Learning Systems. The final authenticated version is available online at:
http://dx.doi.org/10.1109/TNNLS.2012.2228226

Compositional Generative Mapping for Tree-Structured Data - Part II: Topographic Projection Model

Davide Bacciu, *Member, IEEE,* Alessio Micheli, *Member, IEEE,* and Alessandro Sperduti, *Senior Member, IEEE*

***Abstract*—We introduce GTM-SD, the first compositional generative model for topographic mapping of tree-structured data. GTM-SD exploits a scalable bottom-up hidden tree Markov model, introduced in Part I of this paper (D. Bacciu, A. Micheli, A. Sperduti. "Compositional Generative Mapping for Tree-Structured Data - Part I: Bottom-Up Probabilistic Modeling of Trees", IEEE Trans. on Neural Netw. and Learn. Sys., In Press), to achieve a recursive topographic mapping of hierarchical information. The proposed model allows an efficient exploitation of contextual information from shared substructures by a recursive upward propagation on the tree structure which distributes substructure information across the topographic map. With respect to its non-compositional generative counterpart, GTM-SD is shown to allow the topographic mapping of the full sample tree, that includes a projection onto the lattice of all the distinct subtrees rooted in each of its nodes. Experimental results show that the continuous projection space generated by the smooth topographic mapping of GTM-SD yields to a finer grained discrimination of the sample structures with respect to the state-of-the-art recursive neural network approach.**

***Index Terms*—Generative Topographic Mapping, Tree structured data, Recursive bottom-up processing, Hidden Tree Markov Model, Self Organizing Map**

I. Introduction

Effective information visualization is of paramount importance for facilitating the understanding of complex data, especially when dealing with non-flat information such as in tree-structured domains. Tree-structured information is a particular form of relational data composed of atomic pieces of information (the nodes) that are in a hierarchical ancestor-descendant relationship. As such, the composing atomic pieces need to be considered in the context of their relatives, rather than in isolation, in order to capture the full semantics of the compound data. In particular, taking into account the hierarchical nature of tree-structured information, it is straightforward to consider evaluating a node in the context given by its direct descendants, i.e. its children.

Such an approach has the advantage of being coherent with the recursive organization of tree-structured information, where simpler substructures are located at the bottom of the tree (the simplest substructures being the leaf nodes) and are composed, at higher levels of the hierarchy/tree, to realize more complex compound entities (the most complex being the complete tree). By considering a node in the context of its children, we are evaluating a complex structure, rooted on the node, in terms of the knowledge acquired on the substructures rooted on its children. Such a property is defined as *compositionality* and has clear computational advantages typical of a divide-et-impera approach. For instance, a learning/inference procedure can incrementally process a structure by, first, tackling with the less articulated substructures at the bottom of the tree and, then, reusing the extracted information to address the complexity of the higher level nodes. Within the scope of visualization problems, compositionality has the additional advantage of providing a deeper insight into the structure being considered. In particular, a truly compositional approach allows not only the visualization of the tree as a whole, rather, it should permit to project each composing substructure in an orderly manner, thus giving a clearer picture of the regularities in the data, e.g. by highlighting shared substructures among the trees.

In this paper, we present the first generative approach to compositional topographic mapping of tree-structured data. This is achieved by learning an efficient, though approximate, process that generates sample structures from a bottom-up perspective, i.e. from the leaves to the root of the tree. Compositionality, in fact, can only be achieved with a bottom-up approach, such that the tree structure is processed from the leaves to the root with increasing levels of structure complexity. However, the probabilistic modeling of a bottom-up children-to-parent relationship is computationally expensive due to its ariety, i.e. where the relationship describes the association of L nodes to their common ancestor. Instead, top-down relationships can be efficiently modeled as multiple parent-to-child relationships of ariety 2, as they associate a parent with each child independently. This has, so far, prevented the development of compositional probabilistic approaches for hierarchical data, which has only been dealt with using top-down generative models [1]. Here, we define a compositional generative process by exploiting the probabilistic model for trees presented in Part I of this paper [2]. This process is constrained to follow a topological ordering by chaining the generation of the tree substructures to points on a latent topographic map [3], yielding to a topological mapping for each sample tree together with an implicit projection of each distinct subtree in the structures under consideration.

Several works (see [4], [5] for a review) have addressed compositional visualization of structured data from a neural network perspective, in particular by resorting to recurrent extensions of the Self Organizing Map (SOM) (e.g. SOMSD [6]). Recurrent neural architectures, in fact, can deal with tree-structured data as they naturally exploit the recursive/hierarchical aspect of such information. However, no probabilistic compositional model has been proposed until now and, only recently, [7] has introduced a non-compositional generative approach for structured data visualization, known as GTM-HTMM. The idea underlying GTM-HTMM is to extend the Generative Topographic Mapping (GTM) [3] from flat i.i.d samples to independently distributed trees. This is obtained by exploiting the GTM latent space centers as sources for a collection of Hidden Tree Markov Models (HTMMs) [8] that are, in turn, responsible for the top-down generation of the observed input trees, starting from the root and ending up to the leaves, via state transitions from the parent to the children nodes described by the hidden state dependency matrix. In GTM-HTMM, every tree is treated as an atomic entity such that only the tree as a whole is associated to a projection onto the lattice. Indeed, such an approach is not compositional, as top-down tree generation entails that a node is evaluated in the context given by its parent: therefore, an hidden state captures little information concerning the co-occurrence of particular substructures in its child subtrees.

Conversely, by taking inspiration from the recursive neural approach [6], we introduce a GTM for Structured Data (GTM-SD) where single nodes, instead of full trees, are generated in a bottom-up fashion by different latent points on the lattice. This is done similarly to how the GTM Through Time [9] deals with the topographic mapping of sequences. Thanks to the bottom-up approach, it is possible to propagate structural information across the tree structure, so that an ancestor node effectively captures dependency information concerning its descendants. The GTM-SD model is realized by exploiting the Bottom-up Hidden Tree Markov Model with Switching Parents approximation (SP-BTHMM) that has been introduced in Part I [2]. This model introduces an approximation of the bottom-up transition, i.e. from the joint state of the children to the parent, in terms of a mixture of pairwise child-to-parent transitions, thus making learning computationally feasible and scalable also for trees with large out-degree.

A preliminary version of this work has been published in [10]: here we present an extended version providing an in-depth introduction to GTM-SD as well as an extended experimental assessment, comprising a larger number of benchmarks and a more extensive comparative analysis with the state-of-the-art. The remainder of the paper is organizes as follows: Section II briefly recalls the notation introduced in Part I [2] for representing tree-structured data and reviews the main neural and generative approaches to the topographic mapping of non-flat data. Section III, after a brief summary of the SP-BHTMM from Part I [2], describes the proposed compositional model for the probabilistic topographic mapping of trees. In Section IV, we experimentally assess the performance of GTM-SD with respect to state of the art neural and generative models for topographic mapping of structured data, i.e. GTM-HTMM [7] and SOM-SD [6]. Section V discusses the outcome of the experimental evaluation and concludes the paper.

II. Background

A. Definitions and Notation

In the following, we briefly recall the notation for representing tree-structured data introduced in Part I: details of the notation can be found in [2]. As in Part I, we deal with rooted trees, denoted as $\mathbf{y}^n$, that are connected acyclic graphs consisting of a set of nodes $\mathcal{U}_n = \{1, \ldots, U_n\}$ such that a single vertex is denoted as the *root* and any two nodes are connected by exactly one simple path. The index n is used to denote the n-th tree in a dataset of N structures and will be omitted for notational simplicity when the context is clear.

The terms $u, v \in \mathcal{U}_n$ are used to denote generic nodes of a tree $\mathbf{y}^n$, while $pa(u)$ denotes the parent node of u and $ch_l(u)$ is the l-th child of node u. Any two nodes u and v sharing a common parent $pa(u) = pa(v)$ are called *siblings*, while a node without children is called a *leaf*. We denote the set of the leaf nodes of the n-th tree as $\mathcal{LF}_n$. The term $\mathbf{y}_u$ is used to denote the subtree of $\mathbf{y}$ rooted at node u: in particular, $\mathbf{y}_1$ is the whole tree and $\mathbf{y}_{ch_l(u)}$ denotes the l-th child subtree of a node u. Additionally, we use the term $\mathbf{y}_{1\setminus u}$ to denote the tree, rooted at 1, without the $\mathbf{y}_u$ subtree. For the purpose of this paper, we assume trees to have a finite maximum *outdegree* L, i.e. the maximum number of children of a node; these structures are also referred as L-ary trees. Further, we consider only *labeled trees*, where each node u is associated to a label y_u, of dimensionality $D \geq 1$, which can be categorical or continuous.

B. Related Models

Unsupervised learning has long since emerged as a fundamental neural network paradigm for flat data visualization and exploration, especially throughout the use of models allowing topographic projection of high dimensional data into low-dimensional lattices. Among the best known models, in this respect, are the Self-Organizing Map (SOM) [11] and its probabilistic counterpart Generative Topographic Mapping (GTM) [3], which have inspired two prominent approaches to the unsupervised processing and visualization of non-vectorial information, that are the *recursive neural* approach [4], [5] and the *generative probabilistic* model [7]. A third approach exploits extensions of both SOM and GTM using *kernel metrics* for structured data.

The *recursive neural* approach has been formalized in [4], [5] by providing an organic general framework covering several, independently proposed, SOM models for sequential and structured data. The key idea of the recursive approach is to capture the structure of the data by exploiting a recursive *context* to represent the information processed until the current step. This results in recurrent self-organizing models with a common recursive dynamics that adapts the context as computation unrolls on the structure of the input data, differing only in the way in which such context is internally represented by the neural map. For instance, the Temporal Kohonen map (TKM) [12] and the Recurrent SOM (RSOM) [13] originally extend the standard SOM to deal with sequential data. Both models introduce self-recurrent connections that, given the current element of a sequence, provide contextual information regarding the neuron response to the previous element in the sequence. The RecSOM [14], [15] enlarges the context by allowing each neuron to receive feedback connections propagating past activations from all the units in the map. To this end, each RecSOM neuron is equipped, in addition to the standard SOM weight w_i, with a context vector c_i which stores the past activation profile of the whole map, indicating in which sequential context the vector w_i should arise. The SOM for Structured Data (SOM-SD) [6] further extends topographic mapping to deal with more articulated contexts and tree structured information with a fixed outdegree L. Given a node u in the tree, SOM-SD processes its label y_u within the context given by the L child subtrees of u, i.e. $\mathbf{y}_{ch_1(u)}, \ldots, \mathbf{y}_{ch_L(u)}$. Similarly to RecSom, each SOM-SD neuron is equipped with a number of vectors $c_1^i, \ldots, c_L^i$, that encode context as the indices of the winner neurons for the L subtrees of the current node u. Tree processing proceeds bottom-up from the leaves to the root, recursively computing the winning neuron index $I(\mathbf{y}_u)$, given a tree $\mathbf{y}_u$ with root label y_u, as

$$I(\mathbf{y}_u) = \arg\min_i \left\{ \mu_1 \|y_u - w_i\|^2 + \mu_2 \left(\|I(\mathbf{y}_{ch_1(u)}) - c_1^i\|^2 + \right.\right. \\ \left.\left. \cdots + \|I(\mathbf{y}_{ch_L(u)}) - c_L^i\|^2 \right) \right\}. \tag{1}$$

Leaves do not have child subtree, hence are assigned an empty context, typically set to $(-1, -1)$, which is the same used to denote a missing child. The Merge SOM (MSOM) [16] defines a context vector that intuitively combines the sequence history by referring to a merged form of the winner neurons properties, including the weight vector of the previous winner and the context vector computed for the previous element of the sequence. Further, MSOM allows to take into consideration arbitrary lattice topologies such as Neural Gas [17].

