Temporal machine learning for switching control

P. Geurts and L. Wehenkel

University of Liège, Department of Electrical and Computer Engineering
Institut Montefiore, Sart-Tilman B28, B4000 Liège, Belgium
{geurts,lwh}@montefiore.ulg.ac.be

Abstract. In this paper, a temporal machine learning method is presented which is able to automatically construct rules allowing to detect as soon as possible an event using past and present measurements made on a complex system. This method can take as inputs dynamic scenarios directly described by temporal variables and provides easily readable results in the form of detection trees. The application of this method is discussed in the context of switching control. Switching (or discrete event) control of continuous systems consists in changing the structure of a system in such a way as to control its behavior. Given a particular discrete control switch, detection trees are applied to the induction of rules which decide based on the available measurements whether or not to operate a switch. Two practical applications are discussed in the context of electrical power systems emergency control.

1 Introduction

Supervised learning methods, while being very mature, often are only able to handle static input or output values, either numerical or categorical. Nevertheless, many problems whose complexity makes interesting alternative approaches like automatic learning, would need to take into account temporal data. An obvious suboptimal solution is to transform the dynamic problem in order to match the static constraints of traditional learning algorithm but it does not seem to us very appropriate. Besides this simple approach, some recent research on learning classification models with temporal data are described for example in [3],[4] and [5].

In this paper, we present a temporal machine learning method able to automatically construct rules allowing to detect (predict) as soon as possible an event using past and present measurements made on a complex system. This method is able to handle directly temporal attributes and provides interpretable results. We then propose to use this method for the tuning of rules responsible for the triggering of discrete control event in complex systems. Two practical applications are discussed in the context of electrical power systems emergency control. The first one aims at avoiding loss of synchronism in the Western part of the North-American interconnection, while the goal of the second application is the detection of voltage collapse in the South-Eastern part of the French power system.

D.A.Zighed, J.Komorowski, and J.Żytkow (Eds.): PKDD 2000, LNAI 1910, pp. 401-408, 2000.
@Springer-Verlag Berlin Heidelberg 2000

The paper is organized as follows. In Section 2, we define the temporal detection problem. Section 3 is devoted to a description of the learning method we proposed to solve it. In Section 4, we discuss the two practical applications.

2 Temporal detection problem

When dealing with temporal data, we are given a universe U of objects representing dynamic system trajectories (or episodes, scenarios). These objects are described by a certain number of temporal candidate attributes which are functions of object and time, thus defined on $U \times [0, +\infty[$[1]. We denote by $a(o,t)$ the value of the attribute a at time t for the object o, by $\mathbf{a}(o,.)$ the attribute vector $(a^1(o,.),\ldots,a^m(o,.))$ and by $\mathbf{a}_{[0,t]}(o,.)$ its restriction to the interval $[0,t]$.

In a temporal detection problem, we also assume that each scenario o is classified into one of two possible classes $c(o) \in \{+,-\}$. We then call **detection rule** a function of past and present attribute value :

$$d(\mathbf{a}_{[0,t]}(o,.)) \mapsto \{+,-\} \tag{1}$$

which classifies a scenario at each time t and has the following monotonicity property :

$$d(\mathbf{a}_{[0,t]}(o,.)) = + \Leftrightarrow \forall t' \geq t : d(\mathbf{a}_{[0,t']}(o,.)) = +. \tag{2}$$

Thus, we assume that if an object is classified into class + at some time, it will remain so for all later times (monotonicity). We will call the detection time of an object by a detection rule the first time it is classified into class + and we will say that an object is detected by the rule if such a time exists.

The machine learning problem can now be formulated as follows : given a random sample LS of objects whose attributes values are observed for some finite period of time $[0, t_f(o)]$, the objective is to automatically derive a detection rule which would perform as well as possible in detecting objects with class + from U. Clearly a good detection rule is a rule which will detect only those objects which actually belong to class +, and among good rules, the better ones are those having the smallest detection times.

