▩ 연구논문

데이터마이닝에서 기존의 연관규칙을 갱신하는 분할 알고리즘

Partition Algorithm for Updating Discovered Association Rules in Data Mining

이 종 섭*
Rhee, Jong-Sub
황 종 원**
Hwang, Jong-Won
강 맹 규***
Kang, Maing-Kyu

ABSTRACT

This study suggests the partition algorithm for updating the discovered association rules in large database, because a database may allow frequent or occasional updates, and such update may not only invalidate some existing strong association rules, but also turn some weak rules into strong ones. the partition algorithm updates strong association rules efficiently in the whole update database reuseing the information of the old large itemsets. Partition algorithms that is suggested in this study scans an incremental database in view of the fact that it is difficult to find the new set of large itemset in the whole updated database after an incremental database is added to the original database. This method of generating large itemsets is different from that of FUP(Fast Update) and KDP(Kim Dong Pil)

1. 서 론

1. 1. 연구의 배경과 목적

최근 기업이나 백화점에서도 업무환경을 전산화함으로써 모든 데이터를 데이터베이스에 저장하여 관리한다. 특히 제조업에서는 바코드와 같은 입력장치의 발달로 자재, 생산 및 판매 데이터를 실시간으로 데이터베이스에 저장할 수 있게 됨으로써 하루에도 많은 데이터가 데이터베이스에 저장된다. 그러나 데이터를 저장하는 능력에 비해 데이터를 분석하는 능력이 떨어진다. 따라서, 대용량의 데이터베이스로부터 의사결정에 유용한 의미 있는 자료를 찾고, 분석하여 지식을 추론하는 새로운 기술 즉, 데이터마이닝[3,4,5,6]이 출현하게 되었다.

데이터베이스로부터 지식을 탐사할 때 찾고자하는 지식의 종류에 따라 데이터마이닝 기법도 달라지게 된다. 가장 활발히 연구가 진행되는 데이터마이닝 기법에는 트랜잭션 데이터가 주어졌을 때, 동시에 발생하는 항목들간의 관련성을 찾는 연관규칙(association rules), 발생하는 항목들간에 일정한 시간적인 함수관계를 갖는 연속패턴(sequential patterns), 동시에 발생하는 항목들을 가장 잘 나타낼 수 있는 특징을 발견하는 분류규칙(classification) 그리고 분류규칙과 관련되어 있으나 어떤 그룹도 사전에 정의되어 있지 않다는 점에서 분류규칙과 다른 군집화(clustering)등이 있다.

* 한양대학교 산업공학과 박사과정
** 이알정보기술연구소 부소장
*** 한양대학교 산업공학과 교수

본 논문은 데이터마이닝의 기법들에서 연관규칙을 갱신하는 연구를 하고자 한다. 연관규칙 갱신은 백화점등의 대형 유통센타에서 물류관리를 위해 이용되는 POS(Point Of Sales)시스템의 트랜잭션을 대상으로 트랜잭션내의 항목들을 분석하여 연관성이 있는 품목간의 규칙을 찾는다. 예를 들면, "커피와 프림을 구매한 고객의 90%는 설탕도 함께 구매한다" 와 같은 정보를 구하고, 이러한 정보를 판매전략 수립이나 상품 진열 계획, 시장 동향 예측등에 이용한다.

본 논문의 구성은 이후의 제 1.2절에서는 기존의 연관규칙 갱신 알고리즘에 대하여 고찰하고, 제 2장에서는 FUP(Fast Update)와 KDP(Kim Dong Pil)에 대해 서술한 후, 제안하는 갱신을 위한 분할 알고리즘을 서술한다. 제 3장에서는 실험결과를 서술하고, 제 4장의 결론으로 이 논문을 맺고자 한다.

1. 2 기존연구

1. 2. 1 연관규칙 갱신 알고리즘

(1) FUP 알고리즘

FUP 알고리즘[8]은 기존 데이터베이스에 만들어진 빈발 항목집합을 재사용하여 데이터가 추가되어 새로운 빈발 항목집합을 찾을 때 후보 항목집합을 현저히 줄여준다. FUP에서 제시한 과정은 다음과 같다. 최소지지도 s, 트랜잭션 개수 D인 데이터베이스 DB에서 빈발 항목집합 L의 원소 $X \in L$에 대해 지지도를 X.support 라 하자. 갱신이 이루어진 후, 새로운 트랜잭션 개수 d로 구성된 데이터베이스 db가 기존의 데이터베이스 DB에 추가된다. 최소지지도 s하에서 갱신된 전체 데이터베이스 $DB \cup db$의 항목집합 X는 $X.support \geq s \times (D+d)$이면 빈발 항목집합이다. 즉, 갱신된 전체 데이터베이스 $DB \cup db$에서 빈발항목집합 L^*를 찾아낸다.

