Detecting traffic problems with ILP

Sašo Džeroski[1], Nico Jacobs[2], Martin Molina[3], Carlos Moure[3], Stephen Muggleton[4], Wim Van Laer[2]

[1] J. Stefan Institute, Jamova 39, SI-1000 Ljubljana, Slovenia
[2] K.U.Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium
[3] Universidad Politecnica de Madrid, E-28660 Boadilla del Monte, Madrid, Spain
[4] Department of Computer Science, University of York, York, YO1 5DD, UK

Abstract. Expert systems for decision support have recently been successfully introduced in road transport management. These systems include knowledge on traffic problem detection and alleviation. The paper describes experiments in automated acquisition of knowledge on traffic problem detection. The task is to detect road sections where a problem has occured (critical sections) from sensor data. It is necessary to use inductive logic programming (ILP) for this purpose as relational background knowledge on the road network is essential. In this paper, we apply three state-of-the art ILP systems to learn how to detect traffic problems and compare their performance to the performance of a propositional learning system on the same problem.

1 Introduction

Expert systems for decision support have recently been successfully introduced in road transport management. Some of the proposals in this direction are TRYS [5], KITS [4] and ARTIST [9]. From a general perspective, the goal of a real time traffic expert system for decision support is to advise traffic management center operators by proposing control actions to eliminate or reduce problems according to the global state of traffic. To assess the global state of traffic, the system periodically receives readings from sensors on the road, which measure magnitudes such as speed (Km/h), flow (veh/h) and occupancy (percentage of time that the sensor is occupied by vehicles), as well as information about the current state of control devices, such as traffic signals at intersections, traffic signals at sideway on-ramps, CMS (Changeable Message Signs), etc. The system interprets sensor data, detects the presence of a problem, gives the possible cause and proposes recommendations about how to solve or reduce it.

The usual approach to building traffic expert systems is to use knowledge based architectures that support the strategies of reasoning followed by operators. This approach requires to develop knowledge bases using symbolic representations (such as rules, frames, or constraints) that include specific domain knowledge of transport management corresponding to the city for which the system is developed. Among other things, knowledge on detecting specific traffic problems is necessary.

On the other hand, traffic management centers have databases that include basic information about different traffic scenarios, such as congestions at certain locations caused by lack of capacity due to accidents or excess of demand (rush hours). This data, collected from sensors on the road, can be used to either generate or improve the knowledge base for problem (incident) detection of the expert system. The paper explores the possibility to use inductive learning techniques (in particular ILP-inductive logic programming) to generate knowledge on traffic problem detection from historical data that contains parameters recorded by sensors.

The learning experiments described in this paper take place within the context of the traffic management expert system TRYS [5], developed for the cities of Madrid and Barcelona. The system uses knowledge distributed in a collection of knowledge bases that use different representations and address specific tasks (such as data abstraction, incident detection, problem diagnosis, prediction of behaviour, and recommendation of control actions). The knowledge for incident (traffic problem) detection has been formulated by domain experts in a first-order frame-based representation, i.e., in the concept description language CONCEL. Therefore, ILP is a suitable tool for learning to detect traffic problems in this context.

Overall, two kinds of input are available to the learning process. The first type is background knowledge on the road network, which is present in and used by the TRYS system. An object oriented representation is used to capture the different types of road sections, the relations among them, and the placement of sensors on individual road sections. The second type is sensor readings on three basic quantities describing traffic behaviour: speed, flow and occupancy. Both types of input will be described in more detail in Section 2. The goal of the learning process is to identify critical sections (where problems have occurred) by using sensor readings and road geometry. Technically speaking, a critical section is a section of the road which constrains the road capacity the most, e.g., because an accident has occured just after this section in the immediate past. In the paper, the term accident critical section refers to such a section and not to a section where accidents occur frequently.

Let us note at this point that in practice real sensor data are available. However, we have used simulated data in our experiments for three reasons. The first is that real sensor data were not immediately available because of management reasons. The second is missing sensor data from broken sensors (which amounts to approximately 20% of the sensors). Finally, using a simulator makes it possible to easily generate a wide range of different traffic problems (including accidents that should not be artificially produced in the real world).

