데이터 스트림에서 개방 데이터 마이닝 기반의 빈발항목 탐색

장 중 혁[†] · 이 원 석[††]

요 약

기존의 데이터 마이닝 방법들은 기본적으로 지식 발견의 대상이 되는 데이터 집합이 마이닝 작업 시작 이전에 명확히 정의되는 것으로 가정하며 이러한 가정은 고정적으로 정의된 특정 데이터 집합에 내재된 정보 추출이 데이터 마이닝의 목적이 될 때 유효하다. 또한, 기존의 데이터 마이닝 방법들은 대용량의 데이터 집합에 대한 마이닝 결과를 얻는데 있어서 상당한 처리 시간을 요구한다. 따라서, 새로운 트랜잭션 데이터가 지속적으로 추가되는 데이터 스트림에서 추가된 트랜잭션의 정보들을 포함하는 최신의 마이닝 결과를 최대한 빠른 시간 안에 얻기를 기대하는 실시간 처리 환경에서는 기존의 데이터 마이닝 방법을 적용하는 것이 거의 불가능하다. 이러한 목적에 부합하기 위해서 본 논문에서는 새로운 데이터 마이닝 개념인 ***개방 데이터 마이닝***을 제안한다. 개방 데이터 마이닝에서는 새로운 트랜잭션이 발생함에 따라 이전에 발생한 트랜잭션들에 대한 마이닝 결과가 새롭게 갱신되며 따라서 확장된 전체 트랜잭션 집합에 대한 마이닝 결과를 빠르게 얻을 수 있다. 이러한 방법을 효과적으로 구현하기 위해서는 새롭게 출현한 항목에 대한 ***지연추가***와 이전 데이터 집합에 출현한 항목들 중에서 중요하지 않는 항목에 대한 ***전지작업***이 병행되어야 한다. 논문에서 제안하는 알고리즘은 알고리즘의 특성을 파악하기 위한 일련의 다양한 실험을 통해서 검증된다.

Finding Frequent Itemsets based on Open Data Mining in Data Streams

Joong Hyuk Chang[†] · Won Suk Lee[††]

ABSTRACT

The basic assumption of conventional data mining methodology is that the data set of a knowledge discovery process should be fixed and available before the process can proceed. Consequently, this assumption is valid only when the static knowledge embedded in a specific data set is the target of data mining. In addition, a conventional data mining method requires considerable computing time to produce the result of mining from a large data set. Due to these reasons, it is almost impossible to apply the mining method to a real-time analysis task in a data stream where a new transaction is continuously generated and the up-to-dated result of data mining including the newly generated transaction is needed as quickly as possible. In this paper, a new mining concept, ***open data mining*** in a data stream, is proposed for this purpose. In open data mining, whenever each transaction is newly generated, the updated mining result of whole transactions including the newly generated transactions is obtained instantly. In order to implement this mechanism efficiently, it is necessary to incorporate the ***delayed-insertion*** of newly identified information in recent transactions as well as the ***pruning*** of insignificant information in the mining result of past transactions. The proposed algorithm is analyzed through a series of experiments in order to identify the various characteristics of the proposed algorithm.

키워드 : 개방 데이터 마이닝(Open data mining), 데이터 스트림(Data stream), 빈발항목(Frequent itemset), 지연추가(Delayed-insertion), 항목 전지작업(Itemset pruning), 실시간 분석(Real-time analysis)

1. 서 론

기존의 데이터 마이닝 방법들은 기본적으로 지식 발견의 대상이 되는 데이터 집합이 마이닝 작업 시작 이전에 명확히 정의되는 것으로 가정하며 이러한 가정은 고정적인 데이터 집합에 내재된 정보들을 추출하는 것이 데이터 마이닝의 목적이 될 때 유효하다. 또한, 기존의 데이터 마이닝 방법들은 대용량의 데이터 집합에 대한 마이닝 결과를 얻는데 있어서 상당한 처리 시간을 요구한다. 기존의 데이터 마이닝 방법들은 동일한 응용 환경에서 발생되는 서로 다른 데이터 집합에 대해서 각기 다른 마이닝 결과를 구해낸다. 따라서, 기존의 데이터 마이닝 방법에서 가장 중요하게 고려되어야 할 사항 중의 하나는 응용 환경의 특성을 가장 잘 반영하는 적절한 데이터 집합을 정의하는 것이다. 일반적으로 응용 환경에 내재된 지식들은 시간이 흐름에 따라 변화되며 최근에 얻어진 의미있는 마이닝 결과 정보들도 조만간에 무의미한 정보로 변화될 수 있다. 최근에 새롭게

† 준 회 원 : 연세대학교 대학원 컴퓨터과학과
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
논문접수 : 2002년 11월 20일, 심사완료 : 2003년 1월 10일

발생된 트랜잭션에 나타나는 정보들을 포함하는 최신의 마이닝 결과를 얻기 위해서는 전체 데이터 집합에 대한 마이닝 작업이 처음부터 다시 수행되어야 한다. 본 논문에서는 분석 대상 데이터 집합이 사전에 명확히 정의 되어야 하는 이러한 데이터 마이닝 방법을 ***폐쇄 데이터 마이닝(closed data mining)***방법이라 정의한다.

데이터 마이닝은 데이터 분석에 있어서 효율성이 입증됨에 따라 각기 다른 요구 사항을 갖는 다양한 분야에서 널리 이용되고 있다. 그러나 이전의 방법은 마이닝 결과를 얻기 위해서 긴 처리 시간을 필요로 하므로 실시간 처리 환경에 적용하는데 한계가 있다. 온라인 분석 작업(Online analytical processing)[1], 고객 관계 관리(Customer relationship managements)[2] 및 비정상침입 탐지 시스템(Anomaly intrusion detection systems)[3] 등의 분야에서는 지속적으로 발생되는 트랜잭션들에 대한 자동적인 분석을 지원하고 분석 결과를 빠른 시간 내에 얻을 수 있기를 기대한다. 특히, 분석 결과가 즉시 활용되어야 하는 이러한 응용 시스템에서는 대량의 데이터에 대한 실시간 분석이 필수적이다. 비록 실시간 처리를 필요로 하는 응용 환경이 아니더라도 분석 과정이 정형화된 환경에서는 분석 결과를 빠른 시간에 얻을 수 있다는 것은 매우 가치 있는 특성이다.

마이닝 결과를 빠른 시간 내에 얻는데 있어서 폐쇄 데이터 방법은 몇 가지 한계를 갖는다. 폐쇄 데이터 마이닝에서 분석 대상이 되는 전체 데이터 집합은 전처리 과정을 거쳐 마이닝 작업이 시작되기 이전에 명확히 정의되어야만 한다. 또한, 이들 방법은 대량의 데이터 집합에 대해서 다중 탐색(scan)을 필요로 하므로 마이닝 작업 처리 시간이 매우 길다. 폐쇄 데이터 마이닝에 대한 대부분의 이전 연구들[4-7]은 한정적인 데이터 집합에 대한 분석에서 마이닝 수행 시간이나 메모리 사용량을 최소화 하는데 그 목적을 두고 있다. 변화된 응용 도메인에서 발생되는 새로운 요구 사항을 만족시키기 위해서는 비한정적인 데이터 집합에 대한 실시간 분석 능력이 지원되어야 하며, 새로운 트랜잭션이 발생되는 즉시 전체 데이터 집합에 대한 새로운 마이닝 결과를 실시간으로 활용할 수 있어야 한다. 즉, 어느 시점에서나 해당 시점까지의 모든 트랜잭션들을 포함하는 현재 데이터 집합에 대한 마이닝 결과를 얻을 수 있어야 한다. 본 논문에서는 이러한 요구를 만족하는 마이닝 방법을 ***개방 데이터 마이닝(open data mining)***이라 정의한다. 기존의 폐쇄 데이터 마이닝 방법에 있어서도 새로운 트랜잭션이 발생할 때마다 반복적으로 마이닝 작업을 수행함으로써 개방 데이터 마이닝을 구현할 수 있다. 하지만 폐쇄 데이터 마이닝 방법들은 매우 긴 처리 시간을 필요로 하고 지속적으로 확장되는 비한정적인 데이터 집합의 모든 트랜잭션의 정보들을 저장하는 것은 현실적으로 불가능하다.

근래 들어 데이터 스트림(data stream)에 대한 마이닝 방법들[8, 9]이 제안되었다. 데이터 스트림은 지속적으로 발생되는 트랜잭션들로 구성되는 무한 집합으로 정의된다. 따라서, 데이터 스트림의 모든 트랜잭션들을 별도로 저장하는 것은 불가능하므로 각 트랜잭션의 정보를 오직 한 번 읽고 마이닝 결과를 생성해야 한다. 이러한 이유로 이들 마이닝 방법들은 마이닝 결과에 다소의 오차를 포함한다. 이들 데이터 마이닝 방법들의 적용 대상 분야는 데이터 웨어하우스(data warehouse) 등에서와 같이 대량의 새로운 트랜잭션들을 한 번에 추가하는 경우와 보안 및 감시 시스템에서와 같이 지속적으로 증가되는 각 트랜잭션들이 개별적으로 증가되는 경우로 구분되며, 전자를 오프라인 데이터 스트림(off-line data stream)이라 하고 후자는 온라인 데이터 스트림(on-line data stream)이라 한다[8]. 오프라인 데이터 스트림에 대한 마이닝 알고리즘은 점진적으로 확장되는 데이터 집합에 대한 마이닝 알고리즘의 특별한 경우로 간주할 수 있다. 왜냐하면, 이들 두 알고리즘은 증가되는 데이터 양이 많을 때 효율적으로 수행될 수 있으며 최신의 정보들을 포함하는 마이닝 결과를 구하고자 하는 작업이 빈번히 발생하지 않는 경우에 효율적이다. 반면, 온라인 데이터 스트림 알고리즘은 최신정보를 포함하는 마이닝 결과를 보다 빈번히 제공해야 한다. 즉, 새로운 트랜잭션들이 분석 대상에 포함될 때 해당 트랜잭션을 포함하는 최신의 마이닝 결과를 구할 수 있어야 한다. 본 논문에서는 개방 데이터 마이닝을 위한 기본적인 구현 방법을 제시하고, 이러한 환경에서 적합한 효율적인 빈발항목 탐색 방법을 제안한다.

