Artif Intell Rev (2013) 40:517–530
DOI 10.1007/s10462-011-9293-z

An incremental decision tree algorithm based on rough sets and its application in intrusion detection

Feng Jiang · Yuefei Sui · Cungen Cao

Published online: 23 December 2011

Abstract As we know, learning in real world is interactive, incremental and dynamical in multiple dimensions, where new data could be appeared at anytime from anywhere and of any type. Therefore, incremental learning is of more and more importance in real world data mining scenarios. Decision trees, due to their characteristics, have been widely used for incremental learning. In this paper, we propose a novel incremental decision tree algorithm based on rough set theory. To improve the computation efficiency of our algorithm, when a new instance arrives, according to the given decision tree adaptation strategies, the algorithm will only modify some existing leaf node in the currently active decision tree or add a new leaf node to the tree, which can avoid the high time complexity of the traditional incremental methods for rebuilding decision trees too many times. Moreover, the rough set based attribute reduction method is used to filter out the redundant attributes from the original set of attributes. And we adopt the two basic notions of rough sets: significance of attributes and dependency of attributes, as the heuristic information for the selection of splitting attributes. Finally, we apply the proposed algorithm to intrusion detection. The experimental results demonstrate that our algorithm can provide competitive solutions to incremental learning.

Keywords Rough sets · Decision trees · Incremental learning · Significance of attributes · Intrusion detection

F. Jiang (✉)
College of Information Science and Technology, Qingdao University of Science and Technology, Qingdao 266061, People's Republic of China
e-mail: jiangkong@163.net

Y. Sui · C. Cao
Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, People's Republic of China

1 Introduction

Most of the existing machine learning algorithms update learning models mainly by relearning completely. Namely, these algorithms add new training instances to the original training dataset and relearn the model on the whole dataset, which makes the learning efficiency of these algorithms inferior and iterative times of model update limited. An ideal learning model update algorithm should operate continuously and definitely incorporate new instances into the model as they arrive. Such desiderata are fulfilled by incremental learning (Utgoff 1989).

By now, various incremental learning methods have been developed to deal with the problem of dynamic databases in the fields of decision trees, neural networks, association rules, clustering methods, and so on (Schlimmer and Fisher 1986; Utgoff 1989, 1994; Domingos 2000; Shan and Ziarko 1993; Bian 1998; Zheng and Wang 2004; Liu 1999; Borrajo and Veloso 1997; MacLeod and Maxwell 2001). Especially, decision trees, due to their characteristics, have been widely used for incremental classification. For instance, Schlimmer and Fisher first developed a decision tree based incremental learning algorithm ID4 (Schlimmer and Fisher 1986). Then Utgoff further proposed three incremental decision tree algorithms: ID5 (Utgoff 1989), ID5R (Utgoff 1989), and ITI (Utgoff 1994). The current incremental decision tree algorithms can be roughly divided into two categories. The first category uses a greedy search to rebuild the tree, e.g. ITI (Utgoff 1994) or ID5R (Utgoff 1989). These methods often have high time complexity since they need to rebuild decision trees many times. The second category maintains a set of sufficient statistics, i.e. Hoeffding bounds (Domingos 2000), at each decision node and only makes a split-test when there is enough statistical evidence. The second type needs large number of instances (at least several hundred instances for every leaf node) to calculate the Hoeffding bounds, although it could achieve better performance.

Since most of the current decision tree algorithms use all attributes to construct decision trees, and do not concern the relevancy and dependency among attributes (Quinlan 1986, 1993; Han and Kamber 2001; Schlimmer and Fisher 1986; Utgoff 1989, 1994; Domingos 2000), this may lead to extra overhead in terms of memory and computational efforts. To solve that problem, rough set-based decision tree algorithms have been proposed within last few years (Li and Dong 2008; Jiang et al. 2004; Huang et al. 2007; Wei et al. 2002; Bai et al. 2003). Rough set theory was first proposed by Pawlak in 1982 (Pawlak 1991). It is a kind of very useful theory dealing with vagueness or uncertainty. Recently, rough set theory has been proven to be an effective tool in the fields of data mining, pattern recognition, decision support systems, etc. (Wang 2001; Jiang et al. 2009; Huang et al. 2010). Especially, rough set theory has been recognized to be one of the powerful tools for attribute selection (Liu et al. 2003; Xu et al. 2006; Hu 1995; Hu et al. 2003, 2008; Wang et al. 2002; Miao and Hu 1999; Chen et al. 2010), which can be used to eliminate the unnecessary attributes in the dataset and thereby create a simplified version of the original dataset.

Although in rough set theory much attention has been given to the attribute reduction and rule extraction algorithms, there are also numerous studies of incremental rough set based approaches (Shan and Ziarko 1993; Bian 1998; Zheng and Wang 2004; Liu 1999). For instance, Shan et al. proposed an incremental rough set learning algorithm, but it does not support inconsistent data (Shan and Ziarko 1993). To solve this problem, Bian et al. presented an improved algorithm based on Shan's algorithm, using an extended decision matrix to deal with inconsistent data (Bian 1998). Moreover, Liu et al. presented an incremental arithmetic for the smallest reduction of attributes (Liu 1999). Zheng et al. proposed an incremental knowledge acquisition algorithm based on rough sets and rule tree (Zheng and Wang 2004).

