데이터 스트림에서 빈발항목 탐색을 위한 메모리 사용량 최적화

김민정[O], 장중혁, 이원석
연세대학교 컴퓨터 과학과
{guian[O], jhchang, leewo@amadeus.yonsei.ac.kr}

Memory Adaptation in Finding Frequent Itemsets over Data Streams

Minjung Kim[O],Junghyuk Chang, Wonsuk Lee
Dept. of Computer Science, Yonsei University

요 약

컴퓨팅 환경의 발달로 방대한 양의 정보들이 매우 빠른 속도로 생성되고 있다. 구성 요소가 지속적으로 발생되는 무한 집합으로 정의되는 데이터 스트림에 대한 마이닝 방법은 이들 정보로부터 중요한 지식을 효과적으로 얻을 수 있는 방법으로 최근 들어 다양한 방법들이 활발히 제안되고 있다. 이러한 마이닝 방법에서는 지속적으로 확장되는 데이터 스트림의 특성으로 수행과정에서 메모리 사용량을 가용 범위 내로 제한하는 것이 중요한 고려 사항이 되고 있다. 본 논문에서는 데이터 스트림에서 빈발 항목을 탐색하는데 있어서 가용 메모리 범위에서 최적의 메모리를 사용하여 최상의 마이닝 결과를 얻을 수 있도록 하는 메모리 사용량 최적화 방법을 제시한다.

1. 서 론

인터넷의 확산과 네트워크 기술의 발달로 정보의 양은 매우 빠른 시간에 무한히 증가되고 있다. 이런 환경에서 데이터 마이닝은 생성된 많은 정보들 중 가치 있는 지식을 얻기 위하여 다양한 분야에서 널리 이용되고 있다. 온라인 분석 작업, 고객 관계 관리 및 비정상침입 탐지 시스템 등의 분야에서는 지속적으로 발생되는 트랜젝션들에 대한 자동적인 분석을 지원하고 분석 결과를 빠른 시간 내에 얻을 수 있기를 기대한다. 이런 환경을 데이터 스트림(data stream) 환경이라고 한다. 특히, 분석 결과가 즉시 활용 되어야 하는 이러한 시스템에서는 대량의 데이터에 대한 실시간 분석이 필수적이다. 이런 분야에 이전의 마이닝 방법들[1,2,3]을 적용하는데 한계가 있다. 이전의 마이닝 방법들은 마이닝 대상이 사전에 명확히 정의되어 있고 고정되어 있어야 한다. 그리고 두 번 이상의 탐색 과정이 필요하므로 수행시간이 매우 오래 걸리며 메모리 사용량이 매우 크다. 이런 단점 때문에 이전의 데이터 마이닝 방법들을 실시간 처리를 요하는 데이터 스트림 환경에 적용시키기 어렵다.

데이터 스트림은 지속적으로 발생되는 트랜젝션들로 구성되는 무한집합으로 정의된다. 따라서 데이터 스트림의 모든 트랜젝션을 별도로 메모리상에 저장하는 것은 불가능하다. 최근 들어 데이터 스트림에 대한 마이닝 방법들[4,5]이 제안되었다. 이 마이닝 방법들은 각 트랜젝션의 정보를 오직 한번 읽고 마이닝 결과를 생성한다. 이러한 이유로 이들 마이닝 방법들은 마이닝 결과에 다소의 오차를 포함한다. Lossy Counting[4]알고리즘은 최소 지지도와 최대 허용 오차 조건이 주어졌을 때 데이터 스트림에서 발생한 빈발 항목들의 집합을 찾는다. 각 트랜젝션에서 발생하는 빈발 가능한 항목들의 출현 빈도수와 오차를 메모리에서 관리하여 새로 발생한 트랜젝션들은 메인 메모리에 유지되는 고정된 크기의 버퍼에 채워지고 동시에 처리된다. 이 알고리즘에서는 버퍼의 크기가 크면 더 많은 트랜젝션을 동시에 처리할 수 있으나 최신의 마이닝 결과를 임의 시점에서 얻고자 하는 온라인 데이타 스트림에는 한계가 있다.

estAcc[6]방법은 지연추가와 항목 전지작업을 통해 빈발 항목이 될 가능성이 있는 항목들만 메모리 상에 관리한다. 이 방법에서는 사전 정의된 지연 추가 임계값과 항목 전지 작업 임계값에 따라 오차와 메모리 사용량이 달라진다. 이 값들을 크게 설정 하였을 경우 더 큰 오차를 허용하여 마이닝 결과의 전체 오차가 커지는 반면 메모리 사용량은 크게 줄일 수 있다. 그리고 반대로 작게 설정 하였을 경우 마이닝 결과의 오차는 적어지나 메모리 사용량이 매우 크다. 그러나 이 방법은 고정된 임계값들을 기반으로 하기 때문에 마이닝 수행 중에 대상이 되는 데이터 집합에서 빈발이 될 가능성이 높은 항목들이 매우 증가될 수 있다.