논문의 구성은 다음과 같다. 2절에서는 개방 데이터 마이닝을 설계하는데 필요한 관련 연구를 기술하고 3절에서는 개방 데이터 마이닝의 기본 개념 및 빈발항목 탐색을 위한 기본적인 방법을 제시하며, 4절에서는 개방 데이터 마이닝 환경에서 래티스 구조를 이용하는 빈발항목 탐색 방법을 제안한다. 5절에서는 일련의 다양한 실험을 통해서 제안된 방법들을 검증하며, 6절에서는 본 논문의 결론을 맺는다.

2. 관련 연구

일반적으로 빈발항목 탐색이라 함은 한정적인 데이터 집합에서 특정 지지도 임계값보다 큰 지지도를 갖는 모든 빈발항목을 파악하는 작업이다. 여기서 빈발항목이라 함은 해당 항목의 지지도가 사전에 정의된 지지도 임계값보다 큰 항목을 의미한다. 빈발항목 탐색을 위한 가장 대표적인 것은 *Apriori*[4] 알고리즘으로 최대 빈발항목의 차수가 n인 경우(즉, n개의 단위항목으로 구성되는 경우) 데이터 집합을 $n+1$번 탐색한다. *Partition*[6] 알고리즘은 데이터 집합을 최대 두 번 탐색하여 빈발항목 집합을 찾는 방법으로 분석 대상 데이터 집합을 가용 메모리 공간에 적합한 크기의 블록으로 분할한다. *DIC*[5] 알고리즘은 서로 다른 길이인 항

목들의 출현빈도수를 동시에 분석하며 잠재적으로 빈발인 항목들의 출현빈도수만 래티스(lattice)에서 관리된다. 데이터 집합은 몇 개의 블록으로 분할되고 전체 데이터 집합은 최대 두 번 탐색 된다. *Carma*[10] 알고리즘은 데이터 집합의 각 트랜잭션을 하나씩 차례로 탐색하며 다음과 같은 두 단계로 구성된다. 첫 번째 단계에서는 잠재적으로 빈발항목이 될 수 있고 자신의 모든 부분 항목이 이미 래티스에서 관리되고 있는 항목들이 후보 항목으로서 래티스에 추가한다. 두 번째 단계에서는 래티스에 존재하는 항목들에 대해서 해당 항목이 추가 되기 이전의 트랜잭션들을 다시 스캔함으로써 정확한 출현빈도수를 파악한다. 그러나 이들 알고리즘들은 분석 대상 데이터 집합이 마이닝 작업 시작 이전에 한정되어야 하고 데이터 집합을 한번 이상 탐색해야 하므로 개방 데이터 마이닝 방법으로는 적합하지 않다.

데이터 집합이 점진적으로 증가되는 환경에서는 점진적인 마이닝 알고리즘들[11, 12]이 효율적이다. 이 알고리즘들은 이전의 마이닝 결과들을 효율적으로 활용하여 새로 추가된 트랜잭션들의 분석 결과와 통합하여 전체 데이터 집합에 대한 마이닝 결과를 얻을 수 있도록 설계되었다. *BORDERS*[11] 알고리즘은 이전의 트랜잭션 집합에 대한 빈발항목 뿐만 아니라 가장자리-항목(border)들의 출현빈도수를 관리한다. 가장자리-항목이란 빈발항목은 아니지만 해당 항목 자신을 제외한 자신의 모든 부분항목들이 빈발인 항목으로 해당 항목을 포함하는 다른 항목의 출현빈도수를 유추하기 위해서 활용된다. 알고리즘에서는 먼저 새로 추가된 트랜잭션들에서의 빈발항목들을 파악하고 만약 이들이 증가되기 이전의 데이터 집합에서도 빈발항목이라면 해당 항목의 지지도를 갱신하고 새로운 빈발항목인 경우 해당 항목이 전체 데이터 집합에서 빈발인지를 판단하기 위해서 이전의 트랜잭션들을 탐색한다. *DEMON*[12] 알고리즘은 BORDERS 알고리즘에서 정의되는 가장자리-항목에 대해서 각 항목들이 출현한 트랜잭션들의 식별자 TID를 각 항목별로 유지함으로써 새로 추가된 트랜잭션들에서만 빈발인 항목의 출현빈도수를 이전의 트랜잭션들을 실제로 탐색하지 않고 정확히 구할 수 있으므로 BORDERS 알고리즘보다 더 효율적이다. 이러한 점진적 마이닝 알고리즘 방법들은 새로 추가된 트랜잭션들이 많을 때 효율적이나 이전의 트랜잭션들을 다시 탐색해야 하는 경우가 있으므로 개방 데이터 마이닝에는 적합하지 않다.

데이터 스트림에 대한 빈발항목 탐색 방법들[8, 9]에서는 데이터 스트림에 포함되는 이전의 모든 트랜잭션들의 정보를 유지하는 것이 불가능하므로 마이닝 결과로 구해진 빈발항목 집합이나 빈발항목들의 지지도에 다소의 오차를 포함한다. 이들 방법들은 지속적으로 발생되는 트랜잭션들에 출현한 항목들의 출현빈도수 정보를 파악하기 위한 접근 방법에 따라 확률적인 방법과 결정론적인 방법으로 구분된다[8]. 확률적인 방법은 데이터 스트림을 구성하는 정보들에 대한 발생 확률 등을 고려하여 전체 데이터 집합에서의 해당 정보의 출현 가능성을 추정하며 결정론적인 방법에서는 해당 데이터 스트림을 구성하는 각 트랜잭션들을 실제 분석하여 각 정보들의 출현빈도수를 파악한다. Lossy Counting 알고리즘[8]은 결정론적인 방법을 따르는 대표적인 방법으로 최소 지지도와 최대 허용 오차 조건이 주어졌을 때 데이터 스트림에서 발생한 빈발항목들의 집합을 찾는다. 스트림을 구성하는 각 트랜잭션에서 발생하는 빈발가능한 항목들의 출현빈도수와 이들 각각의 오차를 메모리에서 관리하여 새로 발생한 트랜잭션들은 메인 메모리에 유지되는 고정된 크기의 버퍼에 채워지고 동시에 처리된다. 버퍼에 채워진 트랜잭션들에 대해서는 먼저 단위항목들의 출현빈도수가 갱신되며, 버퍼의 트랜잭션들에서 가능한 모든 후보 항목을 생성하고 이의 발생빈도수를 갱신한다. 새로운 빈발가능 항목은 이전에 발생된 트랜잭션의 수를 고려하여 해당 항목에 포함될 수 있는 최대 오차를 추정하며 함께 관리한다.

3. 개방 데이터 마이닝

빈발항목 탐색을 위한 개방 데이터 마이닝 방법의 분석 대상 데이터 집합은 폐쇄 데이터 마이닝[4, 7]의 데이터 집합과 유사하나 다음과 같이 수정되어야 한다.

① $I=\{i_1, i_2, \cdots, i_n\}$는 현재까지의 단위항목(item)의 집합이며 단위항목은 응용 도메인에서 발생한 단위 정보를 의미한다.

② 2^I가 단위항목 집합 I의 멱집합을 나타낼 때, $e \in (2^I - \{\varnothing\})$을 만족하는 e를 항목(itemset)이라 한다. 항목의 길이 $|e|$는 항목 e를 구성하는 단위항목의 수를 의미하며 임의의 항목 e는 해당 항목의 길이에 따라 $|e|$-항목($|e|$-itemset)이라 정의할 수 있다. 한편, 일반적으로 항목 $\{a, b, c\}$는 간단히 abc로 나타낸다.

③ k번째 순서로 데이터 집합에 추가되는 트랜잭션(transaction)을 T_k라 나타내며 이는 단위항목 집합 I의 부분집합이다.

④ 새로운 트랜잭션 T_k가 추가되었을 때 현재의 데이터 집합 D_k는 현재까지 발생하여 추가된 모든 트랜잭션들로 구성된다. 즉, $D_k = \langle T_1, T_2, \cdots, T_k \rangle$.

