http://dx.doi.org/10.3745/KIPSTD.2012.19D.2.139

데이터 스트림 환경에서 효율적인 빈발 항목 집합 탐사 기법

서 복 일[†] · 김 재 인[††] · 황 부 현[†††]

요 약

데이터 마이닝은 다양한 분야에서 축적된 데이터로부터 필요한 지식을 탐사하기 위하여 널리 이용되고 있다. 연관규칙을 탐사하기 위하여 이벤트의 빈발 횟수에 기반을 둔 많은 방법들이 존재하지만, 이들은 이벤트가 연속적으로 발생하는 스트림 환경에는 적합하지 않다. 또한 실시간으로 연관규칙을 탐사해야 하는 스트림 환경에 적용하기에는 많은 비용이 든다. 이 논문에서는 스트림 환경에서 연관규칙을 탐사하기 위한 새로운 방법을 제안한다. 제안하는 방법은 데이터 스트림에서 목적 이벤트의 발생 간격에 따른 가변 윈도우로부터 이벤트의 존재 유무에 근거한 COBJ(Count object) 계산법을 이용하여 데이터 항목을 추출한다. 추출된 데이터는 FPMDSTN(Frequent Pattern Mining over Data Stream using Terminal Node) 알고리즘을 통해 실시간으로 연관규칙을 탐사한다. 실험 결과를 통해 제안하는 방법이 기존의 방법에 비해 스트림 환경에 효율적임을 보인다.

키워드 : 데이터마이닝, 데이터 스트림, 빈발항목집합, 실시간 마이닝

A Method for Frequent Itemsets Mining from Data Stream

BokII Seo[†] · Jaeln Kim[††] · BuHyun Hwang[†††]

ABSTRACT

Data Mining is widely used to discover knowledge in many fields. Although there are many methods to discover association rule, most of them are based on frequency-based approaches. Therefore it is not appropriate for stream environment. Because the stream environment has a property that event data are generated continuously. it is expensive to store all data. In this paper, we propose a new method to discover association rules based on stream environment. Our new method is using a variable window for extracting data items. Variable windows have variable size according to the gap of same target event. Our method extracts data using COBJ(Count object) calculation method. FPMDSTN(Frequent pattern Mining over Data Stream using Terminal Node) discovers association rules from the extracted data items. Through experiment, our method is more efficient to apply stream environment than conventional methods.

Keywords : Data Mining, Data Stream, Frequent Itemsets, Real-time Mining

1. 서 론

데이터 마이닝은 의사 결정에 필요한 새롭고 유익한 정보와 지식을 얻기 위하여 축적된 데이터에서 패턴, 연관, 변화, 예외, 규칙 등 통계적으로 중요한 구조와 사건들을 찾아내는 반자동 시스템이다. 축적된 데이터에 잠재되어 있는 유용한 정보를 추출하기 위해 여러 가지 알고리즘들이 연구되었으며 금융 데이터 분석, 센서 네트워크 모니터링, 약물 부작용 감시, 의료 분야에서와 같이 여러 분야에서 활용되고 있다. 데이터의 발생이 매초마다 무수히 발생되는 현대 사회에서 데이터 마이닝의 활용성은 더욱 커지고 있다[1][2][10].

데이터 마이닝의 기법에는 연관 규칙 탐사(association rule discovery), 항목 분류(classification), 군집화(clustering), 요약(summarization), 순차 패턴 탐사(sequential pattern discovery) 등이 있다. 그 중 연관 규칙 탐사는 데이터들의 동시 발생 가능성을 탐사하는 기법으로써 한 데이터 항목들의 그룹과 다른 데이터 항목들의 그룹 사이에 존재하는 연관성 탐사에 응용된다[1][2]. 연관 규칙은 의료분야에서 특정 질병이 발생하는 데 원인이 되는 다른 이벤트를 탐사하는 데에도 이용된다[10][11].

이 논문에서 다루는 데이터 스트림은 정적인 데이터와 비교했을 때 다음과 같은 특성을 지니고 있다[15].

• 특성 1. 데이터 스트림은 연속적이며 경계가 없다. 정적

※ 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2010-0005647).
† 준 회 원 : 전남대학교 전자컴퓨터공학부 석사과정
†† 준 회 원 : 전남대학교 전자컴퓨터공학부 박사과정
††† 종신회원 : 전남대학교 전자컴퓨터공학부 교수(교신저자)
논문접수 : 2011년 10월 14일
수 정 일 : 1차 2011년 11월 24일, 2차 2011년 12월 19일
심사완료 : 2011년 12월 20일

인 데이터와 달리 데이터가 무한하게 발생하며 트랜잭션에 대한 구분이 없다. 또한, 전체 데이터를 대상으로 지식 탐사를 수행할 수 없기 때문에 데이터 스트림에서 유용한 지식을 탐사하기 위해서는 중요한 이벤트를 추출하는 방법과 실시간으로 탐사를 수행할 수 있는 기법이 필요하다.

• 특성 2. 데이터 스트림은 데이터의 분포가 불규칙적이다. 데이터의 분포는 일반적으로 시간에 따라 변한다. 현재 빈발하지 않은 이벤트가 미래에는 빈발한 이벤트가 될 수도 있고 그 반대가 될 수도 있다.

• 특성 3. 빈발하지 않은 이벤트가 빈발 이벤트보다 더 가치 있는 유용한 데이터일 가능성이 크다. 예를 들어, 환자에게서 발생되는 생체 데이터의 경우, 빈발하게 발생하는 정상적인 이벤트보다 빈발하지 않은 비정상 이벤트가 더 가치가 크다. 따라서 스트림 환경에서 빈발하지 않은 이벤트에 대해서 탐사할 수 있는 새로운 기법이 필요하다.

