Journal of Machine Learning Research 20 (2019) 1-47 Submitted 8/18; Revised 6/19; Published 8/19

Logical Explanations for Deep Relational Machines Using Relevance Information

Ashwin Srinivasan ashwin.srinivasan@wolfson.oxon.org
Department of Computer Sc. & Information Systems
BITS Pilani, K.K. Birla Goa Campus, Goa, India

Lovekesh Vig Lovekesh.Vig@tcs.com
TCS Research, New Delhi, India

Michael Bain m.bain@unsw.edu.au
School of Computer Science and Engineering
University of New South Wales, Sydney, Australia.

Editor: Luc De Raedt

Abstract

Our interest in this paper is in the construction of symbolic explanations for predictions made by a deep neural network. We will focus attention on deep relational machines (DRMs: a term introduced in Lodhi (2013)). A DRM is a deep network in which the input layer consists of Boolean-valued functions (features) that are defined in terms of relations provided as domain, or background, knowledge. Our DRMs differ from those in Lodhi (2013), which uses an Inductive Logic Programming (ILP) engine to first select features (we use random selections from a space of features that satisfies some approximate constraints on logical relevance and non-redundancy). But why do the DRMs predict what they do? One way of answering this was provided in recent work Ribeiro et al. (2016), by constructing readable proxies for a black-box predictor. The proxies are intended only to model the predictions of the black-box in local regions of the instance-space. But readability alone may not be enough: to be understandable, the local models must use relevant concepts in an meaningful manner. We investigate the use of a Bayes-like approach to identify logical proxies for local predictions of a DRM. As a preliminary step, we show that DRM's with our randomised propositionalization method achieve predictive performance that is comparable to the best reports in the ILP literature. Our principal results on logical explanations show: (a) Models in first-order logic can approximate the DRM's prediction closely in a small local region; and (b) Expert-provided relevance information can play the role of a prior to distinguish between logical explanations that perform equivalently on prediction alone.

Keywords: Explainable AI, Inductive Logic Programming, Deep Relational Machines

1. Introduction

In Stewart (1994) a contrast is presented between theories that predict everything, but explain nothing, and those that explain everything, but predict nothing. Both are seen as having limited value in the scientific enterprise, which requires models with both predictive and explanatory power. Michie and Johnston (1984) add a further twist to this by suggesting that the limitations of the human brain may force a "window" on the complexity of explanations that can be feasibly understood, even if they are described in some readable form like a natural or a symbolic formal language.

Complexity limits notwithstanding, it is often assumed that that predictive and explanatory assessments refer to the *same* model. But this does not have to be so: early results in the Machine Learning (ML) literature on behavioural cloning point to areas where people perform predictive tasks using sub-symbolic models, for which entirely separate symbolic explanations were constructed by machine (Michie, 1986). It is not therefore inevitable that machine learning models be either purely sub-symbolic or purely symbolic.

There are, of course, problems for which one or the other is much better suited. Prominent recent examples of successful sub-symbolic learning abound with the use of deep neural networks in automatic caption learning ("woman in a white dress standing with tennis racquet": Karpathy and Li (2015)), speech-recognition (Siri, Google Now) and machine translation (Google Translate). It is equally evident that for learning recursive theories, program synthesis, and learning with complex domain-knowledge, symbolic techniques like Inductive Logic Programming (ILP) have proved to be remarkably effective. But there are also a number of problems that could benefit from an approach that requires not one or the other, but both forms of learning (in his book "Words and Rules", Steven Pinker conjectures language learning as one such task).

Our interest here therefore is in investigating the use of separate models for prediction and explanation. Specifically, we examine the use of a neural model for prediction, and a logical model for explanation. But in order to ensure that the models are about the same kinds of things, the models are not constructed independent of each other. The neural model is constructed in terms of features in (first-order) logic identified by a simple form of propositionalization methods developed in ILP. In turn, the predictions of the neural model are used to construct logical models in terms of the features.[1] Figure 1 shows how the various pieces we have just described fit together.

Having two separate models would appear to be an extravagance that could be ill-afforded if the neural model wasn't a very good predictor, or if the logical model wasn't a very good explainer.

In this paper, we are concerned with the analysis of relational data. That is, a data instance is described not just by values of some pre-defined attributes, but also by by pre-defined relations. A good example of such a data instance is a molecule known to target a malarial parasite. Such a molecule not only has attributes like its molecular weight, solubility, hydrophobicity and so on, but also has connected ring-structures, functional groups, and the like. For predictions on data like these, we use a specific form of neural models called *deep relational machines* or DRMs. The term, originally introduced in Lodhi (2013), can be broadly understood as a class of neural models in which the inputs include Boolean features defined using known relations. For example, a Boolean feature for molecular data may be $TRUE$ if the molecule has "a 7-member ring connected to a lactone ring with a peroxide bridge" and $FALSE$ otherwise. Here definitions of relations like a 7-member ring, lactone ring, and a peroxide bridge are known. In its original formulation, a DRM relied

1. We restrict ourselves in this paper to queries of the form: "What is the class of instance x?", but the ideas extend straightforwardly to tasks other than classification.

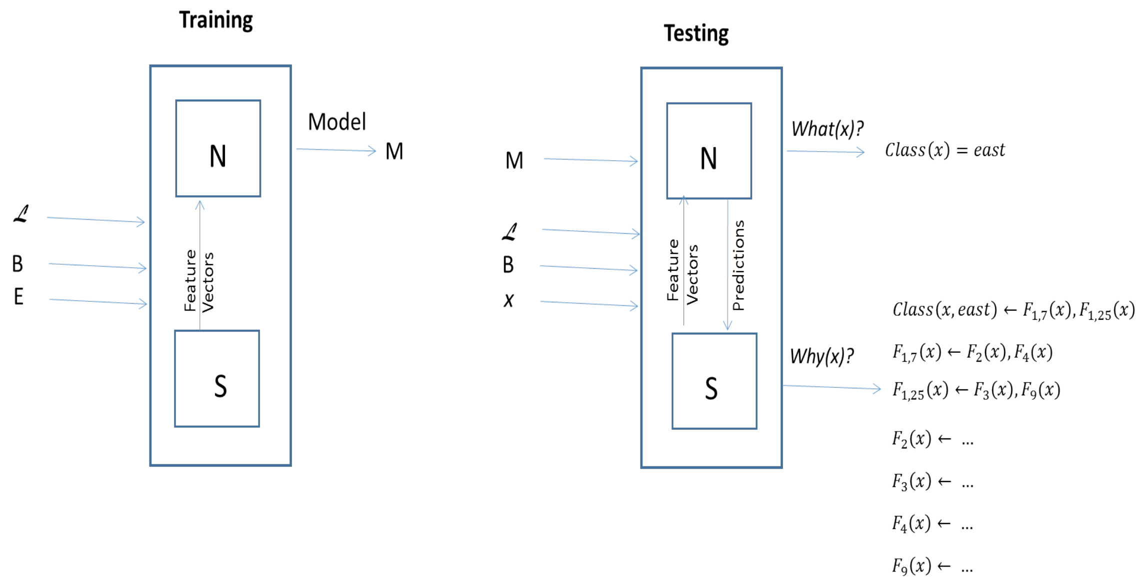


Figure 1: What this paper is about. N is a deep neural network, and S is a symbolic learner. B is domain-specific background knowledge; $\mathcal{L}$ language constraints; E a set of training examples. N is used to construct a predictive model M. Explanations of why M predicts the class of a test instance x are to be obtained as a logical model that uses features, that are pre-defined in B or "invented". Correctly, B and $\mathcal{L}$ on the right are augmentations of those on the left; and the S's on right and left are different instantiations of a first-order symbolic learner that is capable of producing definitions of features (left), or explanations (right).

on selection of Boolean features using Inductive Logic Programming (Muggleton and De Raedt, 1994). More recently, it has been shown in an extensive empirical study that DRMs with stochastic selection of relational features can achieve predictive accuracies that are often better than the state-of-the-art (Dash et al., 2018). We will therefore take this form of a DRM as the basis for prediction. Using the DRMs as predictors, our interest is in developing symbolic explanations for the predictions. Specifically:

- We present a method for extracting local symbolic explanations in terms of the input features of the DRM by employing a Bayes-like trade-off of predictive-fit and prior preference. The latter is obtained using domain-knowledge of relevance of predicates used in the explanations.

The combination of using prediction with relational features and relevance information extends the techniques proposed in Ribeiro et al. (2016) in the direction of making explanations both more expressive (since the features are definitions in first-order logic) and meaningful (by incorporating domain-specific preferences into explanations). We will demonstrate the use of combining predictive performance and relevance on real-life biochemical problems for which we have the necessary relevance information. However, the approach we propose can prove useful to any relational learning task in which there is at least a partial-ordering over domain-predicates. These applications could be as diverse as recommender systems (for example, movies directed by certain directors may be more relevant to recommendations for a particular genre), classification (for example, when paper citations are to be classified as being the same, it may be equally relevant to check similarities in title, or the list of authors, but less relevant to examine the paper's keywords), or function-prediction (for example, when predicting gene-function using relations in the Gene Ontology, predicates with evidence-codes based on experimental evidence may be more relevant than those based on computational evidence). It may not be necessary to have prior knowledge for this — recent work suggests that it may be possible, for any relational learning task, to derive such a partial-ordering automatically from data, using a probabilistic definition of the generality of domain-predicates.[2] A decision-theoretic technique for determining relevant predicates for general relational learning tasks was proposed earlier in Srinivasan (2001).

The rest of the paper is organised as follows. In Section 2 we describe the DRMs we use for prediction. Section 3 describes the notion of explanations as used in this paper. Selection amongst several possible explanations is in Section 4, which introduces the use of a relevance-based prior in Section 4.1. Section 5 presents an empirical evaluation of the predictive and explanatory models, using some benchmark datasets. Appendix C contains details of the domain-specific relevance information used in the experiments.

2. A Deep Relational Machine for Prediction

One of the most remarkable recent advances in the area of Machine Learning is a resurgence of interest in neural networks, resulting from the automated construction of "deep networks" for prediction. Simplistically, Deep Learning is using neural networks with multiple hidden layers. Mathematically speaking, it is a composition of multiple simple non-linear functions trying to learn a hierarchy of intermediate features that most effectively aid the global learning task. Learning such intermediate features with neural networks has been made possible by three separate advances: (a) mathematical techniques that allow the training of neural networks with very large numbers of hidden layers; (b) the availability of very large amounts of data that allow the estimation of parameters (weights) in such complex networks; and (c) the advanced computational capabilities offered by modern day GPUs to train deep models. Despite successes across a wide set of problems, deep learning is unlikely to be sufficient for all kinds of data analysis problems. The principal difficulties appear to lie in the data and computational requirements to train such models. This is especially the case if many hidden layers are needed to encode complex concepts (features). For many industrial processes, acquiring data can incur significant costs, and simulators can be computationally very intensive.

Some of this difficulty may be alleviated if knowledge already available in the area of interest can be taken into account. Consider, for example, a problem in the area of drug-design. Much may be known already about the target of interest: small molecules that have proved to be effective, what can and cannot be synthesized cheaply, and so on. If these concepts are relevant to constructing a model for predicting good drugs, it seems both unnecessary and inefficient to require a deep network to re-discover them (the

2. See: `https://www.doc.ic.ac.uk/∼shm/Papers/relevance18.pdf`

problem is actually worse — it may not even be possible to discover the concepts from first-principles using the data available). It is therefore of significant practical interest to explore ways in which prior domain-knowledge could be used in deep networks to reduce data and computational requirements.

Deep Relational Machines, or DRMs, proposed in Lodhi (2013), are deep neural networks with first-order Boolean functions at the input layer ("function F_1 is true if the instance x is a molecule containing a 7-membered ring connected to a lactone ring" — definitions of relations like 7-membered and lactone rings are expected to be present in the background knowledge).

We note that DRMs are one example of a much broader class of neural-symbolic models, which are surveyed in Besold et al. (2017). In ILP, pioneering work on the combination of neural-networks and symbolic features has been done by d'Avila Garcez and Zaverucha (1999) and extended in França et al. (2014, 2015). For the purposes of this paper, we will use the term "DRM" to mean a neural network with multiple hidden layers, and an input layer with values for first-order Boolean functions. We will further assume that the function definitions employ predicates that are defined as part of domain- or background-knowledge. In this sense, the neural networks in França et al. (2015) for example are DRMs, and the definitions and techniques for constructing explanations employed here apply without change to those models.

2.1. Input Features for a DRM

Ideally, we would like the inputs for a DRM to consist of all possible relational features, and let the network's training process decide on the features that are actually useful (in the usual manner: a feature that has 0 weights for all out-going edges is not useful). The difficulty with this is that the number of such features in first-order logic can be very large, often impractical to enumerate completely. In Lodhi (2013) the functions are learned using an Inductive Logic Programming (ILP) engine[3]. This follows a long line of research, sometimes called *propositionalization*, in which features constructed by ILP have been used by other learning methods like regression, decision-trees, SVMs, topic-models, and multiplicative weight-update linear threshold models, and neural-network models. In each of these, the final model is constructed in two steps: first, a set of features is selected, and then, the final model is constructed using these features, possibly in conjunction with other features already available. Usually the models show significant improvements in predictive performance when an existing feature set is enriched in this manner.

In Lodhi (2013), deep networks with ILP-features are shown to perform well, although the empirical evidence is limited. Recently, a simple approach for stochastic selection of relational features was introduced in Vig et al. (2018) in the context of evaluating embeddings. This form of randomised feature-selection has since been evaluated extensively in Dash et al. (2018) and found to give predictive performance better than the state-of-the-art for classification problems.

In this paper, we will use this form of stochastic selection of relational features as inputs for a DRM. For completeness, we clarify first what we mean by a relational instances and feature.

3. Terminology from Logic Programming and ILP is defined in Appendix A.

Definition 1 Relational Examples for Classification. *The set of examples provided can be defined as a binary relation Class which is a subset of the Cartesian product $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ denotes the set of relational instances (for simplicity, and without loss of generality, we will take each instance to be a ground first-order term), and $\mathcal{Y}$ denotes the finite set of class labels. Any single element of the set of relational examples will be denoted by $Class(a, c)$ where $a \in \mathcal{X}$ and $c \in \mathcal{Y}$.*

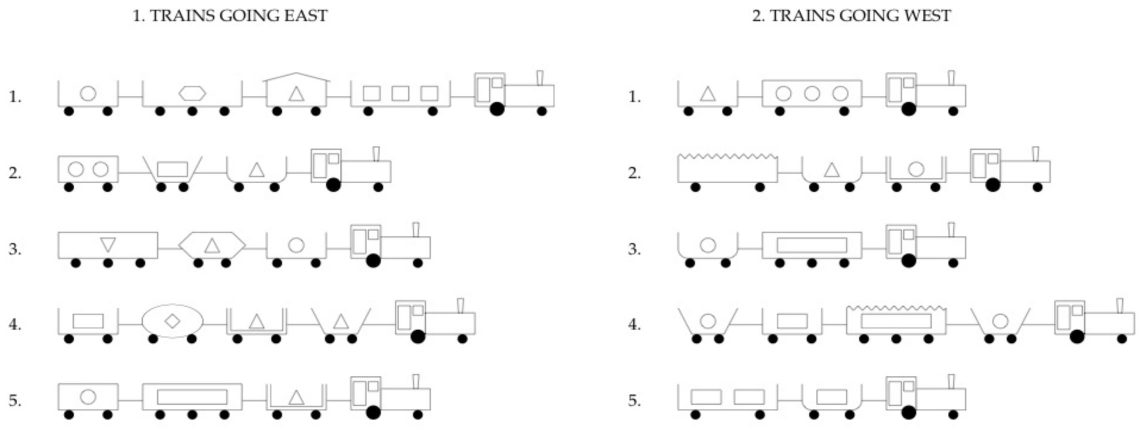
Definition 2 Relational Features. *A relational feature is a unary function $F_i : \mathcal{X} \mapsto \{TRUE, FALSE\}$ defined in terms of a conjunction of predicates $Cp_i(x)$. Each predicate in $Cp_i(x)$ is defined as part of domain- or background-knowledge B. That is, $F_i(x) \mapsto TRUE$ iff $Cp_i(x) \mapsto TRUE$ and $FALSE$ otherwise. We will represent each relational feature as a single definite clause $\forall x\ (F_i(x) \leftarrow Cp_i(x))$ in a logic program, relying on the closed-world assumption for the complete definition of F_i. We will sometimes call the relational feature F_i simply the* feature *F_i, and the definite-clause definition for F_i the* feature-definition *for F_i. If the feature-definition of F_i is in B, we will sometimes say "feature F_i is in B." We will usually denote the set of features in B as $\mathcal{F}_B$ or simply $\mathcal{F}$.*

In terms of the classes of features identified in Saha et al. (2012), our definition corresponds to the least constrained class of features (F_d) in that paper. The relational features selected by an ILP engine in Lodhi (2013) are from this class, and those in França et al. (2014, 2015) are from the class of *independent* features (F_i) in Saha et al. (2012).

