Learning Weighted Model Integration Distributions

Paolo Morettin,[1*] **Samuel Kolb,**[2*] **Stefano Teso,**[1†] **Andrea Passerini**[1]
[1]University of Trento, Italy
{paolo.morettin, stefano.teso, andrea.passerini}@unitn.it
[2]KU Leuven, Belgium
samuel.kolb@cs.kuleuven.be

Abstract

Weighted model integration (WMI) is a framework for probabilistic inference over distributions with discrete and continuous variables and structured supports. Despite the growing popularity of WMI, existing density estimators ignore the problem of learning a structured support, and thus fail to handle unfeasible configurations and piecewise-linear relations between continuous variables. We propose LARIAT, a novel method to tackle this challenging problem. In a first step, our approach induces an SMT($\mathcal{LRA}$) formula representing the support of the structured distribution. Next, it combines the latter with a density learned using a state-of-the-art estimation method. The overall model automatically accounts for the discontinuous nature of the underlying structured distribution. Our experimental results with synthetic and real-world data highlight the promise of the approach.

1 Introduction

Weighted model integration (WMI) is a recent framework for probabilistic inference in hybrid domains, that is, domains with both discrete and continuous variables. A key feature of WMI is that it enables reasoning with highly *structured* distributions that include arbitrary combinations of logical and linear constraints. As previously recognized in work on *discrete* structured distributions (e.g. (Liang, Bekker, and Van den Broeck 2017)), these arise in important real-world applications because of, e.g., physical and mechanical limitations (Afshar, Sanner, and Webers 2016) or safety requirements (Amodei et al. 2016).

Despite their relevance, learning these *WMI distributions* (formally defined in the Background) from data has not been considered so far. Indeed, density estimators for hybrid domains entirely ignore the problem of identifying and learning the support of the distribution. State-of-the-art approaches like Mixed Sum-Product Networks (MSPNs) (Molina et al. 2018) and Density Estimation Trees (DETs) (Ram and Gray 2011) can only model bounding-box constraints over individual variables, and do not admit oblique supports, which are necessary to capture linear interactions between continuous quantities. Failing to account for the support can lead to substantial inaccuracies in density estimation, especially for high-density regions lying at the border of the support. Most importantly, acquiring a faithful support estimate is necessary to guarantee that invalid or harmful configurations cannot be generated, a crucial feature in safety-critical applications, e.g., reliable machine learning and safe AI (Amodei et al. 2016).

Inspired by work on learning *discrete* structured distributions (Liang, Bekker, and Van den Broeck 2017), we contribute LARIAT (LeARning to IntegrATe), a novel approach for learning WMI distributions from data. Our approach breaks the learning problem into two steps. In a first step, a (non-trivial) support is learned from the data. This is accomplished by a novel extension of the approach of (Kolb et al. 2018b) for learning *satisfiability modulo linear real arithmetic* formulas (Barrett et al. 2009). Our generalization extends the original approach to learning from positive-only data, and inherits the ability to natively acquire oblique and non-convex constraints. In a second step, a candidate density is obtained by employing a state-of-the-art non-parametric estimator, e.g., a DET or an MSPN. Finally, INCAL automatically adjusts (normalizes) the learned density to be consistent with the learned support, thus allocating zero mass to infeasible configurations while guaranteeing that the resulting model is a valid WMI distribution.

Summarizing, our main contributions are: (1) INCAL+, an extension of the SMT($\mathcal{LRA}$) learning algorithm of (Kolb et al. 2018b) to support learning; (2) LARIAT, a method to estimate WMI distributions that combines structured supports (either learned by INCAL+ or provided by a domain expert) and state-of-the-art density estimators, e.g., DETs and MSPNs; (3) Extensive experiments showing the advantages of LARIAT over standard density estimators in cases where the underlying distribution has a non-trivial support.

2 Related Work

Non-parametric multivariate density estimation is a venerable field of study. Textbook approaches include histograms (Fix and Hodges Jr 1951) and kernel density estimation (Gray and Moore 2003). Histograms partition the

[*]Equal contributions.
[†]This research was performed while ST was working at KU Leuven.

space into regularly sized bins, each associated to a constant density. Adaptive histograms fit a local regressor (e.g., a spline) at each bin to capture the finer details while allowing the bin size to grow adaptively (Jing, Koch, and Naito 2012). Kernel density estimation (KDE) fits a local density around each datapoint. The runtime of these methods grows quickly with the number of dimensions and/or datapoints. Multivariate adaptive regression splines (MARS) (Friedman 1991) is another classical, greedy space-partitioning model (with polynomial leaves) that tends to be more efficient than histograms and KDE.

Recently, more efficient tree- and grid-based approaches have been introduced. Tree-based methods (Li, Yang, and Wong 2016; Meyer 2018) such as Density Estimation Trees (DETs) (Ram and Gray 2011) achieve considerable scalability by recursively partitioning the space based on axis-aligned splits. The splits are obtained in an adaptive manner by greedily optimizing the integrated square error. Notably, DETs support hybrid domains. Grid-based methods (Peherstorfer, Pflüge, and Bungartz 2014) recursively partition the space too, but they place the splits at fixed, progressively finer-grained locations.

In the discrete case, learning of structured distributions has been explored for probabilistic sentential decision diagrams (Kisa et al. 2014) from both examples and prespecified constraints (Liang, Bekker, and Van den Broeck 2017). These techniques, however, have not been extended to hybrid domains. On the other hand, MSPNs (Molina et al. 2018; Vergari et al. 2019) are state-of-the-art density estimators that extend Sum-Product Networks (Poon and Domingos 2011; Darwiche 2009; Vergari, Di Mauro, and Esposito 2019) by introducing continuous variables and polynomial densities at the leaves. Like DETs, MSPNs allow efficient inference so long as the query formula is axis-aligned, and are learned using a greedy scheme. DETs and MSPNs will be discussed in detail later on. Hybrid SPNs (Bueff, Speichert, and Belle 2018) is another very recent alternative and could be used in place of MPSNs. However, since an implementation is not yet available, we postpone an evaluation to future work.

None of the above models can learn or represent structured, oblique supports, which are crucial in many applications, including safety-critical ones. As our experimental evaluation highlights, taking the support into account can lead to dramatically improved performance in some applications.

