캘린더 패턴 기반의 시간 연관적 분류 기법
(Temporal Associative Classification based on Calendar Patterns)

이 헌 규 † 노 기 용 †† 서 성 보 ††† 류 근 호 ††††
(Heon Gyu Lee) (Gi Young Noh) (Sungbo Seo) (Keun Ho Ryu)

요 약 시간 데이타마이닝은 기존 데이타마이닝에 시간 개념을 추가하여 시간 속성을 가진 데이타로부터 이전에 잘 알려지지는 않았지만 묵시적이고 잠재적으로 유용한 시간 지식을 탐사하는 기술이다. 대표적 데이타마이닝 기법인 연관규칙과 분류기법은 실세계의 여러 응용분야에서 사용된다. 그러나 대부분의 데이타가 시간 속성을 포함함에도 불구하고 기존의 기법들은 시간 속성을 고려하지 않고 주로 정적인 데이타에 대한 지식 탐사만이 진행되었다. 그리고 시간 데이타에 대한 데이타마이닝 연구들은 데이타의 발생 시점과 시간 제약조건을 추가한 지식 탐사에 중점을 두고 있어 데이타가 포함한 시간 의미나 시간 관계를 탐사하는데 부족하였다. 이 논문에서는 시간 클래스 연관규칙에 기반한 시간 연관적 분류기법을 제안한다. 이 기법은 분류규칙 생성을 위해서 연관적 분류에 시간 차원을 포함하여 확장한 시간 클래스 연관규칙에 의해 탐사된 규칙들을 적용하는 것이다. 그러므로 이 기법은 기존의 분류 기법들에 비해 더 유용한 지식 탐사가 가능하다.

키워드 : 시간 데이타마이닝, 시간 연관적 분류, 시간 클래스 연관규칙

Abstract Temporal data mining, the incorporation of temporal semantics to existing data mining techniques, refers to a set of techniques for discovering implicit and useful temporal knowledge from temporal data. Association rules and classification are applied to various applications which are the typical data mining problems. However, these approaches do not consider temporal attribute and have been pursued for discovering knowledge from static data although a large proportion of data contains temporal dimension. Also, data mining researches from temporal data treat problems for discovering knowledge from data stamped with time point and adding time constraint. Therefore, these do not consider temporal semantics and temporal relationships containing data. This paper suggests that temporal associative classification technique based on temporal class association rules. This temporal classification applies rules discovered by temporal class association rules which extends existing associative classification by containing temporal dimension for generating temporal classification rules. Therefore, this technique can discover more useful knowledge in compared with typical classification techniques.

Key words : Temporal data mining, Temporal Associative classification, Temporal Class Association rules

1. 서 론

데이타마이닝은 데이타베이스로부터 이전에 잘 알려지지는 않았지만, 묵시적이고 잠재적으로 유용한 지식을 추출하는 기술이며, 조직의 의사결정 지원 시스템, 전자상거래, 침입탐지를 위한 감사데이타 분석 등 다양한 분야에서 유용하게 사용된다. 데이타마이닝 기술 중 가장 많이 연구된 기법으로는 연관규칙과 분류기법을 들 수 있다. 그러나 기존의 연구들은 대부분 데이타가 시간 속성을 가졌음에도 불구하고 시간 속성을 간과하여 정적인 지식만을 추출하였다. 이는 데이타마이닝 기법 적용 결과 탐사된 지식이 아무리 정확하고 유용하더라도 그 지식을 적용할 시점을 표현하지 않으므로 모호한 지식이라 할 수 있다. 일부, 부분적으로 시간 지식 개념을

· 이 연구는 산업자원부 · 한국산업기술평가원 지정 청주대학교 정보통신연구센터의 지원에 의한 것입니다.
† 비 회 원 : 충북대학교 전자계산학과
hglee@dblab.cbu.ac.kr
†† 비 회 원 : 한국표준과학연구원
kyno@kriss.re.kr
††† 학생회원 : 충북대학교 전산학과
sbseo@dblab.cbu.ac.kr
†††† 종신회원 : 충북대학교 컴퓨터과학과 교수
khryu@dblab.chungbuk.ac.kr
논문접수 : 2004년 4월 30일
심사완료 : 2005년 8월 11일

적용한 연구들 또한 진행되었으나 이러한 연구들에서는 시간을 고려하였다 하더라도 데이타의 발생시점과 시간 제약조건을 추가한 형태로 다루고 있어 실제로 데이타가 포함하고 있는 시간 의미(*semantics*)와 시간 관계(*relationship*)를 탐사하는데 부족하였다. 시간 데이타마이닝은 이러한 기존 연구의 문제점을 해결하기 위해서 기존 데이타마이닝에 시간 개념을 추가하여 시간의 의미와 관계를 가지는 시간 지식을 탐사하기 위한 새로운 데이타마이닝 연구 분야이다. 따라서 시간 데이타마이닝을 통해 시간 데이타들로부터 다양한 형태의 지식을 탐사할 수 있는 것이다. 예를 들어, "금요일 자정에 특정 근원지 IP에서 목적지 IP로의 연결은 침입이다."라는 달력 형식의 시간을 표현하는 캘린더 패턴의 시간 규칙을 탐사할 수 있는 것이다.

이 논문에서는 실세계의 시간 속성을 가진 데이타에서 시간의 변화에 따른 적용 시점이 명확한 지식 탐사가 가능하고, 미래의 예측에 있어 탐사된 규칙과 시간 지식을 이용함으로써 기존의 정적인 분류규칙을 적용한 방법보다 더 정확한 예측을 할 수 있는 새로운 시간 연관적 분류 기법을 제안한다. 논문은 다음과 같은 내용으로 구성된다.

- 시간 연관적 분류규칙은 연관규칙과 분류기법이 통합된 기존의 연관적 분류기법에 시간 속성을 가지는 데이타에 대해서 시간 지식을 포함한 규칙 탐사가 가능하도록 확장한다. 이를 위해서, 다양한 시간 단위에서의 시간 간격과 주기적인 패턴의 표현이 가능하고, 탐사된 지식을 사용자가 쉽게 이해할 수 있도록 달력의 표현방식을 적용한 캘린더 기반의 시간 패턴을 이용한다.
- 캘린더 패턴 표현의 연관적 분류규칙을 기반으로 하여 양질의 분류규칙들만을 추출하기 위한 규칙제거(*rule pruning*) 방법 및 알고리즘을 제안하고 구현한다.
- 제안된 시간 연관적 분류기법의 알고리즘들의 검증과 평가를 위해서 네트워크 상에서의 침입여부를 판단하기 위한 감사데이타의 분석에 이를 적용한다.

논문의 효과적인 이해를 위해서 이 논문의 구성은 다음과 같이 구성하였다. 2장에서는 시간 연관규칙, 연관적 분류 그리고 침입탐지를 위한 데이타마이닝에 대한 정리와 기존 기법들의 문제점들을 분석한다. 3장에서는 기존의 연관적 분류기법에 논문에서 제안된 캘린더 패턴식을 적용한 캘린더 기반 시간 클래스 연관규칙 탐사의 문제정의 및 알고리즘을 기술한다. 4장에서는 시간 클래스 연관규칙의 탐사 결과 생성된 규칙들로부터 다른 시간 간격에서의 다른 연관적 분류규칙들을 생성하는 과정을 설명한다. 5장에서는 제안한 시간 연관적 분류기법의 알고리즘을 구현하여 실험한 후 성능 평가 및 결과를 분석한다. 마지막으로 6장에서는 이 논문에 대한 전체적인 결론과 보완해야 될 문제점 및 향후연구 방향을 제시한다.

2. 관련연구

2.1 시간 연관규칙

시간 연관규칙이란 기존 연관규칙에 시간 개념을 추가하여 시간 데이타로부터 시간 의미와 시간 관계를 가지는 유용한 지식을 탐사는 기법이다[1]. 시간 속성을 가지는 연관규칙에 대한 이전 연구들은 크게 주어진 시간 간격 동안 주기적으로 발생되는 현상, 즉 시간 간격에서의 완전한 주기성을 만족하는 연관규칙을 탐사하는 주기적 연관규칙 탐사[2]와 캘린더로 표현된 시간 패턴을 가지는 연관규칙을 탐사하는 캘린더 기반 연관규칙 탐사[3-6]로 분류할 수 있다.

2.1.1 주기적 연관규칙

주기적 연관규칙이란 트랜잭션이 발생한 전체 시간을 사용자에 기반한 시간 단위(년, 월, 일)에 따라 시간 간격의 집합으로 나누고, 시간 구간에서의 완전한 주기성을 갖는 모든 연관규칙들을 탐사한다[2]. 데이타베이스 D가 주어지고 k번째 시간 간격을 t_k라고 할 때, t_k에 속한 트랜잭션을 $D[k]$라고 정의한다. 임의의 연관규칙 r이 $D[k]$에서 사용자가 지정한 임계값들을 만족한다면 규칙 r은 시간 간격 t_k동안 유효하다고 말한다. 만약, 규칙 r이 최초 시간 간격 t_0에서 시작하여 매번 i번째 시간 간격에서 유효하다면 규칙 r은 주기$(t_0, t_1, ..., t_i)$에서 유효하다고 말한다. 따라서 주기적 연관규칙은 주어진 시간 간격에서의 완전한 주기성을 만족하는 규칙을 탐사하는 것이다. 그러나 실제로 주어진 시간 간격 동안에 완전하게 유지되는 규칙은 존재하지 않으며 대부분 불완전한 주기를 이루고 있다. 또한 주기적 연관규칙에서는 다양한 시간 단위를 표현하지 못하고 단 하나의 시간 단위만을 다룰 수 있다. 따라서 "매달 첫 번째 월요일"과 같은 실제 응용분야에서 적용되는 시간 표현은 불가능하다.

2.1.2 캘린더 기반 연관규칙 탐사

[6]은 주기적 연관규칙을 확장하여 연관규칙 탐사 시, 사용자 기반의 시간 패턴을 탐사하는 기법을 제안했다. 캘린더 연관규칙은 시간 패턴의 명시를 위해서 캘린더 대수(*calendar algebra*)를 사용한다. 캘린더는 시간 간격의 집합으로 정의되고, 캘린더 대수는 시간 연산자를 기반으로 캘린더 사이의 시간관계를 구할 수 있는 연산자를 제공한다. 예를 들어 *"Weeks in Jan 2001 : overlaps : Jan in 2001"*라는 시간 패턴은 "2001년 1월의 주별 중 1월에 포함하는 기간"을 의미하며, 캘린더

연산 결과로 시간 간격의 집합인 {(-1, 6), (7, 13), (14, 23), (21, 27), (28, 31)}을 출력한다. 이 기법은 주기적 연관규칙 보다 유용함과 실용적인 시간 연관규칙을 생성할 수 있으나 시간 패턴 탐사를 위해 사용자 기반의 캘린더 대수 표현이 요구되어지며, 이는 사용자로 하여금 탐사될 시간 패턴에 대한 정확한 이전 지식을 필요로 하는 단점을 가진다.

[5]는 많은 이전 지식을 필요로 하는 캘린더 대수 표현들을 사용하는 것 대신에 시간 패턴 탐사를 위한 프레임워크로 캘린더 스키마를 사용하였다. 이 접근방법은 주어진 캘린더 스키마에 대한 모든 가능한 캘린더 패턴들을 고려하고 [2]에서의 완전한 주기성을 탐사하는 것 대신에 캘린더 스키마에 의해 주어진 시간 간격 동안 충분히 만족되는 불완전 또는 부분적인 주기성을 탐사한다. 따라서 일정 시간 간격에서 만족되는 잠재적인 시간 연관규칙들의 탐사가 가능하다.

