Automatic Bayesian Density Analysis: Supplementary Material

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

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

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

A. Gibbs sampling

Table 1: Average time (in seconds) per iteration for the best ABDA and ISLV models on the density estimation datasets.

	ISLV (*s/iter*)	ABDA (*s/iter*)		ISLV (*s/iter*)	ABDA (*s/iter*)
Abalone	5.21	0.20	Australian	0.89	0.04
Autism	13.56	0.25	Breast	1.30	0.05
Chess	40.6	0.36	Crx	1.10	0.04
Dermatology	2.01	0.14	Diabetes	0.75	0.03
German	4.05	0.08	Student	1.81	0.06
Wine	7.05	0.15			

A1 Implementation and times The proposed Gibbs sampling scheme is simple and efficient, scaling with worst-case time complexity linearly in the number of samples in $\mathbf{X}$ and in the size of $\mathcal{S}$, i.e., $O(N|\mathcal{S}|$. Note that the size of the SPNs learned by LearnSPN/MSPN-like algorithms is generally much smaller than $N \times D$, where D is the number of features in the dataset (see (Vergari, Di Mauro, and Esposito 2015) for further details). Furthermore, note that this time complexity does not depend on the domain size of discrete RVs, unlike ISLV (Valera and Ghahramani 2017) (which also has to perform a costly matrix inversion). Practically, exploiting the underlying SPN to efficiently condition on the LV hierarchy—allowing to keep updates for involved leaf distributions— *enormously speeds Gibbs sampling* in ABDA, also on continuous data. As a reference, see the average times (seconds) per iterations of ISLV and ABDA in Table 1.

Moreover, ABDA lends itself to be implemented parsimoniously if one has access to sampling routines for the parametric models employed in the leaf distributions. See Appendix B for a detailed specification of the likelihood and posterior parametric forms involved in the experiments.

A2. Rao-Blackwellised Gibbs sampler In all our experiments, we observed it converging quickly to high likelihood posterior solutions. We use likelihood traceplots (Cowles and Carlin 1996) to track the sampler convergence, noting that at most 3000 samples for the largest datasets, are generally are needed.

Nevertheless, to further improve convergence, one may devise a Rao-Blackwellised version (Murphy 2012) by collapsing out several parameters in ABDAs. For instance, leaf distribution parameters $\boldsymbol{\eta}^d_{j\ell}$ can be marginalized when sampling assignments to the LVs $\mathbf{Z}$. Indeed, one can draw them from:

$$p(Z^h_n = c|\mathbf{x}_n, \{Z^p_n\}_{p\in\mathsf{anc}(h)}, \boldsymbol{\Omega}) \propto \omega_{hc}\mathcal{S}_c(\mathbf{x}_n|\mathbf{x}_{\setminus n}, \boldsymbol{\Omega}) \tag{1}$$

where $\mathcal{S}_c(\mathbf{x}_n|\mathbf{x}_{\setminus n}, \boldsymbol{\Omega})$ denotes the posterior predictive distribution of sample $\mathbf{x}_n$ given all remaining samples $\mathbf{x}_{\setminus n}$ through SPN $\mathcal{S}$ equipped with weights $\boldsymbol{\Omega}$. Such a quantity can be computed again in time linear in the size of $\mathcal{S}$.

B. Likelihood and prior models

The parametric forms used in our experiments for the likelihood models, and their corresponding priors, are shown below, w.r.t. the statistical data type involved: REAL-valued data, POSitive real-valued data, NUMerical, NOMinal and BINary.

Note that, even if in our experiments we focused on likelihood dictionaries within the exponential family for simplicity, *ABDA can be readily extended to any likelihood model*. E.g., for ordinal data one can just plug the ordinal *probit model* (Valera and Ghahramani 2017) in in the dictionary. Of course, introducing a prior distribution that does not allow to exploit conjugacy, but that could represent valuable prior knowledge about a distribution, would need to derive an approximate sampling routines for the involved likelihood model.

[***real-valued***]
Gaussian $\mathcal{N}(\mu, \sigma^2)$

- **prior:** $p(\mu, \sigma^2|m_0, V_0, \alpha_0, \beta_0) = \mathcal{N}(\mu|m_0, \sigma^2 V_0) IGamma(\sigma^2|\alpha_0, \beta_0)$

- **posterior:** $p(\mu, \sigma^2 | m_0, V_0, \alpha_0, \beta_0) = \mathcal{N}(\mu | \frac{V_0^{-1} m_0 + N\bar{x}}{V_0^{-1}+N}, V_0^{-1} + N) \times IGamma(\alpha_0 + N/2, \beta_0 + \frac{1}{2}\sum_{i=1}^{N}(x_i - \bar{x})^2 + \frac{V_0^{-1} N (\bar{x} - m_0)^2}{2(V_0^{-1}+N)})$

[***positive real-valued***]

Gamma with fixed α, $Gamma(\alpha, \beta)$

- **prior:** $p(\beta|\alpha_0, \beta_0) = Gamma(\alpha_0, \beta_0)$
- **posterior:** $p(\beta|\mathbf{x}, \alpha_0, \beta_0) = Gamma(\alpha_n = \alpha_0 + N\alpha, \beta_n = \beta_0 + \sum_{i=1}^{N} x_i)$

Exponential $Exponential(\lambda)$

- **prior:** $p(\lambda|\alpha_0, \beta_0) = Gamma(\alpha_0, \beta_0)$
- **posterior:** $p(\lambda|\mathbf{x}, \alpha_0, \beta_0) = Gamma(\alpha_n = \alpha_0 + N\alpha, \beta_n = \beta_0 + \sum_{i=1}^{N} x_i)$

[***nominal***]

Categorical $Cat(\{\theta_i\}_{i=1}^{k})$

- **prior:** $p(\{\theta_i\}_{i=1}^{k}|\boldsymbol{\alpha}_i) = Dirichlet(\{\theta_i\}_{i=1}^{k}|\boldsymbol{\alpha}_i)$
- **posterior:** $p(\{\theta_i\}_{i=1}^{k}|\mathbf{x}, \boldsymbol{\alpha}_i) = Dirichlet(\{\theta_i\}_{i=1}^{k}|\boldsymbol{\alpha}_i + N_i)$

[***numerical***]

Poisson $Poisson(\lambda)$

- **prior:** $p(\lambda|\alpha_0, \beta_0) = Gamma(\lambda|\alpha_0, \beta_0)$
- **posterior:** $p(\lambda|\mathbf{x}, \alpha_0, \beta_0) = Gamma(\lambda|\alpha_n = \alpha_0 + \sum_{i=1}^{N} x_i, \beta_n = \beta_0 + N)$

Geometric $Geometric(\theta)$

- **prior:** $p(\theta|\alpha_0, \beta_0) = Beta(\theta|\alpha_0, \beta_0)$
- **posterior:** $p(\theta_i|\mathbf{x}, \alpha_0, \beta_0) = Beta(\theta|\alpha_0 + N, \beta_0 - N + \sum_{i=1}^{N} \mathbf{x}_i)$

[***binary***]

Bernoulli $Ber(\theta)$

- **prior:** $p(\theta|\alpha_0, \beta_0) = Beta(\theta|\alpha_0, \beta_0)$
- **posterior:** $p(\theta_i|\mathbf{x}, \alpha_0, \beta_0) = Beta(\theta|\alpha_0 + \sum_{i=1}^{N} \mathbf{x}_i, \beta_0 + N - \sum_{i=1}^{N} \mathbf{x}_i)$

C. Synthetic data experiments

C1. Synthetic data generation Here we describe in detail the process we adopted for generating the 90 datasets employed in our controlled experiments. As stated in the main article, we consider an increasing number of samples $N \in \{2000, 5000, 10000\}$ and features $D \in \{4, 8, 16\}$ and for each combination of the two we generate 10 independent datasets. Later on, we split them into training, validation and test partitions (70%, 10%, 20% splits).

