18th European Symposium on Computer Aided Process Engineering – ESCAPE 18
Bertrand Braunschweig and Xavier Joulia (Editors)

Inductive Data Mining: Automatic Generation of Decision Trees from Data for QSAR Modelling and Process Historical Data Analysis

Chao Y Ma, Frances V Buontempo and Xue Z Wang*

Institute of Particle Science and Engineering, School of Process, Environmental and Materials Engineering, University of Leeds, Leeds LS2 9JT, UK

Abstract

A new inductive data mining method for automatic generation of decision trees from data (GPTree) is presented. Compared with other decision tree induction techniques that are based upon recursive partitioning employing greedy searches to choose the best splitting attribute and value at each node therefore will necessarily miss regions of the search space, GPTree can overcome the problem. In addition, the approach is extended to a new method (YAdapt) that models the original continuous endpoint by adaptively finding suitable ranges to describe the endpoints during the tree induction process, removing the need for discretization prior to tree induction and allowing the ordinal nature of the endpoint to be taken into account in the models built. A strategy for further improving the predictive performance for previously unseen data is investigated that uses multiple decisions trees, i.e. a decision forest, and a majority voting strategy to give a prediction (GPForest). The methods were applied to QSAR (quantitative structure – activity relationships) modeling for eco-toxicity prediction of chemicals and to the analysis of a historical database for a wastewater treatment plant.

Keywords: inductive data mining, decision trees, genetic programming, QSAR, process historical data analysis

1. Introduction

Collecting data and storing it in databases has now become a routine operation in industry. The data clearly represents a useful 'mine' from which valuable information and knowledge could be extracted. Discovering information and knowledge from data is particularly useful when first-principle models and knowledge are not available or not applicable due to uncertainties and noise in real world applications. Knowledge extracted from data has statistical basis and the advantage of being objective compared to the knowledge and experience of human experts. One of the most attractive knowledge discovery techniques is inductive data mining (IDM) which refers to a class of techniques that can generate transparent

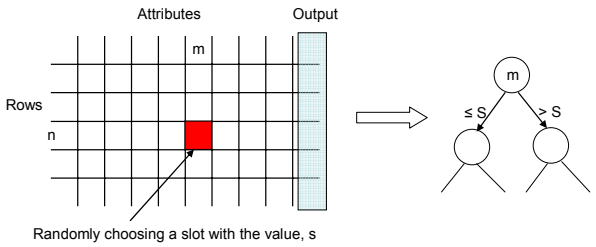


Fig. 1 GPTree generates binary decision trees from the training data by initially growing trees with randomly selected attribute and value pairs from randomly selected rows in the training data to form each splitting node.

and causal models, give both qualitative and quantitative predictions and can incorporate human experts' knowledge, giving advantages over other techniques employed in data mining. When properly trained, IDM models could match the accuracy of more opaque methods such as neural networks. IDM techniques proposed in literature are mainly based upon recursive partitioning employing greedy searches to choose the best splitting attribute and value at each node therefore will necessarily miss regions of the search space. In this paper a genetic programming based approach for decision tree generation is introduced which can overcome the limitations of greedy search based methods. Case studies will be presented in applying the approach for building QSAR (quantitative structure activity relationships) for toxicity prediction of chemicals, and for process historical data analysis for a wastewater treatment plant.

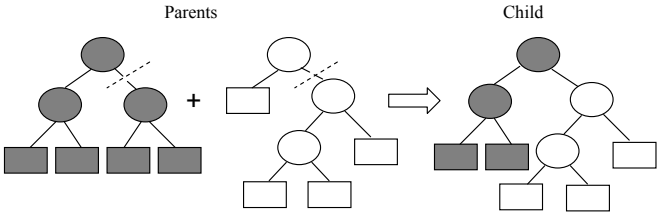


Fig. 2 Two parent trees forming a new child by splitting at a random place, indicated by the dotted lines, and crossing-over to generate a new individual that contains part of the solution encoded in each parent.