Definition 3 Classification clause. *A clause for classifying a relational example $Class(a, c)$ is a clause $\forall x(Class(x, c) \leftarrow Cp(x))$, where $Cp(x)$ is a conjunction of predicates. Each predicate in $Cp(x)$ is defined as part of domain- or background-knowledge B.*

It is evident from the definitions that some or all of a classification clause can be converted into relational features.

Example 1 The trains problem. *The figure below shows a well-known synthetic problem in the ILP literature, Michalski's "Trains" problem, originally posed in Michalski (1983):*



There are two sets of trains: Eastbound and Westbound. Descriptors for the trains include: number, shape and lengths of the car, shape of load, and so on. The task is to determine classification rules to distinguish Eastbound trains from Westbound ones.

For this problem each relational example is a pair, consisting of a ground first-order term and a class label. The ground term represents the train (for example $Train(Car(Long, Open, Rect, 3), Car(Short, Closed, Triangle, 1), \ldots))$ *and the label denotes whether or not it is Eastbound.*

We assume also that we have access to domain (or background) knowledge B *that allows us to access aspects of the relational instance such as* $Has\_Car(Train(\ldots), Car(\ldots))$, $Closed(Car(\ldots))$ *and so on. Then a clause for classifying Eastbound trains is (we leave out the quantifiers for simplicity):*

$$Class(x, East) \leftarrow Has\_Car(x, y), Short(y), Closed(y)$$

Here $Cp(x) = Has\_Car(x, y) \wedge Short(y) \wedge Closed(y)$. *The following relational features can be obtained from this classification clause:*

$$F_1(x) \leftarrow Has\_Car(x, y), Short(y)$$

and:

$$F_2(x) \leftarrow Has\_Car(x, y), Closed(y)$$

The classification clause could now be written as:

$$Class(x, c) \leftarrow F_1(x), F_2(x)$$

Clauses like the last one in the example above will be called feature-clauses. Somewhat informally:

Definition 4 Feature-clause. *A feature-clause is a clause in which all negative literals are relational features.*

We note that "propositionalization" of relational data is simply a conversion of a relational instance into a vector-form, using relational features.

Definition 5 Feature Vector of a Relational Instance. *For a set of relational features* $\mathcal{F}$ *ordered in some canonical sequence* $(F_1, F_2, \ldots, F_d)$ *we obtain a Boolean-vector representation* $\mathbf{x}' \in \{0,1\}^d$ *of a relational instance* $x \in \mathcal{X}$ *using the function* $FV : \mathcal{X} \rightarrow \{0,1\}^d$, *where the* i^{th} *component* $FV_i(x) = 1$ *if* $F_i(x) \mapsto TRUE$, *and 0 otherwise. We will sometimes say* $\mathbf{a}'$ *is the* feature-space *representation of relational instance* a.

In this paper, the components for predicting the class-label of a relational instance a using a DRM M are shown in Fig. 2. This is a 2-step process: (1) The feature-vector representation $\mathbf{a}'$ of a is obtained using the feature-definitions found during the training stage; and (2) $\mathbf{a}'$ is provided as input to the model M found during training, which then computes the class-label (Fig. 3).
Although we are only concerned with DRMs in this paper, the logical explanations we construct apply more broadly to black-box models that use a feature-vector represent ion of relational instances in the sense defined in Defn. 5.

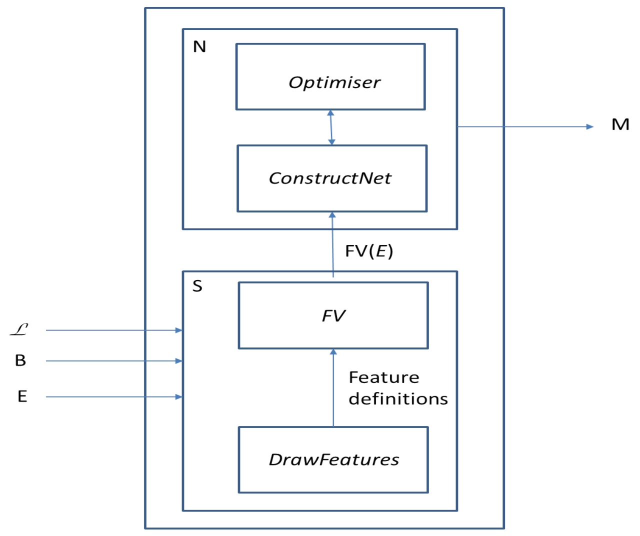


Figure 2: More details on the training component in Fig. 1. $\mathcal{L}$ denotes language constraints on the relational features, and $DrawFeatures$ is the stochastic feature construction procedure described in Vig et al. (2018). Features are used to represent ("propositionalize") each relational instance in the training data as a feature-vector. FV(E) denotes the feature-vectors of all relational instances E in the training data. Model M is a DRM. The optimiser finds the best structure and parameters for the DRM using FV(E).

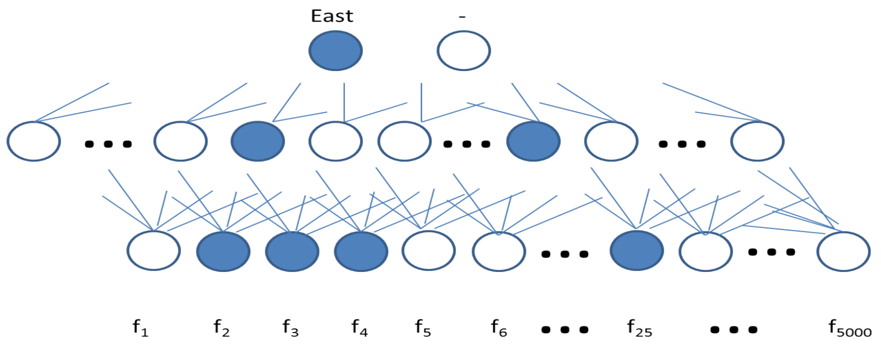


Figure 3: Prediction of the class of a relational instance a by a DRM. The inputs f_i are the component values of the feature-vector $FV_i(a)$. The shaded inputs denote a feature-value of 1. With a DRM, there is more to $f_i = 1$ than just an assignment of 1 to feature: it also means that some conjunction of background predicates $Cp_i(a)$ is $TRUE$ for the relational instance a. The shaded circles in hidden layers denote activated nodes. The shaded output node denotes the prediction of the relational instance is $East$.

3. Logical Explanations using Feature-Clauses

We introduce first a more general notion of a logical explanation.

Definition 6 Logical Explanation. *We assume a set of relational instances $\mathcal{X}$, and a finite set of classes $\mathcal{Y}$. Given a relational instance $a \in \mathcal{X}$ and a label $c \in \mathcal{Y}$, the statement $Class(a, c)$ will denote that the class of a is c. Let $e = Class(a, c)$. Then given background knowledge B, a logical explanation for e is a set of clauses H s.t. $B \cup H \models e$.*

An explanation using feature-clauses is a special case of a logical explanation.

Example 2 An explanation in the trains problem. *One logical explanation $H = \{C\}$ for the first train in the left column of in Example 1 is the feature-clause:*

$$C : Class(t1, East) \leftarrow F_1(t1), F_2(t1), F_3(t1)$$

where B contains:

$$\forall x(F_1(x) \leftarrow \exists y(Has\_Car(x, y), Short(y)))$$
$$\forall x(F_2(x) \leftarrow \exists y(Has\_Car(x, y), Closed(y)))$$
$$\forall x(F_3(x) \leftarrow \exists y(Has\_Car(x, y), Long(y)))$$

Here $t1$ is used as short-form for a structured term, describing the train, and, as before, let us assume that appropriate definitions exist in B for predicates like Has_Car, $Long$, $Short$, and $Closed$ to succeed (or fail) on terms like $t1$.

We will now restrict attention to explanations consisting of feature-clauses. It is useful for this paper to consider such explanations in two categories: those consisting of a single feature-clause, and those consisting of multiple feature-clauses.

3.1. Single-Clause Explanations

Assume we have a set of d features $\mathcal{F}$ with corresponding feature-definitions in the background knowledge B. We will first consider the case where explanation H consists of a single feature-clause. That is, $H = \{C\}$, where C is a definite clause $\forall x$ $(Class(x, c) \leftarrow Body)$. Here $Body$ is a conjunction of $F_i/1$ literals, each of which is defined in B. The ILP practitioner will recognise that finding a single feature-clause explanation for $Class(a, c)$ is an instance of the "Single example clause, single hypothesis clause" situation identified in Muggleton (1994), which forms the basis of Explanation-Based Learning (EBL – see Mitchell et al., 1986).

One explanation for $e = Class(a, c)$ can be obtained immediately as follows. Consider the set $F_a = \{F_i : F_i \in \mathcal{F}$ s.t. $FV_i(a) = 1\}$ (let us call these the set of *active* features for a). By definition, if $FV_i(a) = 1$ then $F_i(a) \mapsto TRUE$. For simplicity, let us suppose that $F_a = \{F_1, \ldots F_k\}$ is the set of active features for a, and let $C :$ $Class(a, c) \leftarrow F_1(a), F_2(a), \ldots, F_k(a)$. Assuming the F_i are defined in B, $H = \{C\}$ is an explanation for e.[4] Algorithm 1 is a simple procedure to construct a single-clause explanation C' that relies on the result that if $C' \preceq_\theta C$, then $C' \models C$. Therefore, if $B \cup \{C\} \models e$,

4. ILP practitioners will recognise C as being analogous to the most-specific clause in Muggleton (1995), and we will call it the most-specific feature-clause for e, given $\mathcal{F}$ and B.

then evidently $B \cup \{C'\} \models e$. That is, $\{C'\}$ will also be an explanation for e. For reasons that will be apparent, we will call $\{C'\}$ an "unstructured" explanation.[5]

Algorithm 1: A non-deterministic procedure for identifying a single-clause unstructured explanation.

```
1 Algorithm ConstructUnstruct($e, B, \mathcal{F}$)
    Input: A relational instance $e$; background knowledge $B$; and a set of features $\mathcal{F}$ with
           definitions in $B$.
    Output: A single feature-clause explanation $H$ s.t. $B \cup H \models e$.
2   Let $e = Class(a, c)$
3   Let $a' = FV(a)$
4   Let $\mathcal{F}'$ be the set of features that map to $TRUE$ in $a'$
5   Let $\mathcal{H}$ be the subset-lattice of $\mathcal{F}'$
6   Let $\mathcal{F}''$ be any non-empty subset in $\mathcal{H}$
7   Let $C' = \forall x(Class(x, c) \leftarrow Body)$
8       where $Body$ is the conjunction of features $F_i(x)$ for $F_i \in \mathcal{F}''$,
9   return $\{C'\}$
```

Example 3 *ConstructUnstruct* **for the trains problem.** *Let the background knowledge B contain the relational features:*

$$\forall x(F_1(x) \leftarrow \exists y(Has\_Car(x, y), Short(y)))$$

$$\forall x(F_2(x) \leftarrow \exists y(Has\_Car(x, y), Closed(y)))$$

$$\forall x(F_3(x) \leftarrow \exists y(Has\_Car(x, y), Long(y)))$$

$$\forall x(F_4 x) \leftarrow \exists y(Has\_Car(x, y), Long(y), Closed(y)))$$

Then, for $ConstructUnstruct(Class(t1, East), B, \{F_1, F_2, F_3, F_4\})$ (where $t1$ denotes the first train on the left in Example 1):

1. *$a' = [1, 1, 1, 0]$;*
2. *$\mathcal{F}' = \{F_1, F_2, F_3\}$;*
3. *$\mathcal{F}''$ can be any non-empty subset of $\{F_1, F_2, F_3\}$ (for example, $\mathcal{F}'' = \{F_1, F_3\}$); and*
4. *C' is the clause $\forall x(Class(x, East) \leftarrow Body)$ where $Body$ is the conjunction of literals $F_i(x)$ where $F_i \in \mathcal{F}''$ (for example, $C' = \forall x(Class(x, East) \leftarrow F_1(x), F_3(x)))$*

In practice, we will need some way of searching the space of subsets of $\mathcal{F}'$. A fidelity-measure introduced later allows us to compare unstructured explanations, and forms the basis for guiding a search strategy.

5. In practice, in Algorithm 1 the lattice in Step 5 would be represented by a graph, and finding an element of the lattice in Step 6 will involve some form of optimal graph-search to find an optimal (or near-optimal) solution. More on this later.

3.2. Multi-Clause Explanations

We now extend explanations to a restricted form of *multi-clause explanation*, able to use features not all of which are defined in background knowledge B. These "invented" features will be required to be (re-)expressible in terms of features already defined in B. In this paper we will require structured and unstructured explanations to be related by the logic-program transformation operations of *folding* and *unfolding*. That is, given an unstructured explanation H and a structured explanation H_1, H can be derived from H_1 using one or more unfolding transformations; and H_1 can be derived from H using one or more folding transformations. We describe the transformations, along with the conditions that ensure that the computable answers do not change when the transformation is applied.

Definition 7 (One-Step) Unfolding of a clause (Hogger, 1990). *Given a set of definite clauses P, w.l.o.g. let $C \in P$ s.t. $C = Head \leftarrow L_1, \ldots, L_i, \ldots, L_k$ $(k \geq 1)$. Let $C' \in P$ be a clause ${L'}_i \leftarrow Body'$ s.t. L_i and ${L'}_i$ unify with m.g.u. θ. Then the (one-step) unfolding of C w.r.t. L_i using P is the clause $Unfold(C) : (Head \leftarrow L_1, \ldots, L_{i-1}, Body', L_{i+1}, \ldots, L_k)\theta$*

Definition 8 (One-Step) Folding of a clause (Hogger, 1990). *Given a set of definite clauses P, w.l.o.g. let $C \in P$ s.t. $C = Head \leftarrow L_1, \ldots, L_{i-1}, Body_i, L_{i+1}, \ldots, L_k$, where $Body_i$ is some literals $B_1, B_2, \ldots, B_j$ $(j, k \geq 1)$. Let C' be a clause ${L'}_i \leftarrow Body'$ s.t. there is a substitution θ that satisfies: (a) $Body_i = Body'_i\theta$; (b) Every existentially quantified variable y in C, $y\theta$ is a variable that occurs only in $Body_i$ and nowhere else; (c) For any pair of distinct existential variables y, z in C $y\theta \neq z\theta$; and (d) C' is the only clause in P whose positive literal unifies with ${L'}_i\theta$. Then the (one-step) folding of C w.r.t. $Body_i$ using P is the clause $Fold(C) : (Head \leftarrow L_1, \ldots, L_{i-1}, {L'}_i\theta, L_{i+1}, \ldots, L_k)$.*

Remark 1 Correctness of Transformations. *In Petterossi and Proietti (1998) the unfold and fold transformations defined above are shown to be correct as replacement rules w.r.t. the minimal-model (MM) semantics. That is, if P is a definite-clause program and $C \in P$, then $MM(P) = MM((P - \{C\}) \cup Trans(C))$, where $Trans(C)$ is $Fold(C)$ or $Unfold(C)$.*

We can use this to construct structured explanations that are correct, in the computational sense just identified. That is, a structured explanation can replace an unstructured explanation without altering the minimal model of the (program containing) the unstructured explanation. It will be convenient to define the notion of an "invented" feature.

Definition 9 Invented Feature. *Given background knowledge B, a feature F is said to be an invented feature if: (1) F is not defined in B; and (2) The definition of F is a feature-clause C whose body contains features in B only; or is a clause that unfolds to a feature-clause C containing features only in B. We will sometimes denote C as $UFC(F)$ (short for "unfolded feature-clause for F").*

That is, $UFC(F)$ unfolds the definition of F only to the extent that the result is a feature-clause (that is, all literals in the definition are features defined in the background knowledge).

Example 4 Invented features in the trains problem. *The features $F_{1,1}$ and $F_{1,2}$ below are invented features:*

$F_{1,1}(x) \leftarrow F_2(x), F_{1,2}(x)$
$F_{1,2}(x) \leftarrow F_4(x), F_9(x)$

where the F_i are features defined in background knowledge:

$F_2(x) \leftarrow Has\_Car(x,y), Short(y)$
$F_4(x) \leftarrow Has\_Car(x,y), Wheels(y,3)$
$F_9(x) \leftarrow Has\_Car(x,y), Load(y, triangle)$

Then $UFC(F_{1,1})$ is:

$F_{1,1,}(x) \leftarrow F_2(x), F_4(x), F_9(x)$

and $UFC(F_{1,2})$ is identical to the definition of $F_{1,2}$.

