Tree generation methods comparison in GAP problems with Low Quality Data

Alba Berzosa, José R. Villar, Javier Sedano, and Marco García-Tamargo

Abstract Data gathered from real world processes include several undesired effects, like the noise in the process, the bias of the sensors and the presence of hysteresis, among other undesirable uncertainty sources. Learning models using the so called Low Quality Data (LQD) is a difficult task which has been barely studied. In a previous study, a method for learning white box models in the presence of LQD that makes use of Multi Objective Simulated Annealing hybridized with genetic operators method for learning models was proposed. This research studies the role of the tree generation methods when learning LQD. The results of this study show the relevance of the role of tree generation methods in the obtained results.

1 Introduction

With the scarce energy sources and the worsening environmental pollution, how to use the existing energy is becoming a very important challenge in various fields of modern engineering [10, 6, 19]. For example, notorious efforts have been made within the area of lighting control systems included in building automation in order to improve the energy efficiency. The aim of lighting control systems is to control

Alba Berzosa
ITCL, Lopez Bravo 70, Pol. Ind. Villalonquéjar 09001 Burgos (SPAIN) e-mail: alba.berzosa@itcl.es

José R. Villar
Universidad de Oviedo, Campus de Viesques s/n 33204 Gijón (SPAIN) e-mail: villarjose@uniovi.es

Javier Sedano
ITCL, Lopez Bravo 70, Pol. Ind. Villalonquéjar 09001 Burgos (SPAIN) e-mail: javier.sedano@itcl.es

Marco García-Tamargo
Universidad de Oviedo, Campus de Viesques s/n 33204 Gijón (SPAIN) e-mail: marco@uniovi.es

the electrical power consumption for the ballast in the installation so the luminance complies with the regulations.

In [20, 21, 22] a lighting control system was considered to show the relevance of the uncertainty for an efficient energy use. The typical control loop includes a light sensor, the light ballasts and a light controller. The sensors measure the amount of light in a room, but they have some drawbacks: they operate with hysteresis and saturation [6] and their measurements depend on the light sensor unit. In the studied literature, when obtaining models for simulation, only crisp values are regarded as the measurements from light sensors. Obviously, the inputs and outputs of the light sensor models obtained are also crisp variables. But several studies have presented the decrease in the performance of crisp algorithms as data uncertainty increases [8].

It is worth noting that all the sensors and industrial instrumentation can be regarded as Low Quality Data (hereinafter LQD). In our opinion, one of the most successful researches in soft computing [2, 3, 4] dealing with LQD is detailed in [5, 12]. In these works the mathematical basis for designing vague data awareness genetic fuzzy systems -both classifiers and models- is shown. The LQD are assumed as fuzzy data, where each $\alpha-$cut represents an interval value for each data. It should be noticed that the fitness functions to train classifiers and models are also fuzzy valued functions when faced with LQD. Hence the learning algorithms should be adapted to such fitness functions [15]. The ideas and principles previously shown have been used in several applications with LQD, both with realistic and real world data sets [13, 14, 18].

An approach for learning white box models when LQD is available is presented in [22], where the variables are assumed with an unknown uncertainty value modelled as a fuzzy number. A Multi Objective Simulated Annealing algorithm hybridized with genetic operators is proposed (hereinafter, SAP), and a random tree generation algorithm to create the individuals is carried out in the evolutionary algorithm.

Nevertheless, the role that the tree generation algorithms play in the learning of SAP when LQD is given has not been studied yet. This research compares different tree generation algorithms to study the performance evolution. The remainder of this manuscript is as follows. Firstly, a light description of the SAP approach for learning white box models with LQD is included. Then, the different tree generation methods employed in this comparison are detailed. The experiments and the results are discussed next, while in the last section the conclusions are drawn and the future work is proposed.

2 White Box Models SAP Learning With LQD

Soft Computing includes the set of techniques that allow learning models using hte knowledge in the data [1, 4, 16, 23]. There are several uncertainty sources in data gathered from real processes [7]. In this study, we adopt the study of data which has been gathered as crisp data but that are highly imprecise, i.e., the data gathered from light sensors [6, 21, 20]. In [22] a SAP approach for learning white box models from this kind of data is presented.

