Mode-Directed Neural-Symbolic Modelling

Ashwin Srinivasan[1] and Lovekesh Vig[2]

[1] Department of Computer Sc. & Information Systems
BITS Pilani, K.K. Birla Goa Campus, Goa
[2] TCS Research, New Delhi.

Abstract. Our interest in this paper is in the development of a hybrid model combining neural and symbolic representations. The hybrid consists of two models: a deep network model for prediction that uses as inputs patterns in first-order logic, and a logical model that uses the network's structure and prediction to construct explanations. The first step in constructing the neural-symbolic model is to provide the neural model with some means of incorporating two of the stand-out aspects of Inductive Logic Programming. These are the use of an expressive subset of first-order logic as a representation language, and the explicit use of background knowledge. A hybrid model consisting of deep networks and ILP was recently proposed in the form of Deep Relational Machines (DRMs), in which ILP-constructed Boolean features were provided as the input layer for a deep network. The approach was promising, but experimental evidence was limited. In this paper, we retain the principle proposed in DRMs of introducing relational features at the input layer. However we do not pre-select good features using an ILP engine. Instead, we construct deep networks with features randomly drawn from a space that satisfies some approximate constraints on relevance and non-redundancy. The second stage of the hybrid consists in constructing hypotheses in first-order logic as explanations for the predictions made by the neural network. No attempt is made to construct a single symbolic model for the entire network. Instead, the symbolic explanations are models that are locally-consistent explanations for the predictions. On several benchmarks tested, the predictive models compare favourably to the best reports in the ILP literature, and the symbolic models produce reasonably high-fidelity explanations.We also demonstrate a method of structuring the symbolic explanations, without losing fidelity, using activations in the neural network.

1 Introduction

It is not inevitable that machine learning models must 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": [9]), 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 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. For example, in his book "Words and Rules", Steven Pinker conjectures language learning as one such task. It is sometimes overlooked that the prominent recent success in the board game Go [22], was in fact not achieved just with deep neural networks, but with a hybrid approach that combined a deep network with classical tree-based search. Our interest here is here is combining neural and symbolic models, with the former being used for prediction, and the latter for explanations. Specifically, we will be using a deep neural network learning to construct a predictive model in terms of randomly drawn first-order features, and in turn, use the neural model to construct an explanatory symbolic model for queries.[3] Following [6], we will call such a model a hybrid, neural-symbolic model. Having two separate models would appear to be an extravagance that could be ill-afforded if the predictive model wasn't very good, or if the explanatory model turned out to be unreflective of the underlying predictive model. In this paper, we present empirical evidence that neither of these appears to happen here.

For ML practioners, the principal points of interest in the paper are these: (a) Expressive first-order features that use domain-knowledge can be incorporated into a deep neural network without requiring the support of a full-fledged ILP engine; (b) A deep neural network equipped with such features can construct good predictive models using (by deep-learning standards) small numbers of data instances; and (c) It is possible to extract

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

symbolic explanations for predictions made by the deep network that is (mostly) consistent with the predictions of the network on similar instances.

A puritanical ILP follower could ask: what do we gain from the deep network, when we can directly learn an ILP model? To this we note the following. First, the deep network guided explanatory models provide one answer to the difficult problem of automatic structuring of hypotheses. But, we do not insist on a single model to explain all (or even most) of the data. This will require a more tolerant view of what constitutes a hypothesis in ILP than is found in the literature. Second, by mapping the problem of learning first-order hypotheses to deep network models, we are able to call on increasingly sophisticated software and hardware implementations for constructing very large models. It is not apparent at this stage if the usual ILP techniques of greedy variants of discrete combinatorial search will be able to scale in a similar manner. It is not inconceivable that some kinds of ILP theories may consist of large collections of weighted first-order clauses obtained from nodes in a deep network. A glimpse of this future is already with us in [23].

2 A Deep Neural Network 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 a rebranding of 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 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 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. Our approach is to provide a deep network with values of first-order features. The features will be defined in terms of predicates in the domain-knowledge.

2.1 First-Order Features as Inputs

Deep Relational Machines, or DRMs, proposed in [12], consists of first-order features learned by an ILP engine as inputs to a deep network of the kind described in the previous section. This follows a long line of research in which features constructed by ILP have been used by other modelling methods like regression, decision-trees, SVMs, topic-models, and multiplicative-weight linear threshold models. In each of these, the final model is constructed in two steps: first, an ILP engine constructs a set of "good" features, 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 [12], the deep network with ILP-features is shown to perform reasonably well (the DRMs do not actually result in the best performance for the problems compared).

In this paper, we do not want to pre-select input features using an ILP engine. Instead, we would (ideally) like the inputs to consist of all possible 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. We clarify first what we mean by a feature.

