Search Space Expansion for Efficient Incremental Inductive Logic Programming from Streamed Data

Mark Law[1] , **Krysia Broda**[2] and **Alessandra Russo**[2]
[1]ILASP Limited, UK
[2]Imperial College London, UK
mark@ilasp.com, {k.broda, a.russo}@imperial.ac.uk

Abstract

In the past decade, several systems for learning Answer Set Programs (ASP) have been proposed, including the recent FastLAS system. Compared to other state-of-the-art approaches to learning ASP, FastLAS is more scalable, as rather than computing the hypothesis space in full, it computes a much smaller subset relative to a given set of examples that is nonetheless guaranteed to contain an optimal solution to the task (called an OPT-sufficient subset). On the other hand, like many other Inductive Logic Programming (ILP) systems, FastLAS is designed to be run on a fixed learning task meaning that if new examples are discovered after learning, the whole process must be run again.

In many real applications, data arrives in a stream. Rerunning an ILP system from scratch each time new examples arrive is inefficient. In this paper we address this problem by presenting *IncrementalLAS*, a system that uses a new technique, called *hypothesis space expansion*, to enable a FastLAS-like OPT-sufficient subset to be expanded each time new examples are discovered. We prove that this preserves FastLAS's guarantee of finding an optimal solution to the full task (including the new examples), while removing the need to repeat previous computations. Through our evaluation, we demonstrate that running IncrementalLAS on tasks updated with sequences of new examples is significantly faster than re-running FastLAS from scratch on each updated task.

1 Introduction

Inductive Logic Programming (ILP) systems aim to find a set of logical rules called a hypothesis, which explain a set of examples in the context of some existing background knowledge. Over the past decade, ILP has been extended to learn programs under the answer set semantics [Ray, 2009; Corapi *et al.*, 2012; Law *et al.*, 2014; Law *et al.*, 2015b; Law *et al.*, 2016; Kazmi *et al.*, 2017]; however, early approaches either did not guarantee finding an optimal hypothesis or relied on computing the full set of rules that could appear in a hypothesis (the *hypothesis space*). Recently, the FastLAS systems [Law *et al.*, 2020; Law *et al.*, 2021] were proposed, introducing the idea of using the examples to compute a smaller subset of the hypothesis space that is guaranteed to contain an optimal solution (this is called an *OPT-sufficient* subset of the hypothesis space).

The FastLAS systems have been shown [Law *et al.*, 2020; Law *et al.*, 2021] to be much more scalable than other state-of-the-art ASP-based ILP systems, while still being guaranteed to compute optimal solutions, meaning that the accuracy of the learned hypotheses is (on average) the same as other such systems. However, like many ASP-based ILP systems, FastLAS is a batch system, meaning that all examples are processed in one go, and if new examples are discovered after learning then the entire process must be rerun from scratch. In many real-world applications of ILP this is likely to be inefficient as data often comes in streams, consisting of "windows" of examples arriving sequentially.

In this paper, we propose a new system called *IncrementalLAS*, inspired by the FastLAS approach of computing an OPT-sufficient subset of the hypothesis space. The novel feature of IncrementalLAS is that when it is used to process data in a stream, rather than recomputing the OPT-sufficient space from scratch each time, it instead uses a new technique called *hypothesis space expansion*, updating the previous OPT-sufficient space in light of the new examples. We prove the correctness of the hypothesis space expansion approach, meaning that, just like FastLAS, IncrementalLAS is guaranteed to find an optimal solution.

The evaluation presents the result of running both FastLAS and IncrementalLAS on datasets that have previously been used to evaluate FastLAS, where each dataset has been converted into a "streamed" dataset by splitting the full set of examples into windows, which are given to the systems sequentially. These experiments show that while FastLAS's running time increases significantly with each window, in many cases IncrementalLAS's time is almost constant.

The rest of the paper is structured as follows. The next section presents the relevant background material, and recaps the FastLAS algorithm. This is followed by a description of the hypothesis space expansion approach and then the evaluation of IncrementalLAS. The paper concludes with discussions of related and future work.

2 Background

Given any atoms $\texttt{h}$, $\texttt{b}_1, \ldots, \texttt{b}_\texttt{n}$, $\texttt{c}_1, \ldots, \texttt{c}_\texttt{m}$, a *normal rule* is $\texttt{h :- b}_1, \ldots, \texttt{b}_\texttt{n}$, $\texttt{not } c_1, \ldots, \texttt{not } c_m$, where $\texttt{h}$ is the *head*, $\texttt{b}_1, \ldots, \texttt{b}_\texttt{n}$, $\texttt{not c}_1, \ldots, \texttt{not c}_\texttt{m}$ (collectively) is the *body* of the rule, and "not" represents negation as failure. The head (resp. body) of a rule R is denoted $head(R)$ (resp. $body(R)$). A rule R is said to be a *sub-rule* of a rule R' iff $head(R) = head(R')$ and $body(R) \subseteq body(R')$ (we call R a *strict-sub-rule* if $R \neq R'$). In this paper, we assume an ASP program to be a set of normal rules. The Herbrand Base of a program P, denoted HB_P, is the set of variable free (ground) atoms that can be formed from predicates and constants in P. Given a program P and an interpretation $I \subseteq HB_P$, the *reduct* P^I is constructed from the grounding of P by removing rules whose bodies contain the negation of an atom in I and removing negative literals from the remaining rules. A set of ground atoms I is an *answer set* of P iff it is the minimal model of P^I.

Examples in this paper are *weighted context dependent partial interpretations* (WCDPI's) [Law *et al.*, 2018]. A *partial interpretation* e is a pair of sets of ground atoms $\langle e^{inc}, e^{exc} \rangle$. An interpretation (i.e. a set of ground atoms) I *extends* e iff $e^{inc} \subseteq I$ and $e^{exc} \cap I = \emptyset$. A WCDPI is a tuple $e = \langle e_{id}, e_{pen}, e_{pi}, e_{ctx} \rangle$, where e_{id} is an identifier for e, e_{pen} is either a positive integer or ∞, called a penalty, e_{pi} is a partial interpretation and e_{ctx} is an ASP program called a *context*. A WCDPI e is *accepted* by a program P iff there is an answer set of $P \cup e_{ctx}$ that extends e_{pi}.

