트랜잭션이 존재하지 않는 데이터베이스 상의 타임 윈도우를 이용한 마이닝 기법

이준섭*, 김민수, 김응모
*성균관대학교 전기전자 및 컴퓨터공학과
e-mail : joonsub@ece.skku.ac.kr

Mining Interesting Rule in Non-Existed Transaction Database Using Time-Windows

Joon-Sub Lee*, Min-Soo Kim, Ung-Mo Kim
*Dept. of Electrical & Computer Engineering, SungKyunKwan University

요 약

기존의 Association Rule 의 적용은 각 사건들이 고유한 연관관계를 갖는 다는 전재 하에 이를 이용하여 Data Mining Association Rule(연관규칙)을 적용해 왔다. 만약 이러한 연관규칙이 포함하지 않는 데이터에 대해서는 기존의 Rule 을 이용하기 위해서는 현재의 데이터를 재 구성해야만 하는 필요성이 존재를 해왔다. 본 논문에서는 위와 같은 데이터의 재 구성없이 연관규칙을 포함하지 않은 데이터로부터 새로운 알고리즘을 이용하여 기존의 Association Rule 을 적용하고자 한다.

1. 서론

데이터의 급속한 팽창으로 인해 기존에 사용하던 간단한 통계분석 방식으로는 더 이상 이를 감당하기에 어려운 상태에 도달했다. 데이터 마이닝은 이러한 기존의 축적된 데이터를 통해서 사용자에게 유용한 정보를 제공해 주는 기술로 많은 분야에서 연구가 진행 중이며 현실에 적용되어 지고 있다. 본 논문의 초점은 데이터 마이닝의 여러 테크닉 중 연관규칙에 초점을 두고 있다. 기존의 연관 규칙은 분석하려 하는 데이터집합 속에 각 데이터간에 일련의 연관 규칙이 존재한다는 전재 아래, 각 관계의 발생 빈도를 측정하는 것이 알고리즘의 주를 이루고 있다. 그러나 현실 세계에는 위와 같이 연관을 가지고 있는 데이터 집합도 존재를 하지만, 그렇지 않은 경우 또한 상당수를 차지 하고 있음을 알 수가 있다. 따라서 연관관계가 존재하지 않는 데이터베이스에 대해서는 연관규칙을 적용하기 이전 단계에서 Data를 기존에 연관규칙에 적용 가능하도록 변경하는 작업이 필요했다. 이는 연관규칙을 적용함에 있어서 두 가지의 Path가 필요함을 의미하며, 연관규칙에서 발생되는 연산시간과 더불어 또 다른 연산 시간을 필요로 한다. 그러므로 이러한 두 가지의 Path를 하나로 하여 구성해야 할 필요성이 대두 된고 볼 수 있다.

본 논문에서는 위에서 제시한 Random 기반의 Data 집합에 대하여 새로운 데이터 집합의 구성없이 연관규칙을 적용할 수 있는 새로운 알고리즘을 제시하고자 한다. 제시하고자 하는 알고리즘은 시간에 따른 사건의 발생을 시간 축을 이용하여 각 사건들을 분류하고 일정한 시간에 발생하는 각 사건들을 하나의 연관된 규칙을 가지는 사건들로 분류를 한다. 따라서 비록 각 사건들이 서로 연관 규칙이 없기 때문에 각 자료들이 데이터베이스 상에서는 서로 독립적인 사건들로 보여 질 수도 있지만, 새롭게 제시되는 알고리즘을 사용하여, 각각의 시간 축에서 발생빈도가 높은 규칙들은 정의함으로 해서, 기존의 알고리즘에서는 찾을 수 없었던, 혹은 적용되는 도매인에서 정의 내리지 못한 규칙들을 새롭게 정의 가능하다고 본다.

본 논문의 기초는 전력계통의 송배전 시스템의 사건, 사고들을 분석에 데이터 마이닝 연관규칙을 사용하기 위하여 고안되었으며, 이를 전력계통이라는 특정한 도매인에 국한 시키지 않고 사용하기 위해 본 논문에서는 일반화된 포맷을 제공하고자 한다.

본 논문의 구성은 2장에서 기존의 마이닝 연관규칙 알고리즘에 대하여 간단히 소개를 한다. 3장에서는 새로운 알고리즘에 대한 소개와 그 효용성에 대하여 논한다. 4장과 5장에서 현재의 연구 진행 사항, 앞으로의 진행 방향과 결론에 대하여 제시한다.

