Bayesian Mixtures of Hidden Tree Markov Models for Structured Data Clustering

Davide Bacciu[a], Daniele Castellana[a]

[a] *Dipartimento di Informatica, Università di Pisa, Italy.*

Abstract

The paper deals with the problem of unsupervised learning with structured data, proposing a mixture model approach to cluster tree samples. First, we discuss how to use the Switching-Parent Hidden Tree Markov Model, a compositional model for learning tree distributions, to define a finite mixture model where the number of components is fixed by a hyperparameter. Then, we show how to relax such an assumption by introducing a Bayesian non-parametric mixture model where the number of necessary hidden tree components is learned from data. Experimental validation on synthetic and real datasets show the benefit of mixture models over simple hidden tree models in clustering applications. Further, we provide a characterization of the behaviour of the two mixture models for different choices of their hyperparameters.

Keywords: Hidden Tree Markov Models, Infinite Mixtures, Dirichlet Process, Tree structured data

1. Introduction

Tree structures are used in multiple contexts to represent hierarchically-organized information. For example, in biology, phylogenetic trees are used to show the evolutionary relationships among various biological species or other entities. In natural language processing, parse trees are used to represent the syntactic structure of sentences. On the web, most of the data (e.g. HTML and XML documents) are represented using the Document Object Model i.e. a tree where each node represents a part of the document. Regardless of the application domain, a tree is composed of atomic entities (i.e. the information attached to the nodes) combined together through the hierarchical relationship encoded by its structure. Hence, dealing with this type of data requires the ability to manage the atomic information along with the contextual information (e.g. the surrounding entities) given by the structure.

Email addresses: bacciu@di.unipi.it (Davide Bacciu), daniele.castellana@di.unipi.it (Daniele Castellana)

Early works on the adaptive processing of tree-structured data date back to the early nineties, mostly focusing on the recursive processing framework consolidated in the seminal work in [1]. Later, there has been a flourishing of works on adaptive tree data processing within different machine learning paradigms. Probabilistic models have been one of the first to be applied to tree data, thanks to an extension of the Hidden Markov Model for sequences to deal with all the root to leaves paths in a tree. This model is referred to as Top-Down Hidden Tree Markov Model (TD-HTMM), where the top-down term denotes the direction of tree visit and generation. The model has been introduced almost coincidentally in the context of documental data processing [2] and for statistical signal processing in the wavelet domain [3]. The Bottom-Up HTMM (BU-HTMM), on the other hand, models a recursive hidden process from the leaves to the root. Here, the tree is modelled as a set of independent processes (i.e. the leaves) which are merging and synchronizing at each level until a single process is obtained at the root. Note that in such a generative process the hidden state of a node depends on the joint hidden state of its children, with clear consequences in terms of combinatorial explosion of the transition distribution for large hidden state sizes. The first practical BU-HTMM has been introduced in [4], where it is proposed an approximation of the state-transition distribution using a mixture model, in a so-called Switching Parent fashion. The model has also been extended to process isomorph structure-to-structure transductions [5].

Kernel methods have also been widely applied to tree-structured data since they allow a straightforward reuse of kernel-based learning machinery for vectorial data by plugging in an appropriately defined tree kernel. There has been a large body of research dealing with the definition of efficient and discriminative tree kernels, including syntactic kernels [6] computing tree similarity by counting the number of common substructures (e.g. subtrees, paths, etc). Syntactic kernels are mostly based on a predefined and hand engineered metric, e.g. path similarity. Hence the resulting structural distance is not really adaptive, while only the classifier machinery built around the kernel is. To surpass this limitation, some authors have proposed building adaptive tree kernels on the top of either neural models [7] or generative tree models [8] such as the HTMM.

Neural network models for tree-structured data have appeared early since the definition of the general framework in [1]. Recently, they have found renewed interest thanks to the deep learning wave, which has led to a widespread use of Long Short-Term Memory (LSTM) units also in the tree-structured domain. The TreeLSTM model in [9] has been the first extension of the LSTM cell to handle tree-structures through a bottom-up approach, which has recently been extended to top-down processing in a transduction learning scenario [10]. An alternative approach is that put forward in the Tree Echo State Network (TreeESN) [11] where the recursive neurons are randomly initialized according to some dynamic system stability criterion and their weights are not adjusted by the training procedure. Recently, the Hidden Tree Markov Networks (HTNs) [12] have been proposed as a hybrid approach integrating probabilistic bottom-up models within a neural architecture and learning scheme.

The large body of research discussed above almost uniquely deals with adap-

tive tree-structured data processing from a supervised learning point of view, whose objective is to build a tree classifier or regressor based on some available ground truth labelling. Applications to unsupervised learning are, on the other hand, more limited. A notable exception is the seminal paper on a general framework for the unsupervised processing of structured data [13]. Within this class of models, the most relevant contributions are related to the extension of topographic mapping models to handle tree data. This is the case, for instance, of the SOM-SD model [14], extending Kohonen's self-organizing maps to structured acyclic data (i.e. including trees as a special case). Extensions of generative topographic mapping to structured data have instead been proposed by [15] and [16], based on top-down and bottom-up approaches, respectively. Despite their unsupervised nature, such models have seldom be used for structured data clustering. The point is that these models are fundamentally solving a different problem than clustering, that is finding a topographic mapping that preserves structural similarities. When applied to clustering, e.g. in [17], the partitioning of samples into clusters is obtained only as a post-processing step, through expert inspection of the projects obtained on the topographic map. In this work, on the other hand, we investigate a model that is fundamentally designed for clustering; in particular, we consider Bayesian non-parametric approaches to relax the necessity of a human overlooking the clustering results (e.g. to determine the number of clusters in the structured population).

The goal of this paper is to introduce a mixture model approach to address the tree clustering problem. Mixture models are generative approaches widely applied in clustering applications for vectorial data, e.g. consider the Gaussian mixture model and its evolutions. Here, we propose a mixture model built on the top of the bottom-up HTMM. The choice of a BU approach as mixture component is driven by the necessity of extracting and representing in the latent state space the maximal amount of structural information from the samples. Earlier works [4] have already shown the superior effectiveness of BU approaches over TD in this respect. In the following, we start by defining a first finite mixture model, where the number of HTMM components is fixed by a hyperparameter. Then we extend the model by introducing the possibility of learning the number of HTMM components directly from the data, by taking a Bayesian approach based on Dirichlet processes [18]. These allow to define a potentially infinite number of mixture components: we will then show how, in practice, this allows to automatically extract a finite number of relevant components to describe clusters in the data. This paper is an extended version of the conference paper [19]: this earlier work only introduced the finite mixture model and provided only preliminary results on a reduced set of data. The current work extends the original conference publication by introducing a completely novel model, that is the infinite mixture approach, together with a completely renewed experimental validation.

The remainder of the paper is organized as follows: in Section 2 we introduce useful definitions and the notation used throughout the paper. In Section 3 we summarize the results obtained in [4] which we use as starting point for our work. In Section 4 we define the finite mixture of BU-HTMM and we derive

its learning algorithm, while in Section 5 we extend this in a non-parametric fashion, defining a new approximated learning procedure. Finally, Section 6 provides the experimental assessment and in Section 7 we draw our conclusions.

2. Definition and Notation

A rooted tree $\mathbf{x}^n$ is a connected acyclic graph consisting of a set of nodes $\mathcal{U}^n = \{1, \dots, U^n\}$, where the root is the node with index 1. The term n is used here to denote the n-th tree in a dataset $\mathcal{D} = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$, where N is the size of the dataset. For the sake of clarity, this index will be omitted when its use is clear by the context.

A rooted tree defines parent-child relations between its nodes (see example in fig. 1). Let $u, v \in \mathcal{U}^n$: by definition of rooted tree, each node has one parent and we use the relation $u = pa(v)$ to indicate the node u is the parent of the node v. Two nodes are siblings if they share the same parent (i.e. $pa(u) = pa(v)$).