We generate each dataset by mimicing the structure learning process of an SPN. Please note that in such a way *we are encompassing highly expressive and complex joint distributions* that might have generated the data (while retaining control over their statistical dependencies, types and likelihood models. Indeed, SPNs have been demonstrated to capture linear (Molina, Natarajan, and Kersting 2017) and highly non-linear correlations (Molina et al. 2018; Vergari, Di Mauro, and Esposito 2018) in the data, being also able to model constrained random vectors, as ones drawn from the simplex (see. (Molina et al. 2018)).

Each dataset is generated in the following way: For every feature, we fix a randomly selected type among real, positive, discrete numerical or nominal. To randomly create a ground truth generative model, i.e., an SPN—denoted as $\mathcal{S}'$—we simulate a stochastic guillotine partitioning of a fictitious $N \times D$ data matrix. Specifically, we follow a LearnSPN-like (Gens and Domingos 2013; Vergari, Di Mauro, and Esposito 2015) structure learning scheme in which columns and rows of this matrix are clustered together, in this case in a random fashion.

At each iteration of the algorithm, it decides to try to split the columns or rows proportionally to a probability $\theta_{\mathsf{split}} = 0.8$. For a tentative column split, the likelihood of a column to be assigned to one of two clusters is drawn from a $\mathsf{Beta}(a = 4, b = 5)$. In case all columns are assigned to a single cluster, no column split is performed and the process moves to the next iteration. Concerning row splits, on the other hand, each row is randomly assigned to one of two clusters with a probability drawn from $\mathsf{Beta}(a = 4, b = 5)$. We alternate these splitting processes until we reach a matrix partition with less than 10% of N number of rows. We assign these final partitions to univariate leave nodes in the spn. For each leaf node, we then randomly select a univariate parametric distribution valid for the type associated with the feature d, as listed in the previous Section.

An example of a synthetically generated dataset for $N = 10000$ samples and $D = 4$ features, and its random partitioning, is shown in Fig. 1. There, samples have been ordered in the data matrix (after performing biclustering) to enhance the visualization of contiguous partitions.

We employ the following priors to draw the corresponding parameters for each distribution: For Gaussian distributions we employ a $\mathsf{Normal} - \mathsf{Inverse} - \mathsf{Gamma}(\mu = 0, V = 30, a = 10, b = 10)$ for the mean and variance, we draw the shape parameter of the Gamma distributions from a $\mathsf{Uniform}(5, 25)$ involved and their scale parameters from a $\mathsf{Gamma}(a = 10, b = 10)$ prior, while for Exponential distributions we randomly select the rate from a $\mathsf{Gamma}(a = 20, b = 5)$. For the discrete data, we draw the number of categories of a Categorical from a discrete $\mathsf{Uniform}(5, 15)$. We then draw the corresponding probabilities from an equally sized and symmetric $\mathsf{Dirichlet}(\alpha = 10)$; the mean parameter of Poissons is drawn from a $\mathsf{Gamma}(a = 100, b = 10)$.

C2. ABDA inference We perform inference on ABDA for each synthetic dataset. We learn the LV structure by letting the RDC independence threshold parameter in $\{0.1, 0.3, 0.5\}$ while fixing to 10% of N the minimum number of samples to continue the partitioning process during SPN structure learning. We select then the best model w.r.t. the log-likelihood scored on a

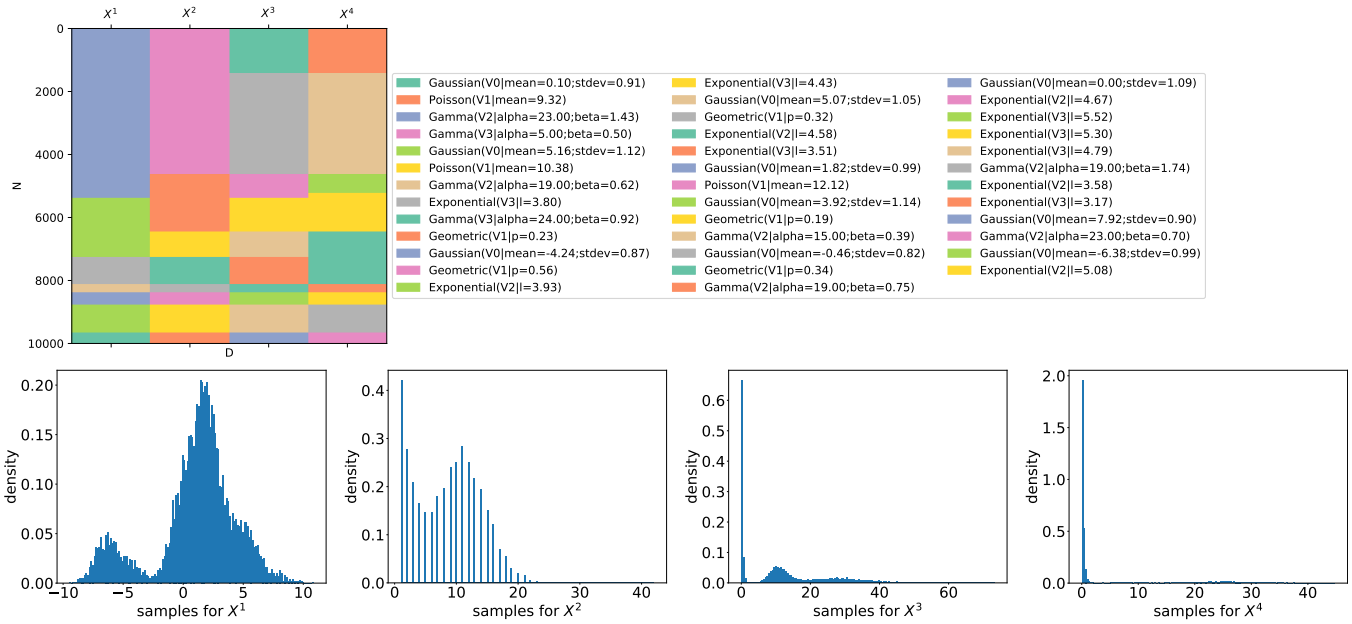

Figure 1: Synthetic data generation. (Top) An example of a partitioning for a synthetic dataset of $N = 10000$ samples and $D = 4$ features, where samples have been re-ordered to have contiguous partitions for the sake of visualization. Each partition is associated with a parametric likelihood model labeled by a color, instantiated with a set of parameters randomly drawn according to a sensible prior (see Appendix B). (Bottom) Histogram visualizations for the marginals of the 4 features generated.

validation set. Then we run the Gibbs sampler for 3000 iterations discarding the first 2000 samples for burn-in.

For the log-likelihoods in Fig 2a, we estimate them as the mean log-likelihood computed by ABDA over the test samples and averaged across the last 1000 samples of the Gibbs chain.

For recovering the global weights measuring uncertainty over the likelihood models (or statistical data types), we average over the predicted arrays belonging to the last 1000 samples of Gibbs and perform the cosine similarity between these vector and the ground truth one.

For the hard predictions, as shown in the confusion matrixes in Fig.2, we select, from each vector, the likelihood model (or statistical data type) scoring the highest weight in the weight vector.

D. Real-world datasets

D1. Density estimation and imputation datasets. For our experiments we use some real-world datasets from the UCI(Dheeru and Taniskidou 2017) repository. The number of instances and features are what we used for the models after preprocessing. We took the datasets as available from (Molina et al. 2018) and (Valera, Pradier, and Ghahramani 2017) and put them in the same tabular formalism where we labeled each feature to be either continuous or discrete. We removed binary features from them and we either randomly selected a certain percentage of missing values for the transductive case or we randomly split them into train, validation and test sets for the inductive case (see main text).

