IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 7, 2020, accepted September 22, 2020, date of publication September 28, 2020,
date of current version December 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3027064

Incremental Learning of Latent Forests

FERNANDO RODRIGUEZ-SANCHEZ, PEDRO LARRAÑAGA, (Member, IEEE), AND CONCHA BIELZA, (Member, IEEE)
Computational Intelligence Group, Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, 28660 Madrid, Spain

Corresponding author: Fernando Rodriguez-Sanchez (fernando.rodriguezs@upm.es)

This work was supported in part by the Spanish Ministry of Economy and Competitiveness through the TIN2016-79684-P and PID2019-109247GB-I00 projects, and in part by the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreements No. 785907 (Human Brain Project SGA2) and No. 945539 (Human Brain Project SGA3).

ABSTRACT In the analysis of real-world data, it is useful to learn a latent variable model that represents the data generation process. In this setting, latent tree models are useful because they are able to capture complex relationships while being easily interpretable. In this paper, we propose two incremental algorithms for learning forests of latent trees. Unlike current methods, the proposed algorithms are based on the variational Bayesian framework, which allows them to introduce uncertainty into the learning process and work with mixed data. The first algorithm, *incremental learner*, determines the forest structure and the cardinality of its latent variables in an iterative search process. The second algorithm, *constrained incremental learner*, modifies the previous method by considering only a subset of the most prominent structures in each step of the search. Although restricting each iteration to a fixed number of candidate models limits the search space, we demonstrate that the second algorithm returns almost identical results for a small fraction of the computational cost. We compare our algorithms with existing methods by conducting a comparative study using both discrete and continuous real-world data. In addition, we demonstrate the effectiveness of the proposed algorithms by applying them to data from the 2018 Spanish Living Conditions Survey. All code, data, and results are available at https://github.com/ferjorosa/incremental-latent-forests.

INDEX TERMS Latent variable model, variational Bayes, latent tree model, hidden variables.

I. INTRODUCTION

Real-world data are often complex and high-dimensional. In this setting, latent variables have proven to be useful in analyzing their generation process, as they are able to represent underlying data concepts by grouping observed variables. A latent variable model that has received considerable attention is the latent tree model (LTM) [1], [2]. The LTM is a tree-structured probabilistic graphical model whose leaf nodes are observed variables and whose internal nodes can be either observed or latent variables.

LTMs are appealing because can capture complex relationships with a simple tree structure that is easily interpretable. Furthermore, they allow for exact probabilistic inference in linear time [3]. For this reason, LTMs have proven to be valuable in many areas, such as classification [4]–[6], topic detection [7], probabilistic inference [8] and cluster analysis [9]–[13].

The associate editor coordinating the review of this manuscript and approving it for publication was Benyun Shi.

Existing literature includes several approaches that address the problem of learning LTMs from data [10]–[21]. However, the majority of them rely on point estimates of the parameters, which do not provide a good assessment of a model's generalization given the data, and do not allow for the incorporation of expert knowledge into the learning process. We address these two problems by making use of the variational Bayes (VB) framework [22]. In this framework, point estimates over parameters are replaced with distributions, thus naturally reflecting the uncertainty over parameter values given the data.

Forests of LTMs are appealing owing to their ability to separate groups of similar variables in different trees. Although several methods for learning latent forests have been studied [15]–[18], they are currently limited to function with either discrete or continuous data. In this work, we propose two new learning algorithms of latent forests that are based on the VB framework, which intrinsically allows them to work with discrete, continuous and mixed datasets.

Our first algorithm, which we refer to as incremental learner (IL), hill-climbs the space of latent forests in a

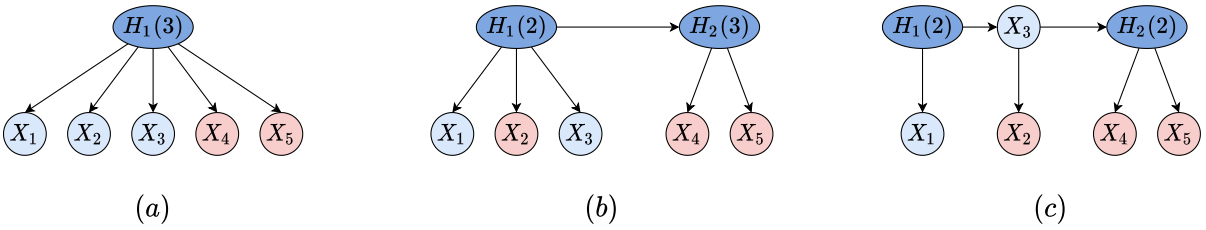


FIGURE 1. Examples of (a) an LCM, (b) an HLCM and (c) a general LTM with observed internal nodes. Discrete variables are colored blue while continuous variables are colored red. The number in parentheses accompanying each latent variable represents its cardinality.

two-phase iterative process. In its first phase, the forest structure is incremented with a new arc or latent variable. In its second phase, the cardinalities of latent variables are estimated. Our second algorithm, which we name constrained incremental learner (CIL), modifies the previous procedure by considering only a subset of candidate structures in each step of the search. We show that by doing this, it is able to return almost identical results for a fraction of the computational cost.

The remainder of this paper is organized as follows. In Section II, we formally introduce LTMs and describe how they can be learned from data using the VB framework. In Section III, we present our approach for incrementally learning forests of LTMs. Then, in Section IV, we explore the performance of our proposed approach with respect to the state of the art for various experimental settings: (i) discrete data, (ii) continuous data, and (iii) mixed data. Finally, we report our conclusions and propose future research directions in Section V.

II. PRELIMINARIES

In this section, we describe LTMs and present an overview of how to learn their parameters and structure from data. For notation, we use capital letters, such as X, Y, Z, to denote variable names, and lower-case letters, such as x, y, z, to denote specific values taken by these variables. Sets of variables are indicated by bold capital letters, such as $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, and assignments of values to these variables are indicated by bold lower-case letters, such as $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$.

For a dataset with N instances, we denote the observed variables and their values in the nth instance by $\mathbf{X}[n]$ and $\mathbf{x}[n]$, respectively, and denote the latent (or hidden) variables and their possible assignments in the nth instance by $\mathbf{H}[n]$ and $\mathbf{h}[n]$, respectively. We use $\mathcal{D} = \cup_n \mathbf{x}[n]$ to denote all observed values, and use $\mathcal{H} = \cup_n \mathbf{h}[n]$ to denote all possible assignments to all hidden values in the dataset. Thus, the pair $(\mathcal{D}, \mathcal{H})$ defines an assignment to all variables in all data instances.

A. LATENT TREE MODELS

A Bayesian network (BN) $\mathcal{B}$ is a probabilistic graphical model [3], [23] that represents a joint probability distribution over a set of observed $\mathbf{X} = \{X_1, X_2, \ldots, X_S\}$ and latent $\mathbf{H} = \{H_1, H_2, \ldots, H_L\}$ random variables. It is defined by a directed acyclic graph $\mathcal{G}$, which represents the network structure that encodes existing conditional independence between triplets of variables, and a set of parameters $\boldsymbol{\theta}$, which comprises the conditional probability distributions of each variable given its parents in $\mathcal{G}$. That is, $p(x_s|\mathbf{pa}_{x_s})$ and $p(h_l|\mathbf{pa}_{h_l})$, where $\mathbf{pa}_{x_s}$ denotes a value in the set of X_s parents, $\mathbf{Pa}_{X_s}$. This is equivalent for H_l. Thus, the joint distribution is factorized as

$$p(\mathbf{x}, \mathbf{h}) = \prod_{s=1}^{S} p(x_s|\mathbf{pa}_{x_s}) \prod_{l=1}^{L} p(h_l|\mathbf{pa}_{h_l}). \tag{1}$$

LTMs are tree-structured BNs whose leaf nodes are observed variables and whose internal nodes can be either observed or latent variables. Subclasses of LTMs, such as phylogenetic trees [24] and latent class models (LCMs) [25], [26], have been studied for decades. In an LCM, all observed variables are conditionally independent given a single latent variable. The hierarchical latent class model (HLCM) generalizes the LCM by allowing multiple latent variables, where all observed variables are leaf nodes. Therefore, an HLCM is a restricted version of an LTM whose internal nodes cannot be observed variables. Fig. 1 presents examples of these models, including an LTM with no restrictions.

To learn the structure of an LTM, the following information must be determined: (i) the number of latent variables, (ii) the cardinality of each latent variable, and (iii) the connections between variables. To learn the structure of a latent forest, it is necessary to determine the structure of each tree that forms it. Structure learning algorithms fall into the following three categories [1]: search-based methods, variable clustering methods, and distance-based methods.

Search-based methods construct LTMs one local move at a time, such as by introducing a latent variable, removing an arc, and increasing the cardinality of a latent variable. Zhang [10] was the first to propose an approach of this kind, called double hill-climbing (DHC) algorithm. DHC starts with an LCM and then explores the space of HLCMs. At each step of the search, it employs a hill-climbing procedure to identify the best HLCM that can be produced by introducing a single arc or latent variable. Once this model is found, DHC optimizes the cardinalities of its latent variables by employing a second hill-climbing procedure (hence the name of the algorithm). However, this results in a very high computational complexity. For this reason, a less computationally

intensive alternative called the single hill-climbing (SHC) algorithm [14] was proposed. SHC balances model quality and computational efficiency by employing a single hill-climbing procedure to explore the space of HLCMs. This method was later improved in [11] by dividing each search step into three stages: expansion, adjustment, and simplification. Each stage uses its own set of operators (e.g., during the expansion stage, the LTM is modified only by introducing a new are or latent variable). For this reason, this algorithm is called EAST (expansion, adjustment, simplification until termination).