3 Background

Satisfiability modulo theories WMI builds on satisfiability modulo linear real arithmetic (SMT($\mathcal{LRA}$)) (Barrett et al. 2009), which we briefly overview here. SMT($\mathcal{LRA}$) extends propositional logic to a combination of Boolean and continuous variables, indicated as $\mathbf{A} = \{A_1, \ldots, A_M\}$ and $\mathbf{X} = \{X_1, \ldots, X_N\}$, respectively. An $\mathcal{LRA}$ *atom* is a formula $\sum_i c_i X_i \bowtie c$ such that $X_i \in \mathbf{X}$, $c_i, c \in \mathbb{R}$ are constants and $\bowtie \in \{=, >, <, \neq, \leq, \geq\}$. We call an $\mathcal{LRA}$ atom *axis-aligned* if at most one coefficient c_i is non-zero, and *oblique* otherwise. An $\mathcal{LRA}$ *formula* φ combines Boolean variables (aka Boolean atoms) and $\mathcal{LRA}$ atoms by means of standard logical connectives, e.g., $\varphi = (A_1 \wedge (X_1 + X_2 \leq -1)) \vee (\neg A_1 \wedge (X_1 + X_2 \geq 1))$. The if-then-else expression $\mathsf{ite}(\varphi, \varphi', \varphi'')$ evaluates to φ' if φ holds and to φ'' otherwise. The above formula can be written $\mathsf{ite}(A_1, X_1 + X_2 \leq -1, X_1 + X_2 \geq 1)$. Given a set of $\mathcal{LRA}$ formulas $\mathbf{\Phi} = \{\varphi_1, \ldots, \varphi_K\}$, a *total truth assignment* $\mu^{\mathbf{\Phi}}$ is an assignment of truth values to all the formulas, e.g., $\mu^{\mathbf{\Phi}} = \{\varphi_1 \mapsto \bot, \ldots, \varphi_K \mapsto \top\}$. Given a formula φ and a total truth assignment μ to all its atoms, we denote by $\mu^{\mathbf{A}}$ the assignment to the Boolean atoms of φ. Later on, we will often call SMT($\mathcal{LRA}$) formulas simply "SMT formulas" and, according to the context, we denote with "μ" either a set of truth assignments or their conjunction.

Weighted model integration Probabilistic inference over discrete variables is often solved with weighted model counting (WMC), i.e., by computing the weighted sum of all satisfying assignments of a propositional formula (Chavira and Darwiche 2008). Weighted model integration (WMI) extends WMC to mixed discrete-continuous domains. Here we follow (Belle, Passerini, and Van den Broeck 2015; Morettin, Passerini, and Sebastiani 2017).

A *WMI distribution* $\langle \chi, w \rangle$ over $\mathbf{X} \times \mathbf{A}$ includes:

- a (not necessarily minimal) *support* $\chi : \mathbf{X} \times \mathbf{A} \to \mathbb{B}$, which encodes the area outside of which w is zero.
- a *conditional weight function* $w : \mathbf{X} \times \mathbf{A} \to \mathbb{R}_{\geq 0}$, which associates a weight to interpretations. Intuitively, w plays the role of a density function. We assume (w.l.o.g.) that w integrates to 1.

Semantically, w is a piecewise-polynomial function over $\mathbf{X} \times \mathbf{A}$. More formally, w is characterized by a set of $\mathcal{LRA}$ formulas $\mathbf{\Psi} = \{\psi_1, \ldots, \psi_K\}$, such that every assignment $\mu^{\mathbf{\Psi}}$ identifies a convex region $\mathbf{X} \times \mathbf{A}$ (namely $\mu^{\mathbf{\Psi}} \wedge \chi$) and defines a *potential function* $w_{[\mu^{\mathbf{\Psi}}]} : \mathbf{X} \times \mathbf{A} \to \mathbb{R}_{\geq 0}$ on it. The potential functions are usually taken to be (multivariate) polynomials[1]. Encodings for w include directed acyclic graphs (Morettin, Passerini, and Sebastiani 2017) (with sum, product, and branching nodes), MSPNs (Molina et al. 2018), and extended algebraic decision diagrams (XADDs) (Kolb et al. 2018a). In XADDs the internal nodes encode the $\mathcal{LRA}$ conditions $\mathbf{\Psi}$ and the leaves are labelled with the polynomials $w_{[\mu^{\mathbf{\Psi}}]}$. From this perspective, an assignment $\mu^{\mathbf{\Psi}}$ merely specifies a path from the root to a leaf, as well as the region of $\mathbf{X} \times \mathbf{A}$ where $w_{[\mu^{\mathbf{\Psi}}]}$ applies.

The support χ is a standard SMT($\mathcal{LRA}$) formula. It is used by SMT-based solvers for pruning zero-density areas from the computations using logical reasoning only.

Example. *Figure 1 illustrates the weight function:*

$$w = \mathsf{ite}(\psi_1, X_1 + X_2, \mathsf{ite}(\psi_2, X_1^2, 0))$$

where $\psi_1 = (X_1 + X_2 \leq 1) \wedge (X_1 \leq 0.5)$ *and* $\psi_2 = \neg A_1 \wedge (X_2 \leq 0.2)$. *Note that* $w_{[\{\psi_1 \mapsto \top, \psi_2 \mapsto \bot\}]} = X_1 + X_2$, *and* $w_{[\{\psi_1 \mapsto \bot, \psi_2 \mapsto \bot\}]} = 0$. *The polynomials at the leaves are unnormalized, for simplicity.*

[1]The potential functions $w_{[\mu^{\mathbf{\Psi}}]}$ must be *feasibly integrable*; polynomials satisfy this condition, since they can be integrated over convex polytopes using numerical (Morettin, Passerini, and Sebastiani 2017) or symbolic (Kolb et al. 2018a) procedures.

