대용량의 고차원 데이터 공간에서 프로젝션 필터링 기반의 부분차원 클러스터링 기법

Partial Dimensional Clustering based on Projection Filtering in High Dimensional Data Space

이혜명(Hye Myung Lee)*, 정종진(Jong Jin Jung)**

초 록

현재 알려진 대부분의 클러스터링 알고리즘들은 고차원 공간에서 데이터가 갖는 고유의 희소성 및 잡음으로 인하여 성능이 급격히 저하되는 경향이 있다. 이에 따라 최근에 클러스터 형성에 연관성이 있는 차원만을 선택하고, 연관성이 적은 차원들을 제거함으로써 클러스터링의 성능을 높일 수 있는 부분차원 클러스터링 기법이 연구되고 있다. 그러나 현재 연구된 부분차원 클러스터링 기법은 그리드 기반 방법으로서 차원의 증가에 따라 그리드 셀의 수가 방대해짐으로써 공간 및 시간적인 효율성이 저하된다. 또한, 대부분의 알고리즘들은 데이터 집합에서 대표객체를 찾아 클러스터 형성에 관계있는 차원만을 조사하기 때문에 대량의 고차원 공간 데이터에 대해서는 최상의 대표객체를 선택하는데 어려움이 많다는 문제점이 있다.

본 논문에서는 입력 차원의 순서와 무관하게 동일한 클러스터를 탐사할 수 있는 효율적인 부분차원 클러스터링 알고리즘인 CLIP을 제안한다. CLIP은 클러스터 형성에 밀접하게 연관된 임의의 차원에서 클러스터를 탐사한 후에, 그에 종속적인 다음 차원에 대해서 점진적인 프로젝션을 이용하여 클러스터를 탐사하는 기법이다. 점진적 프로젝션 기법은 제안된 알고리즘의 핵심 기법으로서 방대한 양의 탐색공간과 클러스터링을 식별하는 계산시간을 크게 줄인다. 이에 따라 CLIP 알고리즘을 평가하기 위해 합성 데이타를 이용한 실험을 통하여 알고리즘의 정확성 및 효율성, 알고리즘 결과의 동등성에 대한 실험 및 비교 분석 결과를 제시한다.

ABSTRACT

In high dimensional data, most of clustering algorithms tend to degrade the performance rapidly because of nature of sparsity and amount of noise. Recently, partial dimensional clustering algorithms have been studied, which have good performance in clustering. These algorithms select the dimensional data closely related to clustering but discard the dimensional data which are not directly related to clustering in entire dimensional data. However, the traditional algorithms have some problems. At first, the algorithms employ grid based techniques but the large amount of grids make worse the performance of algorithm in terms of computational time and memory space. Secondly, the algorithms explore dimensions related to clustering using k-medoid but it is very difficult to determine the best quality of k-medoids in large amount of high dimensional data.

In this paper, we propose an efficient partial dimensional clustering algorithm which is called CLIP. CLIP explores dense regions for cluster on a certain dimension. Then, the algorithm probes dense regions on a next dimension, dependent on the dense regions of the explored dimension using incremental projection. CLIP repeats these probing work in all dimensions. Clustering by Incremental projection can prune the search space largely and reduce the computational time considerably. We evaluate the performance(efficiency, effectiveness and accuracy, etc.) of the proposed algorithm compared with other algorithms using common synthetic data.

키워드 : 클러스터링, 고차원 데이터, 프로젝션
Clustering, High Dimensional Data, Projection.

* 경문대학 인터넷미디어정보과, Dept. of Internet Media Information in Kyung Moon College
** 대진대학교 컴퓨터공학과, Dept. of Computer Engineering in Daejin University

1. 서 론

데이터 마이닝을 위한 기법 중에서 클러스터링은 데이터 집합에 대하여 유사한 특징을 가진 객체들을 집단화하는데 사용되는 매우 유용한 분석방법이다. 특히, 전자상거래에서의 클러스터링은 대용량 데이터베이스에서 유사성 검색, 고객 분류, 경향 분석 등에서 폭넓게 응용되고 있다. 응용분야로서 전자상거래에서 발생되는 방대한 양의 데이터를 클러스터링하여 판매전략, 수요예측, 생산계획 등에 적용할 수 있는 연구가 필요하다.

기존의 다양한 클러스터링 알고리즘들이 개발되고 있으나, 잘 알려진 대부분의 알고리즘들은 고차원 데이터 공간에 있어서 클러스터링의 성능이 크게 저하되는 경향이 있다. 이러한 현상은 고차원 데이터 공간의 데이터 점들이 갖는 고유의 희소성(sparsity)에 기인하는데, 이는 차원 전체가 클러스터 형성에 관련되지 않을 수 있다는 것에서 비롯된다[1,2,3]. 최근에는 이를 해결하기 위한 방법으로, 데이터 점들이 서로 클러스터에 연관성이 있는 차원만을 선택하고, 데이터의 잡음(noise)에 해당하는 클러스터 형성에 관련이 적은 차원들을 제거함으로써 효율성을 높일 수 있는 부분차원 클러스터링 기법이 연구되고 있다.

대표적인 부분차원 클러스터링 알고리즘으로는 CLIQUE[2], PROCLUS[1] 등이 있다. CLIQUE는 부분차원 클러스터링의 첫 번째 연구로서 고차원 공간상의 데이터 점들은 차원의 부분집합에 대해 보다 잘 클러스터링될 수 있다는 사실에 근거한다. 그러나 CLIQUE는 데이터 점들을 서로 소인 집합으로 분할하기 어렵기 때문에 엄밀한 정의의 클러스터 탐색에는 한계가 있다. PROCLUS는 부분차원에 존재하는 클러스터를 효과적으로 찾을 수 있는 접근방법이지만, 주로 특정 차원에 국한된 희소 데이터 분석에만 용이하며, 대용량의 고차원 공간 데이터에 대해 최상의 대표객체를 선택하는데 있어서 어려움이 따른다.

본 연구에서는 위와 같은 부분차원 클러스터링 알고리즘의 단점을 해결하기 위해서 새로운 부분차원 클러스터링 기법인 CLIP (CLustering based on Incremental Projection) 알고리즘을 제안한다. 제안하는 기법은 대용량의 고차원 데이터 공간에서 클러스터 형성에 연관성이 있는 임의의 한 차원을 선택하여 클러스터를 형성하는 밀집영역을 탐사하고, 그에 종속적인 다음 차원에 대해서 점진적인 프로젝션(incremental projection)을 이용하여 고(k)차원까지 탐사하는 기법이다. 이러한 밀도에 근거한 점진적인 프로젝션은 잡음 데이터 및 조사 공간을 효과적으로 줄일 수 있다. 또한, CLIP은 클러스터 형성에 밀접하게 연관성이 있는 차원 및 영역을 식별할 수 있는 부분차원 클러스터링 기법이다. 따라서 고차원 클러스터를 분석하기 위한 목적으로 부분차원의 명세가 요구될 때 이를 지원할 수 있으며, 입력 데이터 레코드에 누락된 값이나 다양한 형태의 데이터를 처리할 수 있다. CLIP 알고리즘은 클러스터 정제 단계를 추가적으로 적용한다. 클러스터 정제는 도메인이나 응용분야에 따라 선택적으로 수행하는데, CLIP의 클러스터링 단계에 의해 찾은 곱집합에 반복적인 2차원 프로젝션을 적용하여

클러스터의 고차원적 잡음을 필터링하고, 프로젝션 방법에서 간과하기 쉬운 클러스터 형태를 보다 구체화할 수 있다. 또한 밀도 임계값을 변화시키면서 클러스터 형태를 일반화 또는 세분화하여 사용자가 원하는 수준으로 클러스터를 정제할 수 있다.

