Advances in Cognitive Systems 6 (2018) 1-20 Submitted 6/2018; published 8/2018

Inductive Learning of Answer Set Programs from Noisy Examples

Mark Law MARK.LAW09@IMPERIAL.AC.UK
Alessandra Russo A.RUSSO@IMPERIAL.AC.UK
Krysia Broda K.BRODA@IMPERIAL.AC.UK
Department of Computing, Imperial College London, London, SW7 2AZ, United Kingdom

Abstract

In recent years, non-monotonic Inductive Logic Programming has received growing interest. Specifically, several new learning frameworks and algorithms have been introduced for learning under the answer set semantics, allowing the learning of common-sense knowledge involving defaults and exceptions, which are essential aspects of human reasoning. In this paper, we present a noise-tolerant generalisation of the *learning from answer sets* framework. We evaluate our ILASP3 system, both on synthetic and on real data sets, represented in the new framework. In particular, we show that on many of the data sets ILASP3 achieves a higher accuracy than other ILP systems that have previously been applied to the data sets, including a recently proposed differentiable learning framework.

1. Introduction

The ultimate aim of cognitive systems research is to achieve human-like intelligence. People are capable of many cognitive activities, such as learning from past experience, predicting outcomes of actions based on what they have learned, and reasoning using this learned knowledge. Each of these cognitive processes uses existing knowledge and generates new knowledge. They are underpinned by our ability to perform *inductive reasoning*, one of our most important high-level cognitive functions. Inductive reasoning is a complex process by which new knowledge is inferred from a series of observations in a way that can be transferred from past experiences to new situations. When performing inductive reasoning, observations perceived through the environment are often noisy and the existing knowledge that we use during the reasoning process is also limited and incomplete. The human inductive reasoning process is therefore capable of handling noise in the observations, reasoning with incomplete and defeasible knowledge, applying knowledge learned in one scenario to many other scenarios, and learning complex knowledge expressed in terms of rules, constraints and preferences that can be communicated to others.

To realise cognitive systems able to perform human-like inductive reasoning, machine learning solutions have to meet the above properties. Research in machine learning has yielded approaches and systems that, although capable of identifying patterns in data sets consisting of millions of (noisy) data points, cannot express the learned knowledge in a form that could be understood by a human. Moreover, their learned knowledge can only be used in exactly the scenario in which it was learned: for example, a system trained to play Go on a standard `19x19` board may not perform very well at Go played on a `20x20` board. Lack of interpretability and transferability of the learned

knowledge make these approaches far from human cognition. On the other hand, Inductive Logic Programming (ILP: Muggleton, 1991) has been shown to be suited for learning knowledge that can be understood by humans and applied to new scenarios. Although approaches for performing ILP in the context of noisy examples have been presented in the literature (e.g., Sandewall & Jansson, 1993; McCreath & Sharma, 1997; Oblak & Bratko, 2010), many existing ILP systems can only learn knowledge expressed as definite logic programs, so they are not capable of learning common-sense knowledge involving defaults and exceptions, which are essential aspects of human reasoning. This type of knowledge can be modelled using *negation as failure*.

Recently, ILP has been extended to enable learning programs containing negation as failure (e.g., Ray, 2009; Sakama & Inoue, 2009), and interpreted under the answer set semantics (Gelfond & Lifschitz, 1988). In particular, our recent results in inductive learning of answer set programs (ILASP: Law et al., 2014, 2016) have demonstrated the ability to support automated acquisition of complex knowledge structures in the language of Answer Set Programming (ASP). The theoretical framework underpinning ILASP, called *Learning from Answer Sets* (LAS), enables the learning of constraints, preferences and non-deterministic concepts. For instance, LAS can learn the concept that a coin may non-deterministically land on either heads or tails, but never both.

When learning, humans are also capable of disregarding information that does not fit the general pattern. Any cognitive system that aims to mimic human-level learning should therefore be capable of learning in the presence of *noisy* data. A realistic approach to cognitive knowledge acquisition is therefore the learning of knowledge that covers the *majority* of the examples, but which at the same time weights coverage against its complexity. In this paper, we present a noise tolerant extension of our LAS framework, *Learning from noisy answer sets* (ILP^{noise}_{LOAS}) and show that our ILASP3 system is capable of learning complex knowledge from noisy data in an effective and scalable way. A collection of data sets, ranging from synthetically generated to real data sets, is used to evaluate the performance of the system with respect to the percentage of noise in the examples and to compare it to existing ILP systems. Specifically, we consider two classes of synthetically generated data sets, called Hamiltonian and Journey preferences, and show that ILASP3 is able in both cases to achieve a high accuracy (of well over 90%), even with 20% of the examples labelled incorrectly. We also evaluate ILASP3 on data sets concerning learning event theories (Katzouris et al., 2016), sentence chunking (Agirre et al., 2016), preference learning (Kamishima et al., 2010; Abbasnejad et al., 2013) and the synthetic data sets of Evans and Grefenstette (2018). Our results show that in most cases the ability of ILASP3 to compute *optimal* solutions for a given learning task allows it to reach higher accuracy than the other systems, which do not guarantee the computation of an optimal solution.

Next, in Section 2, we review relevant background material. Section 3 introduces our new framework for learning ASP from noisy examples; Section 4 discusses the ILASP algorithms; Sections 5 and 6 present an extensive evaluation of our ILASP3 system; and finally, we conclude with a discussion of related and future work.

2. Background

We briefly introduce basic notions and terminologies used throughout the paper. Given any atoms $\mathtt{h}, \mathtt{h}_1, \ldots, \mathtt{h}_k, \mathtt{b}_1, \ldots, \mathtt{b}_n, \mathtt{c}_1, \ldots, \mathtt{c}_m$, a *normal rule* is of the form $\mathtt{h}\ \mathtt{:-}\ \mathtt{b}_1, \ldots, \mathtt{b}_n,\ \mathtt{not}\ \mathtt{c}_1, \ldots,\ \mathtt{not}\ \mathtt{c}_m$, where "not" is negation as failure, h is the head of the rule and $\mathtt{b}_1, \ldots, \mathtt{b}_n,\ \mathtt{not}\ \mathtt{c}_1, \ldots,\ \mathtt{not}\ \mathtt{c}_m$ is the body of the rule. For example, $\mathtt{fly(X)\ :-\ bird(X),\ not\ ab(X)}$ is a normal rule stating that any bird can fly, unless it is abnormal. The negated condition $\mathtt{not\ ab(X)}$ is assumed to hold unless there is a way of proving $\mathtt{ab(X)}$ for some value of X. So, the normal rule essentially models that by default, birds can fly, unless there is a proof that the bird is abnormal. ASP programs include three other types of rule: choice rules, hard and weak constraints. A *choice rule* is of the form $\mathtt{l}\{\mathtt{h}_1, \ldots, \mathtt{h}_k\}\mathtt{u}\ \mathtt{:-}\ \mathtt{b}_1, \ldots, \mathtt{b}_n,\ \mathtt{not}\ \mathtt{c}_1, \ldots,\ \mathtt{not}\ \mathtt{c}_m$, where $\mathtt{l}$ and $\mathtt{u}$ are integers and $\mathtt{l}\{\mathtt{h}_1, \ldots, \mathtt{h}_k\}\mathtt{u}$ is called an *aggregate*. A *hard constraint* is of the form $\mathtt{:-}\ \mathtt{b}_1, \ldots, \mathtt{b}_n,\ \mathtt{not}\ \mathtt{c}_1, \ldots,\ \mathtt{not}\ \mathtt{c}_m$ and a *weak constraint* is of the form $:\sim\ \mathtt{b}_1, \ldots, \mathtt{b}_n,\ \mathtt{not}\ \mathtt{c}_1, \ldots,\ \mathtt{not}\ \mathtt{c}_m.[\mathtt{w@l}, \mathtt{t}_1, \ldots, \mathtt{t}_k]$ where $\mathtt{w}$ and $\mathtt{l}$ are terms specifying *weight* and *priority level*, and $\mathtt{t}_1, \ldots, \mathtt{t}_k$ are terms.

