DATA FITTING
WITH
RULE-BASED REGRESSION

Luis Torgo

LIACC
R.Campo Alegre, 823 - 2o.
4150 PORTO
PORTUGAL

Phone : (+351) 2 600 16 72 - Ext. 115 *Fax* : (+351) 2 600 3654
e-mail : ltorgo@ncc.up.pt *http* ://www.up.pt/~ltorgo

Abstract. In the classical regression theory we try to build one functional model to fit a set of data. In noisy and complex domains this methodology can be highly unreliable and/or demand too complex functional models. Piecewise regression models provide means to overcome these difficulties. Some existing approaches to piecewise regression are based on regression trees. However, rules are known to be more powerful descriptive languages than trees. This paper describes the rule learning system $\mathcal{R}^2$. This system learns a set of regression rules from a classical machine learning data set. Regression rules are IF-THEN rules that have regression models in the conclusion. The conditional part of these rules determines the domain of applicability of the respective model. We believe that by adopting a rule-based formalism, $\mathcal{R}^2$ will out-perform regression trees. The initial set of experiments that we have conducted in artificial data sets show that $\mathcal{R}^2$ compares reasonably to other machine learning systems.

Keywords : Regression, Data Fitting, Inductive Learning, Propositional Rule Learning.

1. INTRODUCTION

Regression is an important problem in data analysis. Given a sample of a set of independent variables $x_1,...,x_n$ together with a dependent variable y, we are trying to approximate the function $y = f(x_1,...,x_n)$.

The machine learning (ML) field has been mainly concerned with the problem of classification. Regression differs from classification because each example is attributed a numeric class instead of a finite set of symbolic labels. One of the main advantages of ML systems compared to systems from other research fields is on comprehensibility of the results [11]. This is a key motivation for trying to apply ML algorithms to regression problems.

In the field of ML most of the existing work on these problems is based on regression trees (CART [2], M5 [12, 13], RETIS [7, 8]). These algorithms have a very simple and efficient algorithm. M5 and RETIS are able to learn trees with linear regression models in the leaves. CART is limited to have average values in the leaves thus proving poor extrapolation from the original data. One exception to regression trees is the work presented in [18] on regression rules. This system transforms the data into a classification problem by a process of discretization. This process will inevitably carry some loss of information.

Regression methods were extensibly studied for many years in the field of statistics. Simple and efficient methods like linear least squares regression have proven to be very effective in many real-world applications. Nevertheless, non-linearity has been receiving increasing attention by the statistics community. Among new methods that appeared in recent years we can refer projection pursuit [6] and MARS [5].

Neural networks (NN's) are also good function approximators and much work exist on these tasks (see for instance the work of [9] with back-propagation NN's).

In this paper we present $\mathcal{R}^2$, a propositional inductive learning system. This system is able to learn a set of regression rules from samples of a function mapping. Compared to regression trees, $\mathcal{R}^2$ has the advantage of richer descriptive language (rules). The target models of these systems are similar[1] to the current version of $\mathcal{R}^2$ (with the exception of CART). Nevertheless, we intend to extend the set of models used by our system in the near future. On the issue of accuracy the tests we have made on some artificial domains shown that $\mathcal{R}^2$ has comparable performance to other ML systems.

$\mathcal{R}^2$ has the advantage of comprehensibility over the statistics and neural nets approaches. This can be an important factor in many domains. However, we should also be concerned with experimental results on accuracy. We still did not make any comparisons with these type of systems. An interesting account on evaluating techniques from all these fields on common data sets can be found in [11].

Regression problems appear in many real world applications. Among them we can refer the field of control. Several authors have tried to develop controllers of dynamic systems based on records of human interaction with the system (see for instance [17]). The controlled variables of these problems are usually numeric so we can look at this as a regression problem. Most of the work on applying ML systems to these tasks has been done using classification systems. Regression systems like $\mathcal{R}^2$ could bring some improvement on the accuracy of these machine-generated controllers.

The following section gives an overview of $\mathcal{R}^2$. We describe its main components and give an illustrative example. We then show some results of comparative experiments carried out on artificial domains. We end with some conclusions and future work.

---

[1] M5 with its default parameters performs term simplification on linear regression models, while $\mathcal{R}^2$ currently does not.