We distinguish between unstructured and structured definitions based on the presence or absence of invented features.

Definition 10 Unstructured and structured explanations. *Let $a \in \mathcal{X}$ and $\mathcal{Y}$ be a set of class labels, with $c \in \mathcal{Y}$. Let $N : \Re^d \rightarrow \mathcal{Y}$ be a predictive model such that $N(FV(a)) = c$. Given background knowledge B, let H be an explanation for $Class(a, c)$ containing the feature-clause $C : \forall x(Class(x, c) \leftarrow Body)$. Let $Body$ consist of features $\mathcal{F}_C$. Let $\mathcal{F}_C$ be partitioned into: (a) $\mathcal{F}_{C,old}$, consisting of features defined in B; and (b) $\mathcal{F}_{C,new}$, consisting of features defined in $H - B$.*

We will call H an unstructured explanation *iff: (1) $\mathcal{F}_C$ contains only old features (that is, $\mathcal{F}_{C,new} = \emptyset$); and (2) $H = \{C\}$.*

We will call H a structured explanation *iff: (1) $\mathcal{F}_C$ contains only invented features (that is, $\mathcal{F}_{C,old} = \emptyset$); and (2) $H = \{C\} \cup InvF$, where $InvF$ only contains clauses defining invented features in $\mathcal{F}_{C,new}$; (3) For each feature F in $\mathcal{F}_{C,new}$ there is a single clause definition in $InvF$ s.t. F unfolds to a unique feature-clause defined using features in B only; and (4) At least one $F \in \mathcal{F}_{C,new}$ unfolds to a feature-clause that contains at least 2 features from B (that is, there is at least one invented feature that is not a trivial rewrite of features in B).*[6]

That is, a structured explanation is a set of clauses containing a classification clause C along with definitions of invented features (and thus is a case of the "single example clause, multiple hypothesis clauses" situation in Muggleton (1994)).

Example 5 Unstructured and structured explanations for the trains problem. *Suppose we are given an instance $a \in \mathcal{X}$, and a set of features $\mathcal{F}$, each defined by feature-definitions in background B. Then we will call the following an unstructured explanation (x is universally quantified):*

$$C : Class(x, East) \leftarrow F_2(x), F_3(x), F_4(x), F_9(x)$$

6. If required, for a structured explanation containing a clause C with a $F \in \mathcal{F}_{C,new}$ defined by a feature-clause whose clause body contains exactly one feature $F' \in \mathcal{F}_{C,old}$, a more compact "semi-structured" explanation can be obtained by a one-step folding. See the example that follows.

The following is a structured explanation H_1:

$Class(x, East) \leftarrow F_{1,1}(x), F_{1,2}(x)$
$F_{1,1}(x) \leftarrow F_2(x), F_3(x)$
$F_{1,2}(x) \leftarrow F_4(x), F_9(x)$

where, as usual, the features $F_{2,3,4,9}$ are defined as background knowledge:

$F_2(x) \leftarrow Has\_Car(x, y), Short(y)$
$F_3(x) \leftarrow Has\_Car(x, y), Closed(y)$
$F_4(x) \leftarrow Has\_Car(x, y), Wheels(y, 3)$
$F_9(x) \leftarrow Has\_Car(x, y), Load(y, triangle)$

Note that H_1 unfolds to H. Another structured explanation that also unfolds to H is H_2 below:

$Class(x, East) \leftarrow F_{1,1}(x), F_{1,2}(x)$
$F_{1,1}(x) \leftarrow F_1(x), F_4(x)$
$F_{1,2}(x) \leftarrow F_3(x), F_9(x)$

We will assume that explanations of the form $Class(x, East) \leftarrow F_{1,1}(x), F_3(x), F_9(x)$ will be represented as H_2. Also, by definition, the following is not a structured explanation, since all new features are trivial rewrites of existing ones:

$Class(x, East) \leftarrow F_{1,1}(x), F_{1,2}(x)$
$F_{1,1}(x) \leftarrow F_1(x)$
$F_{1,2}(x) \leftarrow F_3(x)$

But this is a structured explanation:

$Class(x, East) \leftarrow F_{1,1}(x), F_{1,2}(x)$
$F_{1,1}(x) \leftarrow F_1(x)$
$F_{1,2}(x) \leftarrow F_3(x), F_9(x)$

It is obvious enough that every structured explanation allows the derivation by unfolding of a correct unstructured explanation.

Remark 2 Deriving an Unstructured Explanation from a Structured Explanation. *Let $H = \{C'\} \cup InvF$ be a structured explanation for a relational example $Class(a, c)$ given background knowledge B. W.l.o.g. from Defn. 10, C' is a clause $Class(x, c) \leftarrow \{F'_1(x), F'_2(x), \ldots, F'_k(x)\}$ s.t. each F'_i unfolds using $InvF$ to a unique feature-clause $F'_i(x) \leftarrow F_{i,1}(x), \ldots, F_{i,n_i}(x)$, where the $F_{i,\cdot}$ are defined in B. From Defn.7, unfolding C' w.r.t. the F'_i using $InvF$ results in the clause C : $Class(x, c) \leftarrow F_{1,1}, \ldots, F_{k,n_k}$. From Remark 1, the minimal model of $B \cup H$ will be unchanged by replacing C' with C. It follows that $H = \{C\}$ is an unstructured explanation for $Class(a, c)$ that is computationally equivalent to H'. Here "computationally equivalent" means that the structured explanation computes the same set of answers under the minimal model semantics as the unstructured explanation (see Hogger, 1990).*

One or more correct structured explanations follow from an unstructured explanation if the conditions in Defn. 8 hold. Constraints on the invented features can ensure this.

Remark 3 Deriving Structured Explanations from an Unstructured Explanation. *Let $H = \{C\}$ be an unstructured explanation for a relational example $Class(a, c)$ given background knowledge B. W.l.o.g. from Defn. 10, C is a clause $Class(x, c) \leftarrow \{F_1(x), F_2(x), \ldots, F_n(x)$ s.t. each F_i is defined in B. Let $InvF$ consist of k features ${F'}_1, \ldots, {F'}_k$ s.t. each ${F'}_i$ is uniquely defined by features in a block of a k-partition of $F_1, \ldots, F_n$. Let C' be the clause $Class(x, c) \leftarrow {F'}_1(x), {F'}_2(x), \ldots, {F'}_k(x)$. Then $H' = \{C'\} \cup InvF$ is a structured explanation for $Class(a, c)$ that is computationally equivalent to $B \cup \{C\} \cup InvF$. This follows since the conditions (a)–(d) in Defn. 8 are trivially satisfied (condition (a) follows with θ being a simple renaming substitution; (b)–(c) follow since there are no existential variables in the definitions of the ${F'}_i$; and (d) since there is a single clause definition for each feature in $InvF$).*

Remark 3 suggests a straightforward non-deterministic procedure to construct a correct structured explanation (Algorithm 2).

Algorithm 2: A non-deterministic procedure for obtaining a structured explanation from an unstructured explanation, by inventing k features. The condition in Step 5 is not required by Remark 3, but prevents inventing features that are trivial rewrites of existing features (see Defn. 10).

```
1  Algorithm ConstructStruct($H, k$)
       Input: An unstructured explanation $H$; and a number $k$ ($\geq 2$).
       Output: A structured explanation $H'$ that is computationally equivalent to $H$.
2      Let $H = \forall x(Class(x, c) \leftarrow Body)$.
3      Let $F$ be the set of features in $Body$.
4      Let $P_k$ be a set s.t. one of the following is true:
5          Either (a) $P_k$ is a $k$-partition of $F$ s.t. at least one block in $P_k$ contains 2 or more
             elements;
6          Or (b) $P_k = \emptyset$ (if no such $k$-partition exists).
7      if $P_k = \emptyset$ then
8          return $\emptyset$
9      end
10     else
11         Let $P_k$ consist of blocks $b_1, b_2, \ldots, b_k$.
12         for each block $b_i \in P_k$ do
13             Construct a new feature ${F'}_i$ with definition $C_i = \forall x({F'}_i(x) \leftarrow Body_i)$, where
                 $Body_i$ is the conjunction of the features in $b_i$.
14             Let $C_0 = \forall x(Class(x, c) \leftarrow Body_0)$ where $Body_0$ is the conjunction of the
                 ${F'}_1, {F'}_2, \ldots, {F'}_k$.
15             Let $H' = \bigcup_{i=0}^{k} C_k$.
16         end
17     end
18     return $H'$
```

Example 6 *ConstructStruct* **for the trains problem.** *Let H be the clause $\forall x(Class(x, East) \leftarrow F_2(x), F_3(x)F_4(x), F_9(x))$ where $F_{2,3,4,9}$ are defined in the background knowledge B. Then for $ConstructStruct(H, 2)$:*

1. *P_2 can be any 2-partition of $\{F_2, F_3, F_4, F_9\}$ s.t. at least one block of P_2 contains more than 1 element. (for example, $P_2 = \{[F_2, F_4], [F_3, F_9]\}$)*
2. *Feature-clauses C_1 and C_2 are are constructed using the set of features in each block (for example, $C_1 = F_1'(x) \leftarrow F_2(x), F_4(x)$ and $C_2 = F_2'(x) \leftarrow F_3(x), F_9(x)$)*
3. *A classification clause C_0 is constructed using the new feature-definitions (for example, $C_0 = \forall x(Class(x, East) \leftarrow F_1'(x), F_2'(x))$*
4. *A structured explanation $H' = \{C_0, C_1, C_2\}$ is returned*

In practice, we will need some way of searching the space of 2-partitions (in general, k-partitions). A partial-ordering introduced later allows us to compare structured explanations, and forms the basis for guiding a search strategy.

All explanations so far have only referred to a single relational instance. What is to be done when we seek explanations for more than one instance? It suffices to consider the case of 2 instances e_1 and e_2. Then, an explanation H for both instances is simply an explanation for $e_1 \wedge e_2$ (that is, $B \cup H \models (e_1 \wedge e_2)$). We use this to formulate a notion of "locally consistent" explanations.

3.3. Locally Consistent Explanations

Recent research called LIME ("Local Interpretable Model-agnostic Explanations" – see Ribeiro et al., 2016) proposes producing readable proxies "on-demand", for any kind of black-box predictor. The key feature is that a LIME-style explanatory model is constructed only when a prediction for an instance is sought, and the explanation is required to be *faithful* to the black-box's predictions for the instance and its near-neighbours. The principal intuition underlying LIME is this: a single comprehensible model may not be possible for the black-box's predictions over all instances but, nevertheless, it may be possible to construct multiple readable proxies, each of which is consistent with the predictions for groups of similar ("local") instances.

However, the requirement of having to be consistent with only local instances results in a different kind of problem. Since the set of near-neighbours of an instance may be quite small, there may be insufficient constraints, in data-theoretic terms, to narrow down a unique (or even a small number of) explanations. So, how then is an explanation to be selected? In LIME, this is left to the loss-function. In Bayesian terms, this means defining an appropriate prior to guide selection when the data are insufficient.

We draw on specific situations described in Muggleton (1994) to clarify what is meant by a local explanation, and its data-theoretic evaluation. Later we will introduce a domain-dependent prior to develop a Bayes-like selection of explanations.

The explanations we seek are for predictions of relational instances that are "close" to each other. Readers familiar with the ILP literature will recognise that the task of finding such an explanation corresponds to either the "Multiple examples, single hypothesis clause" or the "Multiple examples, multiple hypothesis clauses" situations in Muggleton (1994), depending on whether unstructured or structured explanations are constructed. We note first that if a logical explanation is extracted for the full DRM in the manner done by França et al. (2015), then alternate explanations for an instance $e = Class(a, c)$ can be provided in terms of the rules that can be used to derive e. Each such explanation would be an unstructured explanation in the terminology of this paper. This does not alter the notion that follows of finding the best local explanation.

We have seen how to construct the most-specific feature-clause for $e = Class(a, c)$ from the set of active features for a. We now seek explanations that are not only consistent with a predictive model on an instance x, but also are consistent with predictions made by the predictive model in a *local neighbourhood* of x.

Definition 11 Neighbourhood. *Given relational instance $a \in \mathcal{X}$, and a set of d features $\mathcal{F}$, and given some $\epsilon \in \Re$, we denote the neighbourhood of a as $Nbd(a) = \{x : x \in \mathcal{X} \text{ and } Dist(FV(a), FV(x)) \leq \epsilon\}$. Here Dist is some appropriate distance measure defined over d-dimensional vectors.*

In practice the dimensionality d can be quite large, and since the standard Euclidean distance is known to be problematic in high-dimensional spaces (Aggarwal et al., 2001) we will use an alternative measure.

Definition 12 Locally Consistent Explanations. *Given relational instance $a \in \mathcal{X}$, and a predictive model N, let $N(FV(a)) = c$. We define the following subsets of $Nbd(a)$: $E^+(a) = \{b : b \in Nbd(a) \text{ and } N(FV(b)) = c\}$ and $E^-(a) = \{b : b \in Nbd(a) \text{ and } N(FV(b)) \neq c\}$. Then an explanation H for $Class(a, c)$ given B is a locally consistent explanation if: (1) for each $a' \in E^+(a)$ H is an explanation for $Class(a', c)$ (that is, $B \cup H \models Class(a', c)$); and (2) for each $a' \in E^-(a)$ $B \wedge H \wedge \neg Class(a', c) \not\models \Box$.*

Example 7 Locally consistent explanation for the trains problem. *For simplicity, we consider the situation where $E^- = \emptyset$. Suppose we now know that the local neighbourhood of the first train ($t1$) in the left column of Example 1 only contains the second train ($t2$) in that column. Let us assume that $\mathcal{F}$ consists just of the functions defined in Example 2. With those definitions, and a predictive model N, let $N(FV(t1)) = N(FV(t2))$ (= East, say). Then $E^+(t1) = \{t1, t2\}$.*

The most-specific feature-clause for $Class(t1, East)$ given $\mathcal{F}$ and B is:

$$C1 : Class(t1, East) \leftarrow F_1(t1), F_2(t1), F_3(t1)$$

and for $Class(t2, East)$ is:

$$C2 : Class(t2, East) \leftarrow F_1(t2), F_2(t2)$$

where the function definitions are as before. Then a locally consistent explanation for $Class(t1, East)$ is the least-general-generalisation (or Lgg) of $C1$ and $C2$:

$$Lgg(C1, C2) : \forall x(Class(x, East) \leftarrow F_1(x), F_2(x))$$

In general, $E^- \neq \emptyset$, and results from ILP tell us that it may not be possible to find a single clause that is locally consistent. We next describe a simple, qualitative form of Bayes Rule that combines likelihood of the data, with a relevance-based prior preference over explanations.

It is useful before we proceed further to have a numerical measure of the extent to which an explanation is locally consistent.

Definition 13 Fidelity. *Let $a \in \mathcal{X}$. Given a predictive model N, let $N(FV(a)) = c$ and $E^+(a)$, $E^-(a)$ as before. Let $D = (E^+, E^-)$ and H be an explanation for $Class(a, c)$ given background knowledge B. Let: (1) $AgreePos(H) = \{b : b \in E^+(a)$ and $B \wedge H \models Class(b, c)\}$; and (2) $AgreeNeg(H) = \{b : b \in E^-(a)$ and $B \cup H \wedge \neg Class(b, c) \not\models \Box\}$. Then $Fidelity(H|D, B) = F_{H,a} = \frac{|AgreePos(H)|+|AgreeNeg(H)|}{|E^+(a)|+|E^-(a)|}$.*

We note that $AgreePos(H)$ and $AgreeNeg(H)$ are the same as true positives and true negatives in the classification literature, and so *fidelity* is a localized form of accuracy (in Ribeiro et al. (2016) the term "local fidelity" is used for a localized form of error). If the predictor allows a probabilistic prediction, then a finer-grained weighted fidelity is computable.

