김일성종합대학학보
(자연과학)
주체104(2015)년 제61권 제10호

JOURNAL OF **KIM IL SUNG** UNIVERSITY
(NATURAL SCIENCE)
Vol. 61 No. 10 JUCHE104(2015).

망침입검출을 위한 련관규칙발굴의 실현

공 혜 옥

자료발굴(DM: Data Mining)은 자료기지로부터 새롭고 쓸모있는 패턴들을 얻어내는 지식발견과정으로서 일정한 알고리듬의 실행과정이다.

자료발굴의 패턴들에는 련관규칙, 분류패턴, 무리패턴, 순서패턴 등이 있으며 그중에서 련관규칙이 대표적이다. 련관규칙발굴과정은 빈발항목모임의 탐색으로 진행된다.

빈발항목모임 전부를 효과적으로 탐색하기 위한 연구가 많이 진행되였으며 그 대부분은 Apriori방법에 의거하고있다. Apriori방법에서는 먼저 빈발항목모임들을 생성하기 위한 후보항목모임들을 만들고 그 후보항목모임들중에서 빈발항목모임들을 찾아낸다. 그리고 발견된 빈발항목모임들은 다음의 반복과정에서 후보항목모임들을 생성하기 위하여 리용되며 이러한 과정은 후보항목모임들을 더는 만들수 없을 때까지 계속된다.

선행연구[2－4]에서는 대부분 후보항목모임들을 줄이거나 동시에 자료기지의 크기를 줄이는 방법에 대하여 론의하였으며 선행연구[1]에서는 자료기지안의 불필요한 트랜잭션 레코드들을 잘라버려 k－항목모임들의 후보모임 C_k의 크기를 감소시키고 I/O소비를 감소시키는 방법으로 Apriori알고리듬을 더 빠르고 더 좋게 갱신하였다. 그러나 이 방법에서도 k－빈발항목모임족 L_k를 생성할 때마다 여전히 자료기지조사가 여러번 반복되기때문에 효과성이 떨어진다. 이 방법에서 주목할만 한 점은 개별적인 트랜잭션의 길이 즉 매 트랜잭션안의 항목들의 개수를 론의하여 k－항목모임들의 후보모임으로부터 k－빈발항목모임을 생성할 때 불필요한 항목들과 트랜잭션들을 제거함으로써 다음단계에서 론의할 자료기지안의 항목들의 개수와 트랜잭션들의 길이가 상대적으로 작아지게 된다는것이다.

우리는 트랜잭션들의 길이가 짧거나 자료기지를 이루는 항목들의 개수가 비교적 적은 자료기지에서 해쉬화수법을 리용하여 L_k를 생성할 때마다 반복과정을 거치지 않고 자료기지를 한번만 조사하면서도 모든 k－부분항목모임들의 지지도를 계산할수 있는 가능성을 찾고 그것을 실현하여 SOT알고리듬[1]을 갱신하였다.

1. 해쉬화수법을 리용한 빈발항목모임 찾기

여기서는 해쉬화수법을 리용한 빈발항목모임 찾기알고리듬의 실행과정을 실례를 통하여 론의한다.

표 1과 같은 항목들과 자료기지에 대하여 론의하자.(최소지지도는 3으로 본다.)

먼저 개별적인 트랜잭션을 읽어 표 2와 같은 k－부분항목모임들을 생성한다.

표 1. 항목들과 자료기지

항목들
A
B
C
D
E
F
G

TID	트랜잭션
T_1	ACG
T_2	BCG
T_3	ABC
T_4	BC
T_5	BCDE
T_6	BC
T_7	ABCDF
T_8	BCDF
T_9	A
T_{10}	AC

표 2. 트랜잭션자료기지와 항목모임들의 생성

TID	1－항목모임	2－항목모임	3－항목모임	4－항목모임	5－항목모임
T_1	A, C, G	AC, AG, CG	ACG		
T_2	B, C, G	BC, BG, CG	BCG		
T_3	A, B, C	AB, AC, BC	ABC		
T_4	B, C	BC			
T_5	B, C, D, E	BC, BD, BE, CD, CE, DE	BCD, BCE, CDE, BDE	BCDE	
T_6	B, C	BC			
T_7	A, B, C, D, F	AB, AC, AD, AF, BC, BD, BF, CD, CF, DF	ABC, ABD, ABF, BCD, BCF, CDF, ACD, ACF, ADF, BDF	ABCD, ABCF, BCDF, ABDF	ABCDF
T_8	B, C, D, F	BC, BD, BF, CD, CF, DF	BCD, BCF, CDF, BDF	BCDF	
T_9	A				
T_{10}	A, C	A, C			