In this paper we consider finite trees: the letter L indicates the maximum output degree of each node (i.e. the maximum number of children). The position of a node with respect to its siblings is indicated by $l = pos(u)$; therefore, $v = ch_l(u)$ indicates the node v is the l-th child of u. The nodes that do not have children are called leaves: we indicate with $\mathcal{LF}^n \subset \mathcal{U}^n$ the set of leaves' indexes.

For the purpose of our paper, we assume that a discrete label is associated to each node: x_u^n is the label related to the node u in the tree $\mathbf{x}^n$.

A rooted tree $\mathbf{x}^n$ can be decomposed in substructures, which consist of a set of nodes and the corresponding edges. We use the term $\mathbf{x}_u^n$ to denote the subtree rooted in u. Similarly, $\mathbf{x}_{1\setminus u}^n$ denotes the whole tree $\mathbf{x}^n$ without the subtree $\mathbf{x}_u^n$.

3. The Switching-Parent Bottom-up Hidden Tree Markov Model

In this section, we provide a summary of the Bottom-Up Hidden Tree Markov Models for labelled trees, introduced by [4], which is used as a building block for the following mixtures. The model is formulated in terms of a hidden Markov model, introducing an approximation of the transition function to avoid a combinatorial explosion of the parameter space. The training procedure is based on the Expectation-Maximisation and it is outlined in Section 3.2

3.1. Model Definition

The Switching-Parent Bottom-up Hidden Tree Markov Model (SP-BHTMM) [4] defines an approximated generative process for a tree $\mathbf{x}$, which goes from the leaves to the root. As in standard HMM, the whole process is split in a hidden and a visible part. The hidden dynamic regulates interactions among hidden states, while the visible one controls the emission of visible labels.

Given a labelled tree $\mathbf{x}$, we build the graphical model of BHTMM associating a hidden random variable $Q_u \in [1, C]$ to each label $x_u \in [1, M]$ in the tree. All the hidden variables are linked together reproducing the same structure of the

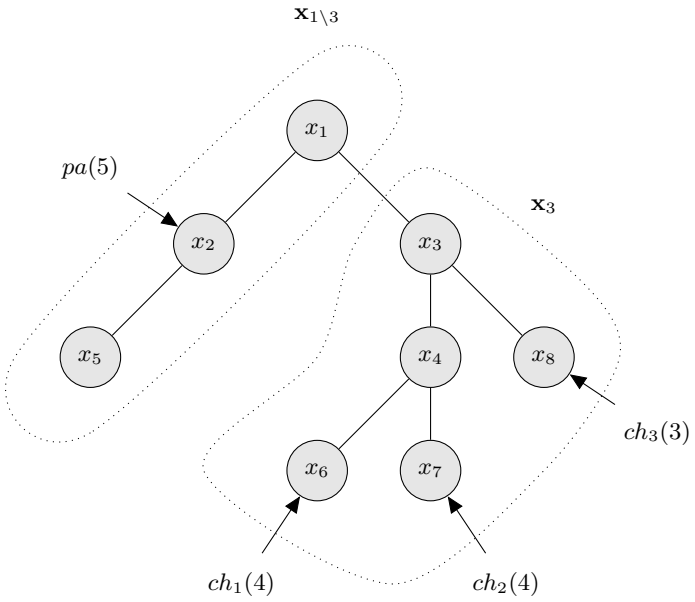


Figure 1: Example of labelled tree with $L = 3$.

visible tree $\mathbf{x}$; the direction of links goes from leaves to the root, assuming the hidden state of a node depends on the joint configuration of its hidden child nodes. The computation of this *state-transition* distribution is impractical, since it grows exponentially with respect to the maximum output degree L. The SP-BHTMM factorises such joint state distribution as a mixture of pairwise child-to-parent transitions: this approximation is called Switching Parents (SP) [4]. Also, SP-BHTMM assumes the hidden state Q_u contains all necessary information to generate the visible label x_u associated.

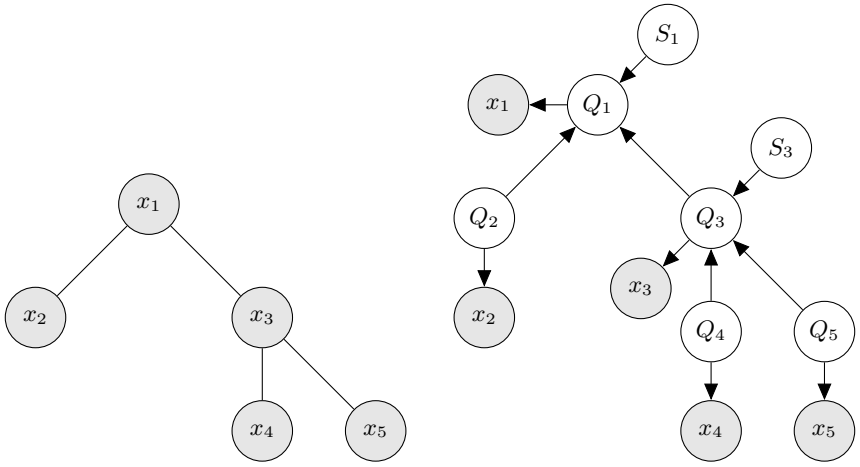
Using the conditional independence assumptions introduced by the SP-BHTMM, we can derive the complete likelihood for a given tree $\mathbf{x}$:

$$\mathcal{L}(\mathbf{x}, \mathbf{Q} \mid \theta) = P(\mathbf{x}, \mathbf{Q} \mid \theta) = \prod_{u \in \mathcal{LF}} \pi_j^l \, b_j(x_u) \times \prod_{v \in \mathcal{U} \setminus \mathcal{LF}} \sum_{l=1}^{L} \phi_l \, A_{i,j}^l \, b_i(x_v) \qquad (1)$$

where $\theta = \{\pi, b, \phi, A\}$ represents all SP-BHTMM model parameters. The likelihood of visible data $P(\mathbf{x} \mid \theta)$ can be obtained summing (1) over the hidden variables $\mathbf{Q}$.

At this point, it is worth spending a few lines to describe each SP-BHTMM parameter in detail. First of all, we should notice that all model parameters are categorical distributions, since both hidden variables and visible labels are finite discrete random variables. Extension to continuous labels is trivial, along with the lines of [4].

The term π denotes the priori distribution, which is defined on leaf hidden



(a) Observed tree $\mathbf{x}$ (b) GM for SP-BHTMM

Figure 2: Graphical model (GM) for SP-BHTMM associated to an observed tree $\mathbf{x}$. The variables $\mathbf{S}$ represent the switching parent variables.

nodes. Since we are dealing with positional trees, the priori distribution depends on the position of the leaf node. Let $u \in \mathcal{LF}$, it holds $P^l(Q_u = i) = \pi_i^l$; the term $l = pos(u)$ indicates the position of the node u. Consequently, the term π is a $C \times L$ matrix.

The term b denotes the emission distribution, which generates the visible labels. The generation of label x_u depends on the state of its hidden variable associated Q_u; therefore, it holds $P(x_u \mid Q_u = i) = b_i(x_u)$. The term b is a $M \times C$ matrix.

The last two terms ϕ and A are related to the state-transition distribution. In particular, $A^l_{i,j} = P^l(Q_v = i \mid Q_{ch_l(v)} = j)$ indicates the dependency between a node and its l-th child while $\phi_l = P(S_v = l)$ is the switching parents distribution and it measures the weight of the contribution of the l-th child to the state transition of node v. The term ϕ is a vector with L elements while A is a $C \times C \times L$ matrix.

3.2. Learning in a SP-BHTMM

Inferring SP-BHTMM parameters from data is achieved trough an Expectation Maximisation approach.

