Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 5, May 2018, pp. 849-856
DOI: 10.23940/ijpe.18.05.p3.849856

Decision Tree Incremental Learning Algorithm Oriented Intelligence Data

Hongbin Wang, Ci Chu, Xiaodong Xie, Nianbin Wang*, and Jing Sun

College of Computer Science and Technology, Harbin Engineering University, Harbin, 150001, China

Abstract

Decision tree is one of the most popular classification methods because of its advantages of easy comprehension. However, the decision tree constructed by existed methods is usually too large and complicated. So, in some applications, the practicability is limited. In this paper, combining NOLCDT with IID5R algorithm, an improved hybrid classifier algorithm, HCS, is proposed. HCS algorithm consists of two phases: building initial decision tree and incremental learning. The initial decision tree is constructed according to the NOLCDT algorithm, and then the incremental learning is performed with IID5R. The NOLCDT algorithm selects the candidate attribute with the largest information gain and divides the node into two branches, which avoids generating too many branches. Thus, this prevents the decision tree is too complex. The NOLCDT algorithm also improves on the selection of the next node to be split, which computes the corresponding nodal splitting measure for all candidate splits, and always selects the node which has largest information gain from all candidate split nodes as the next split node, so that each split has the greatest information gain. In addition, based on ID5R, an improved algorithm IID5R is proposed to evaluate the quality of classification attributes and estimates a minimum number of steps for which these attributes are guaranteed such a selection. HCS takes advantage of the decision tree and the incremental learning method, which is easy to understand and suitable for incremental learning. The contrast experiment between the traditional decision tree algorithm and HCS algorithm with UCI data set is proposed; the experimental results show that HCS can solve the increment problem very well. The decision tree is simpler so that it is easy to understand, and so the incremental phase consumes less time.

Keywords: classification; data mining; decision tree; incremental learning

(Submitted on January 29, 2018; Revised on March 12, 2018; Accepted on April 23, 2018)

1. Introduction

In most of the existed literature, though size of decision trees is also related to the numbers of branches split from the nodes; decision tree can be simplified by shortening the length of paths. A classification attribute may have several different values. A traditional decision tree algorithm, such as C4.5, will generate the same number of branches as the number of possible values for this attribute. Some of the branches may only contain a small number of records and the classification distributions are rather similar in some branches. All these redundant branches would not only make the decision tree complex, but also increase the probability of overfitting.

2. General Approach

Researchers have proposed many decision tree algorithms by now, in which the classical ones include ID3, C4.5, ID4, ID5, and ID5R. [2,6] Among these algorithms, ID3 and C4.5 are not suitable for incremental learning. However, in the existed society, intelligence data generate and change incessantly. Hence, ID3 and C4.5 fail to meet these challenges. In order to meet requirements of increments, researches have proposed ID4 and its promoted version later. However, in the existed algorithms, the number of node branches may be same as the number of possible values of the classification attribute in the process of node splitting. The incessant splitting may make the decision tree too large, which may limit the practicability of decision tree algorithm. Therefore, it is essential to consider how to simplify the decision tree to make it more practical and to enlarge its application scope.

* Corresponding author.
E-mail address: wangnianbin@hrbeu.edu.cn

3. Study on Algorithm of Decision Tree Incremental Learning Oriented Intelligence Data

Practicability of decision tree constructed by traditional decision tree algorithm is limited because of its too much nodes and complexity. [5,8,10] In order to solve this problem, a new decision tree algorithm called NOLCDT is proposed in this chapter. NOLCDT algorithm allows the users to specify the maximum number of nodes; also, when the number of nodes is restricted, it is possible to construct a decision tree of higher accuracy.

3.1. Decision tree algorithm

In order to estimate the advantages of splitting node, a splitting standard is proposed in the algorithm. It defines where the record number of node is the largest information gain when we split node into two branches. In order to estimate the advantages of splitting node n_i, a splitting standard SC (n_i) is proposed in the algorithm and it defines $\mathrm{SC}(n_i) = |D_{n_i}| \times \mathrm{MG}(n_i)$, where $|D_{n_i}|$ is the record number of node n_i, and $\mathrm{MG}(n_i)$ is the largest information gain when we split node n_i into two branches.