The *Herbrand Base* of an ASP program P, denoted HB_P, is the set of ground (variable free) atoms that can be formed from predicates and constants in P. Subsets of HB_P are called (Herbrand) interpretations of P. The semantics of ASP programs P are defined in terms of *answer sets* – a special[1] subset of interpretations of P, denoted as $AS(P)$, that satisfy every rule in P. Given an answer set A, a ground normal or choice rule is satisfied if the head is satisfied by A whenever all positive atoms and none of the negated atoms of the body are in A, that is when the body is satisfied. A ground aggregate $\mathtt{l}\{\mathtt{h}_1, \ldots, \mathtt{h}_k\}\mathtt{u}$ is satisfied by an interpretation I iff $\mathtt{l} \leq |I \cap \{\mathtt{h}_1, \ldots, \mathtt{h}_k\}| \leq \mathtt{u}$. So, informally, a ground choice rule is satisfied by an answer set A if whenever its body is satisfied by an answer set A, a number between $\mathtt{l}$ and $\mathtt{u}$ (inclusive) of the atoms in the aggregate are also in A. A ground constraint is satisfied when its body is not satisfied. A constraint therefore has the effect of eliminating all answer sets that satisfy its body. Weak constraints do not affect what is, or is not, an answer set of a program P. Instead, they create an ordering $\succ_P$ over $AS(P)$ specifying which answer sets are "preferred" to others. Informally, at each *priority level* $\mathtt{l}$, satisfying weak constraints with level l means discarding any answer set that does not minimise the sum of the weights of the ground weak constraints (with level $\mathtt{l}$) whose bodies are satisfied. Higher levels are minimised first. For example, the two weak constraints $:\sim \mathtt{mode(L, walk)}, \mathtt{distance(L, D)}.[\mathtt{D@2, L}]$ and $:\sim \mathtt{cost(L, C)}.[\mathtt{C@1, L}]$ express a preference ordering over alternative journeys. The first constraint (at priority 2) expresses that the total walking distance (the sum of the distances of journey legs whose mode of transport is $\mathtt{walk}$) should be minimised, and the second constraint expresses that the total cost of the journey should be minimised. As the first weak constraint has a higher priority level than the second, it is minimised first – so given a journey j_1 with a higher cost than another journey j_2, j_1 is still preferred to j_2 so long as the walking distance of j_1 is lower than that of j_2. The set $ord(P)$ captures the ordering of interpretations induced by P and generalises the $\succ_P$ relation, so it not only includes $\langle A_1, A_2, < \rangle$ if $A_1 \succ_P A_2$, but includes tuples for each binary comparison operator ($<$, $>$, $=$, $\leq$, $\geq$ and $\neq$).

A *partial interpretation*, e_{pi}, is a pair of sets of ground atoms $\langle e^{inc}, e^{exc}_{pi} \rangle$. An interpretation I *extends* e_{pi} iff $e^{inc}_{pi} \subseteq I$ and $e^{exc}_{pi} \cap I = \emptyset$. Examples for learning come in two forms: *context-*

1. For a formal definition of answer sets of the programs in this paper see Law et al. (2015c).

dependent partial interpretations (CDPIs) and *context-dependent ordering examples* (CDOEs). A CDPI example e is a pair $\langle e_{pi}, e_{ctx} \rangle$, where e_{pi} is a partial interpretation and e_{ctx} is a program with no weak constraints called the context of e. A program P is said to *bravely accept* e if there is at least one answer set A of $P \cup e_{ctx}$ that extends e_{pi} – such an A is called an accepting answer set of P wrt e. Essentially, a CDPI says that the learned program, together with the context of e, should bravely[2] entail all inclusion atoms and none of the exclusion atoms of e. CDPIs can be used for *classification* tasks, as they specify that given contexts should entail given conjunctions of atoms. But as learned programs may have multiple answer sets, accepting a CDPI may require additional assumptions to be made. A CDOE o is a tuple $\langle e_1, e_2, op \rangle$, where the first two elements are CDPIs and op is a binary comparison operator. A program P is said to *bravely respect* o if there is a pair of accepting answer sets, A_1 and A_2, of P wrt e_1 and e_2, respectively, such that $\langle A_1, A_2, op \rangle \in ord(P)$. P is said to *cautiously respect* o if for every pair, A_1 and A_2, of accepting answer sets of P (wrt e_1 and e_2, respectively), $\langle A_1, A_2, op \rangle \in ord(P)$. CDOEs enable *preference learning* as they specify which answer sets should be prefered to other answer sets.

An $ILP^{context}_{LOAS}$ task T consists of an ASP background knowledge B, a hypothesis space S_M, labelled CDPIs, E^+ (positive examples) and E^- (negative examples), and labelled CDOEs, O^b (brave orderings) and O^c (cautious orderings). S_M is the set of rules allowed in hypotheses. A hypothesis $H \subseteq S_M$ *covers* a positive (resp. negative) example e if $B \cup H$ accepts (resp. does not accept) e. H covers a brave (resp. cautious) ordering o if $B \cup H$ bravely (resp. cautiously) respects o. H is an inductive solution of T iff H covers every example in T.

3. Learning Framework

This section presents the ILP^{noise}_{LOAS} framework, which extends our previous (non-noisy) learning framework $ILP^{context}_{LOAS}$ (Law et al. (2016)), by allowing examples to be *weighted context-dependent partial interpretations* and *weighted context-dependent ordering examples*. These are essentially the same as CDPIs and CDOEs, but weighted with a notion of *penalty*. If a hypothesis does not cover an example, we say that it *pays the penalty* of that example. Informally, penalties are used to calculate the *cost* associated with a hypothesis for not covering examples. The cost function of a hypothesis H is the sum over the penalties of all of the examples that are not *covered* by H, augmented with the length of the hypothesis. The goal of ILP^{noise}_{LOAS} is to find a hypothesis that minimises the cost function over a given hypothesis space with respect to a given set of examples.

Definition 3.1. A *weighted context-dependent partial interpretation* e is a tuple $\langle e_{id}, e_{pen}, e_{cdpi} \rangle$, where e_{id} is a constant, called the *identifier* of e (unique to each example), e_{pen} is the penalty of e and e_{cdpi} is a context-dependent partial interpretation. The penalty e_{pen} is either a positive integer, or ∞. A program P *accepts* e iff it accepts e_{cdpi}. A *weighted context-dependent ordering example* o is a tuple $\langle o_{id}, o_{pen}, o_{ord} \rangle$, where o_{id} is a constant, called the *identifier* of o, o_{pen} is the penalty of o and o_{ord} is a CDOE. The penalty o_{pen} is either a positive integer, or ∞. A program P *bravely* (resp. *cautiously*) *respects* o iff it bravely (resp. cautiously) respects o_{ord}.

2. A program P *bravely entails* an atom `a` if there is at least one answer set of P that contains a.

In learning tasks without noise, each example must be covered by any inductive solution. However, when examples are noisy (i.e., they have a weight), inductive solutions need not cover every example, but they incur penalties for each uncovered example. Multiple occurrences of the same CDPI example have different identifiers. So hypotheses that do not cover that example will pay the penalty multiple times (for instance, if a CDPI occurs twice then a hypothesis will have to pay twice the penalty for not covering it). In most of the learning tasks presented in this paper, all examples have the same penalty. In some cases, however, penalties are used to simulate *oversampling*; for example, in tasks with far more positive examples than negative examples, we may choose to give the negative examples a higher weight – otherwise it is likely that the learned hypothesis will treat all negative examples as noisy.

Our learning task with noisy examples consists of an ASP background knowledge, weighted CDPI and CDOE examples and a hypothesis space,[3] which defines the set of rules allowed to be used in constructing solutions of the task. These tasks are *supervised* learning tasks, as all examples are labelled, as positive/negative, or with an operator in the case of the ordering examples.

