# Automatic Bayesian Density Analysis

**Antonio Vergari**
antonio.vergari@tue.mpg.de
MPI-IS, Tuebingen, Germany

**Alejandro Molina**
molina@cs.tu-darmstadt.de
TU Darmstadt, Germany

**Robert Peharz**
rp587@cam.ac.uk
University of Cambridge, UK

**Zoubin Ghahramani**
zoubin@cam.ac.uk
University of Cambridge, UK
Uber AI Labs, USA

**Kristian Kersting**
kersting@cs.tu-darmstadt.de
TU Darmstadt, Germany

**Isabel Valera**
isabel.valera@tue.mpg.de
MPI-IS, Tuebingen, Germany

**Abstract**

Making sense of a dataset in an automatic and unsupervised fashion is a challenging problem in statistics and AI. Classical approaches for exploratory data analysis are usually not flexible enough to deal with the uncertainty inherent to real-world data: they are often restricted to fixed latent interaction models and homogeneous likelihoods; they are sensitive to missing, corrupt and anomalous data; moreover, their expressiveness generally comes at the price of intractable inference. As a result, supervision from statisticians is usually needed to find the right model for the data. However, since domain experts are not necessarily also experts in statistics, we propose Automatic Bayesian Density Analysis (ABDA) to make exploratory data analysis accessible at large. Specifically, ABDA allows for automatic and efficient missing value estimation, statistical data type and likelihood discovery, anomaly detection and dependency structure mining, on top of providing accurate density estimation. Extensive empirical evidence shows that ABDA is a suitable tool for automatic exploratory analysis of mixed continuous and discrete tabular data.

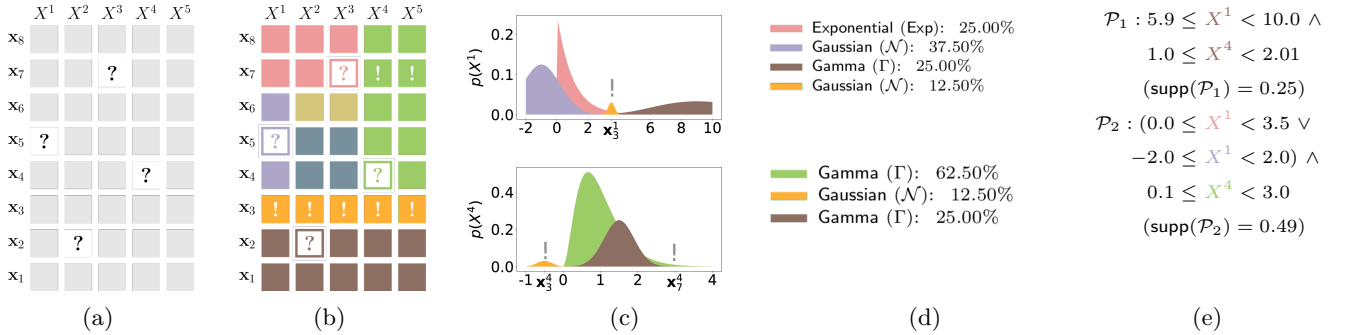
## Introduction

"Making sense" of a dataset—a task often referred to as *data understanding* or *exploratory data analysis*—is a fundamental step that precedes and guides a classical machine learning (ML) pipeline. Without domain experts' background knowledge, a dataset might remain nothing but a list of numbers and arbitrary symbols. On the other hand, without statisticians' supervision, processing the data and extracting useful models from it might go beyond the ability of domain experts who might not be experts in ML or statistics. Therefore, in times of abundant data, but an insufficient number of statisticians, methods which can "understand" and "make sense" of a dataset *with minimal or no supervision* are in high demand.

The idea of machine-assisted data analysis has been pioneered by The Automatic Statistician project (Duvenaud et al. 2013; Lloyd et al. 2014) which proposed to automate model selection for regression and classification tasks via compositional kernel search. Analogously, but with a clear focus on performance optimization, AutoML frameworks (Guyon et al. 2016) automate the choice of supervised ML models for a task-dependent loss. In contrast, we address model selection for a *fully unsupervised task*, with the aim of assisting domain experts in exploratory data analysis, providing them with a probabilistic framework to perform efficient inference and gain useful insights from the data in an automatic way.

In principle, a suitable unsupervised learning approach *to find out what is in the data* is *density estimation* (DE). In order to perform DE on data in a tabular format, a practitioner would first try to heuristically infer the *statistical types* of the data features, e.g., real, positive, numerical and nominal data types (Valera, Pradier, and Ghahramani 2017; Valera and Ghahramani 2017). Based on this, she would need to make assumptions about their distribution, i.e., selecting a *parametric likelihood model* suitable for each marginal, e.g., Gaussian for real, Gamma for positive, Poisson for numerical and Categorical for nominal data. Then, she could start investigating the global interactions among them, i.e., determining the *statistical dependencies* across features. In this process, she would also likely need to deal with *missing values* and reason whether the data may be corrupted or contain *anomalies*.

Unfortunately, classical approaches to DE, even if ubiquitous in ML applications, are still far from delivering automatic tools suitable for performing *all* these steps on *real-world data*. First, general-purpose density estimators usually assume the statistical data types and the likelihood models to be *known* a priori and *homogeneous* across random variables (RVs) or mixture model components (Valera and Ghahramani 2017). Indeed, the standard approach is still to generally treat *all* continuous data as (mixtures of) Gaussian RVs and discrete data as categorical variables. Second, they either assume "shallow" dependency structures that might be too simplistic to capture real-world statistical dependencies or use "deeper" latent structures which cannot be easily learned. As a result, they lack enough flexibility to deal with fine grain statistical dependencies, and to




(a) (b) (c) (d) (e)

Figure 1: Automatic exploratory analysis with ABDA on a tabular dataset comprising samples from mixed continuous and discrete features $\mathbf{X} = \{X^1, \ldots, X^5\}$, potentially containing missing values (denoted as "?") (**1a**). The latent dependency structure inferred by ABDA induces a *hierarchical partition* over samples and features (i.e., a hierarchical co-clustering) given a learned Sum-Product Network (SPN) structure (**1b**). Statistical data types and likelihood models are discovered by estimating each feature distribution as a mixture model over a dictionary of suitable and interpretable parametric model, e.g., Gaussian for real data, Gamma, Exponential distributions for positive data in (**1c**-**1d**). ABDA can efficiently impute missing entries as the most probable values given a partition in 1b by SPN inference routines. Anomalous entries (denoted as "!" in **1b**-**1c**), on the other hand, are presented to the user as low likelihood samples that are relegated to micro-clusters (e.g., $\mathbf{x}_3$) or to the distribution tails (e.g., $\mathbf{x}^4_7$ and $\mathbf{x}^5_7$). Moreover, ABDA allows to automatically discover complex dependency patterns, e.g. conjunctions of confidence intervals, by exploiting the correlations over the induced hierarchical co-clustering (**1e**).

be robust to corrupt data and outliers, especially when a maximum likelihood learning approach is used.

A Bayesian treatment of DE, on the other hand, often struggles to scale up to high-dimensional data and approximate inference routines are needed (Ghahramani and Beal 2000). CrossCat (Mansinghka et al. 2016) shows a clear example of this trade-off. Even though CrossCat models sub-populations in the data with context-specific latent variables, this latent structure is limited to a two-layer hierarchy, and even so inference routines have to be approximated for efficiency. Moreover, it is still limited to fixed and homogeneous statistical data types and therefore, likelihood models.

