다수의 2 차원 배열에서 효율적인 빈발 패턴 탐색 기법

김한슬[0], 이기용
숙명여자대학교 소프트웨어융합학과
{uo3359, kiyonglee}@sookmyung.ac.kr

Efficient Frequent Pattern Mining in Multiple Two-Dimensional Arrays

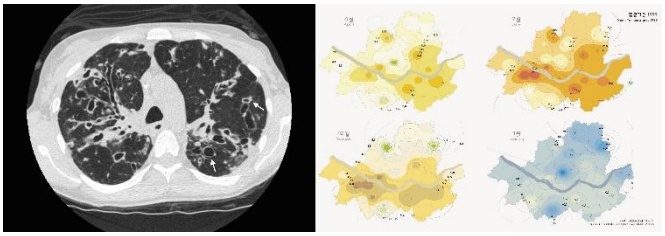
Han-seul Kim[0], Ki Yong Lee
Dept. of Software Convergence, Sookmyung Women's University

요 약

데이터베이스에서의 빈발 패턴 탐색은 일정 횟수 이상 같이 등장하는 항목들의 집합을 찾는 문제이다. 본 논문은 다수의 2 차원 배열들이 주어졌을 때, 이들 내부에서 빈번히 같이 등장하는 부분 구역들을 찾는 문제를 다룬다. 하지만 기존 빈발 패턴 탐색 기법들은 배열 내 원소들의 위치 관계까지 고려하지는 않기 때문에 본 문제에 바로 적용하기는 어렵다. 따라서 본 논문은 다수의 2 차원 배열 내에서 빈번히 같이 발생하는 부분 구역들을 효율적으로 찾는 기법을 제안한다. 본 논문의 선행 연구에서는 주어진 배열들을 두 번 스캔하여 빈발 부분 구역 집합을 찾는 기법을 제안하였다. 본 논문에서는 이를 개선하여 배열들을 한 번만 스캔하고도 빈발 부분 구역 집합을 찾는 효율적인 기법을 제안한다. 이를 위해 제안 방법은 지금까지 탐색된 부분 구역들에 대한 정보를 메모리에 효율적으로 유지한다. 실험결과 제안방법은 기존 방법에 비해 수행시간을 약 30% 단축함을 보였다.

1. 서론

데이터 수집이 이뤄지는 분야가 다양해지면서 여러 분야에서 다양한 형태의 데이터가 생성되고 있다. 그 중 2 차원 배열 데이터는 원소들이 행과 열에 따라 직사각형 모양으로 나열된 데이터를 말한다. 배열 데이터는 그림 1 과 같이 이미지를 나타내거나 또는 어떤 지역의 관측 값들을 나타내는데 널리 사용된다.



(그림 1) CT 촬영 데이터와 서울 기온 및 강수량 데이터

한편 데이터베이스에서의 빈발 패턴 탐색이란 데이터베이스 내의 항목들 중 일정 횟수 이상 같이 등장하는 항목들의 집합을 찾는 문제를 말한다. 지금까지 빈발 패턴 탐색에 대해서는 매우 많은 연구가 이루어져왔다[1][2][3][4]. 본 논문에서는 다수의 2 차원 배열들이 주어졌을 때, 이들 내부에서 빈번히 같이 등장하는 부분 구역들을 찾는 문제를 다룬다. 하지만 기존 빈발 패턴 탐색 기법들은 같이 구매한 상품들의 경우처럼 각 데이터가 항목들의 집합 형태로 표현되는 경우를 주로 고려하며, 배열과 같이 원소들 간에 위, 아래, 좌우와 같은 위치 관계가 존재하는 경우까지 고려하는 연구는 거의 없었다.

