데이터 마이닝에서 샘플링 기법을 이용한 연속패턴 알고리듬

-An Algorithm for Sequential Sampling Method in Data Mining-

홍 지 명*
Hong, Ji Myung
김 낙 현*
Kim, Nak Hyun
김 성 집*
Kim, Sung Jip

Data mining, which is also referred to as *knowledge discovery in database*, means a process of nontrivial extraction of implicit, previously unknown and potentially useful information (such as knowledge rules, constraints, regularities) from data in databases. The discovered knowledge can be applied to information management, decision making, and many other applications.

In this paper, a new data mining problem, *discovering sequential patterns*, is proposed which is to find all sequential patterns using sampling method. Recognizing that the quantity of database is growing exponentially and transaction database is frequently updated, sampling method is a fast algorithm reducing time and cost while extracting the trend of customer behavior. This method analyzes the fraction of database but can in general lead to results of a very high degree of accuracy. The relaxation factor, as well as the sample size, can be properly adjusted so as to improve the result accuracy while minimizing the corresponding execution time.

The superiority of the proposed algorithm will be shown through analyzing accuracy and efficiency by comparing with AprioriAll algorithm.

1. 서론

최근 들어 데이터 마이닝(data mining)은 의사결정 지원, 시장 전략등을 포함한 많은 분야에서 그 다양한 응용 능력 때문에 데이터베이스 분야에서 많은 주목을 끌고 있고 빠르게 성장하고 있다[4]. 과거의 트랜잭션 데이터를 분석한다는 것은 고객의 구입 성향에 대한 매우 가치있는 정보를 제공한다. 이렇게 수집되고 분석되는 데이터의 양은 소프트웨어의 발달과 함께 거대해지는 경향이 있고 거대한 데이터를 효과적으로 분석할 수 있는 도구가 필요하게 되었다. 이러한 배경에서 데이터 마이닝의 역할은 데이터 사이에 존재하는 패턴을 발견하고, 그 패턴을 분석해 정보로서의 가치로 승화시키는 것이다.

데이터 마이닝 기법은 여러 각도에서 연구되고 있다. 이들 중 가장 활발하게 연구되고 있는 분야는 연관 규칙(Association rules)을 찾는 기법이다[1, 2, 5, 6, 8]. 이 기법은 트랜잭션

* 한양대학교 산업공학과

데이터가 주어졌을 때 항목(Item)들간의 관련성을 찾는 문제로 한 트랜잭션에 일부 항목이 존재한다는 것은 같은 트랜잭션에 다른 항목이 존재한다는 관련성을 찾는다. 분류 기법(Mining Classification)은 어떤 공통의 특징에 기초해 데이터들을 집단화 시키는 방법을 찾는 문제이다. 이 문제는 인공지능(AI) 분야에서 연구되고 있다. 군집화(Clustering) 방법은 분산된 형태로 있는 데이터들을 일정한 규모로 규합함으로써 새로운 규칙 및 패턴 가능성을 발견하고자 하는 데이터 마이닝 기법이다.

본 연구에서는 연속 패턴(Sequential Patterns)을 찾는 방법에 샘플링 기법[6]을 적용한 알고리듬을 제시한다. 이 분야는 데이터들의 관계가 시간이 흐르면서 어떠한 패턴을 형성하는지를 발견하고자 하는 데이터 마이닝 기법이다. 오늘날 데이터베이스의 정보가 지수적으로 늘어나고 시간에 따라 수시로 변하는 현실에서 샘플링 기법은 시간과 비용을 줄이면서 추세와 경향을 알 수 있는 효율적이고 빠른 알고리듬이다. 이 방법은 전체 데이터베이스의 일부를 분석함으로써 수행시간을 줄이고 샘플크기와 이완요소를 조정하여 원하는 정확도를 확보하는 매우 효율적인 기법이다.

기존 알고리듬인 AprioriAll과 비교하여 수행시간과 정확도를 분석하여 제안하는 알고리듬의 우수성을 보인다.

2. 기본 개념

2.1 데이터 마이닝(연속 패턴)

바코드의 출현으로 소매 조직들은 거대한 양의 판매 데이터들을 데이터베이스안에 저장할 수 있게 되었고 이 데이터들은 트랜잭션 일수와 거래된 항목으로 구성되어 있다. 특히 신용카드나 단골 고객카드를 이용하여 구입할 경우 고객번호는 트랜잭션 날짜에 포함된다. 소매조직들은 바코드등을 통해 받은 정보의 순서에 따라 데이터를 저장할 수 있다. 경쟁력있는 기업들은 이러한 데이터베이스를 분석하여 의사결정의 중요한 요소로 활용하고 있다.