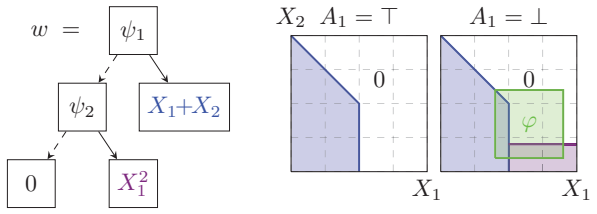

Figure 1: Left: XADD of an example weight function w. Solid (dashed) arrows point to the true (false) child. Right: partition induced by w in $(A_1, X_1, X_2) \in \{\top, \bot\} \times [0, 1] \times [0, 1]$. The green box is an example query $\varphi = \neg A_1 \wedge (0.4 \leq X_1 \leq 0.9) \wedge (0.1 \leq X_2 \leq 0.6)$. (Best viewed in color.)

WMI can answer marginal and conditional queries in hybrid domains. Given a WMI distribution $\langle \chi, w \rangle$, computing the probability of an $\mathcal{LRA}$ formula φ involves integrating the weight of its (potentially uncountably many) models:

$$P(\varphi) = \mathsf{WMI}(\varphi, w \,|\, \mathbf{X}, \mathbf{A}) \equiv \sum_{\mu^{\mathbf{A}} \in \mathbb{B}^M} \int_{\varphi_{[\mu^{\mathbf{A}}]}} w_{[\mu^{\mathbf{A}}]}(\mathbf{X}) d\mathbf{X}$$

Here the sum ranges over all total truth assignments on $\mathbf{A}$ and the integral is over all real variables $\mathbf{X}$. Computing the conditional probability of a query φ given evidence φ_e also reduces to weighted model integration: $P(\varphi \,|\, \varphi_e) = P(\varphi \wedge \varphi_e)/P(\varphi_e)$. Notice that WMI also subsumes volume computation, since $\mathsf{vol}(\varphi) = \mathsf{WMI}(\varphi, 1 \,|\, \mathbf{X}, \mathbf{A})$.

Although WMI is $\#P$-hard in general, several solvers have been proposed, including exact approaches based on SMT solvers (Belle, Passerini, and Van den Broeck 2015; Morettin, Passerini, and Sebastiani 2017; Merrell, Albarghouthi, and D'Antoni 2017), knowledge compilation (Kolb et al. 2018a), and approximate approaches based on hashing (Belle, Van den Broeck, and Passerini 2016) or Monte Carlo sampling (Zuidberg Dos Martires, Dries, and De Raedt 2019).

4 Learning WMI distributions

We are concerned with learning WMI distributions, that is, hybrid distributions with a structured support. Formally, our learning problem can be stated as follows:

Definition (WMI learning)**.** *Given a dataset $\mathcal{D}$ of feasible samples drawn i.i.d. from an unknown hybrid structured distribution P^* with true support χ^*, find a WMI distribution $\langle \hat{w}, \hat{\chi} \rangle$ that well approximates P^*.*

Notice that P^* is not required to be a WMI distribution, and that χ^* is not required to be an SMT formula. In principle this is not an issue, because (complex enough) WMI distributions can approximate any hybrid structured distribution.

We are now ready to introduce our approach, LARIAT (for LeARning to IntegrATe). Let $\mathcal{D} = \{(\mathbf{x}^s, \mathbf{a}^s)\}_{s=1}^S$ be the samples at our disposal, where $\mathbf{x}^s$ and $\mathbf{a}^s$ are the values assigned to $\mathbf{X}$ and $\mathbf{A}$, respectively. LARIAT estimates a WMI distribution from $\mathcal{D}$ by breaking the learning problem into two tasks: 1) learning a support $\hat{\chi}$; and 2) learning a weight function $\hat{w}$ compatible with $\hat{\chi}$. The first step is handled by a novel generalization of INCAL (Kolb et al. 2018b), a state-of-the-art method for learning SMT formulas. The second step uses a hybrid density estimator to learn a weight function. Finally, LARIAT normalizes the latter to trim away the unfeasible areas. We discuss these steps in detail.

Learning the support

Given examples of positive (feasible) and negative (infeasible) variable assignments, INCAL learns an SMT formula φ that covers all positive and no negative examples. Support learning can be reduced to a similar problem, namely finding a formula $\hat{\chi}$ that generalizes from the feasible samples $\mathcal{D}$ to their underlying true support χ^*. Here we briefly overview INCAL, and then introduce INCAL+, our generalization of the former to support learning.

Learning SMT formulas with INCAL Given a set $\mathcal{D}$ of positive and negative examples and a maximum number of clauses k and unique linear inequalities h, INCAL finds a CNF (or DNF) formula φ of the given complexity that correctly classifies all examples. The search is encoded as an SMT($\mathcal{LRA}$) satisfaction problem and solved with an SMT solver, e.g., MathSAT (Cimatti et al. 2013) or Z3 (De Moura and Bjørner 2008); the full SMT encoding can be found in (Kolb et al. 2018b). This non-greedy learning strategy can acquire SMT formulas with non-convex feasible sets and oblique $\mathcal{LRA}$ atoms, which lie at the core of WMI distributions and are beyond the reach of greedier approaches (see (Kolb et al. 2018b) for a discussion).

INCAL employs an incremental scheme whereby a candidate formula is gradually adapted to correctly classify progressively larger subsets of the data. At iteration i, a formula φ_i that correctly classifies a subset $\mathcal{D}_i$ of the data is computed. Next, some of the examples $\mathcal{V}_i \subseteq \mathcal{D} \setminus \mathcal{D}_i$ inconsistent with φ_i are added to $\mathcal{D}_i$ to obtain $\mathcal{D}_{i+1}$ and the process repeats. Empirically, this allows learning a formula by encoding a small fraction of the dataset, with noticeable runtime benefits.

If the target formula complexity (k, h) is not given, INCAL automatically searches for a formula of minimal complexity that covers all positive and no negative examples. This is achieved by gradually increasing k and h (initially set to 1 and 0, respectively) until an appropriate formula is found.

Learning supports with INCAL+ We cast support learning as the problem of finding an SMT formula $\hat{\chi}$ that covers all of the positive examples and does not cover regions too "far away" from them, as determined by some distance measure d over $\mathbf{X} \times \mathbf{A}$ and a user-provided threshold $\theta > 0$ over it.