따라서 본 논문에서는 주어진 다수의 2 차원 배열들 내에서 빈번히 같이 발생하는 부분 구역들을 효율적으로 탐색하는 방법을 제안한다. 제안 방법은 그림 1 과 같은 이미지 또는 관측값을 나타내는 2 차원 배열들이 여러 개 주어졌을 때, 이들 내부에서 어떤 구역에 어떤 값이 나타날 때 다른 구역에 어떤 값이 흔히 나타나는 지에 대한 패턴을 탐색하는데 유용하게 사용될 수 있다. [5]는 본 논문의 사전 연구로서, 주어진 2 차원 배열들 내에서 원소들의 값이 동일한 부분 구역들을 먼저 추출한 뒤 이들 중 동일한 배열 내에서 빈번히 같이 발생하는 부분 구역들을 찾는 방법을 제안하였다. 하지만 [5]의 방법

은 빈발 부분 구역 집합을 찾기 위해 주어진 배열들을 두 번 스캔해야 하며, 따라서 배열들의 개수가 많아지거나 크기가 커질수록 수행시간이 크게 증가한다는 단점이 있다.

이에 비해 본 논문에서 제안하는 방법은 주어진 배열들을 한 번만 스캔하고도 빈발 부분 구역 집합을 효율적으로 탐색한다. 이를 위해 제안 방법은 지금까지 탐색된 부분 구역들에 대한 정보를 메모리에 테이블 형태로 효율적으로 유지한다. 실험 결과 제안 방법은 기존 방법에 비해 수행시간을 대략 30% 정도 감소시킴을 확인할 수 있었다.

본 논문의 구성은 다음과 같다. 2 장에서는 문제를 정의하고, 3 장에서는 기존의 대표적인 빈발 패턴 탐색 기법들과 그들의 한계점을 살펴본다. 4 장에서는 본 논문에서 제안하는 기법을 설명하고, 5 장에서 제안 기법의 실험 결과를 제시한다. 마지막으로 6 장에서는 본 논문의 결론을 맺는다.

2. 문제 정의

본 논문에서는 크기가 $n \times m$으로 동일한 2 차원 배열들이 여러 개 주어져 있다고 가정한다. 여기서 $n\,(n \geq 1)$은 배열의 행 수를, $m\,(m \geq 1)$은 열 수를 나타낸다. 본 논문에서는 배열 A에 대해 그의 $i(i = 0, 1, \ldots, n-1)$번째 행의 $j(j = 0, 1, \ldots, m-1)$번 째 열에 위치한 원소를 A_{i*m+j}로 나타낸다. 본 논문에서는 2 차원 배열 내에서 서로 인접해 있으며 서로 같은 값을 가진 원소들로 이루어진 부분 배열을 부분 구역이라 부른다.

본 논문에서는 여러 개의 2 차원 배열들과 사용자가 지정한 최소 크기($Size_{min}$) 및 최소 지지도 ($Support$)가 주어졌을 때, $Support$번 이상 동일한 배열 내에서 같이 등장하며, 각각의 크기가 $Size_{min}$ 이상인 부분 구역들로 이루어진 빈발 부분 구역 집합을 찾는 문제를 고려한다. 본 논문에서는 이 문제를 2 차원 배열 내 빈발 패턴 탐색 문제라 부른다.

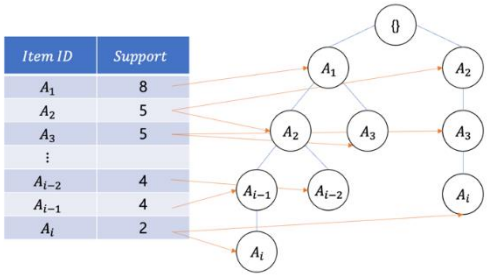
3. 관련 연구

빈발 패턴을 탐색하는 알고리즘은 많은 분야에서 연구되어 왔다[1][2][3][4]. 그 중 $Apriori$ 알고리즘과 $FP-Growth$ 알고리즘은 가장 대표적인 알고리즘으로서 빈발 항목 집합 외에도 빈발 부분 시퀀스, 빈발 부분 그래프 탐색 등 다양한 형태의 데이터에 대한 빈발 패턴 탐색에 폭 넓게 활용되고 있다. 본 장에서는 빈발 패턴 탐색의 대표적인 알고리즘인 $Apriori$ 알고리즘과 $FP-Growth$ 알고리즘 및 그들의 한계점에 대해 간략히 설명한다.

