한국산학기술학회논문지
Vol. 10, No. 8, pp. 1998-2004, 2009

데이터 스트림에서 가중치 지지도 기반 빈발 패턴 추출 방법

김영희[1*], 김원영[1], 김응모[1]
[1]성균관대학교 정보통신공학부

An Efficient Method for Mining Frequent Patterns based on Weighted Support over Data Streams

Young-Hee Kim[1*], Won-Young Kim[1] and Ung-Mo Kim[1]
[1]**School of Information and Communication Engineering, Sungkyunkwan University**

요 약 다양한 저장 장치의 발달과 네트워크의 발전은 대용량의 데이터를 연속적으로 빠르게 생성한다. 데이터 스트림에서의 데이터 마이닝은 처리 시간 및 메모리 사용에 제한적이다. 또한 생성된 데이터를 한 번의 스캔으로 유용한 패턴을 발견할 수 있어야 하고 정보 변화 가능성이 큰 데이터 속성을 갖는 경우 최근의 정보를 반영한 빠른 분석이 가능해야 한다. 기존의 지지도 기반 마이닝 방법들은 일정 기간 동안 미리 정의된 지지도 이상의 빈발 항목에 대하여만 고려하므로 중요도가 높은 항목들을 간과하는 문제점을 가지고 있다. 본 논문에서는 시간의 변화에 따른 가변성을 고려하여 가중치 지지도를 갖는 데이터 항목들에 대하여 보다 의미 있는 정보를 제공하기 위한 효율적인 빈발 패턴 추출 방법을 제안하고자 한다. 제안된 WSFI-Mine(Weighted Support Frequent Itemsets Mine) 방법은 DCT(Data Stream Closed Pattern Tree) 데이터 구조를 이용하여 패쇄 빈발 항목을 탐사한다. 제안된 알고리즘은 DSM-FI와 THUI-Mine 알고리즘과 지지도 변화에 따른 성능을 비교하였고 그 결과 비교 알고리즘 보다 수행 시간이 우수함을 보였고, 빈발 항목을 생성하는 후보 항목의 수를 줄이므로 메모리 사용량을 효율적으로 사용할 수 있음을 보였다.

Abstract Recently, due to technical developments of various storage devices and networks, the amount of data increases rapidly. The large volume of data streams poses unique space and time constraints on the data mining process. The continuous characteristic of streaming data necessitates the use of algorithms that require only one scan over the stream for knowledge discovery. Most of the researches based on the support are concerned with the frequent itemsets, but ignore the infrequent itemsets even if it is crucial. In this paper, we propose an efficient method WSFI-Mine(Weighted Support Frequent Itemsets Mine) to mine all frequent itemsets by one scan from the data stream. This method can discover the closed frequent itemsets using DCT(Data Stream Closed Pattern Tree). We compare the performance of our algorithm with DSM-FI and THUI-Mine, under different minimum supports. As results show that WSFI-Mine not only run significant faster, but also consume less memory.

Key Words : Data Stream, Weighted Support, WSFI-Mine, DCT, Closed Frequent Itemsets

1. 서론

최근 들어, 유비쿼터스 환경에서의 센서 네트워크, 네트워크 침입 탐지, 웹 사용에 따른 클릭 스트림, 네트워크 모니터링에서 성능 및 트래픽 관리 등의 다양한 응용은 데이터 마이닝 분야에서 데이터 스트림 분석의 요구가 더욱 커지고 있다[1,3]. 데이터 스트림 마이닝 과정에서 기본적인 문제는 빈발 항목(Frequent Itemsets)을 찾는 과정으로 기존의 정적인 트랜잭션들은 여러 번의 탐색으로 미리 정의된 특정 임계치(Minimum Support Threshold) 보다 높은 지지도(Support)를 갖는 모든 빈발 항목을 찾는다[2]. 이로 인해 메모리의 사용량이 많고 처리 시간이

본 논문은 2009년도 정부(교육과학기술부)의 재원으로 한국과학재단(No.2009-0075771)의 지원을 받아 수행된 연구임.
*교신저자 : 김영희(younghees@gmail.com)
접수일 09년 06월 19일 수정일 (1차 09년 07월 14일, 2차 09년 08월 06일, 3차 09년 08월 14일) 게재확정일 09년 08월 19일