2. Genetic Programming for Induction of Decision Trees (GPTree)

2.1 The Algorithm

GPTree generates trees from data that is arranged in a tabular form with the columns representing the attributes (i.e. variables), and the rows being the data cases. The data is firstly divided into training and test sets. GPTree then generates binary decision trees from the training data by initially growing trees with randomly selected attribute and value pairs from randomly selected rows in the training data to form each splitting node (Fig. 1). For example randomly picking attribute m with corresponding value s for the randomly selected training row n would form the decision node

$$\text{If attribute } m \leq \text{value } s \qquad (1)$$

Any training data for which this is true is partitioned to the left child node, while the remaining data is partitioned to the right child node. If less than 5% of the training data will be partitioned to one child node, a new row and

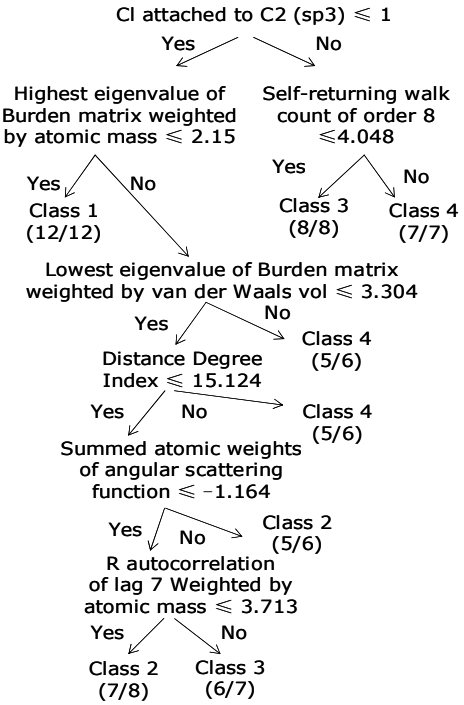


Fig. 3 Decision tree produced by generic programming GPTree for a bacteria data set in generation 37, which has higher accuracies for both training and test data in comparison with See5 generated tree. 7/8 means 7 were correctly classified, and 1 misclassified

attribute is chosen at random. This percentage is user configurable. The child nodes are grown as follows. When less than 10% (or double the pre-specified minimum percentage coverage required at a node) of the training data are partitioned to a branch of the tree, so that any further splits will cover less than 10% of the data, or all the data at that branch are pure (in the same class), a terminal or leaf node is formed. This predicts the class of the data partitioned there as the majority class of the training partitioned to that node. This process continues until all nodes have child nodes or are themselves leaf nodes.

Once the first generation is fully populated, new trees are grown by crossover, splitting the selected parents at a random node and recombining the parts to form new trees (Fig. 2). In order to select the parent trees that will take part in crossover, tournament selection is employed. The number of trees taking part in the tournament is configurable. The fittest tree from the tournament forms the first parent. This process is then repeated to find a second parent. The fitness function uses the accuracy of the trees in the competition since it is enforced that a node must contain a certain number of rows during the tree growth process:

$$Fitness(Tree) = \sum_{i=1}^{i=n} \frac{(Rows\ at\ node\ i\ with\ Class_{m,i})}{Rows} \tag{2}$$

where n is the leaf nodes and $Class_{m,i}$ is the majority class at node i. If the number of leaf nodes was included in the fitness function, the population tended to converge to a relatively small set of trees, decreasing the parts of the search space explored, thereby leading to a slower overall increase in accuracy and often seeming to get stuck in regions containing less accurate trees of the same size as those produced without using the leaf node count in the fitness function. The tree taking part in the tournament maximizing the equation (2) is selected as a parent.

Thirdly, the child trees may possibly be mutated before being added to the next generation. The pre-defined numbers of trees are mutated, with random choice of mutation operators.

(i) Change of split value (i.e. choosing a different row's value for the current attribute).
(ii) Choose a new attribute whilst keeping the same row.
(iii) Choose a new attribute and new row.
(iv) Re-grow part of the tree from any randomly selected node (apart from leaf nodes).