빈발 패턴(frequent pattern)은 데이터 집합에서 빈번하게 발생하는 패턴(항목집합, 부분순차, 부분구조)들이다. 빈발 패턴을 발견하는 일은 데이터 사이의 연관성, 상관성, 그리고 많은 흥미로운 관계를 탐사하는데 필수적인 역할을 한다. 빈발 패턴 마이닝(frequent patterns mining)은 데이터 스트림 마이닝 분야에서 가장 폭 넓게 연구되어지고 있는 분야로써 환경의 특성을 고려하여 얼마나 빠르게 빈발 항목 집합을 탐사할 수 있는지에 따라서 알고리즘의 성능이 좌우된다. 데이터 스트림 환경에서 탐사속도는 가장 큰 연구화제이며 다양한 알고리즘들이 제안되었다[2][5].

이 논문에서는 데이터 스트림 으로부터 실시간으로 유용한 정보를 추출하기 위해 효율적으로 빈발항목 집합을 탐사하는 방법을 소개한다. 제안하는 방법은 다음과 같다. 먼저, 타겟 이벤트의 발생 간격에 따라서 가변적인 크기를 갖는 윈도우를 이용하여 데이터를 선별한다. 가변 윈도우를 통해 타겟 이벤트의 발생과 연관성이 높은 이벤트를 선별할 수 있다. 가변 윈도우를 통해 얻어진 데이터는 COBJ 계산법에 의해서 트랜잭션으로 만들어진다. COBJ 계산법은 이벤트의 빈발 횟수와 관계없이 이벤트의 발생 유무에 중점을 둔다. 예를 들어, 만약 이벤트가 한 번 이상 발생하면 이벤트의 빈발 횟수와 관계없이 해당 이벤트의 빈발 횟수는 1이 된다. 마지막으로 기존의 빈발 패턴 탐사 기법인 FP-Tree를 변형한 FPMDSTN(Frequent pattern Mining over Data Stream using Terminal Node)알고리즘을 이용하여 빈발패턴을 탐사한다. FPMDSTN는 트랜잭션의 끝 이벤트에 대한 정보 테이블을 가지며, 이를 이용해서 트랜잭션의 삽입, 삭제에 대한 연산을 줄일 수 있다.

이 논문의 구성은 다음과 같다. 2장에서는 데이터 스트림 환경에서 빈발항목 집합을 탐사하기 위한 기존 연구에 대해서 소개한다. 3장에서는 이 논문에서 제안하는 방법으로써 가변 윈도우를 이용한 이벤트 선별, COBJ 계산법에 의한 이벤트 추출, 그리고 실시간으로 빈발 항목집합을 탐사하기 위한 FPMDSTN 알고리즘에 대해서 설명한다. 4장에서는 실험 결과에 대해서 기술하고, 5장에서는 실험 결과를 분석한다. 마지막으로 6장에서는 향후 연구에 대해서 기술한다.

2. 관련 연구

데이터 스트림은 크기가 무한하여 전체 데이터를 탐색할 수 없기 때문에 전통적인 마이닝 방법을 통해 지식을 탐사할 수 없다. 따라서 데이터 스트림에서 연관 규칙을 탐사하기 위해서는 여러 번의 데이터 탐색 없이 제약된 환경에서 실시간으로 데이터들의 연관 규칙 정보를 분석하여야 한다. FP-Growth[1] 알고리즘은 이 문제를 해결하기 위해 FP-tree를 사용하여 후보 항목의 생성 없이 두 번의 데이터베이스 탐색만으로 빈발항목 집합을 탐사하였다. 그러나 FP-Growth 알고리즘은 데이터 스트림에 적용하기에 한계가 있다.

데이터 스트림의 또 다른 특성 중의 하나인 연속적인 특성을 고려하여 발생되는 데이터의 일정 주기를 하나의 윈도우로 정의할 수 있다. [11]에서는 슬라이딩 윈도우 방법을 정적인 크기를 가지는 고정 윈도우를 이용하여 데이터 스트림 환경에 적용시키는 방법을 제안하였다. 슬라이딩 윈도우는 모든 데이터 혹은 특정 표본에 대한 계산을 수행하는 대신, 오직 최근 데이터에 기초하여 의사결정을 할 수 있다. 모든 시간 t에서 새로운 원소들이 도착한다고 가정하고 이 원소는 t+w 시간에 소멸한다. 여기서 w는 윈도우의 크기 또는 길이이다. 슬라이딩 윈도우는 데이터의 작은 윈도우만 저장되기 때문에 요구되는 메모리를 절약할 수 있었다. 또한 고정 윈도우는 연속적인 특성을 지니는 스트림 환경에서 전체 데이터 중 부분 데이터만을 대상으로 수행되어 스트림 환경에 적용하기에 효과적이다. 그러나 이 방법은 위에서 언급한 데이터 스트림의 특성 중 두 번째인 분포가 불균형하다는 점을 해결하는 데에는 한계가 있었고 트랜잭션이 발생할 때마다 새로운 트리를 구축하여 마이닝을 수행하였기 때문에 자원의 낭비가 있었다.

[12]에서는 비트를 이용하여 새로운 트리를 구축하지 않고 삭제할 트랜잭션과 삽입할 트랜잭션만으로 트리를 재구성하는 방법을 제안하였다. 제안된 방법은 트리의 전체를 재구성하지 않고 삭제, 삽입할 부분만을 재구성하여 실시간으로 빈발패턴을 탐사하기에 적합하였다. 그러나 모든 트랜잭션의 삽입 순서를 비트로 표현함으로써 새로운 트랜잭션이 삽입 될 때마다 전체 트리를 순회해야 하기 때문에 불필요한 자원 소모가 발생했다.

이 논문에서는 데이터 스트림에서 유용한 트랜잭션을 추출하는 가변 윈도우와, COBJ 계산법, 그리고 FP-Growth 성장 알고리즘을 이용하여 실시간으로 빈발항목 집합을 생성하는 FPMDSTN 알고리즘을 소개한다.