Abalone is a dataset of abalone in Tasmania that includes different physical measurements. It contains 4177 instances, 9 features and no missing values. **Adult** is a "Census Income" dataset used to predict low or high income, it contains 32561 instances and 13 features. **Australia** contains information about credit card applications in Australia, with 690 and 10 features. **Autism** has data about the autistic spectrum disorder screening in adults. It contains 3521 instances and 25 features. **Breast** describes 10 attributes about 681 instances of two types of breast cancer. **Chess** is a chess endgame database, with 28056 and 7 features. **Crx** is a dataset of credit card applications, with 651 instances and 11 attributes. **Dermatology** provides 366 instances and 34 about the Eryhemato-Squamous dermatological disease. **Diabetes** is a dataset of diabetes patient records, with

768 instances and 8 features. **German** is a classification dataset of people described by a set of attributes as good or bad credit risks. It has 1000 instances and 17 attributes. **Student** is a dataset of student performance in secondary education. With 395 instances and 20 features. **Wine** is a dataset about wine quality based on physicochemical tests. It contains 6497 instances, with 12 features.

D2. Outlier detection datasets. The datasets used for anomaly detection come from (Goldstein and Uchida 2016). **Thyroid** is a dataset of thyroid diseases provided by the Garavan Institute of Sydney, Australia. It contains 6916 instances, 21 features, and 250 outliers. **Letter** contains classification data for 26 capital letters of the English alphabet. It contains 1600 instances, 32 features, and 100 outliers. **Pen-*** are datasets about pen-based handwritten digit recognition. Pen-global contains 809 instances, 16 features, and 90 outliers. Pen-local contains 6724, 16 features, and 10 outliers. **Satellite** is a dataset that contains multi-spectral values of pixels in 3x3 neighborhoods in a satellite image and a label for the central pixel. It contains 5100 instances, 36 features, and 75 outliers. **Shuttle** is a dataset about the space shuttle with 46464 instances, 9 features and 878 outliers. **Speech** is a speech recognition dataset with 3686 instances, 400 features and 61 outliers.

E. Missing value estimation

E1. ABDA can efficiently marginalize over missing values during inference In order to be able to deal with missing values at inference time, we first need to provide ABDA with a LV structure that has been induced from the non-missing data entries only. Secondly, we need to marginalize over these entries while performing Gibbs sampling. This can be done efficiently by exploiting SPNs' ability to decompose marginal queries into simpler marginalizations at the leaf distributions (Darwiche 2003; **?**). When updating the posterior parameters for the leaf likelihood models, we keep track of counts belonging only to non-missing entries.

To this end, we adapted the likelihood-agnostic structure learning introduced by MSPNs (Molina et al. 2018). In a nutshell, in presence of missing entries in the training samples, these are not contributing to the evaluation of the RDC both for the partitioning along the rows of the data matrix (clustering), and for the partitioning along RVs or features (group independence seeking). Specifically, while transforming the data matrix through the copula transformation (for details, refer to (Molina et al. 2018)) we do not let missing entries contribute to the estimation of the empirical cumulative distribution. We reserve the same treatment for MSPNs as well.

E2. ABDA can estimate missing values efficiently We assume each sample $\mathbf{x}_n$ to be in the form $\mathbf{x}_n = (\mathbf{x}_n^o, \mathbf{x}_n^m)$, where $\mathbf{x}_n^o$ comprises observed values and $\mathbf{x}_n^m$ missing ones.

To evaluate the effectiveness of a generative model $\mathcal{M}$ in estimating missing values, we employ the probability of seeing the true missing entries $\bar{\mathbf{x}}_n^m$, i.e., $p_{\mathcal{M}}(\bar{\mathbf{x}}_n^m)$, as a standard approach indicating the ability of $\mathcal{M}$ of selecting the true value for a RV (Valera, Pradier, and Ghahramani 2017). This operation requires marginalizing over all observed values $\mathbf{x}_n^o$, which can be done efficiently with SPN-based models (Darwiche 2003; **?**).

To impute missing values, on the other hand, we resolve to compute the the *most probable explanation* (MPE) (Darwiche 2009) for the corresponding RVs, that is:

$$\tilde{\mathbf{x}}_n^m = \arg\max_{\mathbf{x}_n^m} p_{\mathcal{M}}(\mathbf{x}_n^m | \mathbf{x}_n^o) = \arg\max_{\mathbf{x}_n^m} p_{\mathcal{M}}(\mathbf{x}_n^m, \mathbf{x}_n^o)$$

Note that this task is performed by factorization-based models like ISLV efficiently, since each RVs appearing in $\mathbf{x}_n^m$ is imputed *independently* from others. Conversely, ABDA and MSPNs can leverage all other RVs while performing imputation. The price to pay for this is that, for general SPNs, exact MPE imputation is NP-Hard (**?**; Choi and Darwiche 2017). Nevertheless, reasonably accurate approximations to this tasks exist and allow to evaluate an SPN structure in linear time (Darwiche 2003; Chan and Darwiche 2006). In brief, they employ a bottom-up propagation step for observing $\mathbf{x}_n^o$, at first, while performing a Viterbi-like decoding top-down traversal of the SPN structure to retrieve the maximally likely states for the RVs in $\mathbf{x}_n^m$. Please refer to (**?**) for more details.

E3. Missing value estimation experiments For our missing value estimation experiments in the transductive scenarios we monitor both the mean log-likelihood for the missing data entries in a run and the normalized root mean squared error (NRMSE) w.r.t. the imputation provided by each model (please refer to the previous Section for how to compute them in ABDA).

Figures 2 and 3 report the feature-wise plots for the above mentioned metrics. As a general trend, one can see how ISLV performs less well than ABDA and MSPNs and how the difference between these two is often significant only on a small subset of the modeled features. Generally, it seems that ABDA has the advantage of modeling continuous RVs, confirming that the performing discretization as MSPNs do is potentially a problem. However, note also that in such cases, MSPNs log-likelihood values might be inflated by their adoption of piecewise polynomial approximations as likelihood models. Indeed, since they are fitted on the small range of values available during learning, their density is renormalized to integrate to 1. On the other hand, since the likelihood distributions used in ABDA generally have infinite tails, their mass is not concentrated only on that observed range.

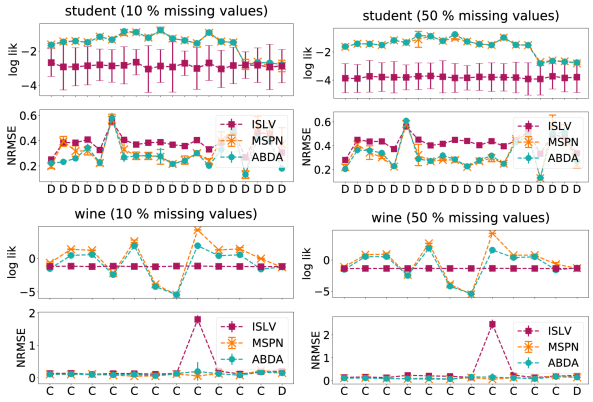

Figure 2: Performance (test log-likelihood, normalized RMSE feature-wise) on real datasets (continued).

F. Anomaly robustness and detection

F1. ABDA is robust to corrupted data and outliers During inference, ABDA will tend to assign anomalous samples into low-weighted mixture components, i.e., sub-networks of the underlying SPN or leaf likelihood models. If more that one anomalous sample is relegated to the same partition, they will form a *micro-cluster* (Chandola, Banerjee, and Kumar 2009).

At test time, ABDA would assign low probability to outliers or novel samples. The (log-)likelihood of the whole joint distribution can be employed as a score for inliers (Schölkopf et al. 2001; Chandola, Banerjee, and Kumar 2009) and by proper thresholding, one can have a decision rule to detect anomalies.

