Knowledge Discovery in Spatial Data by Means of ILP

Luboš Popelínský

Faculty of Informatics, Masaryk University Brno and
Faculty of Electrical Engineering, CTU Prague
Czech Republic

Email: popel@fi.muni.cz

Abstract. We show that inductive logic programming(ILP) is a powerful tool for spatial data mining. We further develop the direction started (or symbolised) by *GeoMiner* [9] and argue that the technique developed for database schema design in deductive object-oriented databases is fully usable for spatial mining and overcome, in expressive power, some other mining methods. An inductive query language, with richer semantics, is proposed and three kinds of inductive queries are described. Two of them are improved versions of *DBMiner* [8] rules. The third kind of rules, dependency rules, allow to compare two or more subsets. Then a description of *GWiM* mining system as well as results reached by the system are given. We conclude with discussion of weaknesses of the method. [1]

1 Knowledge Discovery in Spatial Data

Knowledge discovery in geographic data is, no doubts, challenging and very important. However, the classical KDD, either statistical or based on machine learning, are not convenient for the task. The spatial data have to be managed by means that respect(and exploit) their structural nature. Moreover, non-spatial data need to be used, too, e.g. to find a region with some non-spatial characteristics. Main tasks when mining geographic data [13] are, among others, understanding data, discovering relationship as well as (re)organising geographic databases. This paper addresses those three tasks. We show that inductive logic programming(ILP) is a powerful tool for spatial data mining.

2 Mining Knowledge by Means of ILP

Inductive Logic Programming (ILP) [14] is a research area in the intersection of machine learning and computational logic. The main goal is development of a

[1] The initial portion of this work was done during author's stay in LRI Université Paris-Sud thanks to French government scholarship. This work has been partially supported by Esprit LTR 20237 Inductive Logic Programming II ILP^2.

theory and algorithms for inductive reasoning in first-order logic. ILP aims to construct a theory covering given facts. Given a set of positive examples E^+, a set of negative examples E^-, we construct a logic program P such that $P \vdash E^+$ and $P \nvdash E^-$. In case of noisy data we aim at a first-order logic formula that describes a significant majority of positive examples and may be a non-significant part of negative examples.

Since early 90'th there are attempts to exploit ILP in knowledge discovery in databases [4]. *CLAUDINE* [2] can learn both data dependencies and integrity constraints in relational databases. A problem of mining association rules in multiple relations has been solved in [3]. Interactive system that provides support for inductive database design is presented in [1].

■

In [15] we addressed the possibilities of inductive logic programming (ILP) in restructuring object-oriented database schema. In deductive object-oriented databases, both classes and attributes as well as methods may be defined by rules. We showed that inductive logic programming can help in synthesis of those rules to support the database schema design and modification.

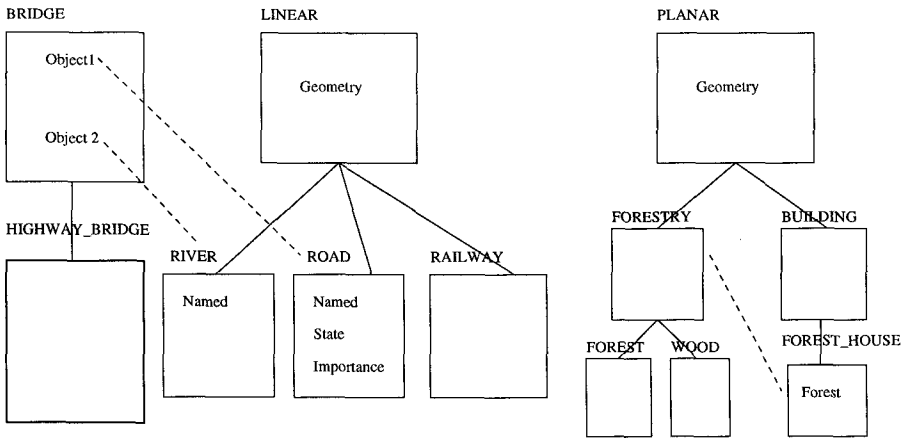
Exploiting that method we further develop the direction started(or symbolised) by *GeoMiner* [9]. We argue that the technique developed for database schema design in deductive object-oriented databases is fully usable for spatial mining. *GWiM* system is presented that outperforms in some aspects *GeoMiner*. Namely *GWiM* can mine a richer class of knowledge, Horn clauses. Background knowledge used in *GeoMiner* may be expressed only in the form of hierarchies. *GWiM* accepts any background knowledge that is expressible in a subset of object-oriented F-logic [12].

