데이타마이닝에서 고차원 대용량 데이타를 위한 셀-기반 클러스터링 방법

(A Cell-based Clustering Method for Large High-dimensional Data in Data Mining)

진 두 석 † 장 재 우 ††
(Du-Seok Jin) (Jae-Woo Chang)

요 약 최근 데이타마이닝 응용분야에서는 고차원 대용량 데이타가 요구되고 있다. 그러나 기존의 대부분의 데이타마이닝을 위한 알고리즘들은 소위 차원의 저주(dimensionality curse)[1] 문제점과 이용 가능한 메모리의 한계 때문에 고차원 대용량 데이타에는 비효율적이다. 따라서, 본 논문에서는 이러한 문제점을 해결하기 위해서 셀-기반 클러스터링 방법을 제안한다. 제안하는 셀-기반 클러스터링 방법은 고차원 대용량 데이타를 효율적으로 처리하기 위한 셀 구성 알고리즘과 필터링에 기반한 저장인덱스 구조를 제공한다. 본 논문에서 제안한 셀-기반 클러스터링 방법을 CLIQUE 방법과 클러스터링 시간, 정확율, 검색시간 관점에서 성능을 비교한다. 마지막으로, 실험결과 제안하는 셀-기반 클러스터링 방법이 CLIQUE 방법에 비해 성능이 우수함을 보인다.

Abstract Recently, data mining applications require a large amount of high-dimensional data. Most algorithms for data mining applications, however, do not work efficiently for high-dimensional large data because of the so-called 'curse of dimensionality'[1] and the limitation of available memory. To overcome these problems, this paper proposes a new cell-based clustering method which is more efficient than the existing algorithms for high-dimensional large data. Our clustering method provides a cell construction algorithm for dealing with high-dimensional large data and a index structure based on filtering. We do performance comparison of our cell-based clustering method with the CLIQUE method in terms of clustering time, precision and retrieval time. Finally, the results from our experiment show that our cell-based clustering method outperform the CLIQUE method.

1. 서 론

데이타마이닝은 대용량의 데이타베이스에서 숨겨진 지식, 패턴, 연관된 규칙 등을 발견하는 방법이다[2]. 이러한 방법들을 통해 기업들이 보유한 기존의 경험적 지식을 재확인하는 역할을 수행함과 동시에 경영 의사결정의 과정을 지원하고 그 결과를 예측할 수 있다. 데이타 마이닝은 기계학습, 통계학, 인공지능을 포함한 다른 연구 결과로부터 발전된 최신의 데이타 분석 기술이다. 하지만, 데이타마이닝은 기존의 방법들과 다른 특징을 가지고 있다. 첫째, 기존 방법들은 주로 고정된 데이타 집합에 적용하는데 반해 데이타 마이닝은 계속적으로 삽입과 삭제가 이루어지는 동적인 데이타 집합에 적용한다. 둘째, 기존 방법들은 정제과정을 통해 정확하고 오류가 없는 데이타를 처리하지만, 데이타마이닝은 정제과정을 거치지 않은 오류가 있거나 값이 누락된 데이타를 처리한다. 셋째, 기존 방법의 데이타는 대략 수 천개 정도 밖에 되지 않으나 데이타마이닝은 수 백만개 이상의 대용량 데이타를 처리한다. 대표적인 데이타 마이닝 기술은 분류(classification)기법, 클러스터링(clustering) 기법, 연관 규칙(association rule) 그리고 경향분석(trend analysis)등이 있다. 이 중에서 특히, 클러스터링 기법은 데이타 마이닝의 중요한 분야 중 하나이다.

· 본 논문은 전북대학교 전기전자회로합성연구소 지원에 의하여 수행되었습니다.
† 비 회 원 : 전북대학교 컴퓨터공학과
dsjin@dblab.chonbuk.ac.kr
†† 종신회원 : 전북대학교 컴퓨터공학과 교수
jwchang@dblab.chonbuk.ac.kr
논문접수 : 2000년 12월 21일
심사완료 : 2001년 7월 12일

클러스터링 기법은 클러스터들 사이에 계층구조를 가지고 있는 계층적(hierarchical) 클러스터링과 데이타 공간을 정해진 수의 클러스터에 따라 분할하고 클러스터링하는 공간분할(space partitioning) 클러스터링 방법이 있다. 그리고 클러스터를 결정하는 기준은 거리(distance)를 사용하는 방식과 밀도(density)를 사용하는 방식으로 구분된다. 기존의 클러스터링 알고리즘에 관한 연구는 데이타 셋이 메모리 상주(memory resident)데이타로 제한되어 있었기 때문에 수 백만건 이상을 가진 대용량의 데이타베이스에는 적합하지 못한 단점이 있다. 또한, 기존의 연구는 저차원 데이타의 클러스터링에는 적합하지만 데이타 셋의 차원이 높아질수록 급격한 성능저하를 보이고 있어 고차원 데이타에는 부적합한 단점이 있다. 최근에는 대용량의 고차원 데이타가 증가하고 있기 때문에 이러한 데이타를 효율적으로 처리할 수 있는 클러스터링방법이 요구되고 있다.

따라서, 본 논문에서는 데이타 공간을 각 차원별 구간으로 분할하는 공간분할 방식과 각 분할구간의 밀도를 구하여 처리하는 밀도기반 방식으로 대용량 고차원의 데이타를 효율적으로 처리할 수 있는 방법을 제안한다. 제안하는 방법은 공간상에서 분할 인덱스를 통해 각 차원의 구간을 정하고 각각 정해진 구간의 겹침 영역에 따라 셀을 구성하며, 구성된 셀의 밀도가 임계값 이상인 셀에 대해서 클러스터링을 처리하는 방법으로 고차원 대용량 데이타에 효율적인 셀 구성 알고리즘이다. 또한 빠른 검색을 위해서 근사 방법(approximation method)을 적용하며 이를 위해서 필터링 기법을 이용한 저장 인덱스 구조를 제안한다.

본 논문의 구성은 다음과 같다. 제2장에서는 기존의 클러스터링 방법에 대해 소개하고 제3장에서는 본 논문에서 제안하는 셀-기반 클러스터링 방법의 셀 구성 알고리즘과 클러스터링 결과를 저장하고 검색하는 저장 인덱스 구조에 대해 설명한다. 제4장에서는 셀-기반 클러스터링 방법의 성능을 제시한다. 마지막으로 5장에서는 결론을 제시한다.