The goal of the Expectation step is to compute the posterior distribution of the hidden variables given the visible ones. The *upward-downward* is an algorithm which computes the posterior by exploiting a recursive factorisation [4]: such factorisation requires an initial *upward* pass and a final *downward* pass on the tree, hence the algorithm name.

The *upward* pass is a recursive procedure over the tree structure, which goes from leaves to the root: the aim is to compute the value $P(Q_u \mid \mathbf{x}_u)$ for

each node u. Conversely, the *downward* pass goes from the root to leaves and computes the posterior $P(Q_u, Q_{ch_l}, S_u = l \mid \mathbf{x})$ for each node u.

The Maximisation step updates the model parameters in order to maximise the expectation of the complete likelihood with respect to the posterior computed in the E-step.

We have voluntarily omitted details and derivations of the learning procedure, which can be found in [4].

4. Mixture of SP-BHTMM

A finite mixture model is able to approximate complex distributions through an appropriate choice of its components to represent the local area of the truth distribution [20]. In this section, we introduce a finite mixture model whose components are SP-BHTMM in order to better represents complex distributions over labelled trees. The number of components is finite and it is an hyper-parameter of the model.

4.1. Model Definition

A finite mixture model is obtained by combining together multiple generative models, which are called *mixture components*. The combination is obtained through a hidden random variable, called *mixture variable*.

Since we are introducing a finite mixture model, the number of components is fixed and it is represented by the hyper-parameter T. In our model, all components are SP-BHTMM, each of them with different parameters $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_T\}$. To better understand how the mixture of SP-BHTMM (MIX-SP-BHTMM) represents the data, it is useful to summarise the underlying generative process for a tree $\mathbf{x}^n$:

$$\begin{aligned} \mathbf{x}^n \mid c_n, \boldsymbol{\theta} &\sim P(\mathbf{x}^n \mid \theta_{c_n}) \\ c_n \mid \mathbf{p} &\sim \text{Discrete}(p_1, \ldots, p_T). \end{aligned} \tag{2}$$

The term c_n indicates the latent class associated to the observed tree $\mathbf{x}_n$, i.e. the index of the component used to generate it. Hence, θ_{c_n} represents the model parameters of the c_n-th mixture component. The value $P(\mathbf{x}^n \mid \theta_{c_n})$ is the likelihood of tree $\mathbf{x}^n$ according to the c_n component (see equation 1). The latent class is drawn from a discrete distribution, which is the distribution of the mixture variable. In Fig. 3, we represent the graphical model which describes this process: for the sake of clarity, the whole tree $\mathbf{x}^n$ is indicated as a single variable.

4.2. Learning in a MIX-SP-BHTMM

Learning MIX-SP-BHTMM parameters has two objectives: the first one is to learn the parameters of the mixture components $\boldsymbol{\theta}$; the second one is to learn the mixing distribution $\mathbf{p}$. In Section 3.2 we have shown how SP-BHTMM parameters can be learned through a specialisation of the EM algorithm. Moreover, the EM algorithm is widely used to estimate the mixing distribution in

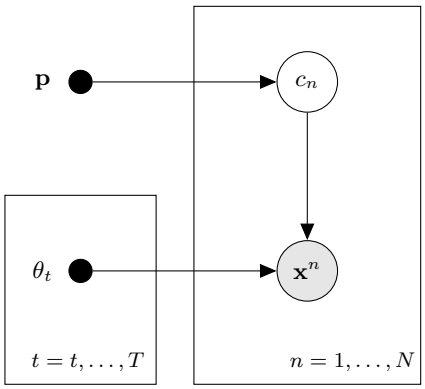


Figure 3: Graphical model for the MIX-SP-BHTMM: empty and shaded nodes denote hidden and observable variables, respectively. Black-point nodes identify model parameters.

finite mixture models [20]. Therefore, we can derive a single EM specialisation which is able to learn all MIX-SP-BHTMM parameters.

The goal of the Expectation phase is to compute the posterior of all hidden variables in the model given visible ones. First of all, we should observe that two mixture component are completely independent given the latent class: the only way to exchange information among components is through the latent class. Hence, each conditional independence assumption made to derive the *upward-downward* algorithm still holds in our model: we can use *upward-downward* algorithm to derive the posterior of hidden variables in each SP-BHTMM component. Combining together the posterior computed for each component, we obtain the posterior $P(\mathbf{Q}^n \mid \mathbf{x}^n, c_n = t, \theta_t)$ where the conditioning over the latent class c_n is explicitly introduced. Unfortunately, we cannot use directly this value since it depends on a hidden variable, i.e. the latent class. However, by applying the chain rule, we obtain:

$$P(\mathbf{Q}^n, c_n = t, \mid \mathbf{x}^n) = P(\mathbf{Q}^n \mid c_n = t, \mathbf{x}^n) P(c_n = t \mid \mathbf{x}^n) \tag{3}$$

where we omit the parameter θ_t since it is implicit in the latent class.

The term $P(c^n = t \mid \mathbf{x}^n)$ represents the posterior of the latent class, which can be easily rewritten as

$$P(c_n = t \mid \mathbf{x}^n) = \frac{P(\mathbf{x}^n \mid c_n = t) P(c_n = t)}{P(\mathbf{x}^n)} \tag{4}$$

which completes the E-step definition, summarised in Algorithm 1.

The M-step updates component parameters $\boldsymbol{\theta}$: it is derived by straightforward application of the formula used for a single SP-BHTMM to the new posterior computed in eq. (3). An additional rule to update the latent class

distribution $\mathbf{p}$ is also needed

$$p_t = \frac{\sum_{n=1}^{N} P(c_n = t \mid \mathbf{x}^n)}{N}. \tag{5}$$

From the computational complexity point of view, the introduction of the mixture increases the computational complexity in time to $O(T \times C_{\text{up-down}})$, where $C_{\text{up-down}}$ is the time complexity of the $upward - downward$ algorithm. The computational complexity in space has the same behaviour: it becomes $O(T \times C_{\text{SP-BHTMM}} + T)$, where $C_{\text{SP-BHTMM}}$ is the space required to store a SP-BHTMM model. The last term T is the space required to store the mixing distribution, which can be neglected.

Algorithm 1 E-step for MIX-SP-BHTMM

Require: A labelled tree $\mathbf{x}^n$, T different SP-BHTMM with parameters $\theta_1 \dots \theta_T$ and a mixture distribution $\mathbf{p}$.
  for t=1 **to** T **do**
    $postQ[t] = \text{UP-DOWN}(\mathbf{x^n}, \theta_t)$
    $lk[t] = \text{LIKELIHOOD}(\mathbf{x^n}, \theta_t)$
    $postP[t] = lk[t] \times p_t$
  end for
  $postP = NORMALISE(postP)$
  for t=1 **to** T **do**
    $postQ[t] = postQ[t] \times postP[t]$
  end for
  return $(postQ, postP)$

5. Infinite MIX-SP-BHTMM

Setting the correct number of components in a finite mixture model is not obvious and a variety of techniques have been developed [20]. In this section we build an infinite mixture of SP-BHTMM (INF-SP-BHTMM), which allows an infinite number of mixture components: in our case, each component is, again, an SP-BHTMM with different parameters. Due to the infinite number of components, the learning procedure requires an approximation, which is discussed in Section 5.2.

5.1. Model Definition

An infinite mixture model is a Bayesian non-parametric extension of a finite mixture model and it typically relies on the use of Dirichlet Processes (DP) [21]. The corresponding generative models can be described as follows [18]

$$\begin{aligned} \mathbf{x}^n \mid \zeta_n &\sim F(\zeta_n) \\ \zeta_n \mid G &\sim G \\ G &\sim DP(G_0, \gamma). \end{aligned} \tag{6}$$

The distribution $F(\zeta_n)$ represents the mixture with mixing distribution ζ_n drawn from G, which is itself distributed according to a DP with concentration parameter γ and base measure G_0. The value G_0 is the expected values of the DP and it represents the priori distribution for the mixture component parameters. For the sake of simplicity, we have ignored the dependency between the function F and the mixture component parameters and the hyper-parameters for the priori G_0. These will be stated more in detail in the remainder of the section.