In that study, the representation of the vagueness in a GP model is represented by two constants C^- and C^+ assigned to each imprecise variable which evolves in the learning process. These constants represent the limits of a triangular membership function for an α-cut$=0$ which is associated with each imprecise variable. Let us suppose the training data set being the set $\{d_i^j\}$, with $i = \{0,\dots,D\}$ for each of the D variables X_i and $j = \{1,\dots,N\}$ and N the number of examples in the data set. Then, whenever an imprecise variable X_i is evaluated for the example j, a fuzzy number with a triangular fuzzy membership defined through the three following values $[d_i^j - C^-, d_i^j, d_i^j + C^+]$ is returned. If symmetrical membership functions are considered, only one constant per imprecise variable is needed. As in classical fuzzy logic literature, crisp values from constants or from crisp variables are extended to fuzzy singletons, so only operations with fuzzy numbers are required. In order to reduce the computational cost, the solution presented in [18] is used, and evaluations are calculated only for certain predefined α-cuts.

An individual in this study is a compound of the equation representation, the constants vector and the specification of the uncertainty, which is provided with the number of constants used to represent the uncertainty and the list of indexes of the input variables in the dataset that are supposed to manage LQD. As in Genetic Programming hybridized with Genetic Algorithms (hereinafter, GAP) models, the equation representation consists of a nodes tree, each internal node corresponds with a valid operator, and the leaf nodes correspond with a variable or a constant index. The number of constants is predefined, so the constant vector in all individuals has the same size. The first group of constants in the constant vector is assigned to the uncertainty management. The generation of a nodes tree is the well-known random strategy given by the GROW method [9].

Evolving GAP individuals introduce four genetic operators, two come from GP evolution (the GP crossover and mutation) and two come from GA (GA crossover and mutation). The GP operators introduce variability in the structure of the model, in other words, the equation itself. The GA operators modify the vector of constants. In all the cases, there is a predefined probability of carrying out each of these genetic operators. In each run, the type of operation to carry out is chosen, that is only GP or GA operations can take place, but never both in the same run. The fitness of an individual is calculated as the mean squared error, which in fact is a fuzzy number. To reduce the width of the intervals and to obtain models that include the desired output crisp data two more objectives have also been considered, so multi-objective techniques are needed. The evolutionary algorithm is the multi-objective simulated annealing proposed in [14, 18].

3 Tree-Generation Algorithms

Tree-creation plays an important role in GP algorithms since a good random tree-creation is needed to create the number of trees that will compose the initial population and the subtrees used in subtree mutation. Besides, as stated in [11], tree creation is also related with tree bloat, that is, the tendency of GP trees to grow during the evolutionary process [17] which causes the slowdown of the evolutionary process by making individuals more resistant to change. This section discusses the role of the tree generation methods when learning LQD. The above mentioned SAP approach is used for learning models with such kind of data, and two different techniques for tree generation are compared: the GROW and the PTC1 methods.

In [9] the so-called GROW GP tree-generation algorithm is described (see algorithm 1). It has been widely used since its formulation despite having several weaknesses. Although originally not designed to control the depth and the size of the tree, it can be easily extended to do so. To generate a tree, the algorithm chooses a node type with equal probability. The choice of a node includes the terminal and non-terminal nodes. Once a node has been chosen, and attending to the arity of the node, the algorithm moves to each of its operands and is executed recursively. This process continues either until all the leaf nodes are terminal nodes, or either the depth limit or the tree size limit is reached. In these two latter cases, the tree is filled with terminal nodes.