지식 발견 과정으로 알려진 데이터 마이닝은 최근에 연구가 활발히 전개되고 있다. 이 분야는 거대한 데이터베이스에서 흥미있는 법칙(regularity)을 효과적으로 캐내는 일로 정의할 수 있다.

연속 패턴 문제는 Agrawal[3]에 의해 처음 제기되었다. 이 문제는 데이터를 시간적으로 분석한다는 의미에서 연관 규칙 문제와는 차이가 있다. 연속 패턴 문제에서는 시간에 따른 연속 데이터(data-sequences)가 입력 데이터가 된다. 각각 연속 데이터는 항목들을 포함하는 트랜잭션들로 이루어지고 각 트랜잭션은 트랜잭션 시간들로 구성되어있다. 결론적으로 사용자가 정한 최소 지지도(minimum support)를 만족하는 연속 패턴을 찾아내는 문제라고 볼 수 있다. 여기에서 최소 지지도는 패턴을 포함하는 연속 데이터의 백분율로 정의한다.

예를 들어 컴퓨터 시스템을 갖춘 비디오 가게가 있다고 하자. 고객의 이름을 치면 과거 6개월간 빌려본 비디오의 리스트와 현재 대여중인 비디오를 알 수 있다. 또한 비디오를 대여한 날짜도 상세하게 기록되어 있고 이 데이터는 시간에 따른 연속 데이터라고 생각할 수 있다. 우리는 이 데이터를 통해 의미 있는 연속 패턴 규칙을 발견할 수 있다. 스피드라는 비디오를 빌려본 15%이상의 고객이 다음에 스피드2라는 비디오를 다음에 터미날 스피드를 빌려본다는 연속 패턴을 발견했다고 하자. 또 어떤 특정 영화배우가 나오는 비디오를 연속적으로 서너 편 빌려본 결과가 나왔다고 하자. 우리는 이 결과들을 의사결정에 활용할 수 있을 것이다. 즉 어떤 시기에 위의 결과들에 기초해 비디오의 배치를 다시 할 수 있고, 또한 고객에 비디오를 권유할 때 활용할 수 있다.

연속 패턴 문제는 소매 산업과 우편을 이용한 마케팅(attached mailing), 부가 세일(add-on sails), 고객 만족(customer satisfaction)등에서 시작되었지만 다른 과학분야나 경영 분야에 적용되고 있다. 예를 들어 의료 분야에서 연속 데이터를 환자의 징후(symptoms)나 질병(diseases)으로 보면 트랜잭션은 의사를 방문하는 시점에서 진단된 징후나 질병으로 볼 수 있다. 이 데이터를 통해 발견된 패턴은 질병연구에 많은 도움을 줄 수 있다.

2.2 연속 패턴 문제

고객 트랜잭션을 가진 데이터베이스가 있고 각 트랜잭션은 고객 번호, 트랜잭션 시간, 트랜잭션에서 거래된 항목들로 이루어졌다고 하자. 어떠한 고객도 하나의 시점에 하나 이상의 트랜잭션 시간은 가질 수 없고 하나의 트랜잭션에 거래된 항목의 수는 제한이 없다.

항목집합(itemset)은 공집합이 아닌 항목들의 집합이고 하나의 시퀀스(sequence)는 항목집합들의 순서화된 리스트(list)이다. 항목의 집합을 정수(contiguous integer)로 매핑했을 때, 항목집합 i는 (i_1 i_2 ... i_m) 로 표기하고 i_j는 하나의 항목이다. 또한 시퀀스 S는 < S_1 S_2 ... S_n >로 표기하고 S_j는 하나의 항목집합이다.

만일 $\mathbf{a}_1 \subseteq \mathbf{b}_{i_1}, \mathbf{a}_2 \subseteq \mathbf{b}_{i_2}, \ldots, \mathbf{a}_n \subseteq \mathbf{b}_{i_n}$와 같이 정수 $i_1 < i_2 < \ldots < i_n$이 존재한다면 하나의 시퀀스 < a_1 a_2 ... a_n >는 다른 시퀀스 < b_1 b_2 ... b_m >를 포함할 수 있다. 예를 들어 시퀀스 < (3) (4 5) (8) >는 시퀀스 < (7) (3 8) (9) (4 5 6) (8) >에 포함된다. 왜냐하면 (3)⊆(3 8), (4 5)⊆(4 5 6), (8)⊆(8) 이 되기 때문이다. 시퀀스의 집합에서 시퀀스 s가 다른 시퀀스에 포함되지 않는다면 이를 최대(maximal) 시퀀스라고 한다.

