사물인터넷융복합논문지 Vol. 9, No. 1, pp. 77-87, 2023 https://doi.org/10.20465/KIOTS.2023.9.1.077

A New Association Rule Mining based on Coverage and Exclusion for Network Intrusion Detection

Tae Yeon Kim[1], KyungHyun Han[2], Seong Oun Hwang[3*]
[1]Student, Department of IT Convergence Engineering, Gachon University
[2]Student, Department of Electronics and Computer Engineering, Hongik University
[3]Professor, Department of Computer Engineering, Gachon University

네트워크 침입 탐지를 위한 Coverage와 Exclusion 기반의 새로운 연관 규칙 마이닝

김태연[1], 한경현[2], 황성운[3*]
[1]가천대학교 IT융합공학과 학생, [2]홍익대학교 전자전산공학과 학생, [3]가천대학교 컴퓨터공학과 교수

Abstract Applying various association rule mining algorithms to the network intrusion detection task involves two critical issues: too large size of generated rule set which is hard to be utilized for IoT systems and hardness of control of false negative/positive rates. In this research, we propose an association rule mining algorithm based on the newly defined measures called coverage and exclusion. Coverage shows how frequently a pattern is discovered among the transactions of a class and exclusion does how frequently a pattern is not discovered in the transactions of the other classes. We compare our algorithm experimentally with the Apriori algorithm which is the most famous algorithm using the public dataset called KDDcup99. Compared to Apriori, the proposed algorithm reduces the resulting rule set size by up to 93.2 percent while keeping accuracy completely. The proposed algorithm also controls perfectly the false negative/positive rates of the generated rules by parameters. Therefore, network analysts can effectively apply the proposed association rule mining to the network intrusion detection task by solving two issues.

Key Words : Network Intrusion Detection, Association Rule Mining, Measure

요 약 네트워크 침입 탐지 작업에 다양한 연관 규칙 마이닝 알고리즘을 적용하는 데에는 두 가지 중요한 문제가 있다. 생성된 규칙 집합의 크기가 너무 커서 IoT 시스템에서 활용하기 어렵고, 거짓 부정/긍정 비율을 제어하기 어렵다. 본 연구에서는 coverage와 exclusion이라는 새로 정의된 척도에 기반을 둔 연관 규칙 마이닝 알고리즘을 제안한다. Coverage는 한 클래스의 트랜잭션에서 패턴이 발견되는 빈도를 나타내고, exclusion은 다른 클래스의 트랜잭션에서 패턴이 발견되지 않는 빈도를 나타낸다. 우리는 KDDcup99라는 공개 데이터 세트를 사용하여 가장 유명한 알고리즘인 Apriori 알고리즘과 실험적으로 제안된 알고리즘을 비교한다. Apriori와 비교하여 제안된 알고리즘은 정확도를 완전히 유지하면서 생성되는 규칙 집합 크기를 최대 93.2%까지 줄인다. 또한, 제안된 알고리즘은 생성된 규칙의 거짓 부정/긍정 비율을 매개변수별로 완벽하게 제어한다. 따라서 네트워크 분석가는 두 가지 문제를 해결함으로써 제안한 연관 규칙 마이닝을 네트워크 침입 탐지 작업에 효과적으로 적용할 수 있다.

주제어 : 네트워크 침입 탐지, 연관 규칙 마이닝, 척도

This work was supported by the Gachon University research fund of 2021 (GCU-202104500001).
*교신저자 : 황성운(sohwang@gachon.ac.kr)
접수일 2022년 11월 6일 수정일 2022년 12월 16일 심사완료일 2022년 12월 18일

1. Introduction

Recently, works for applying machine learning to improve performance are increasing in various areas such as telecommunication networks, market analysis, risk management, and inventory control [1-4]. In particular, analysts in the network intrusion detection area apply association rule mining to find out patterns of normal and anomaly behaviors. It is because association rule mining algorithms generate frequent patterns in a form of rule. This is helpful for generating detection rules which are used in IDS (intrusion detection system).

Abnormal behavior detection research in IoT systems is becoming an important technology [5]. Association rule mining is a prominent method of discovering associations or rules among a set of available attributes in a dataset [6]. Deep learning is widely used recently, but association rule mining-based technology has been studied continuously because the information of associations or rules is useful for intrusion detection systems [7]. In applying association rule mining, we face two major issues. One is too large size of rule sets that association rule mining algorithms generate. Because of too large rule set, network analysts cannot use the rule set for generating detection rules. The other one is hardness of control of false negative/positive rates of rules. Although association rule mining algorithms output the rules, many rules are not useful, because they have lower performance than the other rules. These issues make it difficult for analysts to apply association rule mining to the network traffic dataset.