FUP II 알고리즘[9]은 FUP 알고리즘에서 고려한 데이터 추가이외에 삭제도 이루어진다는 사실을 고려한 일반적인 논문이다.

Apriori[5]나 DHP[10]를 재수행하는 것보다 2~16배 빠른 것으로 밝혀졌다.

(2) KDP 알고리즘

Kim등[1]이 제안한 갱신 알고리즘 KDP는 선택적 검색(Selective Scan)[7]에서 제시한 후보 k항목집합을 생성하는 방법과 FUP에서 제시한 기존정보를 재사용하는 방법을 이용하였다. 후보 항목집합을 생성할 때 추가 데이터베이스 db를 검색한 후 추가 데이터베이스에서의 빈발 1항목집합 L_1을 찾고 해시테이블 H_2에 기반하여 L^*_2에 근사하는 C_2를 찾는다. C_2로부터 모든 단계의 C_k를 생성한 후 기존 데이터베이스 DB에서 빈발 항목집합을 찾는다. 따라서 KDP는 추가된 데이터베이스를 두 번 검색하고, 기존데이터베이스를 한번 검색하여 빈발 항목집합 L^*_k을 찾는다.

결론적으로, KDP는 FUP보다 수행시간을 향상시켰다.

2. 본 론

2. 1 제안하는 분할 알고리즘

제안하는 연관규칙 갱신을 위한 분할 알고리듬은 추가되는 데이터베이스의 양이 많지 않고, 추가되는 데이터베이스로 인하여 갱신된 전체 데이터베이스에서 새로운 빈발 항목집합의 출현은 어렵다는 전제하에 추가 데이터베이스를 먼저 검색하여 빈발 k항목집합을 구하고, 기존 데이터베이스의 빈발 k항목집합에 관한 정보를 재사용함으로써 기존 데이터베이스의 빈발 항목집합이 추가되는 데이터베이스의 빈발 항목집합과 일치 할 경우 기존 데이터베이스를 검색하지 않고도 기존의 연관규칙을 갱신할 수 있다.

이러한 현상은 추가되는 데이터베이스 db의 크기가 기존 데이터베이스 DB의 크기 보다 훨

씬 작고 비슷한 형태의 연관규칙을 가지고 있을 때 더욱 잘 나타나는 경향이 있다. 의사결정자가 수시로 연관규칙을 문의하는 상황에서 데이터가 추가될 때 마다 빠르게 연관규칙을 갱신할 필요가 있다. 따라서, 추가되는 데이터를 메모리에 두고 빈발 항목집합을 구하므로 전체적으로 추가되는 데이터베이스를 한 번 검색함으로써 입/출력 시간을 줄 일수 있다. 그러지 못한 경우에도 추가되는 데이터베이스를 최대 두 번 검색하여 기존의 연관규칙을 갱신할 수 있다.

2. 2 기호정의

본 논문에서 사용된 기호의 정의는 다음과 같다.

DB : 기존 데이터베이스
$db^{(n)}$: 분할 n지역에서 추가되는 데이터베이스
D : 기존 데이터베이스의 트랜잭션의 개수
$d^{(n)}$: 분할 n지역에서 추가되는 데이터베이스의 트랜잭션의 개수
K : 항목집합의 개수
C_k : 후보k항목집합
$C_k^{(n)}$: 분할k지역에서 후보k항목집합
$\bigcup C_k^{(n)}$: 추가 데이터베이스($\bigcup db^{(n)}$)에서 후보k항목집합
$L_k^{(n)}$: 분할 n지역에서 빈발k항목집합
$\bigcup L_k^{(n)}$: 추가 데이터베이스($\bigcup db^{(n)}$)에서 빈발k항목집합
s : 최소 지지도
$X.support$: 항목집합의 지지도
H_2 : 2항목집합의 해시테이블
L_k' : 기존의 빈발 k항목집합
L_k^* : 새로운 빈발 k항목집합
$W_k(=\bigcup C_k)$: 빈발 후보 k항목집합

2. 3 제안하는 분할 알고리즘

본 연구에서 제안하는 갱신 알고리즘은 분할개념[11]과 FUP에서 제시한 기존 정보를 재사용하는 방법을 이용함으로써 데이터베이스가 추가될 때 효율적인 갱신이 이루어지도록 연관규칙을 갱신한다.