The aforementioned algorithms were all designed to work with discrete data. An extension of the EAST algorithm for continuous data was later proposed in [12]. Additionally, for continuous data, Galimberti and Soffritti proposed a greedy method for learning forests of unconnected LCMs. Both methods assume that each observed variable in the model follows a Gaussian distribution.

Although search-based methods usually find high-scoring models, they are computationally expensive, as they involve the evaluation of a large number of models. Methods based on variable clustering are much faster alternatives with a small compromise in model quality. All clustering-based methods rely on two key points: grouping variables to identify new latent variables and constructing models in a hierarchical manner using a bottom-up strategy. Three types of structures have been studied: HLCMs via the bridged-islands (BI) algorithm [13], binary forests (in which each tree node can have at most two children) via the BIN-G and BIN-A algorithms [16], and non-binary forests (in which each tree node has no restrictions on the number of children) via the CFHLC algorithm [17]. These algorithms are all limited to work with discrete data.

Phylogenetic tree reconstruction led to the development of distance-based methods [19], [20]. These algorithms learn an LTM from the information distances of observed variables, connecting variables with a new arc or latent variable. The main advantage of these algorithms is their ability to recover the correct latent structure under some mild conditions. However, they have several drawbacks. First, when applied to discrete data, all variables must share the same state space. Second, in the presence of continuous data, all variables must be continuous; that is, mixing continuous and discrete variables is not possible. The work of Huang *et al.* [21] circumvents these issues by defining a new distance metric and a new parameter learning method based on moments and tensor decompositions [27]. However, similar to its predecessors, it is currently limited to learning a single tree.

The majority of search-based, clustering-based, and distance-based methods estimate model parameters using the expectation-maximization (EM) algorithm [28]. Each iteration t of this algorithm is divided into two steps: (i) the expectation step, in which estimated parameters $\hat{\boldsymbol{\theta}}^{(t)}$ are used to infer the posterior distribution of latent variables $p(\mathcal{H}|\mathcal{D}, \hat{\boldsymbol{\theta}}^{(t)})$; and (ii) the maximization step, in which a new point estimate of the parameters $\hat{\boldsymbol{\theta}}^{(t+1)}$ (usually the maximum likelihood) is computed.

The likelihood function is not appropriate for model selection because complex model structures usually have a large number of parameters and thus score higher, leading to overfitting. For this reason, a penalized version of the likelihood function can be used instead, such as the Bayesian information criterion (BIC) [29]. BIC applies a penalty term to the log-likelihood that depends on the model complexity. For an LTM with a number of $\dim(\mathcal{G})$ parameters, BIC is defined as

$$\mathrm{BIC}(\mathcal{G} : \mathcal{D}) = \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}) - \frac{\dim(\mathcal{G})}{2} \log N. \tag{2}$$

B. BAYESIAN LEARNING WITH LATENT VARIABLES

By using a point estimate of the parameters, there is no measure of uncertainty, and no prior knowledge can be incorporated into the learning process. In Bayesian statistics, prior knowledge is introduced via a prior distribution, and uncertainty is reflected in its posterior distribution. The posterior distribution encodes updated beliefs once prior knowledge and data have been taken into consideration. For a fixed structure $\mathcal{G}$, two quantities are of interest. The first is the posterior distribution over the sets of parameters and latent variables:

$$p(\mathcal{H}, \boldsymbol{\theta}|\mathcal{D}, \mathcal{G}) = \frac{p(\mathcal{D}|\mathcal{H}, \boldsymbol{\theta}, \mathcal{G})p(\mathcal{H}, \boldsymbol{\theta}|\mathcal{G})}{p(\mathcal{D}|\mathcal{G})}. \tag{3}$$

The second quantity is the marginal likelihood, which acts as a normalizing constant for the posterior distribution and is a key element in model selection:

$$p(\mathcal{D}|\mathcal{G}) = \int\int p(\mathcal{D}|\mathcal{H}, \boldsymbol{\theta}, \mathcal{G})p(\mathcal{H}, \boldsymbol{\theta}|\mathcal{G})d\mathcal{H}d\boldsymbol{\theta}. \tag{4}$$

In the Bayesian learning problem, we assume that the learner has a prior distribution over the set of structures $p(\mathcal{G})$. Furthermore, in theory, we should average over all possible structures when estimating the posterior distribution in (3). However, in practice, constraints on storage and computation or ease of interpretability may lead us to select the most probable model structure. In this setting, if we assume that all structures are equally probable for the given data, then model selection is equivalent to choosing a structure with the highest marginal likelihood. Unlike $p(\mathcal{D}|\hat{\boldsymbol{\theta}})$ in the BIC score (2), the marginal likelihood automatically penalizes complex models by integrating out the parameters and latent variables. This is called the Bayesian Occam's razor [30].

The marginal likelihood is tractable to compute for certain types of models, such as fully observed discrete BNs [31]. However, when latent variables are present, such as in LTMs, the marginal likelihood becomes intractable to compute even for moderately sized datasets. In such situations, approximation schemes are required. Two of the most popular approximation schemes are the Markov chain Monte Carlo (MCMC) method [32], [33] and variational inference (VI) [34]. MCMC is a nonparametric method that produces asymptotically exact results but at a high computational cost. In contrast, VI has the

advantage of being faster by using a parametrized approximation, in which the Bayesian inference problem is solved via optimization. In this paper we focus on the VI approach.

C. VARIATIONAL BAYES FRAMEWORK

The goal of VI is to find an approximate distribution $q(\mathcal{H}, \boldsymbol{\theta})$ from some tractable family $\mathcal{Q}$ that closely approximates the true posterior distribution $p(\mathcal{H}, \boldsymbol{\theta}|\mathcal{D}, \mathcal{G})$. For simplicity, we denote these distributions q and p, respectively. The key principle of VI is to solve this problem via optimization, in which a set of variational parameters $\boldsymbol{\varphi}$ that makes q closest to p is identified. The usual cost function for this optimization problem is the reverse Kullback-Leibler (KL) divergence:

$$\begin{aligned}\mathrm{KL}(q||p) &= \int\int q(\mathcal{H}, \boldsymbol{\theta}) \log \frac{q(\mathcal{H}, \boldsymbol{\theta})}{p(\mathcal{H}, \boldsymbol{\theta}|\mathcal{D}, \mathcal{G})} d\mathcal{H} d\boldsymbol{\theta} \\ &= \mathbb{E}_q \left[\log \frac{q(\mathcal{H}, \boldsymbol{\theta})}{p(\mathcal{H}, \boldsymbol{\theta}|\mathcal{D}, \mathcal{G})} \right] \\ &= \mathbb{E}_q \left[\log q(\mathcal{H}, \boldsymbol{\theta}) \right] - \mathbb{E}_q \left[\log p(\mathcal{H}, \boldsymbol{\theta}, \mathcal{D}|\mathcal{G}) \right] \\ &\quad + \log p(\mathcal{D}|\mathcal{G}). \end{aligned} \tag{5}$$

However, we cannot minimize this function, as that requires computing the marginal likelihood in (4). Instead, we can maximize an alternative objective that is equivalent to the reverse KL divergence up to an added constant, $\log p(\mathcal{D}|\mathcal{G})$. This function is called the evidence lower bound (ELBO):

$$\mathrm{ELBO}(q : \mathcal{D}) = \mathbb{E}_q \left[\log p(\mathcal{H}, \boldsymbol{\theta}, \mathcal{D}|\mathcal{G}) \right] - \mathbb{E}_q \left[\log q(\mathcal{H}, \boldsymbol{\theta}) \right]. \tag{6}$$

Two conclusions can be inferred from (5) and (6). First, we can see that maximizing the ELBO is equivalent to minimizing $\mathrm{KL}(q||p)$. Second, we can see that, as its name suggests, the ELBO is a lower bound of $\log p(\mathcal{D}|\mathcal{G})$, which follows from the fact that $\mathrm{KL}(\cdot) \geq 0$. This relationship with the marginal likelihood has led the ELBO to be used as a model selection criterion. However, when applied to models with discrete latent variables (e.g., LTMs), an adjustment must be made to account for the parameters' lack of identifiability. To better understand this problem, consider a conditional distribution of a variable whose parent is a discrete latent variable with K labels. There are $K!$ equivalent settings of parameters for this conditional distribution, which differ merely by permuting the parent labels.

Two common approaches exist for addressing this redundancy: (i) using asymmetric priors; (ii) introducing a small penalty into the ELBO score. We use the second approach when there is a lack of expert knowledge in the learning process. In this context, a simple penalty involves subtracting the term $\log K!$ from the lower bound [35]. We can generalize this penalty for a model with multiple discrete latent variables H_l, each with cardinality K_l. Then, we can use it to define a penalized version of the ELBO:

$$\text{p-ELBO}(q : \mathcal{D}) = \mathrm{ELBO}(q : \mathcal{D}) - \sum_{l=1}^{L} \log K_l!. \tag{7}$$

The complexity of maximizing the ELBO (and therefore, the p-ELBO) is determined by the complexity of the variational family $\mathcal{Q}$. In this paper, we use the VB framework [22], which assumes the following factorization of the variational posterior $q(\mathcal{H}, \boldsymbol{\theta}) = q(\mathcal{H})q(\boldsymbol{\theta})$. The VB framework iteratively maximizes the ELBO with respect to $q(\mathcal{H})$ and $q(\boldsymbol{\theta})$. This results in an iterative algorithm that is directly analogous to the EM (called the VBEM algorithm), which is guaranteed to monotonically increase the ELBO.