Many ILP systems (e.g. [Muggleton, 1995; Ray, 2009; Srinivasan, 2001; Kazmi *et al.*, 2017]) use mode declarations as a form of language bias to specify hypothesis spaces. In this paper, we follow a similar approach. A *mode bias* is defined as a pair of sets of mode declarations $M = \langle M_h, M_b \rangle$, where M_h (resp. M_b) are called the *head* (resp. *body*) *mode declarations*. For simplicity of presentation, we only use propositional mode declarations (i.e. M_h is a set of propositional atoms and M_b is a set of propositional literals) in this paper, but the implementation of IncrementalLAS supports first-order mode declarations in the style of FastLAS [Law *et al.*, 2020][1] and learning first-order rules. Given a mode bias M, the rule space (often called the hypothesis space) S_M is the set of all normal rules R such that $head(R) \in M_h$ and $body(R) \subseteq M_b$.

As IncrementalLAS is based on the FastLAS algorithm, it supports the same learning tasks as the FastLAS system, called *positive non-recursive learning from answer sets tasks*. This task is a simplification of the learning task solved by the ILASP (Inductive Learning of Answer Set Programs) family of systems [Law *et al.*, 2015a; Law, 2022]. As discussed in [Law *et al.*, 2021], FastLAS (and similarly, IncrementalLAS) is much less general than the ILASP system, which solves the full learning from answer set tasks; for example, ILASP is able to learn recursive definitions, whereas FastLAS and IncrementalLAS are not.

Definition 1. *A* Positive Non-recursive Learning from Answer Sets *task is a tuple $T = \langle B, M, E \rangle$ where B is a program, M is a mode bias and E is a finite set of WCDPIs s.t. for each $e \in E$, $B \cup e_{ctx}$ can be partitioned into two programs $bottom_e$ and top_e s.t. no predicate in M_h or the head of a rule in top_e occurs in M_b or the body of a rule in $bottom_e$.*

For any hypothesis $H \subseteq S_M$:

- *For any $e \in E$, H* covers *e iff $B \cup H$ accepts e.*
- *The* score *of H, $\mathcal{S}_{len}(H, T)$,[2] is the number of literals in H, written $|H|$, plus the penalty of each example in T which is not covered by H.*
- *H is an* optimal solution *of T iff $\mathcal{S}_{len}(H, T)$ is finite and there is no $H' \subseteq S_M$ s.t. $\mathcal{S}_{len}(H', T) < \mathcal{S}_{len}(H, T)$.*

2.1 FastLAS Summary

The IncrementalLAS system described in this paper follows the same steps as FastLAS, but with some of the steps modified to allow for the OPT-sufficient space to be expanded rather than constructed from scratch. This section therefore recaps the main steps of the FastLAS.[3] We reformulate some of the definitions using our own notation and terminology in order to simplify the presentation of the rest of the paper.

FastLAS consists of four main steps: (1) *specific rule-space construction*; (2) *generalisation*; (3) *optimisation*; and (4) *solving*. For the remainder of this section, let T be the learning task $\langle B, M, E \rangle$. The rest of this section defines these four steps and exemplifies the whole process.

Specific rule-space construction. The first stage of the FastLAS algorithm computes a set of potential ways of covering each example in a learning task. Given a hypothesis H and a set of rules c, $H \lhd c$ iff there is at least one rule in H that is a sub-rule of a rule in c.

Definition 2. *For any $e \in E$, a* coverage alternative *for e is a tuple $C = \langle \{C_1^+, \ldots, C_n^+\}, C^- \rangle$, such that:*

- *Each of $C_1^+, \ldots, C_n^+, C^-$ is a subset of the rules in S_M.*
- *For each hypothesis $H \subseteq S_M$ such that $H \blacktriangleleft C$, H covers e, where $H \blacktriangleleft C$ iff $\forall i \in [1, n]$, $H \lhd C_i^+$ and $H \not\lhd C^-$.*

A set $\mathcal{C}$ of coverage alternatives for e is complete *if for every hypothesis $H \subseteq S_M$ that covers e, there is at least one $C \in \mathcal{C}$ s.t. $H \blacktriangleleft C$.*

FastLAS computes a complete set of coverage alternatives for each example leading to a *complete set of coverage alternatives* for T, written $\mathcal{C}(T)$, (the union of the sets of coverage alternatives for each example). Although there are often many possible $\mathcal{C}(T)$'s, we will refer to a single $\mathcal{C}(T)$, meaning the complete set of coverage alternatives that are computed by FastLAS. We write $\mathcal{C}^+(T)$ to denote the set of rules that occur in C_i^+ for some $C \in \mathcal{C}(T)$ and $\mathcal{C}^-(T)$ to denote the set of rules that occur in C^- for some $C \in \mathcal{C}(T)$.

[1] The experiments in the evaluation section all make use of first-order mode declarations.

[2] Like FastLAS, IncrementalLAS also supports user-defined scoring functions, but as they are not the subject of this paper, we restrict the discussion to the traditional "length" scoring function.

[3] These steps were presented in [Law *et al.*, 2020] and [Law *et al.*, 2021] and this section presents the combined steps of the most general FastLAS system, called FastNonOPL in [Law *et al.*, 2021].

Generalisation. Coverage alternatives contain extremely specific rules. It is necessary to consider rules that are sub-rules of multiple rules in $\mathcal{C}^+(T)$, as this leads to shorter solutions. Definition 3 formalises the generalised set of rules.