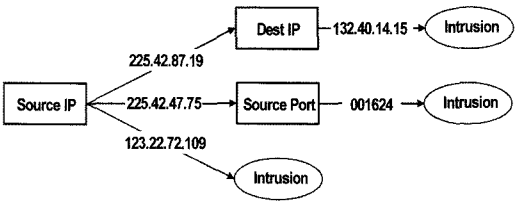
캘린더 패턴에 의한 시간패턴 탐사를 위해 시스템의 달력을 이용한 [3, 4]에서는 시간 연관규칙을 위한 문제제시, 마이닝언어(*DMQL*)와 규칙탐사를 위한 프레임워크를 제안하였고 각 시간 단위와 시간 간격을 이용하여 기존의 *Apriori* 알고리즘에 시간적인 표현을 추가한 형태로 규칙을 탐사한다. 시간 연관규칙은 $<AssRule, TimeExp>$ 형태로 표현되고 *AssoRule*은 $A \rightarrow B$ 형식의 규칙이다. *TimeExp*는 규칙 *AssoRule*에 대한 시간 표현식으로 규칙이 유효한 시간 간격의 집합이다. 또한 이 기법에서는 연관규칙의 효율적 탐사를 위해 주어진 시간 간격들에서 일정기간 동안만 유지되는 규칙을 탐사할 수 있도록 최소발생빈도를 정의하여 발생빈도를 고려한 연관규칙 탐사를 가능하게 하였다. 그러나 시간 표현을 위한 방법으로 캘린더 시스템에 의존한 특정 시점이나 시간 간격의 데이타를 분석하는데 그쳤으며, 시간에 따른 다양한 상호 관련성을 제시하는 측면에서는 한계성을 가진다.

2.2 연관적 분류

연관적 분류란 서로 독립적인 연관규칙과 분류규칙을 일부분 통합시킨 새로운 분류 방식으로 클래스 라벨을 예측하기 위해 연관규칙을 사용한다. 연관적 분류는 기존의 분류기법(결정트리)[7,8]에 비해서 다음과 같은 장점을 가진다. 첫째, 기존의 의사결정트리가 데이타 객체의 분류를 위해 단지 수 백 개의 속성들만을 조작하는 것으로 제한되는 반면, 연관적 분류는 수천의 속성 차원을 조작할 수 있다는 것이다. 둘째, *Naive Bayes*[9]과 유사하게 항목들을 독립적으로 고려할 수 있다. 또한 연관적 분류는 다양한 변수들 사이에서 높은 신뢰도를 갖는 규칙을 탐사하므로 한번에 하나의 변수만을 조사하는 의사결정트리에 의한 제약사항을 극복하였다. 마지막으로 탐사된 규칙은 단순성(*simplicity*)을 가지므로 사용자들은 규칙을 쉽게 이해할 수 있다. 그림 1은 침입탐지 시스템을 위한 데이타마이닝 기법 중 분류기법을 적용하는 것으로 의사결정트리(b)[10]와 연관적 분류(c)기법의 적용 예를 보여준다.

Source IP	Dest IP	Source Port	Dest IP	Intrusion
123.22.72.109	225.14.87.12	001360	000080	True
123.22.72.109	225.14.87.12	001425	000080	True
123.22.72.109	225.14.87.12	001488	000080	True
123.22.72.109	225.14.87.12	001559	000080	True
225.42.47.75	150.216.191.19	001624	000080	True
225.42.87.19	125.25.87.19	004207	000025	True
233.167.15.65	225.142.187.12	001624	000025	False
233.167.15.65	225.142.187.12	004690	000025	False
⋮	⋮	⋮	⋮	⋮

(a) 훈련 데이타



(b) 의사결정트리

Source IP: 123.22.72.109 → Intrusion: True
Source IP: 197.14.72.19 ∧ Source Port: 1624 → Intrusion: True
Source IP: 225.42.72.19 ∧ Dest IP: 132.40.14.15 → Intrusion: False
Source IP: 225.42.72.19 ∧ Dest Port: 113 → Intrusion: False

(c) 클래스 연관규칙

그림 1 의사결정트리와 클래스 연관규칙의 예

분류를 위한 연관규칙의 사용은 [11]에서 처음 소개되었고 기존의 의사결정트리에 비해 더 정확하다는 것이 증명되었다. 연관규칙 기반 분류(*CBA: Classification based on Association*)는 탐사된 규칙들의 결과부(*consequent*)가 클래스 라벨로 제한된 클래스 연관규칙(*CAR: Class Association Rules*)이라는 특별한 연관규칙의 부분집합에 초점을 둔다. 데이타베이스 안의 모든 항목집합 I와 모든 클래스 라벨들의 집합 Y가 주어지면 클래스 연관규칙은 $A \Rightarrow C$ 형태로 표현된다. A는 I 안에 포함되는 항목집합이고 C는 클래스 라벨이다. *CBA*는 빈발한 *CAR* 탐사를 위해 *Apriori*기반 알고리즘을 확장하여 적용하였고 탐사된 많은 빈발한 *CAR*로부터 분류모델 생성을 위한 규칙 제거 방법을 제안하였다.

[12]에서는 *CBA* 보다 더 효율적이고 정확한 분류를 위해 *CBA*를 확장한 *CMAR (Classification based on Multiple Association Rules)*를 제안하였다. *CMAR*은 규칙생성을 위해 *FP-growth* 알고리즘을 적용함으로써

많은 데이타베이스 스캔을 요구하는 *Apriori* 기반의 *CBA*의 문제점을 해결하였고. 많은 중복된 규칙들의 제거와 빠른 규칙들의 탐색과 저장을 위해 *CR-tree*라는 새로운 데이타구조를 정의하였다. 또한 분류모델 생성을 위한 탐사된 규칙들에 대해 상관분석을 적용함으로써 분류모델 생성을 위한 규칙 선택 시 양의 상관관계를 가지는 규칙들을 선택함으로써 에러율을 최소화하는 기법을 제시하였다.

2.3 침입탐지를 위한 데이타마이닝

침입탐지 시스템의 탐지 기법은 오용탐지와 이상행위 탐지으로 나눌 수 있다[13]. 오용탐지는 이미 알려진 침입 행위를 이용하여 규칙을 생성한 후, 입력 데이타와 규칙이 일치하는지의 여부에 따라 침입으로 판정한다. 오용탐지 기법은 공격패턴 정보를 가지고 있으므로 정상행위를 공격행위로 간주하는 오류(*false positive*)가 낮은 대신 알려지지 않은 새로운 공격은 탐지할 수 없는 단점을 가지고 있다. 이상행위탐지는 다양화되는 침입에 대응하기 위해 정상행위의 프로파일을 이용하여 모델링된 정상행위에서 벗어나는 행위 등을 공격행위로 간주하는 탐지기법이다. 그러나 정상행위 프로파일 생성 시 다량의 감사데이타를 분석해야 하므로 정확하고 효율적인 분석을 위해 데이타마이닝 기법을 적용한다[14-16].

지금까지 정상행위를 위한 프로파일 생성에는 빈발에피소드와 연관규칙 탐사 기법이 사용되었다. 그러나 빈발에피소드는 단순히 빈발한 데이타의 발생 순서만을 탐지하고 연관규칙은 시간데이타인 감사데이타를 정적인 데이타로 간주하여 프로파일을 생성하였다. 실제로, 네트워크 연결들에서는 발생된 이벤트들의 패턴은 서로 다른 시간에 따라 매우 다르게 나타날 수 있다. 그러므로 정확하고 효율적인 침입탐지를 위해 시간 데이타마이닝 기법이 필요하다[17-19].

3. 시간 클래스 연관규칙 탐사

시간 데이타마이닝의 시간 패턴 표현 방법으로서 캘린더 스키마를 프레임워크로 적용하고 캘린더 스키마에 의해 정의되는 캘린더 패턴식을 사용한 새로운 시간 클래스 연관규칙 탐사 알고리즘을 제안한다. 이 절에서 제안된 시간 클래스 연관규칙은 시간 연관적 분류 규칙들의 잠재적인 규칙이다.

3.1 캘린더 스키마와 캘린더 패턴

캘린더 스키마는 달력의 개념 계층에 의해 결정되어지고 유효성 제약조건을 갖는 관계형 스키마이다.

정의 3.1: 캘린더 스키마(*CS*: *Calendar Schema*), 캘린더 스키마는 달력 표현의 시간 단위와 그 단위에서의 가능한 도메인의 집합으로 정의되며, 그 형태는 다음과 같다.

$$CS=(G_n:D_n, G_{n-1}:D_{n-1},\ldots,G_1:D_1)$$

$1\le i\le n$에 대해, 속성 G_i는 년, 월, 일 등과 같은 달력 개념에서의 시간 단위이고, 각 D_i는 양의 정수의 유한 집합으로 속성 G_i의 도메인 값의 집합을 나타낸다. ■

캘린더 스키마가 $(G_n, G_{n-1},\ldots,G_1)$이고 $1\le i<n$일 때, 각 시간 단위 G_i는 유일하게 G_{i+1}에 포함된다. 예를 들어 스키마(*month, day*)에 대해, '요일'은 특정 '월'에 포함되므로 유효하나 *(year, month, week)*일 경우, '주'가 특정 '월'에 포함되지 않으므로 잘못된 스키마가 된다. 또한 $CS=(year:1995\sim1999, month:1\sim12, day:1\sim31)$일 경우, $\{1995,1,20\}$은 유효한 시간 표현이 되나, $\{1996,2,31\}$은 유효한 조합이 되지 않는다.

정의 3.2: 캘린더 패턴(*CP*: *Calendar Pattern*), 캘린더 패턴은 주어진 스키마$CS=(G_n:D_n,\ldots,G_1:D_1)$의 인스턴스이며, $CP=\{d_n,\ldots,d_1\}$으로 표현된다. 여기서 각 d_i는 D_i의 도메인 값이거나 문자 '*'이다. 만약 d_i가 '*'이라면 그 의미는 도메인 D_i의 모든 값을 나타내고 "*every*"로 해석한다. ■

예를 들어 캘린더 스키마가 (*month*:{1~3}, *day*:{1~7})로 주어졌을 때, 캘린더 패턴 {*, 3}은 시간 간격 {1, 3}, {2, 3}, {3, 3}을 나타내고 "매달 수요일"을 의미한다. 또한 캘린더 패턴에 포함되어진 '*'의 개수에 따라 표 1과 같이 캘린더 패턴의 표현을 구분한다. i개의 '*'를 가지는 캘린더 패턴은 $i-star\ pattern(CP_i)$이라 부르고 $i-star\ pattern(CP_i)$들의 집합을 $\Phi(CP_i)$로 나타낸다. 특히 '*'를 전혀 포함하지 않는 캘린더 패턴($i-star\ pattern(CP_i)$)에 대해서는 "기본시간단위"라고 부른다.

