Multi-Relational Classification Using Inductive Logic Programming

Vimalkumar B. Vaghela[1], Dr. Kalpesh H. Vandra[2], Dr. Nilesh K. Modi[3]
[1]*Ph.D. Scholar, Department of Computer Science & Engineering, Karpagam University, Coimbatore, Tamilnadu, India*
[2]*Director Academic, C. U. Shah College of Engineering & Technology, Surendranagar, Gujarat India*
3Professor & Head, Department of Computer Science, S V Institute of Computer Studies, Gujarat, India

Abstract

Due to the increase in the amount of relational data that is being collected and the limitations of propositional problem definition in relational domains, multi-relational data mining has arisen to be able to extract patterns from relational data. Multi –relational data mining has become popular due to the limitations of propositional problem definition in structured domains and the tendency of storing data in relational databases. Several relational knowledge discovery systems have been developed employing various search strategies, heuristics, language pattern limitations and hypothesis evaluation criteria, in order to cope with intractably large search space and to be able to generate high-quality patterns. Inductive logic programming (ILP) studies learning from examples, within the framework provided by clausal logic. ILP has become a popular subject in the field of data mining due to its ability to discover patterns in relational domains. Several ILP based concept discovery systems are developed which employs various search strategies, heuristics and language patterns. LINUS, GOLEM, CIGOL, MIS, FOIL, PROGOL, ALEPH and WARMR are well-known ILP-based systems.

***Index Terms*— Classification; Multi-relational data mining; Inductive Logic Programming**

1. INTRODUCTION

Due to the impracticality of single-table data representation, relational databases are needed to store complex data for real life applications. This has led to the development of multi-relational learning systems that are directly applied relational data. Relational upgrades of data mining and concept learning systems generally employ first-order predicate logic as representation language for background knowledge and data structures. The learning systems, which induce logical patterns valid for given background knowledge, have been investigated under a research area, called Inductive Logic Programming (ILP). Stephen Muggleton introduced the name Inductive Logic Programming (ILP) that is the intersection of machine learning and logic programming (Muggleton, 1999).

ILP techniques are widely used for classification and concept discovery in the data mining algorithms. In classification, general rules are created according to data and then they are used for grouping the unclassified data. In concept discovery, interesting rules, if exist, are given to the users of the systems.

Several ILP based systems are developed which employs various search strategies, heuristics and language pattern limitations. LINUS, GOLEM, CIGOL, MIS, FOIL, PROGOL, ALEPH and WARMR [1, 4] are well-known ILP techniques.

2. INDUCTIVE LOGIC PROGRAMMING

A. Overview

ILP has ability to use structured, complex and multi-relational data. ILP system represents examples, background knowledge, hypotheses and target concepts in horn clauses logic. ILP techniques are widely used for classification and concept discovery in the data mining algorithms. The core of ILP is the use of logic for representation and the search for syntactically legal hypotheses constructed from predicates provided by the background knowledge. In ILP systems, the training examples, background knowledge and induced hypothesis are all expressed in a logic program form.

Two measures are used to test the quality of the induced theory. After learning, the theory with background knowledge should cover all positive examples (completeness) and should not cover any negative examples (consistency). Completeness and consistency together form correctness [3, 5, 13].

In ILP, a system often starts with an initial pre-processing phase and ends with a post-processing phase. In pre-processing phase, error (noise) in the given examples can be detected and eliminated. In post-

processing phase, redundant clauses in the induced theory are removed in order to improve its efficiency. There are two approaches for the search direction: top-down and bottom-up.

ILP is the study of learning methods for data and rules that are represented in first-order predicate logic. Predicate logic allows for quantified variables and relations and can represent concepts that are not expressible using examples described as feature vectors. A relational database can be easily translated into first-order logic and be used as a source of data for ILP [4]. As an example, consider the following rules, written in Prolog syntax (where the conclusion appears first), that define the uncle relation:

uncle (X,Y) :- brother (X, Z), parent (Z, Y).
uncle (X,Y) :- husband (X, Z), sister (Z, W), parent (W, Y).