2. 관련연구

여러 개의 어트리뷰트로 구성된 고차원 데이터베이스에서 자동화된 클러스터링은 매우 중요한 문제이며, 고차원 데이터를 적용할 수 있는 다양한 클러스터링 알고리즘이 연구되고 있다. 고차원 데이터 공간에서 클러스터를 탐사하는 기법으로는 크게 전체차원 기반 클러스터링과 부분차원 클러스터링 알고리즘으로 나눌 수 있다.

2.1 전체차원 기반의 클러스터링

대표적인 전체차원 기반의 클러스터링 알고리즘으로는 크게 CLARANS[8]와 같은 분할(partitioning) 알고리즘, 계층적 알고리즘, DBSCAN[4]과 DBCLASD[12] 등의 지역성 기반의 알고리즘이다. 분할 알고리즘의 기본 개념은 데이터베이스를 무게중심(k-means) 또는 대표객체(k-medoid)에 의해 분할하여 k개의 클러스터를 형성하는 것이며 각각의 객체는 가장 가까운 클러스터에 할당된다. 계층적 클러스터링 알고리즘은 데이터베이스를 몇 단계의 분할된 작은 부분집합으로 분해하여 트리구조로 표현하는데, 이러한 계층적 트리는 top-down 또는 bottom-up 방식에 의해 재귀적으로 형성된다. 계층적 알고리즘은 지식탐사에서 매우 효과적이며 대부분의 접근방법들이 계층구조로 확장 가능하지만, 트리의 생성에 소요되는 상당한 비용 때문에 대용량 데이터베이스에 대해서는 비현실적이다[6]. 보다 효율적인 기법은 지역성 기반의 클러스터링 알고리즘으로서 이는 지역적 조건을 기반으로 인접한 데이터 요소들을 클러스터로 그룹화하며 따라서 데이터베이스를 한 번만 스캔하여 클러스터링할 수 있다. 예를 들어, DBSCAN은 클러스터의 밀도 기반 개념을 사용하여 임의 형태의 클러스터를 식별한다. 기본 개념은 클러스터의 각 점에 대하여 이웃에 위치한 데이터 점들의 밀도는 임의의 임계값을 초과해야 한다는 것에 배경을 두고 있다. DBCLASD는 지역성 기반으로 수행하지만 DBSCAN과는 달리 클러스터 내부의 점들은 무작위로 분포하고 있음을 가정하므로, DBCLASD는 어떠한 입력 매개변수 없이 수행한다.

대부분의 접근방법들은 고차원 데이터의 클러스터링에 대해 설계되지 않았으므로 기존 알고리즘들의 성능은 차원이 증가함에 따라 빠르게 저하하는 것이 문제점이다. 효율성을 개선하기 위해 최적화된 클러스터링 기법들이 제안되고 있다. 예로서 그리드 기반 클러스터링, CF-트리에 기반한 BIRCH[11], 추가적 통계정보를 포함하는 Quadtree와 닮은 구조를 사용하는 STING[9], 효율성을 높이기 위해 일정한 그리드를 사용하는 DENCLUE[6] 등이 있다.

이와 같이 클러스터링 알고리즘에 대한 많

은 연구가 이루어지고 있으나 기존의 대부분 알고리즘들은 "차원의 저주(curse of dimensionality)" 현상으로 고차원 공간에서 결과적인 클러스터링의 효과성 및 효율성에 심각한 영향을 준다[6]. 더욱이, 고차원 데이터는 많은 양의 잡음 데이터를 포함하고 있으므로 알고리즘의 추가적인 효과성 문제를 야기한다.

2.2 부분차원 기반의 클러스터링

대부분 클러스터링 알고리즘들이 고차원 공간에서 효율적으로 수행하지 못하는 중요한 이유는 고차원 데이터의 희소성 때문이다. 이에 따라 데이터 점들이 서로 연관되어 있는 특정 차원을 선택하고, 클러스터 형성에 관련이 적은 차원들을 제거함으로써 데이터의 잡음을 감소시킨다는 개념의 부분차원 클러스터링 기법들이 개발되고 있다. 이처럼 관련성 있는 차원이나 공간을 고려하는 알고리즘으로는 CLIQUE[2], PROCLUS[1] 등이 제안되었다.

CLIQUE(CLustering In QUEst)는 고차원 데이터에서 클러스터를 찾는 방법으로 부분공간 즉 부분차원에서의 클러스터링 기법을 제시하였다. CLIQUE는 데이터 공간의 밀도를 간단하게 산정하기 위하여 각 차원을 다수의 일정한 간격으로 나누고 분할된 각 셀(unit) 안에 놓여진 점들의 수를 찾는다. CLIQUE는 다음의 세 단계로 수행된다. 1) 클러스터를 포함하는 부분공간 식별 단계에서는 우선 부분공간에 존재하는 밀집영역(dense unit)을 찾는다. 2) 클러스터 식별 단계에서는 1)의 결과인 밀집 단위들의 집합을 분할하여 출력하는데 각 분할들은 하나의 클러스터이다. 3) 클러스터의 최소명세(minimal description) 생성 단계에서는 최대영역에 대한 최소한의 명세를 DNF식을 사용하여 클러스터를 표현한다.

PROCLUS(PROjected CLUStering) 알고리즘은 데이터 점 및 차원을 기반으로 클러스터를 산출한다. PROCLUS의 프로젝트된 클러스터링 알고리즘은 우선, CLARANS에서 제안한 medoid 탐색기법을 이용하여 클러스터와 차원의 집합을 찾는다. 그 다음, 찾아진 각 medoid와 연관된 차원의 집합을 찾기 위하여 지역성 분석(locality analysis)기법을 사용한다. 즉 PROCLUS 알고리즘은 다음의 3가지 단계로 진행된다. 1) 초기화 단계에서는 CLARANS를 이용하여 데이터 점들의 집합을 줄이는데 것을 목적으로 한다. 2) 반복 단계에서는 보다 좋은 medoid 집합의 탐색을 시도하기 위하여 각 medoid에 대응하는 차원의 집합을 계산하여 medoid에 할당된 점들은 결정된 부분차원에서 최상의 클러스터를 형성한다. 3) 클러스터 정제 단계에서는 클러스터링의 질적인 향상을 위하여 데이터를 한번씩 검사한다.

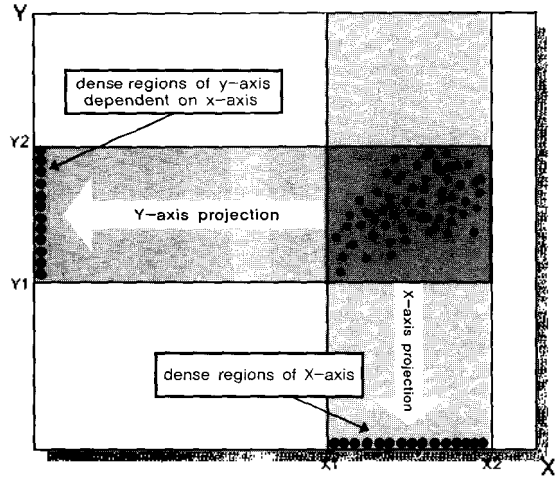
3. CLIP : 고차원 데이터에 대한 부분차원 클러스터링

3.1 CLIP의 기본 개념

CLIP은 부분차원 클러스터링 알고리즘으

로서, 고차원 데이터를 저차원 데이터로 변환하기 위하여 프로젝션을 이용한다. 즉, 대용량의 고차원 데이터 공간에서 클러스터링에 연관된 차원에 대해 선형변환 프로젝션(linear transformation projection)을 수행하여 밀집영역을 찾은 뒤, 그 차원의 밀집영역에 종속적인 그 다음 차원의 밀집영역을 찾는 방법으로 최종 k차원까지 반복적으로 수행하는 기법이다. 이 때 각 차원에서의 밀집영역은 클러스터를 포함하는 부분공간을 찾는데 중요한 사항으로서, 본 연구에서는 이러한 부분공간의 조사방법으로 Monotonicity 정리를 활용한다[2]. 즉 k차원 데이터 공간 R^k에 클러스터가 존재하면, 그 클러스터의 (k-1)차원적 부분집합은 또한 밀집영역을 포함한다는 것이다.

