연관 규칙과 시간 간격을 함께 탐색하는 알고리즘

신윤재, 이기용
숙명여자대학교 컴퓨터과학부
e-mail : yoonjaeshin@sookmyung.ac.kr, kiyonglee@sookmyung.ac.kr

Mining Association Rules with Time Intervals

YoonJae Shin, Ki Yong Lee
Dept. of Computer Science, Sookmyung Women's University

요 약

로그 데이터 속에서 시간차를 두고 발생하는 트랜잭션 혹은 이벤트를 감지하는 일은 유통, 마케팅, 금융 등 다양한 분야에서 활용될 수 있다. 데이터베이스 분야에서 반복되는 패턴을 감지하는 알고리즘은 종종 소개되었지만, 데이터의 특성과 트랜잭션 간의 시간 간격을 고려한 연관 규칙 탐색 알고리즘 연구는 빈약했다. 본 논문에서는 정해진 구간에서 반복되는 패턴을 찾거나 주어진 아이템에 대한 주기를 찾는 등의 기존연구와 달리 전체 데이터베이스를 스캔하여 찾을 수 있는 연관 규칙과 그 연관 규칙이 반복되는 시간 간격을 함께 탐색하는 알고리즘을 제안한다. 또한, 제안하는 알고리즘의 처리시간에 대한 실험을 통해 성능을 확인한다

1. 서론

사물인터넷과 같은 디지털 환경에서 다양하고 많은 양의데이터를 수집되면서, 그 데이터 중 숨겨져 있는 유용한 상관관계를 찾아내는 데이터마이닝 분야의 중요성이 강조되고 있다. 특히, 로그 데이터에서 시간차를 두고 발생하는 트랜잭션 혹은 이벤트를 감지는 일은 일반적이지 않은 이벤트들을 예측, 예상, 감지하는 등 다양한 분야에서 활용할 수 있다.[1]

데이터베이스 분야에서 반복되는 패턴을 찾고자 하는 노력은 꾸준히 있었다. 그 중 대부분 아이템 혹은 이벤트가 발생한 시간 간격보다는 발생 순서에 주목한 순차패턴 혹은 동시에 발생한 이벤트에 의미를 부여하는 연관 규칙 탐색하는 연구가 주를 이뤄왔다. 즉, 알고리즘을 통해 반복하여 등장하는 연관 규칙과 그 시간 간격을 함께 마이닝하는 연구는 큰 주목을 받지 못했는데, 바로 이 주제를 본 논문에서 다루고자 한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 기존의 반복패턴 탐색 알고리즘의 관련 연구에 대해 간략히 소개하고, 3장에서는 시간 간격을 고려하여 연관규칙을 찾아내는 알고리즘을 제안한다. 4장에서는 제안 알고리즘의 성능을 평가하고, 마지막 5장에서는 결론을 내린다.

2. 관련 연구

논문에서 찾고자 하는 패턴은 $A \overset{n}{\Rightarrow} B$와 같은 형태로, A와 B는 아이템 혹은 이벤트들의 집합이며, n은 이들의 발생 시간 간의 차이를 의미한다. 이는 어떤 아이템들이 같은 장바구니 안에 나란히 등장하는지를 탐색하는 연관 규칙 마이닝의 대표적인 기법으로는 Apriori 알고리즘[2]의 출력이 (A, B)였던 것과 차이가 있다. 또, 아이템이 등장하는 순서를 고려하여 패턴을 찾아주는 순차 패턴 마이닝 알고리즘[3]이 A⇒B 형태의 출력을 내놓는 것과도 비교해 볼 수 있다. 단, Apriori 알고리즘과 순차 패턴 탐색 알고리즘 모두 기존에 데이터베이스 상의 타임스탬프를 활용하지 않았지만 본 논문에서는 이를 사용하였다.