If either crossover or mutation gives rise to a node previously classed as a leaf node which is no longer pure or can now usefully be split further, that part of the tree is re-grown. If a node becomes a leaf node during either operation, its previous children will not be copied to the new tree.

Steps (ii) and (iii) are repeated until the required number of trees has been grown for the new generation, and generations are grown up to the requested number.

2.2 GPTree for QSAR Model Development

Quantitative structure – activity relationships (QSAR) correlates molecular structural, biological and physicochemical descriptors to toxicity end-points have been considered as the most promising technique for minimising toxicity assay test using animals. Various techniques have been studied in literature such as, expert systems, neural networks and linear regression and there are also commercial tools available such as TOPKAT, DEREK, Oncologic, and MultiCase. One of the challenges is the lack of knowledge on what molecular descriptors are important to the toxicity prediction and what are not so should not be included in a QSAR model. Inductive data mining based decision tree construction clearly has the potential to address this issue since during construction of a decision tree, the less important variables are excluded from the tree

model. Fig. 3 shows a decision tree built using GPTree for predicting toxicity using data obtained from literature(Zhao, et al., 1998). The data comprises Vibrio fischeri data for 75 compounds. The endpoint is the effect on the bioluminescence of the bacterium Vibrio fischeri measured over 15 min in a single laboratory using a standard protocol. This is scaled as log(1/EC50), ranging from 0.90 to 6.32. Values for the octanol/water partition coefficient, log P, and McGowan characteristic volume, Vx, were taken from the original paper and 1093 further topological, fragment and physical property descriptors with non-zero variation were calculated by DRAGON (a software system that has been widely used for calculating molecular descriptors), after optimisation using HyperChem (Hypercube Inc., Waterloo, Canada). The compounds are diverse mixture of chlorobenzenes, nitrobenzenes, anilines and phenols. Fig. 3 shows the decision tree built by GPTree. The toxicity endpoint log(1/EC50) was partitioned into four equal frequency groups, ≤3.68, ≤4.05, ≤4.50 and > 4.50. In Fig. 3, only a few variables were selected by the decision tree out of all the over a thousand descriptors. A second data set from literature which was tested using algae(Cronin, et al., 2004) was also studied and details can be found in (Buontempo, et al., 2005). See5 (Quinlan, 1993,Quinlan, 1996), probably the most widely used inductive data mining algorithm was also applied to the same two data sets. It was found that for both data sets, GPTree always performed better than See5 in terms of prediction accuracies for the test data.

3. Adaptive Discretization of Real-valued Output

GPTree and other inductive data mining techniques require that data with a continuous endpoint be transformed prior to tree induction, so that the real-valued endpoint is split into classes. Taking advantage of the inherent properties of genetic programming, GPTree has been further developed to a new algorithm, YAdapt, which can adaptively partition the output values to classes. YAdapt uses a different fitness function and introduces new mutation operators. The new fitness function uses the sum of squared differences in rank (SSDR), based on Spearman's rank correlation coefficient in order to utilise the ordinal nature of the data and to compensate for the differences in accuracy between different numbers of classes. SSDR is calculated as follows:

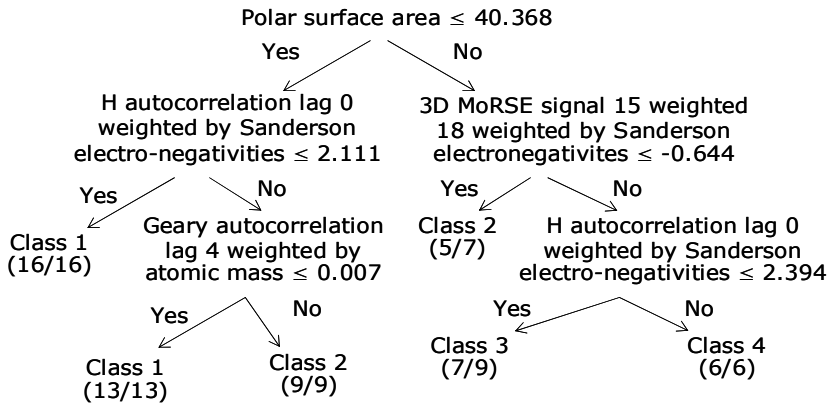
i. Prior to tree induction the data is sorted on the endpoint, from least to greatest.
ii. For each y range, find the middle row. For the sorted data, this is midway between the last and first rows covered by that range. This is used as the range's rank, and as the actual rank for data with an endpoint in that range.
iii. At a leaf node, calculate the sum of the squared differences, d^2, between the ranks of the range each data item covered actually belongs to and the majority range's rank.
iv. Calculate the sum of each leaf node's sum squared differences in rank to find a measure of fitness for the tree. The lower this value, the better the performance of the tree. An SSDR of zero means the tree is 100% accurate.

The SSDR was used in the fitness function instead of the accuracy:

$$\text{Fitness(Tree)} = \sum_{i=1}^{i=n} \text{(SSDR), for all n training rows}$$

$$= \sum_{i=1}^{i=n} ((\text{least}_{act,i}+(\text{greatest}_{act,i}-\text{least}_{act,i})/2) - (\text{least}_{pred,I}+(\text{greatest}_{pred,i} - \text{least}_{pred,i})/2)^2) \quad (5)$$

where the i^{th} training data's endpoint belongs to the range ($\text{least}_{act,i}$, $\text{greatest}_{act,i}$) and the leaf node to which it is designated covers training rows with majority endpoint range ($\text{least}_{pred,i}$, $\text{greatest}_{pred,i}$). Details of YAdapt can be found in Wang et al. (Wang, et al., 2006). Fig. 4 shows a decision tree that was generated for the same bacteria data as described above by adaptively adjusting the splitting of the end point.



Class 1:	Log(1/EC50) ≤ 4.08	29 cases
Class 2:	4.08 < Log(1/EC50) ≤ 4.51	16 cases
Class 3:	4.51 < Log(1/EC50) ≤ 4.99	9 cases
Class 4:	otherwise	6 cases

Fig. 4 A decision tree generated in generation 46 for the bacteria data, found four classes from the continuous endpoint adaptively during induction.

4. Decision Forest - GPForest

Much research on decision trees has been devoted to improving the prediction accuracy in particular for unseen data that was not used in building the tree models. One of the most promising methods is decision forest that uses ensembles of trees. Decision forest combines the results of multiple distinct but comparable decision tree models to reach a consensus prediction. The idea assumes that a single decision tree could not fully represent the relationships between input variables and the output variable or a single tree model is optimum for a specific region of the solution space, while a decision forest captures relationships of different regions using different tree models. The genetic programming method described here, the GPTree, has great advantages in decision forest application because it is designed to generate multiple trees from generation to generation. In addition, by slightly varying the parameter values, more groups of trees can be produced. The GPForest method has the following steps. Firstly, GPTree is run for a given number of times (called experiments), each time with slight variations in parameter values and a given number of generations. Then trees were selected from all the generated trees in all experiments based on some defined criteria such as accuracy for test data, complexity (details are not listed here due to space limitation). In applying the decision forest for prediction, each tree in the decision forest gives an output. The final prediction is determined based on a majority voting strategy. The GPForest method has been applied to the analysis of data collected from a wastewater treatment plant. A total of 527 sets of data were collected which correspond to 527 days of operation. Each dataset has 38 variables, of which 29 correspond to measurements taken at different points of the plant, remaining 9 are calculated performance measures for the plant. In the study, decision forests were developed for four variables: suspended solids in effluent (SS-S), biological oxygen demand in effluent DBO-S, conductivity in effluent COND-S, and chemical oxygen demand in effluent DQO-S. The numbers of decisions trees selected into the four decision forests for predicting SS-S, DBO-S,

DQO-S and COND-S are 24, 25, 23 and 24. Fig.5 shows one of the decision trees selected into the decision forest for predicting SS-S. It was found that the decision forest always gave better prediction accuracy for test data, than GPTree and See5 (Table 1).