We used AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks) [1], a software tool able to reproduce the real traffic conditions of any urban network on a computer. AIMSUN follows a microscopic simulation approach. It means that the behaviour of each individual vehicle in the network is continuously modelled throughout the simulation time period it remains inside the system (i.e., the traffic network), according to several vehicle behaviour models. A model of the urban-ring of the city of Barcelona was developed using this simulator. This model includes exactly the same variables that the real information system records using sensors and was calibrated using information from the real system. Using this model, a collection of examples (including accidents and congestions due to rush hours) were produced for the learning experiments presented in the paper.

The remainder of the paper is organized as follows. Section 2 describes in some detail the background knowledge on the road network and the sensor data used in TRYS. Section 3 describes the particular dataset used in our experiments. Section 4 presents the results of applying three ILP systems to this dataset, and Section 5 concludes with a discussion.

2 Road network and sensor data

In TRYS [5], the road network is represented in an object oriented fashion. The basic object in the road network representation is the section. A section refers to a cross-section of the road and typically has an array of sensors associated to it. There exist several types of sections, such as off-ramp, on-ramp or highway. Relations between sections, such as previous and next, are included in the TRYS knowledge base. The complexity of road structures makes it possible for a section to have more than two previous or next sections.

A link describes a logical group of sections. For instance, the section just before and just after an off-ramp, together with the off-ramp itself, form an off-ramp-link. There are about ten different types of links. TRYS also uses other concepts like nodes, problem areas and measurement points, but these were not used in our experiments.

The information about sections and links is static. Each section is of a certain type and is associated to a number of sensors (as many sensors as there are lanes at that cross-section of the road) and each link is of a certain type and links a predefined set of sections. These relationships can therefore be considered background knowledge for the learning process.

Sensors provide us with a continuous stream of information, sending five readings each minute that refer to the last minute and each of the four minutes preceding it. Typically, flow (number of cars that passed the sensor in the last minute) and occupancy (the proportion of time the sensor is occupied, in thousandths) are measured. Some sensors (which are actually double sensors) also measure the average speed of the cars that passed the sensor during the last minute. The measurements of sensors related to a single section are aggregated: flow is summed across lanes, while occupancy and velocity are averaged across lanes. Saturation is a derived quantity defined as the ratio between the flow and the capacity of a section: the latter depends on the number of lanes and is part of the background knowledge.

The TRYS system stores its information in two formats: in CONCEL format, which is a frame-based format, and in Prolog format. The Prolog format is object-oriented and consists mainly of facts about the predicates `instance` and `value`. For use in ILP, we transformed these facts in the following fashion:

facts of the form `instance(Instance,Class)` are translated to facts of the form `Class(Instance)` and facts of the form `value(Instance,Attribute,Value)` are translated to `Attribute(Instance, Value)`. For example, the fact `valor('RLTL_Ronda_Litoral_en_Rambla_Prim',tipo,carretera)` is transformed to `tipo('RLTL_Ronda_Litoral_en_Rambla_Prim',carretera)`.

3 The dataset and the methodology

After conducting preliminary experiments with CLAUDIEN [7] on a handful of simulated accidents and congestions, we proceeded to create a larger dataset [10]. A dataset containing 66 examples of congestion and 62 examples of accidents on different locations (off-ramp, on-ramp and highway sections) was generated using the AIMSUN simulator. The learning task was formulated as a classification task, with three possible classes `accident`, `congestion` and `noncs` (noncritical section). Each section at a particular moment of time was treated as an example, classified into one of the above classes. In this way we obtained a dataset consisting of 5952 examples altogether.

The background knowledge consisted of facts on sensor values as well as facts on road geometry. More precisely, the background knowledge predicates `velocidadd(Time,Section,Value)`, `ocupaciond(Time,Section,Value)`, and `saturaciond(Time,Section,Value)` were available. These predicates discretize the continuous sensor measurement values, so `Value` takes one of three discrete values: `alta` (high), `media` (medium), and `baja` (low).

As far as road geometry was concerned, the predicates `tipo(Section,Type)` and `secciones_posteriores(Section1,Section2)` were available. The first gives the type of a section, which can be `carretera` (highway), `calle` (street), `rampa_ abandono` (off-ramp), `rampa_incorporacion` (on-ramp), `via_secundaria_abandono` and `via_secundaria_incorporacion`. The second predicate captures the relational structure of the road network: it states that `Section2` follows `Section1`. Note that this is a many to many relation.