This process can be repeated at each node in the underlying SPN structure, thus enabling ABDA to perform *hierarchical anomaly detection* to decide whether a sample—or just a subset of features in a sample (i.e., contextual outliers (Chandola, Banerjee, and Kumar 2009))—is anomalous with respect to the distribution induced at that node.

As a qualitative example, Figure 4 illustrates how ABDA partitioned the Shuttle data, correctly relegating most of the outliers into micro-clusters. Those not associated to a single micro-cluster are still belonging to the tail of the distribution modeling the time feature (x-axis).

F2. Unsupervised outlier detection We quantitatively evaluate how ABDA is able to perform *unsupervised pointwise anomaly detection* (Chandola, Banerjee, and Kumar 2009), that is detect outliers available at training time, without having access to their label.

We follow the experimental setting of (Goldstein and Uchida 2016), in which a set of UCI binary classification datasets have been processed to include all samples from one class (*inliers*) and a small percentage of dissimilar samples drawn from the other class (*outliers*). Please refer to (Goldstein and Uchida 2016) for detailed dataset statistics.

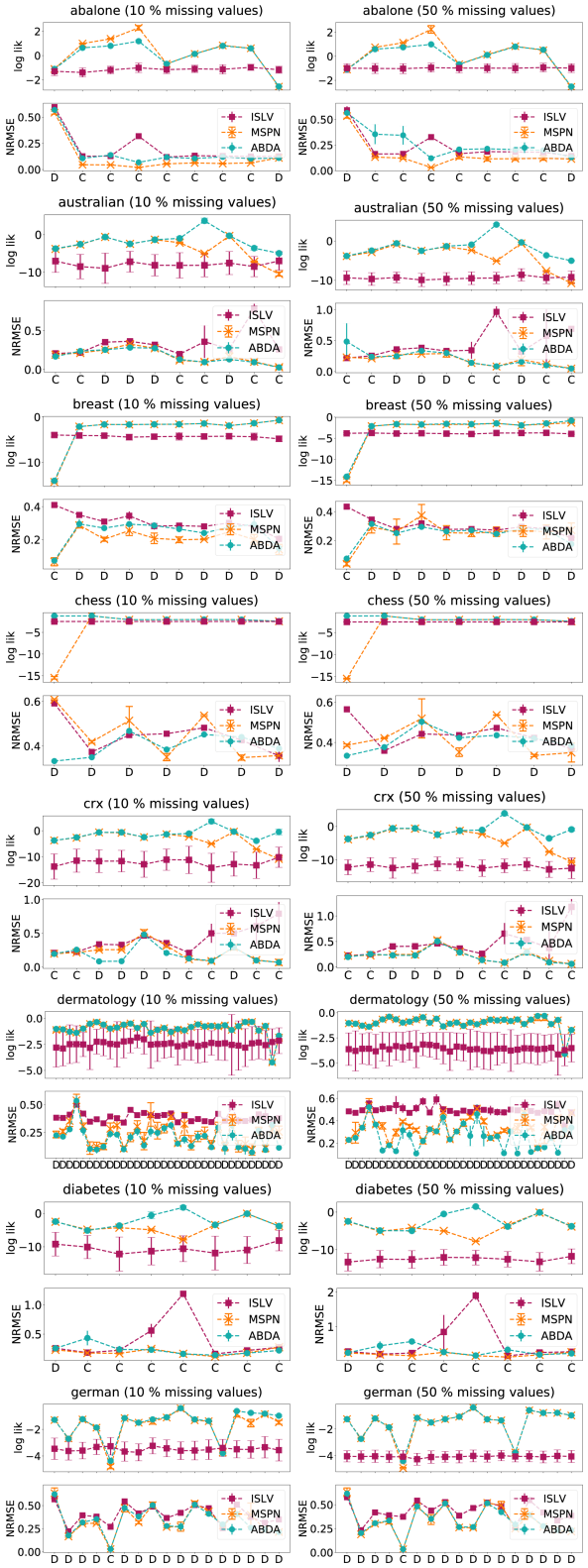

Figure 3: Performance (test log-likelihood, normalized RMSE feature-wise) on real datasets.

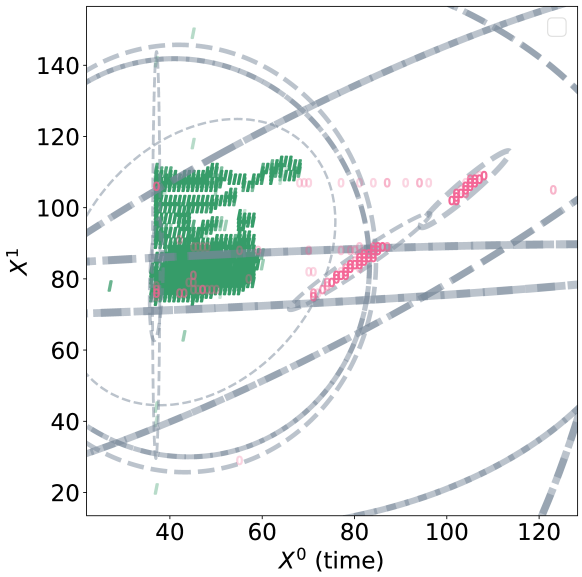

Figure 4: ABDA separates outliers (pink 'O') from inliers (green 'I') via hierarchical partitioning on Shuttle data. Each ellipse represents a partition induced by the underlying SPN w.r.t. the first two features (X^0 and X^1) of the shuttle data. The thickness of the ellipses is inversely proportional to the number of samples in the corresponding partition, therefore it also indicates the depth of the partition—finer lines for finer grain clusters.

Once trained on these datasets, all models involved are required to output a score for each training sample, the higher the most confident the model is about the sample being an outlier. For ABDA, MSPN and ISLV we employ the sample negative log-likelihood as the outlier score.

As competitors we include one-class support vector machines (oSVM) (Schölkopf et al. 2001), the local outlier factor (LOF) (Breunig et al. 2000) and the histogram-based outlier score (HBOS) (Goldstein and Dengel 2012). Accuracy performances for all models are measured in terms of the area under the receiving operating curve (AUC ROC) computed w.r.t. to the outlier scores.

We follow (Goldstein and Uchida 2016), and instead of picking one single model out of a grid search to optimize its hyperparameters, we average over all AUC scores from the cross validated models. We emply an RBF kernel ($\gamma = 0.1$) for oSVMs, varying $\nu \in \{0.2, 0.4, 0.6, 0.8\}$. For LOF, we run it by evaluating a number of nearest neighbors $k \in \{20, 30, 40, 50\}$. For HBOS, instead, we explored these configurations with the Laplacian smoothing factor $\alpha \in \{0.1, 0.2, 0.5\}$ and number of bins in $\{10, 20, 30\}$. For ABDA and MSPNs we set the minimum number of instances m to 5% of the number of instances and explored the RDC coefficient $\rho \in \{0.1, 0.3, 0.5\}$. For ISLV, we run it several times for $k \in \{\lfloor \frac{D}{3} \rfloor, \lfloor \frac{D}{2} \rfloor, \lfloor \frac{2D}{3} \rfloor\}$.

Mean and average AUC scores are reported in Table 2. On the Thyroid dataset we were not able to run ISLV, while on the datasets where there is no value in Table 2 it either did not converge in 72hrs or in 1000 iterates. Otherwise, as for ABDA, we report the average AUC ROC w.r.t. the 100 last Gibbs samples after a burn-in of 3000 iterations. For KDD99, we observed ABDA converging in 1000 iterates (burn-in) and within the 72hrs limit.