The exact form of the variational expectation and maximization equations depends on the functional form of the conditional distributions in the model (e.g., discrete BNs [36]). However, deriving a set of specific update equations for each type of conditional distribution is an arduous task. Fortunately, the *variational message passing* (VMP) framework [37] provides a set of general purpose update equations that work for any BN for which all conditional distributions are in the exponential family, and for which all parent distributions are conjugate.[1] A model in which both of these constraints hold is known as a *conjugate-exponential* (CE) model.

In this paper, we combine the VMP framework with the VBEM algorithm to learn LTMs with discrete and/or continuous variables. Continuous variables are assumed to follow a Gaussian distribution, which belongs to the CE family. The only restriction imposed by the VMP framework is that a discrete variable cannot have a Gaussian parent.

III. INCREMENTAL LEARNING OF LATENT FORESTS

In this section, we propose a search-based method that hill-climbs the space of latent forests using the VB framework. Instead of exploring this space directly, we approach this search as an iterative process with two phases as follows:

- **Structure phase**. The forest structure is incrementally built, in which variables are connected via a new arc or latent variable.
- **Cardinalities phase**. The cardinalities of previously involved latent variables are estimated.

We use two search operators (node addition and arc addition) to modify the forest structure and another two operators (state introduction and state removal) to estimate latent cardinalities.

A. SEARCH OPERATORS

Node addition (NA) involves two variables (observed and latent) and generates a new model by introducing a latent variable as their parent. The cardinality of this variable is 2. To reduce computational complexity, we limit this operator to consider pairs of variables. This restriction is compensated by the following operator.

Arc addition (AA) generates a new model by introducing an arc from one variable to another. In this operator,

[1] A parent distribution $P(X|Y)$ is said to be conjugate to a child distribution $P(W|X)$ if $P(X|Y)$ has the same functional form, with respect to X, as $P(W|X)$.

both directions are considered. However, there is an intrinsic restriction given by the CE family that a continuous variable cannot be the parent of a discrete variable. Therefore, these arcs are ignored by the operator.

State introduction (SI) creates a new model by adding a state to a latent variable. In contrast, state removal (SR) creates a new model by removing a state from a latent variable. The SI is operator is not applicable if the variable has a number of states equal to K_{max}, which can be specified by the user. The SR operator is not applicable if the variable only has two states.

B. SEARCH PROCESS

The search starts with an unconnected forest $\mathcal{M}$ in which no latent variables are present and whose observed variables are independent. These observed variables form the working set $\mathbf{W}$. The parameters of this model are learned via the VBEM algorithm and its corresponding score is stored as a baseline for comparison.

In the structure phase, the NA and AA operators are applied to each pair of variables in $\mathbf{W}$, resulting in the $\mathbf{M}_{NA}$ and $\mathbf{M}_{AA}$ sets of candidate models. Each candidate model is then evaluated by learning its parameters and storing its p-ELBO score. When this process is completed, the highest scoring model $\mathcal{M}_S$ is selected, and all of its variables that were involved in the selected operator are stored in a new set of variables $\mathbf{V}$. We refer to this phase as the `structure` subroutine and it is formally described in Algorithm 1.

Algorithm 1 `structure`

Input : $\mathcal{M}$, $\mathbf{W}$, $\mathcal{D}$

$\mathbf{M}_{NA} \leftarrow \text{NA}(\mathcal{M}, \mathbf{W}, \mathcal{D})$
$\mathbf{M}_{AA} \leftarrow \text{AA}(\mathcal{M}, \mathbf{W}, \mathcal{D})$
$\mathcal{M}_S \leftarrow$ highest scoring model in $\mathbf{M}_{NA} \cup \mathbf{M}_{AA}$
$\mathbf{V} \leftarrow$ variables involved in the selected operator

Output: The resulting latent forest $\mathcal{M}_S$ and $\mathbf{V}$

In the cardinalities phase, the SI and SR operators are applied to the latent variables in $\mathbf{V}$, resulting in the $\mathbf{M}_{SI}$ and $\mathbf{M}_{SR}$ sets of candidate models. In contrast to the structure phase, this is done iteratively. It starts with $\mathcal{M}_C$ and then, at each iteration, applies the SI and SR operators, selecting the best candidate model $\mathcal{M}'_C$. If its p-ELBO is greater than that of $\mathcal{M}_C$, the candidate takes its place and the process continues. Once the score ceases to improve, the resulting model is returned as $\mathcal{M}_C$. We refer to this phase as the `cardinalities` subroutine and it is formally described in Algorithm 2.

After the cardinalities phase, the scores of $\mathcal{M}$ and $\mathcal{M}_C$ are compared. If there is an improvement from the previous iteration, then $\mathcal{M}_C$ becomes the current best model, and the working set $\mathbf{W}$ is updated. The update process depends on the operator that is selected on the structure phase. In the case of node addition, children variables are removed from $\mathbf{W}$ and the new latent variable is added to $\mathbf{W}$. In the case of arc addition, the parent variable remains in $\mathbf{W}$ and the child is removed.

Algorithm 2 `cardinalities`

Input : $\mathcal{M}$, $\mathbf{V}$, $\mathcal{D}$, K_{max}

$\mathcal{M}_C \leftarrow \mathcal{M}$
while *True* **do**
  $\mathbf{M}_{SI} \leftarrow \text{SI}(\mathcal{M}_C, \mathbf{V}, \mathcal{D}, K_{max})$
  $\mathbf{M}_{SR} \leftarrow \text{SR}(\mathcal{M}_C, \mathbf{V}, \mathcal{D})$
  $\mathcal{M}'_C \leftarrow$ highest scoring model in $\mathbf{M}_{SI} \cup \mathbf{M}_{SR}$
  if $p\text{-}ELBO(\mathcal{M}'_C, \mathcal{D}) > p\text{-}ELBO(\mathcal{M}_C, \mathcal{D})$ **then**
    $\mathcal{M}_C \leftarrow \mathcal{M}'_C$
  else
    break /* Stop the loop */

Output: The resulting latent forest $\mathcal{M}_C$

Algorithm 3 Incremental Learner (IL)

Input : $\mathcal{D}$, K_{max}

$\mathbf{W} \leftarrow \mathbf{X}$
Let $\mathcal{M}$ be an unconnected latent forest with nodes $\mathbf{X}$
while $|\mathbf{W}| \neq 1$ **do**
  $\mathcal{M}_S, \mathbf{V} \leftarrow$ `structure`$(\mathcal{M}, \mathbf{W}, \mathcal{D})$
  $\mathcal{M}_C \leftarrow$ `cardinalities`$(\mathcal{M}_S, \mathbf{V}, \mathcal{D}, K_{max})$
  if $p\text{-}ELBO(\mathcal{M}_C, \mathcal{D}) > p\text{-}ELBO(\mathcal{M}, \mathcal{D})$ **then**
    $\mathcal{M} \leftarrow \mathcal{M}_C$
    Update $\mathbf{W}$ with $\mathbf{V}$
  else
    break /* Stop the loop */
Model refinement (see Section III-D for details)

Output: The resulting latent forest $\mathcal{M}$

The algorithm alternates between these two phases until the score ceases to increase or there is only one remaining variable in $\mathbf{W}$. We call this method IL and it is formally described in Algorithm 3.

Fig. 2 provides an example execution of the IL algorithm. It starts with an empty latent forest with sets of two discrete and two continuous observed variables, $\{X_1, X_2\}$ and $\{X_3, X_4\}$, respectively. In its first iteration, the AA operator is selected and X_3 becomes the new parent of X_4, resulting in the removal of X_4 from $\mathbf{W}$. In its second iteration, the NA operator is selected and a new latent variable H_1 is included as the parent of X_2 and X_3, thus removing both X_2 and X_3 from $\mathbf{W}$. The cardinality of H_1 is estimated and it remains at its initial value, 2. In the third and final iteration, the AA operator is selected again, and H_1 is set as the parent of X_1. Unlike the previous iteration, the cardinality of H_1 is increased to 4. Given that only one variable remains in $\mathbf{W}$, the process stops, and if the p-ELBO of the model is greater than that of the previous iteration, then $\mathcal{M}$ is updated and returned.

