2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

빈발 항목집합 추출을 위한 알고리즘

채 덕 진*, 황 부 현*
*전남대학교 전산학과
e-mail : djchai@sunny.chonnam.ac.kr

Algorithm for Extraction of Large itemsets

Duck-Jin Chai*, Bu-Hyun Hwang*
*Dept. of Computer Science, Chonnam National University

요 약

데이터 마이닝이란 대량의 실제 데이터로부터, 이전에 잘 알려지지는 않았지만, 잠재적으로 유용한 정보를 추출하는 작업이라 정의한다. 데이터 마이닝 기술 중에서 현재 가장 활발하게 연구되고 있는 것들 중의 하나가 연관 규칙 탐사이다. 연관 규칙이란 어떤 사건이 일어나면 다른 사건이 일어나는 관련성을 의미한다. 기존의 연관 규칙을 발견하기 위한 알고리즘들은 k-빈발 항목집합을 추출하기 위하여 k-후보 항목집합의 개수를 줄이거나 데이터베이스의 크기를 줄이는데 많은 연구가 이루어져 오고 있다. 본 논문에서는 상대적으로 많은 후보 항목집합의 데이터베이스 스캔을 통하여 추출되는 2-빈발 항목집합은 해쉬 기법을 사용하여 추출하고 k(k>2)-빈발 항목집합은 데이터베이스를 전처리하여 트랜잭션의 길이에 따라 두 개의 트랜잭션 집합으로 분리하고 분리된 데이터베이스에 다른 알고리즘을 사용하여 빈발 항목집합을 찾는 알고리즘을 제안한다. 그리고 성능 평가를 통하여 제안하는 방법의 성능 및 타당성을 보인다.

1. 서론

컴퓨터 시스템기술의 발달과 데이터베이스 이용의 증가로 컴퓨터에 저장되는 데이터의 양은 폭발적으로 증가하고 있다. 데이터 마이닝은 대량의 실제 데이터로부터, 이전에 잘 알려지지는 않았지만, 묵시적이고, 잠재적으로 유용한 정보를 추출하는 작업이라 정의한다[1]. 그리고 데이터에 숨겨진 패턴을 탐사하는 연구 중에서 연관 규칙 탐사에 대하여 많은 연구가 이루어지고 있다[2].

기존의 연관 규칙 탐사 알고리즘들은 먼저 후보 항목집합을 만들고 그 후보 항목집합들 중에서 빈발 항목집합들을 발견한다. 기존의 대부분 알고리즘들의 병목현상은 빈발 항목집합을 발견하기 위해서 많은 후보 항목집합들을 대상으로 대용량 데이터베이스를 스캔하는데 있다[3]. 즉, 후보 항목집합들이 많으면 그 만큼 많은 시간을 소비해야 한다. 특히, 2-후보 항목집합은 다른 k(k>2)-후보 항목집합보다 훨씬 더 많은 후보 항목집합들을 가지므로 알고리즘 수행시간의 대부분을 2-빈발 항목집합을 찾는데 소비된다[3].

본 논문에서는 2-빈발 항목집합을 빠르게 찾을 수 있도록 해쉬 기법을 효과적으로 사용하여 추출하고 나머지 k-빈발 항목집합 추출은 데이터베이스를 트랜잭션의 길이에 따라 두 개의 트랜잭션 집합으로 분리하고 분리된 데이터베이스에 각각 다른 알고리즘을 적용하여 유연하게 k-빈발 항목집합을 발견할 수 있는 FLE(Flexible Large itemsets Extraction) 알고리즘을 제안한다.

논문의 구성은 다음과 같다. 2장에서는 연관 규칙과 연관 규칙 탐사 알고리즘인 Apriori 알고리즘에 대해 설명하고 3장에서는 본 논문에서는 빈발 항목집합 발견 알고리즘을 설명한다. 4장에서는 다른 알고리즘과 실험을 통한 성능 비교를 기술하고 마지막으로 5장에서 결론을 내린다.

† 이 논문은 한국과학재단 1999년도 특정기초연구비 (1999-2-303-006-3) 지원에 의하여 연구되었음.

