Visualisation of Tree-Structured Data through Generative Topographic Mapping

Nikolaos Gianniotis, Peter Tiňo

***Abstract*—We present a probabilistic generative approach for constructing topographic maps of tree-structured data. Our model defines a low dimensional manifold of local noise models, namely (hidden) Markov tree models, induced by a smooth mapping from low dimensional latent space. We contrast our approach with that of topographic map formation using recursive neural-based techniques, namely the Self-Organising Map for Structured Data (SOMSD) [1]. The probabilistic nature of our model brings a number of benefits: (1) naturally defined cost function that drives the model optimisation; (2) principled model comparison and testing for overfitting; (3) a potential for transparent interpretation of the map by inspecting the underlying local noise models; (4) natural accommodation of alternative local noise models implicitly expressing different notions of structured data similarity. Furthermore, in contrast with the recursive neural-based approaches, the smooth nature of the mapping from the latent space to the local model space allows for calculation of magnification factors - a useful tool for the detection of data clusters. We demonstrate our approach on three datasets: a toy dataset, an artificially generated dataset and on a dataset of images represented as quadtrees.**

***Index Terms*—Topographic mapping, structured data, hidden Markov tree model.**

I. Introduction

TOPOGRAPHIC visualisation is a valuable tool for the analysis and interpretation of multivariate data. The Self Organising Map (SOM) [2] is one of the most celebrated tools that is of vast assistance to this task and has become a paradigm inspiring numerous extensions. SOM is a type of neural network that allows a nonlinear projection of data residing in a high dimensional space to a lower dimensional projection space. The lower dimensional space is a discrete lattice of neurons (for visualisation purposes a two dimensional lattice). According to the SOM paradigm, the formation of the map is realised by iterating two steps of competition and cooperation among the neurons. The competition step involves the presentation of an input pattern and calculation of the response of all neurons. The neurons are associated with weights (codebook vectors). The response of a neuron is measured as the Euclidean distance between its weight and the input pattern. The neuron with the greatest response is the winner of the competition. In the cooperation step, the winning neuron is appropriately adjusted to increase its future response to that particular pattern. Moreover, neurons that belong to the neighbourhood (on the lattice) of the winner are also adjusted to increase their future response, albeit proportionally to a (usually) exponentially decaying distance from the winner.

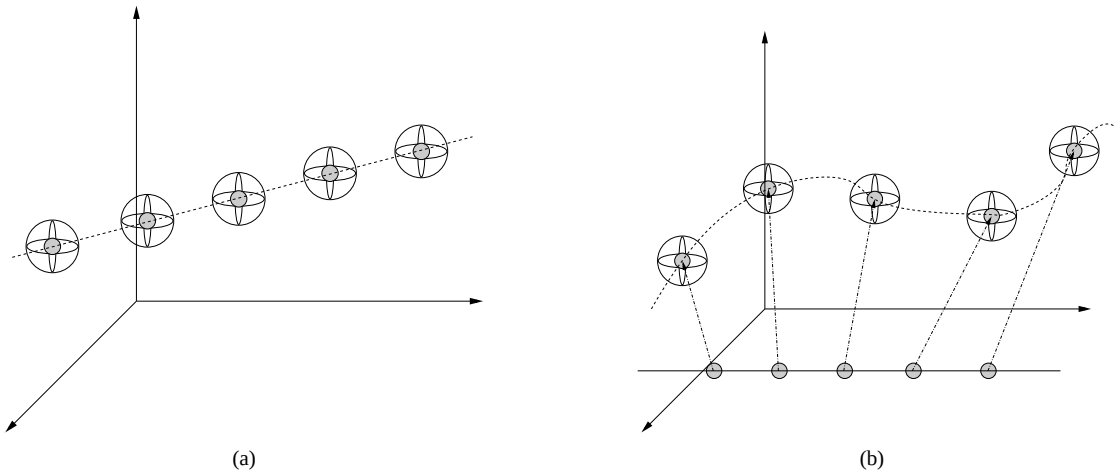
N. Gianniotis and P. Tiňo are with the School of Computer Science, The University of Birmingham, Birmingham, B15 2TT, United Kingdom (e-mail: nxg,pxt@cs.bham.ac.uk).

The heuristic nature of SOM inherently brings about certain limitations, for example the lack of a principled cost function (although see developments in e.g. [3][1]). Comparison of map formations resulting from different initialisations, parameter settings, or optimisation algorithms can be problematic.

The Generative Topographic Map (GTM) algorithm [4] was introduced as a principled probabilistic analog to SOM. As a generative model GTM realises a "noisy" low dimensional manifold in a high dimensional data space. It can be used to model a given training dataset by adjusting its parameters so that the model-generated data lying around the noisy low dimensional manifold match (in the distribution sense) the training data. GTM is a mixture of local generative models (spherical Gaussians) that adheres to topological constraints (constraints on the values that means of the Gaussians can take). A simple example is that of requiring that the means belong to a straight line. This could be useful if we believed that the data are intrinsically one-dimensional and are adequately represented by a "noisy line". This situation is illustrated in Fig.1(a). The line on which the means of local Gaussians are placed can be viewed as an image of a one-dimensional interval under a linear (affine) map. Alternatively, one may want to constrain the Gaussian means to lie on a smooth curve. In that case, the one-dimensional interval would be embedded in the high dimensional data space through a smooth non-linear mapping. This is illustrated in Fig.1(b). The GTM belongs to the class of so called latent-variable models with latent space being the one-dimensional interval through which the Gaussian means are constrained.

SOM has been extended in various ways to deal with non-vectorial forms of data, such as sequences or trees [5], [6]. Several modifications of SOM equip standard SOM with additional feed-back connections that allow for natural processing of recursive data types. Typical examples of such models are Temporal Kohonen Map [7], recurrent SOM [8], feedback SOM [9], recursive SOM [10], merge SOM [11], SOM for structured data [1] and contextual SOM for structured data [12]. These models rely on the same principles of competition and cooperation that govern the SOM formation. Again, formulation of a principled cost function is problematic (although see developments in [13] along the lines of [3]). Also problematic is explanatory interpretation of the visualisation results in such approaches. Clusters may be formed on the map that indicate some close relationship between the concerned structured data items, but there is no explanation on what the

[1]Heskes [3] suggests a modified version of SOM by redefining the codebook vector (winner unit) associated with an input as the one closest to the input with respect to *averaged* distance across its local neighbourhood on the codebook lattice.



(a) (b)

Fig. 1. Spherical Gaussians constrained on a one-dimensional line (a), spherical Gaussians constrained on a one-dimensional curve (b). Note that the straight line (latent space) in (b) does not belong to the data space and is only plotted in the same figure as its image for convenience.

characteristics of the clusters are. Of course one can inspect the individual data points to deduce those relationships once the map has been formed, but reasoning about mapping of new data items (not used for model fitting) can be still challenging.

In this paper, we extend GTM to the visualisation of tree-structured data. We contrast this extension with recursive neural-based approaches and point out potential benefits of a principled probabilistic model-based formulation. For example, the generative nature of our model formulation provides us with an explanatory insight as to how the data might have been generated. By observing the generative process and its parameters we can understand characteristics of clusters of projected data items and/or discern other patterns in the data. Also, the smooth character of the embedding map from the latent space into the model space enables us to use techniques of information geometry to characterise areas on the map of potentially clustered data by calculating local expansion/contraction rates in the statistical manifold of local models. Such knowledge is highly desirable for topographic map understanding, but is impossible to obtain in a principled manner form recursive extensions of SOM. As a candidate member of the recursive neural-based techniques, we use the SOM for structured data (SOMSD) [1].

The paper is organised as follows. Section II gives a brief overview of recursive neural-based approaches for visualising structured data. In Section III we present the GTM algorithm. The class of local generative models used in this paper, the hidden Markov tree model, is introduced in Section IV and its use in extending the GTM to tree-structured data is presented in Section V. We present our experiments and results in Section VI. As a demonstration of the ease of extensibility of this approach, in Section VII we present another local generative model, the Markov tree model, that is used to derive an alternative extension of GTM in the same setting of tree-structured data. Section VIII introduces magnification factors for the GTM extension based on hidden Markov tree models, a useful tool for identifying clusters on the visualisation plot, and experimental results are presented in Section VIII-C. Section IX considers magnification factors for the alternative extension of GTM. A discussion of the presented work follows in Section X. Commonly used notation is summarised in Table IV.

II. Recursive neural approaches to topographic map formation of structured-data

The *original SOM* [2] has inspired various extensions that deal with data of non-vectorial types. An excellent overview of those under a general framework can be found in [13]. The following techniques try to capture the structure of the data, by introducing a notion of *context* that is updated in a recursive manner and is supposed to represent data items processed until the current processing step. Neurons are arranged on a regular 2-dimensional lattice (for visualisation purposes). The same notions of a learning rate η and a neighbourhood function h defined on pairs of neurons on the map, are inherited from SOM:

$$h(i, I(t)) = e^{(-\frac{dist(i, I(t))}{\sigma^2})}, \quad (1)$$

where $dist$ is the distance of neurons i and $I(t)$ on the map and σ controls the neighbourhood size. $I(t)$ denotes the winner neuron at time t. Parameters η and σ are decreased with time to allow for topographic convergence as in SOM [2].

The *Temporal Kohonen Map (TKM)* [7], *Recurrent SOM (RSOM)* [14] and *Recursive SOM (RecSOM)* [10] are SOM extensions designed for the processing of sequences $[\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_T]$ over $\mathbb{R}^d$. In TKM and RSOM each neuron i is equipped with a weight vector $\boldsymbol{w}_i \in \mathbb{R}^d$. In each neuron, one input item is processed at each time step t in the context given by the past activations of that neuron. When a new input is presented, the neurons do not lose their past activity immediately as in SOM, but the context information decays gradually. The gradual decay is controlled by a parameter $\alpha \in (0, 1)$. However, RSOM modifies TKM by summing the deviation of the weights $\boldsymbol{w}_i$ as opposed to distances. The corresponding activations are listed in Table I.

RecSOM takes into account the context of inputs by explicitly augmenting each unit i with a context vector $\boldsymbol{c}_i \in \mathbb{R}^q$ that

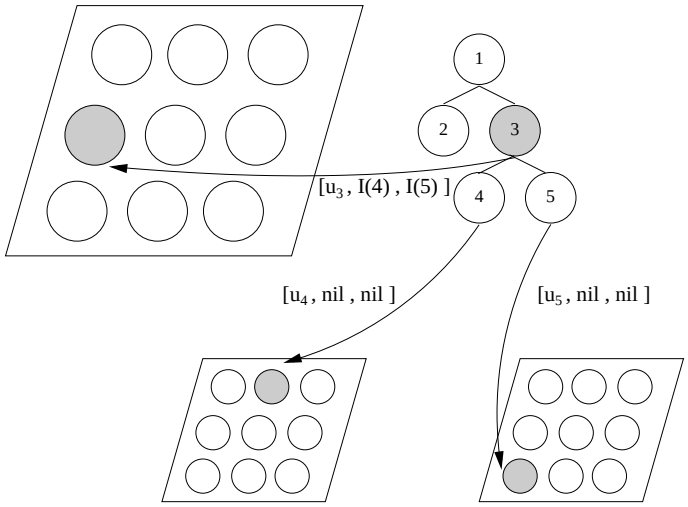


Fig. 2. Activation for label $\boldsymbol{u}_3$ of a tree-pattern: Activation is calculated bottom up, thus the children of input node 3 are processed beforehand. Since nodes 4 and 5 are leaf nodes their contexts are filled in with the special $\boldsymbol{nil}$ vector. The winner neurons $I(4)$ and $I(5)$ of the input labels $\boldsymbol{u}_4$, $\boldsymbol{u}_5$ of nodes 4 and 5 respectively are supplied as the context for node 3.

TABLE I
ACTIVATIONS FOR SOM EXTENSIONS.

Name	Activation
TKM	$y_i(t) = \alpha\|\boldsymbol{s}_t - \boldsymbol{w}_i\|^2 + (1-\alpha)y_i(t-1)$
RSOM	$\boldsymbol{y}_i(t) = \alpha(\boldsymbol{s}_t - \boldsymbol{w}_i) + (1-\alpha)\boldsymbol{y}_i(t-1)$
RecSOM	$y_i(t) = \alpha\|\boldsymbol{s}_t - \boldsymbol{w}_i\|^2 + \beta\|[exp(-y_1(t-1)), \ldots, \exp(-y_N(t-1))]^T - \boldsymbol{c}_i\|^2$

stores the activations $\boldsymbol{y}(t-1) = [y_1(t-1) \ldots y_N(t-1)]^T$ of all the units in the map at the previous time step. Its activation function in Table I uses two positive constants α and β to control the contribution of the weight and context vectors. RecSOM has two Hebbian update rules, one for the weights $\boldsymbol{w}_i$ and one for the context $\boldsymbol{c}_i$, with their own learning rates.

SOM for Structured Data (SOMSD) presented in [1] is an extension of SOM designed to process patterns expressed as directed acyclic graphs (trees and sequences are special cases). Each node v of a graph pattern has a label $\boldsymbol{u}_v \in \mathbb{R}^d$. Neurons i are arranged again on a rectangular lattice structure. The position of each neuron i on the lattice is described by a coordinate vector $\boldsymbol{c}_i$. For the processing of a dataset of graphs of maximum out-degree k, each neuron i besides its weight vector $\boldsymbol{w}_i \in \mathbb{R}^d$ is supplied with k additional coordinate weight vectors $\boldsymbol{c}_j^{(i)} \in \mathbb{R}^2$ with $j = 1, \ldots, k$. Similarly to RecSOM, these additional weight vectors try to capture the expected context of the node v currently processed. This context itself is formed by first calculating the winning neurons $I(j)$ of the k children nodes $j = 1, 2, .., k$ of node v. The coordinate vectors $\boldsymbol{c}_{I(j)}$ of the winning neurons are then collected to form the context $\left[\boldsymbol{c}_{I(1)}, \boldsymbol{c}_{I(2)}, \ldots, \boldsymbol{c}_{I(k)}\right]$. The complete augmented input to SOMSD is formed by the label of the current node and the context: $\left[\boldsymbol{u}_v, \boldsymbol{c}_{I(1)}, \boldsymbol{c}_{I(2)}, \ldots, \boldsymbol{c}_{I(k)}\right]$. The activation of unit i is calculated as:

$$y_i(v) = \mu_1\|\boldsymbol{u}_v - \boldsymbol{w}_i\|^2 + \mu_2(\|c_{I(1)} - \boldsymbol{c}_1^{(i)}\|^2 + \cdots + \|c_{I(k)} - \boldsymbol{c}_k^{(i)}\|^2), \quad (2)$$

where μ_1 and μ_2 are positive constants that control the contribution of the input label $\boldsymbol{u}_v$ and the context vectors $\boldsymbol{c}_j^{(i)}$.

Since, processing a node requires knowing the winning neurons of its children nodes, processing of a graph must proceed in a bottom-up fashion: before a node v can be processed all of its children must be already processed. This is illustrated in Fig.2. Therefore, processing starts from the leaf nodes (nodes without children). When processing a leaf node, its context vectors are set to some default value representing the empty tree $\boldsymbol{nil}$. The same applies to nodes with less than k children where the coordinate vectors $\boldsymbol{c}_j$ of the missing children are substituted by $\boldsymbol{nil}$. The coordinate vector of $\boldsymbol{nil}$ is typically chosen to be $(-1, \ldots, -1)$, so that it resides outside the lattice. SOMSD is trained in a Hebbian fashion and as is usual in SOM-type formulations, the learning rate and the neighbourhood radius decay gradually. The winner is the neuron with the closest weight and context vectors to the augmented input:

$$I(v) = argmin_i y_i(v). \quad (3)$$

If μ_1 is set to 1 and μ_2 is set to 0, SOMSD reduces to the standard SOM algorithm. Also note that for $k = 1$, SOMSD processes sequences.

III. OVERVIEW OF THE GENERATIVE TOPOGRAPHIC MAP

Let us consider a dataset of static d-dimensional vectors $\mathcal{T} = \left\{\boldsymbol{t}^{(1)}, \ldots, \boldsymbol{t}^{(N)}\right\}$ that are independently generated from some underlying distribution in $\mathbb{R}^d$. We model the density with a mixture of C spherical Gaussians:

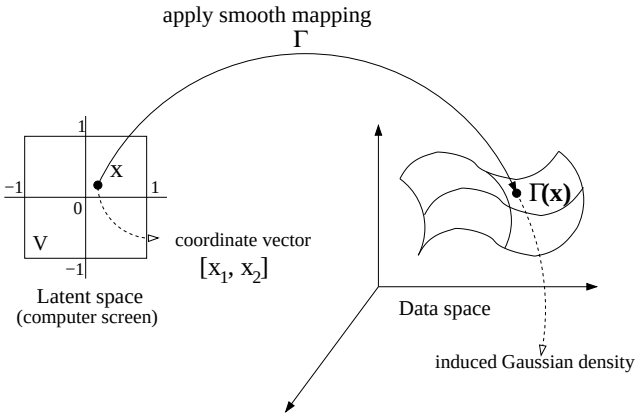


Fig. 3. GTM mapping from latent points to the means of Gaussian components. Adapted from [4].

$$p(\mathcal{T}) = \prod_{n=1}^{N} \sum_{m=1}^{M} P(c) p(\boldsymbol{t}^{(n)}|c) = \prod_{n=1}^{N} \sum_{c=1}^{C} P(c) \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_c, \sigma_c), \tag{4}$$

where $P(c)$ are the mixing coefficients, $\boldsymbol{\mu}_c$ the means of the Gaussians and σ_c the standard deviations. For brevity of presentation we shall assume that $P(c) = \frac{1}{C}$ and that the variance $\sigma_c^2 = \sigma^2$ is fixed. This model is an unconstrained model in the sense that its parameters, the means, do not adhere to any constraints and can move freely. This model which is useful for density modelling can be further extended to capture topographic organisation of the data points.