2. 기존 클러스터링 방법

클러스터링이란 유사한 특성을 갖는 데이타들을 함께 그룹화하고 분할화하는 방법이다. 일반적으로 클러스터링 방법은 거리 기반 클러스터링 방법과 밀도 기반 클러스터링 방법으로 구분할 수 있다. 거리 기반 클러스터링 방법으로는 CLARANS, BIRCH 방법이 있고, 밀도기반 클러스터링 방법으로는 DBSCAN, STING, CLIQUE 방법등이 대표적이다.

CLARANS(Clustering Large Applications based upon RANdomized Search)[3]알고리즘은 PAM(Partition Around Medoids)과 CLARA(Clustering LARge Applications)를 결합한 방법이다[4]. PAM 알고리즘은 데이타의 개수가 n개일때, 각 클러스터를 위한 대표 객체를 먼저 발견하여 k개의 클러스터를 찾는다. 클러스터에서 가장 중앙에 위치한 대표 객체 "medoid"를 구한다. k개의 medoid들을 결정한 후에 medoid와 일반 다른 객체들로 이루어진 모든 가능한 객체들의 쌍을 분석하면서 반복적으로 최선의 modoid들을 선정한다. 그러나, PAM 알고리즘은 모든 데이타를 검색하기 때문에 대용량 데이타에 매우 비효율적이다. CLARA 알고리즘은 모든 데이타를 사용하지 않고 샘플링방법을 사용하여 대표 집합을 추출한 후 대표 집합에 PAM 알고리즘을 적용하므로써 최선의 클러스터를 추출한다. 따라서, 이 방법은 무작위로 추출된 샘플이 적절하다면 매우 효과적인 방법이며 PAM보다 대용량의 데이타를 처리할 수 있는 장점을 가지고 있다. 따라서, PAM 과 CLARA 알고리즘을 결합한 CLARANS 알고리즘은 두 방법보다는 효율적이지만, 모든 객체가 메모리에 적재되어야 한다는 단점을 가지고 있다. 그러므로, 효과적인 샘플링 방법이 필요하며 샘플링 결과에 따라 CLARANS 알고리즘의 성능이 결정된다.

BIRCH(Balanced Iterative Reducing and Clustering using Hierarchies)에서는 대용량의 데이타 집합을 위해서 CF(Clustering Feature)[5]와 CF-tree를 이용한다. CF는 모든 점들을 저장하는 대신에 서브 클러스터의 요약정보를 사용하며, CF-tree는 B(Branching factor)와 T(Threshold)값을 가진 균형 트리로서 CF값을 저장한다. CF-tree의 중간노드는 그 노드의 모든 서브 클러스터들로 구성된 하나의 클러스터를 나타내며 리프 노드에 있는 각각의 엔트리의 지름은 T보다 작은 값을 가지고 있어야 한다. 따라서, T값에 따라 트리의 크기가 결정된다. CF-tree는 데이타의 삽입과정에서 동적으로 구축되며 데이타 점들을 한번만 읽어서 트리를 구성할 수 있다는 장점이 있다.

밀도에 근거한 클러스터링 알고리즘인 DBSCAN (Density Based Spatial Clustering of Applications with Noise)[6][7]은 주어진 반지름(ε) 값을 갖는 클러스터안에 데이타 점들의 밀도가 적어도 MinPtr 이상되어야 한다. 이를위해 DBSCAN은 Eps-neighborhood와 MinPts를 정의한다. Eps-neighborhood($N_{Eps}(p)$) = { $q \in D | dist(p,q) \leq Eps$ }이며, $|N_{Eps}(p)| \geq$ MinPtr 의 조건을 만족해야 한다. 여기서 dist(p,q)는 두점 p,q 사이의 거리를 나타내며 D는 데이타들의 집합이고

$N_{Eps}(p)$는 점 p의 Eps-neighborhood를 나타낸다. DBSCAN의 특징은 저밀도 지역을 이루는 잡음을 클러스터로부터 제거함으로서 잡음이 있는 데이타 집합을 효과적으로 처리할 수 있고 단 하나의 입력 매개변수만을 필요로 하는 장점이 있다.

STING(STatistical INformation Grid)[8]알고리즘은 통계 정보 그리드 방법으로 데이타 공간을 사각형의 셀로 계층분할을 한다. 각 상위 레이어는 상수 k개의 셀로 분할된다. 예를들어, k=4인 경우 1번째 레이어에서는 오직 하나의 셀만을 가지고 있고, 2번째 레이어는 4개의 셀로 분할되며, i번째 레이어에서는 (i-1)*k개의 셀들을 가진다. 각각의 셀들은 다음과 같은 6개의 매개변수를 갖는다. n은 셀에 속한 객체의 개수이고 m은 셀에 속한 모든 값들의 평균이며, s는 셀에 속한 모든 값들의 표준편차이다. min과 max 는 각각 셀의 최소값 및 최대값이며, distr은 각 셀의 애트리뷰트의 분산타입을 나타내며, "normal", "uniform", "exponential", "NONE" 등의 값을 갖는다. STING은 시간 복잡도 O(b)로서 b << n 일 때 높은 효율성을 갖는다. b는 최하위 레벨의 셀의 수이다. 그러나 고차원의 경우 b의 값도 커지므로 고차원의 데이타에 대해 비효율적인 단점이 있다.

마지막으로, CLIQUE(CLustering In QUEst)[9]방법은 관련된 차원만을 선택하여 선택된 부분공간에서 클러스터링을 처리하는 방법으로 고차원 데이타에서 큰 밀도를 가진 영역을 찾는 효과적인 방안을 제시하였다. CLIQUE는 밀도 및 격자 기반 클러스터링 방법이다. 즉 하나의 클러스터는 주위보다 높은 밀도를 가진 영역을 의미한다. CLIQUE는 데이타 점들의 근사 밀도값을 구하기 위해서 각 차원을 일정한 간격(ξ)으로 분할하고, 각 차원에서 이러한 분할 공간의 교차-곱으로 이루어진 Unit에 포함된 점들의 수가 기준밀도(τ)를 초과하면 밀집(dense)하다고 정의한다. 따라서, 각 클러스터는 서로 인접한 밀집 Unit들의 집합을 의미한다. 이와 같이 CLIQUE는 부분공간 클러스터링 방법을 이용하여 고차원 데이타 공간에서 차원의 감소를 시도하여 고차원적인 데이타의 클러스터링에 효과적인 방법을 제시하였으나, 각 차원을 일정한 간격으로 분할하기 때문에 클러스터의 정확한 형태를 표현하기 어려운 단점이 있다.