The class distribution is very skewed, as the frequency of the majority class is 97.85%. Despite showing the potential of ILP to learn to detect traffic problems, preliminary experiments with TILDE [2] on this dataset [10] did not yield satisfactory performance. The skewed distribution was identified as one of the main reasons for this.

To alleviate this problem, we randomly sampled 128 of the 5824 noncritical sections. In all experiments presented in the remainder of the paper, we used the dataset comprising 66 examples of congestion, 62 examples of accidents, and 128 examples of noncritical sections.

From the dataset of 256 examples, we created 10 folds. Each of the ILP systems considered was applied on the entire dataset and the rules produced were shown to the domain experts. Also, the time taken to induce a hypothesis from the 256 examples dataset was recorded. Each of the systems was then applied to the 10 folds, where the induced hypotheses were tested on unseen cases and the overall accuracy recorded.

```
class(accident) :- section(A), timemoment(B), tipo(A,carretera),
  secciones_posteriores(C,A), velocidadd(B,C,baja). % [29,0,0]
class(accident) :- section(A), timemoment(B),
  ocupaciond(B,A,alta), saturaciond(B,A,baja),
  secciones_posteriores(A,C), velocidadd(B,C,alta).  % [36,0,0]
class(accident) :- section(A), timemoment(B), velocidadd(B,A,baja),
  secciones_posteriores(C,A), ocupaciond(B,C,alta).  % [27,0,0]
class(accident) :- section(A), timemoment(B), ocupaciond(B,A,alta),
  saturaciond(B,A,baja), secciones_posteriores(A,C), tipo(C,carretera),
  secciones_posteriores(C,D), ocupaciond(B,D,baja). % [22,0,0]

class(congestion) :- section(A), timemoment(B),
  tipo(A,rampa_incorporacion),
  secciones_posteriores(A,C), saturaciond(B,C,alta),
  secciones_posteriores(D,C), velocidadd(B,D,baja). % [0,31,0]
class(congestion) :- section(A), timemoment(B), tipo(A,rampa_abandono),
  secciones_posteriores(C,A), velocidadd(B,C,baja). % [0,30,0]

class(noncs) :- section(A), timemoment(B), velocidadd(B,A,alta),
  secciones_posteriores(C,A), velocidadd(B,C,alta). % [0,0,46]
class(noncs) :- section(A), timemoment(B), ocupaciond(B,A,baja),
  secciones_posteriores(C,A), velocidadd(B,C,alta),
  secciones_posteriores(D,C),velocidadd(B,D,alta). % [0,0,17]
class(noncs) :- section(A), timemoment(B),  tipo(A,carretera),
  secciones_posteriores(A,C),ocupaciond(B,C,alta),
  secciones_posteriores(C,D),ocupaciond(B,D,alta).  % [0,0,26]
...
```

Fig. 1. An incomplete listing of the rules generated by ICL.

4 Experiments and results

Three ILP systems were applied to the problem formulated above. ICL [8] and TILDE [2] operate within the 'learning from interpretations' [6] setting, while PROGOL [11] operates in the 'learning from entailment' setting. This is the reason for the slightly different form of rules induced by the three systems, i.e., the appearance of the literals `section(A)` and `timemoment(B)` in the bodies of ICL and TILDE rules, which do not appear in the bodies of PROGOL rules.

While ICL and PROGOL learn clauses/rules, TILDE learns logical decision trees. The decision trees can be rewritten as decision lists, where the first rule in the list that applies to the example classified is taken. Decision lists can be viewed as lists of clauses, each followed by a Prolog cut (`!`).