Topographic organisation can be imposed by requiring that the means of the mixture model reside on an image (under a smooth map Γ) of a continuous Euclidean latent space $\mathcal{V}$ of dimension $q < d$ ($q = 2$ for the purposes of visualisation). The non-linear mapping $\Gamma : \mathcal{V} \rightarrow \mathbb{R}^d$ takes the form [4]:

$$\Gamma(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x}), \tag{5}$$

which can be viewed as a RBF network with basis functions $\boldsymbol{\phi}(.)$ and weight matrix $\boldsymbol{W}$. Function Γ maps each latent point $\boldsymbol{x} \in \mathcal{V}$ to a mean $\boldsymbol{\mu}$ of the model in a non-linear manner. Since Γ is smooth, the projected points will retain their local neighbourhood in the higher dimensional data space. For tractability reasons we discretize the space $\mathcal{V}$ by a rectangular grid of points $\boldsymbol{x}_c$, $c = 1, \ldots, C$. The prior distribution on the latent space is then:

$$p(\boldsymbol{x}) = \frac{1}{C} \sum_{c=1}^{C} \delta(\boldsymbol{x}_c - \boldsymbol{x}), \tag{6}$$

where $\delta(\boldsymbol{x})$ denotes the Dirac delta function, which is $\delta(\boldsymbol{x}) = 0, \boldsymbol{x} \neq 0$. Mapping Γ from the latent points to the means of the Gaussian components is illustrated in Fig.3. We can now formulate GTM as a mixture of Gaussians constrained on Γ-images of latent points $\boldsymbol{x} \in \mathcal{V}$. The likelihood function reads:

$$\begin{aligned}
\mathcal{L} = p(\mathcal{T}) &= \prod_{n=1}^{N} \int_{\boldsymbol{x}} p(\boldsymbol{x}) p(\boldsymbol{t}^{(n)}|\boldsymbol{x}) d\boldsymbol{x} \\
&= \prod_{n=1}^{N} \sum_{c=1}^{C} p(\boldsymbol{x}_c) p(\boldsymbol{t}^{(n)}|\boldsymbol{x}_c) \\
&= \prod_{n=1}^{N} \sum_{c=1}^{C} p(\boldsymbol{x}_c) \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{W}\Phi(\boldsymbol{x}_c), \sigma) \\
&= \prod_{n=1}^{N} \sum_{c=1}^{C} p(\boldsymbol{x}_c) \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_c, \sigma) \\
&= \frac{1}{C} \prod_{n=1}^{N} \sum_{c=1}^{C} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_c, \sigma) \\
&\propto \prod_{n=1}^{N} \sum_{c=1}^{C} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_c, \sigma).
\end{aligned} \tag{7}$$

Training the GTM proceeds by maximising the likelihood given the data, $\mathcal{L}$, via the Expectation-Maximisation (E-M) algorithm [4]. Having trained the model, it can be used for visualising the data. To that end we note that the probability of a data point $\boldsymbol{t}^{(n)}$ given a latent point $\boldsymbol{x}$ is:

$$p(\boldsymbol{t}^{(n)}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_{\boldsymbol{x}}, \sigma). \tag{8}$$

We can reverse this probability, using Bayes' theorem, to obtain the posterior of the latent point $\boldsymbol{x}$ given $\boldsymbol{t}^{(n)}$:

$$\begin{aligned}
p(\boldsymbol{x}|\boldsymbol{t}^{(n)}) &= \frac{p(\boldsymbol{t}^{(n)}|\boldsymbol{x}) P(\boldsymbol{x})}{p(\boldsymbol{t}^{(n)})} = \frac{p(\boldsymbol{t}^{(n)}|\boldsymbol{x}) P(\boldsymbol{x})}{\sum_{c'=1}^{C} p(\boldsymbol{t}^{(n)}|\boldsymbol{x}_{c'}) P(\boldsymbol{x}_{c'})} \\
&= \frac{p(\boldsymbol{t}^{(n)}|\boldsymbol{x})}{\sum_{c'=1}^{C} p(\boldsymbol{t}^{(n)}|\boldsymbol{x}_{c'})} = \frac{\mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_{\boldsymbol{x}}, \sigma)}{\sum_{c'=1}^{C} \mathcal{N}(\boldsymbol{t}^{(n)}; \boldsymbol{\mu}_{\boldsymbol{x}_{c'}}, \sigma)}.
\end{aligned} \tag{9}$$

We can then represent each data point $\boldsymbol{t}^{(n)}$ with a point $\mathfrak{p}^{(n)}$ in the latent space given by the expectation of the posterior distribution over all latent points $\boldsymbol{x}$:

$$\mathfrak{p}^{(n)} = \sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{t}^{(n)}) \boldsymbol{x}_c. \tag{10}$$

IV. An overview of Hidden Markov Tree Models

We extend GTM to the visualisation of tree structures by requiring that the points in the latent space generate the parameters of hidden Markov tree models (HMTMs).

A tree $\boldsymbol{y}$ is an acyclic directed graph and as such it consists of a set $\mathcal{U}_{\boldsymbol{y}} = \{1, 2, ..., U_{\boldsymbol{y}}\}$ of nodes $u \in \mathcal{U}_{\boldsymbol{y}}$, a set of directed edges between the nodes (each edge goes from a parent node to a child node) and a set of labels $\boldsymbol{o}_u \in \mathbb{R}^d$ on nodes u. Each node u has a single parent $\rho(u)$ (apart from the node number one, the root node) and each node has a set of children $ch(u)$ (apart from the leaf nodes). Furthermore we designate subtrees. A subtree rooted at node u of a tree $\boldsymbol{y}$ is referred to by $\boldsymbol{y}_u$. Hence, the entire tree $\boldsymbol{y}$ is equivalent to the subtree $\boldsymbol{y}_1$ rooted at its root. Moreover, $\boldsymbol{y}_{1/u}$ denotes the entire tree except for the subtree rooted at node u. This notation is illustrated in Fig.4(a).

We also introduce a model for labels of the trees that captures the structure of the nodes. We associate with each node u a discrete random variable Q_u which can be in one of K (unobservable) states. The variable Q_u stochastically determines the label for node u. Each state $k = 1, 2, ..., K$ is associated with a parametrised emission distribution $f(.; k)$ that produces a label. So given a tree structure $\boldsymbol{y}$, the model can label each of the nodes depending on what state $Q_u \in \{1, 2, ..., K\}$ each node u is in. What states will be entered and ultimately what labels will be produced depends on the structure of the model. By structure we mean a joint probability distribution over state variables that characterises the relationship between states Q_u of all nodes $u \in \mathcal{U}_{\boldsymbol{y}}$. The simplest structure is the one where all the states are independent from each other. In this case a node can enter any state regardless of the state of any other node and the joint distribution simplifies to a product of probabilities. Such a simple structure, however, does not capture the structural information in the trees induced by the parent-child relationship of the nodes. A more appropriate structure is to make each node u dependent on its parent $\rho(u)$. Thus, the state Q_u is conditioned on the state $Q_{\rho(u)}$ of its parent. Such a structure implements a first-order Markov property. Moreover, we assume that when the model labels the tree it does not reveal the states Q_u entered. Thus, the underlying process that generates a tree $\boldsymbol{y}$ is hidden from us, and only the labels $\boldsymbol{o}_u$, $u \in \mathcal{U}_{\boldsymbol{y}}$ are observed. This is illustrated in Fig.4(b).

The resulting model, called the hidden Markov tree model (HMTM) [15], is an extension of the hidden Markov model (HMM) [16] operating on sequences. HMTM models tree structure $\boldsymbol{y}$ by expressing a joint probability density for the set of hidden state variables $Q_1, \ldots, Q_{U_{\boldsymbol{y}}}$, each defined on the support $\{1, 2, \ldots, K\}$, and the set of labels $\boldsymbol{o}_1, \ldots, \boldsymbol{o}_{U_{\boldsymbol{y}}}$ in $\mathbb{R}^d$. The model is called *hidden* because the states cannot be directly observed, while *Markov* refers to the fact that the current state of a node depends only on that of its immediate predecessor (parent).

An HMTM, in the same fashion as an HMM, is defined by three sets of parameters:

- initial probability distribution $\boldsymbol{\pi} = \{p(Q_1 = k)\}_{k=1,\ldots,K}$ – each element expressing the probability of the root node being in state $k \in \{1, 2, \ldots, K\}$.
- transition probability distribution $\boldsymbol{T} = \{p(Q_u = k|Q_{\rho(u)} = l)\}_{l,k=1,\ldots,K}$ – each element expressing the probability of transiting from parent $\rho(u)$ in state l to the child node u in state k. This probability is assumed to be position-invariant.
- the emission parameters that parametrise Gaussian distributions, $f(.; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, one for each state k. Here, $\boldsymbol{\mu}_k \in \mathbb{R}^d$ and $\boldsymbol{\Sigma}_k$ are the mean and covariance matrix, respectively, of the Gaussian associated with emission process in state k.

The Markovian dependencies of hidden states are realised by the following conditions:

- Given the parent state, the child state is conditionally independent of all other states: $p(Q_u = q_u|\{Q_v = q_v\}_{v \neq u}) = p(Q_u = q_u|Q_{\rho(u)} = q_{\rho(u)})$.
- Given the (hidden) state of a node, the corresponding label is conditionally independent of all other variables in the tree, $p(\boldsymbol{O}_u = \boldsymbol{o}_u|\{\boldsymbol{O}_v = \boldsymbol{o}_v\}_{v \neq u}, \{Q_v = q_v\}_{v \neq u}) = p(\boldsymbol{O}_u = \boldsymbol{o}_u|Q_u = q_u)$.

Thus, the HMTM distribution can be factorised as follows:

$$
\begin{aligned}
&p(\boldsymbol{y}, Q_1 = q_1, \ldots, Q_{U_{\boldsymbol{y}}} = q_{U_{\boldsymbol{y}}}) = \\
&p(Q_1 = q_1)\Bigg(\prod_{u \in \mathcal{U}_{\boldsymbol{y}}, u \neq 1} p(Q_u = q_u|Q_{\rho(u)} = q_{\rho(u)})\Bigg) \\
&\times\Bigg(\prod_{u \in \mathcal{U}_{\boldsymbol{y}}} p(\boldsymbol{O}_u = \boldsymbol{o}_u|Q_u = q_u)\Bigg). \qquad (11)
\end{aligned}
$$

Henceforth, for brevity we shall drop stating both random variables and their instantiations, keeping only the latter.

Similarly to the forward-backward algorithm for HMM [16], the likelihood of an HMTM can be efficiently computed by the upward-downward algorithm. The motivation of this algorithm stems from the observation that a direct calculation of likelihood without knowledge of the hidden states requires an exponential number of steps. The upward-downward algorithm defines the following quantities:

$$
\begin{aligned}
&\alpha_k(u) = p(q_u = k, \boldsymbol{y}_{1/u}) && \text{upward probability,} \\
&\beta_k(u) = p(\boldsymbol{y}_u|q_u = k) && \text{downward probability.}
\end{aligned}
$$

The model likelihood, given a tree $\boldsymbol{y}$, can then be calculated using any node $u \in \mathcal{U}_{\boldsymbol{y}}$ as:

$$
p(\boldsymbol{y}) = \sum_{k=1}^{K} p(\boldsymbol{y}, q_u = k) = \sum_{k=1}^{K} \beta_k(u)\alpha_k(u). \qquad (12)
$$

V. HMTMs as noise models for GTM

This Section presents an extension of GTM from vectorial to tree structured data in the spirit of [17], where GTM is extended to visualise sequential data. Analogously to GTM, each latent point $\boldsymbol{x} \in \mathcal{V}$ is mapped to an HMTM via a smooth non-linear mapping Γ. Since the neighbourhood of

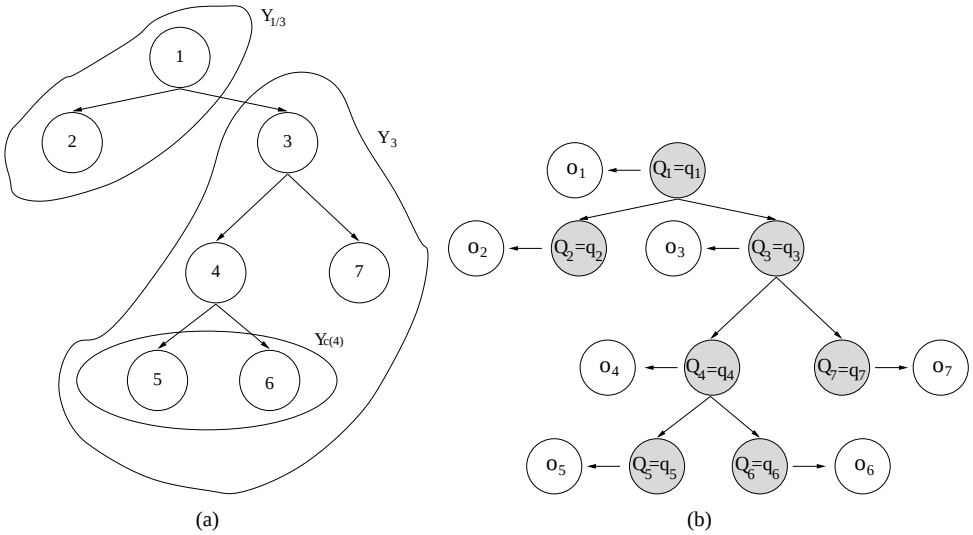


Fig. 4. Notation in tree structures (a), Example of an HMTM where the hidden states Q (states in grey) emit labels $\boldsymbol{o}$ (b).

Γ-images of $\boldsymbol{x}$ is preserved, the resulting HMTMs will be topographically organised. Here the observations are no longer fixed-length vectors $\boldsymbol{t}$, but trees $\boldsymbol{y}$, as described in Section IV. For each latent point $\boldsymbol{x}$ we calculate the likelihood $p(\boldsymbol{y}|\boldsymbol{x})$ (see (12)). Each data item $\boldsymbol{y}$ is subsequently mapped to the location of the map where the $p(\boldsymbol{y}|\boldsymbol{x})$ is expected to be high.

The starting point of our formulation is the form of a standard mixture model. However, this time the components are not spherical Gaussians as in GTM, but HMTMs. Assuming that the given trees $\mathcal{T} = \{\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, ..., \boldsymbol{y}^{(N)}\}$ are independently generated, the likelihood is expressed as:

$$\mathcal{L} = \prod_{n=1}^{N} p(\boldsymbol{y}^{(n)}) = \prod_{n=1}^{N} \sum_{c=1}^{C} p(\boldsymbol{y}^{(n)}|\boldsymbol{x}_c) p(\boldsymbol{x}_c), \qquad (13)$$

where the mixing coefficients can be ignored as $p(\boldsymbol{x}) = \frac{1}{C}$. Denote the number of nodes $U_{\boldsymbol{y}^{(n)}}$ of the n-th tree $\boldsymbol{y}^{(n)}$ by U_n and consider a regular grid $\{\boldsymbol{x}_c\}_{c=1}^{C}$ in the latent space $\mathcal{V}$. The noise models $p(\boldsymbol{y}^{(n)}|\boldsymbol{x}_c)$ are expanded using (11):

$$\mathcal{L} \propto \prod_{n=1}^{N} \sum_{c=1}^{C} \sum_{\boldsymbol{q} \in \{1,2,...,K\}^{U_n}} p(q_1|\boldsymbol{x}_c) \prod_{u=2}^{U_n} p(q_u|q_{\rho(u)}, \boldsymbol{x}_c) \times \prod_{u=1}^{U_n} p(\boldsymbol{o}_u^{(n)}|q_u, \boldsymbol{x}_c). \qquad (14)$$

In order to have the HMTM components topologically organised — e.g. on a two-dimensional equidistant grid — we constrain the mixture of HMTMs,

$$p(\boldsymbol{y}) = \frac{1}{C} \sum_{c=1}^{C} p(\boldsymbol{y}|\boldsymbol{x}_c), \qquad (15)$$

by requiring that the HMTM parameters be generated through a parameterised *smooth* nonlinear mapping from the latent space into the HMTM parameter space:

$$\begin{aligned} \boldsymbol{\pi}_c &= \{p(q_1 = k|\boldsymbol{x}_c)\}_{k=1,...,K} \\ &= \{g_k(\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c))\}_{k=1,...,K}, \end{aligned} \qquad (16)$$

$$\begin{aligned} \boldsymbol{T}_c &= \{p(q_u = k|q_{\rho(u)} = l, \boldsymbol{x}_c)\}_{k,l=1,...,K} \\ &= \{g_k(\boldsymbol{A}^{(\boldsymbol{T}_l)}\boldsymbol{\phi}(\boldsymbol{x}_c))\}_{k,l=1,...,K}, \end{aligned} \qquad (17)$$

$$\begin{aligned} \boldsymbol{B}_c &= \{\boldsymbol{\mu}_k^{(c)}\}_{k=1,...,K} \\ &= \{\boldsymbol{A}^{(\boldsymbol{B}_k)}\boldsymbol{\phi}(\boldsymbol{x}_c)\}_{k=1,...,K}, \end{aligned} \qquad (18)$$

where

- the function $g(\cdot)$ is the softmax function, which is the canonical inverse link function of multinomial distribution and $g_k(\cdot)$ denotes the k-th component returned by the softmax, i.e.
$$g_k\left((a_1, a_2, ..., a_q)^T\right) = \frac{e^{a_k}}{\sum_{i=1}^{q} e^{a_i}}, \quad k = 1, 2, ..., q.$$
Here the softmax function "squashes" the values of $\boldsymbol{A}^{(\boldsymbol{\pi})}\boldsymbol{\phi}(\boldsymbol{x}_c)$ and $\boldsymbol{A}^{(\boldsymbol{T}_l)}\boldsymbol{\phi}(\boldsymbol{x}_c)$, which are unbounded, to values in the range $[0, 1]$. This is necessary as the elements in $\boldsymbol{\pi}_c$ and $\boldsymbol{T}_c$ are probabilities.
- $\boldsymbol{x}_c \in \mathbb{R}^2$ is the c-th grid pointin the latent space $\mathcal{V}$,
- $\boldsymbol{\phi}(\cdot) = (\phi_1(\cdot), ..., \phi_M(\cdot))^T, \phi_m(\cdot) : \mathbb{R}^2 \to \mathbb{R}$ a vector function consisting of M nonlinear smooth basis functions (typically RBFs),
- the matrices $\boldsymbol{A}^{(\boldsymbol{\pi})} \in \mathbb{R}^{K \times M}$, $\boldsymbol{A}^{(\boldsymbol{T}_l)} \in \mathbb{R}^{K \times M}$ and $\boldsymbol{A}^{(\boldsymbol{B}_k)} \in \mathbb{R}^{d \times M}$ are the free parameters of the model.

Note that we decided not to directly model the covariance of the emission distributions. We will elaborate on this point later.

We require the likelihood to be maximised. This can be achieved by adopting an E-M formulation of the problem by writing the (complete data) likelihood in terms of hidden indicator variables z:

$$z_c^n = \begin{cases} 1, \text{if tree } \boldsymbol{y}^{(n)} \text{ was generated by model } c; \\ 0, \text{otherwise.} \end{cases}$$

$$z_{u,k,c}^n = \begin{cases} 1, \text{if tree } \boldsymbol{y}^{(n)} \text{ was generated by model } c \\ \quad \text{and node } u \text{ was in state } k; \\ 0, \text{otherwise.} \end{cases}$$

$$z_{u,k,l,c}^n = \begin{cases} 1, \text{if tree } \boldsymbol{y}^{(n)} \text{ was generated by model } c, \\ \quad \text{node } u \text{ was in state } k \text{ and its parent} \\ \quad \text{node is in state } l; \\ 0, \text{otherwise.} \end{cases}$$

The complete data likelihood and its logarithm read:

$$\begin{aligned} \bar{\mathcal{L}} = & \prod_{n=1}^{N} \prod_{c=1}^{C} \Big[\prod_{k=1}^{K} p(q_1 = k|\boldsymbol{x}_c)^{z_c^n z_{1,k,c}^n} \\ & \times \prod_{u=2}^{U_n} \prod_{l=1}^{K} \prod_{k=1}^{K} p(q_u = k|q_{\rho(u)} = l, \boldsymbol{x}_c)^{z_c^n z_{u,k,l,c}^n} \\ & \times \prod_{u=1}^{U_n} \prod_{k=1}^{K} p(\boldsymbol{o}_u^{(n)}|q_u = k, \boldsymbol{x}_c)^{z_c^n z_{u,k,c}^n} \Big], \\ \log \bar{\mathcal{L}} = & \sum_{n=1}^{N} \sum_{c=1}^{C} z_c^n \Big[z_{1,k,c}^n \sum_{k=1}^{K} \log p(q_1 = k|\boldsymbol{x}_c) \\ & + \sum_{u=2}^{U_n} \sum_{l=1}^{K} \sum_{k=1}^{K} z_{u,k,l,c}^n \log p(q_u = k|q_{\rho(u)} = l, \boldsymbol{x}_c) \\ & + \sum_{u=1}^{U_n} \sum_{k=1}^{K} z_{u,k,c}^n \log p(\boldsymbol{o}_u^{(n)}|q_u = k, \boldsymbol{x}_c) \Big]. \end{aligned} \tag{19}$$

Following the E-M formulation, we maximise the expected complete data log-likelihood with respect to the posterior distribution of the hidden (indicator) variables, given the data and current parameter values. In the E-step, these posteriors (and their expectations) over the hidden variables are estimated:

$$E[z_c^n] = p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}), \tag{20}$$

$$E[z_{u,k,c}^n] = p(q_u = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c), \tag{21}$$

$$E[z_{u,k,l,c}^n] = p(q_u = k, q_{\rho(u)} = l|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c). \tag{22}$$