Intrusion detection systems (IDS) have become an essential component of computer security (Bace and Mell 2001). The concept of IDS was first introduced by Anderson in 1980. Due to the large volumes of security audit data as well as complex and dynamic properties of intrusion behaviors, to optimize the performance of IDS, many artificial intelligence techniques have been utilized, where decision tree technology plays an important role in IDS (Li and Ye 2001).

Most of the current methods to IDS are non-incremental, which usually build a static intrusion detection model on the initial training dataset, and then utilize this model to predict on new network data at the detecting stage (Li and Ye 2001). However, the network behavior profiles defined on the initial training dataset will change continually with the detecting and analyzing process going. Thus the initial model cannot adapt to the new network behavior patterns, which may cause that the detection accuracy of IDS declines. Moreover, with the accumulation of new instances, the training time will continuously increase, which is against the requirement of real-time intrusion detection. To improve the detection accuracy of IDS in dynamic changing network environment and obtain the ability of real-time detection from new instances, we need an incremental intrusion detection method.

In this paper, we propose a novel incremental decision tree algorithm—IDTRS based on rough sets and apply it to IDS. In IDTRS, the significance of attributes and the dependency of attributes in rough sets are adopted as the heuristic information for the selection of splitting attributes. Differing from the traditional incremental decision tree algorithms, the decision trees induced by IDTRS contain two kinds of leaf nodes—*basic leaf node* and *compound leaf node*. For each basic leaf node bln, the path from the root to bln corresponds to one decision rule. However, for each compound leaf node cln, the path from the root to cln corresponds to more than one decision rules, and these rules contradict each other, i.e., they have the same premises, but different consequences. (Note: in the remainder of this paper, when there is no ambiguity, we shall say that each leaf node in a decision tree, instead of the path from the root to the leaf node, corresponds to one or more decision rules.)

When a new instance arrives, IDTRS algorithm will only modify some existing leaf node in the currently active decision tree or adding a new leaf node to the tree, according to the given decision tree adaptation strategies. Moreover, the rough set based attribute reduction method is used to filter out redundant attributes, which is crucial to IDS, since the amount of audit data that IDS needs to examine is usually very large even for a small network. By virtue of the above mechanisms, the computation complexity of IDTRS is relatively low, compared with the traditional algorithms. And the model obtained by IDTRS is similar to those obtained by non-incremental algorithms. Numerical experiments show that our algorithm is efficient for intrusion detection.

The remainder of this paper is organized as follows. In the next section, we present some preliminaries that are relevant to this paper. In Sect. 3, we present our algorithm IDTRS. Moreover, we discuss the application of the algorithm in IDS. Experimental results are given in Sect. 4. Finally, Sect. 5 concludes the paper.

2 Preliminaries

In rough set terminology, a data table is also called an information system, which is formally defined as follows (Pawlak 1991; Wang 2001).

Definition 1 (*Information System*). An information system is a quadruple $IS = (U, A, V, f)$, where:

(1) U is a non-empty finite set of objects;
(2) A is a non-empty finite set of attributes;
(3) V is the union of attribute domains, i.e., $V = \bigcup_{a \in A} V_a$, where V_a denotes the domain of attribute a;
(4) $f : U \times A \to V$ is an information function which associates a unique value of each attribute with every object belonging to U, such that for any $a \in A$ and $x \in U$, $f(x, a) \in V_a$.

If some of the attributes are interpreted as outcomes of classification, the information system $IS = (U, A, V, f)$ can also be defined as a *decision table* by $DT = (U, C, D, V, f)$, where $C \cup D = A$, $C \cap D = \emptyset$, and C is called the *set of condition attributes*, D the *set of decision attributes*, respectively (Pawlak 1991; Wang 2001).

Definition 2 (*Indiscernibility Relation*). Given a decision table $DT = (U, C, D, V, f)$, let $B \subseteq C \cup D$, we call binary relation $IND(B)$ an indiscernibility relation, which is defined as follows:

$$IND(B) = \{(x, y) \in U \times U : \forall a \in B \ (f(x, a) = f(y, a))\}. \tag{1}$$

Since the indiscernibility relation $IND(B)$ is also an equivalence relation on set U, it partitions U into disjoint subsets (or equivalence classes), let $U/IND(B)$ denote the family of all equivalence classes of $IND(B)$. For simplicity, U/B will be written instead of $U/IND(B)$. For every object $x \in U$, let $[x]_B$ denote the equivalence class of relation $IND(B)$ that contains element x, called the equivalence class of x under relation $IND(B)$ (Pawlak 1991; Wang 2001).

Definition 3 (*Positive Region*). Given a decision table $DT = (U, C, D, V, f)$, for any $P \subseteq C$, the positive region $POS_P(D)$ of the partition U/D with respect to P is the set of all objects from U which can be classified with certainty to classes of U/D employing attributes from P, i.e.,

$$POS_P(D) = \bigcup_{E \in U/P \wedge \forall x, y \in E((x,y) \in IND(D))} E, \tag{2}$$

where U/P denotes the partition of domain U induced by relation $IND(P)$ (Pawlak 1991; Wang 2001).

Definition 4 (*Significance of Attribute*) Given a decision table $DT = (U, C, D, V, f)$, let $P \subseteq C$, for any $a \in P$, the significance of attribute a with respect to P and D in DT is defined as follows (Pawlak 1991; Wang 2001):

$$SGF(a, P, D) = POS_P(D) - POS_{P-\{a\}}(D). \tag{3}$$

An important issue in data analysis is discovering dependencies between attributes in a given decision table. Formally such a dependency can be defined in the following way (Pawlak 1991; Wang 2001).