For solving issues as above, we newly define two measures: coverage and exclusion. The reason which we define new measures is that existing measures are not related to the performance of IDS. Our idea is to change helpless existing measures to helpful new ones so that the new algorithm based on the new measures can generate smaller rule sets that include the useful rules.

2. Related Work

2.1 Applications of association rule mining to the network intrusion detection task

In this section, we explain the papers that apply association rule mining to the network intrusion detection task.

In 2004, Ertoz et al. introduced the Minnesota Intrusion Detection System (MINDS) to detect network attack [8]. MINDS first detects abnormal attacks by clustering and making labeled dataset. And it summarizes attack traffics in the labeled dataset as detection rules by mining association rules. Association rule mining algorithms are helpful for generating new detection rules that may be used in intrusion detection module. Network analysts using this method select the rules whose performance is better among other generated rules.

In 2010, Miao et al. also introduced the Intrusion Detection System based on data mining [9]. Using anomaly-based intrusion detection, it learns user's characteristics and generates rules by Apriori with confidence in advance. So, it can detect the abnormal traffic which do not conform to rules.

In 2015, Khamphakdee et al. generated detection rules used in Snort by using association rule mining [10]. They employed the MIT-DARPA 1999 dataset as labeled dataset. In their experiment, the accuracy of generated rules was increased when the number of attributes in the dataset increases. They concluded that the number of attributes has to be increased to improve the accuracy of the generated rules.

The above papers used labeled datasets and algorithms that can set the class attribute, because all rules that do not have the values of classes are not useful in detecting attacks. Most of existing association rule mining algorithms consider both support and confidence. But these are not related to the performance of detection rules. They use the support measure, and use lower threshold for generating useful rules. In this case, they tend to generate too many rules, so analysts must manually and additionally select rules which have a high accuracy among the generated rules in each class.

2.2 Association rule mining algorithms

In 1994, Agrawal et al. proposed the Apriori algorithm for generating rules faster [11]. The Apriori algorithm finds frequent patterns among the patterns that are the combinations of from one item to all items by using the support measure that has the downward closure property which allows to prune the search space. This algorithm is very fast among association rule mining algorithms and is actively used until now. When Apriori is used network intrusion detection area, analysts use lower threshold for generating useful rules, which tends to generate too many rules.

In 2000, Han et al. proposed the FP-growth algorithm for generating rules faster [12]. The Apriori algorithm is fast when many generated rules are short. But there is a case which needs to generate long rules. In this case, the FP-growth algorithm generates rules starting from the longest pattern instead of the shortest pattern like Apriori.

In 2013, Gonzalez et al. proposed a new association rule mining algorithm [13]. When finding frequent patterns, the existing algorithms use the equality measure. But some values are not equal but similar, because some attributes have continuous variables. So, they use similarity instead of equality and their method has the downward closure property. Due to these characteristics, this algorithm can generate rules which have a higher quality.

Many algorithms like above use the support measure because it has the downward closure property, which enables algorithms to prune the search space. But as we mentioned above, the support measure itself is not appropriate measure for network analysts to use. Therefore, we need an alternative measure to satisfy the downward closure property instead of the support measure.

2.3 Measures of association rule mining

In this section, we explain measures that are frequently used in the association rule mining algorithms.

In 1993, Agrawal et al. proposed a measure called confidence [14]. The confidence measure is defined as the ratio of the number of transactions containing the rule's consequent to the number of transactions containing the antecedent. This measure was developed together with the support measure and have been heavily used. It is because confidence is useful considering class information and reducing the number of rules.

In 2007, Hahsler pointed out that the confidence and the lift measures generally used in association rule mining are not suitable for processing random noise [15]. Based on a probabilistic framework, he proposed new measures such as hyper-lift and hyper-confidence for processing random noises. He showed that he could reduce the rule set size by selecting better rules even though the underlying the dataset contains random noises.

In 2014, Benites et al. proposed new measures which can efficiently reduce the size of rule sets in a hierarchically structured dataset [16]. In the particular case of hierarchically organized items and generalized association rules connecting them, their measures that deal appropriately with the hierarchy would be advantageous. The above measures do not satisfy the download closure property. In addition, they do not allow to compute accuracy such as true positive rate and false positive/negative rate. Therefore, they are not appropriate for network detection purpose, either.

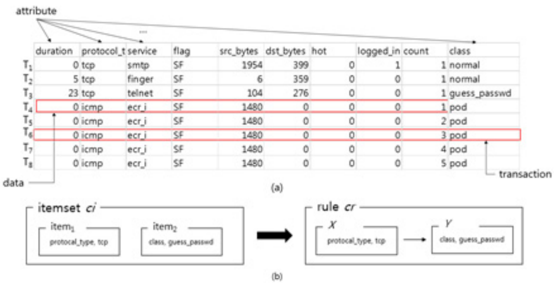
3. Proposed Method

3.1 Measures of association rule mining

We explain the basic concept of Apriori and the existing measures first. Next, we explain the newly defined two measures. Note that Apriori here is a modified version which combines confidence.