데이터 스트림 환경에서는 마이닝을 수행할 때 동적으로 변하는 데이터 집합에 대해 고정된 메모리크기에 맞게 최적화된 마이닝 결과를 얻을 수 있어야 한다. 본 논문에서는 데이터 스트림에서 빈발항목을 탐색하는 estAcc[6] 방법에서 가용 메모리 범위 내에서 최적의 메모리를 사용하여 최상의 마이닝 결과를 얻을 수 있도록 하는 메모리 사용량 최적화 방법을 제시한다.

2. 데이터 스트림에서 빈발항목 탐색

2.1 지연추가와 항목 전지 작업

데이터 스트림 환경에서 발생된 모든 트랜잭션들을 저장하는 것은 불가능하다. 따라서 새로운 트랜잭션이 현재 데이터 집합에 추가되었을 때 해당 트랜잭션의 정보들을 점진적으로 수집해야 한다. 빈발항목 탐색 과정에서 각 트랜잭션에 출현하는 단위 항목들의 조합으로 표현되는 서로 다른 항목들을 효과적으로 표현하기 위해 본 논문에서는 전위-트리(prefix-tree)래티스 구조[2]를 활용한다. 이러한 전위트리 래티스 구조를 이용한 마이닝 방법 중 단순 축적 방법(Acc방법)[6]에 의한 방법은 데이터 스트림을 실시간으로 마이닝 하기 위해 추가되는 전체 데이터 집합의 모든 트랜잭션 정보를 메모리에 관리해야 한다. 그러나 단순축적에 의해 생성된 모든 항목들이 빈발 항목 탐색을 위해 중요한 항목은 아니다. 따라서 이들 모든 항목이 메모리에서 관리될 필요는 없다.

estAcc[6]방법은 메모리상에서 관리되는 레티스에 새 항목을 추가할 때 그 항목이 가까운 미래에 빈발 항목이 될 가능성이 있을 때까지 지연시킨다. 이 방법은 새로 들어온 항목에 대하여 다음의 두 가지 경우에만 메모리상의 래티스에 추가한다. 새로 생성된 항목 e에 대해서 첫 번째 경우는 새로 추가된 트랜잭션에 새로운 1-항목이 출현한 경우이다. 이 경우에는 새로운 항목에 대하여 별도의 추정과정 없이 래티스에 추가된다. 두 번째 경우는 n-항목(n:항목 e의 길이 , n≥2)의 지지도 추정값이 해당 항목의 출현 빈도수를 관리해야 할 만큼 충분히 큰 경우이다. 새로운 트랜잭션에 출현한 항목 e가 메모리상의 래티스에서 관리되고 있지 않을 때 해당 항목의 출현 빈도수는 e의 부분 항목들로 추정한다. n-항목인 e의 (n-1)-부분 항목이 모두 래티스에서 관리되고 그 출현 빈도수가 모두 임계값 이상일 때 래티스에 추가된다. 사전에 정의된 추가를 위한 기준이 되는 임계값을 지연 추가 임계값(S_{ins})라 한다. 지연 추가에 의해 래티스에 들어간 항목이라도 이후 발생되는 데이터 스트림에 의해 출현 빈도수가 사전 정의된 출현 빈도수 임계값 이하가 될 수 있다. 이런 항목들을 메모리 상에서 관리하면 무한히 증가하는 데이터 집합으로 인해 메모리상의 래티스 크기는 매우 커질 수 밖에 없다. 따라서 래티스에서 관리하고 있는 항목 중 빈발 항목이 될 가능성이 적은 항목들을 래티스에서 제거한다. 이런 작업을 항목 전지 작업 임계값(S_{prn})이라 하며 S_{ins}와 S_{prn}은 사전 정의된 최소 지지도의 상대적인 값으로 정의한다. 이 두 방법에 의해 메모리 상에서 관리되는 래티스의 크기를 크게 줄일 수 있다. 그러나 이 두 값을 사전 정의하여 모든 시점에서 똑같이 적용 시킨다면 만약 어느 시점에서 빈발 가능 항목이 한정된 메모리 크기를 초과하여 생성 될 때 마이닝 작업 자체를 수행할 수 없다.