3-1. Apriori

$Apriori$ 알고리즘은 1994 년에 제안된 기법으로 대용량의 장바구니 데이터베이스에 존재하는 아이템들 간의 관계를 분석하기 위해 최초로 제안되었다[1]. $Apriori$ 알고리즘은 데이터베이스에서 사용자가 지정한 최소 지지도 이상 같이 나타나는 빈발 항목 집합을 탐색하는 알고리즘이다.

이 알고리즘은 $(k+1)$개의 항목들로 구성된 빈발 항목 집합을 찾기 위해 k개의 항목들로 구성된 빈발 항목 집합을 이용한다. $Apriori$ 알고리즘에서 빈발 패턴을 탐색하는 순서는 다음과 같다. 우선 데이터베이스를 스캔하면서 최소 지지도 이상 등장하는 항목들을 탐색하여 1 개의 항목들로 구성된 빈발 항목 집합을 구성한다. 이후 이들을 조합하여 2 개의 항목들로 구성된 후보 빈발 항목 집합들을 구성한다. 그리고 다시 데이터베이스를 스캔하여 이들 중 최소 지지도 이상 등장하는 항목 집합들을 탐색하여 2 개의 항목들로 구성된 빈발 항목 집합을 구성한다. 이러한 과정을 k = 1, 2, …에 대해 반복적으로 수행하여 1, 2, …, k개의 항목들로 구성된 빈발 항목 집합을 찾고, 더 이상 빈발 항목 집합이 탐색되지 않으면 알고리즘을 종료한다. 하지만 $Apriori$ 알고리즘은 2 차원 배열과 같이 원소들 간에 위치 관계가 있는 경우까지는 고려하지 않으며, 따라서 본 논문에서 고려하는 2 차원 배열 내 빈발 패턴 탐색 문제에 바로 적용하기는 어렵다.



(그림 2) FP-Growth 에서 사용하는 FP-Tree

3-2. FP-Growth

트리 형태의 구조를 활용하는 $FP-Growth$ 알고리즘은 후보 빈발 집합을 생성하지 않는 방식이다. 이 알고리즘은 우선 주어진 최소 지지도 이상 등장하는 항목들에 대한 정보를 그림 2 와 같은 형태의 트리에 저장한다. $FP-Growth$는 우선 전체 데이터베이스를 스캔하여 동일한 트랜잭션에 같이 등장하는 항목들의 정보를 그림 2 와 같은 트리에 저장한 후, 최소 지지도 이하로 등장하는 항목들을 제거한다. 이때 빈발도가 높은 항목들을 루트 노드에 가깝게 저장하며 이 트리를 $FP-Tree$라고 부른다[3].

$FP-Tree$는 데이터베이스에 존재하는 모든 빈발 항목들에 대한 정보를 가지고 있기 때문에 후보 빈발 항목 집합을 별도로 생성할 필요가 없으며, $FP-Tree$를 재귀적으로 탐색하면 각 빈발 항목에 대해 그를 포함하는 모든 빈발 항목 집합들을 찾아낼 수 있다. 따라서 $FP-Growth$는 $Apriori$ 알고리즘에 비해 후보 빈발 집합을 생성하지 않고도 빈발 항목 집합들을 빠르게 얻어낼 수 있다는 장점을 가지고 있다. 하지만 $FP-Tree$가 매우 커져 메모리에 저장하기 어려

운 경우에는 디스크를 반복적으로 접근해야 하므로 성능이 크게 저하된다는 단점이 있다. 또한 *Apriori* 알고리즘과 마찬가지로 원소들 간에 위치 관계가 있는 경우까지는 고려하지 않으며, 역시 본 논문에서 고려하는 배열 내 빈발 부분 구역 패턴 탐색 문제에 바로 적용하기는 어렵다.

이를 극복하기 위해 본 논문의 선행 연구인 [5]에서는 다수의 2차원 배열들이 주어졌을 때, 동일한 배열 내에서 빈번히 같이 발생하는 부분 구역들을 탐색하는 기법을 제안하였다. [5]에서 제안한 방법은 우선 배열들을 한 번 스캔하여 서로 인접해 있으며 서로 같은 값을 가진 원소들로 이루어진 부분 구역을 추출한다. 이후 배열들을 다시 한 번 스캔하여 탐색된 부분 구역들의 빈도수를 측정한다. 마지막으로 최소 지지도 이상 등장하는 부분 구역들에 대해 *Apriori* 알고리즘이나 $FP-Growth$ 알고리즘을 적용하여 최종적으로 빈발 부분 구역 집합을 찾는다. 따라서 [5]는 주어진 배열들을 두 번 스캔해야 하기 때문에 배열들의 개수가 많아지거나 크기가 커질수록 수행시간이 크게 증가한다는 단점이 있다.

