmannian $\text{Gr}(\rho, D)$, defined as the collections of all linear subspaces of $\mathbb{R}^D$ with rank equal to ρ , where dimension D depends on the learning task at hand: $D = mT_w|\mathcal{L}|$ in state classification, while $D = mT_w$ in community

Figure 1: (a) The Riemannian multi-manifold modeling (RMMM) hypothesis [33, 34]. Landmark points with known labels (red and green points) describe the geometry of each submanifolds/classes. (b) The linear subspace $\hat{S}_k^{(i)}$ serves as an estimate of the “image” of class $\mathcal{C}_k$ in the Euclidean tangent space $T_{x_k^{(i)}}\mathcal{M}$ of $\mathcal{M}$ at $x_k^{(i)}$. Angle $\theta_k^{(i)}$ is critical in the assignment of a class-label to x_i ; see Algorithm 1.

Algorithm 1: OnlineGCT

Input : Features $\mathcal{X} = \{x_i\}_{i \in \mathcal{I}} = \mathcal{X}_{\text{train}} \cup \mathcal{X}_{\text{test}}$.
Classes $\{\mathcal{C}_k\}_{k=1}^K$ so that $\mathcal{X}_{\text{train}} = \bigcup_{k=1}^K \mathcal{C}_k$.

Output : Assign class labels $\{\mathcal{C}_k\}_{k=1}^K$ to points $\mathcal{X}_{\text{test}}$.

- 1 Select landmark points $\mathcal{L}_k$ from class $\mathcal{C}_k \subset \mathcal{X}_{\text{train}}$, $\forall k$ [20].
- 2 **for all points** $x_i \in \mathcal{X}_{\text{test}}$ **do**
- 3 **for all** $k \in \{1, 2, \dots, K\}$ **do**
- 4 Identify the closest point $x_k^{(i)}$ to x_i from $\mathcal{L}_k$ and
compute the geodesic distance $d(x_k^{(i)}, x_i)$.
- 5 Define the K_{NN} -nearest-neighbors $\mathcal{N}_k^{(i)}$ of $x_k^{(i)}$
from points in $\mathcal{C}_k$.
- 6 Map $\mathcal{N}_k^{(i)}$ into the tangent space $T_{x_k^{(i)}}\mathcal{M}$ of
 $\mathcal{M}$ at $x_k^{(i)}$ via the manifold-logarithm mapping:
 $\mathbf{x}_{kj}^{(i)} := \log_{x_k^{(i)}}(x_j)$, $\forall x_j \in \mathcal{N}_k^{(i)}$. Let also
 $\mathbf{x}_{kk}^{(i)} := \log_{x_k^{(i)}}(x_k^{(i)})$ and $\mathbf{x}_{ki}^{(i)} := \log_{x_k^{(i)}}(x_i)$.
- 7 Compute the sample covariance matrix $\hat{\mathbf{C}}_k^{(i)}$ in (1)
and its eigenspace $\hat{S}_k^{(i)}$ by principal component
analysis (PCA).
- 8 Compute angle $\theta_k^{(i)}$ between $\mathbf{x}_{ki}^{(i)} - \mathbf{x}_{kk}^{(i)}$ and $\hat{S}_k^{(i)}$;
cf. Figure 1b.
- 9 Define vector $\mathbf{g}_k^{(i)} := [d(x_k^{(i)}, x_i), \theta_k^{(i)}]^\top$.
- 10 Identify $k_* := \arg \min_{k \in \{1, \dots, K\}} \|\mathbf{g}_k^{(i)}\|_2$ and assign
the class label $\mathcal{C}_{k_*}$ to x_i .

classification. The Grassmannian $\text{Gr}(\rho, D)$ is a well-known Riemannian manifold with dimension equal to $\rho(D - \rho)$ [21, 22].

2.2. Online classification in Riemannian manifolds

The following algorithmic framework is based on the Riemannian multi-manifold modeling (RMMM) hypothesis [33, 34]: Features $\{x_i\}_{i \in \mathcal{I}}$ are located into or close to the union of a finite number of submanifolds $\{\mathcal{C}_k\}_{k=1}^K$ (classes in the current classification setting) in a Riemannian manifold $\mathcal{M}$; $\mathcal{M} := \text{Gr}(\rho, D)$ in Section 2.1. Submanifolds/classes $\{\mathcal{C}_k\}_{k=1}^K$ are allowed to intersect and have different dimensions.