Remark 4 Structuring preserves fidelity. *Let $H' = \{C'\} \cup InvF$ be a structured explanation derived as in Remark 3 from an unstructured explanation $H = \{C\}$ for a relational example $Class(a, c)$, given definite clauses B and a predictor N s.t. $N(FV(a)) = c$. Let $E^+(a)$ and $E^-(a)$ denote relational examples obtained from the neighbourhood of a as before, and $D = (E^+(a), E^-(a))$. Then, $Fidelity(H'|D, B) = Fidelity(H|D, B)$.*

Let S be any finite set of ground $Class/2$ facts and $MM(P)$ be the minimal model of a definite clause program P. Since $InvF$ is a set of definite clauses containing only definitions of invented features, $S \cap MM(B \cup \{C\}) = S \cap MM(B \cup \{C\} \cup InvF)$. Now:

$$AgreePos(H) = \{Class(a', c) : Class(a', c) \in (E^+(a) \cap MM(B \cup H)\}$$
$$AgreeNeg(H) = \{Class(a', c) : Class(a', c) \in (E^-(a) - (E^-(a) \cap MM(B \cup H)\}.$$

Since $E^+(a)$ and $E^-(a)$ are finite sets of ground $Class/2$ facts, and $H = \{C\}$:

$$AgreePos(H) = \{Class(a', c) : Class(a', c) \in (E^+(a) \cap MM(B \cup \{C\} \cup InvF)\}$$
$$AgreeNeg(H) = \{Class(a', c) : Class(a', c) \in (E^-(a) - (E^-(a) \cap MM(B \cup \{C\} \cup InvF)\}.$$

From Remark 3, $MM(B \cup \{C\} \cup InvF) = MM(B \cup \{C'\} \cup InvF)$. Since $H' = \{C\} \cup InvF$, it follows immediately that $AgreePos(H) = AgreePos(H')$ and $AgreeNeg(H) = AgreeNeg(H')$ and therefore $Fidelity(H|D, B) = Fidelity(H'|D, B)$. □

Example 8 (Fidelity-preserving structuring for the trains problem). *The following is a structured explanation H_1:*

$$Class(x, East) \leftarrow F_{1,1}(x), F_{1,2}(x)$$
$$F_{1,1}(x) \leftarrow F_2(x), F_3(x)$$
$$F_{1,2}(x) \leftarrow F_4(x), F_9(x)$$

Since there are no new existentially quantified variables in feature-clauses, and there is a single clause defining each of $F_{1,1}$ and $F_{1,2}$, H_1 will unfold uniquely (see Gabbay et al. (1998)) to the unstructured explanation H:

$$Class(x, East) \leftarrow F_2(x), F_3(x), F_4(x), F_9(x)$$

That is, the minimal-models of H and H_1 will contain the same ground instances of $Class(x, East)$ (Hogger (1990)). Given the data D and background knowledge B it follows that the $AgreePos(H) = AgreePos(H_1)$ and $AgreeNeg(H) = AgreeNeg(H_1)$. Consequently, $Fidelity(H|D, B) = Fidelity(H_1|D, B)$.

From now on, we will take the black-box predictor to be a DRM, although what follows continues to apply to any black-box predictor that uses relational feature values as inputs. In the following section we address selecting a local explanation, but first we draw attention to some general issues for local fidelity.

Remark 5 Determination of local fidelity. *The fidelity measure can be seen, roughly, as a degree of equivalence between two theories, one predictive and the other explanatory, taken as an average of the local fidelities over a set of neighbourhoods of instances. However, this tells us nothing about a strategy to sample the* instances to be explained*. This must be based on the* objectives *for the explanation, which may vary considerably between applications. One strategy in LIME (Ribeiro et al., 2016), for example, is to select, or "pick" test instances that are diverse yet cover well the space of representable features. Since a human user is assumed to have only finite time to inspect explanations, a fixed number of instances is generated using a weighted coverage optimisation algorithm, maximising the use of important features (according to feature weights in the explanatory models) while minimising redundancy between instances. In Section 5.2 of this paper, however, a different strategy is adopted, due to differing assumptions regarding users.*

4. Selecting a Local Explanation

Given a relational instance $a \in \mathcal{X}$, let the prediction of a by a DRM N be c. But, for a definition of a neighbourhood, there may be several explanations for $Class(a, c)$ with the same maximal fidelity. How then should we select a single explanation? Provided we have some reasonable way of specifying prior-preferences, Bayes rule trades-off the fit to the data (likelihood) against prior preference (in Ribeiro et al. (2016), LIME's minimisation of the sum of a loss and a regularisation term can be seen as implementing a form of Bayesian selection).

A general approach for selecting amongst logical formulae is provided by labelled deductive systems (LDS – see Gabbay, 1996), in which logical formulae are extended with labels and an associated algebra. For our purposes, it is sufficient simply to consider labelled explanations.

Definition 14 Labelled Explanation. *Given a relational example $e = Class(a, c)$, and background knowledge B, $\alpha : H$ is a labelled explanation for e given B if: (a) H is an explanation for e given B; and (b) α is a element of a partially-ordered set of ground first-order terms Δ. Δ consists of annotations of all explanations for e given B.*

A comparison of labelled explanations follows simply from the partial ordering on the labels. That is, $\alpha : H_1 \preceq \beta : H_2$ iff $\alpha \preceq \beta$. Here we will take the label of an explanation H to be a pair $\langle L_H, P_H \rangle$, which allows several different kinds of comparisons, given background knowledge B and data D:

Quantitative. This is appropriate when both L_H and P_H are on an interval scale (that is, numeric values). Examples are: (a) the usual Bayesian comparison, using $L_H = P(D|H, B)$ and $P_H = P(H|B)$; and $\langle L_{H_1}, P_{H_1} \rangle : H_1 \preceq \langle L_{H_2}, P_{H_2} \rangle : H_2$ iff $\log L_{H_1} + \log P_{H_1} \leq \log L_{H_2} + \log P_{H_2}$; (b) Good's explicativity (Chapter 23 of Good (1983)), which uses the same L_H, P_H, but uses the function $\log L_H + \gamma \log P_H$ with $0 < \gamma < 1$; and (c) Likelihood-based, using $L_H = P(D|H,B)$ and P_H is the uniform distribution.

Qualitative. Here both L_H and P_H are both on an ordinal scale (that is, only comparisons of values are possible). Examples are: (a) The qualitative Bayesian comparison in the manner proposed by Coletti and Scozzafava (1993). With some abuse of notation, $\langle L_{H_1}, P_{H_1}\rangle : H_1 \preceq \langle L_{H_2}, P_{H_2}\rangle : H_2$ iff $L_{H_1} \preceq L_{H_2}$ and $P_{H_1} \preceq P_{H_2}$. If it is not the case that $\langle L_{H_1}, P_{H_1}\rangle : H_1 \preceq \langle L_{H_2}, P_{H_2}\rangle : H_2$ or $\langle L_{H_2}, P_{H_2}\rangle : H_2 \preceq \langle L_{H_1}, P_{H_1}\rangle : H_1$, then the labelled explanations are not comparable; and (b) A dictionary-ordering, in which $\langle L_{H_1}, P_{H_1}\rangle : H_1 \preceq \langle L_{H_2}, P_{H_2}\rangle : H_2$ iff $L_{H_1} \prec L_{H_2}$, or $L_{H_1} = L_{H_2}$ and $P_{H_1} \preceq P_{H_2}$.[7]

Semi-Quantitative. Here, one of L_H or P_H is on an interval scale, and the other is on an ordinal scale. The qualitative comparisons above can be adapted to this, by replacing $\preceq$ and $\prec$ with $\leq$ and $<$ for the numeric quantity.

We will use the semi-quantitative dictionary ordering with $L_H = Fidelity(H|D,B)$ in this paper, and P_H is an ordinal-valued prior based on an assessment of relevance. Using the semi-quantitative setting and the dictionary ordering has some advantages:

(a) Quantitative selection based on a Bayesian score requires a definition of both $P(D|H,B)$ and $P(H|B)$. While the first can be obtained easily enough, it is not obvious how to specify a prior distribution over explanations. The usual approach of using a mathematically convenient function like $2^{-|H|}$, where $|H|$ is some measure of the size of H, may not be an appropriate translation of prior assessment of relevance of explanations; and

(b) A qualitative Bayesian approach as defined above usually ends up with many incomparable explanations. The dictionary ordering decomposes the task of identifying explanations into two parts: the first part that maximises fidelity, and the second part that maximises the prior amongst maximal fidelity explanations. Under some circumstances (see the Appendix), maximising fidelity is equivalent to maximising log likelihood. In those cases, the dictionary ordering uses the prior to select amongst maximum likelihood explanations. Usefully, there is also an implementation benefit that follows from the result in Remark 4: since structuring does not alter the fidelity, the first part can simply examine unstructured explanations.

We turn now to prior information P_H that captures some aspects of what constitutes a comprehensible explanation.

4.1. A Relevance-Based Prior

In Srinivasan et al. (2003) the authors investigate the utility of including an expert assessment of the relevance of relations included in the background knowledge.

7. This may not yield the same results as the qualitative Bayesian comparison above. Differences arise when $L_{H_1} \prec L_{H_2}$ but $P_{H_2} \preceq P_{H_1}$. Under the dictionary-ordering H_2 would be preferred, but the qualitative Bayesian approach would find H_1 and H_2 incomparable.

Example 9 Relevance information in the trains problem. *Suppose the background knowledge B for the trains problem contains definitions of predicates like $Has\_Car/2$, $Short/1$, $Closed/1$, $Wheels/2$ and $Load/2$. For the problem of classifying trains as east-bound or west-bound, let us assume we are also given domain knowledge in the form of the relevance level of (sets) of predicates as follows: $r_1 : \{Wheels/2, Load/2\}, r_2 : \{Short/1, Closed/1\}$, and $r_1 \prec_r r_2$. That is, $Wheels$ and $Load$ are less relevant to the problem than $Short$ and $Closed$.*

We note that this is different to the notion of logical relevance of features (defined in terms of the entailment relation, $\models$). Here, we are concerned with domain-relevance of predicates used to define those features. This latter form of domain-specific relevance information can also form the basis of a preference ordering over explanations. Here is the view of the domain expert involved in Srinivasan et al. (2003):[8]

> *I think it is reasonable to argue that [a hypothesis using] more relevant prior background knowledge is more probable. I think that what makes hypotheses more probable is also a function of whether the predicates used are related to each other. Often you see hypotheses that seem to mix apples and oranges together, which seems to make little sense. Though of course this mixing of predicates may be because the ML system is trying to express something not easily expressible with the given background predicates.*

This suggests that relevance information can constitute an important source of prior knowledge. One route by which this is exploited by ILP systems is in the form of search constraints ("hypotheses that do not contain oxygens connected to hetero-aromatic rings are irrelevant"), or, as in the case of Srinivasan et al. (2003), in the incremental construction of hypotheses. Our interest here is to extend this use of relevance to select amongst hypotheses, by devising a relevance-based partial ordering over hypotheses.[9]

Definition 15 Relevance-assignments and orderings. *We assume that for some set of predicates $\mathcal{P}$ in background knowledge B, we have domain-specific relevance labels drawn from a set $\mathcal{R}$. A relevance assignment is a function $R : \mathcal{P} \rightarrow \mathcal{R}$. We assume that there is domain-knowledge in the form of a total ordering $\prec_r$ over the elements of $\mathcal{R}$. Then, for $a, b \in \mathcal{R}$, $a \preceq_r b$ iff $a \prec_r b$ or $a = b$. We will call $\prec_r$ a relevance ordering.*

A relevance ordering naturally results in the concept of ordered intervals: $[a, b]$ is an *ordered relevance interval* (or simply a *relevance interval*) if $a, b \in \mathcal{R}$ and $a \preceq_r b$. The ordering $\prec_r$ gives rise in turn to an ordering over relevance intervals.

Definition 16 Ordering over relevance intervals. *Let $\mathcal{I}$ be the set of ordered relevance intervals. Let $[a, b]$ and $[c, d]$ be intervals in $\mathcal{I}$. Then $[a, b] \preceq_i [c, d]$ iff $a \preceq_r c$ and $b \preceq_r d$.*

Remark 6 ($\preceq_i$ is a partial ordering). *Let $\mathcal{I}$ be the set of ordered relevance intervals. Then: (a) Clearly for any interval $[a, b] \in \mathcal{I}$, $[a, b] \preceq_i [a, b]$; (b) For $[a, b] \in \mathcal{I}$ and $[c, d] \in \mathcal{I}$,*

8. R.D. King: personal communication
9. We will often reuse the generic symbols $\preceq$ and $\prec$ to denote partial- and total-orderings. The context will make it clear which sets these relations refer to.

let $[a, b] \preceq_i [c, d]$ and $[c, d] \preceq_i [a, b]$. Then $a \preceq_r c$ and $c \preceq_r a$ and $b \preceq_r d$ and $d \preceq_r b$. Since $\preceq_r$ is a partial order, it follows that $a = c$ and $b = d$; (c) For $[a, b] \in \mathcal{I}$, $[c, d] \in \mathcal{I}$ and $[e, f] \in \mathcal{I}$, let $[a, b] \preceq_i [c, d]$ and $[c, d] \preceq_i [e, f]$. From a similar argument to (b), it follows that $[a, b] \preceq_i [e, f]$. That is, $\preceq_i$ is reflexive, anti-symmetric and transitive.

We will use the following, slightly more general ordering, than $\preceq_i$:

Definition 17 (Ordering over sets of relevance-intervals) *Let S and S' be sets of relevance intervals. Then $S \preceq_s S'$ iff for every interval $[a, b]$ in S there exists at least one interval $[c, d]$ in S' s.t. $[a, b] \preceq_i [c, d]$. That is, $a \preceq_r c$ and $b \preceq_r d$.*

Remark 7 ($\preceq_s$ is a quasi-ordering). *Let $\mathcal{S}$ be the set whose elements are sets of relevance intervals and $\preceq_s$ as in Defn. 16. Then: (a) For $S \in \mathcal{S}$, clearly $S \preceq_s S$; and (b) Let $S_{1,2,3} \in \mathcal{S}$ s.t. $S_1 \preceq_s S_2$ and $S_2 \preceq_s S_3$. Then, from Defn. 16 for every interval $[a, b] \in S_1$ there is an interval $[c, d] \in S_2$ s.t. $[a, b] \preceq_i [c, d]$ and for every $[c, d] \in S_2$ there is an interval $[e, f] \in S_3$ s.t. $[c, d] \preceq_i [e, f]$. Since $\preceq_i$ is a partial order (Defn. 6), it follows that for every $[a, b] \in S_1$ there is an interval $[e, f] \in S_3$ s.t. $[a, b] \preceq_i [e, f]$. That is $S_1 \preceq_s S_3$. So, $\preceq_s$ is reflexive and transitive, and therefore a quasi-order. However, $\preceq_s$ is not anti-symmetric, as witnessed by $S_1 = \{[2, 5], [3, 6]\}$ and $S_2 = \{[3, 6]\}$, with the usual ordering on numbers. Here $S_1 \preceq_s S_2$ and $S_2 \preceq_s S_1$, but $S_1 \neq S_2$.*

We now construct, in stages, the relevance of an explanation.

Definition 18 Relevance of features. *Given background knowledge B, let $\prec_r$ be a relevance ordering over a set of relevance labels $\mathcal{R}$, and let $R : \mathcal{P} \rightarrow \mathcal{R}$ be a relevance assignment for some subset $\mathcal{P}$ of B. Let $\forall x(F(x) \leftarrow Cp(x))$ be the feature-definition for $F/1$ in which $Cp(x)$ is a conjunction containing predicates from $\mathcal{P}$ only. Then the relevance of the feature F is $Relev(F) = [l, h]$, where l is the minimum relevance of predicates in $Cp(x)$ according to $\prec_r$ and R, and h is maximum relevance of predicates in $Cp(x)$ according to $\prec_r$ and R.*

Example 10 (Relevance of features for the trains problem). *Suppose we have the following feature-definitions (omitting quantifiers for simplicity):*

$F_2(x) \leftarrow Has\_Car(x, y), Short(y)$
$F_3(x) \leftarrow Has\_Car(x, y), Closed(y)$
$F_4(x) \leftarrow Has\_Car(x, y), Wheels(y, 3)$
$F_9(x) \leftarrow Has\_Car(x, y), Load(y, triangle)$

Assume we are given a set of relevance labels $\mathcal{R} = \{r1, r2\}$, with $r1 \prec_r r2$. Let us further assume $\{(Wheels/1, r1), (Load/2, r1), (Short/1, r2), (Closed/1, r2)\}$ is our relevance-assignment. Then $Relev(F_2) = Relev(F_3) = [r2, r2]$, $Relev(F_4) = Relev(F_9) = [r1, r1]$.

Definition 19 Relevance of feature-clauses. *Let $\mathcal{F}$ be a set of features. Let C be a feature-clause. W.l.o.g. let the features in C be $\{F_1, F_2, \ldots, F_k\}$ where the $F_i \in \mathcal{F}$. Let $Relev(F_i) = [l_i, h_i]$. Then $Relev(C) = \{[l*, h*]\}$ where $l* = \min(l_1, l_2, \ldots, l_k)$ and $h* = \max(h_1, h_2, \ldots, h_k)$.*

Example 11 (Relevance of a feature-clause for the trains problem). *As before, let there be 4 relational features* F_2, F_3, F_4, *and* F_9, *with* $Relev(F_2) = Relev(F_3) = [r2, r2]$, $Relev(F_4) = Relev(F_9) = [r1, r1]$. *Then, the relevance of the feature-clause* $C : F(x) \leftarrow F_2(x),\ F_3(x), F_4(x), F_9(x)$ *is* $R_C = \{[r1, r2]\}$.