Definition 3. *For any rule $R \in S_M$, let $c^+_R(T)$ be the set of all rules R' in $\mathcal{C}^+(T)$ s.t. R is a sub-rule of R'. A generalised rule-space, $\mathcal{G}(T)$, w.r.t. $\mathcal{C}(T)$ is the set containing the rules $R \in S_M$ s.t. $c^+_R(T) \neq \emptyset$ and s.t. there is no rule $R' \in S_M$ s.t. R is a strict sub-rule of R' and $c^+_R(T) = c^+_{R'}(T)$.*

Optimisation. We say that a subset of the hypothesis space of a task is *OPT-sufficient* iff it either contains at least one optimal solution of the task or the original task is unsatisfiable. The generalised rule space contains rules that have been generalised, but only for cases where it is possible to combine multiple rules. In many cases, it is also necessary to generalise rules in order to optimise with respect to the scoring function (in this case $\mathcal{S}_{len}$). This computation can be done independently for each rule in $\mathcal{G}(T)$. Definition 4 formalises the notion of a *valid optimisation set* of a rule. For any rule r, $v(r, T)$ denotes the set of all rules $r' \in \mathcal{C}^-(T)$ of which r is a sub-rule. For any such rule r', we write that $r' \preceq_T r$ iff $|r'| \leq |r|$ and $v(r', T) \subseteq v(r, T)$; $r' \approx_T r$ iff $|r'| = |r|$ and $v(r', T) = v(r, T)$; and $r' \prec_T r$ iff $r' \preceq_T r$ and $r' \not\approx_T r$.

Definition 4. *For each rule $R \in \mathcal{G}(T)$, a set of rules $\mathcal{O}_{R,T}$ is a* valid optimisation set *of R w.r.t. T iff:*

1. *For each rule $r \in \mathcal{O}_{R,T}$: (i) r is a sub-rule of R for which there is no other sub-rule r' of R s.t. $r' \prec_T r$; and (ii) there is no rule $r' \in \mathcal{O}_{R,T}$ s.t. $r \approx_T r'$.*
2. *No rules can be added to $\mathcal{O}_{R,T}$ without violating (1).*

Note that condition (i) does not forbid there being other sub-rules of r' of R in S_M such that $r' \approx_T r$. In other words, it does not forbid there being other sub-rules that are "equally as good", only that there should not be any "better" sub-rules. Note that as there may be many rules which are "equally as good", and each of them could potentially be chosen as part of a valid optimisation set, in general there are many possible valid optimisation sets. FastLAS computes one such set. Throughout the rest of the paper, $\mathcal{O}_{R,T}$ will denote an arbitrary valid optimisation set of R w.r.t. T.

In [Law *et al.*, 2020], it was shown that $\bigcup_{R \in \mathcal{G}(T)} \mathcal{O}_{R,T}$ is an OPT-sufficient subset of the hypothesis space for T.

Solving. Once an OPT-sufficient subset of the hypothesis space has been computed it is possible to solve the task using an ASP solver such as Clingo 5 [Gebser *et al.*, 2016] to solve the task. By doing this, FastLAS guarantees finding an optimal solution to a given positive non-recursive learning from answer sets task.

3 Hypothesis Space Expansion

In this section, we describe a single run of the IncrementalLAS algorithm. We assume that IncrementalLAS has already been run on a learning task $T = \langle B, M, E \rangle$ and an additional set of examples E' are yet to be processed. The goal is to find an optimal solution to $T' = \langle B, M, E \cup E' \rangle$. IncrementalLAS's algorithm can be divided into two phases: *hypothesis space expansion*, in which the previously computed OPT-sufficient subset of the hypothesis space is expanded in light of the examples in E'; and *solving*, during which this OPT-sufficient space is searched for an optimal solution. The solving stage is identical to the solving stage in FastLAS, so we focus on the new *hypothesis space expansion* method in this section. Much like FastLAS's hypothesis space construction, described in the previous section, hypothesis space expansion consists of three steps, computing a set of coverage alternatives $\mathcal{C}(T')$, a generalised rule space $\mathcal{G}(T')$ and a set of optimisations $\mathcal{O}(T')$. The key difference between the steps described in this section and those used in FastLAS is that IncrementalLAS updates the previous $\mathcal{C}(T)$, $\mathcal{G}(T)$ and $\mathcal{O}(T)$, avoiding needing to repeat computations that were already performed when solving the previous task T. The following definition formalises the state that is reached following performing hypothesis space expansion on a task T – i.e. the pre condition required before running IncrementalLAS on T'. In the remainder of this section, we present the details of the three steps of hypothesis space expansion and prove that the equivalent state for T' holds after IncrementalLAS has been run on T' – i.e. we prove that it is a post condition of IncrementalLAS. This proves that IncrementalLAS can be used on a stream of data, continually expanding the hypothesis space each time new examples arrive, and that the condition will always hold for the task containing the examples seen so far.

Definition 5. *Given a task $T = \langle B, M, E \rangle$ a valid* state after running IncrementalLAS on T *is a tuple of the form $\langle \mathcal{C}(T), \mathcal{G}(T), \mathcal{O}(T) \rangle$, where $\mathcal{C}(T)$ is a complete set of coverage alternatives for T, $\mathcal{G}(T)$ is the generalised rule-space w.r.t. $\mathcal{C}(T)$, and $\mathcal{O}(T)$ is a set consisting of one pair $\langle R, \mathcal{O}_{R,T} \rangle$ for each $R \in \mathcal{G}(T)$.*

Throughout the rest of this section, T (resp. T') is assumed to be a task $\langle B, M, E \rangle$ (resp. $\langle B, M, E \cup E' \rangle$), and $\langle \mathcal{C}(T), \mathcal{G}(T), \mathcal{O}(T) \rangle$ is a valid state after solving T.

Example 1. *(Running Example). Throughout the rest of this section, the following simple propositional[4] learning task, $T = \langle B, M, E \rangle$ is used to illustrate the steps of hypothesis space expansion. The elements of T are as follows: $B = \emptyset$; $M = \langle \{\texttt{p}, \texttt{q}, \texttt{r}\}, \{\texttt{a}, \texttt{b}, \texttt{not b}, \texttt{c}, \texttt{d}\} \rangle$; $E = \{e_1, e_2\}$, where $e_1 = \langle 1, 10, \langle \{\texttt{p}, \texttt{r}\}, \{\texttt{q}\} \rangle, \{\texttt{a. c.}\} \rangle$ and $e_2 = \langle 2, 10, \langle \{\texttt{p}\}, \{\texttt{q}, \texttt{r}\} \rangle, \{\texttt{b. c.}\} \rangle$.*

In this case, the following is a valid state after solving T:

- *$\mathcal{C}(T)$ contains two coverage alternatives (one for each example): $\langle \{\{\texttt{p :- a, not b, c.}\}, \{\texttt{r :- a, not b, c.}\}\}, \{\texttt{q :- a, not b, c.}\} \rangle, \langle \{\{\texttt{p :- b, c.}\}\}, \{\texttt{q :- b,c. r :- b,c.}\} \rangle$.*
- *$\mathcal{G}(T)$ consists of: `p :- a, not b, c`; `p :- b, c`; `p :- c`; and `r :- a, not b, c`.*
- *Let the four rules in $\mathcal{G}(T)$ be denoted $R_1, \ldots, R_4$ (respectively). $O(T)$ contains the pairs $\langle R_1, \{\texttt{p.}\} \rangle$, $\langle R_2, \{\texttt{p.}\} \rangle$, $\langle R_3, \{\texttt{p.}\} \rangle$ and $\langle R_4, \{\texttt{r. r :- a.}\} \rangle$. The union of the valid optimisations is an OPT-sufficient space.*

We now describe how each of the three elements of the state can be computed using the information in the previous state, without needing to compute the entire state from scratch.

[4] IncrementalLAS is of course capable of learning first order rules and solving much larger tasks, but T was chosen to keep the examples in the paper short.

3.1 Computing $\mathcal{C}(T')$

In FastLAS, each example $e \in E$ is processed to compute C_e. As each example is processed separately, $\mathcal{C}(T')$ is equal to $\mathcal{C}(T) \cup \bigcup_{e \in E'} C_e$ – where C_e is the set of coverage alternatives for e computed by FastLAS. Hence, computing the update for $\mathcal{C}$ is simple, as we can use the existing FastLAS methods.

Example 2. *Reconsider the running example and an E' containing a single new example $e_3 = \langle 3, 10, \langle\{\texttt{q}, \texttt{r}\}, \{\texttt{p}\}\rangle, \{\texttt{b. d.}\}\rangle$. $\mathcal{C}(T') = \mathcal{C}(T) \cup \{\langle\{\{\texttt{q :- b, d.}\}, \{\texttt{r :- b, d.}\}\}, \{\texttt{p :- b, d.}\}\rangle$. Note that this coverage alternative can be computed from the new example e_3 using the techniques used in FastLAS. The other coverage alternatives do not need to be recomputed.*

3.2 Computing $\mathcal{G}(T')$

When computing the generalisations of rules in $\mathcal{C}^+(T')$, it is not sufficient to only consider the coverage alternatives derived from new examples in E'. This is illustrated by Example 3.

Example 3. *Reconsider the running example and the E' described in Example 2. If we compute the generalised rule space w.r.t. the single new coverage alternative in Example 2 and add it to $\mathcal{G}(T)$, there are two rules with* `r` *in the head:* `r :- b, d` *and* `r :- a, not b, c`*, with the former coming from the new coverage alternative and the latter coming from $\mathcal{G}(T)$. This is incomplete as the generalised rule space w.r.t. $\mathcal{C}(T')$ also contains the fact* `r`*.*

Algorithm 1 shows the update algorithm for the generalisation phase of IncrementalLAS. The intuition is similar to the implementation of FastLAS,[5] with the key difference being that some execution branches, which are guaranteed not to yield new rules, are pruned – i.e. the execution only considers rules that are sub-rules of at least one rule in $\mathcal{C}^+(T')\backslash\mathcal{C}^+(T)$. The algorithm builds up a set G of pairs $\langle R, \mathit{subs}\rangle$ where R is a rule and *subs* is the set of rules in $\mathcal{C}^+(T')$ for which R is a sub-rule. Initially each rule R is a fact (one of the possible heads) and *subs* is the set of all rules in $\mathcal{C}^+(T')$ which share that head. Each possible body literal *bl* is considered in turn: for each $\langle R, \mathit{subs}\rangle$ in the previous G, we decide whether to retain R in G for the next iteration and whether to add the rule R' constructed by adding *bl* to the body of R. There are three cases: (i) if R' is a sub-rule of each rule in *subs*, we only retain R'; (ii) if R' not a sub-rule of any rule in $\mathcal{C}^+(T')\backslash\mathcal{C}^+(T)$, then we do not need to consider (any extensions of) R', as all useful extensions of it will have been computed in previous iterations, so we only retain R; (iii) if neither (i) nor (ii) applies we must retain both R and R', as there may be useful extensions of both rules that have not been computed in previous iterations. The *Filter* function in line 16 filters out any tuple $\langle R, \mathit{subs}\rangle$ such that there is a body literal in *bls* that has already been processed and that occurs in the body of every rule in *subs* but not in the body of R. This is not strictly necessary for the correctness of IncrementalLAS, but it removes some rules which are not in the generalised rule space and are therefore redundant, meaning that they would lead to a larger OPT-sufficient space than is necessary.

[5] Available for download from https://github.com/spike-imperial/FastLAS.

Algorithm 1 *generalise*$(\mathcal{C}(T), \mathcal{C}(T'))$

```
1:  procedure GENERALISE(C(T), C(T'))
2:      rs = C+(T')\C+(T);
3:      Let hs be the set of heads of rules in rs;
4:      Let bls be the literals in bodies of rules in rs;
5:      G = {⟨h, {R ∈ C+(T') | head(R) = h}⟩ | h ∈ hs};
6:      for bl ∈ bls do
7:          NewG = ∅;
8:          for ⟨(h :- body.), subs⟩ ∈ G do
9:              new_subs = {R ∈ subs | bl ∈ body(R)};
10:             if new_subs = subs then
11:                 NewG.insert(⟨(h :- body, bl.), subs⟩);
12:             else
13:                 NewG.insert(⟨(h :- body.), subs⟩);
14:                 if new_subs ∩ rs ≠ ∅ then
15:                     NewG.insert(⟨(h :- body, bl.), new_subs⟩);
16:         G = Filter(NewG);
17:     return {R | ⟨R, subs⟩ ∈ G};
```

Example 4. *Reconsider the running example. Consider the generalise algorithm being run on $\mathcal{C}(T)$ and $\mathcal{C}(T')$. In the first line, rs is set to* {`q :- b, d.  r :- b, d.`}*. hs =* {`q, r`} *and bls =* {`b, d`}*. The value of G before the first iteration and after each iteration is given in the list below:*