More specifically, let $\mathcal{B}_\theta$ be the union of S bounding boxes, each of size θ, centered around the samples in $\mathcal{D}$:

$$\mathcal{B}_\theta := \bigcup_{s=1}^{S} \{(\mathbf{x}, \mathbf{a}) \,|\, d((\mathbf{x}^s, \mathbf{a}^s), (\mathbf{x}, \mathbf{a})) \leq \theta\}$$

Also, let $\bar{\mathcal{B}}_\theta$ be its complement. Our assumption is that θ can be chosen so that $\mathcal{B}_\theta$ is a reasonably tight overapproximation of the true support χ^*. In this case, if $\hat{\chi}$ approximates

Algorithm 1 The inner loop of the INCAL+ algorithm. FINDFORMULA uses the INCAL encoding to look for an SMT formula of complexity at most (k, h) that correctly classifies $\mathcal{D}_i$.

```
1: procedure LEARN(D: samples, (k, h): complexity)
2:     V_0 ← D_1 ← sample from D,  i ← 1
3:     while |V_{i-1}| > 0 do
4:         χ̂_i ← FINDFORMULA(D_i, k, h)
5:         if the solver returns unsat then
6:             return no support
7:         V_i ← all misclassified samples in D \ D_i
8:         if V_i ≠ ∅ then
9:             V_i ← sample from V_i
10:        else
11:            V_i ← a wrongly covered sample in B̄_θ
12:        D_{i+1} ← D_i ∪ V_i,  i ← i + 1
13:    return χ̂_i
```

$\mathcal{B}_\theta$ and does not cover any of $\bar{\mathcal{B}}_\theta$, then it will also be a good approximation of the true support. (The algorithm can be trivially adapted to allow for $\mathcal{B}_\theta$ to be an underapproximation instead.) Since too complex supports (in terms of k and h) can overfit, we wish to minimize complexity.

This leads to the INCAL+ support learning problem:

$$\min_{\hat{\chi}} \; k + h$$
$$\text{s.t. } (\mathbf{x}^s, \mathbf{a}^s) \models \hat{\chi} \qquad \forall\, (\mathbf{x}^s, \mathbf{a}^s) \in \mathcal{D} \qquad (1)$$
$$(\mathbf{x}^-, \mathbf{a}^-) \not\models \hat{\chi} \qquad \forall\, (\mathbf{x}^-, \mathbf{a}^-) \in \bar{\mathcal{B}}_\theta \qquad (2)$$

where $s = 1, \ldots, S$ ranges over all samples in the dataset. As underlying distance we choose $d((\mathbf{x}, \mathbf{a}), (\mathbf{x}', \mathbf{a}')) = \text{ite}(\mathbf{a} = \mathbf{a}', \max_i(x_i - x'_i), \infty)$, which admits an SMT($\mathcal{LRA}$) representation.

This formulation can not be solved directly, because of the universal quantifier in Eq. 2. For given (k, h), we solve it with INCAL, by exploiting its incremental learning scheme (see Algorithm 1). First, an initial subset $\mathcal{D}_1$ is obtained (at line 2) by sampling examples from $\mathcal{D}$. Then, at each iteration i, INCAL+ computes a formula φ_i that correctly classifies $\mathcal{D}_i \subseteq \mathcal{D}$ (line 4), and a set misclassified variable assignments $\mathcal{V}_i$ is obtained from $\mathcal{D}$; if none are found, a new, misclassified negative is taken from $\bar{\mathcal{B}}_\theta \wedge \hat{\chi}_i$ (lines 7–11). Finding a negative example $(\mathbf{x}^-, \mathbf{a}^-) \in \bar{\mathcal{B}}_\theta$ that is wrongly covered by $\hat{\chi}_i$ (line 11) is done via SMT by solving:

$$\hat{\chi}_i \quad \wedge \quad \bigwedge_s (d((\mathbf{x}^s, \mathbf{a}^s), (\mathbf{x}^-, \mathbf{a}^-)) > \theta) \quad \wedge \quad \text{BK}$$

where s iterates over the examples in $\mathcal{D}$ and BK is background knowledge that should not be relearned, such as mutual exclusivity of Boolean variables used as a one-hot encoding of discrete multi-valued attributes. The violating examples are then added to $\mathcal{D}_i$ to obtain $\mathcal{D}_{i+1}$ and the loop repeats. INCAL+ stops when no violating examples can be found. To improve the runtime of the algorithm, initial negative examples sampled from $\bar{\mathcal{B}}_\theta$ (e.g., through rejection sampling) can be added to the initial dataset.

Like the original algorithm, INCAL+ allows to search for values of (k, h) by wrapping the above procedure into a loop that gradually increases k and h whenever an appropriate formula can not be found (line 6). This converges to a consistent formula if one exists (Kolb et al. 2018b).

To automatically select a viable threshold θ, we dynamically explore various thresholds $\theta_i = m_i \cdot \theta_{nn}$, where θ_{nn} is the average closest distance between neighbors in $\mathcal{D}$. Since smaller thresholds provide more detailed supports but increase the runtime, we use an exponential-search based procedure to quickly find the smallest multiple m_i for which INCAL+ can learn a support within a given time budget t. Starting with an initial multiplier m_1 and given a maximal number of steps N, at each step we run INCAL+ with threshold θ_i for at most t seconds, increasing the multiplier if a support was found, and decreasing it when INCAL+ timed out. The latter controls the trade-off between computational resources and complexity of the learned support. Notice however that a tighter support is not necessarily better, and that the support to be used in LARIAT should selected e.g. on a validation set.

Learning the weight function

Since the weight function w behaves like a structured density function, it can in principle be estimated by any hybrid density estimation technique. We focus on two state-of-the-art models, Density Estimation Trees (DETs) and Mixed Sum-Product Networks (MSPNs), which we introduce next. A major limitation is that these estimators can not learn nor model oblique structured supports. Below we show how LARIAT lifts this limitation by *normalizing* the estimated densities with respect to a learned SMT($\mathcal{LRA}$) support.

Density Estimation Trees DETs are a density estimation analogue of standard decision trees. The internal nodes recursively partition the space with univariate, axis-aligned $\mathcal{LRA}$ conditions (aka *splits*), while the leaves define uniform density functions. DETs are learned from data using an iterative procedure, whereby splits are introduced so as to greedily optimize (a surrogate of) the integrated square error (Ram and Gray 2011). The procedure terminates when a pre-specified minimal number of instances is covered by each leaf. In order to control overfitting, the resulting tree is then pruned. The whole learning procedure allows piecewise constant densities to be efficiently learned from data.