비효율적인 문제를 갖는다. 반면, 데이터 스트림에서는 시간이 경과함에 따라 빈발 항목의 상태(Status)가 변하고 이전에 생성된 데이터 항목을 재 스캔 할 수 없는 특징을 가진다. 다시 말해 이전에는 비 빈발 항목(Infrequent Itemsets)이었으나 현재 시점에서 빈발 항목이 될 수 있는 특성을 포함한다. 따라서 데이터 스트림 마이닝 과정에서는 빈발 항목 집합을 적절히 유지하고 관리할 수 있는 방법이 필요하다. 본 논문에서는 이러한 문제를 개선하기 위해 제한된 가용 메모리상에서 가중치 지지도 패쇄 빈발 항목들을 효율적으로 저장하고 유지할 수 있는 DCT 데이터 구조를 제안한다. 또한 DCT 트리를 이용하여 한 번의 스캔으로 마이닝을 수행하는 WSFI-Mine 알고리즘을 제안한다. 제안된 DCT 트리는 마이닝 과정 동안 모든 후보 항목들에 대한 정확한 지지도 값을 유지할 수 있고 새로운 데이터 스트림의 갱신 작업을 수행 시 변화된 항목의 정보를 효율적으로 유지할 수 있는 장점을 가진다. 본 논문의 구성은 다음과 같다. 제 2장에서는 기존의 데이터 스트림 마이닝 기법과 관련된 연구를 살펴보고 제 3장에서는 데이터 스트림의 문제 정의와 본 논문에서 제안한 DCT 트리 구조 및 저장 기법을 살펴보고, WSFI-Mine알고리즘을 제안한다. 제 4장에서 수행 결과 및 분석을 살펴보고 마지막으로 제 5장에서 결론 및 향후 과제를 기술한다.

2. 관련연구

데이터 스트림에서 빈발 항목 마이닝을 위해 제안된 기존 연구는 다음과 같다. Manku and Motwani는 Apriori 기반 단일 패스 알고리즘인 Lossy Counting 기법을 제안하였다[3]. 이 방법은 메모리 사용량을 일정 범위로 한정하기 위해 메모리 상에 버퍼를 이용하여 일괄 연산을 수행하므로 효율을 높인다. 그러나 빈발 항목 탐색시 필요한 메모리 사용 공간이 증가하고 갱신시 효율적이지 않은 단점이 있다. Chang and Lee는 온라인 데이터 스트림을 대상으로 빈발항목을 탐색하면서 출현 빈도수 관리 대상 항목 집합을 메모리 상에서 관리할 수 있도록 하는 estDec 방법을 제안하였다[1]. estDec 방법에서는 데이터 스트림에서 발생한 항목들에 대하여 사전 정의된 지연 추가 및 전지 임계값 이상의 지지도를 갖는 항목들을 앞으로 빈발 항목이 될 집합으로 간주하고 메모리상에서 관리한다. 이를 통해 온라인 데이터 스트림에 대한 빈발 항목 탐색 과정시 메모리 사용량은 효과적으로 감소하지만 데이터의 변화가 큰 항목집합에 대하여는 메모리 사용량 감소에 한계가 있다. Li et al.는 전위 트리(Prefix Tree) 기반 단일 패스 알고리즘으로 DSM-FI(Data Stream Mining for Frequent Itemsets)와 DSM-MFI (stands for Data Stream Mining for Maximal Frequent Itemsets)를 제안하였다[4,5]. 이들 알고리즘은 히스토리 데이터 스트림 전체에 대하여 모든 빈발 항목을 찾고, 최대 빈발 패턴을 추출하는 알고리즘이다. 윈도우 슬라이딩 기반(Window Sliding Based)의 빈발 항목 마이닝 알고리즘으로 MFI-TransSW(Mining Frequent Itemsets within a Transaction Sensitive Sliding Window)와 MFI-TimeSW(Mining Frequent Itemsets within a Time Sensitive Sliding Window)를 Li와 Lee는 제안하였다[6]. MFI-TransSW는 트랜잭션의 수를 고정하고 항목을 비트 시퀀스(Bit Sequence)로 표현하여 항목의 빈도수를 효율적으로 유지하므로 높은 정확도를 갖는 마이닝 결과를 주고 빠른 실행 속도와 메모리의 효율성을 높였다. 그리고 MFI-TimeSW는 Time Unit을 고정하여 데이터 스트림 마이닝을 수행하는 알고리즘이다. 윈도우 슬라이딩 기반 데이터 스트림 알고리즘에는 이외에도 SWF(Sliding Window Filtering), Moment 알고리즘이 대표적이다[7,8]. Chu et al.는 유틸리티 마이닝의 기법으로 THUI(Temporal High Utility Itemsets)-Mine 방법을 제안하였다. THUI-Mine은 개별 항목에 유틸리티(Utility)를 적용하여 트랜잭션에 가중치를 고려하여 마이닝 하는 방법이다[9,10]. 본 논문에서 기존의 데이터 스트림 마이닝 기법들 중에서 전위 트리 기반 DSM-FI 알고리즘과 트랜잭션에 가중치를 고려한 THUI-Mine 알고리즘과의 성능을 비교하고 분석하고자 한다.