Remark 1 Features for classification. *The set of examples provided can be defined as a binary relation which is a subset of the Cartesian product $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ denotes the set of instances (for simplicity, we will take this to be each instance to be a first-order term) and $\mathcal{Y}$ denotes the finite set of classes. We will represent definite-clauses for classifying examples as $h_{j,c}(x) : \forall x(Class(x,c) \leftarrow Body)$ where Body is a conjunction of literals of the form $F_k(x)$. Each F_k is called a* feature *and represents a Boolean function defined in terms of a conjunction of predicates $Cp_k(x)$ defined in domain- or background-knowledge B. That is, $F_k(x) = 1$ iff $Cp_k(x) = 1$ and 0 otherwise. We will represent these functions as single definite clauses in a logic program ($F_k(x) \leftarrow Cp_k(x)$), relying on the closed-world assumption for the complete definition of the function F_k. The definite clauses will sometimes be called* feature-clauses.

Example 2 Feature-clauses for trains. *A well-known synthetic problem in the ILP literature (Michalski's "Trains" problem) can be used to illustrate this. A clause for classifying eastbound trains using feature-definitions is (we leave out the quantifiers for simplicity) is: $h_j : Class(x, east) \leftarrow F_1(x), F_2(x)$. Feature-clauses could be definitions like: $F_1(x) \leftarrow Has\_Car(x,y), Short(y)$ and $F_2(x) \leftarrow Has\_Car(x,y), Closed(y)$. We assume predicate definitions for Has_Car* etc. *are part of the domain knowledge B.*

Clearly, with a sequence of feature functions $\mathcal{F} = (F_1, F_2, \ldots, F_d)$ we are able to obtain a Boolean-vector representation $a' \in \{0,1\}^d$ of an instance $a \in \mathcal{X}$ using the function $FV : \mathcal{X} \rightarrow \{0,1\}^d$, where the i^{th} component $FV_i(x) = 1$ if $F_i(x) = 1$, and 0 otherwise (correctly, FV depends on $\mathcal{F}$). We will sometimes say a' is the *feature-space* representation of a. In this paper, we will assume that $\mathcal{F}$ contains sufficiently many different feature-functions such that FV is a 1-1 mapping (that is, every element in the object space $\mathcal{X}$ can be uniquely represented in the feature-space). It is evident also that given any d-dimensional Boolean vector $v = (v_1, v_2, \ldots, v_d)$, can be mapped to a conjunction of literals $\bigwedge_{v_i=1} F_i(x)$ that is true for all objects $x \in \mathcal{X}$ for which the conjunction is true.

2.2 Relevance and Redundancy

The class of features is defined purely on syntax. In itself, it does not guarantee that a feature in the class is actually relevant to the problem considered, or that it is redundant given another feature from the same class. For example, the feature $F_1(x) \leftarrow Has\_Car(x,y), Open(y), Closed(y))$ is clearly irrelevant in the trains problem, since no car can be both open and closed. Clearly, $F_2(x) \leftarrow Has\_Car(x,y), Short(y), Closed(y)$ is redundant given $F_3(x) \leftarrow Has\_Car(x,y), Closed(y), Short(y)$ and *vice versa*.

Irrelevance, as we have just described it, is a domain-dependent issue. In its full scope, redundancy is also a semantic notion given the domain knowledge. We will use the some approximations, for which we need some well-understood concepts from the ILP literature:

Remark 3 Clause subsumption. *We use Plotkin's partial ordering on the set of clauses [17]. A clause C subsumes a clause D or $C \succeq_\theta D$, iff there exists a substitution θ s.t. $C\theta \subseteq D$. It is known that if $C \succeq D$ then $C \models D$. Further if $C \succeq_\theta D$ and $D \succeq_\theta C$ we will say $C \equiv_\theta D$.*

Remark 4 Most specific clause in a depth-bounded mode language. *This notion is due to Muggleton [15]. Given a set of modes M, let $\mathcal{L}_d(M)$ be the d-depth mode language. Given background knowledge B and an instance e, let the most specific clause that logically entails e in the sense described in [15] be $\bot(B,e)$. For every $\bot(B,e)$ it is shown in [15] that: (a) there is a $\bot_d(B,e)$ in $\mathcal{L}_d(M)$ s.t. $\bot_d(B,e) \succeq \bot(B,e)$; and (b) $\bot_d(B,e)$ is finite.*

Using these notions, we adopt the following definitions:

Definition 5 Relevance. *Given a set of examples E, and background knowledge B an independent clause C is relevant if $C \succeq \bot_d(B,e)$ for at least one $e \in E$. We will further restrict this to $C \subseteq \bot_d(B,e)$.*

Definition 6 Redundance. *For a pair of independent clauses C and D, we will say D is redundant given C (and*vice-versa*), if $|C| = |D|$ and $C \equiv_\theta D$*

We are, therefore, using the most-specific clause in the sense described in [15] to determine relevance, and a restricted form of clause subsumption for detecting redundancy. Both definitions are approximations, and can result in some errors (assuming relevance when it does not exist, and missing redundance when it does exist): nevertheless, they can be computed reasonably efficiently. They form the basis of a randomised construction of inputs for the deep network.