KDP 알고리즘은 후보 항목집합을 생성할 때 추가 데이터베이스 db를 검색한 후 추가 데이터베이스에서의 빈발 1항목집합(L_1)을 찾고 해시테이블(H_2)에 기반하여 L^*_2에 근사하는 C_2를 찾는다. C_2로부터 모든 단계의 C_k를 생성한 후 기존 데이터베이스(DB)에서 빈발 k항목집합을 찾는다.

제안하는 분할 알고리즘의 주요 절차를 요약하면 아래와 같다. 기존 데이터베이스에서 빈발 k항목집합를 가지고 있다는 전제하에 추가되는 데이터베이스를 논리적(서로 겹치지 않게)으로 분할한다. 기존의 빈발 k항목집합에 대하여 추가되는 데이터베이스(db)에서 각 항목집합의 지지도를 갱신하는 동시에 논리적으로 분할된 각 지역별로 후보 k항목집합(C_k)을 구한다. 추가되는 데이터베이스(db)에서 후보 k항목집합을 구하고 동시에 기존의 빈발 k항목집합에서 전체 최소지지도($(D \bigcup d)*s$)를 만족하지 않은 빈발 k항목집합을 전지(Prune)한다.

빈발 후보 k항목집합(W_k)의 존재 여부에 따라 기존 데이터베이스 검색여부를 결정한다. 추가되는 데이터베이스(db)의 빈발 후보 k항목집합에 대하여 기존 데이터베이스(DB)를 검색하여 전체 최소지지도($(D \bigcup d)*s$)를 만족하지 않은 빈발 k항목집합을 전지(Prune)한다. 최소지지도를 만족하는 항목으로 새로운 빈발 k항목집합으로 결정된다.

지금까지 설명한 개념을 구현한 알고리즘의 주요 절차는 그림 2.1과 같다.

단계 0. [초기화]
- 기존 데이터베이스의 빈발 k항목집합(L'_k)이 주어진다
- 신규 2항목 해시테이블(H_2)를 생성하고 각 원소를 0으로 초기화한다

단계 1. [$L_{k(db)}$을 발견]
- 기존 데이터베이스의 빈발 k항목집합(L'_k)에 대하여 지지도를 갱신한다
- 추가되는 데이터베이스(db)의 빈발 k항목집합을 구한다
- 2항목 해시테이블(H_2)를 이용하여 빈발 2항목집합을 구한다

단계 1-1. [$L_{k(db^1)}$을 발견]
- 기존 데이터베이스의 빈발 k항목집합(L'_k)에 대하여 지지도를 갱신한다
- 추가되는 데이터베이스(db)의 분할 1지역에서 빈발 k항목집합을 구한다
- 2항목 해시테이블(H_2)를 이용하여 빈발 2항목집합($L_{2(db^1)}$)을 구한다

단계 1-2. [$L_{k(db^n)}$을 발견]
- 기존 데이터베이스의 빈발 k항목집합(L'_k)에 대하여 지지도를 갱신한다
- 추가되는 데이터베이스(db)의 분할 n지역에서 빈발 k항목집합을 구한다
- 2항목 해시테이블(H_2)를 이용하여 빈발 2항목집합($L_{2(db^n)}$)을 구한다

단계 2. [기존 빈발 k항목집합(L'_k)의 전지(Prune)]
- 기존의 빈발 k항목집합(L'_k)에 대하여 추가되는 데이터베이스(db)를 검색하여 빈발 k항목집합(L'_k)에 대한 지지도를 계산하고 지지도가 s*(D+d)를 만족하지 않은 빈발 k항목집합(L'_k)은 전지(Prune)한다

단계 3. [W_k을 발견]
- 추가되는 데이터베이스(db)의 빈발 k항목집합($L_{k(db)}$)에서 기존 데이터베이스의 빈발 k항목집합(L'_k)을 뺀 항목집합을 빈발 후보 k항목집합(W_k)라고 한다
- 만일 $W_k \neq \emptyset$이면 단계 5로 간다

