저장공간 축소와 실행시간 개선을 고려한 연관규칙 마이닝

한영우[0] 이수원
숭실대학교 컴퓨터학과 데이터마이닝 연구실
ywhan@valentine.ssu.ac.kr[0], swlee@computing.ssu.ac.kr

Association Rule Mining for Space Reduction and Performance Improvement

Young-Woo Han[0] Soo-Won Lee
Dept. of Computing, Soongsil University

요 약

연관규칙 탐사기법은 거래(사건) 속에 포함된 품목(항목)간의 연관관계를 발견하고자 할 때 사용하는 기법이며, 독특한 형태의 자료구조를 사용하는 다양한 연관규칙 알고리즘들이 제안되었다. 다양한 특성을 갖는 대용량의 데이터에 대해 효율적으로 연관규칙 탐사를 수행하기 위해서는 저장공간과 실행시간을 모두 고려해야 한다. 본 논문에서는 후보항목집합 발생과정 없이 압축빈발항목집합과 동적링크집합을 이용하여 저장공간 축소와 실행시간 개선을 동시에 고려한 연관규칙 알고리즘을 제안하며, 그 우수성을 증명하기 위해 연관규칙 탐사의 대표적인 자료 구조인 FP-struct, H-Struct와의 저장공간 비교 및 이들 저장구조를 사용하는 FP-growth, H-mine 알고리즘과의 실행시간을 비교한다.

1. 서 론

연관규칙 탐사기법은 거래(사건)속에 포함된 품목(항목)간의 연관관계를 찾아내는 기법으로써, 자율학습을 기반으로 하는 지식발견시스템의 핵심이다.

연관 규칙은 근거확률, 신뢰확률, 리프트 등의 확률을 이용하여 생성하며 장바구니 분석, 교차판매 전략, 카탈로그 디자인, 상품 배치, 웹에서의 사용자 접근 패턴 분석 등의 상업분야뿐만 아니라, 최근에는 생명공학 등의 과학연구분야에서도 적극 활용되고 있다.

연관규칙 탐사문제는 최소 지지도 이상의 트랜잭션 지지도를 갖는 항목집합들인 빈발항목집합(Frequent itemset)을 찾아내는 단계와 그것들로부터 연관규칙을 생성하는 단계로 구분되어지며[1], 연관규칙 알고리즘은 빈발항목집합 발견과정에서 후보빈발항목집합 발생의 유무에 따라 Apriori계 알고리즘과 비 Apriori계 알고리즘으로 구분될 수 있다.

연관관계를 관찰하고자 하는 품목의 수와 트랜잭션의 수가 증가할 경우, 분석을 위해 필요한 계산의 수는 기하 급수적으로 증가하며, 그에 필요한 저장구조의 크기 또한 급격히 증가한다. 그러므로 대용량 데이터에 대해 고속으로 연관규칙 탐사를 수행하기 위해서는 저장공간과 실행 시간을 동시에 고려한 알고리즘이 필요하다.

이를 위해 본 논문에서는 후보 빈발항목집합 발생과정 없이 압축빈발항목집합과 동적링크집합을 이용하여 저장공간 축소와 실행시간 개선을 동시에 고려한 연관 규칙 알고리즘을 제안한다.

2장에서는 본 연구에 대한 관련 연구를 소개하고, 3장에서는 제안된 FUSION 알고리즘 기법을 설명하며, 4장에서는 FP-struct, H-struct와의 저장공간 비교 및 이들 저장공간을 사용하는 FP-growth, H-mine 알고리즘과의 실행시간 비교를 통해 제안된 알고리즘의 우수성을 증명한다.

2. 관련 연구

2.1 연관 규칙

$I=\{I_1, I_2, \ldots, I_m\}$을 항목이라 부르는 리터럴(literal)들의 집합이라 할 때 트랜잭션 T는 $T \sqsubseteq I$ 인 집합이다. 항목집합을 A, A의 지지도를 Support(A), 최소지지도를 Smin라 하면, $A \sqsubseteq T$이고 $Support(A) \geq Smin$을 만족할 때 A는 빈발항목집합이 된다. 연관규칙은 빈발항목집합으로 표현된 트랜잭션에서 각 항목간의 연관성을 반영하는 규칙이고, R: $X \rightarrow Y$로 나타내며, 이때 규칙은 $X, Y \subset I$, $X \cap Y = \varnothing$의 특성을 갖는다. 연관 규칙의 타당성을 검증하기 위한 척도로서 지지도(support)와 신뢰도(confidence)가 사용되며 다음과 같이 표현된다.

$$Support(A,B) = P(A \cap B) \quad \text{식(1)}$$

$$Confidence(A,B) = P(B|A) = P(A \cap B)/P(A) \quad \text{식(2)}$$

그 외에 흥미도(interest), 확신도(conviction)등의 척도가 지지도와 신뢰도를 보안하기 위해 사용된다.