Definition 5 (*Degree of Dependency*). Given a decision table $DT = (U, C, D, V, f)$, for any $P \subseteq C$, the degree of dependency of D on P in DT is defined as follows.

$$\gamma_P(D) = \frac{|POS_P(D)|}{|U|}, \tag{4}$$

where $|M|$ denotes the cardinality of set M.

If $\gamma_P(D) = 1$, then we say that D depends totally on P, and if $0 < \gamma_P(D) < 1$, then we say that D depends on P in a degree $\gamma_P(D)$. If $\gamma_P(D) = 0$, then we say that D does not depend on P. Based on the degree of dependency, the concept of reduct can be defined as follows (Pawlak 1991; Wang 2001).

Definition 6 (*Reduct*). Given a decision table $DT = (U, C, D, V, f)$, for any $R \subseteq C$, R is called a reduct of C with respect to D in DT if

$$\gamma_R(D) = \gamma_C(D) \wedge \forall R' \subset R(\gamma_{R'}(D) < \gamma_C(D)). \tag{5}$$

3 Incremental decision tree algorithm IDTRS and its application in IDS

In this section, we shall propose the incremental decision tree algorithm IDTRS, and discuss how to apply IDTRS to intrusion detection.

Next, we first introduce a concept called *support degree of rule*, which is commonly used in association rules mining and rough set based rule induction (Han and Kamber 2001; Shan and Ziarko 1993; Bian 1998; Zheng and Wang 2004). For each decision rule generated by IDTRS algorithm, we use support degree to measure the strength of the rule.

Definition 7 (*Support Degree*) Given a decision table $DT = (U, C, D, V, f)$, let Red be a reduct of C with respect to D in DT, and $DT' = (U', Red, D, V', f')$ the reduced decision table of DT. Assume that Tr is a decision tree induced by IDTRS algorithm on DT', for each rule r in Tr, we define the support degree of rule r in decision table DT as follows.

$$SUP_{DT}(r) = \frac{|S_{DT}(r)|}{|U|}, \tag{6}$$

where $S_{DT}(r) = M_{DT}(r) \cap N_{DT}(r)$, $M_{DT}(r)$ denotes the set of all objects in U that match the premise of rule r (Note: since the decision table DT has been reduced based on the reduct Red before constructing decision tree, only those attributes in Red will occur in r), and $N_{DT}(r)$ denotes the set of all objects in U that match the consequence of r.

An incremental decision tree learning process usually contains two steps: (1) utilize some basic (non-incremental) decision tree algorithm to construct an initial decision tree on the initial training dataset; (2) for each new instance in the incremental training dataset, utilize an incremental decision tree algorithm to modify the currently active decision tree, and obtain a new decision tree as the currently active decision tree. Next, we shall first present a non-incremental decision tree algorithm DTRS based on rough sets. Then we give the incremental decision tree algorithm IDTRS based on rough sets.

Algorithm 1 DTRS.

Input: *decision table* $T_1 = (U, C, D, V, f)$, *i.e.*, *the initial training dataset*, *where* $D = \{d\}$.
Output: *the initial decision tree* IDT, *and set* IDR *of initial decision rules.*

Function Main(T_1)
(1) *For each continuous attribute* $a \in C$, *discretize* a *in* T_1 *by virtue of a discretization approach* (Catlett 1991; Wong and Chiu 1987; Nguyen and Nguyen 1998). *Let* $T_2 = (U_2, C, D, V_2, f_2)$ *denote the discretized dataset.*
(2) *Compute the reduct* Red *of* C *with respect to* D *in* T_2, *using a rough set based attribute reduction approach* (Liu et al. 2003; Xu et al. 2006; Hu 1995; Hu et al. 2003; Wang et al. 2002; Miao and Hu 1999).

(3) *Reduce T_2 based on the reduct Red. Let $T_3 = (U_3, Red, D, V_3, f_3)$ denote the reduced dataset.*
(4) $IDT = Decision\_Tree(T_3)$. *//Call a function defined below to induce the initial decision tree IDT on T_3.*
(5) *Generate a set IDR of initial rules by traversing all the paths from the root to leaf nodes in IDT. And for each rule $r \in IDR$, let $SUP(r) = SUP_{T_2}(r)$, where $SUP(r)$ denotes the support degree of r.*
(6) *Return IDT and IDR.*

Function Decision_Tree($T_{current}$) *// $T_{current} = (U', C', D, V', f')$ is the decision table that is currently in use.*
(1) *For each attribute $a \in C'$, calculate the significance $SGF(a, C', D)$ of a with respect to C' and D in $T_{current}$.*
(2) *Select attribute t from C' with the largest significance as the splitting attribute.*
(3) *If there exist more than one attributes with the largest significance (let LS denote the set of all attributes with the largest significance), then for each $a \in LS$ calculate the degree of dependency $\gamma_{\{a\}}(D)$ in $T_{current}$, and select attribute t with the largest degree of dependency as the splitting attribute (If there still exist more than one attributes with the largest degree of dependency, then arbitrarily select one from them as the splitting attribute).*
(4) *Create a node N, label N with attribute t, and let $C' = C' - \{t\}$.*
(5) *Calculate the partition $U'/\{t\} = \{E_1, \ldots, E_k\}$.*
(6) *For each equivalence $E_i \in U'/\{t\}$, $1 \le i \le k$*
  (6.1) *Generate one branch of N, labeled with $t = f'(o, t)$, where $o \in E_i$;*
  (6.2) *If all objects in E_i belong to the same decision class c_0, then add a basic leaf node bl labeled with c_0;*
  (6.3) *If $C' = \emptyset$, and objects in E_i belong to different decision classes (Let sc be the set of all these decision classes), then add a compound leaf node cl labeled with set sc.*
  (*Note*: *the compound leaf node cl corresponds to a set of rules, the premises of these rules are the same, but the consequences of these rules are different. And the consequence of each rule is some element of set sc.*)
  (6.4) *If $C' \neq \emptyset$, and objects in E_i belong to different classes, then obtain a sub-table $T_{sub} = \{E_i, C', D, V_{sub}, f_{sub}\}$ of $T_{current}$, and call function Decision_Tree (T_{sub}) to construct a sub-tree.*