T_1을 읽으면 트랜잭션의 길이가 3이므로 1, 2, 3－항목모임이 생성된다. 매 항목모임들을 생성하는 방법은 Apriori알고리듬에서와 같다. 즉 A, C, G를 결합하여 2－항목모임들인 AC, AG, CG가 생성되고 3－항목모임인 ACG가 생성된다.

표 3부터 표 11까지는 매 트랜잭션을 조사하면서 하나의 트랜잭션에서 발생가능한 모든 부분항목모임들에 대하여 해쉬함수를 리용하여 새 주소공간은 1이라는 값을 가지면서 생성되고 이미 존재하는 주소공간의 값은 1씩 증가시키는 과정을 보여준다.

표 3. T_1을 읽은 후

k－항목모임들	AC	AG	CG	ACG
지지도	1	1	1	1
주소	13	17	37	137

표 4. T_2를 읽은 후

AC	AG	BC	BG	CG	ACG	BCG
1	1	1	1	2	1	1
13	17	23	27	37	137	237

마찬가지의 방법으로 T_3－T_{10}을 읽어 처리하면 표 5－표 11의 결과를 얻을수 있다.

표 5. T_3을 읽은 후

AB	AC	AG	BC	BG	CG	ABC	ACG
1	2	1	2	1	2	1	1
12	13	17	23	27	37	123	137

표 6. T_4를 읽은 후

AB	AC	AG	BC	BG	CG	ABC	ACG	BCG	BCG
1	2	1	3	1	2	1	1	1	1
12	13	17	23	27	37	123	137	237	237

표 7. T_5를 읽은 후

AB	AC	AG	BC	BD	BE	BG	CD	CE	CG	DE	ABC	ACG	BCD	BCE	BCG	BDE	CDE	BCDE
1	2	1	4	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1
12	13	17	23	24	25	27	34	35	37	45	123	137	234	235	237	245	345	2345

표 8. T_6을 읽은 후

AB	AC	AG	BC	BD	BE	BG	CD	CE	CG	DE	ABC	ACG	BCD	BCE	BCG	BDE	CDE	BCDE
1	2	1	5	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1
12	13	17	23	24	25	27	34	35	37	45	123	137	234	235	237	245	345	2345

표 9. T_7을 읽은 후

AB	AC	AD	AF	AG	BC	BD	BE	BF	BG	CD	CE	CF	CG	DE	DF
1	3	1	1	1	6	2	1	1	1	2	1	1	2	1	1
12	13	14	16	17	23	24	25	26	27	34	35	36	37	45	46

AB	AC	AG	BC	BD	BE	BG	CD	CE	CG	DE	ABC	ACG	BCD	BCE	BCG	BDE	CDE	BCDE
1	2	1	5	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1
12	13	17	23	24	25	27	34	35	37	45	123	137	234	235	237	245	345	2345

표 10. T_8을 읽은 후

AB	AC	AD	AF	AG	BC	BD	BE	BF	BG	CD	CE	CF	CG	DE	DF
1	3	1	1	1	6	2	1	1	1	2	1	1	2	1	1
12	13	14	16	17	23	24	25	26	27	34	35	36	37	45	46

ABC	ABD	ABF	ACG	ACD	ACF	ADF	BCD	BCE	BDF	BCF	BDE	CDE	CDF
2	1	2	1	1	1	1	2	1	1	1	1	1	1
123	124	126	137	134	136	146	234	235	246	236	245	345	346