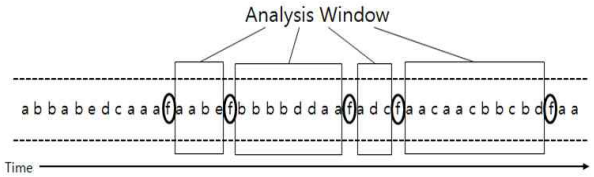
3. 데이터 스트림 환경에 효율적인 빈발 항목 집합 탐사

이 절에서는 이 논문에서 제안하는 데이터 스트림에서 중요한 데이터를 선별하기 위해 가변 윈도우와 이벤트의 발생 유무에 근거한 COBJ 계산법, 그리고 효율적인 빈발 항목 집합을 탐사하기 위한 FPMDSTN 알고리즘을 소개한다.

3.1 가변 윈도우

데이터 스트림은 연속적인 특성을 지니고 있기 때문에 데이터의 시작과 끝에 대한 경계가 없다. 데이터 스트림에서 윈도우는 하나의 트랜잭션을 구성하는 중요한 역할을 한다. 만약 윈도우의 크기가 작으면 중요한 이벤트를 포함하지 않을 가능성이 높고 윈도우의 크기가 크면 불필요한 이벤트를 포함할 가능성이 높아진다. 이처럼 윈도우의 크기에 따라 산출되는 규칙이 달라지기 때문에 윈도우의 크기를 결정하는 문제는 매우 중요하다.

이 절에서는 데이터 스트림에서 데이터를 트랜잭션으로 변환하기 위한 가변 윈도우를 소개한다. 데이터 스트림 환경에서 이벤트는 연속적으로 발생되며 시간 속성을 가진다. 이 절에서 언급되는 타겟 이벤트는 해당 분야의 전문가에 의해 선택되며 다른 이벤트의 발생이나 주변 환경의 변화에 큰 영향을 미치는 이벤트를 의미한다. 제안하는 가변 윈도우는 다음과 같이 크기를 설정한다. 가변 윈도우는 동일한 타입의 타겟 이벤트가 서로 다른 시점에 발생했을 때 두 타겟 이벤트 사이의 간격을 크기로 갖는다. 이렇게 생성된 가변 윈도우는 타겟 이벤트와 연관된 이벤트를 포함하고 있을 가능성이 높다.



(그림 1) 가변 윈도우를 이용한 데이터 선별

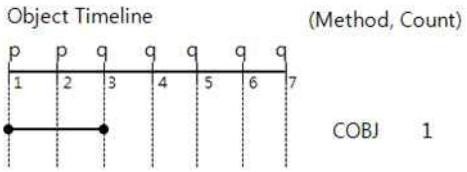
(그림 1)은 데이터 스트림에서 가변 윈도우를 설정하는 예를 보여준다. 각각의 이벤트는 Time 라인에 따라서 시간 속성을 지니며 위의 예에서는 왼쪽(과거)에서 오른쪽(현재)으로 데이터가 발생한다. 또한 데이터 스트림 환경에서는 정상적인 상태의 이벤트보다 비정상 상태를 나타내는 이벤트가 더 의미가 있기 때문에 정상 상태에 대한 이벤트는 제외한다. 기호(a, b, c, d, e)은 하나의 스트림에서 발생되는 비정상적인 상태를 의미한다. 기호 f를 타겟 이벤트라고 가정했을 때, 하나의 타겟 이벤트 f 와 이전에 발생한 타겟 이벤트 f 사이의 간격을 하나의 가변 윈도우로 설정할 수 있다. 위의 그림에서는 기호 f에 대해서 4개의 가변 윈도우(a, a, b, e), (b, b, b, b, d, d, a, a), (a, d, c), (a, a, c, a, a, c, b, b, c, b, d) 를 얻을 수 있다. 처음에 발생한 f의 이전에 발생한 이벤트들(a, b, b, a, b, e, d, c, a, a, a)은 이전 f의 발생이 불분명하기 때문에 가변 윈도우 생성에서 제외된다.

<표 1>은 (그림 1)의 데이터 스트림에서 가변 윈도우를 이용하여 데이터를 선별한 것을 나타낸다. <표 1>에서 W_1에 포함된 이벤트 들은 데이터 스트림에서 두 번째로 발생하는 타겟 이벤트와 관련된 이벤트들을 포함하고 있다. 또한 W_2 에 포함된 이벤트 들은 데이터 스트림에서 세 번째로 발생하는 타겟 이벤트와 관련된 이벤트들을 포함하고 있다. 이처럼 각각의 가변 윈도우에 포함된 이벤트들은 각각의 타겟 이벤트들과 관련된 이벤트들을 포함하고 있기 때문에 좋은 정보를 탐사하기에 유용하다. 선별된 각 윈도우 내에 있는 이벤트는 COBJ 계산법에 의해 트랜잭션으로 변환된다.

〈표 1〉 데이터 스트림에서 가변 윈도우를 이용하여 선별된 이벤트

윈도우	이벤트
W_1	a a b e
W_2	b b b b d d a a
W_3	a d c
W_4	a a c a a c b b c b d

3.2 COBJ계산법을 이용한 이벤트 추출



(그림 2) COBJ에 의한 객체 계산법

이 절에서는 COBJ 방법을 이용한 이벤트 추출법을 소개한다[16]. COBJ 계산법은 최소한 객체의 시간선(Timeline)에서 주어진 순차의 한 번 발생을 찾는다. (그림 2)에서 순차 < (p) (q) > 가 객체의 시간 선에서 여러 번 나타나지만, 단지 한 번 카운트 된다. (p 가 t = 1에서 발생하고 q는 t = 3에서 발생) 트랜잭션 추출방법은 다음과 같다. 3.1절에 소개된 가변 윈도우를 통해 선별된 윈도우내의 이벤트들을 COBJ방법을 이용하여 트랜잭션으로 추출한다. <표 2>는 <표 1>의 가변 윈도우에 포함된 이벤트들을 COBJ 계산법을 이용하여 얻는 것이다. 예를 들어 W_1 은 이벤트(a, a, a, e)를 포함하고 이를 COBJ 방법으로 계산하면 이벤트(a, e)가 된다. 같은 방법으로 W_2의 이벤트(b, b, b, b, d, d, a, a)는 이벤트 (b, d, a)로 계산된다. 각각의 가변 윈도우에 포함된 이벤트들은 COBJ 계산방법에 의해 계산되어 가변 윈도우 당 한 개의 트랜잭션을 이루며 이것은 빈발항목 집합 탐사에 이용된다.