3 Detection rules model : detection trees

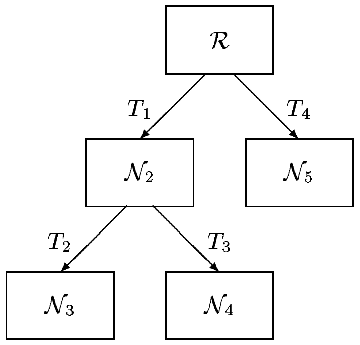
We choose to represent a detection rule by a tree whose arcs have been labeled by elementary tests. Figure 1 shows a simple detection tree and the detection rule it represents. Each test is a logical functional of the attributes :

$$T(\mathbf{a}_{[0\ldots t]}(o,.)) \mapsto \{\mathrm{T},\mathrm{F}\}. \tag{3}$$

which, like detection rules, exhibits a monotonicity property :

$$T(\mathbf{a}_{[0\ldots t]}(o,.)) = \mathrm{T} \Leftrightarrow \forall t' \geq t : T(\mathbf{a}_{[0\ldots t']}(o,.)) = \mathrm{T}. \tag{4}$$

[1] without loss of generality we assume start time of scenario being always 0



$$d(\mathbf{a}_{[0,t]}(o,.)) = + \\ \Updownarrow \\ [T_1 \wedge (T_2 \vee T_3)] \vee T_4$$

Fig. 1. An example detection tree and the corresponding detection rule

Note that the tests are not mutually exclusive; hence, using such a temporal tree, and given a value of t, a scenario is propagated through the tree along all paths starting at the root $\mathcal{R}$ (i.e. the top-node), until a condition T_i along the path is false or a terminal node is reached. If the scenario reaches at least one terminal node, it is classified into class $+$, otherwise it is classified into class $-$ at time t. Notice that the monotonicity property of tests implies the monotonicity of the tree detection.

3.1 Induction algorithm [2]

As in TDIDT (Top Down Induction of Decision Trees [1]), temporal tree induction is based on a quality measure and is composed of two search stages : tree growing and tree pruning. Prior to tree induction, the overall learning set LS is first decomposed in two disjoint subsets : the growing sample (GS) used to evaluate the quality of candidate trees during growing, and the pruning sample (PS) used to evaluate trees during pruning.

Quality measure. In order to assess the quality of a temporal tree from a sample of objects S, we propose the following evaluation function :

$$Q(\mathcal{T}, S) = \frac{(1-\alpha)[(1-\beta)\sum_{o\in\mathcal{T}^+} f(\frac{t_{\mathcal{T}}(o)}{t_f(o)}) + \beta N_{\mathcal{T},+}] + \alpha N_{\mathcal{T},-}}{(1-\alpha)N_+ + \alpha N_-} \in [0,1], \quad (5)$$

where :

- N_+ (resp. N_-) denotes the number of samples of class $+$ (resp. $-$) and $N_{\mathcal{T},+}$ (resp. $N_{\mathcal{T},-}$) the number of them classified $+$ (resp. $-$) by the tree $\mathcal{T}$,
- $\mathcal{T}^+$ denotes the subset of S of objects of class $+$ correctly detected by the tree,
- f(.) is a monotonically decreasing function defined on $[0,1]$ such that $f(0)=1$ and $f(1)=0$ (in our simulation $f(x)=1-x$),
- α and β are two user defined parameters in $[0,1]$,
- $t_{\mathcal{T}}(o)$ is the detection time of o by the rule modeled by the tree,

– $t_f(o)$ denotes the maximal time after which observation of o stops. This time is fixed a priori for each object, and allows one to specify a time after which detection is not considered anymore useful.

This quality measure takes into account two tradeoffs : On one hand, the tradeoff between the selectivity of the rule and the degree of anticipation of the detection and on the other hand the tradeoff between non-detections and false-alarms. The first tradeoff is mainly regulated by β, the second one by α. Indeed, when β is set to 1, detection time does not affect the quality. When α is set to 0 (resp. 1), only correct detections of positive objects via $N_{\mathcal{T},+}$ (resp. correct non-detections of negative objects via $N_{\mathcal{T},-}$) are taken into account in the quality criteria. These tradeoffs are not independent : the more anticipative we want the rule to be, the more false-alarms the system will suffer in general. This relationship appears also in our quality measure as the two parameters actually influence both tradeoffs.

