MGTP Application Study to Inductive Logic Programming

Naoki Yagi, Keiko Shimazu, Koichi Furukawa, Akihiro Kimura

November 25 , 1994

1 Introduction

Inductive logic programming provides a new approach for realizing machine learning based on first order logic. The main advantage of the approach is the ability to utilize background knowledge during learning phase. This ability makes it possible to enhance the expressiveness of the system simply by providing related knowledge as one of its inputs. We applied MGTP to an inductive logic programming system called PROGOL developed by Muggleton et al. We found that the use of MGTP makes the PROGOL complete for deriving the most specific hypothesis which plays an important role to guide the search in concept hierarchy lattice to find a most suitable concept description explaining given positive examples whereas keeping to exclude negative examples under given background knowledge. The essential idea of the use of MGTP in PROGOL is to compute an extended minimal model for the union of background knowledge and a singleton set consisting negation of a positive example. We need both positive and negative literals derivable from the union. We found it possible to compute a set of positive and negative literals using a version of MGTP, called Ground-MGTP/MERC, which accepts both standard and contra-positive representations and computes minimal models.

To realize parallel PROGOL, we started to design a new implementation of PROGOL in Prolog and KL1. Essentially, there are two modules in PROGOL, one for computing a most specific hypothesis to explain a given positive example and the other for finding a most suitable description to explain the given example. The former part can be realized by a direct copy of the ground-MGTP/MERC. The essential algorithm of the latter part is A^* algorithm for searching on a concept hierarchy lattice each node of which is a clause describing a positive example. We are now designing and developing the latter module in Prolog. The implementation of the system in KL1 is scheduled later.

2 The structure of PROGOL

2.1 Computing a most specific hypothesis

Inductive inference is a form of reasoning which is very common within natural sciences. As Muggleton suggests, there have been long time effort to formalize the reasoning. [Muggleton ,1994]

> Aristotle first describes it in his "Posterior Analytics". Frances Bacon, in discussing the empiricism of the new natural sciences in the 17th century gave numerous examples of inductive inference as a paradigm for scientific method. However, despite the efforts of philosophers such as Hume, Mill, Pierce, Popper and Carnap, the foundations of inductive reasoning are still much less clear than those of deductive mathematical logic. Since the 1970's several researchers from within Computer Science have attempted, with varying degrees of success, to find a logical basis for inductive inference. These researchers have included Plotkin, Shapiro and the new school of Inductive Logic Programming.

In the following, we briefly introduce the theoretical background of PROGOL. [Muggleton]

The purpose of PROGOL is to explore inductive inference in the framework of classical first order predicate calculus, Bayesian statics and algorithmic complexity theory. This is achieved by finding a H which satisfies the four conditions. Inductive inference can be formulated as a problem of finding a hypothesis H.

- Prior Satisfiability. $B \wedge E^- \not\models \square$
  (Negative examples not allow us to explain false to B.)
- Posterior Satisfiability. $B \wedge H \wedge E^- \not\models \square$
  (E^- not allow us to explain false to B and H.)
- Prior Necessity. $B \not\models E^+$
  (B not allow us to explain positive examples.)
- Posterior sufficiency. $B \wedge H \models E^+$
  (H allows us to explain E relative to B.)

Also it minimizes $I(T \mid E)$ where $I(T \mid E) = I(T) + I(E \mid T) - I(E)$, where $T = B \wedge H$. This is driven by simply the log form of the Bayes' formula. In this case, T "compress" the examples, since it less information content than they have.

Considering the posterior sufficiency condition ($B \wedge H \models E^+$), according to the Deduction Theorem $F1 \wedge F2 \models F3$ if and only if $F1 \models \overline{F2} \vee F3$, it can be applied left to right $B \models \overline{H} \vee E^+$ and also right to left $B \wedge \overline{E^+} \models \overline{H}$. So the following theorem shows how this relationship can be made one which allows the hypothesis space to be logically derived from background knowledge and examples.

For every hypothesis H such that $B \wedge H \models E^+$, $\overline{H}$ can be derived from $B \wedge \overline{E^+}$ ($B \wedge \overline{E^+} \models \overline{H}$) using any sound and complete derivation method. If H is assumed to be a single clause, then $\overline{H}$ is a conjunction of ground (possible skolemised) literals. The set of ground literals is derivable using linear resolution from $B \wedge \overline{E^+}$.

The problem of generating a single clause hypothesis has been studied extensively. A central notion is that one of the most specific hypothesis relative to B and E^+. This will be denoted by $\perp_H (B, E^+)$. It is defined that $\perp_H (B, E^+)$ is the negation of the conjunction of literals derivable from $B \wedge E^+$.