3.1.1. Information gain of binary nodes

For node n_i, the entropy of the corresponding node, namely the corresponding attribute is shown in Equation (1)

$$E(D_{n_i}) = -\sum_{y=1}^{|C|} p_y \log_2(p_y) \tag{1}$$

Where p_y represents the proportion of the classification attributes c_y in D_{n_i}, calculated with $\left|D_{n_i}(c_y)\right|/|D_{n_i}|$.

In order to control the increasing speed of leaf nodes, the NOLCDT algorithm splits each node into two branches, namely, two nodes in each splitting node. For a node n_i and an attribute a_x, it is stipulated that all the possible values of attribute a_x are divided into two groups represented by $\mathrm{G}L_{n_i}(a_x)$ and $\mathrm{G}R_{n_i}(a_x)$ respectively, and the information gain of the attribute is shown in Equation (2)

$$\mathrm{E}(D_{n_i})\text{-}(\omega_L \times \mathrm{E}\left(D_{n_i}\left(a_x \in \mathrm{G}L_{n_i}(a_x)\right)\right) + \omega_R \times \mathrm{E}\left(D_{n_i}\left(a_x \in \mathrm{G}R_{n_i}(a_x)\right)\right)) \tag{2}$$

Where ω_L and ω_R are the weights calculated with $D_{n_i}(a_x \in \mathrm{G}L_{n_i}(a_x))/\left|D_{n_i}\right|$ and $D_{n_i}(a_x \in \mathrm{G}R_{n_i}(a_x))/|D_{n_i}|$.

3.1.2. Dividing the attribute value into two groups

A node n_i and an attribute a_x of several attribute values are given, and the problem is how to find out the best splitting point or how to split the attribute value so as to divide the records in node into two branches, making the information gain maximum. The attribute of data may be either numeric or category type. For the numeric attribute, the methods in C4.5 algorithm can be used [7,9]. Therefore, all possible splitting points of numeric attribute are tested. If there are n different values $v_1, v_2, \ldots, v_n$ for a numeric attribute, and i > j, $v_i > v_j$, namely, these values are sorted ascending, there will be n-1 splitting points $(v_1 + v_2)/2$, $(v_2 + v_3)/2$,…, $(v_{n-1} + v_n)/2$ that would be tested. The point that would maximize information gain would be selected as the splitting point in the numeric attribute. For example, if we select attribute "age" in node n_0 to classify, then the splitting point would be (22+23)/2=22.5, and the information gain is 0.236. In the node n_0, records would be divided into two subaggregates, in which the age is smaller than 22.5 and bigger than 22.5 respectively. However, in node n_0, the information gain of attribute "age" is 0.236, smaller than 0.278 of attribute "income"; therefore, "income" should be the classification attribute of node n_0.

For the category type attribute, in the traditional decision tree algorithm C4.5, numbers of attribute value decide the numbers of branches after splitting. A node n_i and a classification attribute a_x are given, the node n_i is split with attribute a_x, and $|\mathrm{P}V_{a_x}|$ branches would be generated. For binary splitting, branches should be combined into two with certain methods and minimize the information gain in the branches, namely, make the combined branches as similar as possible. The most direct method is to list all assemble to calculate the information gain of each assemble and select the best one. However, if there are $|\mathrm{P}V_{a_x}|$ possible values, there would be $2|\mathrm{P}V_{a_x}|/2$ assembling methods to classify them into two groups, and the

complexity of time would be increased exponentially. Therefore, the performance of combining values of attributes arbitrarily is too poor.

In order to avoid too much time used in calculation, the algorithm adopts the concept of cluster to combine $|PV_{a_x}|$ branches. The idea of this algorithm is to combine similar classifications into a new branch and the new branch needs less information. About the distribution of classification, we define dis $(D_{n_i}(a_x \in U_1), D_{n_i}(a_x \in U_2))$ to test the distance between two groups of examples $D_{n_i}(a_x \in U_1)$ and $D_{n_i}(a_x \in U_2)$ in order to estimate the similarity of these two branches, and calculate with $\sum_{y=1}^{|C|}(|D_{n_i}(a_x \in U_1) \cap D_{n_i}(c_y)| - |D_{n_i}(a_x \in U_2) \cap D_{n_i}(c_y)|)^2$.