Growing. A greedy approach is used. The algorithm uses a stack of "open" nodes initialized to the root node corresponding to the trivial tree (which classifies all objects into class +). It then proceeds by considering the node at the top of the stack for expansion. At each expansion step, it pushes the created successor on the top of the stack, thus operating in a depth first fashion. For example, the test of the tree at figure 1 would have been added in this order : T_1, T_2, T_3 and T_4. When considering the expansion of a node, it scans the set of candidate tests and evaluates (on the set GS) the increment of detection quality which would result from the addition of each one of these tests individually. If no test can increase quality at this node, then it is removed from the stack and the algorithm proceeds with the next node on the top of the stack. Otherwise, the best test is effectively added to the node, a new successor node is created and pushed on the top of the stack while the node selected for expansion remains on the stack.

Pruning. The tree growing method generally tends to produce overly complex trees, which need to be pruned to avoid overfitting the information contained in the learning set.

The proposed pruning method is similar in principle to the standard tree pruning methods used in TDIDT. It consists in generating a nested sequence of shrinking trees $\mathcal{T}_0, \ldots \mathcal{T}_K$ starting with the full tree and ending with the trivial one, composed of a single node. At each step, the method computes the incremental quality which would be obtained by pruning any one of the terminal nodes of the current tree, and removes the one (together with its test) which maximizes this quantity. It uses an independent set of objects (i.e. different from the learning set) in order to provide honest estimates of the tree qualities, and hence to detect the irrelevant parts of the initial tree. The tree of maximal quality is selected in the sequence.

Candidate tests. The choice of candidate tests depends on the application problem and also on computation efficiency criteria. Indeed, the brute force screening of all candidate tests at each expansion step implies the computation of the quality increment many times and is the most expensive step of the algorithm.

For example, in order to handle numerical attributes in our application, we consider tests in the form :

$$T(o,t) = T \Leftrightarrow \exists t' \leq t \ \mid \ \forall \tau \in [t' - \Delta t \dots t'] \ : a^i(o,\tau) < (>) v_{th}, \qquad (6)$$

thus based only on one attribute. For the sake of efficiency, the user defines for each candidate attribute only a limited number of candidate values for $\triangle t$. Then the method uses a brute force search screening for each candidate attribute and $\triangle t$ value all locally relevant candidate thresholds, computing their incremental quality.

4 Application to switching control of power systems

Switching (or discrete event) control of continuous systems consists of changing the structure of a system in such a way as to control its behavior. In any large scale system (such as power systems) there are generally a large number of possible discrete controls, and an even larger number of possible control laws for each of them. Given a particular discrete control (say on-off status of a switch), the question is to define a rule which decides based on the available measurements whether or not to operate the switch. Typically, measurements are incomplete and noisy to some extent which results in uncertainties. Thus the question is then to process past and present measurement in order to make as reliable decisions as possible. In this section, we will describe two practical applications of detection tree to the induction of control switch triggering rules in the context of the emergency control of power systems.

4.1 Electric power system emergency control

Electric power systems are highly critical infrastructures, responsible for the delivery of electricity over large geographical areas. (For example, the Western European interconnection deserves electric energy to about 300 million customers.) In order to avoid large scale blackouts, these systems are equipped with a large number of control devices operating in different geographical areas and in different time frames. The design of these emergency control systems is difficult due to the highly non-linear, complex and uncertain character of power systems. Because it is not possible to collect data from the field to study the system behavior in highly disturbed conditions, the design is based on simulations. It is generally decomposed into successive steps :

- identification of the main problem causes (type of disturbances and instabilities which are more likely to cause a blackout);
- identification of possible remedies, in terms of control actions (emergency control has to be fast and reliable, and often uses discrete switching control);
- design of control triggering rules (choice of appropriate measurements and decision rules, which should be anticipative and reliable at the same time).