The *generative probabilistic* approach stems as an extension of the Generative Topographic Mapping (GTM) [3], which has been introduced as a principled alternative to SOM for the visualization and clustering of high-dimensional real-valued data. The key idea of the GTM is to learn a generative model for the data by fitting a mixture of probability density functions whose parameters are controlled by an hidden low-dimensional space where data is projected. For instance, the standard GTM models the distribution of the original high-dimensional data $y^n \in \mathbb{R}^D$ as a mixture of Gaussians $\mathcal{N}(y|\Gamma(x_c), \sigma)$, whose mean $\Gamma(x_c)$ is parameterized by latent variables x_c lying on a mono/bi-dimensional lattice (the hidden space), while the spherical variance σ is shared among the centers x_c. The nonlinear transformation $\Gamma : \mathcal{V} \rightarrow \mathbb{R}^D$ serves as mapping from the continuous latent space $\mathcal{V}$ to the data space $\mathbb{R}^D$. Topographic organization is achieved by constraining the means $\Gamma(x_c)$ to lie on a non-Euclidean manifold embedded in data space. Like SOM, the generative topographic mapping has been first extended to sequential data by the GTM Through Time (GTM-TT) model [9]. In the original GTM, observations y^n are assumed to be independent and identically distributed (i.i.d.). Conversely, GTM-TT assumes that two adjacent elements y_{t-1}^n and y_t^n of a sequence $\mathbf{y}^n$ have a Markovian dependence and it describes the generative process of a sequence $\mathbf{y}^n = y_1^n, \ldots, y_T^n$ by an *Hidden Markov Model* (HMM) whose hidden states are constrained to lie on a lattice in the latent space. In a sense, the hidden states of the HMM serve as the neuron indices in the recurrent neural paradigm, providing a context for the current sequence element y_t^n.

Recently, [7] has proposed an extension to the GTM that deals with tree-structured data by exploiting an Hidden Tree Markov Model (HTMM) [8] as a generative model for tree structured data. Differently from GTM-TT, the approach in [7] (referred as GTM-HTMM, in the following) considers each tree $\mathbf{y}^n$ as an atomic i.i.d. sample, i.e. similarly to how flat observations are dealt with in the standard GTM. The emission probability of each i.i.d. tree is then modeled by a constrained HTMM distribution that plays the same role as the constrained Gaussian emission in the standard GTM for vectorial data. Such a process ensures that a tree $\mathbf{y}^n$ is generated as an atomic entity from a single point of the GTM-HTMM latent space. The likelihood of the corresponding mixture model is

$$\mathcal{L} = \prod_{n=1}^{N} \sum_{c=1}^{C} P(x_c) P(\mathbf{y}^n | x_c) \tag{2}$$

where $P(\mathbf{y}^n|x_c)$ is the tree distribution described by an HTMM parameterized by the latent points x_c. The smooth mapping Γ is used, as in flat GTM, to map latent points to the hidden tree models in order to ensure their topological organization. Similarly to an HMM for sequences, the HTMM models an observed tree by a generative process defined by the hidden state variables $\{Q_u\}$ which take values from the discrete set of hidden states $\{1, \ldots, K\}$ and that follows the same indexing as the observed node u (which is the equivalent for trees of the time instant t in sequential data). Differently from the recursive neural approach, the HTMM [8] proceeds in a top-down fashion by assigning an empty context to the root, while each internal and leaf node u is evaluated in the context provided only by its parent $pa(u)$. Hidden states are characterized by a *prior distribution* $P(Q_1 = i)$ for the root node (i.e. corresponding to the empty context), a *state transition probability* $P(Q_u = i | Q_{pa(u)} = j)$, modeling the contextual relation between a node u and its parent, and an *emission distribution* $P(y_u | Q_u = i)$, modeling node label generation. The Markovian assumption for a top-down HTMM dictates that the current state of a node u depends solely on that of its parent $pa(u)$. Given an observed tree $\mathbf{y}^n$ and the latent point assignment x_c, the parameterized GTM-HTMM distribution in eq. (2) factorizes as

$$P(\mathbf{y}^n | x_c) = \sum_{\mathbf{Q} = \{1, \ldots, K\}^{U_n}} P(Q_1 | x_c) p(y_1 | Q_1, x_c) \\ \times \prod_{u=2}^{U_n} P(y_u | Q_u, x_c) P(Q_u | Q_{pa(u)}, x_c) \tag{3}$$

where the sum marginalizes over the hidden states assignment $\mathbf{Q} = Q_1, \ldots, Q_{U_n}$. For notational simplicity, we use Q_u as a short form for the assignments $Q_u = i$ when this is clear from the context. By inserting this result in eq. (2), it yields the likelihood for the GTM-HTMM model [7]. Due to the latent point parametrization, the GTM-HTMM model cannot directly estimate the HTMM state and emission probabilities; rather, it has to obtain them from the smooth mapping $\Gamma(x_c)$, resulting in estimates that need to be passed through a softmax function in order to be transformed into probabilities [7]. Summarizing, the GTM-HTMM associates an hidden tree generative model to each latent point x_c and constraints its parameters by means of the smooth mapping Γ. For each observed tree, it obtains the responsibility of that tree being generated by each of the HTMMs connected to a latent point: then, it projects the tree onto the map at the mean of the posterior distribution over x_c, that is the average of the latent centers x_c weighted by the responsibilities.

The *kernel metrics* approach exploits the so-called *kernel trick* to define distance metrics for non-vectorial data that can be used to replace the Euclidean metric in the activation function of the standard SOM as well as in the Normal distribution of the GTM. For instance, the *Kernel SOM* (KSOM) [18] is a batch algorithm exploiting diffusion kernels to induce a metric for graphs, which include trees as a special case. As with most of the kernel-based approaches, the KSOM does not learn an explicit graph prototype, rather the neuron codebook is used to store the parameters of a linear combination of the input data mapped into the Hilbert space induced by the diffusion kernel. Recently, two groups [19], [20] have independently proposed a substantially equivalent kernelized GTM for graph data which redefines the components of the GTM Gaussian mixture to use a kernel induced metric in place of the standard Euclidean norm. These two models are symmetrically defined, one (named Kernel GTM (KGTM) [20]) in terms of graph similarity and the other (named Relational GTM (RGTM) [19]) in terms of a matrix of graph dissimilarity, but they can be reformulated so that they only differ for a constant term.

Kernel-based models and the GTM-HTMM share a common approach to data processing and representation, as every structure is considered to be an atomic i.i.d observation, so that a topographic projection can only be associated to the structure as a whole. Recursive models, on the other hand, process structured information by focusing on each single node composing the graph, allowing their topographic projection on the map while considering their associated context. Related to this, it is a fundamental property of recursive approaches, known as *compositionality*, which refers to the ability in exploiting the modular nature of the data by first tackling with the less articulated substructures, to allow processing of compound structures by composing the contextual information obtained on their constituents. Another property, which can be used to characterize the models is the *adaptivity* of the metric they use to evaluate structures. Kernel-based models [19], [20] are characterized by a non-adaptive metric, since this is not learned from the data, whereas it is set a priori when choosing the graph kernel. For instance, a particular kernel K_1 can weight structural discrepancies more than label discrepancies, while another kernel K_2 might behave oppositely. Clearly, choosing the most suitable kernel becomes a task and data dependent choice. Conversely, models with an adaptive metric, such as the recurrent neural and generative probabilistic approaches, determine the best suited distance metric by inferring it from the characteristics of the structures in the dataset.

TABLE I
SUMMARY OF TOPOGRAPHIC MAP MODELS FOR NON-VECTORIAL DATA VISUALIZATION EVALUATED IN TERMS OF COMPOSITIONALITY, ADAPTIVITY OF THE STRUCTURE METRIC, AS WELL AS ON TARGET STRUCTURES (I.E. SEQUENCES OR TREES).

Model	Adaptive	Compositional	Sequence	Tree
TKM [12]	√	√	√	×
RSOM [13]	√	√	√	×
RecSom [14]	√	√	√	×
SOM-SD [6]	√	√	√	√
MSOM [16]	√	√	√	√
GTM-TT [9]	√	√	√	×
GTM-HTMM [7]	√	×	√	√
KGTM [20], RGTM [19]	×	×	√	√

Table I summarizes the properties of the approaches discussed so far in this section. SOM-SD and MSOM are the sole models that have specifically been proposed to deal with tree-structured data while retaining both compositionality and adaptivity. RecSOM has been originally proposed to deal with sequential data, but it has recently been used also with tree-structured information [21]. GTM-HTMM fails to achieve compositionality due to the top-down Markovian assumption in its HTMM. A top-down tree generation dynamics, in fact, entails that a node is evaluated in the context of its ancestors rather that its descendants, hence its hidden state cannot be modeled as a function of the co-occurrence of particular substructures in its child subtrees. Therefore, even if hidden states can be mapped to the latent space centers (like with GTM-TT), none of them can be chosen as the representative of the tree, i.e. as the projection of the whole tree on the topographic map. As a consequence, the GTM-HTMM needs to associate each point in the latent space to a separate HTMM, considering trees as atomic entities rather than compound objects. Following the recursive neural approach, compositionality can be achieved by taking a bottom-up approach which processes information recursively from the leaves to the root. In probabilistic terms, this correspond to a bottom-up generative dynamics where state transitions are performed from the hidden state of the children to the parent node. Clearly, such an approach allows evaluating a node in the context of its descendant subtrees. By this means, it is possible to propagate structural information across the tree structure, so that a single node can effectively collect dependency information concerning the whole tree rooted in it. In the remainder of the paper, we describe the details of a novel bottom-up hidden tree Markov model, showing how it can effectively be used to define compositional generative mapping model for tree structured data.

III. GENERATIVE TOPOGRAPHIC MAPPING FOR STRUCTURED DATA (GTM-SD)

In this section, we define the compositional model for the probabilistic topographic mapping of hierarchical information named *Generative Topographic Mapping for Structured Data* (GTM-SD). GTM-SD models an input tree similarly to how GTM-TT [9] represents sequences by means of HMM. We interpret a tree $\mathbf{y}^n$ as a collection of constrained observations $\{y_u^n\}$ following the Markovian dependencies determined by the structural parent-children relationships. As discussed in

Part I [2], we can describe the generative model for such data using an approximated hidden Markov model for trees, named *Switching Parent Bottom-up Hidden Tree Markov Model* (SP-BHTMM). Before delving into the details of the GTM-SD model, we briefly recall the basics of the SP-BHTMM model, whose details can be found in [2].

A. Summary of the SP-BTHMM

HMMs for trees model data by a generative process defined by a set of hidden state variables $\{Q_u\}$ associated to a state transition dynamics determined by a conditional probability of a given order L. A Bottom-Up Hidden Tree Markov Model (BHTMM) defines a generative process that propagates from the leaves to the root of an observed tree $\mathbf{y}^n$, thus modeling the structural and contextual parent-children relationships in the tree by state transitions from the hidden state of the child nodes to the parent. This is modeled by a joint *state transition probability* $P(Q_u = i|Q_{ch_1(u)} = j_1, \dots, Q_{ch_L(u)} = j_L)$ assuming that each node u is conditionally independent of the rest of the tree when the joint hidden state of its direct descendants $Q_{ch_l(u)} = j_l$ is observed.