2. 연관규칙(Data Mining Association Rule)

본 논문은 데이터 마이닝 기법 중 Apriori 알고리즘[2]에 기반을 두고 있다. Apriori 알고리즘은 Association Rule[1]을 발견해 내는 알고리즘 중 가장 최적화 되어있으며, 현재 많은 부분에서 사용이 되고 있다. 2장 연관 규칙에서는 새롭게 제시되는 알고리즘의 기본이 되는 Association Rule의 개요와 Association Algorithm중 Apriori Algorithm에 대한 간단한 소개를 한다. 또한 기존의 알고리즘을 Random한 Data집합 상에서 적용하는데 있어서의 문제점에 대하여 분석하고자 한다.

2.1 Association Rules 개요

현실세계에서 발생하는 사건, 사고들을 item이라 할 때, Transaction이란 각각의 item들의 집합으로 구성되어지는 하나의 item set이라 볼 수 있다. 간단한 예로 item X, Y가 하나의 XY라는 Transaction을 이루고 있다고 할 때, Association Rule(연관규칙)은 X→ Y로 표현된다. 직관적으로 Association Rule을 관찰하여보면, 하나의 Transaction에 X와 Y가 함께 포함되어 있다고 말할 수 가 있다. 이해를 돕기 위해 Association Rule에 대한 간단한 예[1]를 들어보면 다음과 같다. Transaction을 이루고 있는 전체 테이블 중, 30%가 맥주(beer)를 포함하고 있고, 또한 기저귀(diaper)를 포함하고 있다. 이중 2%가 양쪽 모두를 포함하고 있다. 여기서 30%를 Rule의 Confidence라고 하고, 2%를 Rule의 Support라고 한다. 여기서 Association rule의 문제점은 사용자의 min Confidence와 min Support에 맞는 모든 Rule를 찾아야 한다는데 있다. 이는 사용자에게 유용하지 않은 Rule또한 찾아진다는 문제점과 유용한 Rule을 발견하는데 있어 noise와 같은 역할을 한다. 따라서 기존의 연구[2]들은 이러한 문제점을 해결하려는데 초점을 맞추고 있다.

위의 문제들에 대하여 최적화된 기법중의 하나가 Apriori 알고리즘이다. 우리는 여러 가지 연관규칙 알고리즘 중, 효용성과 성능면에서 이미 입증된 Apriori 알고리즘을 사용하였다. 대부분의 연관규칙의 알고리즘들은 아래에서 소개할 Apriori 알고리즘을 기반으로 작성되었다.

2.2 Apriori 알고리즘

```
1)  L1 = {large 1-itemsets};
2)  for ( k = 2; Lk-10; k++) do begin
3)    Ck = apriori-gen(Lk-1);
        //New candidates
4)    forall transactions t∈ D d begin
5)        Ct =subset(Ck ,t);
        // Candidates contained
6)        forall candidates c∈Ct do
7)            c.count++;
8)        end
9)        Lk = { c∈Ct | c.count ≥ minsup}
10)   end
11)   Answer = ∪k Lk
```

[Fig. 1 Apriori 알고리즘]

[Fig.1]은 Apriori 알고리즘을 보여주고 있다. Apriori 알고리즘은 frequent item 집합을 찾기 위해 multiple path를 사용하고 있다. k번째의 path에서 알고리즘은 k개의 모든 아이템 집합을 찾게 되며 이를 k-item 집합이라 한다. 각각의 pass는 두 가지의 단계로 이루어져 있다. 우선 F_k를 frequent item 집합이라 정의를 하고, C_k를 후보 item 집합- frequent item집합이 될 수 있는 잠재성을 가진 집합 -이라 정의를 하자. 첫 번째 단계는 (k-1) item 집합으로부터 후보 생성단계이다. F_{k-1}을 (k-1) pass에서 찾은 frequent item 집합이라고 한다면, 이것을 이용하여 후보 집합인 C_k를 생성한다. 후보 생성 단계의 절차는, 후보 집합인 C_k에 대하여 모든 frequent item 집합의 superset임을 보장한다. 특별히 C_k는 메모리상의 hash-tree 구조를 이용하여 저장이 된다. 그리고 나서 생성되어진 C_k는 다음단계인 support counting 단계를 거치게 된다. 각각의 transaction마다 transaction속에 포함되어져 있는 C_k에 대한 후보들에 대하여 hash-tree 구조와 그들의 support count를 사용하여 증가하게 된다. 마지막 pass에 도달하게 되면, 각각의 C_k에 대하여 과연 그 후보 집합이 frequent item 집합 즉 F_k가 될 수 있는지에 대하여 검사하게 된다. Apriori 알고리즘은 F_k 또는 C_{k+1}이 비워지게 되면 종료하게 된다.