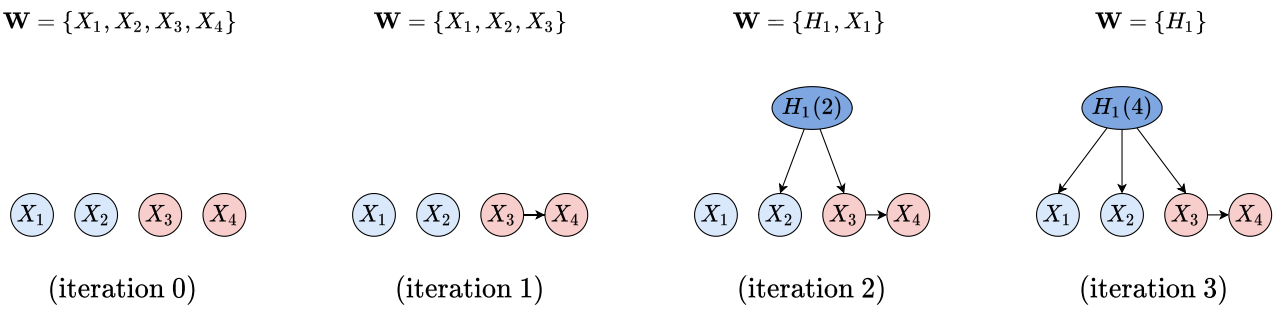


FIGURE 2. Example execution of the incremental learner (IL) algorithm with three iterations. It introduces an arc in its first iteration, a latent variable in its second iteration and an arc on its third (and final) iteration. Colors and parenthesis have the same meaning as in Fig. 1.

However, IL can be very time-consuming due to two factors. First, each iteration requires the evaluation of a large number of models. In the structure phase, the NA and AA operators evaluate a total of $O(3(|\mathbf{W}|^2 - |\mathbf{W}|)/2)$ candidate models. In the cardinalities phase, the number of evaluations depend on the previously selected operator. The maximum number of evaluations occurs when a latent variable is the new parent of two other latent variables, resulting in $O(3K_{max})$ evaluations for SI and SR. Therefore, the total number of candidate models evaluated at each iteration is $O(3(|\mathbf{W}|^2 - |\mathbf{W}|)/2 + 6K_{max})$. The second time-consuming factor is that each candidate evaluation requires running the VBEM algorithm, which is computationally expensive.

To address these challenges, we propose an alternative search procedure that considers fewer candidates in Section III-C, and in Section III-D, we propose an alternative to the VBEM algorithm that speeds up the model evaluation.

C. CONSTRAINED SEARCH

A straightforward approach for increasing the speed of the proposed method is to introduce restrictions in the search process [38], [39]. We achieve this by only evaluating certain structures. More specifically, in the structure phase, instead of considering all pairs of variables in $\mathbf{W}$, we select the pair with the highest mutual information (MI). The rationale for this approach stems from the following observation. Let $p(\mathbf{w})$ be the probability distribution over the set of variables in $\mathbf{W}$. Knowing that all variables in this set are currently considered to be independent of each other, we can approximate this distribution with another distribution, $\hat{p}(\mathbf{w})$, that models the joint distribution of two variables W_i and W_j and the marginal distributions of the remaining variables:

$$\hat{p}(\mathbf{w}) = p(w_i, w_j) \prod_{k \neq i,j} p(w_k) = \frac{p(w_i, w_j)}{p(w_i)p(w_j)} \prod_k p(w_k). \quad (8)$$

Then, our objective is to select the pair of variables whose connection (via an arc or latent variable and represented by their joint probability distribution $p(w_i, w_j)$) makes $\hat{p}(\mathbf{w})$ as close to $p(\mathbf{w})$ as possible. Evaluating this separation using the KL divergence, we have

$$\begin{aligned} \mathrm{KL}(p||\hat{p}) &= \int p(\mathbf{w}) \, log \frac{p(\mathbf{w})}{\hat{p}(\mathbf{w})} d\mathbf{w} \\ &= \int p(\mathbf{w}) \, log \frac{p(\mathbf{w})p(w_i)p(w_j)}{p(w_i, w_j) \prod_k p(w_k)} d\mathbf{w} \\ &= \int p(\mathbf{w}) \, log \frac{p(\mathbf{w})}{\prod_k p(w_k)} dw - I(W_i, W_j) \end{aligned} \quad (9)$$

where $I(W_i, W_j)$ is the MI between W_i and W_j under the distribution $p(w_i, w_j)$ and is defined as

$$I(W_i, W_j) = \int \int p(w_i, w_j) \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)} dw_i dw_j. \quad (10)$$

By selecting the pair of variables with the highest MI, we minimize the KL divergence between $p(\mathbf{w})$ and $\hat{p}(\mathbf{w})$. However, (10) is often computationally intractable given that W_i and W_j can be either discrete or continuous. For this reason, approximate MI estimators, such as binning, k-nearest neighbors [40], or kernel density estimation [41], are required. By approximating the MI, we cannot ensure that the selected pair of variables results in the truly minimal $\mathrm{KL}(p||\hat{p})$. A simple way to alleviate this problem is to consider additional pairs of variables. In this case, we propose considering α pairs with the highest MI.

We apply this idea to IL, resulting in the CIL method, which is formally described in Algorithm 4. CIL is very similar to IL, the main execution differences occur (i) at the beginning, in line 2, where CIL estimates the MI matrix with each pair of variables in $\mathbf{W}$; (ii) in the structure phase, in line 6, where CIL only considers the α pairs of variables in $\mathbf{W}$ with the highest MI; and (iii) at the end of each iteration, in line 12, where CIL updates the MI matrix. The update removes the variables that have been removed from $\mathbf{W}$ and estimates the MI of new variables in $\mathbf{W}$. In this method, α represents a compromise between model quality and computational cost.

CIL generates fewer candidate models than IL at each step of the search. While its cardinalities phase is identical to that of IL, its structure phase depends only on α, which results in a total of $O(2\alpha + 6K_{max})$ evaluations at each step of the search.

Algorithm 4 Constrained Incremental Learner (CIL)

Input : $\mathcal{D}$, K_{max}, α

$\mathbf{W} \leftarrow \mathbf{X}$
Estimate the MI matrix with each pair in $\mathbf{W}$
Let $\mathcal{M}$ be an unconnected latent forest with nodes $\mathbf{X}$
while $|\mathbf{W}| \neq 1$ **do**
  $\mathbf{U} \leftarrow \alpha$ pairs of variables in $\mathbf{W}$ with highest MI
  $\mathcal{M}_S$, $\mathbf{V} \leftarrow$ `structure`$(\mathcal{M}, \mathbf{U}, \mathcal{D})$
  $\mathcal{M}_C \leftarrow$ `cardinalities`$(\mathcal{M}_S, \mathbf{V}, \mathcal{D}, K_{max})$
  if $p\text{-}ELBO(\mathcal{M}_C, \mathcal{D}) > p\text{-}ELBO(\mathcal{M}, \mathcal{D})$ **then**
    $\mathcal{M} \leftarrow \mathcal{M}_C$
    Update $\mathbf{W}$ with $\mathbf{V}$
    Update the MI matrix with respect to $\mathbf{W}$
  else
    break /* Stop the loop */
Model refinement (see Section III-D for details)

Output: The resulting latent forest $\mathcal{M}$

D. EFFICIENT MODEL EVALUATION

Repeatedly evaluating candidate models can become prohibitive when each evaluation involves running the VBEM algorithm. Previously, we addressed this problem by reducing the number of candidates. However, when the number of variables is high, this reduction may be insufficient. One solution that has been successfully applied in the LTM [11], [16] and phylogenetics [42] literature involves optimizing some parameters of the model while the remaining parameters stay constant.

Following this idea, we propose replacing the VBEM algorithm with a more efficient procedure that estimates the variational posterior distribution $q(\mathcal{H}, \boldsymbol{\theta})$ of only several variables while the remaining variables stay constant. We refer to this method as local VBEM. Thus, when evaluating a candidate model using local VBEM, we estimate the variational posteriors of (i) variables involved in the search operator, and (ii) variables belonging to the Markov blankets (MBs) of variables in (i). For any variable in a BN, its MB consists of the set of all its parents, children, and spouses (parents of children).

To illustrate the behavior of local VBEM, let us examine the rightmost example in Fig. 2, which is produced by applying the AA operator on H_1 and X_1. Evaluating this model requires the estimation of variational posteriors of X_1 and H_1 as well as X_2 and X_3, which belong to the MB of H_1. Incidentally, we do not require X_4 because it does not belong to the MB of either H_1 or X_1.

One iteration of local VBEM is computationally much cheaper than one iteration of VBEM because it updates fewer variational parameters. This also implies that a run of local VBEM requires fewer steps to converge than a run of VBEM. However, parameter estimates provided by local VBEM may deviate from those provided by VBEM. To prevent this from affecting the quality of the IL and CIL results, we can perform a run of VBEM after several search steps or before returning the model. In this paper, we run VBEM before returning the model.

Finally, like VBEM, local VBEM may get trapped at local maxima. To avoid this, we use a variant of the multiple-restart approach proposed by Chickering and Heckerman [43] and adapt it for the variational case. First, we sample C initial configurations of the variational parameters $\boldsymbol{\varphi}$. Next, we perform one VBEM step and retain $C/2$ of the configurations that lead to the largest score values. Then, we perform two VBEM steps and retain $C/4$ configurations. We continue this procedure, doubling the number of VBEM steps in each iteration until only one configuration remains.

E. PRIOR SPECIFICATION

A key aspect in Bayesian learning is the specification of prior distributions. This includes two aspects: their form and parameters. Because IL and CIL are designed to work with CE models, the forms of prior distributions are already established, and it is only necessary to focus on their parametrization.