For our purposes, it is convenient to derive the infinite model taking the limit as T goes to infinity of a MIX-SP-BHTMM with T component [18]. Before taking the limit, we define explicitly the prior probability of MIX-SP-BHTMM parameters (i.e. the function G_0). Since all the model parameters are multinomial, we can use its conjugate prior, i.e. the Dirichlet distribution. By the addition of the prior, we obtain the following infinite MIX-SP-BHTMM model

$$\begin{aligned}
\mathbf{x}^n \mid c_n, \boldsymbol{\theta} &\sim P(\mathbf{x}^n \mid \theta_{c_n}) \\
c_n \mid \mathbf{p} &\sim \text{Discrete}(p_1, \ldots, p_T) \\
\pi &\sim \text{Dirichlet}(\alpha_\pi, \ldots, \alpha_\pi) \\
A &\sim \text{Dirichlet}(\alpha_A, \ldots, \alpha_A) \\
b &\sim \text{Dirichlet}(\alpha_b, \ldots, \alpha_b) \\
\phi &\sim \text{Dirichlet}(\alpha_\phi, \ldots, \alpha_\phi) \\
\mathbf{p} &\sim \text{Dirichlet}(\gamma/T, \ldots, \gamma/T).
\end{aligned} \tag{7}$$

For the sake of clarity, we omit the fact that conditional distributions (such as A and b) are obtained by sampling a Dirichlet distribution multiple times.

Since we are using a flat Dirichlet distribution, we have a single hyper-parameter for each prior distribution. Hence, the model hyper-parameters are $\{\alpha_\pi, \alpha_A, \alpha_b, \alpha_\phi, \gamma\}$: the α terms are related to the SP-BHTMM priors (i.e. are parameters of G_0) while the γ term is the concentration parameter of the Dirichlet Process.

5.2. Learning in INF-SP-BHTMM

Computing the exact posterior expectation becomes unfeasible when the model is extended with a DP prior. However, such expectation can be estimated using Monte Carlo methods [18]. A Gibbs sampling algorithm can be applied to the model described in (7), integrating out the mixing proportions $\mathbf{p}$. The idea is to iteratively sample the latent class c for each data point and update the parameters θ for each mixture component, taking into account only those data points assigned to each mixture. Even if there is an infinite number of components, we are able to execute this algorithm since we deal only with mixture components that are currently associated with some observations and, by definition, there is only a finite number of these.

The first step is to assign a latent class to each tree $\mathbf{x}^i$. The Gibbs sampler update requires sampling the latent class of a tree $\mathbf{x}^i$ given the latent class of all other trees. Obviously, the sampling rule depends also on the tree $\mathbf{x}^i$ itself and

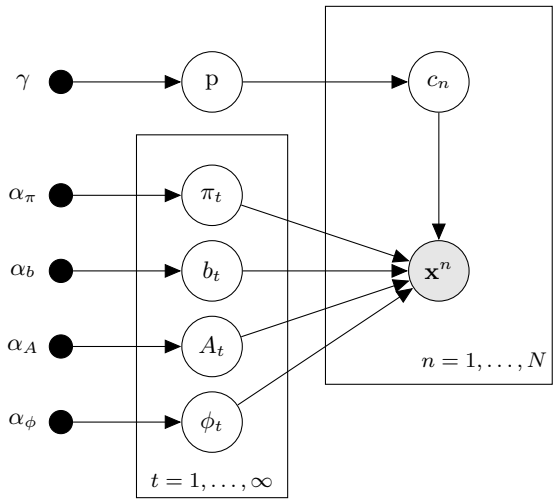


Figure 4: Graphical model for the INF-SP-BHTMM.

all mixture component parameters $\boldsymbol{\theta}$. The conditional probability from which to sample is [18]:

$$P(c_i = c \mid c_{-i}, \mathbf{x}^i, \boldsymbol{\theta}) = \begin{cases} \frac{n_{-i,c}}{Z} P(\mathbf{x}^i \mid \theta_c) & \text{if } \exists\, j \neq i \mid c_j = c \\ \frac{\gamma}{Z} \int P(\mathbf{x}^i \mid \theta) dG_0(\theta) & \text{otherwise} \end{cases} \tag{8}$$

where $n_{-i,c}$ is the number of trees (except $\mathbf{x}^i$) which are already assigned to the c-th class. The value c_{-i} indicates the latent class of all trees in the dataset except $\mathbf{x}^i$, while Z is a normalising constant to ensure that the above probability sum to one.

The equation (8) states that the probability to assign a class c to a tree is proportional to the number of trees that are already assigned to it (i.e. $n_{-i,c}$). Nevertheless, there is a non-zero probability to assign the i-th tree to a new component: unfortunately, we can not consider explicitly all the other components since there an infinite number of them. The solution is to integrate over all the possible mixture component parameters (i.e. all the possible mixture components). The integral is taken over the function $G_0(\theta)$ since it represents the prior for SP-BHTMM parameters. The integral can be solved analytically due to the conjugacy between parameter distributions and their prior: the result is the likelihood of $\mathbf{x}^n$ according to a SP-BHTMM whose parameters are uniform distributions since each prior is a flat Dirichlet distribution. When a new class is sampled, we must create a new mixture component. The new parameters are sampled from the prior distribution $G_0(\theta)$. During the inference procedure, it can also happen that a latent class is no longer assigned to any

trees. From equation (8), it follows there is a null probability to assign such class again. Hence, we can remove the corresponding latent class.

The second step of the inference procedure requires to estimate new parameters θ for all mixture components. Obviously, each component updates its parameters to adapt itself to trees that are assigned to it during the first step. The updates can be performed by applying the procedure summarised in Section 3.2 on the subset of the dataset assigned to each component. The only modification required is in the M-step, which is extended to consider also the prior. Since we choose a conjugate prior, this reduces to add the value $\alpha - 1$ to each counting table. The whole Gibbs sampling method is summarised in algorithm 2.

Algorithm 2 Gibbs sampling method for INF-SP-BHTMM

Require: A dataset of labelled tree $\mathcal{D} = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$, a set of SP-BHTMM parameters $\boldsymbol{\theta} = \{\theta_1 \ldots \theta_T\}$, a random assignment $\mathbf{c} = \{c_i, \ldots, c_N\}$
$S_t = \{n \mid c_n = t\} \quad \forall t \in [1, T]$
repeat
  for $n = 1$ **to** N **do** ▷ Sample step
    $S_{c_n} = S_{c_n} \setminus \{n\}$
    if $S_{c_n} = \emptyset$ **then** ▷ Remove c_n
      $\boldsymbol{\theta} = \boldsymbol{\theta} \setminus \{\theta_{c_n}\}$
      $\mathbf{S} = \mathbf{S} \setminus \{S_{c_n}\}$
      $T = T - 1$
    end if
    $c_n = \text{SAMPLING}(c_{-i}, \mathbf{x}^m, \boldsymbol{\theta})$ ▷ eq. (8)
    if c_n is new **then** ▷ Create c_n
      $\theta_{\text{new}} \sim G_0$
      $\boldsymbol{\theta} = \boldsymbol{\theta} \cup \{\theta_{\text{new}}\}$
      $T = T + 1$
      $S_{c_n} = \emptyset$
    end if
    $S_{c_n} = S_{c_n} \cup \{n\}$
  end for
  for t=1 **to** T **do** ▷ Update step
    $\theta_t = \text{EM-SP-BHTMM}(\theta_t, S_t, G_0)$
  end for
until stopping criteria

Again the computational complexity (both in time and space) increases linearly with respect to the number of components T, when compared to the simple SP-BHTMM model.