3. 셀-기반 클러스터링방법

기존 클러스터링 알고리즘은 메모리의 양에 제한적인 알고리즘으로서 대용량의 데이타를 처리할 수는 있지만 효율적이지 못하거나 또는 차원이 증가함에 따라 셀의 수가 지수적으로 증가하기 때문에 급격한 성능 저하를 보이고 있다. 따라서, 이러한 문제를 해결하기 위하여 셀의 밀도에 기반한 기존의 클러스터링 방법을 확장하여 대용량 고차원 데이타에 효율적인 확장된 셀-기반 클러스터링 방법을 제안한다. 제안하는 셀-기반 클러스터링 방법은 두 가지 특성을 지니고 있다. 첫째, 고차원 대용량 데이타의 효율적인 셀 구성 알고리즘을 위해 공간상에서 분할 인덱스(split index)를 통해 각 차원의 구간을 정하고 각각 정해진 구간의 겹침 영역에 따라 셀을 구성하며, 구성된 셀의 밀도가 임계값 이상인 셀에 대해서 클러스터를 구하는 방법을 사용한다. 둘째, 빠른 검색 처리를 위해서 근사 방법을 적용하며 이를 위해서 필터링 기법을 이용한 저장 인덱스 구조를 사용한다. 따라서, 셀-기반 클러스터링 방법은 크게 고차원 대용량 데이타의 클러스터링에 효율적인 셀 구성 알고리즘과 구성된 셀의 정보를 효율적으로 저장 및 검색하기 위한 저장 인덱스 구성 알고리즘 두 부분으로 구성된다. 셀 구성 알고리즘은 분할 인덱스에 기반한 공간 분할 방법을 사용하며, 저장 인덱스 구성 알고리즘을 위해서는 고차원의 데이타에 적합한 순차 탐색 색인기법을 사용한다. 셀-기반 클러스터링 방법의 전체적인 구조는 그림 1과 같다.

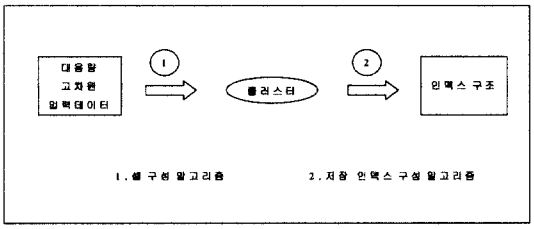

그림 1 셀-기반 클러스터링 방법의 전체 구조

3.1 셀 구성 알고리즘

셀 구성 알고리즘은 분할 인덱스에 따라 분할 구간을 설정하여 각 구간의 겹침 영역을 이용하여 셀을 구성한다. 분할 구간 설정과정에서 사용되는 분할 인덱스는 앤트로피에 기반한 방법으로 다중 그룹의 데이타를 분할하는데 효율적인 분할 인덱스를 사용한다. 셀 구성 알고리즘은 다음과 같다. 각각의 애트리뷰트의 최소값과 최대값을 가지고 반복적으로 최적의 분할 구간을 구하여 각 차원의 분할 구간의 겹침영역으로 이루어지는 셀을 구한다. 이때 반복조건은 최대값과 최소값의 차이가 1이상이거나 분할 후의 분할 인덱스 값이 분할 전의 인덱스 값보다 커야한다. 분할 인덱스 값은 주어진 입력 값에 의해서 분할 할 경우 식(1)에 의해서 그 값이 계산된다[10].

이 방법은 앤트로피에 기반한 방법으로 일반적으로 다중 그룹의 데이타를 분할하는데 효율적인 데이타 분할 방법으로 기존의 연구에서 많이 사용되고 있다. 분할 전, 후의 인덱스 값은 식(2)와 같이 구한다[10].

$$Split(S) = 1 - \sum_{j=1}^{c} p_j^2 \quad (1)$$

$$Split(S) = \frac{n_1}{n} Split(S_1) + \frac{n_2}{n} Split(S_2) \quad (2)$$

식(1)의 분할 인덱스는 데이타 셋 S를 c개의 클래스에 대하여 클래스의 상대적인 밀도 p_j를 구하여 제곱한 값을 모두 합한 결과를 가지고 결정한다. 만약 데이타 셋 S가 S_1과 S_2로 분할되면, 분할 인덱스 값은 식(2)와 같이 구해진다. 여기서 분할 인덱스 값이 분할 전보다 크면, 각각의 분할된 구간에 존재하는 데이타의 순수성이 분할 전보다 높다는 것을 의미한다. 따라서 이 때는 데이타 셋 S를 S_1과 S_2로 분할하는 것이 바람직하다. 한편 만약 분할 인덱스 값이 분할 전보다 크지 않으면, 분할을 수행하지 않고 분할 과정을 마친다. 결과적으로 본 논문에서 제안하는 셀 구성 알고리즘은 기존 알고리즘의 등 간격 셀 구성 알고리즘에 비해 매우 적은 수의 셀이 만들어지며 또한 기존 방법의 단점인 서로 유사한 데이타들이 서로 다른 셀에 포함되는 문제점을 해결하였다. 그림 2는 셀 구성 알고리즘을 나타낸다.