표 1 캘린더 패턴(CP)의 표현 방식

'*'의 개수	캘린더 패턴	표현
0	0-star pattern	CP_0
1	1-star pattern	CP_1
...	...	...
i	i-star pattern	CP_i

정의 3.3: 캘린더 패턴 사이의 포함관계,
$CS=(G_n,G_{n-1},\ldots,G_1)$, $CP=\{d_n,d_{n-1},\ldots,d_1\}$;
$CP'=\{d'_n,d'_{n-1},\ldots,d'_1\}$, $1\le i\le n$인 각 i에 대해, $d_i='*'$이거나 $d_i=d'_i$일 때, CP는 CP'을 포함한다. ■

예를 들어 스키마가 (*week, day, hour*)일 경우, 정의 3.2에 대해 $CP_1=\{2,*,12\}$은 'day'에 대한 주기를 가지며, 정의 3.3에 의해 또 다른 $CP_0=\{2,2,12\}$을 포함한

다. 따라서 기본시간단위 $CP_0 = \{d_n, ..., d_1\}$, $CP_0 \in \Phi(CP_0)$이 주어지고 다른 i개의 '*'를 갖는 CP_i가 CP_0을 포함한다면, 그러한 CP_i들의 집합을 $\Phi(CP_i(CP_0))$로 나타낸다.

3.2 문제 정의

시간 클래스 연관규칙이란 2.2절에서 소개한 클래스 연관규칙에 캘린더 패턴식을 적용하여 확장한 것이다. 먼저 클래스 연관규칙의 정의를 내리고 이로부터 캘린더 기반 시간 클래스 연관규칙을 탐사하는 문제를 단계별로 정의한다.

3.2.1 클래스 연관규칙 탐사

트랜잭션들로 구성된 데이타베이스를 D, $I = \{i_1, i_2, ..., i_n\}$을 모든 항목집합 그리고 $C = \{c_1, c_2, ..., c_m\}$을 클래스 라벨 집합이라고 한다.

정의 3.4: 클래스 연관규칙은 $X \rightarrow c_i$의 형태로 표현되고 $X \subset I$, $c_i \subset C$이다. 여기서 클래스 연관규칙의 전제부를 *ItemSet*라 하고 규칙 자체를 *RuleItem*라 한다. ■

정의 3.5: 클래스 연관규칙의 지지도와 신뢰도,

$$support = \frac{RuleSupCnt}{|D|}, \quad confidence = \frac{RuleSupCnt}{ItemSupCnt}$$

*RuleSupCnt*는 *RuleItem*의 개수 값으로 *ItemSet*을 포함하면서 클래스 c_i로 라벨된 D안의 항목들의 개수이다. *ItemSupCnt*는 *ItemSet*을 포함하는 D안의 항목들의 개수이다. ■

예를 들어 D의 총 개수가 10개이고 최소지지도가 10%, 최소신뢰도가 60%로 주어진다면, 규칙 R : $<(A \wedge B \wedge C) \Rightarrow (Class = 1)>$에 대해, 규칙 R의 *ItemSet*인 $<(A \wedge B \wedge C)>$의 개수가 3이고 *RuleItem*의 지지도는 2일 경우, 규칙 R의 지지도는 2/10이므로 20%가 되고 신뢰도는 2/3이므로 67%이다. 따라서 규칙 R은 최소지지도와 최소신뢰도를 만족하므로 유효한 규칙이다.

정의 3.6: 클래스 연관규칙(*CAR* : *Class Association Rules*)에 대해, 규칙의 전제부(*ItemSet*)에 하나 또는 몇 개의 항목이 추가/제거된 또 다른 규칙을 CAR_{sub}/CAR_{sup}라고 하며, CAR_{sub}/CAR_{sup}을 *CAR*의 *Sub-Rule*/*Supper-Rule*라고 한다. ■

3.2.2 시간 클래스 연관규칙 탐사

시간 클래스 연관규칙 탐사의 문제 정의를 위해서 먼저, 모든 트랜잭션 데이타베이스 D는 트랜잭션 시간이 주어진 타임스탬프와 관련된다고 가정한다.

정의 3.4와 3.5에 캘린더 패턴식을 추가하여 시간 지식탐사 문제를 정의하면 다음과 같다.

정의 3.7: 캘린더 스키마 *CS*에 대한 캘린더 패턴 *CP*가 주어지면, *CP*에 의해 포함되어지는 타임스탬프의 트랜잭션을 $D(CP)$로 나타낸다. 문법적으로 캘린더 연관규칙은 $<CAR, CP>$형태로 표현되며, *CAR*은 클래스 연관규칙이고 *CP*는 *CS*에 대한 캘린더 패턴이다. 또한 $\Phi(CP_0)$은 *CP*의 모든 기본시간단위 집합을 나타낸다. ■

예를 들어 시간 클래스 연관규칙은 기존의 클래스 연관규칙을 확장하여 다음과 같이 표현 된다. $CS = (year : 1999 \sim 2001, month : 1 \sim 12, day : 1 \sim 31)$ 일 때, $<(A \wedge B) \Rightarrow (Class = 1), \{*, 2, 3\}>$은 캘린더 표현과 주기적 표현을 나타내는 것으로 기본시간단위의 집합, $\Phi(CP_0) = 1999, 2, 3, 2000, 2, 3, 2001, 2, 3$에서 규칙이 유효하며, 의미는 "1999년 ~ 2001년 기간 동안 매 2월 3일에 규칙 $<(A \wedge B) \Rightarrow (Class = 1), \{*, 2, 3\}>$이 성립 된다"와 같다.

시간 클래스 연관규칙에서는 기존의 지지도, 신뢰도 외에도 유용성 측면에서 발생도(*frequency*)라는 새로운 임계값을 사용한다.

정의 3.8: 최소발생도(*minimum frequency*), 규칙, $<CAR, CP>$가 $\Phi(CP_0)$에 속한 기본시간단위의 f% 이상을 만족한다면, 규칙, *CAR*은 캘린더 패턴 *CP*를 만족한다. 이 때, f%을 최소발생도, $minFre$이라고 정의한다.■

예를 들어 캘린더 스키마가 $CS = (week : 1 \sim 4, day : 1 \sim 7)$이고 최소발생도가 0.6로 주어졌다면, 시간 클래스 연관규칙 $<(A \wedge B) \Rightarrow (Class = 1), \{*, 3\}>$은 캘린더 패턴 {*, 3}가 포함하는 기본시간단위 {1, 3}, {2, 3}, {3, 3}, {4, 3} 중에서 세 기본시간(예, {1, 3}, {2, 3}, {3, 3}) 동안에 최소지지도와 최소신뢰도를 만족한다면 발생도는 3/4= 0.75%가 되고 최소발생도를 만족하므로 이 클래스 연관규칙은 캘린더 패턴식을 만족한다. 즉, 규칙, <(A∧B)⇒(Class= 1)>은 매주 수요일에 발생된다는 의미이다. 최소발생도의 적용 이유는 실제 응용에서 주어진 시간 간격 동안에 모두 성립되는 규칙은 거의 없기 때문에 유연성을 주기 위함이다.

결론적으로 시간 클래스 연관규칙 탐사는 데이타베이스 D로부터, 사용자가 미리 지정한 최소발생도 $minFre$에 대해서 최소지지도 $minSup$과 최소신뢰도 $minConf$을 만족하는 규칙들을 탐사하는 문제이며, 발생도를 고려함으로써 시간에 따라 변화하는 규칙을 탐사할 수 있다.

3.3 규칙 탐사 알고리즘

시간 클래스 연관규칙에 대한 문제 정의와 함께, 기본적인 규칙탐사 알고리즘의 구조를 묘사할 수 있다. 기존의 여러 알고리즘들이 서로 다름에도 불구하고 그들 모두는 기본적인 스키마를 사용한다.

가정: 탐사될 시간 클래스 연관규칙 중 기본시간단위

에서만 유효한 규칙, $CAR(CP_0)$은 유용하지 못한 규칙으로 간주하여 제외된다. 왜냐하면 탐사된 규칙들은 미래 예측을 위한 목적으로 생성되므로 기본시간단위에서의 규칙들은 특정 시간만을 표현하므로 유용하지 못하다.

모든 시간 클래스 연관규칙들을 구성하기 위해서는 각 기본시간단위에서 데이타베이스에 있는 모든 빈발한 *RuleItem*들의 지지도가 먼저 계산된다. 그러므로 제안한 알고리즘은 다음의 세 단계로 수행된다. 먼저, 주어진 캘린더 스키마에 대해, 모든 *large RuleItem*들을 생성하고 그것들로부터 주어진 신뢰도를 바탕으로 실제 규칙들을 생성한다. 그 다음 기본시간단위에서의 생성된 규칙들을 캘린더 패턴의 포함관계를 고려하여 갱신하게 된다.

표 2 알고리즘을 위한 기호와 의미

기 호	의 미
D	트랜잭션 데이타베이스
d	데이타객체
$\Phi(CP_i)$	*i-star* 캘린더 패턴들의 집합
CP_i	*i-star* 캘린더 패턴
C_k	*k*개 *ItemSet*을 갖는 후보 *RuleItem*
L_k	*k*개 ItemSet을 갖는 *RuleItem*들의 집합
CAR_k	*k*개 *ItemSet*를 갖는 클래스 연관규칙

표 2는 알고리즘을 위한 기호와 설명이고 그림 2는 시간 클래스 연관규칙 탐사의 전체 알고리즘으로 다음의 3단계로 나누어 수행된다.

단계 1: 주어진 $CS=(G_n, ..., G_1)$에 대해, 각 기본시간단위(CP_0)에서 최소지지도와 최소신뢰도를 만족하는 모든 *large RuleItem* 집합을 생성한다. 빈발한 $k-$*RuleItem*은 *k*개의 *ItemSet*을 가지는 *RuleItem*을 나타내고 L_k은 $k-$*RuleItem*들의 집합이다. 집합 L_k의 각 요소는 $<(ItemSet, ItemSupCnt), (ClassLabel, RuleSupCnt)>$이고, 후보 $k-$*RuleItem*들의 집합을 C_k이라 하면 기본시간단위 CP_0에서의 규칙 탐사 과정은 그림 3과 같다.

첫 번째 단계에서의 수행은 세 가지의 주요 연산을 거친다.

(1) $(k-1)$ 패스에서 탐사된 L_{k-1}은 candidateGen()에 의해 C_k를 생성하기 위해 사용되어진다(8라인).

(2) 생성된 C_k는 데이타베이스 스캔을 거쳐 후보 $k-$*RuleItem*들의 지지도를 갱신하게 된다(9~16라인).

