연관규칙 마이닝에서의 Concept 개요

김지혜[0] 김현민 R.S.Ramakrishna
광주과학기술원 정보통신공학과
(jhkim[0], hmkim, rsr)@kjist.ac.kr

Introduction to Concept in Association Rule Mining

Jihye Kim[0] Hyun-min Kim R.S.Ramakrishna
Dept. of Information and Communication
Kwang-ju Institute of Science and Technology

요 약

데이터 마이닝의 대표적인 기법인 연관규칙 마이닝을 위한 다양한 알고리즘들이 제안되었고, 각 알고리즘에 따른 대용량 데이터에 대한 신속한 탐색을 위한 독특한 자료구조가 제안되었다. 각 자료구조의 특성에 따른 알고리즘 성능은 데이터의 패턴에 크게 의존한다. 본 논문에서는 Concept을 형성하는 세가지 대표적인 자료구조인 Hash Tree, Lattice, FP-Tree에 대해 비교·분석해보고, 데이터 패턴에 적합한 효율적인 알고리즘의 설계 위한 framework을 제안한다.

1. 서 론

데이터 마이닝이란 대량의 데이터로부터 쉽게 드러나지 않는 유용한 정보들을 추출하는 과정을 말한다. 데이터 마이닝은 흔히 Knowledge Discovery in Database(정보발견)라고도 불리며, 데이터 마이닝 기법으로는 연관규칙과 서열 연관규칙, 클러스터링, 분류 등이 있다.

연관규칙은 [1]에서 처음 소개되었다. 연관규칙 알고리즘을 통해 찾아낸 규칙들은 소비자 구매형태나 웹에서의 사용자 접근 패턴 등의 상업적인 적용뿐 아니라, 최근 대두되고 있는 바이오 정보학에서의 바이오 데이터와 같은 과학분야의 데이터 분석에도 사용된다.

본 논문에서는 연관규칙에 사용되는 여러 가지 Concept의 비교·분석에 초점을 맞춘다. 특히 데이터의 특성에 따른 적절한 Concept의 선택을 위한 기준을 제안한다.

2. 관련연구

2.1 연관규칙 마이닝

연관규칙이란 동시에 발행하는 사건들을 규칙의 형태로 표현한 것으로 특정 사건이 발생하면 동시에 혹은 일정한 시간 간격 사이에 다른 사건이 일어나는 관련성을 의미한다 $I=\{i_1,i_2,\cdots,i_m\}$ 을 항목이라 부르는 리터럴들의 집합이라하고, D 를 트랜잭션들의 집합이라 했을 때 각 트랜잭션 T 는 $T\subseteq I$ 인 항목들의 집합이다. 트랜잭션들은 *TID* 라 부르는 식별자를 가지고 있으며 X,Y 를 항목들의 집합이라 하면 $X\subseteq T$ 이고, $Y\subseteq T$ 이다. 이때 트랜잭션 T 는 X,Y 를 포함한다고 말한다. 전제 부(X)에 해당하는 조건 항목이 결론 부(Y)에 해당하는 항목들을 야기한다고 정의하며 연관규칙은 $R:X\Rightarrow Y$ 로 나타내고 이때 $X,Y\subseteq I$ 이고 $X\cap Y=\phi$ 이다. 이때 규칙의 타당성을 검증하기 위한 척도로서 지지도(support)와 신뢰도(confidence)가 적용된다. 지지도는 전체 항목 중에 연관규칙 $R:X\Rightarrow Y$ 를 지지하는 비율을 의미하는 척도를 말하며, 신뢰도는 X 의 모든 항목을 포함하고 있는 트랜잭션의 개수에 대하여 Y 또한 포함하는 트랜잭션의 비율을 의미한다.

2.2 연관규칙 마이닝의 Concept과 Dataset

Dataset은 연관규칙을 찾아낼 대상 데이터 베이스를 의미한다 연관규칙 마이닝에서의 Concept이란 연관규칙 알고리즘을 구현하는데 있어 Dataset의 항목들로부터 규칙을 찾아내는데 유용한 자료구조를 말하며,. Dataset의 신속한 접근을 위하여 Hash Tree, Lattice, FP-Tree 등으로 구성되어 있다.

3. 연관규칙 알고리즘의 Concept 비교

3.1 Hash Tree

연관규칙 마이닝의 대표적인 알고리즘인 Apriori[2], DHP[3], Apriori를 기본으로 하는 여러 병렬 알고리즘 – PDM[4], FDM[5], CD[6], DD[6], IDD[7], HD[7], CCPD[8] 등은 모두 크기별 빈발항목을 찾는 동안 내부적으로 Hash Tree를 생성한다.