3.1.1 Basic concept and existing measures

As shown in Fig. 1a), in the network traffic dataset, an attribute is a property that analysts are interested in the network traffic such as duration, protocol type, service, class. A value represents what an attribute has (e.g. protocol type has values such as 'tcp', 'icmp', and so on). A class is a special kind of attribute denoting attack types such as normal, guess_passwd and pod. A data is a set of values corresponding to a given attribute set except a class value (e.g. (0, icmp, ecr_i, SF, 1480, 0, 0, 0, 2)). A transaction is a data with a class value added (e.g. (0, icmp, ecr_i, SF, 1480, 0, 0, 0, 2, pod)). As shown in Fig. 1b), an item is a combination of an attribute and its value (e.g. (protocol type, tcp)). An itemset is a set of items. A pattern is an element of power set of items which are in common in various data (e.g. {(duration, 0), (protocol type, icmp), (service, ecr_i), (flag, SF), (src_bytes, 1480), (dst_bytes, 0), (hot, 0), (logged_in, 0)}). A labeled dataset is a set of transactions.



[Fig. 1] Relationship of terminologies. a) labeled dataset; b) item, itemset and rule.

The Apriori algorithm consists of two phases:

- The first phase finds frequent patterns from an input labeled dataset.
- The second phase outputs rules from the frequent patterns found above.

The Apriori algorithm extracts all items from an input labeled dataset and then combines them into itemsets in the first phase. By calculating the support of each itemset, it finds out itemsets which frequently happen in the dataset. Support for a specific itemset is defined as the ratio of the number of transactions which contain the specific itemset to the number of entire transactions. That is, support shows how frequently each itemset happens among all transactions. Table 1 defines the used notations. Support of an itemset *ci* is computed as *ci.count / |D|*. Taking the example itemset ci as shown in Fig. 1b), support of *ci* is computed as 0.125 (=1/8) because *ci* happens in one transaction (T3) and therefore *ci.count* = 1. The second phase transforms each itemset as a form of X→Y (i.e., if X, then Y). Here, the class value is transformed into Y and the other remaining items are transformed into X. This rule means that if X is detected in data, the data is classified into Y. In this phase, it calculates the confidence of each rule and evaluates its accuracy [8, 10]. Confidence for a specific rule is defined as the ratio of the number of transactions containing the rule's X and Y to the number of transactions containing the rule's X. That is, confidence of a rule *cr* is computed as *cr.count / cr.d_count*. Taking the example rule *cr* as shown in Fig. 1b), *cr.count* = 1 because *cr.count* equals the *ci.count* of itemset *ci* which generates the *cr*. Since X in *cr* detects three transactions (T1, T2, T3), *cr.d_count* = 3. Therefore, confidence of *cr* is computed as 0.333 (=1/3). By applying the generated rules to network traffic as above, we can detect attacks. We can see that cr in Fig. 1b) detects three transactions (T1, T2, T3) as the class of guess_passwd.

〈Table 1〉 Notations

Notation	Meaning
$\|A\|$	Number of transactions contained in A
D	Set of all transactions
D_j	Set of transactions in j-th class
$ci.count$	Number of transactions containing itemset ci
$cr.count$	Number of transactions exactly detecting rule cr
$cr.d\_count$	Number of transactions detecting rule cr
$S_{max}(c_j)$	The maximum support among the supports of the rules with the highest true positive rate in j-th class
S_{min}	The minimum support among $S_{max}(c_j)$ of all classes (excluding normal class)
$Co_{max}(c_j)$	The maximum confidence among the confidences of the rules with the highest true positive rate in j-th class
Co_{min}	The minimum confidence among $C_{max}(c_j)$ of all classes (excluding normal class)
$C_{max}(c_j)$	The maximum coverage among the coverages of the rules with the highest true positive rate in j-th class
$E_{max}(c_j)$	The maximum exclusion among the exclusions of the rules with the highest true positive rate in j-th class
E_{min}	The minimum exclusion among $E_{max}(c_j)$ of all classes (excluding normal class)

3.1.2 Proposed measures: coverage and exclusion

In the network intrusion detection, analysts put much emphasis on false negative/positive rates of rules, which relate to the performance of an intrusion detection system. But support and confidence are not related to false negative/positive rates. So, we define new measures related to false negative/positive rates as follows.