2.3 Drawing Features Randomly

The mechanisms for detecting relevance and redundancy helps reduce the number of possible features within a mode language. But, they still do not guarantee that the numbers will be manageable. We therefore resort to a rejection-based sampling strategy for selecting inputs:

$DrawFeatures(B, M, E, \mathcal{L}, d, MaxDraws)$:

1. Let F be $\langle\rangle$
2. Let $draws = 0$
3. Let $i = 1$
4. Let $Drawn$ be $\emptyset$
5. **while** $draws \leq MaxDraws$ **do**
   (a) Randomly draw with replacement an example $e_i \in E$
   (b) Let $\bot_d(B, e_i)$ be the most specific rule in the depth-limited mode language $\mathcal{L}_d(M)$ that subsumes $\bot(B, e_i)$ (the most-specific clause that entails e, given B).
   (c) Randomly draw an clause C_i s.t. $C_i \subseteq \bot_d(B, e_i)$
   (d) **if** (C_i is not redundant given $Drawn$) **then**
      i. Let $C_i = (Class(x, c) \leftarrow Cp_i(x))$
      ii. Lef $f_i = (F_i(x) \leftarrow Cp_i(x))$
      iii. Update sequence F with f_i
      iv. $Drawn := Drawn \cup \{C_i\}$
      v. increment i
   (e) increment $draws$
6. **done**
7. return F

A procedure for randomly drawing clauses subsuming a most-specific clause is described in [28]. Of course, inputs to the deep network are not the features themselves but the feature-space representations for the data instances E. We will call neural nets constructed in this manner *knowledge-rich deep networks*, or KRDNs for short.

3 A Symbolic Model for Explanation

It has long been understood that while neural models can compute accurate answers for questions like "What is the prediction for $\mathbf{x}$?". But we are not able to say much more about why the answer follows, other than providing a trace of the numerical calculation that was used. Unsurprisingly, there has been a lot of research effort invested into extracting comprehensible models from neural networks (see [6], Chapter 3 for a full description of this line of research). While this effort has mostly been in the direction of extracting symbolic models that guarantee correspondence to the neural model (for example, [27]), or approximate it's behaviour (for example, [3]). It is not evident that when applied to modern-day deep networks, the resulting symbolic models will in fact be any more comprehensible than their neural counterparts. Some of the difficulty stems from the requirement of translating the entire network model into a symbolic form at once. Recent work [18] proposes producing explanatory models "on-demand", restricting them to being locally consistent with the predictive model. That is, the explanatory model is constructed when a prediction for a (usually new) instance is sought, and the explanation is restricted to the instance and its near-neighbours. The intuition is that while comprehensible models may not be possible for the entire predictive model, they may be possible for smaller local predictions. We explore this for the construction of local explanations using a deep network, drawing mainly on two specific situations described in [14], namely that of the "Single example clause, single hypothesis clause", and the "Multiple examples, single hypothesis clause". We clarify first what is meant by an explanation.

Remark 7 Explanations. *In this paper, we will only be concerned with explanations for the classification of an instance. As before, we assume a set of instances $\mathcal{X}$, and a finite set of classes $\mathcal{Y}$. Given an 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, an explanation for e is a set of clauses H s.t. $B \wedge H \models e$.*

We will seek explanations of a restricted kind. Each possible H will be of the form $FP \cup \{C\}$, where FP is a set of feature-clauses of the form described earlier (that is, $F_i(x) \leftarrow Cp_i(x)$). and C is a definite clause for $Class(x, y) \leftarrow Body$, where $Body$ consists of $F_i/1$ literals, defined in FP. It is evident that for any definite clause C' such that $C' \succeq_\theta C$, then $FP \cup \{C'\}$ is also an explanation. Taking the definitions in FP to be an augmentation of B, finding an appropriate explanation for an example e is an instance of the "Single example clause, single hypothesis clause" situation identified in [14], which forms the basis of explanation-based learning (EBL).

One explanation for $e = Class(a, c)$ can be obtained immediately as follows. Let $F \subseteq \mathcal{F}$ denote the set of all feature-functions with value 1 in $a' = FV(a)$. Let us call these the set of *active features* for a', and FP denote the corresponding feature-clauses. For simplicity, suppose the first k features were active for a', and let $C : \ Class(a, c) \leftarrow F_1(a), F_2(a), \ldots, F_k(a)$. Then $H = FP \cup \{C\}$ is an explanation for e. The reader will recognise C as being analogous to the most-specific clause in [15], and we will call it the most-specific explanation of e, given $\mathcal{F}$ and B. We are specifically interested in constructing explanations for the classification of an instance resulting from some (opaque) predictive model, rather than the true class of the instance.

Remark 8 Explanation consistent with a predictive model. *Given an instance $a \in \mathcal{X}$ and a label $c \in \mathcal{Y}$. Let $\mathcal{F}$ be a sequence of d feature-functions, and $a' = FV(a)$ be the feature-space representation of a, given $\mathcal{F}$. Given a predictive model N, let $N(a') = c$. We seek an explanation H for $e = Class(a, c)$. (We have just seen how to construct the most-specific explanation for e from the set of active features for a').*

In fact, we will want a little bit more from an explanation. Following [18], we will seek explanations that are not only consistent with a predictive model on an instance x, but be consistent with predictions made by the predictive model in a local neighbourhood of x.