2. THE $\mathcal{R}^2$ SYSTEM

$\mathcal{R}^2$ resembles in many ways other propositional learning systems. It receives as input a set of examples, each of them described by a set of attributes[2] and labeled by a class (in this case a number). The algorithm of $\mathcal{R}^2$ consists of two main steps. It starts by trying to develop a regression model for the set of examples under consideration. We then check if by specializing the domain of applicability of the resulting model we obtain a better fit of the data. The resulting rule is added to the theory and the system repeats the process until a given criterion is reached. Figure 1 presents this algorithm :

```
UncoveredRegion <- AllDomain
WHILE CurrentCoverage < CoverageRequestedByUser DO
      Select an uncovered region of the Domain
      Build an Unconditional Model for that region
      REPEAT
            Restrict the domain of applicability of the current model
      UNTIL the resulting specialization is worse than the current model
```

Fig. 1 - $\mathcal{R}^2$ main algorithm.

Let us see how $\mathcal{R}^2$ selects an uncovered region. The goal of this step is to find a condition that defines a region of the domain where we can find some examples yet uncovered by the current rules. In the beginning of the learning process this condition is empty as all domain is uncovered. As soon as new rules are learned $\mathcal{R}^2$ finds these uncovered regions by looking at the rule conditions. Imagine that we have the following rules in our theory :

$R_0 \equiv$ IF $X_1 > 13 \wedge X_2 \leq 0.5$ THEN $Y = 10 - 0.5 \times X_3$

$R_1 \equiv$ IF $X_3 > 5$ THEN $Y = -43.6 - 2.5 \times X_1$

$\mathcal{R}^2$ negates each condition of all rules and chooses the condition that is satisfied by more uncovered examples. With the rules presented above one of the conditions of the set $\{ X_1 \leq 13, X_2 > 0.5, X_3 \leq 5\}$[3] is chosen for developing a regression model. This condition is satisfied by a set of examples. Some of them are uncovered but others can already be covered by other rules in the theory. For instance, if the chosen condition was $X_3 \leq 5$, it is possible that an example already covered by the rule R_0 also satisfies this condition. The consequence of this is that $\mathcal{R}^2$ can learn different rule models for the same data. This form of redundancy is similar to the one used in the classification system Yails [16]. In noisy data it has obvious advantages [16] when compared to the more common covering algorithms (like CN_2 [3] or AQ [10]).

The next step in the algorithm consists of developing a regression model for the chosen condition. $\mathcal{R}^2$ uses a lattice of increasingly complex regression model languages as the basis of this process. The system builds a regression model using each of these languages and chooses the one with smaller fitting error. At this stage $\mathcal{R}^2$ is very similar to

---

[2] Currently, the system is restricted to numeric attributes.

[3] We are not saying that these conditions represent all the uncovered areas of the search space. They are just portions of the uncovered area that can be efficiently found.

a classical regression system. Given a set of data it tries to develop a regression model for it. The difference is that the system tries several types of regression models but the algorithm is independent of the strategy followed for this task. As a consequence it is very easy to either enlarge the set of languages or even apply any standard regression package to the data. Currently, as the system is in an initial stage of implementation, we are using a limited set of regression models. This set is equivalent to the one used by M5 or RETIS and it is presented bellow :

$$L_0 \equiv y = \bar{y}$$

$$L_1 \equiv y = K_0 + K_1 \times A_1 + \ldots + K_n \times A_n$$

where y is the variable representing the class
$\bar{y}$ is the average class value
K_i's are constant values
and A_i's are attribute values

The final step in the algorithm consists of trying to specialize the model previously built. This is an iterative search procedure that ends when no further improvements are possible according to a given evaluation criterion that will be explained later.

The set of candidate specializations is built in the following way. Given a rule covering a set of examples $\mathcal{R}^2$ uses the example that the rule worse fits as the source for this process. The algorithm tries to exclude this example from the covered set by adding some condition to the rule. For each attribute of the domain $\mathcal{R}^2$ checks the value of the example and tries conditions on that attribute that guarantee its exclusion. Imagine the following rule with its respective worse fit example :

$R_3 \equiv$ IF $X_3 \leq 5$ THEN $Y = -4 + 0.5 \times X_4 + X_2$

$Ex \equiv X_1 = 10 \wedge X_2 = 0.5 \wedge X_3 = 0.2 \wedge X_4 = 21 \wedge X_5 = 6.32 \Rightarrow Y = 45$

$\mathcal{R}^2$ would generate the set of candidate specializations of rule R_3 by adding each of the following conditions:

$X_1 < 10$	$X_1 > 10$	$X_2 < 0.5$	$X_2 > 0.5$	$X_3 > 0.2$
$X_3 < 0.2$	$X_4 < 21$	$X_4 > 21$	$X_5 < 6.32$	$X_5 > 6.32$

If any of the resulting specializations is better than R_3 according to the evaluation criterion that will be presented bellow, $\mathcal{R}^2$ will consider it the new best rule and will proceed by applying the same worst-fit exclusion algorithm to this new rule.