The goal of *inductive logic programming* (ILP) is to infer rules of this sort given a database of background facts and logical definitions of other relations [2, 13]. For example, an ILP system can learn the above rules for uncle (the *target predicate*) given a set of positive and negative examples of uncle relationships and a set of facts for the relations parent, brother, sister, and husband (the *background predicates*) for the members of a given extended family, such as:

uncle (tom, frank), uncle (bob, john),
uncle (tom, cindy), ¬uncle(bob, tom)
parent (bob, frank), parent (cindy, frank),
parent (alice, john), parent (tom, john),
brother (tom, cindy), sister (cindy, tom),
husband (tom, alice), husband (bob, cindy).

Alternatively, rules that logically define the brother and sister relations could be supplied and these relationships inferred from a more complete set of facts about only the basic predicates: parent, spouse, and gender. If-then rules in first-order logic are formally referred to as *Horn clauses*. A more formal definition of the ILP problem follows:

- **Given:**
  - Background knowledge, B, a set of Horn clauses.
  - Positive examples, P, a set of Horn clauses (typically ground literals).
  - Negative examples, N, a set of Horn clauses (typically ground literals).
- **Find:** A hypothesis, H, a set of Horn clauses such that:
  - $\forall p \in P: H \cup B \models p$ (completeness)
  - $\forall n \in N: H \cup B \not\models n$ (consistency)

A variety of algorithms for the ILP problem have been developed [3, 10, 13] and applied to a variety of important data mining problems [2, 10, 12]. Nevertheless, relational data mining remains an under-appreciated topic in the larger KDD community.

B. Search Strategies

Two basic steps in the search for correct theory are specialization and generalization. If a theory covers negative examples, it means that it is too strong, it needs to be weakened. In other words, a more specific theory should be found. This process is called specialization. If a theory does not imply all positive examples, it means that it is too weak, it needs to be strengthened. In other words, a more general theory should be found. This process is called generalization. Specialization and generalization steps are repeated to adjust the induced theory in the overall learning process.

There are two approaches for the search directions: top-down and bottom-up. Top-down approach starts with an overly general theory and tries to specialize it until it no longer covers negative examples. Specialization (refinement) operators [4, 5, 11] employ two basic operations on a clause: apply a substitution to the clause and add a literal to the body of the clause. Refinement graph is the most popular data structured used in specialization. Bottom-up approach starts with an overly specific theory and tries to generalize it until it cannot further be generalized without covering negative examples. Generalization operators perform two basic syntactic operations on a clause: apply an inverse substitution to the clause and remove a literal from the body of the clause. Relative least general generalization (rlgg) and inverse resolution are basic generalization techniques. Generalization techniques search the hypothesis space in a bottom-up manner. rlgg used in GOLEM and inverse resolution used in CIGOL.

C. Algorithm: Generic ILP based concept learner

Require: TR: Target Relation, B: Background Knowledge

Ensure: H: hypothesis describing the target relation
Start with some initial theory H
repeat

if H is too strong **then**
Specialize H
endif
if H is too weak **then**
Generalize H
endif
until H is consistent and complete
return H

A generic algorithm for an ILP-based concept learner is presented in Algorithm. Target instances are defined as positive, belonging to the target concept, and negative, not belonging to the target concept. Hypothesis is called strong if it does not cover all the positive examples and weak if it covers some negative examples. The induced hypothesis is called complete if it covers all the positive examples, and consistent if it covers none of the negative examples.

3. WELL-KNOWN ILP-BASED SYSTEM

A. Bottom-up System

1. GOLEM

Golem is an inductive logic programming algorithm developed by Stephen Muggleton and Feng. It uses the technique relative least general generalization proposed by Gordon Plotkin. Therefore, only positive examples are used and the search is bottom-up. Negative examples can be used to reduce the size of the hypothesis by deleting useless literals from the body clause.

In order to generate a single clause, GOLEM first randomly picks several pairs of positive examples, computes their rlggs and chooses the one with greatest coverage. If the final clause does not cover all positives, the covering approach will be applied. The covered positives are removed from the input and the algorithm will be applied to the remaining positives (Lavrac˘ & Dz˘eroski, 1994; Muggleton & Feng, 1990).