6. Experimental results

In this section, we provide an experimental validation of the proposed approaches. In particular, we are interested in empirically assessing the ability to recognise clusters in tree-structured data. Evaluating the clustering quality is not trivial and multiple indexes have been defined [22]. In the following experiments, we use the Silhouette index [23] to asses the clustering quality. The Silhouette index is an *internal* measure and therefore it can be computed without any additional knowledge on data (e.g. the true clustering). However, its computation requires the definition of a suitable distance metric among data points. Here we compute the distance between two trees using he Ruzicka distance [24] on their representative matrices, where a representative matrix R^n for a tree $\mathbf{x}^n$ is such that the its element r^n_{lj} counts how many times the label j appears in a node in the l-th position. Let R^n, R^m be two representative matrices for trees $\mathbf{x}^n$ and $\mathbf{x}^m$, respectively, the Ruzicka distance is defined as:

$$d_R(\mathbf{x}^n, \mathbf{x}^m) = 1 - \frac{\sum_l \sum_j \min(r^n_{lj}, r^m_{lj})}{\sum_l \sum_j \max(r^n_{lj}, r^m_{lj})}. \tag{9}$$

For a given tree $\mathbf{x}^n$, the Silhouette index is computed considering the distance between $\mathbf{x}^n$ and both elements that are inside and outside its cluster; its value is always between -1 (worst clustering) and 1 (best clustering). Mathematically speaking, let $\mathbf{x}^n$ an element in the cluster C, the Silhouette index is given by:

$$s(\mathbf{x}^n) = \frac{b(\mathbf{x}^n) - a(\mathbf{x}^n)}{\max(a(\mathbf{x}^n), b(\mathbf{x}^n))}, \tag{10}$$

where $a(\mathbf{x}^n)$ is the average distance of $\mathbf{x}^n$ and its cluster C, while $b(\mathbf{x}^n)$ is the minimum distance between $\mathbf{x}^n$ and a cluster $C' \neq C$. The distance between a tree $\mathbf{x}^n$ and a cluster C is computed as the average distance between $\mathbf{x}^n$ and all trees $\mathbf{x}^m \in C$. In formula, assuming $\mathbf{x}^n \in C$, we obtain:

$$a(\mathbf{x}^n) = \frac{1}{|C|} \sum_{\mathbf{x}^m \in C} d_R(\mathbf{x}^n, \mathbf{x}^m) \qquad b(\mathbf{x}^n) = \min_{C' \neq C} \frac{1}{|C'|} \sum_{\mathbf{x}^{m'} \in C'} d_R(\mathbf{x}^n, \mathbf{x}^{m'}) \tag{11}$$

where $d_R(.,.)$ is the Ruzicka distance defined above.

In Section 6.1 and Section 6.2, we report the results obtained on two clustering tasks; the first is a controlled dataset while the latter deals with real-world data. In Section 6.3 we further investigate the results of the second experiments to highlight the impact of the INF-SP-BHTMM hyper-parameters.

All the experiments have been performed on 2.20GHz Intel® based CPUs. The MATLAB code implementing the models can be found on a public GitLab repository[1].

[1] `http://gitlab.itc.unipi.it/d.castellana/Mixtures_SP_BHTMM`

6.1. Synthetic dataset

The goal of the first experiment is to assess whether the mixture of hidden trees (both finite and infinite) offers an advantage with respect to a single SP-BHTMM in terms of cluster identification. To this end, we test all models (SP-BHTMM, MIX-SP-BHTMM, and INF-SP-BHTMM) on a synthetic clustering problem. The dataset contains ternary trees (i.e. L=3), comprising left-asymmetric, symmetric and right-asymmetric tree, hence defining three clusters. A tree is defined as left-asymmetric (right-asymmetric) if the number of nodes in the leftmost (rightmost) position is greater than the number of nodes in the opposite position. In a symmetric tree, the number of nodes is almost equivalent for each position.

A tree generator has been realised to generate the dataset through a top-down recursive procedure: starting from the root, child nodes are generated according to a distribution which indicates how likely is to generate a node in each position. If new nodes are generated, the same procedure is recursively applied in order to generate the whole tree. The procedure ends when a maximum number of nodes have been generated. This scheme is used to generate all three different types of tree: for each type, a proper distribution to generate child nodes is used. The label of each node encodes structural information since it represents the number of children of the node: therefore the label goes from 0 (i.e. no child nodes) to 3 (i.e. a child node in each position). Moreover, each tree types is generated by setting a different maximum number of nodes in order to add another structural peculiarity. In particular, left-asymmetric trees are the smallest one, while the right-asymmetric are the biggest ones. Symmetric trees have a number of nodes which is between the characteristic sizes of left and right imbalanced trees. Finally, we generate 780 trees (260 for each type) and split them in training set (600 trees, 200 for each type) and test set (180 trees, 60 for each type).

Silhouette index on synthetic dataset			
SP-BHTMM	$C = 3$	$C = 5$	$C = 7$
Root sampling	**0.03** (0.00)	−0.02 (0.03)	−0.08 (0.02)
MIX-SP-BHTMM	$T = 3$	$T = 5$	$T = 7$
$C = 2$	0.41 (0.01)	0.43 (0.03)	0.46 (0.05)
$C = 4$	0.45 (0.05)	**0.47** (0.08)	**0.47** (0.05)
$C = 6$	0.46 (0.05)	0.45 (0.05)	**0.47** (0.06)
INF-SP-BHTMM	$\alpha = 1$	$\alpha = 1.5$	$\alpha = 2$
$C = 2$	0.36 (0.23)	0.45 (0.08)	0.45 (0.07)
$C = 4$	0.43 (0.08)	**0.51** (0.00)	**0.51** (0.00)
$C = 8$	0.33 (0.00)	**0.51** (0.00)	**0.51** (0.00)

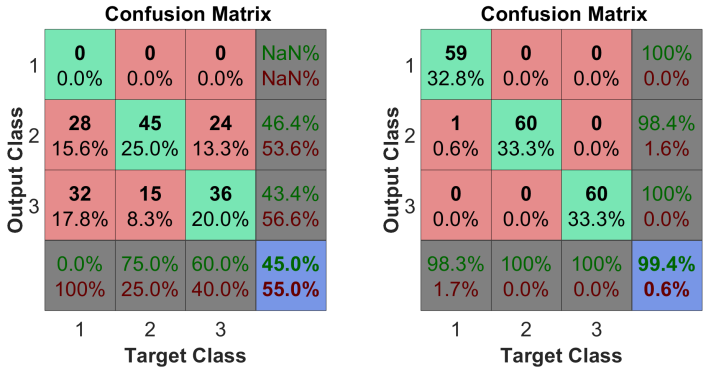
Table 1: Mean Silhouette index over 5 runs (std in brackets) on a synthetic dataset. In bold the best result for each model.

All models (SP-BHTMM, MIX-SP-BHTMM, and INF-SP-BHTMM) have

been trained in an unsupervised setting, i.e. sample classification information is not available at training time. For each model, different configurations have been trained, changing the number of hidden states (i.e. C), the number of mixtures (i.e. T), and the prior hyper-parameters. Thanks to a preliminary analysis, we have noticed that some hyper-parameters of INF-SP-BHTMM do not affect the solution too much. Therefore, to reduce the number of configurations to test, we have used the same value for all the prior hyper-parameters (i.e. α_π, α_b, α_A, α_ϕ): we refer to this value with the letter α. Also, we have fixed the concentration parameters $\gamma = 10$. For a fair comparison, each training algorithm has been executed for a maximum of 30 iterations.