Monte-Carlo simulations together with automatic learning and data mining may be used in order to assist decision making in all these steps [7]. In what

β	α	$P_e(\%)$	$\#fa$	$\#nd$	t_T^{avg}
1.0	0.3	0.17	1	0	3.441
0.5	0.3	1.17	7	0	3.056
0.0	0.3	6.5	39	0	2.239
0.2	0.3	6.83	41	0	2.232
0.2	0.1	11.67	70	0	1.967

Table 1. Results on test set (600 scenarios). P_e = error rate (%), $\#fa$ = number of false alarms (among 485 stable cases), $\#nd$ = number of non detections (among 115 instable cases), T_T^{avg} = average detection time of instable cases (in sec.).

follows, we will show how temporal machine learning of detection rules is suited for the design of control triggering rules. We will consider two different emergency control problems corresponding respectively to detection of loss of synchronism in the Western part of the North-American interconnection, and detection of voltage collapse in the South-Eastern part of the French power system.

4.2 Loss of synchronism

In normal conditions, all the generators in a power system rotate at synchronous speed (corresponding to 50Hz in Europe, 60Hz in North-America). When a large disturbance appears (e.g. a short-circuit due to lightning), some generators start accelerating with respect to others, and, if the disturbance is very large, may loose synchronism. In this context, emergency control aims at anticipating loss of synchronism as quickly as possible so as to open some breakers to avoid cascading outages of lines and generators, which would lead to blackout. Because the phenomenon is extremely fast (a few hundred milliseconds), detection and control must be fully automatic.

The detection tree method was applied to a database composed of 1600 simulations of the North-American power system (about 25% are unstable). The objective is to design a relaying rule used to open the Pacific AC Intertie (a set of lines transmitting energy from Montana and Oregon to California). Opening these lines should reduce the impact of the instability on the overall system behavior [6], and avoid blackout.

Table 1 shows the results (obtained using a LS of 600, a PS of 400 and a TS of 600 scenarios, selected randomly in the database) corresponding to different values of parameters α and β. One can observe how different parameters lead to different types of rule characteristics : smaller values of β (as well as smaller values of α) lead to more anticipative detection (smaller value of average detection time t_T^{avg}) at the price of a higher number of false alarms. The comparison of these results with those obtained using non-temporal machine learning [6], show that our rules are more anticipative and simpler, while being of comparable accuracy.

In all trials, 3 different attributes were used as inputs, corresponding to local measurements in the substation where the breakers are located. The amount of data processed by the detection rule learning algorithm was about 10^6 values,

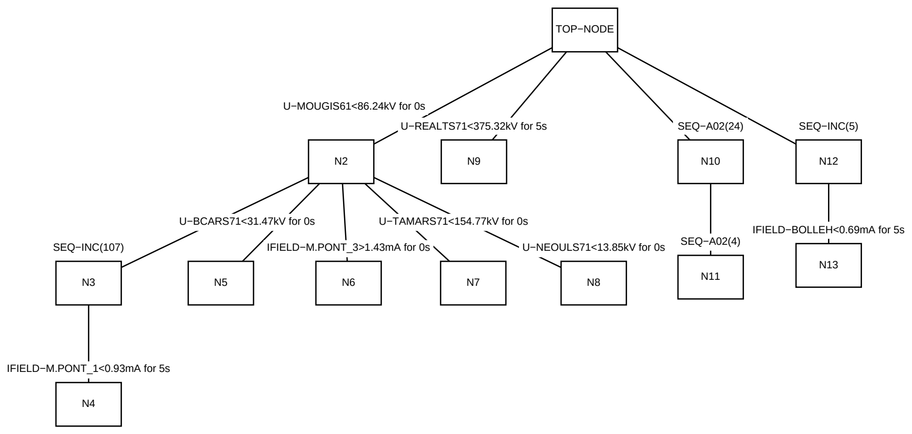

TOP−NODE
U−MOUGIS61<86.24kV for 0s
U−REALTS71<375.32kV for 5s
SEQ−A02(24)
SEQ−INC(5)
N2
N9
N10
N12
U−BCARS71<31.47kV for 0s
U−TAMARS71<154.77kV for 0s
IFIELD−BOLLEH<0.69mA for 5s
SEQ−INC(107)
IFIELD−M.PONT_3>1.43mA for 0s
U−NEOULS71<13.85kV for 0s
SEQ−A02(4)
N3
N5
N6
N7
N8
N11
N13
IFIELD−M.PONT_1<0.93mA for 5s
N4

Fig. 2. A detection tree for the voltage collapse problem. Arcs are labeled with tests. SEQ-XX(n) means that the test is true when at least one out of n events (not described) of type XX occurs. SEQ-INC denotes the sequence of incidents affecting the system and SEQ-A02 denotes the triggering of lines by the installed protections. U-X denotes voltage at node X and IFIELD-Y denotes the excitation current of machine Y.

corresponding to a CPU time of 10 minutes for finalizing tree growing, pruning and testing on a 300 MHz Ultra-Sparc processor.