2. 관련 연구

2.1 연관 규칙의 정의

$I=\{i_1, i_2, \cdots, i_k\}$을 항목이라 부르는 리터럴(literal)들의 집합이라 하자. D를 트랜잭션들의 집합이라 부르고, 각 트랜잭션 T 는 $T \subseteq I$인 항목들의 집합이다. X 를 항목들의 집합이라 하자. ϕ이 아닌 항목집합 X, Y에 대해 $X \subset I$, $Y \subset I$ 에 대한 연관 규칙 $X \rightarrow Y$는 $X \cap Y = \phi$의 특성을 갖는다. X는 규칙의 가정, Y는 규칙의 결과라고 한다. 항목집합 I의 부분집합 X에 대해, $X \subseteq T$이면 T는 X를 만족한다고 정의한다. 최소 지지도(minimum support)를 만족하는 $X \subseteq I$를 빈발 항목집합이라 한다[5].

연관 규칙의 구체적인 예는 다음과 같다. "빵을 구매하는 사람의 40%는 우유도 구매하며, 전체 트랜잭션의 2%는 빵과 우유를 포함하고 있다." 여기서 40%는 이 규칙의 신뢰도(confidence)라고 불리고, 2%는 이 규칙의 지지도(support)라고 불린다.

이와 같이 생성된 연관 규칙은 고객 데이터베이스로부터 고객들의 구매품목들간의 관련성을 발견한다. 이는 개인 구매도 분석, 상품의 교차 매매(cross-marketing), 카탈로그 디자인, 염가 매출품(loss leader)분석, 상품 진열, 구매 성향에 따른 고객 분류 등등 다양하게 사용된다[1].

2.2 Apriori 알고리즘

Agrawal 등은 AIS[6]와 SETM[7]의 문제점을 해결하기 위하여 Apriori 알고리즘을 제안하였다[5]. 연관 규칙 탐사 문제는 빈발 항목집합을 찾는 문제와 추출된 빈발 항목집합에서 연관 규칙을 생성하는 두 개의 부 문제로 분리되어 연구되어 왔다. Apriori 알고리즘은 최초로 연관 규칙 탐사 문제를 두 개의 부문제로 분리하였으며, 효율적으로 빈발 항목집합을 찾기 위하여 먼저 후보 항목집합을 찾는 방법과 찾아낸 후보 항목집합을 최소화하는 방법을 제안하였다. Apriori 알고리즘에서는 이를 Apriori-gen으로 표현하였다.

주어진 데이터베이스로부터 빈발 항목집합을 찾기 위해 Apriori 알고리즘이 사용한 방법을 살펴본다. Apriori는 k-빈발 항목집합 L_k를 구하기 위해 (k-1)-빈발 항목집합 L_{k-1}로부터 k-후보 항목집합 C_k를 구하고 C_k의 지지도를 계산하여 최소지지도 이상을 만족하는 L_k를 구하는 과정을 반복한다.

최초의 반복시행에서 Apriori는 각 항목에 대한 지지도를 계산하기 위해 모든 트랜잭션들을 읽는다. 최소 지지도가 2라고 하면, 2이상의 지지도를 가진 후보의 1-항목집합로 구성된 1-빈발 항목집합의 집합인 L_1이 결정된다. 2-빈발 항목집합의 집합을 찾을 때 Apriori는 항목집합의 후보 집합인 C_2를 생성하기 위해 $L_1 * L_1$ 을 사용한다. 여기서 * 는 접속(concatenation)연산이다. C_2는 $\binom{|L1|}{2}$개의 2-항목집합으로 구성된다. $|L_1|$이 클 때, $\binom{|L1|}{2}$은 매우 커짐을 유의해야 한다. 다음으로 데이터베이스내의 트랜잭션들을 읽고 C_2의 각 후보 항목집합의 지지도를 세어 본다. 그러므로 2-빈발 항목집합의 집합인 L_2는 C_2의 2-후보 항목집합의 지지도를 기반으로 결정된다. 2-빈발 항목집합이 하나도 없으면 알고리즘은 종결되고 2-빈발 항목집합에서 후보 3-항목집합을 만들 수 있으면 과정은 더 이상의 후보 k-항목집합을 만들 수 없을 때까지 반복된다.