3. 가중치 지지도 기반 탐색 기법

각 항목에 가중치를 부여하므로 의미 있는 빈발 항목 탐색 시 간과되는 문제들을 최소화하고 데이터 스트림에서 빈발 횟수에 대한 정보 손실을 최소화 하기 위한 가중치 지지도를 기반으로 한 탐색 기법을 살펴보고자 한다.

3.1 가중치 지지도 기반 빈발 항목

데이터 스트림 항목들의 집합 $I = \{i_1, i_2, ..., i_m\}$과, 트랜잭션의 집합을 $D = \{T_1, T_2, ..., T_n\}$라 할 때, 각 트랜잭션 $T_i \in D$는 I의 부분 집합(Subset)으로 정의된다. 각 트랜잭션은 공집합이 아닌 집합(Non Empty Set)이고, 유일한 트랜잭션 TID를 갖는다. 집합 $X(X \subseteq I)$를 항목집합(Itemsets)이라 하고, k개의 항목

을 갖는 항목집합을 k-항목집합 (k-Itemsets)이라 한다. 트랜잭션에서 항목 X를 포함하는 트랜잭션의 수를 항목 X의 지지도라고 하고, 가중치를 갖는 항목 X의 가중치 지지도 $ws(X)$에 대하여 $ws(X) = (X_{support} \times weight)$로 나타내고, 트랜잭션 데이터베이스에서 각 항목에 중요도를 반영하여 가중치(Weight Value)를 부여한다. 가중치의 범위는 $w_{min}(X) \le w(X) \le w_{max}(X)$로 주어진다. 최소 가중치 지지도와 최대 가중치 지지도는 각 각 다음과 같다.

$$ws_{min}(X) = (X_{support} \times w_{min}(X))$$
$$ws_{max}(X) = (X_{support} \times w_{max}(X)) \quad (1)$$

또한, 가중치를 갖는 의미 있는 빈발 항목을 얻기 위해 사용자 정의 최소 가중치 지지도 임계치(User Defined Minimum Weighted Support Threshold) σ와 최소 가중치 지지도 에러 임계치 (Minimum Weighted Support Error Threshold) ϵ에 대해 다음과 같이 정의한다.

$$\epsilon \le ws(X) \le \sigma,\ \epsilon [0, \sigma] \quad (2)$$

각 각의 임계치 ϵ와 σ는 수식 (3)과 같이 구한다.

$$\sigma = [MAX(ws_{\min}(X)) + MIN(ws_{\max}(X))]/2$$
$$\epsilon = [MIN(ws_{\min}(X)) + MIN(ws_{\max}(X))]/2 \quad (3)$$

3.2 윈도우 슬라이딩 기반 가중치 패턴 유형

최근 N개의 데이터 스트림 트랜잭션에서 빈발 항목을 찾기 위한 윈도우 슬라이딩 기법을 살펴보자. 표 1은 항목 집합 $\sum = \{a, b, c, d, e\}$를 갖는 데이터 스트림 트랜잭션을 나타내고, 표 2는 각 항목에 대한 가중치를 나타낸다. 윈도우 크기를 N = 4로 하고, 항목 X의 가중치 범위를 $0.2 \le w(X) \le 0.8$이라 할 때 가중치를 갖는 항목

집합에 대한 패턴의 유형은 아래의 정의와 같이 각 항목별 최대, 최소 가중치 지지도를 이용하여 분류한다.

[표 1] 실험 데이터 집합

W_1		W_2		W_3	
Tid	Itemsets	Tid	Itemsets	Tid	Itemsets
1	{a,c,d}	2	{b,c}	3	{a,b,c,e}
2	{b,c}	3	{a,b,c,e}	4	{a,c,d}
3	{a,b,c,e}	4	{a,c,d}	5	{a,b,c,d}
4	{a,c,d}	5	{a,b,c,d}	6	{b,c}

[표 2] 항목의 가중치 [0.2≤w(x)≤0.8]

Item	a	b	c	d	e
Weight	0.3	0.2	0.2	0.8	0.4

정의 2.1 가중치 지지도($ws(X, W_i)$)

현재 윈도우 슬라이드(Current Window Slide) W_i에서 항목집합 X에 가중치를 부여한 값으로 $ws(X, W_i) = (X_{support} \times weight)$ 이고, 이때 가중치의 범위는 $w_{min}(X) \le w(X) \le w_{max}(X)$을 만족한다.