CLIP에서 사용하는 프로젝션의 의미는 다음의 (정의 1)과 같으며, 특히 〈그림 1〉과 같이 축-평행 프로젝션(axis parallel projection)을 하는데 특정 축에 해당하는 차원을 제외한 다른 모든 차원의 값은 0으로 간주하고 데이터를 조사하는 것이다. 따라서 CLIP은 1차원적 부분공간에서의 클러스터링에 기본을 두지만, 점진적인 프로젝션을 이용하여 고차원 클러스터를 결정하게 한다. 그리고 프로젝션하는 영역의 선택은 밀도에 근거하므로 단계가 증가함에 따라 잡음 데이터 수는 점차 감



〈그림 1〉 축-평행 프로젝션

(정의 1) 축-평행 프로젝션

전체 k차원 공간 R^k의 데이터 집합에 대한 l-축으로의 프로젝션 P_l을 다음과 같이 정의한다.

$$P_l : R^k \rightarrow R^k \quad ; \quad P_l(x^{(1)}, x^{(2)}, \cdots, x^{(l)}, \cdots, x^{(k)}) = \hat{x}^{(l)}, \quad 1 \le l \le k$$

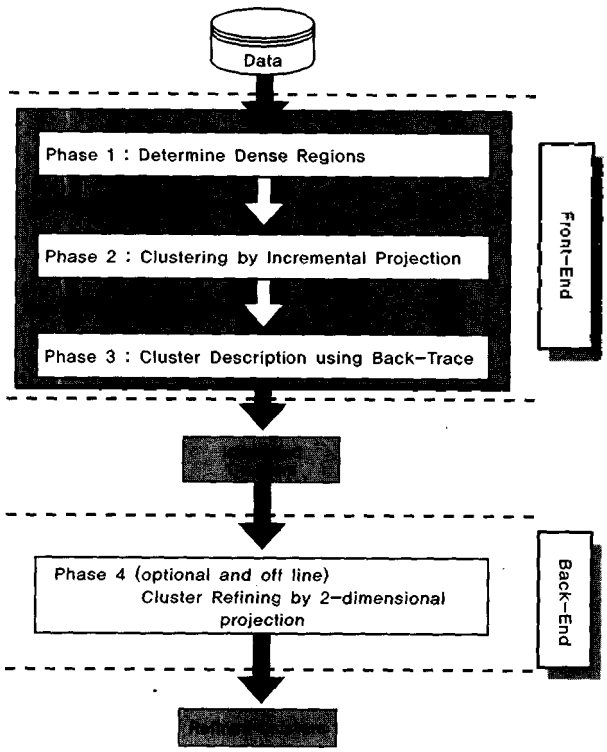
여기서 $\hat{x}^{(l)} = (0, \cdots, 0, x^{(l)}, 0, \cdots, 0)$ 이다. 즉, l-축을 제외한 모든 항은 0(zero)으로 한다.

소하게 되고, 탐사공간도 감소하게 된다. 이때 프로젝션하는 차원의 순서는 임의적일 수 있으며, 차원의 중요도에 따라 우선순위를 부여할 수도 있다.

모집단 $X \subset R^k$의 l축의 프로젝션은 $X_l := P_l(X) = \{P_l(x): x \in X\}$이다. 만일 X의 평균이 $\mu = (\mu_1, \mu_2, \cdots, \mu_k)$이면, X_l의 평균은 $\mu_l = P_l(\mu)$이다. 이와 같이 프로젝션은 본래 고차원 공간의 데이터 집합에 관한 통계적 물리량을 보존한다.

3.2 CLIP의 수행구조

본 논문에서 제안하는 CLIP 알고리즘은 〈그림 2〉와 같이 크게 전반부(front-end)와 후반부(back-end)로 나눌 수 있다. 전반부는 크게 3 단계로 구성되며, 점진적인 프로젝션에 의해 부분차원 클러스터링을 수행하는 단계로서 잡음이 제거된 명확한 클러스터가 생성된다. 1 단계(*phase 1*)에서는 고차원 데이터에 대해 임의의 차원을 선택하여 프로젝션을 통해 밀집영역(클러스터)을 결정한다. 2 단계(*phase 2*)에서는 1 단계에서 결정된 밀집영역에 종속적인 다음 차원에 대해 점진적으로 프로젝션을 적용하여 고(k)차원까지 클러스터링을 수행한다. 이와 같은 과정을 거치면서 많은 잡음이 제거된다. 3 단계(*phase 3*)에서는 조사된 클러스터에 대해서 백트레이스



〈그림 2〉 CLIP의 흐름

(back trace) 과정을 통하여 클러스터의 정확한 명세를 얻게 되며 이 때 잡음은 거의 제거된다. 후반부인 4 단계(*phase 4*)에서는 전반부로부터 생성된 클러스터에 대해 반복적인 2차원 프로젝션을 수행하여 사용자가 원하는 수준으로 클러스터 형태를 구체화하고 고차원의 잡음을 필터링한다. 후반부는 보다 정제된 품질의 클러스터를 얻기 위한 과정으로서 도메인이나 응용분야에 따라 선택적으로 수행한다.

3.3 축-평행 프로젝션에 의한 밀집영역의 결정

CLIP에서는 클러스터링을 수행하는데 있어서 각 차원에서의 밀집영역, 즉 1차원적 클러스터를 찾는 과정을 중요하게 고려한다. 이를 위하여 CLIP의 1단계에서는 고차원 데이터 공간에서 레코드의 어트리뷰트에 해당하는 각 차원의 모든 값에 대하여 축-평행 프로젝션을 수행한 후, 데이터의 밀도가 임계값 τ 이상인 밀집영역을 찾아 클러스터로 결정한다.

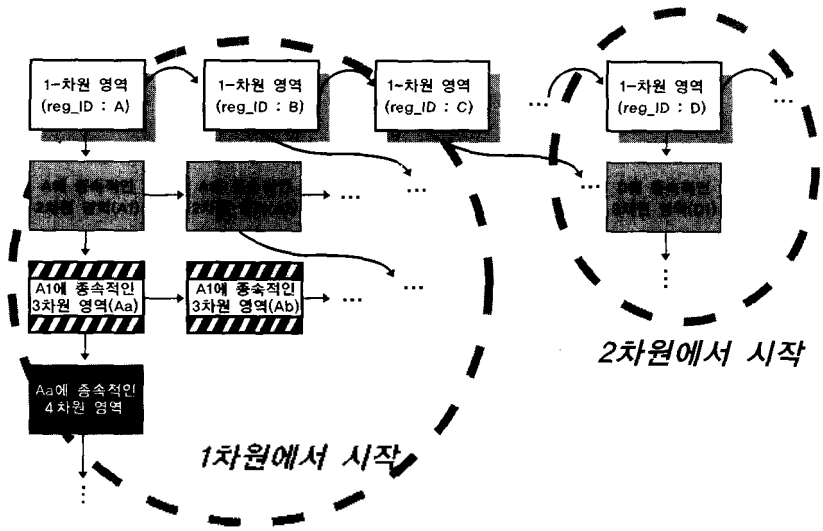
먼저, 데이터 공간의 각 차원을 일정단위(u)로 분할하는데, 각 단위들은 모든 차원을 일정한 ξ 간격으로 나누어 얻어진다. 각 단위 u는 각 어트리뷰트에서 한 간격의 교차점으로서, 각 차원은 집합 $\{u_1, u_2, \cdots, u_d\}$로 정의할 수 있다. 여기서 1개의 간격은 $u_i = [l_i, h_i)$로 표현한다. 이때 한 단위의 선택여부는 단위 안에 포함된 모든 점들의 수로 정의한다. 즉 하나의 "단위 u가 밀집(*dense*)하다"는 것은 $(selectivity(u) \geq \tau)$ 을 만족할 때임을 뜻한다.

