Learning to Manage a Firm

Tristan Cazenave
LAFORIA
Case 169
Université Pierre et Marie Curie
4, place Jussieu, 75252 Paris Cedex 05, France
e-mail: Tristan.Cazenave@laforia.ibp.fr

Abstract

I have developped a general learning system. It uses a representation of knowledge based on an extension of First Order Logic. To initiate the learning process, the system needs a theory of the domain to learn and a definition of the interesting goals to achieve in this domain. The learning process is composed of four parts: problem solving, explanation, generalization and compilation. The point of the paper is to describe the application of this system to the learning of the management of a firm. A firm is represented using four hierarchical levels. The lowest level is the Physical level. It is concerned with the equilibrium between the physical inputs and outputs in the firm. The system learns to choose actions in order to reach this equilibrium. The second level is the Valorized level. At this level, the systems learns to set the prices of the products according to external and internal variables. The third level is the Monetary level. The system learns to have a positive cash flow. The last level is the Financial level. The system learns to have a financial rentability. The approach and the language used in this system are general and can be applied to other domains.

Keywords

Explanation Based Learning, Management of a Firm, Metaknowledge.

1 Introduction

When a domain theory exists, a learning method has been developed: Explanation Based Learning (EBL) [Mitchell 1986] [Dejong 1986]. This learning method is particularly useful in domains that have a strong domain theory. I have developped a learning system which uses a kind of Explanation Based Learning. It is able to learn how to achieve goals given a domain theory, a definition of the goals to achieve and some examples. In order to apply it to the management of a firm, I have used the theory of a firm described in [Alia 1992]. It is a theory which decomposes a firm in four hierarchical levels. Each of these levels is associated to a goal. My system learns to act so as to satisfy the four goals: Equilibrium between inputs and outputs at the Physical level, setting the price of products at the Valorized level, have a positive cash flow at the Monetary level, have a financial rentability at the Financial level.

In a first part, I describe my general learning system using an example of learning at the Monetary level. The Monetary level is the level which is the more complex to learn. In a second part, I show how it has been applied to the learning of the management of a firm. I give some possible extensions to this work in conclusion. The learning algorithm described in this paper can be used in many other domains. My system has also been applied with success to the game of Go [Cazenave 1996a].

2 A General Learning System

In this section, I describe the main components of my general learning system. I begin with knowledge representation and declarativity in the domain theories. Then I explain the four step learning process, composed of : Problem Solving, Explanation, Generalization and Compilation.

2.1 Knowledge Representation

In order to learn correct rules, a learning program must be given a declarative domain theory, otherwise it learns false rules. The declarativity of knowledge presents many aspects. The first one is the explicit representation of knowledge, it allows the program to manipulate the knowledge it uses. This aspect is typical of logic programming techniques, as in Prolog programs or in expert systems. The second constraint that declarativity imposes on knowledge representation is that the instructions of a program must be given without the order to execute them. A declarative program is both explicit and given without a defined order [Pitrat 1990]. This second aspect is very important because it facilitates explanations and learning. I have chosen to represent knowledge using Horn clauses. I use the metapredicate 'exist' which verifies that a fact exists in the working memory, and the metapredicate 'append' which appends a fact in the working memory. My system is also able to manipulate integer and real variables. A variable is usually represented by a word beginning with a question mark.

<table>
<tr>
<td>Rule_Cash_1:

If (exist (Quantity_Sold (?t ?n1))
exist (Sell_Price (?t ?n2))
exist (Delay_of_payment_sell (?t1))
equal (?t2 sum (?t ?t1))
equal (?n3 multiplication (?n1 ?n2))
)
Then (append (Sell_Income (?t2 ?n3)))</td>
<td>Rule_Cash _2:

If (exist (Cash (?t ?n))
exist (Sell_Income (?t ?n1))
exist (Quantity_Work (?t ?n3))
exist (Buy_Outcome (?t ?n4))
equal (?t1 sum (?t 1))
equal (?n5 sub (sum (?n ?n1) ?n3 ?n4)))
Then (append (Cash (?t1 ?n5)))</td>
</tr>
</table>

Table I

Table I gives two examples of first order rules making use of integer variables. These are rules of the Monetary level domain theory which calculate the cash.