Apriori 알고리즘은 많은 후보 항목집합, 특히 2-후보 항목집합에 대하여 모든 트랜잭션을 스캔해야 하므로 빈발 항목집합을 추출하는데 많은 시간이 필요하다.

3. FLE 알고리즘

본 논문에서 제안하는 FLE 알고리즘은 해쉬 기법을 이용하여 2-빈발 항목집합을 추출한다. 그리고 3-빈발 항목집합 추출부터는 전체 데이터베이스를 트랜잭션의 길이에 따라 두 개의 트랜잭션 집합으로 분리하고 두 개의 트랜잭션 집합에 다른 알고리즘을 사용하여 빈발 항목집합을 추출한다.

3.1 해쉬 기법을 사용하여 2-빈발 항목집합 추출

기존의 연관 규칙 탐사 알고리즘에서 대부분의 수행시간은 2-후보 항목집합의 지지도를 계산하기 위하여 데이터베이스를 스캔하는데 있다. FLE 알고리즘에서는 수행시간을 줄이기 위해서 해쉬 기법을 이용하여 2-빈발 항목집합을 추출한다. 데이터베이스를 이루는 트랜잭션 T는 I의 부분집합으로 정의한다($T \subseteq I$). 이때 기존의 알고리즘에서 2-후보 항목집합을 생성하기 위해 사용했던 * 연산을 적용하여 $T * T$ 연산을 수행한다. 접속(concatenation)연산을 수행하여 2-부분집합을 생성하고 이를 기반으로 해쉬 테이블을 생성한다. 각 부분집합은 해쉬 함수를 통하여 하나의 버킷(bucket)를 생성한다. 생성된 버킷과 같은 주소의 버킷이 존재하면 그 버킷의 값을 1 증가시키고 같은 주소의 버킷이 없으면 새로운 주소의 버킷을 생성하고 1로 초기화한다.

TID	Item
100	A C D
200	B C E
300	A B C E
400	B E

TID	2-부분집합
100	AC, AD, CD
200	BC, BE, CE
300	AB, AC, AE, BC, BE, CE
400	BE

(그림 1) T * T 수행의 예

그림 1은 T * T 을 수행한 결과를 보여주는 예이다. 트랜잭션 식별자(TID)가 100인 경우의 트랜잭션은 A, C, D로 이루어져 있다. 이 트랜잭션에 대해서 접속 연산을 수행하면 AC, AD, CD가 생성된다.

TID	2-부분집합
100	AC, AD, CD
200	BC, BE, CE
300	AB, AC, AE, BC, BE, CE
400	BE

AB	AC	AD	AE	BC	BE	CD	CE	2-부분집합
1	2	1	1	2	3	1	2	Count
12	13	14	15	23	25	34	35	Hash address

h(x,y) = (index of x)*10 + (index of y)

(그림 2) 2-부분집합에 대한 해쉬 테이블의 예

그림 2는 2-부분집합에 대해서 해쉬 테이블을 생성하는 예를 보여준다. 해쉬 테이블이 생성되면 각 버킷에 저장되어있는 값과 최소 지지도를 비교하여 2-빈발 항목집합을 추출한다.

3.2 데이터베이스 분류

3.1절에서는 해쉬 기법을 사용하여 2-빈발 항목집합을 빠르게 추출할 수 있는 방법을 설명하였다.

2-빈발 항목집합이 추출되면 3-빈발 항목집합부터는 데이터베이스를 트랜잭션의 길이에 따라 두 개의 트랜잭션 집합으로 분류한다.

데이터베이스는 분류하고자하는 기준에 따라 다양하게 분류될 수 있다. 본 논문에서는 해쉬 기법을 사용하고 해쉬 함수를 통하여 생성된 주소가 유일해야 하므로 주소 공간을 고려하여 데이터베이스를 분류한다. 본 논문에서의 데이터베이스 분류는 전체 트랜잭션의 길이에 대한 평균 트랜잭션의 길이를 $|\hat{t}|$ 라고 했을 때, 길이가 긴 트랜잭션과 짧은 트랜잭션으로 분류하고 이때, 분류 기준은 $|\hat{t}|/2$로 정의한다. 즉, 평균 트랜잭션의 길이가 10이라고 했을 때, 트랜잭션의 길이가 5이하인 트랜잭션을 짧은 트랜잭션, 긴 트랜잭션은 6이상의 트랜잭션을 의미한다.