Distance between each two branches are calculated and combine the two branches with the smallest distance until only two branches are left.

3.1.3. Standard of node splitting

In the calculation of advantages of splitting nodes, if the numbers of leaf nodes are limited, numbers of records of the dividable nodes are also an important factor except for information gain [4,11]. For node n_i, more records of node n_i means that the covering scope of it in training data set is bigger, that is, the node n_i may cover the records of those have not split by now. Therefore, to split the node with more records has more advantages in the future classifications. In addition, to split a node with too little records may increase the risk of overfitting. Therefore, the number of records in the node is used as a weight, and the weighted information gain is used to calculate advantages to split a node, that is, the standard of splitting node is $SC(n_i)=|D_{n_i}| \times MG(n_i)$.

The NOLCDT algorithm is as following:

Algorithm 1 NOLCDT (construct initial decision tree) M

Input: original training set

Output: constructed decision tree T

1. Take node as the root node of decision tree. Calculate splitting value of node, if the node is dividable, set the node into pending node set Q;
2. If the number of nodes in decision tree T is less than the limited maximum leaf node number and the set Q is non empty, repeat the following practice for all nodes in set Q;
3. Select the node with the maximum splitting value in the set Q and delete it from set Q;
4. Split node and calculate the splitting value of the two nodes generated by;
5. For either of the two nodes generated by node, if it is able to split later, put it into set Q;
6. After splitting node, if the number of leaf nodes in the decision tree T is equal to the limited maximum leaf nodes number or the set Q is empty, end the algorithm and define all the nodes in set Q as leaf nodes.

3.2. IID5R algorithm

When a decision tree and a new training sample are given, we should add the new sample into the decision tree with the method of incremental learning [1]. The initial decision tree is constructed by the proposed NOLCDT algorithm based on the training data set. Generally, the test is implemented from the root node, and the classification attribute and candidate attribute of the root node have corresponding information entropy, which may be changed as the new sample is added [3]. It means that in certain time, there may be a condition that there is no better candidate attribute in the classification attribute. Once it occurs, a new better attribute would be in the root node and the original classification attribute is in a lower tier. And the pulling up operation is implemented with this method, which is a structural operation and the attributes are exchanged between the adjacent layers in decision tree. When lowering the original classification attribute of the root node, all the subtree of the root nodes need to be adjusted. When the new sample goes along the path from root node to the leaf nodes in accordance with the attribute value, the entropy of all the attributes in each node will be updated.

An obvious disadvantage of ID5R algorithm is that, except for the updated entropy of candidate attribute, a selection process is asked in each node in the process that the new sample continues decision along the decision path. The selection process is to identify in accordance with the splitting standard that whether the existed classification attribute is the optimal and whether the information gain is largest. In accordance with this algorithm, a key point would be omitted, that is, the existed classification attribute is certainly better than others in the limited procedures, and the classification attribute would not change into a second-superior attribute. The following discussion and analysis are quantification of the limited procedures. In the limited procedures, a blind consideration in the influences of new added samples on classification

attribute would decrease learning speed. The study object is to find out the step that ensures the stability of the node classification attribute and its irreplaceable role (a step is a sample). After these procedures, we should take into consideration that the node needs another new attribute as new classification attribute. Before these procedures, there is no need to worry about whether this operation should be implemented ahead, that is, we don't need to change classification attribute.

In this section, the problem that the updated decision tree based on new added data is studied, and when the new sample is added. It is not learning by combining with the historical data assemble, but to modify the previous model of the historical data so as to promote the classification accuracy of the decision tree. It is stipulated that there are α sample in a node of the decision tree, and there may be several special cases as following: all the new added samples would enlarge the entropy of the classification attribute, which may make the entropy change increase to the maximum value ΔE_{max}, but make entropy change of a candidate attribute minimum as ΔE_{min}, then the new added α samples would make the inequality of the total gap of the identified classification attribute and candidate attribute as Equation (3)

$$\Delta E_{max} - \Delta E_{min} \le (\sum_{i=1}^{a} \frac{log(b_j + i) + loge}{p + n + 1}) + \frac{a}{p + n + 1} \tag{3}$$

