Fast Sequential Clustering in Riemannian Manifolds for Dynamic and Time-Series-Annotated Multilayer Networks

Cong Ye,[1] Konstantinos Slavakis,[1] *Senior Member, IEEE,*
Johan Nakuci,[2] Sarah F. Muldoon,[3] and John Medaglia,[4]

This work exploits Riemannian manifolds to build a sequential-clustering framework able to address a wide variety of clustering tasks in dynamic multilayer (brain) networks via the information extracted from their nodal time-series. The discussion follows a bottom-up path, starting from feature extraction from time-series and reaching up to Riemannian manifolds (feature spaces) to address clustering tasks such as state clustering, community detection (a.k.a. network-topology identification), and subnetwork-sequence tracking. Kernel autoregressive-moving-average modeling and kernel (partial) correlations serve as case studies of generating features in the Riemannian manifolds of Grassmann and positive-(semi)definite matrices, respectively. Feature point-clouds form clusters which are viewed as submanifolds according to Riemannian multi-manifold modeling. A novel sequential-clustering scheme of Riemannian features is also established: feature points are first sampled in a *non-random* way to reveal the underlying geometric information, and, then, a fast sequential-clustering scheme is brought forth that takes advantage of Riemannian distances and the angular information of tangent spaces. By virtue of the landmark points and the sequential processing of the Riemannian features, the computational complexity of the framework is rendered free from the length of the available time-series data. The effectiveness and computational efficiency of the proposed framework are validated by extensive numerical tests against several state-of-the-art manifold-learning and brain-network-clustering schemes on synthetic as well as real functional-magnetic-resonance-imaging (fMRI) and electro-encephalogram (EEG) data.

***Index Terms*—Network, brain, time-series, dynamic, multi-layer, clustering, sequential, Riemannian manifold.**

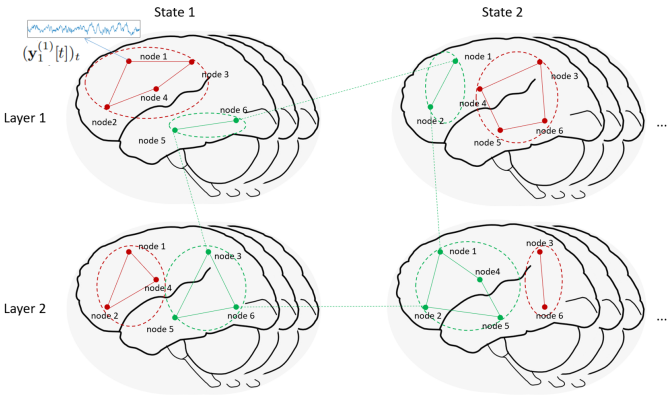


Fig. 1: An example of a brain network with two layers and two states per layer, with the time axis considered to be perpendicular to the figure. Snapshots (observations) of the dynamic network are taken at a number of discrete time instances $t \in \mathbb{Z}$ ($\mathbb{Z}$: the set of all integer numbers). All those collected snapshots are denoted in this figure by the schematic "brains" stacked on top of one another. The real-valued time-series, which annotates node ν in layer l, is denoted by $(y_\nu^{(l)}[t])_{t\in\mathbb{Z}}$. A state corresponds to a specific network topology or nodal connectivity pattern which stays fixed over multiple observations or time instances; an example of a state is the resting state of the brain [10]. A layer corresponds to a "dimension" of connectivity; for example, a layer may represent a subject in a clinical study, or, a frequency band in which nodal time-series are observed. In this figure, each state comprises two "communities" (green and red), and green-colored nodes perform a common task. Such a collaboration can be modeled via the "subnetwork sequence" $\{\{5,6\},\{1,2\},\{1,2,4,5\},\{3,5,6\}\}$, where the integers indicate the node indices that take part in the common effort across all states and layers. Appropriate features, extracted from the time-series of those nodes, are expected to form a cluster in the feature space. The identification of that cluster will reveal the subnetwork sequence.

I. Introduction

A. Problem Statement

Data analytics [1] permeate nowadays social, wireless, power-grid, and brain networks. In the rapidly growing field of network neuroscience, learning from data plays a critical role in revealing the topological and functional structure of the brain [2, 3]. Nodes in a brain network may be neurons (microscopic scale), or even regions (macroscopic scale), while edges may represent anatomical connections such as synapses (structural connectivity at the microscopic scale), or, statistical relationships between regional brain dynamics (functional connectivity at the macroscopic scale). Nodes are usually annotated with time-series, and popular noninvasive modalities that acquire those time-series data are functional magnetic resonance imaging (fMRI), which monitors blood oxygen-level dependent (BOLD) data [4], and electro-encephalography (EEG), where data are collected by electrodes placed on the scalp of a subject. Recent generalizations move towards the direction of multilayer (a.k.a. multiview [5] or multiplex [6]) networks [7–9], where layers are used to model several modes or dimensions of connectivity among network nodes; e.g., a layer may correspond to a subject in a clinical study, or, a frequency band in which nodal time-series are observed (see Fig. 1).

Network clustering is a rich data-analytic paradigm that includes the following learning tasks (see Fig. 1).

Learning task 1: State clustering, where the goal is to identify states of the brain, with "state" referring to a "global" connectivity pattern among network nodes that stays fixed over a time interval. Clustering states in brain networks

[1]C. Ye (contact author; e-mail: congye@buffalo.edu) and K. Slavakis are with the Department of Electrical Engineering, University at Buffalo (UB), The State University of New York (SUNY), NY 14260, USA.
[2]Johan Nakuci is with the Neuroscience Program, UB, SUNY, USA.
[3]S. F. Muldoon is with the Department of Mathematics and the Computational and Data-Enabled Science and Engineering Program, UB, SUNY, USA.
[4]J. Medaglia is with the Department of Psychology, Drexel University, PA 19104, USA, and the Perelman School of Medicine, University of Pennsylvania, PA 19104, USA.

has enhanced understanding of brain disorders such as the Alzheimer disease and autism [11], depression [12], anxiety, epilepsy and schizophrenia [13]. It is worth stressing here that the border between the concepts of "state" and "layer" is blurry; viewing a state, or, observations over a time-window as a layer, for example, often facilitates learning tasks, e.g., [9, 14]. Nevertheless, a subtle distinction is that a layer is usually well defined and known to the user a-priori, e.g., subjects in a clinical study or frequency bands, whereas states may not be provided beforehand and may need to be learned from the brain-network time-series; an epileptic seizure state, for example, may need to be separated from non-seizure ones through EEG time-series observations [15]. The following discussion is developed having in mind those delicate distinctions between states and layers.

Learning task 2: Community detection (a.k.a. network-topology identification), which aims at identifying communities/subgroups of nodes within a single state/layer, with rich applications in network neuroscience, e.g., [16–18].

Learning task 3: Subnetwork-sequence clustering, where a "subnetwork sequence" is defined as a sequence of subgroups of network nodes which emerge from different states/layers, with nodes which may change as the network transitions from one state/layer to another, and with nodal time-series that share and preserve *common* characteristics along the whole evolution path of the sequence of subnetworks; e.g., the green-colored subnetwork sequence of Fig. 1. Subnetwork-sequence clustering aims at a moment-by-moment tracking of the cognitive contents in the brain, and traces the ability of the brain network to stimulate collaboration among its nodes to perform a common cognitive task, e.g., [19].

Following the machine-learning paradigm [20], features will be extracted from the nodal time-series to address the previous network-clustering tasks. Riemannian manifolds will serve as feature spaces, motivated by the fact that popular time-series statistics are strongly connected with Riemannian manifolds, e.g., correlations and low-rank subspaces can be mapped into the manifolds of (symmetric) positive (semi)definite matrices [21] and Grassmann [22] (a.k.a. Grassmannian: the set of all linear subspaces of a fixed rank [23, 24]), respectively. Although most of the Riemannian manifolds used in data analytics are considered to be embedded in Euclidean spaces, there are important cases where points of the Riemannian manifold cannot be represented by Euclidean vectors, e.g., the Grassmannian. Despite Nash's embedding theorem [25, 26], according to which Riemannian manifolds can be embedded in Euclidean spaces under certain conditions, such embeddings may unnecessarily increase the dimensionality of the target feature space and weigh down any computational effort to run sophisticated learning tasks on such features [27].

This work claims that the rich geometry of (not necessarily embedded in Euclidean spaces) Riemannian manifolds allows latent feature patterns to unfold to the benefit of all of the aforementioned network-clustering tasks. With nowadays applications facing large-scale networks and data, e.g., very long nodal time-series, it is of crucial importance in this data-deluge era [28] to develop a *computationally efficient clustering framework* that operates *sequentially* on data and features and carries through *all* of the network clustering tasks in *Riemannian manifolds*. Although the results of this work can be applied to any type of network, this manuscript solidifies arguments via brain networks.

B. Prior Art

Most of the existing studies address single-layer networks, operating in batch and not sequential mode, e.g., [10, 29–32]. Manifold-learning based schemes, not developed originally for network-time-series analysis, e.g., [33], can be tailored to address network-clustering tasks. To save space, a detailed review of those methods is deferred to [31]. Standard manifold-learning techniques are often used as off-the-shelf tools in network-neuroscience studies, e.g., [34, 35]. Works that dig deeper into the specific Riemannian geometry of positive-(semi)definite matrices to offer solutions for neuroscience, but do not consider the wide variety of clustering tasks that this study addresses, can be found in [21, 36].

The majority of batch-clustering frameworks for multilayer networks has been built on popular learning tools such as subspace learning [37, 38], fuzzy clustering [39], the wavelet transform [40], tensor decompositions [41, 42], multilayer modularity maximization [16], and graph signal processing [43]. Batch approaches for multilayer networks include also [9, 44, 45], with [45] being able to address both state clustering and community detection, but not subnetwork-sequence clustering since inter-layer information cannot be accommodated. Correlation matrices and hierarchical clustering were proposed in [14] to detect communities in multilayer brain networks. Works [16, 38, 40] explore inter-layer dependencies, but none of them considers the subnetwork-sequence clustering task. Study [32], as a predecessor of this work, can accommodate all of the aforementioned network clustering tasks, and can be extended straightforwardly to the multilayer network case, but it can only operate in a batch mode, and its computational complexity footprint is thus burdened by the number of available feature-points.

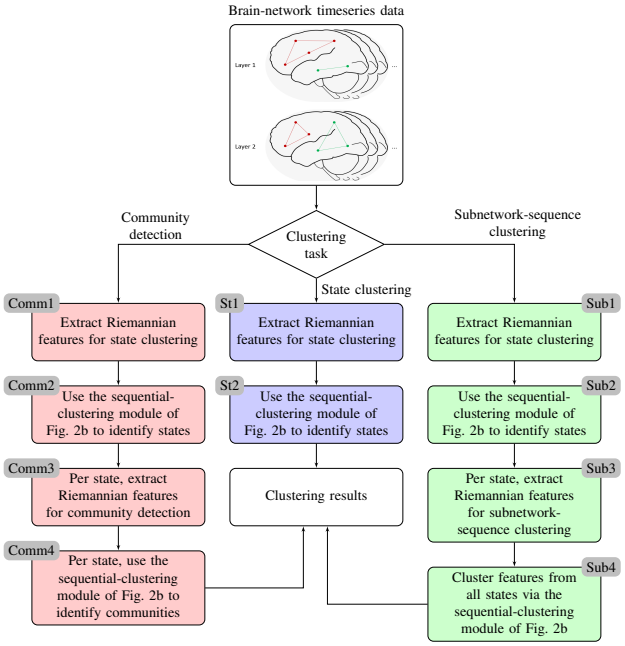
Sequential clustering algorithms, where feature extraction is also performed sequentially, have been reported in [46–48]. Work [47] uses the wavelet features proposed in [40] to introduce a sequential clustering framework for state clustering. A manifold-learning based sequential clustering scheme, not developed originally for network time-series, can be found in [49]. Stochastic block modeling for evolutionary clustering and tracking was proposed in [50, 51], while a clustering algorithm, based on support vector machines and with high computational complexity, was introduced in [52, 53]. Tensor-factorization approaches for streaming data were introduced in [17, 54, 55]. Nevertheless, none of the aforementioned sequential clustering algorithms considers the multilayer-network setting. Study [56] proposed a sequential clustering method for community detection in multilayer networks via modularity-function optimization. Method [56] can also perform subnetwork-sequence clustering by considering each state as a layer, under the assumption that state changes are reflected in a known and measurable property of nodal

time series data. Another sequential clustering algorithm for multilayer networks, based on local linear embedding and spectral clustering, was proposed in [57]. Study [57] cannot perform subnetwork-sequence clustering since it offers no mechanism to incorporate inter-layer dependencies. Work [58] builds on sparse subspace clustering and follows a similar path with [17, 57]: first, communities are detected, and then, based on the "change points" of the communities, states are clustered. Methods [17, 58] can accommodate inter-layer information, and they can be modified to perform also subnetwork-sequence clustering by considering each state as a layer, under the constraint that all states occupy the same time duration, which is dictated by their feature-extraction mechanisms. Since [58] utilizes sparse subspace clustering, it assumes that data lie into or close to a union of unknown affine (linear) Euclidean subspaces, and thus, it accommodates neither any Riemannian geometry, nor Riemannian features which may not be embedded in Euclidean spaces, e.g., the Grassmannian. Method [17] exploits tensor decompositions, but without any consideration of any Riemannian geometry. Among all of the cited frameworks, only [33, 49] can address *all* network-clustering tasks in a Riemannian manifold, after appropriate modifications, since [33, 49] were not developed originally for network time-series clustering. Numerical tests on [17, 33, 49] are reported in Sec. IV.

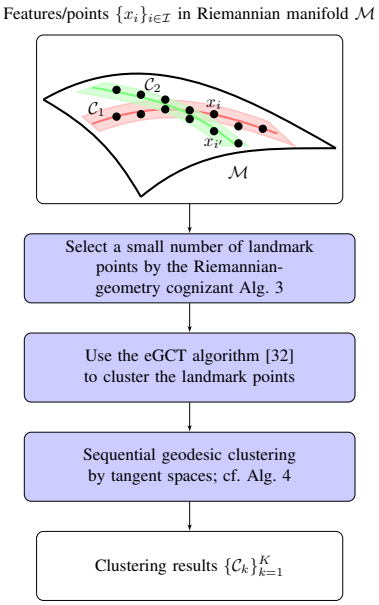
C. Contributions of this Manuscript

First, this work aims at filling the void in Riemannian methods for sequential clustering able to address *all* possible clustering tasks in a multilayer network: state clustering, community detection, and subnetwork-sequence identification/clustering. Following a bottom-up approach, features are first extracted sequentially from raw nodal time-series and then mapped to Riemannian manifolds [23, 24]. To help the reader, kernel autoregressive–moving-average (kARMA) modeling and kernel (partial) correlations are used as case studies to define features in the Riemannian manifolds of Grassmann and positive-(semi)definite matrices, respectively. Kernel functions are used to capture latent non-linear dependencies among the network's time-series. For convenience, a high-level description of the novel clustering framework is provided in Fig. 2a.

Second, rather than turning to popular manifold-learning techniques as off-the-shelf tools to develop the sequential-clustering module that drives the framework of Fig. 2a, this study builds also a novel sequential-clustering method for Riemannian manifolds. Under the hypothesis that data are placed close to or into a union of possibly intersecting submanifolds (Riemannian multi-manifold modeling (RMMM) [59, 60]), a small number of landmark feature points are chosen in a *non-random* way to sketch the geometry of the possibly massive point-cloud of features. After a sketch of the geometry has been obtained, the proposed sequential-clustering scheme considers *both* Riemannian distances and the angular information of tangent spaces [23, 24] to cluster the remaining non-landmark feature points. By virtue of the small number of landmark points (relative to the total number of features) and the sequential operating mode, the computational complexity of the proposed scheme is set *free* from the dependency on the total number of features, surmounting thus the heavy computational-complexity obstacle of its batch predecessors [31, 32] which appears especially in cases where the length of the network nodal time-series is excessively large. A description of the novel sequential clustering scheme is given in Fig. 2b.



(a) The network-clustering framework



(b) Fast sequential clustering in Riemannian manifolds

Fig. 2: (a) The pipeline of the proposed network-clustering framework. (b) The fast sequential-clustering scheme is used in submodules Comm2, Comm4, St2, Sub2 and Sub4 of the framework in Fig. 2a.

D. Organization of the Manuscript

The rest of the manuscript delineates the building blocks/submodules of the framework of Fig. 2. In Sec. II, the feature-extraction scheme is detailed, while Sec. III presents the novel sequential clustering algorithm. Numerical tests on synthetic and real data are reported in Sec. IV. Finally, Sec. V concludes the discussion. Figures and tables that do not fit in the main body of the manuscript, due to space constraints dictated by this Journal, are provided in the supplementary file. Those figures and tables are labeled by the "Supp" tag.

II. Extracting Riemannian Features from Nodal Time-Series

The main mathematical objects of this study are Riemannian manifolds. Loosely speaking, manifolds can be considered as smooth, possibly curved surfaces; see for example Fig. 3. A detailed review on Riemannian manifolds is out of the scope of this study; the interested reader may refer to [24]. Nevertheless, to help the reader and provide details on modules Comm1, Comm3, St1, Sub1, and Sub3 of Fig. 2a, this section showcases the Riemannian manifolds of Grassmann and of positive (semi)definite matrices, as well as the ways to extract and map features from network time-series into those manifolds.