At test time, the SP-BTHMM assign a class for each tree sampling the posterior of the root while the MIX-SP-BHTMM model samples the posterior of the mixture variable. The INF-SP-BHTMM model cannot directly sample from the posterior since this would be intractable; however, the Gibbs sampler (introduced in Section 5.2) can be used to approximate the latent class assignment (skipping the parameters optimisation step). During the test, we limit to 10 the number of iterations of the Gibbs sampler.

In Table 1 we report the mean and standard deviation (in brackets) of the Silhouette index for each configuration over 5 runs. The advantage obtained by introducing a mixture is clear: the single SP-BHTMM reaches the best performance of 0.03, which is far from the best one obtained from both MIX-SP-BHTMM and INF-SP-BHTMM. Instead, the performance obtained by both mixture models is closer to the Silhouette index computed on the ground truth, that is 0.51. In Figure 5 we report two confusion matrices, obtained using INF-SP-BHTMM and SP-BHTMM to show the benefits of mixture models.



(a) SP-BHTMM, $C = 3$.

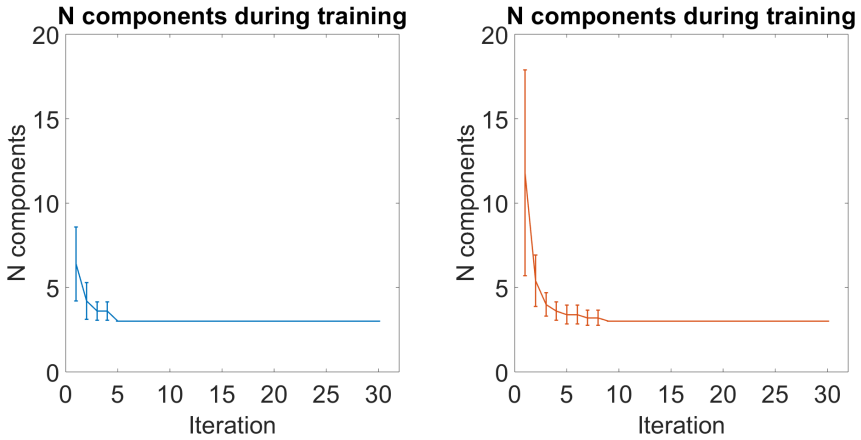
(b) INF-SP-BHTMM, $C = 8$ and $\alpha = 2$.

Figure 5: Confusion matrices for the synthetic dataset using INF-SP-BHTMM (a) and SP-BHTMM (b).

Even if the best results obtained by MIX-SP-BHTMM and INF-SP-BHTMM are similar, there are some key differences. First of all, we should notice that the infinite model reaches the best performance with zero standard deviation, i.e.

the model performed the same in each run. Also, MIX-SP-BHTMM performs better when there are more components than the real number of clusters: most of them are not used by the model. On the other hand, INF-SP-BHTMM is able to find the true number of clusters. In Figure 6, we plot the mean (and standard deviation) number of components during the training for two different configurations of INF-SP-BHTMM. In the first iterations, the model explores the solution space creating a high number of components (with different parameters); then, the model starts adapting the best components to the data, throwing away unused ones. After a few iterations, it reaches a total of 3 component which is the actual number of clusters. The plot also shows a different behaviour between the two configurations: this aspect is examined in depth in Section 6.3.

This behaviour affects also the time required for both training and testing. As stated in previous sections, the complexity in time for both mixture models depends linearly on the number of components: hence, unused mixture components slow down the inference procedure. In Figure 7, we show the training time required by both models on the synthetic dataset. For fairness, both training algorithms are executed without model-specific code optimization. The plot confirms our initial intuition: during the first iterations, INF-SP-BHTMM is slower than MIX-SP-BHTMM due to the high number of active components. However, after few iterations, INF-SP-BHTMM becomes faster than MIX-SP-BHTMM with as little as three components. According to the results in Table 1), MIX-SP-BHTMM and INF-SP-BHTMM perform better when the respective hyperparameters are set to $T = 7$ and $\alpha = 2$: comparing their training time in Figure 7, the benefit of the non-parametric extension is clear.



(a) $C = 8$ and $\alpha = 2$.

(b) $C = 4$ and $\alpha = 1.5$.

Figure 6: Number of active components during the training averaged, over 5 runs, for two different configurations of INF-SP-BHTMM on the synthetic dataset.

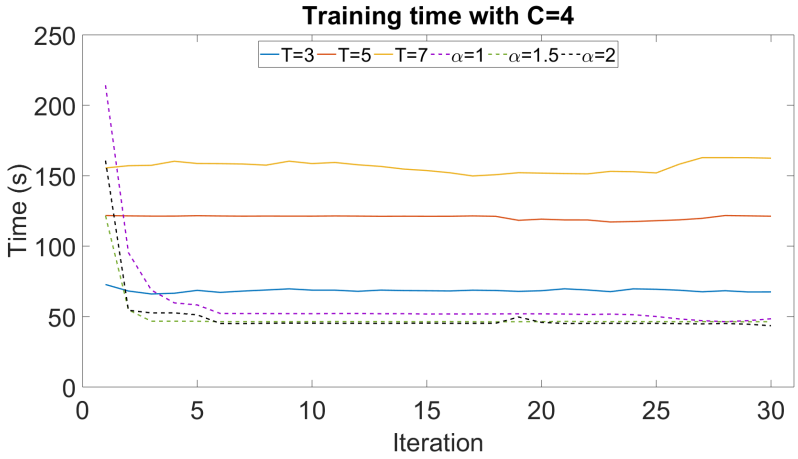


Figure 7: Time spent for a single training iteration by the finite and infinite model. Solid lines denote MIX-SP-BHTMM results for different T choices. Dashed lines refer to INF-SP-BHTMM under different choices for the α hyperparameter. For the sake of clarity, the number of hidden states is fixed to 4 in both models.

6.2. Real-world dataset

Previous experiments show the ability of both MIX-SP-BHTMT and INF-MIX-SP-BHTMT to cluster labelled trees in a completely unsupervised fashion. The goal of this experiment is to assess the clusterization performance of MIX-SP-BHTM and INF-SP-BHTMM on a real-world dataset. Due to the poor performance obtained in the previous experiment, we do not evaluate SP-BHTMM.

The dataset we have chosen is taken from the INEX 2005 competition [25]. It is based on the (m-db-s-0) corpus, comprising 9631 XML-formatted documents represented as trees with maximum output degree $L = 32$ and labelled by 11 thematic categories, which represents the different clusters. Node labels represent XML tags: there are 366 possible labels. The dataset is split in training set (4820 trees) and test set (4811 trees) [25].

Again, we have tested multiple configurations for each model. In particular, in MIX-SP-BHTMM we have varied the number of hidden states $C \in [2, 4, 8]$ and the number of mixture component $T \in [6, 11, 22]$. In INF-SP-BHTMM we have changed the number of hidden states $C \in [2, 4, 8]$ and the hyper-parameter of the SP-BHTMM prior $\alpha \in [1, 1.2, 1.5, 2]$. As in the previous experiment, we have fixed the value of the concentration parameter, i.e. $\gamma = 10$. Each configuration has been trained for a maximum of 30 iterations, while the INF-SP-BHTMM test procedure has been executed for a maximum of 10 iterations.