Stipulating that the new added sample number α is less than the sample number b_j in attribute value segment j, that is, zooming the inequality and it can be obtained as Equation (4)

$$\Delta E_{max} - \Delta E_{min} \le \left(\sum_{i=1}^{a} \frac{log(b_j + b_j) + loge}{p + n + 1}\right) + \frac{a}{p + n + 1} = \frac{a(log(b_j) + 2 + loge)}{p + n + 1} \tag{4}$$

The gap of information entropy between classification attribute and candidate attribute decides the number of new added sample α, and the value α of is shown in Equation (5)

$$\begin{aligned} E_2 - E_1 &\ge \frac{a(\log(b_j) + 2 + \log e)}{p + n + 1} \\ \Leftrightarrow a &\le \frac{(p + n + 1)(E_2 - E_1)}{\log(b_j) + 2 + \log e} \\ \Rightarrow a_{\max} &= \frac{(p + n + 1)(E_2 - E_1)}{\log(b_j) + 2 + \log e} \end{aligned} \tag{5}$$

With the above, a method to modify the incremental decision tree can be obtained in accordance with the above formula:

If the new added sample number a ≤ a_{max}, the entropy of the classification attribute is still less than that of candidate attribute even the new samples are added; then, the information gain of corresponding classification attribute would be larger than that of the corresponding candidate attribute. Therefore, the original classification attribute is regarded as the classification attribute of the node; if the new added sample number a > a_{max}, the entropy of classification attribute would be greater than the candidate attribute after adding new samples, then the information gain of corresponding classification would be less than that of corresponding candidate attribute. Therefore, the original classification attribute should be replaced and take the candidate attribute as the classification attribute to modify the decision tree.

4. Experimental results and analysis

4.1. Data set selection

In order to verify the effectiveness of the proposed algorithm, the Credit approval data set in the UCI machine learning database is selected. The UCI machine learning database contains hundreds of experimental data base and artificial data base for specialists and scholars would to carry out various studies, and is a widely used database system. Many of the world-

famous algorithms have used the data set in the UCI machine learning database; therefore, it is also used in this paper to ensure the reliability of the experiments.

4.2. Results and analysis

In order to prove the simplicity and high-accuracy of the decision tree constructed by the NOLCDT algorithm, and to verify that the proposed IID5R algorithm is able to promote learning speed, two groups of experiments are done as following:

(1) Contrast experiment between NOLCDT algorithm and C4.5 algorithm.

(2) Contrast experiment between ID5R algorithm and IID5R algorithm.

4.2.1. Experiment1: Contrast experiment between NOLCDT algorithm and C4.5 algorithm

First of all, take into consideration the non-incremental condition. Performances of NOLCDT algorithm and traditional non-incremental decision tree algorithm C4.5 are compared from the two aspects including classification accuracy and leaf nodes numbers. The idea of the experiment is to set no limitation to the number of leaf nodes and divide the data set into 10 groups, 9 of which are used to train decision tree and the one is used to test the decision tree generated by the training. Based on the training data set, decision trees are constructed with the NOLCDT algorithm and C4.5 algorithm respectively. The accuracy of the constructed decision tree is verified with the test data set. 10 times of experiments are done and the average value of the 10 experiment results is the final result. The comparison figure of classification accuracy is presented with Figure 1, and the comparison figure of leaf node number is presented with Figure 2.

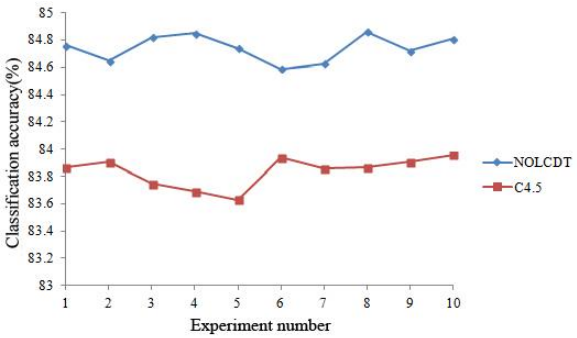

Figure 1. Comparison of classification accuracy without restriction of number of leaf nodes