한 고객의 모든 트랜잭션들은 하나의 시퀀스로 볼 수 있고, 각 트랜잭션은 항목들의 집합이고 트랜잭션 시간에 따라 정리된 트랜잭션의 리스트가 된다. 이러한 시퀀스를 고객 시퀀스(customer-sequence)라 한다. 이를 정식화해보면 트랜잭션 시간에 따라 정렬된 고객의 트랜잭션들을 T_1, T_2, ..., T_n이라 하고 T_i에 있는 항목들의 집합을 itemset(T_i)라고 하자. 그려면 이 고객의 고객 시퀀스는 시퀀스 < itemset(T_1) itemset(T_2) ... itemset(T_n) >라고 표현할 수 있다. 한 고객이 시퀀스 s를 지원한다는 말은 이 고객에 대한 고객 시퀀스가 s를 포함하고 있다는 말과 같다. 시퀀스에 대한 지지도(support)는 이 시퀀스를 포함하는 고객의 수를 나타낸다. 최소 지지도를 만족하는 시퀀스를 빈발 시퀀스(large sequence)라고 한다. 시퀀스의 길이(length)는 시퀀스내에 있는 항목집합들의 수를 나타낸다. 즉 길이가 k인 시퀀스를 k-시퀀스라 한다. 그리고 최소 지지도를 만족하는 항목집합을 빈발 항목집합(large itemset or litemset)이라 한다. 빈발 시퀀스에 있는 각 항목집합들은 최소 지지도를 만족해야 한다. 고객 트랜잭션의 데이터베이스 D가 주어졌을 때, 연속 패턴을 찾아낸다는 것은 최소 지지도를 만족하는 모든 시퀀스에서 최대 시퀀스를 찾아내는 일이다. 이때 각 최대 시퀀스는 연속 패턴을 표현한다.

Customer Id	Transaction Time	Items Bought
1	June 25 93	30
1	June 30 93	90
2	June 10 93	10, 20
2	June 15 93	30
2	June 20 93	40, 60, 70
3	June 25 93	30, 50, 70
4	June 25 93	30
4	June 30 93	40, 70
4	July 25 93	90
5	June 12 93	90

표 2.1 : 고객 번호와 트랜잭션 시간이 저장된 데이터베이스

Customer Id	Customer Sequence
1	< (30) (90) >
2	< (10 20) (30) (40 60 70) >
3	< (30 50 70) >
4	< (30) (40 70) (90) >
5	< (90) >

표 2.2 : 고객 시퀀스

Sequential Patterns with support > 25%
< (30) (90) >
< (30) (40 70) >

표 2.3 : 최대 시퀀스

표 2.1은 데이터베이스에 저장되어 있는 고객번호와 트랜잭션 시간을 나타내고 표 2.2는 이 데이터베이스를 고객 시퀀스로 표현한 것이다.

최소 지지도를 25%로 했을 때 즉 2명의 고객 이상을 포함하는 시퀀스는 표 2.3과 같이 < (30) (90) >와 < (30) (40 70) >가 된다. 이 시퀀스들은 최소 지지도를 만족하는 최대 시퀀스이고 연속 패턴의 해답이다. 연속 패턴 < (30) (90) >은 고객 1과 4가 지원하고 고객 4는 항목 (30)과 (90)사이에 항목집합 (40 70)을 거래했는데 이는 시간적인 흐름을 고려한 것이므로 반드시 인접할 필요는 없다. 연속 패턴 < (30) (40 70) >은 고객 2와 4가 지원한다. 고객 2의 경우 항목 60을 40, 70과 같이 거래했지만 이 패턴을 지원한다. 왜냐하면 항목집합 (40 70)은 항목집합(40 60 70)의 부분집합이기 때문이다.

위의 예에서 시퀀스 < (10 20) (30) >은 최소 지지도를 만족하지 못하고, 시퀀스 < (30) >, < (40) >, < (70) >, < (90) >, < (30) (40) >, < (30) (70) >, < (40 70) >은 최소지지도는 만족하지만 최대 시퀀스는 아니다.

3. 기존 연구

3.1 알고리듬 AprioriAll

연속 패턴을 찾는 문제는 몇 가지 단계로 나눌 수 있다.

1. 정렬 단계

이 단계는 데이터베이스에 있는 자료를 정렬하는 작업인데, 고객번호를 주요키(major key)로 트랜잭션 시간을 보조키(minor key)로 사용한다. 이 단계는 표 2.1과 표 2.2와 같이 초기(original) 트랜잭션 데이터베이스를 고객 시퀀스 데이터베이스로 변환한다.