DETs have a number of useful features. First, under suitable conditions, DETs are provably consistent (Ram and Gray 2011). Just like decision trees can approximate any Boolean function, deep enough DETs can approximate any weight function to arbitrary precision. Second, given a DET, the weighted model integral of any axis-aligned query φ can be easily computed by partitioning φ according to the leaves and summing the resulting integrals.

DETs also have limitations. The shallow DETs that are used in practice can not approximate oblique $\mathcal{LRA}$ splits. Moreover, DET leaves have constant density and may fail to accurately represent non-constant densities. Finally, since leaves always contain at least one example, DETs can only model trivial supports (Ram and Gray 2011).

Mixed Sum-Product Networks Mixed Sum-Product Networks (MSPNs) (Molina et al. 2018) are state-of-the-art models for hybrid domains. MSPNs extend Sum-Product Networks (Poon and Domingos 2011; Darwiche 2009) by introducing continuous variables and densities. They encode a circuit where the leaves define piecewise polynomial distributions over the input variables, and the internal nodes are sums (i.e., mixtures) or products of their child nodes.

Like DETs, MSPNs are learned greedily from data. The learning algorithm recursively splits the dataset by either: i) partitioning the input variables into independent sets, which amounts to introducing a product node, or ii) clustering similar examples together, thus introducing a sum node. If no split can be found (e.g., if there's only one example/variable in the current dataset), a polynomial leaf node is fit on the data. In practice, MSPNs implementations support univariate constant or linear leaves only.

By exploiting context-specific independencies, MSPNs can be much more compact than DETs, while still allowing for tractable inference of marginal and conditional queries. Indeed, if the queried quantity decomposes over the circuit, then inference amounts to a couple of bottom-up evaluations.

Normalization

The third and final step of LARIAT is *normalization*. Given a support $\hat{\chi}$ and a learned piecewise-polynomial weight function $\tilde{w}$, normalization aims at redistributing the density of $\tilde{w}$ away from the unfeasible region outside of $\hat{\chi}$ and inside the feasible one. After normalization, the resulting weight function $\hat{w}$ has to satisfy the properties: p_1) $\mathsf{WMI}(\neg\hat{\chi}, \hat{w}) = 0$; and p_2) $\mathsf{WMI}(\hat{\chi}, \hat{w}) = 1$. For general densities normalization is computationally hard, because computing the mass of infeasible regions is #P-hard. However, it must only be performed once and the guarantees it offers e.g. in safety-critical applications justify the additional processing cost.

Global normalization These two properties give rise to a simple, yet effective normalization scheme which defines:

$$\hat{w} = \mathsf{ite}(\hat{\chi}, \frac{\tilde{w}}{\mathsf{WMI}(\hat{\chi}, \tilde{w})}, 0)$$

It is trivial to see that properties p_1 and p_2 are satisfied, however, to improve the likelihood of the resulting density, we can resort to a more elaborate normalization scheme.

Local Normalization Local density estimators, like DETs, partition the space into sub-regions defined by mutually exclusive SMT formulas χ_i and fit a local density $\tilde{w}_i$ for every sub-region i. The idea behind local normalization is to retain the space partitioning induced by the learner but to (locally) normalize the densities within every subregion such that: $\mathsf{WMI}(\chi_i \wedge \neg\hat{\chi}, \hat{w}_i) = 0$ and $\mathsf{WMI}(\chi_i \wedge \hat{\chi}, \hat{w}_i) = \mathsf{WMI}(\chi_i, \tilde{w}_i)$, where $\hat{w}_i$ is the normalized local density. It follows that, since the χ_i exhaustively partition the entire space and $\mathsf{WMI}(\top, \tilde{w}) = 1$, the properties p_1 and p_2 are both satisfied. The local normalized density $\hat{w}_i$ can be computed as:

$$\mathrm{norm}^* : \hat{w}_i = \tilde{w}_i \cdot \mathsf{WMI}(\chi_i, \tilde{w}_i) \,/\, \mathsf{WMI}(\chi_i \wedge \hat{\chi}, \tilde{w}_i)$$
$$\mathrm{norm}^+ : \hat{w}_i = \tilde{w}_i + \mathsf{WMI}(\chi_i \wedge \neg\hat{\chi}, \tilde{w}_i) \,/\, \mathsf{vol}(\chi_i \wedge \hat{\chi})$$

Local normalization for DETs In the case of DETs, local densities $\tilde{w}_i$ are computed as $\tilde{w}_i = \frac{1}{S}\frac{S_i}{\mathsf{vol}(\chi_i)}$, and we obtain, using $S = |\mathcal{D}|$ for the total number of samples and $S_i = |\mathcal{D} \models \chi_i|$ for the number of samples that satisfy χ_i:

$$\hat{w}_i = \frac{1}{S}\frac{S_i}{\mathsf{vol}(\chi_i)} \cdot \frac{\mathsf{WMI}(\chi_i, \tilde{w}_i)}{\mathsf{WMI}(\chi_i \wedge \hat{\chi}, \tilde{w}_i)} = \frac{1}{S}\frac{S_i}{\mathsf{vol}(\chi_i \wedge \hat{\chi})}.$$

There are three properties of DETs which makes local normalization especially attractive: 1) DETs explicitly describe the sub-regions; 2) norm^* and norm^+ yield the same result since $\tilde{w}_i$ is constant; and 3) since every DET subregion contains at least one sample, no subregion can become entirely infeasible. Local normalization is computationally more expensive than global normalization, however, for locally fit models (like DETs) it obtains more accurate models.