Training data $\mathcal{X}_{\text{train}}$ and testing data $\mathcal{X}_{\text{test}}$ comprise the available kernel-ARMA features/points from Section 2.1 $\mathcal{X} = \{x_i\}_{i \in \mathcal{I}} = \mathcal{X}_{\text{train}} \cup \mathcal{X}_{\text{test}}$. Since the learning task at hand is supervised, the training set is partitioned in known classes $\{\mathcal{C}_k\}_{k=1}^K$ as $\mathcal{X}_{\text{train}} = \bigcup_{k=1}^K \mathcal{C}_k$. To obtain a concise description of the potentially massive in number $\mathcal{X}_{\text{train}}$, landmark points $\mathcal{L}_k$ are selected from each class $\mathcal{C}_k$. Although selecting points randomly from $\mathcal{C}_k$ is attractive from a computational

Table 1: Synthetic data: State-classification results

Methods	10% train 90% test			20% train 80% test			40% train 60% test		
	D1	D2	Time (sec)	D1	D2	Time (sec)	D1	D2	Time (sec)
onlineGCT[kARMA]	1	0.742	177	1	0.785	159	1	0.843	131
GoogLeNet[Corr] [23]	0.927	0.638	38	0.971	0.694	33	0.990	0.723	26
GoogLeNet[WVT] [23]	0.937	0.657	40	0.976	0.676	35	1	0.755	29
GoogLeNet[kARMA] [23]	0.975	0.721	32	0.984	0.766	28	1	0.789	25
DenseNet[Corr] [24]	0.933	0.664	45	0.981	0.663	41	0.994	0.718	33
DenseNet[WVT] [24]	0.945	0.683	48	0.987	0.694	44	1	0.776	36
DenseNet[kARMA] [24]	0.983	0.692	39	0.991	0.743	35	1	0.816	28
ResNet[Corr] [25]	0.930	0.652	41	0.979	0.675	38	0.988	0.737	29
ResNet[WVT] [25]	0.948	0.598	43	0.962	0.680	40	1	0.763	33
ResNet[kARMA] [25]	0.971	0.708	37	0.989	0.759	32	1	0.805	27
VGG-16[Corr] [26]	0.965	0.683	27	0.984	0.718	23	1	0.763	18
VGG-16[WVT] [26]	0.972	0.670	29	1	0.735	24	1	0.780	20
VGG-16[kARMA] [26]	0.989	0.730	21	1	0.772	18	1	0.808	15
VGG-19[Corr] [26]	0.969	0.672	29	0.977	0.713	25	1	0.757	19
VGG-19[WVT] [26]	0.978	0.680	32	1	0.741	27	1	0.788	21
VGG-19[kARMA] [26]	0.992	0.723	24	1	0.768	20	1	0.801	16

Table 2: Synthetic data: Community-classification results

Methods	10% train 90% test			20% train 80% test			40% train 60% test		
	D1	D2	Time (sec)	D1	D2	Time (sec)	D1	D2	Time (sec)
onlineGCT[kARMA]	1	0.849	409	1	0.904	357	1	0.987	271
GoogLeNet[Corr] [23]	0.939	0.726	65	0.955	0.849	57	1	0.921	44
GoogLeNet[WVT] [23]	0.950	0.740	62	0.959	0.864	54	1	0.935	42
GoogLeNet[GAF] [23]	0.966	0.783	72	0.972	0.870	66	1	0.945	51
GoogLeNet[kARMA] [23]	0.989	0.794	57	0.991	0.895	51	1	0.952	38
DenseNet[Corr] [24]	0.916	0.695	77	0.952	0.827	68	1	0.928	53
DenseNet[WVT] [24]	0.942	0.724	73	0.956	0.858	65	1	0.945	50
DenseNet[GAF] [24]	0.951	0.746	82	0.960	0.871	74	1	0.959	57
DenseNet[kARMA] [24]	0.960	0.768	70	0.975	0.887	62	1	0.977	49
ResNet[Corr] [25]	0.924	0.707	70	0.960	0.839	62	1	0.930	47
ResNet[WVT] [25]	0.935	0.736	67	0.964	0.860	60	1	0.942	44
ResNet[GAF] [25]	0.956	0.752	76	0.966	0.875	68	1	0.950	53
ResNet[kARMA] [25]	0.974	0.785	65	0.981	0.890	58	1	0.964	43
VGG-16[Corr] [26]	0.922	0.739	52	0.959	0.862	46	1	0.939	35
VGG-16[WVT] [26]	0.926	0.745	50	0.964	0.869	44	1	0.944	34
VGG-16[GAF] [26]	0.951	0.782	59	0.980	0.892	51	1	0.961	39
VGG-16[kARMA] [26]	0.978	0.816	44	0.987	0.893	42	1	0.958	32
VGG-19[Corr] [26]	0.928	0.725	55	0.961	0.864	48	1	0.946	37
VGG-19[WVT] [26]	0.931	0.740	52	0.968	0.872	47	1	0.949	36
VGG-19[GAF] [26]	0.946	0.780	62	0.982	0.901	54	1	0.962	41
VGG-19[kARMA] [26]	0.973	0.801	53	0.991	0.887	47	1	0.967	36