Table 2: Unsupervised anomaly detection with ABDA. Mean and standard deviation (small) AUC ROC scores for all models and datasets.

	oSVM	LOF	HBOS	ABDA	MSPN	ISLV
Aloi	51.71 ±0.02	**74.19** ±0.70	52.86 ±0.53	47.20 ±0.02	51.86 ±1.04	-
Thyroid	46.18 ±0.39	62.38 ±1.04	62.77 ±3.69	**84.88** ±0.96	77.74 ±0.33	-
Breast	45.77 ±11.12	98.06 ±0.70	94.47 ±0.79	**98.36** ±0.07	92.29 ±0.99	82.82 ±1.31
Letter	63.38 ±17.60	**86.55** ±2.23	60.47 ±1.80	70.36 ±0.01	68.61 ±0.23	61.98 ±0.44
Kdd00	53.40 ±3.63	46.39 ±1.95	87.59 ±4.70	**99.79** ±0.10	70.17 ±15.97	-
Pen-global	46.86 ±1.02	87.25 ±1.94	71.93 ±1.68	**89.87** ±2.87	75.93 ±0.14	75.54 ±5.40
Pen-local	44.11 ±6.07	**98.72** ±0.20	64.30 ±2.70	90.86 ±0.79	79.25 ±5.41	65.52 ±13.08
Satellite	52.14 ±3.08	83.51 ±11.98	90.92 ±0.16	94.55 ±0.68	**95.22** ±0.45	-
Shuttle	89.37 ±5.13	66.29 ±1.69	**98.47** ±0.24	78.61 ±0.02	94.96 ±2.13	-
Speech	45.61 ±3.64	**49.37** ±0.87	47.47 ±0.10	46.96 ±0.01	48.24 ±0.67	-

It is clearly visible from Table 2 how ABDA can perform as good as, or even better in most cases than standard outlier detection models. Moreover, this unsupervised task demonstrates how ABDA is indeed more resilient to outliers and corrupt data w.r.t. MSPNs. Indeed, while the greedy structure learning procedure adopted by both MSPNs and ABDA can get fooled by some outliers, grouping them to inliers in the same partition, Bayesian inference in ABDA might be able to re-assign them to low-probability partitions.

G. Exploratory pattern extraction

Here we discuss how to employ ABDA to unsupervisedly extract patterns among RVs—implying dependency among them—in a similar fashion to what *Association Rule Mining* (ARM) does (Agrawal and Srikant 1994). The aim is, on the one hand, to automatize the way such patterns can be extracted, filtered and presented (as a ranking) to the user, and on the other hand, to extract a small set able to explain to the user—in a more interpretable way—what correlations ABDA has captured.

In ARM, given some data over discrete (generally assumed to be binary for simplicity) RVs $\mathbf{X} = \{X^1, \ldots, X^D\}$ one is interested in finding a set of association rules $\{\mathcal{R}_i\}$ where each rule $\mathcal{R} : \mathcal{A} \rightarrow \mathcal{C}$ is composed by an antecedent, $\mathcal{A} = P_1 \wedge P_2, \ldots, P_a$, and consequent, $\mathcal{C} = P_{a+1} \wedge P_{a+2}, \ldots, P_{a+s}$, part, both of

which are conjunctions of *patterns*, i.e. assignments to some RVs in $\mathbf{X}$. For instance one rule might state:

$$P_1 : X^1 = 0 \wedge P_2 : X^3 = 1 \rightarrow P_3 : X^2 = 1$$

. which would state that whenever the antecedent is observed, i.e., P_1 and P_2 are satisfied, it is *likely* to observe X^2 set to value 1.

To quantify the "importance" of a rule—thus having a quantitative way to *filter* and *rank* rules— one computes for a rule $\mathcal{R}$ measures like its *support* and *confidence*. The former being defined as:

$$supp(\mathcal{R} : \mathcal{A} \rightarrow \mathcal{C}) = \frac{\#\{P_1, \ldots, P_a, P_{a+1}, \ldots, P_{r+s}\}}{N}$$

where the numerator indicates the number of samples for which the patterns $P_1, \ldots, P_{a+s}$ are jointly satisfied and N is the number of samples in the data. Confidence of a rule, instead is the ratio of its support and that of the antecedent:

$$conf(\mathcal{R} : \mathcal{A} \rightarrow \mathcal{C}) = \frac{supp(\mathcal{A}, \mathcal{C})}{supp(\mathcal{A})} = \frac{\#\{P_1, \ldots, P_{a+s}\}}{\#\{P_1, \ldots, P_a\}}$$

G1. Probabilistic patterns in ABDA Clearly, the notion of support is the maximum likelihood estimation for the joint probability of its patterns

$$p(P_1 \wedge \ldots \wedge P_a \wedge P_{a+1} \wedge \ldots \wedge P_{a+s})$$

Analogously, the confidence of a rule $\mathcal{R}$ is estimator for the conditional probability

$$p(P_{a+1} \wedge \ldots \wedge P_{a+s} | P_1 \wedge \ldots \wedge P_a)$$

.

By properly defining what a pattern is in ABDA, we might directly compute the above probabilities efficiently, by exploiting marginalization over the SPN structure of ABDA. We also have to take into consideration how to deal with continuous RVs natively, since the classical formulation of ARM patterns would require binarization.

We define an *interval pattern* over RV X^i as the event $P : \pi^i_{low} \leq X^i < \pi^i_{high}$ where $\pi^i_{low} < \pi^i_{high}$ are two valid values from the domain of X^i. The probability of a single pattern is therefore:

$$p(P : \pi^i_{low} \leq X^i < \pi^i_{high}) = \int_{\pi^i_{low}}^{\pi^i_{high}} f(X^i) dX^i$$

where f is the density function of X^i.

Consider now an dependency rule of the form $P_1, \ldots P_a \rightarrow P_{a+1}, \ldots, P_{a+s}$, its support can then be computed as multivariate integral over $P_1 \ldots P_{a+s}$

$$\int_{\pi^1_{low}}^{\pi^1_{high}} \ldots \int_{\pi^{a+s}_{low}}^{\pi^{a+s}_{high}} f(\mathbf{X}) dX^1 \ldots X^{a+s}$$

If the joint density f decomposes as an SPN structure, solving the above integral would require resolving univariate integrals at the leaves—which is doable assuming tractable univariate distributions there as we do in ABDA[1]— and propagate the computed probabilities upwards.

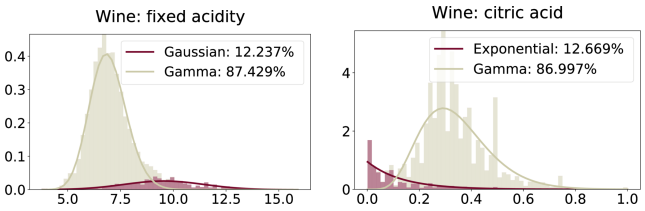

Figure 5: Data exploration on Wine data. Density estimation provided by ABDA on the Wine quality dataset. ABDA identifies the two modalities in the data induced by red and white wine over three different features.

By doing so we have a way to exploit the SPN structure in ABDA to efficiently compute the support of a rule—a collection of dependency pattern—here extended to continuous RVs. Therefore we can rank rules by computing their support. How to extract rules in a (semi-)automatic way?

G2. Automatic pattern mining in ABDA The most straightforward approach to extract patterns and rules via ABDA would mimic *Apriori*, the stereotypical ARM algorithm. In a nutshell, first patterns of length 1 are mined (i.e., involving a single RV), the collection of patterns are combined by enumeration while at the same time filtering out patterns whose support is less than a user-specified threshold ρ (Agrawal and Srikant 1994).

The main issue would be how to determine the atomic patterns in the form $P : \pi^i_{low} \leq X^i < \pi^i_{high}$, since we X^i can also be continuous, and hence we can possibly find an infinite number of intervals $[\pi^i_{low}, \pi^i_{high})$ from its domain. The solution comes from ABDA having already applied this partitioning during inference. Indeed, SPN leaves in ABDA already describe tractable distributions *concentrated* on a portion of the whole domain for a feature. Given a user defined percentile threshold $\lambda \in [0, 1]$, we can determine the interval containing λ percentage of the probability mass of density $f(X^i)$. For instance, for $\lambda = 90\%$, we might easily find $[\pi^i_{low}, \pi^i_{high})$ as the the 5% and 95%percentiles of $f(X^i)$.