2. CIGOL

CIGOL (logic backwards) is interactive bottom-up relational ILP system based on inverse resolution. CIGOL employs three generalization operators which are relational upgrades of absorption, intra-construction and truncation operators. The basic idea is to invert the resolution rule of deductive inference using the generalization operator based on inverse substitution. CIGOL uses the absorption operator. However, CIGOL also needs oracle knowledge to direct the induction process.

B. Top-Down System

1. LINUS

LINUS is one of the most popular attribute-value learning environments in the ILP history. LINUS is a framework that reduces the relational learning problem into a propositional one, employs an attribute-value learning method and transforms the solution hypothesis into relational form. It is a non-interactive ILP system, integrating several ILP attribute-value learning algorithm in a single environment. It can be viewed as a toolkit, in which one or more of the algorithm can be selected in order to find the best solution for the input. The main algorithm behind LINUS[6][7] consists of three steps. In the first step, the learning problem is transformed from relation to attribute-value form. In the second step, the transformed learning problem is solved by an attribute-value learning method. In the final step, the induced hypothesis is transformed back into relational form.

2. MIS

Model Inference System (MIS) is an interactive top-down relational ILP system, which uses refinement graph in the search process (Shapiro, 1983). In its algorithm, at the beginning the hypothesis is empty (H = Φ).Then it reads the examples (either positive or negative) one by one. If the example is negative and covered by some clauses in the Hypothesis set, then incorrect clauses are removed from the solution set. If the example is positive and it is not covered by any clause in the solution set, with breadth-first search, a clause c, which covers the example [6], is developed and added to solution set. The process will continue until the solution set (H) becomes complete and consistent (Lavrac˘ & Dz˘eroski, 1994).

3. FOIL

First-Order Inductive Learner (FOIL) is a non-interactive top-down relational ILP system, which uses refinement graph in the search process as in MIS. It uses the covering approach for the solution having more than one clause. FOIL is a sequential covering algorithm that builds rules one at a time. After building a rule, all positive target tuples satisfying that rule are removed and FOIL will focus on tuples that have not covered by any rule. When building each rule, predicates are added one by one. At each step, every possible predicate is evaluated, and the best one is appended to the current rule. FOIL chooses the clause according to weighted information gain criteria.

4. PROGOL

PROGOL is a top-down relational ILP system, which is based on inverse entailment (Muggleton, 1995; Muggleton & Tamaddoni-Nezhad, 2008).. It

performs a search through the refinement graph. Besides a definite program B as background knowledge and a set of ground facts E as examples, PROGOL requires a set of mode declarations for reducing the hypothesis space.

5. ALEPH

A Learning Engine for Proposing Hypotheses (ALEPH) (Srinivasan, 1999) is a top-down relational ILP system based on inverse entailment similar to PROGOL. The basic algorithm is the same as PROGOL algorithm whereas in the advanced use of ALEPH, it is possible to apply different search strategies, evaluation functions and refinement operators. It is also possible to define more settings in ALEPH such as minimum confidence and support. In the advanced use of ALEPH, contrary to PROGOL, it is possible to select more than one example as a sample at the beginning of the algorithm. The best clause obtained from reducing corresponding bottom clause is then added to the theory. The basic PROGOL algorithm is a ‘‘batch’’ learner in the sense that all examples and background are expected to be in place before learning commences. An incremental mode allows ALEPH to acquire new examples and background information by interacting with the user. The basic PROGOL algorithm does a fairly standard general to specific search and constructs a theory one clause at a time. ALEPH allows the basic procedure for theory construction to be altered in a number of ways by using randomized search methods.

Minpos and minacc are the two parameters representing minimum support and confidence criteria in ALEPH. The default value for minpos is 1 and it sets a lower bound on the number of positive examples to be covered by an acceptable clause. If the best clause covers positive examples below this number, then it is not added to the current theory. The default value for minacc is 0.0 (domain of minacc is the set of floating-point numbers between 0 and 1) and it sets a lower bound on the minimum accuracy of an acceptable clause.