The Grassmannian $\mathrm{Gr}(\rho, M)$ is defined as the collection of all linear subspaces of $\mathbb{R}^M$ with rank equal to ρ; usually, $\rho \ll M$ [23, p. 73], [24]. The Grassmannian $\mathrm{Gr}(\rho, M)$ is a Riemannian manifold with dimension equal to $\rho(M-\rho)$ [23, p. 74], [24]. An element of $\mathrm{Gr}(\rho, M)$, denoted here by $[\mathbf{O}]$, with $\mathbf{O}$ standing for an $M \times \rho$ orthogonal matrix $\mathbf{O}$ whose columns span the linear subspace $[\mathbf{O}]$, cannot be represented uniquely by a Euclidean vector: there is an uncountable number of $M \times \rho$ orthogonal matrices, i.e., orthonormal bases, with columns spanning $[\mathbf{O}]$. In other words, $\mathrm{Gr}(\rho, M)$ is a quotient space [23, 24]. According to Nash's embedding theorem [25, 26], $\mathrm{Gr}(\rho, M)$ can be embedded in a Euclidean space, but of dimensionality considerably larger than $\rho(M-\rho)$ [27]. A Riemannian manifold $\mathcal{M}$ is endowed with a distance measure $\mathrm{dist}_{\mathcal{M}}(\cdot,\cdot)$ [24]. For the linear subspaces $[\mathbf{O}], [\mathbf{O}'] \in \mathrm{Gr}(\rho, M)$, the Riemannian distance between $[\mathbf{O}]$ and $[\mathbf{O}']$ is defined as $\mathrm{dist}_{\mathcal{M}}([\mathbf{O}], [\mathbf{O}']) := (1/2)\|\mathbf{O}\mathbf{O}^{\intercal} - \mathbf{O}'\mathbf{O}'^{\intercal}\|_{\mathrm{F}}$ [61], where $\|\cdot\|_{\mathrm{F}}$ stands for the Frobenius norm. The cost $\mathrm{C}_{\mathrm{dist}}$ of computing $\mathrm{dist}_{\mathcal{M}}(\cdot,\cdot)$ scales in the order of $\mathcal{O}(\rho^2 M)$. Moreover, the (manifold) logarithm map $\log_x^{(\mathcal{M})}(\cdot)$, which maps points of the Riemannian manifold $\mathcal{M}$ into the tangent space at the point $x \in \mathcal{M}$ [24] (see Fig. 4), will be used frequently in the following discussion. A way to compute $\log_x^{(\mathcal{M})}(\cdot)$ in the case where $\mathcal{M}$ is the Grassmannian, via the singular value decomposition (SVD) with a computational cost of order $\mathcal{O}(\rho^2 M)$, can be found in [59, 60, 62].

The set of all $M \times M$ (symmetric) positive-(semi)definite matrices $\mathrm{P(S)D}(M)$ is a Riemannian manifold of dimension $M(M+1)/2$ [21, 36, 63]. This work prefers $\mathrm{PD}(M)$ over $\mathrm{PSD}(M)$ due to the "simple" form that the (manifold) logarithm map $\log_x^{(\mathcal{M})}(\cdot)$ takes in the case of $\mathcal{M} := \mathrm{PD}(M)$ [59, 60, 63]. For matrices $\mathbf{K}, \mathbf{K}' \in \mathcal{M} := \mathrm{PD}(M)$, their Riemannian distance is defined as $\mathrm{dist}_{\mathcal{M}}(\mathbf{K}, \mathbf{K}') := \|\log(\mathbf{K}) - \log(\mathbf{K}')\|_{\mathrm{F}}^2$, where $\log(\cdot)$ denotes the usual matrix logarithm [64]. The computational cost of both $\log_x^{(\mathcal{M})}(\cdot)$ and $\log(\cdot)$ scales in the order of $\mathcal{O}(M^3)$ [63, 64].

A. Notation

Consider a multilayer network/graph $\mathcal{G} := (\mathcal{N}, \mathcal{E}, \mathcal{L})$, 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 $(y_\nu^{(l)}[t])_{t\in\mathbb{Z}}$, where $\mathbb{Z}$ denotes the set of integer numbers; cf. Fig. 1. The physical meaning of $\mathcal{N}$, $\mathcal{L}$ and $(y_\nu^{(l)}[t])_{t\in\mathbb{Z}}$ depends on the underlying data-collection modalities. In fMRI, for example, $\mathcal{N}$ comprises regions of interest (ROI) of the brain which are connected either anatomically or functionally, $\mathcal{L}$ may serve as a set of frequency bands in which time-series are observed, or a group of subjects in a clinical study, and $(y_\nu^{(l)}[t])_{t\in\mathbb{Z}}$ becomes the BOLD time-series of the averaged signal in the νth ROI of layer l [4]; e.g., Fig. 5d.

For a subset/collection $\mathcal{V}$ 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 lth layer of the network at time t. For example, in the case of state clustering, where a "snapshot" of all nodes of the lth layer of the network is needed, $\mathcal{V}$ takes the value of $\mathcal{N}$ and $\mathbf{y}_{\mathcal{N}}^{(l)}[t] := [y_1^{(l)}[t], \ldots, y_{|\mathcal{N}|}^{(l)}[t]]^{\intercal}$, with $q := |\mathcal{N}|$ and with $^{\intercal}$ standing for vector/matrix transposition. On the other hand, in the cases of community detection and subnetwork-sequence identification, where features per nodal time-series need to be extracted, $\mathcal{V} := \nu$, for $\nu \in \mathcal{N}$, so that for a given time-window length $\tau_{\mathrm{w}} \in \mathbb{Z}_{>0}$ and with $q = \tau_{\mathrm{w}}$, $\mathbf{y}_\nu^{(l)}[t] := [y_\nu^{(l)}[t], \ldots, y_\nu^{(l)}[t+\tau_{\mathrm{w}}-1]]^{\intercal}$. Under rigorous notation, $\mathcal{V} \in \mathfrak{N}$, where set $\mathfrak{N} \subset 2^{\mathcal{N}}$ collects all available subsets of nodes, with $2^{\mathcal{N}}$ denoting the power set of $\mathcal{N}$. For a subset of consecutive time instances $\mathcal{T} \subset \mathbb{Z}$ of finite cardinality $|\mathcal{T}| < +\infty$, the following discussion assumes that data $(\mathbf{y}_{\mathcal{V}}^{(l)}[t])_{t\in\mathcal{T}}$ are available to the user, for layers $l \in \mathcal{L}$ and subsets/collections of nodes $\mathcal{V} \in \mathfrak{N}$.

B. Features in the Grassmannian

Motivated by the success of kernel methods in capturing non-linearities in data [65, 66], consider now a user-defined reproducing kernel Hilbert space (RKHS) $\mathcal{H}$, with its associated reproducing kernel function $\kappa(\cdot,\cdot)$ and the induced mapping function $\varphi : \mathbb{R}^q \to \mathcal{H} : \mathbf{y} \mapsto \varphi(\mathbf{y}) := \kappa(\mathbf{y}, \cdot)$ (cf. Appx. A). There can be many choices for $\kappa(\cdot,\cdot)$. Combinations of multiple kernels can also generate new kernel functions (cf. Appx. A). For a user defined parameter $T_{\mathrm{w}} \in \mathbb{Z}_{>0}$, define $\boldsymbol{\varphi}_t^{(l)} := [\varphi(\mathbf{y}_{\mathcal{V}}^{(l)}[t]), \varphi(\mathbf{y}_{\mathcal{V}}^{(l)}[t+1]), \ldots, \varphi(\mathbf{y}_{\mathcal{V}}^{(l)}[t+T_{\mathrm{w}}-1])]^{\intercal} \in \mathcal{H}^{T_{\mathrm{w}}}$. Following the kernel-autoregressive-moving-average (kARMA) model of [32], it is assumed that there exist matrices $\mathbf{C}^{(l)} \in \mathbb{R}^{T_{\mathrm{w}}\times\rho}$, $\mathbf{A}^{(l)} \in \mathbb{R}^{\rho\times\rho}$, the latent $\boldsymbol{\psi}_t^{(l)} \in \mathcal{H}^{\rho}$, and $\boldsymbol{\upsilon}_t^{(l)} \in \mathcal{H}^{T_{\mathrm{w}}}$, $\boldsymbol{\omega}_t^{(l)} \in \mathcal{H}^{\rho}$, which model noise and approximation errors, such that (s.t.) $\forall t$,

$$\boldsymbol{\varphi}_t^{(l)} = \mathbf{C}^{(l)}\boldsymbol{\psi}_t^{(l)} + \boldsymbol{\upsilon}_t^{(l)}\,, \qquad \boldsymbol{\psi}_t^{(l)} = \mathbf{A}^{(l)}\boldsymbol{\psi}_{t-1}^{(l)} + \boldsymbol{\omega}_t^{(l)}\,.$$

Given also parameters $m, \tau_{\rm f}, \tau_{\rm b} \in \mathbb{Z}_{>0}$, define the "forward matrix" in $\mathcal{H}^{mT_{\rm w}\times\tau_{\rm f}}$

$$\boldsymbol{\mathcal{F}}_{\mathcal{V}}^{(l)}[t] := \begin{bmatrix} \boldsymbol{\varphi}_{t}^{(l)} & \boldsymbol{\varphi}_{t+1}^{(l)} & \cdots & \boldsymbol{\varphi}_{t+\tau_{\rm f}-1}^{(l)} \\ \boldsymbol{\varphi}_{t+1}^{(l)} & \boldsymbol{\varphi}_{t+2}^{(l)} & \cdots & \boldsymbol{\varphi}_{t+\tau_{\rm f}}^{(l)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\varphi}_{t+m-1}^{(l)} & \boldsymbol{\varphi}_{t+m}^{(l)} & \cdots & \boldsymbol{\varphi}_{t+\tau_{\rm f}+m-2}^{(l)} \end{bmatrix}, \quad (1a)$$

and the "backward matrix" in $\mathcal{H}^{\tau_{\rm b}T_{\rm w}\times\tau_{\rm f}}$

$$\boldsymbol{\mathcal{B}}_{\mathcal{V}}^{(l)}[t] := \begin{bmatrix} \boldsymbol{\varphi}_{t}^{(l)} & \boldsymbol{\varphi}_{t+1}^{(l)} & \cdots & \boldsymbol{\varphi}_{t+\tau_{\rm f}-1}^{(l)} \\ \boldsymbol{\varphi}_{t-1}^{(l)} & \boldsymbol{\varphi}_{t}^{(l)} & \cdots & \boldsymbol{\varphi}_{t+\tau_{\rm f}-2}^{(l)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\varphi}_{t-\tau_{\rm b}+1}^{(l)} & \boldsymbol{\varphi}_{t-\tau_{\rm b}+2}^{(l)} & \cdots & \boldsymbol{\varphi}_{t+\tau_{\rm f}-\tau_{\rm b}}^{(l)} \end{bmatrix}. \quad (1b)$$

Then, according to [32], there exist matrices $\mathbf{\Pi}^{(l)}[t+1] \in \mathbb{R}^{\rho\times\tau_{\rm b}T_{\rm w}}$ and $\boldsymbol{\mathcal{E}}_{\tau_{\rm f}}^{(l)}[t+1] \in \mathbb{R}^{mT_{\rm w}\times\tau_{\rm b}T_{\rm w}}$ s.t. the following *low-rank* factorization holds true: $(1/\tau_{\rm f})\boldsymbol{\mathcal{F}}_{\mathcal{V}}^{(l)}[t+1] \otimes_{\mathcal{H}} \boldsymbol{\mathcal{B}}_{\mathcal{V}}^{(l)}[t]^{\intercal} = \mathbf{O}_{\mathcal{V}}^{(l)}\mathbf{\Pi}^{(l)}[t+1] + \boldsymbol{\mathcal{E}}_{\tau_{\rm f}}^{(l)}[t+1]$, where product $\otimes_{\mathcal{H}}$ is defined in Appx. B, and $\mathbf{O}_{\mathcal{V}}^{(l)}$ is the so-called *observability* matrix: $\mathbf{O}_{\mathcal{V}}^{(l)} := \left[\mathbf{C}^{(l)\intercal}, (\mathbf{C}^{(l)}\mathbf{A}^{(l)})^{\intercal}, \ldots, (\mathbf{C}^{(l)}(\mathbf{A}^{(l)})^{m-1})^{\intercal}\right]^{\intercal} \in \mathbb{R}^{mT_{\rm w}\times\rho}$. Under certain conditions, it can be shown that $(\lim_{\tau_{\rm f}\to\infty} \boldsymbol{\mathcal{E}}_{\tau_{\rm f}}^{(l)}[t] = \mathbf{0},\ \forall t)$ [32]. Hence, the following task is proposed to obtain an estimate of the observability matrix:

$$\left(\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1], \hat{\mathbf{\Pi}}^{(l)}[t+1]\right) \in \underset{\substack{\mathbf{O}\in\mathbb{R}^{mT_{\rm w}\times\rho} \\ \mathbf{\Pi}\in\mathbb{R}^{\rho\times\tau_{\rm b}T_{\rm w}}}}{\arg\min} \left\| \tfrac{1}{\tau_{\rm f}}\boldsymbol{\mathcal{F}}_{\mathcal{V}}^{(l)}[t+1] \otimes_{\mathcal{H}} \boldsymbol{\mathcal{B}}_{\mathcal{V}}^{(l)}[t]^{\intercal} - \mathbf{O}\mathbf{\Pi} \right\|_{\rm F}^{2}. \quad (2)$$

To solve (2), SVD is applied to obtain $(1/\tau_{\rm f})\boldsymbol{\mathcal{F}}_{\mathcal{V}}^{(l)}[t+1] \otimes_{\mathcal{H}} \boldsymbol{\mathcal{B}}_{\mathcal{V}}^{(l)}[t]^{\intercal} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\intercal}$, where the $(mT_{\rm w} \times mT_{\rm w})$-matrix $\mathbf{U}$ is orthogonal. Assuming that $\rho \leq \operatorname{rank}(\boldsymbol{\mathcal{F}}_{\mathcal{V}}^{(l)}[t+1] \otimes_{\mathcal{H}} \boldsymbol{\mathcal{B}}_{\mathcal{V}}^{(l)}[t]^{\intercal})$, the Schmidt-Mirsky-Eckart-Young theorem [67] provides the estimate $\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1] := \mathbf{U}_{:,1:\rho}$, where $\mathbf{U}_{:,1:\rho}$ is the orthogonal matrix that collects the ρ columns of $\mathbf{U}$ that correspond to the top (principal) ρ singular values in $\mathbf{\Sigma}$. Due to the ambiguity $\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1] \cdot \hat{\mathbf{\Pi}}^{(l)}[t+1] = \hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1]\mathbf{P} \cdot \mathbf{P}^{-1}\hat{\mathbf{\Pi}}^{(l)}[t+1]$, for any non-singular matrix $\mathbf{P}$, and the observation that the column (range) spaces of $\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1]\mathbf{P}$ and $\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1]$ coincide, it becomes preferable to record the column space of $\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1]$, denoted by $[\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1]]$, rather than $\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1]$ itself. Interestingly, for $\rho := \operatorname{rank}[\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1]]$, the linear subspace $[\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1]]$ becomes a point in the Grassmannian $\mathrm{Gr}(\rho, mT_{\rm w})$. The algorithmic procedure of extracting feature $[\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1]]$ is summarized in Alg. 1.

Parameters in Alg. 1 need to be chosen properly to guarantee that feature $[\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1]]$ captures the statistical information of the data. Parameters $T_{\rm w}$, m, and ρ control the dimension of the Grassmannian. The sum of m, $\tau_{\rm f}$ and $\tau_{\rm b}$ should not be greater than the length of the time-series, due to the size of the "forward" and "backward" matrices, while a large value of $\tau_{\rm f}$ can help reduce the estimation error in (2).

Algorithm 1: Extracting Grassmannian features

Input : Data $\{(\mathbf{y}_{\mathcal{V}}^{(l)}[t])_{t\in\mathcal{T}}\}_{\mathcal{V}\in\mathfrak{N},l\in\mathcal{L}}$.
Parameters: Positive integers $T_{\rm w}$, m, ρ, $\tau_{\rm f}$ and $\tau_{\rm b}$.
Output : Grassmannian features $\{x_i^{\rm obs}\}_{i\in\mathcal{I}_{\rm obs}}$.

$i = 0$.
for *all available layers* l **do**
  for *all available collections of nodes* $\mathcal{V}$ **do**
    for *all available time instances* t **do**
      Form $(1/\tau_{\rm f})\boldsymbol{\mathcal{F}}_{\mathcal{V}}^{(l)}[t+1] \otimes_{\mathcal{H}} \boldsymbol{\mathcal{B}}_{\mathcal{V}}^{(l)}[t]^{\intercal}$ via (1) and Appx. B.
      Apply singular value decomposition: $(1/\tau_{\rm f})\boldsymbol{\mathcal{F}}_{\mathcal{V}}^{(l)}[t+1] \otimes_{\mathcal{H}} \boldsymbol{\mathcal{B}}_{\mathcal{V}}^{(l)}[t]^{\intercal} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\intercal}$.
      $i \leftarrow i+1$.
      Feature $x_i^{\rm obs} := [\hat{\mathbf{O}}_{\mathcal{V}}^{(l)}[t+1]] \in \mathrm{Gr}(\rho, mT_{\rm w})$ is the linear subspace spanned by the ρ "principal" columns of $\mathbf{U}$.
Gather all features $\{x_i^{\rm obs}\}_{i\in\mathcal{I}_{\rm obs}}$.

