Using machine learning techniques to interpret results from discrete event simulation

Dunja Mladenić[1], Ivan Bratko[1,2], Ray J. Paul[3], Marko Grobelnik[1]

[1]J. Stefan Institute, Dept. of Computer Sc., Jamova 39, Ljubljana, Slovenia
[2]Faculty of Electr. Eng. and Computer Sc., Tržaška 25, Ljubljana University
[3]Brunel University, Dept. of Computer Sc., Uxbridge, England
e-mail: dunja.mladenic@ijs.si

Abstract. This paper describes an approach to the interpretation of discrete event simulation results using machine learning techniques. The results of two simulators were processed as machine learning problems. Interpretation obtained by the regression tree learning system RETIS was intuitive but obviously expressed in a complicated way. To enable a more powerful knowledge representation, Inductive Logic Programming (ILP) system MARKUS was used, that also highlighted some attribute combinations. These attribute combinations were used as new attributes in further experiments with RETIS and ASSISTANT. Some interesting regularities were thereby automatically discovered.

1 Introduction

Discrete simulation involves several stages: design and implementation of a model for the system under study, performing experiments with the simulator, analysis and presentation of experimental results (Paul and Balmer 1991). Analysis and presentation of experimental results is usually done intuitively. Deriving useful conclusions from the simulation results is a demanding task. In this paper we propose to automate this task by means of machine learning techniques. It should be emphasised that in this application of Machine Learning tools the prediction accuracy is not the main goal. After all, if the user is interested in prediction, she or he can always use the simulator for that. The real goal here is to find interpretation of simulation data and discover regularities, thus helping the user to develop intuitive understanding of the domain.

Discrete event simulation produces example situations that can be used as examples for machine learning. Section 2 describes discrete event simulation domains used in our experiments with machine learning. Section 3 describes three machine learning algorithms used to interpret the results of discrete event simulation. Results are discussed in Section 4.

2 Domains description

In our experiments we used two discrete event simulators, the Supermarket and the Pub. They were suggested and generated by the domain expert as simple and commonly used example domains in the simulation theory.

Each discrete event simulator has input parameters (attributes) that can be changed by the user (our domain expert) and output some statistical indicators (classes) calculated by the simulator. For each simulator and for each output value, a machine learning domain was constructed and experiments with it performed.

In the Supermarket domain, customers arrive and spend some time shopping. After shopping, they have to wait to be served by a cashier. A customer who on arrival sees a queue of more than `QWSize` people waiting to be served, leaves without entering the supermarket. In the described process, there are some attributes whose influence we want to investigate: number of cashiers in the supermarket, `Cashiers` (varied from 1 to 5); maximum queue size tolerated by a customer, `QWSize` (varied from 1 to 10); time between two arrivals, `Arrival` (exponentially distributed with a mean of 1 to 10 minutes). We considered as interesting the following statistical outputs, used as classes in learning: customer "utilisation" (ratio between the number of customers that enter the shop and the number of customers that arrive); cashier utilisation (fraction of cashier's time spent actually serving customers); length of the customers wait queue. Each combination of attribute values corresponds to a run of simulator producing a machine learning example. In this way, domains for all possible attribute values were generated, having $5 \times 10 \times 10 = 500$ examples.

In the pub simulator, customers arrive and wait for the barmaid to pour them a glass of drink. After drinking, they leave or return to the queue for another drink. The barmaids just pour beer and wash glasses. A glass can be either clean (and empty), full (in use) or dirty. Problems, experiments and domain expert's comments for this domain are similar to the Supermarket domain.

3 Machine learning algorithms used

This section gives brief description of machine learning algorithms used to interpret the results of discrete event simulation.

ASSISTANT (Cestnik, Kononenko and Bratko 1987) is an attribute-value learning algorithm for the construction of binary classification trees. RETIS (Karalič 1992) is an attribute-value learning algorithm for the construction of regression trees. Regression trees are similar to binary classification trees constructed by the ASSISTANT algorithm. MARKUS (Grobelnik 1992) is an Inductive Logic Programming (ILP) system designed for inferring Prolog programs from examples and counter examples of there behaviour. The MARKUS algorithm has its basis in Shapiro's MODEL INFERENCE SYSTEM (Shapiro 1983), but extends it in several ways. A Prolog program induced by MARKUS uses background-knowledge predicates defined by the user.

4 Experiments and results

Experiments were performed on six learning domains (three for the shop and three for the pub simulator).