단계 4. [L_k을 발견]
- 빈발 후보 k항목집합(W_k)에 대하여 기존 데이터베이스(DB)를 검색하여 W_k의 지지도를 갱신하고 갱신된 데이터베이스의 최소지지도(s*(D+d))를 만족하면 새로운 빈발 k항목집합(L_k^*)이다

단계 5. [끝냄]
- 끝낸다

그림 2.1 제안하는 알고리즘의 주요 절차

```
Given L'k , H2
For (n=1; db ≠ ∅; n++) {
        H2 ≠ ∅
        Scan db^n to find L1(db^n)
        Generate C2(db^n) based on L1(db^n), H2
        For (k=2; Lk-1(db^n) ≠ ∅; k++) {
            Generate Lk(db^n) from Lk-1(db^n) by Apriori
        }
        If X ∈ L'k then
                L'k = L'k ∪ X.support
        Else
                If X.support ≥ s × d^n then
                        Ck(db) = Ck(db) ∪ Lk(db^n)
        End If
}
For (k=1; L'k ≠ ∅;k++) {
    for_all X ∈ L'k
        if X.support ≥ s × (D+d) then
                Lk* = Lk* ∪ L'k
}
Generate Wk from Ck(db) for k ≥1
If (Wk ≠ ∅) {
Scan DB to count Supports of Itemsets in Wk
for_all X ∈ Lk
    if X.support ≥ s × (D+d) then
            Lk* = Lk* ∪ {X}
}
```

그림 2.2 제안하는 알고리즘의 코드

2. 4 수치예제(최소지지도 : 50%)

식별자	항목집합
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

그림 2.3 기존의 데이터베이스(DB)

식별자	항목집합
500	1 2 3 5
600	1 3 5

그림 2.4 추가되는 데이터베이스의 일부분(db1)

식별자	항목집합
700	2 3 5
800	1 3 5

그림 2.5 추가되는 데이터베이스의 나머지 일부분(db2)

항목집합	지지횟수
1	2
2	3
3	3
5	3

항목집합	지지횟수
1 3	2
2 3	2
2 5	3
3 5	2

항목집합	지지횟수
2 3 5	2

그림 2.6 기존 데이터베이스의 빈발 k 항목집합(L_k)

기존 데이터베이스(그림2-3)와 추가되는 데이터베이스(그림 2-4,5)가 주어지고 기존 데이터베이스에 대한 빈발 k항목집합이 그림 2.6과 같이 주어질 때, 갱신된 전체 데이터베이스에서 빈발 k항목집합은 다음과 같이 구할 수 있다. 수행하는 컴퓨터의 일정메모리까지 추가되는 데이터베이스의 일부분(db1)을 읽어서 그림 2.7과 같이 메모리에 TidList행태로 저장하고 메모리상에서 후보 k항목집합을 구한다.

db1 검색

항목집합	TidList
1	5 6
2	5
3	5 6
4	
5	5 6

항목집합	TidList
1 2	5
1 3	5 6
1 5	5 6
2 3	5
2 5	5
3 5	5 6

항목집합	TidList
1 2 3	5
1 2 5	5
1 3 5	5 6
2 3 5	5

그림 2.7 추가되는 데이터베이스의 후보 k항목집합(db1)

추가되는 데이터베이스의 나머지 부분(db2)에서도 같은 방법으로 후보 k항목집합을 구한다.

db2 검색

항목집합	TidList
1	8
2	7
3	7 8
4	
5	7 8

항목집합	TidList
1 2	
1 3	
1 5	
2 3	7
2 5	7
3 5	7 8

항목집합	TidList
1 2 3	
1 2 5	
1 3 5	
2 3 5	7

그림 2.8 추가되는 데이터베이스의 후보 k항목집합(db2)

분할된 추가 데이터베이스(db1 $\cup$ db2) 각각에 대해 후보 k항목집합을 구하고 이들을 후보 k항목집합별로 합집합을 만든다(그림 2.9). 기존 데이터베이스의 빈발 k항목집합을 뺀 추가 데이터베이스의 후보 k항목집합{{15},{135}}을 그림 2.9 에서 역상으로 표시하였다. 분할된 추가 데이터베이스에서 후보 2항목집합의 수를 줄이기 위하여 해시테이블(H_2)을 사용한다.