그림 1은 Hash Tree를 Concept으로 하는 대표적인 알고리즘 Apriori이다. K단계의 빈발항목집합 L_k 로부터 $(k+1)$-후보항목집합 C_{k+1} 을 만드는 과정에서 C_{k+1} 을 Hash Tree로 구성하여 원래의 데이터 베이스 D 를 스캔하 여 C_{k+1} 로부터 L_{k+1} 를 얻는다.

```
Step 1 Frequent Pattern finding :
  L1 = { frequent 1-itemsets };
  for (k =2; Lk-1 ≠∅; k++ ) do begin
    Ck= apriori-gen ( Lk-1 ); // New candidates
    forall transactions t ∈ D do begin
      Ct = subset ( Ck, t);//Candidates contained int
      forall condidates c ∈ Ct do
            c.count ++;
    end
    Lk = { c ∈ Ck | c.count ≥ minsup }
  end
  Patterns = ∪k Lk ;

Step 2 Rule Generation:
  Rules = rule-generate( Patterns )
```

그림 1 Apriori Algorithm

TID	Items Bought
1	a c d f g i m p
2	a b c f l m o
3	b f h j o
4	b c k s p
5	a c e f l m n p

그림 2 Sample Database

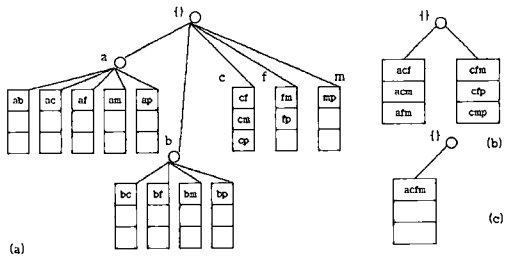

그림 3 (a) 2-size Hash Tree, (b) 3-size Hash Tree, (c) 4-size Hash Tree

그림 1의 Apriori 알고리즘을 바탕으로 하여 샘플 데이터(그림 2)에 대해 지지도 3 이상(minumum support = 3)인 빈발항목들을 찾는다는 조건이 주어졌다고 하자. 샘플 데이터 베이스는 사전적 순서로 정리되었다고 가정하며 그림 3은 그림 2의 샘플 데이터 베이스를 가지고 모든 빈발항목들을 찾는 과정 중에 생성되는 Hash Tree이다(bucket 의 크기는 3).

Hash Tree를 Concept으로 하는 Apriori의 경우, 각 단계에서 생성되는 Hash Tree의 크기는 생성되는 후보 항목의 개수에 비례한다. 각 단계에서의 생성된 후보 항목의 개수 i_k 대해 다음과 같은 Hash Tree의 크기를 예상할 수 있다. Hash Tree 생성 시간(T_{const}^{hash})은 단위 후보항목 생성 시간을 1로 가정하였을 때 식(1)과 같다.

$$T_{const}^{hash} \approx O(i_1 + \sum_{k=1}^{max\_size} i_k C_{k+1}) = O(i_1 C_2) = O((i_1)^2) \qquad (1)$$

$|D|$개의 트랜잭션으로 이루어진 Dataset이 Hash Tree에 저장된 후보항목들을 탐색하는 시간($T_{support}^{hash}$)은 식(2)와 같다.

$$T_{support}^{hash} \approx O(\sum_{k=1}^{max\_size} i_k C_{k+1} \times |D|) = O((i_1)^2 |D|) \qquad (2)$$

3.2 Lattice

Concept의 또 다른 자료구조로서 Lattice 구조는 Hash Tree에 비해 불필요한 내부 노드를 포함하지 않는 구조이다. 이 자료구조를 이용한 알고리즘들(PARTITION[9], Eclat-based[10], DIC[11], CHARM[12])은 Dataset의 형태로 vertical format을 사용한다.

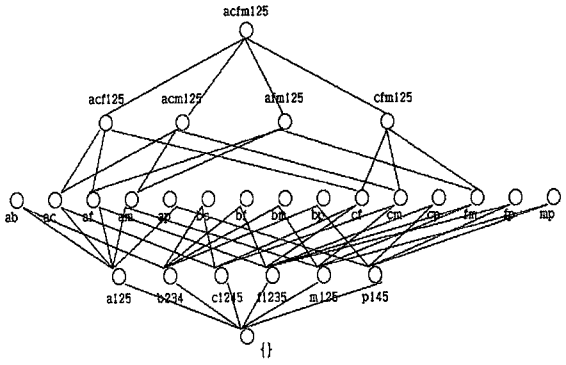

그림 4 Lattice

CHARM을 기본으로 하여 Lattice 구조를 살펴보겠다. CHARM은 두 번의 Dataset 스캔으로 프로세스를 종결한다. 그림 4는 그림 2의 샘플 데이터 베이스를 CHARM을 통해 보인 Lattice구조 이다.

