Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 109C (2017) 1188–1193

Procedia Computer Science

www.elsevier.com/locate/procedia

International Workshop on Adaptive Technology (WAT 2017)

Learning decision rules using adaptive technologies: a hybrid approach based on sequential covering

Renata L. Stange[a,b,*], João J. Neto[a]

[a] *EPUSP, Av. Professor Luciano Gualberto, Travessa 3, 380 – Butantã, São Paulo – SP, Brasil*
[b] *UTFPR, Av. Professora Laura Pacheco Bastos, 800 – Industrial, Guarapuava – PR, Brasil*

Abstract

Sequential covering strategies are commonly used in rule learning algorithms, as they apply separate and conquer approaches, in which the task of finding a complete rule base is reduced to a sequence of subproblems; each solution to a subproblem consists in adding a single rule. We propose an alternative for rule learning, namely the use of adaptive devices whose behavior is defined by a dynamic set of rules. In order to integrate features from rule learning methods and adaptive technologies, this paper presents a hybrid approach to learn classification rules, such that the set of rules is dynamically modified by adding or removing rules. The results have been promising towards applying adaptive technology to learn rules directly from data.

1877-0509 © 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the Conference Program Chairs.

Keywords: adaptive technologies, adaptivity, sequential covering strategy, learn rules

1. Introduction

Machine learning is concerned about the construction of computer programs that automatically improve with experience[1]. Similarly, pattern recognition is interested in the automatic discovery of regularities in data through computer algorithms and the later application of such regularities as decision making actions, such as different data categorizations[2]. Adaptive technology refers to the use of techniques and devices which are expected to react to given inputs by autonomously modifying their own behavior[6]. In a broad sense, computers learn when there is a behavioral change in order to better perform a specific task. Inspired by previous works on these areas[5,3,4], we demonstrate how to behavior of learning systems can be dynamically adjusted by using adaptive techniques. A direct advantage on incorporating adaptivity in learning methods consists on covering a fundamental aspect of learning itself: the dynamic adaptation of the set of rules based on interactions with the environment[7]. Additionally, the use of adaptive technology might be more expressive than traditional mthods[6].

* Corresponding author. Tel.: +55-042-3141-6850.
E-mail address: rlgomes@utfpr.edu.br

1877-0509 © 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the Conference Program Chairs.
10.1016/j.procs.2017.05.396

This paper presents a hybrid approach to extract rules from data using adaptive technologies and supervised learning techniques, such as learning if-then rules from data. The approach is based on sequential covering strategies which involves problem decomposition: the task of finding a complete rule base is reduced to a sequence of subproblems in which the solution to each subproblem is a single rule. The global solution gathers all partial solutions[8]. Additionally, the self-modification feature of adaptive rule-driven devices allows an iterative knowledge inspection, adding rules that improve predictions on the rule base and also replacing one or more rules in order to simplify the knowledge. Algorithms AQ[9] and CN2[10] implement this strategy.

Adaptive technology has been successfully used in pattern recognition and machine learning. Pistori and Neto[11] propose a decision tree induction algorithm using adaptive techniques, combining syntactic and statistical strategies. In[12], Pistori presents an adaptive automaton as device for an automatic recognition process of sign language. Adaptive automata are also reported to be used in syntactic pattern recognition of shapes[13] and in construction of hybrid maps for robot navigation[14]. Other applications of adaptive techniques include skin cancer recognition[15] and optical character recognition[16].

2. Decision Rules

Decision rules are widely used to represent knowledge either obtained from data[17]. An *if-then* rule has a simple construction; for instance, if an certain object swims and has scales, then such object is a fish. Rule-based methods are fundamental do expert systems in artificial intelligence, where classes can be characterized by general relationships among entities. Here we shall focus on a broad class of *if-then* rules for representing and learning such relationships. The general form of a *if-then* rule is `P` $\rightarrow$ `Q` or `if P then Q`, where:

- `P` is a proposition that can contain a conjuntion of n arbitrary attribute/value pairs, $P = \text{condition}_1 \wedge \ldots \wedge \text{condition}_n$. It is important to observe that n is known as the rule length.
- `Q` is the value of the categorical target attribute.