1. $\{\ \langle$`r.`$, \{$`r :- a, not b, c`, `r :- b, d`$\}\rangle, \langle$`q.`$, \{$`q :- b, d`$\}\rangle\ \}$
2. *(bl =* `b`*).* $\left\{ \begin{array}{l} \langle\texttt{r.}, \{\texttt{r :- a, not b, c}, \texttt{r :- b, d}\}\rangle, \\ \langle\texttt{r :- b.}, \{\texttt{r :- b, d}\}\rangle, \langle\texttt{q :- b.}, \{\texttt{q :- b, d}\}\rangle \end{array} \right\}$
3. *(bl =* `d`*).* $\left\{ \begin{array}{l} \langle\texttt{r.}, \{\texttt{r :- a, not b, c}, \texttt{r :- b, d}\}\rangle, \\ \langle\texttt{r :- b, d.}, \{\texttt{r :- b, d}\}\rangle, \\ \langle\texttt{q :- b, d.}, \{\texttt{q :- b, d}\}\rangle \end{array} \right\}$.

Note that the tuple $\langle$`r :- d.`, {`r :- b, d`}$\rangle$ *is added but removed by the filter as the rule does not contain* `b`*.*

Theorem 1. *The set $\mathcal{G}(T') = \mathit{generalise}(\mathcal{C}(T), \mathcal{C}(T')) \cup \mathcal{G}(T)$ is a generalised rule space w.r.t. $\mathcal{C}(T')$.*

The proof of Theorem 1 is given in the online supplementary material document, available at https://github.com/spike-imperial/FastLAS/blob/master/incremental_las_proofs.pdf.

3.3 Computing $\mathcal{O}(T)$

In FastLAS, each rule in $R \in \mathcal{G}(T)$ is optimised, yielding a set of rules $\mathcal{O}_R$. For IncrementalLAS, it is not sufficient to only optimise the new rules in $\mathcal{G}(T')\backslash\mathcal{G}(T)$, as illustrated by the following example.

Example 5. *Reconsider the running example. The OPT-sufficient hypothesis space for T (derived from the optimisations computed in Example 1) is* {`p. r. r :- a.`}*. $\mathcal{G}(T')\backslash\mathcal{G}(T)$ contains no rules with* `p` *in the head. This means that if we only optimise these new generalised rules, we would not generate any extra rules for* `p`*. This would mean that there is no way of covering e_1 and e_3 (nor e_2 and e_3) simultaneously, meaning that the score of any solution must be at least 10.*

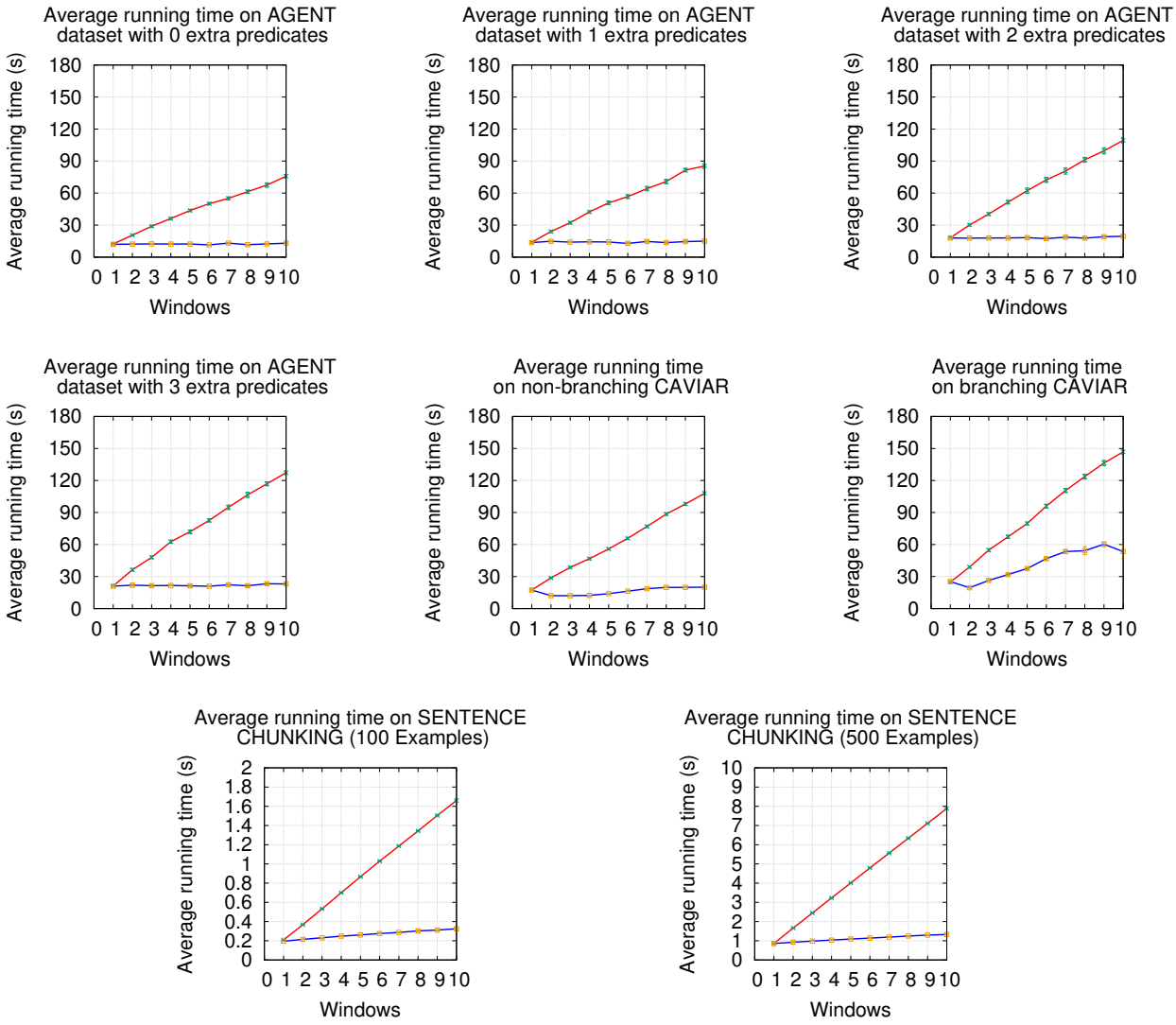

Figure 1: These graphs show the running times of FastLAS and IncrementalLAS on the streamed datasets. In each graph, FastLAS is represented by the red line and IncrementalLAS by the blue.