In the Section 3. we demonstrate the method on the simple mining task. Then an inductive query language is described. A description of *GWiM* mining system is given in Section 5. We conclude with discussion of results and mainly the weaknesses of the method. We propose solutions how to improve the mining method.

3 Example

We will demonstrate a data mining task on the database in Pic.1 . The *BRIDGE* class consists of all road bridges over rivers. Each bridge has two attributes – `Object1` (a road) and `Object2` (a river). Each river (as well as roads and railways) inherits an attribute *Geometry* (a sequence of (x,y) coordinates) from the class *LINEAR*. Objects of a class *RIVER* has no more attributes but `Named` of the river. In a class *ROAD*, the attribute `State` says whether the road is under construction (state=0) or not. The `Importance` defines a kind of the road: 1 stands for highways, 2 for other traffic roads, and 3 for the rest(e.g. private ones).

Our goal is to find a description of the class *HIGHWAY_BRIDGE* in terms of other classes in the given database.



Pic. 1: Object-oriented database schema

That task is expressed in inductive query language (for a full description see Section 4) as

extract characteristic rule
for highway_bridge.

Let us have 2 rivers with object identifiers `river1,river2`, named `Svratka` and `Svitava`, and two roads (with object identifiers `road1, road2`) a highway `D1` and a state road `E7` that cross those rivers. E.g. an object `bridge1` can be expressed in first order logic(FOL) [2] as

```
bridge(bridge1) :- object1(bridge1,road1),
   object2(bridge1,river1).

road(road1) :- named(road1,'D1'),
   geometry(road1,[(375,500),(385,350), ... ]),
   state(road1,1),importance(road1,1).

river(river1) :- named(river1,'Svratka'),
   geometry(river1,[(0,40),(40,60),(77,76), ... ]).
```

All instances of the class *HIGHWAY_BRIDGE* are as positive examples. The rest of members of the class *BRIDGE* serves as negative examples. Then the expected results is as

```
highway_bridge(X):-object1(X,Y),importance(Y,1).
```

GWiM starts with a minimal language which consists of `object1, object2` attributes of the class *HIGHWAY_BRIDGE* itself. The best clause being found

[2] Actually, *GWiM* works with object-oriented F-logic. Here we prefer to use FOL what is an internal representation of *GWiM* system. More on translation from/to F-logic, see [15].

```
highway_bridge(X):-bridge(X),object1(X,Y),object2(X,Z).
```

is over-general as it covers all negative examples. In that case when no solution is found, the language is extended by adding attributes from neighbouring classes(either super/subclasses or referenced classes). If there is a referenced class, the most general superclass is added first. In our case, an attribute `geometry` of the class *LINEAR* (both for *RIVER* and *ROAD*) has been added. As it does not help, the language has been further enriched by attributes of classes *RIVER*, *ROAD*, i.e. `State`, `Importance` and `Named`. For this language, *GWiM* has eventually found a logic formula that successfully discriminates between positive and negative examples.

4 Language for Spatial Data Mining

In this section we present three kinds of inductive queries. Two of them, that ask for characteristic and discriminate rules, are adaptation of *DBMiner* [8] rules. The dependency rules add a new quality to the inductive query language. The general syntactic form, adapted from *DBMiner* of the language is as follows.

extract < KindOfRule > **rule for** < NameOfTarget >
[**from** < ListOfClasses >] [< Constraints >]
[**from point of view** < Explicit Domain Knowledge >] .

Semantics of those rule differs from that of *DBMiner*. Namely < Explicit Domain Knowledge > is a list of predicates and/or hierarchy of predicates. At least one of them has to appear in the answer to the query. Actually a clause **from point of view** enables to specify a criterion of interestingness [5]. The answer to those inductive queries is a first-order logic formula which characterises the subset of the database which is specified by the rule.