2.1. Rule-based methods

According to Mitchell[1], a possible approach to learn sets of rules involves learning a decision tree through an induction algorithm, such as ID3[18], followed by a translation of such tree to an equivalent set of rules. A decision tree can be mapped to a set of rules, transforming each branch into a rule, i.e, each path from the root to one leaf corresponds to a rule. A popular technique is called *sequential covering*. The following steps show a sketch of how a general set covering algorithm works[17]:

1. Create a rule that covers some examples of a certain class an does not cover any examples of other classes.
2. Remove covered examples from training data.
3. If there are some examples not covered by any rule, go to step 1.

These algorithms work through separate and conquer strategies, in which during each pass some examples of the target concept are described by a single rule and then removed from the training data. A new candidate rule can be created either by specialization or generalization; the former adds a new attribute/value pair to the rule condition, while the latter removes an existing attribute/value pair from the rule condition. The set covering algorithm performs a top-down or bottom-up search in the space of all possible rules. Each rule defines a rectangular region in the attribute space.

A prototypical sequential covering algorithm is described in Algorithm 1. As reported by Mitchell[1], Learn-one-rule must return a single rule that covers at least some of the *examples*. Performance is a user-provided subroutine to evaluate the rule quality. This covering algorithm learns rules until it can no longer learn a rule whose performance is above the give *threshold*. Common evaluation functions include[1] relative frequency, m-estimate of accuracy and entropy. For instance, we use a relative frequency w where n denotes the number of examples the rule matches, and n_c denotes the number of examples that it classifies correctly. The relative frequency estimate of rule performance is given by $w = n_c/n$.

Algorithm 1 Sequential Covering algorithm

procedure SEQUENTIAL COVERING(*target attributes, attributes, examples, threshold*)
  learned rules ← {}
  rule ← LEARN-ONE-RULE(*target attributes, attributes, examples*)
  while PERFORMANCE(*rules, examples*) ≤ *threshold* **do**
    learned rules ← *learned rules* + *rule*
    examples ← *examples* − {*examples* correctly classified by *rule*}
    learned rules ← sort *learned rules* according to PERFORMANCE over *examples*
  end while
  return *learned rules*
end procedure

3. Rule-Driven Adaptive Devices

A *rule-driven device ND* is any formal abstraction whose behavior is described by a rule set that maps each possible configuration into a corresponding following one. The complete formulation is be found in [7]. An *adaptive rule-driven device* $AD = (ND_0, AM)$ associates an initial subjacent rule-driven device ND_0, to some adaptive mechanism AM, that can dynamically change its behavior by modifying its defining rules. That is accomplished by executing *non-null adaptive actions* chosen from sets BA and AA of adaptive actions. *Adaptive actions* in BA and AA call functions that map AD's current set AR_t of adaptive rules into AR_{t+1} by inserting to and removing adaptive rules ar from AM. Let AR be the set of all possible sets of adaptive rules for AD. Any $a^k \in AR$ maps the current set of rules $AR_t \in AR$ into $AR_{t+1} \in AR, AD_k = (C_k, AR_k, S, c_k, A, NA, BA, AA)$, where:

- C_k is its set of k possible configurations in step k,
- AR_k is its set of all possible sets of adaptive rules for AD, $AR_k \subseteq BA \times C \times S \times C \times NA \times AA$,
- S is its set of valid input stimuli, and $\epsilon \in C$ and it denotes an empty stimulus,
- $c_k \in C_k$ is its initial configuration,
- $A \subseteq C$ is its set of final configurations,
- NA is its set all possible output symbols of ND, and
- BA and AA are adaptive actions.