There are hypotheses with lower scores that cover all the examples, so this method does not lead to an OPT-sufficient subset of the hypothesis space. The issue is that although the previously optimised rules with p *in the head are not sub-rules of any rules in* $\mathcal{C}^-(T)$*, they are sub-rules of rules in* $\mathcal{C}^-(T')$*.*

The previous example showed that in addition to optimising the new generalised rules, it is also necessary to *re-optimise* some rules. The following definition specifies exactly which rules need to be (re-)optimised.

Definition 6. *The set of* rules that must be (re-)optimised *is given by* $\mathcal{U}(T,T') = (\mathcal{G}(T')\backslash\mathcal{G}(T)) \cup \{R \in \mathcal{G}(T) \mid \langle R, \mathcal{O}_{R,T}\rangle \in \mathcal{O}(T), \mathcal{O}_{R,T} \cap (\mathcal{C}^-(T')\backslash\mathcal{C}^-(T)) \neq \emptyset\}$.

The following theorem shows that it is sufficient to optimise the set of rules in $\mathcal{U}(T,T')$.

Theorem 2. *Let* $\mathcal{O}(T') = \{\langle R, \mathcal{O}_{R,T'}\rangle \mid R \in \mathcal{U}(T,T')\} \cup \{\langle R, \mathcal{O}_{R,T}\rangle \mid R \in \mathcal{G}(T')\backslash\mathcal{U}(T,T')\}$*.* $\langle \mathcal{C}(T'), \mathcal{G}(T'), \mathcal{O}(T')\rangle$ *is a valid state after solving* T'*.*

Proof. As the sets $\mathcal{U}(T,T')$ and $\mathcal{G}(T')\backslash\mathcal{U}(T,T')$ form a partition of the set $\mathcal{G}(T')$, each rule in $\mathcal{G}(T')$ is clearly represented by exactly one pair in $\mathcal{O}(T')$. It therefore remains to show that the second element of this pair is a valid optimisation of R w.r.t. T'.

Case 1: $R \in \mathcal{U}(T,T')$. In this case the second element, $\mathcal{O}_{R,T'}$ is defined according to Definition 4 of the main paper (using $\mathcal{G}(T')$ and $\mathcal{C}(T')$); hence, due to the correctness results proved in [Law *et al.*, 2020] it must be a valid optimisation of R w.r.t. T'.

Case 2: $R \notin \mathcal{U}(T,T')$. In this case, we must show that $\mathcal{O}_{R,T}$ is a valid optimisation of R w.r.t. T. Assume for contradiction that it is not. Then either there is a rule in $\mathcal{O}_{R,T}$ that violates one of the conditions in (1) of Definition 4, or an ad-

ditional rule can be added without violating (1). As R is not in $\mathcal{U}(T, T')$, for each rule in $r \in \mathcal{O}_{R,T}$, $v(r, T) = v(r, T')$. Hence, as for any other rule in $r' \in S_M$, $v(r', T) \subseteq v(r', T')$, this cannot be the case. Contradiction! □

3.4 Proof of Correctness

Similarly to FastLAS, the final step of IncrementalLAS is to search the rule space $\mathcal{O}(T')$ for an optimal solution of the task. This step is no different in IncrementalLAS to how it is defined in [Law *et al.*, 2020], and so to prove that IncrementalLAS is guaranteed to compute an optimal solution to a given task, it remains to show that the rule space $\mathcal{O}(T')$ is guaranteed to contain at least one optimal solution (i.e. it is OPT-sufficient).

Theorem 3. *For any task T and valid state, $\langle \mathcal{C}(T), \mathcal{G}(T), \mathcal{O}(T) \rangle$, after solving T, $\bigcup_{\langle R,S \rangle \in \mathcal{O}(T)} R$ is OPT-sufficient.*

Proof. Assume for contradiction that $O = \bigcup_{\langle R,S \rangle \in \mathcal{O}(T)} R$ is not OPT-sufficient. Then T must be satisfiable. Let H^* be an optimal solution of T. There must be a rule in $h \in H^*$ such that $c_h^+(T) \neq \emptyset$ and there is no rule $h' \in O$ s.t. $|h| = |h'|$, $c_h^+(T) = c_{h'}^+(T)$ and $v(h', T) \subseteq v(h, T)$ – otherwise each rule R could be replaced with its corresponding rule R', and the resulting hypothesis would be an optimal solution.

As $\mathcal{G}(T)$ is a generalised rule space, there must be a rule $h^g \in \mathcal{G}(T)$ such that $c_h^+(T) = c_{h^g}^+(T)$ and h is a sub-rule of h^g. But if this were the case, as $\mathcal{O}(R, T) \subset O$, $\mathcal{O}(R, T)$ must not be a valid optimisation of R w.r.t. T, as h can be added to it without breaking (1) of Definition 4. Contradiction! □

4 Evaluation

This section presents an evaluation of IncrementalLAS on streamed data. Just like FastLAS, IncrementalLAS is guaranteed to return an optimal solution (as shown by the theoretical results in the previous section). Consequently, although on any given run the two systems may return different optimal solutions, over a large enough number of runs their average accuracy will be the same. For instance, over the 400 tasks for the `AGENT` experiments below, the average accuracy of IncrementalLAS was 0.965 and for FastLAS it was 0.963. In fact, over 400 runs there were only 21 tasks for which the two systems did not acheive exactly the same accuracy. For this reason, as FastLAS has already been shown [Law *et al.*, 2020; Law *et al.*, 2021] to outperform other ASP-based ILP systems in terms of accuracy, we focus on a comparison between the running times of FastLAS and IncrementalLAS on a variety of datasets on which FastLAS has been evaluated in the past; specifically, the `AGENT`, `CAVIAR` and `SENTENCE CHUNKING` datasets, which are all available online.[6] Each dataset contains a set of learning tasks and the graphs in Figure 1 each show the average running time over multiple tasks (details of each dataset are given in the following subsections).