데이터베이스를 스캔하면서 각 트랜잭션의 길이와 평균 트랜잭션의 중간 값을 비교하여 길이가 긴 트랜잭션과 짧은 트랜잭션으로 나누어지고 트랜잭션이 분류될 때, 짧은 트랜잭션에 대해서는 동시에 해쉬 기법이 적용되어 분류된다.

3.2 빈발 항목집합 발견

제안하는 FLE 알고리즘은 빈발 항목집합을 추출하기 위하여 트랜잭션의 길이에 따라 서로 다른 알고리즘을 유연하게 적용한다. 트랜잭션의 길이가 짧은 경우에는 해쉬 기법을 사용하는 알고리즘을 사용하고 트랜잭션의 길이가 긴 경우에는 Apriori 알고리즘이나 DHP 알고리즘과 같은 기존의 알고리즘을 사용하여 수행한다(본 논문에서는 Apriori 알고리즘을 사용하여 수행). 그리고 두 알고리즘의 수행이 종결되면 하나의 데이터베이스를 두 개의 데이터베이스로 분류하였기 때문에 서로 다른 알고리즘에서 수행되어 생성된 후보 항목집합의 지지도를 더한 값이 최소 지지도를 만족하는지 확인하여 최종 k-빈발 항목집합을 추출하는 과정이 필요하다. 본 논문에서는 기존의 알고리즘 수행과정은 생략한다.

연관 규칙은 하나의 트랜잭션을 이루는 항목들의 집합 안에 존재한다. 예를 들어, A⇒B라는 연관 규칙이 존재할 때 A와 B는 하나의 트랜잭션 안에 존재하는 부분집합이다. 즉, 트랜잭션에서 만들어질 수 있는 모든 부분집합들을 생성하면 생성된 부분집합들 중 어느 하나는 연관 규칙이 될 수 있다는 말로 바꿀 수 있다. FLE 알고리즘은 기본적으로 이러한 성질을 이용한다

TID	3-부분집합	4-부분집합
100	ACD	
200	BCE	
300	ABC, ABE, ACE, BCE	ABCE

ABC	ABE	ACD	ACE	BCE	ABCE	k-부분집합
1	1	1	1	2	1	Count
123	125	134	135	235	1235	Hash address

h({x y z})=(index of x)*100 + (index of y)*10+(index of z)
H({w x y z}) =(index of w)*1000+(index of x)*100+(index of y)*10+(index of z)

(그림 3) k(k>2)-부분집합에 대한 해쉬 테이블의 예

그림 3은 그림 2와 같은 방법을 짧은 길이를 가진 트랜잭션에 적용하여 해쉬 테이블을 생성하는 예이다. 짧은 길이의 트랜잭션에서 해쉬 테이블을 생성하면 각 버킷에는 각 부분집합들의 지지도를 저장하고 있다.

트랜잭션의 길이가 짧은 데이터베이스에서 빈발 항목집합을 발견하는 알고리즘의 수행이 끝나면 트랜잭션의 길이가 긴 데이터베이스에서 빈발 항목집합을 발견하는 알고리즘을 수행한다. 길이가 긴 경우에 유일한 주소를 부여받는 해쉬 기법을 적용하기 위해서는 데이터베이스를 이루는 항목의 수, 트랜잭션의 길이 그리고 전체 트랜잭션의 개수에 따라 주소 공간이 변할 수 있다. 예를 들면, 항목의 수가 1000개 일 때 생성될 수 있는 2-부분집합에 대해서