Procedure Make-Cell (attributes , input data set S)

Begin

1. **For** each attribute in S **do**
2. **Partition**(attribute, S)

End

Sub-Procedure Partition (attribute, data set S)

Begin

1. **For** each split_points in attribute **do**
2. **Compute** $\frac{n_1}{n}\left(1-\sum p_{j1}^2\right)+\frac{n_2}{n}\left(1-\sum p_{j2}^2\right)$
3. **If** $1-\sum p^2 > MAX\left[\frac{n_1}{n}\left(1-\sum p_{j1}^2\right)+\frac{n_2}{n}\left(1-\sum p_{j2}^2\right)\right]$ **then**
4. **return**
5. **Else**
6. **Split** S into S_1 and S_2 by max split_point
7. **Partition**(attribute, S_1)
8. **Partition**(attribute, S_2)

End

그림 2 셀 구성 알고리즘

그림 3은 셀 구성 알고리즘의 예이다. 입력 데이타가 2차원이고 2개의 클래스를 포함한 20개의 레코드 셋을 분할하는 과정을 설명하며, 그림 4에서 굵은 선으로 나타난 부분이 분할구간이 된다. 분할 과정을 살펴보면, 먼저 x축의 경우 10개의 구간에 대한 분할 인덱스 값을 모두 구하여 분할 인덱스 값이 최대치인 경우가 분할 축으로 선택된다. 예를 들면, 0.4와 0.5일 때 분할 인덱스 값은 각각 0.4인 경우는 0.475 이고 0.5인 경우는 0.501이다. 따라서 최대 분할 축인 0.5와 분할전의 값을 비교해보면 분할 후의 값이 크기 때문에 처음 분할 축은 0.5가 된다. 따라서, x축은 0~0.4 구간과 0.5~1.0사이의 구간으로 나뉘어져서 각각에 분할 과정을 반복한다. 분할 인덱스 값은 아래와 같이 구해진다.

i. X축 분할 전

$$Split(S) = 1 - \sum p_j^2$$

$$1-\left(\left(\frac{10}{20}\right)^2+\left(\frac{10}{20}\right)^2\right)=0.5$$

ii. X축이 0.4인 경우

$$Split(S) = \frac{n_1}{n} Split(S_1) + \frac{n_2}{n} Split(S_2)$$

$$\frac{4}{20}\left(1-\left(\left(\frac{3}{4}\right)^2+\left(\frac{1}{4}\right)^2\right)\right)+\frac{16}{20}\left(1-\left(\left(\frac{7}{16}\right)^2+\left(\frac{9}{16}\right)^2\right)\right)$$

$$=0.475$$

iii. X축이 0.5인 경우

$$\frac{9}{20}\left(1-\left(\left(\frac{4}{9}\right)^2+\left(\frac{5}{9}\right)^2\right)\right)+\frac{11}{20}\left(1-\left(\left(\frac{6}{11}\right)^2+\left(\frac{5}{11}\right)^2\right)\right)$$

$$=0.501$$

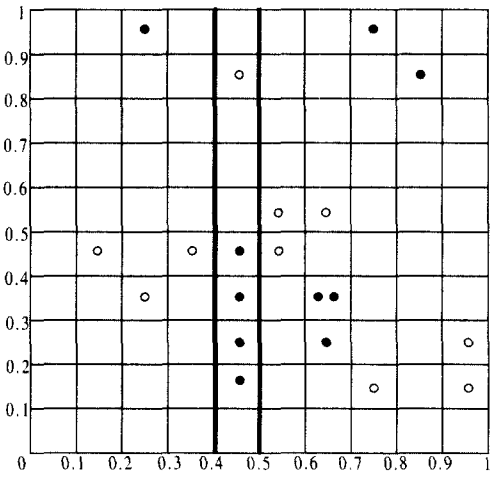

그림 3 셀 구성 알고리즘의 예

만약 n차원인 경우 각 차원에 대한 초기 분할구간이 m인 경우는 기존 셀 구성 알고리즘은 m^n 개의 셀이 구성되지만 제안하는 셀 구성 알고리즘은 K_1*K_2* ... K_n,($1 \le K_1, \ldots K_n \le m$)개의 셀이 구성된다. 예를들어, 그림 4의 데이타 셋은 각 차원이 10개의 구간으로 분할되는 경우 기존 셀 구성 알고리즘은 10*10개의 셀이 만들어진다. 그러나 제안하는 셀 구성 알고리즘은 8*4개의 셀이 만들어진다. 그림 4는 그림 3의 데이타를 기존 셀 구성 알고리즘과 제안하는 셀 구성 알고리즘에 의해 구성된 셀의 결과를 나타낸다.

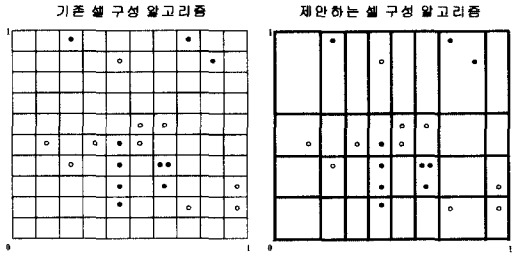

그림 4 셀 구성 결과의 예

3.2 저장 인덱스 구조

입력 데이타 셋의 클러스터는 셀 구성 알고리즘을 통해 구해진다. 구해진 클러스터 정보를 저장하는 방법으로 본 논문에서는 필터링 기법을 이용한 저장 인덱스 구조를 제안한다. 고차원 대용량 데이타의 경우 구성된 클러스터의 수가 매우 많아지기 때문에 검색 과정에 많은 시간이 소요된다. 따라서 이러한 문제점을 보완하기 위해서 본 논문에서는 근사 정보파일을 사용한다. 근사정보파일을 사용함으로서 크기가 매우 큰 클러스터 정보파일에 대한 검색 횟수를 줄여 전체 검색 성능을 향상시킬수 있다. 필터링에 기반한 2단계 저장 인덱스구조는 클러스터 정보를 각각 근사 정보파일(approximation information file)과 클러스터 정보파일(cluster information file)에 저장한다. 셀-기반 클러스터링 알고리즘에 의해 K개의 유용한 클러스터가 구해졌을 경우, X축은 n개의 구간으로 분할되고 Y축은 m개의 구간으로 분할되는 것을 가정할 때, 근사정보파일을 사용할 경우와 사용하지 않았을 경우에 입력 데이타에 대한 검색 처리시간(C)은 다음과 같다. 단, α는 근사정보파일에서 필터링되는 평균 비율이며 D는 입력데이타의 차원 수이고 P는 페이지당 레코드수를 나타낸다. 그리고 R은 차원당 평균 레코드수를 나타낸다.

i) 근사 정보파일 사용하지않은 경우 처리시간

$$C = \lceil K/P \rceil /2 \text{ (I/O수)}$$

ii) 근사 정보파일 사용시 처리시간

$$C = \lceil (D*R)/P \rceil *\alpha + (1-\alpha) \lceil K/P \rceil /2 \text{(I/O수)}$$