6. WARMR.

Design of algorithms for frequent pattern discovery has become a popular topic in data mining. Almost all algorithms have the same of level-wise search known as APRIORI algorithm (Agrawal, Mannila, Srikant, Toivonen, & Verkamo, 1996). The level-wise algorithm is based on a breadth-first search in the lattice spanned by a specialization relation between patterns (Dehaspe & Raedt, 1997; Dehaspe & Toivonen, 2001). The APRIORI method looks at a level of the lattice at a time. It starts from the most general pattern. It iterates between candidate generation and candidate evaluation phases. In candidate generation, the lattice is used for pruning non-frequent patterns from the next level. In candidate evaluation, frequencies of candidates are computed with respect to database. Pruning is based on the monotonicity property with respect to frequency: if a pattern is not frequent then none of its specializations are frequent.

WARMR (Dehaspe & Raedt, 1997; Dehaspe & Toivonen, 2001) is a non-interactive descriptive ILP system that employs APRIORI rule as search heuristics. Therefore, it is not a predictive system, i.e. it does not define the target relation. Instead, it can find the frequent queries including the target relation. Then, it is possible to extract association rules having target relation in the head according to confidence criteria. The target relation is defined as the key relation in WARMR. In WARMR algorithm, at the beginning there are three sets: candidate queries (Q), frequent queries (F) and infrequent queries (I). Q is initialized as having the key predicate. F and I are initialized as empty set. In the first level, the specializations of the item in Q are generated according to language bias (warmode is similar to mode declaration in PROGOL). They are put into current Q set. After this, frequency values of the items in Q are evaluated and infrequent items are put into I and frequent items are put into F. In the next level, Qset is generated according to previous contents of Q, F and I set. The language bias (warmode) defines the types and modes of the parameters of the predicates. The user can define the warmode in the settings file in the input data.

4. LANGUAGE BIAS

ILP algorithms usually use one of the following relational languages:

- general clauses language,
- Horn clauses language.

Construction of the hypothesis in the language frameworks is not always possible, because of the following reasons:

- hypotheses space is huge and/or complex,
- the language used is not expressive enough.

To solve this problem two types of bias are used – a mechanism employed by a leaner to constrain the search for hypotheses:

- language bias – determines the search space itself,

TABLE I:
Language bias in ILP-based systems

System	Language bias					
	mode declarations (input/output) types of the predicates' arguments	function free clauses	ground clauses	ground literals	non-recursive clauses	Horn clauses
LINUS				L_E	L_H	L_H
GOLEM				L_E, L_B		L_H
CIGOL				L_E	L_H	L_H
MIS	L_E, L_H		L_E	L_B		L_H
FOIL	L_E, L_H	L_E		L_B		L_H
PROGOL	L_E, L_H	L_E		L_B		L_H
WARMAR	L_E, L_H	L_E		L_B		L_H, L_B, L_E

TABLE II:
Comparison of ILP-based systems

System	Search Direction	Basic Techniques	Use of mode	Use of negative	Allow recursive
LINUS	Top-down	Attribute-Value Learning	No	No	No
GOLEM	Bottom-up	Rlgg	Yes	Yes	No
CIGOL	Bottom-up	Inverse Resolution	No	No	No
MIS	Top-down	Refinement Graph	Yes	Yes	Yes
FOIL	Top-down	Refinement Graph	Yes	Yes	Yes
PROGOL	Top-down	Inverse Entailment	Yes	Yes	Yes
WARMAR	Top-down	APRIORI	Yes	Yes	Yes

- search bias – determines how the hypothesis space is searched.

There are two categories of language bias:

- the syntactic restrictions of the selected logic formalism,
- the vocabulary of predicate, function, variables and constant symbols: function free clauses, ground clauses (e.g. without variables), non-recursive clauses, mode declarations (input/output) of the predicates' arguments.

To represent examples, hypotheses and BK in the learning task examples' language (L_E), hypotheses language (L_H), and BK language (L_B) have been used.

Each of language restriction above mentioned could be applied to each of these languages independently, or to all of them together (Table 1).