데이터 스트림 환경에서는 실제 환경에서 발생하는 하나의 이벤트를 짧은 시간동안 연속적으로 발생시키기 때문에 수집되는 데이터의 전처리 과정이 필요하다. 따라서 이 절에서는 소개한 COBJ 계산법은 데이터의 발생이 무한한 센서 데이터와 같은 데이터 스트림 환경에 적합하다.

〈표 2〉 COBJ 계산법을 이용해 추출한 이벤트

트랜잭션	이벤트
T_1	a b e
T_2	b d a
T_3	a d c
T_4	a c b d

3.3 FPMDSTN(Frequent Patterns Mining over Data Stream using Terminal Node) 알고리즘

이 절에서는 가변 윈도우와 COBJ 계산법에 의해 추출된 트랜잭션을 통해 연관 규칙을 탐사하는 방법을 소개한다. 제안하는 FPMDSTN 알고리즘은 FP-Growth 알고리즘을 이용하여 실시간으로 연관 규칙을 탐사한다. FPMDSTN 알고리즘은 트랜잭션이 생성될 때마다 새로운 트리를 생성하지 않고 기존에 생성된 트리를 변형하여 새로 발생한 트랜잭션을 트리 구조에 추가시키기 때문에 기존의 FP-Growth 알고리즘을 이용한 FP-Tree에 비해 데이터 스트림에 적용하기에 알맞다.

3.3.1 트리 구축

먼저 트리를 구축하기 위해 <표 3>과 같이 트랜잭션들을 시간순서에 따라 정렬한다. 그리고 각 트랜잭션내의 이벤트들을 알파벳순으로 정렬한다. 이벤트의 정렬 과정을 통해 마이닝 과정에서 불필요한 자원의 낭비를 줄일 수 있다.

〈표 3〉 오름차순으로 정렬된 이벤트

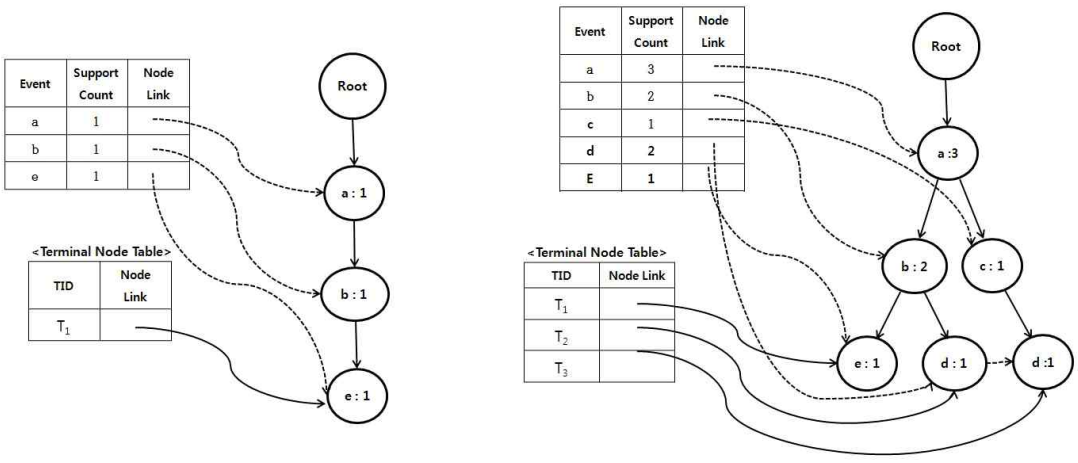
트랜잭션	이벤트
T_1	a b e
T_2	a b d
T_3	a c d
T_4	a b c d

제안하는 FPMDSTN 알고리즘은 FP-Growth 알고리즘을 기반으로 한다. FPMDSTN 알고리즘은 데이터 집합에서 한 번에 한 트랜잭션을 읽어서 각 트랜잭션을 FP-트리에 있는 한 경로로 사상함으로써 구성한다. 서로 다른 트랜잭션들이 공통으로 여러 항목들을 포함할 수도 있으므로 그것들의 경로는 중복된다. 경로들이 서로 중복되면 될수록 FP-트리를 사용하여 더 많이 데이터를 압축할 수 있다. 만약 FP-Tree의 크기가 메인 메모리 내에 맞도록 충분히 작으면, 디스크에 저장된 전체 데이터에 반복된 경로들을 만드는 대신 메모리 내의 구조에서 직접 빈발 항목집합들을 추출할 수 있다.

(그림 3)은 <표 3>의 트랜잭션에 대한 FPMDSTN 알고리즘의 트리 구축 과정을 보여준다. 기존의 FP-Growth 알고리즘과 달리 메모리 공간에 트랜잭션의 마지막 이벤트에 대한 포인터를 저장할 수 있는 자료구조를 추가로 필요로 한다(그림 3(a), (b)의 Terminal Node). 예를 들어, (그림 3(a))는 트랜잭션 1 (a, b, e)에 대한 트리 구조이다. 각각의 이벤트에 대한 지지도 횟수와 노드에 대한 링크를 정보를 저장하는 테이블을 가지고 있으며 추가로 트랜잭션에 대한 마지막 이벤트를 저장할 수 있는 Terminal Node 로 명명된 저장 테이블이 존재한다. (그림 3(a))의 경우 이벤트 e가 트랜잭션의 마지막이므로 이벤트 e에 대한 포인터를 저장하게 된다. (그림 3(b))를 통해 트랜잭션 1, 2, 3 이 삽입된 후의 트리 구성을 확인할 수 있다. 각각의 이벤트에 대한 정보 테이블이 갱신되는 것과 동시에 마지막 이벤트에 대한 포인터가 Terminal Node 테이블에 따로 저장된다. Terminal Node에 저장된 정보는 이 후 트랜잭션의 삭제에 이용된다.