Definition 3.2. An ILP^{noise}_{LOAS} task T is a tuple of the form $\langle B, S_M, \langle E^+, E^-, O^b, O^c\rangle\rangle$, where B is an ASP program, S_M is a hypothesis space, E^+ and E^- are sets of weighted CDPIs and O^b and O^c are sets of weighted CDOEs. Given a hypothesis $H \subseteq S_M$,

1. $uncov(H, T)$ is the set consisting of all examples $e \in E^+$ (resp. E^-) such that $B \cup H$ does not accept (resp. accepts) e and all ordering examples $o \in O^b$ (resp. O^c) such that $B \cup H$ does not bravely (resp. cautiously) respect o.
2. the penalty of H, denoted as $pen(H, T)$, is the sum $\sum_{e \in uncov(H,T)} e_{pen}$.
3. the score of H, denoted as $\mathcal{S}(H, T)$, is the sum $|H| + pen(H, T)$.
4. H is an inductive solution of T (written $H \in ILP^{noise}_{LOAS}(T)$) if and only if $\mathcal{S}(H, T)$ is finite.
5. H is an *optimal inductive solution* of T (written $H \in {}^*ILP^{noise}_{LOAS}(T)$) if and only if $\mathcal{S}(H, T)$ is finite and $\nexists H' \subseteq S_M$ such that $\mathcal{S}(H, T) > \mathcal{S}(H', T)$.

Examples with infinite penalty *must* be covered by any inductive solution, as any hypothesis that does not cover such an example will have an infinite score. An ILP^{noise}_{LOAS} task T is said to be *satisfiable* if $ILP^{noise}_{LOAS}(T)$ is non-empty. If $ILP^{noise}_{LOAS}(T)$ is empty, then T is said to be *unsatisfiable*. Theorem 3.1 shows that for propositional tasks (where all hypothesis spaces, contexts and background knowledge are propositional) the complexity of ILP^{noise}_{LOAS} is the same as $ILP^{context}_{LOAS}$ for the decision problems of *verification* – deciding if a given hypothesis is a solution of a given task – and *satisfiability* – deciding if a given task has any solutions – investigated in Law et al. (2018).

Theorem 3.1.

1. *Deciding verification for an arbitrary propositional ILP^{noise}_{LOAS} task is DP-complete*

3. For details of hypothesis spaces in this paper, see `https://www.doc.ic.ac.uk/~ml1909/ILASP/`.

2. *Deciding satisfiability for an arbitrary propositional ILP^{noise}_{LOAS} task is Σ^P_2-complete*

Like its predecessor $ILP^{context}_{LOAS}$, our new learning framework ILP^{noise}_{LOAS} for noisy examples is capable of learning complex human-interpretable knowledge, containing defaults, non-determinism, exceptions and preferences. The generalisation to allow penalties on the examples means that the new framework can be deployed in realistic settings where examples are not guaranteed to be correctly labelled. Theorem 3.1 shows that this generalisation does not come at any additional cost in terms of the computational complexity of important decision problems of the framework.

4. The ILASP System

ILASP (Inductive Learning of Answer Set Programs: Law et al., 2014, 2015a,b, 2016) is a collection of algorithms for solving LAS tasks. The general idea behind the ILASP approach is to transform a learning task into a meta-level ASP program, which can be iteratively solved (extending the program in each iteration) until the optimal answer sets of the program correspond to solutions of the learning task. Unlike many other ILP systems, such as Muggleton (1995), Ray (2009), and Kazmi et al. (2017), the ILASP algorithms are guaranteed to return an optimal solution of the input learning task (with respect to the cost function). This can of course mean that ILASP may take longer to compute a solution than *approximate* systems (which are not guaranteed to return an optimal solution); however, as we demonstrate in Section 6, the hypotheses found by ILASP are often more accurate than those found by approximate systems.

Each version of ILASP has aimed to address scalability issues of the previous versions. ILASP1 (Law et al., 2014) was a prototype implementation, with a major efficiency issue with respect to negative examples. ILASP2 (Law et al., 2015b) addressed this issue by introducing a notion of an *violating reason*. In each iteration, each answer set of the ILASP2 meta-level program T_{meta} contains a representation of a hypothesis which covers every positive example and every brave ordering example. An answer set representing a hypothesis that is not an inductive solution, contains a "reason" why at least one negative example or cautious ordering is not covered, which can be translated into an ASP representation that, when added to T_{meta}, rules out any hypothesis that is not a solution for this reason. This process is performed iteratively until no more violating reasons are detected. For full details of violating reasons, see Law et al. (2015b).

Both ILASP1 and ILASP2 scale poorly with respect to the number of examples, as the number of rules in the ground instantiation of their meta-level representation is proportional to the number of examples in the learning task. As many examples may be similar, and thus covered by the same hypotheses, in non-noisy tasks (where all examples must be covered), it is often sufficient to consider a small subset of the examples called a *relevant subset* of the examples. ILASP2i (Law et al., 2016) uses this property to further improve the scalability of ILASP2. It starts with an empty set of *relevant examples* RE, and, at each iteration, it calls ILASP2 on a learning task using only the examples in RE. The hypothesis returned by ILASP2 is guaranteed to cover the current relevant examples, but is not necessarily an inductive solution of the original task. So, if ILASP2 returns a hypothesis that does not cover at least one example, then an arbitrary uncovered example is added to RE and the next iteration is started. If no such example exists, then the hypothesis is returned as

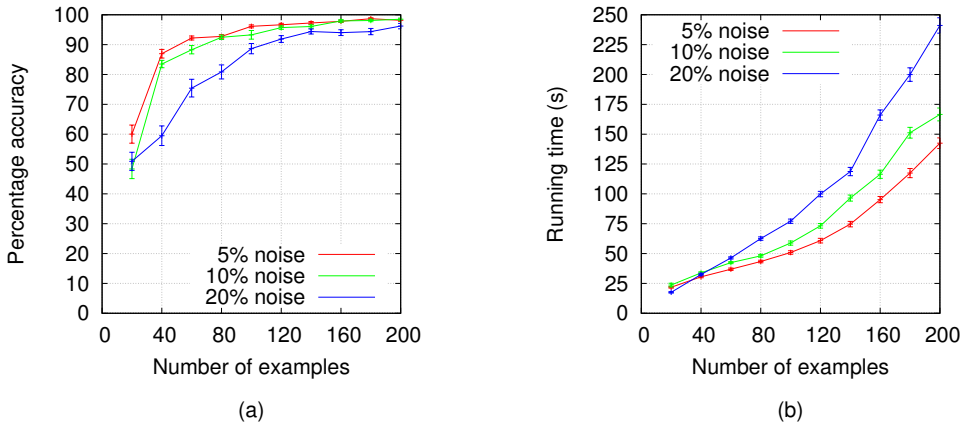

Figure 1. (a) the average computation time and (b) average accuracy of ILASP3 for the Hamilton learning task, with varying numbers of examples, and varying noise.

an optimal inductive solution of the original task. Law et al. (2016) showed that ILASP2i can be up to two orders of magnitude faster than ILASP2 on tasks with 500 (noise-free) examples.

Both ILASP2 and ILASP2i can be extended to solve ILP^{noise}_{LOAS} tasks; however, neither algorithm is well suited to solving tasks with a large number of noise examples with finite penalties. ILASP2 does not scale with respect to the number of examples (regardless of whether examples have finite penalties), and the relevant example feature of ILASP2i is not equally effective when examples have penalties. One reason for this is that many noisy examples may have to be added to the relevant example set before the cost of not covering a particular class of relevant examples is enough to outweigh the cost of learning an extra rule in the hypothesis. The most recent ILASP algorithm, ILASP3, iteratively translates examples into *hypothesis constraints* – constraints on the structure of a hypothesis that are satisfied if and only if the hypothesis covers the example. This leads to a much more compact meta-level program, defined in terms of these hypothesis constraints. Once hypothesis constraints have been computed for one example e, it is possible to compute the set of other examples (which have not yet been translated into hypothesis constraints) that are definitely not covered if e is not covered. This means that one relevant example can effectively have a much higher penalty than just the penalty for that example, meaning that the number of relevant examples that are needed in ILASP3 is often lower than those needed by ILASP2i.

5. Evaluation of ILASP3 on Synthetic Data Sets

In this section ILASP3 is evaluated on two synthetic data sets, the first of which is aimed at learning normal rules, choice rules and hard constraints, while the second is aimed at learning weak constraints. The value of using synthetic data sets is that we can control the amount of noise and investigate how the accuracy and running time of ILASP3 varies with the amount of noise.