A bottom-up state transition ensures compositionality, as simpler substructures (i.e. closer to the leaves) are processed before more articulated trees, and allows to concentrate the largest amount of contextual information in the root node. However, dealing with a joint bottom-up state transition quickly becomes computationally infeasible as the maximum outdegree L grows, since the joint state transition distribution is order of C^{L+1}. For this reason, in [2] we have proposed the *Switching Parents* BHTMM approximation (denoted as SP-BHTMM), that exploits an approximation of the joint transition matrix from L children as a convex combination of L simpler transition matrices. To this end, we introduce an unobserved (latent) variable $S_u \in \{1, \dots, L\}$, named *switching parent*, such that

$$P(Q_u|S_u = l, Q_{ch_1(u)}, \dots, Q_{ch_L(u)}) = P(Q_u|Q_{ch_l(u)}). \quad (4)$$

In other words, the knowledge of the switching parent assignment $S_u = l$ allows only the l-th child $ch_l(u)$, of a non-leaf node u, to have influence on the hidden state of u. The latent variable S_u can be introduced in the joint state transition by marginalization, yielding

$$\begin{aligned} &P(Q_u|Q_{ch_1(u)}, \dots, Q_{ch_L(u)}) \\ &\quad = \sum_{l=1}^{L} P(Q_u, S_u = l|Q_{ch_1(u)}, \dots, Q_{ch_L(u)}) \\ &\quad = \sum_{l=1}^{L} P(S_u = l) P(Q_u|Q_{ch_l(u)}), \end{aligned} \quad (5)$$

where we have used the assumption that S_u is independent of $Q_{ch_1(u)}, \dots, Q_{ch_L(u)}$. Eq. (5) states that the joint state transition can be approximated by a mixture of pairwise *state transitions* $P(Q_u = i_u|Q_{ch_l(u)} = i_{ch_l(u)})$ from the l-th child $ch_l(u)$ to its parent u, where the influence of the l-th child on the state transition to node u is determined by the weight $P(S_u = l)$. The likelihood of the SP-BHTMM model is obtained by inserting the result of eq. (5) in place of the joint state transition, yielding to

$$\begin{aligned} \mathcal{L} = &\prod_{n=1}^{N} \sum_{i_1, \dots, i_{U_n}} \prod_{u' \in \mathcal{LF}_n} P(Q_{u'} = i_{u'}) P(y_{u'}|Q_{u'} = i_{u'}) \\ &\times \prod_{u \in \mathcal{U}_n \setminus \mathcal{LF}_n} P(y_u|Q_u = i_u) \\ &\times \left\{ \sum_{l=1}^{L} P(S_u = l) P(Q_u = i_u|Q_{ch_l(u)} = i_{ch_l(u)}) \right\}, \end{aligned} \quad (6)$$

where $P(Q_u = i)$ is the prior distribution for the hidden states of the leaf nodes. As in the top-down HTMM discussed in Section II-B, each hidden state is characterized also by an *emission model* $P(y_u|Q_u = i)$ describing the distribution of node labels y_u associated to the i-th hidden state.

The SP-BHTMM model is trained by Expectation-Maximization (EM) applied to the likelihood in eq. (6), completed with latent indicator variables z^n_{ui} and t^n_{ul} modeling the (unknown) hidden state and switching parent assignments, respectively. In particular, $z^n_{ui} = 1$ if node u in the n-th tree is in state i and is 0 otherwise (similarly for t^n_{ul}, see [2] for details). By means of such indicator variables, it is possible to reformulate the likelihood in eq. (6) into the equivalent *complete* likelihood

$$\begin{aligned} \mathcal{L}_c(\theta; \mathbf{Y}, \mathcal{Z}) = &\prod_{n=1}^{N} \sum_{i_1, \dots, i_{U_n}} \prod_{u' \in \mathcal{LF}_n} z^n_{u' i_{u'}} \left(\pi_{i_{u'}} \times b_{i_{u'}}(y_{u'}) \right) \\ \times &\prod_{u \in \mathcal{U}_n \setminus \mathcal{LF}_n} z^n_{u i_u} b_{i_u}(y_u) \left\{ \sum_{l=1}^{L} t^n_{ul} z^n_{ch_l(u) i_{ch_l(u)}} \left(\varphi_l \times A^l_{ij} \right) \right\}, \end{aligned} \quad (7)$$

where $\pi_i = P(Q = i)$ is the multinomial prior probability, $b_i(y) = P(y|Q = i)$ denotes the task-dependent emission distribution (e.g. multinomial, Normal, mixture of Gaussians, $\dots$), $\varphi_l = P(S = l)$ is the multinomial switching parent prior and $A^l_{ij} = P(Q = i|Q_{ch_l} = j)$ is the *position dependent* transition distribution (i.e. it depends on the position of the l-th child).

B. The GTM-SD Model

GTM-SD exploits the SP-BHTMM model introduced in Part I [2] to learn a recursive topology-preserving projection of trees with continuous or categorical labels by means of a constrained mixture of Gaussian (resp. Multinomial) emission models. In order to build a topographic mapping on the top of the SP-BHTMM generative process in eq. (6), we need to constrain the hidden states of the Markov model to follow a topographical organization, similarly to how GTM-TT constrains the states of an HMM.

A generic GTM setting [3] comprises a continuous Euclidean latent space $\mathcal{V}$ (the map) of dimension q (typically set to 2 for visualization) that generates the parameters of an emission model through the use of the nonlinear smooth mapping $\Gamma : \mathcal{V} \rightarrow \mathcal{P}$, where $\mathcal{P}$ is the parameter space. For computational tractability, such a nonlinear transformation is defined in terms of a discrete set of C latent centers $x_i \in \mathcal{V}$ arranged to form a squared equispaced grid on the latent space $\mathcal{V}$. Each of the latent point generates the parameters of a different emission model through the mapping $\Gamma(x_i)$, which constrains them to lie on a manifold $\mathcal{S} \subseteq \mathcal{P}$ during learning.

The GTM-SD approach assumes that the hidden states Q_u of the nodes in a SP-BHTMM model are indexed by the C latent centers of a GTM map. Therefore, $Q_u = i$ indicates that the u-th node is assigned to the i-th hidden state which, in turn, is associated with the latent center $x_i \in \mathcal{V}$, which is a point on the GTM topographic map. Figure 1 graphically summarizes the key idea of the GTM-SD: given an input tree, modeled by a SP-BHTMM, we assume to have processed the tree in bottom-up fashion up to nodes 2 and 3, that are the left and right child of the root, respectively. In particular, we assume that such children have been assigned to hidden state j and k, respectively. At the next step, we probabilistically assign an hidden state i to root node 1 based on the state transition probability from j and k as well as based on the probability of generating its label y_1 through the emission model obtained by the smooth mapping $\Gamma(x_i)$. Notice that the latent center x_i straightforwardly provides a projection on the map for the root node and, hence, for the whole input tree.

The complete likelihood of the GTM-SD model follows from (7) by introducing the latent-centers/hidden-states mapping, while constraining the parameters of the emission model to be generated by the smooth mapping Γ. By recalling the positional parametrization introduced in Section III-A and by exploiting the indicator variables to rewrite the sum-marginalization in eq. (7) as a more tractable product over state assignments, we obtain the following GTM-SD complete likelihood

$$\log\mathcal{L}_c = \log \prod_{n=1}^{N} \prod_{u' \in \mathcal{LF}_n} \prod_{i=1}^{C} \left\{ \pi_i^{pos(u')} b_i(y_{u'}|\Gamma(x_i)) \right\}^{z^n_{u'i}} \times \prod_{\substack{u\in \\ \mathcal{U}_n \setminus \mathcal{LF}_n}} \prod_{i=1}^{C} \prod_{j=1}^{C} \prod_{l=1}^{L} \{b_i(y_u|\Gamma(x_i))\}^{z^n_{ui}} \left\{\varphi_l A^l_{i,j}\right\}^{z^n_{ui} t^n_{ul} z^n_{ch_l(u)j}} \tag{8}$$

that is similar to the SP-BHTMM formulation, except for the dependency of the emission model $b_i(y_u|\Gamma(x_i))$ on the smooth mapping Γ.

Learning of the GTM-SD model is performed by Expectation Maximization, by exploiting the *reversed upwards-downwards* algorithm introduced for SP-BHTMM [2] for estimating the posterior of the latent indicator variables

$$\epsilon^{l,n}_{u,ch_l(u)}(i,j) = P(Q_u = i, Q_{ch_l(u)} = j, S_u = l|\mathbf{y}^n), \tag{9}$$

that is the joint posterior probability of a node u in the n-th tree being in state i while its l-th child is in state j. The details of the *reversed upwards-downwards* algorithm are presented in [2] together with an in-depth justification for the learning equations. The key update equations and a pseudo-code summarizing the algorithm for training the GTM-SD model is reported in Appendix A of the Supplemental Material. The update equations for the prior, the transition and the switching parent distributions are identical to those on the SP-BHTMM model [2].

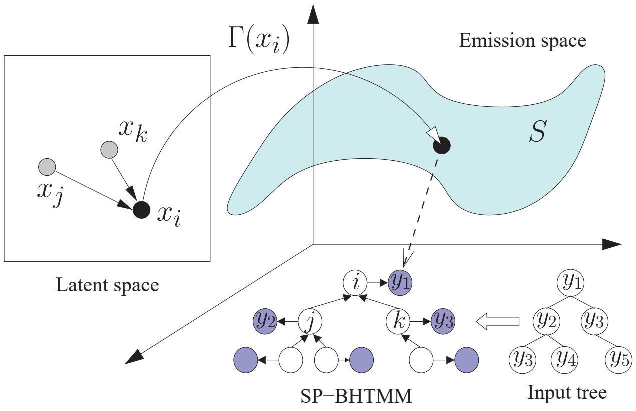


Fig. 1. Generative dynamics of a GTM-SD mapping for an input tree modeled by an SP-BHTMM. The topographic projection for the root node is determined by the hidden state i, that is probabilistically assigned based on the hidden states j and k of its left and right child. Emission for label y_1 is determined by the distribution with parameters $\Gamma(x_i)$ taken from the manifold $\mathcal{S}$ induced by the smooth mapping.

The learning equations for the emission model are modified to account for the dependency on the smooth mapping Γ. Following the ideas in [3], the emission probability for a generic real-valued label y is a Normal distribution whose means μ_i are generated by Γ, i.e.

$$b_i(y|\Gamma(x_i)) = \mathcal{N}(y|W\Phi(x_i), \sigma^2) \tag{10}$$

where σ^2 denotes the variance and $\mu_i = W\Phi(x_i)$ is the Gaussian mean, that is the projection of the i-th latent point in data space by the smooth mapping $\Phi(\cdot) = [\phi_1(\cdot), \ldots, \phi_P(\cdot)]^T$ and the weight matrix $W \in \mathbb{R}^{d\times P}$. Each element of Φ is a vector RBF function $\phi(\cdot) : \mathbb{R}^2 \to \mathbb{R}$ that is applied to the latent point coordinates x_i (usually bi-dimensional); the resulting matrix $\mathbf{\Phi} \in \mathbb{R}^{P\times C}$ has elements $\phi_{pi} = \phi_p(x_i)$ that remain constant throughout the algorithm. The weight matrix W and the variance σ^2 (typically common to all the latent points) are learned as in the standard GTM [3], i.e. given a tree $\mathbf{y}^n$

$$\mathbf{\Phi}\ G_n \mathbf{\Phi}^T {W'_n}^T = \mathbf{\Phi} R_n Y_n \tag{11}$$

where $R_n = [\epsilon^n_u(i)] \in \mathbb{R}^{C\times U_n}$ contains the state occupancies, $Y_n = [(y_u)^T] \in \mathbb{R}^{U_n\times d}$ is the observation matrix and G_n is a $C \times C$ diagonal matrix such that $g_{ii}(n) = \sum_u \epsilon^n_u(i)$. The updated W' on the full dataset can be obtained by solving eq. (11) with respect to ${W'_n}^T$ using standard matrix inversion and summing up the contribution for each tree $\mathbf{y}^n$. Notice that the terms $Y_n R_n^T$ and G_n can be efficiently computed and cached as part of the upwards-downwards algorithm. Variance update is achieved, in its simplest univariate form, as

$$\sigma^2 = \frac{\sum_{n=1}^{N} \sum_{u\in U_n} \sum_{i=1}^{C} \|y_u - W'\Phi(x_i)\| \epsilon^n_u(i)}{\sum_{n=1}^{N} U_n \cdot d}. \tag{12}$$

Such an update can be extended to diagonal and full covariance matrices along the lines of [22].