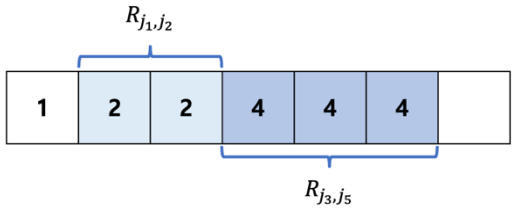
4. 제안 기법

3장에서 설명한 기존의 빈발 패턴 탐색 기법들은 2차원 배열 내 빈발 패턴 탐색 문제에 바로 적용할 수 없거나 비효율적이라는 단점이 있다. 따라서 본 논문에서는 2차원 배열 내 빈발 부분 구역 패턴을 효율적으로 탐색할 수 있는 기법을 제안한다.

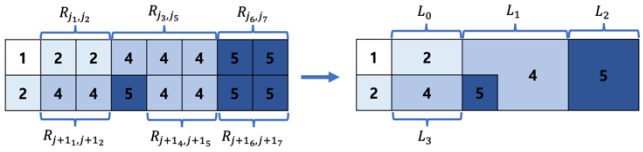
제안 기법은 두 단계로 나뉜다. 첫 번째 단계는 주어진 배열들을 스캔하여 배열 내 부분 구역들을 탐색하는 한편, 동시에 이들의 빈도수를 지속적으로 측정하여 이들 중 최소 지지도 이상 등장하는 부분 구역들만을 찾는 단계이다. 두 번째 단계는 첫 번째 단계에서 탐색된 빈발 부분 구역들을 사용하여 최종 빈발 부분 구역 집합을 찾는 단계이다. 따라서 제안 방법은 [5]와 달리 주어진 배열들을 한 번만 스캔하고도 빈발 부분 구역 집합들을 탐색할 수 있다.

제안 방법의 첫 번째 단계는 주어진 배열들 내에 존재하는 부분 구역들을 탐색하는 한편, 동시에 그들의 빈도수를 측정하는 단계이다. 어떤 배열 내에 존재하는 부분 구역들을 탐색하기 위해, 제안 방법은 해당 배열의 첫 번째 행부터 마지막 행까지 차례대로 탐색한다. 제안 방법은 각 행에 대해서 먼저 해당 행 내에서 서로 연속되어 있으면서 서로 같은 값을 가지는 원소들로 구성된 1차원의 부분 배열을 찾는다. 본 논문에서는 이를 행 단위 부분 구역이라 부른다. 각 행 단위 부분 구역은 R_{j_k,j_l}과 같이 표시하며, 여기에서 k와 l은 각각 R_{j_k,j_l}이 해당 행 내에서 시작되는 시작 열과 끝나는 종료 열을 나타낸다. 그림 3은 하나의 행 내에서 행 단위 부분 구역 R_{j_1,j_2}와 R_{j_3,j_5}를 찾은 예이다.

어떤 행에서 행 단위 부분 구역이 탐색되면 이를 바로 위에 있는 행의 행 단위 부분 구역들과 비교한다. 만약 어떤 행 단위 부분 구역이 바로 위에 있는 행의 행 단위 부분 구역과 원소 값이 동일하고 열의