〈표 1〉 개방 데이터 마이닝의 기본 개념

기 호	의 미
$\|D\|_k$	현재 데이터 집합 D_k에 포함된 트랜잭션의 총수
$C_k(e)$	항목 e의 현재 출현빈도수
$S_k(e)$	항목 e의 현재 지지도, $S_k(e) = C_k(e) / \|D\|_k$
S_{min}	최소 지지도

개방 데이터 마이닝에서는 마이닝 수행 과정의 임의의 모든 시점에서 현재 데이터 집합에 대한 마이닝 결과를 얻을 수 있어야 한다. 따라서, 현재 k번째 트랜잭션이 추가되었을 때 항목의 현재 출현빈도수는 현재까지 추가된 k개의 트랜잭션들 중에서 해당 항목을 포함한 트랜잭션들의 수를 의미한다. 마찬가지로, 항목의 현재 지지도는 현재의 데이터 집합 D_k에 포함된 트랜잭션의 총수 대비 해당 항목의 현재 출현빈도수로써 정의된다. 폐쇄 데이터 마이닝에서 사용되는 기본적인 개념들은 <표 1>에서와 같이 개방 데이터 마이닝에 적합하도록 재정의된다. 개방 데이터 마이닝의 결과는 현재 데이터 집합에 포함된 모든 트랜잭션에 나타난 정보들로부터 얻어지므로 <표 1>에서 나열된 모든 개념들은 현재 데이터 집합 D_k에 포함된 트랜잭션의 수에 의해 표현된다.

개방 데이터 마이닝에서는 데이터 집합이 무한정 확장되므로 모든 트랜잭션들을 저장하는 것은 불가능하다. 따라서 트랜잭션이 현재 데이터 집합에 추가되었을 때 해당 트랜잭션의 정보들을 점진적으로 수집해야 한다. 빈발항목 탐색 과정에서 각 트랜잭션에 출현하는 단위항목들의 조합으로 표현되는 서로 다른 항목을 효율적으로 표현하기 위해서 본 논문에서는 전위-트리(prefix-tree) 래티스 구조[5, 13]를 활용한다. 본 논문에서는 메모리에 존재하는 이와 같은 래티스 구조를 ***모니터링 래티스***(***monitoring lattice***)라 한다.

이러한 개방 데이터 마이닝의 특성을 만족하는 실제 구현 가능한 데이터 마이닝 방법으로 단순 축적에 의한 마이닝 방법(***Acc 방법***)을 고려할 수 있다. 이 방법은 각 트랜잭션에서 발생하는 항목들의 출현 정보를 단순히 축적하는 가장 단순한 방법으로 개방 데이터 마이닝의 실시간 처리 요구를 만족하기 위해서는 추가되는 전체 데이터 집합의 모든 트랜잭션 정보들을 메모리상의 모니터링 래티스에서 관리해야 한다. 만약 빈발항목들만 관리한다면, 현재 빈발항목이 아닌 어떤 항목도 이후 확장되는 데이터 집합에서 빈발항목이 될 수 없다. 결론적으로 마이닝을 위한 메모리 용량이 전체 트랜잭션들의 모든 정보를 유지할 수 있을 만큼 충분히 커야 한다. *Acc* 방법에서 현재 데이터 집합 D_k에 대한 항목 e의 출현빈도수 $C_k(e)$ 및 지지도 $S_k(e)$는 다음과 같이 정의된다.

$$① \ W_i(e) = \begin{cases} 1 & if\ e \in T_i \\ 0 & otherwise \end{cases}$$

$$② \ C_k(e) = \sum_{i=1}^{k} W_i(e)$$

$$③ \ S_k(e) = \frac{1}{k} \times \sum_{i=1}^{k} W_i(e)$$

여기서 $W_i(e)$는 트랜잭션 T_i에서 항목 e의 출현 여부를 나타낸다.

4. 데이터 스트림에서 빈발항목 탐색

개방 데이터 마이닝에서 각 항목의 출현 정보는 신중히 취급돼야 하며 새롭게 추가된 트랜잭션에서 새로운 항목이 출현했을 때, 해당 항목이 단지 현재 빈발항목이 아니라는 이유로 무시될 수는 없다. 만약 해당 항목이 무시된다면, 새로운 항목의 최초 출현시부터 빈발인 경우는 거의 없으므로 영원히 어떤 항목도 빈발항목이 될 수 없다. 이러한 특성에도 불구하고 *Acc* 방법과 같이 데이터 집합에 나타나는 모든 항목의 출현 정보를 관리하는 것은 거의 불가능하다. 만약 모든 정보를 전부 관리하고자 한다면 모니터링 래티스의 크기를 크게 증가시킴으로써 메모리 사용량을 크게 증가시킬 뿐만 아니라 모니터링 래티스에서 관리되는 항목들에 대한 탐색 시간을 증가시킨다. 본 절에서는 개방 데이터 마이닝에서 메모리 사용량을 감소시킬 수 있는 중요한 기법들을 제시하며 이를 바탕으로 데이터 스트림에서의 빈발항목 탐색 방법을 제안한다.

4.1 항목의 출현빈도수 추정

Acc 방법에서는 데이터 집합에 출현한 모든 항목들을 모니터링 래티스에서 관리하지만 이들 항목들 모두가 빈발항목 탐색을 위해서 중요한 항목은 아니다. 즉, 최소지지도보다 매우 작은 지지도를 갖는 항목은 가까운 미래에 빈발항목이 될 가능성이 적기 때문에 그 당시에는 모니터링 래티스에서 관리될 필요가 없다. 따라서, 모니터링 래티스에 대한 새로운 항목의 추가작업은 해당 항목이 가까운 미래에 빈발항목이 될 수 있을 때까지 지연될 수 있으며 정확한 마이닝 결과를 얻기 위해서 각 항목의 지지도는 해당 항목의 추가 시점에 정확히 추정되어야 한다. 즉, 잠재적으로 중요한 항목은 충분히 큰 지지도를 갖는 적절한 시점에 모니터링 래티스에 추가된다. 또한, 효율적인 항목 전지 방법도 메모리 사용량을 감소시킬 수 있는 또 하나의 방법이 될 수 있다. 이전 데이터 집합에서 매우 중요한 항목이었더라도 해당 항목의 현재 지지도가 사전에 정의된 최소지지도 보다 매우 작다면 해당 항목은 모니터링 래티스로부터 전지될 수 있다. 이러한 두 가지 방법들은 모니터링 래티스의 크기를 감소시키는 핵심 기법이다.

Carma[10] 알고리즘에서는 항목의 가능한 최대 출현빈도수가 해당 항목의 부분항목들의 가능한 최대 출현빈도수로부터 추정된다. 새로운 항목은 잠재적으로 빈발항목이면서 해당 항목의 모든 부분항목이 래티스에서 관리되고 있을 때 해당 항목도 래티스에 추가된다. 즉, 해당 항목이 잠재적으로 빈발항목인지를 판단하기 위해서 모든 부분항목들이 고려된다. Carma 알고리즘에서와 마찬가지로 새로운 항목의 출현빈도수는 모니터링 래티스에서 관리되고 있는 해당 항목의 부분항목으로부터 추정될 수 있다. 이를 위해서

[정의 1] 및 [정의 2]에서 정의된 개념들이 활용된다.

[정의 1] n-항목($n \geq 2$) e에 대해서 해당 항목의 부분항목 집합(set of subsets) $P(e)$, m-부분항목 집합(set of m-subsets) $P_m(e)$ 및 m-부분항목들의 출현빈도수 집합(set of counts for m-subsets) $P_m^C(e)$를 다음과 같이 정의한다.

① 부분항목 집합 $P(e)$는 항목 e를 구성하는 단위항목들 중에서 하나 이상의 단위항목으로 구성 가능한 모든 항목들의 집합이다.

$$P(e) = \{a \mid \forall a \ s.t. \ a \in 2^e - \{e\} \text{ and } a \neq \emptyset\}$$

② m-부분항목 집합 $P_m(e)$는 $P(e)$에 속하는 항목들 중에서 m개의 단위항목으로 구성되는 모든 항목들의 집합이다.

$$P_m(e) = \{a \mid \forall a \ s.t. \ a \in p(e) \text{ and } |a| = m\}$$

③ m-부분항목들의 출현빈도수 집합 $P_m^C(e)$는 $P_m(e)$에 속하는 모든 항목들의 출현빈도수 중에서 서로 다른 값들로 구성되는 집합이다.

$$P_m^C(e) = \{C(a) \mid \forall a \ s.t. \ a \in p_m(e)\}$$

여기서, $C(e)$는 데이터 집합에서 항목 e의 출현빈도수를 나타낸다.

□

[정의 2] 두 개의 항목 e_1 및 e_2에 대해서 ***병합항목(union-itemset)*** $e_1 \cup e_2$ 및 ***공통항목(common-itemset)*** $e_1 \cap e_2$를 다음과 같이 정의한다.

① 병합항목 $e_1 \cup e_2$는 e_1 또는 e_2에 속하는 모든 단위항목들로 구성된 항목이다.

② 공통항목 $e_1 \cap e_2$는 e_1과 e_2에 모두 속하는 모든 단위항목들로 구성된 항목이다.

특정 항목에 대해서 해당 항목의 부분항목은 데이터 집합에서 적어도 해당 항목이 출현한 만큼의 출현빈도수를 갖는다. 예를 들어 항목을 구성하는 모든 단위항목들이 항상 동시에 출현하는 경우 해당 항목은 해당 항목의 부분항목들과 동일한 출현빈도수를 갖는다. 따라서, 항목의 출현빈도수는 항목을 구성하는 단위항목들이 얼마나 빈번히 함께 출현하는가에 영향을 받게 된다. 이러한 분석 결과를 고려하여 항목의 출현빈도수의 범위를 결정할 수 있는 두 가지 분포 형태를 [정의 3]에서와 같이 정의한다.