The relevance of explanations is constructed from the relevance of the feature-clauses in the explanation. We distinguish deliberately between unstructured and structured explanations.

Definition 20 Relevance of an explanation *Given background knowledge* B, *let* H *be an explanation for* $Class(a, c)$ *containing the clause* $C : \forall x(Class(x, c) \leftarrow Body)$. *If* H *is an unstructured explanation then* $Relev(H) = Relev(C)$. *Otherwise, if* H *is a structured explanation, then* $Relev(H) = \bigcup_{F_i \in Body} Relev(UFC(F_i))$.

From Defn. 20, the relevance of an explanation is a set of relevance intervals, and from Remark 7, it follows that set of explanations can be quasi-ordered, using the $\preceq_s$ relation over the relevance of explanations.

Example 12 (Relevance of explanations for the trains problem). *Suppose, as before, we have 4 relational features* F_2, F_3, F_4, *and* F_9, *with* $Relev(F_2) = Relev(F_3) = [r2, r2]$, $Relev(F_4) = Relev(F_9) = [r1, r1]$. *Suppose we have an unstructured explanation* $H : Class(x, East) \leftarrow F_2(x),\ F_3(x), F_4(x), F_9(x)$. *Then, from Example 11,* $Relev_H = R_H = \{[r1, r2]\}$.

On the other hand, suppose we have the structured explanation H_1*:*

$$\begin{aligned}
&Class(x, East) \leftarrow F_{1,1}(x), F_{1,2}(x)\\
&F_{1,1}(x) \leftarrow F_2(x), F_3(x)\\
&F_{1,2}(x) \leftarrow F_4(x), F_9(x)
\end{aligned}$$

$F_{1,1}, F_{1,2} \notin \mathcal{F}$, *and are therefore invented features. Let* $\mathcal{F}_2 = \{F_{1,1}, F_{1,2}\}$. *Then,* $Relev(H_1) = R_{H_1} = \bigcup_{F \in \mathcal{F}_2} Relev(UFC(F)) \cup \emptyset$. *Now* $Relev(UFC(F_{1,1})) = \{[r2, r2]\}$, $Relev(UFC(F_{1,2})) = \{[r1, r1]\}$ *and* $R_{H_1} = \{[r1, r1], [r2, r2]\}$.

It is interesting that although a pair of structured explanations may unfold to the same unstructured explanation (and therefore have the same fidelity), their relevance may not be the same. Intuitively, structuring any unstructured explanation will split the relevance-interval of the corresponding feature-clause into a set of intervals (see Defn. 20). In a "good structuring" each interval in this set will be narrower than the unstructured relevance-interval and will therefore be preferred under the relevance ordering (Defn. 17).

Remark 8 Structuring can increase relevance. *Let* $H = \{C\}$ *be an unstructured explanation for* $Class(a, c)$, *and let* H' *be a structured explanation containing a clause* $C' : Class(x, c) \leftarrow Body$ *that unfolds to* C. *Let* $P_H = Relev(H)$ *and* $P_{H'} = Relev(H')$. *Then* $P_H \preceq_s P_{H'}$.
Let C *contain the features* $\{F_1, \ldots, F_l\}$, *where* $Relev(F_i) = [l_i, h_i]$ *Since* H *is an unstructured explanation,* $Relev(H) = \{[l*, h*]\}$, *where* $l* = \min(l_1, \ldots, l_k)$ *and* $h* = \max(h_1, \ldots, h_k)$.

Let C' contain the invented features $\{F'_1, \ldots, F'_j\}$. Since C' unfolds to C, each F'_i unfolds to a clause containing some subset S'_i of $\{F_1, \ldots, F_k\}$, and $\bigcup_{i=1}^{j} S'_i = \{F_1, \ldots, F_k\}$. W.l.o.g. let $h = h_k$. By the constraint imposed on structured explanations, there must be at least one invented feature F'_m that unfolds to a clause containing F_k. Let $Relev(UFC(F'_m)) = [l'_m, h'_m]$. Clearly, $l* \preceq l'_m$ and $h* = h'_m$, and therefore $[l*, h*] \preceq [l'_m, h'_m]$. Since $[l'_m, h'_m] \in Relev(H')$, it follows from Defn. 17 that $Relev(H) \preceq_s Relev(H')$.*

Example 13 Comparing the relevance of explanations in the trains problem. *As before, suppose we have 4 relational features F_2, F_3, F_4, and F_9, with $Relev(F_2) = Relev(F_3) = [r2, r2]$, $Relev(F_4) = Relev(F_9) = [r1, r1]$. The unstructured explanation H: $Class(x, East) \leftarrow F_2(x), F_3(x), F_4(x), F_9(x)$ has relevance $R_H = \{[r1, r2]\}$.*

The following structured explanation H_1 unfolds to H:

$Class(x, East) \leftarrow F_{1,1}(x), F_{1,2}(x)$
$F_{1,1}(x) \leftarrow F_2(x), F_3(x)$
$F_{1,2}(x) \leftarrow F_4(x), F_9(x)$

As we saw in Example 12, $R_{H_1} = \{[r1, r1], [r2, r2]\}$. From Defn. 17, $R_H \preceq_s R_{H_1}$.

But structuring is not guaranteed to increase relevance. The following structured explanation H_2 also unfolds to H:

$Class(x, East) \leftarrow F_{1,3}(x), F_{1,4}$
$F_{1,2}(x) \leftarrow F_3(x), F_4(x)$
$F_{1,4}(x) \leftarrow F_2(x), F_9(x)$

Now $R_{H_2} = \{[r1, r2]\}$, and $R_H \not\preceq_s R_{H_2}$. Thus, although $H_{1,2}$ both unfold to H, a selection criterion that takes relevance into account would prefer H_1 over H_2 and H.

4.2. Other Relevance-Based Priors

Broadly speaking, priors can be categorised as theoretical (derived from domain knowledge), or empirical (derived from data). A relevance ordering over predicates represents just one form of a theoretical prior. Finer-grained theoretical priors based on relevance are possible. For example, some specific combination of predicates may be especially important (say, the presence of a lactone ring, connected to a 7-membered ring).

The machinery of comparing labelled explanations can be used without change if theoretically important features can be encoded as background predicates.

Example 14 (A theoretically important feature for the trains problem). *Let as assume we are given a set of relevance labels $\mathcal{R} = \{r1, r2, r*\}$, with $r1 \prec_r r2$ and $r2 \prec_r r*$. Assume as before, the following relevance-assignment: $\{(Wheels/1, r1), (Load/2, r1), (Short/1, r2), (Closed/1, r2)\}$. Suppose we know that the existence of short, closed cars is especially important. This can be encoded by the predicate in the background knowledge:*

$Short\_Closed(x) \leftarrow Has\_Car(x, y), Short(y),\ Closed(y)$

Augumenting the relevance-assignment with $(Short\_Closed/1, r*)$, we are now able to distinguish between the features:

$F_1(x) \leftarrow Has\_Car(x, y), Short(y)$
$F_3(x) \leftarrow Has\_Car(x, y), Closed(y)$
$F_{10}(x) \leftarrow Short\_Closed(x)$

Now, $Relev(F_1) = Relev(F_3) = [r2, r2]$, but $Relev(F_{10}) = [r*, r*]$. We would expect any explanation that uses F_{10} to be prefered over one that does not.

It may be possible to estimate empirically the relevance of features using data (for example, a feature that occurs frequently in explanations is highly relevant). Incorporating this kind of information will require changes to the definition calculating the relevance of features. The comparison of labelled expressions then proceeds as before: we do not pursue the use of empirical estimates of relevance in this paper.

4.3. Implementation

We finally have the pieces to define a label for an explanation, given data D and background B. Each explanation H will now have the label $\langle L_H, P_H \rangle$, where $L_H = Fidelity(H|D, B)$ and $P_H = Relev(H)$. Using a dictionary-ordering to compare labelled explanations allows us to decompose the task of identifying explanations into two parts: the first that maximises fidelity and the second that maximises the relevance. Further, as we have already seen (Remark 4), structuring cannot increase fidelity, but can increase relevance (Remark 8). Therefore, with a dictionary ordering on labels, it suffices to search first over the space of unstructured explanations, and then over the space of structured explanations that unfold to the unstructured explanations with maximal fidelity. Algorithm 3 extends the previous procedure of finding any unstructured explanation (Algorithm 1) to obtain a maximal-fidelity unstructured explanation.

Algorithm 3: Identifying an unstructured explanation with maximal fidelity. Here, $Clause(e, F)$ denotes the clause $\forall x(Class(x, c) \leftarrow Body)$, where $Body$ is a conjunction of literals $F_i(x)$ with $F_i \in F$. In practice, we will need to extend this procedure to return all unstructured explanations with maximal fidelity.

```
1  Algorithm ConstructUnstructOpt($e, B, \mathcal{F}, E^+, E^-$)
       Input: A relational example $e$; background knowledge $B$; a set of features $\mathcal{F}$ with
              definitions in $B$; and $E^+, E^-$ as defined in Defn. 12.
       Output: A maximal fidelity unstructured explanation $H$ s.t. $B \cup H \models e$.
2      Let $e = Class(a, c)$.
3      Let $a' = FV(a)$.
4      Let $\mathcal{F}'$ be the set of features that map to $TRUE$ in $a'$.
5      Let $D = (E^+, E^-)$.
6      Let $\mathcal{H}$ be the subset-lattice of $\mathcal{F}'$.
7      Let $\mathcal{F}''$ be any element in $\mathcal{H}$ s.t.
8          $Body$ is the conjunction of features in $\mathcal{F}''$,
9          $C = \forall x(Class(x, c) \leftarrow Body)$,
10         $L = Fidelity(\{C\}|D, B)$, and
11         There is no other element $\mathcal{G}$ in $\mathcal{H}$ s.t. $Fidelity(\{Clause(e, \mathcal{G})\}|D, B) > L$.
12     return $\{C\}$
```

It is insufficient to simply get one maximal-fidelity explanation, and we will need to modify the procedure in Algorithm 3 return the set of unstructured explanations with maximal fidelity. For each element H in this set, Algorithm 4 extends *ConstructStruct* in Algorithm 2 to return a structured explanation with higher-relevance than H, if one exists. It is not hard to see that if $P_k = \emptyset$ then $P_{k+1} = \emptyset$. Therefore, it is only needed to call *ConstructExpl* with $k = 2, 3, \ldots$ until $P_k = \emptyset$. In experiments in this paper, we will adopt the even simpler strategy of only considering $k = 2$. That is, we will only consider 2-partitions of the set of features constituting the unstructured explanation H (in effect, seeking structured explanations with higher relevance than H, but using the minimum number of invented features). The structured explanations in Example 5 are examples of structures that can be obtained with $k = 2$.

Algorithm 4: A procedure for obtaining a structured explanation that is at least as relevant as an unstructured explanation H.

```
1  Algorithm ConstructExpl($H, B, k$)
       Input: An unstructured explanation $H$; background knowledge $B$; and a number $k$ ($\geq 2$).
       Output: An explanation $H'$ s.t. $Relev(H|B) \preceq Relev(H'|B)$.
2      Let $H = \forall x(Class(x, c) \leftarrow Body)$.
3      Let $Relev(H) = R_H = \{[\alpha, \gamma]\}$.
4      Let $F$ be the set of features in $Body$.
5      Let $P_k$ be a set s.t. one of the following is true:
6          Either (a) $P_k$ is a $k$-partition of $F$ that satisfies the following: at least one block in $P_k$
             contains 2 or more elements, and there is at least one block in $P_k$ whose elements
             have a minimum relevance $\beta$ and maximum relevance $\gamma$ such that $\alpha \prec_r \beta \preceq_r \gamma$;
7          Or (b) $P_k = \emptyset$ (if no such $k$-partition exists).
8      if $P_k = \emptyset$ then
9          return $H$
10     end
11     else
12         Let $P_k$ consist of blocks $b_1, b_2, \ldots, b_k$
13         for each block $b_i \in P_k$ do
14             Construct a new feature $F'_i$ with definition $C_i = \forall x(F'_i(x) \leftarrow Body_i)$, where
                 $Body_i$ is the conjunction of the features in $b_i$.
15         end
16         Let $C_0 = \forall x(Class(x, c) \leftarrow Body_0)$ where $Body_0$ is the conjunction of the
             $F'_1, F'_2, \ldots, F'_k$.
17         Let $H' = \bigcup_{i=0}^{k} C_k$.
18     end
19     return $\{H'\}$
```

The structured explanation is obtained by inventing k features, the definition of at least one of which has a higher relevance than the unstructured explanation. Any process for obtaining such a k-partition is acceptable. With $k = 2$, we use a simple greedy strategy to identify a high-relevance block (one of the invented features). We start by selecting the feature with the highest relevance in the unstructured explanation. We continue to add

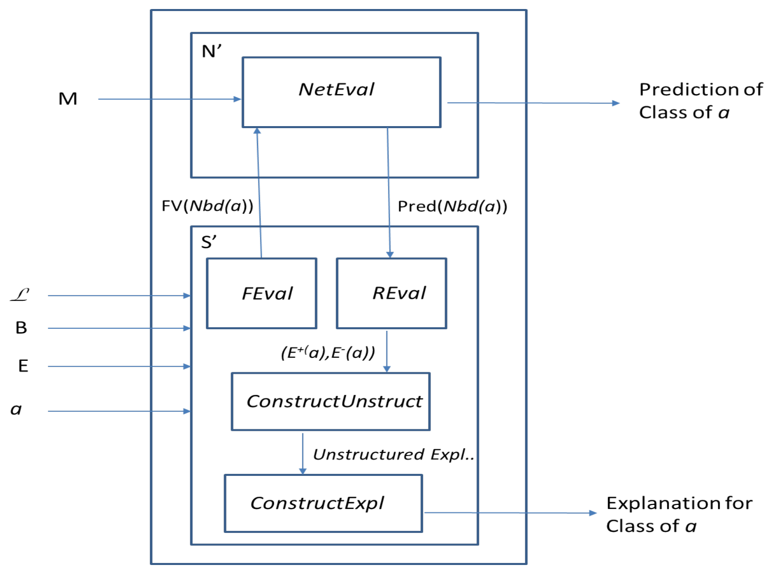


Figure 4: More details on the testing component in Fig. 1. E is the set of relational training instances, a is a test instance, and M is a DRM. FV($Nbd(a)$) denotes the feature-vectors for relational in E that are within the neighbourhood of a. Pred($Nbd(a)$) are the class predictions of instances in $Nbd(a)$ by the DRM. $NetEval$ evaluates the model M on input data; $FEval$ evaluates relational instances and returns their feature-vectors; and $REval$ constructs relational examples. We assume that $FEval$ has access to the feature-definitions found in the training stage.

features as long as the relevance of the corresponding invented feature is higher than that of the unstructured explanation. The second (low-relevance) block then contains all features from the unstructured explanation that are not in the high-relevance block.

Together, $ConstructUnstruct$ and $ConstructExpl$ are used to identify a local explanation for a relational instance e (see Fig. 4).

5. Empirical Evaluation

In this section, we investigate the following:

Explanation. We conduct the following experiments:

Expt. 1: Fidelity. Can we construct a local symbolic explanations for an instance with high fidelity to local predictions made by the DRM?

Expt. 2: Relevance. Does incorporating a prior preference based on relevance have any effect?

Some clarifications are necessary here: (a) By a local symbolic explanation in Expt. 1 we mean the use of a graph-search that returns the unstructured explanation with the highest fidelity, described in Section 4.3; and (b) By prior-preference in Expt. 2, we mean the relevance-based ordering over structured or unstructured explanations. In Expt. 2, we confine ourselves to whether the use of the preference can change the explanation returned (either from a unstructured to a structured one, or from one unstructured explanation to another). We note that incorporation of a prior preference obtained from a human expert is still not sufficient to ensure comprehensibility of explanations by the expert. Evidence for this requires results in the form of cross-comparisons on the use of prior expert preference on explanations against expert comprehensibility of explanations. However, this is outside the scope of this paper.

It is useful, though not mandatory for the experiments here, that the models have good predictive performance. To this end, we report in an Appendix for the following:

Prediction. We conduct the following experiment:

> **Expt. 3: Accuracy.** Does the DRM constructed using randomly drawn features have good predictive performance?

By randomly drawn features in Expt. 3, we mean the rejection-sampling method described in Vig et al. (2018).

5.1. Materials

5.1.1. Data

We report results from experiments conducted using 7 well-studied real world problems from the ILP literature. These are: Mutagenesis (King et al., 1996a); Carcinogenesis (King and Srinivasan, 1996a); DssTox (Muggleton et al., 2008); and 4 datasets arising from the comparison of Alzheimer's drugs denoted here as *Amine*, *Choline*, *Scop* and *Toxic* (Srinivasan et al., 1996). Each of these have been shown to benefit from the use of a first-order representation and domain-knowledge, but there is still room for improvement in predictive accuracy. While good predictive performance is necessary, the principal motivation for the selection of the problems here is that for each dataset we also have access to domain-information about the relevance of predicates for the classification task considered.