The above probabilities are obtained by the upward-downward algorithm as calculated in [18]. We can now express the expected complete-data log-likelihood of the model as:

$$\begin{aligned} E[\log \bar{\mathcal{L}}] = & \sum_{n=1}^{N} \sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}) \\ & \Big[\sum_{k=1}^{K} p(q_1 = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c) \log p(q_1 = k|\boldsymbol{x}_c) \\ & + \sum_{u=2}^{U_n} \sum_{l=1}^{K} \sum_{k=1}^{K} p(q_u = k, q_{\rho(u)} = l|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c) \\ & \times \log p(q_u = k|q_{\rho(u)} = l, \boldsymbol{x}_c) \\ & + \sum_{u=1}^{U_n} \sum_{k=1}^{K} p(q_u = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c) \\ & \times \log p(\boldsymbol{o}_u^{(n)}|q_u = k, \boldsymbol{x}_c) \Big]. \end{aligned} \tag{23}$$

In the M-step, the derivatives of the expected log-likelihood are calculated with respect to the parameters of the model:

$$\frac{\partial E[\log \bar{\mathcal{L}}]}{\partial A^{(\boldsymbol{\pi})}}, \frac{\partial E[\log \bar{\mathcal{L}}]}{\partial A^{(\boldsymbol{T}_l)}}, \frac{\partial E[\log \bar{\mathcal{L}}]}{\partial A^{(\boldsymbol{B}_k)}}.$$

This results in the following update equations:

Element $\alpha_{k,i}$ in matrix $\boldsymbol{A}^{(\boldsymbol{\pi})}$:

$$\begin{aligned} \frac{\partial E[\log \bar{\mathcal{L}}]}{\partial \alpha_{k,i}} = & \sum_{n=1}^{N} \sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}) \phi_i(\boldsymbol{x}_c) \\ & \times \Big(p(q_1 = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c) - p(q_1 = k|\boldsymbol{x}_c) \Big), \end{aligned} \tag{24}$$

Element $\alpha_{k,i}^l$ in matrix $\boldsymbol{A}^{(\boldsymbol{T}_l)}$:

$$\begin{aligned} \frac{\partial E[\log \bar{\mathcal{L}}]}{\partial \alpha_{k,i}^l} = & \sum_{n=1}^{N} \sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}) \phi_i(\boldsymbol{x}_c) \\ & \times \sum_{u=2}^{U_n} \Big(p(q_u = k, q_{\rho(u)} = l|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c) \\ & - p(q_u = k|q_{\rho(u)} = l, \boldsymbol{x}_c) p(q_{\rho(u)} = l|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c) \Big), \end{aligned} \tag{25}$$

Element $\alpha_{k,i}^l$ in matrix $\boldsymbol{A}^{(\boldsymbol{B}_l)}$:

$$\begin{aligned} \frac{\partial E[\log \bar{\mathcal{L}}]}{\partial \alpha_{k,i}^l} = & \sum_{n=1}^{N} \sum_{c=1}^{C} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}) \phi_i(\boldsymbol{x}_c) \\ & \times \sum_{u=1}^{U_n} \Big(p(q_u = l|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c) \boldsymbol{e}_k \boldsymbol{\Sigma}_l^{-1} (\boldsymbol{o}_u^{(n)} - \boldsymbol{\mu}_l^{(c)}) \Big), \end{aligned} \tag{26}$$

where $\boldsymbol{e}_k$ is defined as the row unit-vector which has all elements equal to zero apart from entry k equal to 1, $\boldsymbol{\Sigma}_k$, $k = 1, 2, ..., K$ are the covariance matrices of the state-conditional Gaussian emissions.

Regarding the covariance of the emission distribution, we noticed that higher quality models were obtained when instead of direct modelling of the covariance through the map Γ, the covariance was calculated, in the spirit of [4], at the end of each M-step using standard update equations:

TABLE II
PARAMETERS OF HMTMS FOR CREATING THE TOY DATASET. VARIANCE WAS FIXED TO $\sigma^2 = 1$.

Class	Initial prob	Transition prob	Means of emissions
HMTM 1	0.7 0.3	0.9 0.1	−1.0 4.0
		0.1 0.9	1.0 2.0
HMTM 2	0.7 0.3	0.9 0.1	−2.0 6.0
		0.1 0.9	3.0 0.0
HMTM 3	0.7 0.3	0.1 0.9	−1.0 4.0
		0.9 0.1	1.0 2.0
HMTM 4	0.7 0.3	0.1 0.9	−2.0 6.0
		0.9 0.1	3.0 0.0

$$\Sigma_{k,ij}^{(c)} = \sum_{n=1}^{N} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}) \times \sum_{u=1}^{U_n} p(q_u = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c)(o_{u,i}^{(n)} - \mu_{k,i}^{(c)})(o_{u,j}^{(n)} - \mu_{k,j}^{(c)}) \times \frac{1}{\sum_{n=1}^{N} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}) \sum_{u=1}^{U_n} p(q_u = k|\boldsymbol{y}^{(n)}, \boldsymbol{x}_c)} \quad (27)$$

where $i, j = 1, 2, ..., d$ index the elements of the mean and label vectors $\boldsymbol{\mu}$ and $\boldsymbol{o}$, respectively, as well as the elements of the covariance matrix $\boldsymbol{\Sigma}$.

After the training, to smooth the covariance structure of local HMTMs addressed by arbitrary latent points, we recalculated the covariance matrices using the following scheme: Covariance matrix $\boldsymbol{\Sigma}_k(\boldsymbol{x})$ of the HMTM addressed by $\boldsymbol{x} \in \mathcal{V}$ is expressed as a convex combination of the corresponding covariance matrices[2] $\boldsymbol{\Sigma}_k^{(c)}$ of HMTMs addressed by latent centres $\boldsymbol{x}_c$, $c = 1, 2, ..., C$:

$$\boldsymbol{\Sigma}_k(\boldsymbol{x}) = \sum_{c=1}^{C} \nu_c(\boldsymbol{x}) \boldsymbol{\Sigma}_k^{(c)}, \quad (28)$$

where

$$\nu_c = \frac{\exp(-\beta \|\boldsymbol{x} - \boldsymbol{x}_c\|)}{\sum_{c'=1}^{C} \exp(-\beta \|\boldsymbol{x} - \boldsymbol{x}_{c'}\|)}, \quad (29)$$

and $\|\cdot\|$ denotes the Euclidean norm on $\mathcal{V}$. The parameter $\beta > 0$ quantifies to what degree local neighbourhoods of $\boldsymbol{x}$ are considered.

Here we have set $\beta = 10$, but we have found that the visualisation plots were similar for a wide range of β values. In practice, compared to the obvious choice of directly parameterising the covariance matrices through a smooth mapping from the latent space, we found that this scheme leads to superior models (viewed as density estimators and evaluated on a hold out set) and hence better visualisation plots.

VI. EXPERIMENTATION

A. Datasets

We have used three datasets in our experiments. The first dataset is an artificial toy dataset produced by sampling from 4 HMTMs with 2 hidden states with two-dimensional Gaussian emissions of fixed spherical variance, each corresponding to one artificial class. Each of the 4 classes has 80 samples. All patterns have the topology of a binary tree with 15 nodes. The parameters of the models were set is such a way as to ensure that it would be impossible to distinguish the classes from the observations alone, without taking into account the underlying tree structure. A plot of two-dimensional observations of all the nodes for all trees is presented in Fig.5(a). The parameters of the HMTMs are summarised in Table II.

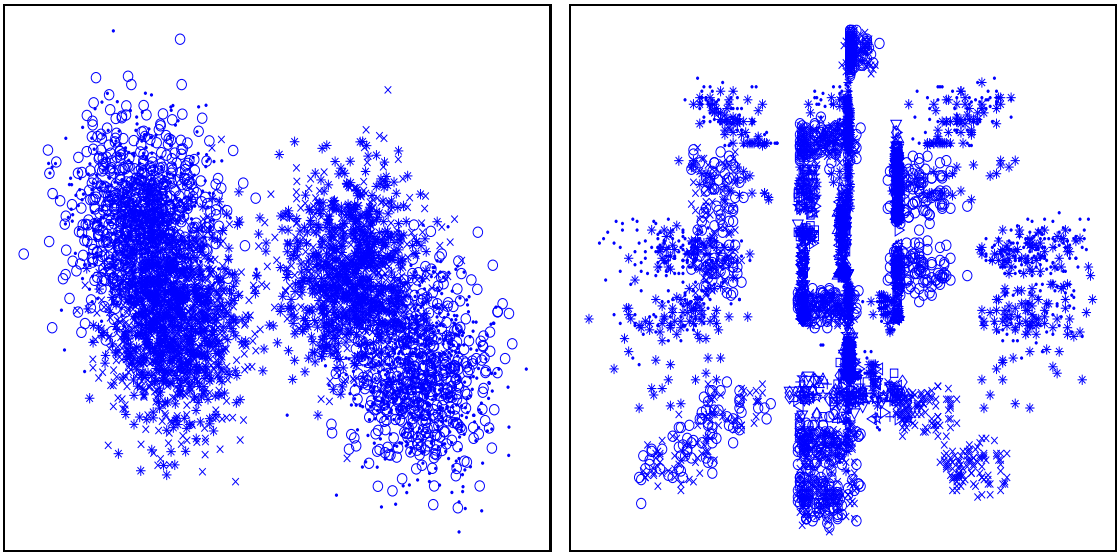
[2]note that a convex combination of symmetric positive definite matrices is again a symmetric positive definite matrix.

TABLE III
CLASSES IN TPB DATASET.

Class	Symbol	Description
A	◯	Policemen with the lowered left arm
B	x	Policeman with the raised left arm
C	*	Ships with two masts
D	•	Ships with three masts
E	△	Houses with one upper right window
F	▽	Houses with upper left and lower left window
G	◁	Houses with two upper windows
H	▷	Houses with lower left and upper right window
I	⋆	Houses with three windows
J	□	Houses with one lower left window
K	+	Houses with no windows
L	♢	Houses with one upper left window

The second dataset consists of benchmark images produced by the *Traffic Policeman Benchmark* (TPB) software [19]. The same software was used to produce a dataset to demonstrate the functionality of SOM for Structured Data (SOMSD) in [1]. This software provides an artificial domain for evaluating learning algorithms that process structured patterns. It produces images that resemble traffic policemen, houses and ships of different shape, size and colour that are products of a rule based grammar. Three sample images of each type are illustrated in the in Fig.6(a), 6(b) and 6(c). Connected components in each image have a parent-child relationship, the object located lower and closer to the left edge being the parent (i.e. the images must be interpreted bottom-up, left to right). In Fig.6(d), 6(e) and 6(f) tree representations of the sample images corresponding to Fig.6(a), 6(b) and 6(c) are displayed. TPB produces general acyclic graph structures, but we restricted it to generate only images expressed as trees. Each node in each tree is labelled with a two-dimensional vector. This two-dimensional vector is a pair of coordinates for the centre of gravity of the component that node stands for. The dataset defines 12 classes, each has 50 samples, that are presented in Table III. Fig.5(b) is a plot of the labels of trees in the dataset. This illustrates what the observed data looks like if the tree structure is ignored.

The third dataset consists of images interpreted as *quadtrees*. A quadtree is a data structure used amongst other things for storage of images [20]. It is a 4-regular tree $\boldsymbol{y}$, i.e. each parent node u has 4 children $v_r \in ch(u), r = 1, \ldots, 4$. A quadtree $\boldsymbol{y}$ stores an image in a recursive manner; the root note u_1 represents the entire image. At the first level of recursion,



(a) (b)

Fig. 5. Labels of toy (a) and TPB (b) dataset. Each marker style indicates class membership of the tree to which each label belongs.

the image is partitioned into four equal square quadrants. At the first level of quadtree $\boldsymbol{y}$, each node $v \in ch(u_1)$ represents a quadrant, and is labelled by a scalar that expresses the mean colour intensity of the quadrant. At the next level of recursion, each quadrant is partitioned further into four quadrants and their mean colour intensities are stored as labels in the nodes at the second level of quadtree $\boldsymbol{y}$. Partitioning continues in this fashion either until a quadrant becomes a single pixel, or when a certain criterion is met. Such a criterion can be a function of the relative change in mean colour intensity between a node u and its parent $\rho(u)$. We note that quadtrees can represent only images of a dimension that is a power of 2 since images are progressively divided into smaller square regions. Other images must be padded with extra pixels or resized in order to become of appropriate dimension.

The images used here, are taken from the Amsterdam Library of Object Images (ALOI) database [21]. We selected 72 images of a single object, a rubber duck, photographed from different viewing angles. The dataset was divided into a training and validation set of 48 and 24 images respectively. The images were created by successively rotating the object by an angle of 5° degrees and photographing it from each viewing angle. The images are colour images of dimension 192×144 (pixels). We converted the images into grayscale and resized them into square images of dimensions 64×64. The number of grayscale levels was then further reduced to 4 levels, which allows enough detail to be discerned relative to the original images. The values of the 4 quantisation levels were determined by first collecting the grayscale intensities of all pixels from all images, and then using the k-means algorithm to select 4 centres in the space of pixel intensities.

All three datasets were normalised in each dimension to zero mean and unit standard deviation.

B. Training

The lattice in the latent space $\mathcal{V} = [-1, 1]^2$ was a 10x10 regular grid (i.e. $C = 100$) and the RBF network consisted of $M = 17$ basis functions; 16 of them were Gaussian radial basis functions of variance $\sigma^2 = 1$ centred on a 4x4 regular grid, and one was a constant function $\phi_{17}(\boldsymbol{x}_c) = 1$ intended to serve as a bias term (analogous to the bias in neural networks).

The state-conditional emission probability distributions were modelled as two-dimensional spherical Gaussians. During training the emission covariance was updated according to (27). Parameters were initialised randomly with uniform distribution in $[-1, 1]$.

We employed scaled conjugate gradient for optimising the cost function (23). The gradient was calculated using (24)–(26).

C. Results and discussion

In Fig.7(a) we see topographic organisation achieved by the GTM-HMTM of the toy dataset for $K = 2$. The covariance of the emission distribution was initially set to $\Sigma_k = 2I$ for both states $k = 1, 2$ where I stands for the identity matrix. We also tried initialising it with $\Sigma_k = 2I, 3I, 5I$ with similar success. Each point on the plot represents an input pattern (tree) and four different markers correspond to the four generative classes used to construct the data set. Training is completely unsupervised and class markers are used only after the training when plotting the projections. A clear topographic organisation of classes has been achieved - there is an evident trend of patterns of the same class to belong to the same cluster.

Fig.7(b) shows the visualisation of the traffic policeman benchmark (TPB) data set produced by GTM-HMTM with $K = 2$. The initial covariance matrix for the emission distribution was set to $\Sigma_k = 2I$ for both states $k = 1, 2$. We also tried initialising the covariance matrix with $\Sigma_k = 1I, 3I$ which yielded similar results and $\Sigma_k = 0.5I$ which failed to

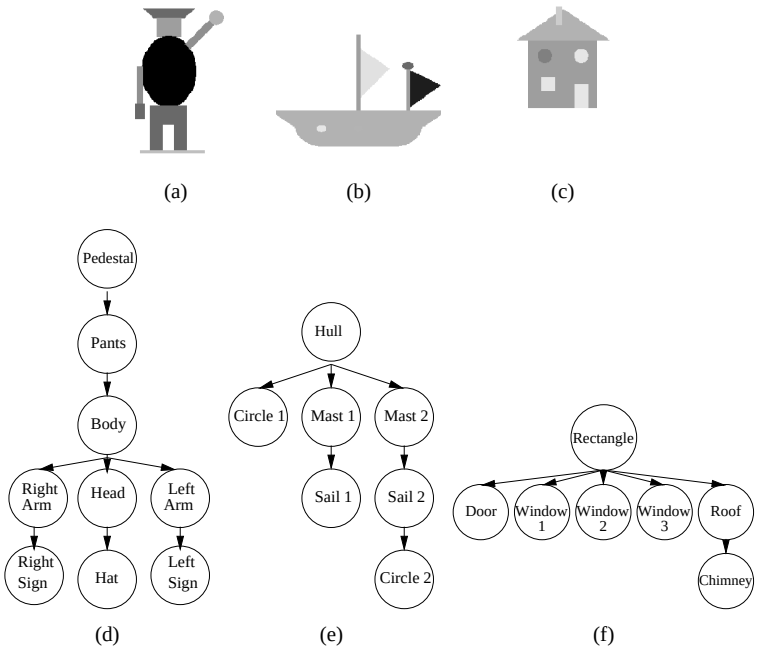


Fig. 6. Sample images from TPB in (a), (b), (c) and their corresponding tree representations in (d), (e), (f).

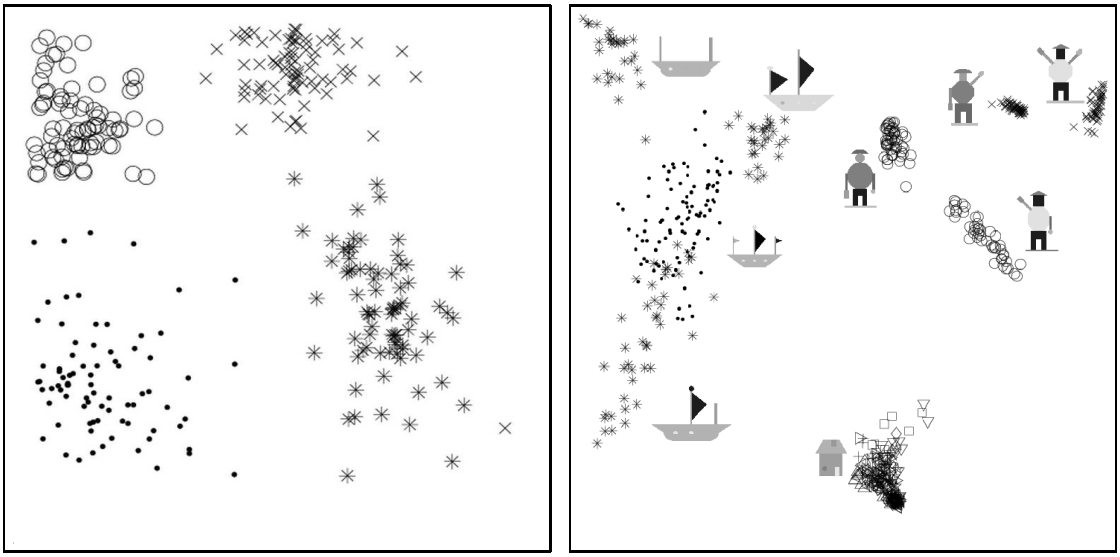


Fig. 7. Visualisation of toy (a) and TPB (b) dataset using GTM-HMTM.

achieve the same level of topographic organisation. Moreover, we attempted training for $K = 3, 4$, but with suboptimal results. One problem that makes training difficult is that as the number of states (and consequently the number of free parameters of the model) increases, it becomes more vital for an E-M trained model to use a good initialisation strategy for the weights. In GTM the initial weights are determined by the linear projection space obtained through principal component analysis [4]. In our case we do not have such a luxury. One way of dealing (at least to certain degree) with the initialisation problem would be to abandon the E-M framework and adopt a more stable parameter fitting strategy (e.g. Bayesian), but this is out of the scope of the present paper.

In Fig.7(b), next to each cluster a representative image is displayed. The model has clearly achieved a level of topographic organisation. It is interesting to note the emerging sub-clusters. Class $\times$ has been split into two sub-clusters, one with policemen with the right arm lowered and one with the right arm raised. The same has happened for class $\bigcirc$ which has been divided into policemen with the right arm lowered and policemen with the arm raised. The sub-clusters of ships are also interesting as not only has class $*$ been divided into three sub-clusters but the sub-clusters that surround class $\bullet$ possibly indicate how the classes are related. Thus, the class

• seems to act as a "link" between the three discovered sub-clusters; class • represents ships with three masts, while the three sub-clusters around class $*$ are composed of ships with either the two masts, with either the left, centre or right mast missing. Nevertheless, the model has not been successful in the visualisation of the classes representing houses. No clusters have been formed as all classes have been merged into one big cluster representing a super-class of all the images of houses. One possible explanation for this inability of discriminating between the classes of houses, is the shallow tree representation of houses; typically they are shorter than ships and traffic-policemen structures.