다음으로, 밀집단위들의 집합에서 밀집영역을 결정한다. 입력은 밀집단위들의 집합이고, 출력은 이로부터의 밀집영역인 $R_1, R_2, \cdots, R_n$로의 분할이다. $R_1, R_2, \cdots, R_n$는 연결된 밀집단위들의 최대집합이다. 이때 의미있는 밀집영역을 결정하기 위해 본 논문에서는 동일한 밀집영역에 대해 그 안의 밀집단위에 속한 데이터 점들의 수를 다음과 같이 계산한다. 여기서 xR_j는 밀집영역 R_j의 중요도로써 참조할 수 있다.

$$xR_j = \sum_{u_i \in R_j} count(u_i)$$

(단, $count(u_i)$는 u_i안에 속한 데이터 점들의 수)

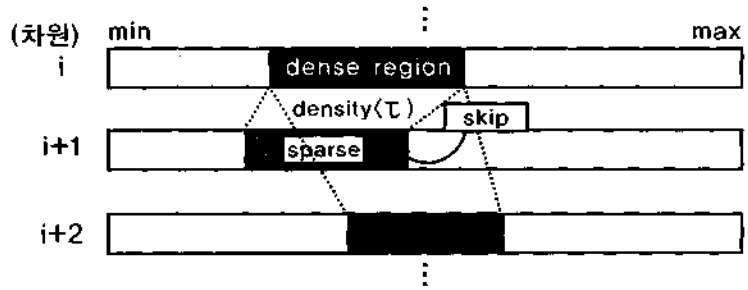
각 차원에서 밀집영역을 결정하면 결정된 밀집영역에 대해 점진적으로 차원을 증가시키면서 이에 종속적인 밀집영역을 구성한다. 그러면 차원들에 대한 밀집영역의 연관성을 트리 구조로 나타낼 수 있다. 〈그림 3〉은 각 차원별로 프로젝션한 데이터를 조사하여 밀집영역을 결정한 후 이와 연관된 차원 및 영역을 연결시키는 리스트 구조를 보이고 있다. 그림에서 1차원에서 프로젝션을 시작하여 데이터를 조사한 결과, 밀집영역으로 조사된 영역은 *reg_ID*가 A, B, C 등이다. 그리고 A, B, C 각각에 종속적인 2차원의 밀집영역이 결정되는데, 여기에 속하는 밀집영역의 개수는 임의의 n개 $(n \geq 0)$이다. 이와 같은 과정은 2차원에서 시작하여 밀집영역으로 결정된 *reg_ID*가 D인 경우에도 동일하게 적용된다.



〈그림 3〉 밀집영역의 연결리스트 구조

3.4 점진적 프로젝션에 의한 클러스터링

2 단계에서는 1단계에서 결정된 클러스터의 차원과 종속적인 다음 차원에 대해서 기준 밀도를 초과하는 초월사각형 부분에 존재하는 레코드에 대해서만 프로젝션한다. 예를 들면, 〈그림 4〉에서 i차원의 밀집영역에 대한 (i+1)차원의 영역은 밀도 임계값 τ를 만족하지 못하므로 (i+1)차원에서는 영역을 선택하지 않고, i차원에 대한 (i+2)차원 영역을 조사하여 밀집영역을 선택한다. 즉, 각 차원에 있어서, 이전 차원에 종속적으로 결정된 차원의 영역 밀도가 임계값을 초과하지 않는 차원은 제외시키고, 그 다음 차원을 조사하여 클러스터를 결정하는 것이다. 이와 같이 CLIP에서는 각 차원에 해당하는 밀집영역을 그 다음 차원에 종속적으로 반영시킴으로써 탐사할



〈그림 4〉 밀집영역이 없는 경우

데이터 공간 및 잡음을 줄일 수 있다. 점진적 프로젝션에 의한 클러스터링 알고리즘은 〈그림 5〉와 같다.

위의 알고리즘에서는 *Num_subspace*라는 변수를 사용하여 클러스터로 인정할 부분차원의 개수를 입력받는다. *state_list*는 클러스터를 형성하는 차원의 연관성을 표현하는 배열로서, 밀집영역의 존재 여부를 의미하는 플래그 값을 포함한다. *dense_region*은 현재 조사하고 있는 차원의 밀집영역에 대한 최소값 및 최대값을 저장하는 배열로서, 차원이 점진적으로 증가함에 따라 해당 차원의 밀집영역에 대한 값으로 갱신된다. *Find_Dense()*는 밀집영역을 결정하여 클러스터링 알고리즘으로 결과를 반환하는 함수이다.

라인 6에서 전체 데이터 공간이 k차원이라 할 때, k개의 어트리뷰트들에 대해 각 어트리뷰트의 중요도에 따라 내림차순으로 정렬한다. 이와 같이 정렬된 k개의 어트리뷰트들에 대해 점진적으로 선형변환 프로젝션을 적용

```
1 : /* Input: Data_rec(set of input data record), ξ(size of grid)
2 :          Num_subspace(number of candidate subspaces for cluster),
3 :  Output: identified cluster, data ID */
4 : procedure GenerateCluster(Data_rec, Num_subspace) {
5 :  Process the linear transformation projection for entire dimensions in Data_rec;
6 :  Sort k dimensions by descending;
7 :  for k dimensions {
8 :    for i = 1 to (k Num_subspaces + 1) do {
9 :        Found = Find_Dense(Data_rec);
10 :       if (Found is equal to 0) then
11 :          i = i + 1;
12 :       for j = (i + 1) to k do {
13 :           Read data IDs on (i + 1)th dimension dependent on ith dimension;
14 :           Found = Find_Dense(Data_rec);
15 :           Assign a value(0, 1 or null) to Found and put it into state_list[i][j];
16 :           if (j is equal to (k Num_subspaces + 2)) then
17 :             Figure out a sum of "1" in state_list[i][j];
18 :             if (the sum = 1) then
19 :               break For (i+1≤j≤k);
20 :
21 :           else j = j + 1;
22 :       }end for
23 :       i = i + 1;
24 :   }end for
25 :   Read the minimum value of data IDs and the maximum value of data IDs
       in dense_region[];
26 :   Decide the scope of dense region between minimum value and maximum
       value for each dimension by back trace process;
27 :  }end for
28 : }end procedure
```

〈그림 5〉 CLIP의 클러스터링 알고리즘

한다. 입력 데이터 레코드의 첫 번째 차원에 해당하는 값을 프로젝션하여 데이터 영역이 결정되면, 여기서 밀도 임계값을 초과하는 밀집영역을 찾게 된다. 그 다음 찾아진 밀집영역에 대한 입력 데이터 ID를 조사하여, 그 데이터 ID들의 두 번째 차원에 해당하는 값들만을 선택하여 프로젝션한다. 이와 같은 과정을 1차원부터 k(k *Num_subspaces* + 1)차원까지 반복하게 되는데, 이것은 클러스터로 인정할 부분차원의 개수가 전체 k개 중에서 최소 *Num_subspace*개(입력된 값)를 만족해야 하기 때문이다. 위의 알고리즘에서 적용하는 점진적 선형변환 프로젝션 기법은 차원들에 종속적인 관계를 이용하여 클러스터를 찾기 때문에 방대한 양의 데이터 공간에서 클러스터와 무관한 영역을 크게 제거할 수 있다. 또한 알고리즘은 부분차원 클러스터링 알고리즘이므로 전체차원 뿐만 아니라 부분차원 공간에 존재하는 클러스터를 거의 찾을 수 있다. 이는 알고리즘의 효과성과 정확성에 해당하는 것으로서 본 논문에서는 4장의 실험 부분에서 그 성능을 다양한 실험결과로써 입증하고 있다.