Remark 9 Locally consistent explanation. *Given an $a \in \mathcal{X}$, and a d-length sequence of feature functions $\mathcal{F}$, let $a' = FV(a)$ be the d-dimensional feature-space represention of a given $\mathcal{F}$. Let $\delta(a')$ be a (finite) set of feature-vectors denoting a local neighbourhood of a' (based on some suitable kernel function, say). For simplicity, we will allow $a' \in \delta(a')$. Given a predictive model N, we have the following subsets of $\delta(a')$: (i) $\delta^+(a')$ s.t. for all $b' \in \delta^+(a')$, $N(b') = N(a')$; and (ii) $\delta^-(a')$ s.t. for all $b' \in \delta^-(a')$, $N(b') \neq N(a')$. Then, we would like explanations for a to be consistent with the labels of instances in δ.*

That is, let $E^+(a) = \{b : a' = FV(a)$ and $b' \in \delta^+(a')$ and $b = FV^{-1}(b')\}$ and $E^-(a) = \{b : a' = FV(a)$ and $b' \in \delta^-(a')$ and $b = FV^{-1}(b')\}$. Given a predictive model N, let $a' = FV(a)$ and $N(a') = c$. We seek an explanation H for $Class(a, c)$ given B s.t.: (1) for each $a' \in E^+(a)$ H is an explanation for $Class(a', c)$ (that is, $B \wedge H \models Class(a', c)$); (2) for each $a' \in E^-(a)$ $B \wedge H \wedge \neg Class(a', c) \not\models \square$

This corresponds directly to the "Multiple examples, single hypothesis clause" situation identified in [14], using the predictive model (and not the oracle) for class labels. It is therefore possible to use the search methods developed in ILP to find the hypothesis clause. These may result in a single-clause hypothesis that is only approximately consistent, and we can attach a *fidelity* to the explanation, based on the proportion of locally instances correctly explained.

Remark 10 Fidelity. *Let $E^+(a)$ and $E^-(a)$ as before, and let H be an explanation for $Class(a, c)$. Let (1) $AgreePos = \{b : b \in E^+(a)$ and $B \wedge H \models Class(b, c)\}$; and (2) $AgreeNeg = \{b : b \in E^-(a)$ and $B \wedge H \wedge \neg Class(b, c) \not\models \square\}$. Then $Fidelity(H) = \frac{|AgreePos|+|AgreeNeg|}{|E^+(a)|+|E^-(a)|}$*

Below we show a search-based procedure for the highest-fidelity explanation for an instance x, given: (a) a predictive model N; (b) a set of features $\mathcal{F}$; (c) a neighbourhood function δ; and (d) background knowledge B.

$ConstructExplanation(x, N, \mathcal{F}, \delta, B)$:

1. Let $x' = FV(x)$
2. Let $c = N(x')$
3. Let $\delta^+(x') \subseteq \delta(x')$ be the neighbours of x' with the same prediction as x'
4. Let $\delta^-(x') = (\delta(x') - \delta^+(x'))$
5. Let $F \subseteq \mathcal{F}$ be the set of features that have the value 1 for x'
6. Let $F' \subseteq F$ be the subset with highest fidelity computed over δ^+ and δ^-
7. Let FP be the set of clausal definitions for features in F'

8. Let $C = \forall x(Class(x, c) \leftarrow Body)$ where $Body$ is the conjunction of literals $\bigwedge FP_i(x)$, where each FP_i is defined in FP.
9. return $FP \cup \{C\}$

The reader will immediately note an anomaly. The search for the best fidelity explanation is described here in the feature-space, but our definition of fidelity was over elements in $\mathcal{X}$-space. This is possible because of the assumption of FV being invertible; and the direct correspondence between features in the instance-space and feature-clauses in the object space. ILP-practitioners will also recognise the procedure as being a feature-space variant of the Progol algorithm ([15]): it is a a general-to-specific search for subsets of a maximal-set of features that are true ($\bot$) for the instance x' (Step 5). It is evident therefore feature-clause explanations arising from subsets of $\bot$ will be explanations for the corresponding instance x. Each such subset will also be true of some of the neighbourhood of x', allowing a calculation of fidelity for the corresponding feature-clause. Also like Progol, a complete search will clearly be of exponential time-complexity, since Step 6 would simply examine subsets of the set F. Practicalities will therefore force a heuristic search, akin to what is done with practical ILP systems.

3.1 Structuring Explanations

Although the usual ILP techniques allow us to construct single-clause explanations we would like explanations with structure. The principal transformations of interest to us are the logic-programming operators of folding and the ILP operators of inverse-resolution (see [19] for the relationship of inverse-resolution and folding). For example, for the trains problem, suppose there was a new feature defined (only) as follows: $NewF(x) \leftarrow F_1(x), F_3(x)$, and our search procedure gave us the the following high-fidelity explanation (let us ignore the feature-clause definitions for simplicity) $H_1 : Class(x, east) \leftarrow F_1(x), F_2(x), F_3(x)$. Then, this clause could be transformed to $H_2 : Class(x, east) \leftarrow NewF(x), F_2(x)$, by folding on $NewF$. Of course, this relies on having the $NewF$ definition in the first place (a point of difference between inverse-resolution operator of absorption and folding is in this step: absorption invents the definition of $NewF$). In our case, we propose using the structure of our predictive model—here, a deep network—to guide the introduction of such new definitions.