5.1 Hamilton Graphs

In this experiment the task is to learn the definition of what it means for a graph to be Hamiltonian. This concept was chosen as it requires learning a hypothesis that contains choice rules, recursive rules and hard constraints, and also contains negation as failure. In these experiments, we show that ILASP3 could learn this hypothesis in the presence of noise, and we test how the running time of ILASP3 is affected by the number of examples and the number of incorrectly labeled examples.

For $n = 20, 40, \ldots, 200$, n random graphs of size one to four were generated, half of which were Hamiltonian. The graphs were labelled as either positive or negative, where positive indicates that the graph is Hamiltonian. The correct ASP representation of Hamiltonian and a discussion of the representation of examples in this task is given in Appendix A.

We ran three sets of experiments to evaluate ILASP3 on the Hamilton learning problem, with 5%, 10% and 20% of the examples being labelled incorrectly. In each experiment, an equal number of Hamiltonian graphs and non-Hamiltonian graphs were randomly generated and 5%, 10% or 20% of the examples were chosen at random to be labelled incorrectly. This set of examples were labelled as positive (resp. negative) if the graph was not (resp. was) Hamiltonian. The remaining examples were labelled correctly (positive if the graph was Hamiltonian; negative if the graph was not Hamiltonian). Figure 5.1 shows the average accuracy and running time of ILASP3 with up to 200 example graphs. Each experiment was repeated 50 times (with different randomly generated examples). In each case, the accuracy was tested by generating a further 1,000 graphs and using the learned hypothesis to classify the graphs as either Hamiltonian or non-Hamiltonian (based on whether the hypothesis was satisfiable when combined with the representation of the graph).

The experiments show that on average ILASP3 is able to achieve a high accuracy (of well over 90%), even with 20% of the examples labelled incorrectly. A larger percentage of noise means that ILASP3 requires a larger number of examples to achieve a high accuracy. This is to be expected, as with few examples, the hypothesis is more likely to "overfit" to the noise, or pay the penalty of some non-noisy examples. With large numbers of examples, it is more likely that ignoring some non-noisy examples would mean not covering others, and thus paying a larger penalty. The computation time rises in all three graphs as the number of examples increases. This is because larger numbers of examples are likely to require larger numbers of iterations of the ILASP3 algorithm. Similarly, more noise is also likely to mean a larger number of iterations.

5.2 Noisy Journey Preferences

The experiment in this section is a noisy extension of the journey preference learning setting used in Law et al. (2016), where the goal is to learn a user's preferences from a set of ordered pairs of journeys. These experiments aim to show that ILASP3 is capable of preference learning in the presence of noise, and to test how the accuracy and running time of ILASP3 are affected by the numbers of examples and the proportion of examples which are incorrectly labelled.

In each experiment, we selected a "target hypothesis" consisting of between one and three weak constraints from a hypothesis space of weak constraints (discussed in Appendix A). For each set of weak constraints, we then ran learning tasks with 0, 20, ..., 200 examples and with 5%, 10% and 20% noise. The ordering examples for these learning tasks were generated from the weak

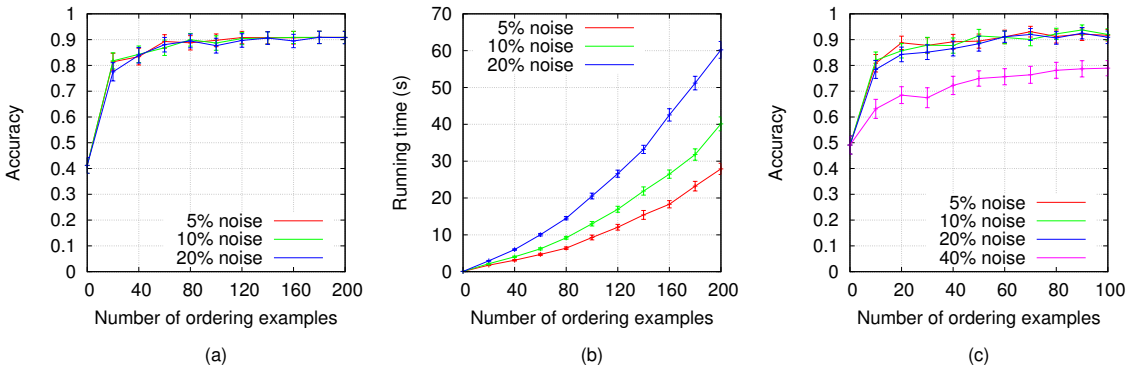

Figure 2. (a) and (c) the average accuracy and (b) average computation time of ILASP3 for the journey preference learning task, with varying numbers of examples, and varying noise. Each point in the graphs is an average over 50 different tasks.

constraints such that half of the (brave) ordering examples represented pairs of journeys J_1 and J_2 where J_1 was strictly preferred to J_2, given the weak constraints, and the other half represented journeys such that J_1 was equally preferred to J_2. Depending on the level of noise, either 5%, 10% or 20% of the examples were given with the wrong operator ($>$ instead of $<$ and $\neq$ instead of $=$). Each ordering example was given a penalty of one.

The results (Figure 2 (a)) show that even with 20% noise, ILASP3 was able to learn hypotheses with an average accuracy of over 90%. There was not much difference between ILASP3's accuracy with 5%, 10% and 20% noise, although the noisier tasks had a higher computation time (this is shown in Figure 2 (b)), as in general ILASP3 requires more iterations on noisier tasks. Even with 20% noise and 200 ordering examples, ILASP3 terminated in just over 60 seconds on average.

As the results for 20% noise were so close to the results for 5% noise, we ran a further set of examples to check that there was some limit to the level of noise where ILASP3 would no longer learn such an accurate hypothesis.[4] In this second set of experiments, we tested ILASP3 with up to 40% noise, and investigated with 0, 10, ..., 100 examples. With 40% noise, the accuracy was lower, but ILASP still achieved an average accuracy of just under 80%.

These experiments show that ILASP3 is able to accurately learn a set of weak constraints from examples of the orderings of answer sets given by these weak constraints, even when 20% of the orderings are incorrect. Although the running time of ILASP3 is affected by the number of examples and the proportion of incorrectly labelled examples, ILASP3 is able to find an optimal solution in an average of 60 seconds, even with 200 ordering examples, 20% of which are incorrectly labelled. Learning weak constraints is significant, as they can be used to represent user preferences. In Sections 6.3 and 6.4, we apply ILASP3 on two real preference learning data sets.

4. If ILASP could achieve such a high accuracy, even with very high levels of noise, then this would indicate that the hypothesis space was too restrictive, and it was impossible to learn anything other than an accurate hypothesis.

6. Comparison with Other Systems

The experiments in this section use data sets that have previously been used to evaluate other ILP systems in the presence of noise. Unlike ILASP3, none of the systems we compare with aim to find optimal solutions. The aim of this set of experiments is therefore to test whether finding optimal solutions leads to any gain in accuracy over systems which may return sub-optimal solutions.

6.1 CAVIAR Data Set

In this experiment ILASP3 was tested on the recent CAVIAR data set that has been used to evaluate the OLED (Katzouris et al., 2016) system, which is an extension of the XHAIL (Ray, 2009) algorithm, for learning Event Calculus (Kowalski & Sergot, 1986) theories. The data set contains data gathered from a video stream. Information such as the positions of people has been extracted from the stream, and humans have annotated the data to specify when any two people are interacting. Specifically, we consider a task from Katzouris et al. (2016), in which the aim is to learn rules to define initiating and terminating conditions for two people meeting. In the evaluation of the OLED system, examples were generated for every pair of consecutive timepoints t and $t+1$. Each example is a pair $\langle N \cup A_t, A_{t+1} \rangle$, where N is the "narrative" at time t (a collection of information about the people in the video stream, such as their location and direction), and A_i is the "annotation" at time i (exactly which pairs of people in the video have been labelled as meeting). This is very simple to express using context-dependent examples. The context of an example is simply the narrative and annotation of time t together with a set of constraints that enforce that the meetings at time t are exactly those in the annotation. The aim of this experiment is to compare ILASP3 to OLED, which was specifically designed to solve this kind of task efficiently. We aimed to discover whether ILASP3 is able to find better quality hypotheses than OLED (in terms of the F_1 measure used to evaluate the hypotheses found by OLED), and whether ILASP3's guarantee of finding an optimal solution comes at a cost in terms of running time.