The Gaussian noise model in eq. (10) is only suitable for the analysis of trees with continuous labels, whereas in many real-world scenarios it is required to deal with categorical/discrete information, e.g. in structured XML document processing. This case can be well modeled by assuming a multinomial noise model in place of the Normal emission in eq. (10). By following the ideas in [23], we define a *Multinomial GTM-SD* (μGTM-SD) whose emission model is the multinomial

$$b_i(y|\Gamma(x_i)) = \prod_{k=1}^{K} (m_{ik})^{y_k} \tag{13}$$

where y is a K-dimensional label of a generic node defined over a discrete K dimensional alphabet, such that its k-th component y_k denotes the number of occurrences of the k-th symbol. Parameter m_{ik} defines the probability of the k-th multinomial class (symbol of the discreet alphabet) for the i-th hidden state, subject to $\sum_{k=1}^{K} m_{ik} = 1$ for each $i \in \{1, \ldots, C\}$. Again, to achieve topographic organization, the means of the multinomial in eq. (13) have to be generated by a smooth mapping from the latent space. This can be attained by obtaining the m_{ik} values throughout a softmax transformation of the form

$$m_{ki} = \frac{\exp(w_k \Phi(x_i))}{\sum_{k'} \exp(w_{k'} \Phi(x_i))} \tag{14}$$

where each term w_k is a row of the $K \times P$ weight matrix W and $\Phi(x_i)$ is the P-dimensional vector of the smooth mapping matrix $\mathbf{\Phi}$ defined previously.

As opposed to the Gaussian noise model, the derivative of eq. (8), with respect to the weight matrix W, has no closed form solution for a multinomial emission [23]. This is due to the nonlinear link function that is used in eq. (14) to obtain the parameters of the Multinomial noise model. The Generalized EM (GEM) approach in [23] addresses the problem by allowing M-step updates that increase, instead of maximizing, the GTM-SD log-likelihood. In particular, for the Multinomial emission model, we obtain the following gradient update

$$W^{(t+1)} = W^{(t)} + \delta \sum_{n=1}^{N} \Delta W_n^{(t)} \tag{15}$$

where δ is the learning step size, superscripts $t, t+1$ identify current and updated parameters and where the contribution from the n-th tree is

$$\Delta W_n^{(t)} = [Y_n R_n^T - M^{(t)} G_n] \mathbf{\Phi}^T. \tag{16}$$

The i-th column of matrix $M^{(t)} = g(W^{(t)}\mathbf{\Phi})$ contains the multinomial parameters $[m_{ki}]_{k=1}^{K}$ for the i-th latent point, computed using (14). The gradient update in eq. (15) is used to perform an inner learning loop in the M-step, where only the matrix of the Multinomial distribution means $M^{(t)}$ needs to be updated at each step of the inner learning loop.

The GTM-SD model defined by eq. (8) can be interpreted as a *twice-constrained* mixture of Gaussian/Multinomials (depending on wether eq. (10) or eq. (13) is used as emission model). The first (*emission*) constraint is imposed by the GTM approach and refers to the fact that the emission means are not chosen freely, whereas they are forced to move on the manifold induced by the smooth mapping. The second (*state*) constraint refers to the Markovian organization that is enforced on the latent space by the SP-BHTMM model, which is required to model observations that are not i.i.d samples, whereas are expected to have a complex causal relationship coherent with the structural relationships in the trees.

C. Topographic Projection and Inference

The GTM-SD defines a continuous smooth mapping from the tree-structured data space onto a topological map defined by the latent points grid. The topological ordering imposed on the map by the smooth mapping ensures that similar structures will be projected to points that are close on the map. In this sense, projecting a generic tree $\mathbf{y}$ on the map entails projecting its root to a point onto the GTM-SD latent space by using its hidden state assignment Q_1. For instance, a *posterior mean* approach maps the tree to the average of the latent point centers x_i weighted by the respective posterior probabilities $P(Q_1 = i|\mathbf{y})$, that is

$$X_{mean}(\mathbf{y}) = \sum_{i=1}^{C} P(Q_1 = i|\mathbf{y}) \cdot x_i. \tag{17}$$

Alternatively, the tree can be mapped to its *posterior mode* by means of

$$X_{mode}(\mathbf{y}) = \arg\max_{x_i} P(Q_1 = i|\mathbf{y}). \tag{18}$$

Several approaches can be used to determine the hidden state assignment for posterior projection. The most straightforward one exploits the $\beta_u(i)$ factor computed by the upwards/downwards recursion employed, in the previous Section, for parameter learning. By definition [2], the upwards parameter for a generic node u in the tree $\mathbf{y}^n$ is

$$\beta_u(i) = P(Q_u = i|\mathbf{y}_u^n),$$

where $\mathbf{y}_u^n$ is the subtree rooted in u in the n-th tree. Clearly, for a root node the upwards parameter is equivalent to the tree posterior given that

$$P(Q_1 = i|\mathbf{y}^n) = P(Q_1 = i|\mathbf{y}_1^n) = \beta_1(i)$$

so that $\beta_1(i)$ can be directly used to obtain the projections in eq. (17) and (18) for the whole tree.

The compositionality of the underlying SP-BTHMM model allows gaining a more articulated insight into the structures. In particular, given an observed tree $\mathbf{y}^n$, the GTM-SD model allows projecting every subtree $\mathbf{y}_u^n$ rooted in each of the nodes u of the observed tree. Such a feature realizes the compositional mapping that has been missing in generative models for structured data (see Section II), marking a fundamental difference from the top-down GTM-HTMM model [7] that defines a generative mapping only for observed trees considered as atomic entities. By means of its principled probabilistic approach, GTM-SD allows a fine grained control over the amount of contextual information used to determine topographic projection. In particular, GTM-SD defines two different subtree projection modalities, that are a *compositional* and a *contextual* approach. The former, by taking a *compositional* approach equivalent to the recursive SOM-SD [6], determines the hidden state assignment for node u based only on the information propagated from the subtree $\mathbf{y}_u^n$, thus discarding the contextual information from the rest of the $\mathbf{y}^n$ structure. Again, this can be done very efficiently by considering u as the root node of an isolated tree $\mathbf{y}_u^n$, thus projecting the subtree $\mathbf{y}_u^n$ on the map using (17) where

$$\beta_u(i) = P(Q_u = i|\mathbf{y}_u^n)$$

is used in place of the true posterior $P(Q_u = i|\mathbf{y}^n)$. The latter, on the other hand, allows projecting the subtree $\mathbf{y}_u^n$ within the context given by the full tree $\mathbf{y}^n$. In other words, the hidden state assignment for node u is determined based on the information propagating from the whole tree $\mathbf{y}^n$, instead of observing only $\mathbf{y}_u^n$. This can be obtained with the reversed upward/downwards algorithm by computing the state occupancy posterior $\epsilon_u(i)$ that, by definition [2], is exactly the contextual posterior $P(Q_u = i|\mathbf{y}_u^n)$ used in eq. (17) and (18) (the equivalence $\epsilon_u(i) = P(Q_u = i|\mathbf{y}_u^n)$ follows straightforwardly also from the definition (9)). Summarizing, to visualize a test tree along with its substructures, it is sufficient to perform an upwards recursion to obtain its compositional projection; the contextual mapping can be obtained at the cost of an additional downwards recursion computing $\epsilon_u(i)$.

Finally, an alternative means for determining the posterior projection is by means of the *reversed Viterbi* algorithm discussed in [2]. Viterbi algorithms are a class of inference procedures that seek the hidden states assignment $\mathbf{Q} = \mathbf{x}$ that maximizes the joint distribution of an observed tree $\mathbf{Y} = \mathbf{y}$ and the hidden states. Clearly, this is a different, though correlated, joint maximization (or decoding) problem, which is wider than directly estimating the posterior distribution needed for (17) and (18). Typically this is computationally more expensive than an upwards recursion, especially if the exact Viterbi is employed [2]. Further, such an approach only allows contextual projection of the subtrees as the hidden state assignment is jointly maximized for the whole tree. In the following, we will focus our experimental analysis on posterior projections obtained by the compositional approach.

D. Model Meta-Parameters and Configuration

The actual instantiation of a GTM-SD model also depends on the selection of a number of meta-parameters whose majority, however, is problem independent or has a minor effect on the performance of the model. Most of this meta-parameters are *inherited* from the underlying generative topographic model: in general, GTM-SD uses the standard meta-parameter choices suggested for the GTM for flat data in its foundational work [3]. These serve to define the exact form of the smooth mapping Φ, e.g. by selecting the number and type of basis functions, or to complete the parametrization of the emission distribution, e.g. by assigning an initial value to the variance σ^2 of the GTM emission. In GTM-SD, like standard GTM, the smooth mapping Φ is defined by RBF basis functions because these define a smooth mapping that is a Generalized Linear Regression model which, under well-defined assumptions, has universal approximation capabilities [3]. In other words, provided that a suitable number of basis function is used, it can represent any mapping. In the two-dimensional GTM map, [3] suggests that 16 is an adequate number of basis functions to represent any typical mapping: this number is usually employed also by GTM-SD. When the data lies on a very low-dimensional manifold, it might be necessary to reduce such number to avoid numerical instabilities: this is the case of the experiment in Sect. IV-A, where we use the smallest non trivial number of basis functions, i.e. 4, to avoid the numerical problems induced by a very degenerate data space of identical node labels.

The standard GTM emission depends on the weight matrix W and on the variance σ^2, that are learned through the EM process. A detailed discussion on how the initialization of W influences the resulting generative maps can be found in [3]: in general, the W initialization can be obtained by sampling random values from a uniform distribution, without noticeable impact on the generated maps. This is the approach used by GTM-SD. The emission variance σ^2 influences the degrees of smoothing that is initially applied to the emission. The GTM for flat data typically uses a standard value of $\sigma^2 = 1$ for all the latent centers [3], with an optional regularization scheme to avoid numerical problems with degenerated data. GTM-SD typically uses the same $\sigma^2 = 1$ initialization and regularization scheme: only in the experiment in Sect. IV-B, we initialize $\sigma^2 = 2$ to preserve coherence with the original setup in [7].

The number of hidden states C is the main GTM-SD meta-parameter that is problem-dependent and whose choice can vary the quality of the obtained maps. The C parameter regulates both the resolution of the map (larger C means more latent centers and an higher resolution) and the memory of the underlying Markov model for structures (larger C means more hidden states to memorize structural information). A standard choice of $C = 100$ is used in the original GTM model and it can be exploited as a starting point to evaluate the maps: in the experiments, we explore several C values to appreciate the effect of different map resolutions and memory capacities.

IV. Experimental Results

In this section, we provide a thorough experimental evaluation of the proposed approach, with the aim of characterizing the capabilities of the topographic mapping (i.e. GTM-SD), while the properties of the underlying probabilistic model for trees (i.e. SP-BHTMM) have been discussed and experimentally assessed in Part I [2]. Section IV-A characterizes the behavior of the GTM-SD by evaluating the label-structure tradeoff, that is the ability of the map to discriminate based on the nodes' labels, as opposed to discriminating based on the structure of the tree. The GTM-HTMM model [7] is a natural term of comparison to benchmark GTM-SD against state of the art models in literature, in particular as regards generative mapping of structured data. Section IV-B presents the experimental comparison between GTM-SD and GTM-HTMM: the source code for the latter algorithm is not available, but the experimental evaluation has been performed on the same datasets and experimental setting in [7], thanks to the data provided by the authors of GTM-HTMM. Finally, Section IV-C compares the discriminative power of the GTM-SD maps against a leading compositional approach, that is the recursive SOM-SD neural network [6]. The two models are compared on two real-world datasets, comprises trees with discrete labels which also allows evaluating the performance of GTM-SD with multinomial emission.