Usually, given a decision table $T_{current} = (U', C', D, V', f')$, for any $B \subseteq C'$, the time complexity for calculating the partition U'/B is $O(n^2)$, where $n = |U'|$ (Liu et al. 2003). However, in algorithm 1, before computing the partition U'/B, we first sort all objects of U' based on the counting sort (Xu et al. 2006), then we can calculate the partition U'/B in $O(m \times n)$ time, where $m = |B|$. And from Definitions 3–5, it is not difficult to see that we can calculate the significance $SGF(a, B, D)$ of attribute a in $O(m \times n)$ time. Moreover, the time complexity for calculating the degree of dependency $\gamma_{\{a\}}(D)$ is $O(n)$. Therefore, in the worst case, the time complexity of step (4) in function Main is $O(|Red|^3 \times |U_3|)$. And the overall time complexity of algorithm 1 is also determined by that of the discretization approach in step (1) of function Main and that of the attribute reduction approach in step (2) of function Main.

As we know, the key issue for constructing a decision tree is how to select an appropriate splitting attribute for each non-leaf node of the tree. Differing from the well-known ID3 and

C4.5 algorithms (Quinlan 1986, 1993), DTRS algorithm employs the significance of attributes and the degree of dependency of attributes as the heuristic information for the selection of splitting attributes. In each step of DTRS for creating a non-leaf node, the condition attribute with the largest significance will be selected as the current splitting attribute. And if there exist more than one attributes with the largest significance, then the attribute with the largest degree of dependency will be selected.

Algorithm 2 IDTRS.

Input: *decision table* $T_{new1} = (U_{new1}, C, D, V_{new1}, f_{new1})$, *i.e.*, *the incremental training dataset, the initial decision tree* IDT, *and the set* IDR *of initial decision rules.*
Output: *The updated decision tree* UDT, *and set* UDR *of updated decision rules.*

---

(1) *For each continuous attribute* $a \in C$, *discretize* a *in* T_{new1} *by virtue of the set of cuts obtained in algorithm* 1. *Let* $T_{new2} = (U_{new2}, C, D, V_{new2}, f_{new2})$ *denote the discretized dataset.*
(2) *Reduce* T_{new2} *based on the reduct* Red *obtained in algorithm* 1. *Let* $T_{new3} = (U_{new3}, Red, D, V_{new3}, f_{new3})$ *denote the reduced dataset.*
(3) *Let* $CDT = IDT$ *be the currently active decision tree, and* $CDR = IDR$ *be the currently active set of rules.*
(4) *For each object* $o \in U_{new3}$
  (4.1) *Let* $|[o]_{Red}|$ *denote the cardinality of equivalence class of* o *under relation* $IND(Red)$ *in decision table* T_{new2}.
  (4.2) *Let the root of* CDT *be the current node* N.
  (4.3) *If* N *is not a leaf node then let* a *be the attribute with respect to* N, *and if there exists a branch* b *of* N *such that* $f_{new3}(o, a) = v$, *where* v *is the attribute value with respect to* b, *then select the child node corresponding to* b *as the new current node* N.
  (4.4) *Repeat* (4.3), *until* N *is a leaf node or there does not exist a branch of* N *such that* $f_{new3}(o, a) = v$.
  (4.5) *If* N *is a basic leaf node, then*
    (*I*) *Let* c_0 *be the decision class with respect to* N, *and assume* N *corresponds to the rule* r.
    (*II*) *If* $f_{new3}(o, d) = c_0$ *then let* $SUP(r) = SUP(r) + |[o]_{Red}|$,
      else let N *be a compound leaf node, and let* $sc = \{c_0, f_{new3}(o, d)\}$ *be the set of decision classes with respect to* N. N *corresponds to two rules* r *and* r_1, *where the consequence of* r_1 *equals* $f_{new3}(o, d)$, *and* $SUP(r_1) = |[o]_{Red}|$. *Add* r_1 *to* CDR.
  (4.6) *If* N *is a compound leaf node, then*
    (*I*) *Let* $sc = \{c_1, \ldots, c_m\}$ *be the set of decision classes with respect to* N, *and we assume that* N *corresponds to a set* $R = \{r_1, \ldots, r_m\}$ *of rules, where the consequence of rule* r_j *equals* c_j, $1 \le j \le m$.
    (*II*) *If there exists a decision class* $c_j \in sc$ *such that* $f_{new3}(o, d) = c_j$, *then let* $SUP(r_j) = SUP(r_j) + |[o]_{Red}|$,
      else let $sc = \{c_1, \ldots, c_m, f_{new3}(o, d)\}$ *be the set of decision classes with respect to* N. *And* N *corresponds to set* $R = \{r_1, \ldots, r_m, r_{m+1}\}$ *of rules, where* $SUP(r_{m+1}) = |[o]_{Red}|$, *and the consequence of rule* r_{m+1} *equals* $f_{new3}(o, d)$. *Add* r_{m+1} *to* CDR.
  (4.7) *If there does not exist a branch of* N *such that* $f_{new3}(o, a) = v$, *then*

add a new branch of N, *labeled with* $a = f_{new3}(o, a)$, *and add a basic leaf node bl to the branch. Let* $f_{new3}(o, d)$ *be the decision class with respect to bl, and assume bl corresponds to the rule r, where the consequence of rule r equals* $f_{new3}(o, d)$, *and* $SUP(r) = |[o]_{Red}|$. *Add r to* CDR.