Of these datasets, the first three (Mut188–DssTox) are predominantly relational in nature, with data in the form of the 2-d structure of the molecules (the atom and bond structure), which are diverse and can be of varying sizes. Some additional bulk properties of entire molecules obtained or estimated from this structure are also available. The Alzheimer datasets (Amine–Toxic) are best thought of as being quasi-relational. The molecules have a fixed template, but vary in number and kinds of substitutions made for positions on the template. A first-order representation has still been found to be useful, since it allows expressing concepts about the existence of one or more substitutions and their properties. The datasets range in size from a few hundred (relational) instances to a few thousands. This is extremely modest by the usual data requirements for deep learning. We refer the reader to the references cited for details of the domain-knowledge used for each problem.

5.1.2. Background Knowledge

For the relational datasets (Mut188–DssTox), background knowledge is in the form of general chemical knowledge of ring-structures and some functional groups. Background-knowledge contains definitions used for concepts like: alcohols, aldehydes, halides, amides, amines, acids. esters, ethers, imines, ketones, nitro groups, hydrogen donors and acceptors, hydrophobic groups, positive- and negatively-charged groups, aromatic rings and non-aromatic rings, hetero-rings, 5- and 6-carbon rings and so on. These have been used in structure-activity applications of ILP before (King et al., 1996b; King and Srinivasan, 1996b). However, we note that none of these definitions are specifically designed for the tasks here. In addition, for Mut188 and Canc330, there are some bulk properties of the molecules that are available. For the Alzheimer problems (Amine–Toxic) domain knowledge consists of properties of the substituents in terms of some standard chemical measures like size, polarity, number of hydrogen donors and acceptors and so on. Predicates are also available to compare these values across substitutions. Again we refer the reader to the relevant ILP literature for more details.

In addition to the domain-predicates just described, we will also have information in the form of a relevance ordering as described in Srinivasan et al. (2003). That paper only refers to the Mut188 and Canc330 datasets. The same information is obtained from the domain-expert involved in that paper for the other problems in this paper. A complete description of the relevance assignment of predicates for each problem is in Appendix C.

5.1.3. Algorithms and Machines

Random features were constructed on an Intel Core i7 laptop computer, using VMware virtual machine running Fedora 13, with an allocation of 2GB for the virtual machine. The Prolog compiler used was Yap. Feature-construction uses the utilities provided by the Aleph ILP system (Srinivasan, 1999) for constructing most-specific clauses in a depth-bounded mode language, and for drawing clauses subsuming such most-specific clauses. No use is made of any of the search procedures within Aleph. The deep networks were constructed using the Keras library with Theano as the backend, and were trained using an NVIDIA K-40 GPU card.

5.2. Methods

The methods used for each of the experiments are straightforward

Explanation. Experiments 1 and 2 are concerned solely with the explanatory performance of the local symbolic models.

For each dataset:

1. Construct a DRM using training data
2. For each test instance, obtain the local symbolic unstructured explanation(s) with the highest fidelity;
3. Estimate the overall fidelity of the symbolic explanations (Expt. 1)
4. Estimate the effect of using the relevance-based prior in hypothesis selection (Expt. 2)

Prediction. Experiment 3 (in the Appendix) is concerned solely with the predictive performance of the DRM on the datasets in the paper.[10]

For each dataset:

1. Obtain a set of random features $\mathcal{F}$;
2. Compute the Boolean-value for each $F \in \mathcal{F}$ for the data;
3. Construct a DRM N using training data and obtain its predictions on test instances;
4. Estimate the overall predictive performance of M (Expt. 1)

Some clarifications are necessary at this point:

- We use a straightforward Deep Neural Network (DNN) architecture. There are multiple, fully connected feedforward layers of rectified linear (ReLU) units followed by Dropout for regularization (see Goodfellow et al. (2016) for a description of these ideas). The model weights were initialized with a Gaussian distribution. The number of layers, number of units for each layer, the optimizers, and other training hyperparameters such as learning rate, were determined via a validation set, which is part of the training data. Since the data is limited for the datasets under consideration, after obtaining the model which yields the best validation score, the chosen model is then retrained on the complete training set (this includes the validation set) until the training loss exceeds the training loss obtained for the chosen model during validation.
- We use the Subtle algorithm (Blockeel and Valevich, 2016) to perform the subsumption-equivalence test used to determine redundant features.
- For all the datasets, 10-fold cross-validated estimates of the predictive performance using ILP methods are available in the ILP literature for comparison. We use the same approach. This requires constructing DRMs separately for each of the cross-validation training sets, and testing them on the corresponding test sets to obtain estimates of the predictive accuracy;
- We use the mode-language and depth constraints for the datasets that have been used previously in the ILP literature. For the construction of features, the rejection-sampler performs at most $10,000$ draws;
- We take *explanatory fidelity* to mean the probability that the prediction made by H on a randomly drawn instance agrees with the prediction made by the corresponding DRM. We use the same 10-fold cross-validation strategy for estimating this probability (for efficiency, we use the same splits as those used to estimate predictive accuracy).

10. A note of clarification is in order here. The purpose of the Experiment 3 is to show that a DRM can achieve predictive accuracies comparable to or better than the best reports in the literature for this data. This is necessary to support the main experiments on assessment of the use of relevance information, i.e., to control for the possibility of deficiencies with the DRM being used. We direct the reader elsewhere to an assessment of DRMs on large numbers of datasets, and against state-of-the-art prediction tools (Dash et al., 2018).

For a given train-test split Tr_i and Te_i, we proceed as follows. We obtain a DRM N_i using Tr_i. We start with a fidelity count of 0. For each instance x'_j in Te_i we obtain the class predicted by N_i for x'_j, and corresponding neighbourhood of x'_j in the training set Tr_i. This strategy ensures that all explanations are based only on data the model was actually trained on, an important consideration in the biological applications our datasets come from. The neighbourhood is partitioned into $\delta^+(x'_j)$ and $\delta^-(x'_j)$ using the predictions by N_i and high-fidelity unstructured explanation(s) H_{ij} are obtained. This is done using a beam-search over the lattice described in Algorithms 1 and 3. The size of the beam is 5 (that is, the top 5 unstructured explanations are returned).

- Assessments of fidelity require the definition of a neigbourhood. For each instance x'_j in Te_i we say any instance $x \in Tr_i$ is in the neighbourhood of x' iff $FV(x')$ and $FV(x)$ differ in no more than k features. This is just the Hamming distance between the pair of Boolean vectors. That is, the neighbourhood of a test instance x' consists of training instances x's whose feature-vector representation are within a k-bit Hamming distance of x'. We will consider $k = 5$ and $k = 10$ in the experiments.

- In all cases, estimates are obtained using the same 10-fold cross-validation splits reported in the ILP literature for the datasets used. This will allow a cross-comparison of the results from Expt.1 to those reports.

5.3. Results

Results of the empirical evaluation are tabulated in Fig. 5. Results on the predictive accuracy of the DRM are in Fig. 10 in Appendix D, with appropriate discussion. Some supplementary results are in Fig. 7. The principal observations that can be made from the main tabulations in Fig. 5 are these: (1) High-fidelity symbolic explanations can be obtained for local predictions made by the DRM; and (2) In 6 of the 7 problems, introducing a prior-preference based on relevance does affect the selection of explanations.
Together, the results provide evidence for the following:

(a) It is possible to extract symbolic explanations for the prediction made by the DRM for a randomly drawn (new) instance. The explanations are largely consistent with the predictions of the network for that instance and its near-neighbours in the training data examined before by the network; and

(b) It is possible to incorporate domain-knowledge in the form of expert assessment of relevance of background predicates into a preference ordering for selecting amongst explanations.

Note that if the inclusion of relevance does not make a difference to selecting an explanation, we would expect values of 0.0 in Fig. 5(b). It is evident that the observed values are clearly not 0, for all cases except $DssTox$. This exception is unsurprising, since all predicates used for this problem have the same relevance (see Appendix C).
Before we examine some critical factors affecting the selection of local explanations, we re-emphasise three important characteristics of explanations:

Problem	Fidelity
$Mut18$	0.99(0.01)
$Canc330$	0.99(0.01)
$DssTox$	0.87(0.03)
$Amine$	0.98(0.01)
$Choline$	0.89(0.01)
$Scop$	0.89(0.02)
$Toxic$	0.94(0.02)

(a)

Problem	$\langle L_H, R_H \rangle > \langle L_H, \emptyset \rangle$
$Mut188$	0.82(0.08)
$Canc330$	0.77(0.07)
$DssTox$	0.00(0.00)
$Amine$	0.38(0.15)
$Choline$	0.27(0.03)
$Scop$	0.27(0.06)
$Toxic$	0.24(0.14)

(b)

Figure 5: Experiments 1 and 2. (a) Mean fidelity of the explanatory (symbolic) model to the predictive (neural) model. The number tabulated is a 10-fold cross-validation estimate of the faithfullness of the symbolic model to a DRM's prediction assessed over the neighbourhood of a test instance. The entries are for the smallest neighbourhood (H_5: see the "Methods" section for how this is computed). The number in parentheses are estimates of standard deviations. (b) Relative frequency estimates of how often we can expect incorporation of a relevance-based prior to affect the selection of explanations. The tabulation is the proportion of explanations for which a Bayes label using both fidelity and relevance ($\langle L_H, R_H \rangle$) is better than one that uses fidelity only ($\langle L_H, \emptyset \rangle$). The explanations are for the H_5 neighbourhood. Again, the estimates are from the same 10-fold cross-validation splits used elsewhere.

	Network		
Problem	$\approx$ #Inputs	#HL	#Units/HL
$Mut188$	7100	3	15
$Canc330$	5400	3	6
$DssTox$	200	4	6
$Amine$	2200	3	12
$Choline$	2500	3	6
$Scop$	2400	4	13
$Toxic$	2300	3	14

Figure 6: Characteristics of DRM models. The numbers are averages across a 10-fold cross-validation. For any fold, the number of input layers and the units in a layer are automatically obtained in an optimisation step.

1. Explanations are constructed to maintain fidelity to the DRM's predictions in a local region, and not to the actual class labels of the instances in the region. Fidelity is agnostic therefore to the predictive accuracy of the DRM, and it is possible to obtain high-fidelity explanations for a low predictive accuracy DRM. This occurs in the results here on the *Canc*330 dataset, which is known to be more noisy than the others. The DRM has low predictive accuracy on this dataset, but nevertheless we are able to construct high-fidelity explanations for it.

2. Local explanations are not constructed for the model's prediction on the entire instance space, and focus instead on just a few (similar) instances. Although the networks can often contain 1000's of input features (see Fig. 6), any single instance usually consists of only a few "active" features. This allows the construction of compact local explanations that only require the presence of a small number of features (more on compactness below).

3. Since (local) fidelity is measured against the DRM's predictions of instances in a region and not against their actual class labels, any subset of instances that are close to the test instance can be chosen. In our experiments, this region is selected from the training set. This ensures that all explanations are based only on data the DRM was actually trained on. An alternative could be by random generation, as is done in Ribeiro et al. (2016), although this may pose some difficulty in ensuring instances are drawn from an appropriate instance space (for biochemical tasks like the ones here, for example, they have to be real molecules).

Unsurprisingly, two key factors affecting the selection of local explanations are the neighbourhood size, and the relevance information provided. We consider each of these in turn.

5.3.1. The Role of the Neighbourhood

Neighbourhood size affects the local explanations constructed (see Fig. 7). In general, the fewer the instances in the local neighbourhood, the less the constraints imposed on the explanations. This leads to smaller explanations (fewer literals), and higher fidelity.

The high-fidelity values in Fig. 7 are not an artifact of instances being very similar within a neighbourhood (that is, class distribution will be predominantly of the same value). Figure 8 shows this is not the case. Although class-ratios are less skewed as the size of the neighbourhood increases (from H_5 to H_{10}), it is not the case that high-fidelity can simply be obtained by default. We believe this is because a 1-bit difference here can mean the presence or absence of a non-trivial relational structure. A change of 5 such structural features can therefore be a very substantial change.

Finally, structured explanations only appear to play role for larger neighbourhoods. We suggest this is not so much to do with the size of the neighbourhood, as to the corresponding increasing in the size of the explanations. In general, larger explanations (those with more features) are likely to benefit from structuring.

5.3.2. The Role of Relevance

Broadly speaking, the relevance-ordering will prefer labelled explanations with higher relevance. The tabulation in Fig. 5(b) tells us the problems for which we can expect this

Problem	Nbd. Size	
	H_5	H_{10}
*Mut*188	5(1)	20(3)
*Canc*330	3(1)	27(4)
DssTox	19(3)	83(10)
Amine	9(1)	23(20
Choline	14(1)	53(3)
Scop	8(1)	26(2)
Toxic	12(2)	42(2)

(a)

Problem	Fidelity	
	H_5	H_{10}
*Mut*188	0.99(0.01)	0.96(0.01)
*Canc*330	0.99(0.01)	0.92(0.02)
DssTox	0.87(0.03)	0.85(0.02)
Amine	0.98(0.01)	0.97(0.01)
Choline	0.89(0.01)	0.83(0.01)
Scop	0.89(0.02)	0.86(0.02)
Toxic	0.94(0.02)	0.90(0.03)

(b)

Problem	Expl. Size	
	H_5	H_{10}
*Mut*188	1(1)	2(1)
*Canc*330	1(1)	3(1)
DssTox	3(1)	4(1)
Amine	2(1)	2(1)
Choline	2(1)	3(1)
Scop	2(1)	2(1)
Toxic	2(1)	3(1)

(c)

Problem	Struc. Expl.	
	H_5	H_{10}
*Mut*188	0.00	0.04
*Canc*330	0.02	0.36
DssTox	0.00	0.00
Amine	0.00	0.00
Choline	0.00	0 00
Scop	0.00	0.00
Toxic	0.00	0.00

(d)

Figure 7: Effect of the neighbourhood. (a) Average numbers of neighbouring instances for local explanations (rounded up); (b) Average fidelity of local explanations; (c) Average number of literals in the maximal-fidelity explanations (rounded up); and (d) Average proportion of structured explanations. All numbers are estimates obtained from 10-fold cross-validation (the numbers in parentheses are estimates of the standard deviation).

Problem	Pos-to-Neg Ratio	
	H_5	H_{10}
*Mut*188	0.77	0.61
*Canc*330	0.63	0.46
DssTox	0.21	0.13
Amine	0.58	0.50
Choline	0.58	0.49
Scop	0.54	0.48
Toxic	0.43	0.33

Figure 8: Average class ratios in neighbourhoods.

information to make a difference, in terms of the proportion of explanations for which we can expect relevance to affect selection.

Since the relevance information provided is an ordering, it is possible, but perhaps unnecessary, to correlate numerically the differences observed in Fig. 5(b) against the ranking. Nevertheless, the specific result in the tabulation for DssTox points to an important issue about the relevance ranking. There are clear differences between the role of the relevance information amongst the datasets (significant effect in Mut188 and Canc330; minor effect in the Alzheimer datasets; and no effect in DssTox). We conjecture the following condition for relevance to have an effect on selection:

> *The larger the range of the relevance assignment, the more likely it is that relevance will play a role in selection of explanations.*

For the datasets here, the range of the relevance assignment has 1 value for DssTox; 2 values for the Alzheimer's datasets; and 4 values each for Mut188 and Canc330. The corresponding proportions of explanations where relevance plays a role in selection are: 0.0 (DssTox); 0.29 (Alzheimer datasets); and 0.80 (Mut188 and Canc330).

Finally, since explanations are generated "on-demand", it is impractical to show the explanations for all test instances. A snapshot is nevertheless useful, and one such is shown in Fig. 9. The example is from the $Canc330$ problem. The unstructured explanation shown has perfect fidelity. The alternate structured explanation shown in Fig. 9 is derived from this unstructured explanation has the same fidelity (as expected), but higher relevance. In effect, what the structuring achieves here is to group predicates with the $Has\_property$ relation, which has high-relevance; and separate out predicates that mix predicates with low- and high-relevance. Some measure of the explanatory convenience provided by the symbolic model is apparent if the reader keeps in mind that the corresponding prediction by the DRM is based on around 2000 input features, about 400 of which are equal to 1 for the test instance. It is interesting that the domain-expert could correctly identify the class of the example when shown the symbolic explanation.