Example. *Consider a learned weight function (DET) consisting of two non-zero subregions whose local densities have been fitted based on the number of samples in those sub-regions ($\frac{1}{4}$th of the samples in the first, and $\frac{3}{4}$th of the samples in the second sub-region):*

$$\tilde{w} = \begin{cases} 0.25 & \textbf{if } (0 \leq x \leq 1) \wedge (0 \leq y \leq 1) \\ 0.75 & \textbf{if } (0 \leq x \leq 1) \wedge (1 < y \leq 2) \end{cases}$$

and a support $\hat{\chi} = (y \geq x)$. The support only cuts away feasible volume from the first sub-region, meaning that the samples falling into the first sub-region are now distributed over a smaller volume. Therefore, the local normalization procedure increases the density over the remaining feasible volume in the first sub-region, while leaving the second sub-region untouched. We obtain the normalized weight function:

$$\hat{w} = \begin{cases} 0.5 & \textbf{if } (0 \leq x \leq 1) \wedge (0 \leq y \leq 1) \wedge (y \geq x) \\ 0.75 & \textbf{if } (0 \leq x \leq 1) \wedge (1 < y \leq 2) \end{cases}$$

Local normalization for generic densities First, to normalize generic weight functions encoded as ASTs (e.g., MSPNs) which only indirectly describe the sub-regions and their densities, we can compile the ASTs into equivalent XADDs (Kolb et al. 2018a) that support the enumeration of sub-regions and integration. Second, the choice between norm^* and norm^+ is left up to the user. Third, the density of completely infeasible sub-regions is redistributed over the entire feasible space akin to global normalization. It is important to note that normalization of models like MSPNs that support tractable inference generally may render inference intractable.

Both normalization schemes are amenable to any support, be it learned with INCAL+ or provided by a domain expert (as can be the case in security applications). Since they can can deal with arbitrary WMI distributions, other density estimators such as MARS (see Related Work), can also be readily converted to WMI and normalized.

5 Empirical Analysis

In this section, we explore the following research questions[2]: **Q1**) Does INCAL+ learn reasonable supports? **Q2**) Does LARIAT improve over state-of-the-art density estimators when the true support is provided? **Q3**) Does LARIAT improve over state-of-the-art density estimators when the support is estimated by INCAL+? We address the first two questions on synthetic datasets of increasing complexity, and the last one on both synthetic and real-world datasets.

Each synthetic dataset was obtained by first generating a random WMI distribution $\langle \chi^*, w^* \rangle$ with a given number of Boolean b and continuous variables r, and then sampling examples (i.e. feasible configurations) from it. More specifically: i) w^* was a random XADD sampled by recursively adding internal and leaf nodes up to a fixed depth $d = 2$, where the internal nodes partition the space with a random SMT($\mathcal{LRA}$) formula, while the leaves host a randomly generated non-negative polynomial of maximum degree $2r$. ii) χ^* was obtained by sampling a CNF formula with h hyperplanes and l literals using the procedure of (Kolb et al. 2018b). Without loss of generality, we restricted the range of the continuous variables to $[0, 1]$.

We evaluated how the complexity of the true distribution impacts the performance of INCAL+ and LARIAT by generating increasingly complex supports, either by fixing $b = 3$, $l = 3$ and varying h and r, or by fixing $r = 3$, $h = 5$ and increasing l and b, for a total of 30 configurations. For each configuration, we generated 20 different ground-truth models and relative dataset, each consisting of 500 training and 50 validation examples.

Learning the support In order to answer **Q1**, we implemented INCAL+ using the MathSAT solver, applied it to the synthetic datasets with a timeout of 300 seconds for each call, and measured the misclassification error between the true and the learned support. This amounts to the volume of the symmetric difference of the two supports, that is, $\mathsf{vol}((\hat{\chi} \wedge \neg\chi^*) \vee (\neg\hat{\chi} \wedge \chi^*))$.

In these experiments we ran INCAL+ on training data $\mathcal{D}_\mathcal{T}$ and used the validation data $\mathcal{D}_\mathcal{V}$ to select the learned support that minimizes:

$$\frac{\mathsf{vol}(\chi \wedge \neg\mathcal{B}'_\theta)}{\mathsf{vol}(\neg\mathcal{B}'_\theta)} + \frac{\sum_{(\mathbf{x}^s, \mathbf{a}^s) \in \mathcal{D}_\mathcal{V}} \mathbb{1}((\mathbf{x}^s, \mathbf{a}^s) \models \chi)}{|\mathcal{D}_\mathcal{V}|}$$

where $\mathbb{1}(\ldots)$ is the indicator function and $\mathcal{B}'_\theta$ is the formula that encodes the union of bounding boxes of size θ around the points in $\mathcal{D}_\mathcal{V}$, being θ the threshold used to learn χ. Figure 2 (left) shows the average and standard deviation misclassification error (normalized in $[0, 1]$) for each set of parameters. INCAL+ is compared to the trivial support, given by the range of each single variable in the dataset. Results show that INCAL+ learns a support that substantially improves on the trivial one in all settings, apart for a few instances for $l = 4$ where it timed out. This allows us to answer question **Q1** affirmatively.

[2]The code is available at: URL ANONYMIZED

Dataset	DET	+LARIAT	MSPN	+LARIAT
anneal-U	-61.2	$\mathbf{-38.5}$	-41.9	$\mathbf{6.0}$
australian	-44.1	$\mathbf{-30.1}$	-32.8	$\mathbf{-27.8}$
auto	-80.8	$\mathbf{-63.5}$	-67.2	$\mathbf{-58.5}$
balance-scale	-7.2	$\mathbf{-6.4}$	-7.5	$\mathbf{-6.4}$
breast	-29.6	$\mathbf{-29.2}$	-26.1	$\mathbf{-25.2}$
breast-cancer	-11.8	$\mathbf{-9.1}$	-11.5	$\mathbf{-8.6}$
cars	-40.1	$\mathbf{-29.5}$	-29.2	$\mathbf{-26.8}$
cleve	-31.2	$\mathbf{-26.9}$	-28.0	$\mathbf{-25.8}$
crx	-48.2	$\mathbf{-32.7}$	-32.7	$\mathbf{-28.8}$
diabetes	-28.8	$\mathbf{-27.9}$	-30.3	$\mathbf{-29.4}$
german	-46.1	$\mathbf{-36.4}$	-39.4	$\mathbf{-33.3}$
german-org	-28.8	$\mathbf{-28.6}$	-27.5	$\mathbf{-25.2}$
glass	-2.1	$\mathbf{1.6}$	0.7	$\mathbf{3.7}$
glass2	4.2	$\mathbf{4.8}$	4.7	$\mathbf{5.2}$
heart	-24.1	$\mathbf{-23.8}$	-24.7	$\mathbf{-22.7}$
hepatitis	-29.3	$\mathbf{-24.0}$	-27.1	$\mathbf{-23.7}$
iris	-4.3	$\mathbf{-3.4}$	-2.9	$\mathbf{-2.7}$
solar	-15.0	$\mathbf{-2.6}$	-7.6	$\mathbf{3.3}$

Table 1: Average log-likelihood on the test set for the UCI/MLC++ experiment. Bold text highlights a statistically significant improvement (p-value < 0.0001) in the test set log-likelihood of LARIAT over the unnormalized model.