(5) *Let* $UDT = CDT$, *and* $UDR = CDR$.

(6) *Return* UDT *and* UDR.

In the worst case, the time complexity of algorithm 2 is $O(|U_{new3}| \times |UDR| + |U_{new1}| \times |C|)$, where $|M|$ denotes the cardinality of set M.

As mentioned in Sect. 1, there exists an essential difference between IDTRS algorithm and the traditional incremental decision tree algorithms. That is, the decision trees induced by IDTRS contain two kinds of leaf nodes: basic leaf node and compound leaf node. And the two kinds of leaf nodes play different roles for the generation of decision rules. Assume that Tr is a decision tree induced by IDTRS, each basic leaf node bln in Tr corresponds to one decision rule. However, each compound leaf node cln in Tr corresponds to more than one decision rules (We may assume that cln corresponds to a set $R = \{r_1, \ldots, r_m\}$ of rules). For any two rules $r_i, r_j \in R$, they contradict each other. Moreover, since we assign a support degree to each rule in Tr, different rules of R may have different support degrees.

During the process of incremental learning, let CDT be the currently active decision tree, when new instances arrive, IDTRS will adopt different decision tree adaptation strategies based on different types of relationships between new instances and CDT. The decision tree adaptation strategies can be divided into four categories. First, if the new instance o is consistent with some existing rule r in CDT, then nothing will be done except modifying the support degree of r. Second, if there does not exist any rule in CDT that is consistent with o but only exists one rule r in CDT that contradicts o, then IDTRS will only modify the basic leaf node bln_r in CDT that corresponds to rule r, i.e., convert bln_r from a basic leaf node to a compound leaf node, and let bln_r correspond to two rules r and r_1, where r_1 is a new rule obtained from o. Third, if there does not exist any rule in CDT that is consistent with o but exists a set R of rules in CDT that contradicts o, then IDTRS will only modify the compound leaf node cln_R in CDT that corresponds to R, i.e., let cln_R correspond to a new set $R \cup \{r'\}$ of rules, where r' is a new rule obtained from o. Finally, if there does not exist any rule in CDT that is consistent with o or contradicts it, then IDTRS will add a new basic leaf node to CDT.

Moreover, to make DTRS and IDTRS algorithms more suitable for intrusion detection, we adopt the following two measures:

(1) Discretization of continuous attribute. Continuous attributes are very common in IDS, for instance, the KDD-99 dataset contains 41 attributes among which 34 are continuous and only 7 are nominal (KDD Cup 99 Dataset 1999; Bay 1999). Since many machine learning approaches such as rough sets and decision trees would work better on discretized or binarized data (Catlett 1991). To utilize such approaches more effectively in IDS, we need to replace all continuous attributes in the dataset with nominal attributes. This process is called "discretization". By now, many discretization algorithms have been proposed, which can be divided into two categories: unsupervised and supervised (Wang 2001). In this paper, for simplicity, we shall mainly use two unsupervised discretization algorithms: Equal Width Binning (Wong and Chiu 1987) and Equal Frequency Binning (Nguyen and Nguyen 1998), to discretize the continuous attributes in IDS.

(2) Attribute reduction. Attribute reduction (or called feature selection) is considered as one of the most critical processes in intrusion detection, since the amount of audit data that

IDS need to examine is very large even for a small network. The elimination of redundant attributes may enhance the accuracy of detection, and by concentrating on the most important attributes we may well improve the time performance of IDS (Chen et al. 2010).

Rough set theory has been recognized to be one of the powerful tools for attribute reduction. Many rough set based attribute reduction methods have been proposed (Liu et al. 2003; Xu et al. 2006; Hu 1995; Hu et al. 2003, 2008; Wang et al. 2002; Miao and Hu 1999; Chen et al. 2010). And the most basic method is to generate all possible reducts and choose any with the minimal cardinality as a minimal reduct (Wang 2001). Obviously, this method is only practical for very small datasets. Therefore, different heuristic approaches to attribute reduction have been proposed in rough sets, which can be roughly divided into three classes: the positive region based method (Liu et al. 2003; Hu 1995), the discernibility matrix based method (Hu et al. 2003), and the information entropy based method (Wang et al. 2002; Miao and Hu 1999). In this paper, we shall mainly use the positive region based method to filter out the redundant attributes in IDS (Liu et al. 2003).

4 Experiments

To evaluate the performance of IDTRS algorithm, we ran it on the KDD-99 dataset obtained from the UCI Machine Learning Repository (KDD Cup 99 Dataset 1999; Bay 1999). Experiments were conducted on a 2.0 GHz Pentium 4 machine with 2GB RAM, running the Windows XP operating system. We compared algorithm IDTRS with three non-incremental decision tree algorithms: DTRS, ID3 and C4.5. Algorithms IDTRS and DTRS were implemented in Pascal. ID3 and C4.5 algorithms are available in Weka (Waikato environment for knowledge analysis) (Witten and Frank 2000). Weka is a collection of machine learning algorithms for data mining tasks that contains tools for data preprocessing, classification, regression, clustering, association rules, and visualization (Witten and Frank 2000).