In total there are 24,530 consecutive pairs in the data set.[5] We performed ten-fold cross validation by randomly partitioning the data set. As there were only twenty-two timepoints where the group of people meeting was different to the timepoint before, these examples were given a high penalty (of 100). Effectively this is the same as oversampling this class of examples. If all examples had been given a penalty of one, then ILASP3 would have likely learned the empty hypothesis, as the twenty-two examples in a task of many thousands of examples would likely be treated as noise.

We compare ILASP3 to OLED on the measures of precision, recall and the F_1 score.[6] ILASP3 achieved a precision of 0.832 and a recall of 0.853, giving an F_1 score of 0.842, compared with OLED's precision of 0.678 and recall of 0.953, with an average F_1 score of 0.792. ILASP3's average running time was significantly higher at 576.3s compared with OLED's 107s. This is explained by the fact that the OLED system computes hypotheses through theory revision, iteratively processing

5. We used the data from `users.iit.demokritos.gr/~nkatz/OLED-data/caviar.json.tar.gz`

6. Let tp, tn, fp, fn represent the number of true positives, true negatives, false positives and false negatives achieved by a classifier on some test data. The *precision* of the classifier (on this test data) is equal to $\frac{tp}{tp+fp}$ and the *recall* is equal to $\frac{tp}{tp+fn}$. The F_1 score is equal to $\frac{2 \times precision \times recall}{precision+recall}$.

examples in sequence to continuously revise its hypothesis. This means that, unlike ILASP3, OLED is not guaranteed to find an optimal solution of a learning task.

We note several key differences between our experiments and those reported in Katzouris et al. (2016). Firstly, to reduce the number of irrelevant answer sets (which lead to slow computation), we constrained the hypothesis space stating that rules for $\texttt{terminatedAt}(\texttt{meeting}(\texttt{V1},\texttt{V2}),\texttt{T})$ had to contain $\texttt{holdsAt}(\texttt{meeting}(\texttt{V1},\texttt{V2}),\texttt{T})$ in the body, which ensures that a fluent can only be terminated if it is currently happening. Similarly, any rule for $\texttt{initiatedAt}(\texttt{meeting}(\texttt{V1},\texttt{V2}),\texttt{T})$ had to contain $\texttt{not holdsAt}(\texttt{meeting}(\texttt{V1},\texttt{V2}),\texttt{T})$ in the body. OLED does not employ this constraint, but when processing an example pair of time points, only considers learning a new rule for `initiatedAt`, for example, if two people are meeting at time $t+1$, but not at time t. The second difference in our experiment is that ILASP3 enumerates the hypothesis space in full. As the hypothesis space in this task is potentially very large, several "common sense" constraints were enforced on the hypothesis space; for instance, two people cannot be both close to and far away from each other at the same time (rules with both conditions in the body were not generated). In total, the hypothesis space contained 3,370 rules. OLED does not enumerate the hypothesis space in full, but uses an approach similar to XHAIL, and derives a "bottom clause" from the background knowledge and the example. In most cases (unless there is noise in the narrative, suggesting that two people are both close to and far away from each other) OLED will therefore only consider rules that respect the "common sense" constraints, as other rules would not be derivable.

This experiment has shown that, at least on this data set, ILASP3's guarantee of finding an optimal solution can lead to better quality hypotheses than those found by OLED; however, this quality comes at a cost, as ILASP3's running time is significantly higher than OLED's.

6.2 Sentence Chunking

In Kazmi et al. (2017), the Inspire system was evaluated on a sentence chunking (Tjong Kim Sang & Buchholz, 2000) data set (Agirre et al., 2016). The task in this setting is to learn to split a sentence into short phrases called chunks. For instance, according to the data set (Agirre et al., 2016), the sentence "Thai opposition party to boycott general election." should be split into the three chunks "Thai opposition party", "to boycott" and "general election". Kazmi et al. (2017) describe how to transform each sentence into a set of facts consisting of part of speach (POS) tags. We use each of these sets of facts as the context of a context dependent example. In Inspire (which is a brave induction system), the facts are all put into the background knowledge. The task is to learn a predicate `split/1`, which expresses where sentences should be split. Inspire does not guarantee finding an optimal solution. The hypothesis can be suboptimal for three reasons: firstly, the abductive phase may find an abductive solution which leads to a suboptimal inductive solution; secondly, Inspire's pruning may remove some hypotheses from the hypothesis space; and finally, Inspire was set to interrupt the inductive phase after 1,800 seconds, returning the most optimal hypothesis found so far. In these experiments, we aimed to show that ILASP3's guarantee of finding an optimal solution leads to a better quality hypotheses than Inspire's approximations, and if so, whether ILASP3's running time was higher Inspire's timeout of 1,800s.

Note that the Inspire tasks in Kazmi et al. (2017) group the multiple `split` examples for a chunk into a single example (using a `goodchunk` predicate); for example, the background knowledge may

Table 1. F_1 scores for Inspire and ILASP3 and ILASP3's average running time on the sentence chunking tasks.

		Inspire F_1 score	ILASP F_1 score	ILASP time (s)
100 examples	Headlines S1	73.1	74.2	351.2
	Headlines S2	70.7	73.0	388.3
	Images S1	81.8	83.0	144.9
	Images S2	73.9	75.2	187.2
	Students S1/S2	67.0	72.5	264.5
500 examples	Headlines S1	69.7	75.3	1,616.6
	Headlines S2	73.4	77.2	1,563.6
	Images S1	75.3	80.8	929.8
	Images S2	71.3	78.9	935.8
	Students S1/S2	66.3	75.6	1,451.3

contain a rule `goodchunk(1):-split(1), not split(2), not split(3),split(4)` expressing that there is a chunk between words one and four of a sentence. It is noted in Kazmi et al. (2017) that this increased performance. This is because there is no benefit in covering some of the `split` atoms that make up a chunk, as hypotheses are tested over full chunks rather than splits. In our framework, we represent this directly with no need for the `goodchunk` rules, with the individual split atoms being inclusions and exclusions in the partial interpretation of the example and the penalty being on the full example. In our learning task, the example corresponding to the rule for `goodchunk(1)` would have the partial interpretation $\langle\{$`split(1), split(4)`$\}, \{$`split(2), split(3)`$\}\rangle$. In Kazmi et al. (2017), eleven-fold cross validation was performed on five different data sets, with 100 and 500 examples. As Inspire has a parameter which determines how aggressive the pruning should be, Kazmi et al. (2017) present several F_1 scores, for different values of this parameter. Each entry for Inspire in Table 1 is Inspire's best F_1 score over all pruning parameters.

Inspire approximates the optimal inductive solution of the task and has a timeout of 1,800s on the inductive phase – in contrast, ILASP3 terminated in less than 1,800 seconds on every task. ILASP3 achieved a higher average F_1 score than Inspire on every one of the ten tasks. This shows that computing the optimal inductive solution of a task can lead to a better quality hypothesis than approximating the optimal solution. Note that for four out of the five data sets, Inspire performs better with 100 examples than with 500 examples. A possible explanation for this is that with more examples, Inspire does not get as close to the optimal solution as it does with fewer examples, thus leading to a lower F_1 score on the test data. With 500 examples, ILASP3 does take longer to terminate than it does for 100 examples, but in four out of the five cases, ILASP's average F_1 score is higher, confirming the expected result that more data should tend to lead to a better hypothesis.

6.3 Car Preference Learning

We tested ILASP3's ability to learn real user preferences with the car preference data set from Abbasnejad et al. (2013). This data set consists of responses from 60 different users, who were each asked to give their preferences about ten cars. They were asked to order each (distinct) pair of cars, leading to 45 orderings. The cars had four attributes, shown in Table 2 (a). Through this

Table 2. (a) The attributes of the car preference data set, along with the possible range of values for each attribute. The integer next to each value is how that value is represented in the data. (b) The accuracy results of ILASP3 compared with the three methods in Qomariyah & Kazakov (2017) on the car preference data set.