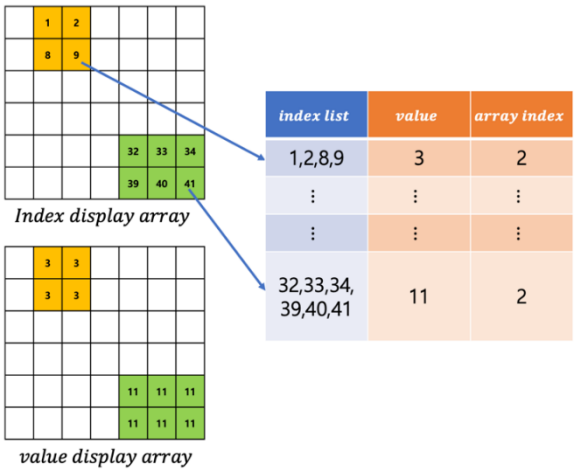


(그림 3) 행 단위 부분 구역의 예



(그림 4) 행 단위 부분 구역들의 결합 예

범위가 겹치면 이들을 결합하여 하나의 부분 구역으로 만든다. 그림 4는 어떤 배열에서 첫 번째 행의 행 단위 부분 구역들을 찾고, 그 후 두 번째 행의 행 단위 부분 구역들을 찾은 뒤, 이들을 결합하여 부분 구역을 확장하는 과정을 나타낸다. 제안 방법은 이러한 과정을 배열의 마지막 행까지 진행하여 해당 배열 내에 존재하는 모든 부분 구역을 탐색한다.



(그림 5) 부분 구역 정보 테이블의 예

상기 방법을 사용하여 주어진 배열들 내에 존재하는 부분 구역들이 탐색되면, 제안 방법은 이들에 대한 정보를 그림 5와 같은 형태의 테이블에 저장한다. 본 논문에서는 이 테이블을 부분 구역 정보 테이블이라 부른다. 부분 구역 정보 테이블의 각 행은 탐색된 각 부분 구역에 대한 3가지 정보를 담고 있다. 첫 번째는 해당 부분 구역을 구성하는 원소들의 배열 내 위치들이고, 두 번째는 해당 부분 구역을 구성하는 원소들이 동일하게 가지고 있는 값이고, 세 번째는 해당 부분 구역이 현재까지 등장한 배열들이다.

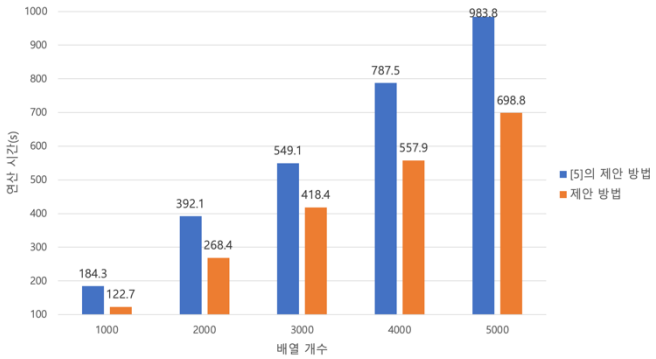
만약 어떤 배열에서 크기가 $Size_{min}$ 이상인 부분 구역이 탐색되면, 제안 방법은 해당 부분 구역의 정보를 부분 구역 정보 테이블에 적재한다. 이 때 부분 구역 정보 테이블에 이미 원소들의 위치와 원소 값이 동일한 부분 구역이 존재하면 해당 부분 구역에 대한

정보에서 해당 부분 구역이 등장하는 배열만 새로 추가한다. 만약 그렇지 않으면 새 부분 구역에 대한 정보를 등록한다. 이렇게 주어진 배열들에 대한 부분 구역들을 모두 탐색하여 부분 구역 정보 테이블이 구축되면, 마지막으로 빈도수가 $Support$ 이하인 부분 구역들에 대한 정보를 제거하여 최종 부분 구역 정보 테이블을 얻는다. 따라서 제안 방법은 주어진 배열들을 한 번만 스캔하고도 배열들 내부에 존재하는 크기가 $Size_{min}$ 이상이고 빈도수가 $Support$ 이상인 모든 부분 구역들을 탐색할 수 있다.

마지막으로 제안 방법은 모든 빈발 부분 구역들과 그들이 등장하는 배열들에 대한 정보를 담고 있는 부분 구역 정보 테이블을 기존의 $Apriori$ 알고리즘이나 $FP-Growth$ 알고리즘의 입력으로 넣어 최종적으로 빈발 부분 구역 집합을 찾는다.