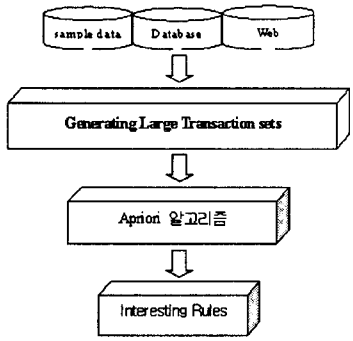
2.3 입력 데이터 형식

Transaction 테이블 T는 두 개의 컬럼을 가지고 있다. 하나는 transaction 정의부(tid)이고 다른 하나는 item 정의부(item)로 정의 할 수 있다. tid에 대한 item의 수는 변수이며, 테이블이 생성되는 동안에는 알 수 가 없다. 하나의 tid에 모든 item들은 하나의 단일 튜플의 컬럼에 나게 되며, 다른 튜플에서도 나타날 수 있다. transaction에 대한 item들의 개수는 DBMS가 지원하는 최대 컬럼의 개수보다 많이 나타는 것을 종종 볼 수 있다. 간단한 실례로 우리의 실세계의 데이터베이스의 최대 각 transaction당의 item들의 개수는 872이거나 다른 경우에는 700정도 밖에 되지 않음을 볼 수 가 있다. 정리하자면 transaction당 item들의 평균 개수는 단지 9.6에서 4.4 정도 밖에 기대 될 수밖에 없다. 정리해보면 하나의 일련의 사건에 대하여 약 5개에서 10정도의 사건정도가 파악 가능한 것이다. 그러나 사용자의 min conf.나 min sup.의 설정에 의해 이는 더 많은 수의 item들도 하나의 transaction에 포함될 수 있기에 앞으로의 연구에는 크게 지장을 주지 안 수 있다.

2.4 문제점

위에서 간략하게 Association Rule과 그 중 최적화된 하나의 기법인 Apriori 알고리즘에 대하여 알아보았다. 위에서 볼 수 있듯이 Apriori 알고리즘은 Transaction이 생성되어 있는 상태에서 적용이 가능하다. 그러나 Random한 기반의 Data, 즉 어떠한 각각의 사건, 사고 사이에 연관규칙이 존재하니 않는 Data에서는 위의 알고리즘을 바로 적용하지 못한다는 점이 문제점으로 대두 된다. 위와 같은 상황에서 기존의 처리 방식은 1차적을 트렌젝션이 존재하지 않은 데이터를 사용자의 요구 조건에 의하여, 일정한 연관관계를 가질 수 있도록 Data를 가공하는 단계를 거치

게 된다. 1차 데이터 가공 단계를 거친 후, 2차적으로 기존의 Apriori등의 연관규칙 알고리즘을 적용하여 사용자에게 유용한 Rule를 발견해 낸다. [Fig.2]에서 위에서 설명한 일련의 단계를 보여주고 있다.



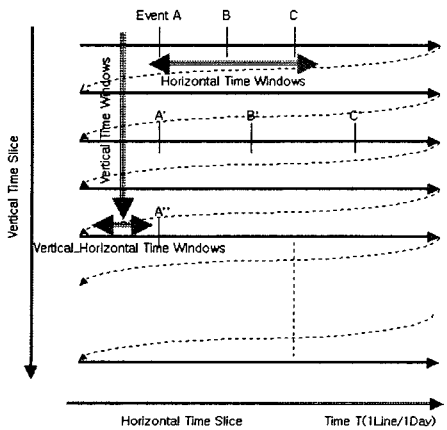
[Fig.2 알고리즘 흐름도]

위의 그림에서 보듯이 Random기반의 Data 집합에서 연관규칙을 이용하기 위해서는 Data를 가공해야 한다는 것이 필요하다. 따라서 위의 Data 가공 부분과 Apriori알고리즘의 통합이 필요하게 된다. 따라서 위의 통합을 위하여 시간이라는 Time축을 이용하여 각각의 사건 사고들에 대하여 연관규칙을 적용하게 되고, 이를 Count함으로 해서 차후에 Apriori를 다시 사용하지 않더라도 사용자에게 안정되고, 고가치의 정보를 제공할 수 있는 알고리즘을 제시하고자 한다.

3. 제안하는 새로운 알고리즘

3.1 새 알고리즘의 개념

새로 제안 하는 알고리즘은 기본적으로 사건(Item)을 시간이라는 축으로 분류하여, 각기 발생하는 시간을 기본으로 하여 각 개별적 Item들을 구별한다. [Fig.3]에서 보듯이 기본적으로 하루를 기본으로 하는 가로축을 Horizontal Time Slice(HTS)라 하고, 포함하는 모든 날들에 대한 파라미터 값으로 -가로축을 하루라고 했을 때- 하는 Vertical Time Slice(VTS)라는 파라미터 값으로 표현 된다.