The latent variable matrix factorization model (ISLV) introduced in (Valera and Ghahramani 2017) is the first attempt to overcome this limitation by *modeling uncertainty over the statistical data types* of the features. However, like CrossCat, it can only perform inference natively in the transductive case, i.e., to data available during training. While ISLV allows one to infer the data type of a feature, this approach still uses a single *ad-hoc* likelihood function for each data type.

Recently, Mixed Sum-Product Networks (MSPNs) (Molina et al. 2018) have been proposed as deep models for heterogeneous data able to perform tractable inference also in the inductive scenario, i.e., on completely unobserved test data. Indeed, MSPNs can exploit context specific independencies to learn latent variable hierarchies that are deeper than CrossCat. However, MSPNs assume piecewise-linear approximations as likelihood models for both continuous and discrete RVs, which are not as interpretable as parametric distributions and also require continuous RVs to undergo a delicate discretization process. As a result, MSPNs are highly prone to overfitting, and learning them via maximum likelihood results into a lack of robustness.

In this paper, we leverage the above models' advantages while addressing their shortcomings by proposing *Automatic Bayesian Density Analysis* (ABDA). Specifically, ABDA relies on sum-product networks (SPNs) to capture statistical dependencies in the data at different granularity through a hierarchical co-clustering. This rich latent structure is learned in an adaptive way, which automates the selection of adequate likelihood models for each data partition, and thus extends ISLV uncertainty modeling over statistical types. As a result, ABDA goes beyond standard density estimation approaches, qualifying as the first approach to *fully automate exploratory analysis for heterogeneous tabular data at large.*

As illustrated in Fig. 1, ABDA allows for:

**i)** inference for *both* the statistical data types *and* (parametric) likelihood models;

**ii)** robust estimation of missing values;

**iii)** detection of corrupt or anomalous data;

**iv)** automatic discovery of the statistical dependencies and local correlation structures in the data.

ABDA relies on Bayesian inference through Gibbs sampling, allowing us to robustly measure uncertainties at performing all the above tasks. In our extensive experimental evaluation, we demonstrate that ABDA effectively assists domain experts in both transductive and inductive settings. Supplementary material and a reference implementation of ABDA are available at github.com/probabilistic-learning/abda.

## Sum-Product Networks (SPNs)

As SPNs provide the hierarchical latent backbone of ABDA, we will now briefly review them. Please refer to (Peharz et al. 2017) for more details.

**Representation.** An SPN $\mathcal{S}$ over a random vector $\mathbf{X} = \{X^1, \dots, X^D\}$ is a probabilistic model defined via a directed acyclic graph. Each leaf node $\mathsf{L}$ represents a probability distribution function over a single RV $X \in \mathbf{X}$, also called its *scope*. Inner nodes represent either *weighted sums* ($\mathsf{S}$) or *products* ($\mathsf{P}$). For inner nodes, the scope is defined as the union of the scopes of its children. The set of children of a node $\mathsf{N}$ is denoted by $\mathsf{ch}(\mathsf{N})$.

A sum node $\mathsf{S}$ encodes a mixture model $\mathcal{S}_\mathsf{S}(\mathbf{x}) = \sum_{\mathsf{N}\in\mathsf{ch}(\mathsf{S})} \omega_{\mathsf{S},\mathsf{N}} \mathcal{S}_\mathsf{N}(\mathbf{x})$ over sub-SPNs rooted at its children $\mathsf{ch}(\mathsf{S})$. We require that the all children of a sum node share the same variable scope—this condition is referred to as *completeness* (Poon and Domingos 2011). The weights of a sum $\mathsf{S}$ are drawn from the standard simplex (i.e., $\omega_{\mathsf{S},\mathsf{N}} \geq 0$, $\sum_{\mathsf{N}\in\mathsf{ch}(\mathsf{S})} \omega_{\mathsf{S},\mathsf{N}} = 1$) and denoted as $\Omega^\mathsf{S}$. A product node $\mathsf{P}$ defines a factorization $\mathcal{S}_\mathsf{P}(\mathbf{x}) = \prod_{\mathsf{N}\in\mathsf{ch}(\mathsf{P})} \mathcal{S}_\mathsf{N}(\mathbf{x})$ over its children distributions defined over disjoint scopes—this condition is referred to as *decomposability* (Poon and Domingos 2011). The parameters of $\mathcal{S}$ are the set of sum weights $\mathbf{\Omega} = \{\Omega^\mathsf{S}\}_{\mathsf{S}\in\mathcal{S}}$ and the set of all leaf distribution parameters $\{\boldsymbol{H}_\mathsf{L}\}_{\mathsf{L}\in\mathcal{S}}$.

Complete and decomposable SPNs are highly expressive deep models and have been successfully employed in several ML domains (Peharz et al. 2015; Molina, Natarajan, and Kersting 2017; Pronobis, Riccio, and Rao 2017; Vergari, Di Mauro, and Esposito 2018) to capture highly complex dependencies in real-world data. Moreover, they also allow to *exactly* evaluate complete evidence, marginal and conditional probabilities in *linear time* w.r.t. their representation size (Darwiche 2003).

**Latent variable representation.** Since each sum node defines a mixture over its children, one may associate a categorical latent variable (LV) $Z^\mathsf{S}$ to each sum $\mathsf{S}$ indicating a component. This results in a *hierarchical* model over the set of all LVs $\mathbf{Z} = \{Z^\mathsf{S}\}_{\mathsf{S}\in\mathcal{S}}$. Specifically, an assignment to $\mathbf{Z}$ selects an *induced tree* in $\mathcal{S}$ (Zhao, Poupart, and Gordon 2016; Peharz et al. 2017; Vergari et al. 2018), i.e., a tree path $\mathcal{T}$ starting from the root and comprising exactly one child for each visited sum node and all child branches for each visited product node (in green in Fig. 2b). It follows from completeness and decomposability that $\mathcal{T}$ selects a subset of $D$ leaves, in a one-to-one correspondence with $\mathbf{X}$. When conditioned on a particular $\mathcal{T}$, the SPN distribution factorizes as $\prod_d \mathsf{L}^d_{j^d}$, where $\mathsf{L}^d_k$ is the $k^{\text{th}}$ leaf for the $d^{\text{th}}$ RV, and $\mathbf{j} = \{j^1, \dots, j^D\}$ are the indices of the leaves selected by $\mathcal{T}$. The overall SPN distribution can be written as a mixture of such factorized distributions, running over all possible induced trees (Zhao, Poupart, and Gordon 2016). We will use this hierarchical LV structure to develop an efficient Gibbs sampling scheme to perform Bayesian inference for ABDA.

**SPN learning.** Existing SPN learning works focus on learning the SPN parameters given a structure (Gens and Domingos 2012; Trapp et al. 2017; Zhao, Poupart, and Gordon 2016) or jointly learn both the structure and the parameters (Dennis and Ventura 2012; 2015; Peharz, Geiger, and Pernkopf 2013). A particular prominent approach is LearnSPN (Gens and Domingos 2013; Vergari, Di Mauro, and Esposito 2015), which recursively partitions a data matrix using hierarchical co-clustering. In particular, LearnSPN alternates between partitioning features into independent groups, inducing a product node, and clustering the data instances, inducing a sum node. As the base step, univariate likelihood models for single features are induced.