6. Other Related Work

We have already noted the key reference to LIME and to reports in the ILP literature of immediate relevance to the work in this paper. Here we comment on other related work. The landmark work on structured induction of symbolic models is that of Shapiro (1987). In Shapiro's work structuring was top-down, with machine learning being used to learn sub-concepts identified by a domain-expert. The structuring is therefore hand-crafted, and with a sufficiently well-developed tool, a domain-expert can, in principle, invoke a machine learning procedure to construct sub-concepts using examples he or she provides. The technique was shown to yield more compact models than an unstructured approach on two large-scale chess problems, using decision-trees induced for sub-concepts.

Clearly the principal difficulty in the Shapiro-style of structured induction is the requirement for human intervention at the structuring step. The following notable efforts in ILP, or closely related areas, have been directed at learning structured theories automatically:

Unstructured explanation:

Label: $\langle L_H = 1.0, P_H = \{[1,4]\}\rangle$
Explanation H:
$Class(x,c) \leftarrow F_{537}(x), F_{1196}(x), F_{610}(x), F_{611}(x), F_{1657}(x)$

Structured explanation(s):

Label: $\langle\ L_{H_1} = 1.0, P_{H_1} = \{[1,4],[4,4]\}\rangle$
Explanation H_1:
$Class(x,c) \leftarrow F_{1,1}(x), F_{1,2}(x)$
$F_{1,1}(x) \leftarrow F_{1657}(x)$
$F_{1,2} \leftarrow F_{537}(x), F_{1196}(x), F_{610}(x), F_{611}(x)$

Feature-definitions:
$F_{537}(x) \leftarrow Atm(x,y,h,3,z), Gteq(z,0.115)$ (Relev = [1,1])
$F_{1196}(x) \leftarrow Atm(x,y,c,22,z), Gteq(z,-0.111), Atm(x,w,c,22,z)$ (Relev = [1,1])
$F_{610}(x) \leftarrow Non\_ar\_hetero\_6\_ring(x,u), Has\_property(x,ames,p)$ (Relev = [2,4])
$F_{611}(x) \leftarrow Has\_property(x,salmonella,n),$
$Has\_property(x,mouse\_lymph,p)$ (Relev = [4,4])
$F_{1657}(x) \leftarrow Has\_property(x,cytogen\_ca,n), Has\_property(x,mouse\_lymph,p),$
$Has\_property(x,cytogen\_sce,p)$ (Relev = [4,4])

Figure 9: Example explanations for a test instance from the $Canc330$ domain. The unstructured explanation H has perfect fidelity to the DRM's local predictions. The structured explanation H_1 is obtained from H and preserves fidelity, but predicates are grouped so that higher relevance is achieved.

- Inverse resolution, especially in the Duce system (Muggleton, 1987) was explicitly aimed at learning structured sets of rules in propositional logic. The sub-concepts are constructed bottom-up;
- Function decomposition, using HINT (Zupan et al., 2001), which learns hierarchies of concepts using automatic top-down function decomposition of propositional concepts;
- First-order theories with exceptions, using the GCWS approach (Bain, 1991), which automatically constructs hiearchical concepts. Structuring is restricted to learning exceptions to concepts learned at a higher level of the hierarchy;
- First-order tree-learning: an example is the TILDE system (Blockeel, 1999). In this, the tree-structure automatically imposes a structuring on the models. In addition, if each node in the tree is allowed a "lookahead"option, then nodes can contain conjunctions of first-order literals, each of which can be seen as defining a new feature. The model is thus a hierarchy of first-order features; and
- Meta-interpretive learning (Muggleton et al., 2015), which allows a very general form of predicate-invention, by allowing an abduction step when employing a meta-interpreter to use higher-order templates of rules that be used to construct proofs (in effect, explanations) for data. In principle, this would allow us not just to construct explanations on-demand, but also invent features on-demand. If the higher-order templates can be specialised to the domain, then it should be possible to control the feature-invention by relevance-information. Of course, this is unrelated to generating explanations for the predictions of a black-box classifier.
- Propositionalisation (relational feature construction) (Kramer, 2001) in principle enables the application of any propositional learner that can learn structured models, such as in (Zelezny and Lavrač̃, 2006), to the task of learning structured first-order models. More recently, propositionalization was applied in learning topic models, a class of two-level structured probabilistic models (Srinivasan et al., 2012). ILP propositionalization typically uses relational features constructed before learning commences (Srinivasan and King, 1996). For construction of relational explanations "on-demand", e.g., for local neighbourhoods of a test instance, there have been a number of approaches in ILP in which propositionalization occurs *during* the learning process. This appears to have been described first by (Alphonse and Rouveirol, 2000) where it was called "lazy propositionalization". Following this were the approaches of (Kramer, 2001), (Popescul and Ungar, 2004), and (Landwehr et al., 2006) in which this was termed "dynamic propositionalization".

Unsurprisingly, there has been a lot of research effort invested into extracting explanations for the predictions made by a neural network (see d'Avila Garcez et al. (2002), Chapter 3 for a full description of this line of research). Most of this effort has been in the direction of translating the network into a single logical model that guarantees correspondence to the neural model (for example, see Thrun, 1994). Alternatively, a single symbolic model can be learned that approximates the behaviour of the neural model over all inputs (for example, Craven and Shavlik, 1994; França et al., 2015)). Although distinct in their aims, both approaches still result in two separate models, one neural and the other symbolic. In both cases the symbolic model is intended to be a readable proxy for the neural model, which can then form a basis for an explanation of "why" questions. But there are some inherent trade-offs:

- If we try to replicate exactly the behaviour of the neural network with a symbolic model (as is done, say, in Thrun, 1994), then the resulting model may not be any more comprehensible than the neural network; and
- If we try only to approximate the behaviour of the network using logical predicates (as is done, say, in Craven and Shavlik, 1994), then we run the risk of not being able to replicate the network's behaviour sufficiently accurately over all instances, because of inadequacies of the logical predicates available.

Both these issues are exacerbated for modern-day deep networks, with many hidden layers and large numbers of inputs. One way to side-step these difficulties is simply to drop the requirement—as is done here—of translating the *entire* network model into a single symbolic model.

An entirely different, and much more sophisticated kind of hybrid model combining connectionist and logical components has been proposed recently in the form of Lifted Relational Neural Networks (LRNNs: Sourek et al., 2015)). In LRNNs, the logical component is used to provide a template for ground neural network models, which are used to learn weights on the logical formulae. While we have largely stayed within the confines of classical ILP both for obtaining features and explanations, LRNNs are closely related to probabilistic models for ILP. An area of common interest arises though in the use of the network structure to invent new features (although in the LRNN case, this is not for local models as we proposed here).

7. Concluding Remarks

The recent successes of deep neural networks on predictive tasks have not, to any large extent, used either domain knowledge or representations significantly more expressive than simple relations (usually over sequences). The price for this has been a requirement for very large amounts of data, which provide the network with sufficient correlations necessary to identify predictively useful models. This works for problems where large amounts of data are being routinely generated automatically, and about which there may be little or no domain knowledge. The situation with scientific data is quite the opposite: data are sparse, but there is significant domain-knowledge, often built up over decades of painstaking experimental work. We would like powerful predictive models in such domains, but for this, the network would need a way of capturing what is known already, and be able to explain its predictions, in a language that is sufficiently expressive.

In this paper the predictive models are knowledge-rich deep networks (DRMs), which results suggest can achieve, and often exceed, the levels of predictivity reached by full first-order learners. However, our interests extend beyond prediction. We want to construct understandable models. For this we start with the approach taken in Ribeiro et al. (2016) and propose the use of a proxy for the DRM that acts as a readable explanation. The proxy in this paper is in the form of a symbolic model for the predictions made by the DRM, constructed using techniques developed in Inductive Logic Programming (ILP). But there are at least three limitations we see arising from using the approach in Ribeiro et al. (2016). First, the goal is to generate a readable proxy for the *prediction* made by a black-box model. For us, the DRM is the black-box, but the approach here is sufficiently general to construct logical explanations for any predictor that uses relational features as input. We must emphasise that the logical explanation need not be the same as a readable proxy for the (true-)value of the instance. Second, readability does not guarantee comprehensibility: in Michie (1995) examples are shown of readable, but still incomprehensible, models. Thirdly, an important quality normally required of good explanations, causality, does not explicitly play a role. The first issue is inherent to the purpose of the model, and we have not attempted to change it here. The incorporation of a semantic prior based on relevance is a first attempt to address directly the second issue, and indirectly may address the third partially (non-causal explanations should have low relevance). To construct causal explanations correctly we will need more information than assigning relevance labels to predicates. We will also need the explanation-generator to pose counterfactual queries to the black-box, and the black-box to be able answer such queries with high accuracy. At this point, there is some evidence that symbolic learning could be adapted to suggest new experiments (see, for example, King et al., 2004), but it is not known how well DRMs will perform if the distribution of input values is very different to those that were used to train the network. So, at this point, we have restricted ourselves to constructing readable explanations for predictions that take into account prior preferences.

In this paper we have proposed the use of DRMs and the results here are consistent with previous work showing that such models are powerful predictors. However, it is important to understand also what the experimental evidence presented does not tell us. It does not tell us, for example, that a deep network without first-order features will not achieve the same performance as those tabulated here. However, it is not immediately apparent how this conjecture could be tested, since no more data are available for the problems. However it may be possible to transfer features constructed by a network trained on other problems with more data.

There are at least three separate directions in which we plan to extend the work here. First, interactions with the domain-expert suggests that the relevance information we have used here can be made much more fine-grained. It is possible, for example that certain combinations of predicates may be more relevant than others (or, importantly, certain combinations are definitely not relevant). None of this is accounted for in the current feature-generation process, and we intend to investigate relevance-guided sampling in place of the simple random sampling we use at present. Secondly, we would like to explore the construction of causal explanations for DRMs, by combining counterfactual reasoning with the use of a generative deep network capable of generating new instances. Thirdly, it is necessary at some point in the future, to establish the link between local symbolic explanations and human comprehensibility. For this, we would need to conduct a cross-comparison of structured and unstructured explanations and ratings of their comprehensibility by a domain-expert. Recently (Schmid et al., 2016) experiments have been reported in the ILP literature on assessing comprehensibility, when invention of predicates is allowed. Similar experiments will help assess the human-comprehensibility of local symbolic explanations for black-box classifiers.

Acknowledgments

A.S. is a Visiting Professorial Fellow, School of CSE, UNSW Sydney. A.S. is supported by the SERB grant EMR/2016/002766.

Appendix A. Some terminology from Logic Programming and ILP

In this section we cover only terminology used in the paper. For additional background and further terminology see Chang and Lee (1973); Nilsson (1980); Lloyd (1987); Muggleton and De Raedt (1994).

A language of first order logic programs has a vocabulary of constants, variables, function symbols, predicate symbols, logical implication '$\leftarrow$', and punctuation symbols. A function or predicate can have a number of arguments known as *terms*. Terms are defined recursively. A constant symbol (or simply "constant") is a term. A variable symbol (or simply "variable") is a term. If f is an m-ary function symbol, and $t_1, \ldots, t_m$ are terms, then the function $f(t_1, \ldots, t_m)$ is a term. A term is said to be *ground* if it contains no variables. If p is an n-ary predicate symbol, and $t_1, \ldots, t_n$ are terms, then the predicate $p(t_1, \ldots, t_n)$ is an atom. Predicates with the same predicate symbol but different arities are distinguished by the notation p/n where p is a predicate of arity n. We follow the standard Prolog syntax conventions. Constant symbols are written as a lower-case letter followed by a string of lower- or upper-case letters, digits or underscores ('_'). Variables are written similarly, except that the first letter must be upper-case. A literal is either an atom or the negation of an atom. If a literal is an atom it is referred to as a positive literal, otherwise it is a negative literal. A clause is a disjunction of the form $A_1 \vee \ldots \vee A_i \vee \neg A_{i+1} \vee \ldots \vee \neg A_k$, where each A_j is an atom. Alternatively, such a clause may be represented as an implication (or "rule") $A_1, \ldots, A_i \leftarrow A_{i+1}, \ldots, A_k$. A definite clause $A_1 \leftarrow A_2, \ldots, A_k$ has exactly one positive literal, called the *head* of the clause, with the literals $A_2, \ldots, A_k$ known as the *body* of the clause. A definite clause with a single positive literal is called a *unit* clause, and a clause with at most one positive literal is called a Horn clause. A Horn clause with no positive literal is called a *goal* clause. A set of Horn clauses is referred to as a logic program. It is often useful to represent a clause as a set of literals.

A *substitution* θ is a finite set $\{v_1/t_1, \ldots, v_n/t_n\}$ mapping a set of n distinct variables v_i, $1 \leq i \leq n$, to terms t_j, $1 \leq j \leq n$ such that no term is identical to any of the variables. A substitution containing only ground terms is a *ground* substitution. For substitution θ and clause C the expression $C\theta$ denotes the clause where every occurrence in C of a variable from θ is replaced by the corresponding term from θ. If θ is a ground substitution then $C\theta$ is called a ground clause. Since a clause is a set, for two clauses C, D, the set inclusion $C\theta \subseteq D$ is a partial order called *subsumption*, usually written C θ-subsumes D and denoted by $C \preceq D$. For a set of clauses S and the subsumption ordering $\preceq$, we have that for every pair of clauses $C, D \in S$, there is a least upper bound and greatest lower bound, called, respectively, the least general generalisation (lgg) and most general unifier (mgu) of C and

D, which are unique up to variable renaming. The subsumption partial ordering on clauses enables the definition of a lattice, called the *subsumption lattice*.

Appendix B. Fidelity and Likelihood

The model for noisy data in McCreath and Sharma (1998) can be adapted to the construction of local explanations that are not completely consistent with local predictions (that is, fidelity < 1).

Remark 9 (Bayesian Posterior McCreath and Sharma (1998)) *Given a relational instance $a \in \mathcal{X}$, let N be predictive model s.t. $N(FV(a)) = c$ for $c \in \mathcal{Y}$. Let sets E^+ and E^- denote the local neighbourhood of a and $D = (E^+, E^-)$. Let H be a local explanation for a (not necessarily consistent), using background knowledge B. Then, if $\theta(H)$ is the proportion of all instances in $\mathcal{X} \times \mathcal{Y}$ covered by H, and ϵ is an estimate of inconsistency (noise) allowed $\epsilon \in [0,1]$), the log posterior is given by McCreath and Sharma (1998):*

$$\log P(H|D,B) \ = \ \log P(D|H,B) + \log P(H|B) - \log(D|B)$$

where $P(H|B)$ denotes the prior probability, and:

$$\log P(D|H,B) \ = \ |TP(H)|\log\left(\frac{1-\epsilon}{\theta(H)} + \epsilon\right) + |TN(H)|\log\left(\frac{1-\epsilon}{1-\theta(H)} + \epsilon\right) + |FPN(H)|\log(\epsilon)$$

is the log-likelihood. Here $TP(H)$ is the set $\{e : e \in E^+ \text{ and } B \wedge H \models e\}$; $TN(H)$ is the set $\{e : e \in E^- \text{ and } B \wedge H \wedge \neg e \not\models \Box\}$; and $FPN(H) = D - (TP(H) \cup TN(H))$.

It is not hard to see that TP and TN correspond to $AgreePos$ and $AgreeNeg$ in Defn. 13. In some cases, it is in fact sufficient to maximise fidelity, to maximise the log-likelihood.

Remark 10 (Fidelity and Log-Likelihood) *Let $H_{1,2}$ be local explanations for a relational example $Class(a,c)$, given D, B. If $\theta(H_1) = \theta(H_2)$, $|TP(H_2)| \geq |TP(H_1)|$ and $|TN(H_2)| \geq |TN(H_1)|$ then: (a) $Fidelity(H_2|D,B) \geq Fidelity(H_1|D,B)$; and*
(b) $\log P(D|H_2,B) \geq \log P(D|H_1,B)$.

For given D, B, from Defn. 9 the log-likelihood is $\log\ P(D|H,B)$. Then, from Defn. 13, $Fidelity(H_1|D,B) = \frac{|TP(H_1)|+|TN(H_1)|}{|D|}$, and $Fidelity(H_2|D,B) = \frac{|TP(H_2)|+|TN(H_2)|}{|D|}$. Since $|TP(H_2) \geq |TP(H_1)|$, and $|TN(H_2)| \geq |TN(H_1)|$, trivially we obtain $Fidelity(H_2|D,B) \geq Fidelity(H_1|D,B)$.