(3) 지지도 카운트가 진행 된 후, 임계값 $minSup$를 만족하는 *RuleItem*들만을 선택한다(17라인).

```
1   input transaction D, minSup, minConf, minFre
2   output <CAR_k(CP_i), CP_i> for all i-star patterns

3    for each 0-star pattern CP_0 ∈Φ(CP_0)  do
4          L_1(CP_0) = { large 1-RuleItems in D[ CP_0] }
5    end
6    for (k=2;L_{k-1}(CP_0)≠Φ for at least one CP_0∈Φ(CP_0);k++) do
7        for each 0-star pattern CP_0∈Φ(CP_0) do
           //PHASE Ⅰ: Generate RuleItems.
8            C_k(CP_0)=candidateGen(L_{k-1}(CP_0));
9            for each data case d∈D[ CP_0] do
10                C_d=subset(C_k(CP_0), d);
11                for each candidate c∈C_d do
12                    c.ItemSupCnt++;
13                    if d.class=c.class then
14                        c.RuleSupCnt++;
15                end
16           end
17           L_k(CP_0)={ c∈C_k | c.RuleSupCnt≥minSup} ;
           //PHASE Ⅱ: Generate class association in CP_0
18           CAR_k(CP_0)=genRule(L_k(CP_0));
           //PHASE Ⅲ: Update for star calendar patterns
19           for (i=1;i≤n;i++)
20               To_calUpdate_i(CP_0 , CAR_k(CP_0));
21           end
22       end
23       for (i=1;i≤n;i++)
24       ∪_{CP_i∈Φ(CP_0)} CAR_k(CP_i)=From_calUpdate_i();
25       end
26   end
```

그림 2 시간 클래스 연관규칙 탐사 알고리즘

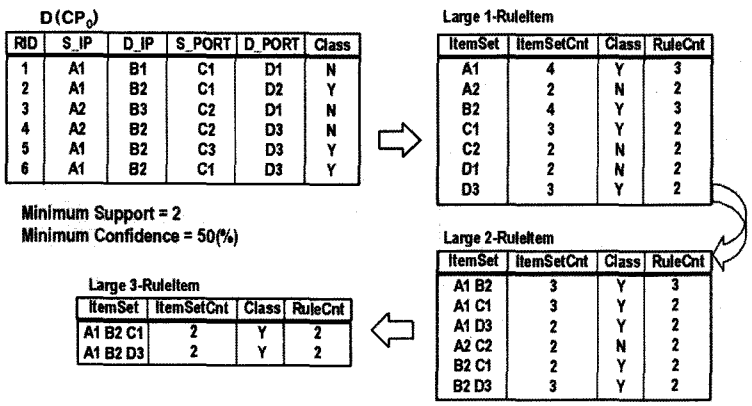


그림 3 기본시간단위에서의 클래스 연관규칙 탐사 과정

단계 2: 생성된 $k-RuleItem m$들의 집합 L_k로부터 genRules()에 의해 k개의 $ItemSet$과 클래스 라벨을 가진 CAR_k를 생성한다. 생성된 L_k들은 신뢰도가 계산되어지고 임계값, $minConf$을 만족하는 규칙들만을 생성한다(18라인). 그러나 만약, $ItemSet$이 같고 클래스 라벨이 다른 $RuleItem$들이 존재할 경우, 높은 신뢰도를 가지는 $RuleItem$을 선택한다. 예를 들어 $ItemSet$이 같은 다음의 두 $RuleItem$이 존재한다고 할 때,

1. $<ABC \rightarrow class : Y>$ 2. $<ABC \rightarrow class : N>$

여기서, $|D(CP_0)|$=10, $minConf$=0.3이고 $ItemSet$의 지지도가 3, 첫 번째 $RuleItem$의 지지도는 2, 두 번째 $RuleItem$의 지지도를 1이라고 가정하자. 그러면 $RuleItem$ 1의 신뢰도는 67%이 되고, $RuleItem$ 2의 신뢰는 33%가 된다. 따라서 생성되는 $RuleItem$은 $<ABC \rightarrow class : Y>[sup=0.2, conf=0.67\%]$이 된다.

단계 3: 최소신뢰도를 만족하는 모든 $CAR_k(CP_0)$들을 생성한 후, 각 $i-star pattern$에 대한 빈발한 $CAR_k(CP_i)$들을 계산한다. 이 단계에서는 $i-star pattern$으로 저장과 검색을 위해 해시트리를 이용한다.

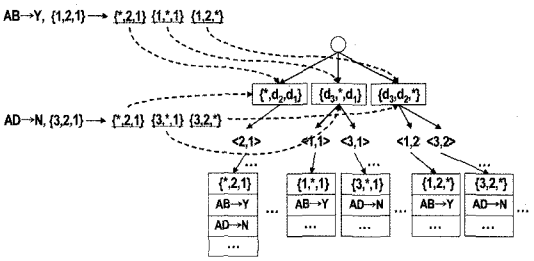
(1) 루트 노드를 시작으로 하여 CP_i안의 서로 다른 '*'의 분포로부터 루트의 자식노드로 차례로 매핑이 이루어진다. 예를 들어 만약, n=3이고 i=2이면 (n은 캘린더 패턴의 시간 단위의 개수, i는 '*'의 개수) 해시트리의 루트는 $\{*,*,d_1\},\{*,d_2,*\},\{d_3,*,*\}$으로 표현되는 3개의 자식노드를 갖는다.

(2) 루트의 한 자식노드를 시작으로 CP_i가 저장될 위치를 결정한다. 여기서 자식노드는 해시벡터를 이용하여 단말노드를 위치시킨다. 해시벡터는 CP_i안의 모든 양의 정수로 구성된다. 예를 들어 CP_i={1,*,2}일 경우, 해시벡터는 <1,2>가 된다.

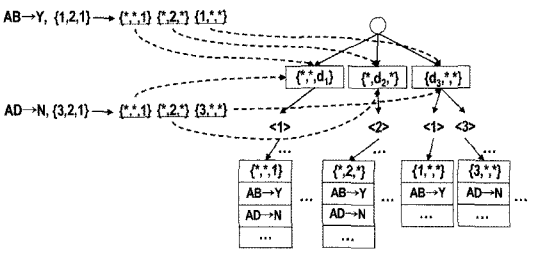
(3) 해시벡터가 계산된 후, 캘린더 패턴 CP_i 그리고 CP_i과 관련된 규칙인 $CAR_k(CP_i)$은 단말노드에 저장된다. 해시트리의 구성단계를 그림 4에 나타내었다(단, n=3이고 CP_0={3,2,1}로 가정한다.)

그림 2에서 To_calUpdate$_i$은 $i-star\ pattern$에 대해, 규칙, $CAR_k(CP_i)$을 갱신하기 위해서 사용된다. $i-star\ pattern$ 패턴으로의 갱신 과정은 그림 5와 같다.

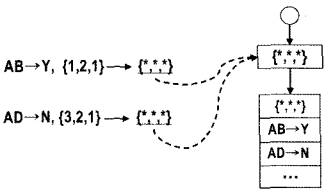
각 $i-star\ pattern$에 대한 규칙들은 각각의 해당되는 카운터를 가지고 있다. 만약 갱신 과정이 처음 이루어지는 단계(4~6라인)이거나 기본시간단위에서의 규칙이 처음 갱신 되어질 경우(8~10라인), $CAR_k(CP_0)$를 $CAR_k(CP_i)$으로 할당하고 카운터를 1로 설정한다. 그렇지 않은 경우(11~13라인), 현재의 카운터를 1씩 증가시킨다.



(a) CalTree$_1$



(b) CalTree$_2$



(c) CalTree$_3$

그림 4 캘린더 해시트리에서의 저장 검색

```
1  input CARk(CP0),CP0, minFre
2  output CARk(CPi),CPi satisfy minimum frequency

3   if CARk(CPi) is first time updated then
4        CARk(CPi) = CARk(CP0);
5        updateCnt = 1;
6    else
7       for each CAR ∈ CARk(CP0) - CARk(CPi)
8           CAR.updateCnt = 1;
9       end
10      for each CAR ∈ CARk(CP0) ∩ CARk(CPi)
11          CAR.updateCnt++;
12      end
13      CARk(CPi) = { CAR.updateCnt ≥ minFre·Ni}
14  end if
```

그림 5 $i-star$ 캘린더 패턴 갱신 알고리즘

만약 CP_i에 의해 포함되는 CP_0가 N_i개, 현재 $CAR_k(CP_i)$으로 n번째 갱신 단계라고 가정할 경우, $CAR_k(CP_i)$의 각 카운터는 공식, $updateCnt + (N_i - n) \geq minFre \cdot N_i$을 만족하는 $CAR_k(CP_i)$들만을 유효한 규칙으로 생성한다. 기본시간단위에서의 규칙을 갱신하는 단계 대한 예를 그림 6에 나타내었다.

CAR_2(*, 3} (before update)	CAR_2(3, 3}	CAR_2(*, 3} (after update)
AB →1 updateCnt=2	AB →1	AB →1 updateCnt=3
AC →0 updateCnt=1	AC →0	AC →0 updateCnt=2
AD →1 updateCnt=1		AD →1 updateCnt=1 X
BC →0 updateCnt=2	BC →0	BC →0 updateCnt=3
	BD →0	BD →0 updateCnt=1 X

그림 6 캘린더 패턴으로의 갱신 과정

위의 예에서 캘린더 스키마는 (*week*:{1~4}, *day*:{1~5}), $minFre = 0.8$, CP_i={*,3}에 의해 포함되는 CP_0의 개수를 5, 그리고 CP_0={3,3}이 3번째로 갱신되는 단계일 경우의 갱신 후의 결과를 나타낸다(갱신 후 단계에서 **X**는 최소발생도를 만족 못하는 규칙이다).

알고리즘 1에서 갱신 과정이 종료된 다음에 함수 From_CalUpdate(24라인)에 의해 최종적인 규칙인 $CAR_k(CP_i)$들은 빈발한 규칙들의 조합, $\bigcup_{CP_i \in \Phi(CP_i)} CAR_k(CP_i)$으로 생성된다.

시간 클래스 연관규칙 탐사 결과인 $CAR_k(CP_i)$은 사용자 기반 캘린더 스키마 정의와 임계값인 최소지지도, 최소신뢰도, 최소발생도에 의존적이다. 따라서 5장의 실험 평가 부분에서 탐사된 규칙의 캘린더 스키마와의 관계와 유용성 측정 지표인 임계값들에 대해 비교 분석해 본다.

4. 시간 연관적 분류규칙 탐사

3장에서 생성된 시간 클래스 연관규칙을 이용하여 시간에 따른 연관적 분류규칙 탐사를 위한 규칙 생성 기법을 소개한다.

4.1 문제 정의

시간 클래스 연관규칙 탐사로 생성된 규칙들은 신속하고 효율적인 분류규칙 생성을 위해 사용된다. 이 논문에서의 침입탐지 시스템을 위한 시간 연관적 분류규칙 탐사의 전체 프레임워크는 그림 7과 같다.

시간 연관적 분류규칙 탐사를 위해서 먼저, 시간 클래스 연관규칙 탐사 알고리즘을 적용하여 모든 규칙들을 생성한 후에 4.2절에서 소개될 규칙 선택과 제거 단계를 거쳐 양질의 분류규칙을 생성한다. 그 다음 생성된 분류규칙들은 테스트 데이타를 이용하여 정확성 여부를 검사한다. 만약 정확한 분류규칙들이 생성된다면 클래스 라벨이 없는 새로운 데이타를 시간 속성을 고려하여 분류하게 된다.

탐사된 시간 클래스 연관규칙들로부터의 분류규칙 생성을 위해서 모든 규칙들은 각 캘린더 패턴 CP_i ($CP_i \in \Phi(CP_i)$)에 의해 그룹핑 된다. 또한 각 그룹(같은 캘린더 패턴)의 규칙($CAR_k(CP_i)$)들은 정의 4.1의 기준에 의해 내림차순으로 정렬된다.

정의 4.1: 두 규칙 $CAR(CP_i)_m$, $CAR(CP_i)_n$가 주어지면, 조건 (1), (2)에 의해 $CAR(CP_i)_m$은 $CAR(CP_i)_n$보다 높은 우선순위($CAR(CP_i)_m > CAR(CP_i)_n$)를 갖는다.