Lattice를 Concept으로 하는 경우, 항목들의 합집합으로 후보항목을 만들고, Tidlist[12]의 교집합으로 빈발항목을 가려낸다. 따라서 매 단계마다 Concept을 생성해야 하는 Hash Tree의 경우와 달리 후보항목을 만드는 순간 빈발항목여부를 가려내므로 Concept의 중복을 줄여준다. Lattice의 생성 시간($T_{const}^{lattice}$)과 support 시간($T_{support}^{lattice}$)은 다음과 같다.

$$T_{const}^{lattice} + T_{support}^{lattice} \approx O(\sum_{k=1}^{max\_size} i_k C_{k+1}) = O(i_1 C_2) = O((i_1)^2) \qquad (3)$$

3.3 FP-Tree

Dataset을 압축한 형태의 Concept을 FP-Tree(Frequent Pattern –Tree)[13]로 구성하는 기법은 Lattice 구성과 마찬가지로 두 번의 데이터 베이스 스캔으로 모든 빈발 항목을 선출하며 Apriori 나 CHARM 알고리즘과 달리 후보항목을 생성하지 않고 데이터 베이스 자체에서 빈발항목을 뽑아낸다. 후보항목에 대한 중복성을 배제하므로 공간과 시간면에서 효율적인 Concept이라 할 수 있다.

그림 5, 6, 7 각각은 FP-Tree를 구성하는 알고리즘, FP-Tree를 이용한 FP-growth 알고리즘, 그림 2의 샘플 데이터 베이스를 이용해 생성한 FP-Tree의 예제이다.

FP-Tree 는 데이터 베이스를 압축하는 트리 구조를 이루어 빈발항목의 개수가 적을수록, 빈발항목의 지지도가 높을수록 FP-Tree가 차지하는 공간이 줄어든다. 지지도가 낮은 경우라 할지라도 원래의 데이터 베이스 크기 이상을 차지하지 않으므로 FP-Tree 생성 시간(T_{const}^{fp})은 빈발 1크기의 항목의 개수 i_1 에 대해 $O(i_1)$ 이며, 탐색 시간($T_{support}^{fp}$)은 식(4)와 같이 얻을 수 있다.

$$T_{support}^{fp} \approx O(i_1 \times |D|) \qquad (4)$$

1.데이터 베이스를 스캔. 크기가 1인 빈발항목을 선출하여 지지도순으로 정렬한 헤더테이블을 만든다.
2.FP-Tree의 루트를 null로 하고, 데이터 베이스의 트랜잭션을 읽어들여, 지지도 순으로 정렬하고, 이를 루트에서 한가지로 만든다. 다음 트랜잭션을 읽어들일 때 공통된 빈발항목에 대해서는 지지도를 1 증가하고, 새로운 빈발항목에 대해서는 노드를 추가한다. 이때 1에서 생성한 헤더테이블에 해당하는 노드들을 차례로 링크시킨다

그림 5 FP-Tree Construction Algorithm

Procedure FP-growth(Tree, α)
{
  if Tree contains a single path P
    then for each combination (denoted as β) of the nodes in the path P
      do
    generate pattern $\beta \cup \alpha$ with the support = minimum support of nodes in β;
  else for each a_i in the header of Tree do {
    generate pattern $\beta = a_i \cup \alpha$ with support = a_i.support;
    construct β's conditional pattern base and
    then β's conditional FP-Tree $Tree_\beta$;
    if $Tree_\beta \neq 0$
    then call FP-growth($Tree_\beta$, β) }
}

그림 6 FP-growth Algorithm

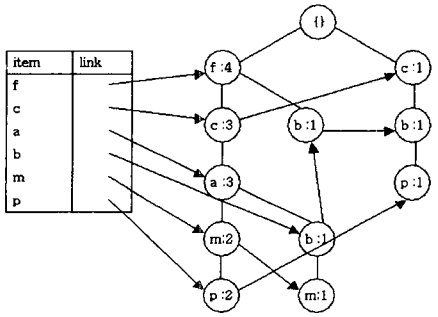

그림 7 FP-Tree

4.결론

. 표 1 Concept Comparison($n=i_k$)