In Table 2, we report the mean and standard deviation (in brackets) of the Silhouette index computed, for each configuration, over 5 training-test runs. The advantage of the infinite model is not clear, even if it reaches the best performance on the INEX2005 dataset. Rather than comparing only the performance results, it is interesting to compare the resulting clustering produced

Silhouette index on INEX 2005 dataset				
MIX-SP-BHTMM	$T = 6$	$T = 11$	$T = 22$	
$C = 2$	0.12 (0.01)	0.13 (0.07)	**0.20** (0.04)	
$C = 4$	0.13 (0.09)	0.17 (0.02)	0.15 (0.02)	
$C = 8$	0.08 (0.00)	0.11 (0.05)	0.17 (0.06)	
INF-SP-BHTMM	$\alpha = 1$	$\alpha = 1.2$	$\alpha = 1.5$	$\alpha = 2$
$C = 2$	0.15 (0.02)	0.15 (0.05)	0.19 (0.02)	**0.21** (0.04)
$C = 4$	0.07 (0.04)	0.20 (0.04)	0.16 (0.07)	0.18 (0.03)
$C = 8$	0.05 (0.10)	0.15 (0.05)	0.13 (0.02)	0.15 (0.06)

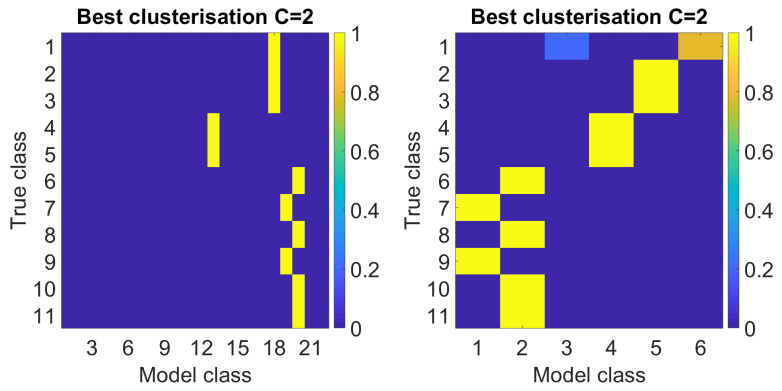
Table 2: Mean Silhouette index over 5 runs (std in brackets) on the INEX05 dataset. In bold the best result for each model.

by each model. In Figure 8, we report clusters obtained using the best configuration of both models. The plot shows how trees in each true class (on the y-axis) are distributed with respect the model-predicted clusters (on the x-axis). The clustering obtained using MIX-SP-BHTMM (Figure 8a) is made up of only 4 active clusters (even if there are 22 components): the first cluster contains all trees with ground-truth class labels $\{1, 2, 3\}$, the second cluster contains all trees with labels $\{4, 5\}$, the third cluster contains all trees with labels $\{6, 8, 9, 11\}$ and the last one contains all trees with labels $\{7, 9\}$. The clustering obtained using the INF-SP-BHTMM (fig. 8b) are almost the same, but there are two main differences. The first one is the number of clusters used, that is only 6 since the components with no data are thrown away during the training, thus reducing their impact on computational complexity. The second difference is that the model creates two new clusters to contain trees with ground-truth label 1: even if the model creates a spurious cluster, it is able to learn the difference between trees from category 1 and trees from all other categories.

The clustering produced by both models exploit the structural and label information contained in INEX2005 trees. In Figure 9 we report a similarity measure between categories in the INEX2005 training set. The similarity between two categories C_1 and C_2 is computed taking the mean of the Ruzicka similarity [24] between all C_1 trees and all C_2 trees. The plot shows clearly that categories with high similarity are the ones that are clustered together by our models. It is curious to observe that trees with label 6 are more similar to trees in class 11 than trees within the same class.

6.3. The importance of hyper-parameters

The experiments reported so far highlight how important is choosing the right value of hyper-parameters in order to obtain satisfactory results using both mixture models. In this section, we analyse the results obtained on INEX05 to emphasise the effects of each hyper-parameter. In particular, we study the effect of the hyper-parameters on the number of clusters discovered by the models. In Table 3 we report the mean and standard deviation of the number of clusters for each MIX-SP-BHTMM and INF-SP-BHTMM configuration over 5 runs.



(a) MIX-SP-BHTMM, $C = 2$ and $T = 22$.

(b) INF-SP-BHTMM, $C = 2$ and $\alpha = 2$.

Figure 8: Clustering obtained by MIX-SP-BHTMM (a) and INF-SP-BHTMM (b) using the best (model selected) configuration on the INEX05 dataset.

Number of active components				
MIX-SP-BHTMM	$T = 6$	$T = 11$	$T = 22$	
$C = 2$	1.60 (0.55)	2.00 (0.71)	3.40 (0.55)	
$C = 4$	2.00 (1.00)	2.20 (0.84)	1.80 (0.45)	
$C = 8$	1.20 (0.45)	2.00 (0.00)	2.80 (0.45)	
INF-SP-BHTMM	$\alpha = 1$	$\alpha = 1.2$	$\alpha = 1.5$	$\alpha = 2$
$C = 2$	44.20 (14.79)	33.00 (13.69)	8.80 (3.11)	4.60 (1.95)
$C = 4$	23.80 (26.36)	11.60 (4.83)	3.20 (1.30)	2.40 (1.67)
$C = 8$	45.80 (35.81)	5.80 (3.03)	2.40 (0.55)	1.80 (0.84)

Table 3: Mean number of non-empty clusters over 5 runs (std in brackets) on INEX05 dataset.

The MIX-SP-BHTMM is characterised by two hyper-parameters: the number of hidden states C and the number of mixture components T. By increasing the number of hidden states, we obtain more expressive SP-BHTMMs. Therefore, with higher values of C, the model tends to use fewer components since each component can be expressive enough to represent different clusters. The number of components T indicates how many SP-BHTMM components are used by the model. By taking a deeper look at the results in Table 2, it is clear that increasing the number of components helps to obtain better performances. However, even if a high number of components is selected, the number of clusters being identified is always small (see Table 3). We argue that increasing the value of T allows more exploration in the solution space: each component has a random configuration that can be suitable or not to describe the data. Creating more components, it is more likely to guess a better initialisation. In Figure 10a, we plot the average number of clusters over 5 runs for each MIX-SP-BHTMM configuration. From the plot is clear that higher values of T lead

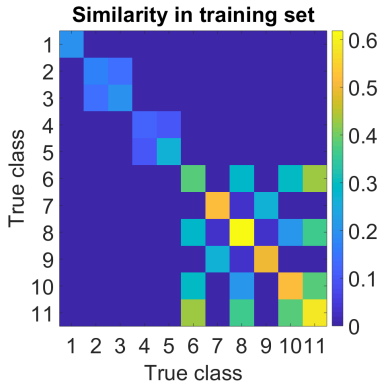
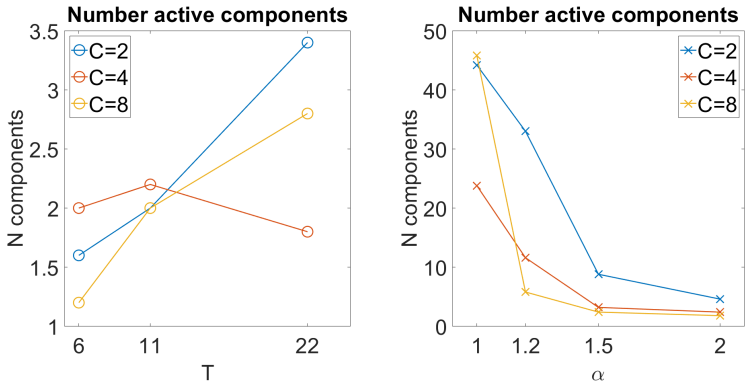


Figure 9: Ruzicka similarity between categories on the INEX2005 training set. Blue colours denote low similarity while yellow indicates an high similarity.

to higher numbers of active clusters. It is also visible the influence of C: the configuration with $C = 2$ has more active components than the configuration with $C = 8$.



(a) MIX-SP-BHTMM (b) INF-SP-BHTMM

Figure 10: Number of active components as a function of hyper-parameters for both mixture models on INEX05.