Tuning prior parameters is largely dependent on the availability of expert information. This information can be obtained from a person or from other directly related studies (see Bayesian meta-analysis [44]). When expert information is available, prior parameter values can be selected to best reflect the expert knowledge. When this information is unavailable, we propose using the following strategy, which varies for observed and latent variables. First, for observed variables, we use an empirical Bayes [45] approach and assign maximum likelihood estimates to their prior parameters. Second, for latent variables, we assume a symmetric Dirichlet prior with a total concentration of 1.

IV. EXPERIMENTS

In this section, we evaluate the performances of IL and CIL in terms of data fitting and computational complexity. To this end we first conducted a comparative study using both discrete (Section IV-A) and continuous (Section IV-B) real-world data. Then, as described in Section IV-C, we analyzed real social data from the Spanish Living Conditions Survey of 2018, which contains both discrete and continuous variables.

In these experiments, IL and CIL were compared to several LTM methods of the state of the art. However, given that most of these algorithms are specific to the discrete data domain, we complemented this study with two other approaches that are not based on LTMs but can work with mixed datasets. The first one, called MSPN [46], combines sum-product networks [47] with nonparametric estimation to learn hierarchically structured latent variable models that do not require the specification of variables' parametric forms. The second one is a version of kernel density estimation that can work with continuous and discrete variables and uses Silverman's rule of thumb for bandwidth selection [48].

TABLE 1. Average 10-fold cross-validated predictive log-likelihood for various discrete datasets (rows) and algorithms (columns).

	m	N	BIN	EAST	BI	MSPN	IL	CIL ($\alpha = 1$)	CIL ($\alpha = 10$)
HIV-test	4	428	-346.10	-1485.89	-85.61	-104.84	**-84.22**	-84.48	**-84.22**
House-building	4	1185	-296.12	-295.67	**-293.54**	-296.27	-299.40	-299.40	-299.40
Hayes-roth	5	160	-93.89	-89.72	-89.98	**-88.89**	-89.09	-89.09	-89.09
Balance-scale	5	625	-570.61	-492.48	-491.67	-493.10	**-487.16**	**-487.16**	**-487.16**
Car-evaluation	7	1728	-1723.65	-12095.26	-1610.46	-1690.90	**-1503.81**	-1721.09	**-1503.81**
Nursery	9	12960	-119161.70	-264750.76	-20307.37	-16158.19	**-13626.52**	-14153.90	-13649.92
Breast-cancer	10	277	-267.60	-321.55	**-258.56**	-310.73	-261.36	-262.44	-261.36
Vote	17	232	-200.34	-216.01	-176.49	**-174.27**	-193.73	-194.44	-193.73
Mushroom	19	5644	-13616.74	**-5519.58**	-6079.81	-24784.02	-9179.16	-9173.26	-9179.16
Pascal-Voc-2007	20	10425	-12523.78	-14654.98	**-10500.84**	-54124.60	-10859.13	-10867.30	-10859.35
Spect-heart	23	267	-303.94	**-288.35**	-289.78	-316.32	-299.42	-299.42	-299.42
Alarm	36	1000	-1655.89	**-1179.96**	-1352.71	-4142.37	-1461.45	-1430.75	-1452.13
Coil-42	42	5822	-6480.32	-30597.77	-6978.14	-6879.65	– – –	-6316.36	**-6269.04**
News-100	100	16242	-27213.93	– – –	**-25343.09**	-26768.92	– – –	-26301.65	-26325.91
Webkb-336	336	1038	-10304.10	– – –	**-8430.17**	-9211.57	– – –	-9440.12	-9439.57

The winner in each row is indicated in bold. Results from the incremental learner (IL) and constrained incremental learner (CIL) that outperformed those from existing methods without being in first place ar colored in blue. Dashed lines indicate that the algorithm was unable to complete a fold in 24 hours.

Experiments were conducted on a computer with an Intel Core i7-6700K CPU at 4.00 GHz with 64GB RAM, running Windows 10 Enterprise. All experimental runs were performed on a single thread to allow a fair comparison between methods. For reproducibility, all code (including implementations and state-of-the-art executables), data, and results can be downloaded from our repository at https://github.com/ferjorosa/incremental-latent-forests.

A. COMPARATIVE STUDY ON DISCRETE DATA

The algorithms we considered in this comparative study are summarized as follows. The LCM algorithm starts with a latent class model of cardinality 2 and greedily increases its cardinality until the score ceases to improve. The BIN algorithm generates a binary latent forest following the agglomerative clustering variant proposed in [16]. The EAST algorithm learns an HLCM and refers to the method of the same name proposed in [11]. The BI algorithm learns an HLCM and refers to the method of the same name presented in [13]. The MSPN algorithm learns a mixed sum-product network and refers to the method presented in [46]. Finally, the IL and CIL algorithms are the methods proposed in this paper. For CIL, we used α values of 1 and 10. The goal of using these values was to evaluate whether an increase in α produced better results or simply an increase in computational complexity.

To conduct the study, we used 15 real-world datasets of varied dimensionalities (S) and sample sizes (N). The majority of these datasets have been used in previous studies. We also added several publicly available datasets from the UCI repository[2] for a balanced set of examples in terms of S and N. None of these datasets provided expert knowledge to set Bayesian priors. For this reason, we set the prior parameters of the IL and CIL models using an empirical Bayes approach.

[2]https://archive.ics.uci.edu/ml/index.php

As a performance measure, we used the 10-fold cross-validated predictive log-likelihood (CVPLL). That is, we divided each dataset into 10 equal-sized folds, trained a model on nine of them, and computed the predictive log-likelihood on the remaining fold. Tables 1 and 2 display the average results for the CVPLL and learning time, respectively. The maximum time allowed per fold was 24 hours. For a better comparison between IL and CIL, we colored in blue those results from CIL that were in second place behind IL. Our objective was to evaluate whether CIL was able to produce top results in less time than IL. Due to space limitations, we were unable to place the standard deviation values in the tables. This informations is provided in the supplementary material.

Tables 1 and 2 indicate that IL and CIL produced competitive results in terms of both model quality and execution time. Table 1 indicates that, IL and CIL had the best performance on 5 out of 15 datasets. In addition, CIL with $\alpha = 10$ was in second place on another dataset. EAST and IL were unable to compete on high-dimensional datasets due to their computational complexity. In contrast, both BI and CIL were able to produce competitive results on all types of datasets. Finally, in terms of speed, the MSPN algorithm was the fastest of the seven methods evaluated, followed by CIL with $\alpha = 1$.

It can be observed that there were no substantial score differences between IL and CIL. This observation is independent of the selected α value because the MI estimation is exact for discrete data (see Section III-C). In fact, for all datasets except for *Car-evaluation*, the results of CIL with $\alpha = 1$ were almost identical to those of CIL with $\alpha = 10$ and IL.

B. COMPARATIVE STUDY ON CONTINUOUS DATA

The algorithms we considered in this comparative study are summarized as follows. The KDE algorithm corresponds to the kernel density method proposed in [48]. The GS algorithm corresponds to the method proposed by

TABLE 2. Average 10-fold cross-validated learning time (in seconds) for various discrete datasets (rows) and algorithms (columns).

	m	N	BIN	EAST	BI	MSPN	IL	CIL ($\alpha = 1$)	CIL ($\alpha = 10$)
HIV-test	4	428	2.06	0.66	0.57	**0.30**	1.22	0.51	1.00
House-building	4	1185	6.92	1.11	1.19	**0.25**	4.13	1.61	3.24
Hayes-roth	5	160	1.03	2.15	1.71	1.50	1.45	**0.41**	0.99
Balance-scale	5	625	5.70	7.94	5.63	5.28	3.51	**2.23**	4.19
Car-evaluation	7	1728	21.18	243.66	38.83	**4.04**	32.09	7.29	27.85
Nursery	9	12960	403.18	6977.54	496.93	**6.23**	800.37	44.87	449.10
Breast-cancer	10	277	4.13	44.63	8.50	**3.86**	30.86	9.05	23.25
Vote	17	232	13.17	96.00	13.77	3.61	64.31	**2.42**	12.71
Mushroom	19	5644	595.59	21124.53	968.38	**56.56**	6114.58	193.81	974.22
Pascal-Voc-2007	20	10425	378.12	8520.83	310.81	**82.51**	17685.83	616.19	2364.90
Spect-heart	23	267	18.76	328.52	12.84	8.46	237.04	**4.66**	27.80
Alarm	36	1000	231.82	4592.03	67.42	75.98	6209.85	**40.67**	293.53
Coil-42	42	5822	670.53	27531.16	254.00	**33.55**	– – –	261.85	2446.87
News-100	100	16242	3714.78	– – –	2537.71	**142.22**	– – –	4809.26	38156.47
Webkb-336	336	1038	2505.02	– – –	**1244.90**	2139.86	– – –	2266.56	22448.31

The winner in each row is indicated in bold. Results from the incremental learner (IL) and constrained incremental learner (CIL) that outperformed those from existing methods without being in first place ar colored in blue. Dashed lines indicate that the algorithm was unable to complete a fold in 24 hours.