The PROGOL system derives such a most specific clause when carrying out generalizations. However, with PROGOL the most specific clause is variabilised according to "mode" declarations. ¿From $B \not\models E^+$, $\overline{\perp_H (B, \overline{E^+})}$ is made of conjunction of literals derivable from $B \wedge \overline{E^+}$ and H is made as a set of clauses. $B \wedge H \models E^+$ if and only if $H \rightarrow \perp_H (B, E^+)$. (All hypothesis implies the most specific hypothesis.) This theorem is proved as follows.

In "if part", $H \rightarrow \perp_H (B, E^+)$ and $B \wedge H \not\models E^+$ are assumed, but $B \wedge \perp_H (B, E^+) \models E^+$ and $B \wedge H \models B \wedge \perp_H (B, E^+)$ are derived. Therefore $B \wedge H \models E^+$ contradicts assumption. In the other hand, in "only if part", $B \wedge H \models E^+$ and $H \not\rightarrow \perp_H (B, E^+)$, H does not imply, are assumed. This implies $\overline{(\overline{H} \vee \perp_H (B, E^+))}$ therefore $H \wedge \overline{\perp_H (B, E^+)}$. But $B \wedge \overline{E} \models \overline{H}$ which means $B \wedge \overline{E^+}$ must be unsatisfiable because H is true. Then $B \models E^+$ contradicts the assumption.

The PROGOL system searches though all clauses which subsume $\perp_H (B, E^+)$ to find that which has maximum posterior probability. It's the approximation of implication by subsumption test. (If A implies B, then A subsume B. But the inverse is not true.)

2.2 Searching a conceptual hierarchy lattice

The PROGOL system searches a conceptual hierarchy lattice to find the best hypothesis among a set of clauses which subsume the most specific clause. In this search the A^* algorithm is used. The best hypothesis satisfies the four conditions mentioned above and minimizes $I(T \mid E)$. In A^* algorithm, a cost function "f" is associated to each candidate is defined as follows: $f = p - (c + n + h)$.

- p — the number of positive examples which are explained by the candidate clause.
- n — the number of negative examples which are explained by the candidate clause.
- c — The number of atoms in the body of the candidate clause.
- h — the least number of atoms required to complete the relationship between variable in the head of candidate clause.

During the negative search process, many irrelevant branches are pruned by the use of mode information, allowance of literals and number of noise, and the depth of search. In PROGOL, background knowledge is used as a concept hierarchy lattice. It is easy to use definitions on concept hierarchy as a part of background knowledge. This additional information besides individual facts sometimes speeds up and/or proceeds further generalization depends upon how such hierarchical knowledge fits to the target structure. On the contrary, if the given structure does not reflect the target structure, the addition may expand search space and slow down the computation. In other words, the computation cost for deriving a hypothesis increase time when background knowledge increases. The present version of PROGOL cannot handle real data which has large quantities or has various background knowledge.

Therefore, it is worth parallelizing the program to solve the time complexity.

3 Design of parallel PROGOL

In the process of PROGOL, there are the following two main modules.

1. To calculate the most specific clause which explain the examples.
2. To choose the best clause by A^* algorithm taking the most specific clause into account.

For the former computation, MGTP can be used without any significant modification for the other process, when the specific example is generalized to derive hypothesis, the new hypothesis is derived from the rest of the examples after removing those which are explained by the previous hypothesis. This process is repeated until the examples are tested on generalization. In PROGOL the generalization is processed from the topmost clause of the input file. So example sets with different order generate different sets of hypotheses. To design the paralleled PROGOL, there are two approaches. One is to follow the idea of present sequential algorithm of PROGOL, and parallelizes such algorithm as search in concept lattice requiring heavy computation. The other is that all generalizations of hypothesis are derived in parallel. The latter architecture promises fundamental improvement against the current sequential PROGOL. There are many issues to be solved such as the design of communicating hypothesis generalization.

4 Conclusion

In this we described a promising application of MGTP in Inductive Logic Programming. It is very natural to embed MGTP in PROGOL, the most advanced ILP system developed by Muggleton and Feng. Moreover, it is expected to remove a substantial limitation of the current implementation of PROGOL which computes only a Horn clause subset of the most specific

hypothesis clause. We are designing paralleled PROGOL aiming to make it complete and efficient. As we mentioned in the last chapter, there are many issues to be solved to parallelize PROGOL.

There is another possibility to utilize MGTP in removing redundant literals. This problem is left for the future research.

References

[1] Muggleton et al.*PROGOL: Inductice Logic Programming system,* 1993

[2] S.Muggleton, *Foundation of Inductive Logic Programming,* Draft,1993

[3] Ryuzo Hasegawa and Masayuki Fujita, *Parallel Theorem Provers and Their Applications,* Proc. FGCS'92 1992