2.1 EVALUATING THE SPECIALIZATIONS OF A RULE

The goal of the evaluation function is to compare two rules (one being a specialization of the other). There are two consequences of specializing a rule. We have a gain in terms of fitting, but we loose coverage as the domain is restricted[4]. The evaluation function should weigh these factors producing a quality measure of the specialization that enables the comparison to the original rule. $\mathcal{R}^2$ uses a weighted average of these two factors as evaluation function.

[4] There is also a lost in terms of simplicity of the rule, but at this stage we are not considering this factor.

$\mathcal{R}^2$ measures de degree of fitting of a model using a statistic called Mean Absolute Deviation (MAD) which is defined in equation 1 :

$$MAD = \frac{\sum_{i \in Exs} |y'_i - y_i|}{\#\, Exs} \qquad \textbf{(1)}$$

where y'_i is the system's prediction
y_i is the real class value
#Exs is the number of examples from which the model was built

Coverage is measured by the number of examples that satisfy the rule. Notice that we start this specialization process with a rule and we are trying to compare it with its specializations. For each candidate specialization we calculate the gain in fitting error and the loss in coverage compared to this original rule. These factors are calculated by :

$$GainMAD = \frac{MAD_{init} - MAD_{spec}}{MAD_{init}} \quad \text{and} \quad LossCOV = \frac{COV_{init} - COV_{spec}}{COV_{spec}} \qquad \textbf{(2)}$$

where MAD_{init} is the MAD of the initial rule (before the specialization process)
MAD_{spec} is the MAD of the candidate specialization
COV_{init} is the coverage of the initial rule
and COV_{spec} is the coverage of the candidate specialization

The *quality* of the candidate specialization is calculated as a weighted average of these two values[5]. It remains open the question on how to set the weight of each of the factors. These weights represent a trade-off between generality and correctness of the learned rules. The bigger the weight on GainMAD the more specific the rules get. As a result the final theory will probably have too many rules. On the contrary, if we favor coverage we will get fewer rules but probably less accurate. This can be interesting in noisy domains where we don't want to overfit the noise by producing too specific rules. As this is highly domain dependent we let the user tune these weights[6]. $\mathcal{R}^2$ introduces a further degree of flexibility by allowing some limited variation on these weights, as we will explain bellow. The formula for calculating the quality of a specialization is as follows :

$$Q = GainMAD \times w_{gain} + (1 - LossCov) \times (1 - w_{gain}) \qquad \textbf{(3)}$$

$\mathcal{R}^2$ lets the user specify an interval for the value of w_{gain}. This interval states the minimum and maximum value for the weight of GainMAD in the quality of a rule. The actual value of the parameter is calculated as a function of the value of GainMAD. The idea is that if the value of GainMAD is very high then we put more weight on this factor, and vice-versa. In resume, given an interval $[w_1 \ldots w_2]$ the actual value of w_{gain} is calculated as :

$$w_{gain} = w_1 + (w_2 - w_1) \times GainMAD \qquad \textbf{(4)}$$

---

[5] As the loss of coverage is a negative effect we use the value (1-LossCOV) on this calculation.

[6] Being a weighted average it is in effect one weight as the quality formula will have the form :
Q = GainMAD × weight + (1-LossCOV) × (1-weight)

This formulation of a flexible weighted average is very similar to the one used in system Yails [16]. We intend to compare this simple approach to one using the Minimum Description Length (MDL) theory [14]. This methodology would give an elegant framework for integrating these properties with the issue of rule complexity. This could be important if we want to stress the advantages of ML-based systems in terms of comprehensibility.