(a)

Attribute	Values
Body type	`sedan`(1), `suv`(2)
Transmission	`manual`(1), `automatic`(2)
Engine Capacity	2.5L, 3.5L, 4.5L, 5.5L, 6.2L
Fuel Consumed	`hybrid`(1), `non_hybrid`(2)

(b)

Method	Accuracy
SVM	0.832
DT	0.747
Aleph	0.729
ILASP3 A	0.880
ILASP3 B	0.863

experiment, we aim to show that ILASP3 is capable of learning real user preferences, encoded as weak constraints. There is not much work on applying ILP systems to preference learning, but one such work (Qomariyah & Kazakov, 2017) applied the Aleph (Srinivasan, 2001) system to the car preference data set. Aleph is not guaranteed to find an optimal solution,[7] and is only capable of learning rules (and not of learning weak constraints). Qomariyah & Kazakov (2017) used Aleph to learn rules defining the predicate `bt`$/2$, where `bt`$(\texttt{c}_1, \texttt{c}_2)$ represents that $\texttt{c}_1$ is preferred to $\texttt{c}_2$. For comparison, we present the results of Qomariyah & Kazakov (2017) on this data set.

Our initial experiment was based on an experiment in Qomariyah & Kazakov (2017), where the Aleph (Srinivasan, 2001) system was used to learn the preferences of each user in the data set and compared with support vector machines (SVM) and decision trees (DT). Ten-fold cross validation was performed for each of the 60 users on the 45 orderings. In each fold, 10% of the orderings were omitted from the training data and used to test the learned hypothesis. The flaw in this approach is that the omitted examples will often be implied by the rest of the examples (i.e., if $a \prec b$ and $b \prec c$ are given as examples it does not make sense to omit $a \prec c$). For this reason, we also experimented with leaving out all the examples for a single car in each fold (i.e., every pair that contains that car), and using these examples to test (again leading to ten folds). This new task corresponds to learning preferences from a complete ordering of nine cars, and testing the preferences on an unseen car.

Table 2 (b) shows the accuracy of the approach in Qomariyah & Kazakov (2017) and ILASP3 accuracy on the two versions of the experiment. The easier task (with 10% of the orderings omitted) is denoted as experiment A in the table, and the harder task is denoted as experiment B. In fact, even on the harder version of the task, ILASP3 performs better than the approaches in Qomariyah & Kazakov (2017) perform on the easier version of the task. In one fold for the first user (in experiment A), ILASP3 learns the following weak constraints: $:\sim$ `fuel`$(2).[1@4]$; $:\sim$ `body`$(1), $`transmission`$(2).[-1@3]$; $:\sim$ `engine_cap`$(\texttt{V0}).[\texttt{V0}@2, \texttt{V0}]$; $:\sim$ `body`$(1).[-1@1]$. This hypothesis corresponds to the following set of prioritised preferences (ordered from most to least important): the user (1) prefers hybrid cars to non-hybrid cars; (2) likes automatic sedans; (3) would like to minimise the engine capacity of the car; and (4) prefers sedans to SUVs.

7. Aleph processes the examples sequentially, and searches for the best clause to add in each iteration (in terms of coverage). Although each iteration adds the best clause, this may still lead to a sub-optimal hypothesis overall.

Table 3. (a) the attributes of the SUSHI preference data set, along with the range of values for each attribute, and (b) the average accuracy of ILASP3 compared with the methods used in Qomariyah & Kazakov (2017).

(a)

Attribute	Values
Style	`maki(0)`, `non_maki(1)`
Major group	`seafood(0)`, `non_seafood(1)`
Minor group	`0`, ..., `11`
Oiliness	$[0, 4]$
Frequency Eaten	$[0, 3]$
Normalised Price	$[0, 5]$
Frequency Sold	$[0, 1]$

(b)

Method	Accuracy
SVM	0.76
DT	0.81
Aleph	0.78
ILASP3	0.84

The noise in this experiment comes from the fact that some of the answers given by participants in the survey may contradict other answers. Some participants gave inconsistent orderings (breaking transitivity) meaning that there is no set of weak constraints that covers every ordering example.

The results of these experiments have shown that ILASP3 is able to learn hypotheses that accurately represent real user preferences, even in the presence of noise. On average, ILASP3 learns a hypothesis with a higher accuracy than the hypothesis learned by Qomariyah & Kazakov (2017). This could be for two reasons: (1) the fact that Aleph might return a sub-optimal inductive solution; or (2), the representation of hypotheses as weak constraints allows for preferences to be expressed that cannot be expressed using the definite search space in Qomariyah & Kazakov (2017).

6.4 SUSHI Preference Learning

Another data set for preference learning is the SUSHI data set (Kamishima et al., 2010). The data set is comprised of peoples' preference orderings over different types of sushi. The purpose of these experiments is to show that ILASP3 is capable of learning weak constraints that accurately capture real user preferences. Qomariyah & Kazakov (2017) also tested their approach on these data sets, and we compare ILASP3's accuracy with their results in order to test whether the optimal solution found by ILASP3 is more accurate than their solutions.

Each type of sushi has several attributes, described in Table 3 (a). There is a mix of categorical and continuous attributes. In the language bias for these experiments, the categorical attributes are used as constants, whereas the continuous attributes are variables that can be used as the weight of the weak constraint. This allows weak constraints to express that the continuous attributes should be minimised or maximised. The data set was constructed from a survey in which people were asked to order ten different types of sushi. This ordering leads to 45 ordering examples per person. This experiment is based on a similar experiment in Qomariyah & Kazakov (2017). For each of the first 60 people in the data set ten-fold cross validation was performed, omitting 10% of the orderings in each fold. This experiment suffers from the same flaw as Experiment A on the car data set in that some of the omitted examples may be implied by the training examples, but we give the results for a comparison to Qomariyah & Kazakov (2017). As shown in Table 3 (b), ILASP3 achieved an average accuracy of 0.84, comparing favourably to each result from Qomariyah & Kazakov (2017).

Although in this experiment each participant gave a consistent total ordering of the ten types of sushi, it might be the case that there is no hypothesis in the hypothesis space that covers all of the examples. This could be the case when we are not modelling a feature of the sushi that the participant considers to be important. For this reason, we treated this as a noisy learning setting, and used ILASP3 to maximise the coverage of the examples.

This experiment has shown that ILASP3 is capable of learning weak constraints that accurately capture users' preferences, and that ILASP3's approach of finding an optimal hypothesis comprising of weak constraints is (on average) more accurate than the approach of Qomariyah & Kazakov (2017), which finds a (potentially sub-optimal) set of definite clauses.

6.5 Comparison to δILP

Although the work in this paper concerns learning ASP programs from noisy examples, work has been done in the area of extending definite clause learning to handle noisy examples (for example, Sandewall & Jansson (1993); Srinivasan (2001); Oblak & Bratko (2010)). In Evans & Grefenstette (2018), it was claimed that ILP approaches are unable "to handle noisy, erroneous, or ambiguous data" and that "If the positive or negative examples contain any mislabelled data, [ILP approaches] will not be able to learn the intended rule". The experiments in this section aim to refute this claim.

To learn from noisy data, Evans & Grefenstette (2018) introduced the δILP algorithm, based on artificial neural networks. They demonstrated that δILP is able to achieve a high accuracy even with a large proportion of noise in the examples. Evans & Grefenstette (2018) evaluated δILP on six synthetic data sets, with noise ranging from 0% to 90%. In these experiments, we investigated the accuracy of ILASP3 on five of these six data sets.[8] In the original experiments, examples were atoms, and noise corresponded to swapping positive and negative examples. In each of the ILP^{noise}_{LOAS} tasks, we ensured that the hypothesis space was such that for each $H \subseteq S_M$, $B \cup H \cup e_{ctx}$ was stratified for each example e. This allowed atomic examples to be represented as (positive) partial interpretations – a positive example `e` was represented as a partial interpretation $\langle \{\texttt{e}\}, \emptyset \rangle$, and a negative example `e` was represented as a partial interpretation $\langle \emptyset, \{\texttt{e}\} \rangle$. Due to the differences in language biases used by ILASP and δILP, the hypothesis spaces of the two systems are not equivalent.