5. 실험 결과

실험은 intel i7-9700 3.00GHz CPU 와 32 GB RAM 을 장착한 단일 컴퓨터에서 수행하였으며, 제안 방법과 [5]에서 제안한 방법을 비교하였다. [5]에서 제안한 방법은 먼저 주어진 배열들을 스캔하여 각 배열 내에 존재하는 부분 구역들을 찾는다. 이후 주어진 배열들을 다시 한 번 스캔하여 각 부분 구역의 빈도수를 측정하고, 이들 중 최소 빈도수 이상 등장하는 부분 구역들만 탐색한다. 이후 탐색된 빈발 부분 구역들에 대해 $Apriori$ 알고리즘이나 $FP-Growth$ 알고리즘을 사용하여 최종 빈발 부분 구역 집합을 찾는다. 본 실험에서는 제안 방법과 [5]에서 제안한 방법 모두 마지막 단계에서 $FP-Growth$ 알고리즘을 사용하도록 하였다.



(그림 6) 배열 개수 증가에 따른 실험 결과

본 실험에서는 배열의 크기를 100 × 100 으로 고정하고, 배열의 개수를 1,000 개에서 5,000 개까지 1,000 개 단위로 증가시켜가며 두 방법의 수행 시간을 비교하였다. 실험에서는 부분 구역의 최소 크기 $Size_{min}$는 2, 최소 지지도 $Support$는 4 로 설정하였다. 그림 6 은 두 방법의 수행 시간을 비교한 실험 결과이다. 실험 결과를 관찰해보면 기존 방법에 비해 제안 방법의 수행 시간이 약 30% 정도 감소했음을 볼 수 있다. 이것은 제안 방법이 기존 방법과 달리 주어진 배열들을 한 번만 스캔하고도 빈발 부분 구역들을 탐색할 수 있기 때문이다. 따라서 제안 방법은 기존 방법에 비해 빈발 부분 구역 집합을 효율적으로 탐색할 수 있음을 알 수 있다.

6. 결론

본 논문은 2 차원 배열 내 빈발 패턴 탐색을 위한 효율적인 기법을 제안하였다. 기존의 빈발 패턴 탐색 기법들은 각 데이터가 항목들의 집합으로 표현되는 경우를 주로 고려하며, 따라서 원소들의 위치관계까지 고려해야 하는 2 차원 배열 내 빈발 패턴 탐색 문제에 바로 적용하기는 어렵다. 따라서 본 연구의 선행 연구에서는 배열 내 존재하는 부분 구역들을 먼저 추출하고, 이들에 대해 빈발 패턴 탐색을 찾는 기법을 제안하였으나 주어진 배열들을 두 번 스캔해야 한다는 단점이 있었다. 이에 비해 본 논문의 제안 방법은 배열들을 한 번만 스캔하고도 배열들 내부에 존재하는 빈발 부분 구역들을 효율적으로 탐색할 수 있다. 실험 결과 제안 방법은 기존 방법에 비해 수행시간을 약 30% 단축함을 확인하였다. 추후 연구로는 부분 구역을 구성하는 원소들의 값이 동일하지 않은 더 일반적인 부분 구역들에 대한 빈발 패턴 탐색을 연구할 예정이다.

이 논문은 2020 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No.2018R1D1A1B07045643).

참고문헌

[1] Agrawal, R. and Srikant, R. “Fast algorithms for mining association rules,” In Proc. of the 20th VLDB Conference, pp.487-499, 1994.

[2] X Yuan, “An improved Apriori algorithm for mining association rules,” In AIP Conference Proceedings, vol. 1820, no. 1, 2017.

[3] SS Maw, “An Improvement of FP-Growth Mining Algorithm Using Linked list,” In Proc. of the Eighteenth International Conference On Computer Applications (ICCA 2020), 2020.

[4] Christian Borgelt, “An Implementation of the FP-growth Algorithm”, In OSDM’05 Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations, pp. 1-5, 2005.

[5] 김한슬, 이기용, “2 차원 배열 내 구역 간 연관 규칙 탐색을 위한 효율적인 탐색 기법,” 한국소프트웨어종합학술대회(KSC 2020), pp.59-61, 2020.