ABDA resorts to the SPN learning technique employed by MSPNs (Molina et al. 2018), as it is the only available approach to build an SPN structure in a likelihood-agnostic way, therefore being suitable for our heterogeneous setting. It performs a partitioning over mixed continuous and discrete data by exploiting a randomized approximation of the Hirschfeld-Gebelein-Rényi Maximum Correlation Coefficient (RDC) (Lopez-Paz, Hennig, and Schölkopf 2013). ABDA employs it to automatically select an *initial*, global LV structure, which is provided to its Bayesian inference routines.

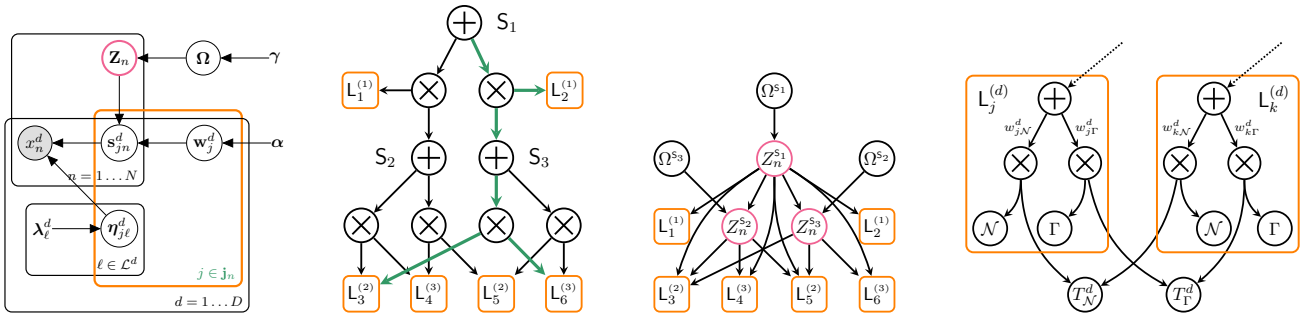
## Automatic Bayesian Density Analysis

For a general discussion, we extend $\mathbf{X}$ to a whole data matrix containing $N$ samples (rows) and $D$ features (columns). RVs which are local to a row (column) receive now a sub-script $n$ (super-script $d$).

Our proposed Automatic Bayesian Density Analysis (ABDA) model can be thought as being organized in two levels: a *global* level, capturing dependencies and correlations among the features, and a level which is *local* w.r.t. each feature, consisting of dictionaries of likelihood models for that feature. This model is illustrated Fig. 2, via its graphical model representation, and corresponding SPN representations. In this hierarchy, the global level captures context specific dependencies via recursive data partitioning, leveraging the LV structure of an SPN. The local level represents context-specific uncertainties about the variable types, conditioned on the global context.

In contrast to classical works on DE, where typically fixed likelihood models are used as mixture components, e.g., see (Poon and Domingos 2011; Vergari, Di Mauro, and Esposito 2015)), ABDA assumes a *heterogeneous* mixture model combining several likelihood models from a user-provided *likelihood dictionary*. This dictionary may contain likelihood models for diverse types of discrete (e.g. Poisson, Geometric, Categorical,...) and continuous (e.g., Gaussian, Gamma, Exponential,...) data. It can be built in a generous automatic way, incorporating arbitrary rich collections of domain-agnostic likelihood functions. Alternatively, its construction can be limited to a sensible subset of likelihood models reflecting domain knowledge, e.g. a Gompertz and Weibull distributions might be suitable for demographics data.

In what follows, assume that for each feature $d$ we have readily selected a dictionary $\{p^d_\ell\}_{\ell\in\mathcal{L}^d}$ of likelihood




(a) Graphical model (b) SPN (c) LV interpretation (d) Type-augmented SPN

Figure 2: **(a)** Plate representation of ABDA, showing the *global* LV (pink) and *local* (orange) levels. **(b)** The SPN representation with likelihood-modeling leaves $L_j^{(d)}$ over features $d$=1,2,3 selected by an induced tree (green), **(c)** and its corresponding hierarchy over LVs $\mathbf{Z}$. **(d)** A global mixture model over Gaussian ($\mathcal{N}$) and Gamma ($\Gamma$) likelihoods interpreted as sub-SPNs sharing global auxiliary LVs $T^d$ across the local likelihood mixture models for feature $d$.

models, indexed by some set $\mathcal{L}^d$. We next describe the process generating $\mathbf{X}$, also depicted in Fig. 2a, in detail.

**Generative model.** The global level of ABDA contains latent vectors $\mathbf{Z}_n$ for each sample $\mathbf{x}_n$, associated with the SPN sum nodes (see previous section). Each LV $Z_n^{\mathsf{S}}$ in $\mathbf{Z}_n$ is drawn according to the sum-weights $\Omega^{\mathsf{S}} \in \boldsymbol{\Omega}$, which are associated a Dirichlet prior, parameterized with hyper-parameters $\boldsymbol{\gamma}$: $Z_n^{\mathsf{S}} \sim \mathsf{Cat}(\Omega^{\mathsf{S}})$, $\Omega^{\mathsf{S}} \sim \mathsf{Dir}(\boldsymbol{\gamma})$.

As previously discussed, an assignment of $\mathbf{Z}_n$ determines an induced tree through the SPN, selecting a set of indices $\mathbf{j}_n = (j_n^1, \dots, j_n^D)$, such that the joint distribution of $\mathbf{x}_n = (x_n^1, \dots, x_n^D)$ factorizes as

$$p(\mathbf{x}_n \,|\, \mathbf{j}_n, \boldsymbol{\eta}) = \prod_{d=1}^{D} \mathsf{L}_{j_n^d}^d(x_n^d \,|\, \boldsymbol{\eta}_{j_n^d}^d), \tag{1}$$

where $\mathsf{L}_j^d$ is the $j^{\text{th}}$ leaf for feature $d$ and $\boldsymbol{\eta}_j^d = \{\boldsymbol{\eta}_{j,\ell}^d\}_{\ell \in \mathcal{L}^d}$ is the set of parameters belonging to the likelihood models associated to it. More precisely, the $j^{\text{th}}$ leaf distribution $\mathsf{L}_j^d$ is modeled as a mixture over the likelihood dictionaries provided by the user, i.e.,

$$\mathsf{L}_j^d(x_n^d \,|\, \boldsymbol{\eta}_j^d) = \sum\nolimits_{\ell \in \mathcal{L}^d} w_{j,\ell}^d \, p_\ell^d(x_n^d \,|\, \boldsymbol{\eta}_{j,\ell}^d). \tag{2}$$

Note that the likelihood models $p_\ell^d$ are *shared* among all leaves with the same scope $d$, but each leaf has its private parameter $\boldsymbol{\eta}_{j,\ell}^d$ for it. Moreover, each $\boldsymbol{\eta}_{j,\ell}^d$ is also equipped with a suitable prior distribution parameterized with hyper-parameters $\boldsymbol{\lambda}_\ell^d$. For a discussion on the selection of suitable prior distributions, refer to Appendix B in the supplementary material.