2.2 연관 규칙 알고리즘의 분류

빈발항목집합 발견과정의 관점에서 볼 때, 연관규칙 알고리즘은 후보빈발항목집합 발생의 유무에 따라 Apriori계 알고리즘과 비 Apriori계 알고리즘으로 구분될 수 있다. 후보 빈발항목집합을 발생하는 Apriori계 알고리즘은 긴 빈발패턴 또는 많은 수의 빈발패턴이 존재할 경우, 거대한 빈발항목집합의 발생과 과다한 트랜잭션 데이터

베이스 스캔으로 인한 오버헤드가 발생한다[2]. 예를 들어서, 만약 1-아이템으로 이루어진 10^4개의 빈발항목집합이 있다면 Apriori 알고리즘은 10^7개 이상의 2-아이템으로 이루어진 후보빈발항목 집합을 발생시킬 것이다. 또한 100-아이템으로 이루어진 빈발항목집합을 발견하기 위해서는 모두 $2^{100} \approx 10^{30}$개의 후보빈발항목집합을 생성해야 한다[2]. 이와 같은 병목 현상을 감소시키기 위해 AprioriTID, AprioriHybrid, DHP, Partition, Sampling, DIC과 같은 다수의 알고리즘들이 제안되었다. 이에 반해 비 Apriori계 알고리즘은 최대 두 번의 트랜잭션 데이터베이스 스캔과정을 통해 후보항목집합 발생과정 없이 빈발항목집합을 발견한다. 따라서 후보항목집합 발생 및 반복되는 트랜잭션 데이터베이스 스캔으로 인한 공간적, 시간적 오버헤드를 제거할 수 있는 장점이 있다.

2.3 FP-growth

FP-growth 알고리즘은 [2]에서 소개되었으며, 대표적인 비 Apriori계 알고리즘이다. 전체 트랜잭션 데이터베이스를 공통된 이전(prefix) 항목들에 대해 같은 트리 노드를 사용하는 FP-tree로 압축한 후, 재귀적으로 조건적 FP-struct 생성과정을 통해 후보항목집합 발생과정 없이 빈발항목집합을 발견한다. 표1은 원본 트랜잭션 데이터베이스와 최소 지지도가 3일때 지지도 순으로 정렬된 빈발항목집합들을 나타내며, 그림1은 표1로부터 생성된 헤더테이블과 FP-tree로 구성된 FP-struct를 나타낸다.

표1 전체트랜잭션과 빈발 아이템집합

TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, I, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

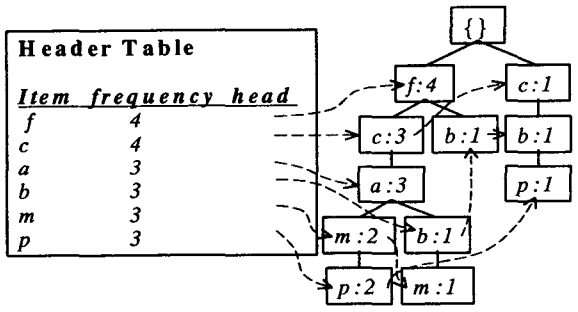

그림1 FP-struct

헤더테이블의 각 항목들에 대한 조건적 FP-tree를 FP-tree가 단일 패스로 구성될 때까지 분할 생성한 후, 항목과 각 트리 노드와의 모든 조합을 통해 빈발항목집합을 발견한다. 예를 들어서, 항목 m을 포함하는 모든 빈발항목집합을 발견하기 위해 항목 m의 조건적 패턴 집합인 {(f:2, c:2, a:2), (f:1, c:1, a:1, b:1)}을 통해 m의 조건적 FP-tree트리 {(f:3, c:3, a:3)}|m을 생성한다. 이 트리와 m의 조합을 통해 m을 포함하는 모든 빈발항목집합을 발견한다.

2.4 H-mine

H-mine 알고리즘은 [3]에서 소개되었으며, 전체 트랜잭션 데이터베이스를 H-struct라는 자료구조로 변환한 후, 재귀적 탐사과정을 통해 빈발항목집합을 발견한다. FP-growth가 반복적으로 조건적 FP-struct를 생성하는데 비해, H-mine은 임시 저장공간인 헤더테이블만을 반복적으로 생성한다. 빈발항목집합 발견시 Frequent Projections가 상대적으로 sparse하면 H-struct를 사용하고, dense하면 FP-struct를 생성하여 빈발항목집합을 발견한다. 그림2는 표1에 대해 생성된 H-struct의 예제이다.