항목집합	TidList
1	5 6 8
2	5 7
3	5 6 7 8
4	
5	5 6 7 8

항목집합	TidList
1 2	5
1 3	5 6 8
1 5	5 6 8
2 3	5 7
2 5	5 7
3 5	5 6 7 8

항목집합	TidList
1 2 3	5
1 2 5	5
1 3 5	5 6 8
2 3 5	5 7

그림 2.9 분할된 추가 데이터베이스(db1 ∪ db2)에서 후보 k항목집합의 합집합

기존 빈발 k항목집합에 속하지 않은 빈발 후보 k항목집합의 원소인 {15}, {135}에서 기존 데이터베이스를 검색하여 최소지지도4(8 ×0.5)를 만족하는 {15}(지지횟수:4), {135}(지지횟수:4)가 빈발 2, 3항목집합에 추가로 포함된다. 최종적으로 갱신된 데이터베이스의 빈발 k항목집합은 그림 2.10과 같다

항목집합	지지횟수
1	5
2	5
3	7
5	7

항목집합	지지횟수
1 3	5
1 5	4
2 3	4
2 5	5
3 5	6

항목집합	지지횟수
1 3 5	4
2 3 5	4

그림 2.10 갱신된 데이터베이스의 빈발 k항목집합(L_k^*)

3. 실 험

제안하는 알고리즘의 성능평가를 위해 많은 실험을 수행하였다. Visual C++ 6.0을 사용하여 프로그램하였다. 실험은 CPU 166MHz, 메모리 32MB를 가진 컴퓨터에서 수행하였다.

3. 1. 실험 데이터의 생성

본 연구에서는 실험 데이터를 생성하기 위하여 사용한 방법은 Apriori나 DHP(Direct Hashing and Pruning)에서 제시된 것과 같다. 소매환경에서 소비자의 구매행위를 모방하는 것이다. 그림 3.1은 본 실험에서 사용된 파라미터들의 목록이다.

트랜잭션의 길이는 평균 |T|인 포아송분포에 의해서 결정된다. 트랜잭션은 트랜잭션의 길이를 초과하지 않는 범위내에서 잠재 최대 빈발 항목집합으로부터 반복적으로 할당된 항목들이다.

잠재 빈발 항목집합에서 항목집합의 길이는 평균 |I|인 포아송 분포에 따라 결정된다. 첫 번째

항목집합에 있는 항목들은 항목집합으로부터 임의로 선택되어진다. 이어지는 항목집합에 있는 항목들은 항목들의 일부가 평균이 상관수준(correlation level)과 같은 지수분포의 확률변수에 의해서 결정되는 이전 항목집합과 공통점을 가지는 집합으로 선택되어진다. 나머지 항목들은 임의로 선택되어진다. 상관관계의 수준은 0.5이다. 잠재 빈발 항목집합에 있는 각각 항목집합은 이 항목집합이 선택될 확률을 결정하는 관계된 가중치를 가진다. 가중치는 평균이 1인 지수분포에 따라 선택되어진다. 항목집합에 있는 모든 항목들이 트랜잭션에 할당되지는 않는다. Apriori 에서와 같이, 빈발 항목집합들의 모든 항목들이 항상 한꺼번에 선택되는 현상을 방지하기 위해 트랜잭션을 만드는 동안 몇 개의 항목들을 탈락시키는데 이 항목들은 0과 1사이의 균일하게 발생하는 확률변수가 탈락수준(corruption level, c)이하일 경우에 한하여 탈락한다. 항목집합에 대한 퇴화수준은 평균이 0.5이고 분산이 0.1인 정규분포에 의해서 결정된다.

각 트랜잭션은 <트랜잭션 식별자, 항목들의 수, 항목들>의 형식으로 하나의 파일 시스템에 저장된다. 실험에 사용되는 해시테이블(H_2)의 크기는 DHP에서 제시된 것과 같은 2^{19} 버켓으로 구성되게 하였다. 실험에서 사용된 파라미터의 값은 Apriori 나 DHP에서와 같이 D=m(천단위), d=n(천단위), N=1000, |L|=2000, |T|=10, |I|=4를 사용하였다. 예를 들어 데이터베이스 T10.I4.D100.d1은 각각 트랜잭션에 있는 항목들의 평균 개수가 10이고, 최대의 잠재적인 빈발 항목집합의 평균 항목들의 개수가 4이고, 기존 데이터베이스의 트랜잭션들의 개수가 100,000이고, 추가되는 데이터베이스의 트랜잭션들의 개수가 1000임을 의미한다.