2. 빈발 항목집합 단계

여기에서는 모든 빈발 항목집합 L을 찾아내는 일인데 이 집합은 { < ℓ > | $\ell \in$ L }로 나타낼 수 있기 때문에 동시에 빈발 1-시퀀스의 집합을 구할 수 있다. 표 2.2에서 보면 최소 지지도를 만족하는 빈발 항목집단들은 (30), (40), (70), (40 70), (90)임을 알 수 있고 이를 정수의 집합으로 매핑한다. 표 3.1는 매핑한 예를 보여주고 있는데 이는 항목집단을 단일 개체로 취급하기 위함이다.

Large Itemsets	Mapped To
(30)	1
(40)	2
(70)	3
(40 70)	4
(90)	5

표 3.1 : 빈발 항목집합

3. 변환 단계

주어진 빈발 항목집단들 중 어느것이 고객 시퀀스에 포함되는지 반복적으로 결정해야 할 필요가 있다. 이 과정을 빠르게 하기 위해서 고객 시퀀스를 다른 표현으로 전환한다. 변환된 고객 시퀀스에서, 각 트랜잭션은 그 트랜잭션에 포함된 모든 빈발 항목집합의 집합으로 나타난다. 만일 고객 시퀀스가 어떤 빈발 항목집합을 갖지 않는다면 변환된 데이터베이스로부터 제거된다. 이제 고객 시퀀스는 빈발 항목집합들의 리스트로 표현되고 각 빈발 항목집합들의 집합은 $\{ \ell_1, \ell_2, \ldots, \ell_n \}$으로 나타나고 여기에서 ℓ_i는 반발 항목집합이다. 표 2.2의 고객 시퀀스 데이터베이스를 변환한 예가 표 3.2에 나타나 있다. 이 예에서 고객번호 2의 트랜잭션 (10 20)은 제거 되었는데 이는 어떤 빈발 항목집합을 포함하지 않았기 때문이다. 그리고 트랜잭션 (40 60 70)은 빈발 항목집합의 집합인 { (40), (70), (40 70) }으로 바뀌었다.

Customer Id	Original Customer Sequence	Transformed Customer Sequence
1	< (30) (90) >	< {(30)} {(90)} >
2	< (10 20) (30) (40 60 70) >	< {(30)} {(40), (70), (40 70)} >
3	< (30 50 70) >	< {(30), (70)} >
4	< (30) (40 70) (90) >	< {(30)} {(40), (70), (40 70)} {(90)} >
5	< (90) >	< {(90)} >

표 3.2 : 변환된 데이터베이스

Customer Id	After Mapping
1	< { 1 } { 5 } >
2	< { 1 } { 2, 3, 4 } >
3	< { 1, 3 } >
4	< { 1 }, { 2, 3, 4 } { 5 } >
5	< { 5 } >

표 3.3 : 매핑된 데이터베이스

4. 시퀀스 단계

표 3.1에 기초해서 표 3.2의 데이터베이스를 표 3.3으로 바꿀 수 있다. 이 단계에서는 데이터에 대해 반복적인 과정을 거쳐 빈발 시퀀스를 구한다. 각 단계에서 잠재적인 빈발 시퀀스를

구하기 위해 후보 시퀀스를 이용한다. 후보 시퀀스는 연관 규칙의 Apriori에서와 같이 L_1*L_1을 사용한다[2]. 데이터베이스를 통해 후보 시퀀스의 지지도를 계산한 후 빈발 시퀀스를 결정한다. 처음 단계에서는 표 3.1의 1-시퀀스를 이용한다. 빈발 k-시퀀스를 구하는 과정이 표 3.4에 나타나 있다.

L_1

1-Sequences	Support
< 1 >	4
< 2 >	2
< 3 >	3
< 4 >	2
< 5 >	3

L_2

2-Sequences	Support
< 1 2 >	2
< 1 3 >	3
< 1 4 >	2
< 1 5 >	2
< 2 3 >	2
< 2 4 >	2
< 3 4 >	2

L_3

3-Sequences	Support
< 1 2 3 >	2
< 1 2 4 >	2
< 1 3 4 >	2
< 2 3 4 >	2

L_4

4-Sequences	Support
< 1 2 3 4 >	2

표 3.4 : 빈발 시퀀스

5. 최대 시퀀스 단계

빈발 시퀀스들의 집합에서 최대 시퀀스를 찾는 과정이다. 최대(maximal) 시퀀스는 시퀀스의 집합에서 시퀀스가 다른 시퀀스에 포함되지 않는 시퀀스이므로 표 3.4에서 보면 < 1 5 > 와 < 1 2 3 4 >가 된다.