As an example of regression trees generated by RETIS, let us consider the tree for customer utilisation (which we can not include here for space reasons). In the part of regression tree with `Cashiers` = 1 and `Arrival` < 4, customer utilisation monotonically increases between 0.31 and 0.63 and is independent of customer tolerance (`QWSize`). This can be explained as follows: for a small number of cashiers and a high arrival rate of customers (attribute `Arrival` small), the waiting queue in the steady state of the system is always long and occasionally drops below `QWSize` thus not influencing customer utilisation. `QWSize` begins to affect the results for the combinations `Arrival` $\geq$ 6 or `Arrival` $\geq$ 2 and `Cashiers` $\geq$ 3. This means, at lower arrival rates and/or higher number of cashiers, `QWSize` becomes important.

We analysed the same simulation results with MARKUS with respect to the question: what combinations of the supermarket attributes will result in customer utilisation below or equal to 90%. We defined the following predicates as background knowledge for MARKUS:

```
mult( A, B, C ) :- C is A * B.
add( A, B, C ) :- C is A + B.
gtr1( X ) :- X > 1. gtr2( X ) :- X > 2. ... gtr9( X ) :- X > 9.
```

The target predicate for learning was: `shop( Cashs, QWSize, Arriv)`, which is true for customer utilisation $\leq$ 90%. The following two Prolog clauses were induced by MARKUS:

```
shop( Cashs, QWSize, Arriv) :-
  mult( Arriv, Cashs, X ), not gtr5( X ).
shop( Cashs, QWSize, Arriv) :-
  mult( Arriv, Cashs, X ), not gtr1( QWSize ), not gtr6( X ).
```

These clauses can be read as: **if** `Cashs*Arriv` $\leq$ 5 or `Cashs*Arriv` $\leq$ 6 and `QWSize` = 1 **then** `CustomerUtil` $\leq$ 90%.

MARKUS nicely discovered the law that holds in situations with high input of customers and low number of cashiers (consistent with the previous regression tree). At lower input rates or higher number of cashiers, the parameter `QWSize` begins to affect the performance which can also be seen in the rule above.

After seeing MARKUS results, we constructed new attributes (in similar way as the ILP system LINUS (Lavrač, Džeroski and Grobelnik 1990) does) with four basic algebraic operations ($*$, /, $+$, $-$) applied on pairs of original attributes. Experiments with RETIS were repeated on such data sets. Additional experiments (problem reformulated as for MARKUS), were performed by ASSISTANT. Both attribute learning systems confirmed the results obtained by MARKUS, moreover in all three domains for the Supermarket simulator the same combined attribute (`Cashiers*Arrival`) is recognised as the "best" one. The role of

parameter `QWSize` is also very similar. It begins to affect the performance when `Cashiers*Arrival` is high (> 3 for RETIS, > 5 for MARKUS and ASSISTANT).

5 Conclusion

In the paper, we experimentally analysed the results of two simple discrete event simulators by means of machine learning programs. Two attribute value and one ILP systems were used on six domains (three for each simulator). The obtained results are encouraging in that the induced descriptions provide some useful insights into the dependencies between the parameters of a discrete event system and its performance. Descriptions obtained by the ILP approach can be particularly concise and interesting.

The induced descriptions have been confirmed to be of interest for the simulation domain experts. Some descriptions were new to the expert. This is important because the success of the present study should not be measured in terms of prediction accuracy, but in terms of interpretation value of the induced descriptions.

6 Acknowledgements

We thank Aram Karalič for providing a copy of the RETIS program used in the experiments in this paper. This work was partially supported by the Slovenian Ministry for Science and Technology and the European ESPRIT III Basic Research Action Project No.6020 Inductive Logic Programming.

References

1. Cestnik, B., Kononenko, I. and Bratko, I., (1987), ASSISTANT 86: A Knowledge-Elicitation Tool for Sophisticated Users, in *Progress in Machine Learning*, ed. by I.Bratko and N.Lavrač, Sigma Press, Wilmslow.
2. Grobelnik, M. (1992) Markus - an optimized Model Inference System, *ECAI-92 Workshop on Logical Approaches to Machine Learning*, Vienna, Austria.
3. Karalič, A. (1992) Employing Linear Regression in Regression Tree Leaves. *Proc. of ECAI 92*, pp. 440-441. Vienna, Austria, John Wiley & Sons.
4. Lavrač, N., Džeroski, S. and Grobelnik, M. (1991) Learning nonrecursive definitions of relations with LINUS. *Proc. of EWSL 91*, pp. 265-281. Porto, Portugal, Springer-Verlag.
5. Mladenić, D. (1990) Machine learning system Magnus Assistant (in Slovene). *BSc Thesis*, University of Ljubljana, Faculty of Electrical Engineering and Computer Science.
6. Paul, R.J., Balmer, D.W. (1991) *Simulation Modelling*, Chartwell-Bratt, Stockholm.
7. Shapiro, E.Y. (1983) *Algorithmic Program Debugging*. Cambridge, MA: MIT Press.