D	기존의 데이터베이스 DB에서의 트랜잭션들의 개수
d	추가 데이터베이스 db에서의 트랜잭션들의 개수
|T|	전체 트랜잭션의 항목들의 평균개수
|I|	최대 잠재적인 빈발 항목집합들의 평균 개수
|L|	최대 잠재적인 빈발 항목집합들의 개수
N	항목들의 개수

그림 3.1 실험 데이터 생성에 사용된 파라미터 목록

3. 2. 실험결과

실험은 FUP와의 비교를 위해 Apriori나 DHP에서 수행한 것과 같은 방법을 따랐다.
상대시간은 FUP와 KDP에서의 수행시간을 100으로 보았을 때의 제안하는 알고리즘에서의 시간이며, {(제안하는 알고리즘의 수행시간(JSR)/FUP의 수행시간) ×100}과 {(제안하는 알고리즘의 수행시간(JSR)/KDP의 수행시간) ×100}으로 계산된다.

그림 3.2는 트랜잭션에 있는 항목들의 평균개수가 10이고, 최대의 잠재적인 빈발 항목집합의 평균항목들의 개수가 100,000이고, 추가되는 데이터베이스의 트랜잭션들의 개수가 1000인 데이터베이스 T10.I4.D100.d1에 대해서 최소지지도를 0.25%에서 1%까지 단계적으로 변화시켰을 때의 상대시간을 표시 한 것이다. 일정한 성능개선이 이루어짐을 알 수 있다.

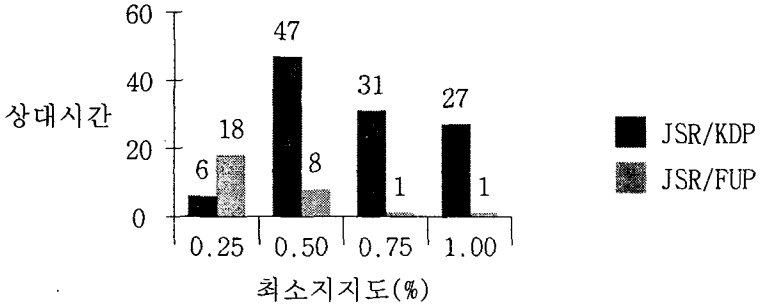

그림 3.2 최소지지도의 변화에 따른 수행시간 비교(기존 데이터베이스 : 100000. 추가 데이터베이스 : 1000)

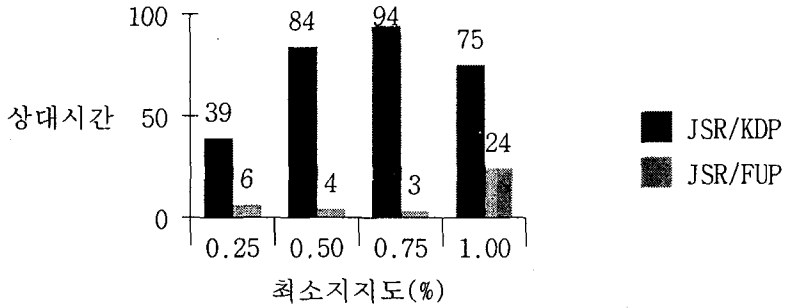

그림 3.3 최소지지도의 변화에 따른 수행시간 비교(기존 데이터베이스 : 1000000, 추가 데이터베이스 : 10000)

4. 결 론

데이터마이닝에서 기존 연관규칙을 효율적으로 관리하는 문제가 중요하다. 따라서 데이터베이스에 자료가 추가될 때 기존에 발견한 연관규칙을 재활용하여 갱신된 데이터베이스에서 새로운 연관규칙을 발견하는 방법에 관하여 연구하는 것은 바람직하다.

제안하는 분할 알고리즘은 해시테이블(H_2)과 추가되는 데이터베이스에 대하여 수평분할하여 Tidlist를 생성하고 Tidlist를 결합하여 빈발 k항목집합을 구하였다. 실험결과와 같이 기존 알고리즘인 FUP와 KDP에 비해 수행시간의 개선이 있었을 뿐만 아니라 데이버베이스 검색횟수에 있어서도 추가되는 데이터베이스를 두 번, 기존 데이터베이스 한번으로 빈발 k항목집합을 생성함으로써 진전이 있었다.