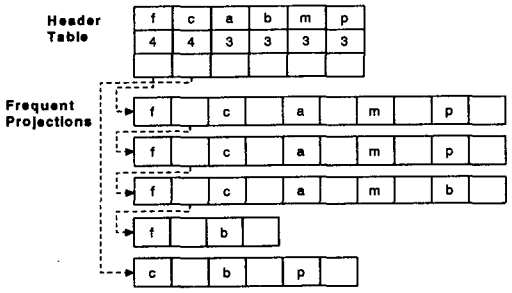

그림2 H-struct

빈발항목집합 발견시, 해당 항목에 대한 링크 집합이 완전하지 않을 경우, 이전 항목들의 큐를 다시 탐사하여 부족한 링크들을 찾는 backtracking이 필요하다. 예를 들어, 그림2에서 'c'를 포함하는 모든 빈발항목집합을 발견하기 위해서는 'c'에 대한 링크가 하나밖에 없기 때문에 이전 항목인 'f'의 큐를 탐사하여 나머지 링크 3개를 찾아 연결한 후 탐사를 수행한다.

3. FUSION 연관 규칙 알고리즘

3.1 기존 알고리즘의 문제점

FP-growth는 빈발항목집합 발견시 반복적으로 조건적 FP-struct를 생성함으로써 공간적 시간적 낭비를 초래한다. 또한 대용량의 sparse한 데이터인 경우 FP-tree의 압축율이 낮아지고, 최소 지지도가 매우 낮을 경우 원본 트랜잭션 데이터베이스보다 FP-struct가 더 커질 경우가 발생할 수 있다.

H-Mine은 입력 데이터가 dense/sparse한지에 대한 명확한 기준 제시가 어려울뿐더러, dense한 경우 FP-struct를 사용함으로 FP-growth가 가지고 있는 단점을 내재하게되고, sparse한 경우 H-struct를 사용함으로 압축 효과를 기대하기가 어렵다. 또한 H-struct를 사용할 경우 사용하지 않는 링크공간으로 인한 저장공간의 낭비가 발생하고, 빈발항목집합 발견과정시 해당 항목에 대한 링크가 부족할 경우 backtracking을 해야 하는 문제점이 있다.

3.2 FUSION

본 논문에서 제안한 FUSION 연관규칙 알고리즘은 저장공간 축소와 실행시간 개선을 동시에 고려하기 위해

제안되었다. 실행시간 개선을 위해 후보항목집합 발생과정 없이 2번의 트랜잭션 데이터베이스 스캔을 통해 빈발항목집합을 발견하며, 같은 항목들로 구성된 Frequent Projection에 대해 같은 저장공간을 사용함으로서 검색횟수의 감소와 저장공간 축소 효과를 얻을 수 있다. 또한 빈발항목집합 발견시 사용되는 동적링크집합을 임시 구조인 헤더테이블에 포함시켜 사용하지 않는 링크공간으로 인한 저장공간의 낭비를 제거하였다.

Frequent Projection을 스캔시, 두 번째 아이템에 대한 링크를 헤더테이블에 추가시킴으로써 backtracking을 생략할 수 있고, 탐사가 끝난 후 관련 링크집합을 헤더테이블로부터 제거함으로써 임시 저장공간의 확대를 방지하였다. 그림3은 FUSION-struct의 예제이다.

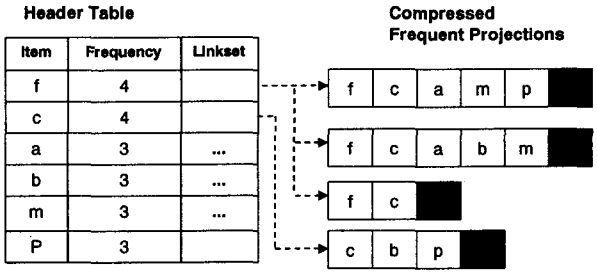

그림3 FUSION-struct

2개의 Frequent Projection {f, c, a, m, p}가 같은 저장 공간을 사용하고 있으며, 각 항목 사이에 불필요한 링크공간이 없음을 확인할 수 있다. 'f'에 대한 빈발항목집합 탐사를 하면서 두 번째 아이템인 'c'에 대한 링크를 헤더테이블에 추가하여 'f'에 대한 탐사가 끝난 후, bactracking 작업 없이 'c'에 대한 완전한 링크 정보를 얻을 수 있고, 'f'의 링크 집합을 헤더테이블로부터 제거하여 임시 저장공간의 확대를 방지한다.