Learning synthetic distributions We addressed **Q2** and **Q3** by comparing the distribution learned by the density estimator alone (DET or MSPN) with the one learned by LARIAT using ground truth (**Q2**) and INCAL+ estimated (**Q3**) support.

Contrary to the previous support learning experiment, INCAL+ was run on $\mathcal{D}_\mathcal{T} \cup \mathcal{D}_\mathcal{V}$. Weight learning was performed on $\mathcal{D}_\mathcal{T}$ only, reserving the validation set $\mathcal{D}_\mathcal{V}$ to select the best support among those learned by INCAL+, as the one yielding the highest log-likelihood on $\mathcal{D}_\mathcal{V}$. For DETs, the validation set was also used to prune the tree.

We measured the quality of the learned distributions $\langle \hat{\chi}, \hat{w} \rangle$ by measuring the integrated absolute error (IAE) with respect to the true distribution:

$$\mathsf{IAE}(\hat{w}, w^*) = \sum_{\mathbf{A}} \int_{\hat{\chi} \vee \chi^*} |\hat{w}(\mathbf{X}, \mathbf{A}) - w^*(\mathbf{X}, \mathbf{A})| \, d\mathbf{X}$$

Since computing the exact value of IAE is computationally prohibitive, we performed a Monte Carlo approximation with 1,000,000 samples.

The average IAEs and their standard deviations are shown in Figure 2 (middle) and (right) when using DET and MSPN estimators respectively. The results affirmatively answer **Q2**, as LARIAT with ground truth support achieves substantial improvements over the density estimators alone, for both DET and MSPN. The same holds for **Q3** when using a DET estimator, even if the improvements are clearly more limited. Conversely, the supports learned appear not as effective when applied to the global normalization.

Learning real-world distributions In order to check whether the improvements shown by LARIAT in the synthetic experiments hold in real world scenarios, we tested its

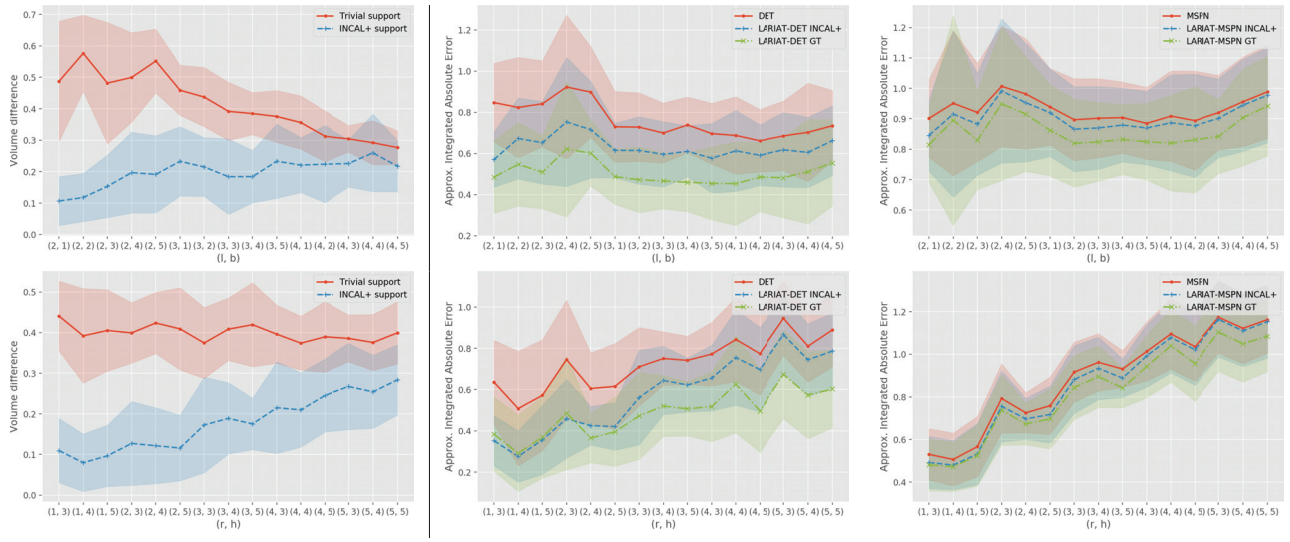

Figure 2: Results on the synthetic datasets. Left: volume of the difference of the true support and the learned (blue) or trivial (red) ones for varying $\langle l, b\rangle$ (top) and $\langle r, h\rangle$ (bottom). Middle: IAE of DETs (red) vs. LARIAT-DETs with learned (blue) and ground truth (green) support. Right: IAE of MSPNs (red) vs. LARIAT-MSPNs with learned (blue) and ground truth (green) support. (Best viewed in color.)

performance on real-world datasets (question **Q3**). The categorical features were converted to Booleans with a *one-hot-encoding*. The resulting mutual exclusivity constraints were used to initialize both INCAL+ and the sampling procedures. Support and weight learning were run using the same setting used for the synthetic distributions.

We evaluated LARIAT on the hybrid UCI benchmarks contained in the MLC++ library, which includes 18 hybrid datasets from different real-world domains. On some datasets, INCAL+ was not able to discover any non-trivial support on the full hybrid space before timing out. In those cases, the search was then performed on the numerical subspace, allowing INCAL+ to learn linear relationships among the continuous variables. Table 1 shows the average log-likelihood computed on the test set for each dataset. (Notice that the log-likelihood of test points falling outside the model support is $-\infty$. In order to apply the metric in our structured and hybrid spaces, we follow the approach of (Molina et al. 2018) and substitute each $\log(0)$ in the average with a large negative constant.)

For the most complex tasks, LARIAT-DET is sometimes unable to complete the local normalization within reasonable time. If the local normalization timeout (set to $1200s$) is reached, LARIAT's fallback strategy is to apply the faster, global normalization.