We will now illustrate these ideas with the example we have been using. Imagine that after the step of building a model for an uncovered region we have as result the rule R_3 previously shown. Suppose that this rule covers 15 examples and has a MAD of 0.45. One possible specialization of this rule could be[7] :

$$R_{31} \equiv \text{IF } X_3 \leq 5 \wedge X_2 < 2 \text{ THEN } Y = -4.2 + 0.6 \times X_4 + 0.99 \times X_2$$

If this new rule covered only 12 of the previous examples but had a MAD of 0.24 then we would get :

$$GainMAD = \frac{0.45 - 0.24}{0.45} = 0.467 \quad \text{and} \quad LossCOV = \frac{15 - 12}{15} = 0.2$$

If the interval for the weight on GainMAD was [0.6..0.85] then w_{gain} would be given by $0.6 + (0.85\text{-}0.6) \times 0.467 = 0.717$, and finally the quality of this specialization would be:

$$Q = 0.467 \times 0.717 + (1\text{-}0.2) \times (1\text{-}0.717) = 0.561$$

$\mathcal{R}^2$ would now compare this value to the quality of the original rule and decide whether this specialization is better.

We still did not refer how the quality of the original rule is assessed[8]. We decided to use the difference 1-MAD has the quality of these original models. In this case rule R_3 would have a value of 0.55. This makes the specialization more attractive and the next step followed by $\mathcal{R}^2$ would be to try to specialize this new rule and so on until no better specialization of the current best rule exists.

3. EXPERIMENTS

In this section we present some experiments carried out with $\mathcal{R}^2$. At this stage of development our system is not yet ready in terms of computational efficiency for real-world high dimensionality domains so we have decided to test it in some artificial data sets.

We now present the data sets used in our experiments. We give a kind of specification used in the generation of the data sets :

- *Linear 1 (L1)* - 100 examples

  This data set has 1 real valued attribute with values randomly generated from the interval [-10..10]. The class for each case is obtained using the rules :

[7] Please note that when we specialize a rule we get a different set of examples so the model in the conclusion needs to be refined.

[8] Notice that it is impossible to use the same formulation for obvious reasons.

$$\text{IF } a_1 \geq 1 \text{ THEN } y = -5 - 0.67 \times a_1 + \varepsilon$$
$$\text{IF } a_1 < 1 \text{ THEN } y = 10 + 0.3 \times a_1 + \varepsilon$$
where ε is a random value in [0..2]

- *Linear 2 (L2)* - 100 examples

  This domain has 4 real valued attributes. The class is calculated as :

$$\text{IF } a_3 \geq 1 \text{ THEN } y = -5 - 0.67 \times a_1$$
$$\text{IF } a_3 < 1 \text{ THEN } y = 10 + 3 \times a_1$$
$$\text{OTHERWISE } y = 100 - 0.5 \times a_1 + 0.3 \times a_3 + a_2$$

- *Non-linear 1(NL1)*[9] - 100 examples

  The examples of this domain are described by 5 attributes. The first has equally probable values 1 and 2. The others are random real numbers. The classes are obtained using the rules :

$$\text{IF } a_1 = 1 \text{ THEN } y = 1 + 2 \times a_2 + a_3 - e^{(-2\times(a4+a5))}$$
$$\text{IF } a_1 = 2 \text{ THEN } y = 1 - 1.2 \times a_2 - 3.1 \times a_3 + e^{(-3\times(a4-a5))}$$

- *Non-linear 2 (NL2)* - 100 examples

  This domain uses 3 real valued attributes and the classes are obtained by :

$$\text{IF } a_1 \in]-10..5[\wedge a_2 \in]0.9..3[\text{ THEN } y = -5.3 + 0.3 \times a_1 \times a_2$$
$$\text{IF } a_1 \in]-10..5[\wedge a_3 \in]1..2[\text{ THEN } y = -5.3 + 0.3 \times a_1 \times a_2$$
$$\text{IF } a_2 \in]0.5..0.9[\wedge a_3 \in]2..3[\text{ THEN } y = 0.5 \times a_1^2 + a_3 / a_2$$
$$\text{OTHERWISE } y = 0.5 \times a_1 + 0.3 \times a_3^2 + a_1 \times a_2$$

We compared $\mathcal{R}^2$ with M5. Different parameter settings permit M5 to simulate several other algorithms (like CART, standard linear regression models, etc.). The variations of M5 that we have tried were :

- Default parameter values (M5).
- Standard multiple linear regression models (M5a).
- Regression trees (M5b) - simulating CART, i.e. trees with average values in the leaves (no linear regression performed).
- Instance based (M5c) - simulates an instance-based predictor, namely M5 uses a predictor similar to David Aha's IB1 [1].
- Full models, no smoothing (M5d) - with these settings M5 does not simplify the regression models at the leaves and it also does not perform smoothing [12] when classifying.