4. 실험 및 분석

4.1 실험 데이터 및 실험 방법

실험 데이터는 백화점 거래데이터를 사용하였으며, 표1은 실험 데이터 특성을 나타낸다.

표1 실험 데이터 특성

데이터	아이템 수	각 트랜잭션 당 평균아이템 수	전체 트랜잭션 수	사이즈
백화점 데이터(D1)	6619개	7개	26580개	4830Kb

실험에 사용된 FP-growth, H-mine, FUSION 알고리즘은 모두 비 Apriori계 알고리즘들로써 JAVA로 구현하였고, 실험 데이터 D1에 대해 저장공간 크기와 실행속도를 실험하였다. 본 실험은 450MHz Pentium, 192Mem, 40G hard disk로 구성된 PC에서 이루어졌다.

4.2 실험 결과 및 분석

그림4는 지지도 임계치 변화에 대한 각 알고리즘의 저장공간 크기 결과를 나타내고 있다. 실험 결과 전 구간에 걸쳐 FUSION-struct가 가장 적은 저장공간을 사용하고, 지지도 임계치가 높아질수록 데이터가 상대적으로 dense해짐에 따라 지지도 임계치 약 0.4%를 기점으로 FP-struct가 H-struct보다 적은 양의 저장공간을 사용하고 있음을 확인 할 수 있다.

그림5는 지지도 임계치 변화에 대한 각 알고리즘의 실행속도 결과를 나타내고 있다. 실험 결과 전 구간에 걸쳐 FUSION이 가장 빠른 실행 속도를 보이며, 지지도 임계치가 낮아질수록 데이터가 상대적으로 sparse해짐에 따라, FP-growth와의 차이가 점점 커지는 것을 확인할 수 있다. H-mine은 FP-growth에 비해 적은 차이를 보이고 있지만, 그림4에서 FUSION-struct가 H-struct에 비해 뚜렷한 저장공간 축소효과가 있음을 확인 할 수 있다.

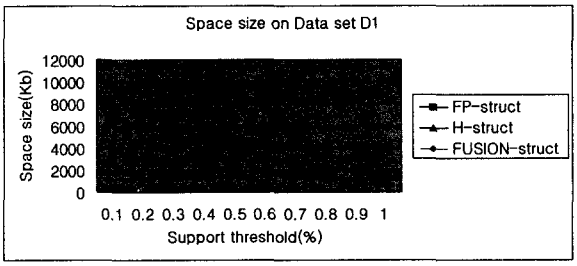

그림4 Space size on Data set D1

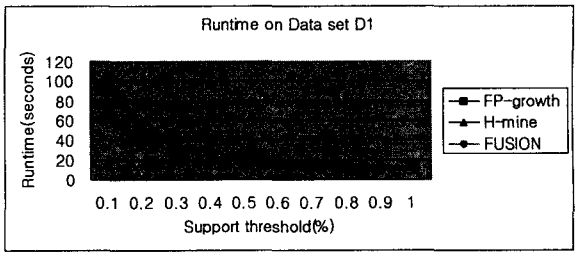

그림5 Runtime on Data set D1

5. 결론

본 논문에서는 후보항목집합 발생과정 없이 압축빈발항목집합과 동적링크집합을 이용하여 저장공간 축소와 실행시간 개선을 고려한 연관규칙 알고리즘을 제안하였다. 웹 환경등에서 실시간으로 연관 규칙 탐사를 수행하기 위해서는 고속의 연관 규칙 알고리즘이 필요하며, 특히 대용량 데이터에 대해 적은 저장공간을 사용하여 연관규칙을 수행할 경우, 보다 많은 양의 데이터를 메모리상에서 처리할 수 있기 때문에, 저장공간의 축소가 실행속도의 향상에 직접적인 영향을 줄 수 있다. 이와 같은 관점에서 볼 때 저장공간 축소와 실행시간 개선을 고려한 FUSION 연관규칙 알고리즘은 많은 분야에서 유용하게 사용될 수 있다.

6. 참고 문헌

[1] R.Agrawal, R.Srikant Fast algorithms for Mining Association Rules , Proc. Of the 20th Intl conference on Very Large Database, Santiago, Chile, Sept. 1994

[2] J.Han, J.Pei, and Y.Yin. Mining frequent patterns without candidate generation. In SIGMOD'00, pages 1-12, 2000

[3] J.Pei, J.Han, H.Lu, S.Nishio, S.Tang and D.Yang. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. ICDM, pages 441-448, 2001