IncrementalLAS can be run by downloading the latest version of FastLAS (currently 2.1.0) from https://spike-imperial.github.io/FastLAS/ and running FastLAS with the `--read-cache` and `--write-cache` flags (which represent the files where the cache is read from and written to). Setting both cache flags to the same (initially empty) file, will cause the system to start from an empty cache and continually update this as the system is run on tasks with increasing sets of examples.

In order to simulate data arriving in a stream of windows, we partitioned the examples in each learning task into 10 equal sized windows of examples. For each system, the i^{th} point on each graph shows the average running time of the system when processing the i^{th} window. For FastLAS, this means generating the OPT-sufficient subset w.r.t. windows 1 to i and then searching for a solution, whereas for IncrementalLAS, this means using the hypothesis space expansion method to expand a previous OPT-sufficient space for windows 1 to $(i-1)$ in light of the examples in window i and then searching for a solution.

Agent. The `AGENT` dataset [Law *et al.*, 2014] concerns a problem in which an agent must navigate a grid world to achieve a goal, whilst simultaneously learning the rules of the grid world. We consider a non-observational predicate learning (non-OPL) version of this learning task, which has previously been used to evaluate the FastLAS system [Law *et al.*, 2021]. Non-OPL tasks address the problem of learning concepts which are not directly observable from the examples, but are instead only linked through the background knowledge. In this particular case, each example corresponds to a single trace through the grid world and is labelled only as either valid or invalid. The learner must take advantage of a background knowledge definition which says that a trace is valid if all moves within it are valid (and invalid otherwise) to learn a definition of valid move. This is an instance of a *branching* [Law *et al.*, 2021] non-OPL task, because each invalid trace gives rise to multiple possibilities about which individual move(s) were invalid (the learner only knows that at least one move in an invalid trace is invalid, but does not know which one). There are four versions of the dataset, the simplest of which has a mode bias containing only those predicates necessary to learn the target hypothesis and the other three with 1, 2 and 3 extra predicates which are irrelevant to the concept of valid move. The versions with extra predicates show how learning systems perform when the size of the hypothesis space is increased. In all four cases FastLAS's time increases approximately linearly with the number of windows as it is repeating all of the previous computations with each new window. On the other hand, IncrementalLAS's time remains more or less constant as the number of windows increases. In each case, the time processing the last window is over 5 times higher for FastLAS than for IncrementalLAS.

CAVIAR. The `CAVIAR` dataset [Fisher *et al.*, 2004] consists of structured data extracted from video data. The structured data consists of information such as the positions of people and when two people are interacting. We consider two scenarios which have previously been used to evaluate FastLAS [Law *et al.*, 2021], in which the aim is to learn initiating and terminating conditions for two people meeting. The first scenario is non-branching as the examples explicitly state

[6] Available for download from https://github.com/spike-imperial/FastLAS.

which pairs of people are meeting. The second is branching, as each example only specifies whether at least one pair of people is meeting. In examples with more than two people, this leads to multiple possibilities for which pairs of people are meeting. The non-branching version shows a similar trend to the AGENT experiments, with FastLAS's running time increasing approximately linearly and IncrementalLAS's time remaining low. It is worth noting that the early windows after the first window are faster than the first window. This is because the first window causes most of the final OPT-sufficient subset of the space to be generated and although there is still some work to be done in generating $\mathcal{C}(T')$, there is little extra work in the rest of the hypothesis space expansion process. The branching task is the only experiment in this evaluation in which IncrementalLAS's running time increases significantly with the number of windows. The reason for this increase is that of IncrementalLAS's two phases (hypothesis space expansion, and searching for an optimal solution), the latter phase takes a far greater proportion of the total time in this experiment than in the other experiments. Even so, IncrementalLAS is considerably faster than FastLAS on this task.

Sentence Chunking. The goal of sentence chunking [Tjong Kim Sang and Buchholz, 2000] is to learn to split a sentence into short phrases called "chunks". For instance, according to the dataset [Agirre *et al.*, 2016] used in this paper, the newspaper headline "Thai opposition party to boycott general election" should be split into "Thai opposition party", "to boycott" and "general election". This dataset has been used to evaluate INSPIRE [Kazmi *et al.*, 2017] and ILASP [Law *et al.*, 2018; Law, 2018]. In [Law *et al.*, 2020] it was shown that FastLAS outperformed INSPIRE and ILASP, achieving a similar F_1 score to ILASP – which (like FastLAS) is guaranteed to find an optimal solution – and a significantly lower running time. There are two versions of the dataset, one with 100 sentences and the other with 500 sentences. In both cases, the running time of IncrementalLAS is significantly (almost an order of magnitude) lower than that of FastLAS by the 10^{th} window.

5 Related Work

Many traditional ILP systems (e.g. [Muggleton, 1995; Srinivasan, 2001; Ray *et al.*, 2003]) use a cover loop, in which positive examples are processed one at a time. In each iteration an uncovered positive example is selected and a rule is added to the hypothesis which covers that example. Cover loops have several disadvantages: (1) they are incapable of handling non-monotonicity [Ray, 2009; Law, 2018], which is vital when learning ASP programs; (2) even if the rules in each iteration are optimal, the combined solution is not guaranteed to be optimal; and (3) many such systems are incapable of learning from noisy data. In addition, although they learn from one positive example at a time, they assume the presence of all negative examples to avoid learning rules which prove a negative example. This makes them unsuitable for learning from streamed data in which some negative examples may come later than others in the stream.