일반적으로 K는 매우 큰 값을 가지므로 근사 정보파일을 사용하는 경우 필터링된 평균 비율 즉, α값이 클수록 처리시간 C는 작아진다. 또한 차원값, D에 비례하여 K값이 지수적으로 커지므로 고차원인 경우에 근사 정보파일을 사용한 2단계 저장 인덱스 구조가 효율적이다. 필터링에 기반한 2단계 저장 인덱스 구조의 구체적인 구성 과정은 다음과 같다. 첫째, 모든 차원에 대한 각각의 구간별 빈도수를 구하여 구간 임계값 이상인 구간과 미만인 구간을 구분하여 근사 정보파일에 0과 1로서 저장하고, 클러스터 정보파일에는 각 셀에 대해 빈도수를 구하여 빈도수가 셀 임계값 이상인 셀의 번호 및 빈도수를 저장한다. 셀 임계값과 구간 임계값은 식(3)과 같다.

$$\text{구간임계값 } (\lambda) = \begin{cases} \lambda = \dfrac{NR}{NI} \times F \\ NI : \text{입력데이터의수} \\ NR : \text{각 차원별 구간 수} \\ F : \text{'1'로 받아들이는 구간내 최소빈도수} \end{cases}$$

$$\text{셀임계값 } (\tau) : \text{양의정수값} \qquad (3)$$

만들어진 셀의 클러스터 정보를 가지고 저장 인덱스를 구성하는 알고리즘은 다음과 같다. 우선, 셀 임계값 이상인 셀을 클러스터 정보파일(cluster information file)에 저장하고 또한, 근사 정보파일(approximation information file)에 저장한다. 그림 5는 저장 인덱스 구성 알고리즘을 나타내며, 그림 6은 K개의 클러스터가 주어졌을 때 저장 알고리즘을 통해 만들어진 저장 인덱스 구조를 나타낸다.

```
Procedure Insert Cell ( cells )
Begin
1.    For each cells which is made form make cell do
2.        Compare the cell-threshold with cell density
3.        If cell density > cell-threshold then
4.            Insert  cell information into cluster info file

5.        Compare the volume-threshold with volume density
6.        If volume density > volume-threshold then
7.            approximation info file[volume] = 1
8.        else
9.            approximation info file[volume] = 0
End
```

그림 5 저장 인덱스 구성 알고리즘

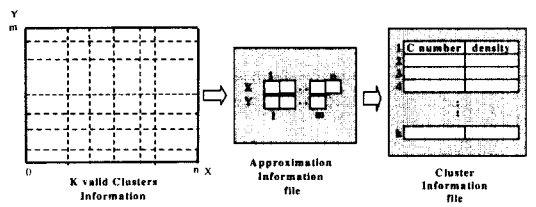

그림 6 저장 인덱스구조

셀-기반 클러스터링 방법의 질의 처리 방법은 새로운 데이타의 입력시 우선 각 차원의 해당구간을 구한다. 모든 차원의 해당구간이 근사 정보파일의 1인 구간으로 나타나는 경우 셀 번호를 구해서 클러스터 정보파일을 검색하고, 그렇지 않은 경우 클러스터 정보파일을 검색하지 않음으로서 검색성능을 향상시킨다. 차원이 증가할수록 적어도 하나의 차원에 대하여 근사 정보파일의 값이 0 이 나올 확률이 커지므로 고차원의 데이타에서 크기가 큰 클러스터 정보파일을 검색하는 횟수를 줄일 수 있다. 그림 7은 셀 임계값과 구간 임계값이 1 인 경우에 구성된 근사 정보파일과 클러스터 정보파일에서 질의 레코드를 처리하는 과정을 보여준다.

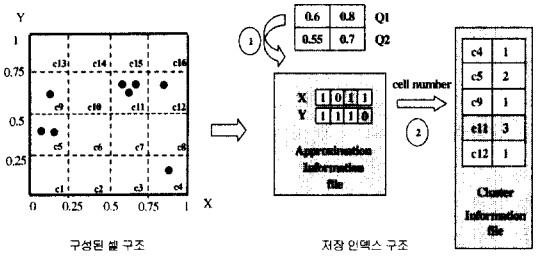

그림 7 질의처리 과정

질의 레코드가 Q1인 경우 X축 0.6의 값은 3번째 구간으로 구분되고 Y축의 0.8의 값은 4번째 구간으로 구분된다. 이때 근사정보파일의 내용을 보면 X축의 값은 1에 해당하고 Y축의 값은 0에 해당한다. 만약 하나이상의 0에 해당하는 구간이 나오면 질의 레코드는 의미 없는 레코드로 처리되어 클러스터정보파일을 검색하지 않고 단계 1에서 끝난다. 따라서 Q1의 경우 1단계에서 의미 없는 레코드로 처리된다. 질의 레코드가 Q2인 경우에는 X축 0.55의 값과 Y축 0.7은 모두 3번째 구간으로 구분된다. 이때 근사 정보파일의 내용을 보면 모두 1에 해당하는 값을 가진다. 따라서 이 경우 셀의 번호를 구하고 클러스터 정보파일을 순차검색 하여 매칭된 셀의 번호를 찾아 클러스터 결과와 빈도수를 구한다. 따라서 Q1의 경우 1단계에서 의미 없는 레코드로 처리되며 Q2의 경우 클러스터정보파일에서 셀 번호 11의 값에 해당하는 정보를 반환한다.

4. 구현 및 성능 평가

본 장에서는 제안된 셀-기반 클러스터링 방법을 구현하여 성능 평가를 수행한다. 실험을 통하여 제안된 셀-기반 클러스터링 방법이 고차원 대용량 데이타에 대한 처리가 효율적임을 보이고, CLIQUE 방법과의 성능을 비교한다. 또한 각 차원의 최대 구간값과 데이타 셋의 분포에 따른 성능을 측정한다.