[Fig.3 알고리즘 개념]

위의 값들은 기본적으로 하나의 HTS의 값을 어떻게 지정하느냐에 따라 가변적일 수 있는 값으로 표현된다. 데이터 분석의 임계 값으로 사용하는 것은 Horizontal Time Windows(HTW), Vertical Time Windows(VTW) 그리고 Vertical_Horizontal Time Windows(VHTW)로 정의 할 수 있다. 위의 값들은 사용자가 각각의 분석하려는 도매인에 특성에 따라 가변적으로 정의 가능하다. 정의한 HTW와 VTW, VHTW에 대하여 예를 들어 설명하면 다음과 같다.

위 그림에서 보듯이 각각의 독립적인 사건 ABC가 하나의 시간 선 안에 존재를 한다. A라는 사건에 대하여 사용자가 정의 한 시간의 범위(HTW)가 시작되게 된다. 그 시간 범위 안에 B혹은 C라는 사건이 포함이 된다고 가정하자. 만약 다른 시간 선 안에 A와 같은 사건인 A'가 발생을 하게 되고 같은 시간 범위 안에서 B', C'가 발생을 하게 된다. 이러한 Transaction이 다수 발생을 하게 된다면 A라는 사건이 일어나게 되면, B라는 사건이 일어나게 되고, 이어서 C라는 사건도 발생하게 된다는 규칙이 발생하게 된다. 즉 A→B→C라고 정의 될 수 있다. VTW는 VHTW와 함께 적용이 된다. 어떠한 사건에 대하여, Test되는 사건이 몇일 동안(VTW), 어느 정도의 시간대(VHTW)에서 공통으로 일어나는 지에 대하여 사용자에게 유용한 정보를 제공 가능하다. VHTW는 정의된 사건이 발생된 Horizontal Time Windows상에서의 시간을 Middle Point로 한다. 따라서 VHTW는 정의된 사건이 같은 시간에 발생되었는지에 대하여 파악할 수 있는 좌우측의 임계값으로(±Horizontal Time Windows상에서의 Item 발생시간) 적용된다. 위의 그림에서 보여 지듯 A라는 사건은 일정한 시간에 각 Vertical Time Windows 상에서 나타남을 볼 수 있다. 그러나 발생시간이 항상 일정할 수는 없음으로 VHTW를 사용하여 그 범위 안에 발생하는 사건에 대해서는 같은 시간에 발생하는 사건으로 파악 가능한 것이다.

아래의 [Table.1]은 알고리즘에 사용되는 상수들을 정의하고 있다. [Fig.4]는 Horizontal 상에서의 각 Item사이에의 규칙을 생성하는 알고리즘이며, [Fig.5]는 Vertical 상에서의 각 Item사이에의 규칙을 생성하는 알고리즘을 보여주고 있다. 아래의 알고리즘은 기본적으로 위에서 설명한 내용을 기본으로

상수명	정 의
L_E	생성전의 데이터 소스
C_H	생성된 후보 Horizontal Transaction set
C_V	생성된 후보 Vertical Transaction set
L_H	생성된 결과 Horizontal Transaction Set
L_V	생성된 결과 Vertical Transaction Set
.name	L_T 테이블의 컬럼(transaction 이름)
.count	L_T 테이블의 컬럼(transaction 발생횟수)
M	L_E 테이터 소스에 속한 모든 item들의 개수
T_H	사용자 정의 Horizontal 시간범위
T_V	사용자 정의 Vertical 시간 범위
T_{V_H}	사용자 정의 Vertical_Horizontal 시간범위
min_sup	사용자 정의 Minimum Support
M_T	Vertical Time Slice의 개수

[Table.1 상수 정의]

하여 작성되었다.

```
1) L_E={events};
      //각각의 event들은 서로 독립적인 사건
2) Let C_H=0;    // Candidate sets
3) for(m=0;m=m-1;m++)
4)    for(f=0; f=(발생된 Event의 갯수)in T_H; f++)
6)      Temp = make transaction set;
//as like L_T+L_T+1     (ex. AB, ABC, ABCD ··· )
7)      if (Temp ∈ C_H)
8)      C_H.count++ where Temp = C_H.name;
9)      else
10)     Add Temp to C_H.tranname & set C_H.count=1;
11)   End
12) End
13) forall Transaction c∈C_H do
14)     L_H = { c∈C_H | C_H.count ≥ minsup}
15) End
16)     Answer = ∪_H L_H
```