TABLE 3. Average 10-fold cross-validated predictive log-likelihood for various continuous datasets (rows) and algorithms (columns).

	m	N	KDE	GS	GEAST	MSPN	IL	CIL ($\alpha = 1$)	CIL ($\alpha = 10$)
Iris	4	150	-50.92	-47.24	**-36.71**	-243.08	-51.32	-63.45	-51.32
Buddymove	6	249	-663.69	-697.21	-671.54	-1258.76	**-662.24**	-675.27	**-662.25**
Yeast	8	1484	1556.29	1407.45	**1921.62**	658.60	1175.08	1175.08	1175.23
Glass	9	214	-5.31	-59.09	**74.98**	-133.78	-2.54	-47.40	-33.58
Ilpd	9	579	-1834.17	-1495.48	-1567.17	-2125.03	**-1443.60**	-1772.20	-1448.62
Alcohol	10	125	-611.08	-620.97	-553.53	-1544.25	**-493.29**	-527.98	-498.38
Wine	13	178	-365.86	-399.20	**-360.01**	-1093.08	-400.20	-406.13	-398.06
Leaf	14	340	694.85	285.34	**902.82**	-2412.34	603.59	484.33	619.96
NBA	18	441	-694.60	-680.86	**-399.21**	-2537.31	-660.75	-762.68	-681.09
Vehicle	18	846	-5312.55	-4968.41	**-4547.36**	-9544.40	-4896.24	-5304.40	-4982.69
Wdbc	30	569	360.47	-533.95	**1168.41**	-4383.39	721.22	353.83	615.81
Ionosphere	33	351	-657.48	**-617.47**	– – –	-6579.40	-794.62	-877.04	-808.51
Waveform-noise	40	5000	-32643.86	– – –	– – –	-30816.14	**-30370.28**	-31238.63	-30580.09
100-plants	63	1600	**31175.92**	22236.09	– – –	-13220.36	28239.13	28193.23	28197.25
Geo-music	70	1059	-10029.05	– – –	– – –	**-7584.17**	– – –	-10749.99	-9005.82

The winner in each row is indicated in bold. Results from the incremental learner (IL) and constrained incremental learner (CIL) that outperformed other methods without being in first place are colored blue. Dashed lines indicate that the algorithm was unable to complete a fold in 24 hours.

Galimberti and Soffritti in [15], which produces a forest of unconnected LCMs whose leaf variables are all Gaussian. The GEAST algorithm [12] refers to the Gaussian version of the EAST algorithm. The MSPN algorithm is the same method of the previous comparative study.

As with discrete LTMs, we ran experiments on 15 real-world datasets taken from the UCI repository[2] and state-of-the-art papers. We also used CVPLL as the evaluation measure and allowed a maximum learning time of 24 hours per fold. In addition, given the absence of expert knowledge, we used an empirical Bayes approach to set the prior parameters of the IL and CIL models. Tables 3 and 4 display the average results of this study. As with discrete LTMs, we were unable to place the standard deviation values in the tables due to space limitations; however, this information is provided in the supplementary material.

Tables 3 and 4 indicate that in terms of model quality, GEAST and IL had the highest performance. The performance of IL was similar to that in the discrete case in terms of the number of top results: IL had the best performance on 4 out of 15 datasets. However, it was outperformed by the Gaussian version of EAST, which performed considerably better than its discrete counterpart, achieving the best performance on 8 out of 15 datasets. However, GEAST still had the same computational problems as with high-dimensional datasets and was unable to complete in four of the experiments. In terms of speed, the KDE algorithm was the fastest method on all of the datasets. However, this high performance in terms of execution time was not accompanied by high performance in terms of model quality.

Although there were no substantial differences between the performance of IL and CIL with discrete data, the opposite behavior was observed with continuous data. More specifically, there was only one dataset (*Buddymove*) for which CIL with $\alpha = 1$ resulted in the same performance as IL. This coincides with our intuition from Section III-C. When handling continuous data, we must rely on approximate MI methods, which may result in suboptimal solutions (we used

TABLE 4. Average 10-fold cross-validated learning time (in seconds) for various continuous datasets (rows) and algorithms (columns).

	m	N	KDE	GS	GEAST	MSPN	IL	CIL ($\alpha = 1$)	CIL ($\alpha = 10$)
Iris	4	150	**0.00**	3.34	12.93	1.28	0.63	0.39	0.77
Buddymove	6	249	**0.00**	25.67	213.78	4.48	2.80	0.58	2.54
Yeast	8	1484	**0.08**	251.19	1540.01	22.80	29.49	4.15	15.24
Glass	9	214	**0.01**	59.03	486.43	9.85	10.88	0.65	3.99
Ilpd	9	579	**0.02**	442.72	1757.13	15.89	43.16	2.49	29.00
Alcohol	10	125	**0.00**	76.59	316.98	5.36	6.46	1.07	3.13
Wine	13	178	**0.01**	126.85	795.98	13.89	21.59	0.94	4.40
Leaf	14	340	**0.02**	289.41	3941.66	9.76	61.19	0.59	12.77
NBA	18	441	**0.03**	2336.86	13608.33	59.87	257.69	5.61	49.60
Vehicle	18	846	**0.07**	7198.36	74520.49	109.92	510.65	11.71	116.55
Wdbc	30	569	**0.08**	9582.42	50784.86	52.60	2436.78	8.90	57.90
Ionosphere	33	351	**0.03**	4231.51	– – –	69.90	1433.53	3.59	46.41
Waveform-noise	40	5000	**3.31**	– – –	– – –	17.72	45831.88	383.15	1314.81
100-plants	63	1600	**0.68**	27992.68	– – –	604.52	6936.40	61.20	58.20
Geo-music	70	1059	**0.37**	– – –	– – –	724.22	– – –	79.11	2186.30

The winner in each row is indicated in bold. Results from the incremental learner (IL) and constrained incremental learner (CIL) that outperformed other methods without being in first place are colored blue. Dashed lines indicate that the algorithm was unable to complete a fold in 24 hours.

the k-nearest neighbors approach proposed in [40]). As previously mentioned in Section III-C, increasing the number of candidate models in each step improves the quality of the results, as illustrated in Table 3. There were only small differences between IL and CIL with $\alpha = 10$, which makes CIL particularly promising for high-dimensional data for which IL takes a much longer time to complete.

C. SPANISH LIVING CONDITIONS DATASET

The Spanish Living Conditions Survey[3] is an annual survey conducted by Eurostat to analyze income, poverty, social exclusion, and livfing conditions in the European Union. The objective of this survey is the cross-sectional study of Spanish households. This information will be later used in a comparative study that includes other European countries. In this experiment, we used data from the last published survey, from 2018. Our objective for using these data was twofold: (i) to test the performance of our proposed methods on a mixed dataset; and (ii) to obtain knowledge from this study by learning a latent forest.

After preprocessing, the data consisted of 13,222 instances and 22 observed variables. Variables were organized into two main groups: (i) house quality (more specifically, the location, furniture, and appliances); and (ii) family economy, which includes household income and the ability to afford essential services and products. For a better comparison of different types of households according to the number of individuals that constitute them and the individuals' ages, the concept of equivalent income was used. This concept standardizes households according to the number of equivalent consumption units that constitute them. The number of units in each household was determined using the OECD-modified equivalence scale.

Most LTM algorithms used in the comparative studies of Sections IV-A and IV-B are specific to discrete or continuous data. Therefore, in this experiment we evaluated the performance of IL and CIL with respect to other methods that could also handle mixed datasets such as KDE, MSPN and LCM. The LCM algorithm starts with a latent class model of cardinality 2 and greedily increases its cardinality until the score ceases to improve.

As in the previous experiments, we lacked an expert who could provide us with prior information about the observed variables in the model. Instead, we based our prior information on another study: the Continuous Survey of Spanish Households.[4] This annual survey collects information from more than 100,000 households, including the number of family members, number of rooms in the house, and its tenure regime. We used this information to set the parameter values of the prior distributions from the corresponding variables in our study. For variables in the living conditions survey that did not have an analogous variable (e.g., *equivalent_income*, *problem_pollution*, *afford_meal*), we set their prior parameters using an empirical Bayes approach. A more detailed explanation of each variable and the specification of its prior parameter values are provided in the supplementary material.

The comparative results of this experiment (both CVPLL scores and execution times) are displayed on Table 5. As with previous studies, we were unable to place the standard deviation values in the table due to space limitations; however, this information is provided in the supplementary material. Table 5 indicates that even though the MSPN and KDE algorithms were much faster than the rest, their resulting models were unable to correctly represent the underlying probability distribution. For the proposed methods, in terms of model quality there was a clear underperformance by CIL with $\alpha = 1$, which coincides with our reasoning on the inherent problems of approximate MI methods. IL and CIL with $\alpha = 10$ produced almost identical results, although CIL obtained a slightly better average CVPLL. In terms of speed,

[3]https://www.ine.es/uc/nnQwhLRQ

[4]https://www.ine.es/uc/4C5vfjY1

TABLE 5. Results for the Spanish living conditions dataset.