(1) $Frequency(CAR(CP_i)_m) > Frequency(CAR(CP_i)_n)$

또는,

(2) $Freuency(CAR(CP_i)_m) = Frequency(CAR(CP_i)_n$

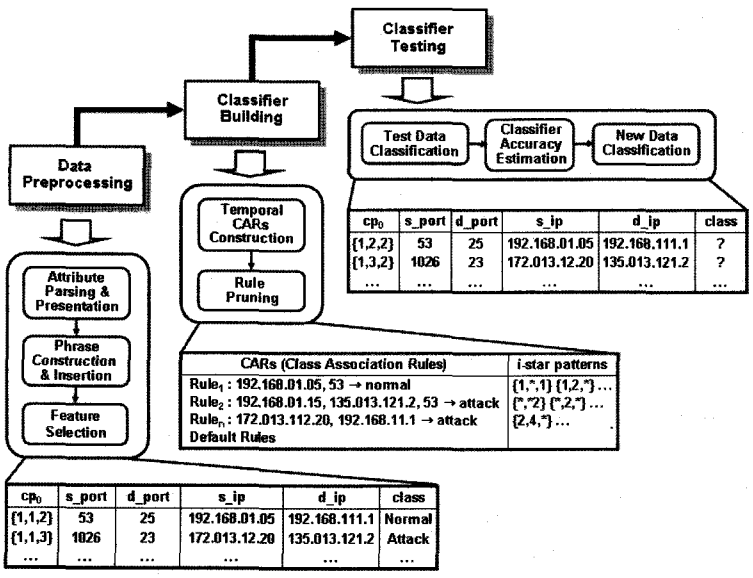


그림 7 시간 분류규칙 탐사 과정 프레임워크

$\wedge$ $CAR(CP_i)_m$이 $CAR(CP_i)_n$의 *Sub-Rule*이다. ($\because$ $CAR(CP_i)_m = (CAR(CP_i)_n)_{sub}$). ■

정의 4.1은 분류규칙 생성을 위한 규칙선택 시, $i-star\ pattern$에서 가장 발생도가 높은 규칙을 우선적으로 선택하는 것을 보장한다. 그러나 만약 두 규칙의 발생도가 같은 경우에는 일반화된 규칙을 선택하도록 한다.

정의 4.2: 캘린더 패턴에 대한 분류규칙의 형식 $< CAR(CP_i)_1, CAR(CP_i)_2, ..., CAR(CP_i)_n, deRule, \{CP_i\} >$.

여기서, $CAR(CP_i)_j$는 탐사된 시간 클래스 연관규칙이고 *deRule*는 디폴트 규칙, $\{CP_i\}$는 그 규칙들이 유효한 캘린더 패턴 표현이다. ■

시간 클래스 연관규칙에 의한 연관적 분류규칙 생성의 기본접근법은 다음과 같다.

(1) 규칙들이 포함된 각 캘린더 패턴에 대해, 훈련 데이타의 기본시간단위가 캘린더 패턴에 포함되어지는 모든 데이타를 선택한다.

(2) 각각의 규칙 $CAR(CP_i)_j$가 정확하게 어떤 데이타를 포함(covering)할 경우, 규칙 $CAR(CP_i)_j$을 분류규칙 생성을 위한 규칙으로 선택한다.

정의 4.3: 데이타 $d(CP_0)$가 규칙 $CAR(CP_i)_j$의 *ItemSet*을 모두 포함한다면 $d(CP_0)$은 $CAR(CP_i)_j$에 의해 포함(*covered*)된다고 하고 또는, 규칙 $CAR(CP_i)_j$가 훈련 데이타 $d(CP_0)$를 포함(*covering*)한다고 말한다. ■

분류규칙의 생성 후에 클래스 라벨이 없는 새로운 데이타의 분류에 대해, 새로운 데이타는 그 데이타를 포함하는 첫 번째 규칙의 클래스 라벨을 할당 받는 것을 원칙으로 한다.

4.2 분류규칙 생성

시간 연관규칙 탐사 결과 생성된 잠재적 분류규칙들은 많고 중복규칙들을 포함하고 있다. 따라서 이 절에서는 많은 잠재적인 규칙들로부터 정확한 규칙들만을 추출하기 위해 세 가지의 규칙제거 또는 선택 방법과 알고리즘을 기술하고 생성된 규칙들로부터의 분류과정을 기술한다.

4.2.1 캘린더 패턴들의 포함관계에 의한 규칙제거

시간 연관규칙 탐사 알고리즘의 각 캘린더 패턴에 대한 갱신 과정을 거친 후, 생성된 규칙들에 대해서 다음 조건을 만족하는 규칙들을 제거한다.

n을 캘린더 스키마에 정의된 시간단위의 개수라고 할 때, $1 \le i < j \le n$인 두 캘린더 패턴이 $CP_i \in CP_j$을 만족하고, $CAR_k(CP_i) = CAR_k(CP_j)$일 경우, 규칙 $CAR_k(CP_i)$은 제거된다.

캘린더 패턴의 포함관계에 대한 규칙제거의 예를 그림 8에 나타내었다.

4.2.2 중복규칙의 제거

상세하고 낮은 우선순위를 갖는 규칙들을 제거하기 위해서 일반화된 규칙과 정의 4.1의 기준에 의해 순서화된 높은 우선순위의 규칙을 사용한다.

정의 4.4: 규칙 $CAR(CP_i)$에 대해, $CAR(CP_i)$의 부분규칙인 $CAR(CP_i)$가 $CAR(CP_i)$보다 높은 우선순위를 갖는다면, 규칙 $CAR(CP_i)$을 중복규칙이라 정의한다. ■

예를 들어 $A' \subset A$의 관계인 두 규칙 $A \rightarrow c$과 $A' \rightarrow c'$,

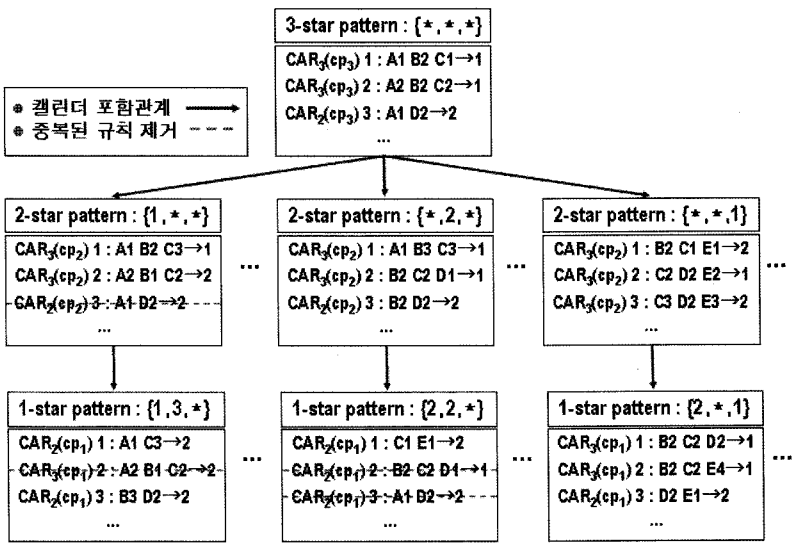


그림 8 캘린더 패턴의 포함 관계에 따른 규칙제거

($c \neq c'$인 두 클래스)에 대해, 규칙 $A' \rightarrow c'$은 $A \rightarrow c$에 관하여 일반화된 규칙이다. 또한 $A' \rightarrow c'$의 우선순위가 더 높을 경우, 규칙 $A \rightarrow c$은 중복규칙이므로 제거된다.

중복규칙 제거의 원리는 다음과 같다. 만약 규칙 $CAR(CP_i)$이 데이타 $d(CP_0)$을 포함한다면 $CAR(CP_i)$ 또한 $d(CP_0)$을 포함한다. 그리고 $CAR(CP_i)$은 더 높은 우선순위를 가지므로 항상 선택된다.

4.2.3 데이타베이스 *coverage*에 의한 규칙 선택

데이타베이스 *coverage*[11,12]는 생성된 모든 분류규칙들 중에서 훈련데이타들에 매치되는 규칙들만을 선택하기 위한 것이며, 이는 훈련데이타가 생성된 분류규칙에 의해 분류되어질 수 있다면, 테스트 데이타 역시 같은 분류규칙에 의해 분류될 수 있다는 가정을 포함한다. 데이타베이스 *coverage*에 기반한 삭제에는 새로운 임계값 *dbCover*를 적용한다. 그림 9는 데이타베이스 *coverage*에 따른 규칙 제거 알고리즘이며, 수행과정은 다음과 같다.

(1) $D(CP_0)$의 모든 데이타 $d(CP_0)$의 카운터(*dbCover*)를 0으로 초기화한다(3~5라인).

(2) 각 캘린더 패턴 cp_i 에서의 규칙들에 대해, 규칙의 캘린더 패턴이 $d(CP_0)$의 기본시간단위을 포함하고 규칙 $CAR(CP_i)$이 $d(CP_0)$을 정확하게 분류할 수 있다면, 규칙은 분류규칙으로 선택된다. 그리고 규칙 $CAR(CP_i)$에 포함되는 $d(CP_0)$의 카운터, $d(CP_0).coverCnt$을 1로 증가시킨다.

여기서 주어진 임계값 *dbCover*을 $d(CP_0).coverCnt$이

```
1  input A set of CARk(cpi), coverage threshold dbCover
2  output A subset of CARk(cpi) for classifier

3    for all d(cp0)∈D(cp0) do
4        d(cp0).coverCnt=0;
5    end
6    for the training data set and rule set=Ø do
7        for each CARk(cpi) ∈rule set do
8            for each d(cp0)∈D(cp0) do
9                if (d(cp0).cp0∈CARk(cpi).cpi)&&
                    (CARk(cpi)⊆d(cp0)) then
10                   d(cp0).coverCnt++;
11                   select CARk(cpi);
12               if d(cp0).coverCnt≥dbCover then
13                  delete d(cp0) in D(cp0);
14           end
15       end
16   end
```

그림 9 *dbCover*에 기반한 규칙 선택 알고리즘

만족한다면($d(CP_0).coverCnt \geq$ dbCover), $d(CP_0)$을 훈련데이타에서 삭제한다(6~16라인).

알고리즘 수행 후, 모든 규칙들에 의해 분류 되어질 수 없는 새로운 데이타들을 위해, *default* 규칙을 적용한다. *default* 규칙은 데이타를 포함하는 규칙이 전혀 없을 경우, 남아있는 훈련 데이타들의 다수 클래스 라벨을 이용하여 예측을 한다. 만약, 훈련 데이타가 남아있지 않을 경우, 원시 데이타베이스의 다수 클래스를 *default* 규칙으로 적용한다. 그림 10은 *dbCover*를 이용한 분류규칙 선택 과정의 한 예를 보여준다.

데이타베이스 coverage를 이용한 규칙선택은 분류규칙 생성을 위한 규칙들의 선택을 증가시키기 위한 것이고, 또한 *default* 규칙의 클래스 라벨의 할당 횟수를 줄이기 위한 것이다.

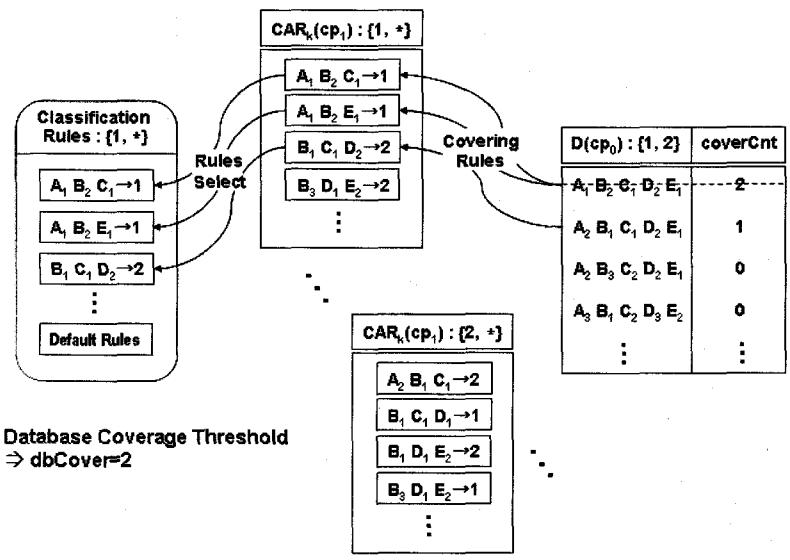


그림 10 데이타베이스 *coverage*에 의한 규칙선택

4.2.4 분류규칙에 의한 데이타 분류과정

분류를 위한 모든 규칙들을 선택한 후에 새로운 데이타를 분류한다. 먼저 분류될 새로운 데이타의 기본시간 단위를 포함하고 그 데이타를 정확하게 분류할 수 있는 모든 규칙들 $CAR(CP_i)$를 분류규칙으로 선택한다. 만약 모든 분류규칙들이 같은 클래스를 가지면 단순히 그 클래스를 새로운 데이타에 할당한다. 그러나 다른 클래스를 가질 경우, 규칙들을 각 클래스 별로 그룹핑한다. 그런 다음 각 그룹의 해당 클래스에 대한 강도를 *CMAR* [12]의 *weighted*-χ^2 상관분석을 적용하여 측정한 뒤 가장 강한 그룹의 클래스를 할당한다. *weighted*-χ^2의 계산을 위해 먼저 각 클래스 별 그룹의 모든 분류규칙들에 대한 카이제곱(χ^2) 값의 계산이 필요하다.