4.3 Voltage collapse

In the recent years, voltage collapse has been one of the most studied problems in the area of power system emergency control, in particular due to the fact that it has been the cause of several large scale blackouts throughout the world. Usually, this phenomenon is slower than loss of synchronism (in the order of minutes), but still too fast to allow human decision making. Typical control actions consist of disconnecting temporarily some customers so as to avoid the system collapse. Of course, such control actions should be triggered only when absolutely necessary but early enough to be effective, hence the need to design appropriate decision rules which are both anticipative and selective.

We will illustrate the detection tree method on a database from a large scale study carried out on the system of Electricité de France [8]. It is composed of 1400 simulations, characterized all in all by about 800 temporal attributes (total of 2GB of raw data). In our simulations, we used only 42 of these attributes pre-selected as being potentially relevant for the problem under consideration (29 numerical time-series, and 12 sequences of events). Still, the amount of data is quite important (12.10^6 values all in all).

Figure 2 shows one particular detection tree thus obtained. To build the tree, the method has automatically selected a reduced set of attributes, together with appropriate thresholds and delays. If compared to presently used detection rules, this rule happens to be much more anticipative and accurate. Notice that the delays selected by the machine learning method are here in the order of a few seconds (compared to 50 to 100 ms in the loss of synchronism problem), which shows that the method is indeed able to identify the appropriate dynamic features of the problem.

Here, the overall CPU time required to build the rule was about 3 hours on a 300 MHz Ultra-Sparc processor, which in this particular application is negligible with respect to the time required to generate the database by simulations (3 months of CPU using a cluster of 10 high-end workstations [8]).

5 Conclusions

In this paper, we have discussed the development of an automatic induction method of detection rules and its application was proposed to the inference of triggering rules of control switch. This general framework was applied to the emergency control of electric power systems in the case of two different kinds of system failures, loss of synchronism and voltage collapse.

In the future, we will consider extensions of our detection tree method along two axes : hypothesis space and search strategy. The semantics of detection trees should be extended to allow the discovery of more complex temporal features. At its actual stage (and for efficiency criteria), the method does not permit to represent temporal constraints on the verification of tests, as for example in rules like "if T_1 becomes true and T_2 becomes true strictly after T_1 then ...". This capability has already been added to our system, but tends to be ineffective in terms of accuracy improvement on the studied problems. Expansion of the hypothesis space also will need to call into question the greedy search strategy which may not be appropriate anymore.

References

1. L. Breiman, J.H. Friedman, R.A. Olsen, and C.J. Stone. *Classification and Regression Trees.* Wadsworth International (California), 1984.
2. P. Geurts and L. Wehenkel. Early prediction of electric power system blackouts by temporal machine learning. In *Proc. of ICML-AAAI'98 Workshop on "AI Approaches to Times-series Analysis"*, Madison (Wisconsin), 1998.
3. M. W. Kadous. Learning comprehensible descriptions of multivariate time series. In *Proceedings of the Sixteenth International Conference on Machine Learning, ICML'99*, pages 454–463, Bled, Slovenia, 1999.
4. S. Manganaris. *Supervised classification with temporal data.* PhD thesis, Vanderbilt University, 1997.
5. V. Petridis and A. Kehagias. *Predictive modular neural network : applications to time series.* Kluwer Academic Publishers, 1998.
6. S. Rovnyak, C. Taylor, and Y. Sheng. Decision trees using apparent resistance to detect impending loss of synchronism. *To appear in IEEE Transactions on Power Delivery*, 2000.
7. L. Wehenkel. *Automatic learning techniques in power systems.* Kluwer Academic, Boston, 1998.
8. L. Wehenkel, C. Lebrevelec, M. Trotignon, and J. Batut. Probabilistic design of power-system special stability controls. *Control Engineering Practice*, 7(2):183–194, 1999.