Nevertheless, in every setting LARIAT improves the performance of the underlying model. We compared the log-likelihood of test set points using a Wilcoxon test, confirming that the improvement is significant for both DETs and MSPNs (p-value < 0.0001).

The tasks that benefited the most from LARIAT are characterized by a large amount of categorical and few continuous variables. A more extensive investigation of the factors contributing to the performance gains is left for future work.

6 Conclusion and Future Work

We presented LARIAT, a novel approach for learning WMI distributions from data. LARIAT first learns a non-trivial SMT($\mathcal{LRA}$) support from examples—using an extension of a recent SMT($\mathcal{LRA}$) constraint learner to positive-only data—and then combines said support with state-of-the-art hybrid density estimators. Our experiments with DETs and MSPNs show that combining support learning and density estimation allows to more accurately recover the underlying density. Our approach is modular, and each component can be replaced with alternative solutions whenever appropriate. For instance, solutions based on search-based program synthesis (Alur et al. 2018) could be explored as a replacement for INCAL+ in learning supports. Concerning the density estimator, we showed the advantages of integrating support learning and density estimation for both DET and MSPN, but other density estimators like the recently proposed HSPN (Bueff, Speichert, and Belle 2018) could be tried as soon as implementations will be made available.

Acknowledgements

We are grateful to Antonio Vergari for helping with the experiments. This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. [694980] SYNTH: Synthesising Inductive Data Models) and from the European Union's Horizon 2020 FET Proactive project *"WeNet – The Internet of us"*, grant agreement No. 823783

References

Afshar, H. M.; Sanner, S.; and Webers, C. 2016. Closed-form gibbs sampling for graphical models with algebraic

constraints. In *AAAI*, 3287–3293.

Alur, R.; Singh, R.; Fisman, D.; and Solar-Lezama, A. 2018. Search-based program synthesis. *Commun. ACM* 61(12):84–93.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Mané, D. 2016. Concrete Problems in AI Safety. *arXiv preprint arXiv:1606.06565*.

Barrett, C.; Sebastiani, R.; Seshia, S.; and Tinelli, C. 2009. Satisfiability modulo theories. In *Handbook of Satisfiability*. IOS Press. chapter 26, 825–885.

Belle, V.; Passerini, A.; and Van den Broeck, G. 2015. Probabilistic inference in hybrid domains by weighted model integration. In *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2770–2776.

Belle, V.; Van den Broeck, G.; and Passerini, A. 2016. Hashing-Based Approximate Probabilistic Inference in Hybrid Domains: An Abridged Report. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI), Sister Conference Best Paper Track*.

Bueff, A.; Speichert, S.; and Belle, V. 2018. Tractable querying and learning in hybrid domains via sum-product networks. In *KR Workshop on Hybrid Reasoning and Learning*.

Chavira, M., and Darwiche, A. 2008. On probabilistic inference by weighted model counting. *Artificial Intelligence* 172(6-7):772–799.

Cimatti, A.; Griggio, A.; Schaafsma, B. J.; and Sebastiani, R. 2013. The MathSAT5 SMT Solver. In *TACAS*, volume 7795, 93–107. Springer.

Darwiche, A. 2009. *Modeling and reasoning with Bayesian networks*. Cambridge university press.

De Moura, L., and Bjørner, N. 2008. Z3: An efficient SMT solver. *Tools and Algorithms for the Construction and Analysis of Systems* 337–340.

Fix, E., and Hodges Jr, J. L. 1951. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, California Univ Berkeley.

Friedman, J. H. 1991. Multivariate adaptive regression splines. *The annals of statistics* 1–67.

Gray, A. G., and Moore, A. W. 2003. Nonparametric density estimation: Toward computational tractability. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, 203–211. SIAM.

Jing, J.; Koch, I.; and Naito, K. 2012. Polynomial histograms for multivariate density and mode estimation. *Scandinavian Journal of Statistics* 39(1):75–96.

Kisa, D.; Van den Broeck, G.; Choi, A.; and Darwiche, A. 2014. Probabilistic sentential decision diagrams. In *KR*.

Kolb, S.; Mladenov, M.; Sanner, S.; Belle, V.; and Kersting, K. 2018a. Efficient symbolic integration for probabilistic inference. In *IJCAI*, 5031–5037.

Kolb, S.; Teso, S.; Passerini, A.; and De Raedt, L. 2018b. Learning SMT(LRA) Constraints using SMT Solvers. In *IJCAI*, 2333–2340.

Li, D.; Yang, K.; and Wong, W. H. 2016. Density estimation via discrepancy based adaptive sequential partition. In *Advances in Neural Information Processing Systems*, 1091–1099.

Liang, Y.; Bekker, J.; and Van den Broeck, G. 2017. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*.

Merrell, D.; Albarghouthi, A.; and D'Antoni, L. 2017. Weighted model integration with orthogonal transformations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, 4610–4616. AAAI Press.

Meyer, D. W. 2018. Density estimation with distribution element trees. *Statistics and Computing* 28(3):609–632.

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 *Thirty-second AAAI conference on artificial intelligence*.

Morettin, P.; Passerini, A.; and Sebastiani, R. 2017. Efficient Weighted Model Integration via SMT-based Predicate Abstraction. *Proceedings of IJCAI'17*.

Peherstorfer, B.; Pflüge, D.; and Bungartz, H.-J. 2014. Density estimation with adaptive sparse grids for large data sets. In *Proceedings of the 2014 SIAM international conference on data mining*, 443–451. SIAM.

Poon, H., and Domingos, P. 2011. Sum-product networks: A new deep architecture. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, 689–690. IEEE.

Ram, P., and Gray, A. G. 2011. Density estimation trees. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 627–635. ACM.

Vergari, A.; Molina, A.; Peharz, R.; Ghahramani, Z.; Kersting, K.; and Valera, I. 2019. Automatic bayesian density analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 5207–5215.

Vergari, A.; Di Mauro, N.; and Esposito, F. 2019. Visualizing and understanding sum-product networks. *Machine Learning* 108(4):551–573.

Zuidberg Dos Martires, P. M.; Dries, A.; and De Raedt, L. 2019. Exact and approximate weighted model integration withprobability density functions using knowledge compilation. In *Proceedings of the 30th Conference on Artificial Intelligence*. AAAI Press.