Adaptive actions may be defined as abstractions called adaptive functions, in a similar usage as function calls and function declarations in a usual programming language. In order to specifying adaptive functions, further information is needed: the *name* of the corresponding adaptive function, the *set of parameters* to be used, the *elementary adaptive actions* to be applied and the *parameters*, *variables* and *generators* to be employed. An adaptive functions assumes the form $A_i(\langle\text{parameters}\rangle)\{\langle\text{list of elementary adaptive actions}\rangle\}$.

An *elementary adaptive action* denotes an operation to be performed on the set of rules. The format of an elementary adaptive action is presented as $(? \mid - \mid +)\langle\text{pattern}\rangle$. Note that *pattern* orresponds to the representation of a rule $a^k \in AR$. The notation ?(pattern), −(pattern), and +(pattern) represents search, removal and insertion of adaptive rules that follow the *pattern* template, respectively. In summary:

- ?(⟨pattern⟩): the action search for rules that obey a certain pattern.
- −(⟨pattern⟩): the action removes rules that match a certain pattern from the current set of rules.
- +(⟨pattern⟩): the action inserts a rule that corresponds to a certain pattern in the current set of rules.

For practical reasons, up to two adaptive actions are allowed in an adaptive rule [7], such that one action is performed prior to the execution of an underlying rule, and another action is performed after the execution.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	Normal	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	High	Strong	Yes
D8	Sunny	Mild	Normal	Weak	No
D9	Sunny	Hot	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Cool	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Fig. 1. Training examples

4. A Hybrid Method to Learn Rules

Our approach of an adaptivity model is inspired on the previous sequential covering description, with subtle yet significant differences. According to Algorithm 1, subroutine LEARN-ONE-RULE accepts a set of positive and negative training examples as input and returns as output a single rule that covers many of the positive examples and few of the negative examples. We propose a modification to this subroutine, such that it now returns two rules that cover both positive and negative examples (we have also renamed such subroutine to LEARN-RULES). We also have a different treatment on the set of examples, such that LEARN-RULE is invoked on all available training examples and removing positive or negative examples completely covered by the rule it learns, i.e, whenever *relative* frequency is 1.

The sequential covering strategy uses a greedy strategy that takes the best local actions where it iteratively learns a single rule, so that each learned rule is added to the rule base. Generally, those rules being added in each iteration keep being part of the rule base until the end of the learning process. Thus, our proposal aims to modify of the sequential covering strategy, making it able to review already learned rules through adaptive functions. In each step, it is possible to decide between two options:

1. add the rule that improves the prediction capability of the rule base (however, if it has a relative frequency lower than 1, it could be marked for removal), or
2. add the best rule that replaces one or more rules of the rule base, as an attempt to increase the prediction capability, when the relative frequency of the rule is 1.

A possible approach to implement the LEARN-ONE-RULE subroutine consists of organizing the hypothesis space search similarly to ID3 algorithm's behavior[18], but restricting the search to the most promising tree branch at each step. Our approach uses a breadth-first search to construct the next rule. Since the rule consequent must be the given class (in this case, positive or negative), only the antecedent needs to be constructed; this is achieved by starting with an empty antecedent and iteratively adding an attribute/value pair for all attribute values. The relative frequency is used to rate candidate rules. This breadth-first search continues until the resulting rule is specific enough. In general, the approach assumes that:

1. the algorithm obtains obtains a data set containing training examples as input,
2. the dataset consist of nominal attributes,
3. the attributes hold a predefined order, and
4. initially, the learner starts with an set of rules containing a single adaptive rule.

4.1. An example illustration the algorithm

For didactic purposes, we present an example on how the proposed method can be used to generate the classification rules from a data set. In order to ease the understanding, we have selected a well-known data set proposed by Quinlan[18], named *play tennis*. It is presented in Fig. 1

Finding the first rule The main loop searches for a default rule R_i in a set of rules AR_k. Default rule R_i is defined as $R_i = \texttt{P} \rightarrow \texttt{Q (w)}[A_i]$.