정의 2.2 최소 가중치 지지도($ws_{min}(X,W_i)$)

W_i에서 항목집합 X의 지지도에 최소 가중치 $w_{min}(X)$를 곱한 값, $ws_{min}(X) = (X_{support} \times w_{min}(X)$를 나타낸다.

정의 2.3 최대 가중치 지지도($ws_{max}(X,W_i)$)

W_i에서 항목집합 X의 지지도에 최대 가중치 $w_{max}(X)$를 곱한 값, $ws_{max}(X) = (X_{support} \times w_{max}(X))$를 나타낸다.

정의 2.4 가중치 지지도 빈발 항목(WSFI)

WSFI는 $ws(X, W_i)$가 최소 가중치 지지도 보다 크고, 최대 가중치 지지도 보다 작은 조건을 만족하며 사용자 정의 최소 가중치 지지도 이상의 빈발 항목을 나타낸다.

예를 들어 표 1의 데이터 집합 W_1에서 항목 a의 지지도는 3이고 a의 가중치 값은 0.3이다. 이때 $ws(a, W_1)$은 0.9 = 3 × 0.3 가 된다. 정의 2.2와 정의 2.3에서 각 각 $(ws_{min}(a,W_1))$은 0.6 = 3×0.2 이고 $(ws_{max}(a,W_1))$은 2.4 = 3×0.8이므로 항목 a의 가중치 지지도는 0.6 ≤0.9 ≤2.4 를 만족한다.

연속적으로 생성되는 데이터 스트림은 시간과 공간의 제약으로 인해 모든 빈발 항목에 대하여 정확한 지지도를 유지하는 것은 매우 어려운 문제이다. 뿐만 아니라 현재 시점에서 비빈발 항목이 앞으로 빈발 항목으로 될 잠재된 가능성이 있음에도 불구하고 유용한 정보로 선택되지 않는 문제가 발생될 수 있다. 따라서, 빈발 항목을 추출하는 과정에서 각 데이터 항목에 대하여 사용자 정의 최소 가중치 지지도 임계치 σ와 최소 가중치 지지도 에러 임계치 ϵ를 고려하여 빈발 항목, 잠재적 빈발 항목, 비빈발 항목으로 분류한다. 본 논문에서 제안된 빈발 항목의 유형은 다음과 같이 정의된다.

빈발 항목 항목 X의 가중치 지지도 $ws(X)$의 값이 σ 보다 크면 빈발 항목이라 한다. σ는 각 각의 항목들의 최

소 가중치 지지도 내에서 가장 큰 값과 최대 가중치 지지도 내의 가장 작은 값의 평균값으로 나타낸다.

잠재적 빈발 항목 항목 X의 가중치 지지도 $ws(X)$의 값이 최소 가중치 지지도 에러 임계치 ϵ보다 크거나 같고, 사용자 정의 최소 가중치 지지도 임계치 σ보다 작으면 잠재적 빈발 항목으로 나타낸다.

비빈발 항목 항목 X의 가중치 지지도 $ws(X)$의 값이 최소 가중치 지지도 에러 임계치 ϵ보다 작은 항목을 의미한다.

3.3 DCT 저장 구조

전위 트리 구조는 트랜잭션에서 발생한 개별 항목을 각각 노드로 관리하므로 항목 집합의 수가 증가 될수록 트리의 크기가 커지며 최대 빈발 항목을 찾을 때 유용하지만 마이닝 수행과정에서 메모리 사용량을 증가시키는 단점을 가진다. 반면 DCT 트리는 확장된 전위 트리(Extended Prefix Tree)로 빈발 항목들을 탐색하기 위해 폐쇄 빈발 항목(Closed Frequent Itemsets)들을 효율적으로 유지하고 관리하는 새로운 기법의 데이터 구조이다. 본 장에서는 DCT 트리의 생성과 패쇄 빈발 항목을 탐색하는 과정을 살펴본다.

3.3.1 사용자 정의 가중치 지지도

그림 1은 3.2절에서 정의된 각 개별 항목들의 최소, 최대 가중치 지지도를 나타낸다.