정의 4.5: 카이제곱(χ^2)의 계산은 관찰된 값(O)과 기대 값(E)을 이용하며, 식 (1)과 같다.

$$\chi^2 = \sum \frac{(O-E)^2}{E} \quad (1)$$

예를 들어 분류규칙 $A \rightarrow C$에 대한 χ^2값의 계산은 그림 11 과정의 실제 측정된 관찰 분할표(observed contingency table)와 기대 분할 테이블(expected contingency table)을 통해 구할 수 있다.

정의 4.6: 각 클래스별 그룹의 *weighted*-χ^2의 계산은 먼저 각 규칙들의 *maximum*-χ^2 값을 구한다. 규칙 $R=A \rightarrow C$에 대한 *maximum*-χ^2 식은 다음과 같다.

	C	*!C*	*∑*
A	*6*	*2*	*8*
!A	*6*	*18*	*24*
∑	*12*	*20*	*32*

(a) 관찰된 분할표

	C	*!C*	*∑*
A	*(12*8)/32=3*	*(20*8)/32=5*	*8*
!A	*(12*24)/32=9*	*(20*24)/32=15*	*24*
∑	*12*	*20*	*32*

(b) 기대 분할표

O	*E*	*(O-E)²*	*(O-E)²/E*
6	*3*	*9*	*3.0*
6	*9*	*9*	*1.0*
2	*5*	*9*	*1.8*
18	*15*	*9*	*0.6*
∑ (O-E)²/E			*6.4*

(c) 카이제곱(χ^2)의 계산

그림 11 분류규칙의 χ^2 계산

$$maximum-\chi^2 = \left(\frac{support(A)support(C)}{N}\right)^2 \cdot N \cdot e \quad (2)$$

여기서 N은 훈련데이타의 레코드 수이고 e는 식 (3)이다.

$$e = \frac{1}{support(A)support(C)} + \frac{1}{support(A)(N-support(C))} + \frac{1}{(N-support(A))support(C)} + \frac{1}{(N-support(A))(N-support(C))} \quad (3)$$

따라서, 정의 4.5의 χ^2와 식 (2)의 *maximum*-χ^2을 이용하여 각 그룹의 *weighted*-χ^2은 식 (3)으로 정의되며, 가장 큰 값을 가진 그룹의 클래스를 분류될 데이타에 할당한다.

$$weighted-\chi^2 = \sum \frac{\chi^2\chi^2}{maximum-\chi^2} \quad (4)$$

5. 실험 및 평가

실험에 사용된 데이타로는 네트워크 기반 침입탐지 시스템에서 시스템에 대한 침입 여부를 판별하는, [20]에서 제공한 *TCP-dump list file* 데이타를 사용하며, 제안한 시간 분류 탐사 알고리즘을 구현한 후에 여러 실험을 통해 알고리즘의 성능을 평가하고 입력 파라미터들이 탐사된 규칙들에 어떻게 영향을 미치는지를 분석해 보았다.

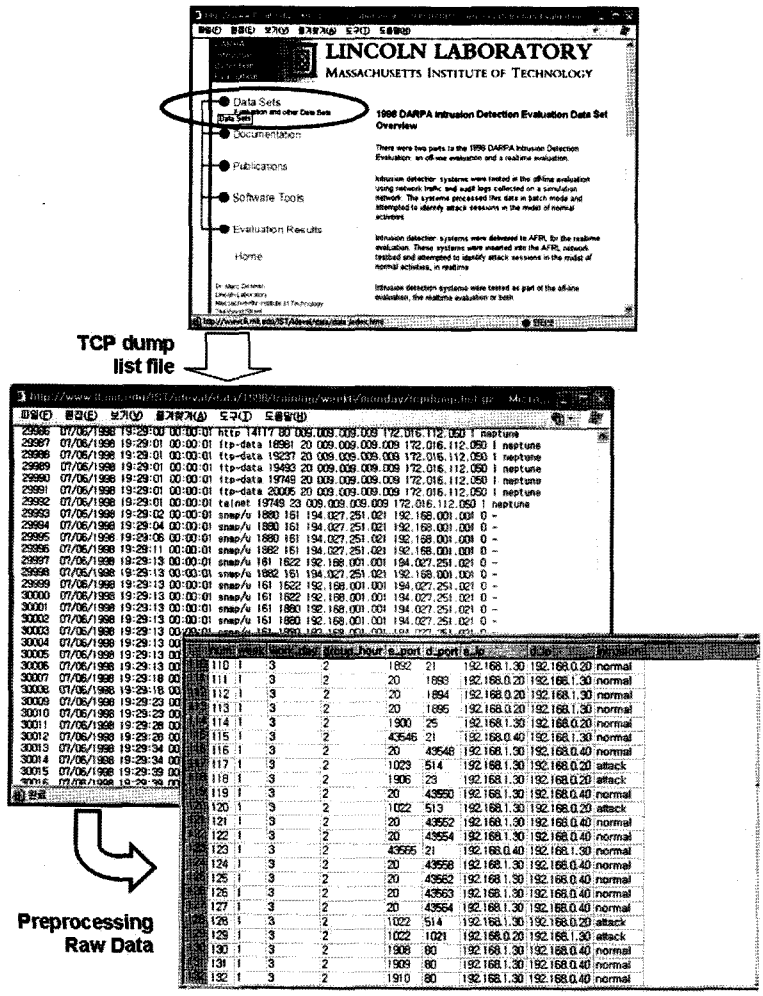
5.1 실험 데이타 생성

*TCP-dump data*는 텍스트 파일 형태의 원시 데이타이다. 따라서 체계적인 저장과 관리를 위해 관계형 데이타베이스를 사용하였다. 원시 데이타의 전처리 과정을 거쳐 데이타베이스에 저장한 결과는 그림 12에 나타내었다.

데이타베이스의 저장된 훈련데이타 집합의 테이블 스키마는 다음과 같다.

$$(RID, week, work\_day, group\_hour, s\_port, d\_port, s\_ip, d\_ip, intrusion)$$

*RID*는 테이블의 각 레코드에 대한 식별번호이고, *week, work_day, group_hour*는 원시데이타의 타임스탬프 값을 캘린더 시간 표현으로 전 처리한 것이다. *s_ip, d_ip*는 각각 근원지와 목적지 주소이고 *s_port, d_port*는 근원지와 목적지의 포트번호이다. 마지막 속성인 *intrusion*은 데이타 집합의 클래스 라벨로 *normal*과 *attack*으로 구성된다. 제안된 알고리즘에서는 시간지식의 표현으로 캘린더 스키마에 의한 시간 패턴식을 사용하였다. 이 캘린더 스키마는 사용자에 의해 정의되는 것이고, 또한 다양한 시간을 적용할 수 있다. 따라서 데이



#	Start Date	Start Time	Duration	Service	Src Port	Dest Port	Src IP address	Dest IP address	Attack Score/Name
1	07/03/1998	08:00:01	00:00:01	ecoñ	321	-	192.168.001.005	192.168.001.001	0 -
2	07/03/1998	08:00:02	00:00:01	domainu	53	53	172.106.112.020	192.168.112.020	0 -
42	07/03/1998	08:01:06	00:00:02	ftp	1027	25	172.106.113.084	135.106.113.191	0 -
43	07/03/1998	08:01:50	00:00:29	http	1106	80	172.016.112.149	167.008.112.015	0 -
9966	07/03/1998	11:49:39	00:00:01	tcpmux	1234	1	205.160.208.190	172.016.113.050	1 portsweep
...	...	...	...	...	...	...	...	...	...

pre-Processing

Class Label

#	Week	Work Day	Group Hour	Src Port	Dest Port	Src IP address	Dest IP address	Intrusion
1	1	2	1	321	-	192.168.001.005	192.168.001.001	Normal
2	1	2	1	53	53	172.106.112.020	192.168.112.020	Normal
42	1	2	1	1027	25	172.106.113.084	135.106.113.191	Normal
43	1	2	1	1106	80	172.016.112.149	167.008.112.015	Normal
9966	1	2	3	1234	1	205.160.208.190	172.016.113.050	Attack
...	...	...	...	...	...	...	...	...

그림 12 TCP dump 데이타의 전처리 과정

타 집합의 생성을 위해서 캘린더 스키마 $CS_{DARPA98}$ = (*week, work_day, group_hour*)을 정의하고, 시간대의 표현 단위인 *group_hour*에 대해 두 가지의 서로 다른 도메인 값을 표 3과 같이 나타내었다.

표 3 *group_hour*에 대한 도메인 값 정의

	group_hour	시간의 분할 범위
TG_1 (Time Granul$_1$)	{1~5}	1=[00시-06시) 2=[06시-09시) 3=[09시-18시) 4=[18시-20시) 5=[20시-24시)
TG_2 (Time Granul$_2$)	{1~3}	1=[00시-09시) 2=[09시-17시) 3=[17시-24시)
NonTemporal	-	-

전 처리 단계를 거친 훈련데이타 집합은 1주~4주로 총 데이타는 1,404,322개이고 각각의 기본시간단위에 포함된 데이타 수는 최대 200,117개, 최소 0개이다.

*TCP-dump data*에 대한 시간 클래스 연관규칙 알고리즘의 수행 결과 의미 있는 규칙 탐사를 위해 생성될 규칙에 대한 제약조건을 추가한다. 기본적으로 다차원 속성을 가지는 데이타를 위한 연관규칙 방법[22]을 따르나 알고리즘의 수행 결과 생성될 규칙에 대한 빈발 *ItemSet*의 형식은 표 4에 명시된 형태로 제한한다.

표 4 탐사될 규칙의 ItemSet 형식

형식	ItemSet
1	s_ip, d_ip
2	s_ip, d_port
3	s_ip, d_ip, d_port
4	s_ip, s_port, d_ip
5	s_ip, s_port, d_port
6	s_ip, s_port, d_ip, d_port

5.2 성능 평가

제안한 알고리즘의 성능평가를 위해 정적인 데이타마이닝에서의 입력파라미터 이외에 시간속성을 고려한 파라미터를 추가하여 실험하며, 각 파라미터 변수의 종류와 설명을 표 5와 같다.

이 절에서의 실험 내용은 크게 두 부분으로 구성된다. 첫 번째는 시간 클래스 연관규칙 탐사에 대한 것으로 입력 파라미터, 즉 사용자에 의해 주어지는 임계값들에 대한 분석을 위한 것이다. 두 번째 실험 내용은 첫 번째 실험에서의 얻어진 결과 규칙을 적용하는 것으로서 시간 클래스 연관규칙에 기반 하여 생성된 분류규칙에 대한 성능 평가로 구성된다.