Due to the imbalance of positive and negative examples in many of the tasks, we weight the positive examples at $w \times |E^-|/(|E^+|+|E^-|)$ and the negative examples at $w \times |E^+|/(|E^+|+|E^-|)$, where in this experiment w is 100. The weight for each example class (positive or negative) is equal to w multiplied by the proportion of the whole set of examples which are in the other class. This "corrects" any imbalance between positive and negative examples (i.e., the penalty for not covering a proportion of the positive examples is the same as the penalty for not covering the same proportion of negative examples). The constant w can be thought of as the difference in importance between the hypothesis length and the number of examples covered. In these experiments we chose 100, as it is high enough to ensure that coverage is considered far more important than hypothesis length.

Figure 3 shows the mean squared error of the two systems, where the results for δILP are taken from Evans & Grefenstette (2018). In most tasks ILASP3 achieves similar results to δILP when

8. The authors of (Evans & Grefenstette, 2018) provided us with the training and test data for these five problems.

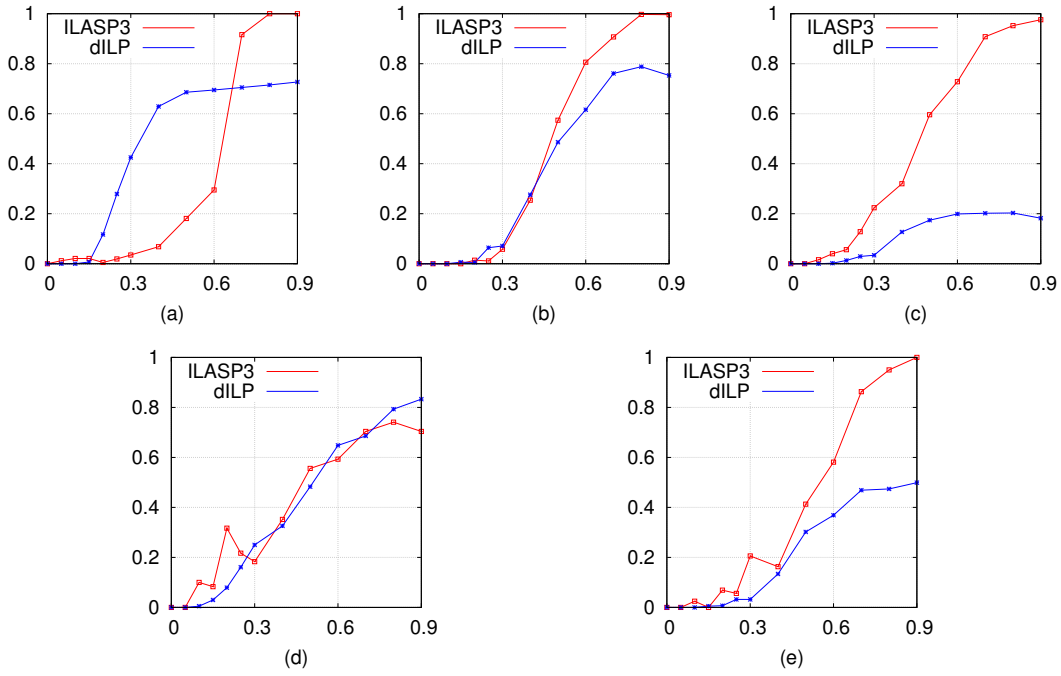

Figure 3. A comparison of δILP and ILASP3 on five data sets from Evans and Grefenstette (2018). Specifically the graphs correspond to the (a) predecessor, (b) less than, (c) member, (d) connected and (e) undirected edge experiments in Evans and Grefenstette (2018). In each graph, the x and y axes represent the noise level and mean squared error, respectively.

the noise is in the range of 0% to 40%. However, at the other end of the scale (with more than 50% noise), there are some tasks where ILASP3 finds hypotheses with close to 100% error, where δILP's error is much lower (less than 20% in the "member" problem). We argue that when the noisy examples outnumber the correctly labelled examples, the learner should start learning the negation of the target hypothesis; for instance, in the case of "less than", ILASP3 correctly learned the "greater than or equal to" relation. The ideal outcome of these kinds of experiments, where the proportion of noise is varied, is that the learner achieves close to 0% error until around 50% noise and close to 100% error thereafter. This is roughly what seems to happen for ILASP3 in the "predecessor", "less than", "member" and "undirected edge" experiments. In "predecessor", the graph is less symmetric, with the "crossover" from low to high error occurring later. This is likely because the hypothesis for "not predecessor" is longer than the hypothesis for "predecessor". The failure of δILP to get close to 100% error in many of the tasks (for example in "member", δILP has an error of less than 20% with the noise level at 90%) may indicate that the negation of the target concept is not representable given the language bias used by δILP in these experiments, instead of δILP being particularly robust to noise. In some cases (such as "member"), this is likely because the negation of the concept requires negation as failure (which is not supported by δILP), but in others such as "less than", the negation of the concept is expressible without negation as failure.

These results show that, on the ILP problems investigated by Evans & Grefenstette (2018), ILASP3 is certainly robust to noise, thus refuting their claim that ILP systems cannot handle noise.

7. Related Work

Several other ILP systems use ASP solvers in the search for hypotheses. For example, Balduccini (2007) presented an early system for learning action descriptions, where the search for inductive solutions is encoded in ASP. Many of these systems, such as those reported by Balduccini (2007), Athakravi et al. (2013), Bragaglia and Ray (2014), and Kazmi et al. (2017), operate under a *brave* semantics – the learned program should have at least one answer set that satisfies some given properties (such as covering examples). But our results on the generality of learning frameworks in Law et al. (2018) prove that there are ASP programs that can be learned by our framework and that cannot be learned by any of these systems. For example, brave induction systems cannot learn hard or weak constraints, no matter what examples are given.

In a different line of research, Sridharan et al. (2017) and Sridharan and Meadows (2017) present an architecture that combines relational reinforcement learning with ASP-based inference and decision tree induction to identify a set of candidate axioms. The candidates deemed to have the highest likelihood are then represented in an ASP program, which the system uses for planning.

Early approaches to relational learning (e.g., Langley, 1987; Mooney & Ourston, 1991; Cohen, 1995) were able to learn definite rules from noisy data. Mooney and Ourston (1991) presented an ILP system based on theory revision, where hypotheses are only modified if the modification leads to the additional coverage of more than one example. In practice, however, it is possible that given a large enough set of examples, two noisy examples may be covered by exactly the same class of hypotheses. Under the ILP^{noise}_{LOAS} approach, the penalty for not covering a set of examples which forms a small proportion of examples is low, even if there are multiple examples in this set. Cohen (1995) introduces algorithms which learn from noisy examples, learning one clause at a time. ILP systems which iteratively learn single clauses, removing covered positive examples after each iteration, are common when the target hypotheses are definite logic programs (with no negation), as the programs being learned are *monotonic*. Learning *non-monotonic* ASP programs with negation (allowing for the learning of exceptions) requires a different approach (Ray, 2009). This is because, due to the non-monotonicity of the learned programs, examples which are covered in one iteration may become uncovered when further rules are learned.