< 1 5 > => < (30) (90) >

< 1 2 3 4 > => < (30) (40) (70) (40 70) > => < (30) (40 70) >

표 2.3의 결과와 같음을 알 수 있다.

3.2 연속 패턴 알고리듬

앞에서 언급한 5단계를 거쳐 최대 시퀀스를 구하는 알고리듬이 AprioriAll이다. 다른 대표적인 연속 패턴 알고리듬으로 GSP[7]가 있다. 이 알고리듬은 연속 패턴 문제에 일반화(generalization)를 적용시켰고 시간 제약(time constraint), 윈도우(Sliding time windows), 분류(Taxonomy : is-a hierarchy)개념이 도입된 연속 패턴 알고리듬이다.

4. 제안하는 알고리듬

본 연구에서는 후보 시퀀스 집합의 생성에 매우 효과적인 해시 기법(hash technique)을 이용한다[5]. 이는 초기 반복 시행에서 Ck의 항목들의 수를 현저하게 감소시켜 전체 프로세서의 병목현상을 개선한다. 또한 샘플링 기법을 이용하여 정확성을 유지하면서 알고리듬 효율성을 높이도록 한다.

4.1 기호 설명

본 연구에 사용되는 기호는 다음과 같다.

α: 이완 요소

s : 최소지지도

αs : 이완된 지지도

L_k : 데이터베이스의 빈발 k-시퀀스들의 집합

C_k : 후보 시퀀스들의 집합

H_2 : 2-시퀀스 집합에 대한 해시 테이블

L_k' : 샘플링 데이터의 빈발 k-시퀀스들의 집합

C_k' : 샘플링 데이터의 후보k-시퀀스들의 집합

H_2' : 샘플링 데이터의 2-시퀀스 집합에 대한 해시 테이블

F : L_1에서 제외된 항목들의 집합

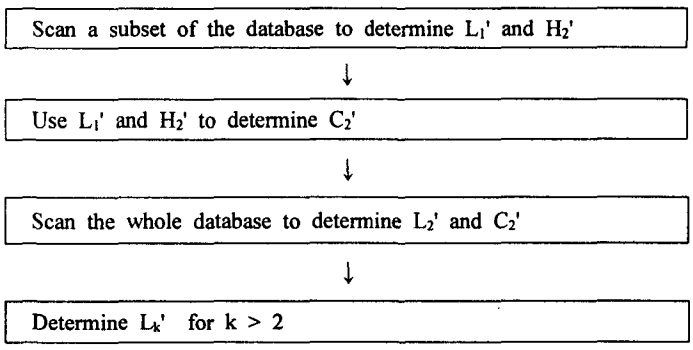
4.2 알고리듬 설명

먼저 고객 데이터베이스의 일부를 샘플크기만큼 선택한다. 이 선택된 데이터로부터 L_1'과 H_2'를 구한다. L_1에 포함된 1-항목에 이용된 최소 지지도는 s로 선택한다. 는 이완 요소로 [0, 1]사이의 값을 갖는다. s가 s보다 적으므로 L_1는 실제 L_1보다 많은 항목을 가지는 경향이 있다. 샘플크기뿐만 아니라 이완요소는 수행시간을 최소화시키면서 정확성을 개선하기 위해 적당히 조정될 수 있다.

L_1'과 H_2'를 구한 후에 이로부터 후보 2-시퀀스인 C_2'를 구한다. AprioriAll에서 빈발 k-시퀀스 L_k는 빈발 (k-1)-시퀀스에서 유도된다. 즉 빈발 k-시퀀스에 나타난 항목들은 반드시 빈발 (k-1)-시퀀스에 있어야 한다. 다른 말로 L_1에 없는 항목은 L_k(k > 1)의 구성요소에서 제거된다. 이 개념을 이용하여 수행시간을 줄일 수 있다. L_1'과 H_2'에서 C_2'를 구할 때 C_2'에 없는 항목은 L_1'에서 제거된 항목과 같다. 이 항목들을 모아서 F로 정의하고 빈발 시퀀스의 효율성을 증가시키는 필터로서 이용한다.

C_2'를 구한 후에 전체 데이터베이스를 읽어 L_2'를 결정한다. 이때 F에 속하는 항목이 있다면 빈발 2-시퀀스 L_2'를 만들 때 이 항목을 제거한다. 지금까지 전체 데이터베이스의 일부고객 트랜잭션으로 분석을 했지만 이 단계에서는 전체 데이터베이스를 검색한다. L_2' 이전 단계에서 샘플링 기법을 사용한 이유는 수행시간의 대부분이 L_2' 이전 단계에서 소비되기 때문이다.

