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, 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.

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 [4], KITS [3] and ARTIST [6]. 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 asses 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 (such as ILP-inductive logic programming) to generate knowledge on traffic problem detection from 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 [4], 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. 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 occured) 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 organised as follows. Section 2 describes in some detail the background knowledge on the road network and the sensor data used in the experiments. Section 3 describes the dataset used in our experiments. Section 4 presents the results of applying three ILP systems to this datasets, and Scetion 5 consludes with a discussion.

2 Road network and sensor data

In TRYS [4], 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 pro mille of time the sensor is occupied) 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 simplicity reasons, we transform 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 `instance(salida_a_rambla_Prim,off_ramp)` is transformed to `off_ramp(salida_a_rambla_Prim)`.

3 The dataset and the methodology

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 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.

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, e.g., `carretera` (highway). The second captures the relational structure of the road network: it states that `Section2` follows `Section1`. Note that this is a many to many relation.

Since the class distribution is very skewed, we randomly sampled 128 of the 5952 noncritical sections. In all experiments presented in the remainder of the paper, we used the dataset consisting 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 sysetms applied 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.

Table 1. Performance of three ILP systems at detecting traffic problems.

System	Time	Accuracy	Rules
ICL	82s	93.36 %	18
PROGOL	27min	93.75 %	13
TILDE	28s	94.14 %	12

```
Time = Time on entire dataset
Accuracy = Accuracy on unseen cases as measured by 10-fold CV
Rules = Number of rules induced from entire dataset
(for TILDE number of leaves of the induced tree)
```

4 Results

Three ILP systems were applied to the ILP problem formulated above. ICL and TILDE operate within the 'learning from interpretations' setting, while PROGOL operates in the 'learning from entailment' setting. This causes the slightly different form of rules induced by the three systems.

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 a significance threshold of 99%, instead of the default 90%. TILDE also used the limit of at least three examples in each leaf, as well as a lookahead of two. PROGOL used its default settings.

The rules induced by each of ICL, PROGOL and TILDE are given in Figures 2, 3, and 4, respectively. 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 three systems on unseen cases, as estimated by 10 fold crossvalidation. The running time and number of rules (res. leaves) on the 256 examples dataset are also listed in the table.

Acknowledgements

Nico Jacobs is financed by a specialisation grant of the Flemish Institute for supporting scientific-technological research in the industry (IWT). This work was supported by the ESPRIT IV Project 20237 ILP2.

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] 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.

[4] Cuena, J., Hernandez, J., and Molina, M. (1995). Knowledge-based models for adaptive traffic management systems. *Transportation Research: Part C*, 3(5): 311-337.

[5] De Raedt, L., and Dehaspe, L. (1997). Clausal discovery. *Machine Learning*, 26: 99–146.

[6] 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.

```
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]
class(noncs) :- section(A), timemoment(B), tipo(A,rampa_incorporacion),
  ocupaciond(B,A,baja), saturaciond(B,A,baja). % [0,0,8]
class(noncs) :- section(A), timemoment(B), saturaciond(B,A,media),
  secciones_posteriores(C,A), velocidadd(B,C,alta), ocupaciond(B,C,alta),
  secciones_posteriores(C,D), ocupaciond(B,D,baja). % [0,0,21]
class(noncs) :- section(A), timemoment(B), saturaciond(B,A,media),
  secciones_posteriores(A,C), saturaciond(B,C,media),
  secciones_posteriores(D,C),velocidadd(B,D ,alta). % [0,0,23]
class(noncs) :- section(A), timemoment(B), tipo(A,rampa_abandono),
  secciones_posteriores(C,A), velocidadd(B,C,media),
  secciones_posteriores(D,C), saturaciond(B,D,media). % [0,0,6]
class(noncs) :- section(A), timemoment(B), tipo(A,carretera),
  saturaciond(B,A,alta). % [0,0,30]
class(noncs) :- section(A), timemoment(B), saturaciond(B,A,media),
  secciones_posteriores(A,C), velocidadd(B,C,media),
  secciones_posteriores(D,C), velocidadd(B,D,media). % [0,0,6]
class(noncs) :- section(A), timemoment(B), ocupaciond(B,A,alta),
  secciones_posteriores(C,A), ocupaciond(B,C,baja). % [0,0,15]
class(noncs) :-  section(A), timemoment(B), tipo(A,rampa_incorporacion),
  secciones_posteriores(A,C), velocidadd(B,C,baja), saturaciond(B,C,media).
  % [0,0,3]
class(noncs) :- section(A), timemoment(B), saturaciond(B,A,baja),
  secciones_posteriores(C,A), ocupaciond(B,C,media),
  secciones_posteriores(D,C),saturaciond(B,D,baja).  % [1,0,8]
```

Figure 2. The set of rules generated by ICL from the entire traffic problem dataset.

```
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]
```

Figure 3. The set of rules generated by PROGOL from the entire traffic problem dataset.

```
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]
```

```
class(accident) :- section(A), timemoment(B),
  secciones_posteriores(A,C), saturaciond(B,A,baja),
  secciones_posteriores(D,A), ocupaciond(B,D,alta), !.
class(accident) :- section(A), timemoment(B),
  secciones_posteriores(A,C), saturaciond(B,A,baja),
  ocupaciond(B,A,alta), velocidadd(B,C,alta), !.
class(accident) :- section(A), timemoment(B),
  secciones_posteriores(A,C), saturaciond(B,A,baja),
  ocupaciond(B,A,alta), secciones_posteriores(C,D), ocupaciond(B,D,baja), !.
class(noncs) :- section(A), timemoment(B),
  secciones_posteriores(A,C), saturaciond(B,A,baja), ocupaciond(B,A,alta), !.
class(noncs) :- section(A), timemoment(B),
  secciones_posteriores(A,C), saturaciond(B,A,baja), !.
class(accident) :- section(A), timemoment(B),
  tipo(A,carretera), secciones_posteriores(C,A), velocidadd(B,C,baja), !.
class(noncs) :- section(A), timemoment(B), tipo(A,carretera), !.
class(noncs) :- section(A), timemoment(B),
  secciones_posteriores(C,A), velocidadd(B,C,alta), !.
class(congestion) :- section(A), timemoment(B),
  secciones_posteriores(C,A), velocidadd(B,C,baja), !.
class(congestion) :- section(A), timemoment(B),
  secciones_posteriores(A,C), ocupaciond(B,A,alta),
  secciones_posteriores(D,C), velocidadd(B,D,baja), !.
class(noncs) :- section(A), timemoment(B),
  secciones_posteriores(A,C), ocupaciond(B,A,alta), !.
class(noncs) :- true, !.
```

Figure 4. The tree generated by TILDE from the entire traffic problem dataset and the equivalent Prolog program.