4.1 Data source

The KDD-99 dataset is a common benchmark for evaluation of intrusion detection techniques. It contains a wide variety of intrusions simulated in a military network environment. Each instance of the dataset describes a network connection record obtained from the raw network data gathered during the simulated intrusions. And the dataset includes a set of 41 attributes derived from each connection and a label which specifies the status of connection records as either normal or specific attack type (KDD Cup 99 Dataset 1999; Bay 1999).

Since the full KDD-99 training dataset contains 4,898,431 records, which is too large for our purposes, a concise subset of KDD-99, known as "10%KDD", will be discussed here. The 10%KDD dataset contains 22 different attack types and normal records. The 22 attack types fall into four main categories (KDD Cup 99 Dataset 1999; Bay 1999): (1) Probing (PROBE); (2) Denial of Service Attacks (DoS); (3) User to Root Attacks (U2R); (4) Remote to User Attacks (R2L).

Although the 10%KDD dataset is a concise subset of the full KDD-99 training dataset, it contains 494,021 records, which is still very large for our purposes. On the other hand, the 10%KDD dataset is biased to specific attack types. In order to cope with these problems, here following the experimental technique of Chen et al. (2010), we did *random sampling without replacement* from the 10%KDD dataset to form a smaller dataset called "Final_Dataset".

Table 1 The numbers of instances selected

Attack type	Number of instances	Number of instances selected
Back	2,203	441
buffer_overflow	30	30
ftp_write	8	8
guess_passwd	53	53
imap	12	12
ipsweep	1,247	249
Land	21	21
Loadmodule	9	9
Multihop	7	7
Neptune	107,201	2,144
nmap	231	231
Normal	97,278	19,456
Perl	3	3
phf	4	4
pod	264	264
Portsweep	1,040	208
Rootkit	10	10
Satan	1,589	318
Smurf	280,790	5,616
Spy	2	2
Teardrop	979	196
Warezclient	1,020	204
Warezmaster	20	20
Total	494,021	29,506

In the process of random sampling, if an attack is under-represented, i.e., there are few samples to carry random sampling, all attack instances were selected. Table 1 gives the numbers of instances that are selected for different attack types and normal connections.

4.2 Experimental procedures and setup

There are four steps in our experiments: data preparation, data discretization, the construction of decision tree as well as rules generation, and data classification.

(I) The first step is to prepare the dataset. As mentioned in Sect. 4.1, we did *random sampling without replacement* from the 10%KDD dataset to obtain a new dataset called Final_Dataset. Final_Dataset is a subset of 10%KDD dataset, which contains 29,506 instances, and the distribution of different attack types in Final_Dataset has been given in Table 1.

(II) Next is the step of discretization. We employed two unsupervised discretization algorithms: Equal Width Binning (EW) and Equal Frequency Binning (EF), to respectively discretize the continuous attributes in Final_Dataset (Wong and Chiu 1987; Nguyen and Nguyen

1998). EW and EF algorithms are also available in Weka (Witten and Frank 2000). And for both of the two algorithms, the numbers of bins are set to 3. It should be noted that although Final_Dataset contains 34 continuous attributes, we did not discretize all of them. Since for some continuous attributes in the dataset, the number of different attribute values is too small, it is not meaningful to discretize them. In our experiments, the following 26 attributes were selected to be discretized:

(1) Duration; (2) Src_bytes; (3) Dst_bytes; (4) Hot; (5) Num_compromised; (6) Num_root; (7) Num_file_creations; (8) Count; (9) Srv_count; (10) Serror_rate; (11) Srv_serror_rate; (12) Rerror_rate; (13) Srv_rerror_rate; (14) Same_srv_rate; (15) Diff_srv_rate; (16) Srv_diff_host_rate; (17) Dst_host_count; (18) Dst_host_ srv_count; (19) Dst_host_same_ srv_rate; (20) Dst_host_diff_srv_rate; (21) Dst_ host_same_src _port_rate; (22) Dst_host _srv_diff_host_rate; (23) Dst_host_serror_ rate; (24) Dst_host_srv_serror_rate; (25) Dst_ host_rerror_rate; (26) Dst_host_srv_ rerror_rate.

Moreover, for non-incremental decision tree algorithms (including DTRS, ID3 and C4.5), each discretized Final_Dataset was randomly divided into a training dataset (66.67% of the data) and a test dataset (33.33% of the data). And for algorithm IDTRS, each discretized Final_Dataset was randomly divided into three parts: (1) the initial training dataset (33.34% of the data); (2) the incremental training dataset (33.33% of the data); (3) the test dataset (33.33% of the data).

(III) The third step is constructing decision trees on training datasets using algorithm IDTRS and three non-incremental algorithms: DTRS, ID3 and C4.5, respectively. Then different sets of rules can be generated from these trees. As mentioned above, algorithms IDTRS and DTRS were implemented in Pascal. And ID3 and C4.5 algorithms are available in Weka (Witten and Frank 2000).

Since IDTRS is an incremental algorithm, there exist two stages for the construction of decision trees using IDTRS, and we should provide different training datasets at different stages. First, the non-incremental algorithm DTRS was used to induce the initial decision trees on the initial training datasets; Then, algorithm IDTRS was used to update the initial decision trees and induce the incremental decision trees on the incremental training datasets.