The following settings were used: ICL used a minimal coverage of three examples, instead of the default one, and the default significance threshold of 90%. TILDE also used the limit of at least three examples in each leaf, as well as a lookahead of two.

```
accident(A,B) :- velocidadd(B,A,baja), saturaciond(B,A,baja). % [27,0,0]
accident(A,B) :- secciones_posteriores(C,A), tipo(A,carretera),
  velocidadd(B,C,baja). % [29,0,0]
accident(A,B) :- ocupaciond(B,A,alta), saturaciond(B,A,baja),
  secciones_posteriores(A,C), velocidadd(B,C,alta). % [36,0,0]

congestion(A,B) :- secciones_posteriores(C,A), tipo(A,rampa_abandono),
velocidadd(B,C,baja). % [0,30,0]
congestion(A,B) :- secciones_posteriores(A,C), saturaciond(B,C,alta),
  secciones_posteriores(D,C), velocidadd(B,D,baja). % [0,31,0]

noncs(A,B) :- velocidadd(B,A,alta), saturaciond(B,A,media). % [0,0,32]
noncs(A,B) :- saturaciond(B,A,alta), tipo(A,carretera). % [0,0,30]
noncs(A,B) :- secciones_posteriores(C,A), ocupaciond(B,C,baja). %[0,0,25]
noncs(A,B) :- ocupaciond(B,A,baja), saturaciond(B,A,baja),
  tipo(A,rampa_incorporacion). % [0,0,8]
noncs(A,B) :- ocupaciond(B,A,baja), secciones_posteriores(A,C),
  saturaciond(B,C,media). % [0,0,5]
noncs(A,B) :- ocupaciond(B,A,baja), secciones_posteriores(C,A),
  saturaciond(B,C,alta). % [0,0,11]
noncs(A,B) :- ocupaciond(B,A,baja), saturaciond(B,A,baja),
  secciones_posteriores(C,A), ocupaciond(B,C,media). % [0,0,11]
noncs(A,B) :- secciones_posteriores(C,A), saturaciond(B,C,alta),
  secciones_posteriores(C,D), velocidadd(B,D,media). % [0,0,12]
```

Fig. 2. The set of rules generated by PROGOL.

PROGOL used its default settings. Since it works with positive and negative examples only, a set of rules was induced for each of the three predicates `accident(Section,Time)`, `congestion(Section,Time)`, and `noncs(Section, Time)`. Positive examples of one predicates were treated as negative examples of the other two predicates, e.g., positive examples for `congestion(Section, Time)` and `noncs(Section,Time)` were treated as negative examples for `accident(Section,Time)`. To the rules induced, a classification procedure similar to the one of CN2 [3] and ICL was applied: all rules whose conditions apply to a testing example are taken and the number of training examples of each class covered by the rules are summed up. The class with the largest sum is assigned to the testing example.

In addition to the three ILP systems, the decision tree learning system C4.5 [12] was used for comparison. C4.5 only took into account sensor values for the focus section and could not access sensor values of neighboring sections. Since a section can have several previous and/or next sections, the problem cannot be transformed to propositional form easily without loss of information. The default parameters were used in C4.5.

The rules induced by each of ICL, PROGOL and TILDE are given in Figures 1, 2, and 3, respectively. The numbers in square brackets are the numbers of examples covered by the respective clauses / leaves. The second ICL rule and the first PROGOL rules for congestions thus cover 30 examples of the correct class and no examples of the other classes (`[0,30,0]`). For TILDE, the first leaf covers 36 examples, all of which represent accidents (`[36 / 36]`), and the last one covers 14 examples, 12 of which are noncritical sections (`[12 / 14]`).

In all three cases, two rules are used to characterize congestions. PROGOL uses three rules to characterize accidents and leaves eight accidents uncovered, while ICL and TILDE use four rules, resp. leaves, each.

Table 1 summarizes the accuracies of the four systems on unseen cases, as estimated by 10 fold crossvalidation. The numbers in brackets are the standard errors. For example, ICL achieves an average accuracy of 93.36% over the 10 folds with a standard error of 4.42%.

The running time (on a SUN SPARC Ultra 2) and the number of rules (res. leaves) for each system on the 256 examples dataset are also listed in the table. In terms of induction speed, ICL and TILDE are much faster than PROGOL in this particular domain. In terms of classification accuracy, all three algorithms perform at a similar level.

C4.5 is much faster than the three ILP systems. However, its accuracy is lower than the accuracies achieved by the ILP systems. The relational information on the road structure and the sensor values at neighboring sections allow ILP systems to achieve better classification of the focus section.