G3. Wine data As a concrete example, see Figure 5, depicting two marginal distributions as learned by ABDA for two features of the Wine dataset, $X^1 =$ FixAcid and $X^2 =$ CitAcid. One might extract the following four patterns from them by fixing a threshold of 80% of probability:

$$P_1 : 5.8 \leq \text{FixAcid} < 8.1, \quad \text{FixAcid} \sim \mathcal{N}$$

$$P_2 : 0.2 \leq \text{CitAcid} < 0.5, \quad \text{CitAcid} \sim Gamma$$

[1]Even if some densities would require to approximate it, it would be still doable

$$P_3 : 7.1 \leq \text{FixAcid} < 12.0, \quad \text{FixAcid} \sim Gamma$$
$$P_4 : 0.0 \leq \text{CitAcid} < 0.3, \quad \text{CitAcid} \sim Exp$$

After these atomic patterns have been extracted from leaves, one can first determine their support according to the whole SPN $\mathcal{S}$ Then, conjunctions of patterns may be mechanically combined as in Apriori, their support computed and filtered out if it is lower than the user-defined threshold ρ.

Back to the Wine features in Figure 5, one could compose the following composite pattern via conjunctions, by noting that the extracted patterns are belonging to product node children and thus referring to samples belonging the same partition (same color across histograms):

$$P_1 : 5.2 \leq \text{FixAcid} < 8.1 \wedge P_2 : 0.2 \leq \text{CitAcid} < 0.5$$
$$P_3 : 7.1 \leq \text{FixAcid} < 12.0 \wedge P_4 : 0.0 \leq \text{CitAcid} < 0.3$$

G4. Abalone data For a more complex example, we employ ABDA on the Abalone dataset (see Appendix D). The dataset contains physical measurements (features) over abalone samples, namely: the Sex of the specimen ('male', 'female' or 'infant'), its Length, Diameter, Height, Whole weight, Shucked Weight and Viscera Weight, its Shell Weight and the number of rings in it.

We employ ABDA on it by running it for 5000 iterates, stopping the SPN LV structure learning until 10% of the data was reached and employing 0.7 as the RDC independence threshold.

Figure 6 shows the marginal distributions for all features upon which the densities as fit by ABDA. Each density is colored by a unique color—shared across all features—indicating the partition $\mathcal{K}$ induced by the SPN $\mathcal{S}$. Each of such partitions is also labeled with an integer, which serves the purpose to indicate the path—appearing in each legend entry—inside $\mathcal{S}$ leading from the top partition (numbered as 0, indicating the whole data matrix) to the finer grained one. For instance, the path $0 \to 2 \to 20$ appearing in the legend of the Length attribute, and associated to the purple partition numbered 20, indicates that such a partition is contained in the partition number 2 which in turn belongs to the initial partition.

By starting from the Sex feature, one can see that two partitions are discovered (1 and 2) in which the first mostly represents samples belonging to the first and last mode of the categorical features, indicating 'male' and 'infant' abalone samples, while the second comprises mostly 'female' individuals. From these initial *conditioning* on the sex, all other feature correlations are characterized.

For instance, the green partition (11) is a sub-partition of the 'Male' clustering, and shows a cross correlation across the Height feature and the Weight one, as one would expect. Additionally, the green partition itself later on splits into two sub-population, the gray (52) and yellow (53) representing correlations among the Shucked and Viscera weights, indicating a mode in the abalone population for specific ranges of these features.

By looking at all partitions and at each density belonging to them, one can build in a mechanical way (composite) patterns and rank them by their support. For this analysis of the Abalone data, these are the top 5 patterns that the ABDA automatically extracts:

$$\begin{aligned}
\mathcal{P}_1 : \quad & 0.088 \leq \text{Height} < 0.224 \quad \wedge \\
& 0.158 \leq \text{ShellWeight} < 0.425 \quad (supp(\mathcal{P}_1) = 0.507) \\
\mathcal{P}_2 : \quad & 0.483 \leq \text{Length} < 0.684 \quad \wedge \\
& 0.364 \leq \text{Diameter} < 0.547 \quad (supp(\mathcal{P}_2) = 0.489) \\
\mathcal{P}_3 : \quad & 0.596 \leq \text{WholeWeight} < 1.040 \quad \wedge \\
& 0.205 \leq \text{ShuckedWeight} < 0.491 \quad \wedge \\
& 0.076 \leq \text{VisceraWeight} < 0.281 \quad (supp(\mathcal{P}_3) = 0.223) \\
\mathcal{P}_4 : \quad & 0.596 \leq \text{WholeWeight} < 1.040 \quad \wedge \\
& 0.205 \leq \text{ShuckedWeight} < 0.491 \quad \wedge \\
& 0.076 \leq \text{VisceraWeight} < 0.281 \quad (supp(\mathcal{P}_4) = 0.202) \\
\mathcal{P}_5 : \quad & 0.313 \leq \text{Length} < 0.554 \quad \wedge \\
& 0.230 \leq \text{Diameter} < 0.438 \quad \wedge \\
& 0.023 \leq \text{Height} < 0.197 \quad \wedge \\
& 0.165 \leq \text{WholeWeight} < 0.639 \quad \wedge \\
& 0.037 \leq \text{ShuckedWeight} < 0.408 \quad \wedge \\
& 0.019 \leq \text{VisceraWeight} < 0.192 \quad \wedge \\
& 0.027 \leq \text{ShellWeight} < 0.27 \quad \wedge \\
& 5 \leq \text{Rings} < 13 \quad (supp(\mathcal{P}_5) = 0.111)
\end{aligned}$$

H. ABDA vs MSPN: accuracy and robustness

In our experimental section, we selected diverse datasets—w.r.t. size and feature heterogeneity—from both the ISLV and MSPN original papers with the aim to have a common, fair experimental setting. Moreover, we run for ABDA and MSPN a grid search in the same hyperparameter space, *to block the effect of building the SPN LV hierarchy*. Striving for automatic density analysis, such a grid search has been limited only to the independence test (RDC) threshold parameter, while original MSPNs were validated over more than five parameters.

For a more straightforward comparison in the original experimental setting of (Molina et al. 2018), we cross-validate ABDA also on one additional parameter, $m \in \{20\%, 10\%, 5\%\}$, the minimum number of samples in a partition (as a percentage over the whole number of samples) to stop the learning process to recursively further partition it. This parameter governs the degree of overparametrization of the learned LV structure.