3.3.2 터미널 노드 테이블(Terminal Node Table)을 이용한 트랜잭션의 삭제와 삽입

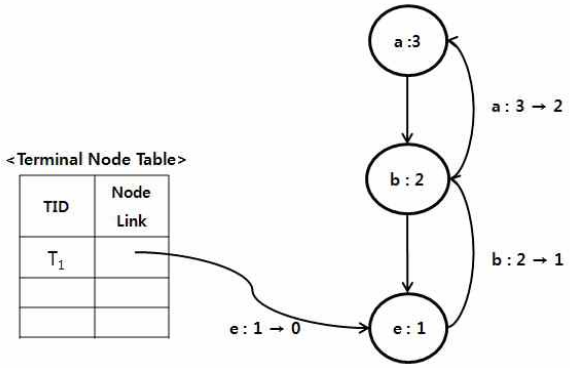
(그림 4)는 터미널 이벤트 테이블을 사용하여 트랜잭션을 삭제하는 과정을 보여준다. 터미널 노드 테이블은 각각의 트랜잭션의 마지막 이벤트에 대한 포인터를 갖고 있다. 이를 통해 트랜잭션의 삭제가 이루어져야 할 경우, 해당 트랜잭션에 대한 정보를 터미널 노드 테이블에서 얻어올 수 있다. (그림 4)의 경우, 윈도우의 크기를 3이라고 정의했을 때, 우리는 트랜잭션 1의 정보를 삭제하고 트랜잭션 4의 정보를 트리에 삽입해야한다. 따라서 터미널 이벤트 테이블에서 트랜잭션 1의 마지막 이벤트 노드에 대한 정보를 얻어 와서 bottom-up 방식으로 루트까지 트리를 탐색하며 지지도 횟수를 감소시켜준다. 예를 들어, 트랜잭션 1을 삭제하려 할 경우 노드 (e)는 지지도 횟수가 1에서 0으로 감소하고 노드



(a) 트랜잭션 1에 대한 FPMDSTN (b) 트랜잭션 1, 2, 3에 대한 FPMDSTN

(그림 3) 〈표 3〉의 트랜잭션에 대한 FPMDSTN 알고리즘의 트리 구축 과정

(b) 는 2에서 1로, 그리고 노드 (a)는 3에서 2로 감소하게 된다. 삭제 과정은 루트노드까지 진행된다. 이 때, 지지도 횟수가 0이 되는 노드 (e)의 경우는 트리에서 삭제가 되며 이 벤트 e에 대한 정보 테이블 또한 갱신되게 된다. 제안하는 방법은 전체 트리를 순회하지 않고 삭제할 트랜잭션의 노드만을 방문하기 때문에 자원의 낭비를 줄일 수 있다.

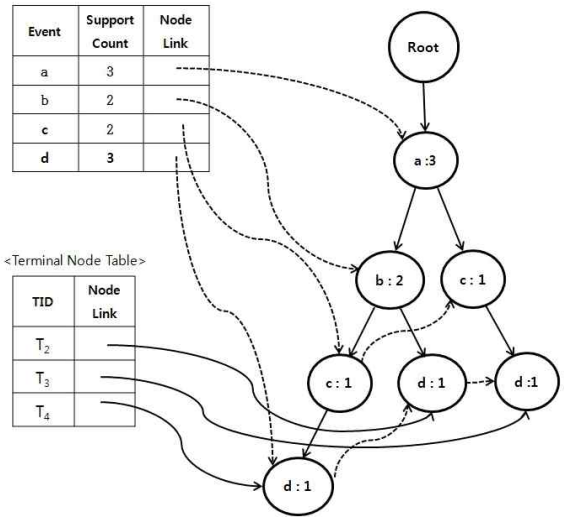


(그림 4) 트랜잭션 1의 삭제 과정

(그림 5)는 트랜잭션 1을 삭제한 후 트랜잭션 4을 삽입한 트리의 모습이다. 트랜잭션 1을 삭제하기 전의 터미널 이벤트 테이블과 달리 트랜잭션 4을 삽입한 후의 터미널 이벤트 테이블은 삭제한 트랜잭션에 대한 포인터가 삭제되고 새로운 트랜잭션의 마지막 이벤트를 가리키는 포인터가 새로 추가된 것을 볼 수 있다. 또한 추가된 트랜잭션에 맞추어 정보 테이블이 갱신되었으며 Node Link 또한 새롭게 갱신된다. 이처럼 터미널 이벤트 테이블은 트랜잭션의 삽입과 삭제와 밀접한 관계를 갖고 있으며 트랜잭션의 삽입과 삭제가 발생할 때마다 동시에 갱신된다.

3.3.3 마이닝 과정

FPMDSTN 알고리즘의 마이닝은 다음과 같이 진행된다.



(그림 5) 트랜잭션 1을 삭제한 후 트랜잭션 4을 삽입한 트리

먼저 길이 1인 빈발 패턴에서 시작하여 조건부 패턴 베이스를 생성하고, 조건부 트리를 생성한다. 최소 지지도 횟수를 만족하지 않는 이벤트는 생성 과정에서 제외된다. 그리고 이 트리에 대해서 재귀적으로 마이닝을 수행한다. 패턴 증가는 접미부 패턴과 조건부 트리로부터 생성된 빈발 패턴을 접합함으로써 얻어진다.