연관 규칙과 그 시간 간격을 마이닝하기 위해 Apriori 알고리즘을 적용하면, 우선 데이터베이스 상의 등장하는 모든 단일 아이템의 빈도수를 계산하고, 사용자가 정의한 최소 지지도를 만족하는 아이템들을 이용하여 다음 단계(두 개의 아이템을 고려하는 단계)에서 빈발할 가능성이 있는 후보 집합을 생성한다. 이 때, 연관 규칙 (A, B)와 (B, A)가 같은 후보를 의미했던 것에 비해 $A \overset{n}{\Rightarrow} B$와, $B \overset{n}{\Rightarrow} A$는 서로 다른 후보이며, $A \overset{1}{\Rightarrow} B$와 $A \overset{2}{\Rightarrow} B$ 역시 다른 후보임에 주의한다. 즉, Apriori 알고리즘에서 (A,B)로 나타난 집합이 $A \overset{1}{\Rightarrow} B$, $A \overset{2}{\Rightarrow} B$, …, $A \overset{n}{\Rightarrow} B$, $B \overset{1}{\Rightarrow} A$, $B \overset{2}{\Rightarrow} A$, …, $B \overset{n}{\Rightarrow} A$로 더 많은 수의 후보로 나타나게 된다.

순차 패턴 마이닝 중 하나인 GSP 알고리즘[4]에서는 A⇒B와 B⇒A가 서로 다른 패턴을 의미한다는 점에서 본 논문에서 다루고자 하는 문제와 조금 더 유사하지만, Apriori 알고리즘과 마찬가지로 시간 간격을 고려하여야 하므로 많은 양의 후보 집합을 생성하게 된다. 후보 패턴을 만들지 않으면서 빈번한 패턴을 찾아내는 것으로 알려진 PrefixSpan 알고리즘[5]도, 역시 가능한 모든 시간 간격에 대해 PrefixSpan 트리를 작성해야 하므로 본 문제 해결에 적합하지 않다.

그러므로 본 논문에서는 사용자가 정의한 최소 지지도를 만족하는 연관 규칙과 그 시간 간격을 함께 추출하는 알고리즘을 제안하고자 한다.

3. 제안 알고리즘

3.1 문제 정의

본 논문에서 풀고자 하는 문제는 Table 1과 최소 지지도(minimum support)와 로그 데이터 D를 입력으로 받는다. D는 모든 트랜잭션의 집합으로 각각의 트랜잭션은 timestamp와 유일한 트랜잭션 ID(TID)를 가지며 아이템의 집합 I의 부분집합을 포함하고 있다. Table 1에서는 timestamp가 1일 때, 아이템 a, c가 속한 트랜잭션 T_1, 아이템 d, e가 속한 트랜잭션 T_2가 등장한 것이다. 동시간 대에 발생한 아이템은 중복되지 않는다고 가정한다.

출력은 I의 부분집합인 X, Y에 대해 $X \overset{n}{\Rightarrow} Y$의 형태의 연관규칙의 집합이며 이는 X가 등장한 후 n만큼의 시간 간격 후에 B가 로그 데이터에 사용자가 제시한 기준(최소 지지도)이상 등장하는 것을 의미한다.

TID	timestamp	itemsets				
T_1	1	a		c		
T_2	1				d	e
T_3	2	a		c		
T_4	3					e
T_5	3				d	
T_6	3		b			
T_7	4	a				
T_8	4			c		
T_9	5		b		d	
T_{10}	6	a				
T_{11}	6					
T_{12}	7			c		e
T_{13}	8		b		d	
T_{14}	9	a	b	c		

Table 1. 입력 로그 데이터 D

3.2 제안 알고리즘

X, Y에 속한 총 아이템의 수가 1개이면서 최소 지지도를 만족하는 연관 규칙의 집합을 AR_1이라고 하자. 본 논문에서는 $X \overset{n}{\Rightarrow} Y$ 패턴을 찾을 때, X, Y에 속한 총 아이템의 수가 1개인 연관규칙 집합 AR_1부터 AR_2, AR_3, …, AR_l을 순서대로 차례로 탐색한다.

제안하는 알고리즘은 크게 두 부분으로 구성된다. 첫 번째는 입력 데이터 D로부터 AR_1를 생성하고, AR_1을 이용하여 AR_2 구하는 알고리즘이다. 두 번째 알고리즘은 이 AR_2를 가지고 AR_3, AR_4, …, AR_l로 확장해 나가는 것이다.

첫 번째 알고리즘은 기존의 알고리즘과 제안 알고리즘이 공통으로 사용하는 부분으로, 빈발하는 아이템으로만 이루어진 AR_2를 찾기 위해 입력 데이터 D를 스캔하여 I에 속한 각각의 단일아이템의 등장횟수를 먼저 계산한다. 이중 사용자가 입력한 최소 지지도를 만족하는 아이템들의 집합이 AR_1에 해당한다. AR_1에 포함된 아이템을 조합하여 l=2인 연관규칙의 후보 집합을 만들고, 가능한 모든 시간 간격에 대해 D에서 등장한 횟수를 센다. 이중 최소 지지도를 만족시키는 연관 규칙만 AR_2에 추가한다.