W_1-list	$sup(X,W_1)$	$ws_{min}(X) \le ws(X) \le ws_{max}(X)$	$ws(X)$
a	[3:0.3]	$0.6 \le ws(a) \le 2.4$	0.9
b	[2:0.2]	$0.4 \le ws(b) \le 1.6$	0.4
c	[4:0.2]	$0.8 \le ws(c) \le 3.2$	0.8
d	[2:0.8]	$0.4 \le ws(d) \le 1.6$	1.6
e	[1:0.4]	$0.2 \le ws(e) \le 0.8$	0.4
W_2-list	$sup(X,W_2)$	$ws_{min}(X) \le ws(X) \le ws_{max}(X)$	$ws(X)$
a	[3:0.3]	$0.6 \le ws(a) \le 2.4$	0.9
b	[3:0.2]	$0.6 \le ws(b) \le 2.4$	0.6
c	[4:0.2]	$0.8 \le ws(c) \le 3.2$	0.8
d	[2:0.8]	$0.4 \le ws(d) \le 1.6$	1.6
e	[1:0.4]	$0.2 \le ws(e) \le 0.8$	0.4
W_3-list	$sup(X,W_3)$	$ws_{min}(X) \le ws(X) \le ws_{max}(X)$	$ws(X)$
a	[3:0.3]	$0.6 \le ws(a) \le 2.4$	0.9
b	[3:0.2]	$0.6 \le ws(b) \le 2.4$	0.6
c	[4:0.2]	$1.2 \le ws(c) \le 3.2$	0.8
d	[2:0.8]	$0.4 \le ws(d) \le 1.6$	1.6
e	[1:0.4]	$0.2 \le ws(e) \le 0.8$	0.4

[그림 1] 윈도우 슬라이드 1에서의 가중치 지지도

식 3에 의해 윈도우 슬라이드 W_1에서 σ는 모든 항목의 최소 가중치 지지도 [0.6, 0.4, 0.8, 0.4, 0.2]의 최대값 0.8과 최대 가중치 지지도 [2.4, 1.6, 3.2, 1.6, 0.8]의 최소값 0.8의 평균인 0.8이 된다. 또한, 최소 가중치 지지도 에러 임계치 ϵ는 최소 가중치 지지도 [0.6, 0.4, 0.8, 0.4, 0.2]의 최소값 0.2와 최대 가중치 지지도 [2.4, 1.6, 3.2, 1.6, 0.8]의 최소값 0.8의 평균인 0.5가 된다. 위의 조건에 따라 W_1에서의 항목 X의 가중치 지지도가 0.8 이상이면 빈발 항목으로 간주되고 만약 0.5보다 크거나 같고 0.8보다 작으면 잠재적 빈발 항목으로, 0.5 보다 작은 가중치 지지도를 갖는 항목에 대하여는 비빈발 항목으로 간주한다.

3.3.2 DCT 생성

본 논문에서 제안한 DCT 트리는 지지도를 기반으로 한 기존의 CP(Closed Pattern) 트리에 가중치 지지도 정보를 추가하여 트리를 생성한다. 트리 생성 과정은 다음과 같은 단계를 수행한다.

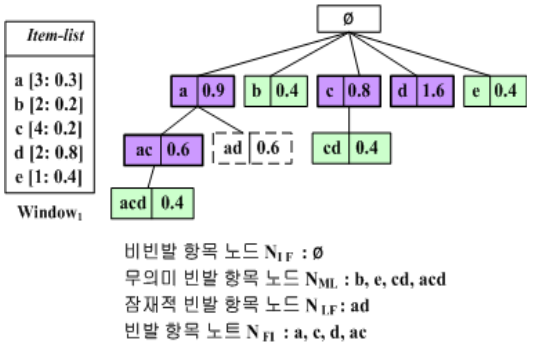
단계 1 데이터 스트림 트랜잭션을 스캔 후 개별 항목에 대하여 지지도를 구하고 항목에 주어진 가중치를 이용하여 1-항목의 가중치 지지도를 구한다.

단계 2 길이가 1인 항목에 대한 노드를 생성하고 트리를 확장하여 다음 레벨의 후보 빈발 항목 노드를 생성한다. DCT 트리의 각 노드는 <X, $ws(X)$>로 항목과 각 항목의 가중치 지지도 값을 유지한다.

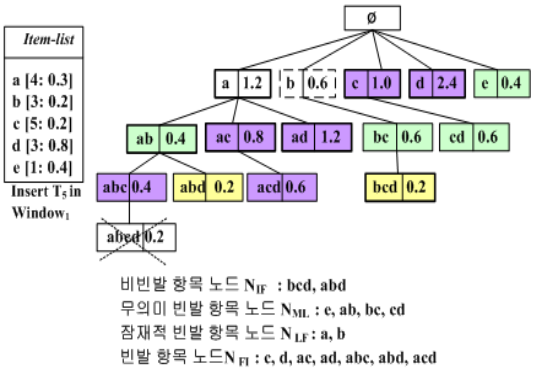
단계 3 DCT 트리를 확장시 최소 가중치 지지도 에러 임계치 미만의 노드는 더 이상 확장하지 않고 빈발 항목 노드에 대하여만 확장 하므로 트리의 노드 정보를 유지 할 때 최소 가용 메모리만 제한적으로 사용한다. 단계 1-3을 수행한 후 생성된 트리는 그림 2와 같다.