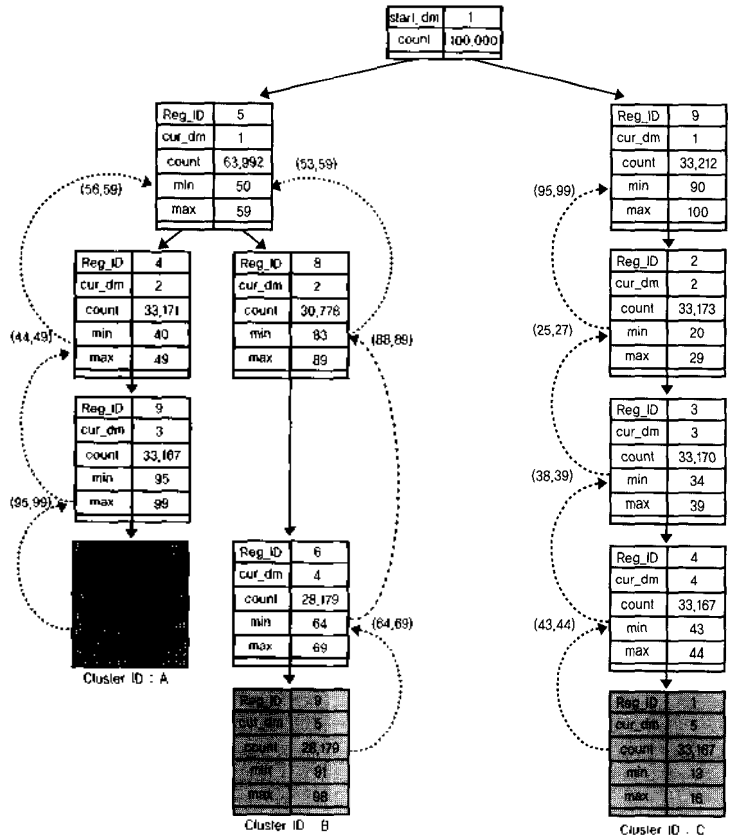
3.5 백트레이스에 의한 클러스터 명세

2 단계에서 점진적인 프로젝션을 이용하여 결정한 밀집영역은 이에 종속적인 다음 차원에 대해서 여전히 다소의 잡음 데이터를 포함하고 있다. 따라서 CLIP은 이러한 잡음을 제거하고 클러스터에 포함된 데이터들의 영역을 정확히 명세하기 위해서 백트레이스를 수행한다.

〈그림 6〉은 점진적인 프로젝션을 수행하여 조사된 클러스터의 생성과정 및 백트레이스 과정을 연결리스트 형태로 표현한 예를 보여주고 있다. 실험 데이터 집합은 전체 5차원의 100,000개 데이터를 포함하고, 값의 범위는 [0, 100], 그리드 간격 ξ는 10이며 클러스터로 인정할 부분차원의 개수(*Num_subspace*)는 3이다. 그림에서 점선으로 표시한 것처럼 백트레이스를 수행하면서 클러스터의 각 차원에 대한 최소값 및 최대값을 다시 명세한다. 예를 들어, *Cluster ID*가 A인 클러스터에 대해 백트레이스를 이용한 각 차원의 명세를 살펴보면 다음과 같다. 클러스터 A의 데이터 개수는 33,167이며, 클러스터를 형성하는 4차원 값의 범위는 [30, 31], 3차원은 [95, 99], 2차원은 [44, 49] 그리고 1차원은 [56, 59]이다. 클러스터 A의 1차원 값의 경우, 클러스터 결정 과정에서 조사된 영역의 범위는 [50, 59]인데 백트레이스를 통해 명세된 범위는 [56, 59]로서 더욱 정확한 차원 명세를 얻을 수 있다. 또한 점진적으로 차원이 증가함에 따라 데이터 수(*count*)가 줄고 있는 현상을 볼 수 있는데, 이는 단계적으로 잡음 데이터를 필터링하고 있음을 의미한다.

3.5 반복적 2차원 프로젝션에 의한 클러스터 정제

〈그림 2〉의 후반부인 클러스터 정제 과정은 기존의 CLIP 알고리즘으로 식별된 클러스터에 속한 데이터 집합을 대상으로 한다. 이 과정의 목적은 첫째 프로젝션 방법에서 간과하기 쉬운 클러스터의 실제 형태를 예측하



〈그림 6〉 백트레이스에 의한 차원 명세

고, 둘째 고차원의 잡음을 필터링하며 셋째, 사용자가 원하는 수준의 클러스터를 찾는데 있다. 점진적인 1차원적 프로젝션으로 결정된 클러스터는 각 차원에 대해 초월사각형 형태의 영역이다. 이때 실제 클러스터 영역의 크기는 이보다 작을 수 크다. 즉, *Size*(*CLIP*에 의한 초월사각형 영역) ≥ *Size*(실제 클러스터 영역)이며, 두 영역의 차이는 잡음으로 간주할 수 있다. 잡음 데이터 판단의 경우, 1차원 프로젝션 방법에 의하여 1차원에서는 완전한 클러스터를 찾았으므로 더 이상 1차원적 잡음은 존재하지 않으나 고차원의 잡음은 존재할 수 있다. 그러나 직접적인 고차원 데이터 필터링은 연산의 복잡성에 의하여 매우 큰 계산비용이 소요된다. 따라서 적은 비용으로 실현 가능한 2차원 필터링을 반복적으로 적용하여 고차원 필터링을 근사적으로 구현하고자 하는데, (정의 2)와 같은 2차원 프로젝션을 적용한다.

정제 방법은 전체 k차원 클러스터의 정제를 위하여 임의 2차원의 쌍 (i,j)에 대하여 반복적으로 프로젝션을 하는 것이다. 이 경우 연산횟수가 총 ${}_kC_2 = k(k-1)/2$번이라는 많은 반복이 필요하다. 따라서 본 논문에서

는 이 중 일부의 (i,j) 쌍을 선택하는 부분적 정제(partial refining)를 적용한다. CLIP의 전반부에 의해 결정된 클러스터를 대상으로 하는 클러스터 정제 알고리즘은 〈그림 8〉과 같다.

2차원 프로젝션에 적용한 그리드(ξ')는 알고리즘 전반부의 클러스터링에서 사용했던 것보다 세분화된 크기이며, 클러스터링을 조절하는 밀도 임계값(τ')은 입력받는다. 이때 임계값을 크게 하면 많은 객체가 포함된 셀들이 선택되어 보다 일반화되고, 작게 하면 적은 객체가 포함된 셀들도 선택되어 좀더 세분화된다. 그러므로 사용자는 임계값을 변화시키면서 원하는 수준의 클러스터를 발견할 수 있다. 임의의 (i,j) 축으로 이루어진 2차원 부분공간에서, 각 i축 ($1 \le i \le k$)은 n_i개의 그리드 셀로 나뉘며, 2차원 셀들의 총 개수는 $n_i \times n_j$이다. 〈그림 7〉의 *min*과 *max*는 CLIP에 의해 식별된 클러스터에 존재하는 값에 대한 각 축의 최소값 및 최대값을 의미한다. 따라서 i축 그리드 셀의 개수(n_i)는 입력된 그리드 크기(ξ_i)에 의해 $\left\lceil \frac{(\max_i - \min_i)}{\xi_i} \right\rceil$로 계산된다. n_i의 값이 커지면 그리드 셀의 크기가 작아져 효율성이 낮아지고 의미있는 클러스터를 발견하기 어려워진다. n_i의 값이 1에 근접하면, 즉 그리드 셀의 크기가 커지면 셀에 포함된 객체의 개수가 증가하여 사용자가 원하는 다양한 수준의 클러스터를 발견하기 어렵다. 그러므로 ξ_i를 결정하기 위해서는

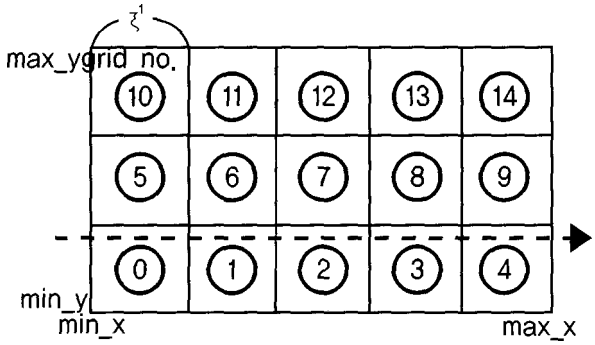
(정의 2) 2차원 프로젝션

전체 k차원 데이터 집합에서 (i,j)축으로의 2차원 프로젝션 $P_{i,j}$는 다음과 같이 정의한다.