C. Features in the Positive-Definite-Matrices Manifold

This section demonstrates ways to map correlations, extracted from multi-layer network time-series $\{(y_{\nu}^{(l)}[t])_{t\in\mathcal{T}}\}_{\nu\in\mathcal{N},l\in\mathcal{L}}$, to the Riemannian manifold of positive-(semi)definite matrices. To this end, define the offset $\mu_{\nu}^{(l)} := (1/|\mathcal{T}|)\sum_{t\in\mathcal{T}} y_{\nu}^{(l)}[t]$ and let the "centered" time-series $\tilde{y}_{\nu}^{(l)}[t] := y_{\nu}^{(l)}[t] - \mu_{\nu}^{(l)}$. Consider, then, a sliding window of length $\tau_{\rm w} \in \mathbb{Z}_{>0}$, and define the $(\tau_{\rm w}\times 1)$-vector $\tilde{\mathbf{y}}_{\nu}^{(l)}[t] := [\tilde{y}_{\nu}^{(l)}[t], \tilde{y}_{\nu}^{(l)}[t+1], \ldots, \tilde{y}_{\nu}^{(l)}[t+\tau_{\rm w}-1]]^{\intercal}$. Aiming at kernel correlations, consider also a user-defined RKHS $\mathcal{H}$, with its reproducing kernel function $\kappa(\cdot,\cdot)$ and the induced non-linear mapping $\varphi(\cdot)$ (cf. Appx. A).

1) Kernel Correlations

To address community detection and subnetwork-sequence clustering tasks in multilayer networks, kernel correlations will be extracted from $(\tilde{y}_{\nu}^{(l)}[t])_{t\in\mathcal{T}}$ for every ν. To this end, define $\boldsymbol{\varphi}_{t}^{(l)} := [\varphi(\tilde{\mathbf{y}}_{\nu}^{(l)}[t]), \varphi(\tilde{\mathbf{y}}_{\nu}^{(l)}[t+1]), \ldots, \varphi(\tilde{\mathbf{y}}_{\nu}^{(l)}[t+T_{\rm w}-1])]^{\intercal} \in \mathcal{H}^{T_{\rm w}}$ to capture non-linear correlations, where the sliding time-window parameter $T_{\rm w} \in \mathbb{Z}_{>0}$. The kernel correlation matrix is defined as the $T_{\rm w}\times T_{\rm w}$ matrix $\mathbf{K}_{\nu}^{(l)}[t] := \boldsymbol{\varphi}_{t}^{(l)} \otimes_{\mathcal{H}} \boldsymbol{\varphi}_{t}^{(l)\intercal}$, whose (ξ,ξ')th entry is $\kappa(\tilde{\mathbf{y}}_{\nu}^{(l)}[t+\xi-1], \tilde{\mathbf{y}}_{\nu}^{(l)}[t+\xi'-1])$. In the case where kernel $\kappa(\cdot,\cdot)$ becomes the linear one (cf. Appx. A), then each entry $\kappa(\tilde{\mathbf{y}}_{\nu}^{(l)}[t+\xi-1], \tilde{\mathbf{y}}_{\nu}^{(l)}[t+\xi'-1])$ boils down to the classical dot-vector product $\tilde{\mathbf{y}}_{\nu}^{(l)}[t+\xi-1]^{\intercal}\tilde{\mathbf{y}}_{\nu}^{(l)}[t+\xi'-1]$, which is, in turn, a scaled version of the classical sample correlation $(1/\tau_{\rm w})\sum_{\tau=0}^{\tau_{\rm w}-1} \tilde{y}_{\nu}[t+\xi-1+\tau]\tilde{y}_{\nu}[t+\xi'-1+\tau]$ of lag $\xi-\xi'$. Since $\mathbf{K}_{\nu}^{(l)}[t]$ is symmetric, all of its information is included in the $T_{\rm w}(T_{\rm w}+1)/2$ entries located in its main diagonal and upper triangular parts.

In contrast to the conventional approach of vectorizing those entries to use the resultant Euclidean $[T_{\rm w}(T_{\rm w}+1)/2]\times 1$ vector as a feature, this study chooses the matrix $\mathbf{K}_{\nu}^{(l)}[t]$ itself as a feature in the Riemannian manifold P(S)D$(T_{\rm w})$ of positive-(semi)definite matrices, of dimension $T_{\rm w}(T_{\rm w}+1)/2$, to take advantage of the rich Riemannian geometry of P(S)D$(T_{\rm w})$.

Algorithm 2: Extracting kernel (partial) correlations

Input : Data $\{(y_\nu^{(l)}[t])_{t\in\mathcal{T}}\}_{\nu\in\mathcal{N},l\in\mathcal{L}}$.
Parameters: Time-window lengths $\tau_{\rm w}$ and $T_{\rm w}$, $\epsilon>0$, and kernel function $\kappa(\cdot,\cdot)$.
Output : Features $\{x_i^{\rm kc}\}_{i\in\mathcal{I}_{\rm kc}}$ and $\{x_{i'}^{\rm kpc}\}_{i'\in\mathcal{I}_{\rm kpc}}$.

1 $i=0$; $i'=0$.
2 **for** *all available layers* l **do**
3 **for** *all available time instances* t **do**
4 **for** *all nodes* ν **do**
5 Let $\mu_\nu^{(l)} := (1/|\mathcal{T}|)\sum_{t\in\mathcal{T}} y_\nu^{(l)}[t]$.
6 $\tilde{y}_\nu^{(l)}[t] := y_\nu^{(l)}[t]-\mu_\nu^{(l)}$.
7 $\tilde{\mathbf{y}}_\nu^{(l)}[t] :=$ $[\tilde{y}_\nu^{(l)}[t],\tilde{y}_\nu^{(l)}[t+1],\ldots,\tilde{y}_\nu^{(l)}[t+\tau_{\rm w}-1]]^\intercal$.
8 Generate the $T_{\rm w}\times T_{\rm w}$ kernel matrix $\mathbf{K}_\nu^{(l)}[t]$ whose (ξ,ξ')th entry is $\kappa(\tilde{\mathbf{y}}_\nu^{(l)}[t+\xi-1],\tilde{\mathbf{y}}_\nu^{(l)}[t+\xi'-1])$.
9 **if** $\mathbf{K}_\nu^{(l)}[t]$ *is singular* **then**
10 $\mathbf{K}_\nu^{(l)}[t]\leftarrow\mathbf{K}_\nu^{(l)}[t]+\epsilon\mathbf{I}_{T_{\rm w}}$.
11 $i\leftarrow i+1$.
12 Kernel-correlation feature: $x_i^{\rm kc}:=\mathbf{K}_\nu^{(l)}[t]$.
13 Generate the $|\mathcal{N}|\times|\mathcal{N}|$ kernel matrix $\mathbf{K}_\mathcal{N}^{(l)}[t]$ with (ν,ν')th entry $\kappa(\tilde{\mathbf{y}}_\nu^{(l)}[t],\tilde{\mathbf{y}}_{\nu'}^{(l)}[t])$.
14 **if** $\mathbf{K}_\mathcal{N}^{(l)}[t]$ *is singular* **then**
15 $\mathbf{K}_\mathcal{N}^{(l)}[t]\leftarrow\mathbf{K}_\mathcal{N}^{(l)}[t]+\epsilon\mathbf{I}_{|\mathcal{N}|}$.
16 $i'\leftarrow i'+1$.
17 Kernel-partial-correlation feature: $x_{i'}^{\rm kpc}:=(\text{diag}\mathbf{K}_\mathcal{N}^{(l)}[t]^{-1})^{-1/2}\cdot\mathbf{K}_\mathcal{N}^{(l)}[t]^{-1}\cdot(\text{diag}\mathbf{K}_\mathcal{N}^{(l)}[t]^{-1})^{-1/2}$.
18 Gather all features $\{x_i^{\rm kc}\}_{i\in\mathcal{I}_{\rm kc}}$ and $\{x_{i'}^{\rm kpc}\}_{i'\in\mathcal{I}_{\rm kpc}}$.

The inverse correlation matrix was used as a feature in identifying network topology in [68, 69]. The majority of state-of-the-art methods, such as the 4D-tensor method [17], the graph-signal-processing approach of [43], as well as [14], use a "sliding-window" correlation matrix as the feature that reveals network communities. However, only the present study exploits the entailing Riemannian geometry of the (kernel) correlation matrix to tackle community detection and even the rarely addressed subnetwork-sequence tracking.

2) Kernel Partial Correlations

To address state clustering, a snapshot of the network needs to be taken at every time instance t to monitor the evolving network topology. The straightforward approach is to keep track of the feature $\mathbf{K}_\mathcal{N}^{(l)}[t]$, i.e., the $|\mathcal{N}|\times|\mathcal{N}|$ kernel matrix whose (ν,ν')th entry is $[\mathbf{K}_\mathcal{N}^{(l)}[t]]_{\nu,\nu'} := \kappa(\mathbf{y}_\nu^{(l)}[t],\mathbf{y}_{\nu'}^{(l)}[t])$. Nevertheless, motivated by [31], kernel partial correlations are used here to reveal the "proximity" of the time-series of any two nodes ν,ν', after removing the "effect" that the time-series of the rest of the nodes $\mathcal{N}\setminus\{\nu,\nu'\}$ have on the time-series of ν and ν'. To this end, define for $\iota\in\{\nu,\nu'\}$ the residual signal $r_\iota^{(l)}[t] := \varphi(\tilde{\mathbf{y}}_\iota^{(l)}[t])-\sum_{j\in\mathcal{N}\setminus\{\nu,\nu'\}}\beta_{\iota j}^*\varphi(\tilde{\mathbf{y}}_j^{(l)}[t])$,

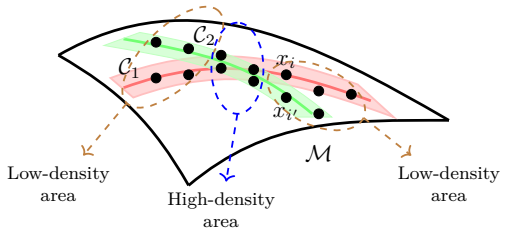


Fig. 3: The Riemannian multi-manifold modeling (RMMM) hypothesis: the feature point-cloud is assumed to be placed close to or into the union of a finite number of low-dimensional submanifolds. To facilitate the selection of landmark points, the point-cloud is separated in high- and low-density areas.

where a specific linear combination of $\{\varphi(\tilde{\mathbf{y}}_j^{(l)}[t]) : j\in\mathcal{N}\setminus\{\nu,\nu'\}\}$ is subtracted from vectors $\varphi(\tilde{\mathbf{y}}_\nu^{(l)}[t])$ and $\varphi(\tilde{\mathbf{y}}_{\nu'}^{(l)}[t])$. Coefficients $\{\beta_{\iota j}^*\}_{j\in\mathcal{N}\setminus\{\nu,\nu'\}}$ are the solutions of the following least-squares estimation task $\{\beta_{\iota j}^*\}_{j\in\mathcal{N}\setminus\{\nu,\nu'\}}\in\arg\min_{\beta_{\iota j}}\|\varphi(\tilde{\mathbf{y}}_\iota^{(l)}[t])-\sum_{j\in\mathcal{N}\setminus\{\nu,\nu'\}}\beta_{\iota j}\varphi(\tilde{\mathbf{y}}_j^{(l)}[t])\|_\mathcal{H}^2$, which renders $r_\iota^{(l)}[t]$ orthogonal to the linear subspace spanned by $\{\varphi(\tilde{\mathbf{y}}_j^{(l)}[t]) : j\in\mathcal{N}\setminus\{\nu,\nu'\}\}$. For any pair of nodes (ν,ν'), the kernel partial correlation is then defined as [31]

$$\text{kPCorr}_{\nu\nu'}^{(l)}[t] := \frac{\langle r_\nu^{(l)}[t]\mid r_{\nu'}^{(l)}[t]\rangle_\mathcal{H}}{\|r_\nu^{(l)}[t]\|_\mathcal{H}\cdot\|r_{\nu'}^{(l)}[t]\|_\mathcal{H}}.$$

According to [31], if $\mathbf{K}_\mathcal{N}^{(l)}[t]$ is non-singular, then $\text{kPCorr}_{\nu\nu'}^{(l)}[t]$ is the (ν,ν')th entry of the positive-definite matrix $(\text{diag}\mathbf{K}_\mathcal{N}^{(l)}[t]^{-1})^{-1/2}\cdot\mathbf{K}_\mathcal{N}^{(l)}[t]^{-1}\cdot(\text{diag}\mathbf{K}_\mathcal{N}^{(l)}[t]^{-1})^{-1/2}\in\text{PD}(|\mathcal{N}|)$, where $\text{diag}\mathbf{K}_\mathcal{N}^{(l)}[t]^{-1}$ is the diagonal matrix whose main diagonal entries are equal to those of $\mathbf{K}_\mathcal{N}^{(l)}[t]^{-1}$. Apart from [31], partial correlations were also used in [70, 71] for brain-network state clustering, showing advantages over standard correlations. However, unlike [31] and the present study, [70, 71] use neither kernels nor any Riemannian geometry.

The steps that extract kernel (partial) correlations and their corresponding Riemannian features are summarized in Alg. 2. In the case where $\mathbf{K}_\nu^{(l)}[t]$ and $\mathbf{K}_\mathcal{N}^{(l)}[t]$ are singular, "diagonal loading" via a small positive real number ϵ is used in steps 9 and 14 of Alg. 2 to render those matrices positive definite, and simplify thus the computation of the manifold logarithm map $\log_x^{(\mathcal{M})}(\cdot)$, as explained in the introductory part of Sec. II.

III. Fast Sequential Clustering in Riemannian Manifolds

This section details the novel sequential scheme of Fig. 2b which is employed in submodules Comm2, Comm4, St2, Sub2 and Sub4 of the framework in Fig. 2a. It is assumed that the Riemannian features/points $\mathcal{X} := \{x_i\}_{i\in\mathcal{I}}$ are available to the user through the extraction procedures of Sec. II. In other words, $\{x_i\}_{i\in\mathcal{I}}$ represents here any of the collections $\{x_i^{\rm obs}\}_{i\in\mathcal{I}_{\rm obs}}$ (Alg. 1), $\{x_i^{\rm kc}\}_{i\in\mathcal{I}_{\rm kc}}$ (Alg. 2), or $\{x_i^{\rm kpc}\}_{i\in\mathcal{I}_{\rm kpc}}$ (Alg. 2).

A. Selecting Landmark Points

The first step in Fig. 2b is to select a small number of "landmark," or, representative points $\Lambda := \{\ell_i\}_{i\in\mathcal{I}_{\rm Land}}$ from

$\mathcal{X} \coloneqq \{x_i\}_{i\in\mathcal{I}}$, i.e., $\Lambda \subset \mathcal{X}$, that provides an approximate description (sketch) of the geometry of the possibly massive point-cloud $\mathcal{X}$.

The strategy of selecting Λ randomly, under a uniform distribution, from $\mathcal{X}$ (validated in Sec. IV under the tag "RM"), albeit computationally efficient, seems inappropriate to capture the latent geometry of the $\mathcal{X}$. Another approach, called systematic selection (Sys) [72], samples one point from $\mathcal{X}$ every a user-defined number of features, in the order of their appearance after the extraction schemes of Sec. II. Furthermore, a classical point-selection scheme is importance sampling [73]. Nevertheless, importance sampling requires prior knowledge or estimation of the underlying probability distribution functions of the data-points, via training data, which is not viable in this context due to the unsupervised nature of the clustering task and the absence of training data. Another popular scheme, which combines point sampling and clustering, usually via k-means, is the random sample and consensus (RANSAC) algorithm [74]. Although vastly used to combat outliers within data, RANSAC does not utilize any Riemannian geometric information. RANSAC is also considered as one of the competing methods in Sec. IV.

This study focuses on the greedy and *non-random* maxmin algorithm [75]. Originally developed for Euclidean spaces, the maxmin algorithm is tailored here to fit the present context by substituting the Euclidean distance with the Riemannian one. The maxmin algorithm is an iterative procedure that updates the pool of landmark points sequentially. At step s of the maxmin algorithm, the minimum of the distances of an arbitrarily chosen non-landmark point from the existing landmark ones Λ_s is computed. Such a minimum distance is computed for every non-landmark point. The non-landmark point x_{i_*} which scores the maximum such minimum distance is "promoted" to a new landmark point, and the landmark points are updated as $\Lambda_{s+1} \coloneqq \Lambda_s \cup \{x_{i_*}\}$, before moving on to step $s+1$ of the algorithm.

The maxmin algorithm [75] is modified further in Alg. 3, and coined *density-based manifold* maxmin (DM-Maxmin) algorithm, to offer a data-driven scheme that includes more elements of the geometry of $\mathcal{X}$ than the original [75] does. More specifically, DM-Maxmin samples a number N_{High} of landmark points from a "high-density" area $\mathcal{A}_{\text{High}}$ of the point-cloud, larger than the number N_{Low} of landmark points collected from the "low-density" area $\mathcal{A}_{\text{Low}}$. Whether a feature point x_i belogs to $\mathcal{A}_{\text{High}}$ or $\mathcal{A}_{\text{Low}}$ depends on the number n_i of feature points $\{x_{i'} \in \mathcal{X} \,|\, \text{dist}_{\mathcal{M}}(x_i, x_{i'}) \leq d_{\text{dense}}\}$ included in the Riemannian ball $\{z \in \mathcal{M} \,|\, \text{dist}_{\mathcal{M}}(x_i, z) \leq d_{\text{dense}}\}$ centered at x_i with radius d_{dense}, where $\text{dist}_{\mathcal{M}}(x_i, x_{i'})$ stands for the Riemannian distance between points x_i and $x_{i'}$. If the number n_i of those points exceeds a user-defined threshold N_{dense}, then $x_i \in \mathcal{A}_{\text{High}}$; otherwise, $x_i \in \mathcal{A}_{\text{Low}}$. After $\mathcal{A}_{\text{High}}$ and $\mathcal{A}_{\text{Low}}$ have been determined, the aforementioned Riemannian-distance variant of the maxmin algorithm is applied to the points in $\mathcal{A}_{\text{High}}$ and $\mathcal{A}_{\text{Low}}$ to select N_{High} and N_{Low} landmark points Λ_{High} and Λ_{Low}, respectively. Finally, all landmark points are gathered in $\Lambda \coloneqq \Lambda_{\text{Low}} \cup \Lambda_{\text{Low}}$.