In order to search for good hypotheses, ILP systems often use a cost function, defined in terms of the coverage of the examples and the length of the hypothesis (e.g., Srinivasan, 2001; Muggleton, 1995; Bragaglia & Ray, 2014). When examples are noisy, this cost function is sometimes combined with a notion of maximum threshold, and the search is not for a hypothesis that minimises the cost function, but for a hypothesis that does not fail to cover more than a defined maximum threshold number of examples (e.g., Srinivasan, 2001; Oblak & Bratko, 2010; Athakravi et al., 2013). In this way, once an acceptable hypothesis (i.e., a hypothesis that covers a sufficient number of examples) is computed the system does not search for a better one. As such, the computational task is simpler, and therefore the time needed to compute a hypothesis is shorter, but there may be other hypotheses which have a lower cost. Furthermore, to guess the "correct" maximum threshold requires some

idea of how much noise there is in the given set of examples. For instance, one of the inputs to the HYPER/N (Oblak & Bratko, 2010) system is the proportion of noise in the examples. When the proportion of noise is unknown, too small a threshold could result in the learning task being unsatisfiable, or in learning a hypothesis that overfits the data. On the other hand, too high a threshold could result in poor accuracy, as the hypothesis may not cover many of the examples. Our ILP^{noise}_{LOAS} framework addresses the problem of computing optimal solutions (with respect to the cost function) and in doing so does not require knowledge a priori of the level of noise in the data. Note that optimal hypotheses are not *guaranteed* to outperform other hypotheses on unseen data, but based on the evidence (i.e., the training examples) they minimise the cost function, and so if the cost function is reasonable, they should be more likely to be correct. This can be seen in the sentence chunking experiments, where we used ILASP with the same cost function as Inspire (which does not guarantee minimising the cost function). In future work, we intend to explore alternative cost functions, and formalise what makes a cost function "reasonable" in a given learning setting.

8. Conclusion

Learning interpretable knowledge is a key requirement for cognitive systems that are required to communicate with each other or with humans. Our research has addressed the problem of learning ASP programs, which are capable of representing complex knowledge, such as defaults, exceptions and preferences. In practice, cognitive systems are required to learn knowledge from noisy data sources, where there is no guarantee that all examples are perfectly labelled.

This paper has presented the ILP^{noise}_{LOAS} framework for learning answer set programs from noisy examples, described the ILASP3 system that implements this framework, and reported experiments that evaluated its abilities. We used several synthetic data sets to show that ILASP3 can learn even in the presence of high proportions noisy examples. We also tested ILASP3's performance on several data sets used by other ILP systems. The results of these experiments show that, in most cases, ILASP3 can learn with a higher accuracy than the other systems, which, unlike ILASP3, are not guaranteed to find optimal solutions of the tasks. Although ILASP3 is a significant improvement on previous ILASP systems with respect to the running time on noisy tasks, some scalability issues remain, especially with the size of the hypothesis space. Every ILASP system begins by computing the hypothesis space in full, which limits the feasible size of the hypothesis space. In future work, we plan to design ILASP systems which do not begin by computing the hypothesis space in full.

Acknowledgements

We would like to thank the reviewers for their useful comments and suggestions.

References

Abbasnejad, E., Sanner, S., Bonilla, E. V., & Poupart, P. (2013). Learning community-based preferences via Dirichlet process mixtures of Gaussian processes. *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence* (pp. 1213–1219). Beijing, China: AAAI Press.

Agirre, E., Gonzalez Agirre, A., Lopez-Gazpio, I., Maritxalar, M., Rigau Claramunt, G., & Uria, L. (2016). Semeval-2016 task 2: Interpretable semantic textual similarity. *Proceedings of the Tenth International Workshop on Semantic Evaluation* (pp. 512–524). San Diego, CA: Association for Computational Linguistics.

Athakravi, D., Corapi, D., Broda, K., & Russo, A. (2013). Learning through hypothesis refinement using answer set programming. *Proceedings of the Twenty-Third International Conference on Inductive Logic Programming* (pp. 31–46). Rio de Janeiro, Brazil: Springer.

Balduccini, M. (2007). Learning action descriptions with A-Prolog: Action language C. *Proceedings of the 2007 AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning* (pp. 13–18). Palo Alto, CA: AAAI Press.

Bragaglia, S., & Ray, O. (2014). Nonmonotonic learning in large biological networks. *Proceedings of the Twenty-Fourth International Conference on Inductive Logic Programming* (pp. 33–48). Nancy, France: Springer.

Cohen, W. W. (1995). Fast effective rule induction. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 115–123). Tahoe City, CA: Morgan Kaufmann.

Evans, R., & Grefenstette, E. (2018). Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, *61*, 1–64.

Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. *Proceedings of the Fifth International Conference and Symposium on Logic Programming* (pp. 1070–1080). Seattle, WA: MIT Press.

Kamishima, T., Kazawa, H., & Akaho, S. (2010). A survey and empirical comparison of object ranking methods. In J. Fürnkranz & E. Hüllermeier (Eds.), *Preference learning*, 181–201. Berlin: Springer-Verlag.

Katzouris, N., Artikis, A., & Paliouras, G. (2016). Online learning of event definitions. *Theory and Practice of Logic Programming*, *16*, 817–833.

Kazmi, M., Schüller, P., & Saygın, Y. (2017). Improving scalability of inductive logic programming via pruning and best-effort optimisation. *Expert Systems with Applications*, *87*, 291–303.

Kowalski, R. A., & Sergot, M. J. (1986). A logic-based calculus of events. *New Generation Computing*, *4*, 67–95.

Langley, P. (1987). A general theory of discrimination learning. *Production system models of learning and development*, (pp. 99–161).

Law, M., Russo, A., & Broda, K. (2014). Inductive learning of answer set programs. *Proceedings of the Fourteenth European Conference on Logics in Artificial Intelligence* (pp. 311–325). Funchal, Madeira, Portugal: Springer.

Law, M., Russo, A., & Broda, K. (2015a). The ILASP system for learning answer set programs. Retrieved 30 July, 2018, from `https://www.doc.ic.ac.uk/~ml1909/ILASP`.

Law, M., Russo, A., & Broda, K. (2015b). Learning weak constraints in answer set programming. *Theory and Practice of Logic Programming*, *15*, 511–525.

Law, M., Russo, A., & Broda, K. (2015c). *Simplified reduct for choice rules in ASP*. Technical report, DTR2015-2, Department of Computing, Imperial College London, London.

Law, M., Russo, A., & Broda, K. (2016). Iterative learning of answer set programs from context dependent examples. *Theory and Practice of Logic Programming*, *16*, 834–848.

Law, M., Russo, A., & Broda, K. (2018). The complexity and generality of learning answer set programs. *Artificial Intelligence*, *259*, 110–146.

McCreath, E., & Sharma, A. (1997). ILP with noise and fixed example size: A Bayesian approach. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (pp. 1310–1315). Nagoya, Japan: Morgan Kaufmann.

Mooney, R. J., & Ourston, D. (1991). *Theory refinement with noisy data*. Technical Report AI-91-153, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, Texas.

Muggleton, S. (1991). Inductive logic programming. *New Generation Computing*, *8*, 295–318.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing*, *13*, 245–286.

Oblak, A., & Bratko, I. (2010). Learning from noisy data using a non-covering ILP algorithm. *Proceedings of the Twentieth International Conference on Inductive Logic Programming* (pp. 190–197). Florence, Italy: Springer.

Qomariyah, N. N., & Kazakov, D. (2017). Learning binary preference relations. *Proceedings of the Fourth Joint Workshop on Interfaces and Human Decision Making for Recommender Systems* (pp. 30–34). Como, Italy: CEUR.

Ray, O. (2009). Nonmonotonic abductive inductive learning. *Journal of Applied Logic*, *7*, 329–340.

Sakama, C., & Inoue, K. (2009). Brave induction: A logical framework for learning from incomplete information. *Machine Learning*, *76*, 3–35.

Sandewall, E., & Jansson, C. (1993). Handling imperfect data in inductive logic programming. *Proceedings of the Fourth Scandinavian Conference on Artificial Intelligence*. Stockholm, Sweden: IOS Press.

Sridharan, M., & Meadows, B. (2017). An architecture for discovering affordances, causal laws, and executability conditions. *Advances in Cognitive Systems*, *5*, 1–16.

Sridharan, M., Meadows, B., & Gómez, R. (2017). What can I not do? towards an architecture for reasoning about and learning affordances. *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling* (pp. 461–470). Pittsburgh, PA: ICAPS.

Srinivasan, A. (2001). The Aleph manual. Machine Learning at the Computing Laboratory, Oxford University. Retrieved 30 July, 2018, from `https://www.cs.ox.ac.uk/activities/machlearn/Aleph/`.

Tjong Kim Sang, E. F., & Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. *Proceedings of the Fourth Conference on Computational Natural Language Learning, and the Second Workshop on Learning Language in Logic* (pp. 127–132). Lisbon, Portugal: Association for Computational Linguistics.