Cash (0 5000)	Quantity_Products_Bought (0 10)	Quantity_Work (1 700)
Quantity_Sold (0 10)	Quantity_Products_Bought (1 10)	Quantity_Work (2 700)
Quantity_Sold (1 10)	Quantity_Products_Bought (2 10)	Quantity_Work (3 700)
Quantity_Sold (2 10)	Quantity_Products_Bought (3 10)	Quantity_Work (4 1050)
Quantity_Sold (3 10)	Quantity_Products_Bought (4 15)	Quantity_Work (5 1050)
Quantity_Sold (4 15)	Quantity_Products_Bought (5 15)	Quantity_Sold (0 10)
Quantity_Sold (5 15)	Price_Product (0 70)	Quantity_Sold (1 10)
Sell_Price (0 170)	Price_Product (1 70)	Quantity_Sold (2 10)
Sell_Price (1 170)	Price_Product (2 70)	Quantity_Sold (3 10)
Sell_Price (2 170)	Price_Product (3 70)	Quantity_Sold (4 15)
Sell_Price (3 170)	Price_Product (4 70)	Quantity_Sold (5 15)
Sell_Price (4 160)	Price_Product (5 70)	Delay_of_payment_buy (3)
Sell_Price (5 160)	Quantity_Work (0 700)	Begin_activity (0)

Table II

Table II gives an example of a working memory for the monetary level. Working memories contains only predicates and constants. Wheras rules can also contain metapredicates and variables.

2.2 Problem Solving

Problem solving is the deductive step which consists in firing rules until no new fact can be deduced. The facts deduced during problem solving are the facts created by the action performed. Table III gives the facts deduced using our domain theory and the working memory in Table II. The actions performed in this example are the setting of the 'Delay_of_payment_sell' at each time.

Delay_of_payment_sell (2)	Buy_Outcome (3 700)	Cash (4 4900)
Sell_Income (0 0)	Buy_Outcome (4 700)	Positive_Cash (4)
Sell_Income (1 0)	Buy_Outcome (5 700)	Cash (5 4850)
Sell_Income (2 1700)	Buy_Outcome (6 700)	Positive_Cash (5)
Sell_Income (3 1700)	Buy_Outcome (7 1050)	Cash (6 4800)
Sell_Income (4 1700)	Buy_Outcome (8 1050)	Positive_Cash (6)
Sell_Income (5 1700)	Cash (1 4300)	Cash (7 6500)
Sell_Income (6 2400)	Positive_Cash (1)	Positive_Cash (7)
Sell_Income (7 2400)	Cash (2 3600)	Cash (8 7850)
Buy_Outcome (0 0)	Positive_Cash (2)	Positive_Cash (8)
Buy_Outcome (1 0)	Cash (3 4600)	Cash (9 6800)
Buy_Outcome (2 0)	Positive_Cash (3)	Positive_Cash (9)

Table III

2.3 Explanation

The explanation module finds the facts which are responsible for the deduction of an interesting fact. Its goal is to create a rule explaining why this fact is present using only facts that represent the situation before it was deduced. To do this, it goes back through the rules fired during problem solving, replacing facts representing the situation at time t by facts representing the position before time t. In our example, it selects the fact 'Positive_Cash (4)' and finds the explanation given in Table IV:

Cash (0 5000)	equal (2 sum (1 1))	greater_than (6 4)
Quantity_Sold (0 10)	equal (3 sum (3 1))	Delay_of_payment_buy (3)
Quantity_Sold (1 10)	equal (4 sum (3 1))	Quantity_Products_Bought (0 10)
Sell_Price (0 170)	equal (2 sum (0 2))	equal (1700 multiplication (170 10))
Sell_Price (1 170)	equal (3 sum (1 2))	equal (1700 multiplication (170 10))
Sell_Price (2 170)	equal (4 sum (2 2))	equal (700 multiplication (70 10))
Price_Product (0 70)	equal (5 sum (3 2))	equal (4300 sub (sum (5000 0) 0 700))
Quantity_Work (0 700)	equal (3 sum (0 3))	equal (3600 sub (sum (4300 0) 0 700))
Quantity_Work (1 700)	equal (4 sum (1 3))	equal (4600 sub (sum (3600 1700) 0 700))
Quantity_Work (2 700)	equal (5 sum (2 3))	equal (4900 sub (sum (4600 1700) 700 700))
Quantity_Work (3 700)	greater_than (5 4)	greater_than (4900 0)
equal (1 sum (0 1))	equal (6 sum (3 3))	Begin_activity (0)

Table IV

This explanation accounts for the fact that to have a positive cash at time 4, setting the delay of payment of sells to 2 works well.