aspect, the non-random way of [20], via a Riemannian modification of the Maxmin algorithm [35], is adopted also here to collect geometric information that could be neglected by random sampling techniques. More details and comparisons between the non-random approach and random sampling techniques can be found in [20].

The testing data $\mathcal{X}_{\text{test}}$ are classified sequentially according to Algorithm 1. Each testing datum is visited sequentially/online in Line 2. For each testing datum x_i , Line 4 identifies its closest, in a geodesic-distance sense, landmark point $x_k^{(i)} \in \mathcal{L}_k$. A user-defined number K_{NN} of the nearest neighbors of $x_k^{(i)}$ from the landmark points $\mathcal{L}_k$ are mapped to the tangent space [21, 22] of the Riemannian manifold at point $x_k^{(i)}$ via the manifold logarithmic map [22] in Lines 5 and 6; see also Figure 1b. Ways to compute the manifold logarithmic map for a couple of well-known Riemannian manifolds, such as the Grassmannian one, are detailed in [20, 31]. The computation of the manifold logarithmic map seems to be the most computationally intensive task in Algorithm 1, and examples of such complexities can be found in [20, 31]. Since the tangent space is a Euclidean space, the sample covariance matrix

$$\hat{\mathbf{C}}_k^{(i)} := \frac{1}{|\mathcal{N}_k^{(i)}|-1} \sum_{x_j \in \mathcal{N}_k^{(i)}} (\mathbf{x}_{kj}^{(i)} - \bar{\mathbf{x}}_k^{(i)}) (\mathbf{x}_{kj}^{(i)} - \bar{\mathbf{x}}_k^{(i)})^\top, \quad (1)$$

can be also computed in Line 7, where the sample average is defined as $\bar{\mathbf{x}}_k^{(i)} := (1/|\mathcal{N}_k^{(i)}|) \sum_{x_j \in \mathcal{N}_k^{(i)}} \mathbf{x}_{kj}^{(i)}$. The principal eigenspace $\hat{\mathcal{S}}_k^{(i)}$ serves as an estimation of the “image” of the submanifold $\mathcal{C}_k$ in $T_{x_k^{(i)}} \mathcal{M}$ (see Figure 1b). The angle $\theta_k^{(i)}$ between the vector $\mathbf{x}_{ki}^{(i)} - \bar{\mathbf{x}}_k^{(i)}$ and $\hat{\mathcal{S}}_k^{(i)}$ is computed in Line 8. The assignment of a class-label to the testing feature x_i under query is provided in Line 10 and is based on a least-squares criterion on the vector comprising the computed distance and angle information.

3. NUMERICAL TESTS

Algorithm 1 is validated on synthetic and real data against the state-of-the-art DL models GoogLeNet [23], DenseNet [24], ResNet [25], VGG-16 and VGG-19 [26]. Tags kARMA, Corr, WVT and GAF indicate the following feature-extraction methods for the learning tasks at hand: The proposed kernel-ARMA modeling of Section 2.1, correlations, wavelet transform [36], and Gramian angular field [37], respectively. Since GAF can only extract statistical features from nodal time series, it is used only in community classification. All aforementioned feature-extraction methods generate matrices as features, which can be thus viewed as “grayscale-image” input data by [23–26], widely used in bio-medical image classification [27, 28].