[정의 3] 다수의 트랜잭션으로 구성되는 데이터 집합에서 출현한 임의의 두 항목에 대해서 이들 두 항목이 가능한 많은 트랜잭션에서 함께 출현한 경우를 ***최소배타분포(Least Exclusively Distributed : LED)***라 정의한다. 이와 반대로, 이들 두 항목이 최대한 배타적으로 출현한 경우를 ***최대배타분포(Most Exclusively Distributed : MED)***라 정의한다.

□

예를 들어 10개의 트랜잭션으로 구성된 데이터 집합에서 항목 e_1은 6개의 트랜잭션에서 출현하였으며 다른 항목 e_2는 7개의 트랜잭션에서 출현하였다고 가정하자. 만약 두 항목이 최소배타분포 상태라면 가능한 많은 트랜잭션에서 동시에 출현한 경우이며, 따라서 두 항목의 병합항목 $e_1 \cup e_2$의 최대 출현빈도수는 6이 된다. 반면에, 두 항목이 최대배타분포 상태라면 항목 e_2는 e_1이 출현하지 않은 4개의 트랜잭션에서 독립적으로 출현해야 한다. 즉, 두 항목은 3개의 트랜잭션에서 동시에 출현한다. 따라서, 두 항목의 병합항목의 출현빈도수는 3이상이며 6이하라고 간주할 수 있다.

[정리 1] 다수의 트랜잭션으로 구성되는 데이터 집합에 D에 대해서, $|D|$는 데이터 집합 D에 포함된 트랜잭션의 총수를 나타내며 $min(V)$ 및 $max(V)$는 숫자값들의 집합 V에 대해서 각각 최소값 및 최대값을 나타낸다. 두 항목 e_1 및 e_2에 대해서 병합항목의 최대 출현빈도수 $C^{max}(e_1 \cup e_2)$ 및 최소 출현빈도수 $C^{min}(e_1 \cup e_2)$는 다음과 같이 구할 수 있다.

① $C^{max}(e_1 \cup e_2) = min(C(e_1), C(e_2))$

② $C^{min}(e_1 \cup e_2)$

$$= \begin{cases} C(e_1) + C(e_2) - C(e_1 \cap e_2) & (\text{if } e_1 \cap e_2 \neq \emptyset) \\ \max(0, C(e_1) + C(e_2) - |D|) & (\text{if } e_1 \cap e_2 = \emptyset) \end{cases}$$

이때, 두 항목 e_1 및 e_2의 병합항목 $e_1 \cup e_2$는 적어도 $C^{min}(e_1 \cup e_2)$ 만큼의 트랜잭션에서 출현하고 최대 $C^{max}(e_1 \cup e_2)$이하의 트랜잭션에서 출현하는 것으로 간주될 수 있다.

(증명) 병합항목의 최대 출현빈도수 $C^{max}(e_1 \cup e_2)$는 두 항목이 최소배타분포(LED) 상태라고 가정하여 추정한다. 반면에, 병합항목의 최소 출현빈도수 $C^{min}(e_1 \cup e_2)$는 두 항목이 최대배타분포(MED) 상태라고 가정하여 추정한다. 다수의 트랜잭션들로 구성되는 데이터 집합 D에서 $TS(e)$는 항목 e를 포함하고 있는 트랜잭션들의 수를 나타낸다. 먼저, [정의 2]에 의해 식 (1)이 성립한다.

$$TS(e_1 \cup e_2) = TS(e_1) \cap TS(e_2) \quad (1)$$

만약 두 항목 e_1 및 e_2가 최소배타분포 상태라면 $TS(e_1) \subseteq TS(e_2)$ 또는 $TS(e_1) \supseteq TS(e_2)$ 관계가 성립한다. 그러므로 $C^{max}(e_1 \cup e_2)$는 두 항목의 출현빈도수 $C(e_2)$ 및 $C(e_2)$의 최소값 $min(C(e_1), C(e_2))$와 동일한 값을 갖게 된다. 한편,

[정의 2]에 의해 공통항목 $e_1 \cap e_2$에 대해서 식 (2)가 성립된다. 따라서 출현빈도수 관점에서 해석할 때 식 (3)이 성립되며 병합항목의 출현빈도수를 구하기 위한 식으로 전개하면 식 (4)가 성립된다.

$$TS(e_1 \cap e_2) = TS(e_1) \cup TS(e_2) \quad (2)$$

$$C(e_1 \cap e_2) = C(e_1) + C(e2) - C(e_1 \cup e_2) \quad (3)$$

$$C(e_1 \cup e_2) = C(e_1) + C(e_2) - C(e_1 \cap e_2) \quad (4)$$

만약 공통항목 $e_1 \cap e_2$이 존재하지 않는다면 병합항목의 출현빈도수는 데이터 집합에 포함되는 트랜잭션의 총수를 고려하여 구할 수 있다. $TS(e_1)$ 및 $TS(e_2)$가 데이터 집합 D의 부분집합이므로 식 (5)의 관계가 성립한다. 이러한 관계를 항목들의 출현빈도수 관점에서 해석할 때 식 (6) 및 식 (7)의 형태로 전개될 수 있다.

$$TS(e_1) \cup TS(e_2) \subseteq D \quad (5)$$

$$C(e_1) + C(e_2) - C(e_1 \cup e_2) \leq |D| \quad (6)$$

$$C(e_1) + C(e_2) - |D| \leq C(e_1 \cup e_2) \quad (7)$$

만약 두 항목의 출현빈도수 $C(e_1)$ 및 $C(e_2)$의 합이 $|D|$보다 작다면, 병합항목의 출현빈도수 $C(e_1 \cup e_2)$의 하한값은 0보다 작은 값을 가질 수 있다. 그러나 $C(e_1 \cup e_2)$는 항목의 출현빈도수이므로 하한값은 0이 된다. 따라서 두 항목이 최대배타분포일 때, 병합항목의 출현빈도 최소값 $C^{min}(e_1 \cup e_2)$는 다음과 같이 구해진다.

$$C^{min}(e_1 \cup e_2) = \begin{cases} C(e_1) + C(e_2) - C(e_1 \cap e_2) & (\text{if } e_1 \cap e_2 \neq \emptyset) \\ \max(0, C(e_1) + C(e_2) - |D|) & (\text{if } e_1 \cap e_2 = \emptyset) \end{cases} \quad (8)$$

□

데이터 집합 D에서 n-항목 e의 출현빈도수 추정값은 해당 항목의 부분항목들의 출현빈도수로부터 구할 수 있다. 모든 부분항목들이 최소배타분포 상태일 때, 부분항목들의 출현빈도수 중에서 최소값이 해당 항목의 출현빈도수 최대값 $C^{max}(e)$로 구해진다. 그러나 $(n-1)$-부분항목들은 해당 n-항목의 출현빈도수에 가장 유사한 정보를 제공할 수 있으므로 $C^{max}(e)$는 단지 $(n-1)$-부분항목들의 출현빈도수만을 고려함으로써 추정할 수 있다. 따라서, 해당 항목 e의 출현빈도수 최대값 $C^{max}(e)$는 [정리 1]에 의해 다음과 같이 구해진다.

$$C^{max}(e) = min(P^{C}_{n-1}(e))$$

마찬가지로 n-항목 e의 출현빈도수 최소값 $C^{min}(e)$도 해당 항목의 $(n-1)$-부분항목들의 출현빈도수로부터 추정할 수 있다. $P_{n-1}(e)$의 원소인 $(n-1)$-부분항목 a_i 및 a_j로 구성되는 각각의 서로 다른 조합 (a_i, a_j)에 대해서 두 부분항목이 최대배타분포 상태일 때, 해당 항목 e의 출현빈도수 최소값 $C^{min}(e)$는 [정리 1]에 의해 구해진다. 그러므로 모든 서로 다른 조합들에서 구해지는 출현빈도수 최소값 중에서 최대값이 부분항목들이 최대배타분포일 때 해당 항목의 출현빈도수로 간주된다. 즉, 해당 항목 e의 출현빈도수의 최소값 $C^{min}(e)$는 다음과 같이 구해진다.

$$C^{min}(e) = max(\{C(a_i \cup a_j) | \forall a_i, a_j \in P_{n-1}(e) \text{ and } i \neq j)$$

항목 e의 출현빈도수 최대값은 해당 항목의 출현빈도수를 추정해야 될 때 추정값으로 활용된다. 따라서, 출현빈도수 최대값은 데이터 집합에서 해당 항목이 출현할 수 있는 최대 트랜잭션수이므로 해당 항목의 출현빈도수는 추정오차를 포함할 수 있다. 한편, 해당 항목은 적어도 출현빈도수 최소값 $C^{min}(e)$ 만큼의 트랜잭션에서 출현한다고 보장할 수 있다. 따라서, $C^{max}(e)$와 $C^{min}(e)$의 차이가 해당 항목의 ***추정 오차*** $\varepsilon(e)$로 정의된다.

4.2 지연추가 및 항목 전지작업

개방 데이터 마이닝에서 최초 출현한 시점에서 항목의 출현빈도수는 1이다. 반면에, 데이터 집합이 지속적으로 확장되므로 트랜잭션의 총수는 일반적으로 매우 큰 값이다. 따라서, 최초 출현 시점에서 각 항목의 지지도는 최소 지지도에 비해 매우 작은 값이다. 결론적으로, 그 시점에서는 해당 항목을 중요하지 않은 항목으로 간주할 수 있다. 따라서, 해당 항목의 지지도가 마이닝 결과에 영향을 미칠 정도로 충분히 커질 때까지 해당 항목의 지지도를 관리하지 않아도 된다.