The likelihood mixture coefficients $w_{j,\ell}^d$ are drawn from a Dirichlet distribution with hyper-parameters $\boldsymbol{\alpha}$, i.e., $\mathbf{w}_j^d \sim \mathsf{Dir}(\boldsymbol{\alpha})$. For each entry $x_n^d$, the likelihood model is selected by an additional categorical LV $s_{j,n}^d \sim \mathsf{Cat}(\mathbf{w}_j^d)$. Finally, the observation entry $x_n^d$ is sampled from the selected likelihood model: $x_n^d \sim p_{s_{j,n}^d}^d$.

## Bayesian Inference

The hierarchical LV structure of SPNs allows ABDA to perform Bayesian inference via a simple and effective Gibbs sampling scheme. To initialize the global structure, we use the likelihood-agnostic SPN structure learning algorithm proposed for MSPNs (Molina et al. 2018). Specifically, we apply the RDC to split samples and features while extending it to deal with missing data (see supplementary for details). The local level in ABDA is constructed by equipping each leaf node in $\mathcal{S}$ with the dictionaries of likelihood models as described above. Additionally, one can introduce global type-RVs, responsible for selecting a specific likelihood model (or data type) for each feature $d$ (see Fig. 2d and further explanations below).

Furthermore note that, in contrast to MSPNs, ABDA is not constrained to the LV structure provided by structure learning. Indeed, inference in ABDA accounts for *uncertainty also on the underlying latent structure*. As a consequence, a wrongly overparameterized LV structure provided to ABDA can still be turned into a simpler one by our algorithm by using a sparse prior on the SPN

**Algorithm 1** Gibbs sampling inference in ABDA

**Input:** $N \times D$ data matrix $\mathbf{X}$, SPN $\mathcal{S}$, $\mathbf{Z} = \{\mathbf{Z}_n\}$, $\mathbf{s} = \{s_{j,n}^d\}$, $\boldsymbol{\Omega} = \{\boldsymbol{\Omega}^{\mathsf{S}}\}$, $\mathbf{w} = \{\mathbf{w}_j^d\}$, $\boldsymbol{\eta} = \{\boldsymbol{\eta}_{j,\ell}^d\}$, max iters $I$, burn-in $B$

$\mathcal{D} \leftarrow \emptyset$
Initialize $\mathbf{Z}, \mathbf{s}, \boldsymbol{\Omega}, \mathbf{w}, \boldsymbol{\eta}$
**for** $i = 1, \dots, I$ **do**
  **for** $n = 1, \dots, N$ **do**
    Sample $\mathbf{Z}_n$, $s_{:,n} \,|\, \mathbf{X}, \boldsymbol{\Omega}, \boldsymbol{\eta}$
  **for** $d \in \{1 \dots D\}$ **do**
    **for** $j \in \{j : \mathsf{L}_j^d \in \mathcal{S}\}$ **do**
      Sample $\mathbf{w}_j^d \,|\, \mathbf{Z}, \mathbf{s}$
      **for** $\ell \in \mathcal{L}^d$ **do**
        Sample $\boldsymbol{\eta}_{j,\ell}^d \,|\, \mathbf{X}, \mathbf{Z}, \mathbf{s}$
  Sample $\boldsymbol{\Omega} \,|\, \mathbf{Z}$
  **if** $i > B$ **then**
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathbf{Z}, \mathbf{s}, \boldsymbol{\Omega}, \mathbf{w}, \boldsymbol{\eta}\}$
**Output**: $\mathcal{D}$

weights (see Appendix H).

To perform inference, we draw samples $\mathcal{D}$ from the posterior distribution $p(\mathbf{Z}, \mathbf{s}, \boldsymbol{\Omega}, \mathbf{w}, \boldsymbol{\eta} \,|\, \mathbf{X})$ via Gibbs sampling, where $\mathbf{Z}$ is the set of the SPN's LVs, $\mathbf{s}$ is the set of all local LVs selecting the likelihood models, $\boldsymbol{\Omega}$ is the set of all sum-weights, $\mathbf{w}$ are the distributions of $\mathbf{s}$, and $\boldsymbol{\eta}$ is the set collecting all parameters of all likelihood models. Next, we describe each routine involved to sample from the conditionals for each of these RVs in turn. Algorithm 1 summarizes the full Gibbs sampling scheme. A Rao-Blackwellised version to improve mixing is discussed in Appendix A.

**Sampling LVs Z.** Given the hierarchical LV structure of $\mathcal{S}$, it is easy to produce a sample for $\mathbf{Z}_n$ by ancestral sampling, i.e. by sampling an induced tree $\mathcal{T}_n$. To this end we condition on a sample $\mathbf{x}_n$, and current $\boldsymbol{\Omega}$, $\boldsymbol{\eta}$ and $\mathbf{w}$. Starting from the root of $\mathcal{S}$, for each sum node $\mathsf{S}$ we encounter, we sample a child branch $c$ from

$$p(Z_n^{\mathsf{S}} = c \,|\, , \mathbf{x}_n, \boldsymbol{\Omega}, \boldsymbol{\eta}, \mathbf{w}) \propto \omega_{\mathsf{S},c}\, \mathcal{S}_c(\mathbf{x}_n \,|\, \boldsymbol{\Omega}, \boldsymbol{\eta}, \mathbf{w}). \quad (3)$$

Note that we are effectively conditioning on the states of the ancestors of $\mathsf{S}$. Moreover, we have marginalized out all $\boldsymbol{Z}$ below $\mathsf{S}$ and all $\boldsymbol{s}$, which just amounts to evaluating $\mathcal{S}_c$ bottom-up, for given parameters $\boldsymbol{\Omega}$, $\boldsymbol{\eta}$, and $\mathbf{w}$ (Poon and Domingos 2011). Sampling a tree $\mathcal{T}_n$ does in general not reach all sum nodes. Since these sum nodes are "detached" from the data, we need to sample their LVs from the prior (Peharz et al. 2017).

**Sampling likelihood model assignments s.** Similarly as for LVs $\mathbf{Z}_n$, we sample $s_{j,n}^d$ from the posterior distribution $p(s_{j,n}^d = \ell \,|\, \mathbf{w}, \mathbf{j}, \boldsymbol{\eta}) \propto w_{j,\ell}^d\, p_\ell^d(x_n^d \,|\, \boldsymbol{\eta})$ if $j = j_n^d$.

**Sampling leaf parameters $\boldsymbol{\eta}$.** Sampling $\mathbf{Z}_n$ gives rise to the leaf indices $\mathbf{j}_n$, which assign samples to leaves. Within leaves, $\mathbf{s}$ further assign samples to likelihood models. For parameters $\boldsymbol{\eta}_{j,\ell}^d$ let $\mathbf{X}_{j,\ell}^d = \{x_n^d \,|\, \forall n \colon j_n^d = j \wedge s_{j,n}^d = \ell\}$, i.e., the samples in the $d^{\text{th}}$ column of $\mathbf{X}$ which have been assigned to the $\ell^{\text{th}}$ model in leaf $\mathsf{L}_j^d$. Then, $\boldsymbol{\eta}_{j,\ell}^d$ is updated according to $\boldsymbol{\eta}_{j,\ell}^d \sim \prod_{x \in \mathbf{X}_{j,\ell}^d} p_\ell^d(x \,|\, \boldsymbol{\eta}_{j,\ell}^d)\, p(\boldsymbol{\eta}_{j,\ell}^d \,|\, \boldsymbol{\lambda}_\ell^d)$. When the likelihood models $p_\ell^d$ are equipped with conjugate priors, these updates are straightforward. Moreover, also for non-conjugate priors they can easily be approximated using numerical methods, since we are dealing with single-dimensional problems.