Initialization Initially, the set of rules is defined as $AR_0 = \{R_0 : \texttt{P} \rightarrow \texttt{Q (1)}[A_0], \quad R_1 : \texttt{P} \rightarrow \texttt{Q (w)}[A_1]\}$.

First step through the loop The relative frequency is computed for all candidate rules in the form $\texttt{P} \rightarrow \texttt{Q (w)}$, as seen in Table 1 ($D = \{D_1, D_2, \ldots D_{14}\}$).

Table 1. Candidate rules to attribute outlook = $\{o_1 = \text{sunny}, o_2 = \text{rain}, o_3 = \text{overcast}\}$

Input stimulus	Search and application	Adaptive action performed
$s_1 = o_1 \rightarrow$ yes $(5/2)(D_9, D_{D11})$	Let AR_0 apply R_1, call A_1	$A_1(o_1, \text{yes}, w = 2, 5)\{var^1 = ?(o_1 \rightarrow \texttt{Q})(w), +(0_1 \rightarrow \text{yes } (2, 5)[A_2])\}$
$s_2 = o_1 \rightarrow$ no $(5/3)(D_1, D_2, D_8)$	Let AR_1 apply R_1, call A_1	$A_1(o_1, \text{no}, w = 1, 6)\{var^2 = ?(o_1 \rightarrow \texttt{Q})(w), -(var), +(0_1 \rightarrow \text{no } (1, 6)[A_3])\}$
$s_3 = o_2 \rightarrow$ yes $(5/3)(D_4, D_5, D_{D10})$	Let AR_2 apply R_1, call A_1	$A_1(o_2, \text{yes}, w = 1, 6)\{var^3 = ?(o_2 \rightarrow \texttt{Q})(w), +(0_2 \rightarrow \text{yes } (1, 6)[A_4])\}$
$s_4 = o_2 \rightarrow$ no $(5/2)(D_6, D_{D14})$	Let AR_3 apply R_1, call A_1	$A_1(o_2, \text{no}, w = 2, 5)\{var^4 = ?(o_2 \rightarrow \texttt{Q})(w)\}$
$s_5 = o_3 \rightarrow$ yes $(4/4)(D_3, D_7, D_{12}, D_{13})$	Let AR_3 apply R_0, call A_0	$A_0(o_3, \text{yes}, w = 1)\{var^5 = ?(o_3 \rightarrow \texttt{Q})(w), +(0_3 \rightarrow \text{yes } (1))\}$

1 `var = NULL then add` R_2, $AR_0 \rightarrow AR_1$
2 `var =` R_2 `and` $(w < t)$ `then replace` R_2, $AR_1 \rightarrow AR_2$
3 `var = NULL then add` R_3, $AR_2 \rightarrow AR_3$
4 `var =` R_3 `and` $(w > t)$ `then do not anything`
5 `var = NULL then add` R_4, $AR_3 \rightarrow AR_4$

After the first step the rule set is as follows:

$AR_4 = \{$
R_0: P $\rightarrow$ Q (1) $[A_0]$,
R_1: P $\rightarrow$ Q (w) $[A_1]$,
$R_2 : 0_1 \wedge \alpha \rightarrow$ no (1,6) $[A_3]$,
$R_3 : 0_2 \wedge \alpha \rightarrow$ yes (1,6) $[A_4]$,
$R_4 : 0_3 \wedge \alpha \rightarrow$ yes (1) $\}$ *This rule will not be modified, because it covers*

The symbol α is used to represent any value in the attribute domain. This completes the first pass through the inner loop in the sequential covering algorithm. Since there is still one uncovered remaining instance, another step is performed in order to generate additional rules.