Definition 1. (Coverage) Coverage in a specific itemset is defined as the ratio of the number of transactions related to a given itemset to the number of transactions containing its relevant class. It shows how frequently each itemset is discovered in the transactions of a class. That is, the coverage of an itemset ci in j-th class is computed as $ci.count\ /\ |D_j|$. Here we define the coverage as 1 in case there exists no item for the class in the itemset.

Definition 2. (Exclusion) In a given rule, exclusion is defined as the ratio of the number of transactions which contain neither X nor Y to the number of transactions which do not contain Y. That is, the exclusion of a rule cr is computed as $1 - ((cr.d\_count - cr.count) / (|D| - |D_j|))$.

3.1.3 Analysis of proposed measures

Proposition 1. Coverage meets the downward closure property.

Proof. We say that a measure has the downward closure property if all measures of $(k-1)$-itemsets are greater than or equal to those of k-itemsets, where k-itemsets can be made from $(k-1)$-itemsets [11]. Note that the set of transactions containing k-itemsets is a subset of the set of transactions containing $(k-1)$-itemsets. Therefore, the number of transactions containing k-itemsets is less than or equal to the number of transactions containing $(k-1)$-itemsets. Since the number of transactions of relevant class is fixed, the coverage of k-itemsets is less than or equal to that of $(k-1)$-itemsets.□

Property 1. The proposed algorithm can control the false negative rates of the generated rules.

Previously, we defined the coverage in a specific itemset as the ratio of the number of transactions related to a given itemset to the number of transactions containing its relevant class. When applying the given rule to a test dataset, true positive rate is the ratio of the number of detected data in a class to the number of data in the class. Under the assumption that input datasets are similar to test datasets, which is generally accepted in machine learning, coverage is identical with true positive rate. Note that false negative rate is 1 − true positive rate. In this way, if the coverage of each rule increases, then the false negative rate of the relevant rule decreases. Therefore, we can control the false negative rate(s) by changing the coverage threshold value(s).

Property 2. The proposed algorithm can control the false positive rates of the generated rules.

Previously, we defined the exclusion in a specific rule as the ratio of the number of transactions which contains neither X nor Y to the number of transactions which do not contain Y. When applying the given rule to a test dataset, true negative rate is the ratio of the number of undetected data out of a class to the number of data out of the class. Under the assumption that input datasets are similar to test datasets, which is generally accepted in machine learning, exclusion is identical with true negative rate. Note that false positive rate is 1 − true negative rate. In this way, if the exclusion of each rule increases, then the false positive rate of the relevant rule decreases. Therefore, we can control the false positive rate(s) by changing the exclusion threshold value(s).

Note that Proposition 1 is important, because coverage is required to satisfy the downward property which allows to prune the search space. When pruning the search space, it is important to reserve desired rules. In network intrusion detection, desired rules are ones which minimize false negative/positive rate. Coverage can control the false negative rate as Property 1 shows. Therefore, the proposed algorithm removes only undesired rules by using coverage.

3.2 Proposed algorithm

Apriori uses support and confidence to generate rules. The proposed measures are also used in generating rules. Therefore, we construct the proposed algorithm by replacing support and confidence in Apriori with coverage and exclusion, respectively, while leaving the remaining parts such as generation of itemsets or rules unchanged. The reason we use Apriori is because it is a representative and popular algorithm in association rule mining.

3.2.1 Explanation of proposed algorithm

Our algorithm takes a labeled dataset as input, like the existing ones, and produces rules as output as shown Fig. 2. The proposed algorithm consists of two phases:

```
1) Input: dataset: D = D_1, D_2, D_3, …, D_n, Cov_tv for each class: mincov[n], Exc_tv: minexc
2) Output: rule set: R
3) L_1 = {large 1-itemsets};
4) for ( k = 2; L_{k−1} = ∅; k ++) do begin
5)     CI_k = apriori-gen(L_{k−1}); // New candidates of k-itemset
6)     for ( j = 1; j ≤ n; j ++ ) do begin // n : the number of class
7)         forall transactions t ∈ D_j do begin
8)             CI_t = subset-i(CI_k, t); // Candidates of itemset contained in t
9)             forall candidates ci ∈ CI_t do ci.count ++;
10)        end
11)        L_k = L_k + { ci ∈ CI_k | ci.count / |D_j| ≥ mincov[j], class-check(ci) == j}; // coverage
12)    end
13)    L_k = L_k + { ci ∈ CI_k | class-check(ci) == 0};
14) end
15) L = ∪_{j=1}^{k−1} L_j;
16) CR = rule-gen(L); // New candidates of rule
17) forall transactions t ∈ D do begin
18)     CR_t = subset-r(CR, t); // Candidates of rule contained in t
19)     forall candidates cr ∈ CR_t  do cr.d_count ++;
20) end
21) for ( j = 1; j ≤ n; j ++ ) do begin
22)     R = { cr ∈ CR | 1 − ((cr.d_count − cr.count ) / (|D| − |D_j|)) ≥ minexc, class-check(cr) == j}; // exclusion
23) end
24) Answer = R;
```

[Fig. 2] Proposed algorithm

- The first phase is used to find frequent patterns based on coverage (lines 3-15).
- The second phase is used to output rules from frequent patterns based on exclusion (lines 16-24).