Moreover, it should be noted that before applying algorithms DTRS and IDTRS to each training dataset, we first reduced the dataset by virtue of the attribute reduction algorithm proposed by Liu et al. (2003).

(IV) The last step is applying the rules generated in (III) to classify the corresponding test dataset based on a given classifier. For ID3 and C4.5 algorithms, the process of data classification was implemented in Weka. And for DTRS and IDTRS, we adopted the Standard Voting classifier available in Rosetta (Øhrn 1999). The corresponding options are: CLASSIFIER = StandardVoter; FALLBACK = True; FALLBACK.CLASS = 0. (Note: in our experiments, the attack categories DoS, R2L, U2R, Probe and Normal were respectively denoted by 0, 1, 2, 3, 4)

4.3 Experimental results

Table 2 details the classification results of algorithm IDTRS and three non-incremental decision tree algorithms on the dataset discretized by EW algorithm.

In Table 2, "Number of attributes" denotes the number of attributes in the training datasets that are used to build decision trees. For IDTRS and DTRS algorithms, we apply the attribute reduction algorithm proposed by Liu et al. to the training datasets and generate two reducts Red_1 and Red_2, where the former contains 23 attributes and the latter contains 26 attributes. And only those attributes in the reducts are used to build decision trees. However, for ID3

Table 2 A comparison of performances of our algorithm and other algorithms on the dataset discretized by EW

Decision tree algorithm	Type of algorithm	Number of attributes	Classification accuracy (%)	Time taken to build model (s)
IDTRS	Incremental	23	97.9258	0.266
DTRS	Non-incremental	26	97.9969	0.391
ID3	Non-incremental	41	97.8546	2.39
C4.5	Non-incremental	41	97.9156	3.91

Table 3 A comparison of performances of our algorithm and other algorithms on the dataset discretized by EF

Decision tree algorithm	Type of algorithm	Number of attributes	Classification accuracy (%)	Time taken to build model (s)
IDTRS	Incremental	14	99.4001	0.156
DTRS	Non-incremental	20	99.5323	0.281
ID3	Non-incremental	41	99.4509	2.03
C4.5	Non-incremental	41	99.3899	2.16

and C4.5, we use all the 41 attributes in the training dataset to build decision trees. Moreover, "Time taken to build model" denotes the times taken by different algorithms for constructing decision trees, where for IDTRS the time taken to build model includes the time taken to build the initial decision tree and the time taken to build the incremental decision tree.

From Table 2, we can see that for the dataset discretized by EW, IDTRS algorithm performs better than ID3 and C4.5, since the detection rate (i.e., classification accuracy) of our algorithm is higher than those of ID3 and C4.5, while the training time (i.e., time taken to build model) of our algorithm is much lower than those of ID3 and C4.5. Moreover, compared with DTRS, our algorithm yields significant gains in terms of training time and memory requirements at the expense of slightly lower detection rate. Therefore, this experiment shows that our algorithm can generate a better rule learning scheme for IDS than the three non-incremental algorithms.

Moreover, Table 3 details the classification results of algorithm IDTRS and three non-incremental algorithms on the dataset discretized by EF algorithm.

Table 3 is similar to Table 2. From Table 3, it is easy to see that for the dataset discretized by EF, the training time of our algorithm is much lower than those of non-incremental algorithms, and the detection rate of our algorithm is the same as or slightly lower than those of non-incremental algorithms. Hence, this experiment also demonstrates the effectiveness of our incremental decision tree algorithm for intrusion detection.

5 Conclusion

Incremental learning has attracted more and more attention recently, both in theory and application. For instance, many incremental decision tree algorithms have been proposed. Moreover, rough set researchers are also focusing on the issue of incremental learning and have proposed different incremental algorithms for attribute reduction and rule extraction. In this paper, we proposed a novel incremental decision tree algorithm based on rough set

theory and applied it to IDS. By virtue of the rough set based attribute reduction technology and the specific decision tree adaptation strategies, our algorithm can avoid the high time complexity of traditional methods. Hence, our algorithm is suitable to deal with the mass data in IDS. Experimental results on the KDD-99 dataset showed that our algorithm generated a better rule learning scheme for IDS than non-incremental algorithms, since the detection rate of our algorithm is close to those of non-incremental algorithms, while the training time of our algorithm is much lower than those of non-incremental algorithms.

Our incremental decision tree algorithm was proposed based on the Pawlak's classical rough set model (Pawlak 1982, 1991). As the classical rough set model can just be used to deal with discretized attributes, we need to replace all continuous attributes with discretized attributes by virtue of the process of discretization. However, discretization of continuous attributes may cause information loss. Hence, in the future work, we plan to extend our algorithm to the neighborhood rough set model proposed by Hu et al. (2008) which can deal with both continuous and discretized attributes without discretization.

Acknowledgments This work is supported by the National Natural Science Foundation of China (grant nos. 60802042, 61035004), the Natural Science Foundation of Shandong Province, China (grant nos. ZR2011FQ005, ZR2010FQ027), and the Project of Shandong Province Higher Educational Science and Technology Program (grant no. J11LG05).