Table 3: EEG data: State-classification results

Methods	10% train 90% test		20% train 80% test		40% train 60% test	
	Accuracy	Time (sec)	Accuracy	Time (sec)	Accuracy	Time (sec)
onlineGCT[kARMA]	0.962	2675	0.977	2324	1	1857
GoogLeNet[Corr] [23]	0.864	315	0.897	289	0.968	223
GoogLeNet[WVT] [23]	0.895	324	0.929	296	1	230
GoogLeNet[kARMA] [23]	0.932	292	0.954	261	1	204
DenseNet[Corr] [24]	0.845	330	0.912	307	0.980	245
DenseNet[WVT] [24]	0.882	349	0.938	315	1	267
DenseNet[kARMA] [24]	0.918	316	0.955	280	1	226
ResNet[Corr] [25]	0.871	321	0.904	299	0.984	237
ResNet[WVT] [25]	0.906	325	0.943	303	1	238
ResNet[kARMA] [25]	0.941	306	0.959	275	1	216
VGG-16[Corr] [26]	0.842	274	0.878	240	0.959	185
VGG-16[WVT] [26]	0.863	290	0.897	256	1	192
VGG-16[kARMA] [26]	0.923	259	0.945	228	1	171
VGG-19[Corr] [26]	0.846	281	0.890	250	0.967	189
VGG-19[WVT] [26]	0.869	297	0.905	262	1	197
VGG-19[kARMA] [26]	0.917	264	0.941	234	1	175

“Classification accuracy” in the following tables and figures is defined as the average (over multiple independent tests) of the ratios of the number of correctly classified features over the total number of features in the testing dataset. Average computational times are provided in seconds.

3.1. Synthetic data

Synthetic fMRI data were generated by the open-source Matlab SimTB toolbox [38]. A multilayer network with 2 layers and 10 nodes, transitioning between 4 distinct network states per layer, is considered. Each state corresponds to a connectivity matrix which is modeled as the superposition of three matrices: (i) the ground-truth (noiseless) connectivity matrix, where each entry takes the value of 0 (no connection) or 1 (strongest possible connection), and which generates the “clean” training data via the SimTB toolbox [38]; (ii) a symmetric “noise” matrix whose entries are drawn independently from a zero-mean Gaussian distribution with standard deviation σ ; and (iii) a symmetric matrix where a number μ of non-zero entries with values equal to 0.5 serve as outliers. All entries of the superimposed “contaminated” connectivity matrix are thresholded to fall within the range of $[0, 1]$. Driven by the previous connectivity matrices, two datasets (training and testing) are created by the SimTB toolbox [38]. For dataset 1, $(\mu, \sigma) = (5, 10^{-1})$, while for dataset 2, $(\mu, \sigma) = (30, 10^{-0.5})$. Parameters σ and μ stay fixed for all states. Each state contributes 300 signal samples, for a total of $4 \times 300 = 1,200$ samples per node. In community classification, three communities are defined via the ground-truth connectivity matrix, where each community included nodes from different layers (cross-layer communities). For a fair comparison, the number of features used by each method, both for training and testing, are the same. There are 400×2 (layers) = 800 features for state classification and 250×10 (nodes) $\times$ 2 (layers) = 5,000 features for community classification for each dataset. Parameters are tuned to achieve best classification accuracies for every method. In onlineGCT, the number of landmark points is equal to the number of features in training set.

Tables 1 and 2 show that onlineGCT[kARMA] achieves the highest classification accuracies on all datasets, while DL methods score the least computational time. As expected, the larger the number of training data, the larger the classification accuracies for all competing schemes. DenseNet [24] achieves the largest improvement among all DL models when increasing the number of training data, since it employs the largest number of layers in its neural-network architecture.

3.2. Real data

The open-source EEG data [39] were used. Data comprise neonatal EEG recordings, annotated for seizure and non-seizure states (=

Figure 2: Classification accuracies on the synthetic dataset 2. All learning methods use kARMA features with different numbers of training data.