4.1 성능평가기준

실험에 사용된 시스템 환경은 CPU 650 MHz dual, 메모리 512MB의 리눅스 서버에서 수행하였다. 사용된 데이타는 IBM Quest Data mining project[11]에서 사용한 Synthetic Data Generation Code for Classification 을 이용하여 각각 8차원, 16차원의 100만 건의 데이타를 만들어 사용하였다. 8차원의 데이타는 salary, commission, age, elevel, zipcode, hvalue, hyears, loan으로 구성된 레코드 셋을 사용하였고 16차원의 데이타는 8차원의 데이타에 area, children, tax, interest, ctype, cyear, job, balance가 추가된 셋이다. 여기에서, salary, commission, age, hvalue, hyears, loan, tax, interest, cyear, balance는 수치(numeric) 애트리뷰트에 속하고 elevel, zipcode, area, children, ctype, job은 범주(categorical) 애트리뷰트에 속한다. 따라서, 8차원과 16차원의 데이타 모두 수치 애트리뷰트와 범주 애트리뷰트를 모두 포함한 데이타 셋이다. 표 2는 본 논문에서 사용한 실험 데이타를 나타낸다.

아울러 실험에 사용된 데이타 셋은 세 가지로, 랜덤 분포를 따르는 데이타 셋, 분산이 1인 표준 정규분포를 따르는 데이타 셋, 마지막으로 skewed 된 데이타 셋을 위해 분산이 0.5인 정규분포를 따르는 데이타 셋을 사용한다. 또한 구간값에 따른 실험을 위하여 수치 애트리뷰트의 최대 구간값을 각각 5와 10을 사용한다. 실험에 사용된 데이타의 분포와 구간값에 따른 실험 방법들은 표 3과 같다. 표 3에서와 같이 본 논문에서 제안한 셀-기반 클러스터링방법(CBCM)을 CLIQUE 방법과 세 가지의 데이타 셋과 두가지 구간값에 대하여 성능을 비교한다. 비교대상으로 CLIQUE 방법을 선택한 것은 기존의 클러스터링 방법 가운데 CLIQUE 방법이 고차원 대용량 데이타 처리에 가장 효율적이기 때문이다. 아울러 성능비교는 클러스터링 시간(클러스터 생성시간), 검색시간, 정확율의 측정을 통해 수행된다.

표 2 실험데이타의 애트리뷰트 속성

Attribute	Description	Value
salary	salary	uniformly distributed from 20000 to 150000
commission	commission	salary >= 75000 => commission =0 else uniformly distributed from 10000 to 75000
age	age	uniformly distributed from 20 to 80
elevel	education level	uniformly chosen from 0 to 4
zipcode	zip code of the town	uniformly chosen from 9 available zipcodes
hvalue	value house owned	uniformly distributed from 0.5k100000 to 1.5k100000 where k∈{0...9} depends on zipcode
hyears	years house owned	uniformly distributed from 1 to 30
loan	total loan amount	uniformly distributed from 0 to 500000
area	area code	uniformly chosen from 20 available area codes
children	the number of children	uniformly chosen from 0 to 4
tax	tax	salary < 60000 => tax =0 else = salary*0.01
interest	interest	loan < 100000 => interest = loan*0.01 else interest = loan*0.02
ctype	car types	uniformly chosen from 10 available car types
job	job types	uniformly chosen from 20 available job types
cyears	years car owned	uniformly distributed from 1 to 10
balance	total balance amount	uniformly distributed from 0 to 500000

표 3 데이타 분포와 구간값에 따른 실험방법

Methods	Description
CBCM-5R CLIQUE-5R	최대 구간값 : 5 분산 타입 : random data set
CBCM-10R CLIQUE-10R	최대 구간값 : 10 분산 타입 : random data set
CBCM-5SND CLIQUE-5SND	최대 구간값 : 5 분산 타입 : standard normal distribution data set
CBCM-10SND CLIQUE-10SND	최대 구간값 : 10 분산 타입 : standard normal distribution
CBCM-5ND(0.5) CLIQUE-5ND(0.5)	최대 구간값 : 10 분산 타입 : normal distribution (variation = 0.5)
CBCM-10ND(0.5) CLIQUE-10ND(0.5)	최대 구간값 : 10 분산 타입 : normal distribution (variation = 0.5)

4.2 성능평가

동등한 조건으로 제안하는 CBCM과 CLIQUE 방법의 클러스터링 시간을 측정하기 위하여, 먼저 구간 임계값은 0 으로 설정하고 클러스터링 시간을 측정한다. 그림 8은 8차원 100만건의 데이타를 클러스터링하는 시간을 나타낸다. 두 방법 모두 데이타가 증가함에 따라 클러스터링 시간이 선형적으로 증가함을 보이고 있다. 이것은 대용량의 데이타를 처리할 때 매우 바람직한 특성이다. 왜냐하면 데이타의 증가에 따른 클러스터링 시간이 지수적으로 증가하는 방법들은 대용량의 데이타베이스에는 적합하지 않기 때문이다. 실험 결과 100 만건의 데이타를 클러스터링하는 시간은, 세가지 데이타 셋의 분포에 따라 약간의 차이가 있지만, CLIQUE 방법은 약 350초 정도이고, 제안된 CBCM 방법은 약 50초 정도이다. 제안된 CBCM 방법이 성능이 현저하게 좋은 이유는, CBCM 방법이 CLIQUE 방법에 비해 매우 적은 수의 셀을 생성하며, 따라서 클러스터링 시간이 약 85%가 감소되기 때문이다. 그림 9는 16차원 데이타의 클러스터링 시간을 나타낸다. 16차원 데이타도 8차원과 비슷한 형태를 보이고 있으며 CLIQUE 방법은 약 650초, 제안하는 CBCM 방법은 약100초 정도 소요된다.