**Sampling weights $\boldsymbol{\Omega}$ and w.** For each sum node $\mathsf{S}$ we sample its associated weights from the posterior $p(\Omega^{\mathsf{S}} \,|\, \{\mathbf{Z}_n\}_{n=1}^N)$, which is a Dirichlet distribution with parameters $\boldsymbol{\gamma} + \sum_{n=1}^N \mathbb{1}\{(\mathsf{S}, c) \in \mathcal{T}_n\}$. Similarly, we can sample the likelihood weights $\mathbf{w}_j^d$ from a Dirichlet distribution with parameters $\boldsymbol{\alpha} + [\sum_{n=1}^N \mathbb{1}\{j_n^d = j \wedge s_{j,n}^d = \ell\}]_{\ell \in \mathcal{L}^d}$.

## Automating Exploratory Data Analysis

In this section, we discuss how inference in ABDA can be exploited to perform common exploratory data analysis tasks in an automatic way. For all inference tasks (e.g., computing $\log p(\mathbf{x}_n)$), one can either condition on the model parameters, e.g., by using the maximum likelihood parameters within posterior samples $\mathcal{D}$, or perform a Monte Carlo estimate over $\mathcal{D}$. While the former allows for efficient computations, the latter allows for quantifying the model uncertainty. We refer to Appendixes E-G in the supplementary for more details.

**Missing value imputation.** Given a sample $\mathbf{x}_n = (\mathbf{x}_n^o, \mathbf{x}_n^m)$ comprising observed $\mathbf{x}_n^o$ and missing $\mathbf{x}_n^m$ values, ABDA can efficiently impute the latter as the *most probable explanation* $\tilde{\mathbf{x}}_n^m = \arg\max_{\mathbf{x}_n^m} \mathcal{S}(\mathbf{x}_n^m \,|\, \mathbf{x}_n^o)$ via efficient approximate SPN routines (Peharz et al. 2017).

**Anomaly detection.** ABDA is robust to outliers and corrupted values since, during inference, it will tend to assign anomalous samples into low-weighted mixture components, i.e., sub-networks of the underlying SPN or leaf likelihood models. Outliers will tend to be either grouped into anomalous *micro-clusters* (Chandola, Banerjee, and Kumar 2009) or assigned to the tails of a likelihood model. Therefore, $\log p(\mathbf{x}_n)$ can be used as a strong signal to indicate $\mathbf{x}_n$ is an outlier (in the transductive case) or a *novelty* (in the inductive case) (Goldstein and Uchida 2016).

**Data type and likelihood discovery.** ABDA automatically estimates *local* uncertainty over likelihood models and statistical types by inferring the dictionary coefficients $w_{j,\ell}^d$ for a leaf $\mathsf{L}_j^d$. However, we can extend ABDA to reason about data type also on a *global* level by explicitly introducing a *type variable* $T^d$ for feature $d$. This type variable might either represent a *parametric type*, e.g., Gaussian or Gamma, or a *data type*, e.g., real-valued or positive-real-valued, in a similar way to ISLV (Valera and Ghahramani 2017). To this end, we introduce state-indicators for each type variable $T^d$, and connect them into each likelihood model in leaf $\mathsf{L}_j^d$, as shown in Fig. 2d. In this example, we introduced the state-indicators $T_{\mathcal{N}}^d$ and $T_{\Gamma}^d$ to represent a global type $T^d$ which distinguishes between Gaussian and Gamma distribution. Note that this technique is akin to the *augmentation* of SPNs (Peharz et al. 2017), i.e., explicitly introducing the LVs for sum nodes. Here, however, the introduced $\{T^d\}_{d=1}^D$ have an explicit intended semantic as global data type variables. Crucially, after introducing $\{T^d\}_{d=1}^D$, the underlying SPN is still complete and decomposable, i.e., we can easily perform *inference* over them. In particular, we can estimate the posterior probability for a feature $d$ to have a particular type as:

$$p(T^d \,|\, \mathbf{X}) \approx \frac{1}{|\mathcal{D}|} \sum_{\{\boldsymbol{\Omega}, \mathbf{w}, \boldsymbol{\eta}\} \in \mathcal{D}} \mathcal{S}(T^d \,|\, \Omega, \mathbf{w}, \boldsymbol{\eta}). \quad (4)$$

The marginal terms $\mathcal{S}(T^d \,|\, \Omega, \mathbf{w}, \boldsymbol{\eta})$ are easily obtained via SPN inference – we simply need to set all leaves and all indicators for $\{T^{d'}\}_{d' \neq d}$ equal to 1 (Poon and Domingos 2011).

**Dependency pattern mining.** ABDA is able to retrieve *global dependencies*, e.g., by computing pairwise hybrid mutual information, in a similar way to

MSPNs (Molina et al. 2018). Additionally, ABDA can provide users *local* patterns in the form of dependencies within a data partition $\mathbf{X}^{\mathsf{N}} \subseteq \mathbf{X}$ associated with any node $\mathsf{N}$ in $\mathcal{S}$. In particular, let $\mathbf{X}^{\mathsf{N}}$ contain all entries $x_n^d$ such that $d$ is in the scope of $\mathsf{N}$ and $n$ such that $\mathbf{Z}_n$ yield an induced tree from the SPN's root to $\mathsf{N}$. Then, for each leaf $\mathsf{L}_j^d$ and likelihood model $\ell \in \mathcal{L}^d$ one can extract a pattern of the form $\mathcal{P}\colon \pi_l^d \leq X^d < \pi_h^d$, where $[\pi_l^d, \pi_h^d)$ is an interval in the domain of $X^d$. The pattern can be deemed as present, when its probability exceeds a user-defined threshold $\theta$: $p_{\mathsf{L}_j^d}(\mathcal{P}) \geq \theta$. A conjunction of patterns $\mathcal{P}^{\mathsf{N}} = \mathcal{P}_1 \wedge \ldots \wedge \mathcal{P}_{|\mathsf{sc}(\mathsf{N})|}$ represents the correlation among features in $\mathbf{X}^{\mathsf{N}}$, and its *relevance* can be quantified as $p_{\mathcal{S}}(\mathcal{P}^{\mathsf{N}})$. This technique relates to the notion of *support* in association rule mining (Agrawal and Srikant 1994), whose binary patterns are here generalized to also support continuous and (non-binary) discrete RVs.

## Experimental Evaluation

We empirically evaluate ABDA on synthetic and real-world datasets both as a density estimator and as a tool to perform several exploratory data analysis tasks. Specifically, we investigate the following questions:

**(Q1)** How does ABDA estimate likelihoods and statistical data types when a ground truth is available?

**(Q2)** How accurately does ABDA perform density estimation and imputation over unseen real-world data?

**(Q3)** How robust is ABDA w.r.t. anomalous data?

**(Q4)** How can ABDA be exploited to unsupervisedly extract dependency patterns?

**Experimental setting.** We implemented ABDA by leveraging the SPFlow library[1]. In all experiments, we use a symmetric Dirichlet prior with $\gamma = 10$ for sum weights $\mathbf{\Omega}$ and a sparse symmetric prior with $\alpha = 0.1$ for the leaf likelihood weights $\mathbf{w}_j^d$. We consider the following likelihoods for continuous data: Gaussian distributions ($\mathcal{N}$) for REAL-valued data; Gammas ($\Gamma$) and exponential (Exp) for POSitive real-valued data; and, for discrete data, we consider Poisson (Poi) and Geometric (Geo) distributions for NUMerical data, and Categorical (Cat) for NOMinal data, while a Bernoulli for Binary data. For details on the prior distributions employed, for the likelihood parameters and their hyper-parameters, please refer to the Appendix.