2.4 Generalization

The generalization step consists in transforming the rule which specifically applies on the example, and which contains only constants, in a more general rule which will match on many more examples and which contains variables. A constant is replaced by a variable only in some special cases to avoid to be too general in replacing constants by variables. I only generalize the constants that are instanciations of variables, not the 'true' constants that are also constants in the fired rules. The conditions of the new general rule are given in Table V.

Cash (?t ?n) Quantity_Sold (?t ?q) Quantity_Sold (?t1 ?q1) Sell_Price (?t ?sp) Sell_Price (?t1 ?sp1) Price_Product (?t ?p) Quantity_Work (?t ?qw) Quantity_Work (?t1 ?qw1) Quantity_Work (?t2 ?qw2) Quantity_Work (?t3 ?qw3) equal (?t1 sum (?t 1)) equal (?t2 sum (?t1 1))	equal (?t3 sum (?t2 1)) equal (?t4 sum (?t3 1)) equal (?t2 sum (?t ?dps)) equal (?t3 sum (?t1 ?dps)) equal (?t4 sum (?t2 ?dps)) equal (?t5 sum (?t3 ?dps)) equal (?t3 sum (?t ?dpb)) equal (?t4 sum (?t1 ?dpb)) equal (?t5 sum (?t2 ?dpb)) greater_than (?t5 ?t4) equal (?t6 sum (?t3 ?dpb))	greater_than (?t6 ?t4) Delay_of_payment_buy (?dpb) Quantity_Products_Bought (?t ?qb) equal (?si multiplication (?sp ?q)) equal (?si1 multiplication (?sp1 ?q1)) equal (?bo multiplication (?p ?qb)) equal (?n1 sub (sum (?n 0) 0 ?qw)) equal (?n2 sub (sum (?n1 0) 0 ?qw1)) equal (?n3 sub (sum (?n2 ?si) 0 ?qw2)) equal (?n4 sub (sum (?n3 ?si1) ?bo ?qw3)) greater_than (?n4 0) Begin_activity (?t)

Table V

The conclusion of the learned rule is ‘append (Delay_payment_sell (?t4 ?dps))’.

2.5 Compilation

After the generalization process, the rules are true and general but not efficient. In order to fire them efficiently, I compile them by folding and simplifying expressions and by reordering predicates. Expression simplification is done by example by replacing each term ?t1 by the term ‘sum (?t 1)’ in the rule. Folding is done by replacing the expression ‘sum (1 1)’ by the constant 2. Folding saves time because it execute portions of a rule at the compilation time, these portions will not be reevaluated at execution time. Predicate ordering is done by ordering the predicate in the rule so as to make the less instanciations possible. It can make a rule fire orders of magnitude faster [Cazenave 1996b]. Compilation transforms the list of conditions of Table V in the list of conditions of Table VI:

Delay_of_payment_buy (3)
Begin_activity (?t)
Cash (?t ?n)
Quantity_Sold (?t ?q)
Quantity_Sold (sum (?t 1) ?q1)
Sell_Price (?t ?sp)
Sell_Price (sum (?t 1) ?sp1)
Quantity_Products_Bought (?t ?qb)
Price_Product (?t ?p)
Quantity_Work (?t ?qw)
Quantity_Work (sum (?t 1) ?qw1)
Quantity_Work (sum (?t 2) ?qw2)
Quantity_Work (sum (?t 3) ?qw3)
greater_than (sub (sum (?n multiplication (?sp ?q) multiplication (?sp1 ?q1)) ?qw ?qw1 ?qw2 multiplication (?p ?qb) ?qw3) 0)
equal (?t4 sum (?t 4))

Table VI

The conclusion of the rule is now 'append (Delay_of_payment_sell (?t4 2))'.