ABCD	ABCF	ABDF	ACDF	BCDE	BCDF	ABCDF
1	1	1	1	1	1	1
1234	1236	1246	1346	2345	2346	12346

표 11. T_9, T_{10}을 읽은 후

AB	AC	AD	AF	AG	BC	BD	BE	BF	BG	CD	CE	CF	CG	DE	DF
2	4	1	1	1	7	3	1	2	1	3	1	2	2	1	2
12	13	14	16	17	23	24	25	26	27	34	35	36	37	45	46

ABC	ABD	ABF	ACG	ACD	ACF	ADF	BCD	BCE	BDF	BCG	BCF	BDE	CDE	CDF
2	1	2	1	1	1	1	3	1	2	1	2	1	1	2
123	124	126	137	134	136	146	234	235	246	237	236	245	345	346

ABCD	ABCF	ABDF	ACDF	BCDE	BCDF	ABCDF
1	1	1	1	1	2	1
1234	1236	1246	1346	2345	2346	12346

T_1로부터 생성된 AC, AG, CG, ACG는 해쉬함수에 의해 생성되는 주소공간 13, 17, 37, 137에 값 1을 보관한다.(표 3) 다음 T_2의 항목모임들을 생성한 후 대응되는 주소공간이 존재하면 그 주소공간의 값을 1만큼 증가시키고 대응되는 주소공간이 없으면 새로운 주소공간을 생성하고 초기값을 1로 한다.(표 4)

Apriori알고리듬에서는 1－항목모임들에 대한 지지도를 먼저 확인하고 그것들의 1－빈발항목모임들이 될수 있는 가능성여부를 확인한다. 다시 1－빈발항목모임들로부터 후보항목모임들을 생성하는 알고리듬에 의해서 2－후보항목모임들이 생성된다. 이러한 순차적인 반복과정이 대부분의 기존알고리듬들에서 적용되지만 이 론문에서 제기하는 알고리듬에서는 그러한 반복과정을 거치지 않고 자료기지를 한번 조사한 후에 모든 부분항목모임들의 지지도가 계산되기때문에 매 단계마다 확인할 필요가 없게 된다.

련관규칙탐색에서 1－항목모임은 다음 단계의 후보를 생성하기 위해서는 필요하지만 항목들의 련관성을 찾는 탐색에서 1－빈발항목모임은 의미가 없다.

표 2에서 보는바와 같이 분류된 매 항목모임들에 대하여 해쉬함수를 리용하여 생성된 매 주소공간에 그것들의 지지도가 보관된다. 자료기지를 이루는 매 트랜잭션에서 발생가능한 모든 부분항목모임들은 해쉬함수에 의해 유일한 주소를 가지게 되며 그 주소공간에서는 동일한 부분항목모임들이 존재할 때마다 값이 1씩 증가하게 된다.

다음으로 매 주소공간에 보관되여있는 값과 최소지지도를 비교하여 최소지지도이상의 값을 가지고있는 주소공간들을 추출하고 그것들의 매 주소에 해당되는 부분항목모임을 대응시키면 결국 우리가 바라는 빈발항목모임들을 찾게 된다.(표 12)

표 12. 빈발항목모임들

빈발항목모임	AC	BC	BD	CD	BCD
지지도	4	7	3	3	3
주소	13	23	24	34	234

결과적으로 단 한번의 자료기지조사로 모든 빈발항목모임들을 찾을수 있다. 물론 트랜잭션의 길이가 길고 후보항목들의 개수가 많은 대용량자료기지에서는 많은 후보항목모임들을 대상하기때문에 기억공간문제를 고려해야 한다. 그러나 응용분야에 따라 트랜잭션의 길이가 많이 차이난다. 더우기 SOT알고리듬에서 빈발항목모임들을 생성할 때마다 항목들의 개수가 줄어들고 따라서 트랜잭션의 길이가 작아지므로 론문에서 제기한 알고리듬을 적용할수 있는 가능성이 생긴다.

2. 트랜잭션제거와 해쉬화수법의 결합

SOT알고리듬에서는 k의 값에 따라 순차적으로 빈발항목모임들을 발견해나가면서 k와 SOT값이 일치하는 트랜잭션들을 자료기지에서 제거한다.

표 1의 실례자료기지에 대하여 SOT알고리듬에서 빈발항목모임을 발견해나가는 과정은 그림과 같다.