표 5 실험에 사용되는 파라미터 변수

파라미터 변수	설 명
minSup	최소지지도 (절대적(개수), 상대적(%))
minConf	최소신뢰도 (%)
minFre	최소발생도 (%)
dbCover	데이타베이스 coverage count 수

5.2.1 시간 클래스 연관규칙 탐사 관련 실험

그림 13은 서로 다른 도메인을 갖는 캘린더 스키마의 두 가지 데이타 집합과 시간을 고려하지 않은 데이타 집합을 적용한 실험이다. 이 실험에서는 최소지지도를 150개, 최소발생도를 0.5로 하여 최소신뢰도 변화에 따른 생성 규칙 수의 변화를 실험하였고 실험 결과 분석은 두 가지 측면에서 볼 수 있다. 첫 번째 분석은 정적인 것보다는 시간 차원을 고려한 규칙 탐사에서 대부분 더 많은 규칙들이 생성되었음을 알 수 있다는 것이다. 그리고 두 번째는 정적인 데이타에 대한 규칙들은 신뢰도의 영향을 거의 받지 않는 것이고, 또한 시간을 고려한 집합 TG_1과 TG_2 모두 기본시간단위 동안 생성된 규칙 수의 차이가 적었다는 것이다. 그러나 두 집합은 *i-star* 캘린더 패턴으로 갱신 단계를 거치므로 규칙 수는 그림에서 보듯이 낮은 신뢰도에서 더 많은 규칙들이 생성됨을 알 수 있다. 이는 규칙 수의 변화가 신뢰도가 아닌 지지도나 발생도에 의한 것임을 의미한다. 따라서 이를 증명하기 위해서 데이타 집합 TG_1에 대해, 최소지지도와 최소발생도 변화에 따른 생성된 시간 클래스 연관규칙의 수를 비교해 보았다.

그림 14의 실험에서는 각 기본시간단위에서의 상대적인 지지도를 사용한 것과 절대적인 지지도를 주어 두 가지 경우로 실험을 하였다. 그림 13과 달리 이 실험에서는 최소지지도와 최소발생도 변화에 따른 생성 규칙 수가 많은 영향을 받는다는 것을 알 수 있으며, 최소발생도가 낮은 곳에서 규칙들의 수가 급격히 증가되었다. 이것은 생성될 규칙 수에 대해 발생도가 큰 영향을 주고 클래스 연관규칙은 알고리즘의 각 패스에서 결과부를 클래스 라벨로 고정한 형태로 규칙을 생성함으로 지지도 변하에 따라 생성 규칙 수의 변화가 매우 크게 된

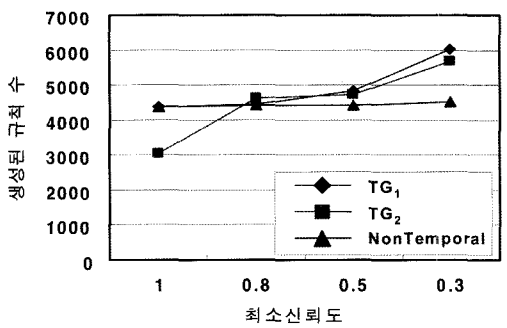


그림 13 최소신뢰도 변화에 따른 생성규칙 수

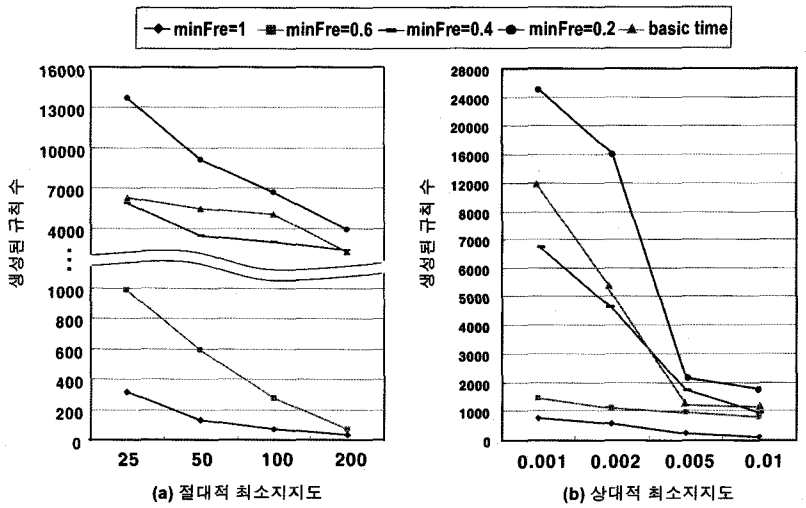
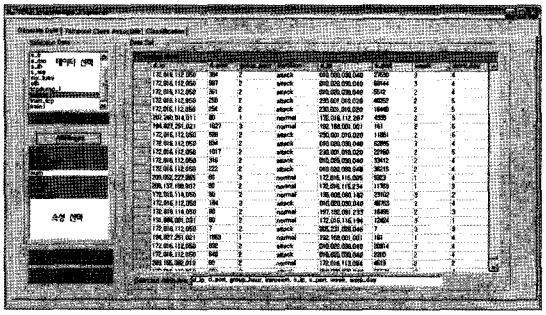
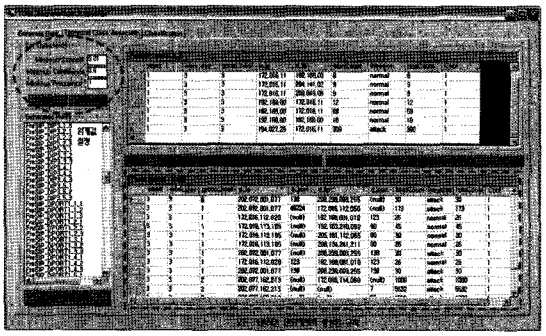


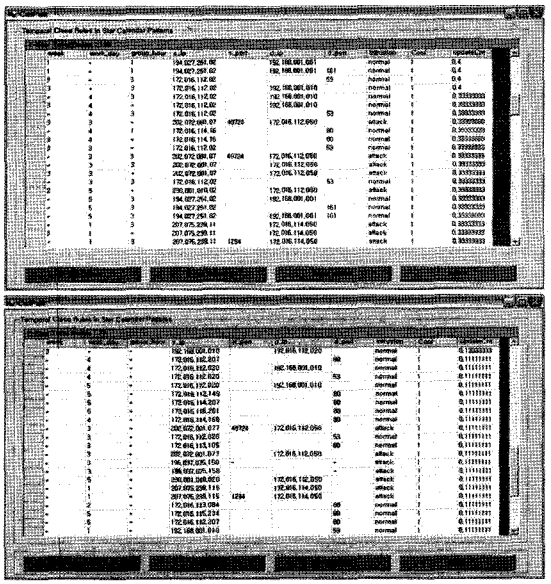
그림 14 절대적 상대적 최소지지도 변화에 따른 생성 규칙 수



(a) 훈련 데이타 선택 및 속성 선택 과정



(b) 기본시간 단위에서의 시간 클래스 연관규칙 생성 예



(c) 캘린더 패턴으로의 갱신 결과 예

그림 15 시간 클래스 연관규칙 탐사 과정

것이다. 따라서 그림 13와 14의 실험 결과, 적절한 임계치의 설정을 위해 기본시간단위에서의 생성 규칙 수 보다 적은 수를 나타내는 최소지지도와 최소발생도를 마이닝을 위한 임계값으로 설정한다. 왜냐하면 기본시간단위에서 보다 생성된 규칙 수가 더 많은 경우, 그 규칙들에는 많은 중복규칙을 포함하기 때문이다.

그림 15는 *TCP-dump* 데이타를 적용하여 시간 클래스 연관규칙을 탐사하는 과정이다.

5.2.2 분류규칙의 성능평가

이 실험에서는 그림 13, 14의 실험 결과를 바탕으로 설정된 임계값을 사용하여 탐사된 시간 클래스 연관규칙들을 분류규칙 생성을 위한 기본 규칙으로 사용한다.

또한 분류규칙의 정확성을 측정하기 위해서 *intrusion* 클래스를 *Positive* 샘플로하고 *normal* 클래스를 *Negative* 샘플로 하여 민감도와 특이도 척도를 사용하며, 최종적으로 정확도를 민감도와 특이도의 함수로 표현하였다.

$$sensitivity = \frac{True - Positive}{Positive} \quad (5)$$

$$specificity = \frac{True - Negtive}{Negtive} \quad (6)$$

$$accuracy = sens.\frac{Positive}{(Positive + Negative)} + spec.\frac{Negative}{(Positive + Negative)} \quad (7)$$

그림 16은 데이타베이스 *coverage*에 따른 생성된 분류규칙들의 정확성을 실험한 내용이다. 테스트 데이타는 *DARPA*의 *tcpdump* 데이타 중 890개의 레코드 수를 추출하였으며, 최소지지도 0.05, 최소신뢰도 0.6으로 하여 실험하였다.

시간 클래스 연관규칙으로부터의 생성된 분류규칙들의 실험을 위해 *DARPA* 데이타의 7주간의 훈련데이타 중 5주차의 데이타 집합을 테스트 데이타 집합으로 생성하였다. 각 요일별 데이타의 크기는 최대 283,859개, 최소 22,002개이다. 분류규칙의 생성을 위한 시간 클래스 연관규칙 탐사에 적용된 임계값 설정은 절대적 최소지지도 30(개), 최소신뢰도 0.6, 최소발생도 0.4로 하였다. 그림 17과 18은 21,468개의 레코드 수를 갖는 5주차의 데이타에 대한 분류규칙의 적용 결과를 보여준다. 또한 그림 17의 실험에서는 기존의 *CMAR*[21]에 대해서

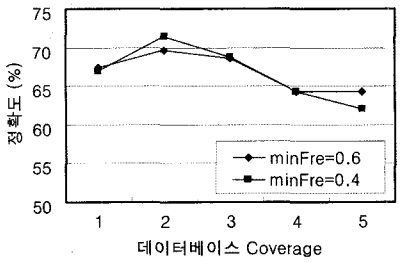


그림 16 dbCover 변화에 따른 정확도

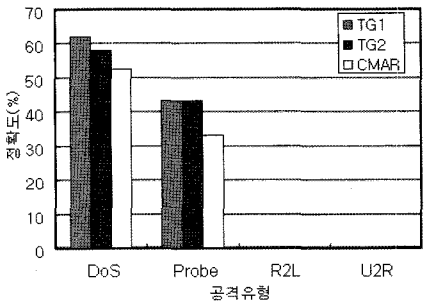


그림 17 공격 유형별 분류 결과

만 정확도 비교를 하였다. 왜냐하면, 이미 *CMAR* 알고리즘이 *CBA*와 결정트리(*C4.5*) 보다 성능이 우수한 것으로 입증되었기 때문이다.