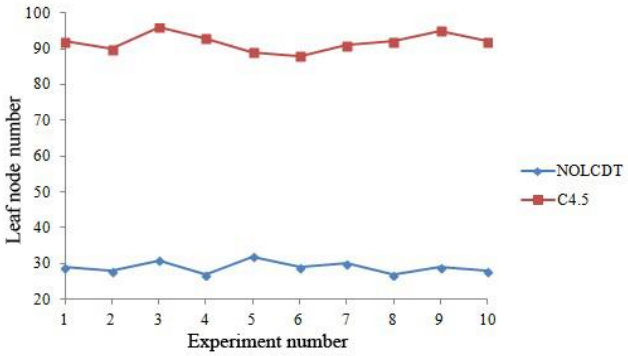

Figure 2. Comparison of the leaf node number without restriction under the conditions of the leaf node number

It can be seen from Figure 1 and 2 that the accuracy of NOLCDT algorithm is higher than that of the C4.5 algorithm; there is no restriction on the leaf node number, and the leaf node number of the NOLCDT algorithm is less than that of C4.5 algorithm.

Then, restrictions are added on the leaf node number, and the performance of NOLCDT algorithm is evaluated by changing the restriction value. Because the numbers of leaf node numbers are restricted, most of the algorithm fails to generate a complete decision tree; the limit of leaf node number is not identified casually, but needs to be identified by a specialist or to refer to previous experiments. When the leaf node numbers are not restricted, the end condition of the

decision tree is that all the experimental cases are the same type. However, once the leaf node number is restricted, the leaf node number may reach the limit before all the experimental cases reach the same type, namely, it cannot end in accordance with the previous standard. Therefore, in algorithm C.5, the sequence of node splitting is of great importance. In this experiment, splitting nodes would be selected in depth-first sequence and breadth-first sequence, which would be presented by C4.5-DF and C4.5-BF respectively.

The results in Figure 1 are used to identify the limit of maximum lead node number. The leaf node number of decision tree generated by C4.5 algorithm is presented as TMaxLeaves. In order to observe the influence of deficiency of leaf nodes on decision tree performance, several experiments are done and the leaf node numbers are decreased gradually from the TMaxLeaves, in which the numbers are TMaxLeaves, 90% of TMaxLeaves, 80% of TMaxLeaves, ..., and 20% of TMaxLeaves. The experimental results are shown in Figure 1.

Table 1. Experimental results of changes in leaf node number limit

Data set	Algorithm	Leaf node number limit（20%×TMaxLeaver-100%×TMaxLeaves）								
		20%	30%	40%	50%	60%	70%	80%	90%	100%
Credit approval	C4.5-DF	83.23	83.38	83.84	84.00	84.00	84.00	84.00	83.84	83.84
	C4.5-BF	85.15	84.71	84.63	84.30	83.58	83.69	83.69	83.69	83.84
	NOLCDT	86.00	84.76	84.76	84.76	84.76	84.76	84.76	84.76	84.76

Then, represent leaf node number in the form of numerical value. When the leaf node numbers are restricted, classification accuracy of NOLCDT algorithm and C4.5 algorithm are shown in Figure 3.

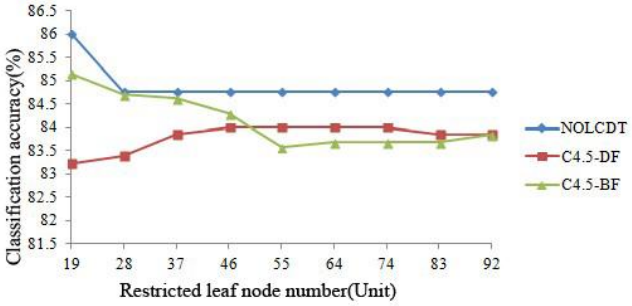

Figure 3. Experimental results of changing the leaf node number limit

In accordance with the experimental results in Figure 3, it can be seen that the accuracy rate of algorithm NOLCDT is higher than that of C4.5. These results proved that when splitting the nodes, the optimal splitting node can be selected by combining the branches with a new splitting standard, which enables the algorithm NOLCDT to obtain a simpler decision tree of higher-accuracy with limited leaf nodes.

4.2.2. Experiment 2: Contrast experiments of algorithm ID5R and IID5R

Because the algorithm C4.5 fails to perform incremental learning, while the ID5R algorithm is able to do it, in order to test the effectiveness of the strategy to modify ID5R, the second experiment is done, namely, the comparison experiment of algorithm ID5R and IID5R.