3.2 Model-Guided Structured Explanations

We will assume our predictive model is a neural network that uses first-order features constructed as described in Section 2.3. Recall that there is an input node in the deep network for each feature constructed. For any instance a, assigned class c by the network, let us assume that we have obtained an explanation for $Class(a, c)$. This explanation consists of a single clause of the form $\forall x(Class(x, c) \leftarrow Body$, where $Body$ contains the feature-literals that need to be true for the explanation. Activating the corresponding input nodes in the network will result in progressively activating nodes in hidden layers until the output node for c is reached.

We propose to introduce a new feature for each active node in hidden layers terms of active nodes of preceding layer. Specifically suppose a high-fidelity explanation results in j active nodes in a layer l, and k active nodes in layer $(l-1)$. We use the trained neural network to determine a small subset of the k nodes in layer $(l-1)$ that are needed to activate each of the active j nodes in layer L, and a corresponding new feature is constructed. We sketch the procedure for a pair of layers in a network N given an unstructured (presumably high-fidelity) explanation H. The entire process simply iterates this for every pair of adjacent layers (with some attention needed for the start and finish layers).

$ConstructNewF(N, H, l)$:

1. Let J be the set of active nodes in layer l of the network N
2. If $l-1$ is the input layer, then K is the set of active nodes in the network corresponding to explanation H; otherwise K is the set of active nodes in layer $l-1$ of N.
3. Let $j = |J|$ and $k = |K|$
4. Let $FP = \emptyset$
5. Let $JK = \{(i, K_i) :\ i \in J$ *and* $K_i \subseteq K\}$ s.t.
   (a) The K_i must cover K (that is, $K = \bigcup K_i$
   (b) The K_i are not empty, and are as small as possible
   (c) For each (i, K_i) in JK the nodes in K_i are sufficient to activate node i, given the network N
6. For each $(i, K_i) \in JK$:
   (a) Associate node i with the feature $F_{l,i}$

(b) $FP := FP \cup \{\forall x(F_{l,i}(x) \leftarrow Body)\}$ where $Body$ is the conjunction of literals $\bigwedge_{j \in K_i} F_{l-1,j}(x)$, where the $F_{l-1,j}$ is the feature associated with node $j \in K_i$

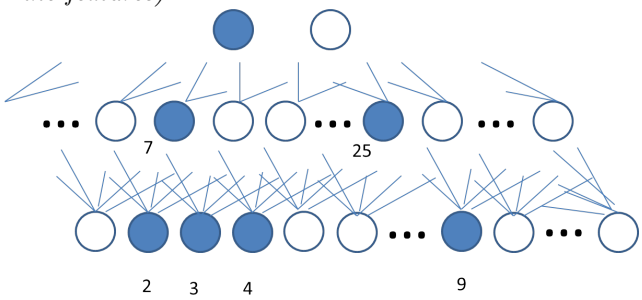
7. return FP

In $ConstructNewF$: (a) The is exactly one new feature-clause for every active node; (b) Each new feature-clause does not introduce any new variables; and (c) The covering requirement in Step 5a ensures the conjunction of feature-clauses at the final hidden layer can be unfolded to the feature-clauses in the unstructured explanation H. It is not hard to see that a procedure that uses $ConstructNewF$ over pairs of adjacent layers will allow us to transform the unstructured explanation into a sequence of new programs, each resulting from a folding step. It can be shown that the conditions (a)–(c) are sufficient to ensure correctness of the folding steps. Each program in the sequence should therefore have the same fidelity as the unstructured explanation.

Example 11 Structured explanation for the trains. *Suppose we are given an instance $x \in \mathcal{X}$ and a deep network N that results in an explanation with the following clause (we leave out the quantifiers for the example):*

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

Further, let us assume this clause results in the following activations (shown shaded) in the network below (a). $ConstructNewF$ uses the weights linking active nodes to result in the the new features shown in (b); and the corresponding structured explanation after two successive folding steps is in (c) (we leave out the definitions of the features):



(a)

$F_{1,7}(x) \leftarrow F_2(x), F_4(x)$
$F_{1,25}(x) \leftarrow F_3(x), F_9(x)$

(b)

$Class(x, east) \leftarrow F_{1,7}(x), F_{1,25}(x)$
$F_{1,7}(x) \leftarrow F_2(x), F_4(x)$
$F_{1,25}(x) \leftarrow F_3(x), F_9(x)$

(c)

By construction, the structured explanation in (c) unfolds to the original unstructured explanation. In general, we expect that there will only be a few nodes in the entire network would be active when attempting to explain a test instance.

We have deliberately left unspecified how to obtain the subsets K_i in $ConstructNewF$. A simple-minded approach of trying all subsets is clearly impractical with large numbers of active nodes. A simple greedy procedures can be devised however, that simply associates an active node i in layer l with the subset of active nodes K_i in layer $l-1$ that has positive edge-weights to i. This can result in a program with redundancies, which can be removed.[4] To ensure we the folding steps are correct, we do still have to ensure that the resulting program unfolds to the original unstructured explanation.

We now investigate the performance of the predictive and explanatory models constructed using the basic procedures $DrawFeatures$, $ConstrucExplanation$ and $ConstructNewF$

4 Empirical Evaluation

In this section, we evaluate empirically the predictive performance of KRDNs and the explanatory models derived from them. Our aim is investigate the following conjectures:

Predictive Accuracy. A KRDN constructed using randomly drawn features from a depth-limited mode language can perform at least as well as an ILP engine given the same language.