A. GTM-SD: Evaluating the Structure-Emission Trade-off

In this section, we evaluate the effect of homogeneous labels on structure discrimination in the GTM-SD topographic mapping. To this end, we have devised a synthetic tree generator[1] [2] to populate a dataset, named Artificial1, comprising 200 *left sequences*, i.e. represented as binary trees with only left children, 200 *right sequences* and 200 non-complete binary trees. The trees and sequences in the dataset are allowed to have different depths, varying between 1 and 6: notice that the dataset includes also degenerated trees comprising only a single node. In order to evaluate the effect of homogenous non-Markovian node labels on structure discrimination, we have sampled the emission of all the trees and sequences from the same bivariate Gaussian with full covariance.

The GTM-SD has been initialized with a 5×5 latent space lattice, corresponding to a SP-BHTMM with $C = 25$ hidden states. Following the standard setup of the original GTM [22], the smooth mapping is defined by 16 RBFs with unit variance centered on a 4×4 grid, plus a constant bias (i.e. $P = 17$). The weight matrix W has been initialized with random values in $[-1, 1]$ and the variance of the GTM emission has been set to $\sigma^2 = 1$ for each hidden state. For the purpose of this experiment, we assume a SP-BHTMM with positional stationarity. In the binary tree structures, left children are identified by the child index $l = 1$, while right children are associated to index $l = 2$. Therefore in the positional parametrization, the term A^1_{ij} denotes the state transition from a left child (subscript $l = 1$) in state j to a parent in state i (similarly A^2_{ij} refers to a right child).

Figure 2 shows the posterior mean projections obtained by the GTM-SD: clearly there is no topographic organization of the space with respect to the tree structure. We have experimented with several hidden state numbers (i.e. finer scaled lattices), still obtaining the same results. In fact, such lack of structural organization is the consequence of the predominance of the label emission over the tree structures. The way trees distribute on the hidden space follows the Gaussian distribution generating the node labels. To better understand this behavior, we should consider the generative model underpinning GTM-SD: this essentially expects data to be generated from an hidden process such that there exist some sort of Markovian dependence between the child and the parent emissions. This is not the case for the data in Artificial1, as node emissions are i.i.d. from the originating Gaussian. In a sense, the best generative model for such data is a single Gaussian, which would provide the results in Fig. 2.

A question that follows straightforwardly from such results is whether GTM-SD is an algebraic model or if it can effectively detect the tree structure when labels have less relevance. To this aim, we have generated a second dataset, that is structurally identical to the Artificial1, but where the emissions are now set to be exactly the same for each node and tree in the dataset. Notice this is a degenerated setting for a Gaussian GTM model, but its use is intended only to deepen the understanding on the structure discrimination ability of the GTM-SD, given that samples in this dataset can only be differentiated by their structure. In order to avoid numerical problems, we reduced the number of basis functions to 4 and use the σ^2 regularization scheme in [22] with parameter $\lambda = 0.1$. The other parameters are left unchanged.

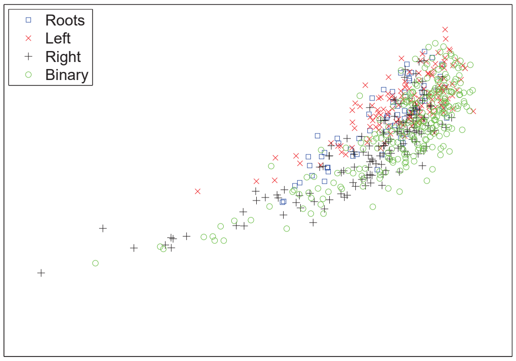
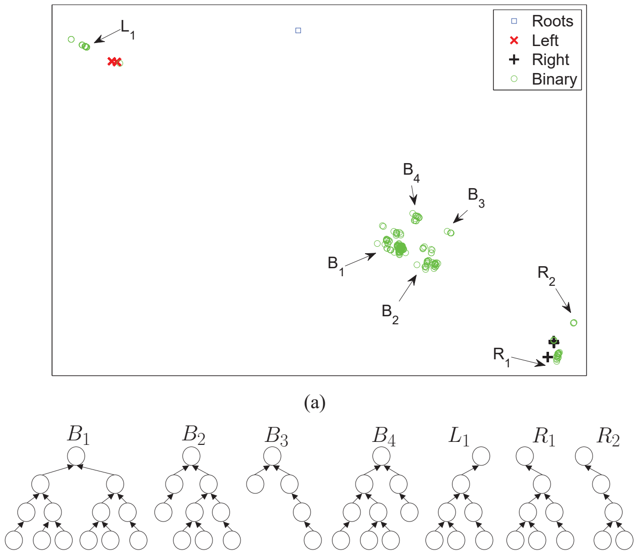


Fig. 2. GTM-SD Topographic mapping obtained for the Artificial1 data with emissions sampled from a single Gaussian: the resulting fitted distribution is, expectably, a Gaussian.

The corresponding topographic mapping is shown in Fig. 3.a: GTM-SD discriminates quite clearly the 4 structure classes, with the left and right sequences positioned at the opposite corners of the map, while most of the binary trees are concentrated at the center of the map. Both sequence types tend to split into 2 sub-groups (although separation is less clear for left sequences), one comprising sequences of length 2 and the other including the remainder of the elements. Figure 3.b shows a graphical interpretation for the binary tree subgroups identified in Fig. 3.a. Interestingly, some binary trees have been positioned close to the sequence areas, i.e. subgroups L_1, R_1 and R_2. These corresponds to *unbalanced* trees that are missing either the right root subtree (L_1) or the left root subtree (R_1 and R_2). Moreover, R_1 and R_2 differentiate further into trees with a *balanced* right root subtree (R_1) and structures with a pseudo-sequential behavior R_2. Structures in the middle correspond to more balanced trees whose root have non-null left and right children. In particular, subgroup B_1 comprises the balanced structures, while B_2 and B_4 cluster those showing more nodes in the right and left top-subtree, respectively. Finally, subgroup B_3 is the binary tree counterpart of the pseudo-sequential structures in R_2.

The results in Fig. 3.a show that GTM-SD can effectively discriminate trees based on their structure; only, it needs to find the correct balance between the influence of the node labels and the tree structure, especially when the labels are counter-informative as in the dataset in Fig. 2. A straightforward way to reduce the discriminative effect of node labels is by smoothing the response of the GTM-SD activation using a larger variance initialization. Figure 4 shows the results obtained on the same

[1] Available for download here: http://www.di.unipi.it/~bacciu/artree.html



(a)

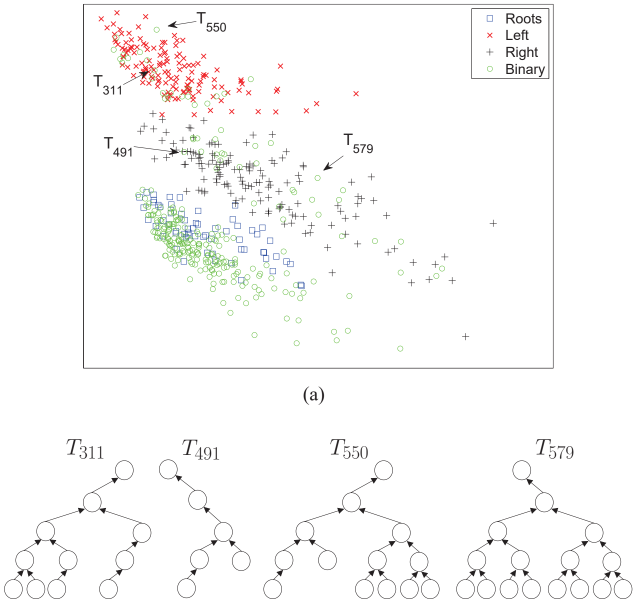
(b)

Fig. 3. GTM-SD Topographic mapping obtained for the Artificial1 data with identical emissions: (a) structures are organized top-left to bottom-right on the basis of balancing of nodes in the left and right root subtrees; (b) prototypical structures corresponding to the map labels.

data in Fig. 2 (i.e. with no identical emissions as in Fig. 3), but using a larger initial emission variance equal to $\sigma^2 = 100$. The resulting topographic mapping is coarser grained than in Fig. 3 as it accommodates both the influence of the emission as well as of the structure. The results show that GTM-SD can effectively cluster left and right sequences as well as the balanced binary trees (on the lower left). Interestingly, the unbalanced binary trees (i.e. corresponding to clusters L_1, R_1 and R_2 in Fig. 3) are again mapped onto the corresponding sequence area. Figure 4.b shows examples of unbalanced structures mapped to the corresponding sequence area, where more unbalanced structures (e.g. T_{311} and T_{491}) are more mixed with the sequence clusters with respect to more regular structures (e.g. T_{550} and T_{579}). Notice that, due to stochastic fluctuations in the generation of the artificial dataset, there are a number of extremely unbalanced right-structures such as T_{491}, but no equivalent degenerate left-structures. This is reflected in the organization of the map in Fig. 3.a where there exists two clusters for unbalanced (i.e. R_1) and extremely unbalanced (i.e. R_2) right structures, but a single cluster for left-unbalanced trees (i.e. L_1).

B. GTM-SD Comparison with GTM-HTMM

This section evaluates GTM-SD compared to a closely related generative model in literature, that is GTM-HTTM [7]. The experimental comparison is based on the two datasets used by [7] and kindly provided by the authors of GTM-HTTM. The simulation setup is chosen in agreement with that in [7] to allow a fair comparison of the results. We employ a fully stationary GTM-SD with $C = 100$ hidden state organized on a 10×10 latent space lattice. The smooth mapping and the parameter initialization has been chosen as



(a)

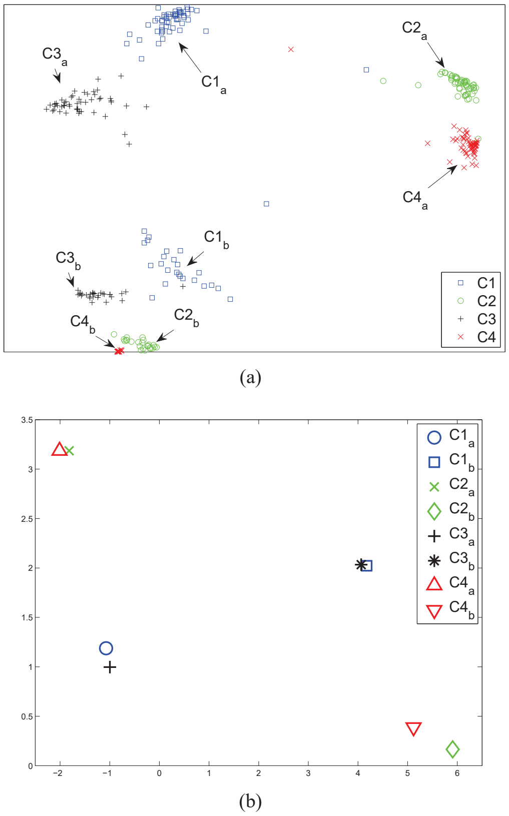
(b)

Fig. 4. GTM-SD Topographic mapping obtained for the Artificial1 data with smoothed emission: (a) binary trees mixed with the sequences correspond to the unbalanced sequence-like structures clustered as L_1, R_1 and R_2 in Fig. 3; (b) examples of unbalanced binary trees corresponding to the map labels. Notice that structures that are projected in the center of the sequence clusters (e.g. T_{311} and T_{491}) are more unbalanced than those close to the cluster boundaries (e.g. T_{550} and T_{579}).

in the experiment in Section IV-A, except for the variance that has been initialized to $\sigma^2 = 2$ as in [7].