예를 들어, (그림 3(a)) 경우에서 최소 지지도 횟수를 2라고 하면 이벤트 c 와 e 는 마이닝 과정에서 제외된다. 이벤트 d에 대해서 조건부 패턴 베이스({ a, b : 1 }, { a : 1 }) 를 생성한 후 조건부 트리({ a : 2 })를 생성한다. 조건부 트리 생성 과정에서 이벤트 b는 최소 지지도 카운트 2를 만족하지 않기 때문에 제외된다. 이 후 절차는 FP-Growth 알고리즘[1]과 동일하기 때문에 이 논문에서는 생략한다. FPMDSTN에서 마이닝은 최초 윈도우 생성 후 수행되며 트랜잭션의 삽입과 삭제가 발생되어 트리가 변형될 때마다 수행된다.

3.3.4 FPMDSTN 알고리즘

데이터 스트림에서 발생하는 데이터를 FPMDSTN 알고리즘에 적용하기 위해 실시되는 처리 과정은 다음과 같다. 먼저 해당 분야의 전문가에 의해서 마이닝 대상이 되는 타겟 이벤트를 지정한다. 이 후 타겟 이벤트의 발생 간격에 따라서 데이터가 선별된다. 선별된 데이터는 COBJ계산법에 의해서 트랜잭션으로 추출된다. FPMDSTN 알고리즘은 사용자가 입력한 윈도우의 크기를 유지하면서 빈발항목집합을 탐사한다.

제안하는 FPMDSTN 알고리즘의 수행순서는 아래와 같다. 먼저 “null” 로 표기된 노드를 생성한 뒤(a), 트랜잭션이 생성될 때마다 오름차순으로 트랜잭션 내의 이벤트를 정렬하고 트리를 형성해나간다.(b) 이 때, 트랜잭션의 마지막 이벤트를 저장하는 노드의 주소 포인터를 Terminal Node Table 에 저장한다. 저장하는 순서는 First-in-First-out를 유지하기 위해 큐를 이용한다.(c) 생성된 트랜잭션의 수가 최초 사용자가 지정한 윈도우의 크기와 같다면 빈발항목집합을 탐사한다.(d) 빈발항목집합 탐사 과정은 기존의 FP-Tree와 동일하기 때문에 설명은 생략한다. 이 후, 새로운 트랜잭션이 발생할 때마다 Terminal Node Table을 이용하여 가장 오래된 트랜잭션을 삭제하며 삭제과정은 "null"노드까지 부모 노드를 방문하며 지지도 카운트를 1 감소시켜 준다.(e) (f)에서는 다시 과정 (b)를 반복하여 새로 생성된 트랜잭션을 FPMSDTN에 삽입한다.

Algorithm : FPMDSTN
Input : To maintain the size of window, minimum support count
Output : Frequent Itemsets
Method :
(a) create the root of an FPMSDTN, and label it as "null"
(b) if transaction occurred, transaction is sorted in

ascending order. and after insert into FPMSDTN
(c) a pointer of the last event of transaction stored to the terminal node
(d) if count of transaction is equal to or greater than size of window then FPMDSTN(FPMDSTN, α)
(e) when a new transaction occurs, delete oldest transaction in the FPMSDTN
(f) return to (b)

procedure FPMDSTN_growth(FPMDSTN, α)
(1) if FPMDSTN contains a single path P then
(2) for each combination(denoted as β) of the nodes in the path p
(3) generate pattern β ∪ α with support_count = minimum support count of nodes in β;
(4) else for each ai in the header of FPMDSTN{
(5) generate pattern β = a_i ∪ α with support_count = $a_{i.support\_count}$;
(6) construct β′s conditional pattern base and then β′s conditional $FPMDSTN_\beta$;
(7) if $FPMDSTN_\beta \neq \varnothing$ then
(8) call FPMDSTN_growth($FPMDSTN_\beta$, β); }

4. 실험결과

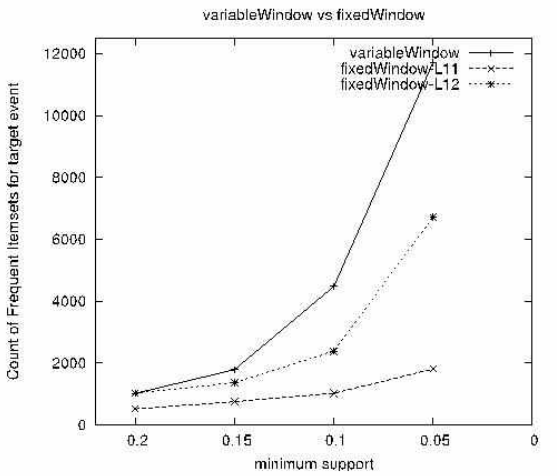
이 절에서는 제안하는 가변 윈도우와 고정 윈도우의 효율성을 비교하고 FPMDSTN 알고리즘의 성능을 분석한다. 실험은 Windows 7, 4GB RAM, Quad Core 2.40MHz 시스템에서 자바 언어로 작성하여 수행한다.

4.1 타겟 이벤트에 대한 가변 윈도우와 고정 윈도우의 비교

빈발 항목집합의 개수는 마이닝의 결과물로써 특정 알고리즘의 탐사 성능을 비교하기 위해서 이용된다. 실험을 위해 타겟 이벤트에 대한 빈발 항목집합의 개수를 이용하여 가변 윈도우와 고정 윈도우의 효율성을 비교한다. 실험 데이터는 pima.D38.N768.C2.num 데이터 셋을 대상으로 한다. pima.D38.N768.C2.num 은 38개의 아이템과 768개의 트랜잭션으로 구성되어 있고 트랜잭션의 평균 길이는 9이다.