Taking into consideration the incremental learning, performances of ID5R incremental decision tree algorithm before and after modification are compared from the two aspects including classification accuracy and algorithm execution time. Because the main purpose of this experiment is to judge whether the strategy on ID5R algorithm is able to promote learning speed, it is of no problem if the initial decision trees are the same. The experimental results in the former section illustrated that when the leaf node number is limited as the 20% of TMaxLeaves, the accuracy rate is the highest. Therefore, the leaf node limit is identified as 20% of TMaxLeaves in the experiment. Because the incremental learning capability needs to be tested, the training data set are divided into 11 groups in the experiment. The number of data in first group is half of the data set, and the other 10 data sets are used as the incremental data for incremental learning. Then, algorithm ID5R and IID5R are used respectively to implement incremental learning for new data set after adding data each time. After learning, calculate the algorithm execution time and classification accuracy rate, and the experimental results are shown in Table 2, and the results of all incremental completion are shown in Table 3.

Table 2. ID5R algorithm and incremental learning results

Incremental scale（%）	ID5R		IID5R	
	Execution time（s）	Accuracy rate（%）	Execution time（s）	Accuracy rate（%）
10	2579.5	85.69	2574.1	85.71
20	2787.4	85.70	2780.3	85.74
30	2997.9	85.60	2986.6	86.62
40	3105.4	85.69	3089.7	85.72
50	3317.5	85.70	3295.5	85.79
60	3375.6	85.73	3336.7	85.81
70	3424.8	85.75	3377.1	85.85
80	3474.9	85.76	3418.8	85.87
90	3526.2	85.79	3460.5	85.87
100	3578.3	85.81	3501.5	85.89

Table 3. Final result of the incremental completion of the ID5R and IID5R algorithms

Data set	algorithm	Execution time（s）	Accuracy rate（%）
Credit approval	ID5R	3578.3	85.81
	IID5R	3501.5	85.89

The results in Table 2 show the learning time and corresponding accuracy rate of ID5R and IID5R when data are added. Figure 4 represents the accuracy rate and data scale in Table 2. Figure 5 represents the execution time and data scale. It can be seen from Figure 4 that the classification accuracy rate of IID5R is higher than that of the traditional incremental learning algorithm ID5R, and it can be seen from the curve trend that the classification rate of decision tree can be increased when incessant incremental learning is added in original decision tree.

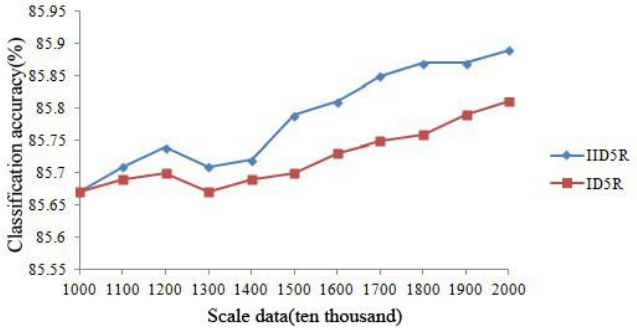

Figure 4. Comparison chart of classification accuracy under different scale data

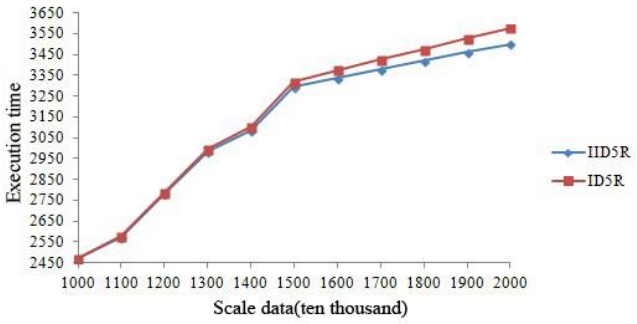

Figure 5. Comparison of execution time under different data size

In accordance with the experimental results in Figure 5, the proposed strategy is able to improve learning speed, which can shorten the execution time. When an initial decision tree and the incremental data set are given, there may be no change in decision tree before the new samples are added. In order to solve how to judge whether the structure of decision tree needs to be adjusted, the methods adopted in this paper is to calculate the minimum record number of each classification attribute that may be substituted by the candidate attribute in accordance with the existed sample. The concrete method is to calculate the minimum record number of the classification attribute that can be substituted by the candidate attribute in line with the formula concluded from the proof reasoning. If the new added record reaches the minimum record of certain

candidate attribute, exchange the candidate attribute with the classification attribute by pulling up algorithm; if there is no minimum record number that reaches the candidate attribute, just record the new samples directly, which would have no impact on the selection of classification attribute in the node.