Second step through the loop The relative frequency is computed for all candidate rules ($D = \{D_1, D_2, D_4, D_5, D_6, D_8, D_9, D_{10}, D_{11}, D_{14}\}$). See Table 2 for more details.

Table 2. Candidate rules to attribute outlook = $\{o_1 = \text{sunny}, o_2 = \text{rain}\}$ and temperature = $\{t_1 = \text{hot}, t_2 = \text{mild}, y_3 = \text{cool}\}$

Input stimulus	Search and application of rules	Adaptive action performed
$s_6 = o_1 \wedge t_1 \rightarrow$ yes $(5/1)(D_9)$	Let AR_4 apply R_2 and call A_3	var = R_2 and $(w > t)$ then do not anything
$s_7 = o_1 \wedge t_1 \rightarrow$ no $(5/2)(D_1, D_2)$	Let AR_4 apply R_2 and call A_3	var = R_2 and $(w > t)$ then do not anything
$s_8 = o_1 \wedge t_2 \rightarrow$ no $(5/1)(D_8)$	Let AR_5 apply R_2 and call A_3	var = R_2 and $(w > t)$ then do not anything
$s_9 = o_1 \wedge t_3 \rightarrow$ yes $(5/1)(D_{11})$	Let AR_5 apply R_3 and call A_4	var = R_3 and $(w > t)$ then do not anything
$s_{10} = o_2 \wedge t_2 \rightarrow$ yes $(5/2)(D_4, D_{10})$	Let AR_5 apply R_3 and call A_4	var = R_3 and $(w > t)$ then do not anything
$s_{11} = o_2 \wedge t_2 \rightarrow$ no $(5/1)(D_{14})$	Let AR_5 apply R_3 and call A_4	var = R_3 and $(w > t)$ then do not anything
$s_{12} = o_2 \wedge t_3 \rightarrow$ yes $(5/1)(D_5)$	Let AR_5 apply R_3 and call A_4	var = R_3 and $(w > t)$ then do not anything
$s_{13} = o_2 \wedge t_3 \rightarrow$ no $(5/1)(D_6)$	Let AR_5 apply R_3 and call A_4	var = R_3 and $(w > t)$ then do not anything

After the loop, the set of rules is then updated by adding new rules. Nevertheless, set AR_5 is not updated; based on such result, we do not consider the attribute *temperature* in the upcoming iterations. This strategy follows until all

instances have been covered. Intermediate steps were omitted for simplicity's sake, as they were not relevant to the method explanation.

Last step through the loop The relative frequency is computed for all candidate rules.

Let us assume that:

$AR_7 = \{$
R_0: P → Q (1) [A_0],
R_1: P → Q (w) [A_1],
R_4: $0_3 \wedge \alpha \rightarrow$ yes (1),
R_5: $0_1 \wedge h_1 \rightarrow$ no (w) [A_5],
R_6: $0_1 \wedge h_2 \rightarrow$ yes (w) [A_6],
R_7: $0_2 \wedge w_1 \rightarrow$ no (w) [A_7],
R_8: $0_2 \wedge w_2 \rightarrow$ yes (w) [A_8] }

Note that all rules but R_4 can be improved as new training instances become available.

5. Conclusions and Future Work

This paper presented an approach towards the definition of an adaptive learning algorithm to infer rules. This approach is based on modifications of conventional sequential covering strategies in order to adapt acquired knowledge during the learning process. Studies show that adaptive technologies allow rule set fitting. As future work, comparative experiments with other sequential coverage algorithms are planned, in order to verify the model accuracy. Prospectively, we aim at studying algorithm adaptability when new examples are available, as well as cases that missing values must be considered.