In Fig.8 the underlying state transitions are visualised. The plot is organised as a grid of $K \times K = 2 \times 2$ state transition matrices $p(q_u = l|q_{\rho(u)} = k)$, each transition matrix corresponding to an underlying local noise model (HMTM). Topographic organisation of local noise models with respect to their transition structure is evident in Fig.8 as state transitions vary smoothly with their "latent space addresses". In Fig.8 we see that state 1 acts as a "trap" state for the entire plot, that is if the model visits state 1, it is extremely unlikely for it to ever visit state 2. Regarding transitions from state 2 we observe a more interesting behaviour. A strong tendency for self-loops in state 2 is observed at the upper-left and bottom-left corner of Fig.8. However, this behaviour gradually changes as we move towards the centre of the map; transitions to state 2 progressively lose their strength benefiting transitions to state 1. Around the centre of the map transitions to state 1 narrowly dominate transitions to state 2. Moving further towards the upper right part of the plot, transitions to state 1 and 2 become almost equally likely. Moving from the centre towards the bottom-right corner, transitions to state 2 regain their power, albeit not to the same strength as in the upper-left and bottom-left corner of the plot.

The respective plot for the initial probabilities is not presented, as a particularly simple structure has emerged as a result of the GTM-HMTM training; the initial probability vector of all models is practically equal to $\boldsymbol{\pi} = [0\ 1]^T$. Thus, effectively all models pick the second state as their starting state, $q_1 = 2$.

In Fig.9 the underlying means of the emissions are visualised. This plot is organised as a grid of subplots. Each subplot presents the space of emissions $\mathbb{R}^d$, where labels $\boldsymbol{o}_u$ reside, in which the means for states $k = 1$ and $k = 2$ marked with circles and crosses, respectively. Evidently, the means of the emissions are topographically organised as well as the state transitions, as the positions of means change gradually as we move in the plot. We note that images in the TPB dataset are interpreted bottom-up (see Fig.6), and that x-coordinates of the labels decrease leftwards while y-coordinates decrease upwards. Thus, components located at the lower part of the TPB images have higher y-coordinates than components located closer to the top of the TPB images. We observe that since state 2 is effectively the starting state for all models and since images are interpreted bottom-up, the mean for state 2 naturally has a greater y-coordinate than the mean for state 1 in the entire plot. We also observe the following three general behaviours in the plot. The means close to the upper-left corner of the plot lie far apart in the x-axis, while being close in the y-axis. This behaviour progressively changes as we move towards the upper-right corner of the plot, where the means have similar x-coordinates but are distant in the y-axis. Moving towards the bottom-centre and bottom-right regions of the plot we notice that the means approach each other. This behaviour reflects the nature of the data points (trees) mapped in particular regions of the latent space. In particular, the ship-classes reserve the left part of the visualisation plot in Fig.7(b). As it can be seen in Fig.6, ships are generally "wide" and "short" structures. Policeman are concentrated at the right upper part of Fig.7(b), and are generally "narrow" and "thin"(see Fig.6), while houses are clustered densely at the bottom-centre of Fig.7(b) and appear to be relatively "compact" (see Fig.6). In order to confirm these observations, we measured the variance for the three classes of ships, policeman and houses. We found that the variance was $1.83, 0.69, 0.14$ in the x-axis and $0.58, 1.53, 0.42$ in the y-axis for the three classes respectively.

Inspecting Fig.8 in conjunction with Fig.9 we make the following observations. In general, the mean for state 1 concentrates more on modelling the labels of lower y-coordinates, while the mean for state 2 seems to concentrate more on the labels of higher y-coordinates. The classes of ships reserve the area that corresponds to the left area in Fig.8 of self-loops for state 2, thus favouring the projection of "short" classes[3]. Furthermore, the upper right area of the latent space, in Fig.7(b), is reserved for the policemen classes, which are "tall" structures. As noted, this corresponding area in the state-transition plot of Fig.8 is where transitions from state 2 to states 1 and 2 become almost equally likely, thus favouring such "tall" structures. Of course, if transitions from state 2 to state 1 were further strengthened at the expense of transitions from state 2 to itself, the projection of the policemen classes to the corresponding area would be favoured even further. This particular area in Fig.8 is the most favourable for the projection of the policeman classes with respect to other regions of the latent space. Finally, the respective area of the house classes in Fig.8 corresponds to the area where a strong tendency for self-loops for state 2 occurs, that favours the mapping of "short" structures. Clearly, despite of the similarity in the state-transition probabilities in the respective areas of the ship and house classes, the two classes are projected in well separated areas due to the different underlying structure of the means. The model-based nature of our visualisation plots has a potential to bring a degree of transparency in analysing and understanding of how the data items are organised in the visualisation plot in Fig.7(b).

We also trained GTM-HMTM on the dataset of quadtrees. We set $K = 3$ and the variance of the one-dimensional emissions equal to 1.0. However during training, GTM-HMTM displayed numerical problems that prevented us from using the dataset at the 64×64 resolution that we determined earlier. Thus, we reduced the images from 64×64 to 16×16 pixels. The results for GTM-HMTM on the quadtree dataset are

[3]recall that the values of y-coordinate in TPB data increase in a top-down direction.

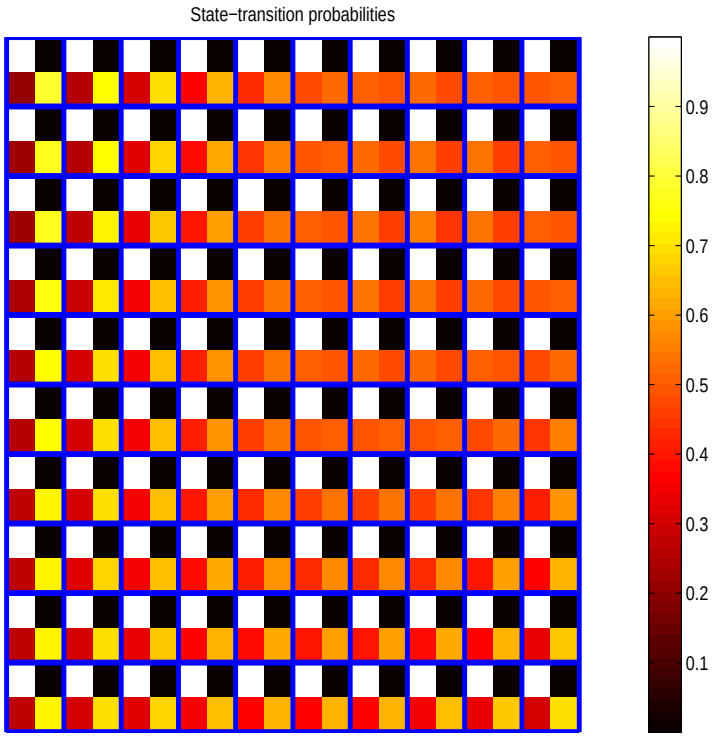


Fig. 8. TPB task: the plot is organised as a grid of $K \times K = 2 \times 2$ transition matrices, with each transition matrix corresponding to one local HMTMs underlying the visualisation plot.

displayed in Fig.10. Unfortunately, although a certain level of topographic organisation is evident, the model does not seem to be particularly successful at this task. We note certain trends such as the presence of images at the bottom of the plot of ducks facing to the right, while at the centre-left we come across images facing to the left. The top right is dominated to images of frontal views. Finally, close to the centre and slightly to the left, we note an overlapping of images of different orientations that have not been successfully organised on the map.

Further insight regarding the topographic organisation for the quadtree dataset can be gained by observing the plots for the state transitions in Fig.11 and the means of the emissions in Fig.12(a),12(b) and 12(c). The state transitions are very similar across the entire plot and only subtle variations are noticeable. All three plots for the means exhibit a very similar structure, with abrupt changes close to the centre of the respective plots. These observations suggest that the underlying local models are very similar in terms of transition probabilities, and that it is the means that mostly drive the topographical organisation. The abrupt changes noted in the the plots of the means, seem to be related to the overlapping of images of different orientations, noted at about the same location in Fig.10 (close to the centre of the plot).

Next, we present the results obtained by using SOMSD on the three datasets. We tried numerous parameter settings for SOMSD, all with rectangular lattices, Gaussian neighbourhood functions and 600 training iterations, and picked the best results, for the toy and TPB datasets where class information is provided, according to the criterion described below. For the toy dataset we found that the best parameters were a lattice of dimensions 28×28, a learning rate of 0.5, an initial radius of 5 and weighting coefficients of $\mu_1 = 0.99$ and $\mu_2 = 0.01$. For the TPB dataset the we chose a network of dimensions 114×87, a learning rate of 1.5, an initial radius of 60 and weighting coefficients of $\mu_1 = 0.01$ and $\mu_2 = 0.99$. Finally for the quadtree dataset, the parameters were a network of dimensions 90×90, a learning rate of 0.5, initial radius of 90 and weighting coefficients of $\mu_1 = 0.05$ and $\mu_2 = 0.95$. By inspecting the plots we can see that GTM-HMTM is better at the toy dataset, while SOMSD is better at the TPB dataset as it manages to distinguish between all of the classes, especially the classes of houses that are problematic in GTM-HMTM. This is interesting, because SOMSD seems to be more sensitive than GTM-HMTM to data items of shallow structure. On the other hand, SOMSD does not discover the sub-classes that GTM-HMTM does for the policemen and ships. Regarding the quadtree dataset, although we tried numerous parameter settings we could not obtain a good result for the same dataset of 16×16 of images. Nevertheless, when we further reduced the dimensions of the images down to 8×8 pixels, SOMSD was able to a achieve good topographic organisation, displayed in in Fig.14, indicating that the transformed images preserve sufficient information. However, SOMSD does not seem to utilise the entire map when projecting the data, as it does for the toy and TPB dataset (the same problem also appeared when training with smaller maps). Thus, in Fig.14 only the region of the map containing projections is displayed.

The toy data set is clearly biased towards GTM-HMTM and SOMSD was not able to cluster the trees in a fashion reflecting

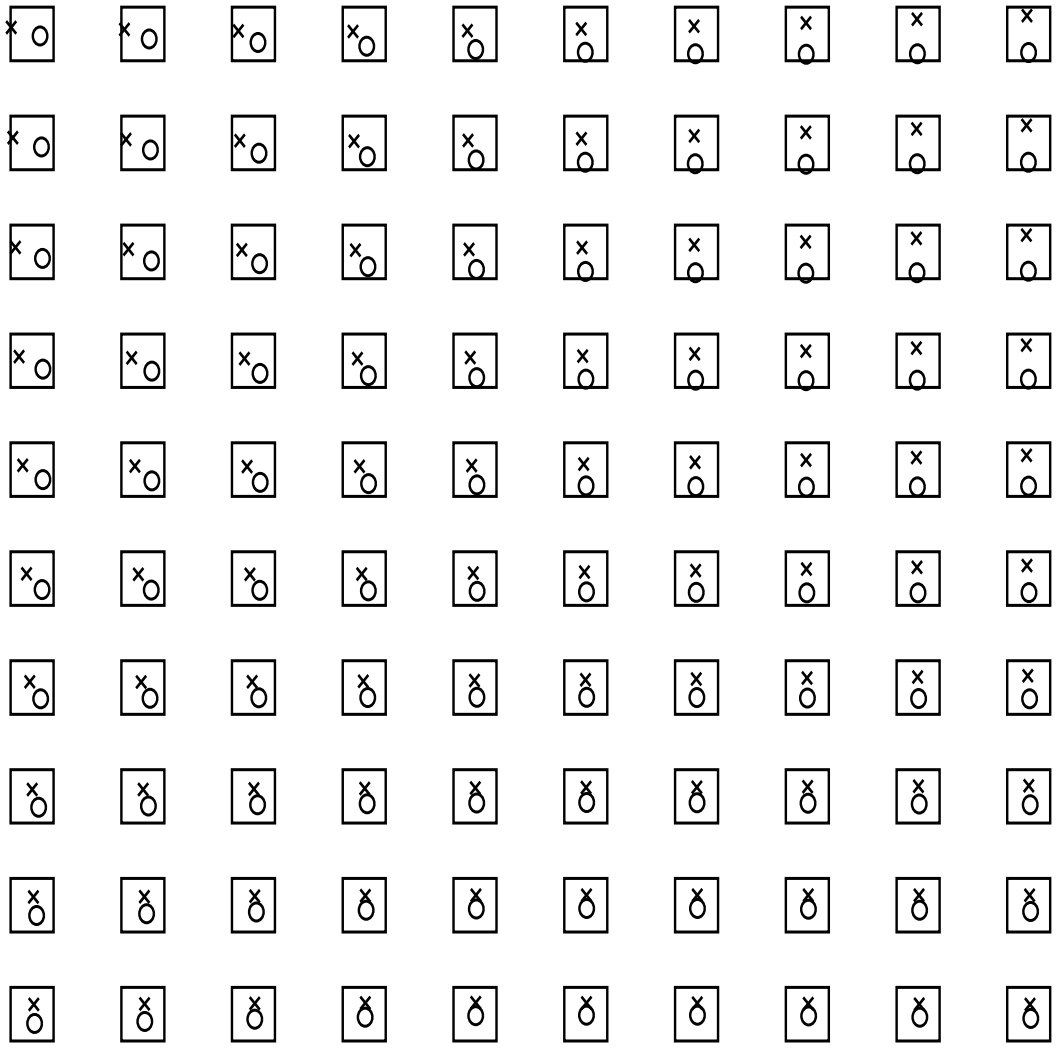


Fig. 9. TPB task: means of emissions for states $k = 1, 2$ corresponding to the local HMTMs. Means corresponding to states 1 and 2 are marked with circles and crosses respectively.

the organisation of the underlying generative process. This raises an important point we would like to stress. Of course, there is no single best model for topographic organisation of data of a given form. This issue is even more pronounced in the case of unsupervised learning in structured domains, where for models such as SOMSD a clear cost functional being minimised during the parameter fitting process is missing. Besides not knowing exactly what the model is optimised for, there is an additional difficulty: recursive models such as SOMSD are non-autonomous dynamical systems that can be difficult to understand. But without a clear understanding of the underlying dynamics, we can never know exactly what is driving topographical organisation of the projections. As a consequence, given a new tree, it might be possible to guess where its image on the SOMSD map will lie, but understanding the process of its formation will remain problematic. Consequently, it is difficult to grasp the structure of a trained topographic map on a deeper level - one is forced to produce only verbal descriptions as in the previous two paragraphs.

In contrast, a clear model-based formulation of GTM-HMTM enables us not only to analyse and understand the trained model (and hence understand the organisation of the map in terms of organisation of local prototype HMTM noise models), but also to understand exactly what kinds of data our model is suitable for. It is also important to understand that the class of noise models (in our case HMTM) inherently dictates along what lines will the data projections/representations be organised on the visualisation plot. Close regions on the computer screen (latent space $\mathcal{V}$) will correspond to "close" noise models (HMTMs) and hence trees will be organised on the map with respect to how closely they adhere to different HMTMs defined by different regions on the map. We will study the issues of metric on the latent space induced by the choice of noise models in the Section VIII.

There are two problems in applying HMTM to the third dataset of quadtrees. First, the particular HMTM emission model (Gaussian) does not correspond well to the discrete nature of quadtree labels employed here - binomial or multinomial distribution would fit the bill much better. Second, dendencies in the quadtree structures can be better modeled

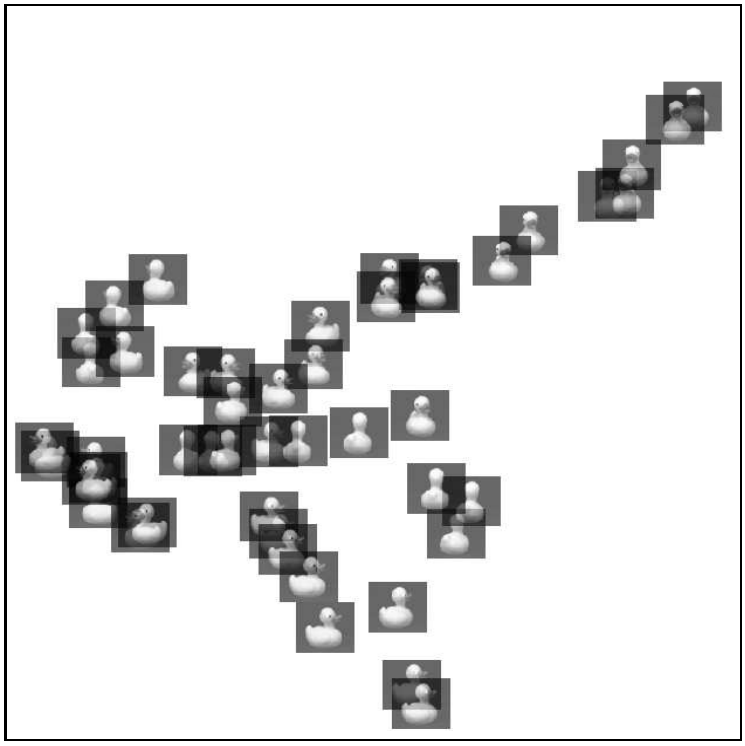


Fig. 10. Visualisation of reduced resolution quadtree dataset (16×16) for GTM-HMTM. Images are plotted as transparent to allow visibility of overlapping ones.

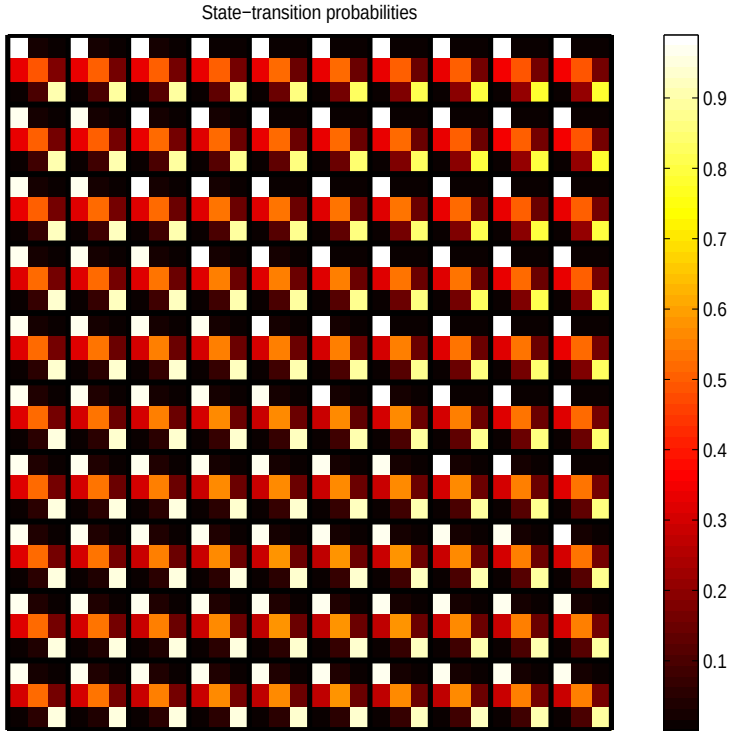
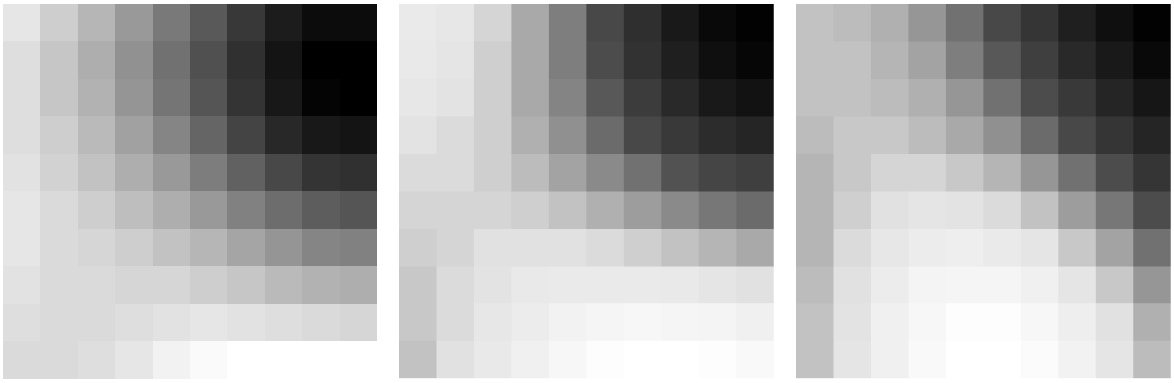


Fig. 11. Quadtree task: the plot is organised as a grid of $K \times K = 3 \times 3$ transition matrices, with each transition matrix corresponding to one local HMTM underlying the visualisation plot.