It takes dataset *D*, *mincoverage[n]* (coverage threshold value for each class), and *minexclusion* (exclusion threshold value) as input, and outputs *R*, a set of rules. Each class in the dataset is assigned a number starting from *1* up to *n*, the number of all classes. D_j denotes the set of transactions in the *j*-th class. L_k denotes a set of *k*-itemsets (i.e., itemset consisting of *k* items) whose coverages are greater than or equal to the coverage threshold value designated by analysts. CI_k denotes a set of *k*-itemsets which are combinations of *(k-1)*-itemsets in L_{k-1}. It means that k-itemsets in CI_k are candidate *k*-itemsets in L_k.

The functions used in the algorithm are as follows:

- apriori-gen() takes L_{k-1} as input, combines all the *(k-1)*-itemsets in L_{k-1} and outputs CI_k, the set of *k*-itemsets.
- subset-i() takes both CI_k and transaction *t* as input, and outputs CI_t, which consists of only items in *t* among the itemsets in CI_k.
- class-check() takes an itemset *ci* or a rule *cr* as input, and outputs its corresponding class number. In case class information is

not available with input itemsets, it outputs 0.

- rule-gen() takes L, the set of all generated itemsets, transforms itemsets in L to rules and outputs the resulting rule set CR. While transforming, itemsets whose transformed rules do not have the Y part are removed. Even after the transformation process, *ci.count* of each itemset in L is kept as *cr.count* in the relevant rule in CR.
- subset-r() takes both CR and t, and outputs CR_t, the set of rules which detect t among the rules in CR.

In the first phase of the algorithm, it sets coverage of all items as 1 in the input labeled dataset D, generates L_1 (line 3) and repeats the following three tasks, starting from k = 2 up to the time L_{k-1} becomes the empty set (lines 4-14): (i) It generates CI_k, a candidate set of k-itemsets, from L_{k-1} (line 5); (ii) It computes *ci.count*, the number of transactions where each itemset *ci* is relevant (lines 7-10); (iii) Based on the coverage, it generates L_k, the set of k-itemsets whose coverage is greater than *mincoverage[j]* (lines 6-13), where L_k contains all *ci*'s which meet the aforementioned coverage threshold value condition in the generated CI_k. In CI_k, k-itemsets with class information are included in L_k (line 11) and k-itemsets without class information are included in L_k (line 13). Finally, all itemsets belonging to L_1 through L_{k-1} are combined into L, the set of itemsets (line 15).

In the second phase of the algorithm, it starts transforming all the previously generated itemsets into CR, the set of candidate rules (line 16). Particularly in case there exists a class designated by analysts, it makes the item for the class into Y and the other remaining items into X. Next, it computes *cr.d_count*, the number of transactions which are detected by X in each rule of the generated CR (lines 17-20). Finally, it generates a rule set R, which contains all the rules satisfying the aforementioned exclusion threshold value condition in the generated CR (lines 21-24).

3.2.2 Analysis of proposed algorithm

Property 3. The proposed algorithm keeps the same accuracy rate with Apriori.

Rules to detect transactions in j-th class have higher accuracy measures as they exactly detect transactions whose number comes closer to $|D_j|$. Note that the way rules are generated in Apriori is the same as ours. It is because the proposed algorithm was achieved by replacing support and confidence in the Apriori algorithm with coverage and exclusion, respectively, while leaving generation of itemsets or rules unchanged. Both algorithms generate rules to exactly detect transactions of the number close to $|D_j|$ in j-th class. Therefore, we can conclude that the proposed algorithm keeps the same accuracy rate with Apriori.

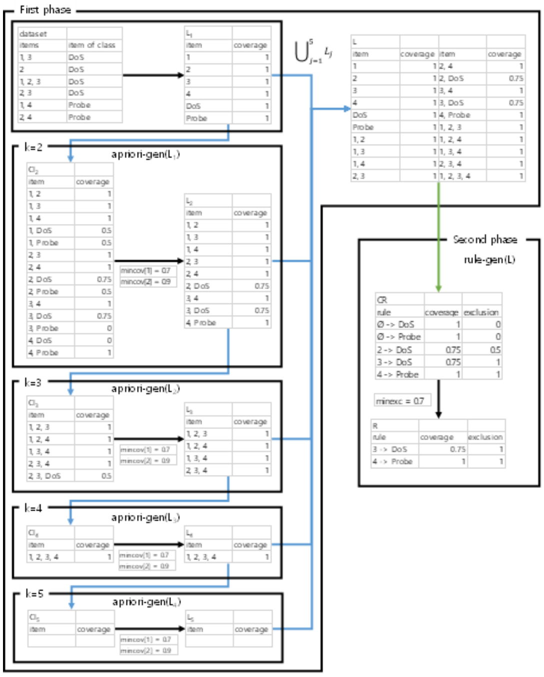
Property 4. The proposed algorithm reduces the generated rule set size compared to Apriori.