D

TID	트랜잭션
T_1	ACG
T_2	BCG
T_3	ABC
T_4	BC
T_5	BCDE
T_6	BC
T_7	ABCDF
T_8	BCDF
T_9	A
T_{10}	AC

→

TID	트랜잭션	SOT
T_1	ACG	3
T_2	BCG	3
T_3	ABC	3
T_4	BC	2
T_5	BCDE	4
T_6	BC	2
T_7	ABCDF	5
T_8	BCDF	4
T_9	A	1
T_{10}	AC	2

C_1 →

Itemset	Support
A	5
B	7
C	9
D	3
E	1
F	2
G	2

L_1 →

Itemset	Support
A	5
B	7
C	9
D	3

D_1

TID	트랜잭션	SOT
T_1	AC	2
T_2	BC	2
T_3	ABC	3
T_4	BC	2
T_5	BCD	3
T_6	BC	2
T_7	ABCD	4
T_8	BCD	3
T_{10}	AC	2

C_2 →

Itemset	Support
AB	2
AC	4
AD	1
BC	7
BD	3
CD	3

L_2 →

Itemset	Support
AC	4
BC	7
BD	3
CD	3

D_2

TID	트랜잭션	SOT
T_3	ABC	3
T_5	BCD	3
T_7	ABCD	4
T_8	BCD	3

C_3 →

Itemset	Support
ABC	1
BCD	3

L_3 →

Itemset	Support
BCD	3

그림. SOT알고리듬의 실행과정

먼저 자료기지 D가 주어지면 SOT를 계산한다.

이때 첫 자료기지조사(10개 트랜잭션읽기)가 진행된다. 그다음 알고리듬을 리용하여 1－항목모임들의 후보모임 C_1을 얻기 위한 두번째 자료기지조사(10개 트랜잭션읽기)가 진행되며 매 1－항목모임의 지지도를 구하기 위한 7번의 자료기지조사($10\times7=70$개 트랜잭

션읽기)가 진행된다. 다음 최소지지도 3을 리용하여 1－빈발항목모임족 L_1이 얻어진다.

다음으로 D로부터 k=SOT를 만족시키는 트랜잭션의 제거와 비빈발1－항목부분모임들을 제거하고 자료기지 D_1을 얻기 위한 자료기지조사(10개 트랜잭션읽기)가 진행된다. 그리고 D_1로부터 SOT계산을 위한 자료기지조사(9개 트랜잭션읽기), 2－항목모임들의 후보모임 C_2의 지지도를 얻기 위한 자료기지조사($9\times6=54$개 트랜잭션읽기)가 진행된다. 다음 최소지지도 3을 리용하여 2－빈발항목모임족 L_2가 얻어진다. 마찬가지의 방법으로 D_1로부터 k=SOT를 만족시키는 트랜잭션들을 제거하고 자료기지 D_2를 얻기 위한 자료기지조사(9개 트랜잭션읽기)가 진행되며 D_2로부터 SOT계산을 위한 자료기지조사(4개 트랜잭션읽기), 3－항목모임들의 후보모임 C_3의 지지도를 얻기 위한 자료기지조사($4\times2=8$개 트랜잭션읽기)가 진행된다. 다음 최소지지도 3을 리용하여 3－빈발항목모임족 L_3이 얻어진다.

우의 반복과정을 모두 합치면 SOT알고리듬으로 빈발항목모임을 발견하는데 총 22번의 자료기지조사(184개 트랜잭션읽기), 그중 지지도계산을 위한 15번의 자료기지조사(132개 트랜잭션읽기)가 진행된다.

론문에서 제기하는 해쉬화수법을 SOT알고리듬에 결합하면 그림과 같은 3단계의 순차적과정에서 후보항목모임 C_1, C_2, C_3들의 지지도계산을 위한 자료기지조사를 따로 할 필요가 없으며 SOT계산을 위한 자료기지조사에서 직접 지지도계산까지 할수 있다.

비교결과는 표 13과 같다.

표 13. 비교결과

구분	SOT방법	해쉬화방법
빈발항목모임을 찾기 위한 자료기지조사회수 (트랜잭션개수)	22(184)	7(52)
그중 지지도계산을 위한 자료기지조사회수 (트랜잭션개수)	15(132)	0

우의 실례와 같이 실행되는 트랜잭션제거와 해쉬화수법이 결합된 빈발항목모임찾기 알고리듬은 다음과 같다.