An important aspect of the suitability of the induced rules for use in an expert system is their comprehensibility to domain experts, in this case traffic engineers. The rules listed in Figures 1, 2, and 3 were inspected by the domain experts. Many of them were judged to be correct, but a substantial number were qualified as overly general, i.e., as potentially covering examples from other classes.

The TILDE decision tree and the corresponding rules fare the lowest in terms of comprehensibility: the dependence of rules in the decision list on rules above them makes them difficult to understand. On the positive side, only the first two accident rules/leaves produced by TILDE were judged to be correct, whereas all other accident rules were judged to be overly general. The two remaining accident leaves are expected to also cover noncritical sections.

Table 1. Performance of three ILP systems and one propositional system at detecting traffic problems.

System	Time	Accuracy	Rules
ICL	82s	93.36 (4.42) %	18
PROGOL	27min	93.75 (2.64) %	13
TILDE	28s	94.14 (3.63) %	12
C4.5	20ms	87.90 (5.34) %	14

Let us take a look at the first two accident rules/leaves by TILDE. The first states that there is an accident at section `A` if the saturation at `A` is low and the occupation at the section preceding `A` is high. The second states that there is an accident at section `A` if the saturation is low and the occupation is high at `A`, whereas the velocity at the next section is high.

```
section(A), timemoment(B)
secciones_posteriores(A,C), saturaciond(B,A,baja)?
+--yes: secciones_posteriores(D,A), ocupaciond(B,D,alta)?
|       +--yes: accident [36 / 36]
|       +--no:  ocupaciond(B,A,alta)?
|               +--yes: velocidadd(B,C,alta)?
|               |       +--yes: accident [15 / 15]
|               |       +--no:  secciones_posteriores(C,E),
|                               ocupaciond(B,E,baja)?
|               |               +--yes: accident [3 / 3]
|               |               +--no:  noncs [4 / 4]
|               +--no:  noncs [13 / 13]
+--no:  tipo(A,carretera)?
        +--yes: secciones_posteriores(F,A), velocidadd(B,F,baja)?
        |       +--yes: accident [5 / 5]
        |       +--no:  noncs [73 / 75]
        +--no:  secciones_posteriores(G,A), velocidadd(B,G,alta)?
                +--yes: noncs [21 / 23]
                +--no:  secciones_posteriores(H,A), velocidadd(B,H,baja)?
                        +--yes: congestion [30 / 30]
                        +--no:  secciones_posteriores(A,I),
                                ocupaciond(B,A,alta)?
                                +--yes: secciones_posteriores(J,I),
                                        velocidadd(B,J,baja)?
                                |       +--yes: congestion [31 / 31]
                                |       +--no:  noncs [5 / 7]
                                +--no:  noncs [12 / 14]
```

Fig. 3. The logical decision tree generated by TILDE.

The first, second and fourth accident rules by ICL are expected to cover congestions as well, whereas the third accident rules and the second congestion rule are expected to also cover noncritical sections. The remaining ICL rules are judged to be correct. This includes all rules on noncritical sections and the first rule on congestions.

Let us take a look at the correct rule about congestions. It concerns a section `A` of type on-ramp (`rampa_incorporacion`). There is a congestion at such a section if the saturation at a following section `C` is high and the velocity at a section preceding `C` is low.

Finally, all PROGOL rules on noncritical sections are judged correct, as well as one rule on congestions which is identical to the ICL correct rule on congestions. As stated above, all PROGOL accident rules are considered overly general.

5 Discussion

We have presented an application of inductive logic programming in a novel domain, namely the domain of detecting traffic problems. The task addressed was to learn rules that identify road sections where accidents or congestions have occurred in the immediate past. As shown by a comparison to a propositional learning system, background knowledge on road geometry was essential, requiring the use of ILP for this task. While simulated data were used for our experiments, it should be noted that the simulator is very realistic and has been calibrated using real-world data.

We applied three different ILP systems to this problem and compared their performance in terms of efficiency of induction, accuracy on unseen cases and comprehensibility of induced rules. All three systems perform at roughly the same level as far as accuracy is concerned. TILDE performs best in terms of efficiency, but its rules are overall more difficult to understand. All three ILP systems are much slower than C4.5, but perform better in terms of classification accuracy.