2.2 동적 메모리 관리

앞 절에서 살펴본 S_{ins}와 S_{prn}두 값을 함께 S_{ip}라 한다. S_{ip}의 값에 따라 메모리상의 레티스의 크기가 결정된다. S_{ip}가 크면 지연 임계값이 커지므로 상대적으로 적은 항목이 레티스에 추가되고 항목 전지 임계값도 커지므로 많은 항목들이 전지된다. 그러나 S_{ip}가 너무 큰 값을 가지면 메모리 상의 래티스에서 관리되는 항목들의 수가 적으므로 지연 추가를 위한 더 많은 출현 빈도수 추정과정을 거치므로 더 큰 오차를 가지게 된다. 반대로 S_{ip}가 작은 값이면 지연 추가 임계값이 작으므로 더 많은 항목이 레티스에 추가되며 상대적으로 적은 항목이 래티스에서 전지된다. 따라서 추정되는 항목이 적어지므로 전체 오차가 작아진다. 이런 이유로 메모리의 크기와 허용 오차 사이의 적절한 S_{ip}가 결정 되어야 한다. 데이터 스트림 환경에서 데이터 집합은 비 한정적이며 유동적이다. 따라서 어느 시점에서 빈발 가능 항목이 갑자기 많이 생성되거나 적어 질 수 있다. 메모리는 한정되어 있으므로 메모리 크기와 메모리에서 관리 할 수 있는 래티스 크기 사이에 적절한 S_{ip}를 각 시점마다 결정할 수 있다면 효과적으로 메모리 상의 래티스 크기를 조절 할 수 있다.

현재 시점에서 들어온 트랜잭션 $T_k = \{ i_1, i_2, \ldots , i_n\}$으로 인해 새로운 노드를 메모리상의 래티스에 삽입 할 때 현재 설정된 S_{ip}를 old_S_{ip}, 새로 갱신될 S_{ip}를 new_S_{ip}, 현재 래티스를 구성하고 있는 노드수를 C_nodes, 제한한 최대 노드수를 L_nodes라 할 때 마이닝 수행 중에 S_{ip}를 다음과 같이 고려한다.

$$\begin{cases} new\_S_{ip} = old\_S_{ip} + \alpha \text{ , when } C\_nodes > L\_nodes \\ new\_S_{ip} = old\_S_{ip} - \alpha \text{ , when } C\_nodes < L\_nodes \end{cases}$$

S_{ip}는 최소 지지도 임계값과 상대적인 값을 가지므로 비율(%)로 정의된다. 따라서 S_{ip}가 변화 될 수 있는 값의 범위는 0%에서 100%까지 이므로 α 는 0 이상의 변화된 비율을 의미하게 된다. 그리고 L_nodes값을 설정할 때 이 값을 메모리상에 생성될 수 있는 최대 노드수로 잡으면 그 이상의 노드들에 대해서는 생성 자체가 불가능 하므로

```
Input : A data stream D
Output : The set of frequent itemsets
ML : The lattice in the memory

1: for each new transaction in D
2: read current transaction & increase the number of transactions
3:  for all itemset e
4:    if (e∈ML) then
5:       update frequency of e ;
6:       if (frequency of e < S_prn) then
7:          eliminate e and child_nodes from ML ;
8:    else // e∉ML
9:         if ( |e|=1 ) then  insert e into ML;
10:        else
11:          estimate frequency of e ;
12:          if ( estimated_frequncy > S_ins )  then
13:               insert e into ML ;
14:               if ( C_nodes > L_nodes )  then S_ip = S_ip + α ;
15:               else  S_ip = S_ip - α ;
16:               end
17:        end
18:   end
19: end
20: Find frequent itemsets;
21:end
```

그림 1. 데이터 스트림에서의 빈발항목 탐색 알고리즘

최대 노드수 보다 작은 값으로 잡으면 임계값 이상의 노드도 생성 되면서 S_{ip}를 조절 할 수 있으므로 효과적이다.

2.3 데이터 스트림에서 빈발 항목 찾기

본 절은 데이터 스트림 환경에서 빈발 항목을 찾기 위한 방법을 제안한다. 전체적인 알고리즘은 그림 1과 같다. 이 방법은 매개변수 갱신 단계(1-2번째 줄)와 출현 빈도수 갱신단계(4-5), 지연 추가 단계(8-19), 빈발항목 선택 단계(20)로 구성되며 빈도수 갱신 단계에선 항목 전지작업(6-7)을 포함한다. 노드를 메모리 상에 만들 때 제한된 메모리 크기에 맞게 효율적으로 빈발 가능한 항목을 저장하기 위해 S_{ip} 조절하는 부분이(14-15)이 추가된다. 지연 추가 과정을 거쳐 새로운 항목이 래티스에 추가 될 때 제한된 노드 수와 비교하여 S_{ip}를 늘리거나 줄인다.

3. 실험 및 결과 분석

앞 장에서 소개한 알고리즘의 성능을 검증하기 위해 [1]에서와 같은 방법으로 서로 다른 단위 항목의 개수가 1000개인 T5.I4.D1000K 데이터 집합을 생성하였다. 이 실험에서 데이터 집합의 각 트랜젝션은 데이터 스트림 환경에서처럼 하나씩 처리된다. S_{ins}와 S_{prn}은 S_{ip}로 동일한 값을 가지도록 하고 S_{ip}가 p%로 표현 되었을 때 두 임계값 S_{ins}및 S_{prn}의 실제값은 사전 정의된 최소 지지도(S_{min}) × (p/100)으로 지정된다.