마지막으로 남은 L_k'(k > = 3)를 구하는 과정은 AprioriAll과 같지만 C_k'에서 L_k'를 만드는 과정에서 L_{k+1}'를 만드는데 사용되지 않는 고객 트랜잭션은 데이터베이스에서 제거된다. 이 전지(Pruning)기법[5]은 결과적으로 여러 단계를 거치는 동안 트랜잭션 크기는 상당히 줄어들게 된다.



단계 1 : 트랜잭션 데이터베이스에서 일부를 선택하여 최소지지도 αs로 L_1', H_2'를 결정

단계 2 : L_1' 과 H_2'에서 후보 k-시퀀스인 C_2'를 구하고 C_2'를 만드는데 사용되지 않은 항목을 F로 정의한다.

단계 3 : 전체 데이터베이스를 검색하여 C_2'로부터 L_2'를 구한다. 만일 트랜잭션의 항목이 F에 속한다면 이 항목을 L_2'를 만드는데 필터링한다.

단계 4 : L_k' (k > 2)를 구한다.

그림 4.1 : 전개도와 단계

5. 실 험 및 결과

제안한 알고리듬의 성능평가와 증명을 위해 실험을 수행하였다. 프로그램 언어는 C++이고 컴파일러는 VisualC++을 사용했고 기계는 CPU 150MHZ, 주메모리 32MB를 사용하였다.

5.1 실험 데이터 생성

실험 트랜잭션을 생성하는데 각 트랜잭션은 N항목으로부터 만들어진다. 실험 트랜잭션을 만드는 방법은 Agrawal[2]에서 제시한 방법과 비슷하다. 실험 데이터베이스는 고객 트랜잭션으로 이루어져 있는데 고객의 수는 |D|가 되고 고객당 트랜잭션의 수는 |C|, |T|는 트랜잭션이 가지고 있는 평균 항목의 수를 나타낸다.

|D|	고객의 수
|C|	고객당 트랜잭션의 평균 개수
|T|	트랜잭션당 항목의 평균 개수
|S|	최대 잠재적인 빈발 시퀀스의 평균 길이
|I|	최대 잠재적인 빈발 시퀀스에서 항목집합들의 평균 크기
N_s	최대 잠재적인 빈발 시퀀스의 개수
N_I	최대 잠재적인 빈발 항목들의 개수
N	항목들의 개수

그림 5.1 : 고객 트랜잭션 생성에 사용된 파라메타 목록

실험에서 사용한 파라메타의 값은 N = 100, N_s = 2000, N_I = 10000, |C| = 5, |T| = 5, |S| = 4, |I| = 1.2을 사용하였다. 예를 들어 C10-T5-S4-I1.25의 의미는 한 고객에 대한 트랜잭션의 평균 개수가 10이고, 한 트랜잭션당 항목들의 평균 개수가 5이고, 최대 잠재적인 빈발 시퀀스의 평균 길이는 4이고, 최대 잠재적인 빈발 시퀀스에서 항목집합들의 평균 크기는 1.25임을 의미한다.

5.2 실험 결과

본 실험은 수행시간과 정확성을 비교하기 위해 3개의 알고리듬을 비교하였다. AprioriAll과 제안하는 알고리듬에서 샘플링을 사용하지않은 경우와 사용한 경우로 나누어진다. 실험 데이터는 C5-T5-S4-I1.25에 따라 생성되었고 고객의 수는 10000으로 하였다. 그림 5.2는 최소 지지

도를 0.5 %에서 2 %까지 단계적으로 변화 시켰을 때 상대시간을 표시한 것이다. 샘플링에 사용된 파라메타는 이완요소 0.75와 샘플크기는 고객의 10 %로 하였다.

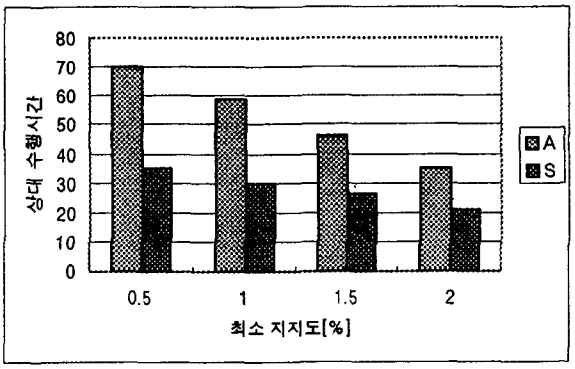


그림 5.2 : 최소 지지도의 변화에 따른 수행시간의 비교

그림 5.2에서 보여지듯이 샘플링을 사용한 알고리듬이 수행시간에서 개선 효과가 있음을 확연히 알 수 있다. A는 기존 알고리듬이고 S는 샘플링 기법을 이용한 알고리듬이다. 최소 지지도가 0.5 %인 경우 그림에서는 0.5로 표기되었다.