For any explanation H, s.t. $0 < \theta)(H) < 1$ and for a fixed ϵ s.t. $0 \leq \epsilon \leq 1$, the log-multipliers $k_{1,2}(\theta(H))$ of $|TP(H)|$ and $|TN(H)|$ in the expression for $\log\ P(D|H,B)$ are both positive; and the log-multiplier k_3 of $|FPN(H)|$ is at most 0. Therefore $\log P(D|H)$ is $k_1(\theta(H))|TP| + k_2(\theta(H))|TN| - k_3(|D|) + k_3(|TP| + |TN|)$. If $\theta(H_1) = \theta(H_2)$, then $k_1(\theta(H_1)) = k_1(\theta(H_2)) = k_1$, say, and $k_2(\theta(H_1) = k_2(\theta(H_2)) = k_2$, say. Then $\log P(D|H_1)$ is $k_1|TP(H_1)| + k_2|TN(H_1)| - k_3(|D|) + k_3(|TP(H_1)| + |TN(H_1)|)$ and $\log P(D|H_2)$ is $k_1|TP(H_2)| + k_2|TN(H_2)| - k_3(|D|) + k_3(|TP(H_2)| + |TN(H_2)|)$, Since $|TP(H_2) \geq |TP(H_1)|$ and $|TN(H_2)| \geq |TN(H_1)|$, and $k_{1,2} > 0$ and $k_3 \geq 0$ it follows trivially that $\log P(D|H_2,B) \geq \log P(D|H_1,B)$.

Problem	Accuracy			
	OptILP	*Stat*	*DRM*	*DRM*
	Srinivasan and Ramakrishnan (2011)	Saha et al. (2012)	Lodhi (2013)	(here)
*Mut*188	0.88(0.02)	0.85(0.05)	0.90(0.06)	0.91(0.06)
*Canc*330	0.58(0.03)	0.60(0.02)	–	0.68(0.03)
DssTox	0.73(0.02)	0.72(0.01)	0.66(0.02)	0.70(06)
Amine	0.80(0.02)	0.81(0.00)	–	0.89(0.04)
Choline	0.77(0.01)	0.74(0.00)	–	0.81(0.03)
Scop	0.67(0.02)	0.72(0.02)	–	0.82(0.06)
Toxic	0.87(0.01)	0.84(0.01)	–	0.93(0.03)

Figure 10: Experiment 3. Estimated predictive accuracies of DRMs against some of the best reported performances in the ILP literature. All estimates are from the same 10-fold cross-validation splits in the reports cited.

Appendix C. Relevance Information

The following problem-specific relevance assignments for predicates in the background knowledge were obtained from R.D. King, University of Manchester.

Problem	Relevance	Predicates
Mut188	1	Atoms and bonds
	2	3-dimensional distance
	3	Functional groups and rings
	4	LUMO, hydrophobicity
	5	Expert-identified indicator variables
Canc330	1	Atoms and bonds
	2	Functional groups and rings
	3	Carcinogenic alerts
	4	Outcome of genetic tests
DssTox	1	Atoms and bonds
Alzh.	1	Substitutions at templates
Datasets	2	Hansch-type predicates (size, polarity *etc.*)

Appendix D. Predictive Accuracy of the DRM

The principal observations that can be made from the main tabulations in Fig. 10 are that the predictive accuracy of the DRMs clearly compare favourably to the best reports in the literature. Therefore the results provide evidence that a deep relational machine (DRM) equipped with domain knowledge and randomly drawn first-order features can construct good predictive models using (by deep-learning standards) very few data instances.

Quantitative assessments of the results are also possible, with the usual cautions associated with small numbers and multiple comparisons. The appropriate test for comparing the predictive accuracy of the DRM is the Wilcoxon signed-rank test, with the null hypothesis that the DRM's accuracy is the same as the method being compared. This yields P-values of < 0.05 for the comparisons against *OptILP* and *Stat* (we omit a comparison against the DRM in Lodhi (2013), due to lack of data).

Appendix E. Example Explanations

The following explanations are for a test-instance in the $Canc330$ problem (specifically, test-instance 2 on the 3rd cross-validation split). This example was chosen since it illustrates a number of interesting aspects: all explanations have perfect fidelity; structuring increases relevance; and there are several structured explanations possible. The DRM has 2196 features, of which 397 are active for this instance. The DRM correctly predicts the instance as belonging to the "positive" class. All the symbolic explanations below predict the same class-values as the DRM for the test-instance and its neighbours.

Unstructured explanation:

Label: $\langle L_H = 1.0, P_H = \{[1,4]\}\rangle$
Explanation H:
$Class(x,c) \leftarrow F_{537}(x), F_{1196}(x), F_{610}(x), F_{611}(x), F_{1657}(x)$

Structured explanation(s):

Label: $\langle\ L_{H_1} = 1.0, P_{H_1} = \{[1,4],[4,4]\}\rangle$
Explanation H_1:
$Class(x,c) \leftarrow F_{1,1}(x), F_{1,2}(x)$
$F_{1,1}(x) \leftarrow F_{1657}(x)$
$F_{1,2} \leftarrow F_{537}(x), F_{1196}(x), F_{610}(x), F_{611}(x)$

Label: $\langle\ L_{H_2} = 1.0, P_{H_2} = \{[1,4],[4,4]\}\rangle$
Explanation H_2:
$Class(x,c) \leftarrow F_{1,1}(x), F_{1,2}(x)$
$F_{1,1}(x) \leftarrow F_{611}(x)$
$F_{1,2} \leftarrow F_{537}(x), F_{1196}(x), F_{610}(x), F_{1657}(x)$

Label: $\langle\ L_{H_3} = 1.0, P_{H_3} = \{[1,4],[4,4]\}\rangle$
Explanation H_3:
$Class(x,c) \leftarrow F_{1,1}(x), F_{1,2}(x)$
$F_{1,1}(x) \leftarrow F_{1657}(x), F_{611}(x)$
$F_{1,2} \leftarrow F_{537}(x), F_{1196}(x), F_{610}(x)$

Feature-definitions:
$F_{537}(x) \leftarrow Atm(x,y,h,3,z), Gteq(z,0.115)$ (Relev = [1,1])
$F_{1196}(x) \leftarrow Atm(x,y,c,22,z), Gteq(z,-0.111), Atm(x,w,c,22,z)$ (Relev = [1,1])
$F_{610}(x) \leftarrow Non\_ar\_hetero\_6\_ring(x,u), Has\_property(x,ames,p)$ (Relev = [2,4])
$F_{611}(x) \leftarrow Has\_property(x,salmonella,n),$
$Has\_property(x,mouse\_lymph,p)$ (Relev = [4,4])
$F_{1657}(x) \leftarrow Has\_property(x,cytogen\_ca,n), Has\_property(x,mouse\_lymph,p),$
$Has\_property(x,cytogen\_sce,p)$ (Relev = [4,4])

References

Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings.*, pages 420–434, 2001.

E. Alphonse and C. Rouveirol. Lazy propositionalisation for Relational Learning. In W. Horn, editor, *ECAI-2000: Proc. 14th European Conf. on Artificial Intelligence*, pages 256–260, 2000.

Michael Bain. Experiments in non-monotonic learning. In *Proceedings of the Eighth International Workshop (ML91), Northwestern University, Evanston, Illinois, USA*, pages 380–384, 1991.

Tarek R. Besold, Artur S. d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro M. Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luís C. Lamb, Daniel Lowd, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, Hoifung Poon, and Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. *CoRR*, abs/1711.03902, 2017.

H. Blockeel and S. Valevich. Subtle. Available at: https://dtai.cs.kuleuven.be/software/subtle/, 2016.

Hendrik Blockeel. Top-down induction of first order logical decision trees. *AI Commun.*, 12(1-2):119–120, 1999.

C-L. Chang and R. C-T. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, London, 1973.

G. Coletti and R. Scozzafava. A coherent qualitative Bayes' theorem and its application in artificial intelligence. In *Proc. 2nd Intl. Symp. Uncertainty Modeling and Analysis*, pages 40–44, 1993.

Mark Craven and Jude W. Shavlik. Using sampling and queries to extract rules from trained neural networks. In *Machine Learning, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, July 10-13, 1994*, pages 37–45, 1994.

Tirtharaj Dash, Ashwin Srinivasan, Lovekesh Vig, Oghenejokpeme Orhobor, and Ross King. Large scale assessment of deep relational machines. In E. Bellodi F. Riguzzi and R. Zese, editors, *Proceedings of the 28th International Conference on Inductive Logic Programming*, volume 11105 of *Lecture Notes in Computer Science*, pages 22–37. Springer, 2018.

Artur S. d'Avila Garcez and Gerson Zaverucha. The connectionist inductive learning and logic programming system. *Appl. Intell.*, 11(1):59–77, 1999.

Artur S. d'Avila Garcez, Krysia B. Broda, and Dov M. Gabbay. *Neural-Symbolic Learning Systems: Foundations and Applications.* Perspectives in Neural Computing. Springer, 2002.

Manoel V. M. França, Gerson Zaverucha, and Artur S. d'Avila Garcez. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine Learning*, 94(1):81–104, 2014.

Manoel Vitor Macedo França, Artur S. d'Avila Garcez, and Gerson Zaverucha. Relational knowledge extraction from neural networks. In *Proceedings of the NIPS Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches co-located with the 29th Annual Conference on Neural Information Processing Systems (NIPS 2015), Montreal, Canada, December 11-12, 2015.*, 2015.

D. Gabbay. *Labelled Deductive Systems*, volume 1. Clarendon Press, 1996.

Dov M. Gabbay, C.J. Hogger, and J.A. Robinson. *Logic Programming.* Vol. 5 of Handbook of Logic in Artificial Intelligence and Logic Programming. Clarendon Press, Oxford, 1998.

I.J. Good. *Good Thinking: The Foundations of Probability and its Applications.* University of Minnesota, Minneapolis, 1983.

Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning.* Adaptive computation and machine learning. MIT Press, 2016.

C.J. Hogger. *Essentials of Logic Programming.* Clarendon Press, Oxford, 1990.

Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3128–3137, 2015.

R. D. King and A. Srinivasan. Prediction of rodent carcinogenicity bioassays from molecular structure using inductive logic programming. *Environmental Health Perspectives*, 104:pp. 1031–1040, Oct. 1996a.

R. D. King, S. H. Muggleton, A. Srinivasan, and M J Sternberg. Structure-activity relationships derived by machine learning: the use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 93(1):438–42, January 1996a.

R.D. King and A. Srinivasan. Prediction of rodent carcinogenicity bioassays from molecular structure using inductive logic programming. *Environmental Health Perspectives*, 104(5): 1031–1040, 1996b.

R.D. King, S.H. Muggleton, A. Srinivasan, and M.J.E. Sternberg. Structure-activity relationships derived by machine learning: The use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proc. of the National Academy of Sciences*, 93:438–442, 1996b.

Ross D. King, Kenneth E. Whelan, Ffion M. Jones, Philip G. K. Reiser, Christopher H. Bryant, Stephen H. Muggleton, Douglas B. Kell, and Stephen G. Oliver. Functional genomic hypothesis generation and experimentation by robot scientist. *Nature*, 427:247–52, 2004.

S. Kramer. Demand-Driven Construction of Structural Features in ILP. In C. Rouveirol and M. Sebag, editors, *ILP 2001: Proc. 11th Intl. Conference on Inductive Logic Programming*, number 2157 in LNAI, Berlin, 2001. Springer.

N. Landwehr, A. Passerini, L. De Raedt, and P. Frasconi. K-Foil: Learning Simple Relational Kernels. In Y. Gil and R. Mooney, editors, *AAAI-2006: Proc. 21st National Conference on Artificial Intelligence*, pages 389–394, 2006.

J. W. Lloyd. *Logic Programming, 2nd Edition.* Springer-Verlag, Berlin, 1987.

Huma Lodhi. Deep relational machines. In *Neural Information Processing - 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part II*, pages 212–219, 2013.

Eric McCreath and Arun Sharma. LIME: A system for learning relations. In *Algorithmic Learning Theory, 9th International Conference, ALT '98, Otzenhausen, Germany, October 8-10, 1998, Proceedings*, pages 336–374, 1998.

R.S. Michalski. A theory and methodology of inductive learning. In R. Michalski, J. Carbonnel, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 83–134. Tioga, Palo Alto, CA, 1983.

D. Michie. The superarticulacy phenomenon in the context of software manufacture. *Proc. R. Soc. Lond. A*, 405:185–212, 1986.

D. Michie and R. Johnston. *The Creative Computer: Machine Intelligence and Human Knowledge.* Viking Press, 1984.

Donald Michie. Consciousness as an engineering issue, part 2. *Journal of Consciousness Studies*, 2(1):52–66, 1995.

T.M. Mitchell, R.M. Keller, and S.T. Kedar-Cabelli. Explanation-Based Generalization: A Unifying View. *Machine Learning*, 1(1):47–80, 1986.

S. Muggleton. Inductive Logic Programming: derivations, successes and shortcomings. *SIGART Bulletin*, 5(1):5–11, 1994.

S. Muggleton. Inverse Entailment and Progol. *New Gen. Comput.*, 13:245–286, 1995.

S. Muggleton and L. De Raedt. Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19(20):629–679, 1994.

S.H. Muggleton. Duce, an oracle based approach to constructive induction. In *IJCAI-87*, pages 287–292. Kaufmann, 1987.

Stephen Muggleton, José Carlos Almeida Santos, and Alireza Tamaddoni-Nezhad. Toplog: ILP using a logic program declarative bias. In *Logic Programming, 24th International Conference, ICLP 2008, Udine, Italy, December 9-13 2008, Proceedings*, pages 687–692, 2008.

Stephen H. Muggleton, Dianhuan Lin, and Alireza Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. *Machine Learning*, 100(1):49–73, 2015.

N. J. Nilsson. *Principles of Artificial Intelligence*. Springer-Verlag, Berlin, 1980.

A. Petterossi and M. Proietti. Transformation of Logic Programs. In D. Gabbay. Hogger and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 5 - Logic Programming*, pages 697–787. Clarendon Press, Oxford, 1998.

A. Popescul and L. Ungar. Dynamic Feature Generation for Relational Learning. In *3rd International Workshop on Multi-Relational Data Mining*, 2004.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

Amrita Saha, Ashwin Srinivasan, and Ganesh Ramakrishnan. What kinds of relational features are useful for statistical learning? In *ILP*, 2012.

Ute Schmid, Christina Zeller, Tarek R. Besold, Alireza Tamaddoni-Nezhad, and Stephen Muggleton. How does predicate invention affect human comprehensibility? In *Inductive Logic Programming - 26th International Conference, ILP 2016, London, UK, September 4-6, 2016, Revised Selected Papers*, pages 52–67, 2016.

A.D. Shapiro. *Structured Induction in Expert Systems*. Addison-Wesley, Wokingham, 1987.

Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezný, and Ondrej Kuzelka. Lifted relational neural networks. In *Proceedings of the NIPS Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches co-located with the 29th Annual Conference on Neural Information Processing Systems (NIPS 2015), Montreal, Canada, December 11-12, 2015.*, 2015.

A. Srinivasan. The Aleph Manual. Available at http://www.comlab.ox.ac.uk/oucl/ research/areas/machlearn/Aleph/, 1999.

A. Srinivasan and R. King. Feature construction with inductive logic programming: a study of quantitative predictions of biological activity aided by structural attributes. In S. Muggleton, editor, *ILP'96: Proc. 6th Inductive Logic Programming Workshop*, volume LNAI 1314, pages 89–104, 1996.

A. Srinivasan, S. H. Muggleton, M. J. E. Sternberg, and R. D. King. Theories for mutagenicity: A study in first-order and feature-based induction. *Artif. Intell.*, 85(1-2): 277–299, 1996.

A. Srinivasan, R.D. King, and M.E. Bain. An empirical study of the use of relevance information in Inductive Logic Programming. *Machine Learning Research*, 4(Jul):369–383, 2003.

Ashwin Srinivasan. Extracting context-sensitive models in inductive logic programming. *Machine Learning*, 44(3):301–324, 2001.

Ashwin Srinivasan and Ganesh Ramakrishnan. Parameter screening and optimisation for ILP using designed experiments. *Journal of Machine Learning Research*, 12:627–662, 2011. URL `http://portal.acm.org/citation.cfm?id=1953067`.

Ashwin Srinivasan, Tanveer Faruquie, Indrajit Bhattacharya, and Ross King. Topic models with relational features for drug design. In *ILP*, 2012.

Ian Stewart. The Ultimate in Anty-Particles. *Scientific American*, July, 1994.

Sebastian Thrun. Extracting rules from artifical neural networks with distributed representations. In *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*, pages 505–512, 1994.

L. Vig, A. Srinivasan, M. Bain, and A. Verma. An investigation into the role of domain-knowledge on the use of embeddings. In N. Lachiche and C. Vrain, editors, *Inductive Logic Programming. ILP 2017.*, volume 10759 of *Lecture Notes in Computer Science*, Cham, 2018. Springer.

F. Zelezny and N. Lavrač. Propositionalization-based relational subgroup discovery with RSD. *Machine Learning*, 62:33–63, 2006.

Blaz Zupan, Ivan Bratko, Marko Bohanec, and Janez Demsar. Function decomposition in machine learning. In *Machine Learning and Its Applications, Advanced Lectures*, pages 71–101, 2001.