2.6 Conclusion

Before learning, to find a possible value for the delay of payment of sells, the system had to try a set of possible values, compute the cash at each time of the simulation for each value, and select the value which resulted in a positive cash at each time of the simulation. After learning, the system can deduce a possible delay of payment for each time only firing the learned rules. Learning allows it to find the solution to the problem of setting the delay of payment more rapidly.

3 Application to the learning of the management of a firm

This learning method has been applied to the learning of the management of a firm, using the formal analysis of a firm given in [Alia 1992]. This model has four hierarchical levels represented in Figure 1. Each level is related to a goal. My system learns to achieve this goal for each level.

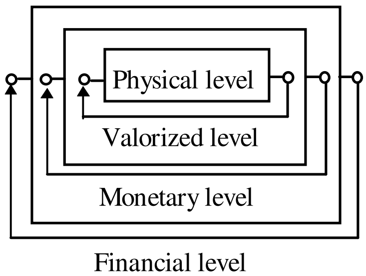

Figure 1

On the physical level, it learns to buy and to produce according to the expected sales. Table VII gives some rules of the domain theory of the Physical level. The system learns to set the value of the variables ?n and ?n1 in the predicates 'Quantity_Products_Bought (?t ?n)' and 'Quantity_Work (?t ?n1)'.

Rule_Manufactured_Products_1 : If (exist (Stock_MP_After_Sale (?t ?n)) exist (Stock_Products (?t ?n1)) exist (Quantity_Products_Bought (?t ?n2)) equal (?n3 sum (?n1 ?n2)) exist (Quantity_Work (?t ?n4)) greater_than (?n3 ?n4) equal (?n5 sum (?n ?n3 ?n4)) equal (?t1 sum (?t 1))) Then (append (Stock_MP_Before_Sale (?t1 ?n5)))	Rule_Sale_1: If (exist (Stock_MP_Before_Sale (?t ?n)) exist (Quantity_Sold (?t ?n1)) greater_than (?n ?n1) equal (?n2 substraction (?n ?n1))) Then (append (Stock_MP_After_Sale (?t ?n2)))

Table VII

On the valorized level, it learns to calculate the price the product should be sold. The system learns to set the value of the variable ?p in the predicate 'Sell_Price (?t ?p)'.

On the monetary level, it learns how to have a positive cash. The behavior of the system has been extensively described in section 2.

On the financial level, it learns how to have a good rentability.

4 Conclusion

I have described a method to learn rules of management of a firm using a knowledge representation based on an extension of First Order Logic to the use of integer and real variables. This method allows a system to find the solution to a problem faster than by making a simulation of the consequences of the different possible choices. It is a general method which has already been applied to other domains with success. There are several possible ways to enhance the learning system described in this paper:

- Have it find by itself the hierarchical order of goals only given the firm domain theory and the goal of having a financial rentability.

- Test it on management problems taken from the real world.

- Apply it to other domains.

- Making the matching of the rules faster by sharing common subexpressions of the learned rules.

References

Alia C. (1992). *Conception et réalisation d'un modèle didactique d'enseignement de la gestion en milieu professionnel*. Ph.D. Thesis, Montpellier II University, 1994.

Cazenave T. (1996a). *Learning to Forecast by Explaining the Consequences of Actions.* First International Workshop on Machine Learning, Forecasting and Optimization, Madrid, 1996.

Cazenave T. (1996b). *Automatic Ordering of Predicates by Metarules.* 4th International Workshop on Metaprogramming and Metareasonning in Logic, Bonn, 1996.

Dejong G., Mooney R. (1986). *Explanation Based Learning : an alternative view.* Machine Learning 2, 1986.

Minton S. (1988). *Learning Search Control Knowledge - An Explanation Based Approach*. Kluwer Academic, Boston, 1988.

Mitchell T. M., Keller R. M., Kedar-Kabelli S. T. (1986). *Explanation-based Generalization : A unifying view*. Machine Learning 1 (1), 1986.

Pitrat J. (1990). *Métaconnaissances*. Hermès, 1990.