Algorithm 1 Grow Algorithm

```
Initialize $P_0$
Evaluate $P_0$
while not stop criterion do
    parents ← selectParents ($P_t$)
    offspring ← variation (parents)
    evaluate offspring (and if necessary $P_t$ )
    select the new population $P_{t+1}$ from $P_t$ and offspring
    $t = t + 1$
end while
```

On the other hand, [11] offers an alternative tree-creation algorithm, the Probabilistic Tree Creation 1 (PTC1). This algorithm gives the user control over the expected tree size E_{tree}, a method parameter. Instead of attempting to generate completely uniformly distributed tree structures, PTC1 allows user-defined probabilities of appearance of functions within the tree plus a desired maximum depth D, providing in addition, very low computational complexity. However, PTC1 does not provide any control over the variance in tree size generated, which limits its usefulness.

The PTC1, as described in [11] is as follows (see algorithm 2). The set of functions F is divided into two disjoint subsets: terminals nodes set T, each one with probability q_t, and non-terminals nodes N, each one with probability q_t. The recursive method chooses between terminal and non-terminal nodes type for the current node with probability p (see Eq. 1), where b_n is the arity of non-terminal n. When a terminal node type is chosen, variable or constant is decided according to q_t. In case of non-terminal nodes, the node type will be selected according to q_n. Each of its operators is chosen recursively. Both $\{q_t\}$ and $\{q_n\}$ are given by the user.

$$p = \frac{1 - \frac{1}{E_{tree}}}{\sum_{n \in N} q_n b_n} \tag{1}$$

Algorithm 2 PTC1 Algorithm

```
PTC1 (depth $d$, probability $p$, maximum depth $D$)
*Returns:* a tree of depht $\leq D - d$
**if** $d = D$ **then**
    return a terminal from $T$ (by $q_t$ probabilities)
**else if** probability $p$ **then**
    Choose a non-terminal $n$ from $N$ (by $q_n$ probabilities)
    **for** each argument $a$ of $n$ **do**
        Fill $a$ with PTC1 ($d+1$, $p$, $D$)
        Return $n$ with filled arguments
    **end for**
**else**
    Return a terminal from $T$ (by $q_t$ probabilities)
**end if**
```

4 Experimentation and results

In the experimentation stage, the performance of GROW and PTC1 algorithms when learning models with LQD is compared when both regression and time series problems are faced. Several different synthetic data set are generated, obviously, all of them are LQD. In order to obtain such LQD, a two step procedure has been carried out: firstly, the crisp data sets have been generated; secondly, the uncertainty have been introduced to the data.

Up to four different problems are proposed, three of them correspond with regression problems and one problem belongs to time series problems. One of the differences among the two types of the problem is that in regression problems the model can not be a function of the output variable, while in the time series problems this variable can be included in the equations. Obviously, should the output variable be included in an equation then it is mandatory that it has to be affected with one delay operator at least. Nevertheless, the main difference between regression problems and time series problems relies on that the time series problems models should be evaluated recursively, that is, the output at time t should be calculated using the previously calculated values of the output variable.

The regression problems are defined through f_1, f_2 and f_3 in Table 1, while f_4 is the equation of the output variable and represents a time series problem. In all the cases, four input variables are considered ($\{x_0, x_1, x_2, x_3\}$ in Table 1). Recall that the time, represented as t, is not an input variable of the data sets but is used to calculate the values of the examples $\{x_i, \forall i\}$. Therefore, all example in each data set includes the four above-mentioned variables and the output variable calculated with the corresponding formulae f_i.

The second step for generating the LQD data sets is the aggregation of uncertainty to the data. Uncertainty is generated as follows: for each variable in an example, a random values in the range $[-1e^{-4}, 2e^{-4}]$ is added. All the involved variables in a data set are affected, including the output variable. In all the problems, the variables are parameterized as imprecise.

Table 1 Formulae for the data sets generation.