새로운 항목은 다음과 같은 두 가지 경우에만 모니터링 래티스에 추가된다. 첫 번째 경우는 새로 추가된 트랜잭션에 새로운 *1*-항목이 출현한 경우이다. 이 경우 해당 항목은 별도의 출현빈도수 추정 과정을 거치지 않고 즉시 모니터링 래티스에 추가된다. 따라서, 모든 *1*-항목의 출현빈도수는 추정값이 아니라 실제값이다. 두 번째 경우는 n-항목 (*n2*)의 지지도 추정값이 해당 항목의 출현빈도수를 관리해야 할 만큼 충분히 큰 경우이다. 새로운 트랜잭션에 출현한 항목 e가 모니터링 래티스에서 관리되고 있지 않을 때, 해당 항목의 출현빈도수는 4.1절에서 기술된 출현빈도수 최대값 $C^{max}(e)$ 값으로 추정된다. 해당 항목의 지지도 추정값은 현재 데이터 집합의 트랜잭션 총수 $|D|_k$ 대비 항목의 출현빈도수의 비율로 구할 수 있다. 만약 $P_{n-1}(e)$의 원소인 $(n-1)$-부분항목들 중에서 하나 이상의 항목이 현재 모니터링 래티스에서 관리되지 않고 있다면, 해당 항목의 $C^{max}(e)$ 값이 항상 0이므로 해당 항목의 출현빈도수는 별도의 추정 과정을 수행하지 않는다.

지지도 추정값이 사전에 정의된 임계값 이상이라면 해당 항목은 모니터링 래티스에 추가된다. 항목이 추가되었을 때 해당 항목의 출현빈도수는 출현빈도수 최대값 $C^{max}(e)$로 초기화 되며 더불어 추정오차 $\varepsilon(e)$도 동시에 관리된다. 이러한 작업 과정을 ***지연추가(delayed-insertion)***라 정의하며 사전에 정의된 추가를 위한 기준이 되는 임계값을 ***지연추가 임계값(threshold for delayed-insertion)*** S_{ins}라 정의한다. 항목의 출현빈도수 최대값 $C^{max}(e)$가 해당 항목의 실제 출현빈도수로 간주되므로 [속성 1]에서 정의된 바와 같이 항목들의 출현빈도수에 있어서 부정적 오차(negative error)는 발생하지 않는다. 또한, 항목의 지지도 오차는 새로운 트랜잭션이 추가로 발생함에 따라 [속성 2]에서와 같이 계속적으로 감소된다. 지연추가에 기반하여 모니터링 래티스에서 관리되는 항목의 수는 최소화될 수 있으며 이로 인해 메모리 사용량이 감소될 뿐만 아니라 마이닝 작업 시간을 단축시킬 수 있다.

[속성 1] 부정적 오차 무발생 속성 : n-항목($n \geq 2$) e에 대해서 출현빈도수 최대값 $C^{max}(e)$가 해당 항목의 실제 출현빈도수 $C(e)$로 간주되므로 항목의 출현빈도수에는 부정적 오차가 존재하지 않는다. 다시 말해서, 항목이 모니터링 래티스에 추가되었을 때 추정에 의해 부여되는 해당 항목의 출현빈도수는 해당 항목이 실제 출현한 트랜잭션의 수보다 많거나 같다.

□

[속성 2] 지지도 오차 감소 속성 : 트랜잭션 T_k에 출현한 새로운 항목 e가 모니터링 래티스에 추가되었을 때, 해당 항목의 추정 오차 (e)는 트랜잭션이 추가되더라도 변하지 않는 상수값이며 해당 항목의 지지도 오차는 $\frac{\varepsilon(e)}{|D|_k}$ 이다. n개의 트랜잭션이 추가 되었을 때, 트랜잭션의 총수는 $|D|_{k+n}$로 변화되며 $|D|_{k+n} > |D|_k$ 관계를 만족한다. 따라서 트랜잭션이 증가된 시점에서의 지지도 오차는 $\frac{\varepsilon(e)}{|D|_{k+n}}$ 로 변화되며 $\frac{\varepsilon(e)}{|D|_{k+n}} < \frac{\varepsilon(e)}{|D|_k}$ 관계를 만족한다. n 값이 지속적으로 증가되었을 때, $\frac{\varepsilon(e)}{|D|_{k+n}}$는 0으로 수렴된다. 즉, $\lim_{n\to\infty} \frac{\varepsilon(e)}{|D|_{k+n}} \approx 0$이며 따라서 무시될 수 있는 값이다.

□

폐쇄 데이터 마이닝에서 일련의 데이터 마이닝 과정에서 빈발항목이 될 수 없는 것으로 확인된 항목은 이후 분석 과정에서는 무시된다. 그러나 개방 데이터 마이닝에서는 현재 빈발항목이 아니더라도 이후 발생하는 트랜잭션에서 빈번히 출현하는 경우 빈발항목으로 변화될 수 있으므로 어떤 항목도 영원히 빈발항목이 아닌 것으로 간주할 수 없다. 하지만, 모니터링 래티스에서 관리되고 있는 항목이 이후 발생한 다수의 트랜잭션에서 출현하지 않는다면, 트랜잭션의 총수가 지속적으로 증가됨에 따라 해당 항목의 지지도는 감소될 것이며 사전에 정의된 임계값보다 작은 값으로 감소되었을 때, 해당 항목은 중요하지 않은 항목으로 간주할 수 있다. 따라서 기존의 래티스 기반의 데이터 마이닝 방법[5, 13]에서의 항목 전지작업과 마찬가지로 해당 항목은 모니터링 래티스로부터 제거될 수 있다. 그러나 만약 1-항목이 모니터링 래티스로부터 제거된다면, 이후에 해당 1-항목의 출현빈도수를 추정하는 것은 불가능하다. 따라서, 1-항목은 모니터링 래티스로부터 제거하지 않는다. 이러한 작업 과정을 ***항목 전지작업(itemset pruning)***이라 정의하며 항목 전지작업의 기준이 되는 지지도 임계값을 ***항목 전지작업 임계값(threshold for itemset pruning)*** S_{prn}이라 정의한다. 현재 시점에서 전지된 항목은 이후 새롭게 발생되는 트랜잭션에서 빈번히 발생되는 경우 지연추가 과정을 거쳐 모니터링 래티스에 추가될 수 있다. 동일한 항목에 대한 불필요한 전지 및 지연추가 과정이 반복되는 것을 방지하기 위해서 항목 전지작업 임계값은 지연추가 임계값 S_{ins} 이하의 값으로 정의되어야 한다. 더불어 모든 빈발항목을 보다 정확히 얻기 위해서는 두 임계값 S_{ins} 및 S_{prn}은 최소 지지도 S_{min}보다 작은 값으로 정의되어야 한다.

4.1절에서 언급된 바와 같이 모니터링 래티스에서 관리되는 항목의 출현빈도수는 출현빈도수 추정 과정에서 발생한 추정오차를 포함한다. 그러나, 추정오차를 고려하더라도 전지되는 항목들의 실제 출현빈도수는 [속성 1]에 의해 S_{prn}보다 작은 값을 갖는다. 한편, 모니터링 래티스에서 관리되는 각 항목은 각 항목마다 다른 정도의 추정오차를 갖는다. 왜냐하면 항목의 출현빈도수 추정시 발생하는 추정오차가 서로 다르기 때문이다. 각 항목들은 현재 시점에서 추정오차를 근거로 해당 항목의 지지도의 정확성을 평가할 수 있다.

4.3 *estAcc* 방법

본 절에서는 빈발항목 탐색을 위한 개방 데이터 마이닝 방법으로 *estAcc* 방법을 제안한다. 이 방법은 매개변수 갱신 단계, 출현빈도수 갱신 단계, 지연추가 단계 및 빈발항목 선택 단계와 같은 네 단계로 구성된다. 각 단계의 상세한 내용은 (그림 1)에 도시하였다. 마이닝 작업 과정에서 새로운 트랜잭션이 현재 데이터 집합에 추가될 때마다 빈발항목 탐색 단계를 제외한 다른 단계들이 순차적으로 수행된다. 매개변수 갱신 단계((그림 2)의 4번째 줄)에서는 현재의 데이터 집합의 트랜잭션의 총수 $|D|_k$가 갱신된다. 새로운 트랜잭션이 추가되었으므로 트랜잭션의 총수는 1증가된다. 출현빈도수 갱신 단계(5~9번째 줄)에서는 새로 추가된 트랜잭션에 출현한 항목들 중에서 모니터링 래티스에서 관리되고 있는 항목들의 출현빈도수가 갱신된다. 만약 항목의 갱신된 지지도가 S_{prn}보다 작다면 해당 항목은 모니터링

래티스로부터 제거된다. 해당 항목에 대한 출현빈도수 갱신 작업이 완료되면 새롭게 출현한 항목들 중에서 가까운 미래에 빈발 항목이 될 가능성이 있는 항목들을 찾기 위해서 지연추가 단계(10~19번째 줄)가 시작된다. 이 단계에서는 새로 추가된 트랜잭션에 출현한 항목들 중에서 모니터링 래티스에서 관리되고 있는 않은 항목들의 출현빈도수를 4.1절에서 기술된 과정을 거쳐 추정한다. 만약 항목의 출현빈도수가 S_{ins}보다 작은 경우 해당 항목을 부분항목으로 포함하는 모든 병합 항목의 출현빈도수는 별도의 추정 과정을 거치지 않아도 S_{ins}보다 작은 값임을 판단할 수 있다. 전위-트리 래티스의 특징으로 인해 보다 작은수의 단위항목들로 구성되는 항목이 래티스 상에서 먼저 탐색된다. 따라서 한 항목의 출현빈도수 추정값이 S_{ins}보다 작다면 해당 항목의 자식 노드로 표현된 항목들은 출현빈도수 추정 과정을 수행할 필요가 없다.