Characteristic Rule. Characteristic rules serve for a description of a class which exists in the database or for a description of a subset of a database. The example of that kind of rule has been shown in Section 3 . See 6.1, too.

Discriminate Rule. Discriminate rules find a difference between two classes which exist in the database, or between two subsets of the database. They allow to find a quantitative description of a class in contrast to another one.

extract discriminate rule
for < NameOfClass >
[**where** < ConstraintOnClass >]
in contrast to < ClassOfCounterExamples >
[**where** < ConstraintOnCounterExamples >]
[**from point of view** < DomainKnowledge >] .

Positive examples of the concept < NameOfTarget > are described by

for < NameOfClass >
where < ConstraintOfListOfClasses >

negative examples are described by

from < ListOfClasses >
in contrast to < ClassOfCounterExamples >
where < Constraint On Counterexamples >

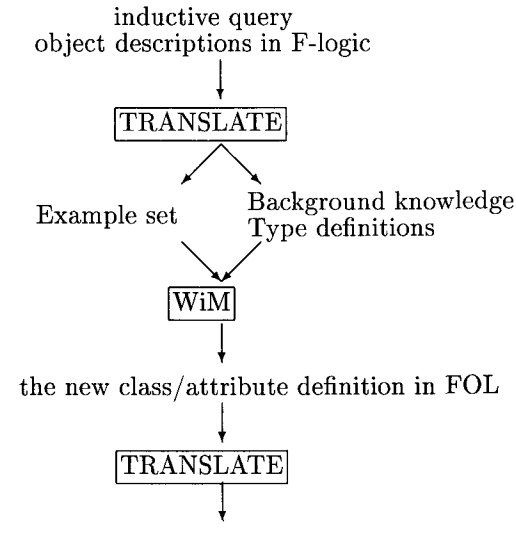
E.g. forests have an area greater than 100 hectares. Woods serve as counterexamples there. For an example, see 6.2.

Dependency Rule. Dependency rules aim to find a dependency between different classes. In opposite to discriminate rules, dependency rules look for a qualitative characterisation of a difference between two classes.

extract dependency rule
for < NameOfClass >
from < ListOfClasses >
[**where** < ConstraintOnClasses >]
[**from point of view** < DomainKnowledge >] .

The objects are defined by the **from ... where ... from point of view ...** formula. The target predicate < NameOfTarget > is of arity equal to a number of classes in < ListOfClasses >. E.g. for forests and woods, an area of a forest is always greater than an area of a wood. See 6.3 for an example.

5 Description of *GWiM*

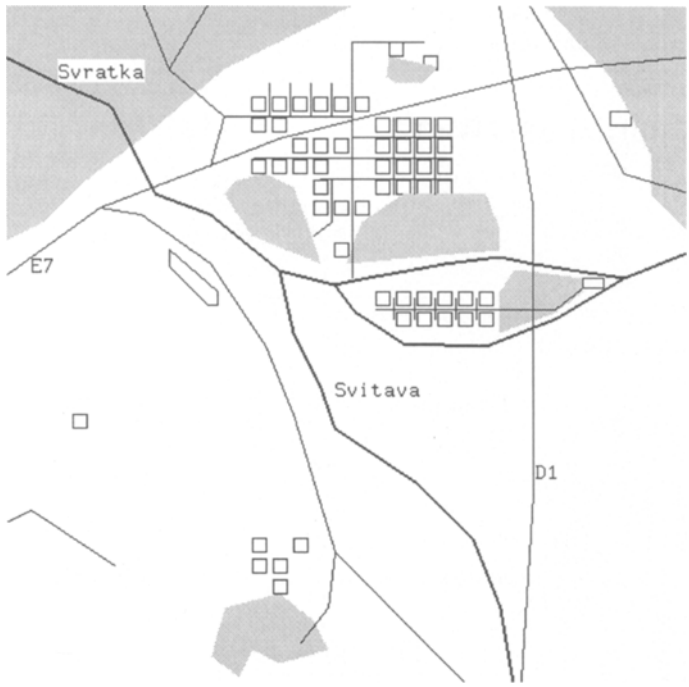


Pic. 2: *GWiM* **schema**

GWiM is built upon the *WiM* ILP system. *WiM* [6, 11], a program for synthesis of closed Horn clauses further elaborates the approach of *MIS* [17] and *Markus* [7]. It works in top-down manner and uses shift of language bias. Moreover, a second-order schema, as a part of *WiM* truth maintenance system, can be defined which the learned program has to much. This schema definition can significantly increase an efficiency of the learning process because only the synthesised programs which match the schema are verified on the learning set.