단계 4 그림 3은 현재 윈도우 슬라이드 1에 새로운 트랜잭션이 추가된 후 트리의 구성을 나타낸다.

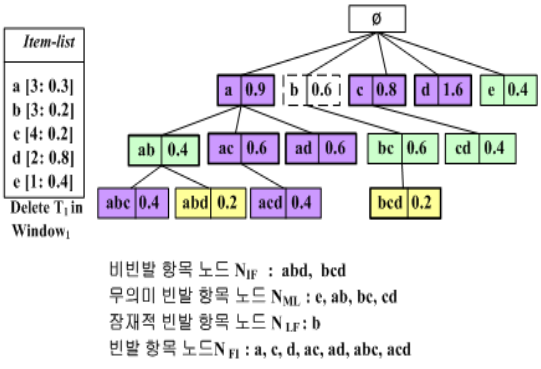
단계 5 그림 4는 현재 윈도우 슬라이드 1에서 단계 4를 수행 후 트랜잭션 T_1을 삭제한 후 트리의 구성을 나타낸다.



[그림 2] 윈도우 슬라이드 1에서의 DCT 구성



[그림 3] 윈도우 슬라이드 1에서의 T5 삽입



[그림 4] 윈도우 슬라이드 1에서의 T_1 삭제

3.3.3 제안된 WSFI-Mine 알고리즘

데이터 스트림에서 패턴의 정보는 윈도우 슬라이딩된 시간의 변화에 점진적으로 갱신된다. 다음은 WSFI-Mine 알고리즘의 원시코드를 나타낸다.

알고리즘 1. Find_WSFI()

입력: T_{DS} //N개의 데이터 스트림
W //윈도우 크기
ω //각 항목에 대한 가중치
R //최대, 최소 가중치 범위

출력: k^{th}-항목 집합//가중치를 갖는 패쇄 빈발 항목 집합

1. Bucket = Null; //윈도우 슬라이드 크기의 임시 저장소
2. Compute(ωs, σ, ϵ) //현재 슬라이드의 L_1의 가중치 지지도, 최소 가중치 지지도, 에러 임계치
3. While(!W=Full) {
4. Construct_DCT(s,ω) //현재 윈도우 슬라이드에 대한 트리 생성
5. If Node(ωs) $\geq$ σ Then FI$\cup N_{FI}$;
6. Else If Node(ωs) $\geq$ ϵ and Node(ωs) $\leq$ σ Then LF$\cup N_{LF}$;
7. Else If Node(ωs) = ChildeNode(ωs) and Node(ωs) $\geq$ ϵ Then ML$\cup N_{ML}$;
8. Else IF$\cup N_{IF}$;

4. 성능 평가

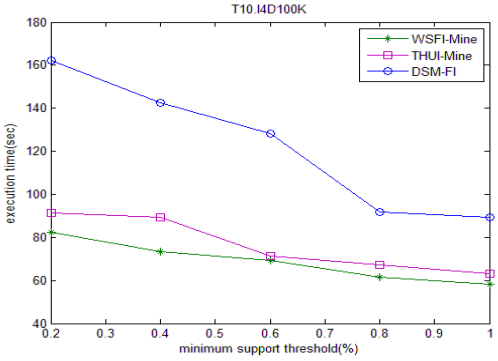
본 장에서는 제안된 알고리즘의 성능을 분석하기 위해 다음과 같은 실험 방법을 사용하였다. 모든 실험은 Microsoft Windows XP Professional 운영체제에서 3GHz CPU, 1GB 메인 메모리에서 C++을 이용하여 구현하였다. 실험에서 사용된 데이터 집합은 표 3과 같다[11]. 데이터 집합 T10I4D100K에서 T는 트랜잭션의 평균 크기를 나타내고 I는 빈발 항목의 평균 길이를 나타낸다. Mushroom은 AI분야에 널리 사용되는 데이터 집합이다. 가중치 범위는 0.2 ~ 0.8 사이의 값을 랜덤하게 각 항목에 적용하였다. 실험은 크게 Landmark Time Based 기법의 전위 트리 기반 DSM-FI 알고리즘과 트랜잭션에 가중치를 고려한 Window Sliding Based 기법의 THUI-Mine 알고리즘과 성능을 비교하였다.