Input : A data stream D
Output : A complete set of frequent itemsets L_k

ML : A monitoring lattice

1 : $ML = \varnothing$;
2 : **for each** new *transaction* in D **do**
3 : **read** current transaction T_k ;
/* Parameter Updating Phase */
4 : $|D|_k = |D|_{k-1} + 1$;

/* Count Updating Phase */
5 : **for all** itemset e s.t. $e \in (2^{T_k} - \{\varnothing\})$ and $e \in ML$ **do**
6 : $C_k(e) = C_{k-1}(e) + 1$;
/* Pruning */
7 : **if** $S_k(e) < S_{prn}$ and $|e| > 1$ **then**
8 : Eliminate e and *it's child node* from L ;
9 : **end**

/* Delayed-insertion Phase */
10 : ItemFiltering(T_k) ;
11 : **for all** e s.t. $e \in (2^{T_k} - \{\varnothing\})$ and $e \notin ML$ **do**
12 : **if** $|e| = 1$ **then**
13 : Insert e into ML ; $C_k(e) = 1$; $(e) = 0$;
14 : **else**
15 : Estimate $C^{max}(e)$ and $C^{min}(e)$;
16 : **if** $\frac{C^{max}(e)}{|D|_k} \geq S_{ins}$ **then**
17 : Insert e into ML ; $C(e) = C^{max}(e)$;
$\varepsilon(e) = C^{max}(e) - C^{min}(e)$;
18 : **end**
19 : **end**

/* Frequent Itemset Selection Phase */
20 : $L_k = \varnothing$;
21 : **for all** $e \in ML$ **do**
22 : **if** $S_k(e) \geq S_{min}$ **then**
23 : $L_k = L_k + e$;
24 : **end**

(그림 1) *estAcc* 방법

빈발항목 선택 단계(20~23번째 줄)는 현재 데이터 집합에 대한 마이닝 결과를 구하고자 할 때에만 수행된다. 이 단계에서는 모니터링 래티스에 존재하는 모든 빈방항목을 전위-트리 래티스 구조에 기반한 기존의 데이터 마이닝 방법들에서와 동일한 방법으로 찾는다. 출현빈도수와 각 빈발항목들의 추정오차도 모니터링 래티스에서 관리되므로 빈발항목 탐색 시점에서 해당 항목의 지지도 오차를 구할 수 있다. 이 정보를 바탕으로 큰 지지도 오차를 갖는 빈발항목을 마이닝 결과 집합에서 제외함으로써 마이닝 결과의 정확도를 조절할 수 있다.

5. 실험 결과

본 절에서는 <표 2>에 나열된 데이터 집합을 대상으로 본 논문에서 제안한 개방 데이터 마이닝 방법의 성능을 검증하였다. <표 2>에서 $|T|$, $|I|$, $|D|$ 및 N은 각각 트랜잭션의 평균 길이(average transaction size), 잠재적 최대 빈발항목의 평균 길이(average maximal potentially frequent itemset size), 트랜잭션의 총수 및 데이터 집합을 구성하는 단위항목의 총수를 나타낸다. 각 데이터 집합은 [4]에서 설명된 방법으로 생성되었다. 데이터 집합의 특성을 나타내기 위해서 데이터 집합의 단위항목 총수 대비 트랜잭션의 평균길이의 비율을 각 데이터 집합의 ***트랜잭션-밀도***(transaction-density) $\rho_T(D)$라 정의한다.

$$\rho_T(D) = \frac{|T|}{N} \times 100(\%)$$

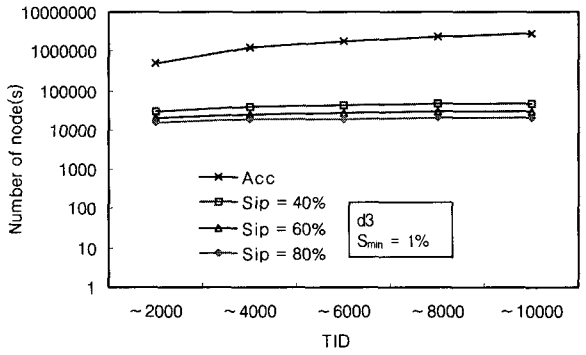
데이터 집합의 트랜잭션-밀도는 데이터 집합에서 단위항목들이 밀집되어 분포하는 정도를 평가하는데 활용된다. 데이터 집합 *d1*, *d2*, *d3* 및 *d4*의 트랜잭션-밀도는 각각 50%, 25%, 5% 및 2.5%이다.

〈표 2〉 실험 데이터 집합

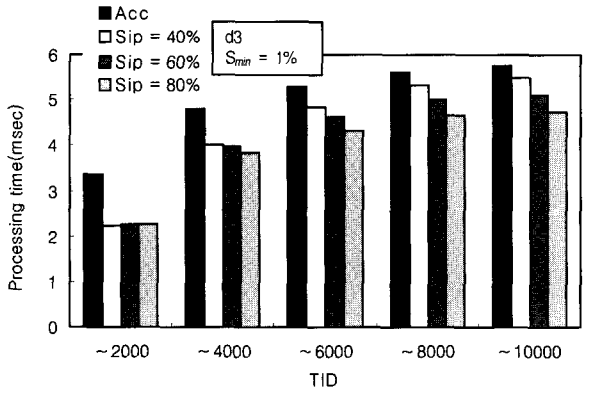
데이터 집합 D	$\|T\|$	$\|I\|$	$\|D\|$	N
d1	5	4	10K	10
d2	5	4	10K	20
d3	5	4	10K	100
d4	25	10	10K	1K

실험에서 각 데이터 집합의 트랜잭션들은 개방 데이터 마이닝 환경처럼 하나씩 차례대로 탐색된다. 개방 데이터 마이닝에서는 데이터 집합내의 모든 트랜잭션을 처리하기 위한 총 마이닝 수행 시간보다 각 트랜잭션을 빠른 시간내에 처리하는 것이 보다 더 중요한 의미를 갖는다. *Acc* 방법은 단위항목의 수가 매우 많은 대량의 데이터 집합에 대해서는 마이닝 작업을 수행할 수 없으므로 *estAcc* 방법의 정확성을 보다 세밀히 분석하기 위해서 비교 대상 결과 집합을 상대적으로 작은 규모의 데이터 집합인 *d1*, *d2* 및 *d3*에 대

해서 실험하였다. 모든 실험에서 두 임계값 S_{ins} 및 S_{prn}은 사전에 정의되는 최소 지지도 S_{min} 값에 대한 상대적인 값으로 정의하였다. 이들 두 임계값은 실제로 동일한 값으로 정의될 필요는 없지만 본 논문의 실험들에서는 이들을 서로 동일한 값으로 설정하였으며 간단히 하나의 기호를 사용하여 S_{ip}로 표현한다. 임계값 S_{ip}가 p%로 표현되었을 때, 두 임계값 S_{ins} 및 S_{prn}의 실제값은 $S_{min}\times(p/100)$으로 지정된다. 모든 실험들은 500MB의 램(RAM)을 가진 1.8GHz 팬티엄 컴퓨터와 리눅스 7.3 환경에서 실험되었으며 모든 프로그램은 C언어로 구현되었다.