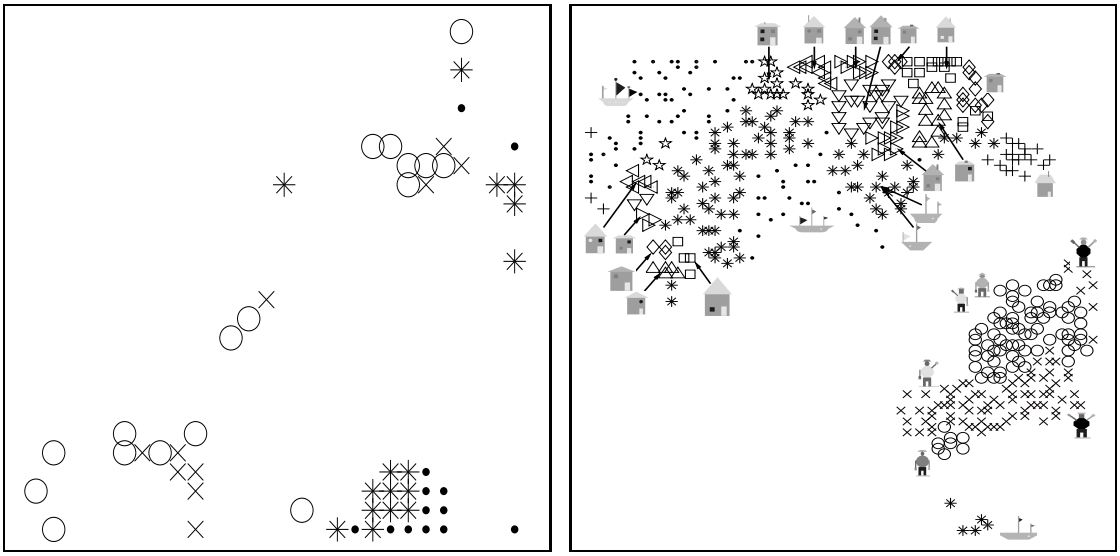
using observable and position-dependent first-order transitions, This corresponds directly to the nature of the process by which quadtrees are generated from images. In our framework, it is easy to modify local noise models for trees such that disrete nature of emissions and dependency structure of quadtrees are accounted for. This will be done in section VII.

Because of the absence of a clear cost function, the performance of SOMSD was measured in [1] as the accuracy of



(a) (b) (c)

Fig. 12. Quadtree task: means of emissions of GTM-HMTM for quadtree dataset. Each subplot corresponds to a state.



(a) (b)

Fig. 13. Visualisation of toy (a) and TPB (b) dataset using SOMSD.

classification of data into known classes (the class information was not used during the training) using data representations on the map. After the map formation, a secondary hold-out test dataset was used. Items from the test set were represented on the trained map and each test item was predicted to have the class label of its closest neighbour (from the training set) on the map. The accuracy was then defined as the percentage of correctly classified test points. The results of this measure on the toy dataset were 90% and 60% for GMT-HMTM and SOMSD respectively. The results were reversed for the TPB dataset: GMT-HMTM and SOMSD achieved 55% and 95% of accuracy respectively. Regarding the quadtree dataset, where data items cannot be classified as they do not belong to distinct classes, we formulated an analogous performance criterion. Again after map formation, the items from a secondary hold-out test dataset were represented on the trained map. The difference of the viewing angle of the photographed rubber duck of each test item and its closest neighbour (from the training set) was calculated. The performance criterion was then calculated as the average of the absolute differences of the viewing angles (a lower score is preferable). According to this criterion GTM-HMTM achieved a score of 30.833 and SOMSD a score of 23.26.

We stress again, that such a procedure makes sense only when the class organisation of the data correlates with the driving force behind the topographic map formation. If, for example, the classes of trees are organised along the lines that cannot be reasonably captured by HMTM modelling, there is simply no reason why the achieved classification accuracy of GMT-HMTM should be high. But low classification rate would just mean that our model-driven topographic map formation does not correlate with the particular class labelling scheme. In such cases one can simply switch to local noise models that are more correlated with the class labelling. Alternatively, one might say that he/she wanted to see topographically organised data representations driven by aspects captured by HMTM (or any other noise model employed) and stick with the obtained topographic maps, irrespective of the class labels. This is an unsupervised learning setting after all. Again, without knowing the exact mechanism behind the topographic map formation, it is problematic to assign any performance-related interpretation to the classification rate obtained on the trained map.

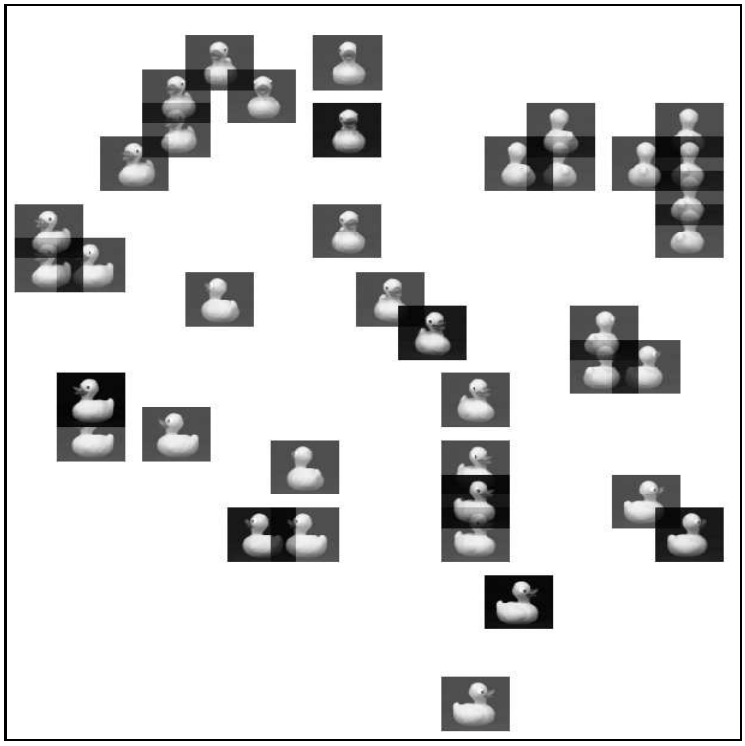


Fig. 14. Visualisation of reduced resolution quadtree dataset (8×8) for SOMSD.

VII. Alternative local noise model formulations

In general, GTM consists of two components, a generative probabilistic model, and a suitable constrained parametrisation. For a given data type, there can be many different choices of local noise models in the data space. Since in the GTM formulation, two data items are viewed as being "close" if they are generated by the same (or similar) local noise model, the nature of the topographic map is determined by our choice of the noise model. The methodology is general and can easily accommodate different notions of similarity between structured data items[4]. As a brief demonstration we introduce another generative model on trees, which we call the *Markov Tree Model* (MTM). Unlike HMTM, MTM has observable states (which makes their interpretation and estimation easier) and are capable of limited accommodation of positional information in the trees.

A MTM is an observable process operating on trees of a particular class, trees where each parent node has exactly R children. The process generates a discrete label $\boldsymbol{o}_u \in \{1, \ldots, K\}$ for each node $u \in \mathcal{U}_{\boldsymbol{y}}$ of tree $\boldsymbol{y}$. Labelling of the tree proceeds in a top-down fashion, starting from the root node u_1, and working down towards the leaves. At transition from a parent node $\rho(u)$ to its child node u, a label $\boldsymbol{o}_u$ is assigned. The label assignment is conditionally dependent on label $\boldsymbol{o}_{\rho(u)}$ of the parent $\rho(u)$ and the position $pos(u) \in \{1, 2, ..., R\}$ of the child. This dependency is expressed as a probability $p(\boldsymbol{o}_u | \boldsymbol{o}_{\rho(u)}, pos(u))$. A MTM is the first-order Markov process, where the label of a node is conditionally independent from all the labels that belong to ancestor nodes, given its position and parent node. Transitions are governed by R transition matrices $\boldsymbol{B}^{(r)}$, one for each child position $r = 1, \ldots, R$, with entries $b^r_{kl} = p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r)$ for $k, l = 1, \ldots K$.

We impose a non-informative flat initial state probability distribution for the root node, $p(\boldsymbol{o}_1) = \frac{1}{K}$. The scaled[5] model log-likelihood for a dataset $\mathcal{T} = \{\boldsymbol{y}^{(1)}, ..., \boldsymbol{y}^{(N)}\}$ is then:

$$
\begin{aligned}
\log p(\boldsymbol{y}) &\propto \sum_{n=1}^{N} \sum_{u=2}^{U_n} \sum_{k=1}^{K} \sum_{l=1}^{K} \sum_{r=1}^{R} \delta_{\boldsymbol{o}_{\rho(u)},k} \delta_{\boldsymbol{o}_u,l} \delta_{pos(u),r} \\
&\quad \times \log p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{l=1}^{K} \sum_{r=1}^{R} \left(\sum_{u=2}^{U_n} \delta_{\boldsymbol{o}_{\rho(u)},k} \delta_{\boldsymbol{o}_u,l} \delta_{pos(u),r} \right) \\
&\quad \times \log p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{l=1}^{K} \sum_{r=1}^{R} \nu^{(n)}_{rkl} \\
&\quad \times \log p(\boldsymbol{o}_u = l | \boldsymbol{o}_{\rho(u)} = k, pos(u) = r),
\end{aligned}
\tag{30}
$$

where $\delta_{\cdot,\cdot}$ is the Kronecker delta and $\nu^{(n)}_{rkl}$ is the number of times the transition from a parent node labelled by k to the r-th child labelled by l occurs in tree $\boldsymbol{y}^{(n)}$.

In order to built a GTM that utilises MTMs as noise models, we need a suitable parametrisation of the multinomial emissions along the lines of GTM-HMTM (see section V):

$$
\boldsymbol{B}^r(\boldsymbol{x}_c) = \{g_l(\boldsymbol{W}^{r,k} \boldsymbol{\phi}(\boldsymbol{x}))\}_{k,l=1 \ldots K}. \tag{31}
$$

[4] or, more generally, different data types.

[5] discarding $p(\boldsymbol{o}_1) = \frac{1}{K}$.

For C latent points organised on a regular grid in space $\mathcal{V}$ and dataset $\mathcal{T}$, a constrained mixture of C MTMs is formed. Using (30), the complete data log-likelihood (upto a constant factor) of GTM-MTM is:

$$\begin{aligned}\log \mathcal{L} &= \sum_{n=1}^{N} \log \sum_{c=1}^{C} \prod_{u=2}^{U_n} p(\boldsymbol{o}_u|\boldsymbol{o}_{\rho(u)}, pos(u)) \\ &= \sum_{n=1}^{N} \log \sum_{c=1}^{C} \prod_{u=2}^{U_n} \prod_{k=1}^{K} \prod_{l=1}^{K} \prod_{r=1}^{R} \\ &\quad p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r)^{\delta_{\boldsymbol{o}_{\rho(u)},k}\delta_{\boldsymbol{o}_u,l}\delta_{pos(u),r}}.\end{aligned} \tag{32}$$

Employing the E-M formulation, we seek to maximise the expected complete-data log-likelihood, calculated in the E-step:

$$\begin{aligned}E[\log \mathcal{L}] = &\sum_{n=1}^{N} \sum_{c=1}^{C} \sum_{k=1}^{K} \sum_{l=1}^{K} \sum_{r=1}^{R} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}) \nu_{rkl}^{(n)} \\ &\times \log p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r).\end{aligned} \tag{33}$$

Partial derivatives of (33) with respect to elements of parameter matrices $\boldsymbol{W}^{s,t}$ to be used in the M-step read:

$$\begin{aligned}\frac{\partial}{\partial w_{jm}^{st}} E[\log \mathcal{L}] = &\sum_{n=1}^{N} \sum_{c=1}^{C} \sum_{l=1}^{K} p(\boldsymbol{x}_c|\boldsymbol{y}^{(n)}) \nu_{stl}^{(n)} \\ &\times \frac{1}{p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s)} \\ &\times \phi_m(\boldsymbol{x}_c) p(\boldsymbol{o}_u = j|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s) \\ &\times \Big(\delta_{l,j} - p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = t, pos(u) = s)\Big).\end{aligned} \tag{34}$$

We experimented with two datasets, a toy dataset constructed by sampling three MTM models, and the quadtree dataset that was used in GTM-HMTM.

The results for the toy dataset are displayed in Fig.15. The three clusters are clearly discerned. Data points are numbered to indicate the MTM they originate from. Regarding the quadtree dataset, the results are displayed in Fig.16. Training GTM-MTM with the original resolution of 64×64 did not yield any numerical problems as experienced in GTM-MTM. Clearly, GTM-MTM has achieved a much higher quality of topographic organisation than GTM-HMTM. The upper-left corner is dominated by images of ducks facing to the right, while the lower-right corner is dominated by images facing to the left. In between the two, close to the centre of the map, we find images of frontal views. Finally, as we move from the lower left corner towards the top, we come across images of rear views.

Furthermore, we measured the performance of GTM-MTM using the same criterion as we did for GTM-HMTM and SOMSD on the quadtree dataset in Section VI-C. Recall that GTM-HMTM and SOMSD scored 30.833 and 23.26 respectively, while GTM-MTM achieved the lowest score of 18.125.

VIII. Magnification factors for GTM-HMTM

The topographic organisation of the data on the two-dimensional space allows the inspection of spatial data relationships in the high dimensional space and the inference of potential clusters and relations between the data points. However, we must keep in mind that the data-points are projected on the latent space in a non-linear way. This means that data points that are distant in the data space (assuming some notion of a metric in the data space) may be projected close to each other. Thus, even though the smooth mapping does preserve the neighbourhood structure, it does not necessarily preserve distances in the latent space.

Magnification factors for the GTM are introduced in [22]. Each point $\boldsymbol{x}$ of latent space $\mathcal{V}$ is mapped to the mean $\boldsymbol{\mu} \in \mathbf{R}^d$ of a spherical Gaussian density via $\Gamma(\boldsymbol{x}) = \boldsymbol{\mu} = \boldsymbol{W}\boldsymbol{\phi}(\boldsymbol{x})$. Thus, a smooth two-dimensional statistical manifold of isotropic Gaussians is induced. An infinitisimal displacement $d\boldsymbol{x}$ at a latent point $\boldsymbol{x} \in \mathcal{V}$ is mapped to a displacement $d\boldsymbol{y}$ at $\Gamma(\boldsymbol{x})$ on the manifold of Gaussian means: $d\boldsymbol{y} = \boldsymbol{J}d\boldsymbol{x}$, where $\boldsymbol{J}$ is Jacobian of the mapping Γ at $\boldsymbol{x}$. Consider an infinitesimal rectangle located at $\boldsymbol{x} \in \mathcal{V}$ and defined by displacements $d\boldsymbol{x}_1 = dx_1\boldsymbol{e}_1$, $d\boldsymbol{x}_2 = dx_2\boldsymbol{e}_2$ along a cartesian coordinate system $\{\boldsymbol{e}_1, \boldsymbol{e}_2\}$ in $\mathcal{V}$. Its area is equal to $dA = dx_1 dx_2$. The area of the Γ-image of this rectangle[6] is equal to $dA' = dA \cdot \det(\boldsymbol{J}^T\boldsymbol{J})$ [22]. In [22] magnification factors are calculated as the ratio $dA'/dA = \det(\boldsymbol{J}^T\boldsymbol{J})$ by which the area of minute local patches at $\boldsymbol{x} \in \mathcal{V}$ expands/contracs as the patches get embedded in the high-dimensional data space via the non-linear mapping mapping Γ.

However, as pointed out in [22], this definition of magnification factors gives no information as to what directions in the latent space correspond to dominant stretching/contraction. Moreover, expansion in one direction can be compensated by contraction in another (orthogonal) one. Bishop, Svensén and Williams [22] suggest to perform eigenanalysis of the local metric tensors $\boldsymbol{J}^T\boldsymbol{J}$. In this spirit, we will quantify magnification by examining the effect of minute local directional displacements in the latent space on the corresponding noise models (HMTM or MTM).

It should be noted that the approach of [22] concentrating on expansions/contractions on the manifold of means of local spherical Gaussians in the data space cannot be directly applied here. The generative probabilistic visualization models studied in this paper naturally induce a metric in the structured data space. Loosely speaking, two data items (trees) are considered to be close (or similar) if both of them are well-explained by the same underlying noise model (e.g. HMTM) from the two-dimensional manifold of noise models. We emphasise, that in this framework, the distance between structured data items is implicitly defined by the local noise models that drive the topographic map formation. If the noise model changes, the perception of what kind of data items are considered similar changes as well. In this paper, we quantify the extend to which small positional changes in the latent space lead to changes in the distributions defined by the corresponding noise models. It is important to

[6] the image is not necessarily a rectangle

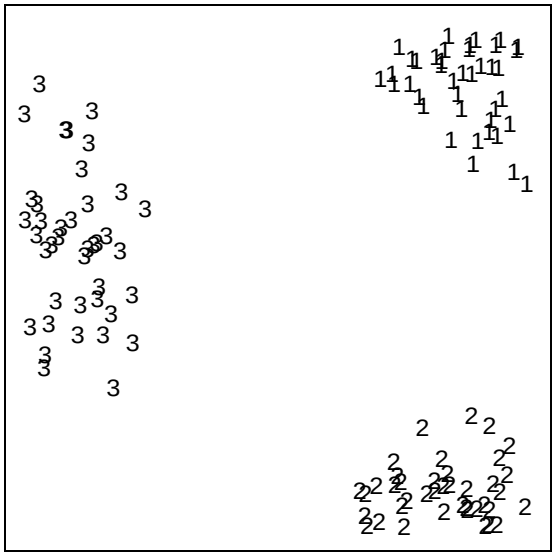


Fig. 15. Visualisation of toy dataset for GTM-MTM.

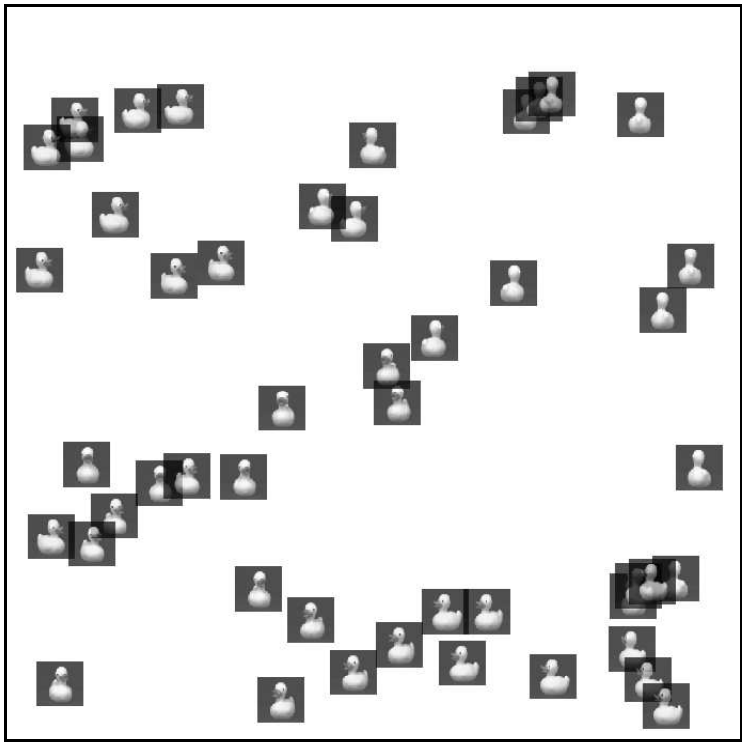


Fig. 16. Visualisation of quadtree dataset (64×64) for GTM-MTM.

quantify the changes in a parametrization-free manner - we use approximations of Kullback-Leibler divergence.