We performed a 10-fold cross validation test on each data set and collected averages on three measures of prediction errors :

- Mean Absolute Error - same as MAD (see equation 1).
- Mean Standard Error - MSE

$$MSE = \frac{\sum_{i \in Exs} (y_i' - y_i)^2}{\#\, Exs} \qquad \textbf{(5)}$$

[9] Adaptation of the LEXP artificial problem presented in [8].

- Normalized Mean Standard Error - NMSE

$$NMSE = \frac{\sum_{i \in Exs} (y_i' - y_i)^2}{\sum_{i \in Exs} (y_i - \bar{y})^2} \quad \textbf{(6)}$$

The results on the data sets presented before are shown in table 1. The first line represents the average error over the 10 tests and the second line the corresponding standard deviation.

	L1			**L2**			**NL1**			**NL2**		
	MAD	*MSE*	*NMSE*	*MAD*	*MSE*	*NMSE*	*MAD*	*MSE*	*NMSE*	*MAD*	*MSE*	*NMSE*
$\mathcal{R}^2$	**0.7**	**2.9**	**0.04**	**0.9**	26.1	0.82	0.9	1.4	0.66	21.1	2679.6	1.37
	0.5	*8.0*	*0.13*	*2.8*	*82.3*	*2.60*	*0.2*	*1.1*	*0.4*	*15.5*	*4389.6*	*0.80*
M5	0.9	3.0	**0.04**	1.5	**11.7**	**0.07**	**0.3**	0.5	**0.12**	18.0	1669.9	2.71
	0.42	*6.7*	*0.09*	*0.9*	*23.8*	*0.13*	*0.2*	*0.7*	*0.13*	*9.8*	*1795.9*	*3.32*
M5a	3.5	19.5	0.28	7.7	81.0	0.68	1.1	2.7	0.77	20.8	3360.6	**0.87**
	0.7	*7.8*	*0.09*	*1.4*	*21.5*	*0.30*	*0.6*	*3.9*	*0.28*	*17.8*	*4871.6*	*0.48*
M5b	0.8	3.2	0.04	2.0	17.2	0.09	0.7	1.6	0.46	21.9	2816.9	3.33
	0.5	*8.4*	*0.12*	*1.3*	*31.7*	*0.12*	*0.3*	*1.6*	*0.33*	*16.5*	*3620.2*	*6.10*
M5c	2.0	10.3	0.15	4.6	45.7	0.38	0.5	0.7	0.21	**15.4**	**1267.9**	2.72
	0.7	*6.3*	*0.09*	*1.4*	*22.4*	*0.21*	*0.2*	*0.9*	*0.23*	*11.5*	*1653.6*	*7.36*
M5d	**0.7**	**2.9**	**0.04**	1.0	24.5	0.16	**0.3**	**0.4**	0.15	16.6	2280.7	4.69
	0.4	*7.9*	*0.13*	*1.2*	*34.9*	*0.22*	*0.1*	*0.5*	*0.17*	*14.6*	*3406.7*	*11.95*

Table 1. Comparative results on the artificial data sets.

The results of $\mathcal{R}^2$ compare reasonably well to other systems on most of the data sets. We believe that if we add some strategy for eliminating terms[10] on linear regression models we could get some improvement. We hope that by incorporating other non-linear models we will get much better results on non-linear data. Nevertheless, this results indicate that our approach is valid although needing some more improvements before it can clearly win over these ML-based systems[11].

We observed that standard deviation varies quite a lot. This variability of the results could derive from small amount of training cases used in each run (90 examples). The same can be said about the test cases as our measures were based on 10 predictions. We confirmed that the standard deviation decreases significantly if we extend the size of the data. For instance, M5 on a extended L2 data set (1000 examples) got a value of 0.279 of MAD with a standard deviation of 0.062. The reason for generating so small data sets is the fact that $\mathcal{R}^2$ is currently implemented in Prolog and it has several problems in efficiency. In effect several parts of the system demand intensive numeric computations, namely matrix algebra. Prolog is clearly not the ideal programming language for these tasks. However, on the part of specialization it is much more suitable than other languages

[10] See section 6.6 of the book *Multivariate Analysis* [4] for an excellent overview of several methods.

[11] M5a is not an ML-based system, of course.

due to the symbolic character of the task. We will need to reach some compromise in order to be able to deal with real-world problems.

4. CONCLUSIONS AND FUTURE WORK

Although regression is an important problem in data analysis, it has not been dealt much by the ML community. In this paper we presented a system capable of learning regression rules. The system integrates the task of developing a regression model from data, with the technique of searching for logical conditions that enable a better fitting error by the model. Although regression trees also follow a similar strategy, $\mathcal{R}^2$ uses a more powerful descriptive language.