예를 들면, Table 1의 로그 데이터로부터 minimum support =3인 연관규칙과 그 시간 간격을 찾아보자. Table 2은 로그데이터에서 등장하는 모든 아이템과 그 support값을 나타낸 것이다. 이 경우, 모든 아이템이 minimum support를 만족시키므로 가능한 모든 아이템 조합에 대해 Table 3과 같이 시간 간격 별로 등장 횟수를 조사한다.

Table3은 time interval이 1일 때의 테이블로, $\{b\} \overset{1}{\Rightarrow} \{a\}$, $\{c\} \overset{1}{\Rightarrow} \{b\}$, $\{c\} \overset{1}{\Rightarrow} \{d\}$, $\{d\} \overset{1}{\Rightarrow} \{a\}$, $\{d\} \overset{1}{\Rightarrow} \{c\}$가 AR_2에 속하는 것을 알 수 있다. 마찬가지로 Table4, Table 5로부터 $\{b\} \overset{2}{\Rightarrow} \{a\}$, $\{a\} \overset{3}{\Rightarrow} \{c\}$도 AR_2에 속한다.

item	support
a	5
b	4
c	4
d	4
e	3

Table 2. AR_1

item	support
b⇒a	3
c⇒b	3
c⇒d	3
d⇒a	4
d⇒c	3

Table 3. AR_2 (interval =1)

Figure 1은 위의 과정을 pseudocode로 나타낸 것이다.

```
Input: log data, minimum support
Output: A set of association rules with time
intervals
t: timestamp

Generate AR1
For (n = 0; n < N; n++)
  For (t = 0; t < N; t++)
    For every possible item pairs(X ⇒(t) Y)
      count(X ⇒(t) Y) ++
    End
  End
  If count(X ⇒(t) Y) ≧ minimum_support
    AR2 = AR2 ∪ { X ⇒(t) Y }
End
```

Figure 1. AR_2을 구하는 알고리즘

다음은 위에서 구한 AR_2를 바탕으로 AR_3를 구하는 과정이다. 여기에서는 앞서 구한 AR_2중 시간 간격 t가 Figure 2를 따라 동일한 패턴들을 중복되지 않게 조합하고, minimum support를 만족하는 경우 AR_3에 추가하고 같은 방법으로 확장하여 AR_4, AR_5, …, AR_l을 구한다.

Table 1에서 생성된 AR_2로부터 {b, d}$\overset{1}{\Rightarrow}${a}, {b, d}$\overset{1}{\Rightarrow}${c}, {b}$\overset{1}{\Rightarrow}${a, c}, {c}$\overset{1}{\Rightarrow}${b, d}, {d}$\overset{1}{\Rightarrow}${a, c}가 AR_3의 후보셋이 된다. 이 중 {b}$\overset{1}{\Rightarrow}${a, c}를 제외한 모든 조합이 실제 로그 데이터에서 3번 이상 등장하기 때문에 이르 모두 AR_3에 추가한다. Figure 2는 AR_l을 확장하는 과정을 pseudocode로 나타낸 것이다.

```
Input: AR_2
Output: AR_l (2 < l < N)

For ( l = 2; l < N; l++)
  For ( t = 1; t < N; t++) in AR_2
    If ( A_1 =t=> B_1 and A_2 =t=> B_2 with same time
    intervals t )
      If ( A_1 = A_2 || (| B_1 - B_2 | = 1))
        A_1 =t=> (B_1∩B_2) ∪ (B_1 - B_2) ∪ (B_2 -
        B_1)
      If ( B_1 = B_2 || (| A_1 - A_2 | = 1)):
        (A_1∩A_2) ∪ (A_1 - A_2) ∪ (A_2 - A_1) =t=>
        B_1
        If this rule appears more than
        minimum support, add id to AR_l
  End
End
```

Figure 2. AR_2를 입력으로 AR_l을 확장하는 알고리즘

4. 성능평가

실험에는 Intel i7-5820K (3.30GHz) 단일 머신을 사용하였다. 기존의 연관규칙 탐색 알고리즘을 본 논문의 문제에 맞춰 확장한 알고리즘과 제안하는 알고리즘을 Java로 구현했다.