The numerical tests of Sec. IV indeed demonstrate the ability of Alg. 3 to capture data geometries, especially in the "high-density" intersection areas of the possibly overlapping clusters in the Riemannian manifold; e.g., Fig. 3.

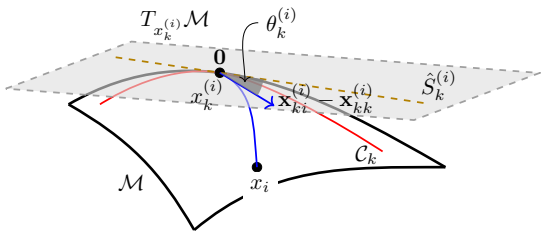


Fig. 4: The linear subspace $\hat{S}_k^{(i)}$ serves as an "image" of the cluster/submanifold $\mathcal{C}_k$ in the Euclidean tangent space $T_{x_k^{(i)}}\mathcal{M}$ of $\mathcal{M}$ at $x_k^{(i)}$. The angle $\theta_k^{(i)}$ between vector $\mathbf{x}_{ki}^{(i)} - \mathbf{x}_{kk}^{(i)}$ and $\hat{S}_k^{(i)}$ is assigned to each non-landmark point x_i; see Alg. 4.

Algorithm 3: Density-based manifold maxmin (DM-Maxmin) algorithm

Input : Riemannian feature-points $\mathcal{X} \coloneqq \{x_i\}_{i\in\mathcal{I}}$.
Parameters: Distance threshold d_{dense}, minimum number of neighbors N_{dense}, desired number N_{High} of landmark points from the "high-density" area $\mathcal{A}_{\text{High}}$, and desired number N_{Low} of landmark points from the low-density area $\mathcal{A}_{\text{Low}}$.
Output : Landmark points $\Lambda \coloneqq \{\ell_i\}_{i\in\mathcal{I}_{\text{Land}}}$ of cardinality $|\Lambda| = N_{\text{High}} + N_{\text{Low}}$.

for *all available points/features* $x_i \in \mathcal{X}$ **do**
  Count the number n_i of data-points $\{x_{i'} \,|\, \text{dist}_{\mathcal{M}}(x_i, x_{i'}) \leq d_{\text{dense}}\}$ contained in a "Riemannian ball" with center at x_i and radius d_{dense}.
  if $n_i \geq N_{\text{dense}}$ **then**
    $x_i \in \mathcal{A}_{\text{High}}$.
  else
    $x_i \in \mathcal{A}_{\text{Low}}$.
Apply the Riemannian-distance variant of the maxmin algorithm to the high-density-area points $\mathcal{A}_{\text{High}}$ to select N_{High} landmark points Λ_{High}.
Apply the Riemannian-distance variant of the maxmin algorithm to the low-density-area points $\mathcal{A}_{\text{Low}}$ to select N_{Low} landmark points Λ_{Low}.
Gather all landmark points in $\Lambda \coloneqq \Lambda_{\text{High}} \cup \Lambda_{\text{Low}}$.

B. Clustering Landmark Points

Having sketched the geometry of $\mathcal{X}$ via the landmark points Λ of Sec. III-A, the next step in the pipeline of Fig. 2b is to cluster Λ to obtain a rough estimate of clusters $\{\mathcal{C}_k\}_{k=1}^K$. This rough estimate of $\{\mathcal{C}_k\}_{k=1}^K$ serves as the starting point for the sequential clustering scheme of Alg. 4 (see step 1), which adds the finer details in $\{\mathcal{C}_k\}_{k=1}^K$ by visiting sequentially all non-landmark points $\mathcal{X} \setminus \Lambda$.

The data-modeling assumption, under which $\mathcal{X}$ are clustered, is the Riemannian multi-manifold modeling (RMMM) hypothesis [59, 60]: feature-points $\mathcal{X}$ are assumed to be

located close to or into a union of finite number of low-dimensional, and of possibly different dimensionality, submanifolds/clusters $\{\mathcal{C}_k\}_{k=1}^K$ (see Fig. 3 for the case of $K = 2$ clusters). In contrast to conventional hypotheses, where clusters (better, their convex hulls) are assumed to be "sufficiently well separated," e.g., K-means, clusters $\{\mathcal{C}_k\}_{k=1}^K$ are allowed to intersect according to the RMMM hypothesis.

Clustering of Λ is achieved by the very recently introduced method of extended geodesic clustering by tangent spaces (eGCT) [32]. In a nutshell, eGCT computes the affinity matrix of all points in Λ via information about sparse data approximations and the angular information hidden behind tangent spaces in Riemannian manifolds. Hierarchical clustering is applied to that affinity matrix to obtain clusters $\{\mathcal{C}_k\}_{k=1}^K$ without any need to know in advance the number K of clusters. The eGCT is a variant of GCT [31, 59, 60], where the number K of clusters is assumed to be known beforehand, and where spectral clustering is used instead of hierarchical clustering. It is also worth stressing here that eGCT is a batch method, with a computational complexity that scales quadratically in the number of feature points [32]. This dependency renders eGCT impractical if applied directly to $\mathcal{X}$ with $|\mathcal{X}|$ being excessively large. Moreover, such a complexity justifies the introduction of the landmark points Λ, where $|\Lambda| \ll |\mathcal{X}|$, and the application of eGCT to Λ. To save space, a detailed description of eGCT is deferred to [32].

C. Sequential Geodesic Clustering by Tangent Spaces

The final submodule in the flowchart of Fig. 2b, detailed in Alg. 4, clusters the non-landmark points $\mathcal{X}\setminus\Lambda$ which comprise the majority of the feature points, since, by design, $|\Lambda| \ll |\mathcal{X}|$. Alg. 4 builds on the arguments that originated GCT [59, 60] and its variants [31, 32], but offers an efficient scheme that visits Riemannian features sequentially, as opposed to its batch and computationally heavy predecessors [31, 32, 59, 60].

After landmark points Λ are clustered to provide a first estimate of the clusters $\{\mathcal{C}_k\}_{k=1}^K$ in step 1, the remaining non-landmark points $\mathcal{X} \setminus \Lambda$ are visited sequentially in step 2 of Alg. 4. For the non-landmark point x_i under query, its closest point $x_k^{(i)}$ and distance $d_k^{(i)}$ from cluster $\mathcal{C}_k$ are identified in step 4. To obtain a view of the geometry of the feature point-cloud around $x_k^{(i)}$, the K_{NN} nearest neighbors of $x_k^{(i)}$ from cluster $\mathcal{C}_k$ are first identified in step 5, and mapped, together with the point x_i under query, into the Euclidean tangent space $T_{x_k^{(i)}}\mathcal{M}$ of $\mathcal{M}$ at $x_k^{(i)}$ via the Riemannian manifold logarithm mapping $\log_{x_k^{(i)}}^{(\mathcal{M})}(\cdot)$ [24, 59, 60] in step 6. Since $T_{x_k^{(i)}}\mathcal{M}$ is a Euclidean space, the mapped vectors $\{\mathbf{x}_{kj}^{(i)} \,|\, x_j \in \mathscr{N}_{\text{NN}}(x_k^{(i)})\}$ can be used to extract statistical information via the sample covariance matrix

$$\hat{\mathbf{C}}_k^{(i)} \coloneqq \frac{1}{K_{\text{NN}}-1} \cdot \sum_{x_j \in \mathscr{N}_{\text{NN}}(x_k^{(i)})} (\mathbf{x}_{kj}^{(i)} - \bar{\mathbf{x}}_k^{(i)})(\mathbf{x}_{kj}^{(i)} - \bar{\mathbf{x}}_k^{(i)})^\intercal \,, \qquad (3)$$

where the sample average is defined as $\bar{\mathbf{x}}_k^{(i)} \coloneqq (1/K_{\text{NN}}) \sum_{x_j \in \mathscr{N}_{\text{NN}}(x_k^{(i)})} \mathbf{x}_{kj}^{(i)}$. The classical tool of principal component analysis (PCA) [76] is then applied to extract a principal eigenspace $\hat{\mathcal{S}}_k^{(i)}$ from $\hat{\mathbf{C}}_k^{(i)}$ in step 7. The eigenspace $\hat{\mathcal{S}}_k^{(i)}$ is determined here by identifying the largest gap between eigenvalues in the eigenvalue decomposition of $\hat{\mathbf{C}}_k^{(i)}$. In other words, the linear subspace $\hat{\mathcal{S}}_k^{(i)}$ serves as an "image" of the submanifold $\mathcal{C}_k$ in $T_{x_k^{(i)}}\mathcal{M}$; see Fig. 3.

The angle $\theta_k^{(i)}$ between the vector $\mathbf{x}_{ki}^{(i)} - \mathbf{x}_{kk}^{(i)}$ and $\hat{\mathcal{S}}_k^{(i)}$ is computed in step 8. Vector $\mathbf{g}_k^{(i)} \coloneqq [d_k^{(i)}, \theta_k^{(i)}]^\intercal$ is formed for every cluster $\mathcal{C}_k$, and the cluster $\mathcal{C}_{k_*}$, which scores the smallest possible ℓ_2-norm $\|\mathbf{g}_k^{(i)}\|_2$ across all clusters $\{\mathcal{C}_k\}_{k=1}^K$, is updated as $\mathcal{C}_{k_*} \cup \{x_i\}$ to include the point x_i under query. After cluster $\mathcal{C}_{k_*}$ is updated, the next of the remaining non-landmark points is examined in step 2. The previous procedure is repeated until all non-landmark points are visited and clustered.

Algorithm 4: Sequential geodesic clustering by tangent spaces

Input : Features $\mathcal{X} \coloneqq \{x_i\}_{i\in\mathcal{I}}$ and landmark points $\Lambda = \{\ell_i\}_{i\in\mathcal{I}_{\text{Land}}}$.
Parameters: $K_{\text{NN}} \in \mathbb{Z}_{>0}$.
Output : Clusters $\{\mathcal{C}_k\}_{k=1}^K$.

1 Apply eGCT [32] to Λ to obtain the first estimate of clusters $\{\mathcal{C}_k\}_{k=1}^K$.
2 **for** *all non-landmark points* $\mathcal{X} \setminus \Lambda$ **do**
3 **for** *all* k *in* $\{1, 2, \ldots, K\}$ **do**
4 Identify the closest point to x_i from $\mathcal{C}_k$: $x_k^{(i)} \coloneqq \arg\min_{x\in\mathcal{C}_k} \text{dist}_{\mathcal{M}}(x, x_i)$. Moreover, let $d_k^{(i)} \coloneqq \text{dist}_{\mathcal{M}}(x_k^{(i)}, x_i)$.
5 Define the K_{NN}-nearest-neighbors $\mathscr{N}_{\text{NN}}(x_k^{(i)})$ of $x_k^{(i)}$ from points of $\mathcal{C}_k$.
6 Map $\mathscr{N}_{\text{NN}}(x_k^{(i)})$ into the tangent space $T_{x_k^{(i)}}\mathcal{M}$ of the Riemannian manifold $\mathcal{M}$ at $x_k^{(i)}$ via the logarithm map: $\mathbf{x}_{kj}^{(i)} \coloneqq \log_{x_k^{(i)}}(x_j)$, $\forall x_j \in \mathscr{N}_{\text{NN}}(x_k^{(i)})$. Let $\mathbf{x}_{kk}^{(i)} \coloneqq \log_{x_k^{(i)}}(x_k^{(i)})$. Map also x_i into the tangent space $T_{x_k^{(i)}}\mathcal{M}$: $\mathbf{x}_{ki}^{(i)} \coloneqq \log_{x_k^{(i)}}(x_i)$.
7 Compute the sample covariance matrix $\hat{\mathbf{C}}_k^{(i)}$ in (3) and its eigenspace $\hat{\mathcal{S}}_k^{(i)}$ by principal component analysis.
8 Compute angle $\theta_k^{(i)}$ between vector $\mathbf{x}_{ki}^{(i)} - \mathbf{x}_{kk}^{(i)}$ and $\hat{\mathcal{S}}_k^{(i)}$.
9 Define vector $\mathbf{g}_k^{(i)} \coloneqq [d_k^{(i)}, \theta_k^{(i)}]^\intercal$.
10 Identify $k_* \coloneqq \arg\min_{k\in\{1,\ldots,K\}} \|\mathbf{g}_k^{(i)}\|_2$.
11 Assign x_i to $\mathcal{C}_{k_*}$, and update $\mathcal{C}_{k_*}$ by $\mathcal{C}_{k_*} \cup \{x_i\}$.

D. Computational Complexity

The computational burdens in the framework of Fig. 2a come from the submodules of feature extraction (Sec. II), landmark-point selection (Sec. III-A), and clustering (Secs. III-B and III-C).

In Sec. II, the computational complexity in Alg. 1 comes from $\text{C}_{\otimes_{\mathcal{H}}}$, i.e., the cost of computing the $mT_{\text{w}} \times \tau_b T_{\text{w}}$ matrix

$\boldsymbol{\mathcal{F}}_{\mathcal{V}}^{(l)}[t+1] \otimes_{\mathcal{H}} \boldsymbol{\mathcal{B}}_{\mathcal{V}}^{(l)}[t]^{\intercal}$, but mainly from $\mathrm{C}_{\mathrm{SVD}}$, which is the cost of the matrix's SVD in (2) that scales in the order of $\mathcal{O}[\min(T_{\mathrm{w}}^3 m^2 \tau_{\mathrm{b}}, T_{\mathrm{w}}^3 \tau_{\mathrm{b}}^2 m)]$. Recall that T_{w} is the user-defined size of the time-sliding window in Sec. II-A. Overall, the computational cost of Alg. 1 is $\mathcal{O}\{|\mathcal{X}|[\mathcal{C}_{\otimes_{\mathcal{H}}} + T_{\mathrm{w}}^3 m \tau_{\mathrm{b}} \min(m, \tau_{\mathrm{b}})]\}$. In Alg. 2, the complexity to extract the kernel-correlation feature is $\mathcal{O}(|\mathcal{N}|^2)$, while that for a kernel-partial-correlation feature is $\mathcal{O}(|\mathcal{N}|^3)$, due to the matrix-inversion operation. Clearly, complexities $\mathcal{O}(|\mathcal{N}|^2)$, $\mathcal{O}(|\mathcal{N}|^3)$ are prohibitive in the case of massive networks where $|\mathcal{N}|$ is large. Ways to surmount this complexity obstacle are outside of the scope of this paper and will be presented elsewhere.

In landmark-point selection (Sec. III-A), Riemannian distances among all features need to be computed. If $\mathrm{C}_{\mathrm{dist}}$ denotes the cost of computing distances between two Riemannian features (see Secs. II-B and II-C1), and since $|\mathcal{X}|(|\mathcal{X}|-1)/2$ distances need to be computed for a number $|\mathcal{X}|$ of features, the overall complexity of computing distances scales in the order of $\mathcal{O}(\mathrm{C}_{\mathrm{dist}}|\mathcal{X}|^2)$.

The very recently introduced batch eGCT algorithm [32] is employed in step 1 of Alg. 4 to cluster the landmark points Λ. Per landmark point, the complexity of eGCT accounts for the following sub-tasks: **(i)** Identify the K_{NN} nearest neighbors of the landmark point under query, with a complexity of $\mathcal{O}(K_{\mathrm{NN}} \log|\Lambda|)$ which amounts to sorting the Riemannian distances $\{\mathrm{dist}_{\mathcal{M}}(\ell_i, \ell_{i'}) \,|\, (i,i') \in \mathcal{I}_{\mathrm{Land}}^2\}$, assumed to have been computed prior to employing Alg. 4. **(ii)** Map those K_{NN} neighbors into the tangent space of the manifold at the landmark point under query with a complexity which scales in the order of $\mathcal{O}(K_{\mathrm{NN}}\mathrm{C}_{\log})$, where $\mathrm{C}_{\log}$ denotes the cost of computing the manifold logarithm mapping, provided in the introductory part of Sec. II. The cost of computing the angle between a vector and a principal subspace in the tangent space, similarly to step 8 of Alg. 4, is of order $\mathcal{O}[K_{\mathrm{NN}}^2 \dim \mathcal{M}]$ due to the operations needed for the SVD of matrix (3), and under the assumption that $K_{\mathrm{NN}} < \dim \mathcal{M}$. **(iii)** Solve a sparse-coding task, of cost C_{sc}, to detect parsimonious relations of the landmark point under query with its K_{NN} nearest neighbors. After the previous steps are run for all landmark points, hierarchical clustering (Louvain method [32, 77]) is applied to an affinity matrix, which gathers all sparse-coding coefficients and angles, to yield clusters with a computational complexity of order $\mathcal{O}(|\Lambda| \log|\Lambda|)$ [32]. The Louvain method belongs to the family of hierarchical-clustering algorithms, which maximize a modularity loss that monitors the intra- and inter-cluster density of links/edges. Overall, the complexity of employing the batch eGCT becomes $\mathcal{O}[|\Lambda|(K_{\mathrm{NN}} \log|\Lambda| + K_{\mathrm{NN}}\mathrm{C}_{\log} + K_{\mathrm{NN}}^2 \dim \mathcal{M} + \mathrm{C}_{\mathrm{sc}})]$, which becomes manageable in the premises of this framework, where by design $|\Lambda| \ll |\mathcal{X}|$.