6 Results

In this section we will demonstrate (in)capabilities of *GWiM*. The geographic data set contains descriptions of 31 roads, 4 rails, 7 forest/woods, and 59 buildings [3]. Geographic data are on the Pic.3. The thick lines are rivers. Learning sets that has been generated from those raw data by *GWiM* are in the following table.



Pic. 3: Raw data

[3] See http://www.gmd.de/ml-archive/frames/datasets/ilprev/ilprev-frames.html for more information

	# positive	# negative
bridge	2	1
forest	3	4
forestOrWood	7	7

Tab.1: Summary of results

In following paragraph, particular mining tasks are described.

6.1 Bridge with Additional Domain Knowledge

Find a description of bridge in terms of attributes of classes *road*, *river*, using a predicate *common*(X, Y). That predicate succeeds if geometries X and Y has a common point.

extract characteristic rule
for bridge
from road, river
from point of view common.

```
bridge(X,Y):-
 object1(X,Z),geometry(Z,G1),
 object2(Y,U),geometry(U,G2),
 common(G1,G2).
```

6.2 Discrimination of Forests and Woods

Find a difference between forests and woods from the point of view of area. *area* is the name of set of predicates like *area*$(Geometry, Area)$.

extract discriminate rule
for forest
in contrast to wood
from point of view area.

```
forest(F) :-
 geometry(F,GForest),
 area(GForest,Area),
 100 < Area.
```

6.3 Relation between Forests and Woods

Find a relation between forests and woods from the point of view of area. *area* is the name of set of predicates like *area*$(Geometry, Area)$.

extract dependency rule
for forestOrWood
from forest, wood
from point of view area.

```
forestOrWood(F,W) :-
 geometry(F,GF),area(GF,FA),
 geometry(W,GW), area(GW,WA),
 WA<GA.
```

7 Towards Discovery in Real Databases

The query language is quite powerful. It means that even quite complex queries can be formulated. However, the price that user has to pay for is sometimes too big. Namely an explicit domain knowledge has to be declared almost exactly (see examples above). To avoid that drawback, more meta-knowledge is needed,

both on the particular database and on the spatial relations. We point out that the meta-knowledge can be easily expressed in F-logic and exploited in *GWiM* truth maintenance system.

Another fault concerns *GWiM*'s incapability to process large amount of data. However, even there is a way out. The general-to-specific learning method of *GWiM* consists of 2 phases. In the first step a promising clause is generated which is, in the second step, tested on the example set. Complexity of the generation step does not practicaly depend on cardinality of the example set. Moreover, in the testing step, it seems be straightforward to employ the database management system(DBMS) itself. Actually, we only need to know how many examples (both positive and negative) is (un)covered by the promising clause. It remains to implement a communication channel connecting an ILP system, with the DBMS. *WiM* can work in interactive mode [16]. There are oracles that allow to ask for examples an external device and even evaluate a given hypothesis on external data. A hypothesis is first translated into a (sequence of) SQL queries. The answers are then evaluate and the hypothesis is either specialised(if some positive examples are newly covered by the hypothesis and some negative examples are covered, too) or accepted as an answer to the particular inductive query.

We will demonstrate it again on the same example as in Section 3. Let us have 2 positive examples and 1 negative example and let the minimal success rate for a hypothesis to be accepted be set to 1 – all positive examples need to be covered and none of negative ones does. An initial portion of examples is randomly chosen from the database employing *random oracle*. Let we ask for just 1 positive example. The learned hypothesis

```
highway_bridge(X) :- bridge(X),object1(X,Y),object2(X,Z)
```

has to be verified using the full database. A *success rate oracle* is called that returns the percentage of correctly covered positive examples and correctly uncovered negative examples. The result is 2/3 as there is 1 negative example covered. Thus the hypothesis is over-general and need to be further specialised. The process continues until the limit for a minimal success rate is reached.