All ILP systems use some language bias. Mode declarations and learning of non-recursive clauses are necessary for narrowing search in the hypotheses space, but other language restrictions are imposed from the theory. For example, such a hypothesis does not exist in general case when both set of examples and BK set consist of Horn clauses.

5. COMPARISON OF ILP-BASED SYSTEMS

In this section, we compare the aforementioned ILP systems in terms of basic techniques they employ, availability of basic features in concept discovery and their concept discovery performance in different domains. In comparison, we included the features of search direction, use of mode declaration, use of negative data and handling recursive rules.

Search direction of the systems is either top-down or bottom-up. In top-down systems, the search starts with a general clause and at each turn, it makes the clause more specific until it covers no negative examples. On the other hand, in bottom-up systems, the search starts with a specialized clause, and at each turn, it generalizes the clause so that it covers more positive examples, until no more improvement is possible.

6. ACCURACY AND TIME CHARACTERISTICS

The following characteristics are measured in the classical chess and endgame domain "White King and Rook versus Black King". The results of the experiment are presented in the following table III: the classification accuracy is given by the percentage of correctly classified testing instance and by the standard deviation (sd), averaged over 5 experiments.

TABLE III:
Experimental comparison of ILP-based systems

System	100 training examples		1000 training examples	
	Accuracy	Time	Accuracy	Time
CIGOL	77.2%	21.5 h	N/A	N/A
FOIL	90.8%	31.6 s	99.4%	4.0 min
LINUS	98.1%	55.0 s	99.7%	9.6 min
PROGOL	95.3%	53.9 s	99.6%	8.3 min

Although LINUS is better than other algorithms in small and large training sets, it has one major disadvantage - does not provide features for handling BK. From the rest algorithms PROGOL has better accuracy, but it is slower.

7. CONCLUSION

This work aims to present a review of ILP-based concept discovery systems through illustrations of the working mechanisms of these systems. The well-known systems, LINUS, GOLEM, CIGOL, MIS, FOIL, PROGOL, ALEPH and WARMR, are the systems that are covered in this comparative study. The comparison is based on basic characteristics of the systems. Although search strategies of FOIL and its family algorithms make them very efficient, they have a considerable disadvantage these algorithms in the search process sometimes can prune searched hypotheses.

If the target concept is not clear enough, using WARMR is a wise choice in order to see various frequent clauses hidden in the data. If the user is familiar with mode declarations, search and evaluation mechanisms, PROGOL and ALEPH are suitable choices. User may try different mode declarations, search and evaluation mechanisms to find the settings and rule structures that best identify the target concept. Many inverse resolution algorithms increase the concept description language by constructing predictor descriptors (i.e., predicates), but are either limited to deduction or require an oracle to maintain reasonable efficiency.

ACKNOWLEDGMENT

We are thankful to the great GOD for making us able to do something.

This research is the part of Ph.D. programme in Computer Science & Engineering, Karpagam University, India.

REFERENCES

[1] Arno J. Knobbe, Arno Siebes, Hendrik Blockeel, Daniël van der Wallen, "Multi-Relational Data Mining, using UML for ILP", PKDD '00 Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery Springer-Verlag London, UK ©2000

[2] Yusuf Kavurucu, Pinar Senkul, Ismail Hakki Toroslu," AGGREGATION IN CONFIDENCE-BASED CONCEPT DISCOVERY FOR MULTI-RELATIONAL DATA MINING", IADIS European Conference Data Mining 2008.

[3] Raymond J. Mooney, Prem Melville, Lappoon Rupert Tang, "Relational Data Mining with Inductive Logic Programming for Link Discovery",*National Science Foundation Workshop on Next Generation Data Mining, Nov. 2002, Baltimore, MD.*

[4] Yusuf Kavurucu, Pinar Senkul, Ismail Hakki Toroslu, "Confidence-based Concept Discovery in Multi-Relational Data Mining", International MultiConference of Engineers and Computer Scientists 2008 Vol I IMECS 2008, 19-21 March, 2008, Hong Kong.