In contrast to eGCT, the novel scheme of Alg. 4 operates sequentially on the non-landmark points $\mathcal{X} \setminus \Lambda$, without involving any sparse-coding and hierarchical-clustering subtasks. Hence, its computational complexity scales in the order of $\mathcal{O}[K_{\mathrm{NN}}(\log|\mathcal{X}\setminus\Lambda| + \mathrm{C}_{\log} + K_{\mathrm{NN}} \dim \mathcal{M})]$ per feature point, which leads to large savings in computational times as the following Sec. IV demonstrates.

IV. Numerical Tests

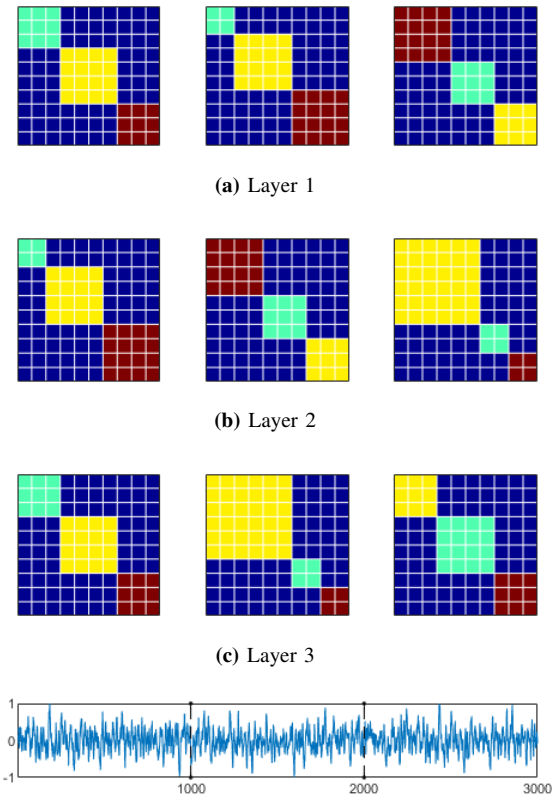
This section validates the proposed framework on synthetic and real data. Tags fastGCT[kARMA], fastGCT[kC], and fastGCT[kPC] indicate the proposed framework whenever features are extracted by multi-kernel ARMA modeling, kernel correlations, and kernel partial correlations, respectively. Apart from the classical K-means, other competing frameworks are: **(i)** Sparse manifold clustering and embedding (SMCE) [33]; **(ii)** interaction K-means with PCA (IKM-PCA) [78]; **(iii)** graph shift operator estimation (GOE) [43]; **(iv)** 4D-windowed tensor approach (4D-WTA) [17]; **(v)** multivariate Granger causality (MVGC) [79, 80]; **(vi)** wavelet–transform based algorithm (Wavelet) [40, 47]; **(vii)** streaming Isomap (S-Isomap) [49]; and **(viii)** RANSAC-based subspace clustering (RANSAC) [74].

SMCE, 4DWTA, Wavelet, S-Isomap, RANSAC, and the classical K-means will be used in state clustering, while SMCE, IKM-PCA, 4DWTA, GOE, S-Isomap, MVGC, RANSAC, Wavelet, and K-means will be employed in community detection. Since none of IKM-PCA, GOA, MVGC and 4DWTA can perform subnetwork-sequence clustering across multiple states, only the results of the proposed framework, SMCE, S-Isomap, RANSAC, and Wavelet are reported. Moreover, since the performance of fastGCT[kC] appears to be better than fastGCT[kPC] in community detection and subnetwork-sequence clustering, fastGCT[kC] results will be reported only for community detection and subnetwork-sequence clustering, while those of fastGCT[kPC] only for state clustering. Although RANSAC can be applied to any type of features, only results for features extracted by multi-kernel ARMA modeling are reported, since results on kernel (partial) correlations were not competitive with their multi-kernel ARMA counterparts.

The assessment criteria for all competing methods are: **(i)** Clustering accuracy, defined as the number of correctly clustered data-points (ground-truth labels are known) over the total number of points; and **(ii)** normalized mutual information (NMI) [81]. In the following discussion, every reported numerical value is the result of the uniform average of 20 independently performed tests. The time duration of computations (run-times) that are reported refer only to clustering operations, while run-times for feature extraction are not included in the subsequent tables/figures. Parameters were carefully tuned so that each employed method achieves its best performance for the scenario at hand. All experiments are performed via Matlab (R2018b) on an i7-7700 CPU with 32GB RAM.

A. Synthetic fMRI Data

FMRI data were generated by the open-source Matlab SimTB toolbox [82]. A multilayer network with 3 layers and 10 nodes, transitioning between 3 distinct network states for every layer, is considered here. Each state corresponds to a connectivity matrix, which is modeled as the superposition of three matrices: **(1)** The ground-truth (noiseless) connectivity matrix, where nodes sharing the same color belong to the same cluster and collaborate to perform a common task (cf. Fig. 5); **(2)** a symmetric matrix whose entries are drawn



(a) Layer 1

(b) Layer 2

(c) Layer 3

(d) BOLD time series of node #6, layer #2 dataset #2

Fig. 5: Synthetic data generated by the Matlab SimTB toolbox [82]. (a)-(c) Noiseless and outlier-free connectivity matrices corresponding to three network layers. Nodes that share the same color cooperate to perform a common task.

independently from a zero-mean Gaussian distribution with standard deviation σ_{n} to model noise; **(3)** a symmetric outlier matrix where 36 entries are equal to μ to capture outlier neural activity. Different states may share different outlier matrices, controlled by μ. Aiming at extensive numerical tests, six datasets were generated (corresponding to the columns of Table I) by choosing six sets of parameters [including the signal-to-noise ratio (SNR)] in the modeling of the connectivity matrices and the SimTB toolbox. Table Supp1 provides parameters of those 6 datasets: Datasets 1, 2, and 3 were created without outliers, while datasets 4, 5 and 6 include outlier matrices with different μs in different states. Driven by the previous connectivity matrices, the SimTB toolbox generates BOLD time series [4], e.g., Fig. 5d. Each state contributes 1000 signal samples, for a total of $3\times1000 = 3000$ samples, to every nodal time series; an example is depicted in Fig. 5d.

1) State Clustering

Table I demonstrates the results of state clustering. The parameters used for eGCT[kARMA] and fastGCT[kARMA] are: $T_{\mathrm{w}} = 50$, $m = 3$, $\rho = 3$, $\tau_{\mathrm{f}} = 300$, $\tau_{\mathrm{b}} = 50$. The multi-kernel function $\kappa(\cdot,\cdot) := 0.6\,\kappa_{\mathrm{G};0.8}(\cdot,\cdot) + 0.4\,\kappa_{\mathrm{L};1}(\cdot,\cdot)$ scored the best results among several other choices. Parameters for fastGCT[kPC] are: $\epsilon = 1$, $T_{\mathrm{w}} = 400$, $\tau_{\mathrm{w}} = 50$. There are 2596 features in extracted by fastGCT[kARMA] and 2551 features extracted by fastGCT[kPC]. One hundred landmark points are selected in fastGCT[kARMA] and fastGCT[kPC]. Table I demonstrates the average run-times of all algorithms over all data samples. Tables I and II show that fastGCT[kARMA] takes only about 10% of the run-times of eGCT[kARMA] with a slightly decreased accuracy. Fig. Supp1 depicts also the standard deviations of the results in Tables I and II, after independent repetitions of the test. To save space, the figures of the standard deviations of the following tests will be omitted.

Among all methods, the batch eGCT[kARMA] scores the highest clustering accuracy and NMI over all six datasets, while fastGCT[kARMA] is one of the top performers among sequential schemes, with much shorter running time than that of eGCT[kARMA]. It can be observed by Table I that the existence of outliers affects negatively the ability of all methods to cluster data. The main reason is that the algorithms tend to detect outliers and gather those in clusters different from the nominal ones. Ways to reject outliers are outside of the scope of this study and will be provided in a future publication.

2) Community Detection

Table III presents results on community detection. The numerical values in Table III stand for the average values over the all states and layers for each one of the datasets. Parameters of eGCT[kARMA] and fastGCT[kARMA] were as follows: $T_{\mathrm{w}} = 30$, $\tau_{\mathrm{w}} = 20$, $m = 3$, $\rho = 2$, $\tau_{\mathrm{f}} = 340$, $\tau_{\mathrm{b}} = 10$, while the multi-kernel function is defined as $\kappa(\cdot,\cdot) := 0.5\,\kappa_{\mathrm{G};0.5}(\cdot,\cdot) + 0.5\,\kappa_{\mathrm{L0};1}(\cdot,\cdot)$. Parameters for fastGCT[kC] were: $\epsilon = 1$, $\tau_{\mathrm{w}} = 30$, $T_{\mathrm{w}} = 400$. Eighty landmark points were selected for fastGCT[kARMA] and fastGCT[kC]. Table III demonstrates that eGCT[kARMA] consistently outperforms all other methods across all datasets and even for the case where outliers contaminate the data. Table IV demonstrates the average run-times of each algorithm, and shows that fastGCT[kARMA] is much faster than its batch predecessors eGCT[kARMA] with a moderate accuracy loss. Fig. Supp2 depicts also the standard deviations of the results of Tables III and IV.

3) Subnetwork-Sequence Clustering

Table V illustrates the results of task clustering on the six synthetic-fMRI datasets. The parameters of eGCT[kARMA], fastGCT[kARMA] were set as follows: $T_{\mathrm{w}} = 50$, $\tau_{\mathrm{w}} = 50$, $m = 3$, $\rho = 2$, $\tau_{\mathrm{f}} = 350$, $\tau_{\mathrm{b}} = 10$. The parameters for fastGCT[kPC] were: $\epsilon = 1$, $\tau_{\mathrm{w}} = 30$, $T_{\mathrm{w}} = 400$. There are 25370 features extracted by fastGCT[kARMA] and 25710 features extracted by fastGCT[kC]. A number of 450 landmark points were selected for fastGCT[kARMA] and fastGCT[kC]. The kernel functions used in eGCT[kARMA], fastGCT[kARMA] and fastGCT[kC] are identical to those employed in Table III. Similarly to the previous cases, eGCT[kARMA] outperforms all other methods across all datasets and scenarios on both clustering accuracy and NMI, and fastGCT[kARMA] exhibits good clustering accuracies with excellent computation-time footprints. Fig. Supp3 depicts also the standard deviations of the results of Table V.

To summarize the results on synthetic data, the batch eGCT[kARMA] outperforms all other methods across all datasets in all clustering tasks, while fastGCT[kARMA] outperforms all competing sequential methods in Tables III and

TABLE I: Synthetic fMRI data: State-clustering results

	Methods	Without Outliers						With Outliers					
		Clustering Accuracy			NMI			Clustering Accuracy			NMI		
		D1	D2	D3	D1	D2	D3	D4	D5	D6	D4	D5	D6
Batch	eGCT[kARMA] [32]	**1**	**0.853**	**0.680**	**1**	**0.811**	**0.597**	**0.966**	**0.785**	**0.593**	**0.934**	**0.676**	**0.395**
	SMCE [33]	0.947	0.780	0.545	0.882	0.701	0.457	0.880	0.714	0.496	0.828	0.574	0.375
	RANSAC [74]	0.929	0.745	0.563	0.859	0.636	0.483	0.907	0.708	0.502	0.846	0.570	0.316
	Kmeans	0.859	0.672	0.395	0.774	0.605	0.311	0.768	0.621	0.337	0.476	0.403	0.148
Sequential	fastGCT[kARMA]	**1**	0.815	**0.656**	**1**	0.762	**0.568**	0.945	0.752	**0.552**	0.912	0.629	**0.347**
	fastGCT[kPC]	**1**	0.791	0.634	**1**	0.705	0.516	0.906	0.733	0.520	0.837	0.609	0.315
	4DWTA [17]	**1**	**0.826**	0.642	**1**	**0.789**	0.522	**0.951**	**0.766**	0.509	**0.927**	**0.635**	0.327
	Wavelet [47]	0.967	0.764	0.552	0.913	0.683	0.483	0.924	0.702	0.471	0.855	0.562	0.309
	S-Isomap [49]	0.983	0.787	0.601	0.935	0.693	0.509	0.936	0.729	0.511	0.867	0.581	0.335

TABLE II: Synthetic fMRI data: State-clustering run-time

	Methods	Runtime (sec)
Batch	eGCT[kARMA] [32]	1421
	SMCE [33]	1186
	RANSAC [74]	102.51
	Kmeans	8.43
Sequential	fastGCT[kARMA]	130.41
	fastGCT[kPC]	157.36
	4DWTA [17]	1477
	Wavelet [47]	95.71
	S-Isomap [49]	206.49

V. In datasets 1–3 and in the absence of outliers ($\mu = 0$), the accuracies and NMI of all methods are influenced mainly by the presence of noise. As expected, the accuracies and NMI decrease as the standard deviation σ_{n} of noise increases. The comparison between the results of datasets 1 and 4 indicates that if σ_{n} remains the same, but the outlier neural activity μ intensifies, the accuracies and NMI of all methods decrease. Similar results can be observed by the comparison between datasets 2 and 5, as well as 3 and 6.

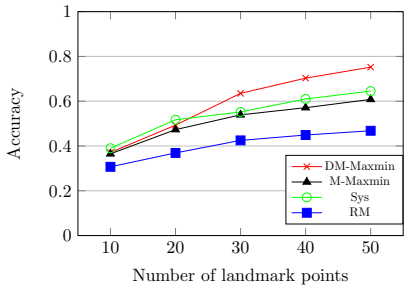
B. Validating Landmark-Selection Schemes

This section illustrates the advantage of DM-Maxmin algorithm (Alg. 3) over the following landmark-point selection methods: **(1)** Random selection (RM); **(2)** Systematic selection (Sys) [72]; and **(3)** Manifold-Maxmin (M-Maxmin) [75], where the Riemannian distance is used in the place of the Euclidean one. To present a challenging task to all these methods, Dataset 5, which includes noise and outliers, is considered for fastGCT[kARMA].

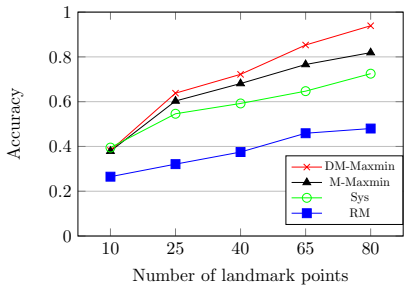
The clustering results for all the aforementioned landmark-selection methods are shown in Tables VII, VIII, IX, and Fig. 6. Although Sys performs well with few landmark points, the accuracy of DM-Maxmin increases faster than other methods as the number of landmark points increases. Besides, comparison between DM-Maxmin and Maxmin shows that the density-based DM-Maxmin is a necessary ingredient for the proposed fastGCT framework.

C. Real EEG Data

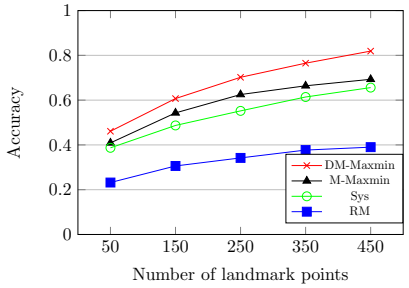
The open-source EEG data [15] were used. EEGs were selected from an archive of neonatal EEG recordings, requested from the clinical team due to suspicion of seizures. All



(a) State clustering



(b) Community detection



(c) Subnetwork-sequence clustering

Fig. 6: Clustering accuracies of different landmark selection methods for all the network-clustering tasks. (a) State clustering; (b) Community detection; (c) Subnetwork-sequence clustering.

recordings were performed at the Helsinki University Hospital, Finland. Each EEG recording lasted approximately one hour; median recording duration was 74 mins. The EEG 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. The data were annotated for seizure and non-seizure

TABLE III: Synthetic fMRI data: Community-detection results

	Methods	Without Outliers						With Outliers					
		Clustering accuracy			NMI			Clustering accuracy			NMI		
		D1	D2	D3	D1	D2	D3	D4	D5	D6	D4	D5	D6
Batch	eGCT[kARMA] [32]	**1**	**0.974**	**0.956**	**1**	**0.931**	**0.882**	**0.983**	**0.939**	**0.833**	**0.940**	**0.836**	**0.755**
	SMCE [33]	0.981	0.916	0.825	0.900	0.842	0.719	0.932	0.826	0.755	0.801	0.663	0.577
	MVGC [79, 80]	0.979	0.903	0.807	0.877	0.810	0.684	0.885	0.773	0.721	0.705	0.591	0.503
	GOE [43]	**1**	0.894	0.786	**1**	0.818	0.695	0.913	0.752	0.696	0.819	0.610	0.427
	RANSAC [74]	0.962	0.885	0.774	0.907	0.809	0.675	0.924	0.815	0.709	0.804	0.683	0.521
	Kmeans	0.895	0.814	0.690	0.802	0.721	0.525	0.836	0.621	0.567	0.645	0.409	0.327
Sequential	fastGCT[kARMA]	**1**	**0.960**	**0.917**	**1**	**0.887**	**0.825**	**0.971**	**0.913**	**0.804**	**0.928**	**0.802**	**0.714**
	fastGCT[kC]	**1**	0.945	0.861	**1**	0.850	0.765	0.941	0.883	0.774	0.890	0.724	0.682
	4DWTA [17]	**1**	0.938	0.887	**1**	0.848	0.742	0.946	0.891	0.780	0.875	0.724	0.665
	Wavelet [40]	0.954	0.870	0.762	0.893	0.805	0.645	0.915	0.762	0.693	0.793	0.647	0.572
	S-Isomap [49]	0.991	0.924	0.861	0.973	0.815	0.703	0.921	0.843	0.727	0.779	0.701	0.539
	IKM-PCA [78]	0.941	0.910	0.789	0.886	0.829	0.681	0.898	0.769	0.705	0.742	0.635	0.494

TABLE IV: Synthetic fMRI data: Community-detection runtime

	Methods	Runtimes (sec)
Batch	eGCT[kARMA] [32]	1835
	SMCE [33]	3976
	MVGC [79, 80]	2087
	GOE [43]	46.46
	RANSAC [74]	173.38
	Kmeans	11.38
Sequential	fastGCT[kARMA]	217.76
	fastGCT[kC]	249.22
	4DWTA [17]	1361
	Wavelet [40]	120.65
	S-Isomap [49]	387.79
	IKM-PCA [78]	60.35

states by experts. Each neonate defines a layer in the present context of multilayer networks: Ten neonates are considered, so that the number of layers is 10. Each layer presents a 40sec segment, so that the length of time series is $40 \cdot 256 = 10,240$. The network has 18 nodes, i.e., $|\mathcal{N}| := 18$.