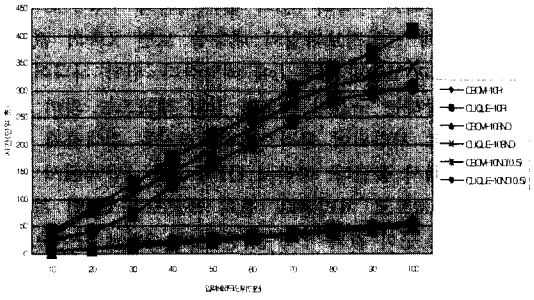

그림 8 8차원 데이타의 클러스터링 시간

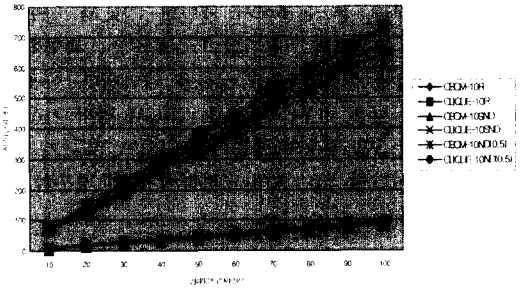

그림 9 16차원 데이타의 클러스터링 시간

그림 10은 8차원 100만건의 데이타를 클러스터로 구성한 후, 질의에 대한 평균검색 시간을 나타낸다. 최대 구간값이 10인 경우, 데이타 셋의 분포에 따라 차이가 있지만 CLIQUE 방법은 약 12~22초 정도 소요되며, 제안하는 CBCM 방법은 약 2초 정도 소요된다. CBCM 방법이 검색시간이 빠른 이유는 셀 구성 알고리즘을 통해 적은 수의 셀을 만들며, 또한 근사 정보파일을 사용하여 필터링 효과를 얻기 때문이다. 아울러 최대 구간값이 5인 경우는 만들어진 클러스터의 개수가 현저하게 적기 때문에, 데이타 셋의 분포에 따라 차이가 있지만 CLIQUE 방법은 약 5~9초 정도 소요되며, 제안하는 CBCM 방법은 약 1초 정도 소요된다. 또한, 데이타의 셋의 분포에 따른 차이를 살펴보면, 램덤 분포, 표준 정규분포, skewed 분포(분산 0.5인 정규분포) 순으로 검색시간이 줄어들며, 이는 데이타 셋의 분산이 적을수록 클러스터의 개수가 적어짐으로써 검색 시간이 줄어드는 결과를 보인다. 그림 11은 16차원 데이타에 대한 결과를 나타내며, 16차원도 8차원과 비슷한 형태의 성능을 보이고 있다.

그림 12와 13은 각각 8차원과 16차원에서 CLIQUE 방법과 제안된 CBCM 방법의 정확율을 나타낸다.

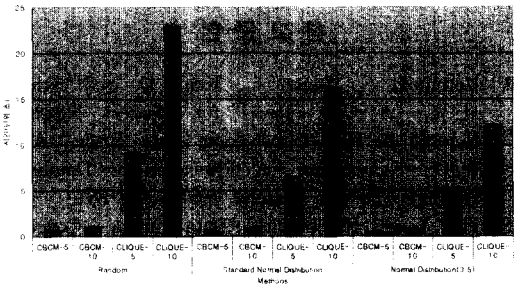

그림 10 8차원 데이타의 검색시간

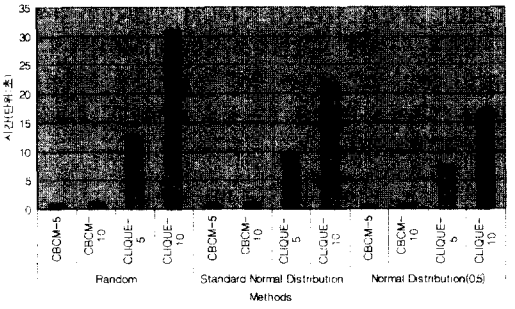

그림 11 16차원 데이타의 검색시간

CLIQUE 방법은 구간 임계값을 사용하지 않기 때문에 구간 임계값을 0으로 하여 정확율을 측정하였다. CLIQUE 방법은 모든 경우에서 약 95% 의 정확율을 보이며, 제안하는 CBCM 방법은 최대 구간값이 10인 경우에는 90% 이상의 정확율을 보이는 반면 최대 구간값이 5인경우에는 정확율이 약 80% 정도로 떨어진다. 이 이유는 최대 구간값이 5인경우에는 생성되는 클러스터의 개수는 적으나 클러스터의 정확성이 떨어지기 때문이다.

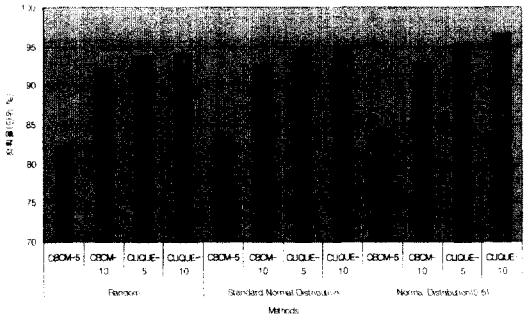

그림 12 8차원 데이타의 정확율

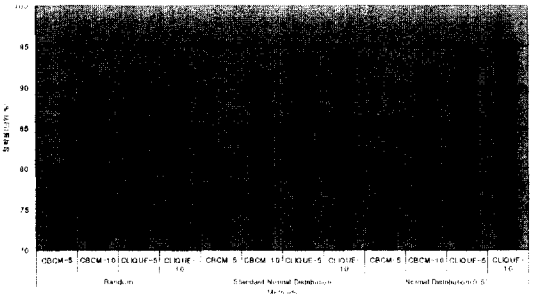

그림 13 16차원 데이타의 정확율

CLIQUE 방법과 제안하는 CMCM 방법간에 검색시간과 정확율 사이에 트레이드오프(trade-off)관계가 존재하므로, 단일 비교 인자를 통한 성능비교를 위하여 검색시간과 정확율을 결합한 효율성을 측정한다. 효율성은 식4와 같이 정의된다.

$$E_{MD} = W_p \cdot \frac{P_{MD}}{P_{MAX}} + W_t \cdot \frac{1}{\frac{T_{MD}}{T_{MIN}}} \quad (4)$$

E_{MD}는 표3의 방법(MD:method)들에 대한 검색시간과 정확율간의 효율성을 나타내고, 정확율에 대한 가중치는 W_p 그리고 검색시간에 대한 가중치를 W_t로 정의한다. 각각 P_{MD}와 T_{MD}는 각 방법들에 대한 정확율과 검색시간을 나타내며, P_{MAX}는 랜덤, 분산이 1인 표준 정규분포, 분산이 0.5인 정규분포를 따르는 데이타 셋에 대해 CBCM-5, CBCM-10, CLIQUE-5 그리고 CLIQUE-10 방법중 정확율이 가장 높은 값을 나타내며, T_{MIN}은 이들 방법들 가운데 검색시간이 가장 적은 값을 나타낸다. 실험 데이타는 16차원 100만 건의 데이타를 사용하였고, 정확율과 검색 시간에 대한 가중치는 둘의 가중치가 같은 경우와(W_p=W_t=0.5) 정확률의 가중치가 3배인 경우(W_p=0.75, W_t=0.25) 로 측정하였다. 그림 14와 15는 효율성에 대한 성능평가 결과를 나타낸다.