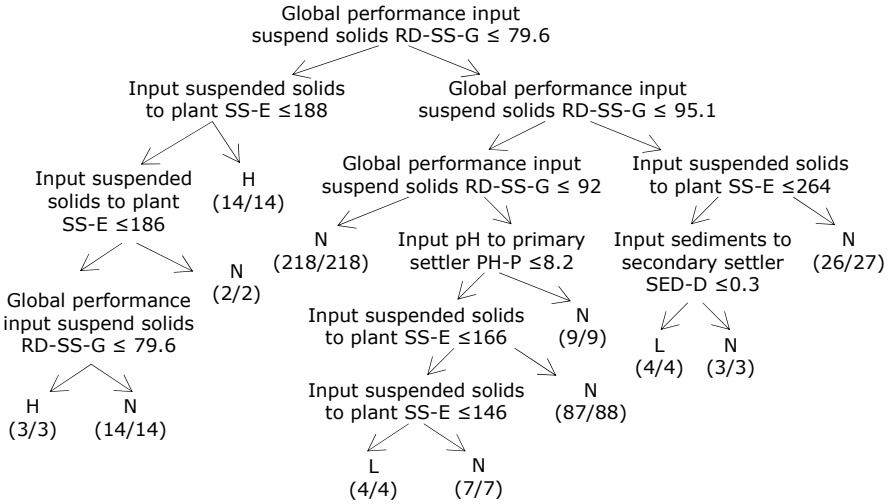


Fig. 5 A decision tree selected into the decision forest which was generated in generation 69 of experiment 14 for predicting suspended solids in effluents SS-S. The training and test accuracies by this tree are 99% and 94% respectively.

5. Conclusions

The applications on developing QSAR models for chemical toxicity prediction and for analysis of the data collected from a wastewater treatment plant proved that GPTree has great advanatges in comparison with algorthms that employ a greedy search strategy. The extension of it, the YAdapt, allows adaptive partitioning of the output values into classes during the decision tree building process, which was not possible with previous methods. In addition, GPTree provides a natural way for developing decision forest (GPForest) that proved to be an effective strategy for improving the prediction performance for previously unseen data.

Table 1 Comparison between GPForest, GPTree and See 5 in predicting the four outputs[a]

Results		GPForest	See5	GPTree
SS-S	Training	99.5 %	96.7 %	99.5%
	Test	99.3 %	98.5 %	98.5%
DBO-S	Training	99.5 %	99.2 %	99.0%
	Test	99.3 %	95.5 %	99.3%
DQO-S	Training	99.7 %	99.2 %	99.7%
	Test	97.8 %	96.3 %	97.8%
COND-S	Training	100.0 %	99.0 %	100.0%
	Test	98.5%	96.3 %	97.8%

[a] SS-S: suspended solids in effluents; DBO-S: effluent biological oxygen demand; DQO-S: effluent chemical oxygen demand; COND-S: effluent conductivity.

References

F.V. Buontempo, X.Z. Wang, M. Mwense, N. Horan, A. Young, D. Osborn, 2005, J Chem Inf Model, 45, 904-912.
M.T.D. Cronin, T.I. Netzeva, J.C. Dearden, R. Edwards, D.P. Worgan, 2004, Chem Res Toxic, 17, 545 - 554.
J.R. Quinlan, 1993, (Morgan Kaufmann Publishers Inc., 302.
J.R. Quinlan, 1996, J Artif Intell Res, 4, 77-90.
X.Z. Wang, F.V. Buontempo, A. Young, D. Osborn, 2006, SAR QSAR Env Res, 17, 451-471.
Y.H. Zhao, M.T.D. Cronin, J.C. Dearden, 1998, QSAR, 17, 131-138.