In our model, each point $\boldsymbol{x}$ in the latent space $\mathcal{V}$ maps to a HMTM $p(\cdot|\boldsymbol{x})$. The entire latent space induces a smooth two-dimensional manifold $\mathcal{M}$ of HMTMs embedded in the manifold $\mathcal{H}$ of all HMTMs of the same structural form. In order to appreciate how the latent space is stretched or *magnified* we perturb a latent point $\boldsymbol{x} \in \mathcal{V}$ by an infinitesimally small perturbation $d\boldsymbol{x}$. The new point $\boldsymbol{x}+d\boldsymbol{x}$ maps to a new HMTM $p(\cdot|\boldsymbol{x} + d\boldsymbol{x})$. We measure the statistical "distance" between the two HMTMs, $p(\cdot|\boldsymbol{x})$ and $p(\cdot|\boldsymbol{x} + d\boldsymbol{x})$, by employing the Kullback-Leibler divergence (KLD). The KLD between two HMTMs cannot be analytically calculated (as HMTMs are latent variable models), but it can be practically measured as the *observed* KLD, $\hat{D}_{KL}$. In particular, for two HMTMs $p(\cdot|\boldsymbol{x})$ and $p(\cdot|\boldsymbol{x} + \boldsymbol{dx})$, and a set of N trees $\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, ..., \boldsymbol{y}^{(N)}$, generated by $p(\cdot|\boldsymbol{x})$, the observed KLD is

$$\hat{D}_{KL}[p(\cdot|\boldsymbol{x})||p(\cdot|\boldsymbol{x} + \boldsymbol{dx})] = \frac{1}{N}\sum_{n=1}^{N} \log \frac{p(\boldsymbol{y}^{(n)}|\boldsymbol{x})}{p(\boldsymbol{y}^{(n)}|\boldsymbol{x} + \boldsymbol{dx})}. \tag{35}$$

We employ two different approaches for measuring KLD. The first approach is an approach that can be applied to any

noise model while the second one applies specifically for HMTMs (and HMMs as a special case).

A. KLD as Fisher information matrix

Each latent point has two coordinates $\boldsymbol{x} = (x_1, x_2)^T \in \mathcal{V}$ is mapped via a smooth non-linear mapping to an HMTM $p(\cdot|\boldsymbol{x})$ on the manifold $\mathcal{M}$ of HMTMs. If we displace $\boldsymbol{x}$ by an infinitesimally small perturbation $d\boldsymbol{x}$, the KLD $D_{KL}[p(\cdot|\boldsymbol{x})\|p(\cdot|\boldsymbol{x}+d\boldsymbol{x})]$ between the corresponding noise models $p(\cdot|\boldsymbol{x}), p(\cdot|\boldsymbol{x}+d\boldsymbol{x}) \in \mathcal{M}$ can be approximated via Fisher information matrix

$$\boldsymbol{F}(\boldsymbol{x}) = -E_{p(\cdot|\boldsymbol{x})}[\nabla^2 \log p(.|\boldsymbol{x})], \tag{36}$$

that acts like a metric tensor on the Riemannian manifold $\mathcal{M}$ [24]:

$$D_{KL}[p(\cdot|\boldsymbol{x})\|p(\cdot|\boldsymbol{x}+d\boldsymbol{x})] = d\boldsymbol{x}^T \boldsymbol{F}(\boldsymbol{x})\ d\boldsymbol{x}. \tag{37}$$

The situation is illustrated in Fig.17. We base the calculation of the observed Fisher information matrix on the upward recursion of the likelihood estimation for HMTMs [18].

- The recursion starts from the leaves u of the tree:

$$\beta_k(u;\boldsymbol{x}) = p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x}). \tag{38}$$

- Recursive step for non-leaves nodes u:

$$\begin{aligned}\beta_k(u;\boldsymbol{x}) &= p(\boldsymbol{y}_u|q_u = k;\boldsymbol{x})\\ &= \Big\{ \prod_{v\in c(u)} p(\boldsymbol{y}_v|q_u = k;\boldsymbol{x})\Big\}\\ &\times p(\boldsymbol{o}_u|q_u = k;\boldsymbol{x})\\ &= \Big\{ \prod_{v\in c(u)} \sum_{i=1}^{K} p(\boldsymbol{y}_v|q_v = i;\boldsymbol{x})\\ &\times p(q_v = i|q_u = k;\boldsymbol{x})\Big\} p(\boldsymbol{o}_u|q_u = k;\boldsymbol{x})\\ &= \Big\{ \prod_{v\in c(u)} \sum_{i=1}^{K} \beta_i(v;\boldsymbol{x}) p(q_v = i|q_u = k;\boldsymbol{x})\Big\}\\ &\times p(\boldsymbol{o}_u|q_u = k;\boldsymbol{x}).\end{aligned} \tag{39}$$

- Final step:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{k=1}^{K} \beta_k(1;\boldsymbol{x}) p(q_1 = k|\boldsymbol{x}). \tag{40}$$

Starting again from the leaves of the tree $\boldsymbol{y}$, we recursively evaluate 1st-order derivatives of likelihood with respect to the latent coordinates x_1, x_2. Let $r \in \{1, 2\}$, 1st-order derivative is (details can be found in Appendix B):

$$\begin{aligned}\frac{\partial}{\partial x_r} p(\boldsymbol{y}|\boldsymbol{x}) = &\sum_{k=1}^{K} \Big\{ \frac{\partial}{\partial x_r} \beta_k(1;\boldsymbol{x})\Big\} p(q_1 = k|\boldsymbol{x})\\ &+ \beta_k(1;\boldsymbol{x}) \Big\{ \frac{\partial}{\partial x_r} p(q_1 = k|\boldsymbol{x})\Big\}.\end{aligned} \tag{41}$$

The recursion is repeated once more, this time calculating the 2nd-order derivatives. Let $r, s \in \{1, 2\}$:

$$\begin{aligned}\frac{\partial^2}{\partial x_r \partial x_s} p(\boldsymbol{y}|\boldsymbol{x}) = &\sum_{k=1}^{K} \Big\{ \frac{\partial^2}{\partial x_r \partial x_s} \beta_k(1;\boldsymbol{x})\Big\} p(q_1 = k|\boldsymbol{x})\\ &+ \Big\{ \frac{\partial}{\partial x_r} \beta_k(1;\boldsymbol{x})\Big\} \Big\{ \frac{\partial}{\partial x_s} p(q_1 = k|\boldsymbol{x})\Big\}\\ &+ \Big\{ \frac{\partial}{\partial x_s} \beta_k(1;\boldsymbol{x})\Big\} \Big\{ \frac{\partial}{\partial x_r} p(q_1 = k|\boldsymbol{x})\Big\}\\ &+ \beta_k(1;\boldsymbol{x}) \Big\{ \frac{\partial^2}{\partial x_r \partial x_s} p(q_1 = k|\boldsymbol{x})\Big\}.\end{aligned} \tag{42}$$

Finally, we need the derivatives of the log-likelihood:

$$\begin{aligned}&\frac{\partial^2}{\partial x_r \partial x_s} \log p(\boldsymbol{y}|\boldsymbol{x}) =\\ &\frac{p(\boldsymbol{y}|\boldsymbol{x}) \frac{\partial^2}{\partial x_r \partial x_s} p(\boldsymbol{y}|\boldsymbol{x}) - \frac{\partial}{\partial x_r} p(\boldsymbol{y}|\boldsymbol{x}) \frac{\partial}{\partial x_s} p(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y}|\boldsymbol{x})^2}.\end{aligned} \tag{43}$$

The elements of the information matrix are calculated given a set of trees $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$ sampled by model $p(\cdot|\boldsymbol{x})$:

$$\hat{\mathbf{F}}(\boldsymbol{x})_{r,s} = -\frac{1}{N} \sum_{n=1}^{N} \frac{\partial^2}{\partial x_r \partial x_s} \log p(\boldsymbol{y}^{(n)}|\boldsymbol{x}). \tag{44}$$

The above calculations depend on the 1st- and 2nd-order derivatives of initial state, state transition and state-conditional emission probabilities with respect to the latent coordinates:

1st-order derivatives for initial probability for state k according to (16):

$$\begin{aligned}&\frac{\partial}{\partial x_r} p(q_1 = k|\boldsymbol{x}) = g_k(\boldsymbol{A}^{(\boldsymbol{\pi})} \boldsymbol{\phi}(\boldsymbol{x}))\\ &\times \Big(\boldsymbol{A}_k^{(\boldsymbol{\pi})} \frac{\partial \phi(\boldsymbol{x})}{\partial x_r} - \sum_{i=1}^{K} g_i(\boldsymbol{A}^{(\boldsymbol{\pi})} \boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{A}_i^{(\boldsymbol{\pi})} \frac{\partial \phi(\boldsymbol{x})}{\partial x_r}\Big).\end{aligned} \tag{45}$$

2nd-order derivatives for initial probability for state k:

$$\begin{aligned}&\frac{\partial^2}{\partial x_r \partial x_s} p(q_1 = k|\boldsymbol{x}) = \frac{\partial}{\partial x_s} g_k(\boldsymbol{A}^{(\boldsymbol{\pi})} \boldsymbol{\phi}(\boldsymbol{x}))\\ &\times \Big(\boldsymbol{A}_k^{(\boldsymbol{\pi})} \frac{\partial \phi(\boldsymbol{x})}{\partial x_r} - \sum_{i=1}^{K} g_i(\boldsymbol{A}^{(\boldsymbol{\pi})} \boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{A}_i^{(\boldsymbol{\pi})} \frac{\partial \phi(\boldsymbol{x})}{\partial x_r}\Big)\\ &+ g_k(\boldsymbol{A}^{(\boldsymbol{\pi})} \boldsymbol{\phi}(\boldsymbol{x})) \Big(\boldsymbol{A}_k^{(\boldsymbol{\pi})} \frac{\partial^2 \phi(\boldsymbol{x})}{\partial x_r \partial x_s}\\ &- \sum_{i=1}^{K} \Big[\frac{\partial}{\partial x_s} g_i(\boldsymbol{A}^{(\boldsymbol{\pi})} \boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{A}_i^{(\boldsymbol{\pi})} \frac{\partial \phi(\boldsymbol{x})}{\partial x_r}\\ &+ g_i(\boldsymbol{A}^{(\boldsymbol{\pi})} \boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{A}_i^{(\boldsymbol{\pi})} \frac{\partial^2 \phi(\boldsymbol{x})}{\partial x_r \partial x_s}\Big]\Big).\end{aligned} \tag{46}$$

1st-order derivatives for transition probability from state l to state k according to (17):

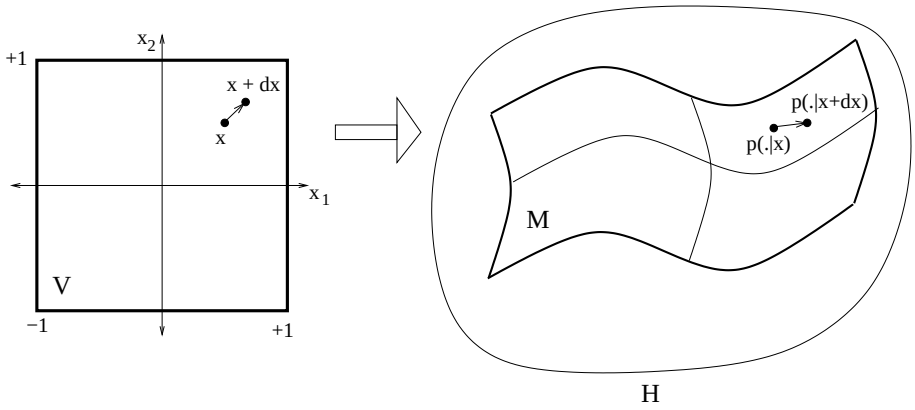


Fig. 17. Two-dimensional manifold $\mathcal{M}$ of local noise models $p(\cdot|\boldsymbol{x})$ parametrised by the latent space $\mathcal{V}$ through (14) and (16)-(18). The manifold is embedded in manifold $\mathcal{H}$ of all noise models of the same form. Latent coordinates $\boldsymbol{x}$ are displaced to $\boldsymbol{x}+\boldsymbol{dx}$. Kullback-Leibler divergence $D_{KL}[p(\cdot|\boldsymbol{x})\|p(\cdot|\boldsymbol{x}+\boldsymbol{dx})]$ between the corresponding noise models $p(\cdot|\boldsymbol{x}), p(\cdot|\boldsymbol{x}+\boldsymbol{dx}) \in \mathcal{M}$ can be determined via Fisher information matrix $\boldsymbol{F}(\boldsymbol{x})$ that acts like a metric tensor on the Riemannian manifold $\mathcal{M}$.

$$\frac{\partial}{\partial x_r} p(q_u = k|q_{\rho(u)} = l; \boldsymbol{x}) = g_k(\boldsymbol{A}^{(\boldsymbol{T}_l)}\boldsymbol{\phi}(\boldsymbol{x})) \times \left(\boldsymbol{A}_k^{(\boldsymbol{T}_l)} \frac{\partial \phi(\boldsymbol{x})}{\partial x_r} - \sum_{i=1}^{K} g_i(\boldsymbol{A}^{(\boldsymbol{T}_l)}\boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{A}_i^{(\boldsymbol{T}_l)} \frac{\partial \phi(\boldsymbol{x})}{\partial x_r} \right). \tag{47}$$

2nd-order derivatives for transition probability from state l to state k:

$$\begin{aligned} \frac{\partial^2}{\partial x_r \partial x_s} p(q_u = k|q_{\rho(u)} = l; \boldsymbol{x}) &= \frac{\partial}{\partial x_s} g_k(\boldsymbol{A}^{(\boldsymbol{T}_l)}\boldsymbol{\phi}(\boldsymbol{x})) \\ &\times \left(\boldsymbol{A}_k^{(\boldsymbol{T}_l)} \frac{\partial \phi(\boldsymbol{x})}{\partial x_r} - \sum_{i=1}^{K} g_i(\boldsymbol{A}^{(\boldsymbol{T}_l)}\boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{A}_i^{(\boldsymbol{T}_l)} \frac{\partial \phi(\boldsymbol{x})}{\partial x_r} \right) \\ &+ g_k(\boldsymbol{A}^{(\boldsymbol{T}_l)}\boldsymbol{\phi}(\boldsymbol{x})) \Bigg(\boldsymbol{A}_k^{(\boldsymbol{T}_l)} \frac{\partial^2 \phi(\boldsymbol{x})}{\partial x_r \partial x_s} \\ &- \sum_{i=1}^{K} \left[\frac{\partial}{\partial x_s} g_i(\boldsymbol{A}^{(\boldsymbol{T}_l)}\boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{A}_i^{(\boldsymbol{T}_l)} \frac{\partial \phi(\boldsymbol{x})}{\partial x_r} \right. \\ &\left. + g_i(\boldsymbol{A}^{(\boldsymbol{T}_l)}\boldsymbol{\phi}(\boldsymbol{x})) \boldsymbol{A}_i^{(\boldsymbol{T}_l)} \frac{\partial^2 \phi(\boldsymbol{x})}{\partial x_r \partial x_s} \right] \Bigg). \end{aligned} \tag{48}$$

1st-order derivatives for means of the emission distribution for state k according to (18):

$$\frac{\partial}{\partial x_r} \boldsymbol{\mu}_k = \frac{\partial}{\partial x_r} \boldsymbol{A}^{(\boldsymbol{B}_k)} \phi(\boldsymbol{x}) = \boldsymbol{A}^{(\boldsymbol{B}_k)} \frac{\partial}{\partial x_r} \phi(\boldsymbol{x}). \tag{49}$$

2nd-order derivatives for means of the emission distribution for state k:

$$\frac{\partial^2}{\partial x_r \partial x_s} \boldsymbol{\mu}_k = \frac{\partial^2}{\partial x_r \partial x_s} \boldsymbol{A}^{(\boldsymbol{B}_k)} \phi(\boldsymbol{x}) = \boldsymbol{A}^{(\boldsymbol{B}_k)} \frac{\partial^2}{\partial x_r \partial x_s} \phi(\boldsymbol{x}). \tag{50}$$

Finally, we need to calculate the 1st- and 2nd-order derivatives of the basis functions:

$$\frac{\partial}{\partial x_r} \phi(\boldsymbol{x}) = \left[\frac{\partial \phi_1(\boldsymbol{x})}{\partial x_r}, \ldots, \frac{\partial \phi_M(\boldsymbol{x})}{\partial x_r} \right], \tag{51}$$

where for the RBF kernels:

$$\frac{\partial}{\partial x_r} \phi_m(\boldsymbol{x}) = -\frac{1}{\sigma^2} \phi_m(\boldsymbol{x})(\boldsymbol{x}_r - \boldsymbol{\mu}_{m,r}), \tag{52}$$

$$\frac{\partial^2}{\partial x_r \partial x_s} \phi(\boldsymbol{x}) = \left[\frac{\partial^2}{\partial x_r \partial x_s} \phi_1(\boldsymbol{x}), \ldots, \frac{\partial^2}{\partial x_r \partial x_s} \phi_M(\boldsymbol{x}) \right], \tag{53}$$

and

$$\begin{aligned} \frac{\partial^2}{\partial x_r \partial x_s} \phi_m(\boldsymbol{x}) &= -\frac{1}{\sigma^2} \phi_m(\boldsymbol{x}) \\ &+ (\boldsymbol{x}_r - \boldsymbol{\mu}_{m,r})(\boldsymbol{x}_s - \boldsymbol{\mu}_{m,s}) \frac{1}{\sigma^4} \phi_m(\boldsymbol{x}). \end{aligned} \tag{54}$$

In order to illustrate the magnification factors on manifold $\mathcal{M}$, we calculate the observed Fisher information matrix for each latent centre $\boldsymbol{x}_c, c = 1, 2, \ldots, C$. We can then compute the KLD between each $\boldsymbol{x}_c$ and its perturbation $\boldsymbol{x}_c + d\boldsymbol{x}$ by (37). Here we perturb each latent centre $\boldsymbol{x}_c$ in 16 regularly spaced directions on a small circle (we have set its radius to 10^{-5}). This is illustrated in Fig.18 for 8 directions. We note that alternatively, we could have used SVD decomposition of the Fisher information matrix to find and quantify the local dominant stretching directions in the latent space.

B. Direct recursive approximation of KLD

In [25] an efficient method for approximating the KLD between two HMTMs is presented. The approximation is based on the upward recursion and calculates an upper bound for KLD. It is particularly fast compared to the previous approach as its computations rely only on the parameters of the models and does not need to calculate the likelihoods of samples generated by the models.

With each hidden state $k = 1, 2..., K$ we associate an upward probability $\beta_k(u; \boldsymbol{x}) = p(\boldsymbol{y}_u|q_u = k, \boldsymbol{x})$. Given a tree $\boldsymbol{y}$, the likelihood $p(\boldsymbol{y}|\boldsymbol{x})$ can be efficiently calculated by the upward recursion, as in (38), (39) and (40).