Methods eGCT[kARMA], fastGCT[kARMA], Fast[kPC], Wavelet, S-Isomap, SMCE, RANSAC, and K-means were validated. 4DWTA did not perform well on this dataset, and its results are not reported. Parameters of eGCT[kARMA] and fastGCT[kARMA] are defined as: $T_{\rm w} = 800$, $m = 3$, $\rho = 2$, $\tau_{\rm f} = 5200$, $\tau_{\rm b} = 100$. The kernel function is defined as $\kappa(\cdot,\cdot) := 0.5\,\kappa_{{\rm G};0.37}(\cdot,\cdot) + 0.5\,\kappa_{{\rm L};0.85}(\cdot,\cdot)$. Parameters of fastGCT[kPC] are: $\epsilon = 0.8$, $\tau_{\rm w} = 100$, $T_{\rm w} = 6000$. The number of landmark points in fastGCT[kARMA] and fastGCT[kPC] is set equal to 200. Due to the sliding-window implementation in the proposed framework, there are cases where the sliding window captures samples from both seizure and non-seizure state. The features extracted from those cases are labeled as cluster #3. The seizure and non-seizure state clustering accuracies and run-times are demonstrated in Table X and XI. Fig. Supp4 depicts also the standard deviations of the results of Table X.

D. Real fMRI Data

The community-detection capability of the proposed framework is tested on multilayer networks formed by the fMRI data of 99 subjects, taken from the S1200 data-set of the Human Connectome Project (HCP) [84]. Rather than providing definite answers to open neuroscience problems, the following discussion aims at revealing only a small part of the complexity of cognitive systems, and at highlighting the rich potential of the proposed framework to serve as a data-analytic toolbox for neuroscience research.

Per subject, features are extracted from the temporal activity of 116 brain (structural) regions, defined via the automated anatomical labeling (AAL) atlas [85]. Those 116 regions are considered as the nodes of the brain network; see Fig. 7a. The communities of the network are defined by the 7-network parcellation scheme of the Schaefer-100 atlas, which was based on the resting-state fMRI data of $1,489$ healthy subjects [86]. Here, nine communities are considered and listed in the second row of Table XII. The definition of node labels, i.e., assignment of a node into a community, was based on the proximity, via the Euclidean distance, of the centroid of an AAL region (node) from the corresponding community in the Schaefer-100 atlas. This label assignment is color coded and shown in Figs. 7a and 7b.

It is worth stressing here that the ground-truth node labels are *unknown.* The earlier defined labels may not correspond to the actual ones, since brain regions (nodes) may be actually associated with different communities; indeed, recent studies suggest that subcortical and cerebellum are associated also with other communities [87–89]. Also, it has been observed that the same brain region (node) across different people is not necessarily associated with the same community: recent studies suggest that a region, which is part of a community in a group atlas, becomes part of another community in an atlas tailored to a specific subject, e.g., [90].

With regards to time-series pre-processing and cleansing, the data of each voxel are normalized by firstly subtracting the temporal mean, then applying linear regression on the data and regressing out global signal [91]. The temporal activity of a node was computed by averaging the signal over all voxels within the region (node). Only the part of cleansed volume data in single run with left-to-right phase encoding direction was employed. Specifically, motion outliers were used to estimate framewise displacement (FD) [92], and volumes with FD $>$ 0.2mm were removed from further analysis.

Each of the 99 subjects of the HCP data-set is considered as a layer of the multilayer network. Nodes (brain regions)

TABLE V: Synthetic fMRI data: Subnetwork-sequence clustering results

	Methods	Without Outliers						With Outliers					
		Clustering accuracy			NMI			Clustering accuracy			NMI		
		D1	D2	D3	D1	D2	D3	D4	D5	D6	D4	D5	D6
Batch	eGCT[kARMA] [32]	**1**	**0.890**	**0.787**	**1**	**0.826**	**0.703**	**0.941**	**0.819**	**0.647**	**0.881**	**0.671**	**0.513**
	SMCE [33]	0.916	0.704	0.600	0.794	0.585	0.417	0.829	0.607	0.467	0.646	0.355	0.273
	RANSAC [74]	0.920	0.729	0.611	0.809	0.603	0.423	0.871	0.675	0.535	0.726	0.459	0.340
Sequential	fastGCT[kARMA]	**0.971**	**0.825**	**0.702**	**0.904**	**0.740**	**0.609**	**0.921**	**0.745**	**0.628**	**0.837**	**0.615**	**0.462**
	fastGCT[kC]	0.953	0.794	0.681	0.862	0.667	0.553	0.884	0.718	0.592	0.774	0.562	0.413
	Wavelet [40, 47]	0.923	0.712	0.569	0.812	0.593	0.360	0.802	0.574	0.421	0.619	0.308	0.197
	S-Isomap [49]	0.928	0.735	0.651	0.821	0.619	0.497	0.852	0.656	0.523	0.715	0.427	0.301

TABLE VI: Synthetic fMRI data: subnetwork-sequence tracking runtime

	Methods	Runtime (sec)
Batch	eGCT[kARMA] [32]	16 219
	SMCE [33]	35 875
	RANSAC [74]	1723
Sequential	fastGCT[kARMA]	2142
	fastGCT[kC]	2477
	Wavelet [40]	1378
	S-Isomap [49]	3605

TABLE VII: Synthetic fMRI Data: Landmark selection methods in state clustering

Methods	Number of landmark points				
	10	20	30	40	50
DM-Maxmin	0.371	0.493	**0.635**	**0.703**	**0.752**
M-Maxmin	0.365	0.473	0.539	0.571	0.608
Sys	**0.390**	**0.517**	0.552	0.610	0.645
RM	0.307	0.369	0.425	0.449	0.468

TABLE VIII: Synthetic fMRI Data: Landmark selection methods in community detection

Methods	Number of landmark points				
	10	25	40	65	80
DM-Maxmin	0.382	**0.638**	**0.722**	**0.853**	**0.939**
M-Maxmin	0.379	0.603	0.681	0.766	0.819
Sys	**0.395**	0.546	0.592	0.647	0.725
RM	0.265	0.321	0.375	0.459	0.480

TABLE IX: Synthetic fMRI Data: Landmark selection methods in subnetwork-sequence clustering

Methods	Number of landmark points				
	50	150	250	350	450
DM-Maxmin	**0.461**	**0.607**	**0.702**	**0.765**	**0.819**
M-Maxmin	0.409	0.543	0.625	0.664	0.698
Sys	0.387	0.487	0.552	0.614	0.656
RM	0.232	0.306	0.342	0.377	0.390

in all layers are clustered in communities by the competing clustering schemes. Per community (see Table XII), the ratio of the total number of correctly clustered nodes in all layers over the product of the number of layers with the cardinality of the community is recorded. The averages of those ratios across 10 independently performed tests are coined "fitting rates" and are listed in Table XII. Fig. Supp5 depicts also the standard deviation of those results. Since the number of sought communities is known a-priori, the GCT algorithm proposed in [17] with kernel ARMA features can be also used here, where instead of the hierarchical Louvain method in eGCT, spectral clustering is applied. The highest score per community is marked with bold-faced fonts. The parameters of fastGCT[kARMA] were tuned to obtain 9 communities: $T_{\mathrm{w}} := 100$, $m := 3$, $\rho := 3$, $\tau_f := 550$, $\tau_b := 10$. The kernel function was defined as $\kappa(\cdot,\cdot) := 0.7\,\kappa_{\mathrm{G};0.5}(\cdot,\cdot) + 0.3\,\kappa_{\mathrm{L};0.1}(\cdot,\cdot)$. The parameters of fastGCT[kC] were also tuned to obtain 9 clusters: $T_{\mathrm{w}} := 200$, $\tau_{\mathrm{w}} := 180$, $\epsilon := 1$. The kernel function of fastGCT[kC] was defined as $\kappa(\cdot,\cdot) := 0.35\,\kappa_{\mathrm{G};0.8}(\cdot,\cdot) + 0.25\,\kappa_{\mathrm{G};0.5}(\cdot,\cdot) + 0.4\,\kappa_{\mathrm{L};0.5}(\cdot,\cdot)$. The number of landmark points in fastGCT[kARMA] and fastGCT[kC] is set equal to 660. The low values of fitting rates, recorded by *all* clustering methods for several communities in Table XII, together with the observation that the ground-truth node labels are unknown, indicate that the aforementioned node-label assignment might not be the adequate one for this data-set, and that there may be another way of parcellating the brain-network nodes of these 99 subjects in communities. Clearly, these results illustrate the complexity of brain-network problems and the variability in how cognitive systems are organized. Nevertheless, both GCT[kARMA] and fastGCT[kARMA] appear to score the highest fitting rates across all competing methods within their respective groups. The color-coded results of GCT[kARMA] and fastGCT[kARMA] are depicted in Figs. 7c and 7e, respectively, where only the mis-clustered nodes are shown. A deeper and more extensive discussion on this specific task is beyond the scope of this Journal, and will be reported in publication venues which are more tailored to neuroscience and cognitive-systems research.

V. Conclusions

This paper introduced a computationally efficient framework to address all possible clustering tasks, i.e., state clustering, community detection, and subnetwork-sequence identification, in dynamic multilayer networks where nodes are annotated with real-valued time-series. A full learning path was offered, which starts from low-level feature extraction and reaches up to network clustering. Features in Riemannian

TABLE X: Real EEG data: State-clustering results

	Methods	Clustering accuracy	NMI
Batch	eGCT[kARMA] [32]	**1**	**1**
	SMCE [33]	0.965	0.910
	RANSAC [74]	0.932	0.873
	Kmeans	0.845	0.725
Sequential	fastGCT[kARMA]	**1**	**1**
	fastGCT[kPC]	**1**	**1**
	Wavelet [40]	0.925	0.856
	S-Isomap [49]	0.986	0.951

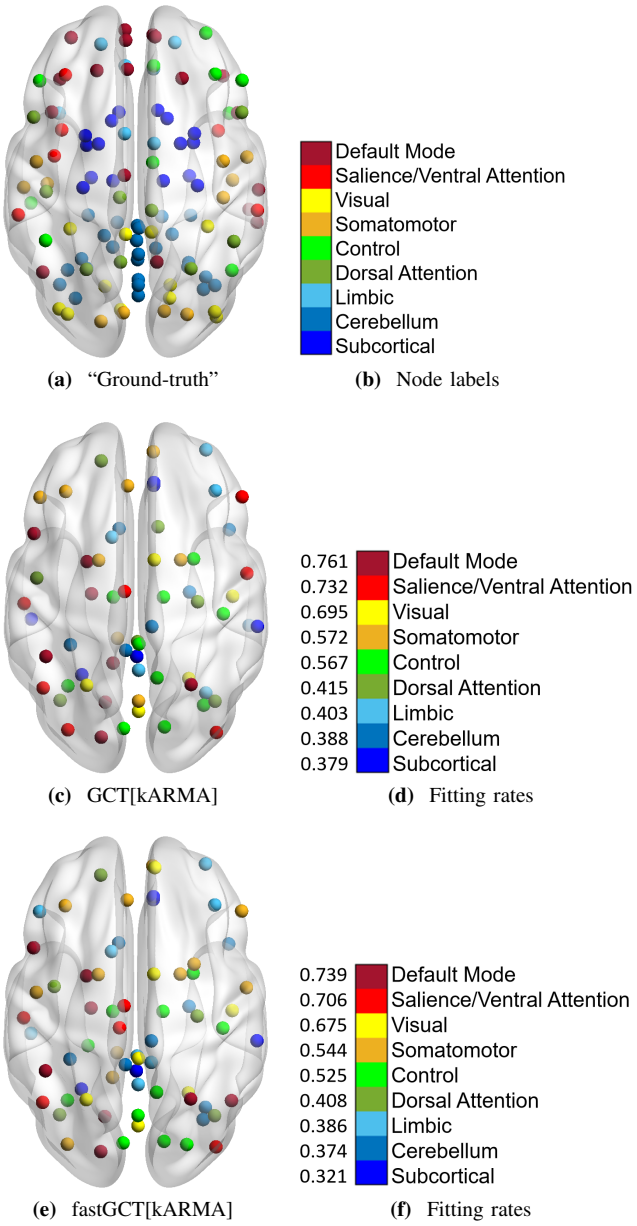
TABLE XI: Real EEG data: State-clustering run-time

	Methods	Runtime (sec)
Batch	eGCT[kARMA] [32]	42 954
	SMCE [33]	38 690
	RANSAC [74]	3395
	Kmeans	183
Sequential	fastGCT[kARMA]	4748
	fastGCT[kPC]	5064
	Wavelet [47]	3144
	S-Isomap [49]	6388

manifolds were considered, anticipating the emergence of latent data patterns, which may not be easily observed in classical Euclidean feature spaces. The manuscript provided also non-trivial feature-extraction examples: Kernel autoregressive-moving-average modeling and kernel (partial) correlations are used to generate features in the Riemannian manifolds of Grassmann and positive-(semi)definite matrices, respectively, where kernels were used to capture non-linear correlations among nodal time-series. Clusters were viewed as submanifolds via the recently introduced Riemannian multi-manifold modeling, and a fast sequential clustering scheme was established that takes advantage of both Riemannian distances and the angular information within tangent spaces of the manifolds. The case of brain networks was considered to solidify arguments. Extensive numerical tests on synthetic and real fMRI/EEG data were used to demonstrate that the proposed framework outperforms several state-of-the-art sequential manifold-learning and brain-network-clustering schemes, while staying competitive against much computationally heavier batch methods. Current research effort includes the performance analysis of the sequential clustering scheme, as well as the development of new Riemannian-manifold techniques to reject network-wide outliers. The Matlab software code that supports the proposed sequential-clustering framework will be made available online, via GitHub, as soon as the manuscript receives publication permissions.

VI. Acknowledgements

The brain network data [84] were provided by the Human Connectome Project, MGH-USC Consortium (Principal Investigators: Bruce R. Rosen, Arthur W. Toga and Van Wedeen; U01MH093765) funded by the NIH Blueprint Initiative for Neuroscience Research grant; the National Institutes of Health grant P41EB015896; and the Instrumentation Grants S10RR023043, 1S10RR023401, 1S10RR019307.



(a) "Ground-truth" **(b)** Node labels

(c) GCT[kARMA] **(d)** Fitting rates

(e) fastGCT[kARMA] **(f)** Fitting rates

Fig. 7: Brain network with 116 nodes (AAL atlas) [85]. The nodes are parcellated in nine communities according to the Schaefer-100 atlas [86]: (a) color-coded node labels (cf. Sec. IV-D); (b) "ground-truth" communities of cognitive system with AAL atlas; (c) subnetwork-sequence clustering results of GCT[kARMA] in layer #50 (only the nodes that were not correctly clustered are plotted; for example, the cyan node in the top right of Fig. 7c was assigned to be in the Limbic community, whereas it should have been assigned to the Control community); (d) fitting rates of GCT[kARMA]; (e) subnetwork-sequence clustering results of fastGCT[kARMA] in layer #50 (only the nodes that were not correctly clustered; for example, the green node in the top left of Fig. 7e was assigned to be in the Dorsal Attention community, whereas it should have been assigned to the Limbic community); (f) fitting rates of fastGCT[kARMA]; (cf. Table XII).