$f_1 = x_1 + x_0 * (x_2 - 0.5)$	$t = \{1, \ldots, 100\}$
$f_2 = 2 * x_1 * x_2$	$x_0 = abs(\cos(t))$
$f_3 = \cos(x_0) * (x_2 - x_1)$	$x_1 = abs(\sin(t))$
$f_4 = 2 * x_2 * delay(f_4)$	$x_2 = abs(\cos(t) * \sin(t))$
	$x_3 =$ random in the range $[0, 1]$

The parameters to be used are presented in Table 2. The number of iterations is relatively small for this kind of problems in order to determine if the algorithms are able to find good models in the whole search space when the temperature in the Simulated Annealing is still high. It is worth noting that the total number of generations is fixed to 1000 and that each experiment is artificially stopped at a certain iteration. This is due to the use of simulated annealing and its temperature evolution parameter: if a reduced number of iterations is fixed then the whole temperature variation will be covered, thus the variability of models in the SA will be penalized. The aim of this is to evaluate how the algorithms improves when the temperature is still high.

Results from experiments are shown in Table 3 and Table 4. In the former, the results of the individual with lower mean squared error (MSE) is shown. In the latter, the MSE of the individual closer to the point (0,0,0) is shown. From results it can easily be extracted that both methods are valid for evolving SAP models. Even though the initial individual is initialized far enough from the suboptimal, the ability of the generation methods to find models in the whole search space seems to be guaranteed. The main drawback of GROW is that it needs more iterations, while the PTC1 will probably have a somewhat lower error level. This is mainly due to the knowledge of the domain expressed as the operators probability. But besides, this benefit is twofold: if the probability distribution is not suitable the obtained error of the method will be higher.

Table 2 Parameters used in the experimentation stage. When LQD is assumed, all the variables are set as imprecise variables. In all the cases symmetrical triangular membership functions are used, so only one constant in the GA constants vector is needed per variable. Whenever time series learning is carried out the models are evaluated recursively.

Parameter	Value	Parameter	Value
$\alpha-$cut	0.95	population size	50
Constants range	[-10,10]	GP mutation prob.	0.25
$C^- == C^+$ range	[0, 0.01] %	GP crossover prob.	1
SA Δ value	0.1	GA mutation prob.	0.5
SA initial temperature	1	GP crossover prob.	1
SA final temperature	0	stop generation	{50, 100, 300}
PTC1 Constant prob.	0.3	generations	1000
PTC1 Operators probability	+ 0.145 - 0.145 * 0.145 / 0.145 max 0.08 min 0.08 delay 0.1 sin 0.08 cos 0.08		
Fitness functions	MSE + number of covered examples + mean output width	Maximum depth	5
N^o of GA constants	7	Maximum size	5

Table 3 Experimentation results: comparison of lower MSE values found. Results shown should be multiplied by 10^{-3}. DS and Itera stand for data set and number of iterations carried out, respectively.

DS	Itera	Initial MSE		Lower MSE found	
		GROW	PTC1	GROW	PTC1
f_1	50	[3.348, 3.353]	[2.097, 2.123]	[0.2763, 0.2940]	[0.2743, 0.2760]
	100	[4.175, 4.186]	[115.2, 119.3]	[0.2635, 0.2875]	[0.2660, 0.2821]
	300	[9.351, 9.781]	[3.725, 3.840]	[0.2707, 0.2779]	[0.2051, 0.2412]
f_2	50	[1.204, 1.246]	[2.409, 2.541]	[0.226, 0.230]	[0.357, 0.360]
	100	[1.506, 1.521]	[13.379, 13.699]	[0.227, 0.232]	[0.165, 0.188]
	300	[5741.45, 137350]	[1952.27, 696342]	[0.210, 0.246]	[0.210, 0.210]
f_3	50	[10.432, 10.642]	[3.173, 3.234]	[0.0085, 0.0132]	[0.0064, 0.0167]
	100	[10.431, 10.641]	[4.570, 4.600]	[0.0085, 0.0132]	[0.0104, 0.0109]
	300	[16.951, 20.167]	[8.669, 8.825]	[0.0093, 0.0122]	[0.0065, 0.0165]
f_4	50	[8.508, 8.551]	[6.068, 6.307]	[0.779, 0.857]	[0.668, 0.681]
	100	[10.073, 10.332]	[2024.89, 1439640]	[0.811, 0.823]	[0.807, 0.854]
	300	[4.849, 4.870]	[7.483, 7.627]	[0.669, 0.714]	[0.644, 0.657]