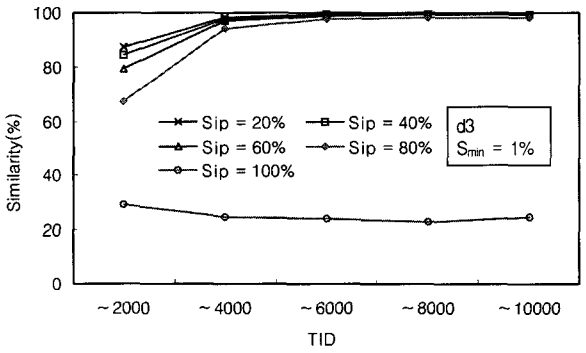


(그림 2) *d3*의 메모리 사용량



(그림 3) *d3*의 수행 시간

(그림 2)는 데이터 집합 *d3*의 각 트랜잭션들이 일련의 순서로 하나씩 탐색 되었을 때 *Acc* 및 *estAcc* 두 가지 방법의 메모리 사용량을 보여준다. 그림에서 추가되는 트랜잭션들은 순서에 따라 2000개씩의 트랜잭션으로 구성되는 5개의 구간으로 구분된다. 모니터링 래티스에서 생성되는 각 노드는 서로 동일한 크기이므로 각 구간의 메모리 사용량은 모니터링 래티스의 최대 노드수로 표현된다. *Acc* 방법은 단 하나의 트랜잭션에서 출현한 항목들도 모두 모니터링 래티스에서 관리하므로 많은 메모리 공간을 사용한다. 반면에 임계값 S_{ip}가 증가됨에 따라 *estAcc* 방법의 메모리 사용량은 감소된다. (그림 3)은 데이터 집합 *d3*의 각 트랜잭션이 (그림 2)에서와 동일한 순서로 탐색 되었을 때 각 구간에서 *estAcc* 방법에서 평균 수행 시간을 보여준다. 수행 시간은 하나의 트랜잭션이 추가되었을 때 처리하는데 소모되는 시간으로서 새로운 트랜잭션이 추가되어 완전한 빈발항목 집합을 얻는데 걸리는 시간으로 측정하였으며 최신의 마이닝 결과를 얻는데 필요한 평균 수행 시간이 6/1000000초 이하이다. 그러나 새로운 트랜잭션이 추가될 때마다 매번 빈발항목 선택 단계를 수행할 필요는 없으므로 만약 *estAcc* 방법에서 빈발항목 선택 단계를 제외한다면 평균 수행 시간은 30%정도 감소한다. 새롭게 추가된 트랜잭션의 처리 시간은 지연 추가 과정에서 출현빈도수를 추정해야 하는 새 항목들의 수뿐만 아니라 현재의 모니터링 래티스 크기에도 영향을 받는다. 특히, 모니터링 래티스를 마이닝 수행과정에서 빈번히 탐색하므로 수행 시간은 모니터링 래티스 크기에 주로 영향을 받는다. (그림 2)에서 보듯이 모니터링 래티스에서 관리되는 항목의 수는 S_{ip} 값에 반비례한다. 그러므로 S_{ip}가 증가함에 따라 새롭게 추가된 트랜잭션 처리에 필요한 수행 시간은 감소된다.



(그림 4) *d3*의 마이닝 결과 정확도

(그림 4)는 데이터 집합 *d3*의 트랜잭션들이 (그림 2)에서와 동일한 순서로 탐색 되고 임계값 S_{ip}를 다양하게 변화시켰을 때 *estAcc* 방법에서 얻어진 마이닝 결과를 *Acc* 방법에서 얻어지는 마이닝 결과와 비교한 마이닝 결과의 정확도를 보여준다. *Acc* 방법의 마이닝 결과는 *Apriori* 알고리즘 등과 같은 빈발항목 탐색을 위한 폐쇄 데이터 마이닝 방법의 마이닝 결과와 동일하다. 그러므로 *estAcc* 방법의 정확도는 해당 결과를 *Acc* 방법의 결과와 비교하여 측정한다. 두 결과 집합 R_1과 R_2의 유사도를 표현하기 위해서 [14]에서 정의된 다음과 같은 $similarity(R_1, R_2)$ 함수를 이용한다.

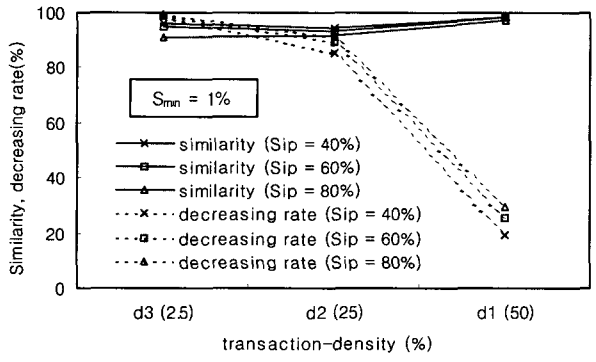
$$similarity(R_1, R_2) = \frac{|R_1 \cap R_2|}{|R_1 \cup R_2|}$$

두 방법에서 얻어진 마이닝 결과들의 유사도를 (그림 4)에서 보여준다. 유사도가 증가될수록 *estAcc* 방법에서 얻어진 마이닝 결과의 정확도가 높아진다. 결과 유사도는 S_{ip}의

값에 영향을 받는다. 해당 임계값이 낮게 설정되면 보다 많은 항목들이 모니터링 래티스에서 관리되고 이는 마이닝 결과의 정확도를 향상시킬 수 있다. 그림에서 첫 번째 구간의 유사도가 다른 구간들의 유사도에 비해 낮은 것을 알 수 있다. 첫 번째 구간에서는 데이터 집합에 포함되는 트랜잭션의 총 수가 상대적으로 작으므로 지연추가와 항목 전지작업이 보다 빈번히 수행된다. 임계값 S_{ip}가 증가됨에 따라 *estAcc* 방법은 메모리 사용량이 감소되고 마이닝 결과를 보다 빠르게 구할 수 있으나 마이닝 결과의 정확도는 낮아진다. 항목 전지작업 임계값 S_{prn}이 지연추가 임계값 S_{ins}보다 작은 값으로 지정되는 경우 마이닝 결과의 정확도 및 수행 시간 측면에서 약간의 성능 향상을 볼 수 있음을 확인하였으나 결과 그래프는 본 논문에서 생략하였다. 두 임계값의 차이가 커지면 서로 동일한 값을 갖는 경우에 비해 모니터링 래티스의 크기는 다소 증가되지만 전지된 항목이 다시 지연추가 되는 횟수가 감소됨에 따라 항목의 출현빈도수 추정 횟수가 감소됨으로써 수행 시간은 감소된다.

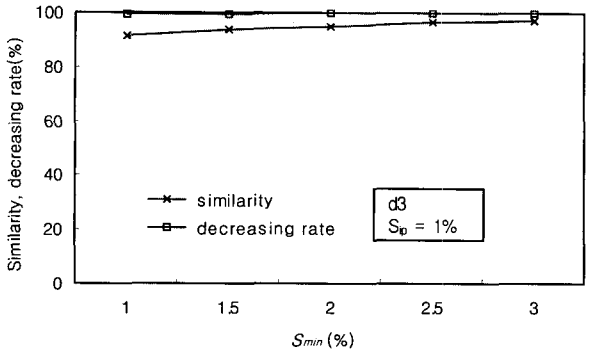
데이터 집합의 메모리 사용량은 해당 데이터 집합에 존재하는 트랜잭션들을 처리하는 과정에서 모니터링 래티스에서 관리되는 최대 노드수로서 표현되며 메모리 감소율(decreasing rate of memory usage)은 *estAcc* 방법에서 감소되는 노드수에 의해 정의된다. $num(X, D)$가 데이터 집합 D를 마이닝 방법 X로 마이닝을 수행했을 때 생성되는 최대 노드수라고 할 때 데이터 집합 D의 메모리 감소율은 다음과 같이 *decreasing rate*(D) 함수에 의해 정의된다.

$$decreasing\ rate(D) = \frac{num(Acc, D) - num(estAcc, D)}{num(Acc, D)}$$

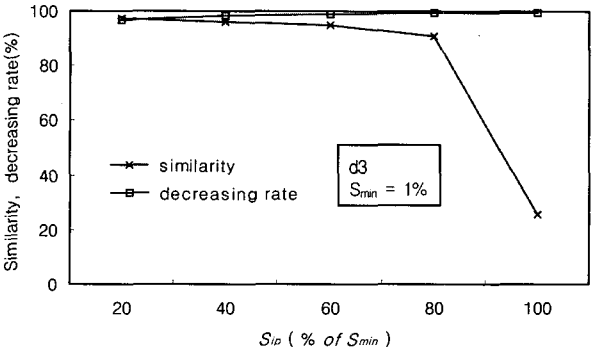


(그림 5) ρ_T 변화에 따른 마이닝 결과 변화

(그림 5)는 데이터 집합 *d1*, *d2* 및 *d3*에 대해서 *estAcc* 방법의 메모리 감소율 및 마이닝 결과 정확도를 보여준다. 마이닝 결과의 정확도는 데이터 집합의 트랜잭션-밀도 ρ_T에 상관없이 항상 높은 정확도를 나타낸다. 그러나 데이터 집합 *d1*의 메모리 감소율은 급격히 낮아진다. 일반적으로 데이터 집합의 트랜잭션-밀도가 높다면 해당 데이터 집합에 출현하는 항목들의 평균 지지도가 높다. 따라서 보다 많은 항목들이 모니터링 래티스에서 관리되어야 하므로 메모리 감소율은 낮아진다.