Explanatory Fidelity. The explanatory model produces locally consistent models of reasonably high fidelity.

[4] Using the usual checks for redundancy (C is redundant in a program P iff $(P - \{C\} \equiv P$, and a literal l is redundant in a clause $C = C1 \vee l$ iff $C \equiv C1$).

Some clarifications are necessary here: (a) By randomly constructing features, we will mean the rejection-sampling method described in Section 2.3; (2) By a local explanatory model for a KRDN, we will mean the iterative method described in Section 3; (3) The fidelity does not change with the structuring process, which preserves the consequences of the unstructured explanation. So the explanatory fidelity is assessed using the unstructured explanations; and (4) We are aware that the structured explanatory models need to be assessed on human-comprehensibility. However, this is outside the scope of this paper, which is concerned with investingating some necessary conditions before undertaking the more complex task of assessing comprehensibility.

4.1 Materials

Data We report results from experiments conducted using 7 well-studied real world problems from the ILP literature. These are: Mutagenesis [10]; Carcinogenesis [11]; DssTox [**?**]; and 4 datasets arising from the comparison of Alzheimer's drugs denoted here as *Amine*, *Choline*, *Scop* and *Toxic* [25]. Each of these have shown to benefit from the use of a first-order representation, and domain-knowledge but there is still room for improvement in predictive accuracies.

Of these datasets, the first three (Mut188–DssTox) are predominantly relational in inature, with data in the form of the 2-d structure of the molecules (the atom and bond structure), which can be of varying sizes, and diverse. 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 1000. This is extremely modest by the usual data requirements for deep learning.

For reasons of space, we refer the reader to the references cited for details of the domain-knowledge used for each problem.

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 [24] 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.

4.2 Method

Our method is straightforward:

For each dataset:
1. Obtain a set of random features F;
2. Provide the and their values for F for the data;
3. Construct a KRDN N for the data and estimate its predictive performance;
4. Construct a symbolic explanation H for N obtain an estimate of its explanatory fidelity

Some clarifications are necessary at this point:

- We use a straightforward Deep Neural Network (DNNs) architecture. There are multiple, fully connected feedforward layers of rectified linear (ReLU) units followed by Dropout for regularization (see [7] 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 [1] to perform the subsumption-equivalence test used to determine redundant features.

- For all the datasets, estimates of predictive performance using ILP methods are available in the ILP literature using 10-fold cross-validation. We use the same approach. This requires constructing KRDNs separately for 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 KRDN. 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). For a given train-test split Tr_i and Te_i, we proceed as follows. We obtain a KRDN 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 predicticted by N_i for x'_j, and corresponding neighbourhood of x'_j in the training set Tr_i. The neighbourhood is partitioned into $\delta^+(x'_j)$ and $\delta^-(x'_j)$ using the predictions by N_i and the best fidelity explanation H_{ij} obtained.
- Assessments of explanatory fidelity will be presented over two different sized neighbourhoods (instances within a 5-bit Hamming distance; and instances within a 10-bit Hamming distance).

4.3 Results

The principal results of the empirical evaluation are tabulated in Fig. 1. Some supplementary results are in Fig. 2. The principal observations that can be made from these tabulations are these: (1) The predictive accuracy of the KRDNs clearly compare favourably to the best reports in the literature; (2) Explanatory fidelity obtained from the symbolic models depends on the size of the neighbourhood. Clearly, smaller the neighbourhood, greater the explanatory fidelity: but the results show that reasonably good fidelity is obtainable even with large neighbourhoods. Together, the results provide evidence for the following:

(a) Domain-knowledge and expressive first-order features can be incorporated into a deep neural network without requiring the support of a full-fledged ILP engine;
(b) A deep neural network equipped with domain knowledge and randomly drawn first-order features can construct good predictive models using (by deep-learning standards) very few data instances;
(c) It is possible to extract symbolic explanations for the prediction made by the deep network for a randomly drawn (new) instance that is (mostly) consistent with the predictions of the network for that instance and its near-neighbours in data examined before by the network.

We also note the following additional points: (a) Changes in the neighbourhood size can change the number of instances significantly. In general, the fewer the instances, the lesser the constraints imposed on the explanations. This leads to smaller explanations (fewer literals), and higher fidelity; (b) Although the networks can often contain 1000's of input features, a high-fidelity locally-consistent explanation may only contain a few active features. Correspondingly, this results only in a few active nodes in the other layers of the network. This is what makes it possible to extract compact explanations even from large networks (we are not attempting to extract a complete symbolic model for the entire network).