When applying association rule mining algorithms, support and coverage primarily have influence on the number of generated rules. Support shows how much data a rule is related to in a given entire data set. In some class with a small data set, the support threshold should be lower, which results in a larger rule set. However, coverage shows how much data a rule is related to in a given class. Therefore, the proposed algorithm can reduce the generated rule set size by setting appropriate coverage threshold per class.

3.3 An example

We show how this algorithm works by taking an example.

Consider the example in Fig. 3. We set the coverage threshold values as *mincoverage[1]* = 0.7 for DoS and *mincoverage[2]* = 0.9 for Probe, and the exclusion threshold value as *minexclusion* = 0.7. The first phase starts generating L_1 by



[Fig. 3] Example to help understand the proposed algorithm

extracting all items from the input labeled dataset. When k = 2, it generates CI_2, the set of 2-itemsets by combining 1-itemsets in L_1. For the example {2, DoS} in CI_2, the *ci.count* of {2, DoS} is 3, because {2, DoS} is relevant to three transactions ({2, DoS}, {1, 2, 3, DoS}, {2, 3, DoS}), and $|D_1|$ is 4, because the number of transactions in DoS class is 4. Therefore, the coverage of {2, DoS} is computed as 0.75 (=3/4). Next, it generates L_2, the set of 2-itemsets, which consists of itemsets in CI_2 with the coverage greater than or equal to the specified *mincoverage* of its related class. This process repeats as k increments and terminates when L_{k-1} becomes the empty set. In the example, when k = 6, L_5 becomes the empty set and all the itemsets in L_1 through L_5 belong to L. The second phase generates CR, a set of rules corresponding to itemsets in L. Note that ci.count of each itemset in L is kept as cr.count in the relevant rule in CR. Next, we compute exclusion for each rule in CR as follows. For the example {2→DoS} in CR, the *cr.d_count* of {2→DoS} is 4, because {2} detects four transactions ({2, DoS}, {1, 2, 3, DoS}, {2, 3, DoS}, {2, 4, Probe}), and *cr.count* of {2→DoS} is 3, because *ci.count* of {2, DoS} is kept as *cr.count*, $|D|$ is 6, and $|D_1|$ is 4. Therefore, the exclusion of {2→DoS} is computed as 0.5 (=1-(4-3)/(6-4)). Finally, from the generated CR, it generates a rule set R, which consists of rules with exclusion greater than or equal to the *minexclusion* of 0.7.

Unlike from the above example, real datasets surely have very large data. Therefore, association rule mining algorithms will generate much more diverse rules. Analysts prefer rules with lower false negative/positive rates among the generated rules. Our algorithm enable them to set up false negative/positive rates first and then get the reduced sized rule set accordingly.

4. Experiments

We will show how the proposed algorithm can resolve the aforementioned issues. First, in Experiment 1, we will find threshold values of four kinds of measures. Next, in Experiment 2, we will apply the found threshold values to Apriori and the proposed algorithm to generate rules, respectively.

Publicly available datasets widely used in the network intrusion detection area include DARPA

⟨Table 2⟩ Found measures

	$\|D_i\|$	S_{max} (c_j)	Co_{max} (c_j)	C_{max} (c_j)	E_{max} (c_j)
Normal	952	0.1039	0.9922	0.6691	0.9990
Guess_passwd	53	0.0086	0.9138	1.0000	0.9992
Nmap	231	0.0168	1.0000	0.4459	1.0000
Pod	264	0.0423	1.0000	0.9811	1.0000
Portsweep	1040	0.1266	1.0000	0.7462	1.0000
Satan	1589	0.2081	1.0000	0.8024	1.0000
Teardrop	979	0.1598	1.0000	1.0000	1.0000
Warezclient	1020	0.1164	0.9532	0.6990	0.9931
Minimum	53	0.0086	0.9138	0.4459	0.9931