Appendix A
Reproducing Kernel Hilbert Spaces

A reproducing kernel Hilbert space $\mathcal{H}$, equipped with inner product $\langle \cdot \mid \cdot \rangle_{\mathcal{H}}$, is a functional space where each point $g \in \mathcal{H}$ is a function $g : \mathbb{R}^q \to \mathbb{R} : \mathbf{y} \mapsto g(\mathbf{y})$, for some $q \in \mathbb{Z}_{>0}$, s.t. the mapping $g \mapsto g(\mathbf{y})$ is continuous, for any choice of $\mathbf{y}$ [65, 66, 93]. There exists a kernel function

TABLE XII: Real fMRI data: Community detection

	Methods	Fitting rates								
		Cerebellum	Control	Default Mode	Dorsal Attention	Limbic	Ventral Attention	Somatomotor	Subcortical	Visual
Batch	GCT[kARMA] [32]	0.388	0.567	**0.761**	0.415	0.403	**0.732**	0.572	0.379	**0.695**
	eGCT[kARMA] [32]	0.383	0.540	0.757	0.439	0.397	0.720	0.565	0.367	0.681
	SMCE [33]	0.319	**0.582**	0.647	0.367	0.330	0.695	0.528	**0.418**	0.647
	MVGC [79, 80]	**0.445**	0.471	0.638	0.408	0.295	0.650	0.536	0.394	0.624
	GOE [43]	0.371	0.450	0.626	**0.461**	0.435	0.621	0.479	0.345	0.592
	RANSAC [83]	0.355	0.387	0.596	0.412	**0.429**	0.627	**0.581**	0.275	0.533
	Kmeans	0.247	0.329	0.518	0.332	0.226	0.575	0.392	0.287	0.461
Sequential	fastGCT[kARMA]	0.374	0.525	**0.739**	0.408	0.386	**0.706**	0.544	0.321	**0.675**
	fastGCT[kC]	0.342	0.493	0.712	0.415	0.374	0.686	0.529	0.347	0.639
	4DWTA [17]	0.439	**0.541**	0.659	0.423	**0.417**	0.652	0.577	0.304	0.593
	Wavelet [40, 47]	0.326	0.414	0.683	**0.447**	0.274	0.679	0.473	**0.349**	0.625
	S-Isomap [49]	**0.487**	0.508	0.618	0.367	0.391	0.640	**0.615**	0.296	0.514
	IKM-PCA [78]	0.361	0.392	0.621	0.424	0.360	0.638	0.542	0.316	0.507

$\kappa(\cdot,\cdot) : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$, unique to $\mathcal{H}$, s.t. $\varphi(\mathbf{y}) \coloneqq \kappa(\mathbf{y},\cdot) \in \mathcal{H}$ and $g(\mathbf{y}) = \langle g \mid \varphi(\mathbf{y})\rangle_{\mathcal{H}}$, for any $g \in \mathcal{H}$ and any $\mathbf{y} \in \mathbb{R}^q$ [66, 93]. The latter property is the reason for calling kernel κ reproducing, and yields the celebrated "kernel trick": $\kappa(\mathbf{y}_1,\mathbf{y}_2) = \langle \kappa(\mathbf{y}_1,\cdot) \mid \kappa(\mathbf{y}_2,\cdot)\rangle_{\mathcal{H}} = \langle \varphi(\mathbf{y}_1) \mid \varphi(\mathbf{y}_2)\rangle_{\mathcal{H}}$, for any $\mathbf{y}_1,\mathbf{y}_2 \in \mathbb{R}^q$.

Popular examples of reproducing kernels are: **(i)** The linear $\kappa_{\text{lin}}(\mathbf{y}_1,\mathbf{y}_2) \coloneqq \mathbf{y}_1^{\intercal}\mathbf{y}_2$, where space $\mathcal{H}$ is nothing but $\mathbb{R}^q$; **(ii)** the Gaussian $\kappa_{\text{G};\sigma}(\mathbf{y}_1,\mathbf{y}_2) \coloneqq \exp[-\|\mathbf{y}_1-\mathbf{y}_2\|^2/(2\sigma^2)]$, where $\sigma \in \mathbb{R}_{>0}$ and $\|\cdot\|$ is the standard Euclidean norm. In this case, $\mathcal{H}$ is infinite dimensional [66]; **(iii)** the polynomial $\kappa_{\text{poly};r}(\mathbf{y}_1,\mathbf{y}_2) \coloneqq (\mathbf{y}_1^{\intercal}\mathbf{y}_2+1)^r$, for some parameter $r \in \mathbb{Z}_{>0}$. There are several ways of generating reproducing kernels via certain operations on well-known kernel functions such as convex combinations, products, etc. [65]. For example, given a set of parameters $\{\sigma_j\}_{j=1}^J$, with $J \in \mathbb{Z}_{>0}$, used to cover an interval where the "ideal" σ_* is known to belong to, and the Gaussian kernels $\{\kappa_{\text{G};\sigma_j}(\cdot,\cdot)\}_{j=1}^J$, a reproducing kernel function can be then defined as the convex combination $\kappa(\cdot,\cdot) \coloneqq \sum_{j=1}^J \gamma_j \kappa_{\text{G};\sigma_j}(\cdot,\cdot)$, where $\{\gamma_j\}_{j=1}^J$ are convex weights, i.e., non-negative real numbers s.t. $\sum_{j=1}^J \gamma_j = 1$ [65].

APPENDIX B
THE PRODUCT $\otimes_{\mathcal{H}}$

Define $\mathcal{H}^p$, for some $p \in \mathbb{Z}_{>0}$, as the space whose points take the following form: $\boldsymbol{g} \coloneqq [g_1,\ldots,g_p]^{\intercal} \in \mathcal{H}^p$ s.t. $g_j \in \mathcal{H}$, $\forall j \in \overline{1,p}$, where $\overline{1,p}$ is a compact notation for $\{1,\ldots,p\}$. For $p' \in \mathbb{Z}_{>0}$ and given a matrix $\mathbf{A} \coloneqq [a_{ij}] \in \mathbb{R}^{p'\times p}$, the product $\mathbf{A}\boldsymbol{g} \in \mathcal{H}^{p'}$ stands for the vector-valued function whose ith entry is $\sum_{j=1}^p a_{ij}g_j$. Similarly, define $\mathcal{H}^{p_1\times p_2}$, for some $p_1,p_2 \in \mathbb{Z}_{>0}$, as the space comprising all

$$\boldsymbol{\mathcal{G}} \coloneqq \begin{bmatrix} g_{11} & \cdots & g_{1p_2} \\ \vdots & \ddots & \vdots \\ g_{p_1 1} & \cdots & g_{p_1 p_2} \end{bmatrix} \in \mathcal{H}^{p_1\times p_2},$$

s.t. $g_{ij} \in \mathcal{H}$, $\forall i \in \overline{1,p_1}$, $\forall j \in \overline{1,p_2}$. Moreover, given $\boldsymbol{\mathcal{G}} \in \mathcal{H}^{p_1\times p}$ and $\boldsymbol{\mathcal{G}}' \in \mathcal{H}^{p\times p_2}$, define the "product" $\boldsymbol{\mathcal{G}} \otimes_{\mathcal{H}} \boldsymbol{\mathcal{G}}'$ as the $p_1 \times p_2$ matrix:

$$\boldsymbol{\mathcal{G}} \otimes_{\mathcal{H}} \boldsymbol{\mathcal{G}}' \coloneqq \begin{bmatrix} \sum_{l=1}^p \langle g_{1l} \mid g'_{l1}\rangle_{\mathcal{H}} & \cdots & \sum_{l=1}^p \langle g_{1l} \mid g'_{lp_2}\rangle_{\mathcal{H}} \\ \vdots & \ddots & \vdots \\ \sum_{l=1}^p \langle g_{p_1 l} \mid g'_{l1}\rangle_{\mathcal{H}} & \cdots & \sum_{l=1}^p \langle g_{p_1 l} \mid g'_{lp_2}\rangle_{\mathcal{H}} \end{bmatrix}.$$

In the case where $g_{il} \coloneqq \varphi(\mathbf{y}_{il}) = \kappa(\mathbf{y}_{il},\cdot)$ and $g'_{lj} \coloneqq \varphi(\mathbf{y}'_{lj}) = \kappa(\mathbf{y}'_{lj},\cdot)$, for some $\mathbf{y}_{il},\mathbf{y}'_{lj}$, as in (2), then the kernel trick suggests that the previous product simplifies to

$$\boldsymbol{\mathcal{G}} \otimes_{\mathcal{H}} \boldsymbol{\mathcal{G}}' = \begin{bmatrix} \sum_{l=1}^p \kappa(\mathbf{y}_{1l},\mathbf{y}'_{l1}) & \cdots & \sum_{l=1}^p \kappa(\mathbf{y}_{1l},\mathbf{y}'_{lp_2}) \\ \vdots & \ddots & \vdots \\ \sum_{l=1}^p \kappa(\mathbf{y}_{p_1 l},\mathbf{y}'_{l1}) & \cdots & \sum_{l=1}^p \kappa(\mathbf{y}_{p_1 l},\mathbf{y}'_{lp_2}) \end{bmatrix}.$$

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] E. Bullmore and O. Sporns, "Complex brain networks: Graph theoretical analysis of structural and functional systems," *Nature Reviews Neuroscience*, vol. 10, no. 3, pp. 186–198, Feb. 2009.

[4] 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.

[5] C. Zhang *et al.*, "Generalized latent multi-view subspace clustering," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 42, no. 1, pp. 86–99, 2018.

[6] A. Vörös and T. A. Snijders, "Cluster analysis of multiplex networks: Defining composite network measures," *Social Networks*, vol. 49, pp. 93–112, 2017.

[7] M. Kivelä *et al.*, "Multilayer networks," *J. Complex Networks*, vol. 2, no. 3, pp. 203–271, 2014.

[8] R. F. Betzel and D. S. Bassett, "Multi-scale brain networks," *NeuroImage*, vol. 160, pp. 73–83, 2017.

[9] 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.

[10] A. Liu, X. Chen, X. Dan, M. J. McKeown, and Z. J. Wang, "A combined static and dynamic model for resting-state brain connectivity networks," *IEEE J. Selected Topics in Signal Process.*, vol. 10, no. 7, pp. 1172–1181, 2016.

[11] C. J. Stam *et al.*, "Graph theoretical analysis of magnetoencephalographic functional connectivity in Alzheimer's disease," *Brain*, vol. 132, no. 1, pp. 213–224, 2009.

[12] M. D. Greicius *et al.*, "Resting-state functional connectivity in major depression: Abnormally increased contributions from subgenual cingulate cortex and thalamus," *Biological Psychiatry*, vol. 62, no. 5, pp. 429–437, 2007.

[13] S. J. Broyd *et al.*, "Default-mode brain dysfunction in mental disorders: A systematic review," *Neuroscience & Biobehavioral Reviews*, vol. 33, no. 3, pp. 279–296, 2009.

[14] G. Gifford *et al.*, "Resting state fMRI based multilayer network configuration in patients with schizophrenia," *NeuroImage: Clinical*, p. 102 169, 2020.

[15] 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.

[16] R. F. Betzel *et al.*, "The community structure of functional brain networks exhibits scale-specific patterns of inter-and intra-subject variability," *NeuroImage*, vol. 202, p. 115 990, 2019.

[17] E. Al-Sharoa, M. Al-Khassaweneh, and S. Aviyente, "Tensor based temporal and multi-layer community detection for studying brain dynamics during resting state fMRI," *IEEE Trans. Biomedical Engineering*, vol. 66, no. 3, pp. 695–709, 2019.

[18] M. Vaiana, E. M. Goldberg, and S. F. Muldoon, "Optimizing state change detection in functional temporal networks through dynamic community detection," *J. Complex Networks*, Dec. 2018.

[19] L. Bréchet *et al.*, "Capturing the spatiotemporal dynamics of self-generated, task-initiated thoughts with EEG and fMRI," *NeuroImage*, vol. 194, pp. 82–92, 2019.

[20] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*, 2nd ed. Academic Press, 2020.

[21] P. T. Fletcher and S. Joshi, "Riemannian geometry for the statistical analysis of diffusion tensor data," *Signal Processing*, vol. 87, no. 2, pp. 250–262, 2007.

[22] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa, "Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition," *IEEE Trans. Pattern Analysis Machine Intell.*, vol. 33, no. 11, pp. 2273–2286, 2011.

[23] L. W. Tu, *An Introduction to Manifolds*. Springer, 2008.

[24] J. W. Robbin and D. A. Salamon, *Introduction to Differential Geometry*. 2019. [Online]. Available: https://people.math.ethz.ch/~salamon/PREPRINTS/diffgeo.pdf.

[25] J. Nash, "The imbedding problem for Riemannian manifolds," *Annals of Mathematics*, pp. 20–63, 1956.

[26] Q. Han and J. X. Hong, *Isometric Embedding of Riemannian Manifolds in Euclidean Spaces*, v. 13. American Mathematical Society, 2006.

[27] A. Machado and I. Salavessa, "Grassmannian manifolds as subsets of Euclidean spaces," *Res. Notes in Math*, vol. 131, pp. 85–102, 1985.

[28] K. Slavakis, G. B. Giannakis, and G. Mateos, "Modeling and optimization for big data analytics: (Statistical) learning tools for our era of data deluge," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 18–31, Sep. 2014.

[29] R. Zarei, J. He, S. Siuly, and Y. Zhang, "A PCA aided cross-covariance scheme for discriminative feature extraction from EEG signals," *Computer Methods and Programs in Biomedicine*, vol. 146, pp. 47–57, 2017.

[30] N. Mammone, C. Ieracitano, H. Adeli, A. Bramanti, and F. C. Morabito, "Permutation Jaccard distance-based hierarchical clustering to estimate EEG network density modifications in MCI subjects," *IEEE Trans. Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 5122–5135, 2018.

[31] K. Slavakis *et al.*, "Clustering brain-network time series by Riemannian geometry," *IEEE Trans. Signal and Information Processing over Networks*, vol. 4, no. 3, pp. 519–533, 2018.

[32] C. Ye *et al.*, "Network clustering via kernel-ARMA modeling and the Grassmannian: The brain-network case," *arXiv:2002.09943*, 2020.

[33] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[34] K. V. Haak, A. F. Marquand, and C. F. Beckmann, "Connectopic mapping with resting-state fMRI," *NeuroImage*, vol. 170, pp. 83–94, 2018.

[35] H. Shen, L. Wang, Y. Liu, and D. Hu, "Discriminative analysis of resting-state functional connectivity patterns of schizophrenia using low dimensional embedding of fMRI," *NeuroImage*, vol. 49, no. 4, pp. 3110–3121, 2010.

[36] H. J. Kim *et al.*, "Canonical correlation analysis on SPD(n) manifolds," in *Riemannian Computing in Computer Vision*, Springer, 2016, pp. 69–100.

[37] G. Duan, W. Hu, and Z. Zhang, "A novel multilayer data clustering framework based on feature selection and modified k-means algorithm," *Int J. Signal Process. Image Process. Pattern Recognit.*, vol. 9, no. 4, pp. 81–90, 2016.

[38] M. Abdolali, N. Gillis, and M. Rahmati, "Scalable and robust sparse subspace clustering using randomized clustering and multilayer graphs," *Signal Processing*, vol. 163, pp. 166–180, 2019.

[39] Z. Jiao, X. Ming, Y. Cao, C. Cheng, and S.-H. Wang, "Module partitioning for multilayer brain functional network using weighted clustering ensemble," *J. Ambient Intelligence and Humanized Computing*, pp. 1–11, 2019.

[40] A. Medda *et al.*, "Wavelet-based clustering of resting state MRI data in the rat," *Magnetic Resonance Imaging*, vol. 34, no. 1, pp. 35–43, 2016.

[41] D. Wang, H. Wang, and X. Zou, "Identifying key nodes in multilayer networks based on tensor decomposition," *Chaos*, vol. 27, no. 6, p. 063 108, 2017.

[42] D. Xie, W. Xia, Q. Wang, Q. Gao, and S. Xiao, "Multi-view clustering by joint manifold learning and tensor nuclear norm," *Neurocomputing*, vol. 380, pp. 105–114, 2020.

[43] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 467–483, 2017.

[44] F. Liu, D. Choi, L. Xie, and K. Roeder, "Global spectral clustering in dynamic networks," *Proc. National Academy of Sciences*, vol. 115, no. 5, pp. 927–932, 2018.

[45] D. R. DeFord and S. D. Pauls, "Spectral clustering methods for multiplex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 533, p. 121 949, 2019.

[46] A. Abuarqoub *et al.*, "Dynamic clustering and management of mobile wireless sensor networks," *Computer Networks*, vol. 117, pp. 62–75, 2017.

[47] A. Dal Col *et al.*, "Wavelet-based visual analysis of dynamic networks," *IEEE Trans. Visualization and Computer Graphics*, vol. 24, no. 8, pp. 2456–2469, 2018.

[48] 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.

[49] S. Mahapatra and V. Chandola, "S-Isomap++: Multi manifold learning from streaming data," in *IEEE International Conference on Big Data*, IEEE, 2017, pp. 716–725.

[50] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin, "Detecting communities and their evolutions in dynamic social networks—a Bayesian approach," *Machine Learning*, vol. 82, no. 2, pp. 157–189, 2011.

[51] K. S. Xu and A. O. Hero, "Dynamic stochastic blockmodels for time-evolving social networks," *IEEE J. Selected Topics in Signal Processing*, vol. 8, no. 4, pp. 552–562, 2014.

[52] C.-D. Wang, J.-H. Lai, D. Huang, and W.-S. Zheng, "SVStream: A support vector-based algorithm for clustering data streams," *IEEE Trans. Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1410–1424, 2011.

[53] D. Huang, C.-D. Wang, and J.-H. Lai, "Locally weighted ensemble clustering," *IEEE Trans. Cybernetics*, vol. 48, no. 5, pp. 1460–1473, 2017.

[54] L. Gauvin, A. Panisson, and C. Cattuto, "Detecting the community structure and activity patterns of temporal networks: A non-negative tensor factorization approach," *PloS One*, vol. 9, no. 1, 2014.

[55] K. Tu, B. Ribeiro, A. Swami, and D. Towsley, "Temporal clustering in dynamic networks with tensor decomposition," *arXiv:1605.08074*, 2016.

[56] M. Vaiana, E. M. Goldberg, and S. F. Muldoon, "Optimizing state change detection in functional temporal networks through dynamic community detection," *J. Complex Networks*, vol. 7, no. 4, pp. 529–553, 2019.

