Design of Database Interface to ILP for building knowledged base.

4J－9 Keiko SHIMAZU, Koichi FURUKAWA, Naoki YAGI
Graduate School of Media and Governance, Keio University,

1 Introduction

One of the current interests in machine learning is whether its techniques can be successfully used in knowledge acquisition, which would be its justification from the engineering point of view [1]. A new school of machine learning called inductive logic programming (ILP) has been recently growing and many attempts have been made applying its new techniques to problems in various fields:

design of medicine, genome analysis, medical diagnosis and so on.

This means that many inductive logic programming applications have been developed as machine learning systems, inductive learner. Further, because of employing background knowledge, ILP may represent a breaking of the "Feigenbaum" bottleneck of knowledge acquisition or knowledge discovery. However, for naive users, it is not easy to apply the system to their own problems. If naive users could apply ILP systems to extract rules or hypotheses as clauses, they could then them as input data for knowledge bases.

2 IGES problem and difficulties of ILP

Initial Graphic Exchange Specification (IGES) format data were used as real data. The format is an international 3D format and it is adopted by such application software as Intergraph [5]. In the IGES database , it is easy to identify the various shapes, (100 for circle, 110 for line, and so on), by their numbers. However, if any composite shapes are constructed by a set of lines, there is no way to identify what kind of shape the data represent, unless we compute the geometry of the shapes. Especially, in the three dimensional representation, it is hard to identify shapes by manual operation with eye inspection without a precise rotation operation.

AS we stated preciously, many inductive logic programming applications have been developed using machine learning systems, but it is not easy for naive users to apply the system to solve their own problems. Mainly, there are two types of difficulty. One is that naive users can not use the ILP system directly on real data, even if they want to generate positive examples. The other difficulty is in generating data sets as negative examples and background knowledge. It is said that a super concept has high relevancy as background knowledge and that near-miss data has high relevancy as a some of negative examples. But still it is hard to prepare an appropriate data set for the ILP system. This means that a large amount of data may have to be created in order to be obtain the target hypothesis.

3 Input Data design

3.1 Preparation of artificial raw data

An obvious solution to this problem is to ask IGES database creators to add oracle type information to sets of points, as comments. An alternative solution is to develop a user interface program displaying 3D objects with suitable rotation operations to make it possible for users identify target objects by visual inspection. In this experimental study, the users specified which figures were diamonds, by using menu interface which we developed. They did this at the time if drawing the shapes. So, we assumed oracle type information and made an artificial IGES database, as raw data to ILP. This was "artificial raw data" with four parts: raw data clauses, component clauses and super concepts (not yet in clause form) of background knowledge, mode declaration clauses, and type information clauses. In our interface to the database, background knowledge has two roles in inductive logic programming. One is for providing the definition of components. The other is for defining a "super concept" for the positive examples. The former is for computing specific data using their instance values.

3.2. Generating positive examples

In our artificial raw data, through the "oracle" operation menu which we have previously described, some line data are described as "diamond". So, by extracting four the line clauses following "diamond" assertion, positive examples were supplied automatically, by programmed operation. Owing to type information and mode declaration, our positive examples were represented by four mode clauses information .

Endo 5322, Fujisawa, Kanagawa, Japan
kshimazu@mag.keio.ac.jp, furukawa@mag.keio.ac.jp, nyagi@mag.keio.ac.jp,

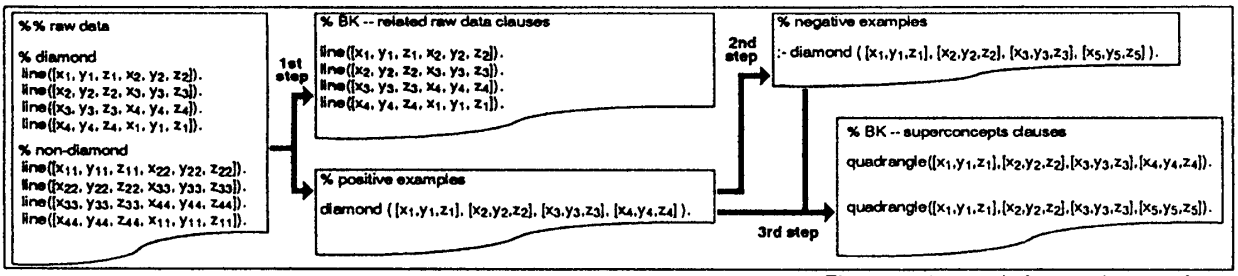

Figure : Automatical generating procedure.

3.3 Generating appropriate negative examples

In our trial, a negative example was restricted to be similar to a positive examples, by having three if its four elements as nodes. To obtain such near miss data, our negative examples were made of positives, which were modified only one element (see the 2nd step in Figure). Our procedure of generating appropriate negative examples is as follows. Note that this is presented as two dimensional data for better understanding.

3.4 Generating background knowledge

Usually, background knowledge has two roles in inductive logic programming. We added another role, as related raw data, to these two background knowledge roles. Through type information, we succeeded in using these related raw data clauses as positive examples. Related raw data clauses were generated from artificial raw data, by extracting sets of line clauses just following a "diamond". This was done at the same time as generating the positive examples (see 1st step in Figure). So, we succeeded in generating appropriate background knowledge (see 3rd step in Figure).

4 Result our experimental study

We inferred the target concept as the rule:

if a quadrangle has nodes , A, B, C and D,
and four square distances of A to B, B to C, C to D and D to A are same,
it must be a diamond.

5 Conclusion

In our experiment, we tried to establish a design principle for supplying appropriate input data to an ILP system, Progol, from a database. We noted the importance of finding an appropriate method of generating positive examples from the database. Usually, a database has no "oracle" for a machine learning system, so it is important to know how to add oracle type information. Our "oracle" generated artificial raw data. By extracting clauses as related raw data to background knowledge, and by setting type information clause and mode declaration, we realized a filter from the raw data to positive examples. Generating negative examples is another critical issue. We were able to show the importance of generating near miss negative examples from the given positive examples. As for background knowledge, the main source should be the given database itself. We also noted the importance of introducing components and a super concept relative to target concepts as items of relevant vocabulary, through inductive inference. Finally, we demonstrated the importance of defining a finite domain in terms of all the constants appearing in a given database. We successfully defined it by using the type information and mode declaration of Progol.

This report concerns the first step of constructing a general interface between a data base and an ILP system. Such an interface has a big possibility of generating a knowledge base cheaply from a huge database .

References

[1] Nada Cavrac and Saso Dzeroski "INDUCTIVE LOGIC PROGRAMMING" pp3-4, 1994.
[2] Stephen Muggleton, et al. 1995
[3] Initial Graphics Exchange Specification, AM AMERICAN NATIONAL STANDARD, ASME Y14.26M-1989
[4] Nada Cavrac and Saso Dzeroski "INDUCTIVE LOGIC PROGRAMMING" pp17-21, 1994.
[5] ©1988,1989,1990,1992,1993,1994 INTERGRAPH CORPORATION
[6] Stephen Muggleton et al. "Foundations of Inductive Logic Programming" pp44-46,1993.
[7] Nada Cavrac and Saso Dzeroski "INDUCTIVE LOGIC PROGRAMMING" pp15-17, 1994.