기존의 FUP 알고리즘은 각 단계별로 데이터베이스를 한 번씩 검색하여야 한다. 따라서 데이터베이스 검색횟수는 빈발 k항목집합의 수에 의존하여 k번 수행한다.

KDP는 추가되는 데이터베이스에서의 후보 2항목집합을 기반으로 모든 단계의 후보 항목집합을 한 번에 생성하고 추가되는 데이터베이스를 검색하여 최소지지도를 만족하는 후보 항목집합에 대해 기존의 데이터베이스를 검색하여 전체 데이터베이스에서 연관규칙을 갱신한다. 그러나 제안하는 알고리즘은 추가되는 데이터베이스에서 발생한 빈발 k항목집합이 기존 데이터베이스의 빈발항목집합과 일치하는 경우에는 단지 추가되는 데이터베이스를 한 번 검색하고도 빈발 k항목집합의 지지도를 갱신 할 수 있다. 그리고 추가되는 데이터베이스에서 새로운 빈발

k항목집합이 발생하여 기존 데이터베이스를 검색할 때에도 기존 빈발 k항목집합의 지지도는 첫 번째 검색에서 지지도가 갱신되고 추가 데이터베이스의 빈발 후보 k항목집합에서 기존 빈발 항목집합을 제거한 더 적은 항목집합에 대하여 기존 데이터베이스를 검색한다.

제안하는 분할 알고리즘은 트랜잭션에 있는 항목들의 평균 개수가 10이고, 최대의 잠재적인 빈발 항목집합의 평균 항목들의 개수가 4이고, 기존 데이터베이스의 트랜잭션들의 개수가 100,000이고, 추가되는 데이터베이스의 트랜잭션들의 개수가 1000인 데이터베이스 T10.I4.D100.d1에 대해서 최소지지도를 0.25%에서 1%까지 단계적으로 변화시키며 갱신작업을 수행할 때 기존 갱신 알고리즘인 FUP와 KDP보다 평균 92.91%, 평균 72.3%의 속도향상을 가져왔다.

또한 추가되는 데이터베이스의 크기를 증가시켰을 때, 전체 데이터베이스의 크기를 증가 시켰을 때에도 제안하는 갱신 알고리듬은 기존 갱신 알고리즘보다 속도향상을 가져왔다.

5. 참고문헌

1. 김동필, 지영근, 황종원, 강맹규, "데이터 마이닝에서 기존의 연관규칙을 갱신하는 효율적인 앨고리듬," *공업경영학회, 제 21권 제 45집*, pp. 121-133, 1998.
2. 이동명, 지영근, 황종원, 강맹규, "데이터 마이닝에서 기존의 연관규칙을 갱신하는 앨고리듬 개발," *공업경영학회, 제 20권 제 43집*, pp. 261-276, 1998.
3. Adriaans, P and D. Zantinge, Data Mining, Addison-Wesley, London, England,1996.
4 Agrawal, R., T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Database," *Proceedings of the 1993 ACM SIGMOD Conference,* May 1993.
5. Agrawal, R. and R. Srikant, "Fast Algorithms for Mining Association Rules," Proceedings of the 20th VLDB Conference, 1994.
6. Chen, M. S., J. Han, and P. S. Yu, "Data Mining: An Overview from Database Perspective," *IEEE Transaction on Knowledge and Data Engineering, Vol. 8, No.* 6, December 1996.
7. Chen, M. S., J. S. Park, and P. S. Yu, "Data Mining for Path Traversal Patterns in a Web Environment", Proceedings of the 16th *International Conference on Distributed Computing Systems, May, 1996.*
8. Cheung, D. W., J. Han, V. T. Ng, and C. Y. Wang, "Maintenance of Discovered Rules in Large Databases: An Incremental Updating Technique," *Proceedings of 12th International Conference on Data Engineering*, February 1996.
9. Cheung, D. W., S. D. Lee, and B. Kao, "A General Incremental Technique for Maintaining Discovered Association Rules," *Proceedings of the 5th International Conference on Database Systems for Advanced Applications,* April 1997.
10. Park, J. S., M. S. Chen, and P. S. Yu, "An Effective Hash-based Algorithm for Mining Association Rules," *Proceedings of ACM SIGMOD International Conference on Management of Data,* May 1995.
11. Savasere. A., E. Omiecinski, and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Database," *Proceedings of 21st VLDB Conference,* pp. 432-444, 1995.