Figure 3: State classification accuracies on the real EEG data. All learning methods use kARMA features with different numbers of training data.

classes) by experts. Signals were recorded with an EEG amplifier (sampling frequency of 256Hz) and EEG caps (sintered Ag/AgCl electrodes) with 19 electrodes positioned, including a recording reference at midline. Each neonate defines a layer in the present context of multilayer networks, and 10 neonates/layers are considered. The length of time series is 10,240 samples, with 18 nodes per layer. The number of features extracted is $2,000 \times 10$ (layers) = 20,000 for all methods. In onlineGCT, the number of landmark points is set equal to 80% of the number of features in the training data. Due to the sequential mode of operation, there are cases where the extracted features are computed by samples from both seizure and non-seizure states. Features extracted from those cases are labeled as state #3. There are no ground-truth communities defined for this dataset, and thus, only the state-classification task is considered here. Classification accuracies are shown in Table 3, where onlineGCT scores the highest accuracies at the expense of increased computational times. All methods get 100% classification accuracies when 40% or more of the total number of features are used as training data.

4. CONCLUSIONS

Aiming at a classification framework that achieves high classification accuracy results while operating sequentially on data, and motivated by the very recently sequential clustering scheme of [20], this work introduced an online classification framework for multilayer network time series by means of Riemannian manifolds, coined online geodesic clustering by tangent spaces (onlineGCT). The paper presented also the case study of extracting features in the Grassmann manifold (Grassmannian) by fitting kernel-ARMA models to the network nodal time series. Extensive numerical tests that validate the onlineGCT against state-of-the-art deep-learning (DL) models were also reported. According to the numerical tests, onlineGCT consistently scored better classification accuracies than DL models, especially in cases where the number of training data is small with respect to the number of testing data, at the expense of larger computational times. Interestingly, in all classification tasks, all DL models equipped with the proposed Grassmannian features of Section 2.1 performed better than when fed with features provided by any of the rest of the considered feature-extraction schemes.