5 Conclusions and future work

The relevance of the tree generation algorithms in SAP learning problems with LQD has been studied. A SAP approach that makes use of fuzzy fitness functions is proposed. The Multi-Objective Simulated Annealing algorithm is used as the SAP learning evolutionary strategy. Consequently, the relevance of the tree generation methods is focused on the first generations due to the temperature grading and the need to search in the whole variables space. Two algorithms have been compared, the GROW and the PTC1, and both of them behave in a similar way. The PTC1 has slightly lower error values than the GROW but needs a-priori knowledge in setting the operators probability distribution. Future work includes the study of statistics

Table 4 Experimentation results: comparison of MSE values when all the fitness values have the same relevance and the nearest to the origin individual found is chosen. Results shown should be multiplied by 10^{-3}. DS and Itera stands for data set and number of iterations carried out, respectively.

DS	Itera	Initial MSE		Lower MSE found	
		GROW	PTC1	GROW	PTC1
f_1	50	[3.348, 3.353]	[2.097, 2.123]	[1.578, 1.595]	[0.301, 0.302]
	100	[4.175, 4.186]	[115.2, 119.3]	[0.264, 0.287]	[0.266, 0.282]
	300	[9.351, 9.781]	[3.725, 3.840]	[0.301, 0.302]	[0.272,0.278]
f_2	50	[1.204, 1.246]	[2.409, 2.541]	[1.804, 1.820]	[0.693, 0.751]
	100	[1.506, 1.521]	[13.379, 13.699]	[0.227, 0.232]	[0.535, 0.591]
	300	[5741.45, 137350]	[1952.27, 696342]	[1.570, 1.585]	[0.924, 1.015]
f_3	50	[10.432, 10.642]	[3.173, 3.234]	[1.785, 1.786]	[2.482, 2.484]
	100	[10.431, 10.641]	[4.570, 4.600]	[1.784, 1.786]	[1.678, 1.679]
	300	[16.951, 20.167]	[8.669, 8.825]	[1.126, 1.359]	[0.0065, 0.0165]
f_4	50	[8.508, 8.551]	[6.068, 6.307]	[1.313, 1.346]	[1.853, 1.894]
	100	[10.073, 10.332]	[2024.89, 1439640]	[4.462, 4.472]	[1.776, 1.781]
	300	[4.849, 4.870]	[7.483, 7.627]	[1.338, 1.338]	[1.253, 1.258]

with fuzzy fitness functions, so the typical cross validations test could be carried out. Also, the study of the evolution of the diversity that each of the tree generation algorithms induces through the temperature evolution in the MOSA learning process, should be considered.

Acknowledgements This research has been funded by the Spanish Ministry of Science and Innovation, under project TIN2008-06681-C06-04, the Spanish Ministry of Science and Innovation [PID 560300-2009-11] and by the ITCL project CONSOCO.

References

1. Alcalá-Fdez, J., Sánchez, L., Garca, S., del Jesús, M.J., Ventura, S., Garrell i Guiu, J.M., Otero, J., Romero, C., Bacardit, J., Rivas, V.M., Fernndez, J.C., Herrera, F.: KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Computing **13(3)**, 307–318 (2009).
2. Banerjee, T.P., Das, S., Roychoudhury, J., Abraham, A.: Implementation of a New Hybrid Methodology for Fault Signal Classification Using Short -Time Fourier Transform and Support Vector Machines. oft Computing Models in Ind. and Environ. Appl., 5th Int. Workshop (SOCO 2010), Advances in Soft Computing **73**, 219–225, Springer, Berlin (2010).
3. Corchado, E., Arroyo, A., Tricio, V.: Soft computing models to Identify typical meteorological days. Logic J. of thel IGPL (2010) doi:10.1093/jigpal/jzq035.
4. Corchado, E., Herrero, A.: Neural visualization of network traffic data for intrusion detection. Applied Soft Computing (2010) doi:10.1016/j.asoc.2010.07.002.
5. Couso, I., Sánchez, L.: Higher order models for fuzzy random variables. Fuzzy Sets Syst. **159**, 237–258, (2008).
6. De Keyser, R., Ionescu, C.: Modelling and simulation of a lighting control system. Simul. Model. Pract. and Theory **18(2)**, 165–176 (2010).