〈Table 3〉 Comparison between Apriori and the proposed method

	Apriori (Support: 0.0086, Confidence: 0.9138)					Proposed algorithm (Coverage: set for each class, Exclusion: 0.9931)				
Class [Coverage]	No. of rules	TPR	FNR	FPR	F1-measure	No. of rules	TPR	FNR	FPR	F1-measure
Normal [0.6691]	1708	0.6691	-	-	-	2	0.6691	-	-	-
Guess_passwd [1.0000]	2	1.0000	0.0000	0.0008	0.9550	2	1.0000	0.0000	0.0008	0.9550
Nmap [0.4459]	640	0.4459	0.5541	0.0000	0.6168	128	0.4459	0.5541	0.0000	0.6168
Pod [0.9811]	608	0.9811	0.0189	0.0000	0.9904	192	0.9811	0.0189	0.0000	0.9904
Portsweep [0.7462]	810	0.7462	0.2538	0.0000	0.8546	16	0.7462	0.2538	0.0000	0.8546
Satan [0.8024]	752	0.8024	0.1976	0.0000	0.8904	2	0.8024	0.1976	0.0000	0.8904
Teardrop [1.0000]	876	1.0000	0.0000	0.0000	1.0000	128	1.0000	0.0000	0.0000	1.0000
Warezclient [0.6990]	1564	0.6990	0.3010	0.0069	0.8066	2	0.6990	0.3010	0.0069	0.8066
Sum	6960	-	-	-	-	472	-	-	-	-
Average	-	0.8106	0.1894	0.0011	0.8734	-	0.8106	0.1894	0.0011	0.8734
Maximum	-	1.0000	0.5541	0.0069	1.0000	-	1.0000	0.5541	0.0069	1.0000
Minimum	-	0.4459	0.0000	0.0000	0.6168	-	0.4459	0.0000	0.0000	0.6168

98, KDDcup99 [17], and NSL-KDD [18]. In the KDDcup99 dataset, various attributes were already extracted enough to classify each attack and transactions are classified according to various attacks. We used the KDDcup99 dataset.

The best threshold values are different depending on each dataset. We find the best threshold value which will be used to generate the selected rules in next experiment.

Table 2 shows the values of selected rules in each class. The minimum values of support, confidence and exclusion in Table 2 will be used as threshold values in next experiment, because rules with higher value than threshold are generated. All values of coverage in Table 2 will be used as threshold, because coverage can be set per each class. Note that the other three measures can be set per dataset.

Now we perform Experiment 2 and analyze the result.

Table 3 shows the number of generated rules and the best accuracy measures (i.e. TPR, FNR, FPR, F1-measure) of the rules per each class. As we can see in Table 3, the proposed algorithm reduces the resulting rule set size by 93.2 percent from 6960 to 472 while keeping the same accuracy measures compared to Apriori.

Table 3 also shows the set threshold values in the proposed algorithm and the false negative/positive rates of the generated rules. In each class, two conditions “false negative rate ≤ 1 − coverage” and “false positive rate ≤ 1 − exclusion” are satisfied. This means that the false negative/positive rates can be controlled by coverage and exclusion.

5. Conclusion

Applying association rule mining to the network traffic analysis involves critical issues such as too large size of generated rule set and hardness of control of false negative/positive rates. To address these issues, we proposed a new association rule mining algorithm by newly defining measures such as coverage and exclusion. Compared to Apriori, we showed experimentally that it reduces the resulting rule set size by up to 93.2 percent and controls the false negative/positive rates. In addition, we also showed theoretically that it can reduce the resulting rule set size while keeping the Apriori's accuracy. Therefore, analysts can effectively apply the proposed association rule mining to the network intrusion detection task.

The core part of the proposed method is measures, which serve as generating rule set upon the controlled accuracy rate as well as reducing the resulting rule set size. The network intrusion detection area puts much value on both the accuracy rate and the size of rule set at the same time. In the market analysis area, accuracy is considered more important, while reducing the rule set size is so in the text analysis area. In next research, we will apply our proposed method to these areas.

Currently, all the existing association rule mining algorithms including the proposed one are batch styled. But these algorithms are hard to be applied in data stream or big data environments where data items are continuously added to dataset over time. Another future work is to adapt the proposed algorithm to incremental learning one.

REFERENCES

[1] A.S.Sadh and N.Shukla, "Association Rules Optimization: A Survey," Int. J. of Advanced Comput. Res., Vol.3, No.1, pp.111-115, 2013.

[2] G.J.Simon, P.J.Caraballo, T.M.Therneau, S.S.Cha, M.R.Castro, and P.W.Li, "Extending Association Rule Summarization Techniques to Assess Risk of Diabetes Mellitus," IEEE Trans. Knowl. Data Eng., Vol.27, No.1, pp.130-141, 2015.

[3] I.F.Videla-Cavieres and S.A.Rios, "Extending market basket analysis with graph mining techniques: A real case," Expert Syst. Appl., Vol.41, No.4, pp.1928-1936, 2014.