Finally, the quantitative results clearly do not have anything to say on explanatory structure. Here we simply present evidence that structure-extraction is feasible, using activations in the deep network (Fig. 3). We focus on the DssTox problem, since this has both the most complex unstructured explanations (see "#Literals" in Fig.2(c)), and the representation is at a very sparse kind, consisting only of atom-and-bond structure of the molecules. Sufficient theory exists in the area of logic-programming to tell us when fold operations will be correct: for us, the structured explanations in (b) and (c) will have the same fidelity as the unstructured explanation. From Fig. 1, this means the explanation will agree with the neural predictions about 85–90% of the time. (by construction, it will always agree with the neural prediction on the test instance). However the extent to which they provide more (or less) comprehensible explanations has to be assessed by a separate study. It is instructive nevertheless, in Fig. 3 to compare the complexity of the symbolic explanation against the full predictive model, which has 200 input units, and 4 hidden layers with 5, 10, 15, and 20 units respectively (this structure is automatically obtained during the training of the network). We conjecture that providing a symbolic explanation (even of the form in (a)) is more comprehensible than an explanation in terms of numeric weights of the network.

Problem	Accuracy			
	OptILP [26]	*Stat* [20]	*DRM* [12]	*KRDN*
*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)	–	0.80(0.05)
Toxic	0.87(0.01)	0.84(0.01)	–	0.93(0.03)

(a)

Problem	Fidelity	
	H_5	H_{10}
*Mut*188	0.86(0.05)	0.86(0.04)
*Canc*330	0.73(0.09)	0.73(0.09)
DssTox	0.92(0.04)	0.85(0.03)
Amine	0.87(0.03)	0.83(0.04)
Choline	0.79*0.02)	0.73(0.03)
Scop	0.75(0.03)	0.70(0.02)
Toxic	0.81(0.02)	0.77(0.02)

(b)

Fig. 1. (a) Estimated predictive accuracies of KRDNs against some of the best reported performances in the ILP literature. All estimates are from a 10-fold cross-validation; (b) 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 KRDN's prediction assessed over the neighbourhood of a test instance. The neighbourhood is obtained from the training data. H_n refers to the neighbourhood of training instances that differ by at most n feature-values from the test instance are treated as neighbouring instances (that is, these are the training instances within an n-bit Hamming distances from the test instance., The number in parentheses are estimates of standard deviations.

Problem	Network		
	#Inputs	#HL	#Units/HL
*Mut*188	7097	3	15
*Canc*330	5365	3	6
DssTox	209	4	6
Amine	2195	3	12
Choline	2516	3	6
Scop	2420	4	13
Toxic	2327	3	14

(a)

Problem	#Neigbours	
	H_5	H_{10}
*Mut*188	5	20
*Canc*330	6	27
DssTox	11	82
Amine	9	23
Choline	15	63
Scop	9	26
Toxic	12	42

(b)

Problem	#Literals	
	H_5	H_{10}
*Mut*188	2	3
*Canc*330	2	2
DssTox	4	6
Amine	2	3
Choline	2	4
Scop	2	3
Toxic	3	4

(c)

Fig. 2. (a) Network characteristics, including average number of input features arising from rejection sampling using 10,000 draws, number of hidden layers and number of units per hidden layer (the network structure is determined automatically using a validation set); (b) Average numbers of neighbouring instances in two different sized neighbourhoods (H_n denotes an n-bit Hamming distance); and (c) Average number of literals in the unstructured high-fidelity explanation obtained using instances in the neighbourhood of a test instance. All numbers are estimates obtained from 10-fold cross-validation.

$Class(x, Active) \leftarrow F_7(x), F_{39}(x), F_{62}(x), F_{63}(x)$

$F_7(x) \leftarrow Atom(x, w, c), Atom(x, y, Hydrogen), Bond(x, w, Carbon, z, Oxygen, Single)$
$F_{39}(x) \leftarrow Atom(x, w, Oxygen)$
$F_{62}(x) \leftarrow Atom(x, w, Carbon), Bond(x, w, Carbon, y, Hydrogen, Single), bond(x, w, Carbon, z, Bromine, Single)$
$F_{63}(x) \leftarrow Atom(x, w, Nitrogen), Atom(x, y, Carbon), Bond(x, w, Nitrogen, z, Oxygen, Single)$

(a)

$Class(x, Active) \leftarrow F_{4,11}(x)$

$F_{4,11}(x) \leftarrow F_{3,4}(x)$

$F_{3,4}(x) \leftarrow F_{2,2}(x)$

$F_{2,2}(x) \leftarrow F_{1,3}(x), F_{1,4}(x)$

$F_{1,3}(x) \leftarrow F_7(x), F_{39}(x), F_{62}(x)$
$F_{1,4}(x) \leftarrow F_{62}(x), F_{63}(x)$

(b)

$Class(x, Active) \leftarrow F_{1,3}(x), F_{1,4}(x)$

$F_{1,3}(x) \leftarrow F_7(x), F_{39}(x), F_{62}(x)$
$F_{1,4}(x) \leftarrow F_{62}(x), F_{63}(x)$

(c)

Fig. 3. Example of: (a) an unstructured explanation for a test instance; and (b) a structured explanation using the active nodes in the network resulting from atest instance. The test instance here is from the DssTox problem. The simple greedy procedure we use of selecting all active nodes with positive weights results in redundancies. The structured explanation in (b) is shown with redundancies removed. The explanation in (c) is a transformation of (b) after a series of unfolding steps.