이 실험에서 *R2L*과 *U2R* 유형의 공격은 두 유형의 특성 상, 빈발한 규칙을 이루지 않으므로 탐지가 불가능하였고, *DoS*와 *Probe* 유형의 공격에 대해서만 탐지가

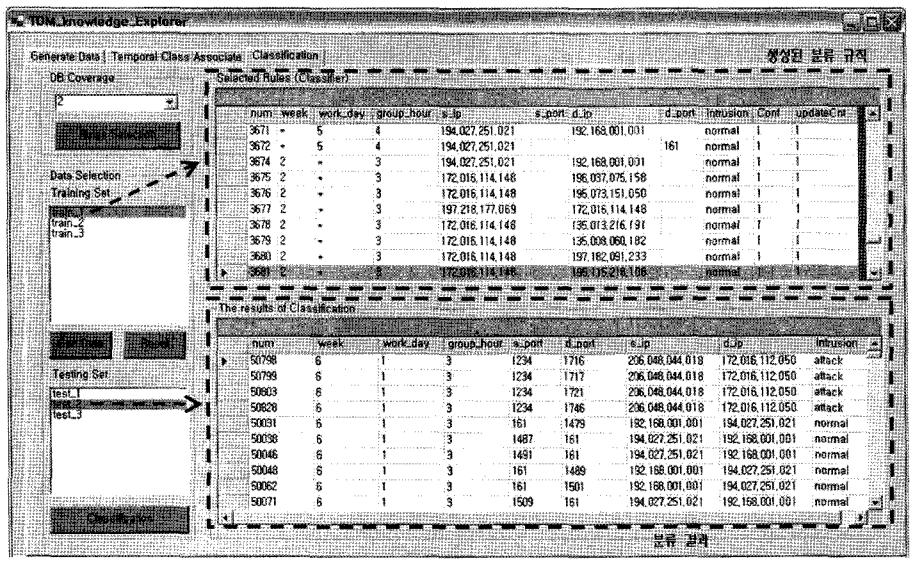


그림 18 DARPA data 분류 규칙 생성 및 분류 결과

표 6 DARPA'98 DATA에서의 공격 유형

주 \ 유형	DoS	Probe	R2L	U2R
1	smurf, neptune, teardrop	-	dict, ffb	loadmodule, perlmagic, format
2	land, back, syslog	ipsweep, portsweep	ftp-write, imap	guest
3	land, back, smurf, neptune	satan, nmap, portsweep, ipsweep,	phf, ffb, ftp-write, imap,	warez, warezmaster
4	pod, smurf, neptune, teardrop, syslog	satan, ipsweep, portsweep	ffb	warezclient, format, loadmodule
5	teardrop, smurf, pod, syslog, neptune, land	satan, ipsweep, portsweep	ffb, eject	perlmagic, format, loadmodule
6	neptune, pod, land, back, smurf, teardrop	satan, portsweep, ipsweep	phf, eject, ffb, dict	perlmagic
7	syslog, land, pod, teardrop, smurf, neptune, back	satan, portsweep, ipsweep	phf, ffb, eject	perlmagic, loadmodule

가능하였다.

표 6은 *1998 DARPA data*에 포함된 주별 모든 공격 유형을 나타낸다.

이 장에서는 사간 속성을 고려한 분류가 정적인 분류 규칙 탐사보다 정확하다는 것에 초점을 두었으며, 그 실험 결과 위의 값을 얻을 수 있었다. 그리고 제안된 시간 클래스 연관규칙의 생성 수에는 신뢰도가 아닌 지지도와 발생도에 의해 결정되었다. 또한 최적의 최소발생도는 0.4에서 0.6 사이가 가장 좋은 결과를 얻을 수 있음을 알게 되었다. 이것은 탐사된 규칙에서 강한 시간 패턴을 탐사할 수 없었음을 의미한다. 이러한 결과는 *DARPA'98* 데이타가 인위적으로 만들어진 데이타이기 때문이며, 만약 실제 네트워크 상에서의 비정상행위자들의 기록 데이타를 실험 데이타로 한다면 생성된 규칙에서 강한 시간 패턴을 탐사가 가능할 것이다.

6. 결 론

시간 데이타마이닝에 대한 기존 연구는 트랜잭션의 발생 시점이나 시간 제약조건을 추가한 형태로 시간 데이타를 다루고 있으며 시간간격을 가진 데이타에서 시간의 의미와 관계를 고려하고 있지 않다. 그러나 실세계의 많은 응용분야들에는 다양한 시간간격 데이타가 존재하며 이로부터 여러 유용한 지식을 탐사 해낼 수 있다. 따라서 이 연구에서는 캘린더 기반의 연관규칙을 기반으로 하여 시간 분류규칙을 탐사하는 기법을 제안하였다. 체계적인 연구를 위하여 시간 데이타마이닝 기법을 제안하기에 앞서, 기존 연구 내용을 분석하여 정리한 후, 이를 취합하여 새로운 시간 분류기법을 제안하였다.

이 논문에서 제안한 시간 데이타마이닝 기법은 기존의 연관규칙과 분류가 통합된 형태의 분류기법인 연관적 분류에, 다양한 시간단위에서의 시간패턴 탐사가 가능하고 탐사된 시간지식을 쉽게 사용자들이 이해 할 수 있는 캘린더 패턴식을 추가하여 확장하였다. 이 기법은 단일 시간 단위에서의 분류탐사가 이루어지는 기존의 시간 분류에 비해서, 첫째 사용자 기반의 달력 기반 시간 패턴을 탐사할 수 있고, 둘째 다양한 시간 단위를 캘린더 스키마로 정의하여 시간 간격과 주기적인 패턴을 표현할 수 있다. 뿐만 아니라 캘린더 패턴식을 포함한 클래스 연관규칙으로부터 양질의 분류규칙들만을 추출하기 위해 규칙 제거 단계를 수행하며, 이로 인해 정확한 규칙들만을 추출할 수 있다.

이 연구에서 제시된 알고리즘의 이와 같은 특성을 검증하기 위하여 *1998 DARPA tcpdump* 데이타를 적용하여 실험을 하였고, 기존의 정적인 분류(CMAR)보다 정확성 높은 예측을 할 수 있었다. 그러나 이 실험에서의 보완해야 할 점은 강한 시간 패턴을 탐사하여 검증할 수 있는 데이타에 대한 알고리즘의 적용이 필요하고, *R2L*과 *U2R* 유형의 공격을 탐지할 수 있는 방법의 연구가 필요하다. 또한 시간 클래스 연관규칙 탐사과정에서 기존의 *CBA* 알고리즘을 확장하였으므로 분류규칙 탐사에 높은 실행시간을 나타냈다. 따라서 향후 연구 방향은 제안한 알고리즘의 실행 시간을 줄일 수 있도록 [12]에서 제안된 *FP-growth*와 같은 향상된 알고리즘으로 확장하는 것이며, 캘린더 표현의 강한 시간 패턴을 탐사 할 수 있는 실제 데이타에 알고리즘을 적용하여 검증하는 것이다.

참 고 문 헌

[1] J. F. Roddick and M. Spiliopoulou, "Temporal data mining: survey and issues," Research Report ACRC-99-007, University of South Australia, 1999.

[2] B. Ozden, S. Ramaswamy and A. Silberschatz, "Cyclic association rules," 11th International Conference on Data Engineering, Orlando, May. 1998.

[3] X. Chen and I. Petrounias, "A Framework for temporal data mining," 9th International Conference on Database and Expert System Applications, 1998.

[4] X. Chen, I. Petrounias and H. Heathfield, "Discovering temporal association rules in temporal database," International Workshop on Issues and Applications of Database Technology, 1998.

[5] Y. Li and P. Ning, "Discovering Calendar-based Temporal Association Rules," In Proc. of the 8th International Symposium on Temporal Representation and Reasoning, 2001.

[6] S. Ramaswsmy, S. Mahajan and A. Silberschatz, "On the discovery of interesting patterns in association rules," the VLDB Conference, New York City, Sep., 1998.

[7] J. R. Quinlan, "Induction of decision trees, Machine Learning," Tom M. Mitchell Publishers, 1986.

[8] J. R. Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufman Publishers, 1993.

[9] J. Han and M. Kamber, "Data Mining: Concepts and Techniques," Morgan Kaufmann Publishers, 2000.

[10] Chris Sinclair, Lyn Pierce and Sara Matzner, "An Application of Machine Learning to Network Intrusion Detection," 15th Annual Computer Security Applications Conference Dec., Phoenix, Arizona, 1999.

[11] B. Liu, W. Hsu and Y. Ma, "Integrating classification and association rule mining," In Proc. of the 4th International Conference Knowledge Discovery and Data Mining, 1998.

[12] W. Li, J. Han and J. Pei, "CMAR: Accurate and Efficient Classification Based on Multiple Association Rules," In Proc. 2001 International Conference on Data Mining.

[13] W. Lee and S. Stolfo, "Data Mining Approaches for Intrusion Detection," In Proc. of the 7th USENIX security Symposium, 1998.

[14] W. Lee and S. Stolfo, "A Data Mining Framework for Building Intrusion Detection Models," IEEE Symposium on Security and Privacy, 1999.

[15] M. J. Lee, M. S. Shin, K. H. Ryu and K. Y. Kim, "Design and Implementation of Alert Analyzer with Data Mining Engine," In Proc. of the 5th International Conference on Intelligent Data Engineering and Automated Learning, 2003.

[16] M. S. Shin and K. H. Ryu, "Data Mining Methods for Alert Correlation Analysis," International Journal of Computer and Information Science (IJCIS), Vol.4, No.4, Dec., 2003.

[17] D. Barbara, J. Couto and N. Wu, "ADAM: Detection Intrusion by Data Mining," In Proc. of 2th IEEE Information Assurance Workshop, 2001.

[18] 이현규, 이양우, 김룡, 서성보, 류근호, 박진수, "시간연관규칙과 분류규칙을 이용한 비정상행위 탐지 기법", 한국정보처리학회 춘계학술발표 논문집, 제10권 제1호, pp. 1579-1582, 2003.

[19] Jin Suk Kim, Hohn Gyu Lee, Sungbo Seo and Keun Ho Ryu, "CTAR: Classification based on Temporal Class-Association Rules for Intrusion Detection," In Proc. of the 4th International Workshop on Information Security Applications, Vol.4, pp. 101-113, 2003.

[20] "1998 DARPA Intrusion Detection Evaluation Data," http://www.ll.mit.edu/IST/ideval/data/data_index.html

[21] http://www.csc.liv.ac.uk/~frans/KDD/Software/CMAR/cmar.html

[22] R. Srikant and R. Agrawal, "Mining Quantitative Association Rules in Large Relational Tables," In Proc. of the 1996 ACM SIGMOD International Conference on Management of Data, pp. 1-12, 1996.

이 현 규

2002년 경기대학교 전자계산학과 졸업(이학사). 2004년 충북대학교 대학원 전자계산학과 석사 졸업(이학석사). 2005년~현재 충북대학교 대학원 전자계산학과 박사 과정. 관심분야는 시간 데이타베이스, 시공간 데이타베이스, 데이타마이닝, 생체신호 처리, 바이오인포매틱스 등

노 기 용

1981년 충남대학교 물리학과 졸업(이학사). 1995년 충남대학교 대학원 전자계산학과 졸업(이학석사). 2004년 충북대학교 대학원 전자계산학과 박사 졸업(이학박사). 1988년~현재 한국표준과학연구원. 관심분야는 데이타베이스 설계, 이미지 처리, ATM 등

서 성 보

1999년 서원대학교 전산학과 졸업(학사) 2001년 충북대학교 전산학과 졸업(이학석사). 2001년~2003년 한국전자통신연구원 우정기술연구센터 위촉 파견연구원 2005년 미국 North Carolina State Univ. 방문 연구원. 2002년~현재 충북대학교 전산학과 박사과정. 관심분야는 시공간 데이타베이스, 데이타마이닝, 다차원 센서 데이타 분석 등

류 근 호

정보과학회논문지 : 데이타베이스
제 32 권 제 5 호 참조