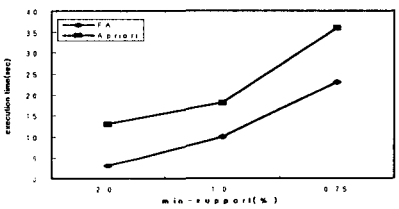
각각 유일한 주소를 부여하기 위해서는 $\binom{1000}{2}$개의 주소 공간이 필요하다. k-부분집합에 이러한 주소 공간 부여문제 때문에 길이가 긴 트랜잭션에 대해서는 기존의 알고리즘을 적용한다.

```
Forall transaction t ∈ D do begin
    Generate hash table for 2-subsets on T and T;
end
Forall b ∈ buckets do begin
    if (b.value > minsup) Generate 2-large itemset;
end
Forall transaction t ∈ D do begin /* |t̂| is length of average transaction */
    if (|t| > |t̂|) filter long transaction;
    else {
        filter short transaction;
        Generate k-hash table;
    }
end
Forall transaction t ∈ LD(long transactions) do begin
    Generate on using previous algorithm and using k-hashtable;
end
```

(그림 4) FLE 알고리즘

길이가 긴 트랜잭션에 대해서 기존의 알고리즘을 적용하여 구하고자 하는 3-빈발 항목집합부터는 전체 데이터베이스를 스캔하고 해쉬 기법을 사용하여 추출한 2-빈발 항목집합에서 생성된다. 다시 이러한 반복 과정을 거쳐 더 이상의 후보 항목집합이 생성되어질 수 없을 때, 분리된 두 개의 데이터베이스에 대해서 모든 알고리즘의 수행이 끝나면 두 개의 데이터베이스를 스캔한 결과의 지지도가 남게되고 이들을 더하여 분리되기 전의 전체 데이터베이스를 스캔한 결과를 얻게되면 전체 데이터베이스에 대한 3 이상의 k-빈발 항목집합을 추출할 수 있다. 그림 4는 FLE 알고리즘을 기술한 것이다.

4. 성능 비교



(그림 5) FLE와 Apriori 알고리즘의 성능 비교

그림 5는 2절과 3절에서 설명되어진 빈발 항목집합을 발견하기 위하여 Apriori 알고리즘과 FLE 알고리즘을 사용한 경우의 성능을 보여주고 있다. 주어진 데이터의 트랜잭션들의 개수가 10,000일 때 최소 지지도를 0.5%에서 2.0%로 변화시키면서 측정한 결과이다. 결과에서 보듯이 지지도에 따라 제안한 알고리즘의 성능이 더 우수하게 나타나는 것을 알 수 있다.

5. 결론

본 논문에서는 트랜잭션의 길이에 따라 다른 알고리즘을 유연하게 적용하는 연관 규칙 탐사문제를 다루었다. 빈발 항목집합을 발견하는 문제는 먼저 항목집합의 후보 집합을 구성하고 이 후보 집합 내에서 빈발 항목집합의 요구 조건을 만족시키는 항목집합을 규명함으로써 해결되었다. 본 논문은 데이터베이스를 이루는 각 트랜잭션의 길이가 다른 경우에 이를 분리하고 분리된 데이터베이스에 대하여 서로 다른 알고리즘을 적용하여 빈발 항목집합을 발견하는 알고리즘을 제안하였고, 실험을 통해 기존의 알고리즘을 적용했을 때 보다 성능이 더 좋다는 것을 입증하였다.

참고문헌

[1] 김정자, 이도헌 "데이타 마이닝 기술 및 연구동향", 한국 정보과학회지 제 16권 제 9호, pages 6-14, 1998.

[2] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases", In *Proceedings of ACM SIGMOD Conference on Management of Data, Washington* D.C., pp. 207-216, May 1993.

[3] J.S. Park, M.-S. Chen, and P.S. Yu, "An Effective Hash-Based Algorithm for Mining Association Rules", *Proceedings of ACM SIGMOD*, pages 175-186, May 1995.

[4] 박종수, "대용량 데이터베이스상의 효과적인 관련규칙 탐사를 위한 데이터 전지기법", 한국 정보과학회 데이터베이스 연구회지 제12권 제 4호, pages 59-75, 1996.

[5] R. Agrawal and R. Srikant, "Fast algorithms for mining Association Rules in Large Databases", In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, Sept., 1994.

[6] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules in large databases", In *Proceedings of ACM SIGMOD Conference on Management of Data,* Washington D.C., pages. 207-216, May 1993.

[7] M. Houtsma and A. Swami, "Set-Oriented mining for association rules", IBM Research Report, RJ 9567 (83573) October 22, 1993.