	LCM	KDE	MSPN	IL	CIL ($\alpha = 1$)	CIL ($\alpha = 10$)
CVPLL	-26932.06	-31227.11	-49206.72	-23724.60	-24322.80	**-23719.42**
Time (s)	256.16	**12.18**	13.85	17687.78	900.12	2582.00

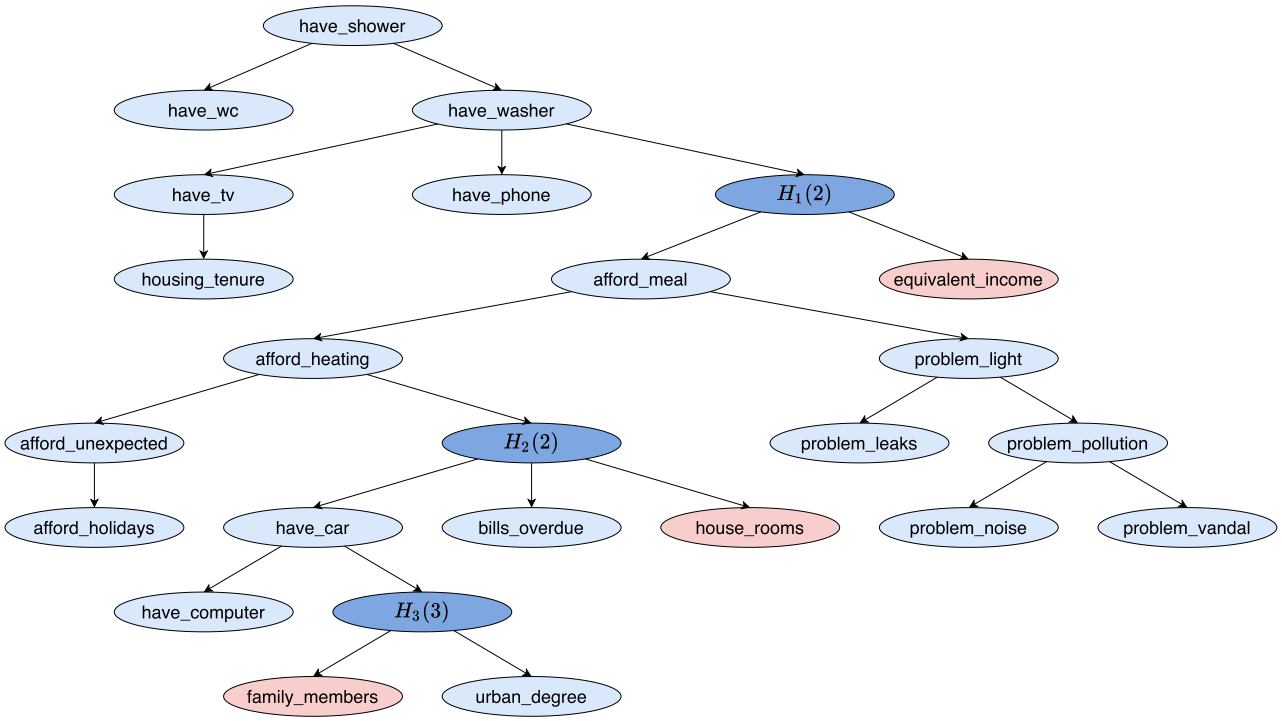


FIGURE 3. Latent forest learned by the constrained incremental learner (CIL) algorithm with an α value of 10 for the 2018 Spanish living conditions dataset.

CIL with $\alpha = 1$ behaved similarly to previous studies and was the fastest of the three.

Fig. 3 displays the latent forest inferred by CIL with $\alpha = 10$. We selected this model due to its good performance in terms of score and speed. The learned forest was formed by a single tree with three latent variables. Theoretically, each latent variable represents a soft partition of data, where the meaning of each partition is determined by the conditional probability distributions of the latent variable's children. To analyze this model and its latent variables, we used Genie,[5] a common tool for interpreting BNs. Some of our findings are as follows.

H_1 divides households into two groups: those with an income and those without an income. Unsurprisingly, those without an income are more likely to be unable to afford a meal (the probability of being unable to afford a meal increases from 3% to 14%).

H_2 divides households into two groups according to the number of rooms in the house: small-medium-sized houses (whose conditional Gaussian distribution has a mean value of 4.29 with a variance of 0.69) and large-sized houses (mean value of 6.00 with a variance of 0.00). In addition, H_2 relates the size of the house to other aspects, such as the possession of a car and delays in paying house bills. We observed that families with a larger house had a smaller probability of delay (3% versus 7%).

H_3 divides households into three groups according to the number of individuals living in the house: single family, couple, and family with children or relatives (more than two family members). H_3 also relates family size to the degree of urbanization of the city. We observed that larger families usually preferred less populated areas rather than highly populated areas (48% versus 54%).

There were other interesting aspects of the model that were not directly related to the latent variables. For example, we observed that houses located in highly polluted areas had an increased probability of vandalism (from 16% to 55%) and noise (from 11% to 37%).

The results of this study demonstrate the ability of latent forests to group variables in meaningful ways and extract insightful information. Models generated by CIL with $\alpha = 10$ as well as the remaining methods can be found in our GitHub repository. They are in XDSL format, which is directly supported by Genie. as well as the remaining methods can be found in our GitHub repository.

V. CONCLUSION

In this paper, we propose an incremental method that, in combination with the VB framework, is able to learn forests

[5]https://www.bayesfusion.com/genie/

of LTMs from data composed of discrete and/or continuous variables. Considering that directly searching this space requires the evaluation of a large number of candidate structures, a constrained variant that only evaluates the most prominent α candidates of each iteration is also developed. As demonstrated by the experimental results, the constrained approach is a valid alternative to the brute-force search method. It returns almost identical results for discrete data experiments and displays only slightly worse performance for continuous data. We believe that these differences are caused by the use of approximate MI estimation, and demonstrate that they can be reduced by increasing α.

Although restricting the structure search to an incremental process limits the space of possible models, our experiments demonstrate that this restriction still leads to competitive results. Furthermore, due to this restriction, our proposed method is able to work effectively with both low- and high-dimensional dataset. Other methods (e.g., GEAST) may perform better by considering a larger number of candidate models at each iteration, but their computational complexity makes them infeasible when the number of variables is large. Additionally and in contrast to current LTM methods, by taking advantage of the variational Bayesian framework, we are able to provide a means incorporating prior knowledge and produce a superior evaluation of the generalization properties of a model given data.

There are various future research directions. First, inspired by factor analysis, our methods can be modified to work with Gaussian latent variables, where the number of factors can be estimated analogously to the cardinality of discrete latent variables. Second, we can extend parameter estimation by using more flexible variational families [49], [50] and nonconjugate priors [51], but at the cost of a more difficult variational optimization problem. Third, we can add more flexibility to the structure search by replacing the arc addition operator with a variational version of Friedman's structural EM [52]. This can allow us to efficiently add and remove arcs without relying on an incremental process. Finally, selecting the most probable latent forest may not be suitable when we are interested in quantifying our confidence in the forest structure or when we have a low number of data instances. For these cases, Bayesian inference can also be introduced into the search process by averaging over the set of possible latent forests (i.e., by using Bayesian model averaging [53], [54]). For this, we can define a Markov chain over the space of possible latent forests (given our search operators) and then do a random walk in this Markov chain.

REFERENCES

[1] R. Mourad, C. Sinoquet, N. L. Zhang, T. Liu, and P. Leray, "A survey on latent tree models and applications," *J. Artif. Intell. Res.*, vol. 47, pp. 157–203, May 2013.

[2] P. Zwiernik, "Latent tree models," in *Handbook of Graphical Models*. Boca Raton, FL, USA: CRC Press, 2018, ch. 11, pp. 267–290.

[3] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA, USA: Morgan Kaufmann, 1988.

[4] H. Langseth and T. D. Nielsen, "Latent classification models for binary data," *Pattern Recognit.*, vol. 42, no. 11, pp. 2724–2736, Nov. 2009.

[5] Y. Wang, N. L. Zhang, T. Chen, and L. K. M. Poon, "LTC: A latent tree approach to classification," *Int. J. Approx. Reasoning*, vol. 54, no. 4, pp. 560–572, Jun. 2013.

[6] T. Nimmagadda and A. Anandkumar, "Multi-object classification and unsupervised scene understanding using deep learning features and latent tree probabilistic models," 2015, *arXiv:1505.00308*. [Online]. Available: http://arxiv.org/abs/1505.00308

[7] P. Chen, N. L. Zhang, T. Liu, L. K. M. Poon, Z. Chen, and F. Khawar, "Latent tree models for hierarchical topic detection," *Artif. Intell.*, vol. 250, pp. 105–124, Sep. 2017.

[8] Y. Wang, N. L. Zhang, and T. Chen, "Latent tree models and approximate inference in Bayesian networks," *J. Artif. Intell. Res.*, vol. 32, pp. 879–900, Aug. 2008.

[9] X. Li, Z. Chen, L. K. Poon and N. L. Zhang, "Learning latent superstructures in variational autoencoders for deep multidimensional clustering," presented at the 6th Int. Conf. Learn. Represent., New Orleans, LA, USA, May 2019.

[10] N. L. Zhang, "Hierarchical latent class models for cluster analysis," *J. Mach. Learn. Res.*, vol. 5, pp. 697–723, Dec. 2004.

[11] T. Chen, N. L. Zhang, T. Liu, K. M. Poon, and Y. Wang, "Model-based multidimensional clustering of categorical data," *Artif. Intell.*, vol. 176, no. 1, pp. 2246–2269, Jan. 2012.

[12] L. K. M. Poon, N. L. Zhang, T. Liu, and A. H. Liu, "Model-based clustering of high-dimensional data: Variable selection versus facet determination," *Int. J. Approx. Reasoning*, vol. 54, no. 1, pp. 196–215, Jan. 2013.