7. Ferson, S., Kreinovich, V., Hajagos, J., Oberkampf, W., Ginzburg, L.: Experimental uncertainty estimation and statistics for data having interval uncertainty. RAMAS Technical Report SAND2007-0939 (2007) Available via http://www.ramas.com/intstats.pdf.
8. Folleco, A., Khoshgoftaar, T.M., Van Hulse, J., Napolitano, A.: Identifying learners robust to low quality data. Informatica (Slovenia) **33(3)**, 245–259 (2009).
9. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992).
10. Li, D.H.W., Cheung, K.L., Wong, S.L., Lam, T.N.T.: An analysis of energy-efficient light fittings and lighting controls. Appl. Energy **87(2)**, 558–567 (2010).
11. Luke, S.: Two fast tree-creation algorithms for genetic programming. IEEE Trans. on Evol. Comput. **4(3)**, 274–283 (2000).
12. Sánchez, L., Couso, I.: Advocating the use of imprecisely observed data in genetic fuzzy systems. IEEE Trans. on Fuzzy Systems **15(4)**, 551–562 (2007).
13. Sánchez, L., Otero, J.: Learning fuzzy linguistic models from low quality data by genetic algorithms. IEEE Int. Conf. on Fuzzy Systems FUZZ-IEEE 2007, 1–6 (2007).
14. Sánchez, L., Suárez, M.R., Villar, J. R., Couso, I.: Mutual information-based feature selection and partition design in fuzzy rule-based classifiers from vague data. Int. J. Approx. Reasoning **49**, 607–622 (2008).
15. Sánchez, L., Couso, I., Casillas, J.: Genetic learning of fuzzy rules based on low quality data. Fuzzy Sets and Systems **160(17)** 2524–2552 (2009).
16. Sedano, J., Curiel, L., Corchado, E., de la Cal, E., Villar, J.R.: A soft computing method for detecting lifetime building thermal insulation failures. Integr. Computer-Aided Eng. **17(2)**, 103–115 (2010).
17. Soule, T., Foster, J.A., Dickinson, J.: Code growth in genetic programming. In Proc. of the First Annual Conf. on Genetic Programming, GECCO 96, 215–223, MIT Press, Cambridge (1996).
18. Villar, J.R., Otero, A., Otero, J., Sánchez, L.: Taximeter verification with gps and soft computing techniques. Soft Comput. **14**, 405–418 (2009).
19. Villar, J.R., de la Cal, E., Sedano, J.: A fuzzy logic based efficient energy saving approach for domestic heating systems. Integr. Computer-Aided Eng. **16**, 151–163 (2009).
20. Villar, J.R., de la Cal, E., Sedano, J., García-Tamargo, M.: Analysing the low quality of the data in lighting control systems. Hybrid Artificial Intelligent Systems, Lecture Notes in Computer Science **6076**, 421–428, Springer, Berlin (2010).
21. Villar, J.R., de la Cal, E., Sedano, J., García, M.: Evaluating the low quality measurements in lighting control systems. Soft Computing Models in Ind. and Environ. Appl., 5th Int. Workshop (SOCO 2010), Advances in Soft Computing **73**, 119–126, Springer, Berlin (2010).
22. Villar, J.R., Berzosa, A., de la Cal, E., Sedano, J., García-Tamargo, M.: Multi-objecve simulated annealing in genetic algorithm and programming learning with low quality data. Submitted to Neural Comput. (2010).
23. Yu, W.-D., Liu Y.-C.: Hybridization of CBR and numeric soft computing techniques for mining of scarce construction databases. Autom. in Constr. **15(1)**, 33–46 (2006).