$P_{i,j}(x^1, x^2, \cdots, x^k) = (x^i, x^j)$ 혹은

$P_{i,j}(x_1, x_2, \cdots, x_k) = (0, \cdots, 0, x_i, 0, \cdots, 0, x_j, 0 \cdots, 0)$

즉 (x_i, x_j)을 제외한 모든 항은 0(zero)이다.



〈그림 7〉 2차원 프로젝션에 의한 (3×5) 그리드

```
1 : /* Input: object IDs in the identified Cluster, size of grid ξ'
2 :         threshold for density τ', pairs of dimensional clusters
3 :   Output: refined cluster */
4 : procedure RefineCluster() {
5 :   for pairs of two dimensional clusters {
6 :     Process two dimensional projection for data set;
7 :     Execute Cartesian Product for regions within the scope of (density≥τ');
8 :     Traverse two dimensional cells in the region from Cartesian Product
        by DFS;
9 :     for all traversed cells {
10 :        if (density≥τ') then
11 :          Select data which are included in cell;
12 :        else
13 :          Discard data which are included in cell;
14 :    }end for
15 :  }end for
16 : return cluster }
```

〈그림 8〉 클러스터 정제 알고리즘

먼저 데이터 집합의 분포를 분석하는 것이 바람직하다.

4. 실험 및 성능평가

4.1 실험환경 및 평가요소

본 논문에서 제시한 CLIP 알고리즘의 성능평가를 위하여 다양한 실험을 하였다. 실험은 256M 메인 메모리의 200-MHz SUN-Ultra SPARC Ⅱ 워크스테이션 환경 하에서 수행하였고, 데이터는 12GB SCSI 디스크에 저장하였다. 또한, C++언어와 LEDA-4.2 라이브러리[7], 그리고 GNU g++ 컴파일러를 사용하였다. CLIP 알고리즘의 실험을 위한 데이터는 IBM Almaden 연구소에서 개발한 마이닝 프로젝트인 QUEST에서 사용한 합성 데이터 집합이다. 합성 데이터는 다음 절들에서 비교하는 CLIQUE와 PROCLUS에서 사용한 것으로서, 보다 객관적인 비교를 위하여 본 논문에서도 사용하도록 하였다. 데이터 생성기는 데이터 레코드의 수, 어트리뷰트의 수, 각 어트리뷰트 값의 범위 등의 매개 변수들을 통하여 데이터 집합의 구조와 크기를 제어할 수 있다[2,10].

CLIP 알고리즘의 실험 평가는 효과성(effectiveness), 효율성(efficiency), 결과의 동등성(equivalency)의 측면에서 수행한다. 효과성은 고차원 데이터 공간의 전체 및 임의 부분차원에서 클러스터를 발견할 수 있는가를 실험하는 것이다. 효율성은 데이터베이스의 크기, 데이터 공간의 차원, 클러스터의 차원 증가에 따라 수행시간이 어떻게 변하는가를 다른 알고리즘과 비교 분석한다.

4.2 CLIP의 클러스터링 실험 결과 분석

4.2.1 효과성 분석

CLIP의 효과성 실험을 위한 데이터는 합성 데이터 생성기(synthetic data generator)에 의해 데이터 레코드의 수, 어트리뷰트의 수, 각 어트리뷰트 값의 범위 등을 제어할 수 있으며, 값의 범위는 모든 어트리뷰트에 대해 [0, 100]으로 고정하였다. 데이터 개수는 100,000개이며, 3개의 클러스터와 20%의 잡음 데이터를 포함하고 있다.

〈표 1〉에서 입력 *input* A는 10차원 26,667개 데이터로 형성된 전체차원 클러스터이며, *input* B는 26,667개 데이터로 3, 4, 6, 9, 10차원으로, *input* C는 26,666개 데이터로 1, 2, 3, 4, 5, 6, 7, 8, 10차원으로 형성된 부분차원 클러스터이다. 이러한 입력 클러스터에 대해 CLIP의 수행결과로서 식별된 클러스터는 *found* Ⅰ, Ⅱ, Ⅲ이다. 입력 클러스터와 CLIP이 식별한 결과를 비교해보면 알 수 있듯이 CLIP은 입력으로 주어진 클러스터들인 *input A, B, C*를 거의 정확하게 탐색하는 성능을 발휘하였다.

〈표 2〉는 입력 클러스터의 영역에 대해서 CLIP을 수행한 후, 출력 클러스터의 영역에 대해서 각 차원의 [최소값, 최대값]을 명세한 것이다. 전체차원 클러스터에서 *입력(A)*의 7차원의 경우 입력 클러스터 영역은 [80, 90]인데 *출력(Ⅰ)*은 [80, 89]로 조사되었고, *입력(B)*의 4차원의 경우는 [64, 70]인데 *출력(Ⅱ)*은 [64, 69]로 조사되었으며, *입력(C)*의 2차원과 5차원의 경우는 [24, 30]인데 *출력(Ⅲ)*은 [24, 29]로 조사되었다. 이것은 그리드 방식의 경계에 의해 발생하는 현상으로서, 출력 클러스터의 데이터 수가 입력 클러스터 데이터 수의 약 8.5%~15% 정도 감소되는 원인이 된다. CLIP에서는 그리드 단위를 우측 개구간 간격(right-open interval)으로 정의함에 따라

〈표 1〉 입력 클러스터에 대한 출력 클러스터의 비교

input	dimensions	points
A	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	26667
B	3, 4, 6, 9, 10	26667
C	1, 2, 3, 4, 5, 6, 7, 8, 10	26666
noises	-	20000

found	dimensions	points
Ⅰ	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	24130
Ⅱ	3, 4, 6, 9, 10	24398
Ⅲ	1, 2, 3, 4, 5, 6, 7, 8, 10	22399
noises	-	29073

〈표 2〉 입·출력 클러스터의 영역 명세

		전체차원 클러스터		부분차원 클러스터			
		입력(A)	출력(Ⅰ)	입력(B)	출력(Ⅱ)	입력(C)	출력(Ⅲ)
데이터 수		26,667	24,130	26,667	24,398	26,666	22,399
차원별 영역 명세 [최소, 최대]	1	[74, 79]	[74, 79]			[10, 16]	[10, 16]
	2	[21, 24]	[21, 24]			[24, 30]	[24, 29]
	3	[33, 36]	[33, 36]	[88, 89]	[88, 89]	[53, 55]	[53, 55]
	4	[72, 77]	[72, 77]	[64, 70]	[64, 69]	[46, 49]	[46, 49]
	5	[81, 89]	[81, 89]			[24, 30]	[24, 29]
	6	[40, 50]	[40, 49]	[91, 98]	[91, 98]	[54, 58]	[54, 58]
	7	[80, 90]	[80, 89]			[12, 15]	[12, 15]
	8	[89, 90]	[89, 90]			[43, 46]	[43, 46]
	9	[40, 44]	[40, 44]	[41, 44]	[41, 44]		
	10	[11, 14]	[11, 14]	[31, 38]	[31, 38]	[63, 64]	[63, 64]

그리드의 오른쪽 경계에 해당하는 데이터는 클러스터 영역으로부터 제외된 것으로 분석된다. 그러나 이 문제는 그리드 간격의 적절한 조절을 통해 해소시킴으로써 클러스터링의 정확성을 더욱 향상시킬 수 있을 것으로 판단된다.