5. Conclusions

In this paper, decision tree classification and incremental learning method in data mining are deeply studied. A hybrid classifier algorithm called HCS is proposed, by combining the NOLCDT algorithm with IID5R algorithm. The HCS algorithm consists two parts: building the initial decision tree and incremental learning. The initial decision tree is constructed in accordance with the NOLCDT algorithm, based on the initial decision tree, and incremental learning is performed with the IID5R algorithm. HCS takes advantages of NOLCDT algorithm and IID5R algorithm, which is of better interpretability and higher incremental learning ability.

Acknowledgements

This work was funded by the National Natural Science Foundation of China under Grant (No. 61772152 and No. 61502037), and the Basic Research Project (No. JCKY2016206B001, JCKY2014206C002 and JCKY2017604C010).

References

1. K. Adhatrao, A. Gaykar, A. Dhawan, R. Jha, V. Honrao, "Predicting Students' Performance Using ID3 And C4.5 Classification Algorithms," *International Journal of Data Mining & Knowledge Management Process*, vol. 3, no. 5, pp., 2013
2. J. M. Cherry, "The Saccharomyces Genome Database: Advanced Searching Methods and Data Mining," *Cold Spring Harb Protoc*, vol. 2015, no. 12, pp. pdb.prot088906, 2015
3. D. U. Hong-Le, Y. Zhang, "Incremental Support Vector Machine Algorithm on Dynamic Cost," *Journal of Shangluo University*, vol., no., pp., 2017
4. D. Kalles, T. Morris, "Efficient Incremental Induction of Decision Trees," *Machine Learning*, vol. 24, no. 3, pp. 231-242, 1996
5. Liu, Huan, Motoda, Hiroshi, "Feature Selection for Knowledge Discovery and Data Mining," *Springer International*, vol., no. 4, pp. xviii, 1998
6. Z. Q. J. Lu, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction," *Mathematical Intelligencer*, vol. 27, no. 2, pp. 83-85, 2005
7. C. J. Mantas, J. Abellán, "Credal-C4.5: Decision Tree Based on Imprecise Probabilities to Classify Noisy Data," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4625-4637, 2014
8. J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, no. 1, pp. 81-106, 1986
9. L. Rutkowski, M. Jaworski, L. Pietruczuk, P. Duda, "The CART Decision Tree for Mining Data Streams," *Information Sciences*, vol. 266, no. 5, pp. 1-15, 2014
10. C. W. Tsai, C. F. Lai, M. C. Chiang, L. T. Yang, "Data Mining for Internet of Things: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 77-97, 2014
11. C. C. Wu, Y. L. Chen, Y. H. Liu, X. Y. Yang, "Decision Tree Induction with a Constrained Number of Leaf Nodes," *Applied Intelligence*, vol. 45, no. 3, pp. 1-13, 2016

Hongbin Wang received his PhD degree in computer application technology from Harbin Engineering University in 2010. He is currently an associate professor in the college of Computer Science and Technology of Harbin Engineering University. His research interests include information management, data integration, data space, semantic web, ontology and information system design.

Ci Chu received his Bachelor's degree in computer application technology from Harbin Engineering University in 2018. He is now studying for a Master's degree at the Harbin Engineering University. His research interests include deep learning, web architecture, data integration, data space, and data classification.

Xiaodong Xie is now studying for a Bachelor’s degree in computer application technology at the Harbin Engineering University in 2018. His research interests include deep learning, big data, AI, and machine learning.

Nianbin Wang received his PhD degree in computer science and technology from Harbin Institute of Technology in 2001. He is member of CCF, he has been a Professor with the Department of Computer Science and Technology, Harbin Engineering University. His interests include: dataspace, deep learning, and data integration.

Jing Sun received her master degree in software engineering from Harbin Engineering University in 2016. Her research interests include information management, data integration, and data classification.