Num. of AprioriAll	Num. of S	Difference
| L2| = 212	| L2| = 218	6
| L3| = 145	| L3| = 147	2
| L4| = 26	| L4| = 26	0
| L5| = 8	| L5| = 8	0
| L6| = 2	| L6| = 2	0

그림 5.3 : s가 0.8일때 실험결과

샘플링 기법을 이용할 경우 수행 속도는 빠르지만 직관적으로 정확도를 보장하기가 힘들다. 그림 5.3은 s가 0.8일때 정확도의 차이가 존재함을 알 수 있다. 파라메타들의 변화로 정확도의 차이가 어떻게 변하는지를 살펴보았다.

그림 5.4는 샘플링을 이용한 알고리듬에서 샘플크기에 대한 정확성의 변화를 나타낸 것이다. 샘플링 방법을 이용할 경우 그림 5.3처럼 전체 고객 트랜잭션을 비교한 결과와 일치하지 않을 가능성이 존재한다. 샘플링을 사용한 빈발 시퀀스 L_k'($k > 1$)와 실제 빈발 시퀀스 L_k($k > 1$) 사이에 차이가 있다면 이를 불일치라고 하고 그 결과들의 차이를 비교오차라 하자.

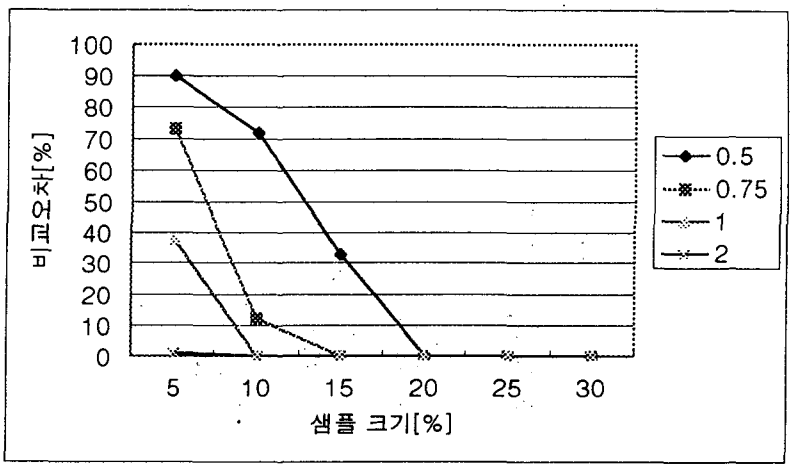


그림 5.4 : 샘플크기에 따른 비교오차

그림에서 보듯이 샘플크기를 5 %에서 30 %까지 단계적으로 증가 시켰을 때 비교오차의 비율은 줄어듬을 알 수 있다. 샘플링 기법을 적용했을 때 낮은 최소지지도와 적은 샘플크기를 선택할 때 정확도는 낮아지지만 샘플크기가 10 %에서 최소 지지도 1.0 %이상일 때는 거의 차이가 없음을 알 수 있다.

그림 5.5는 이완요소 α에 대한 비교오차의 양을 그래프로 나타낸 것이다. 샘플의 크기는 10 %로 고정되어 있다. α값을 0.6에서 1까지 단계적으로 변화했을 때 3가지 최소 지지도에 대한 비교오차의 변화를 나타낸 것이다.

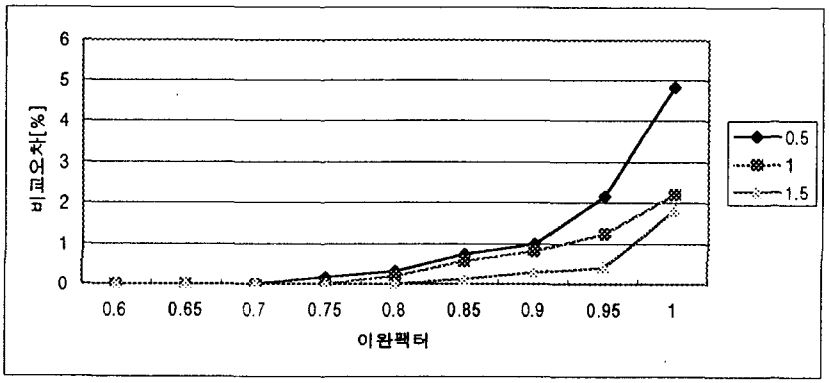


그림 5.5 : 이완요소 α의 변화에 대한 불일치의 양

비교오차를 구하는 식은 다음과 같다.