(그림 6) *d3*에서 S_{min} 변화에 따른 변화

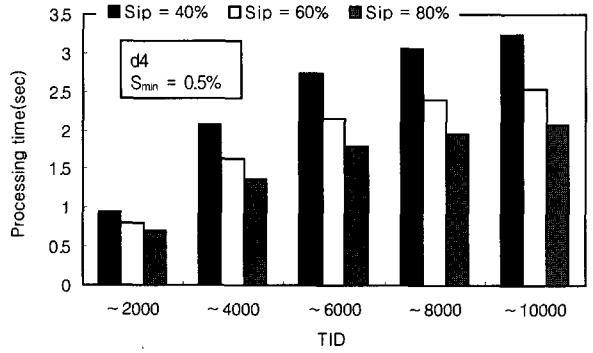


(그림 7) *d3*에서 S_{ip} 변화에 따른 변화

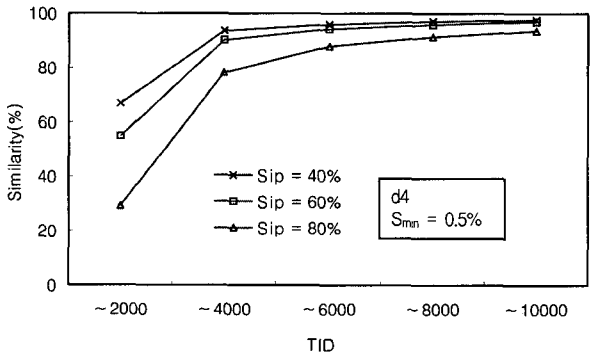
(그림 6)은 데이터 집합 *d3*에 대해서 최소 지지도 S_{min}이 변화되었을 때 *estAcc* 방법의 메모리 감소율 및 마이닝 결과 집합의 정화도 변화를 보여준다. 본 실험에서는 메모리 감소율은 거의 동일하다. 그러나 S_{min} 값이 감소됨에 따라 마이닝 결과 집합의 정확도는 다소 감소된다. S_{min}이 작아질수록 S_{min} 값과 S_{ip} 값의 차이가 작아진다. 이로 인해 보다 많은 빈발항목이 상대적으로 큰 지지도 오차를 갖게 되므로 *estAcc* 방법의 정확도는 감소된다. (그림 7)은 데이터 집합 *d3*에 대해서 임계값 S_{ip}가 변화되었을 때 *estAcc* 방법의 메모리 감소율 및 마이닝 결과 집합의 정화도 변화를 보여준다. S_{ip}의 값이 증가됨에 따라 모니터링 래티스에 추가되는 항목의 수가 감소되므로 메모리 감소율은 증가된다. 반면에, S_{ip} 값이 큰 경우 마이닝 결과 집합의 정확도가 급격히 감소한다. 이는 (그림 6)에서와 같은 이유이다. 큰 S_{ip} 값에 대해서는 큰 추정오차를 포함하는 빈발항목의 수가 증가된다.

규모가 큰 데이터 집합에 대해서 *estAcc* 방법의 성능을 분석하기 위해서 데이터 집합 *d4*를 실험하였다. (그림 8)은 데이터 집합 *d4*에 포함되는 트랜잭션들이 하나씩 차례로 탐색 되었을 때 하나의 트랜잭션을 처리 하는데 필요한 수행 시간을 각 구간별 평균 수행 시간으로 보여준다. 각 구

간에서 트랜잭션당 평균 수행 시간은 4초 이내이다. (그림 3)에서 제시된 실험에서와 같이 새롭게 추가된 트랜잭션을 처리 하는데 필요한 수행 시간은 S_{ip} 값에 주로 영향을 받는다. (그림 9)는 각 구간에서 *estAcc* 방법의 마이닝 결과 정확도를 보여준다. 데이터 집합 *d4*에 대해서는 *Acc* 방법을 적용할 수 없으므로 *estAcc* 방법의 정확도는 *Apriori*[4] 알고리즘으로 수행한 결과와 비교하여 측정되었다. 각 구간에 있어서 *Apriori* 알고리즘과 *estAcc* 방법에 의해 구해진 두 마이닝 결과 집합의 유사도는 매 500개의 트랜잭션들이 추가 되었을 때마다 유사도를 구하고 구간안에 포함되는 네 지점에서의 유사도를 평균하여 해당 구간의 평균 유사도를 구한다. 데이터 집합에 포함되는 트랜잭션의 수가 증가됨에 따라 *estAcc* 방법의 정확도는 크게 증가된다.



(그림 8) *d4*의 수행 시간



(그림 9) *d4*의 마이닝 결과 정확도

6. 결 론

응용 도메인에서 해당 도메인의 특성을 잘 표현할 수 있는 정확한 마이닝 결과가 중요하므로 폐쇄 데이터 마이닝 방법들은 데이터 집합이 응용 도메인의 지식들을 최상으로 표현할 수 있는 트랜잭션들로 구성될 때에 가장 유효하다. 더불어 폐쇄 데이터 마이닝에서 얻어진 결과가 응용 도메인의 특성을 잘 표현하고 있는지 여부를 판단하는 것은 매우 어려운 일이다. 따라서 특정한 데이터 집합에 대해 수행된 데이터 마이닝의 결과는 해당 데이터 집합이 생성된 응용 도메인에서 파악된 지식들의 일반적인 가능성만을 제시한다. 한편, 새로운 트랜잭션의 발생에 따른 마이닝 결과의 변화도 응용 도메인에 내재된 특성을 다양하게 분석할 수 있는 또 다른 중요한 지식이다. 이것은 응용 도메인에서 나타나는 정보들의 최근의 변화 동향을 분석할 수 있기 때문이다. 본 논문에서는 이를 지원하는 데이터 마이닝 기법들을 제시하였으며 빈발항목 탐색을 위한 구현 방법을 제시하였다. 본 논문에서 제안된 방법은 마이닝 결과의 정확도와 메모리나 CPU 등과 같은 컴퓨팅 자원에 대한 요구량 사이에서 유연하게 적응할 수 있도록 지원한다. 결과적으로 이 방법은 실시간 분석 작업들에 효율적으로 적용될 수 있다. 다양한 실험을 통해 본 논문에서 제안한 방법의 특성을 분석하였다. 본 논문에서 제안된 방법은 데이터 집합의 규모에 상관없이 마이닝 수행 시간을 단축 시킬 수 있도록 마이닝 결과 정확도를 조절할 수 있다. 즉, 보다 빠른 처리 시간을 필요로 하는 실시간 분석 작업에서는 마이닝 결과 정확도가 조금 낮아지는 것을 감수함으로써 마이닝 수행 시간을 감소 시킬 수 있다.

참 고 문 헌

[1] A. Berson and S. J. Smith, Data Warehousing, Data Mining, and OLAP : On-Line Analytical Processing, McGraw-Hill, New York, pp.247-266, 1997.

[2] S. Gallant, G. Piatetsky-Shapiro and M. Tan, Value-based data mining for CRM. In *tutorial notes of the 7th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, SanFrancisco, CA, Aug., 2001.

[3] S. J. Stolfo, A. L. Prodromidis, S. Tselepis, W. Lee, D. Fan and P. K. Chan, JAM : Java agents for meta-learning over distributed databases, In *Proc. of the KDD and AAAI Workshop on AI Methods in Fraud and Risk Management*, 1997.

[4] R. Agrawal and R. Srikant, Fast algorithms for mining association rules, In *Proc. of the 20th Int'l Conference on Very Large Databases*, Santiago, Chile, Sept., 1994.

[5] S. Brin, R. Motwani, J. D. Ullman and S. Tsur, Dynamic itemset counting and implication rules for market basket data. In *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, Tucson, AZ, pp.255-264, May, 1997.

[6] A. Savasers, E. Omiecinski and S. Navathe, An efficient algorithm for mining association rules in large databases, In *Proc. of the 21st Int'l Conference on Very Large Database*, Zurich, Switzerland, pp.432-444, Sept., 1995.

[7] S. Guha, R. Rastogi and K. Shim, CURE : A clustering algorithm for large databases, In *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, Seattle, WA, pp.73-84, June, 1998.

[8] G. S. Manku and R. Motwani, Approximate frequency counts over data streams, In *Proc. of the 28th Int'l Conference on Very Large Databases*, Hong Kong, China, Aug., 1994.

[9] M. Charikar, K. Chen and M. Farach-Colton, Finding Frequent Items In Data Streams, In *Proc. of the 29th Int'l Colloq. on Automata, Language and Programming*, 2002.

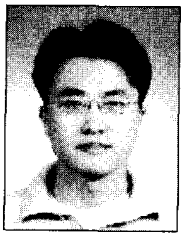
[10] C. Hidber, Online association rule mining, In *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, Philadelphia, PA, pp.145-156, May, 1999.

[11] Y. Aumann, R. Feldman, O. Lipshtat and H. Manilla, Borders : An efficient algorithm for association generation in dynamic databases, In Journal of Intelligent Information System, Vol.12, No.1, pp.61-73, 1999.

[12] V. Ganti, J. Gehrke, and R. Ramakrishnan, DEMON : Mining and monitoring evolving data, In *Proc. of the 16th Int'l Conference on Data Engineering*, San Diego, California, pp.439-448, Feb., 2000.

[13] R. C. Agarwal, C. C. Aggarwal and V. V. V. Prasad, Depth first generation of long patterns, In *Proc. of the 6th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, Boston, MA, pp.108-118, Sept., 2000.

[14] S. Cuha, R. Rastogi and K. Shim, ROCK : A robust clustering algorithm for categorical attributes, In *Proc. of the 15th Int'l Conference on Data Engineering*, Sydney, Australia, pp.512-521, May, 1999.



장 중 혁

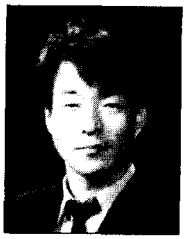
e-mail : jhchang@amadeus.yonsei.ac.kr

1996년 연세대학교 컴퓨터과학과(학사)

1998년 연세대학교 대학원 컴퓨터과학과 (석사)

2001년~현재 연세대학교 대학원 컴퓨터과학과 박사과정 재학

관심분야 : 데이터 스트림, 데이터마이닝, 멀티미디어 데이터 마이닝, 생물정보학



이 원 석

e-mail : leewo@amades.yonsei.ac.kr

1985년 미국 보스턴대학교 컴퓨터과학과 (학사)

1987년 미국 퍼듀대학교 컴퓨터공학과 (석사)

1990년 미국 퍼듀대학교 컴퓨터공학과 (박사)

1990년~1992년 삼성전자 선임연구원

1993년~1999년 연세대학교 컴퓨터과학과 조교수

1999년~현재 연세대학교 컴퓨터과학과 부교수

관심분야 : 분산 데이터베이스, 멀티미디어 데이터베이스, 객체지향 시스템, 데이터마이닝