[표 3] 실험 데이터 집합

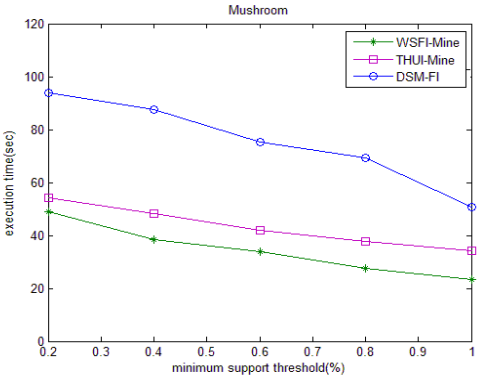
DB	파라메터			
	항목 수	평균 길이	레코드 수	윈도우 크기
T10I4D100K	1,000	10	100,000	10K
Mushroom	120	23	8124	10K

실험은 본 논문에서 제안하는 DCT 기반 WSFI-Mine과 두 가지 비교 알고리즘과의 수행 시간을 비교 하였고,

메모리의 사용량을 알아보기 위해 생성된 후보 빈발 항목 수를 살펴보았다. 그림 5의 (a)와 (b)는 각 각 T10I4D100K와 Mushroom 데이터 집합에 지지도를 변화시키며 수행 시간을 비교한 것이다. 제안된 알고리즘과 THUI-Mine 알고리즘은 윈도우 슬라이딩 기반에서 수행되기 때문에 전체 데이터 집합을 모두 스캔하여 빈발 항목을 탐색하는 DSM-FI 알고리즘보다 성능의 우수함을 알 수 있었고 또한 WSFI-Mine은 트랜잭션 전체 항목의 유틸리티의 합을 이용하여 트랜잭션 단위의 계산을 하는 THUI-Mine 알고리즘보다 계산 비용을 줄이므로 수행 속도가 우수함을 알 수 있다. 그림 6의 (a)와 (b)는 전위 트리 기반의 DSM-FI 알고리즘이 모든 후보 빈발 항목의 정보를 유지하므로 DCT 기반의 제안된 알고리즘에서 패쇄 빈발 항목의 정보만을 유지하는 WSFI-Mine 알고리즘보다 많은 후보 항목을 생성함을 알 수 있었다. 반면 WSFI-Mine 알고리즘과 THUI-Mine 알고리즘은 생성된 후보 항목의 수는 차이가 적어짐을 알 수 있었다. 따라서 제안된 알고리즘은 마이닝 수행 과정시 생성되는 후보 항목을 줄이고 최소의 저장 공간을 사용하여 실행이 가능함을 알 수 있다.

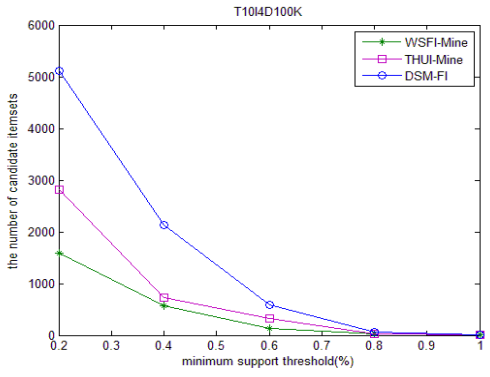


(a) 수행시간 변화 (T10I4100K)

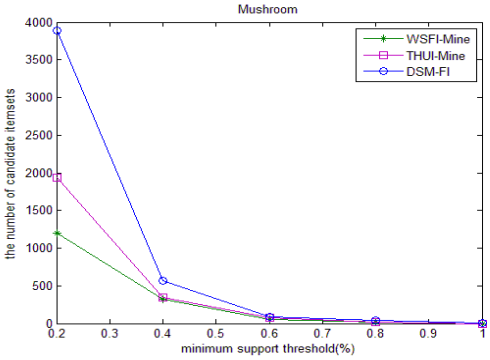


(b) 수행시간 변화 (Mushroom)

[그림 6] 지지도 변화에 따른 수행 시간



(a) 빈발 패턴 생성 수 (T10I4100K)



(b) 빈발 패턴 생성 수 (Mushroom)