The first dataset, named Artificial2, comprises synthetic data sampled from four Top-down Hidden Tree Markov Models (THTMMs): details of the THTMM setting from [7] are reported in Appendix B of the Supplemental Material. Each element is a complete binary tree comprising 15 nodes and there are a total of 80 elements for each of the 4 classes corresponding to the generating THTMMs.

The samples in this dataset cannot be discriminated by their structure, as they all are complete binary trees with the same number of nodes. Further, as shown in [7], the four classes cannot be discriminated based only on their observations since these overlap in the bi-dimensional space. Figure 5.a shows that the topographic mapping obtained by GTM-SD can separate such 4 classes, although, differently from GTM-HTMM, it generates additional subgroups. The correspondig GTM-HTMM map is in Fig. 6, which reproduces Fig. 7.a in [7]. In particular, by looking at the root labels of the eight clusters in Fig. 5.a, it appears that GTM-SD is differentiating the samples with respect to the root emission or, in other words, with respect to its hidden state. In fact, each node emission can be sampled from one out of two Gaussian distributions, each corresponding to an hidden state of one of the four THTMMs [7]. Figure 5.b shows the means of the root labels for each of the 8 clusters identified by GTM-SD: each of them approximate the means of the emission distribution of the 8 hidden states that are generating the node labels (confront with the emission means in Appendix B of the Supplemental Material).



(a)

(b)

Fig. 5. Artificial2 dataset: (a) Topographic mapping obtained by GTM-SD; (b) Means of the root labels for each cluster discovered by GTM-SD.

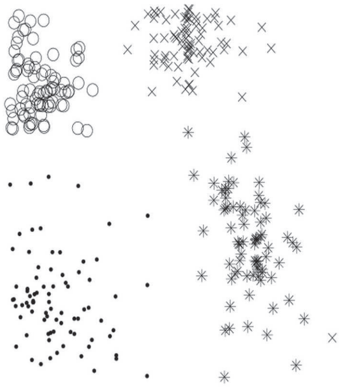


Fig. 6. Topographic mapping obtained by GTM-HTMM on the Artificial2 dataset: this image is an authorized reproduction of Fig. 7.a in [7]. The four THTMM classes are represented by different placeholders.

The structure of the topographic map in Fig. 5 is quite clear: the top-bottom separation, corresponding to the bi-partition of subclasses with a subscript from those with b subscript, corresponds to separating trees whose root emission has been sampled from a Gaussian positioned on the second quadrant (i.e. $C1_b$ to $C4_b$) from those sampled from a Gaussian positioned on the first quadrant (i.e. $C1_a$ to $C4_a$). Figure 5.b graphically shows that subclasses $C2_a$ and $C4_a$ are close because their roots share the same generating Gaussian, but their projections are not mixed because GTM-SD can discriminate the different structure of their generating THTMMs. A similar discussion applies to the couples $C1_a$-$C3_a$, $C2_b$-$C4_b$, etc. These results point out how GTM-SD discriminates strongly based on the state of the root node, which is not surprising, given that it exploits a bottom-up generative approach where structural information is conveyed to the root node and captured into its hidden state assignment.

The comparison of the GTM-SD map with that generated by GTM-HTMM does not suggest a clear answer on which model achieves the best data visualization. In fact, we are convinced that in an exploratory task there is no such best view in general, whereas the two models provide with different perspectives over the data, depending on the assumptions underlying the respective generative processes. By its own bottom-up nature, GTM-SD tends to provide more discriminative projections for the substructures close to the root of the tree, than for the leaves: the practical advantage of having such discriminative projections is very much application dependent. In the particular case in Fig. 5, the GTM-SD map has identified a regularity in the data, that is the fact the root labels from each class can come from two different Gaussian sources, which corresponds to an actual property of the original generative process used to produce the trees. Interestingly, the results in [7] show that an alternative bottom-up compositional model, i.e. SOM-SD [6], cannot effectively discriminate the structures in a way that reflects the underlying THTMM generative process. This is probably due to the fact that SOM-SD is biased to discriminate trees based on their structure while, in this dataset, structure does not provide sufficient information to tell the classes apart. In fact, the only way to discriminate the structures in this dataset, is to learn the Markovian dependencies among the observations corresponding to the node labels. GTM-SD is capable of achieving this by means of the underlying SP-BHTMM model, which can capture such generative dynamics even if the originating process is top-down. From a theoretical point of view this is not surprising, as it is known that the class of trees that can be recognized by a deterministic top-down automata is a proper subset of the tree languages recognizable by bottom-up automata [24].

The second data set from [7] comprises 600 images generated by the *Traffic Policeman Benchmark* (TPB) software [25], that provides an artificial domain for evaluating learning algorithms for structured data. The dataset consists of images categorized into 12 classes (50 pictures per class) that resemble traffic policemen, houses and ships of different spatial/chromatical configurations that are generated by a rule based grammar. Images are represented as trees such that connected components in each image have a parent-child relationship, the object located lower and closer to the left edge being the parent. Each node is labeled by a two-dimensional vector that represents the center of gravity of the component that node stands for.

Figure 8 shows the topographic mapping obtained by GTM-SD: symbols identify different classes following the notation in [7]. GTM-SD achieves a clear separation between the structures of the 3 top-classes, whereas it does not differentiate at all between the two subgroups in the Policeman class

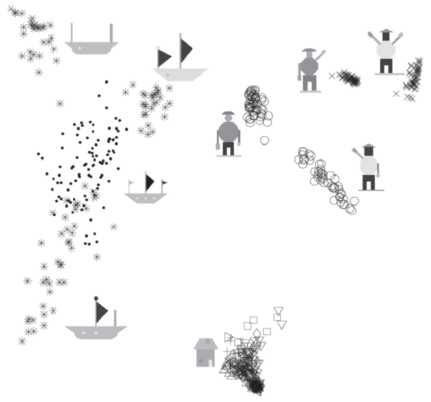
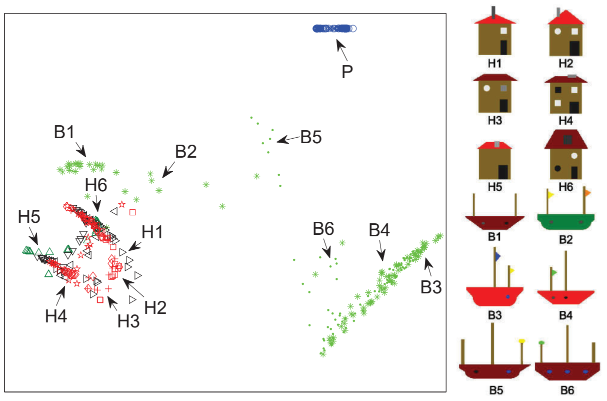


Fig. 7. Topographic mapping obtained by GTM-HTMM on the TPB data: this image is an authorized reproduction of Fig. 7.b in [7]. Policeman trees are well separated, while the house structures appear consistently mixed.

(marked as P in Fig. 8). These subgroups include trees that are structurally identical, being differentiated only by the leaves emission: bottom-up processing fails to separate the two subgroups because, as shown in the previous experiment, it tends to discriminate more based on nodes closer to the root. Conversely, a top-down model separates more based on the information of nodes close to the leaves: in fact, GTM-HTMM yields to a better topographic mapping of the policeman subgroups (see Fig. 7, reproducing the original map in Fig. 7.b of [7]), that is also superior to that achieved by the bottom-up SOM-SD [7].

Like GTM-HTMM, GTM-SD separates the subgroup corresponding to ships missing the central mast, that are $B1$ and $B2$, where the latter further differentiates those ships that have a larger hull. Subgroups $B5$ and $B6$ correspond to ships with three masts with an unbalanced structure, i.e. with sails only on the rightmost and leftmost mast, respectively. The remainder of the ship subgroups are quite mixed in the large cluster on the right, whereas GTM-HTMM can better separate the structures corresponding to ships missing the left or right mast (see Fig. 7). On the other hand, the GTM-HTMM fails to find any type of topographic ordering in the house class. Conversely, GTM-SD provides some partial topographic organization based on the presence and position of the chimney. The chimney appears and its position moves upwards as one goes from the bottom to the top of the map: $H3$ comprises houses without a chimney, $H4$ includes houses with a low chimney positioned on the right, while $H1$ and $H2$ comprise houses with non-right chimneys having one and two windows, respectively. The large and mixed groups $H6$, on the other hand, maps to houses with an higher center of gravity for the chimney. Chimneys appear to be discriminative both for their structural relevance, given that they increase the tree depth from 2 to 3, as well as because GTM-SD associates the different chimney positions with separate leaf priors, whose effect propagates well to the root in the shallow house structures.

One of the main advantages of a compositional approach such as GTM-SD over a *monolithic* model such as GTM-HTMM is the possibility of projecting and visualizing the substructures within the dataset, which allows a clearer under-



(a) (b)

Fig. 8. GTM-SD Topographic mapping of the TPB dataset in [7]: topographic map (a) and prototypical images for the discovered subgroups (b).

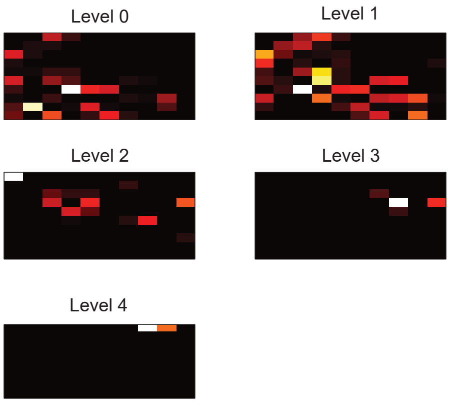


Fig. 9. Topographic maps for the substructures in the TPB dataset: level numbers refer to the distance from the leaves. Hotter colors denote a larger responsibility (white and black denote high and low activation, respectively).

standing of the topographic organization of the latent space. Figure 9 shows the responsibility maps for the TPD task: plots are organized based on distance from the leaves, e.g. *level 1* denotes subtrees that are rooted in nodes that are parents of a leaf. The map shows a clear topographic organization of the space that recall that obtained by SOM-SD [6]: leaves are projected on the bottom left while subtrees with depth 2 are mapped close-by. More articulated substructures are projected in an orderly manner on the top of the map, with depth increasing from the left to right. This is coherent with the topographic organization in Fig. 8, where shallower structures are mapped to the bottom-left with depth increasing as their projection moves to the top of the map, which is responsible for deeper structures.

Generally speaking, the bottom-up approach, postulated by GTM-SD, and the top-down approach, taken by GTM-HTMM, provide two complementary views over the structured data. The suitability of one approach over the other is application dependent, such that for some tasks and/or datasets one approach can be superior to the other and that for other tasks and/or datasets the opposite can be true. Nevertheless, there are some clear properties associated with the GTM-SD and GTM-HTMM models that we would like to discuss here to complement their experimental comparison.

From a modeling point of view, GTM-SD enforces a Markovian organization of the topographic map that is associated to a unique, large hidden Markov model that can generate all possible structures and substructures in the dataset. Trees are modeled as compound entities whose constituents follow complex causal relationships induced by the structural properties. Conversely, GTM-HTMM defines multiple, separate hidden Markov models that are responsible for the generation of complete (atomic) observed trees. Such modeling difference can have an impact on the convergence of the learning process: as noted by the authors in [7], the GTM-HTMM model is prone to slow convergence and local optima, which is due to the approximations introduced in the EM algorithm to generate the parameters of the HTMM noise models from the smooth mapping. Conversely, the EM algorithm for GTM-SD has an exact closed form solution (at least for Gaussian emissions, see Sect. III-B) that is more likely to converge to global optimum.