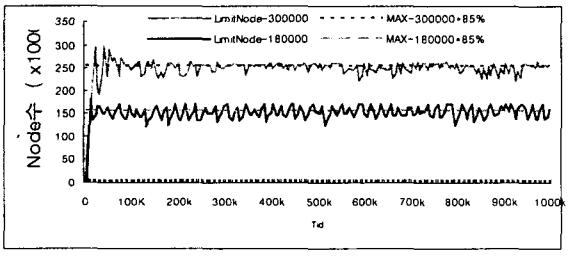

그림 2. 노드수 변화

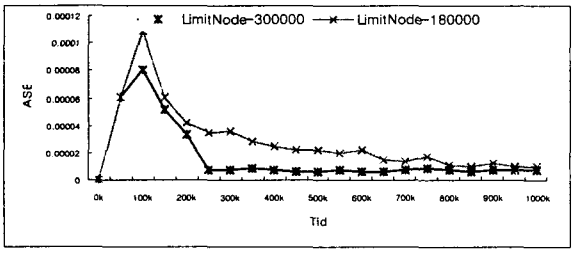

그림 3. 평균 지지도 오차 (ASE) 변화

이 실험에서 생성 될 수 있는 최대 노드수를 각각 300000개와 180000개로 제한하고 제한된 메모리를 최대한 이용할 수 있도록 S_{ip}는 85%선 즉 255000개와 153000개에서 조절하도록 하였다. 그리고 최대 노드수 이상으로 생성되는 노드들에 대해서는 생성하지 않고 S_{ip}는 1%씩 조절된다. 그림 2는 트랜젝션이 들어와 새롭게 노드가 생성 될 때 제한된 노드의 수와 현재 메모리 상에서 관리되는 노드수를 고려하여 S_{ip}를 늘리거나 줄임으로써 제한된 메모리 크기에 맞게 래티스의 크기를 관리하는 것을 보여주고 있다. 그림 3은 각 시점에서 노드수를 조절 할 때 평균 지지도 오차(ASE: Average Support Error)[7]가 변화되는 것을 보여준다.

위 실험에서 제한된 메모리의 크기에 맞게 S_{ip}를 조절함으로써 래티스의 노드수를 효과적으로 조절 할 수 있다.

4. 결론

데이터 스트림 환경에서 데이터 집합은 사전 정의되지 있지 않고 마이닝을 수행하는 중에 동적으로 변화된다. 따라서 이런 데이터 스트림에서 빈발 항목을 탐색하기 위해 본 논문에서는 제한된 메모리에서 마이닝 수행과정 중에 동적으로 메모리 사용량을 조절하는 것을 실험을 통해 보였다. 결과적으로 이 방법은 무한하게 증가되는 데이터와 메모리와 같은 제한된 컴퓨팅 자원 사이에서 가장 최적화 된 결과를 얻을 수 있도록 지원한다. 본 논문에서 제안한 방법은 데이터 집합의 크기에 상관없이 한정된 메모리의 크기에 맞게 마이닝 수행과정 중 메모리 사용량을 조절함으로써 데이터 스트림 환경에서 빈발 항목을 효율적으로 탐색 할 수 있는 방법을 제시하였다.

5. 참고 문헌

[1] R. Agrawal, and R. Srikant. Fast algorithms for mining association rules. In Proc. of the 20th Int'l *Conference on Very Large Databases*, Santiago, Chile, Sept. 1994.

[2] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for *market basket data*. In Proc. of the ACM SIGMOD Int'l Conference on Management of Data, pages 255-264, Tucson, AZ, May 1997.

[3] A. Savasers, E. Omiecinski, and S. Navathe. An *efficient algorithm for mining* association rules in large databases. In Proc. of the 21st Int'l Conference on Very Large Databases, pages 432-444, Zurich, Switzerland, Sept, 1995.

[4] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In Proc. of the 28th Int'l Conference on Very Large Databases, Hong Kong, China, Aug. 2002.

[5] M. Charikar, K. Chen and M. Farach-Colton. Finding Frequent Items In Data Streams. In Proc. of the 29th Int'l Colloq. on Automata, Language and Programming, 2002

[6] 장중혁, 이원석. 데이터 스트림에서 개방 데이터 마이닝 기반의 빈발 항목 탐색. 정보처리학회 논문지, 10-D권 3호, pp.447-458,2003년 6월

[7] J. H. Chang and W. S. Lee. Finding recent frequent itemsets adaptively over online data streams. In Proc. Of the 9th ACM SIGKDD Int' l. Conf. on Knowledge Discovery and Data Mining, Washington, DC, Aug. 2003