**(Q1) Likelihood and statistical type uncertainty.** We use synthetic data in order to have control over the ground-truth distribution of the data. To this end, we generate 90 synthetic datasets with different combinations of likelihood models and dependency structures with different numbers of samples $N \in \{2000, 5000, 10000\}$ and numbers of features $D \in \{4, 8, 16\}$. For each possible combination of values, we create ten independent datasets of $N$ samples (reserving 20% of them for testing), yielding 90 data partitionings randomly built by mimicking the SPN learning process of (Gens and Domingos 2013; Vergari, Di Mauro, and Esposito 2015). The leaf distributions have been randomly drawn from the aforementioned likelihood dictionaries. We then perform density estimation with ABDA on these datasets. See Appendix C for details on ABDA inference and the data generation process.

Fig. 3 summarizes our results. In Fig. 3a we see that ABDA's likelihood matches the true model closely in all settings, indicating that ABDA is an accurate density estimator. Additionally, as shown in Fig. 3b, ABDA is able to capture the uncertainty over data types, as it achieves high average cosine similarity between the ground truth type distribution and the inferred posterior over data type $p(T^d \mid \mathbf{X})$ (see Eq. (4)), both for i) likelihood functions (using $\mathcal{N}$, $\Gamma$ and Exp for continuous and Pos, Geo and Cat for discrete features); and ii) the corresponding statistical data types (using POS and REAL for continuous and NUM and NOM for discrete features).

Furthermore, when forcing a hard decision on distributions and data types, ABDA delivers accurate predictions. As shown in the confusion matrices in Fig. 3c, selecting the most probable likelihood (data type) based on ABDA inference matches the ground truth up to the expected indiscernibility due to finite sample size. For further discussions, please refer to (Valera and Ghahramani 2017).

**(Q2) Density estimation and imputation.** We evaluate ABDA both in a *transductive* scenario, where we aim to estimate (or even impute) the missing values in the data used for inference/training; and in an *inductive* scenario, where we aim to estimate (impute) data that was not available during inference/training. We compare against ISLV (Valera and Ghahramani 2017), which directly accounts for data type (but not likelihood model) uncertainty, and MSPNs (Molina et al. 2018), to observe the effect of modeling uncertainty over the RV dependency structure via an SPN LV hierarchy.

From ISLV and MSPN original works we select 12 real-world datasets differing w.r.t. size and feature heterogeneity. Appendix C reports detailed dataset information, while Appendix H contains additional experiments in the MSPN original setting. Specifically, for the transductive setting, we randomly remove either 10% or 50% of the data entries, reserving an additional 2% as a validation set for hyperparameter tuning (when required), and repeating five times this process for robust evaluation. For the inductive scenario, we split the data into train, validation, and test (70%, 10%, and 20% splits).

For ABDA and ISLV, we run 5000 iterations of Gibbs sampling [2], discarding the first 4000 for burn-in. We set for ISLV the number of latent factors to $\lfloor D/2 \rfloor$. We learn MSPNs with the same hyper-parameters as for ABDA structure learning, i.e., stopping to grow the network when the data to be split is less than 10% of the dataset,

[1] https://github.com/SPFlow/SPFlow

[2] On the Adult dataset, ISLV did not converge in 72hr.

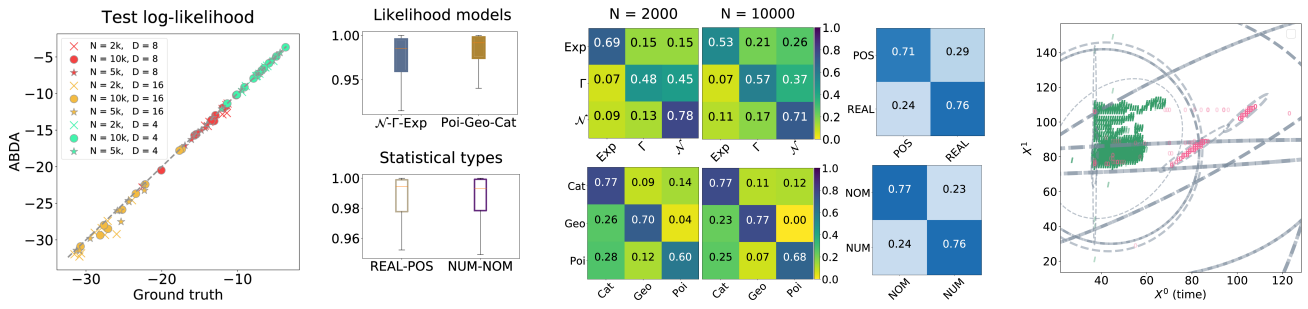



Figure 3: **(a)** Mean test log-likelihood on synthetic data w.r.t. the ground truth. **(b)** Distributions of the mean cosine similarity between true and retrieved uncertainty weights over likelihood models (top) and statistical types (bottom). **(c)** Confusion matrices for the most likely likelihood model resp. statistical type. **(d)** ABDA separates outliers (pink 'O') from inliers (green 'I') via hierarchical partitioning on Shuttle data, see Appendix F. Best viewed in colors.

Table 1: Density estimation and anomaly detection. Mean test log-likelihood on real-world benchmark datasets for trans-/inductive scenarios and mean AUC ROC for outlier detection. Best values are bold.