While the complexity of the INF-SP-BHTMM model is also controlled by the number of hidden states C, there is no hyper-parameter explicitly determining the number of mixture components. However, this is strictly correlated to choice of the α value. In fact, α determines how strong is our prior belief on the SP-BHTMM parameters: a stronger belief means that components will not adapt to the data too much (preventing over-fitting), while weaker beliefs lead to a completely data-driven solution. Hence, higher values of α tend to create solutions with fewer clusters, while a small value has the opposite effects. The value of the hyper-parameters C has the same influence described before on

MIX-SP-BHTMM. In Figure 10b, we plot the average number of active components for each INF-SP-BHTMM configuration, averaged on 5 runs. The effect of the choice of α is evident: the number of components reduces from more than 20 to around 5, independently of the value of C. The effect of the choice of C is also clear: the number of components obtained with $C = 2$ is greater than the one obtained with $C = 4$, which is greater than the one obtained with $C = 8$. Furthermore, the influence of C is evident when reporting the best clustering obtained for each C value (see Figure 11): selecting $C = 8$, all trees in the first five categories are merged together. On the other hand, by selecting $C = 2$, we do not have a SP-BHTMM expressive enough to represent trees in the first category: hence, the model uses two components to represent them.

(a) $C = 2$. (b) $C = 4$. (c) $C = 8$.

Figure 11: Best clusters obtained using INF-SP-BHTMM with different values of C on INEX05 dataset.

7. Conclusion

Learning models for tree-structured data have found application mostly to supervised tasks. Also, generative models, like SP-BHTMM, have been used mostly for such tasks, often within kernel-based frameworks for increased precision. This despite the fact that generative models traditionally find competitive applications in unsupervised/explorative analysis. In this work, we have first highlighted the limitations of SP-BHTMM in realizing unsupervised clustering analysis. Motivated by this, we have shown how to build on SP-BHTMM ability to learn structural patterns within a mixture model framework for clustering applications. Two different forms of mixtures of hidden tree models have been introduced. The first is a finite mixture (MIX-SP-BHTMM) which requires a fixed number of components to be supplied as hyper-parameters, while model parameters are learned by EM. The second is an infinite-mixtures model (INF-SP-BHTMM), addressing the problem of components specification by allowing an infinite number of components within the model. Despite the potentially infinite nature of the mixture, only a finite set of components is actually used during training, while the learning procedure can create (or remove) components

on the fly. Learning in the infinite model is not trivial and a Gibbs sampling method is required to approximate the intractable posterior.

The experiments have shown the benefit of mixture models in an unsupervised task. Even on controlled data, the SP-BHTMM was not able to perform an effective clustering. On the other hand, performances of both finite and infinite mixture models are nearly equivalent in both experiments (the INF-SP-BHTMM is slight better). The major advantage of the infinite model is its ability to learn the number of clusters directly from data. The experiment analysis has also been used to show how the behaviour of INF-SP-BHTMM depends on the configuration of its hyper-parameters. In particular, we have highlighted how the prior hyper-parameter plays a fundamental role to avoid the generation of single components for each sample in the dataset.

Further developments of this work can lead to a complete non-parametric model, such that the SP-BHTMM is able to use an infinite number of hidden states. Also, the inference procedure can be extended in order to adapt the hyper-parameters to the model, as stated in [18].

Acknowledgements

This work has been supported by the Italian Ministry of Education, University, and Research (MIUR) under project SIR 2014 LIST-IT (grant n. RBSI14STDE).

8. References

[1] P. Frasconi, M. Gori, A. Sperduti, A general framework for adaptive processing of data structures, IEEE transactions on Neural Networks 9 (1998) 768–786.

[2] M. Diligenti, P. Frasconi, M. Gori, Hidden tree markov models for document image classification, IEEE Trans. Pattern Anal. Mach. Intell. 25 (2003) 519–523.

[3] M. S. Crouse, R. D. Nowak, R. G. Baraniuk, Wavelet-based statistical signal processing using hidden markov models, IEEE Transactions on Signal Processing 46 (1998) 886–902.

[4] D. Bacciu, A. Micheli, A. Sperduti, Compositional generative mapping for tree-structured data; part i: Bottom-up probabilistic modeling of trees, Neural Networks and Learning Systems, IEEE Transactions on 23 (2012) 1987 –2002.

[5] D. Bacciu, A. Micheli, A. Sperduti, An input-output hidden Markov model for tree transductions, Neurocomputing 112 (2013) 34–46.

[6] T. Gärtner, A survey of kernels for structured data, SIGKDD Explorations 5 (2003) 49–58.

[7] F. Aiolli, G. Da San Martino, M. Hagenbuchner, A. Sperduti, Learning nonsparse kernels by self-organizing maps for structured data, IEEE Trans. on Neural Netw. 20 (2009) 1938 –1949.

[8] D. Bacciu, A. Micheli, A. Sperduti, Generative kernels for tree-structured data, IEEE Transactions on Neural Networks and Learning Systems PP (2018) 1–15.

[9] K. S. Tai, R. Socher, C. D. Manning, Improved semantic representations from tree-structured long short-term memory networks, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, 2015, pp. 1556–1566. URL: http://www.aclweb.org/anthology/P15-1150. doi:10.3115/v1/P15-1150.

[10] D. Bacciu, A. Bruno, Text summarization as tree transduction by top-down TreeLSTM, in: Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI'18), IEEE, 2018.

[11] A. M. C. Gallicchio, Tree echo state networks, Neurocomputing 101 (2013) 319 – 337.

[12] D. Bacciu, Hidden tree markov networks: Deep and wide learning for structured data, in: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), 2017, pp. 1–8.

[13] B. Hammer, A. Micheli, A. Sperduti, M. Strickert, A general framework for unsupervised processing of structured data, Neurocomputing 57 (2004) 3–35.

[14] M. Hagenbuchner, A. Sperduti, A. Tsoi, A self-organizing map for adaptive processing of structured data, IEEE Trans. Neural Networks 14 (2003) 491–505.

[15] N. Gianniotis, P. Tino, Visualization of Tree-Structured Data Through Generative Topographic Mapping, IEEE Trans. on Neural Netw. 19 (2008) 1468–1493.

[16] D. Bacciu, A. Micheli, A. Sperduti, Compositional generative mapping for tree-structured data - part II: Topographic projection model, IEEE Trans. Neural Netw. Learning Syst. 24 (2013) 231–247.

[17] M. Hagenbuchner, A. Sperduti, A. C. Tsoi, F. Trentini, F. Scarselli, M. Gori, Clustering xml documents using self-organizing maps for structures, in: N. Fuhr, M. Lalmas, S. Malik, G. Kazai (Eds.), Advances in XML Information Retrieval and Evaluation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 481–496.

[18] R. M. Neal, Markov chain sampling methods for dirichlet process mixture models, Journal of computational and graphical statistics 9 (2000) 249–265.

[19] D. Bacciu, D. Castellana, Mixture of hidden markov models as tree encoder, in: M. Verleysen (Ed.), Proc.of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN'18), i6doc.com, 2018, pp. 543–548.

[20] G. McLachlan, D. Peel, Finite Mixture Models, Wiley series in probability and statistics: Applied probability and statistics, Wiley, 2004.

[21] T. S. Ferguson, A bayesian analysis of some nonparametric problems, The annals of statistics (1973) 209–230.

[22] E. Rendón, I. Abundez, A. Arizmendi, E. Quiroz, Internal versus external cluster validation indexes, Int. J. Comp. and Comm. 5 (2011) 27–34.

[23] P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53 – 65.

[24] M. Deza, E. Deza, Encyclopedia of distances, in: Encyclopedia of Distances, Springer, 2009, pp. 1–583.

[25] L. Denoyer, P. Gallinari, Report on the xml mining track at inex 2005 and inex 2006: categorization and clustering of xml documents, in: ACM SIGIR Forum, volume 41, ACM, 2007, pp. 79–90.