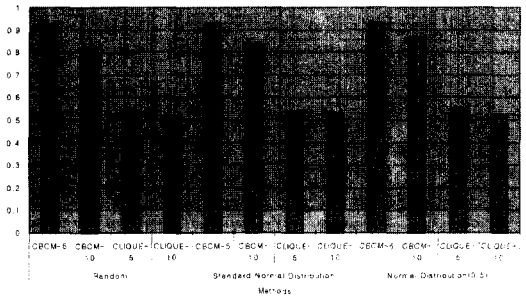

그림 14 효율성(Wp : Wt = 1 : 1)

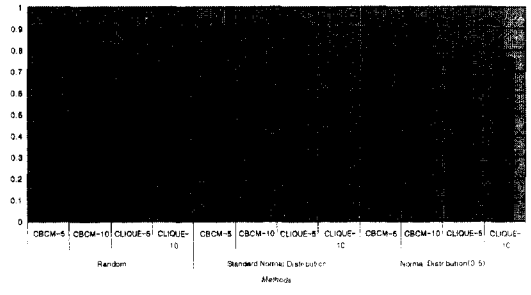

그림 15 효율성(Wp : Wt = 3 : 1)

성능평가 결과, 본 논문에서 제안한 CBCM 방법이 데이타 셋의 분포와 상관없이 효율성 측면에서 CLIQUE 방법에 비해 우수한 성능을 나타낸다. 아울러 W_p 와 W_t 가 같은 경우(그림 14), CBCM 방법 가운데 최대 구간값이 5인 경우의 성능이 우수하고, W_p 가 W_t의 3배인 경우(그림 15), 최대 구간값이 10인 경우의 성능이 우수하다.

5. 결 론

최근 데이타마이닝 분야에서 고차원 대용량 데이타에 대한 클러스터링 방법이 중요한 이슈가 되고 있다. 그러나 기존의 클러스터링에 방법들은 고차원 대용량 데이타를 효율적으로 처리하지 못하고 있다. 이러한 문제를 해결하기 위해서, 본 논문에서는 셀-기반 클러스터링 방법을 제안하였다. 셀-기반 클러스터링 방법의 두 가지 특징은 첫째, 공간상에서 분할인덱스를 통해 각 차원의 구간을 정하고 각각 정해진 구간의 겹침 영역에 따라 셀을 구성하며, 구성된 셀의 밀도가 임계값 이상인 셀에 대해서 클러스터링을 처리하는 방법으로 고차원 대용량 데이타에 효율적인 셀 구성 알고리즘을 제시하였다. 둘째, 빠른 클러스터링 처리를 위해서 근사 방법을 적용하며 이를 위해서 필터링 기법을 이용한 저장 인덱스구조를 사용한다. 성능평가를 위해 기존의 방법 중 고차원 대용량 데이타를 효율적으로 처리하는 CLIQUE 방법과 본 논문에서 제안하는 셀-기반 클러스터링 방법을 정확율과 클러스터링 시간 그리고 검색시간에 관하여 성능비교를 하였다. 성능평가 결과 본 논문에서 제안하는 셀-기반 클러스터링 방법은 정확율 측면에서는 약간 낮은 성능을 보이나 대용량 데이타를 처리하는데 중요한 클러스터링 시간이나 검색시간 측면에서는 우수한 성능을 보인다. 아울러, 검색시간과 정확율을 함께 고려한 효율성 지표에서도 본 논문에서 제안하는 클러스터링 방법이 CLIQUE 방법에 비해 우수한 성능을 나타낸다.

참 고 문 헌

[1] Berchtold S., Bohm C., Keim D. and Kriegel H.-P., "A Cost Model for Nearest Neighbor Serarch in High-Dimensional Data Space," ACM PODS Symposium on Principles of Databases Systems, Tucson, Arizona, 1997, pp.78-86.

[2] Han J. and Kamber M., "Data Mining : Concepts and Techniques," Morgan Kaufmann, 2000.

[3] Ng R.T. and Han J., "Efficient and Effective Clustering Methods for Spatial Data Mining," Proc. 20th Int. Conf. on Very Large Data Bases,

1994, pp.144-155.
[4] Kaufman L. and Rousseeuw P.J., "Finding Groups in Data : An Introduction to Cluster Analysis," John Wiley & Sons, 1990.
[5] Zhang T., Ramakrishnan R. and Linvy M., "BIRCH : An Efficient Data Clustering Method for Very Large Databases," Proc. ACM SIGMOD Int. Conf. on Management of Data, 1996, pp. 103-114.
[6] Ester M., Kriegel H.-P., Sander J. and Xu X., "A Density Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, 1996, pp.226-231.
[7] Ester M., Kriegel H.-P., Sander J. and Xu X., "Density-Connected Set and Their Application for Trend Detection in Spatial Databases," Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining, 1997, pp.10-15.
[8] Wang W., Yang J. and Muntz R., "STING : A Statistical Information Grid Approach to Spatial Data Mining," Proc. 23rd Int. Conf. on Very Large Data Bases, 1997, pp.186-195.
[9] Agrawal R., Gehrke J., Gunopulos D. and Raghavan P., "Automatic Subspace Clustering of High Dimensional Data Mining Applications," Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp.94-105.
[10] Breiman L., Friedman J. H., Olshen R. A. and Stone C. J., "Classification and Regression Trees," Wadsworth, Belmont, 1984.
[11] http://www.almaden.ibm.com/cs/quest.

진 두 석
1999년 전북대학교 컴퓨터공학과(공학사). 2001년 전북대학교 컴퓨터공학과(공학박사). 관심분야는 멀티미디어 데이타베이스, 멀티미디어 정보검색, 하부저장구조

장 재 우
정보과학회논문지 : 데이타베이스
제 28 권 제 2 호 참조