The rules induced by PROGOL and ICL are very similar and receive similar comprehensibility marks from the domain experts. They also perform at the same level as far as accuracy is concerned. ICL is more efficient, however, and marginally the best choice of the three ILP systems considered.

Many of the rules were judged as overly general by the domain experts. A possible reason for this are errors in identifying the critical section in an area where an accident has occurred. The critical section is the one that constrains capacity the most, but in some cases a previous or a following section may have been stated as critical.

Regarding related work, several ILP systems have been compared to C4.5 on the problem of identifying accidents caused by young male drivers [13]. However, the data considered there is propositional. The ILP systems perform at approximately the same level as C4.5, while being much slower.

Much work remains to be done. Immediate further work concerns the use of nondiscretized sensor values. Since all three ILP systems considered can handle real values this should not be a problem. Exepriments are already underway.

A practical issue of utmost importance is the issue of using real sensor data instead of simulated data. Missing sensor values are a problem that has to be dealt with here and redundant rules will have to be built for this purpose.

Other issues to be addressed include mapping the induced problem detection rules into a frame-based representation with which experts are familiar and using the time series of sensor values instead of the current values only. The domain of traffic control also holds other challenges for machine learning techniques. Detecting traffic problems is only one step of the traffic management process: suggesting actions to alleviate the problems is the natural next step. Since examples of operator actions in response to detected problems exist, there is hope that the problem of suggesting appropriate actions for alleviating traffic problems can also be addressed using machine learning and inductive logic programming.

Acknowledgements This work was supported in part by the ESPRIT IV Project 20237 ILP2. Nico Jacobs is supported by the Flemish Institute for the Promotion of Scientific and Technological Research in Industry (IWT). Wim Van Laer is supported by the Fund for Scientific Research, Flanders.

References

1. Barcelo, J., Ferrer J.L., and Montero, L. (1989). *AIMSUN: Advanced Interactive Microscopic Simulator for Urban Networks. Vol I: System Description, and Vol II: User's Manual.* Departamento de Estadistica e Investigacion Operativa, Facultad de Informatica, Universidad Politecnica de Cataluna, Barcelona, Spain.
2. Blockeel, H., and De Raedt, L. (1997). Lookahead and discretization in ILP. In *Proc. 7th Intl. Workshop on Inductive Logic Programming*, pages 77–84, Springer, Berlin.
3. Clark, P. and Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In *Proc. Fifth European Working Session on Learning*, pages 151–163. Springer, Berlin.
4. Cuena, J., Ambrosino, G., and Boero M. (1992). A general knowledge-based architecture for traffic control: The KITS approach. In *Proc. Intl. Conf. on Artificial Intelligence Applications in Transportation Engineering.* San Buenaventura, CA.
5. Cuena, J., Hernandez, J., and Molina, M. (1995). Knowledge-based models for adaptive traffic management systems. *Transportation Research: Part C*, 3(5): 311-337.
6. De Raedt, L. (1997). Logical settings for learning. *Artificial Intelligence.*
7. De Raedt, L., and Dehaspe, L. (1997). Clausal discovery. *Machine Learning*, 26: 99–146.
8. De Raedt, L., and Van Laer, V. (1995). Inductive constraint logic. *Proc. Sixth International Workshop on Algorithmic Learning Theory*, pp. 80-94. Berlin: Springer.
9. Deeter, D.L., and Ritchie, S.G. (1993). A prototype real-time expert system for surface street traffic management and control. In *Proc. 3rd Intl. Conf. on Applications of Advanced Technologies in Transportation Engineering*, Seattle, WA.
10. Džeroski, S., Jacobs, N., Molina, M., Moure, C. (1998). ILP experiments in detecting traffic problems. In *Proc. Eleventh European Conference on Machine Learning.* Springer, Berlin. To appear.
11. Muggleton, S. (1992). Inverse entailment and PROGOL. *New Generation Computing* 13: 245–286.
12. Quinlan, J.R. (1993) *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA.
13. Roberts, S., Van Laer, W., Jacobs, N., Muggleton, S., Broughton, J. (1998) A comparison of ILP and popositional systems on propositional traffic data. In *Proc. Eighth International Conference on Inductive Logic Programming.* Springer, Berlin. This volume.