[5] Lappoon R. Tang, Raymond J. Mooney, and Prem Melville, "Scaling Up ILP to Large Examples: Results on Link Discovery for Counter-Terrorism", KDD-2003 Workshop on Multi-Relational Data Mining (MRDM-2003), pp.107-121, Washington DC, August, 2003.

[6] Seda Daglar Toprak,Pinar Senkul, "A New ILP-based Concept Discovery Method for Business Intelligence", 2007, IEEE.

[7] Alexessander Alves, Rui Camacho and Eugenio Oliveira, "Discovery of Functional Relationships in Multi-relational Data using Inductive Logic Programming", Fourth IEEE International Conference on Data Mining (ICDM '04).

[8] Dzeroski, S., Lavtac, N. 2001. eds, Relational data mining, Berlin: Springer.

[9] Han, J., Kamber, M. 2007. Data Mining: Concepts and Techniques", 2nd Edition, Morgan Kaufmann.

[10] Wrobel S, "Inductive Logic Programming for Knowledge Discovery in Databases: Relational Data Mining", Berlin: Springer, pp.74-101, 2001.

[11] I. Bratko. Prolog Programming for Arti_cial Intelli-gence, 3rd edition. Addison-Wesley, Harlow, England, 2001.

[12] W. Buntine. Generalized subsumption and its applications to induction and redundancy. Artificial Intelligence, 36(2): 149-176, 1988.

[13] L. De Raedt, editor. Advances in Inductive Logic Programming. IOS Press, Amsterdam, 1996.

AUTHOR'S PROFILE

Prof. Vimalkumar B Vaghela, is currently doing his Ph.D. in Computer Science & Engineering at Karpagam University, India. This author is Young Scientist awarded from Who's Whos Science & Engineering 2010-2011 & also his biography is included in American Biographical Institute in 2011. His publication is also available in ieeeexplorer and also in spocus online database. He is currently working as a Assistant Professor in Computer Engineering Department at L. D. College of Engineering, Ahmedabad, Gujarat, India. He received the B.E. degree in Computer Engineering from C. U. Shah College of Engineering and Technology, in 2002 & M.E. degree in Computer Engineering from Dharmsinh Desai Univrsity, in 2009. His research areas are Relational Data Mining, Ensemble Classifier, Pattern Mining. Author has published / presented more than 5 international papers and 5 national papers.

Dr. Kalpesh H Vandra received the B.E. degree in Electronics & Communication form North Gujarat University, Patan, in 1995, the M.E degree in Microprocessor System Applications from M.S. University, Vadodara, in 1999 and PhD from Saurashtra University, Rajkot, in 2009. He is working as Director Academic and Head Of Computer Engineering & Information Technology at C. U Shah College of Engineering & Technology, wadhwan city, Gujarat, INDIA, Author having more than 15 years of UG & 6 years of PG (MCA and M.E) Teaching Experience. Author had written more than 10 Books related to Computer & IT related area. He has published / presented 10 International and 7 national Research Papers & Guided more than 10 PG Students and more than 155 UG Students Dissertation work. Dr. K H.Wandra is a Chairman of Board of Studies Computer Engineering At Saurashtra University, Rajkot, Core Committee member For Syllabus of UG & PG at Gujarat Technology University, Ahmedabad. Section Managing Committee Member of ISTE Gujarat Section, Life member of ISTE, member of IEEE and CSI, Interested for working in the area of Wireless communication, Networks, Advanced Microprocessors, Data Mining.

Dr. Nilesh K Modi received the MCA degree from A.M.P. Institute of Computer Studies, Kherva, Gujarat, India and PhD from Bhavnagar University in 2006. He is working as a professor & head of MCA department in Sarva Vidyalaya's Institute of Computer Studied, S V Campus, Kadi, Gujarat, India. He is Associate Life Member in Computer Society of India (CSI) Mumbai, Senior Associate Member in International Association of Computer Science and Information Technology (IACSIT) Singapore, Senior Member in International Association of Engineers (IAEng) Hong Kong, Senior Member in Computer Security Institute New York, Member in Data Security Council of India (DSCI) a NASSCOM initiative New Delhi. Author has published / presented more than 18 international papers and more than 25 national papers. His areas of research interest are Data mining, Computer network, information security.