Piecewise regression models like the ones built by $\mathcal{R}^2$ have advantages of achieving better prediction accuracy when compared to one-model approaches. These advantages come at the cost of more specific models. $\mathcal{R}^2$ implements a flexible compromise between model generality and correctness.

Our system compares reasonably to other ML regression algorithms and even outperforms them on some data sets. However, due to the size of the data sets many differences are not statistically significant.

In future we plan to extend our comparisons and weigh both accuracy and comprehensibility. These comparisons should include systems from other non-symbolic fields. We believe that this will show the advantage of $\mathcal{R}^2$ over other sub-symbolic methods.

We also intend to explore techniques that enable our system to deal with large scale domains. We think that methods of sampling and iterative regression modeling will help to overcome these problems.

ACKNOWLEDGMENTS

I would like to thank Prof. Ross Quinlan for making available to us the system M5. I would also like to thank my supervisor Prof. Pavel Brazdil for the interesting discussions we had throughout the development of system $\mathcal{R}^2$.

References

[1]. Aha,D. Kibler,D. (1991): Instance-Based Learning Algorithms. In *Machine Learning, vol. 6 - 1*. Kluwer Academic Publishers.

[2]. Breiman, L. , Friedman,J.H., Olshen,R.A. & Stone,C.J. (1984): *Classification and Regression Trees*, Wadsworth Int. Group, Belmont, California, USA.

[3]. Clark, P., Niblett, T. : Induction in noisy domains, in *Proc. of the 2th European Working Session on Learning* , Bratko,I. and Lavrac,N. (eds.), Sigma Press, Wilmslow, 1987.

[4]. Dillon,W., Goldstein,M. (1984) : *Multivariate Analysis methods and applications*. John Wiley & Sons.

[5]. Friedman,J. (1991): Mutivariate Adaptative Regression Splines. In *Annals of Statistics* , 19:1.

[6]. Friedman,J. Stuetzle,W. (1981): Projection Pursuit Regression. In *J. American Statistics Association* 76.

[7]. Karalic, A.. (1991): The Bayesian Approach to Tree-Structured Regression. In *Proceedings of ITI-91* , Cavtat, Croatia, 1991.

[8]. Karalic, A..(1992): Employing Linear Regression in Regression Tree Leaves. In *Proceedings of ECAI-92* , Wiley & Sons, 1992.

[9]. McClelland,J. Rumelhart,D. (1988): *Explorations in Parallel Distributed Processing*. Cambridge, Ma. : MIT Press.

[10]. Michalski, R.S. , Mozetic, I. , Hong, J., Lavrac, N. : The multi-purpose incremental learning system AQ15 and its testing application to three medical domains, in *Proceedings of AAAI-86*, 1986.

[11]. Michie,D., Spiegelhalter,D.J., Taylor,C. (1994) : *Machine Learning, Neural and Statistical Classification*. Ellis Horwood series in Artificial Intelligence. Ellis Horwood.

[12]. Quinlan, J.R. (1992): Learning with Continuos Classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*. Singapore: World Scientific, 1992.

[13]. Quinlan, J.R. (1993): Combining Instance-Based and Model-Based Learning. In *Proceedings of 10th IML*, Utgoff,P. (ed.). Morgan Kaufmann Publishers.

[14]. Rissanen,J. (1983) : A universal prior for integers and estimation by minimum description length. In Annals of Statistics 11, 2.

[15]. Torgo,L. (1993) : Controlled Redundancy in Incremental Rule Learning. In *Proceedings of ECML-93*, Brazdil,P. (ed.). Lecture Notes in Artificial Intelligence - 667. Springer-Verlag.

[16]. Torgo,L. (1995) : Applying Propositional Learning to Time Series Prediction. In *ECML-95 workshop on Statistics, Machine Learning and Knowledge Discovery in Databases*. Kodratoff, Y. et al. (eds.).

[17]. Urbancic,T., Bratko,I. (1994): Reconstructing Human Skill with Machine Learning. In *Proceedings of the European Conference in Artificial Intelligence (ECAI-94)*, Cohn, A.G. (ed.). John Wiley & Sons.

[18]. Weiss,S.M., Indurkhya,N. (1993): Rule-Based Regression. In *Proceedings of IJCAI-93*, Bajesy,R. (ed.). Morgan Kaufmann Publishers.