4.2.2 효율성 분석

CLIP의 효율성 실험을 위해, 데이터 차원은 5~25차원으로, 데이터 수는 10만~50만으로, 전체 25차원 데이터 공간에서 숨겨진 클러스터의 차원은 3~11차원으로 확장시키면서 실험하였다. 실험에서 값의 범위는 모든 어트리뷰트에 대해 [0, 100]으로 고정하였으며, 그리드 셀의 크기인 ξ는 10으로 하였다. 수행에 소요된 모든 시간은 초 단위이다. 참고로 이 실험에서 그리드의 크기와 밀도 임계값은 기존의 그리드-기반 부분차원 알고리즘과의 성능비교를 위해 CLIQUE와 동일하게 적용하였다.

(1) 데이터베이스의 크기가 변할 때

〈그림 9〉는 100,000에서 500,000 레코드까지 데이터베이스 크기의 증가에 따른 확장성을 보여준다. 데이터 공간은 50차원이고 각각 서로 다른 5차원 부분공간에 5개의 클러스터를 포함하며, 밀도 임계값 τ는 10%로 하였다. 예상대로, CLIP의 수행시간은 데이터베이스 크기에 대해 선형적으로 증가하였는데 이는 데이터베이스를 스캔하는 횟수는 변하지 않기 때문이다. 또한 PROCLUS보다 약 10배, CLIQUE보다는 약 100배 빠른 수행 속도를

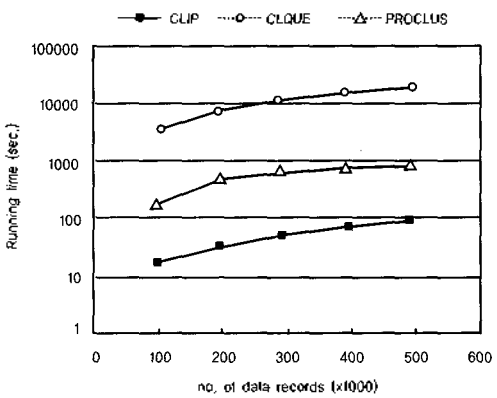
보인다. CLIP은 다른 알고리즘들과는 달리 밀집영역을 탐색할 때 선형변환 프로젝션을 이용하여 차원들간의 종속적인 부분들에 대해서 수행하기 때문에 탐색시간을 크게 줄일 수 있는 것이다.

(2) 데이터 공간의 차원이 변할 때

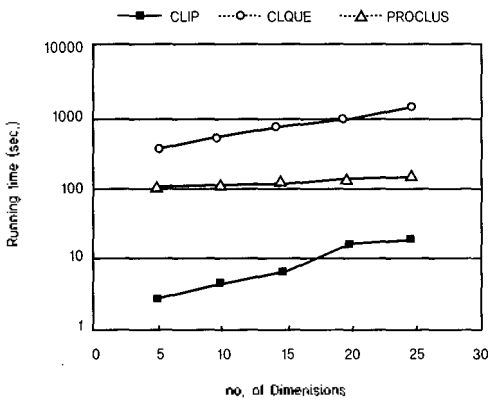
〈그림 10〉은 데이터 공간의 차원을 5에서 25까지 증가시킴에 따른 확장성을 보이고 있다. 데이터베이스는 100,000 레코드를 가지며 5개 클러스터가 있는데, 클러스터는 각각 서로 다른 5차원 부분공간에 존재하며 밀도 임계값 τ는 10%로 정한다. 흥미있는 부분차원을 찾는 문제는 데이터 공간의 차원이 증가하여도 잘 확장하지 않는다는 것에 주목할 수 있다. 이 경우에 있어서, 5개의 차원에 존재하는 클러스터를 모두 조사하였다.

4.2.3 CLIP의 정제 실험

CLIP의 클러스터 정제부분에 관한 성능 실험에서는 임의의 2차원 쌍을 선택한다. 이 논문에서는 수행한 반복횟수는 (k-1)번이며, 이는 공통된 차원으로 연결된 2차원 쌍에 대해 실험하고자 하였다. 2차원 프로젝션을 반복적으로 적용한 결과는 〈표 3〉과 같다. CLIP에 의한 클러스터 정제 에서 실험의 대상이 되는 데이터 집합은 CLIP 알고리즘의 전반부에 의해 조사된 전체차원 및 부분차원 클러스터이다. 표에서 나타내는 그리드 번호는 클러스터의 각 2차원에서 선택된 셀 번호를, m은 데이터 개수를 의미한다. 〈표 3〉의 실험 I-1에서 실험 I-3은 밀도 임계값(τ′)를 2%에서 10%까지 증가시킴에 따른 변화를 보이고 있다. ξ′=2이고 τ′=2%일 경우는 전반부의 클러스터링과 동일한 비율의 밀도이므로 2차원 프로젝션을 반복하여도 데이터 개수에 변화가 없다. 그러나 τ′를 5%, 10%로 증가시키면 제외되는 그리드가 발생하며 이에 따라 데이터의 수도 감소하는 현상을 볼 수 있다. 클러스터 ID가 A인 경우를 분석해 보면 다음과 같다. CLIP에 의한 클러스터링으로 식별된 클러스터의 개수는 24,130인데, τ′가 2%인 실험 I-1의 수행에서는 데이터 개수에 변화가 없다. 그러나 τ′를 5%로 증가시키면 (5,6)차



〈그림 9〉 데이터 개수별 수행시간



〈그림 10〉 데이터 차원별 수행시간

〈표 3〉 반복적 2차원 프로젝션 결과

Cluster ID	차원쌍	CLIP에 의해 선택된 그리드		(k-1)번의 반복적 2차원 프로젝션					
		총 그리드	그리드 번호	실험 I -1		실험 I -2		실험 I -3	
				$\xi'=2$, $\tau'=2\%$		$\xi'=2$, $\tau'=5\%$		$\xi'=2$, $\tau'=10\%$	
				그리드	m	그리드	m	그리드	m
A	(1,2)	(1×2)	⓪,①	동일	24,130	동일	24,130	동일	24,130
	(2,3)	(1×1)	⓪	동일	24,130	동일	24,130	동일	24,130
	(3,4)	(2×1)	⓪,①	동일	24,130	동일	24,130	동일	24,130
	(4,5)	(4×2)	⓪ ~⑦	동일	24,130	동일	24,130	⓪ 제외	16,740
	(5,6)	(4×4)	⓪ ~⑮	동일	24,130	⓪,①제외	18,478	⓪ 제외	5,625
	(6,7)	(4×4)	⓪ ~⑮	동일	24,130	⓪,①,②제외	14,524	⓪①②④⑤⑥⑧⑨⑩⑫⑬⑭제외	5,625
	(7,8)	(1×4)	⓪ ~③	동일	24,130	동일	14,524	동일	5,625
	(8,9)	(2×1)	⓪ ~②	동일	24,130	동일	14,524	동일	5,625
	(9,10)	(1×2)	⓪,①	동일	24,130	동일	14,524	동일	5,625
B	(1,2)	(2×3)	⓪ ~⑤	동일	67,293	동일	67,293	⓪ 제외	56,648
	(2,3)	(2×1)	⓪,①	동일	67,293	동일	67,293	동일	56,648
	(3,4)	(1×1)	⓪	동일	67,293	동일	67,293	동일	56,648
	(4,5)	(2×1)	⓪,①	동일	67,293	동일	67,293	동일	56,648
	(5,6)	(2×2)	⓪ ~③	동일	67,293	동일	67,293	동일	56,648
	(6,7)	(1×2)	⓪,①	동일	67,293	동일	67,293	동일	56,648
	(7,8)	(1×1)	⓪	동일	67,293	동일	67,293	동일	56,648
	(8,10)	(1×1)	⓪	동일	67,293	동일	67,293	동일	56,648
C	(3,4)	(2×3)	⓪ ~⑤	②,⑤	122,139	②,⑤	122,139	②,⑤	122,139
	(4,6)	(3×2)	⓪ ~⑤	동일	122,139	동일	122,139	⓪②제외	105,447
	(6,9)	(3×3)	⓪ ~⑧	⑥⑦⑧ 제외	122,139	⑥⑦⑧ 제외	122,139	⑥⑦⑧ 제외	86,435
	(9,10)	(4×3)	⓪ ~⑪	②⑤⑧⑪ 제외	122,139	⓪①②⑤⑧⑪제외	113,569	⓪①②⑤⑧⑪ 제외	80,329