Table 3 reports the average test log-likelihoods for the best ABDA model on the validation set, while MSPN results, considering unimodal isotonic regression applied to piecewise linear leaf distribution approximations, are directly taken from (Molina et al. 2018). As one can

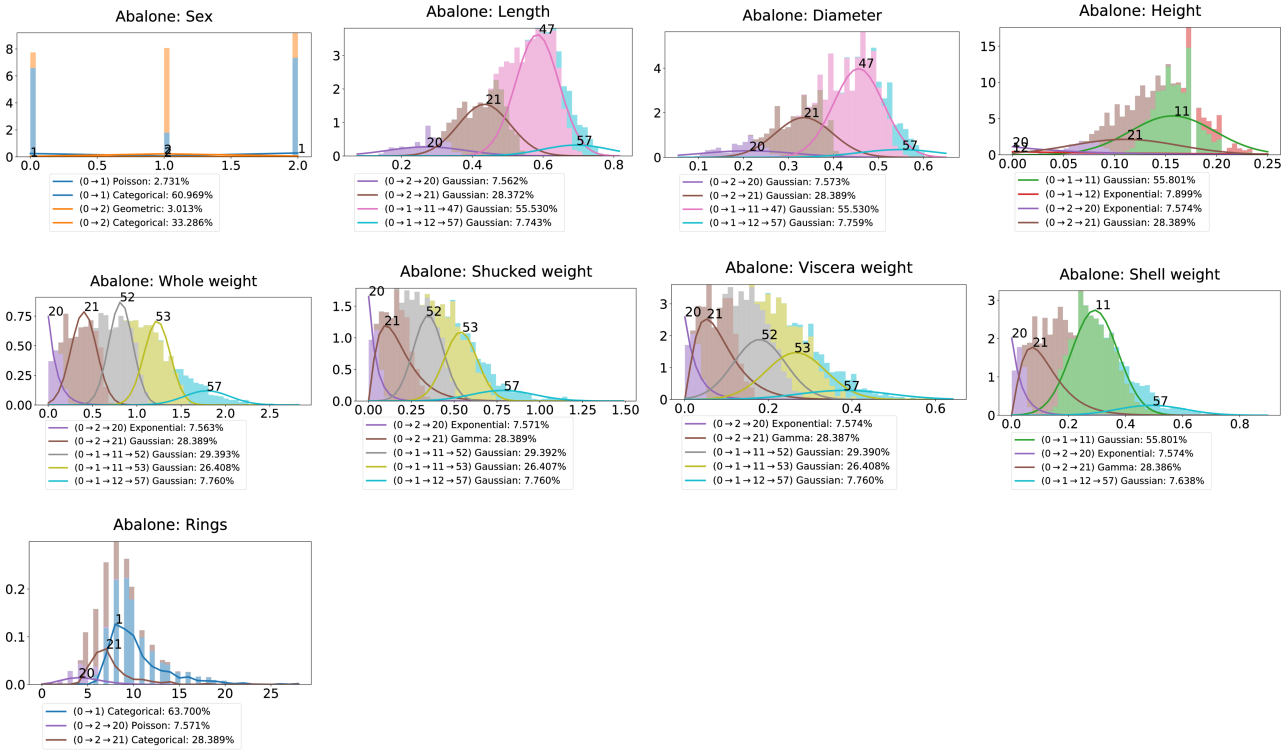

Figure 6: Data exploration on Abalone. Density estimation and likelihood model discovery provided for each feature of the Abalone dataset. Likelihood models are labeled by the partition id comprising the associated sampled and whose color is shared across the plots.

Table 3: Density estimation by ABDA vs MSPN over the original 14 datasets. Best average test log-likelihoods in bold.

	ABDA	MSPN		ABDA	MSPN
Anneal-U	**-2.65**	-38.31	Austr.	**-17.70**	-31.02
Auto	-70.62	**-70.06**	Bal-scale	**-7.13**	-7.30
Breast	-25.46	**-24.04**	Bre.-cancer	**-9.61**	-9.99
Cars	-35.04	**-30.52**	Cleve	**-22.60**	-25.44
Crx	**-15.53**	-31.72	Diabete	**-17.48**	-27.24
German	**-32.10**	-32.36	Ger.-org	**-26.29**	-27.29
Heart	**-23.39**	-25.90	Iris	-2.96	**-2.84**

see, *ABDA provides competitive results to MSPNs* even in this experimental setting, still requiring less parameter tuning. This is due to i) the likelihood mixtures in ABDA better generalizing than piecewise models—since their support is limited to samples seen during training and possibly infinite tails of a distribution are not natively captured—and ii) our Bayesian inference providing indeed robust models.

To get a better understanding of how Bayesian inference in ABDA affects its robustness, we compare ABDA and MSPNs in an inductive scenario when we provide both with increasingly parametrized LV structures. More specifically, for both we explore deeper and deeper structures by letting the minimum sample percentage parameter m vary in $\{5\%, 1\%, 0.5\%\}$, while we fix the RDC independence threshold to 0.5. As one could expect, ABDA is at advantage since it performs an additional parameter learning step after the structure is provided. Nevertheless, it is worth measuring by *how much more robust* ABDA models are to an over-parametrization than MSPNs and, at the same time, evaluate *how much sparser* the SPN structure in ABDA gets, that is, quantifying how "blindly" one user can run ABDA with limited or no possibility to cross-validate.

To this end, we measure the relative percentage of decreasing mean test log-likelihoods w.r.t. the mean test log-likelihood achieved by each model in the case in which $m = 5\%$. Moreover, we measure the sparsity of a SPN structure in ABDA as the ratio between the number of *relevant* product nodes and leaf likelihood components over the number of all nodes and likelihood models. In this case, the relevancy of a node is measured by the fact that at least some samples have been associated to the corresponding node (or likelihood component) partition.

Table 4 reports these values for the 12 datasets involved in our density estimation and imputation experiments. Firstly, one can see that on the Chess and German datasets, growing a deeper SPN structure is not even possible, in general. However, for all other datasets, it is clear that MSPN accuracy tends to drop more quickly than ABDA's, whose test predictions are not

Algorithm 1 LearnStructureABDA ($\mathbf{X}$, $N_{\min}$, ρ)

Input: an $N \times D$ data matrix $\mathbf{X}$; $N_{\min}$: minimum number of instances to split; ρ: threshold of significance

if $D = 1$ **then**
  $\{\mathbf{X}_c\}_{c=1}^{C} \leftarrow$ clusterSamplesRDC($\mathbf{X}$)
  if $C = 1$ **then**
    $\mathcal{S} \leftarrow$ createLikelihoodDictionary($\mathbf{X}$)
  else
    $\mathcal{S} \leftarrow \sum_{c=1}^{C} \frac{|\mathbf{X}_c|}{N}$ LearnStructureABDA($\mathbf{X}_c, N_{\min}, \rho$)
else if $N < N_{\min}$ **then**
  $\mathcal{S} \leftarrow \prod_{d=1}^{D}$ createLikelihoodDictionary($\mathbf{X}^d$)
else
  $\{\mathbf{X}_c\}_{c=1}^{C} \leftarrow$ splitFeaturesRDC($\mathbf{X}, \rho$)
  if $C > 1$ **then**
    $\mathcal{S} \leftarrow \prod_{c=1}^{C}$ LearnStructureABDA($\mathbf{X}_c, N_{\min}, \rho$))
  else
    $\{\mathbf{X}_c\}_{c=1}^{C} \leftarrow$ clusterSamplesRDC($\mathbf{X}$)
    $\mathcal{S} \leftarrow \sum_{c=1}^{C} \frac{|\mathbf{X}_c|}{N}$ LearnStructureABDA($\mathbf{X}_c, N_{\min}, \rho$)

Output: $\mathcal{S}$

only more stable for different values of m but sometimes even better. Lastly, the sparsity of the SPN structure in ABDA is indeed significantly increasing as m becomes smaller[2].

I. LV Structure Learning

To provide ABDA an initial hierarchy of LVs $\mathbf{Z}$, we devise a variant of the structure learning algorithm introduced in (Molina et al. 2018) to learn MSPNs. Specifically, we i) adopt the same RDC-based (Lopez-Paz, Hennig, and Schölkopf 2013) data partitioning (over samples and features) to introduce sum and product nodes into the SPN to be learned, while ii) extending it to deal with missing entries in the data matrix. Lastly, iii) we model univariate distributions as the SPN leaves, as mixture models over user-provided likelihood dictionaries.

Algorithm 1 illustrates the main procedure called for the structure learning process. In order to handle missing data, we need to modify the sample partitioning and the feature partitioning sub-procedures sketched in Algorithms 3 and 4 respectively. Both procedures rely on the computation of the empirical cumulative distribution function (ECDF), illustrated in Algorithm 2. There, the marginal ECDFs are computed only on the observed data entries. Afterwards, the random projection of the full data matrix is obtained by treating missing entries as zeros, such that their contribution in the scalar products is none.