The GTM-SD approach allows a more effective information sharing than GTM-HTMM as regards substructures that are common to different sample trees. As shown in Fig. 9, GTM-SD straightforwardly yields to a topographic projection for every substructure in the dataset, whereas in top-down GTM-HTMM this has to be enforced by explicitly training the model on an augmented dataset comprising a tree for each different (proper) substructure in the original data. However, even with such an augmented dataset, the GTM-HTMM still considers each tree and proper subtree as a separate atomic entity, without information sharing among the HTMM noise models generated by different latent points. Consider an example tree $\mathbf{y}$ which contains two non-trivial subtrees $\mathbf{y}_a$ and $\mathbf{y}_b$. These structures may be generated by 3 different latent points, corresponding to 3 separate HTMMs with different distribution parameters: hence the results of the upwards-downwards procedure of one HTMM cannot be reused to simplify the computation of another HTMM. For instance, the HTMM model responsible for $\mathbf{y}$ will generate by itself the whole tree, irrespectively of the fact that some latent point on the map may be capable of generating the substructures $\mathbf{y}_a$ and $\mathbf{y}_b$. This lack of information sharing prevents GTM-HTMM from achieving compositionality.

Computationally, GTM-SD is asymptotically less efficient than GTM-HTMM for small structures, whereas we expect it to be more efficient and scalable for large-size data, as GTM-HTMM would need to train C (i.e. one for each of the GTM latent points) large HTMM models with K hidden states, while GTM-SD naturally accommodates such structures in its large constrained space of C hidden states. For instance, the total number of parameters of the positional state transition matrices of a GTM-HTMM with C latent points is order of $M_{htmm} = C(K^2 \cdot L)$, where K is the number of hidden states of the single HTMM noise models. The corresponding number of parameters for GTM-SD is $M_{sd} = C^2 \cdot L$, where C is both the number of hidden states and the number of latent points on the map. Usually, the number of latent points on the topographic map is chosen about $C = 100$ to ensure a sufficient map resolution (in both the approaches). The number of hidden states required to capture the structural information into the Markov model grows with the complexity of the trees. For instance, a minimum of $10 - 20$ hidden states is required to capture enough information from real world trees such as the INEX 2005 data [26] discussed in the following subsection. For a map of $C = 100$ latent points, a GTM-HTMM with $K = 10$ hidden states in its noise models has the same number of state transition parameters then the corresponding GTM-SD. The latter, however, has a much larger state space of $C = 100$ states that can capture more structural information. For $K > 10$, GTM-HTMM needs a larger number of parameters than GTM-SD.

By its compositional nature, GTM-SD can also better reuse information from shared substructures, achieving a more effective information compression than GTM-HTMM. Notice that augmenting the dataset to obtain substructure projection consistently further increases the computational costs of GTM-HTMM with respect to GTM-SD. In fact, the GTM-HTMM learning process has to be iterated over a larger dataset including all proper subtrees in the original data, while GTM-SD can straightforwardly achieve such projection by parsing only the non-extended dataset.

C. Experimental Comparison with SOM-SD

In this section, we evaluate GTM-SD against the SOM-SD model [6], a recursive neural network for structured data also founding on a bottom-up approach. The experimental comparison is based on two datasets: the (m-db-s-0) corpus, that has been used for the 2005 INEX competition for structured data classification and that has been won by the SOM-SD model [26], and the Melody corpus [27].

The INEX 2005 dataset comprises 9361 XML formatted documents represented as trees with maximum outdegree $L = 32$, labeled by 11 thematic categories, with consistently varied class distributions, such that node labels represent 1 out of 366 possible XML-tags. Standard splits into training and test sets are available for both datasets [6], such that INEX 2005 comprises 4820 training structures and 4811 test samples. Node labels for the INEX data represent categorical information which, in our generative framework, is best modeled by a multinomial emission distribution, resulting in the μGTM-SD model described by equations (13)-(16). As discussed in Section III-B, parameter update for such a constrained multinomial emission model requires to perform a gradient descent loop regulated by a learning step δ. A preliminary experimental evaluation has been performed to study the behavior of μGTM-SD learning as a function of the δ metaparameter. In particular, a 7×7 μGTM-SD has been trained for 50 epochs on the INEX 2005 data (training set only), using different δ values from $\{0.005, 0.001, 0.0005, 0.0001\}$ and with an inner multinomial learning loop of 100 iterations. No information concerning tree classification has been used to evaluate the metaparameter choice: rather, only learning stability has been taken into account by evaluating the log-likelihood behavior as a function of the learning epochs. Figure 10 shows the log-likelihood corresponding to the different δ values: values

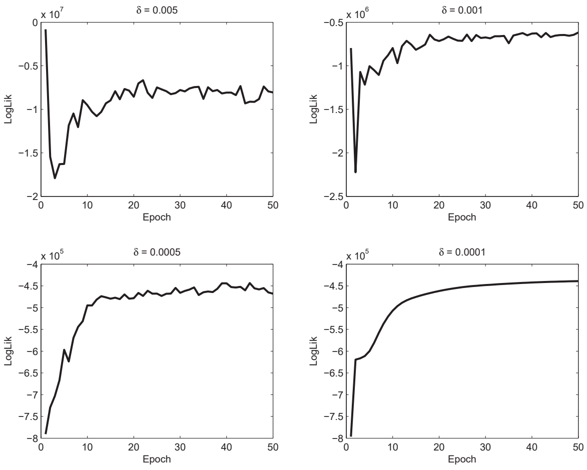


Fig. 10. Behavior of the log-likelihood for different values of the learning step size δ in the inner learning loop of the multinomial μGTM-SD (see (15)). Log-likelihood refers to a 7×7 μGTM-SD trained on the INEX 2005 data for 50 epochs, with an inner multinomial learning loop of 100 iterations. Values of $\delta \geq 0.01$ have been tested and result in strong numerical problems, with impaired learning convergence.

of $\delta > 0.0001$, yield to unstable learning with oscillating log-likelihoods, while for $\delta = 0.0001$ the log-likelihood shows the expected smooth behavior, which suggest to use this value (and a maximum of 100 inner loop iterations) for the experimental comparison with SOM-SD. Notice that the EM process is expected to produce a monotonically increasing likelihood: the decreasing parts of the curves that can be observed in Fig. 10 for $\delta > 0.0001$ are due to the gradient descent loop that is used to approximate the update of the constrained multinomial emission. Such gradient descent loop is not part of the EM and produces the non-monotonic behavior due to the well known numerical problems introduced by large learning rates. An adequate choice of the δ values restores the correct monotonic behavior (e.g. see $\delta = 0.0001$ in Fig. 10). Larger learning rates can be used if supported by an appropriate exponential decay policy that reduces the learning step as the iterations of the gradient descend loop progress.

Classification performance on the INEX 2005 data has been evaluated for varying configurations of the neural and generative topographic maps: [28] describes an extensive validation of SOM-SD performance for varying network configurations (i.e. 45 networks), that include different map sizes, maximum number of training iterations as well as several values of the metaparameter μ, which regulates the structure/label tradeoff (values of μ close to 1 give more credit to the label over the structural part). For the purpose of this paper, we confront GTM-SD with the top-5 SOM-SD configurations in [28] for each of the map-sizes. Table II shows the metaparameters values and the corresponding classification error on the INEX 2005 test set: notice how, coherently with the results in [28], SOM-SD shows a marked sensitivity with respect to the choice of the metaparameters. Classification on a *fresh* test tree is obtained by a simple 1 nearest neighbor (1-NN) rule on a SOM-SD fitted to the training set [28].

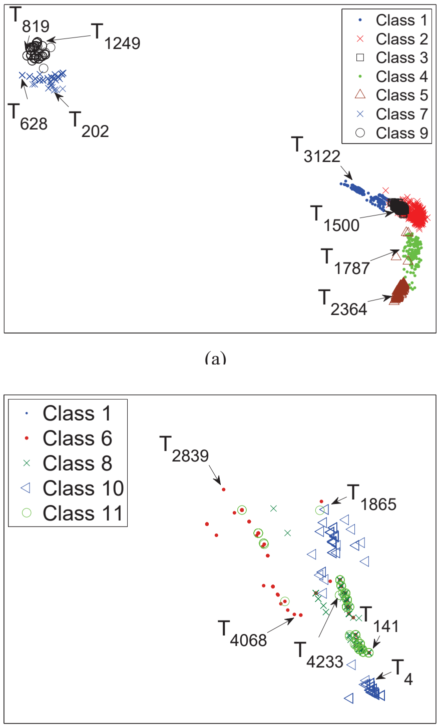
As regards μGTM-SD, we have tested 5 positional configurations corresponding to maps with a grid of 7×7, 9×9, 10×10, 15×15 and 20×20 latent points. The number of maximum EM iterations has been set to 50, as experimental evidence shows that learning reaches stable log-likelihood levels after 30-35 epochs: values of the maximum number of training iterations above such threshold do not yield to substantial variations in the results. Other relevant GTM-SD metaparameters are the maximum number of inner loop iterations and the learning step δ, that are set to 100 and 0.0001, respectively, based on the results in Fig. 10. As any EM process, GTM-SD is prone to performance fluctuations due to different initial (random) assignments of the model parameters. To account for this effect, performance results, for each map-size, are based on 5 repetitions of the training-test procedure with different random initialization of the parameters.

TABLE II
CLASSIFICATION ERROR (I.E. PERCENTAGE OF INCORRECTLY CATEGORIZED TEST TREES) FOR DIFFERENT CONFIGURATIONS OF THE SOM-SD MAPS ON THE INEX 2005 TASK[28]

Map #	Size	Training iterations	μ	Best test error
1	55×40	32	0.25	32.488
2	55×40	128	0.85	22.51
3	77×56	32	0.65	18.62
4	110×80	32	0.05	12.62
5	110×80	128	0.85	8.65
			Average	18.9776
			Std Dev	9.2495

Figure 11 shows the root projection on the μGTM-SD map for trees in the INEX 2005 training set. Two areas of the μGTM-SD map are responsible for generating the roots, that are the top-left and the bottom-right corner (visualized separately in Fig. 11). Example trees are labeled on the map with their dataset index (numbering follows that of the original data [6]). The first portion in Fig. 11.a shows a clear organization with trees on the top-left corner (e.g. T_{819}, T_{202}, $\ldots$) corresponding to shallow structures (i.e. depth $2-3$) with small outdegree (i.e. $4-5$), while trees on the bottom-right (e.g. T_{3122}, T_{1787}, $\ldots$) correspond to deeper structures. In particular, trees close to T_{3122} have the most complex structure (e.g outdegree 32 and depth 5) which becomes increasingly simpler as one moves to the bottom-right corner, where T_{1500} is shallower than T_{3122} but has a larger outdegree than T_{1787}, and T_{2364} has the smallest outdegree. Trees in Fig. 11.b correspond to medium-sized structures that are smaller than those on the bottom-right of Fig. 11.a, having a maximum outdegree of 10. They become increasingly deeper and with a more complex structure on the deep levels as one moves from top-left (T_{2839}) to bottom-right (T_4).

Table III shows the μGTM-SD classification errors on the training and test sets, averaged across the 5 training-test repetitions. Class predictions on the test set for μGTM-SD have been obtained by a simple 1-NN nearest neighbor rule in the bi-dimensional latent space: i.e. a fresh tree is categorized with the class of its nearest training tree projected on the map. Training set classification is based on the second nearest neighbor in the training set (the closest match would be, clearly, the sample itself): Table III shows that test and training error have a comparable magnitude, suggesting that the learned models do not show significant overfitting. Further,



(a)

(b)

Fig. 11. μGTM-SD maps for the INEX 2005 dataset showing root projection for trees in the training set. Two separate areas of the map are used to project roots, that is the top-left corner (a) and bottom right corner (b), which are visualized separately for the sake of clarity.