REFERENCES

- [1] E. D. Kolaczyk, *Statistical Analysis of Network Data: Methods and Models*. New York: Springer, 2009.
- [2] D. S. Bassett and O. Sporns, "Network neuroscience," *Nature Neuroscience*, vol. 20, no. 3, p. 353, 2017.
- [3] S.-Y. Huang and H. Chen, "Exploring the online underground marketplaces through topic-based social network and clustering," in *Proc. IEEE Conference on Intelligence and Security Informatics (ISI)*, IEEE, 2016, pp. 145–150.
- [4] A. J. Wixted, P. Kinnaird, H. Larijani, *et al.*, "Evaluation of LoRa and LoRaWAN for wireless sensor networks," in *2016 IEEE SENSORS*, IEEE, 2016, pp. 1–3.
- [5] S. Ogawa, T.-M. Lee, A. R. Kay, and D. W. Tank, "Brain magnetic resonance imaging with contrast dependent on blood oxygenation," *Proc. National Academy of Sciences*, vol. 87, no. 24, pp. 9868–9872, 1990.
- [6] M. Kivelä, A. Arenas, M. Barthelemy, *et al.*, "Multilayer networks," *J. Complex Networks*, vol. 2, no. 3, pp. 203–271, 2014.
- [7] R. F. Betzel and D. S. Bassett, "Multi-scale brain networks," *NeuroImage*, vol. 160, pp. 73–83, 2017.
- [8] P.-Y. Chen and A. O. Hero, "Multilayer spectral graph clustering via convex layer aggregation: Theory and algorithms," *IEEE Trans. Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 553–567, 2017.
- [9] A. Vörös and T. A. Snijders, "Cluster analysis of multiplex networks: Defining composite network measures," *Social Networks*, vol. 49, pp. 93–112, 2017.
- [10] C. J. Stam, W. de Haan, A. Daffertshofer, *et al.*, "Graph theoretical analysis of magnetoencephalographic functional connectivity in Alzheimer's disease," *Brain*, vol. 132, no. 1, pp. 213–224, Oct. 2008.
- [11] A. Khorasani and M. R. Daliri, "HMM for classification of Parkinson's disease based on the raw gait data," *Journal of Medical Systems*, vol. 38, no. 12, p. 147, 2014.
- [12] S. Röblitz and M. Weber, "Fuzzy spectral clustering by PCCA+: Application to Markov state models and data classification," *Advances in Data Analysis and Classification*, vol. 7, no. 2, pp. 147–179, 2013.
- [13] R. Nema and A. Pandey, "Community kernels detection in OSN using SVM clustering and classification," *International Journal of Computer Applications*, vol. 113, no. 11, 2015.
- [14] S. Wang, P. Dehghanian, and Y. Gu, "A novel multi-resolution wavelet transform for online power grid waveform classification," in *2019 International Conference on Smart Grid Synchronized Measurements and Analytics (SGSMA)*, IEEE, 2019, pp. 1–8.
- [15] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.
- [16] P. Molchanov, X. Yang, S. Gupta, *et al.*, "Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network," in *IEEE CVPR*, 2016, pp. 4207–4215.
- [17] C. Pelletier, G. I. Webb, and F. Petitjean, "Temporal convolutional neural network for the classification of satellite image time series," *Remote Sensing*, vol. 11, no. 5, p. 523, 2019.
- [18] S. Chambon, M. N. Galtier, P. J. Arnal, G. Wainrib, and A. Gramfort, "A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 4, pp. 758–769, 2018.
- [19] R. Wan, S. Mei, J. Wang, M. Liu, and F. Yang, "Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting," *Electronics*, vol. 8, no. 8, p. 876, 2019.
- [20] C. Ye, K. Slavakis, J. Nakuci, S. F. Muldoon, and J. Medaglia, "Fast sequential clustering in Riemannian manifolds for dynamic and time-series-annotated multilayer networks," *IEEE Open Journal of Signal Processing*, 2021, to appear. DOI: 10.1109/OJSP.2021.3051453.
- [21] L. W. Tu, *An Introduction to Manifolds*. Springer, 2008.
- [22] J. W. Robbin and D. A. Salamon, *Introduction to Differential Geometry*. 2019. [Online]. Available: <https://people.math.ethz.ch/~salamon/PREPRINTS/diffgeo.pdf>.
- [23] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *IEEE CVPR*, 2015, pp. 1–9.
- [24] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE CVPR*, 2017, pp. 4700–4708.
- [25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016, pp. 770–778.
- [26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015*, 2015.
- [27] J.-H. Kim, S.-Y. Seo, C.-G. Song, and K.-S. Kim, "Assessment of electrocardiogram rhythms by GoogLeNet deep neural network architecture," *Journal of Healthcare Engineering*, vol. 2019, 2019.
- [28] X. Chen, B. M. Williams, S. R. Vallabhaneni, *et al.*, "Learning active contour models for medical image segmentation," in *IEEE CVPR*, 2019, pp. 11 632–11 640.
- [29] L. Nannia, S. Ghidoni, and S. Brahmam, "Ensemble of convolutional neural networks for bioimage classification," *Applied Computing and Informatics*, 2020.
- [30] B. Scholkopf and A. J. Smola, *Learning with Kernels*. MIT Press, 2001.
- [31] C. Ye, K. Slavakis, P. V. Patil, *et al.*, "Network clustering via kernel-ARMA modeling and the Grassmannian: The brain-network case," *Signal Processing*, 2021, to appear. DOI: 10.1016/j.sigpro.2020.107834.
- [32] L. Ljung, "System identification," in *Signal Analysis and Prediction*, Springer, 1998, pp. 163–173.
- [33] X. Wang, K. Slavakis, and G. Lerman, "Riemannian multi-manifold modeling," *arXiv:1410.0095*, 2014.
- [34] —, "Multi-manifold modeling in non-Euclidean spaces," in *Proc. AISTATS*, San Diego: California: USA, May 2015.
- [35] V. De Silva and J. B. Tenenbaum, "Sparse multidimensional scaling using landmark points," Technical report, Stanford University, Tech. Rep., 2004.
- [36] P. Chaovalit, A. Gangopadhyay, G. Karabatis, and Z. Chen, "Discrete wavelet transform-based time series analysis and mining," *ACM Computing Surveys (CSUR)*, vol. 43, no. 2, pp. 1–37, 2011.
- [37] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, vol. 1, 2015.
- [38] E. A. Allen, E. Damaraju, S. M. Plis, *et al.*, "Tracking whole-brain connectivity dynamics in the resting state," *Cerebral Cortex*, vol. 24, no. 3, pp. 663–676, 2014.
- [39] N. Stevenson, K. Tapani, L. Lauronen, and S. Vanhatalo, "A dataset of neonatal EEG recordings with seizure annotations," *Scientific Data*, vol. 6, p. 190 039, 2019.