5 Other Related Work

We have already noted 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 [21]. There, 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:

- Inverse resolution, especially in the Duce system [16] was explicitly aimed at learning structured sets of rules in propositional logic. The sub-concepts are constructed bottom-up;
- Function decomposition, using HINT [29], which learns hierarchies of concepts using automatic top-down function decomposition of propositional concepts;
- First-order theories with exceptions, using the GCWS approach [13], 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: [2]. 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;

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: [23]). In this, 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).

6 Concluding Remarks

In [8], a contrast is presented between theories that predict everything, but explain nothing; and those that explain everything, but predict nothing. Both are seen as limited value in the scientific enterprise, which is thought to require models with both predictive and explanatory power. Michie [5] adds 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 natural language. Complexity limits notwithstanding, it is often assumed that that predictive and explanatory assessments refer the *same* model. But this does not have to be so: early results in the machine learning literature on behavioural cloning point to areas where people perform predictive tasks using sub-symbolic models, and provide entirely separate symbolic explanations [4]. In this paper, we have proposed the use of two different kinds of models: one used purely for prediction; and a proxy, possibly within Michie's Human Window, for explanation. Our interests were in investigating—using scientific data—the use of a deep neural network for prediction, and symbolic model in first-order logic as a proxy.

The remarkable 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, in a language that is sufficiently expressive. The knowledge-rich deep networks (KRDNs) we have proposed here is one way forward, and the results suggest that we can achieve the levels of predictivity reached by full first-order learners. Interestingly, we are able to accompany these predictions with a first-order model that provides explanations of why the network is predicting what it does. To the best of our knowledge, this is the first time ILP model has been used to complement a neural model in this manner. It is important to understand that the experimental evidence presented here does not tell us that a deep network cannot achieve the same performance with the same or more data, but without first-order features. It is not immediately apparent how this conjecture could be tested, although it may be possible to transfer features constructed by a network trained on other related problems to overcome the lack of first-order features.

It can be persuasively argued that the predictive model we have constructed here is a deep relational machine, although not of the kind proposed in [12]. Some important differences are: (a) The principal motivation in [12] appears to be prediction. There is therefore no counterpart to constructing a second, explanatory model; (b) The network in [12] requires an ILP engine. Here, we require only the ability to construct most-specific clauses within a mode language; and (c) the experimental evidence we have presented is more extensive.

There are at least two separate directions in which we plan to extend the work here. First, following [20], it is of interest to see if sub-classes of simpler features work just as well for KRDNs. Second, it would be useful to conduct a controlled study on the comprehensibility of structured explanations derived from the deep network.

Acknowledgements

A.S. is a Visiting Professor in the Department of Computer Science, University of Oxford; and Visiting Professorial Fellow, School of CSE, UNSW Sydney. A.S. is supported by the SERB grant EMR/2016/002766.

References

1. H. Blockeel and S. Valevich. Subtle. Available at: https://dtai.cs.kuleuven.be/software/subtle/, 2016.
2. Hendrik Blockeel. Top-down induction of first order logical decision trees. *AI Commun.*, 12(1-2):119–120, 1999.
3. 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.
4. D. Michie. The superarticulacy phenomenon in the context of software manufacture. *Proc. R. Soc. Lond. A*, 405:185–212, 1986.
5. D. Michie and R. Johnston. *The Creative Computer: Machine Intelligence and Human Knowledge.* Viking Press, 1984.

6. 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.
7. Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.
8. Ian Stewart. The Ultimate in Anty-Particles. *Scientific American*, July, 1994.
9. 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.
10. 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 1996.
11. 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. 1996.
12. 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.
13. Michael Bain. Experiments in non-monotonic learning. In *Proceedings of the Eighth International Workshop (ML91), Northwestern University, Evanston, Illinois, USA*, pages 380–384, 1991.
14. S. Muggleton. Inductive Logic Programming: derivations, successes and shortcomings. *SIGART Bulletin*, 5(1):5–11, 1994.
15. S. Muggleton. Inverse Entailment and Progol. *New Gen. Comput.*, 13:245–286, 1995.
16. S.H. Muggleton. Duce, an oracle based approach to constructive induction. In *IJCAI-87*, pages 287–292. Kaufmann, 1987.
17. G.D. Plotkin. A note on inductive generalisation. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pages 153–163. Elsevier North Holland, New York, 1970.
18. 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.
19. David A. Rosenblueth. Incorporating a folding rule into inductive logic programming. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 1630–1631, 2005.
20. Amrita Saha, Ashwin Srinivasan, and Ganesh Ramakrishnan. What kinds of relational features are useful for statistical learning? In *ILP*, 2012.
21. A.D. Shapiro. *Structured Induction in Expert Systems*. Addison-Wesley, Wokingham, 1987.
22. David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
23. 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.
24. A. Srinivasan. The Aleph Manual. Available at http://www.comlab.ox.ac.uk/oucl/ research/areas/machlearn/Aleph/, 1999.
25. 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.
26. Ashwin Srinivasan and Ganesh Ramakrishnan. Parameter screening and optimisation for ILP using designed experiments. *Journal of Machine Learning Research*, 12:627–662, 2011.
27. 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.
28. F. Zelezny, A. Srinivasan, and C.D. Page. Randomised Restarted Search in ILP. *Machine Learning Journal*, 64(1,2,3), 2006.
29. 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.