The approximation relies on two results. First we note again that by definition, the KLD between two multidimensional distributions $\boldsymbol{w}$ and $\tilde{\boldsymbol{w}}$ is:

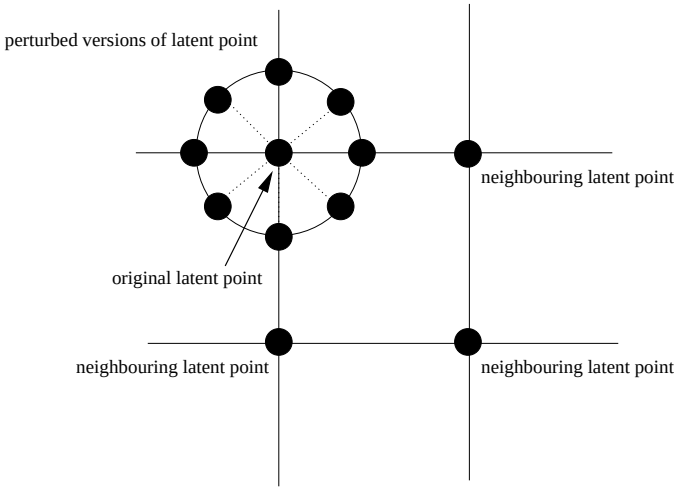


Fig. 18. A latent point in space $\mathcal{V}$ is perturbed in 8 regularly spaced directions on a small circle in order to measure the local magnification factor.

$$\kappa[\boldsymbol{w}, \tilde{\boldsymbol{w}}] = \sum_i w_i \log \frac{w_i}{\tilde{w}_i}. \tag{55}$$

Secondly, given two mixtures $F = \sum_i w_i f_i$ and $F' = \sum_i w'_i f'_i$ the following lemma is proved [25] for the KLD between them:

$$D_{KL}[F||F'] = D_{KL}[\sum_i w_i f_i || \sum_i w'_i f'_i] \leq \kappa[\boldsymbol{w}, \tilde{\boldsymbol{w}}] + \sum_i w_i D_{KL}[f_i||f'_i]. \tag{56}$$

Furthermore, in the case of D-dimensional Gaussian emissions we have:

$$D_{KL}[N(.;\boldsymbol{\mu}, \boldsymbol{C})||N(.;\boldsymbol{\mu}', \boldsymbol{C}')] = \frac{1}{2}\big[\log(\frac{det\boldsymbol{C}'}{det\boldsymbol{C}}) - D + tr(\boldsymbol{C}'^{-1}\boldsymbol{C}) + (\boldsymbol{\mu} - \boldsymbol{\mu}')^T \boldsymbol{C}'^{-1}(\boldsymbol{\mu} - \boldsymbol{\mu}')\big]. \tag{57}$$

This result is important because we will treat emissions $p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x})$ as mixtures of Gaussians (in our case presented here, the mixtures are simplified to single d-dimensional Gaussians). The KLD between two HMTMs $p(.|\boldsymbol{x})$ and $p(.|\boldsymbol{x} + \boldsymbol{dx})$ can be approximated as follows:

- Recursion starts at leaf nodes u; we treat emissions $p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x})$ as a mixture and apply (56) to obtain approximation D_k:

$$D_k[u; \boldsymbol{x}, \boldsymbol{x} + d\boldsymbol{x}] = D_{KL}[p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x})||p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x} + d\boldsymbol{x})] = D_{KL}[N(\boldsymbol{o}_u; \boldsymbol{\mu}_k, \boldsymbol{C})||N(\boldsymbol{o}_u; \boldsymbol{\mu}'_k, \boldsymbol{C}')]. \tag{58}$$

Primed quantities correspond to model $p(\cdot|\boldsymbol{x} + d\boldsymbol{x})$.

- Recursive step; for internal nodes u use (56) to rewrite (39):

$$\begin{aligned} D_k[u; \boldsymbol{x}, \boldsymbol{x} + d\boldsymbol{x}] = & \\ & D_{KL}[p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x})||p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x} + d\boldsymbol{x})] \\ & + \sum_{v \in c(u)} D_{KL}\Big[\sum_{i=1}^K \beta_i(v; \boldsymbol{x}) p(q_v = i|q_u = k; \boldsymbol{x})|| \\ & \sum_{i=1}^K \beta_i(v; \boldsymbol{x} + \boldsymbol{dx}) p(q_v = i|q_u = k; \boldsymbol{x} + d\boldsymbol{x})\Big] \\ \leq & D_{KL}[N(\boldsymbol{o}_u; \boldsymbol{\mu}_k, \sigma_k)||N(\boldsymbol{o}_u; \boldsymbol{\mu}'_k, \sigma'_k)] \\ & + \sum_{v \in c(u)} \Big(\kappa[p(q_v|q_u = k, \boldsymbol{x}), p(q_v|q_u = k, \boldsymbol{x} + d\boldsymbol{x})] \\ & + \sum_{i=1}^K p(q_v = i|q_u = k, \boldsymbol{x}) D_i[v; \boldsymbol{x}, \boldsymbol{x} + d\boldsymbol{x}]\Big). \end{aligned} \tag{59}$$

- Final step; the upper bound of KLD is found by again using (56) to rewrite (40) in the root node (node number 1):

$$\mathcal{K}[\boldsymbol{y}; \boldsymbol{x}, \boldsymbol{x} + d\boldsymbol{x}] = \kappa[p(q_1|\boldsymbol{x}), p(q_1|\boldsymbol{x} + d\boldsymbol{x})] + \sum_{k=1}^K p(q_1 = k|\boldsymbol{x}) D_k[1; \boldsymbol{x}, \boldsymbol{x} + d\boldsymbol{x}]. \tag{60}$$

Given a set of trees $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$ sampled by model $p(\cdot|\boldsymbol{x})$ an estimate of KLD is approximated as

$$\hat{D}_{KL}[p(\cdot|\boldsymbol{x})||p(\cdot|\boldsymbol{x}+d\boldsymbol{x})] = \frac{1}{N}\sum_{n=1}^N \mathcal{K}(\boldsymbol{y}^{(n)}, \boldsymbol{x}, \boldsymbol{x}+d\boldsymbol{x}). \tag{61}$$

The above formula requires some explanation. It can be seen in all equations above, that the *labels* of the trees are not utilised for estimating the KLD approximation as they eventually vanish due to (57), i.e. KLD between Gaussians

can be calculated via a closed-form formula. The sample of trees is still needed though, because the recursion above is driven by their topological *structure*. Note however, that in our sample, all trees have the same topology. In this case the KLD can be approximated using a single tree:

$$\hat{D}_{KL}[p(\cdot|\boldsymbol{x})||p(\cdot|\boldsymbol{x}+d\boldsymbol{x})] = \mathcal{K}(\boldsymbol{y}^{(1)}, \boldsymbol{x}, \boldsymbol{x}+d\boldsymbol{x}). \quad (62)$$

The same procedure as for the Fisher information matrix is applied here too, namely perturbing a latent point in 16 regularly spaced directions on a small circle (again the radius is set to 10^{-5}) and measuring the KLD between the original HMTM model $\boldsymbol{x}$ and the perturbed model $\boldsymbol{x}+d\boldsymbol{x}$.

C. Results on Magnification Factors

Magnification plots for the toy and TPB datasets have been produced following the approaches of Sections VIII-A and VIII-B. In both approaches we define on the latent space a rectangular grid of 25x25 latent points. This divides the grid in 625 squares with a latent point at the centre of each square. Increasing the number of grid points results to finer magnification plots. Each latent centre is perturbed in 16 regularly spaced directions on a small circle of radius 10^{-5}. KLD between the noise model corresponding to the original latent point and the noise model corresponding to its perturbed version is measured via both approaches. For each latent point, out of the 16 directions of perturbation, the direction of maximal magnification is represented by a straight line drawn through the centre of the corresponding square. The length of the line signifies the level of the magnification, i.e. shorter lines indicate lower magnification, longer lines indicate higher magnification. The shading of each square also signifies the level of magnification; brighter squares are associated with higher magnification, whereas darker squares are associated with lower magnification.

We observe that both plots of Fig.19(a) and 19(b) illustrate very similar magnifications for the toy dataset. It can be seen in both figures that the data have been organised in 4 distinct clusters, well separated by light regions that signify that the clusters have clear boundaries and are indeed different from each other (compare with Fig.7(a)).

Fig.20(a) and 20(b) illustrate the resulting plots for the TPB dataset. The light region concentrated in the left upper corner of the plot concerns the topographic organisation of the two ship classes (see Fig.7(b)). The high magnification indicates that data points projected in this area are very dissimilar to each other as abrupt changes occur in the underlying models. This is verified by the fact that we have identified that class $*$, the class of ships with two masts, has been split into three sub-clusters. On the other hand, the classes of policemen exhibit a gentler separation between them as indicated by the moderately light area close to the right upper corner. Finally the region of the classes of houses, which as we saw earlier have all been projected in one super-cluster, is characterised by very low magnification. This suggests, that indeed that this super-cluster is dense and that the underlying models fail to discern differences between house patterns.

In Fig.21 magnification factors for the quadtree dataset are displayed using the KLD approximation method. The plot does not impart information on the presence of any clusters. However, when inspected in conjunction with the state transitions in Fig.11 and the means of the emissions in Fig.12(a),12(b) and 12(c), we can see how it reflects the situation of the visualisation plot in Fig.10 where an overlapping of images of different orientations occurs. We recall that transition probabilities vary only little across the plot. Also, the plots of means exhibit a common abrupt change close to their respective centres. We see that the magnification factors effectively summarise the behaviour of the means corresponding to the three states: a high magnification is observed at the centre of the plot where the means change the most.

IX. Magnification factors for GTM-MTM

We also calculate magnification factors GTM-MTM. For MTMs we can resort to a fast approximation of KLD by assuming that the trees are sufficiently deep and that the compared MTMs are not too dissimilar. Using a result from [26], we calculate the KLD between a MTM addressed by $p(\cdot|\boldsymbol{x})$ and its perturbed version $p(\cdot|\boldsymbol{x}+d\boldsymbol{x})$ by:

$$\begin{aligned}&\hat{D}_{KL}[p(\cdot|\boldsymbol{x})||p(\cdot|\boldsymbol{x}+d\boldsymbol{x})] = \\ &\sum_{r=1}^{R}\sum_{k=1}^{K}\pi_k^r \sum_{l=1}^{K} p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r|\boldsymbol{x}) \\ &\times \log \frac{p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r|\boldsymbol{x})}{p(\boldsymbol{o}_u = l|\boldsymbol{o}_{\rho(u)} = k, pos(u) = r|\boldsymbol{x}+d\boldsymbol{x})}, \end{aligned} \quad (63)$$

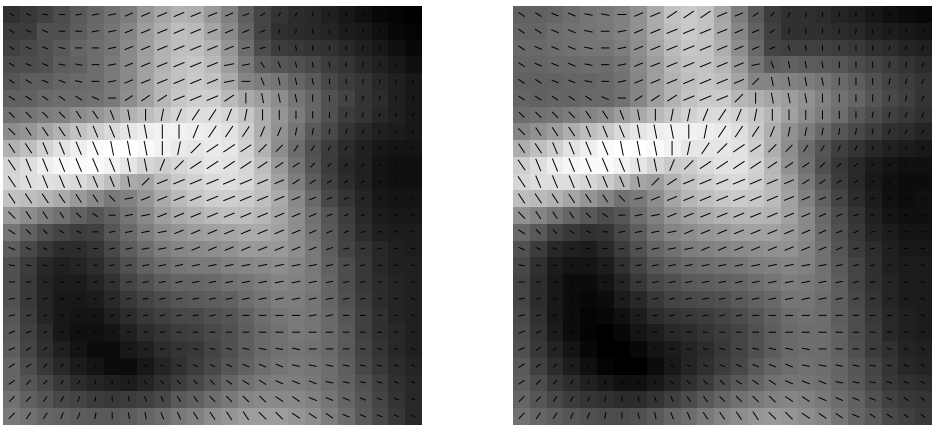
where probabilities π_k^r are obtained as the normalised left eigenvector of the state transition matrix with eigenvalue 1.

The magnification factors for the toy dataset are presented in Fig.22(a). The clusters illustrated in Fig.15 are visible here too, clearly separated by bright boundaries signifying stretches in the latent space. Inspecting the state transitions for the toy dataset (similar to Fig.8), a clear structure is observed. For example in Fig.22(b), where the transition probabilities that correspond to the 3-rd child are illustrated, the regions underlying the three clusters indicate different trends. In the case of quadtree data set, the *local* metric structure of GTM-MTM was varying rather slowly and so the magnification factor plot (reflecting local differentiable structure of the noise manifold) was almost flat. The topographic organisation is driven by small local changes in the noise models. Maps of magnification factors are not well suited for such situations.

We also calculated magnification factors via Fisher information matrices. The magnification factor plots were virtually identical to the ones obtained via KLD approximation.

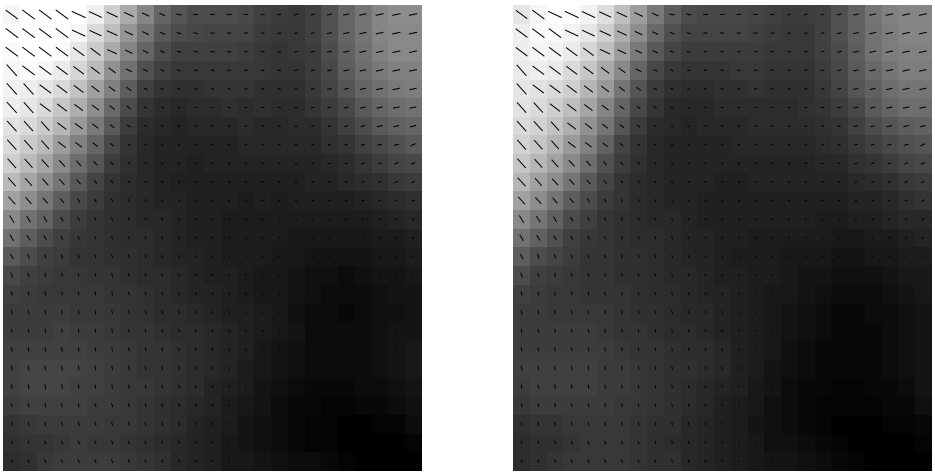
X. Discussion

Unlike in recursive neural-based approaches to topographic maps formation, the optimisation of the free parameters of GTM-HMTM is driven by a well defined cost function - negative log of the likelihood function (13). The complexity of the E-step is $\mathcal{O}(NCUK^2)$, where U is the average number



(a) (b)

Fig. 19. Fisher information (a) and KLD approximation (b) for GTM-HMTM on toy dataset.



(a) (b)

Fig. 20. Fisher information (a) and KLD approximation (b) for GTM-HMTM on TPB dataset.

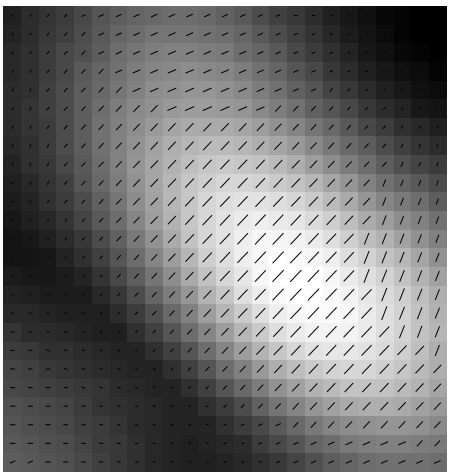
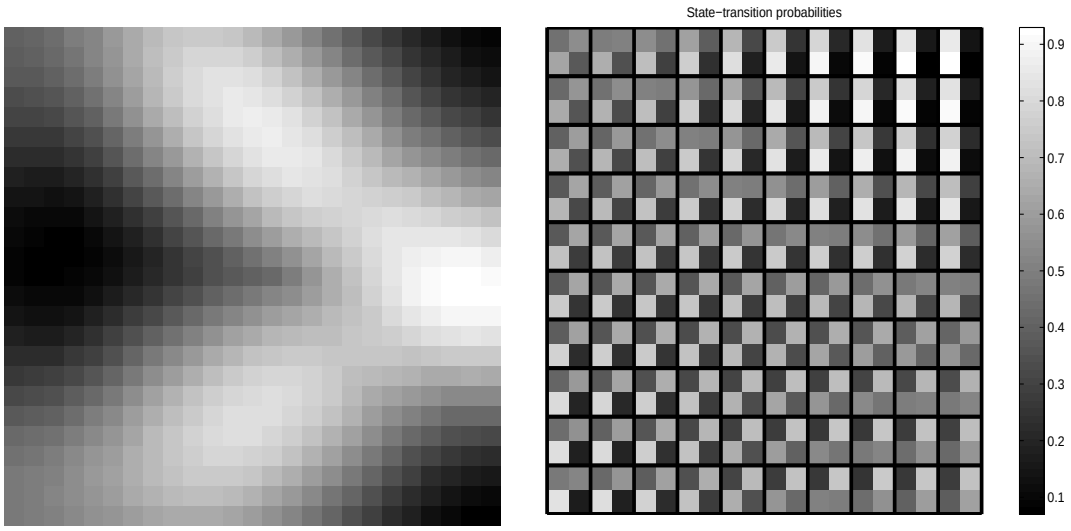


Fig. 21. KLD approximation for GTM-HMTM on quadtree dataset.

of nodes in a tree data item. The complexity of the M-step depends on the optimisation procedure employed. A scaled conjugate gradient implementation has a complexity of $\mathcal{O}(2W^2)$, where W is the number of parameters of the



(a) (b)

Fig. 22. Magnification factors for GTM-MTM on toy dataset (a). State transitions for 3-rd child node for toy dataset (b).

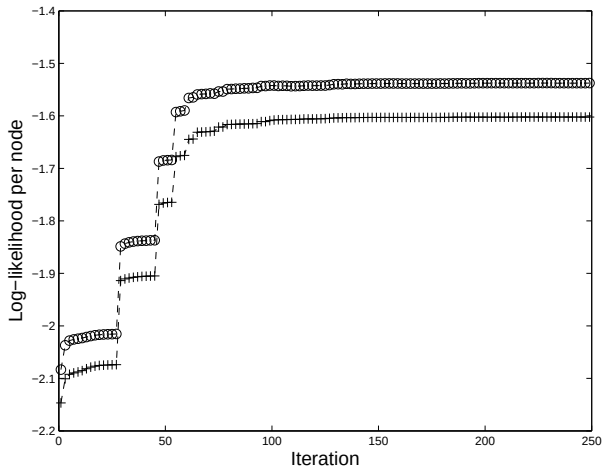


Fig. 23. Evolution of log-likelihood for GTM-HMTM on training (line with + marker) and validation (line with o marker) set.

model. For GTM-HMTM this is $W = M(K + K^2 + Kd)$ (covariance of emissions is not modelled directly). Despite the high number of parameters W, we found that in practice this does not present a significant difficulty in training the model because of its highly constrained nature. What seems to make training difficult is the lack of a good initialisation procedure.

Having trained GTM-HMTM via E-M, the data points are projected from the data space on the latent space. To this end we calculate the responsibilities (posterior probabilities) of the underlying HMTMs models. Our mixture of noise models (HMTMs) is a constrained mixture, constrained by the smooth two-dimensional structure of the latent space. Hence, neighbouring latent points correspond to noise models (local HMTMs) that lead to similar answers (responsibilities) when queried about a certain data point (tree). The data point can then be placed at the mean location of the responsibilities (10) in order to reflect the contribution of all local models. The same also applies to a newly incoming data point. The method is transparent in the sense that we can understand why a certain point was placed in a particular position in the visualisation (latent) space by inspecting the underlying local models. This level of transparency is not readily provided when visualising new trees using recursive neural-based techniques such as SOMSD. Also, in such models it is difficult to quantify what the induced notion of similarity between trees really means. For example, why exactly does SOMSD place some trees close together and some further apart? Can one have some form of control over the shaping of visualisation plots? In GTM-HMTM this is done by imposing the form of local noise models. Then two data items (trees) are viewed as “similar”, if they are highly probable under the same local noise model (HMTM or MTM). Of course, it is possible to have different notions of similarity even for the same data set. It can well be that there are two users that would like to see the same data items organised on the visualisation plot using different criteria for item similarity, depending on what aspects of the data they are interested in. Then it is up to the user to formulate the appropriate noise model and let the data visualisation be driven

by it. This capability of accommodating alternative noise models was demonstrated in Section VII, where an alternative noise model, the MTM, was introduced. The same machinery for parametrisation and optimisation as for the GTM-HMTM was adopted to derive GTM-MTM. The two extensions rely on the same principles for achieving topographic organisation but work with different notions of similarity. The latent variable nature of HMTM gives it potentially a richer expressive power, but at the price of introducing additional level of unobserved variables in the GTM-HMTM formulation. HMTM also allows for continuous observations. On the other hand, HMTM does not discriminate between children nodes. In contrast, MTM is not a latent variable model and works with discrete emissions. Crucially, it does consider an ordering on the children nodes.