[57] Y. Yang, L. Wang, Y. Lei, Y. Zhu, and H. Shen, "Manifold learning of dynamic functional connectivity reliably identifies functionally consistent coupling patterns in human brains," *Brain Sciences*, vol. 9, no. 11, p. 309, 2019.

[58] J. Guo, W. Yin, Y. Sun, and Y. Hu, "Multi-view subspace clustering with block diagonal representation," *IEEE Access*, vol. 7, pp. 84 829–84 838, 2019.

[59] X. Wang, K. Slavakis, and G. Lerman, "Multi-manifold modeling in non-Euclidean spaces," in *Proc. AISTATS*, San Diego: California: USA, May 2015.

[60] ——, "Riemannian multi-manifold modeling," *arXiv:1410.0095*, 2014.

[61] J. Hamm and D. D. Lee, "Grassmann discriminant analysis: A unifying view on subspace-based learning," in *Proc. International Conference on Machine Learning*, 2008, pp. 376–383.

[62] K. Gallivan, A. Srivastava, X. Liu, and P. V. Dooren, "Efficient algorithms for inferences on Grassmann manifolds," in *Proc. SSP*, 2003, pp. 315–318.

[63] O. Tuzel, F. Porikli, and P. Meer, "Human detection via classification on Riemannian manifolds," in *IEEE Conf. Computer Vision & Pattern Recognition*, 2007, pp. 1–8.

[64] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, "Geometric means in a novel vector space structure on symmetric positive-definite matrices," *SIAM J. Matrix Analysis and Applications*, vol. 29, no. 1, pp. 328–347, 2007.

[65] B. Scholkopf and A. J. Smola, *Learning with Kernels*. MIT Press, 2001.

[66] K. Slavakis, P. Bouboulis, and S. Theodoridis, "Online learning in reproducing kernel Hilbert spaces," in *Academic Press Library in Signal Processing: Signal Processing Theory and Machine Learning*, vol. 1, 2014, pp. 883–987.

[67] A. Ben-Israel and T. N. Greville, *Generalized Inverses: Theory and Applications*. Springer Science & Business Media, 2003, vol. 15.

[68] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical LASSO," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[69] C.-J. Hsieh, I. S. Dhillon, P. K. Ravikumar, and M. A. Sustik, "Sparse inverse covariance matrix estimation using quadratic approximation," in *Advances in Neural Information Processing Systems*, 2011, pp. 2330–2338.

[70] N. Masuda, M. Sakaki, T. Ezaki, and T. Watanabe, "Clustering coefficients for correlation networks," *Frontiers in Neuroinformatics*, vol. 12, p. 7, 2018.

[71] K. Dadi *et al.*, "Benchmarking functional connectome-based predictive models for resting-state fMRI," *NeuroImage*, vol. 192, pp. 115–134, 2019.

[72] W. M. Cates, "Systematic selection and implementation of graphical user interface metaphors," *Computers & Education*, vol. 38, no. 4, pp. 385–397, 2002.

[73] J.-F. Richard and W. Zhang, "Efficient high-dimensional importance sampling," *J. Econometrics*, vol. 141, no. 2, pp. 1385–1411, 2007.

[74] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[75] V. De Silva and J. B. Tenenbaum, "Sparse multidimensional scaling using landmark points," Technical report, Stanford University, Tech. Rep., 2004.

[76] I. T. Jolliffe, *Principal Component Analysis*, ser. Springer Series in Statistics. New York: Springer-Verlag, 2002.

[77] T. Aynaud and J.-L. Guillaume, "Static community detection algorithms for evolving networks," in *Proc. IEEE WiOpt*, 2010, pp. 513–519.

[78] K. Vijay and K. Selvakumar, "Brain fMRI clustering using interaction K-means algorithm with PCA," in *Proc. IEEE ICCSP*, 2015, pp. 909–913.

[79] L. Barnett and A. K. Seth, "The MVGC multivariate Granger causality toolbox: A new approach to Granger-causal inference," *J. Neuroscience Methods*, vol. 223, pp. 50–68, 2014.

[80] A. Duggento *et al.*, "Multivariate Granger causality unveils directed parietal to prefrontal cortex connectivity during task-free MRI," *Scientific Reports*, vol. 8, no. 1, p. 5571, 2018.

[81] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to Information Retrieval*. Cambridge University Press, 2008, vol. 39.

[82] E. A. Allen *et al.*, "Tracking whole-brain connectivity dynamics in the resting state," *Cerebral Cortex*, vol. 24, no. 3, pp. 663–676, 2014.

[83] E. Arias-Castro and J. Wang, "RANSAC algorithms for subspace recovery and subspace clustering," *arXiv:1711.11220*, 2017.

[84] M. F. Glasser *et al.*, "The minimal preprocessing pipelines for the human connectome project," *NeuroImage*, vol. 80, pp. 105–124, 2013.

[85] N. Tzourio-Mazoyer *et al.*, "Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain," *NeuroImage*, vol. 15, no. 1, pp. 273–289, 2002.

[86] A. Schaefer *et al.*, "Local-global parcellation of the human cerebral cortex from intrinsic functional connectivity MRI," *Cerebral Cortex*, vol. 28, no. 9, pp. 3095–3114, 2018.

[87] E. Y. Choi, B. T. Yeo, and R. L. Buckner, "The organization of the human striatum estimated by intrinsic functional connectivity," *J. Neurophysiology*, vol. 108, no. 2, pp. 2242–2263, 2012.

[88] D. J. Greene *et al.*, "Integrative and network-specific connectivity of the basal ganglia and thalamus defined in individuals," *Neuron*, vol. 105, no. 4, 742–758.e6, 2020, ISSN: 0896-6273.

[89] S. Marek *et al.*, "Spatial and temporal organization of the individual human cerebellum," *Neuron*, vol. 100, no. 4, 977–993.e7, 2018.

[90] D. Wang *et al.*, "Parcellating cortical functional networks in individuals," *Nature Neuroscience*, vol. 18, pp. 1853–1860, 2015.

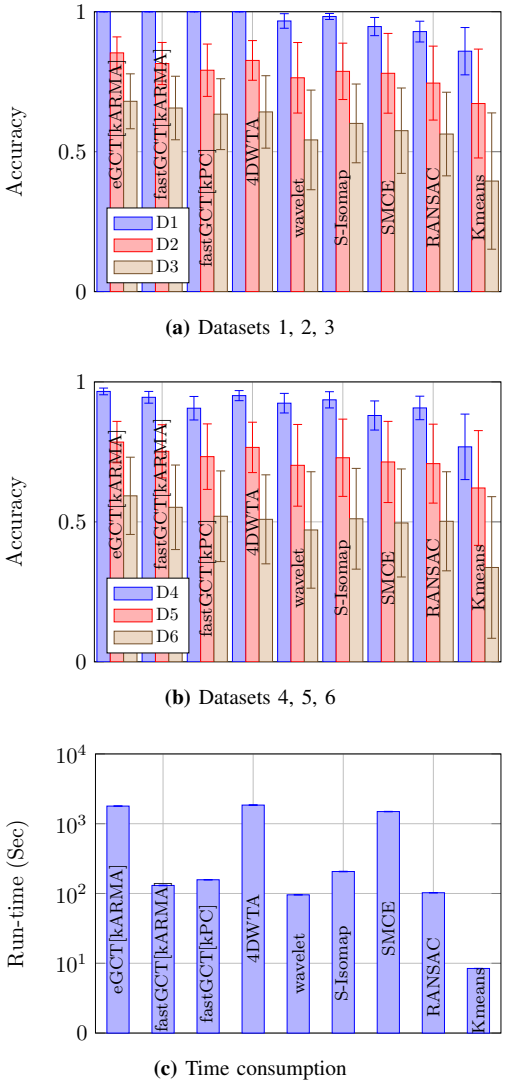
[91] P. M. Macey, K. E. Macey, R. Kumar, and R. M. Harper, "A method for removal of global effects from fMRI time series," *NeuroImage*, vol. 22, no. 1, pp. 360–366, 2004.

[92] M. Jenkinson, P. Bannister, M. Brady, and S. Smith, "Improved optimization for the robust and accurate linear registration and motion correction of brain images," *NeuroImage*, vol. 17, no. 2, pp. 825–841, 2002.

[93] N. Aronszajn, "Theory of reproducing kernels," *Trans. American Mathematical Society*, vol. 68, no. 3, pp. 337–404, 1950.

SUPPLEMENTARY DOCUMENT

This supplementary document includes figures and tables that do not fit in the main body of the manuscript, due to space constraints dictated by this Journal. The following figures and tables are labeled by the "Supp" tag.

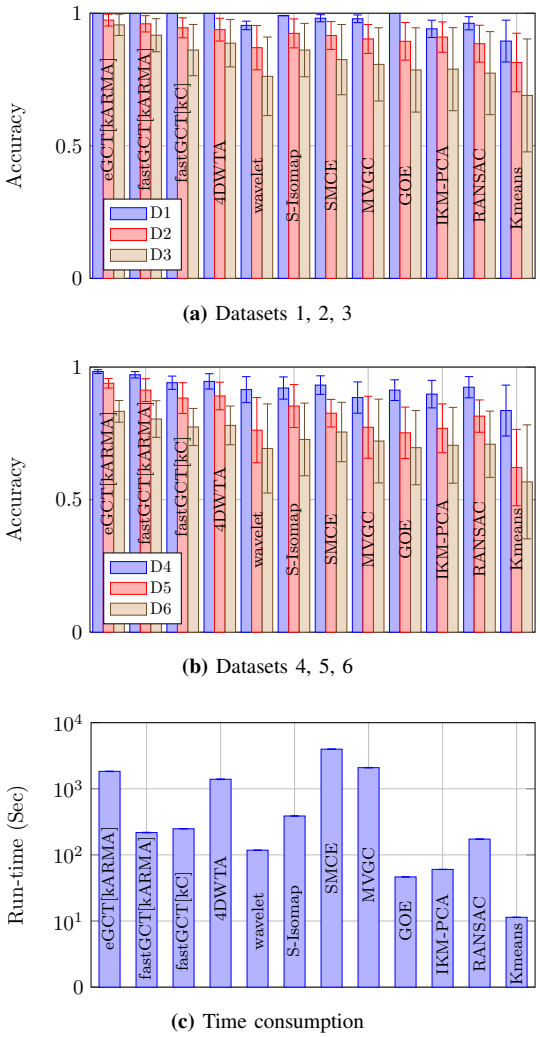


(a) Datasets 1, 2, 3

(b) Datasets 4, 5, 6

(c) Time consumption

Fig. Supp1: State clustering results of synthetic fMRI datasets. (a) Data without an independent event; (b) Data with an independent event. (c) Average time consumption of proposed and competing algorithms.



(a) Datasets 1, 2, 3

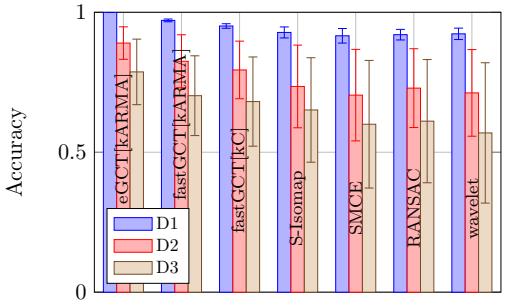
(b) Datasets 4, 5, 6

(c) Time consumption

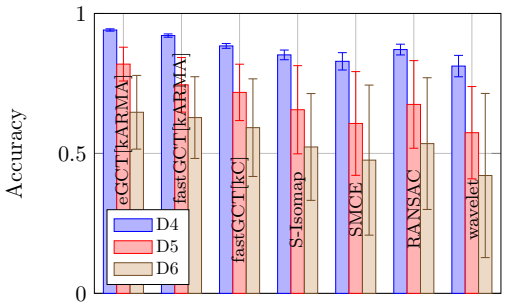
Fig. Supp2: Community detection results of synthetic fMRI datasets. (a) Data without independent event; (b) Data with independent event. (c) Average time consumption of proposed and competing algorithms.

TABLE Supp1: Parameters $(\mu, \sigma_{\mathrm{n}})$ used to generate synthetic BOLD time series

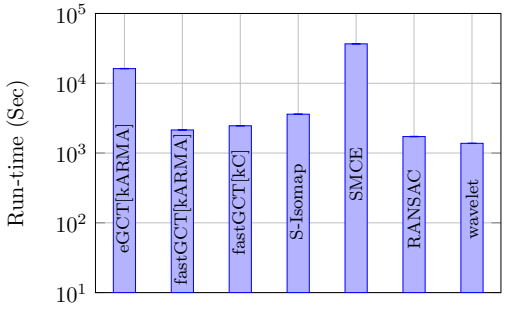
Dataset	State 1	State 2	State 3	State 4
1	$(0, -10\mathrm{dB})$	$(0, -10\mathrm{dB})$	$(0, -10\mathrm{dB})$	$(0, -10\mathrm{dB})$
2	$(0, -8\mathrm{dB})$	$(0, -8\mathrm{dB})$	$(0, -8\mathrm{dB})$	$(0, -8\mathrm{dB})$
3	$(0, -6\mathrm{dB})$	$(0, -6\mathrm{dB})$	$(0, -6\mathrm{dB})$	$(0, -6\mathrm{dB})$
4	$(0.2, -10\mathrm{dB})$	$(0.3, -10\mathrm{dB})$	$(0.4, -10\mathrm{dB})$	$(0.5, -10\mathrm{dB})$
5	$(0.2, -8\mathrm{dB})$	$(0.3, -8\mathrm{dB})$	$(0.4, -8\mathrm{dB})$	$(0.5, -8\mathrm{dB})$
6	$(0.2, -6\mathrm{dB})$	$(0.3, -6\mathrm{dB})$	$(0.4, -6\mathrm{dB})$	$(0.5, -6\mathrm{dB})$



(a) Datasets 1, 2, 3

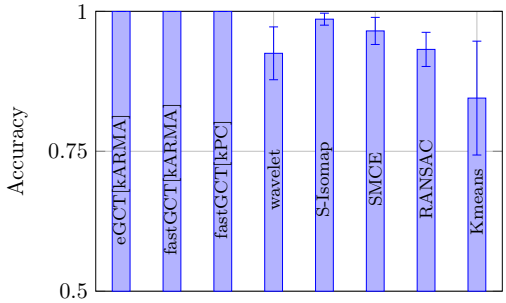


(b) Datasets 4, 5, 6

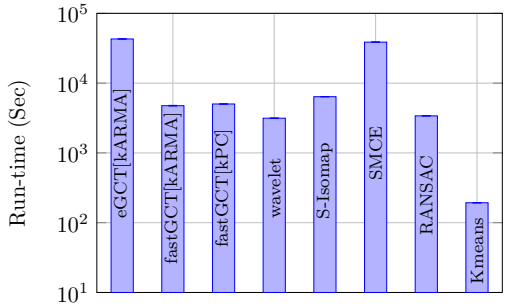


(c) Time consumption

Fig. Supp3: Subnetwork-sequence clustering results of synthetic fMRI datasets. (a) Data without independent event; (b) Data with independent event. (c) Average time consumption of proposed and competing algorithms.

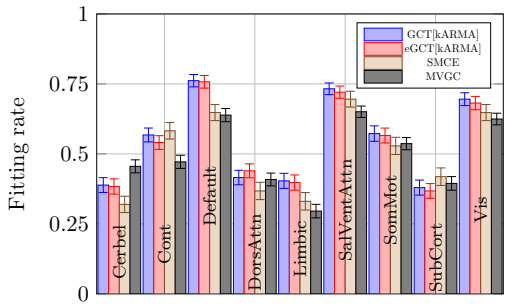


(a) Accuracy

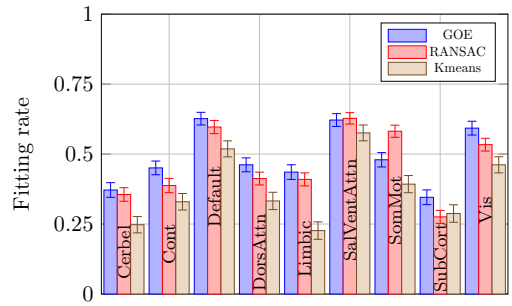


(b) Run-times

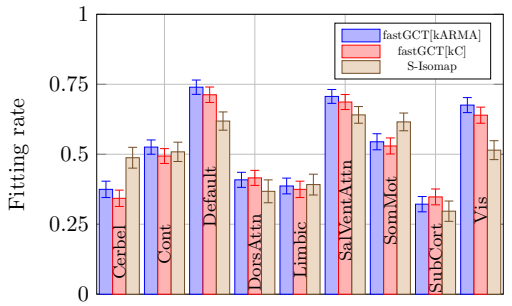
Fig. Supp4: State clustering results of real EEG dataset. (a) Average clustering accuracy. (b) Average time consumption.



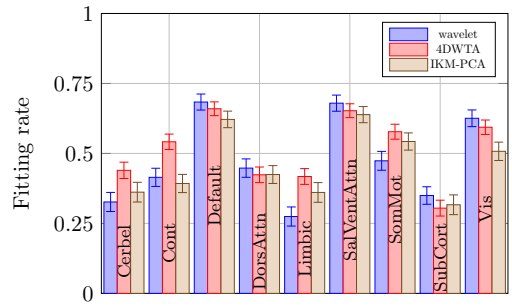
(a) Batch.I



(b) Batch.II



(c) Sequential.I



(d) Sequential.II

Fig. Supp5: Community detection results for the real fMRI dataset. (a) GCT, EGCT, SMCE and MVGC. (b) GOE, RANSAC and Kmeans. (c) FastGCT and S-Isomap. (d) Wavelet, 4DWTA and IKM-PCA.