입력자료 D: 자료기지, min_sup: 최소지지도

출력자료 L: D의 빈발항목모임족

Address = ϕ

```
for ∀t ∈ D {
    HL1 = 1－항목모임족(t);
    for(k = 2; HLk−1 ≠ φ; k++){
        HCk = apriori_gen(HLk−1);
        for ∀c ∈ HCk {
            Address(k, c) = Hash(c);
            if Address(k, c) ∉ Address{
                Address = Address ∪ Address(k, c);
                Value(k, c) = 1;}
            else
                Value(k, c)++;
        }
```

```
            HL_k = HC_k;
 If (k ≥ 2){
        delete_datavalue(D, HL_k, HL_{k-1});
        delete_datarow(D, HL_k);}
}
}
for ∀ Address(k, c) ∈ Address{
      L_k = { Hash−1(Address(k, c))|Value(k, c) ≥ min_sup};
}
return L = ⋃k L_k ;

          Procedure apriori_gen(L_{k-1}: (k−1)−빈발항목모임족)
for ∀ l_1 ∈ L_{k-1}{
      for ∀ l_2 ∈ L_{k-1}{
        if(l_1[1] = l_2[1]) ∧ (l_1[2] = l_2[2]) ∧ ⋯ ∧ (l_1[k−2] = l_2[k−2]) ∧ (l_1[k−1]<l_2[k−1]) then{
            c=l_1 ∞ l_2;
            for ∀ l_1 ∈ L_{k-1}{
                for ∀ c ∈ C_k{
                    if l_1 ⊆ c then
                        c.num++; }}}}}
C'k = {c ∈ C_k |c.num = k};
return C'_k;

          Procedure delete_datavalue (D: 자료기지; L_k: k−빈발항목모임족;
                                      L_{k-1}: (k−1)−빈발항목모임족)
for ∀ i ∈ L_{k-1} and i ∉ L_k{
      for ∀ t ∈ D{
          for ∀ datavalue ∈ t{
              if (datavalue = i)
                  update datavalue = null; }
      }
}

          Procedure delete_datarow (D: 자료기지; L_k: k−빈발항목모임족)
for ∀ t ∈ D{
      for ∀ datavalue ∈ t{
          if(datavalue! = null and datavalue! = 0){
              datarow.count++;}}
    if (datarow.count < k){
        delete datarow;}
}
```

맺 는 말

론문에서 제기한 해쉬화수법에 의한 빈발항목모임찾기방법은 단 한번의 자료기지조사로 지지도계산을 진행함으로써 트랜잭션안의 항목들의 개수를 고려하여 단계별로 불필요한 항목들과 트랜잭션들을 제거해나가는 SOT알고리듬의 우점을 더욱 살려주고있다.

또한 트랜잭션들의 길이가 짧거나 자료기지를 이루는 항목들의 개수가 비교적 적은 자료기지인 경우에는 빈발항목모임찾기에 직접 적용할수 있는 우월한 방법으로 된다. 이 방법은 최소지지도에 따라 성능이 많이 달라지는 Apriori방법과 비교해보아도 최소지지도의 영향을 거의나 받지 않으므로 우월한 방법으로 된다.

참 고 문 헌

[1] J. Singh et al.; International Journal of Scientific and Research Publications, 3, 1, 1, 2013.

[2] J. J. Jadav et al.; International Journal of Engineering Research and Applications, 2, 2, 1147, 2012.

[3] M. J. Zaki; IEEE Trans. Knowledge and Data Engineering, 12, 3, 372, 2000.

[4] M. Z. Ashrafi et al.; LNCS, 2660, 978, 2003.

주체104(2015)년 6월 5일 원고접수

Implementation of Association Rule Discovering for Network Intrusion Detection

Kong Hye Ok

We propose an improved algorithm for finding frequent-itemsets upon databases where lengths of transaction records are not relatively so long and the numbers of items are not relatively so many, which can find supports of frequent-itemsets by only one scan of database by applying hashing-method.

Key word: frequent-itemset