TABLE III
CLASSIFICATION ERROR FOR DIFFERENT CONFIGURATIONS OF THE μGTM-SD MAPS ON THE INEX 2005 TASK: TRAINING AND TEST ERRORS ARE AVERAGED ACROSS 5 REPETITIONS.

Map #	Size	Training error	Mean test error (dev)
1	7×7	8.44	10.27 (1.74)
2	9×9	9.16	10.60 (1.55)
3	10×10	6.96	8.36 (0.57)
4	15×15	8.16	7.84 (1.01)
5	20×20	5.55	7.42 (0.40)
	Average	7.65	8.90
	Std Dev	1.42	1.44

Table III shows that, overall, the standard deviation of the test error between the 5 repetitions is not significant, witnessing a good robustness of the model with respect to parameter initialization. Results show that GTM-SD reaches a competitive classification performance already with small 7×7 maps, that achieve best classification accuracies, comparable to that of the top 110×80 SOM-SD map which has roughly 70% more free parameters than a 7×7 positional GTM-SD.

The Melody corpus [27] is also based on real-world data, such that each tree represents a music piece, where node labels identify the notes being played, while the depth of the node defines the duration of the corresponding note (i.e. the shorter a note the deeper it is in the tree). The corpus comprises 420 monophonic themes of 20 worldwide well known themes of different musical genres. Each theme has been played by different musicians with different embellishments and variations due to performance errors: for each of the 20 original scores, 21 different variations have been built (all of them with 4/4 meter signature) [27]. Again this is a classification task, where each tree has to be assigned to the correct theme out of the 20 possible. The tree labels come from a multinomial alphabet of 12 symbols (notes) and the maximum outdegree in the dataset is $L = 8$. The dataset is provided by the authors in a standard 3-fold split, where each split contains 280 training trees and 140 test structures. As for the INEX 2005 data, we have tested several SOM-SD and μGTM-SD configurations. In this case, only a single data fold has been used to select the best performing model. The SOM-SD map sizes and meta-parameters used for INEX 2005, and shown in Table II, have been tested; similarly, we have tested different sizes of μGTM-SD maps as in Table III. Table IV reports the classification error on the best SOM-SD and μGTM-SD maps, averaged over the 3 test sets in the folds (also standard deviation is reported). Again, μGTM-SD shows a competitive classification performance with respect to SOM-SD, with a lower classification error. The moderate standard deviation of the former model confirms that μGTM-SD is a robust model that is not subject to abrupt performance variations for different training data.

TABLE IV
CLASSIFICATION ERROR FOR THE BEST SOM-SD AND μGTM-SD MAPS ON THE 3 FOLDS OF THE MELODY CORPUS [27].

Model	Mean test error	Std. Dev.
SOM-SD	42.65	12.32
GTM-SD	40.48	3.59

V. CONCLUSIONS

Compositionality is a fundamental property when dealing with tree structured data, as it closely reflects the compound nature of hierarchical information. Instead of evaluating complex structures as whole atomic entities, throughout a compositional approach we are allowed to assess them in terms of their constituent elements. We have introduced GTM-SD, the first generative approach to compositional topographic mapping of tree-structured data that founds on a scalable bottom-up hidden tree Markov model, named SP-BHTMM and proposed in Part I of this paper [2]. SP-BHTMM circumvents the typical strong computational requirements imposed by the exploding state space of a bottom-up state transition through the use of a (finite) mixture of multinomials approximation, allowing for parameter learning and inference procedures of the same computational class of its top-down counterpart.

GTM-SD enforces a Markovian organization of the latent space serving as topographic map, defining a unique, large hidden Markov model that can generate all possible structures and substructures in the dataset. By this means, trees can be modeled as compound entities whose constituents follow complex causal relationships induced by the structural properties, rather than being considered atomic i.i.d. samples as in the top-down GTM-HTMM [7]. The substructure information sharing capability of GTM-SD straightforwardly yields to a topographic projection for every substructure in the dataset. Substructure projection is a fundamental capability when addressing the exploratory analysis of collections of tree-structured data, as it allows to determine which substructures are shared between different trees by simply inspecting their projections on the 2D topographic map. Such an intuition has motivated the development of a novel adaptive kernel for structures that exploits Euclidean distances among projections on the GTM-SD map [29] and that has state-of-the-art performance on classification of tree-structured data.

Experimental results have shown that GTM-SD can effectively generate topographically ordered maps of the sample trees and their substructures, with competitive performance with respect to both the top-down generative GTM-HTMM as well as against the recursive neural approach of SOM-SD. With respect to the latter, the experimental results highlight that the continuous topographic map generated by the smooth generative mapping of GTM-SD allows a finer grained discrimination among the projected structures.

An issue that has emerged from the experimental analysis is related to regulating the GTM-SD trade-off between structure-based and label-based discrimination. The probabilistic formulation of the model may itself provide well-founded tools to address this problem. For instance, following the experimental intuition, it might be interesting to explore the use of Bayesian priors to fine tune the smoothness of the emission, thus regulating the model bias with respect to the structure. Further, it will be interesting to study the effect of adding contextual information to substructure projection. In this paper, we have focused on compositionality and, to this end, we have taken into consideration only the compositional projection mode based on the upwards parameter β_i. However, as discussed in Section III-C, GTM-SD defines a straightforward means for contextualizing the projection of a subtree through the use of the posterior distribution, at the little cost of an additional downwards recursion. Notice that neither GTM-HTMM nor SOM-SD offer contextualized projections since the former, does not allow projection of substructures, while the latter is a purely compositional model where context information only flows from the leaves to the root.

ACKNOWLEDGMENT

The authors would like to thank Nikolaos Gianniotis, for having provided experimental data and pictures from [7], and Giovanni Da San Martino, for his support with the SOM-SD experiments. The work by D. Bacciu was supported in part by the University of Manchester and LJMU. The work by Alessandro Sperduti was supported by the Italian Ministry of Education, University, and Research (MIUR) under Project PRIN 2009 2009LNP494_005.

REFERENCES

[1] M. Diligenti, P. Frasconi, and M. Gori, "Hidden tree markov models for document image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 4, pp. 519–523, 2003.

[2] D. Bacciu, A. Micheli, and A. Sperduti, "Compositional generative mapping for tree-structured data - part I: Bottom-up probabilistic modeling of trees," *IEEE Trans. on Neural Netw. and Learn. Sys.*, In Press.

[3] C. Bishop, M. Svensén, and C. Williams, "GTM: The generative topographic mapping," *Neural Comput.*, vol. 10, no. 1, pp. 215–234, 1998.

[4] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert, "A general framework for unsupervised processing of structured data," *Neurocomputing*, vol. 57, pp. 3–35, 2004.

[5] ——, "Recursive self-organizing network models," *Neural Networks*, vol. 17, no. 8-9, pp. 1061–1085, 2004.

[6] M. Hagenbuchner, A. Sperduti, A. Tsoi *et al.*, "A self-organizing map for adaptive processing of structured data," *IEEE Trans. Neural Networks*, vol. 14, no. 3, pp. 491–505, 2003.

[7] N. Gianniotis and P. Tino, "Visualization of Tree-Structured Data Through Generative Topographic Mapping," *IEEE Trans. on Neural Netw.*, vol. 19, no. 8, pp. 1468–1493, 2008.

[8] M. Crouse, R. Nowak, and R. Baraniuk, "Wavelet-based statistical signal-processing using hidden markov-models," *IEEE Trans. Signal Process.*, vol. 46, no. 4, pp. 886–902, April 1998.

[9] C. Bishop, G. Hinton, and I. Strachan, "GTM through time," in *Proc. of the 5th Int. Conf. on Artif. Neural Networks*, Jul 1997, pp. 111–116.

[10] D. Bacciu, A. Micheli, and A. Sperduti, "Compositional generative mapping of structured data," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, 2010, pp. 1–8.

[11] T. Kohonen, *Self-organization and associative memory*. Springer-Verlag New York, Inc. New York, NY, USA, 1989.

[12] M. Varsta, J. Heikkonen, J. Lampinen, and J. Millán, "Temporal Kohonen map and the recurrent self-organizing map: analytical and experimental comparison," *Neural processing letters*, vol. 13, no. 3, pp. 237–251, 2001.

[13] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski, "Recurrent SOM with local linear models in time series prediction," in *6th Europ. Symp. on Artif. Neural Netw.* Citeseer, 1998, pp. 167–172.

[14] T. Voegtlin, "Recursive self-organizing maps," *Neural Netw.*, vol. 15, pp. 979–991, October 2002.

[15] P. Tiňo, I. Farkaš, and J. van Mourik, "Dynamics and topographic organization of recursive self-organizing maps," *Neural Computation*, vol. 18, no. 10, pp. 2529–2567, 2006.

[16] M. Strickert and B. Hammer, "Merge som for temporal data," *Neurocomputing*, vol. 64, pp. 39 – 71, 2005, trends in Neurocomputing: 12th European Symposium on Artificial Neural Networks 2004.

[17] T. M. Martinetz and K. J. Shulten, "A "neural-gas" network learns topologies"," in *Artificial Neural Networks*, T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, Eds., 1991, pp. 397–402.

[18] R. Boulet, B. Jouve, F. Rossi, and N. Villa, "Batch kernel som and related laplacian methods for social network analysis," *Neurocomputing*, vol. 71, no. 7-9, pp. 1257 – 1273, 2008.

[19] A. Gisbrecht, B. Mokbel, and B. Hammer, "Kernel generative topographic mapping." in *Procs. of the 18th European Symposiun on Artificial Neural Networks (ESANN 2010)*, Apr 2010, pp. 277–282.

[20] I. Olier, A. Vellido, and J. Giraldo, "Kernel generative topographic mapping." in *Procs. of the 18th European Symposiun on Artificial Neural Networks (ESANN 2010)*, Apr 2010, pp. 481–486.

[21] P. Vanco and I. Farkas, "Experimental comparison of recursive self-organizing maps for processing tree-structured data," *Neurocomputing*, vol. 73, no. 7-9, pp. 1362–1375, 2010.

[22] C. Bishop, M. Svensen, and C. Williams, "Developments of the generative topographic mapping," *Neurocomp.*, vol. 21, no. 1-3, pp. 203–224, 1998.

[23] A. Kabán and M. Girolami, "A combined latent class and trait model for the analysis and visualization of discrete data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, pp. 859–872, August 2001.

[24] J. L. Verdu-Mas, R. C. Carrasco, and J. Calera-Rubio, "Parsing with probabilistic strictly locally testable tree languages," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 7, 2005.

[25] M. Hagenbuchner and A. Tsoi, "The traffic policeman benchmark," in *European Symposium on Artificial Neural Networks*, M. Verleysen, Ed., Apr 1999, pp. pp. 63–68.

[26] L. Denoyer and P. Gallinari, "Report on the XML mining track at INEX 2005 and INEX 2006: categorization and clustering of XML documents," *SIGIR Forum*, vol. 41, no. 1, pp. 79–90, 2007.

[27] J. F. Bernabeu, J. Calera-rubio, J. M. Iesta, and D. Rizo, "A probabilistic approach to melodic similarity," in *Proc. of the 2nd Int. Work. on Machine Lear. and Music (MML)*, 2009, pp. 48–53.

[28] F. Aiolli, G. Da San Martino, M. Hagenbuchner, and A. Sperduti, "Learning nonsparse kernels by self-organizing maps for structured data," *IEEE Trans. on Neural Netw.*, vol. 20, no. 12, pp. 1938 –1949, 2009.

[29] D. Bacciu, A. Micheli, and A. Sperduti, "Adaptive tree kernel by multinomial generative topographic mapping," in *Proc. of the 2011 Int. Joint Conf. on Neural Networks (IJCNN'11)*. IEEE, 2011.