References

Anderson JP (1980) Computer security threat monitoring and surveillance. James P. Anderson Co., Fort Washington

Bace R, Mell P (2001) Intrusion detection systems. NIST special publication on intrusion detection system, SP 800-31

Bai JS, Fan B, Xue JY (2003) Knowledge representation and acquisition approach based on decision tree. In: Proceedings of the international conference on natural language processing and knowledge engineering, pp 533–538

Bay SD (1999) The UCI KDD repository. Available online at: http://kdd.ics.uci.edu

Bian YH (1998) An incremental algorithm for learning certain rules from inconsistent examples. J East China Shipbuild Inst 12(1):25–30

Borrajo D, Veloso M (1997) Lazy incremental learning of control knowledge for efficiently obtaining quality plans. Artif Intell Rev 11(1-5):371–405

Catlett J (1991) On Changing Continuous Attributes into Ordered Discrete attributes. In: Proceecings of European working session on learning, Springer LNCS, vol 482, pp 164–178

Chen ST, Chen GL, Guo WZ, Liu YH (2010) Feature selection of the intrusion detection data based on particle swarm optimization and neighborhood reduction. J Comput Res Dev 47(7):1261–1267

Domingos P, Hulten G (2000) Mining high-speed data streams. In: Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining, Boston, USA, pp 71–80

Han JW, Kamber M (2001) Data mining: concepts and techniques. Morgan Kaufmann Publishers, San Francisco

Hu XH (1995) Knowledge discovery in databases: an attribute-oriented rough set approach. Ph.D. thesis, Regina University, Canada

Hu KY, Lu YC, Shi CY (2003) Feature ranking in rough sets. AI Commun 16(1):41–50

Hu QH, Yu DR, Liu JF, Wu CX (2008) Neighborhood rough set based heterogeneous feature subset selection. Inf Sci 178(18):3577–3594

Huang LJ, Huang MH, Guo B (2007) A new method for constructing decision tree based on rough set theory. In: Proceedings of 2007 IEEE international conference on granular computing, pp 241–244

Huang ZX, Lu XD, Duan HL (2010) Context-aware recommendation using rough set model and collaborative filtering. Artif Intell Rev 35(1):85–99

Jiang Y, Li ZH, Zhang Q, Liu Y (2004) New method for constructing decision tree based on rough set theory. Comput Appl 24(8):21–23

Jiang F, Sui YF, Cao CG (2009) Some issues about outlier detection in rough set theory. Expert Syst Appl 36(3):4680–4687

KDD Cup 99 Dataset (1999) Available online at: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
Li XP, Dong M (2008) An algorithm for constructing decision tree based on variable precision rough set model. In: Proceedings of the 4th international conference on natural computation, pp 280–283
Li XY, Ye N (2001) Decision tree classifiers for computer intrusion detection. J Parallel Distrib Comput Pract 4(2):179–190
Liu ZT (1999) An incremental arithmetic for the smallest reduction of attributes. Chin J Electron 27(11):96–98
Liu SH, Sheng QJ, Wu B, Shi ZZ (2003) Research on efficient algorithms for rough set methods. Chin J Comput 26(5):525–529
MacLeod C, Maxwell GM (2001) Incremental evolution in ANNs: neural nets which grow. Artif Intell Rev 16(3):201–224
Miao DQ, Hu GR (1999) An heuristic algorithm of knowledge reduction. Comput Res Dev 36(6):681–684
Nguyen HS, Nguyen SH (1998) Discretization methods in data mining. In: Polkowski L, Skowron A (eds) Rough sets in knowledge discovery. Physica, pp 451–482
Øhrn A (1999) Rosetta technical reference manual. Available online at: http://www.idi.ntnu.no/_aleks/rosetta
Pawlak Z (1982) Rough sets. Int J Comput Inf Sci 11(5):341–356
Pawlak Z (1991) Rough sets: theoretical aspects of reasoning about data. Kluwer Academic Publishing, Dordrecht
Quinlan R (1986) Induction of decision trees. Mach Learn 1(1):81–106
Quinlan R (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Francisco
Schlimmer JC, Fisher D (1986) A case study of incremental concept induction. In: Proceedings of the fifth national conference on artificial intelligence, pp 496–501
Shan N, Ziarko W (1993) An incremental learning algorithm for constructing decision rules. In: Rough sets, fuzzy sets and knowledge discovery. Springer, Heidelberg, pp 326–334
Utgoff PE (1989) Incremental induction of decision trees. Mach Learn 4(2):161–186
Utgoff PE (1994) An improved algorithm for incremental induction of decision trees. In: Proceedings of the 11th international conference on machine learning, pp 318–325
Wang GY (2001) Rough set theory and knowledge acquisition. Xian Jiaotong University Press, Xian
Wang GY, Yu H, Yang DC (2002) Decision table reduction based on conditional information entropy. Chin J Comput 25(7):759–766
Wei JM, Huang D, Wang SQ, Ma ZY (2002) Rough set based decision tree. In: Proceedings of the 4th world congress on intelligent control and automation, pp 426–431
Witten IH, Frank E (2000) Data mining: practical machine learning tools and techniques with Java implementations. Morgan Kaufmann, San Francisco
Wong AKC, Chiu DKY (1987) Synthesizing statistical knowledge from incomplete mixed-mode data. IEEE Trans Pattern Anal Mach Intell 9(6):796–805
Xu ZY, Liu ZP, Yang BR, Song W (2006) A quick attribute reduction algorithm with complexity of $\max(O(|C||U|), O(|C|^2|U/C|))$. Chin J Comput 29(3):391–399
Zheng Z, Wang GY (2004) RRIA: a rough set and rule tree based incremental knowledge acquisition algorithm. Fundamenta Informaticae 59(2/3):299–313