타겟 이벤트를 '38'로 지정하고 가변 윈도우를 이용하면 평균 트랜잭션 길이가 11.17인 트랜잭션이 268개가 생성 된다(variableWindow). 고정 윈도우는 각각의 윈도우 크기를 11 (fixedWindow-L11), 12 (fixedWindow-L12) 로 하여 생성하며 각각 364, 254 개의 트랜잭션을 가진다.

(그림 6)은 생성된 세 개의 데이터 셋에서 타겟 이벤트 '38'과 연관된 빈발 항목집합의 개수를 나타낸 것이다. 아래 그림과 같이 가변 윈도우는 고정 윈도우보다 '38'과 연관된 빈발 항목집합을 더 많이 탐사할 수 있다. 따라서 고정 윈도우를 통해서는 탐사할 수 없었던 타겟 이벤트에 대한 연관 정보를 가변 윈도우에서는 탐사할 수 있기 때문에 사용자의 의사 결정에 큰 도움을 줄 수 있다.

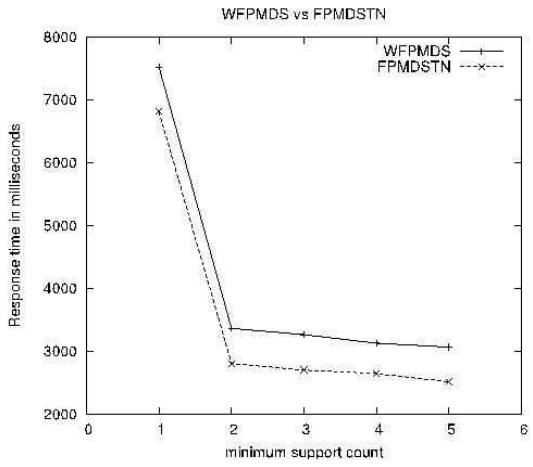


(그림 6) variableWindow vs fixedWindow

4.2 FPMDSTN 알고리즘과 WFPMDS 알고리즘의 수행 속도 비교

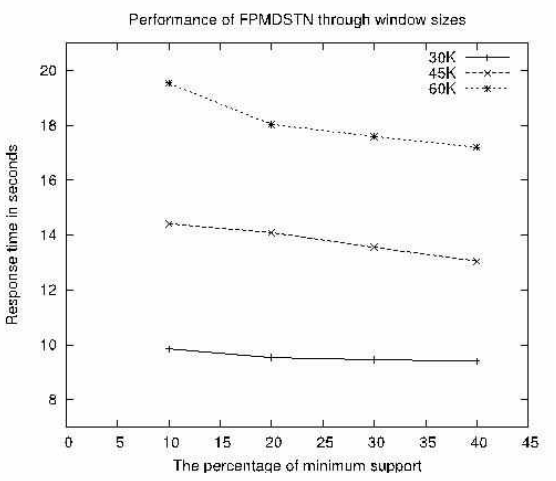
이 절에서는 FPMDSTN 알고리즘과 WFPMDS 알고리즘의 수행 속도를 비교하여 FPMDSTN 알고리즘의 실시간 환경에서 대한 적합성을 실험한다. 실험 데이터는 인공 데이터인 T10I4D100K dataset을 대상으로 실시한다. T10I4D100K dataset은 총 100,000개의 트랜잭션과 870개의 아이템으로 구성되어 있고 평균 트랜잭션의 길이는 10.1 개다.

(그림 7)은 T10I4D100K dataset 에 대해서 기존의 알고리즘(WFPMDS)와 제안하는 알고리즘(FPMDSTN)의 성능을 비교한 것이다. 윈도우의 크기는 8개의 트랜잭션으로 고정하고 WFPMDS 알고리즘에서 트랜잭션의 추가와 삭제 부분만을 반영하여 비교하였다. 그 결과 최소 지지도 횟수에 따라서 제안하는 FPMDSTN이 WFPMDS에 비해 약 10%-15% 정도 성능이 우수함을 보였다.



(그림 7) WFPMDS vs FPMDSTN

(그림 8)은 T10I4D100K dataset 에 대해서 WFPMDS 알고리즘의 윈도우 크기를 트랜잭션 30개, 45개, 60개로 실험한 것이다. 그 결과, 윈도우의 크기가 작을수록 더 빠른 응답 속도를 보였다. 따라서 시스템의 특성에 알맞게 윈도우의 크기를 조절하는 것이 중요하다.



(그림 8) 윈도우 크기에 따른 성능 비교

5. 토 의

데이터 스트림은 데이터의 발생이 무한하며 연속적인 특성을 지니고 있다. 데이터 사이에서 연관성을 찾기 위한 여러 알고리즘이 존재하지만 기존의 알고리즘들은 스트림 환경에는 적합하지 않다.

이 논문에서는 스트림 환경에 적합하게 이벤트를 추출할 수 있는 가변 윈도우와 COBJ 계산법, 그리고 실시간으로 연관 규칙을 탐사할 수 있는 FPMDSTN 알고리즘을 소개했다. 제안하는 가변 윈도우는 타겟 이벤트의 발생간격에 따라서 윈도우의 크기를 변화시킴으로써 데이터 스트림에서 의미 있는 데이터를 선별할 수 있도록 했다. 또한 존재 유무에 기반한 COBJ 계산법은 동일한 이벤트가 연속적으로 발생하는 스트림 환경에서 이벤트의 빈발 횟수와는 관계없이 이벤트를 선별함으로써 빈발하지 않은 이벤트에 대한 섣부른 가지치기를 방지했다. 그리고 스트림 환경에서 데이터의 불균형적인 분포로 인한 문제는 FPMDSTN에 슬라이딩 윈도우를 적용하여 해결하였다. 또한 FPMDSTN은 터미널 이벤트 테이블을 추가하여 트랜잭션의 추가와 삭제로 인한 비용을 감소시켰다.