The ILED [Katzouris *et al.*, 2015] system enables incremental learning of event definitions. Although ILED has similar aims of processing data in streams, it makes different assumptions/guarantees to those assumed/guaranteed in this paper: (1) all examples are non-noisy; (2) each window is processed only once and is not stored – while IncrementalLAS does not rely on storing the examples themselves, the coverage alternatives and key parts of the examples are stored; (3) ILED revises the hypothesis learned in the previous iteration and therefore does not guarantee finding an optimal solution overall, whereas IncrementalLAS guarantees finding an optimal solution to the overall task. The successor to ILED, OLED (Online Learning of Event Definitions) [Katzouris *et al.*, 2016] enables learning from noisy examples, but still makes the assumption that each window is processed only once, meaning that it is still not guaranteed to find an optimal solution to a given task. It has previously been shown [Law *et al.*, 2021] that OLED is faster than FastLAS, but that due to its ability to find an optimal solution, FastLAS returns a better quality hypothesis in terms of the F_1 score on the non-branching CAVIAR dataset – OLED is incapable of solving branching tasks, so cannot be applied to the branching version of CAVIAR. Although not shown in the paper, the comparison between OLED and IncrementalLAS would be similar. In particular, as discussed in the previous section, both IncrementalLAS and FastLAS return arbitrary optimal solutions, so their average F_1 scores would be the same. On the other hand, IncrementalLAS's average total time for processing the CAVIAR dataset was 163s, whereas OLED takes 107s [Katzouris *et al.*, 2016].

6 Conclusion and Future Work

Learning efficiently and accurately from streamed data is an important challenge for ILP systems. This paper has presented the new IncrementalLAS system for learning from streamed data, based on a technique called hypothesis space expansion. We have proved that IncrementalLAS preserves the ability of other state-of-the-art LAS systems to find an optimal solution to a given task, meaning that on average the accuracy of returned solutions will be the same. On the other hand, through our evaluation, we have shown that IncrementalLAS is significantly more efficient at learning from streamed data than the state-of-the-art FastLAS system.

This paper makes the assumption that while data is coming from a stream, the ground truth is not changing over time. Future work will explore extending IncrementalLAS to allow for concept drift.

References

[Agirre *et al.*, 2016] Eneko Agirre, Aitor Gonzalez Agirre, Inigo Lopez-Gazpio, Montserrat Maritxalar, German Rigau Claramunt, and Larraitz Uria. Semeval-2016 task 2: Interpretable semantic textual similarity. In *SemEval-2016. 10th International Workshop on Semantic Evaluation.* ACL, 2016.

[Corapi *et al.*, 2012] Domenico Corapi, Alessandra Russo, and Emil Lupu. Inductive logic programming in answer set programming. In *Inductive Logic Programming*, pages 91–97. Springer, 2012.

[Fisher *et al.*, 2004] Robert Fisher, Jose Santos-Victor, and James Crowley. CAVIAR: Context aware vision using image-based active recognition. http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/, 2004. Accessed: 2019-08-28.

[Gebser *et al.*, 2016] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Philipp Wanko. Theory solving made easy with clingo 5. In *Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[Katzouris *et al.*, 2015] Nikos Katzouris, Alexander Artikis, and Georgios Paliouras. Incremental learning of event definitions with inductive logic programming. *Machine Learning*, 100(2-3):555–585, 2015.

[Katzouris *et al.*, 2016] Nikos Katzouris, Alexander Artikis, and Georgios Paliouras. Online learning of event definitions. *Theory and Practice of Logic Programming*, 16(5-6):817–833, 2016.

[Kazmi *et al.*, 2017] Mishal Kazmi, Peter Schüller, and Yücel Saygın. Improving scalability of inductive logic programming via pruning and best-effort optimisation. *Expert Systems with Applications*, 87:291–303, 2017.

[Law *et al.*, 2014] Mark Law, Alessandra Russo, and Krysia Broda. Inductive learning of answer set programs. In Eduardo Fermé and João Leite, editors, *Proceedings of the Fourteenth European Conference on Logics in Artificial Intelligence, 2014, Funchal, Madeira, Portugal, September 24-26, 2014.*, volume 8761 of *Lecture Notes in Computer Science*, pages 311–325. Springer, 2014.

[Law *et al.*, 2015a] Mark Law, Alessandra Russo, and Krysia Broda. The ILASP system for learning answer set programs. www.ilasp.com, 2015. Accessed: 2022-05-27.

[Law *et al.*, 2015b] Mark Law, Alessandra Russo, and Krysia Broda. Learning weak constraints in answer set programming. *Theory and Practice of Logic Programming*, 15(4-5):511–525, 2015.

[Law *et al.*, 2016] Mark Law, Alessandra Russo, and Krysia Broda. Iterative learning of answer set programs from context dependent examples. *Theory and Practice of Logic Programming*, 16(5-6):834–848, 2016.

[Law *et al.*, 2018] Mark Law, Alessandra Russo, and Krysia Broda. Inductive learning of answer set programs from noisy examples. *Advances in Cognitive Systems*, 2018.

[Law *et al.*, 2020] Mark Law, Alessandra Russo, Elisa Bertino, Krysia Broda, and Jorge Lobo. FastLAS: Scalable inductive logic programming incorporating domain-specific optimisation criteria. In *AAAI*. Association for the Advancement of Artificial Intelligence, 2020.

[Law *et al.*, 2021] Mark Law, Alessandra Russo, Krysia Broda, and Elisa Bertino. Scalable non-observational predicate learning in ASP. *IJCAI*, 1943(10.24963), 2021.

[Law, 2018] Mark Law. *Inductive Learning of Answer Set Programs*. PhD thesis, Imperial College London, 2018.

[Law, 2022] Mark Law. Conflict-driven inductive logic programming. *Theory and Practice of Logic Programming*, page 1–28, 2022.

[Muggleton, 1995] Stephen Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13(3-4):245–286, 1995.

[Ray *et al.*, 2003] Oliver Ray, Krysia Broda, and Alessandra Russo. Hybrid abductive inductive learning: A generalisation of progol. In *Inductive Logic Programming*, pages 311–328. Springer, 2003.

[Ray, 2009] Oliver Ray. Nonmonotonic abductive inductive learning. *Journal of Applied Logic*, 7(3):329–340, 2009.

[Srinivasan, 2001] Ashwin Srinivasan. The aleph manual. *Machine Learning at the Computing Laboratory, Oxford University*, 2001.

[Tjong Kim Sang and Buchholz, 2000] Erik F Tjong Kim Sang and Sabine Buchholz. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics, 2000.