$$\varepsilon = \frac{\sum_{k=2}^{c} m_k}{\sum_{k=2}^{c} |L_k|}$$

p는 공집합이 아닌 최대 빈발 j-시퀀스의 집합을 나타낼 때 마지막 j를 나타내고 m_k는 샘플링으로 인한 빠진 k-시퀀스의 개수이다. 이완요소가 증가하면 비교오차의 양은 많아짐을 알 수 있다. 그 이유는 최소 지지도 s대신에 αs를 사용할 때 후보 2-시퀀스 C_2'에 더 많은 항목들이 포함되기 때문이다. 값이 낮아지면 C_2'에서 L_2'를 구할 때 L_2'값은 실제 L_2에 더 가까워지고 비교오차는 제로값에 근접한다. 하지만 C_2'가 늘어나므로 수행시간은 상대적으로 늘어난다.

그림 5.4에서 샘플크기가 줄어들면 정확도는 떨어짐을 알 수 있다. 즉 샘플크기와 정확도는 절충(trade-off)관계임을 알 수 있고 그림 5.5에서 이완요소 α도 정확도를 통제하는 또 하나의 요소임을 보여주고 있다. 이는 샘플링을 이용할 때 샘플크기와 이완요소는 주요한 변수임을 알 수 있고 샘플크기를 늘리고 이완요소를 줄임으로서 정확도를 향상시키는 반면 수행시간은 줄일 수 있다. 실험결과 최소지지도 1 %이상에서 α = 0.6, 샘플크기 5 %와 α = 0.75, 샘플크기 10 %인 경우 정확성에서 거의 차이가 없이 수행시간을 줄일 수 있음을 알 수 있다.

6. 결론 및 추후 연구과제

데이터 마이닝에서 시간에 따라 시퀀스를 찾는 연속 패턴문제는 최근에 연구가 활발히 진행되고 있고 샘플링을 이용한 방법은 분석해야 하는 데이터베이스의 양을 상당히 줄여주는 효율적인 기법이다. 샘플링의 크기와 이완요소를 조정함으로써 수행시간을 줄이고 원하는 정확도의 수준에 접근할 수 있다.

오늘날 데이터베이스의 크기는 지수적으로 늘어나고 정보는 하루에도 수십 번 추가되거나 갱신된다. 이러한 정보를 빠르게, 자주 분석하여 정보의 추세나 흐름을 찾는 것이 더 효율적일 때가 있다. 샘플링을 이용한 방법은 이런 상황에서 비용과 시간을 줄이고 효율을 높이는데 필요하다.

추후 연구과제로 추가되는 데이터베이스의 양이 많고 정보의 내용이 기존 데이터베이스의 내용과 상관관계가 적을 때 샘플링 적용시에 이들 정보에 어떻게 비중을 둘 것인가에 대한 연구가 기대된다.

참고문헌

1. Agrawal R., T. Imielski, and A. Swami, Mining Association Rules between Sets of Items in the Large Datebase, *Proceeding of the ACM SIGMOD Conference*, May, 1993.
2. Agrawal, R. and R. Srikant, Fast Algorithms for Mining Association Rules, *Proceedings of the 20th VLDB Conference*, 1994.
3. Agrawal R., and R. Srikant. Mining Sequential Patterns, *Proceeding of the 11th International Conference on Data Engineering*, Taipei, Taiwan, March 1995.
4. Chen, M. S., J. Han, and P. S. Yu, Data Mining : An Overview from Database Perspective, *IEEE Transaction on Knowledge and Data engineering,* Vol.8, No.6, December, 1996.
5. Park, J. S., M. S. Chen, and P. S. Yu, An Effective Hash-Based Algorithm for Mining Association Rules, *Proceeding of ACM SIGMOD International Conference on Management of Data*, May, 1995.
6. Park, J. S., P. S. Yu, M. S. Chen, Mining Association Rules with Adjustable Accuracy, *IBM Research Report*, RC 20695, 25Pages, 1/17/97.

7. Srikant and R. Agrawal. Mining Sequential Patterns : Generalizations and Performance Improvements, *Research Report RJ 9994, IBM Almaden Research Center*, San Jose, California, December 1995.
8. 이동명, 데이터 마이닝에서 기존의 연관 규칙을 갱신하는 앨고리듬 개발 한양대학교 대학원 석사논문.