[그림 7] 지지도 변화에 따른 빈발 패턴 생성 수

5. 결론 및 향후 과제

데이터 스트림에서의 데이터 마이닝은 처리 시간 및 메모리 사용의 제한과 마이닝 처리 결과의 정확성을 보장할 수 있는 방법이 필요하다. 또한 정보 변화 가능성이 큰 데이터 속성을 갖는 데이터 집합에 대하여 최근의 정보 변화를 빠르게 분석할 수 있는 방법이 필요하다. 본 논문에서는 기존의 지지도 기반 알고리즘에서 간과되는 문제를 효율적으로 해결하고자 중요도가 높은 항목들에 대하여 가중치 지지도를 고려하여 제한된 가용 메모리상에서 가중치 지지도 패쇄 빈발 항목들을 효율적으로 저장하고 유지할 수 있는 DCT 데이터 구조를 제안하였다. 그리고 DCT 트리를 이용하여 한 번의 스캔으로 마이닝을 수행하는 WSFI-Mine 알고리즘을 제안하였다. 제안된 알고리즘은 DCT 트리에서 생성되는 노드의 수를 줄이고 폐쇄 빈발 노드의 지지도를 효율적으로 유지하므로 생성되는 후보 항목의 수를 줄이므로 가용 메모리를 최소화하였다. 그 결과 수행 시간이 비교된 두 개의 알고리즘보

다 향상됨을 보였고, 빈발 항목을 생성하는 후보 항목의 수를 줄이므로 메모리 사용량을 효율적으로 사용할 수 있음을 보였다.

향후 연구 과제로는 유비쿼터스 환경에서의 다양한 응용에 적용 가능한 데이터 스트림 마이닝 기법의 개발과 시간의 변화에 보다 정확한 마이닝 결과를 줄 수 있는 다양한 알고리즘에 관한 연구가 필요하다.

참고문헌

[1] Chang, J., Lee, W.: A Sliding Window Method for Finding Recently Frequent Itemsets over Online Data Streams. Journal of Information Science and Engineering, Vol. 20, No. 4, July, 2004.

[2] Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In Conf. of the 20th VLDB conference, pp. 487-499, 1994.

[3] Manku, G. S., Motwani, R.: Approximate Frequency Counts Over Data Streams. In Proc. of the 28th VLDB, pp. 346-357, 2002.

[4] Li., H.F., Lee, S.Y., Shan, M. K.: An Efficient Algorithm for Mining Frequent Itemsets over the Entire History of Data Streams. In Proceedings of First International Workshop on Knowledge Discovery in Data Streams 9IWKDDS, 2004.

[5] Li., H. F., Lee, S.Y., Shan, M. K.: Online Mining (Recently) Maximal Frequent Itemsets over Data Streams. In Proceedings of the 15th IEEE International Workshop on Research Issues on Data Engineering(RIDE), 2005.

[6] Li, H. F., Lee, S. Y.: Mining frequent itemsets over data streams using efficient window sliding techniques. Expert Systems with Applications, 2008.

[7] Lee, C. H., Lin, C. R., Chen, M. S.: Sliding window filtering: An efficient method for incremental mining on a time-variant database. Information Systems, 30, pp. 227-244, 2005.

[8] Chi, Y., Wang, H., Yu, P. S., Muntz, R. R.: Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window. In Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04), 2004.

[9] Chu, C.J., Tseng, V.S., Mao, R.: An efficient algorithm for mining temporal high utility itemsets from data streams. The Journal of System and Software, no81, pp. 1105-1117, 2008.

[10] Yun, U., Leggett, J, J.: WFIM: Weighted Frequent Itemset Mining with a weight range and a minimum weight. Proceedings of the Fourth SIAM International Conference on Data Mining, pp636-640, 2005.

[11] http://fimi.cs.helsinki.fi/data/

김 영 희(Young Hee Kim) **[정회원]**

- 1997년 8월 : 순천향대학교 일반대학원 전산학과 (전산학석사)
- 2005년 2월 : 성균관대학교 일반대학원 컴퓨터공학과 (박사수료)
- 2003년 3월 ~ 2005년 2월 : 그리스도대학교 객원교수
- 2007년 3월 ~ 2009년 2월 : 인천시립대학 겸임교수
- 1998년 8월 ~ 현재 : 백석대학교, 순천향대학교 컴퓨터학부 외래 교수

<관심분야>
웹데이타베이스, 웹/데이터 스트림 마이닝, RFID 마이닝

김 원 영(Won Young Kim) **[준회원]**

- 2009년 2월 : 성균관대학교 컴퓨터 공학과 (공학사)
- 2009년 3월 ~ 현재 : 성균관대학교 전자전기컴퓨터 공학과(석사과정)

<관심분야>
오피니언 마이닝, 데이터 마이닝, 트리 마이닝

김 응 모(Ung Mo Kim) **[정회원]**

- 1981년 2월 : 성균관대학교 수학과 (이학학사)
- 1986년 5월 : Old Dominion University Computer Science (공학석사)
- 1990년 2월 : Northwestern University Computer Science (공학박사)
- 1990년 9월 ~ 현재 : 성균관대학교 정보통신공학부 교수

<관심분야>
XML/데이터 마이닝, 데이터베이스 보안, GIS, 정보검색