[Fig.4 Horizontal 알고리즘]

```
1) L_E={events};
2) Let C_V=0;    // Candidate sets
3) for(m=0;m=m-1;m++)
4)      for(M_T=0;M_T=M_T-1;M_T++)
5)          if l_m.itemname ∈ l_m.rosetime ± T_V_H/2
6)              if l_m.itemname ∈ C_V.itemname
                    within C_V.rosetime ± T_V_H/2
7)                      Add C_V.rosetime++
                        Where l_m.itemname=C_V.itemname;
8)                  else
9)                      Add l_m.itemname to C_V.tranname
                            & set C_V.count=1;
10)     End
11) End
12) forall Transaction c∈C_V do
13)     L_H = { c∈C_V | C_V.count ≥ minsup}
13) End
14)     Answer = ∪_V L_V
```

[Fig.5 Vertical 알고리즘]

4. 연구 진행

현재 본 논문에서 제시된 알고리즘은 전력계통의 사고를 분석하고 이를 복구하기 위한 방식으로 현재 연구 중에 있다. 일차적으로 Horizontal에 대한 알고리즘의 평가는 이미 이루어져 있으며, 현재 Vertical 알고리즘에 대한 평가에 대하여 연구진행 중이다. 추후 연구과제는 위의 두 알고리즘에 대하여 서로 연관을 주어 통합하는 것과, Sample Data가 아닌 실제적인 데이터를 이용한 성능평가가 주요할 것이라고 생각 된다.

5. 결론

본 논문에서는 트랜젝션이 존재하지 않는 데이터베이스 상에서 Time Windows를 이용한 마이닝 알고리즘을 제시 하였다. 위의 알고리즘과 방식을 이용하여 서로 연관 관계가 전혀 없는 Item(사건, 사고)등에 대하여 추론이 가능할 것이라 본다. 또한 위와 같은 특징을 보이는 데이터 집합들이 무수히 존재 함으로 이를 이용한다면 그 동안 찾지 못한 많은 정보들을 찾을 수 있을 것이다.

참고문헌

[1] R. Agrawal, T. Imielinski, and A. Swami., "Mining association rules between sets of items in large database", ACM-SIGMOD, pp 207-216, In Proc. 1993

[2] R. Agrawal, R. Srikant., "Fast algorithms for mining association rules", VLDB, pp 487-499, In Proc. 1994

[3] Robert Cooley, Bamshad Mobacher, Jaideep Srivastava., "Data preparation for mining world wide web browsing patterns", Knowledge and Information Systems, In Proc. 1999

[4] Brian Lent, Arun Swami, Jennifer Widom., "Clustering association rules", ICDE, In Proc. 1997

[5] M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen, "Finding interesting rules form large sets of discoverd association rules", In 3rd International Conference on Information and Knowledge, November 1994

[6] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo, "Efficient algorithms for discovering association rules", KDD-94, AAAI Workshop on Knowledge Discovery in Database, July 1994

[7] G. Piatestsky-Shapiro, "Discovery, analysis, and presentation of strong rules", In G.Piatestky-Shaprio, editor, Knowledge Discovry in Database. AAAI/MIT Press, 1991

[8] C. Schaffer, "Domain-Independent Function Finding", PhD thesis, Rutgers Univeristy, 1990

[9] Heikki Mannila, Hannu Toivonen, A. Inkeri Verkamo, "Discovery of Frequent wpusodes in event sequences", Department of Computer Science Series of Publication C Report C-1997-15, University of Helsinki Finland, 1997

[10] Han, J., Dong, G. and Yin, Y. 1999. 'Efficient Mining of Partial Periodic Patterns in Time Series Database'. In *Proc. Fifteenth International Conference on Data Engineering*, Sydney, Australia. IEEE Computer Society. 106-115.

[11] Bayardo Jr, R.J. 1998. 'Efficiently mining long patterns from databases'. In *Proc. ACM SIGMOD Conference on the Management of Data*, Seattle, WA. ACM Press. 85-93.

[12] Berndt, D.J. and Clifford, J. 1995. 'Finding patterns in time series: a dynamic programming approach'. In *Advances in Knowledge Discovery and Data Mining*. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, et al. (eds.), AAAI Press/ MIT Press, 229-248.

[13] Chen, X., Petrounias, I. and Heathfield, H. 1998. 'Discovering temporal association rules in temporal databases'. In *Proc. International Workshop on Issues and Applications of Database Technology (IADT'98)*, 312-319.

[14] Han, J., Gong, W. and Yin, Y. 1998. 'Mining segment-wise periodic patterns in time-related databases'. In *Proc. Fourth International Conference on Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park. 214-218.