8 Conclusion

We have presented *GWiM* system for mining spatial data. The system outperforms in some aspects *GeoMiner*, namely it can mine a richer class of knowledge, Horn clauses. *GWiM*, too, accepts any background knowledge that is expressible in a subset of object-oriented F-logic i.e. richer than those one exploited by *GeoMiner*. It seems not to be hopeless to apply that approach in a warehouse technology [10]. The most serious problem is the computational complexity of the current implementation. A way how to solve it as been proposed in the previous section.

Acknowledgements

The main part of the work has been done during my stay at LRI Université Paris-Sud. Thanks are mainly due to Yves Kodratoff and I&A group members. I thank, too, database research team at LRI Universite Paris-Sud for their assistance. The first version of *GWiM* has been implemented by Petr Lecián. Last but not least, I thank to Olga Štěpánková and members of Gerstner Laboratory at CTU Prague for fruitful discussions and to anonymous referees for their comments.

References

1. Blockeel H., De Raedt L.: Inductive Database Design. Accepted for Applied Artificial Intelligence, special issue on first order knowledge discovery in databases.
2. Dehaspe L., Van Laer W., De Raedt L.: Applications of a logical discovery engine. In: [18].
3. Dehaspe L., De Raedt L.: Mining Association Rules in Multiple Relations. In Proc. of ILP'97 pp.125-133, LNAI Springer-Verlag
4. Džeroski S., Lavrač: Inductive Learning in Deductive Databases. IEEE Trans. on Knowledge and Data Engineering, Vol 5. No. 6, December 1993.
5. Fayyad, U.M., Piatetski-Shapiro G., Smyth P.: From Data Mining to Knowledge Discovery: An Overview. In: Advances in Knowledge Discovery and Data Mining. AAAI Press/ The MIT Press, Menlo Park, California (1996) 1-34.
6. Flener P., Popelínský L. Štěpánková: ILP nad Automatic Programming: Towards three approaches. In: [18].
7. Grobelnik M.: Induction of Prolog programs with Markus. In Deville Y.(ed.) Proceedings of LOPSTR'93. Workshops in Computing Series, pages 57-63,Springer-Verlag, 1994.
8. Han et al.: DMQL: A Data Mining Query Language for Relational Databases. In: ACM-SIGMOD'96 Workshop on Data Mining
9. J. Han, K. Koperski, and N. Stefanovic: GeoMiner: A System Prototype for Spatial Data Mining, Proc. 1997 ACM-SIGMOD Int'l Conf. on Management of Data(SIGMOD'97), Tucson, Arizona, May1997(System prototype demonstration)
10. Kouba Z., Matoušek K., Mikšovský P., Štěpánková O. : On Updating the Data Warehouse from Multiple Data Sources. In: Proceedings of DEXA'98, Vienna 1998.
11. Lavrač N., Džeroski, Kazakov D., Štěpánková O.: ILPNET repositories on WWW: Inductive Logic Programming systems, datasets and bibliography. AI Communications Vol.9, No.4, 1996, pp. 157-206 .
12. Kifer M., Lausen G., Wu J.: Logical Foundations of Object-Oriented and Frame-Based Languages. TR 93/06, Dept. of Comp. Sci. SUNY at Stony Brook, NY, March 1994 (accepted to Journal of ACM).
13. Koperski K., Han J., Adhikary J.: Mining Knowledge in Geographical Data. To appear in Comm.of ACM 1998
14. Muggleton S., De Raedt L.: Inductive Logic Programming: Theory And Methods. J. Logic Programming 1994:19,20:629-679.
15. Popelínský L.: Inductive inference to support object-oriented analysis and design. In: Proc. of 3rd JCKBSE Conference, Smolenice 1998, IOS Press.
16. Popelínský L.: Interactive *WiM*: User Guide. 1998
17. Shapiro Y.: Algorithmic Program Debugging. MIT Press, 1983.
18. Wrobel S.(ed.): Proc. of 4th Workshop on Inductive Logic Programming ILP'94, Bad Honeff,Germany, 1994.