In both GTM formulations, there is a potential of gaining an insight about the driving forces behind the topographic map formations by inspecting the learnt structure local noise models as in Fig.8, 9, 11, 12(a), 12(b), 12(c), 19(a), 19(b), 20(a), 20(b), 21, 22(a) and 22(b).

Another important advantage of the principled probabilistic model formulation is the possibility to inspect the tendency of the model to overfit the training data, by measuring the log-likelihood on an independent validation set. For example, for the GTM-HMTM and the TPB task, the validation set consists of 88 patterns produced in the same manner as the training set. During training, the log-likelihoods of the model on the training and validation sets were calculated in each iteration. The evolution of the log-likelihood for both data sets is presented in Fig.23. It is apparent that the constrained nature of our model prevents it from overfitting the training sample. For the case of GTM-MTM and quadtree dataset, we show log-likelihood evolutions in Fig.24(a) and 24(b). We examine two training sessions. The first one corresponds to a training session where overfitting occurs. In order to avoid overfitting we changed the variance of the radial basis functions ϕ of the RBF network from 1.0 to 2.0 and performed a second training session. Increasing the variance in the RBF network, has the effect of making the basis functions wider, less localised, blending them to a higher degree in their overlapping regions. In terms of the GTM, this has the effect of not allowing local noise models to become very different from their neighbours in terms of parameters, which enforces a form of regularisation. The evolution of the log-likelihood with RBF variance equal to 2.0 is illustrated in Fig.24(b). Even though, we observed very similar topographic organisations in both training sessions, the second session achieves better generalisation performance, which is advantageous if new data items are to be projected on the map. We stress, that checking the model likelihood on a hold-out sample is a natural way of detecting possible overfitting. In case of a highly overfitted model, it would be difficult to interpret visualisation plots as representing any general tendency in the data. We also note that dealing with the overfitting issue in case of recursive neural based formulations is problematic - it is not clear how one would go about quantifying the level of overfitting in the first place.

As for the measure of accuracy used in Section VI-C to quantify the quality of visualisations, we note that, as discussed at the end of Section VI-C, it is not directly related to what the models are optimised for and therefore does not constitute an objective criterion to discriminate between the models.

The generative nature of GTM-HMTM allows further data exploration after a first impression of the visualisation through hierarchical visualisation in the spirit of [27]. Also, the incorporation of priors on the model parameters is straightforward. MAP estimation is possible by a adding to (23) an additional term, namely the log-likelihood of the prior density on the parameters.

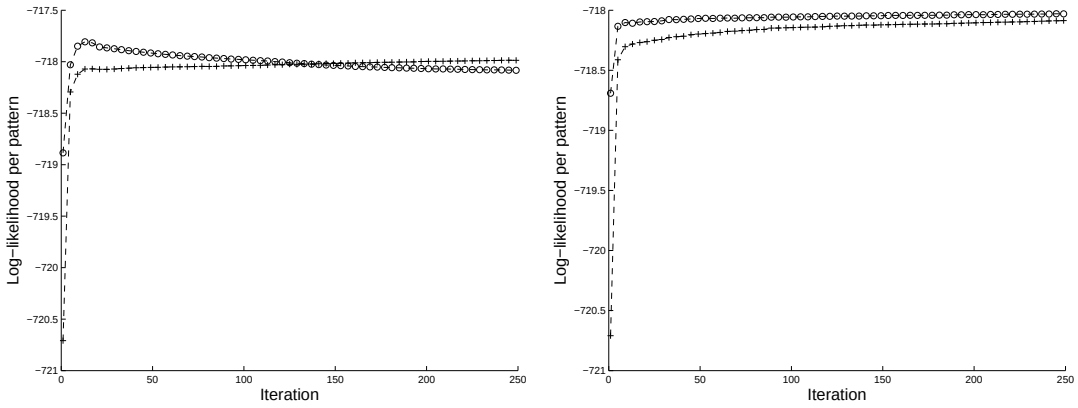
Furthermore, the fact that the non-linear mapping of GTM-HMTM from the latent space to the local model space is smooth allows us to calculate magnification factors. We have presented two approaches toward this end, precise Fisher information matrix and KLD approximation, which have been verified by experiments. This constitutes a useful tool for the study of clusters and can be used to further interpret the visualisation plot as magnifications of the manifold $\mathcal{M}$ of local models. The presence of low magnification in a certain region can help us infer the presence of a potential cluster as we expect the generative process of the underlying local models to change only slightly as we move in that region. On the contrary, high magnification signifies the volatility of the local models and hence that data points in the region are expected to differ significantly from each other.

XI. Conclusions

We have presented GTM-HMTM, an alternative method for topographic organisation of tree-structured data. The model is an extension of the GTM algorithm and thus is based on a sound probabilistic formulation. Compared with recursive neural based approaches, the main advantages of GTM-HMTM include:

1) The model-based nature of GTM-HMTM may provide a degree of transparency of the visualisation plot formation and a principled interpretation of the data visualisations.
2) There is a well defined cost function driving the model training that can be used for principled model comparison.
3) Trained models can be checked in a natural way for possible overfitting by comparing log-likelihoods on training and validation sets.
4) Alternative local noise model formulations allow the user to express in a natural way a notion of structured data similarity that will be driving the topographical organisation in visualisation plots.
5) Smooth mapping from the latent soace to the local noise model space enables the calculation of magnification factors, a useful tool that supplements our understanding of the visualisation plots.
6) It is straightforward to extend the methodology to include hierarchical visualisations for detailed user-guided exploration of subsets of data.

For illustration purposes we compared visualisation plots obtained with GTM formulations with those obtained by SOMSD - a representative of recursive neural-based topographic map constructions.



(a) (b)

Fig. 24. Evolution of log-likelihood for GTM-MTM on training (line with + marker) and validation (line with o marker) set. In (a) the variance of the radial basis functions is set to 1.0 and in (b) it is set to 2.0.

APPENDIX A
NOTATIONS

Commonly used symbols and notation are summarised in Table IV.

TABLE IV
COMMONLY USED SYMBOLS AND NOTATION.

Notation	Meaning
$\boldsymbol{A}^{(\boldsymbol{\pi})}$	matrix parametrising initial probabilities for GTM-HMTM
$\boldsymbol{A}^{(\boldsymbol{T}_k)}$	matrix parametrising transition probabilities from state k for GTM-HMTM
$\boldsymbol{A}^{(\boldsymbol{B}_k)}$	matrix parametrising means of Gaussians at state k for GTM-HMTM
$D_{KL}[\cdot\|\|\cdot]$	Kullback-Leibler divergence
$E[\cdot]$	expectation operator
$\mathcal{L}$	model likelihood
$\mathcal{N}$	Gaussian density
g	softmax function
$\boldsymbol{t}$	instance of vector variable
$\boldsymbol{y}$	instance of tree structure variable
$\boldsymbol{o}_u$	label of node u
q_u	instantiation of state variable of node u
$\beta_k(u)$	downward probability for node u at state $q_u = k$
$\mathcal{M}$	two-dimensional manifold of HMTMs constrained on $\mathcal{V}$
$\mathcal{V}$	continuous Euclidean latent space
$\boldsymbol{x}$	latent point, belongs to $\mathcal{V}$
$p(\cdot\|\boldsymbol{x})$	probabilistic model addressed by latent point $\boldsymbol{x}$
$p(\cdot\|\boldsymbol{x}+\boldsymbol{dx})$	perturbed version of probabilistic model $p(\cdot\|\boldsymbol{x})$ addressed by latent point $(\boldsymbol{x}+\boldsymbol{dx})$
ϕ	non-linear smooth basis function
z	hidden indicator variable defined in E-M training

APPENDIX B
DERIVATIVES FOR KLD AS FISHER INFORMATION

Based on the recursive evaluation of likelihood of (38), (39) and (40), we recursively calculate 1st-order derivatives of the likelihood with respect to the latent coordinates x_1, x_2. Let $r \in \{1, 2\}$:

- The recursion starts from the leaves of the tree:
$$\frac{\partial}{\partial x_r}\beta_k(u;\boldsymbol{x}) = \frac{\partial}{\partial x_r}p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x}). \tag{64}$$
- Recursive step for non-leaves nodes u:
$$\begin{aligned}\frac{\partial}{\partial x_r}\beta_k(u;\boldsymbol{x}) &= \frac{\partial}{\partial x_r}p(\boldsymbol{y}_u|q_u = k;\boldsymbol{x}) \\ &= \left\{\frac{\partial}{\partial x_r}\gamma_k(u;\boldsymbol{x})\right\}p(\boldsymbol{o}_u|q_u = k;\boldsymbol{x}) \\ &\quad + \gamma_k(u;\boldsymbol{x})\left\{\frac{\partial}{\partial x_r}p(\boldsymbol{o}_u|q_u = k;\boldsymbol{x})\right\}, \end{aligned}\tag{65}$$
where
$$\begin{aligned}\gamma_k(u;\boldsymbol{x}) &= \prod_{v\in c(u)}\zeta_k(u,v;\boldsymbol{x}), \\ \zeta_k(u,v;\boldsymbol{x}) &= \sum_{i=1}^{K}\beta_i(v;\boldsymbol{x})p(q_v = i|q_u = k;\boldsymbol{x}), \\ \frac{\partial}{\partial x_r}\gamma_k(u;\boldsymbol{x}) &= \gamma_k(u;\boldsymbol{x})\sum_{v\in c(u)}\frac{\partial}{\partial x_r}\log\zeta_k(u,v;\boldsymbol{x}), \\ \frac{\partial}{\partial x_r}\zeta_k(u,v;\boldsymbol{x}) &= \sum_{i=1}^{K}\frac{\partial}{\partial x_r}\beta_i(v;\boldsymbol{x})p(q_v = i|q_u = k;\boldsymbol{x}) \\ &\quad + \sum_{i=1}^{K}\beta_i(v;\boldsymbol{x})\frac{\partial}{\partial x_r}p(q_v = i|q_u = k;\boldsymbol{x}). \end{aligned}\tag{66}$$
- Final step:
$$\begin{aligned}\frac{\partial}{\partial x_r}p(\boldsymbol{y}|\boldsymbol{x}) &= \sum_{k=1}^{K}\left\{\frac{\partial}{\partial x_r}\beta_k(1;\boldsymbol{x})\right\}p(q_1 = k|\boldsymbol{x}) \\ &\quad + \beta_k(1;\boldsymbol{x})\left\{\frac{\partial}{\partial x_r}p(q_1 = k|\boldsymbol{x})\right\}. \end{aligned}\tag{67}$$

We repeat the recursion once more, this time calculating the 2nd-order derivatives. Let $r, s \in \{1, 2\}$:

- The recursion starts from the leaves of the tree:
$$\frac{\partial^2}{\partial x_r \partial x_s}\beta_k(u;\boldsymbol{x}) = \frac{\partial^2}{\partial x_r \partial x_s}p(\boldsymbol{o}_u|q_u = k, \boldsymbol{x}). \tag{68}$$

- Recursive step:

$$\begin{aligned}\frac{\partial^2}{\partial x_r \partial x_s}\beta_k(u;\boldsymbol{x}) &= \frac{\partial^2}{\partial x_r \partial x_s}p(\boldsymbol{y}_u|q_u=k;\boldsymbol{x}) \\ &= \frac{\partial^2}{\partial x_r \partial x_s}\gamma_k(u;\boldsymbol{x})p(\boldsymbol{o}_u|q_u=k;\boldsymbol{x}) \\ &+ \frac{\partial}{\partial x_r}\gamma_k(u;\boldsymbol{x})\frac{\partial}{\partial x_s}p(\boldsymbol{o}_u|q_u=k;\boldsymbol{x}) \\ &+ \frac{\partial}{\partial x_s}\gamma_k(u;\boldsymbol{x})\frac{\partial}{\partial x_r}p(\boldsymbol{o}_u|q_u=k;\boldsymbol{x}) \\ &+ \gamma_k(u;\boldsymbol{x})\frac{\partial^2}{\partial x_r \partial x_s}p(\boldsymbol{o}_u|q_u=k;\boldsymbol{x}),\end{aligned} \tag{69}$$

where

$$\begin{aligned}\frac{\partial^2}{\partial x_r \partial x_s}\gamma_k(u;\boldsymbol{x}) &= \gamma_k(u;\boldsymbol{x})\Bigg(\sum_{v\in c(u)}\frac{\partial}{\partial x_r}\log\zeta_k(u,v;\boldsymbol{x})\Bigg) \\ &\times\Bigg(\sum_{v\in c(u)}\frac{\partial}{\partial x_s}\log\zeta_k(u,v;\boldsymbol{x})\Bigg) \\ &+ \gamma_k(u;\boldsymbol{x})\Bigg(\sum_{i=1}^{K}\frac{-1}{(\zeta_k(u,v;\boldsymbol{x}))^2} \\ &\times\frac{\partial}{\partial x_r}\zeta_k(u,v;\boldsymbol{x})\frac{\partial}{\partial x_s}\zeta_k(u,v;\boldsymbol{x}) \\ &+ \frac{1}{\zeta_k(u,v;\boldsymbol{x})}\frac{\partial^2}{\partial x_r \partial x_s}\zeta_k(u,v;\boldsymbol{x})\Bigg).\end{aligned} \tag{70}$$

and

$$\begin{aligned}\frac{\partial^2}{\partial x_r \partial x_s}\zeta_k(u,v;\boldsymbol{x}) &= \sum_{i=1}^{K}\frac{\partial^2}{\partial x_r \partial x_s}\beta_i(v;\boldsymbol{x}) \\ &\times p(q_v=i|q_u=k;\boldsymbol{x}) \\ &+ \frac{\partial}{\partial x_r}\beta_i(v;\boldsymbol{x})\frac{\partial}{\partial x_s}p(q_v=i|q_u=k;\boldsymbol{x}) \\ &+ \frac{\partial}{\partial x_s}\beta_i(v;\boldsymbol{x})\frac{\partial}{\partial x_r}p(q_v=i|q_u=k;\boldsymbol{x}) \\ &+ \beta_i(v;\boldsymbol{x})\frac{\partial^2}{\partial x_r \partial x_s}p(q_v=i|q_u=k;\boldsymbol{x}).\end{aligned} \tag{71}$$

- Final step:

$$\begin{aligned}\frac{\partial^2}{\partial x_r \partial x_s}p(\boldsymbol{y}|\boldsymbol{x}) &= \sum_{k=1}^{K}\Bigg\{\frac{\partial^2}{\partial x_r \partial x_s}\beta_k(1;\boldsymbol{x})\Bigg\}p(q_1=k|\boldsymbol{x}) \\ &+ \Bigg\{\frac{\partial}{\partial x_r}\beta_k(1;\boldsymbol{x})\Bigg\}\Bigg\{\frac{\partial}{\partial x_s}p(q_1=k|\boldsymbol{x})\Bigg\} \\ &+ \Bigg\{\frac{\partial}{\partial x_s}\beta_k(1;\boldsymbol{x})\Bigg\}\Bigg\{\frac{\partial}{\partial x_r}p(q_1=k|\boldsymbol{x})\Bigg\} \\ &+ \beta_k(1;\boldsymbol{x})\Bigg\{\frac{\partial^2}{\partial x_r \partial x_s}p(q_1=k|\boldsymbol{x})\Bigg\}.\end{aligned} \tag{72}$$

ACKNOWLEDGMENT

The authors would like to thank Markus Hagenbuchner and Alessandro Sperduti for providing them with the SOMSD software.

REFERENCES

[1] M. Hagenbuchner, A. Sperduti, and A. C. Tsoi, "A self-organizing map for adaptive processing of structured data," *Neural Networks, IEEE Transactions on*, vol. 14, no. 3, pp. 491–505, 2003.
[2] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.
[3] T. Heskes, "Energy functions for self-organizing maps," in *Kohonen Maps*, S. Oja and E. Kaski, Eds. Amsterdam: Elsevier, 1999, pp. 303–315.
[4] C. M. Bishop, M. Svensén, and C. K. I. Williams, "GTM: The generative topographic mapping," *Neural Computation*, vol. 10, no. 1, pp. 215–234, 1998.
[5] G. de A. Barreto, A. Araújo, and S. Kremer, "A taxanomy of spatiotemporal connectionist networks revisited: The unsupervised case," *Neural Computation*, vol. 15, pp. 1255–1320, 2003.
[6] B. Hammer, A. Micheli, M. Strickert, and A. Sperduti, "A general framework for unsupervised processing of structured data," *Neurocomputing*, vol. 57, pp. 3–35, 2004.
[7] G. Chappell and J. Taylor, "The temporal kohonen map," *Neural Networks*, vol. 6, pp. 441–445, 1993.
[8] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski, "Recurrent SOM with local linear models in time series prediction," *6th European Symposium on Artificial Neural Networks*, pp. 167–172, 1998.
[9] K. Horio and T. Yamakawa, "Feedback self-organizing map and its application to spatio-temporal pattern classification," *International Journal of Computational Intelligence and Applications*, vol. 1, no. 1, pp. 1–18, 2001.
[10] T. Voegtlin, "Recursive self-organizing maps," *Neural Networks*, vol. 15, no. 8-9, pp. 979–991, 2002.
[11] M. Strickert and B. Hammer, "Merge SOM for temporal data," *Neurocomputing*, vol. 64, pp. 39–72, 2005.
[12] M. Hagenbuchner, A. Sperduti, and A. C. Tsoi, "Contextual processing of graphs using self-organizing maps," in *Proceedings of the European Syposium on Artificial Neural Networks (ESANN)*, 2005, pp. 399–404.
[13] B. Hammer, A. Micheli, A. Sperduti, and M. Strickert, "Recursive self-organizing network models," *Neural Networks*, vol. 17, no. 8-9, pp. 1061–1085, 2004.
[14] M. Varsta, J. Heikkonen, and J. del R. Millan, "Context learning with the self organizing map," *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4–6*, pp. 197–202, 1997.
[15] J.-B. Durand, P. Goncalves, and Y. Guedon, "Computational methods for hidden markov tree models-an application to wavelet trees," *Signal Processing, IEEE Transactions on*, vol. 52, no. 9, pp. 2552–2560, 2004.
[16] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
[17] P. Tino, A. Kaban, and Y. Sun, "A generative probabilistic approach to visualizing sets of symbolic sequences," *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–706, 2004.
[18] M. Crouse, R. Nowak, and R. Baraniuk, "Wavelet -Based Statistical Signal Processing Using Hidden Markov Models," *IEEE Transactions on Signal Processing*, vol. 46, no. 4, pp. 886–902, 1998.
[19] M. Hagenbuchner and A. Tsoi, "The traffic policeman benchmark," in *European Symposium on Artificial Neural Networks*, M. Verleysen, Ed. D-Facto, April 1999, pp. 63–68.
[20] H. Samet, *The design and analysis of spatial data structures*. Addison Wesley, 1990.
[21] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, "The Amsterdam library of object images," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 103–112, 2005.
[22] C. M. Bishop, M. Svensén, and C. K. I. Williams, "Magnification factors for the GTM algorithm," in *Proceedings IEE Fifth International Conference on Artificial Neural Networks*, 1997, pp. 64–69.
[23] T. M. Cover and J. A. Thomas, *Elements of information theory*. Wiley-Interscience, 1991.
[24] S. Kullback, *Information theory and statistics*. Wiley, New York, NY, 1959.

[25] M. N. Do, "Fast approximation of Kullback-Leibler distance for dependence trees and hidden Markov models," *Signal Processing Letters, IEEE*, vol. 10, no. 4, pp. 115–118, 2003.
[26] M. Falkhausen, H. Reininger, and D. Wolf, "Calculation of distance measures between hidden markov models," in *EUROSPEECH-1995*, 1995, pp. 1487–1490.
[27] P. Tiňo and I. T. Nabney, "Hierarchical GTM: Constructing localized nonlinear projection manifolds in a principled way," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 24, no. 5, pp. 639–656, 2002.