입력데이터로는 동일한 랜덤 데이터 D에 대하여 레코드의 수를 100, 200, 300, 400, 500으로 조정한 부분 데이터를 사용하였으며, minimum support를 8로 지정하여 각 경우에 대하여 두 알고리즘의 CPU time과 후보집합의 원소 수를 비교하였다.

실험 결과 Figure 3에서 나타난 것과 같이 제안 알고리즘을 활용하여 후보집합에 포함된 연관 규칙의 수를 효과적으로 줄일 수 있음을 확인하였다. 후보집합을 탐색하는데 소요된 시간 역시 제안 알고리즘을 사용할 때 크게 단축되었음을 Figure 4에서 확인할 수 있다.

5. 결론

본 논문에서는 로그 데이터로부터 등장하는 아이템(혹은 아이템 집합)간의 시간 간격과 연관 규칙을 함께 탐색하는 알고리즘을 제안했다. 기존의 연규 규칙 탐색에 사용되었던 Apriori를 활용하되, 시간 간격이 동일한 후보 규칙들에 대해 일종의 join연산을 적용하여 후보 규칙들을 확장하였다. 그 결과 기존의 연관 규칙 탐색 알고리즘을 확장한 알고리즘에서 생성되는 무분별한 후보규칙의 수를 줄이며, 그에 따른 전체 연산에 소요되는 처리 시간을 효과적으로 줄일 수 있었다.

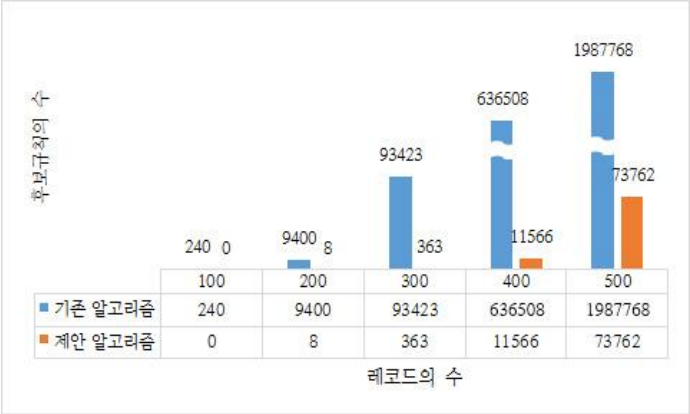

Figure 3. 레코드 수에 따른 생성된 후보규칙의 수

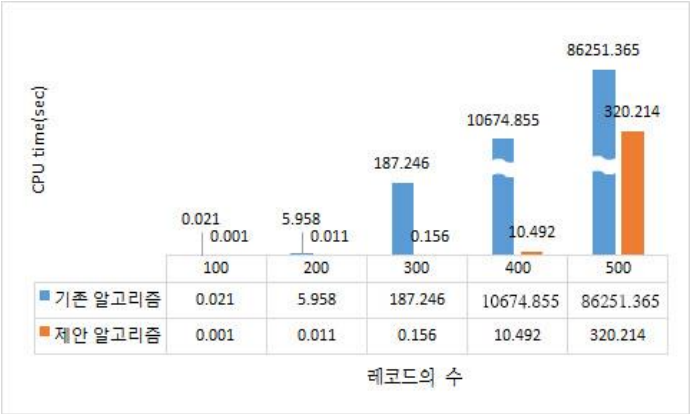

Figure 4. 레코드 수에 따른 CPU time

Acknowledgement

이 논문은 2016 년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.NRF-2015R1C1A1A02037071).

참고문헌

[1] F. Rasheed, M. Alshalalfa, and R. Alhajj. Efficient periodicity mining in time series databases using suffix trees. *IEEE TKDE*, 23:79-94, 2011

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*. Vol. 1215. 1994.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *Data Engineering, 1995.* In *Proc. of the Eleventh International Conference on*. IEEE, 1995.

[4] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In: *International Conference on Extending Database Technology*. Springer Berlin Heidelberg, 1996. p. 1-17.

[5] Han, Jiawei, et al. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proc. of the 17th international conference on data engineering*. 2001.