| | transductive setting (10% mv) | | | transductive setting (50% mv) | | | inductive setting | | outlier detection | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ISLV | ABDA | MSPN | ISLV | ABDA | MSPN | ABDA | MSPN | | 1SVM | LOF | HBOS | ABDA |
| Abalone | -1.15±0.12 | -0.02±0.03 | **0.20** | -0.89±0.36 | -0.05±0.02 | **0.14** | 2.22±0.02 | **9.73** | Aloi | 51.71±0.02 | **74.19**±0.70 | 52.86±0.53 | 47.20±0.02 |
| Adult | - | **-0.60±0.02** | -3.46 | - | **-0.69±0.01** | -5.83 | **-5.91±0.01** | -44.07 | Thyroid | 46.18±0.39 | 62.38±1.04 | 62.77±3.69 | **84.88**±0.96 |
| Austral. | -7.92±0.96 | **-1.74±0.19** | -3.85 | -9.37±0.69 | **-1.63±0.04** | -3.76 | **-16.44±0.04** | -36.14 | Breast | 45.77±11.1 | 98.06±0.70 | 94.47±0.79 | **98.36**±0.07 |
| Autism | -2.22±0.06 | **-1.23±0.02** | -1.54 | -2.67±0.16 | **-1.24±0.01** | -1.57 | **-27.93±0.02** | -39.20 | Kdd99 | 53.40±3.63 | 46.39±1.95 | 87.59±4.70 | **99.79**±0.10 |
| Breast | -3.84±0.05 | -2.78±0.07 | **-2.69** | -4.29±0.17 | **-2.85±0.01** | -3.06 | **-25.48±0.05** | -28.01 | Letter | 63.38±17.6 | **86.55**±2.23 | 60.47±1.80 | 70.36±0.01 |
| Chess | -2.49±0.04 | **-1.87±0.01** | -3.94 | -2.58±0.04 | **-1.87±0.01** | -3.92 | **-12.30±0.00** | -13.01 | Pen-glo | 46.86±1.02 | 87.25±1.94 | 71.93±1.68 | **89.87**±2.87 |
| Crx | -12.17±1.41 | **-1.19±0.12** | -3.28 | -11.96±1.01 | **-1.20±0.04** | -3.51 | **-12.82±0.07** | -36.26 | Pen-loc | 44.11±6.07 | **98.72**±0.20 | 64.30±2.70 | 90.86±0.79 |
| Dermat. | -2.44±0.23 | **-0.96±0.02** | -1.00 | -3.57±0.32 | **-0.99±0.01** | -1.01 | **-24.98±0.19** | -27.71 | Satellite | 52.14±3.08 | 83.51±11.98 | 90.92±0.16 | **94.55**±0.68 |
| Diabetes | -10.53±1.51 | **-2.21±0.09** | -3.88 | -12.52±0.52 | **-2.37±0.09** | -4.01 | **-17.48±0.05** | -31.22 | Shuttle | 89.37±5.13 | 66.29±1.69 | **98.47**±0.24 | 78.61±0.02 |
| German | -3.49±0.21 | **-1.54±0.01** | -1.58 | -4.06±0.28 | **-1.55±0.01** | -1.60 | **-25.83±0.05** | -26.05 | Speech | 45.61±3.64 | **49.37**±0.87 | 47.47±0.10 | 46.96±0.01 |
| Student | -2.83±0.27 | **-1.56±0.03** | -1.57 | -3.80±0.29 | **-1.57±0.01** | -1.58 | **-28.73±0.10** | -30.18 | | | | | |
| Wine | -1.19±0.02 | -0.90±0.02 | **-0.13** | -1.34±0.01 | -0.92±0.01 | **-0.41** | -10.12±0.01 | **-0.13** | | | | | |

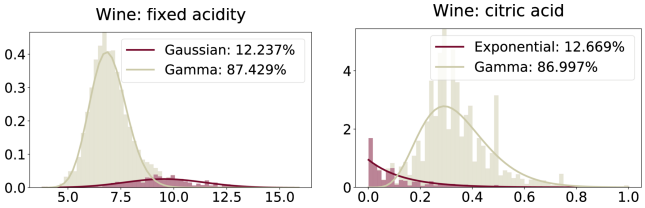



Figure 4: Data exploration and dependency discovery with ABDA on the Wine quality dataset. ABDA identifies the two modalities in the data induced by red and white wines, and extracts the following patterns: $5.8 \leq$ FixAcid $< 8.1 \wedge 0.2 \leq$ CitAcid $< 0.5$ and $7.1 \leq$ FixAcid $< 12.0 \wedge 0.0 \leq$ CitAcid $< 0.3$ ($\theta = 0.9$).

while employing a grid search in $\{0.3, 0.5, 0.7\}$ for the RDC dependency test threshold.[3]

Tab. 1 reports the mean test-log likelihoods–evaluated on missing values in the transductive or on completely unseen test samples in the inductive cases–for all datasets. Here, we can see that ABDA outperforms both ISLVs and MSPNs in most cases for both scenarios. Moreover, since aggregated evaluations of heterogeneous likelihoods might be dominated by a subset features, we also report in Appendix E the average test log-likelihood and the normalized root mean squared error (NRMSE) of the imputed missing values (normalized by the range of each RV separately) for each feature in the data. Here, we observe that ABDA is, in general, more accurate and robust across different features and data types than competitors. We finally remark that due to the piecewise approximation of the likelihood adopted by the MSPN, evaluations of the likelihood provided by this approach might be boosted by the fact that it renormalizes an infinite support distribution to a bounded one.

**(Q3) Anomaly detection.** We follow the *unsupervised outlier detection* experimental setting in (Goldstein and Uchida 2016) to evaluate the ability of ABDA to detect anomalous samples on a set of standard benchmarks. As a qualitative example, we can observe in Fig 3d that ABDA either clusters outliers together or relegates them to leaf distribution tails, assigning them low probabilities. Tab. 1 compares, in terms of the mean AUC ROC, ABDA—for which we use the negative log-likelihood as the outlier score—with staple outlier detection methods like one-class SVMs (1SVM) (Schölkopf et al. 2001), local outlier factor (LOF) (Breunig et al. 2000) and histogram-based outlier score (HBOS) (Goldstein and Dengel 2012). It is clearly visible that ABDA perform

[3] In Appendix H we also evaluate ABDA and MSPN robustness to overparametrized structures.

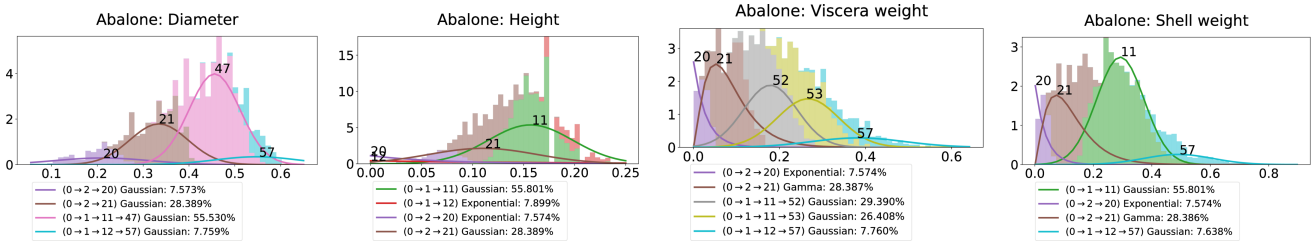



Figure 5: Data exploration and pattern discovery on the Abalone dataset, comprising physical measurements (e.g. diameter, height, weight) of different specimens. Each colored density belongs to a unique partition $\mathbf{X}^{\mathsf{N}}$ discovered by ABDA and here labeled by an integer indicating the corresponding node $\mathsf{N}$ (0 indicates the root of the SPN). Hierarchies over partitions are shown in the legend as the path connecting the root node to $\mathsf{N}$ (e.g., $0 \to 1 \to 11 \to 47$ indicates that the partition #11 (green) is responsible for the green densities and it decomposes into the partitions #52 (gray), #53 (yellow) and #47 (pink), highlighting different correlation patterns across feature intervals). For instance, $\mathcal{P}_1$ : $0.088 \leq \mathsf{Height} < 0.224 \quad \wedge 0.158 \leq \mathsf{ShellWeight} < 0.425 \quad (supp(\mathcal{P}_1) = 0.507)$ is a pattern within such a partition, correlating the height and the shell weight features. Additional patterns, and the other estimates for the dataset features, are reported in Appendix G.

as good as—or even better—in most cases than methods tailored for outlier detection and not being usable for other data analysis tasks. Refer to Appendix F for further experimental details and results.

**(Q4) Dependency discovery.** Finally, we illustrate how ABDA may be used to find underlying dependency structure in the data on the Wine and Abalone datasets as use cases. By performing marginal inference for each feature, and by collapsing the resulting deep mixture distribution into a shallow one, with ABDA we can recover the data modes and reason about the likelihood distributions associated to them.