[2]For a reference, consider that the same structure in an MSPN is never sparse, since a partition in the training data must contain at least one sample. Therefore the sparsity level for $m = 5\%$ is already meaningful for our investigation.

Table 4: Robustness of MSPN and ABDA w.r.t. over-parametrized structures, i.e. when the minimum percentage of samples (m) to split a partition is let vary up to 0.5% Best relative test log-likelihoods improvement w.r.t. test log likelihoods for $m = 5\%$ is reported in bold (in parenthesis). The last column reports the sparsity of ABDA structures (s) after inference, for each setting.

dataset	m	MSPN		ABDA		s
abalone	5%	10.33		4.94		0.15
	1%	10.16	(−**1.68**%)	4.74	(−4.13%)	0.04
	0.5%	9.59	(−7.15%)	5.00	(+**1.30**%)	0.03
chess	5%	-16.21		-12.54		0.66
	1%	-16.21	(−**0.00**%)	-12.54	(−**0.00**%)	0.69
	0.5%	-16.21	(−**0.00**%)	-12.54	(−0.00%)	0.69
german	5%	-31.85		-25.86		0.56
	1%	-31.85	(−**0.00**%)	-26.21	(−1.35%)	0.45
	0.5%	-31.85	(−**0.00**%)	-25.87	(−**0.05**%)	0.59
student	5%	-36.86		-28.93		0.25
	1%	-44.86	(−21.70%)	-29.16	(−**0.78**%)	0.09
	0.5%	-49.95	(−35.49%)	-29.32	(−**1.35**%)	0.05
wine	5%	-0.10		-9.20		0.14
	1%	-0.31	(−198.38%)	-8.43	(+**8.37**%)	0.17
	0.5%	-0.51	(−391.74%)	-8.73	(+**5.05**%)	0.07
dermat.	5%	-35.77		-24.96		0.36
	1%	-46.82	(−30.91%)	-25.14	(−**0.72**%)	0.20
	0.5%	-57.42	(−60.54%)	-25.16	(−**0.78**%)	0.14
anneal-U	5%	-151.09		7.68		0.12
	1%	-156.60	(−**3.64**%)	3.80	(−50.53%)	0.14
	0.5%	-166.26	(−**10.04**%)	4.33	(−43.53%)	0.08
austral.	5%	-37.67		-15.89		0.27
	1%	-38.93	(−3.32%)	-16.27	(−**2.38**%)	0.21
	0.5%	-40.08	(−6.37%)	-16.36	(−**2.98**%)	0.13
autism	5%	-41.16		-27.69		0.23
	1%	-42.30	(−2.78%)	-27.60	(+**0.35**%)	0.12
	0.5%	-42.54	(−3.34%)	-27.33	(+**1.29**%)	0.10
breast	5%	-33.42		-25.52		0.23
	1%	-34.87	(−4.32%)	-26.01	(−**1.89**%)	0.20
	0.5%	-35.84	(−7.23%)	-25.64	(−**0.46**%)	0.10
crx	5%	-37.57		-12.86		0.45
	1%	-38.79	(−3.26%)	-13.03	(−**1.37**%)	0.18
	0.5%	-39.39	(−4.87%)	-13.00	(−**1.07**%)	0.14
diabetes	5%	-31.55		-16.47		0.41
	1%	-31.70	(−**0.45**%)	-18.96	(−15.07%)	0.31
	0.5%	-31.65	(−**0.31**%)	-17.91	(−8.75%)	0.27
adult	5%	-74.98		-5.50		0.28
	1%	-76.15	(−1.55%)	-5.37	(+**2.50**%)	0.17
	0.5%	-77.25	(−3.03%)	-5.36	(+**2.62**%)	0.13

References

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

[Breunig et al. 2000] Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. LOF: identifying density-

Algorithm 2 ECDF $(\mathbf{X}^d, x)$

Input: a random feature $\mathbf{X}^d$ and an entry x

1: **if** $x \neq \text{NA}$ **then**
2: $\quad e \to \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}\{x_n^d \leq x\}$
3: **else**
4: $\quad e \to 0$

Output: e

Algorithm 3 clusterSamplesRDC $(\mathbf{X})$

Input: an $N \times D$ data matrix $\mathbf{X}$; k: number of random features (default: 10); s: variance of the random features (default: 1/6)

1: $\mathcal{C} \leftarrow \left\{ \mathsf{ECDF}(\mathbf{X}^d, x_i) \middle| x_i \in \mathbf{X}^d \right\}_{d=1}^{D}$
2: $(\mathbf{w}, b) \sim \mathcal{N}(\mathbf{0}_k, s\mathbf{I}_{k\times k})$
3: $\phi(\mathcal{C}_d) \leftarrow \sin(\mathbf{w} \cdot \mathcal{C}_d^T + b)$
4: $\mathcal{E} \leftarrow \{\phi(\mathcal{C}_1), \ldots, \phi(\mathcal{C}_D)\}$

Output: $\mathsf{KMeans}(\mathcal{E}, 2)$ //clustering into two components

Algorithm 4 splitFeaturesRDC $(\mathbf{X}, \rho)$

Input: an $N \times D$ data matrix $\mathbf{X}$; ρ: threshold of significance; k: number of random features (default: 10); s: variance of the random features (default: 1/6)

1: **for each** $d \in \{1 \ldots \mathbf{D}\}$ **do**
2: $\quad \mathcal{C}_d \leftarrow \mathsf{ECDF}(\mathbf{X}^d, x_i)$
3: $\quad (\mathbf{w}, b) \sim \mathcal{N}(\mathbf{0}_k, s\mathbf{I}_{k\times k})$
4: $\quad \phi(\mathcal{C}_d) \leftarrow \sin(\mathbf{w} \cdot \mathcal{C}_d^T + b)$
5: $\mathcal{G} \leftarrow \mathsf{Graph}(\{\})$
6: **for each** $i, j \in \{1 \ldots \mathbf{D}\}$ **do**
7: $\quad c_{i,j} \leftarrow \mathsf{CCA}(\phi(\mathcal{C}_i), \phi(\mathcal{C}_j))$
8: $\quad$ **if** $c_{i,j} > \alpha$ **then**
9: $\quad\quad \mathcal{G} \leftarrow \mathcal{G} \cup \{(i, j)\}$

Output: $\mathsf{ConnectedComponents}(\mathcal{G})$

based local outliers. In *ACM sigmod record*, volume 29, 93–104.

[Chan and Darwiche 2006] Chan, H., and Darwiche, A. 2006. On the robustness of most probable explanations. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'06, 63–71. Arlington, Virginia, United States: AUAI Press.

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

[Choi and Darwiche 2017] Choi, A., and Darwiche, A. 2017. On relaxing determinism in arithmetic circuits. In *Proceedings of ICML*, 825–833.

[Cowles and Carlin 1996] Cowles, M. K., and Carlin, B. P. 1996. Markov chain monte carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association* 91(434):883–904.

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

[Darwiche 2009] Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge.

[Dheeru and Taniskidou 2017] Dheeru, D., and Taniskidou, E. K. 2017. Uci machine learning repository. *University of California, Irvine, School of Information and Computer Sciences*.

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

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

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

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

[Molina et al. 2018] 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, Natarajan, and Kersting 2017] 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.

[Murphy 2012] Murphy, K. P. 2012. *Machine Learning: A Probabilistic Perspective*. MIT Press.

[Schölkopf et al. 2001] 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.

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

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

[Vergari, Di Mauro, and Esposito 2015] 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, Di Mauro, and Esposito 2018] Vergari, A.; Di Mauro, N.; and Esposito, F. 2018. Visualizing and understanding sum-product networks. *MLJ*.