References

1. Mitchell, T. M.: *Machine Learning*. 1a Ed. McGraw-Hill, 1997. ISBN: 0070428077.
2. Bishop, C. M.: *Pattern Recognition and Machine Learning*. Berlin: Springer, 2006. ISBN 0-387-31073-8.
3. Stange, R. L.; Neto, J. J. *Applying Adaptive Technology in Machine Learning*. Revista IEEE América Latina, Vol.12(7), pp.1298-1306, 2014.
4. Stange, R. L.; Neto, J. J. *Utilização de técnicas adaptativas em aprendizagem de máquina e reconhecimento de padrões* (resumo). Em: Memórias do X Workshop de Tecnologia Adaptativa. EPUSP, São Paulo. ISBN: 978-85-86686-86-3, p. 5. 28 e 29 de Janeiro, 2016.
5. Stange, R. L.; Neto, J. J.: *Aprendizagem Incremental Usando Tabelas De Decisão Adaptativas*. Memórias do WTA 2011 -Workshop De Tecnologia Adaptativa, EPUSP, São Paulo, 2011.
6. Neto, J. J.: *Solving complex problems with Adaptive Automata*. Lecture Notes in Computer Science. S. Yu, A. Paun (Eds.): Implementation and Application of Automata 5th International Conference, CIAA 2000, Vol.2088, London, Canada, Springer-Verlag, pp.340, 2000.
7. Neto, J. J. Ad*aptive Rule-Driven Devices: General Formulation and Case Study*. Revista de Engenharia de Computação e Sistemas Digitais, São Paulo, v.1, n.1, p. 45-57, 2001.
8. Fürnkranz J., *Separate-and-Conquer Rule Learning*. Artificial Intelligence Review, vol. 13, p.3-54, 1999.
9. Michalski, R. S. *On the quasi-minimal solution of the covering problem*. Proceedings of the 5th International Symposium on Information Processing (FCIP-69) (pp. 125–128). Bled, Yugoslavia, 1969.
10. Clark, P., Niblett, T.*The CN2 induction algorithm*, Machine Learning 3(4), pp. 261-283, 1987.
11. Pistori, H. e Neto, J. J. *AdapTree – Proposta de um Algoritmo para Indução de Árvores de Decisão Baseado em Técnicas Adaptativa*s. Anais Conferência Latino Americana de Informática -CLEI 2002. Montevideo, Uruguai, Novembro, 2002.
12. Pistori, H., Neto, J. J. *An Experiment on Handshape Sign Recognition using Adaptive Technology: Preliminary Results*. XVII Brazilian Symposium on Artificial Intelligence - SBIA 04. São Luis, September 29 - October 1, 2004.
13. Costa, E.R. and Hirakawa, A.R.and Neto, J. J. A*n Adaptive Alternative for Syntactic Pattern Recognition*. Proceeding of 3rd International Symposium on Robotics and Automation, ISRA 2002, Toluca, Mexico, September 1-4, 2002, pp. 409-413.
14. Hirakawa, A. R.; Saraiva, A. M.; Cugnasca, C. E. *Autômatos Adaptativos Aplicados em Automação e Robótica*. Revista IEEE América Latina. Vol. 5, Num. 7, ISSN: 1548-0992, pp. 539-543, Novembro 2007.
15. Ganzeli, Bottesini, Paz e Ribeiro. *SKAN, Skin Scanner, software para o reconhecimento de câncer de pele utilizando técnicas adaptativas*. In: Quarto Workshop de Tecnologia Adaptativa – WTA 2010. EPUSP, 2010.
16. Doy, Souza e Jankauskas. *AOCR, adaptive optical character recognition*. In: Quarto Workshop de Tecnologia Adaptativa -WTA 2010. EPUSP, 2010.
17. Berka, Petr and J. Rauch. *Machine Learning and Association Rules*. In 19th Int. Conf. On Computational Statistics COMPSTAT 2010. 2010.
18. Quinlan, J. R. *Induction of decision trees*. Machine Learning, 1, 81-106, 1986.
19. Fürnkranz J., Gamberger D., Lavrač N., *Foundations of Rule Learning*. Cognitive Technologies, 2012.