[4] A.C.Squicciarini, D.Lin, S.Sundareswaran, and J.Wede, "Privacy Policy Inference of User-Uploaded Images on Content Sharing Sites," IEEE Trans. Knowl. Data Eng., Vol.27, No.1, pp.193-206, 2015.

[5] K.H.Lee, "A Scheme on Anomaly Prevention for Systems in IoT Environment," Journal of The Korea Internet of Things Society, Vol.5, No.2, pp.95-101, 2019.

[6] I.H.Sarker, A.I.Khan, Y.B.Abushark, and F.Alsolami, "Internet of things (iot) security intelligence: a comprehensive overview, machine learning solutions and research directions," Mobile Networks and Applications, pp.1-17, 2022.

[7] D.Sellappan, and R.Srinivasan, "Association rule-mining-based intrusion detection system with entropy-based feature selection: Intrusion detection system," In Handbook of Research on Intelligent Data Processing and Information Security Systems, pp.1-24, 2020.

[8] L.Ertoz, E.Eilertson, A.Lazarevic, P.N.Tan, V.Kumar, J.Srivastava, and P.Dokas, "MINDS - Minnesota Intrusion Detection System," Next Generation Data Mining, MIT Press, 2004.

[9] C.Miao, and W.Chen, "A Study of Intrusion Detection System Based on Data Mining," 2010 IEEE Int. Conf. on Inform. Theory and Inform. Security, pp.186-189, 2010.

[10] N.Khamphakdee, N.Benjamas, S.Saiyod, "Improving Intrusion Detection System Based on Snort Rules for Network Probe Attacks Detection with Association Rules Technique of Data Mining," J. ICT Res. Appl., Vol.8, No.3, pp.234-250, 2015.

[11] R.Agrawal and R.Srikant, "Fast Algorithms for Mining Association Rules," Proc. 20th VLDB Conf., Vol.1215, pp.487-499, 1994.

[12] J.Han, J.Pei, and Y.Yin, "Mining Frequent Patterns without Candidate Generation," ACM Sigmod Rec., Vol.29, No.2, pp.1-12, 2000.

[13] A.Y.R.Gonzalez, J.F.Martinez-Trinidad, J.A.Carrasco-Ochoa, J.Ruiz-Shulcloper, "Mining frequent patterns and association rules using similarities," Expert Syst. Appl., Vol.40, pp.6823-6836, 2013.

[14] R.Agrawal, T.Imielinski, and A.Swami, "Mining association rules between sets of items in large databases," ACM Sigmod Rec., Vol.22, No.2, pp.207-216, 1993.

[15] M.Hahsler, "New Probabilistic Interest Measures for Association Rules," Intelligent Data Anal., Vol.11, No.5, pp.437-455, 2007.

[16] F.Benites and E.Sapozhnikova, "Evaluation of Hierarchical Interestingness Measures for Mining Pairwise Generalized Association Rules," IEEE Trans. Knowl. Data Eng., Vol.26, No.12, pp.3012-3025, 2014.

[17] University of California, Irvine, "KDD Cup 1999 Data," https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[18] University of New Brunswick, "The NSL-KDD Data Set," https://web.archive.org/web/20150205070216/http://nsl.cs.unb.ca/NSL-KDD/

김 태 연(Tae Yeon KIM) [준회원]

■ 2021년 2월 : 홍익대학교 컴퓨터 정보통신공학과(공학학사)
■ 2021년 3월 ~ 현재 : 가천대학교 일반대학원 IT융합공학과 (공학 석사과정)

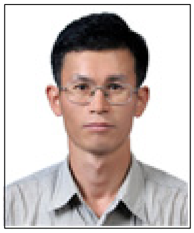
〈관심분야〉
딥러닝, 경량화, 네트워크보안

한 경 현(KyungHyun Han) [정회원]

■ 2015년 2월 : 홍익대학교 컴퓨터 정보통신학과(공학학사)
■ 2017년 2월 : 홍익대학교 일반대학원 전자전산공학과 (공학석사)
■ 2017년 3월 ~ 현재 : 홍익대학교 일반대학원 전자전산공학과 (공학 박사과정)

〈관심분야〉
사이버보안

황 성 운(Seong Oun Hwang) [정회원]



■ 1993년 8월 : 서울대학교 수학과 (이학사)
■ 1998년 2월 : 포항공과대학교대학원 정보통신학과 (공학석사)
■ 2004년 8월 : 한국과학기술원 전자전산학과 (공학박사)
■ 2006년 1월 ~ 2006년 12월 : University of Michigan 박사 후 연구원
■ 2008년 3월 ~ 2020년 2월 : 홍익대학교 컴퓨터공학과 교수
■ 2020년 3월 ~ 현재 : 가천대학교 컴퓨터공학과 교수

〈관심분야〉
정보보호, 사이버보안, 기계학습