제안하는 방법은 데이터가 실시간으로 무한하게 발생하는 금융 데이터 분석, 센서 네트워크 모니터링, 약물 부작용 감시, 의료 분야에 이용될 수 있다. 특히 IT와 의료가 결합하여 최근 각광 받고 있는 U-Health분야에 적용하기에 유용하다. 환자의 생체 데이터를 이용하여 환자에게 심각한 영향을 미칠 수 있는 이벤트를 타겟 이벤트로 지정하고 환자의 상태를 관찰하면 해당 타겟 이벤트에 대한 연관 규칙을 탐사할 수 있다. 탐사되는 규칙을 통해서 질병의 발생을 예측하고 미리 조치하여 환자의 건강을 관리할 수 있다.

6. 결 론

기존의 여러 마이닝 알고리즘은 대용량 데이터에서 연관 규칙, 순차 패턴, 항목 분류, 군집화 등 의사 결정에 필요한 유용한 정보를 탐사하였다. 그러나 기존의 알고리즘을 스트림 환경에 적용하기에는 한계가 있었고 스트림 환경에 적합한 새로운 알고리즘이 필요하다.

이 논문에서는 스트림 환경에서 실시간으로 연관 규칙을 탐사할 수 있는 알고리즘을 소개하였다. 제안하는 가변 윈도우와, COBJ 계산방법은 무한하고 연속적인 특성을 지닌 스트림 환경에서 의미 있는 이벤트를 추출하기에 유용했고, FPMDSTN 알고리즘은 빠른 연산을 요구하는 실시간 환경에서 효율적으로 연관 규칙을 탐사하기에 적합함을 보였다.

제안하는 방법은 환자의 질병 예측, 센서 네트워크 모니터링, 약물의 부작용 감시등에 유용하며 향후 특정 분야에 알맞은 규칙을 찾기 위한 탐사 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] J. Han, J. Pei, Y. Yin, and R. Mao, “Mining frequent patterns without candidate generation: a frequent pattern tree approach.”, Data Mining and Knowledge Discovery, Vol.8, pp.53-87, 2004.

[2] R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases”, proc. 12th ACM SIGMOD Int'l Conf. on Management of Data, pp.207-216, 1993.

[3] R. Agrawal, R. Srikant, “Fast algorithms for mining association rules.” , The VLDB Conference, Santiago, Chile, 1994. Sep.

[4] R. Agrawal, R. Srikant, “Mining Sequential Patterns.”, Proc. of 11th International Conference on Data Engineering, ICDE, pp.3-14, 1995.

[5] Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim, “SPRIT: Sequential Pattern Mining with Regular Expression Constraints.”, Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, pp.223-234, 1999.

[6] K.Y.Huang, C.H.Chang, “SMCA: A General Model for Mining Asynchronous Periodic Patterns in Temporal Databases.”, IEEE Transactions on Knowledge and Data Engineering, Vol.17, No.6, 2005, June.

[7] S.Laxman, P.S.Sastry, and K.Unnikrishnan, “Discovering frequent generalized episodes when events persist for different durations.”, IEEE Transactions on Knowledge and Data Engineering Vol.19, 2007, Sep.

[8] H. Mannila, H. Toivonen and A. I. Verkamo, “Discovering frequent episodes in sequences.”, proc. of International Conference on Knowledge Discovery and Data Mining(KDD-95), 1995.

[9] W.Pijls and R.Potharst, “Classification and target group

selection based upon frequent patterns.", Proc. of Twelfth Belgium-netherlands artificial intelligence Conference (BNAIC00), pp.125-132, 2000.

[10] H.Jin, J.Chen, H.He, C.Kelman, D.McAullay, and C.M.O'Keefe, "Signaling Potential Adverse Drug Reactions from Administrative Health Databases.", IEEE Transactions on Knowledge and Data Engineering, Vol.22, No.6, 2010, August.

[11] N. Srinivasa, Q. Jiang, and L. G. Barajas, "High-Impact Event Prediction by Temporal Data Mining Through Genetic Algorithms.", proc. of 4th IEEE International Conference on Natual Computation, 2008.

[12] C. F. Ahmed, S. K. Tanbeer, and B.S. Jeong, "Efficient Mining of Weighted Frequent Patterns Over Data Streams", 11th IEEE International Conference on High Performance Computing and Communications", 2009.

[13] G. Chen, X. Wu, and X. Zhu, "Mining Sequential Patterns Across Data Streams," Univ. of Vermont Computer Science Technical Report(CS-05-04), 2005(3).

[14] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.Hsu, "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach," IEEE Transactions on Knowledge and Data Engineering, Vol.16, No.11, 2004(11).

[15] C. K. S. Leung, B. Hao, "Mining of Frequent Itemsets from Streams of Uncertain Data", IEEE International Conference on Data Engineering, 2009.

[16] Pang-Ning Tan, Michael Steinbach, Vipin Kumar 저, 용환승, 나연묵, 박종수 역, "데이터 마이닝".

서 복 일

e-mail : seo.boxer@gmail.com

2010년 전남대학교 전자컴퓨터공학부 (공학사)

2010년~현 재 전남대학교 전자컴퓨터 공학부 석사과정

관심분야 : 스트림 데이터 마이닝, 데이터 베이스 시스템, 이동컴퓨팅

김 재 인

e-mail : sereno3@naver.com

2010년 전남대학교 전자컴퓨터공학과(석사)

2010년~현 재 전남대학교 전자컴퓨터 공학부 박사과정

관심분야 : 스트림 데이터 처리, 스트림 데이터 마이닝, USN응용

황 부 현

e-mail : bhhwang@chonnam.ac.kr

1978년 숭실대학교 전산학과(학사)

1980년 한국과학기술원 전산학과(공학석사)

1994년 한국과학기술원 전산학과(공학박사)

1980년~현 재 전남대학교 전자컴퓨터 공학부 교수

관심분야 : 스트림 데이터 마이닝, 분산 시스템, 분산 데이터베이스