Concept	생성시간	Support시간
Hash Tree	$O(n^2)$	$O(n^2 \mid D \mid)$
Lattice	$O(n^2)$	
FP-Tree	$O(n)$	$O(n\times \mid D \mid)$

표 1은 각각의 자료구조에 대해 생성시간과 support시간에 대해 정리한 것이다. 크기 1의 초기 빈발 항목 n과 $|D|$에 따른 데이터의 특성에 따라 각각의 Concept은 다음과 성능차이를 보인다. $|D|$ 값이 크고 sparse한 경우 Lattice의 Support 시간이 FP-Tree에 비해 빠름을 예상할 수 있다. 이는 Lattice Support방법이 vertical data형태의 TID-list의 교집합 연산으로 이루어 지기 때문이다. 그러나 n이 큰 경우에는 교집합 연산의 횟수가 n^2에 비례해 증가하기 때문에 FP-Tree에 비해 성능이 떨어진다.

구현상의 space complexity가 중요한 성능의 요인으로 작용할 때 생성 시간 또는 생성되는 Concept의 크기는 Lattice>Hash Tree>FP-Tree순임을 고려해야 한다. Lattice의 경우 모든 후보항목을 하나의 lattice에 저장하므로 많은 저장공간을 필요로 하는 컴퓨팅 환경에서 적당하다. Dense한 데이터의 경우 또는 동일 데이터에서 지지도를 낮게 정하는 경우에는 FP-Tree의 성능이 다른 자료구조의 성능보다 월등함을 볼 수 있다. FP-Tree는 후보항목을 생성하지 않고 축소된 Dataset형태로 Concept을 구성하기 때문에 저장 공간과 실행 속도면에서 탁월하다. 빈발항목의 수가 적고 Dense한 데이터는 특히 Bioinformatics 처리를 위한 Biodata (DNA, RNA, Amino Acid항목)에서 자주 볼 수 있다. FP-Tree는 적은 빈발항목에 대한 저장공간 축소기능이 뛰어나기 때문에 많은 저장용량을 요구하는 biodata를 처리하는데 효과적이다.

4. 참조논문

[1] R.agrawal,T.Imielinkski,A.Swami"Mining Associations between Sets of Items in massive Databaes"Proc. Of the ACM-SIGMOD 1993 Int'l conference on management of Data,Washington D.C.,May(1993)

[2] R.Agrawal, R.Srikant "Fast algorithms for Mining Association Rules", Proc. Of the 20th Int'l conference on Very Large Database, Santiago, Chile, Sept. (1994)

[3] Jong Soo Park, Ming-Syan Chen and Philip S. Yu, "An effective Hash-Based Algorithm for Mining Association Rules," Proc. of the ACM SIGMOD Int'l Conference on Management of Data, San Jose, CA, (1995).

[4] J. S. Park, M. S. Chen and P. S. Yu, "Efficient Hash-Based Algorithm for Mining Association Rules," Proc. Int'l Conference Information and Knowledge Management, Baltomore, Md., Nov. (1995).

[5] D. Cheung et. al., "A Fast Distributed Algorithm for Minng Association Rules," Proc, 4th Int'l Conference Parallel and Distributed Information Systems, IEEE Computer Soc. Press, Los Alamitos, Calif, pp. 32-42, (1996).

[6] R. Agrawal, J.C. Shafer: "Parallel Mining of Association Rules", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, December (1996).

[7] Han, E.-H.; Karypis, G.; and Kumar, V. "Scalable parallel data mining for association rules," In ACM SIGMOD Conference Management of Data, (1997).

[8] M.J. Zaki et al., "Parallel Data Mining for Association Rules on Shared-Memory Multi-Processors," Proc. Supercomputing '96, IEEE Computer Soc. Press, Los Alamitos, Calif, (1996).

[9] A. Savasere, E. Omiecinski, and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases," VLDB 95, Zurich, Sept. (1995).

[10] M.J. Zaki et al., "Parallel Algorithms for Fast Discovery of Association Rules," Data Mining and Knowledge Discovery: An Int'l J., Vol. 1,No. 4, pp. 343-373, Dec. (1997).

[11] S. Brin, R. Motwani, J. D. Ullman, S. Tsur, "Dynamic Itemset Counting and Implication Rules for Market Basket Data," SIGMOD 97, AZ. (1997).

[12] M.J.Zaki and C.Hsiao. Charm:An efficient algorithm for closed association rule mining. In Technical Report 99-10, Coputer Science, Rensselaer Polytechnic Institute (1999)

[13] J.Han, J.Pei, and Y.Yin. Mining frequent patterns without candidate generation. In Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data(SIGMOD'00), Dallas, TX, May(2000)