As an example, ABDA is able to discern the two modes in the Wine data which correspond to the two types of wine, red and white wines, information not given as input to ABDA (Fig 4). Moreover, ABDA is also able to assign to the two modes accurate and meaningful likelihood models: Gamma and Exponential distributions are generally captured for the features fixed and citric acidity, since they are a ratio and indeed follow a positive distribution, while being more skewed and decaying than a Gaussian, employed for the fixed acidity of red wines. Note that since ABDA partitions the data into white and red wines sub-populations, it allows us to reason about statistical dependencies in the data in the form of simple conjunctive patterns (see Fig 4), as discussed in the previous Section. Here, we observe an anti-correlation between fixed and citric acidity: as the former increases the latter tends to zero.

A more involved analysis is carried on the Abalone dataset and summarized in Fig 5. There, retrieved data partitions clearly highlight correlations across features and samples of the data. For instance, it is possible to see how abalone samples differing by weight, height and diameters form neat sub-populations in the data. See Appendix G in the supplementary for a detailed discussion and more results.

## Conclusions

Towards the goal of fully automating exploratory data analysis via density estimation and probabilistic inference, we introduced Automatic Bayesian Density Analysis (ABDA). It automates both data modeling and selection of adequate likelihood models for estimating densities via joint, robust and accurate Bayesian inference. We found that the inferred structures are able of accurately analyzing complex data and discovering the data types, the likelihood models and feature interactions. Overall, it outperformed state-of-the-art in different tasks and scenarios in which domain experts would perform exploratory data analysis by hand.

ABDA opens many interesting avenues for future work. First, we aim at inferring also prior models in an automatic way; abstracting from inference implementation details by integrating probabilistic programming into ABDA; and casting the LV structure learning as nonparametric Bayesian inference. Second, we plan on integrating ABDA in a full pipeline for exploratory data analysis, where probabilistic and logical reasoning can be performed over the extracted densities and patterns to generate human-readable reports, and be treated as input into other ML tasks.

**Acknowledgements** We thank the anonymous reviewers and Wittawat Jitkrittum for their valuable feedback. IV is funded by the MPG Minerva Fast Track Program. This work has benefited from the DFG project CAML (KE 1686/3-1), as part of the SPP 1999, and from the BMBF project MADESI (01IS18043B). This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 797223.

## References

Agrawal, R., and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proceedings of VLDB*, volume 1215, 487–499.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. LOF: identifying density-based local outliers. In *ACM sigmod record*, volume 29, 93–104.

Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41(3):15:1–15:58.

Darwiche, A. 2003. A differential approach to inference in Bayesian networks. *Journal of the ACM (JACM)* 50(3):280–305.

Dennis, A., and Ventura, D. 2012. Learning the architecture of sum-product networks using clustering on variables. In *Proceedings of NIPS*, 2033–2041.

Dennis, A., and Ventura, D. 2015. Greedy structure search for sum-product networks. In *Proceedings of IJCAI*, 932–938.

Duvenaud, D. K.; Lloyd, J. R.; Grosse, R. B.; Tenenbaum, J. B.; and Ghahramani, Z. 2013. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of ICML*, 1166–1174.

Gens, R., and Domingos, P. 2012. Discriminative learning of sum-product networks. In *Proceedings of NIPS*, 3239–3247.

Gens, R., and Domingos, P. 2013. Learning the structure of sum-product networks. In *Proceedings of ICML*, 873–880.

Ghahramani, Z., and Beal, M. J. 2000. Variational inference for Bayesian mixtures of factor analysers. In *Proceedings of NIPS*, 449–455.

Goldstein, M., and Dengel, A. 2012. Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm. *KI-2012* 59–63.

Goldstein, M., and Uchida, S. 2016. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one* 11(4).

Guyon, I.; Chaabane, I.; Escalante, J. H.; and Escalera, S. 2016. A brief review of the ChaLearn AutoML challenge: Any-time any-dataset learning without human intervention. In *ICML Workshop on AutoML*, 21–30.

Lloyd, J. R.; Duvenaud, D. K.; Grosse, R. B.; Tenenbaum, J. B.; and Ghahramani, Z. 2014. Automatic construction and natural-language description of nonparametric regression models. In *Proceedings of AAAI*, 1242–1250.

Lopez-Paz, D.; Hennig, P.; and Schölkopf, B. 2013. The randomized dependence coefficient. In *Proceedings of NIPS*, 1–9.

Mansinghka, V.; Shafto, P.; Jonas, E.; Petschulat, C.; Gasner, M.; and Tenenbaum, J. B. 2016. Crosscat: A fully Bayesian nonparametric method for analyzing heterogeneous, high dimensional data. *JMLR* 17(1):4760–4808.

Molina, A.; Vergari, A.; Di Mauro, N.; Natarajan, S.; Esposito, F.; and Kersting, K. 2018. Mixed sum-product networks: A deep architecture for hybrid domains. In *Proceedings of AAAI*.

Molina, A.; Natarajan, S.; and Kersting, K. 2017. Poisson sum-product networks: A deep architecture for tractable multivariate poisson distributions. In *Proceedings of AAAI*, 2357–2363.

Peharz, R.; Tschiatschek, S.; Pernkopf, F.; and Domingos, P. 2015. On theoretical properties of sum-product networks. *Proceedings of AISTATS* 744–752.

Peharz, R.; Gens, R.; Pernkopf, F.; and Domingos, P. 2017. On the latent variable interpretation in sum-product networks. *IEEE TPAMI* 39(10):2030–2044.

Peharz, R.; Geiger, B.; and Pernkopf, F. 2013. Greedy part-wise learning of sum-product networks. In *Proceedings of ECML-PKDD*, 612–627.

Poon, H., and Domingos, P. 2011. Sum-product networks: a new deep architecture. In *Proceedings of UAI*, 337–346.

Pronobis, A.; Riccio, F.; and Rao, R. P. 2017. Deep spatial affordance hierarchy: Spatial knowledge representation for planning in large-scale environments. In *ICAPS 2017 Workshop*.

Schölkopf, B.; Platt, J. C.; Shawe-Taylor, J. C.; Smola, A. J.; and Williamson, R. C. 2001. Estimating the support of a high-dimensional distribution. *Neural Comput.* 13(7):1443–1471.

Trapp, M.; Madl, T.; Peharz, R.; Pernkopf, F.; and Trappl, R. 2017. Safe semi-supervised learning of sum-product networks. *Proceedings of UAI*.

Valera, I., and Ghahramani, Z. 2017. Automatic discovery of the statistical types of variables in a dataset. In *Proceedings of ICML*, 3521–3529.

Valera, I.; Pradier, M. F.; and Ghahramani, Z. 2017. General Latent Feature Models for Heterogeneous Datasets. *ArXiv e-prints*.

Vergari, A.; Peharz, R.; Di Mauro, N.; Molina, A.; Kersting, K.; and Esposito, F. 2018. Sum-product autoencoding: Encoding and decoding representations using sum-product networks. In *Proceedings of AAAI*.

Vergari, A.; Di Mauro, N.; and Esposito, F. 2015. Simplifying, regularizing and strengthening sum-product network structure learning. In *Proceedings of ECML-PKDD*, 343–358.

Vergari, A.; Di Mauro, N.; and Esposito, F. 2018. Visualizing and understanding sum-product networks. *MLJ*.

Zhao, H.; Poupart, P.; and Gordon, G. J. 2016. A unified approach for learning the parameters of sum-product networks. In *Proceedings of NIPS*, 433–441.