원에서는 ⓪,①번 그리드가 제외되고, (6,7) 차원에서는 ⓪,①,②번 그리드가 제외되므로 결과적인 데이터 개수는 14,524개로 감소된다. 또한 τ′를 10%로 증가시키면 (4,5) 및 (5,6) 차원에서는 ⓪번 그리드가 제외되고, (6,7)차원에서는 ⓪~⑭번 그리드가 제외되며 결과적인 데이터 개수는 5,625로 감소된다. 이는 상대적으로 밀도가 희소한 그리드 셀은 제외된 결과이며, 밀도 임계값이 클수록 많은 데이터 객체가 포함된 그리드 셀들이 선택되어 더욱 일반화된 클러스터가 조사된 것이다. 결과적으로, 반복적인 2차원 프로젝션 후의 데이터 개수 m은 CLIP에 의한 클러스터의 데이터 수보다 감소하는 현상을 볼 수 있는데, 이는 밀도 임계값(τ′)의 크기에 따라 그리드 셀이 제외되기 때문이다. 실험에서 τ′의 값이 2%일 경우에는 데이터 개수의 변화가 거의 없는 반면, 5%나 10%일 경우에는 데이터 개수의 차이를 점차 크게 보인다. 이는 각각 사용자가 원하는 수준의 클러스터가 조사된 것을 의미한다.

5. 결 론

본 논문에서는 고차원 데이터 공간에서 효과적으로 클러스터를 식별하기 위한 부분차원 기반의 클러스터링 기법인 CLIP 알고리즘을 제안하였다. CLIP에 의해 클러스터링을 할 때의 효과성은 선형변환 프로젝션을 이용함으로써 클러스터의 형성에 전체차원이든, 부분차원이든 간에 연관성이 있는 모든 차원을 식별할 수 있음을 〈표 1〉, 〈표 2〉를 통하여 보였다. 또한 클러스터링의 효율성 측면에서 데이터베이스 크기 증가와 데이터 공간의 차원 증가에 대한 클러스터링 수행시간을 각각 〈그림 9〉와 〈그림 10〉에서 보였다. 결과적으로 CLIP은 PROCLUS보다 약 10배, CLIQUE보다 약 100배 이상 빠른 수행 속도를 보였다. 그리고 〈표 3〉의 클러스터 정제 실험에서는 반복적인 2차원 프로젝션 기법을 통하여 식별된 클러스터를 정제함으로써 클러스터의 실제 형태를 예측하고, 고차원의 잡음을 재필터링하며, 사용자가 원하는 수준의 클러스터를 찾도록 할 수 있음을 보였다. 실험 결과들을 통하여 제안된 CLIP 알고리즘의 성능은 기존의 알고리즘보다 여러 가지 측면에서 좀더 효율적이고 진보된 형태의 알고리즘임을 입증하였다.

그러나 클러스터의 품질을 더욱 개선하기 위해서는 그리드 단위의 크기, 임계 밀도, 유사한 영역간의 병합 등에 관한 추가적인 연구가 필요하다. 또한 본 논문에서는 CLIP의 우수성에 대한 타당성 입증 작업을 실험적인 차원에서 수행하였으나, 실제적인 차원에서 효용성 및 효과성을 보이기 위한 연구가 요구된다. 따라서 현재 2차적인 연구로서 전자상거래의 고객 관계 관리와 CLIP을 연계하여 고객 분류 및 선호 제품 추천 기능, 판맨 전략 수립 및 수요 예측 기능 등을 수행하는 시스템 개발을 진행하고 있다.

참 고 문 헌

[1] Charu C. Aggrawal, Cecilia Procopiuc, Joel L. Wolf, Philip S. Yu, and Jong Soo Prk, "Fast Algorithms for Projected Clustering." *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, pp. 61-72, 1999.

[2] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan, "Automatic Subspace Clustering on High Dimensional Data Mining Applications," *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, pp. 94-105, 1998.

[3] S. Berchtold, D. A. Keim, C. Böm, H.-P. Kriegel, "A Cost Model For Nearest Neighbor Search in High-Dimensional Data Space," *Proc. of the 16th Symposium on Principles of Database Systems (PODS)*, pp. 78-86, 1997.

[4] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Database with Noise," *Proc. of Int. Conf. on Knowledge Discovery and Data Mining*, 1996.

[5] Christos Faloutsos, "Fast Searching by Content in Multimedia Database," *Data Engineering Bulletin*, 18(4), 1995.

[6] Hinneburg A., Keim D. A, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," *Proc. of 4th Int. Conf. on Knowledge Discovery and Data Mining*, 1998.

[7] Mehlhorn K., Naher S., "LEDA, a library of efficient data types and algorithms," University of Saarland, FB Informatik, TR A 04/89.

[8] Raymond T. Ng, Jiawei Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," *Proc. of 20th Int. Conf. on VLDB*, pp. 144-155, 1994.

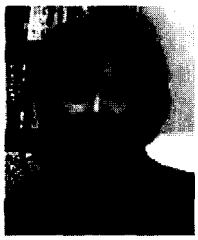
[9] Wei Wang, Jiong Yang, and Richard Muntz, "STING: A Statistical Information Grid Approach to Spatial Data Mining," *Proc. of 23rd Int. Conf. on VLDB*, pp. 186-195, 1997.

[10] M. Zait, H. Messatfa, "A comparative study of clustering methods." *Future Generation Computer Systems*, 13(2-3), pp. 149-159, 1997.

[11] Tian Zhang, Raghu Ramakrishnan, and Miron Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases." *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, pp. 103-114, 1996.

[12] Xiaowei Xu, Martin Ester, Hans-Peter Kriegel and Jorg Sander, "A Nonparametric Clustering Algorithm for Knowledge Discovery in Large Spatial Databases." *Proc. of IEEE Int. Conf. on Data Engineering*, 1998.

저 자 소 개



이해명 (E-mail : hmlee@kmc.ac.kr)
1989. 명지대학교 전자계산학(공학사)
1993. 명지대학교 전자계산학(석사)
2002. 명지대학교 컴퓨터공학(박사)
1998. ~ 현재 경문대학 인터넷미디어정보과 조교수
관심 분야 : 데이터마이닝, 웹 DB, 전자상거래 등

정종진 (E-mail : jjjung@daejin.ac.kr)
1992. 인하대학교 전자계산학(공학사)
1995. 인하대학교 전자계산공학(석사)
2000. 인하대학교 전자계산공학(박사)
1998 ~ 2002. 경문대학 인터넷미디어정보과 조교수
2002 ~ 현재 대진대학교 컴퓨터공학과 조교수
관심 분야 : 데이터마이닝, 전자상거래, 지능형 에이전트, 지식 기반 스케줄링 등