[13] T. F. Liu, N. L. Zhang, P. Chen, A. H. Liu, L. K. Poon, and Y. Wang, "Greedy learning of latent tree models for multidimensional clustering," *Mach. Learn.*, vol. 98, nos. 1–2, pp. 301–330, 2015.

[14] N. L. Zhang and T. Kocka, "Efficient learning of hierarchical latent class models," in *Proc. 16th IEEE Int. Conf. Tools with Artif. Intell.*, Nov. 2004, pp. 585–593.

[15] G. Galimberti and G. Soffritti, "Model-based methods to identify multiple cluster structures in a data set," *Comput. Statist. Data Anal.*, vol. 52, no. 1, pp. 520–536, Sep. 2007.

[16] S. Harmeling and C. K. I. Williams, "Greedy learning of binary latent trees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 6, pp. 1087–1097, Jun. 2011.

[17] R. Mourad, C. Sinoquet, and P. Leray, "A hierarchical Bayesian network approach for linkage disequilibrium modeling and data-dimensionality reduction prior to genome-wide association studies," *BMC Bioinf.*, vol. 12, no. 1, pp. 1–20, Dec. 2011.

[18] M. Drton, S. Lin, L. Weihs, and P. Zwiernik, "Marginal likelihood and model selection for Gaussian latent tree and forest models," *Bernoulli*, vol. 23, no. 2, pp. 1202–1232, May 2017.

[19] J. A. Lake, "Reconstructing evolutionary trees from DNA and protein sequences: Paralinear distances.," *Proc. Nat. Acad. Sci. USA*, vol. 91, no. 4, pp. 1455–1459, Feb. 1994.

[20] M. J. Choi, V. Y. Tan, A. Anandkumar, and A. S. Willsky, "Learning latent tree graphical models," *J. Mach. Learn. Res.*, vol. 12, pp. 1771–1812, 2011.

[21] F. Huang, I. Perros, R. Chen, J. Sun, and A. Anandkumar, "Guaranteed scalable learning of latent tree models," presented at the 35th Conf. Uncertain. Artif. Intel., Tel Aviv, Israel, Jul. 2019.

[22] H. Attias, "A variational Baysian framework for graphical models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 209–215.

[23] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA, USA: MIT Press, 2009.

[24] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[25] P. F. Lazarsfeld and N. W. Henry, *Latent Structure Analysis*. Boston, MA, USA: Houghton Mifflin, 1968.

[26] D. J. Bartholomew, M. Knott, and I. Moustaki, *Latent Variable Models and Factor Analysis: A Unified Approach*. New York, NY, USA: Wiley, 2011.

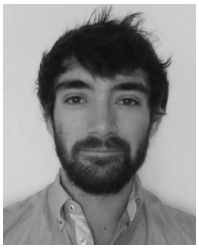
[27] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2773–2832, 2014.

[28] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. B, Methodol.*, vol. 39, no. 1, pp. 1–38, 1977.

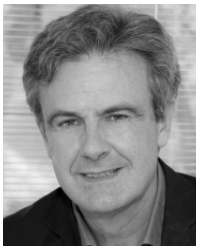
[29] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, Mar. 1978.

[30] W. H. Jefferys and J. O. Berger, "Ockham's razor and Bayesian analysis," *Amer. Sci.*, vol. 80, no. 1, pp. 64–72, 1992.

[31] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," *Mach. Learn.*, vol. 20, no. 3, pp. 197–243, Sep. 1995.
[32] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, Apr. 1970.
[33] A. Gelfand and A. Smith, "Sampling-based approaches to calculating marginal densities," *J. Amer. Statist. Assoc.*, vol. 85, no. 410, pp. 398–409, 1990.
[34] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *J. Amer. Stat. Assoc.*, vol. 112, no. 518, pp. 859–877, Apr. 2017.
[35] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.
[36] M. J. Beal and Z. Ghahramani, "Variational Bayesian learning of directed graphical models with hidden variables," *Bayesian Anal.*, vol. 1, no. 4, pp. 793–831, Dec. 2006.
[37] J. Winn and C. M. Bishop, "Variational message passing," *J. Mach. Learn. Res.*, vol. 6, pp. 661–694, Apr. 2005.
[38] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing Bayesian network structure learning algorithm," *Mach. Learn.*, vol. 65, no. 1, pp. 31–78, Oct. 2006.
[39] J. A. Gámez, J. L. Mateo, and J. M. Puerta, "Learning Bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood," *Data Mining Knowl. Discovery*, vol. 22, nos. 1–2, pp. 106–148, Jan. 2011.
[40] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 69, no. 6, Jun. 2004, Art. no. 066138.
[41] Y.-I. Moon, B. Rajagopalan, and U. Lall, "Estimation of mutual information using kernel density estimators," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 52, no. 3, pp. 2318–2321, Sep. 1995.
[42] S. Guindon and O. Gascuel, "A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood," *Systematic Biol.*, vol. 52, no. 5, pp. 696–704, Oct. 2003.
[43] D. M. Chickering and D. Heckerman, "Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables," *Mach. Learn.*, vol. 29, nos. 2–3, pp. 181–212, 1997.
[44] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis*. Boca Raton, FL, USA: CRC Press, 2013.
[45] G. Casella, "An introduction to empirical Bayes data analysis," *Amer. Statistician*, vol. 39, no. 2, pp. 83–87, May 1985.
[46] A. Molina, A. Vergari, N. Di Mauro, S. Natarajan, F. Esposito, and K. Kersting, "Mixed sum-product networks: A deep architecture for hybrid domains," presented at the 32nd AAAI Conf. Artif. Intell., New Orleans, LA, USA, Feb. 2018.
[47] H. Poon and P. Domingos, "Sum-product networks: A new deep architecture," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCV Workshops)*, Nov. 2011, pp. 337–346.
[48] Q. Li and J. Racine, "Nonparametric estimation of distributions with categorical and continuous data," *J. Multivariate Anal.*, vol. 86, no. 2, pp. 266–292, Aug. 2003.
[49] D. Barber and W. Wiegerinck, "Tractable variational structures for approximating graphical models," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 183–189.
[50] C. M. Bishop, N. D. Lawrence, T. Jaakola, and M. I. Jordan, "Approximating posterior distributions in belief networks using mixtures," in *Proc. Adv. Neural Inf. Process. Syst.*, 1998, pp. 416–422.
[51] C. Wang and D. M. Blei, "Variational inference in nonconjugate models," *J. Mach. Learn. Res.*, vol. 14, pp. 1005–1031, Jan. 2013.
[52] N. Friedman, "The Bayesian structural EM algorithm," in *Proc. 14th Conf. Uncertain. Artif. Intell.*, 1998, pp. 129–138.
[53] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, "Bayesian model averaging: A tutorial," *Stat. Sci.*, pp. 382–401, Nov. 1999.
[54] T. M. Fragoso, W. Bertoli, and F. Louzada, "Bayesian model averaging: A systematic review and conceptual classification," *Int. Stat. Rev.*, vol. 86, no. 1, pp. 1–28, Apr. 2018.



FERNANDO RODRIGUEZ-SANCHEZ received the B.Sc. degree in software engineering from the Universidad de Oviedo, Spain, in 2015, and the M.Sc. degree in artificial intelligence from the Universidad Politécnica de Madrid (UPM), in 2016. He is currently pursuing the Ph.D. degree with the Artificial Intelligence Department, UPM, where he collaborates on the Human Brain Project. His research interests include the areas of probabilistic graphical models, cluster analysis, decision theory, and probabilistic programming.



PEDRO LARRAÑAGA (Member, IEEE) received the M.Sc. degree in mathematics (statistics) from the University of Valladolid and the Ph.D. degree in computer science from the University of the Basque Country (excellence award). He has been a Full Professor in computer science and artificial intelligence with the Universidad Politécnica de Madrid (UPM) since 2007. Before moving to UPM, his academic career developed at the University of the Basque Country (UPV-EHU) at several faculty ranks, i.e., Assistant Professor (1985–1998), Associate Professor (1998–2004), and Full Professor (2004–2007). He received the distinction for Full Professor in 2003. He has published over 200 papers in high-impact factor journals. He has supervised over 30 Ph.D. theses. His research interests include the areas of probabilistic graphical models, metaheuristics for optimization, data mining, classification models, and real applications, such as biomedicine, bioinformatics, neuroscience, industry 4.0, and sports. He is a Fellow of the European Association for Artificial Intelligence since 2012, and a Fellow of the Academia Europea since 2018. He has received the 2013 Spanish National Prize in Computer Science and the prize of the Spanish Association for Artificial Intelligence in 2018, and the Amity University Machine Learning Award (India) in 2020.

CONCHA BIELZA (Member, IEEE) received the M.S. degree in mathematics from the Universidad Complutense de Madrid, Madrid, Spain, in 1989, and the Ph.D. degree in computer science from the Universidad Politécnica de Madrid (UPM), Madrid, in 1996 (extraordinary doctorate award). Since 2010, she has been a Full Professor of Statistics and Operations Research with the Departamento de Inteligencia Artificial, UPM. Her research interests are primarily in the areas of machine learning, probabilistic graphical models, decision analysis, metaheuristics for optimization, and real applications, such as biomedicine, bioinformatics, neuroscience, industry 4.0, and sports. She has published more than 120 papers in impact factor journals and has supervised 17 Ph.D. theses. She was awarded the 2014 UPM Research Prize and the Amity University Machine Learning Award (India) in 2020.