고차원 데이타에 대한 투영 클러스터링에서 특성 가중치 부여

(Feature Weighting in Projected Clustering for High Dimensional Data)

박 종 수 †

(Jong Soo Park)

요 약 투영 클러스터링은 고 차원 데이타집합에서 서로 다른 부분공간들에서 클러스터들을 찾으려고 모색한다. 사용자가 출력 클러스터들의 개수와 투영 클러스터들의 부분공간의 평균 차원수를 지정하지 않아도, 거의 최적인 투영 클러스터들을 탐사해내는 알고리즘을 제안한다. 클러스터링의 각 단계에서 알고리즘의 목적 함수는 투영 에너지, 품질, 그리고 이상치들의 개수를 계산한다. 클러스터링에서 투영 에너지를 최소화하고 품질을 최대화하기 위하여, 전체 차원의 표준 편차들을 비교함으로 입력 점들의 밀도 상에서 각 클러스터의 최선의 부분영역을 찾기 시작한다. 부분공간의 각 차원에 대한 가중치 요소가 투영 거리 측정에서 확률 오차를 없애기 위하여 사용된다. 제안된 알고리즘이 투영 클러스터들을 정확하게 발견해내고 대 용량의 데이타 집합에서 비례확장성을 갖는다는 것을 여러 가지 실험으로 보여준다.

키워드 : 분할, 합병, 클러스터링, 투영 클러스터링, 가중치, 알고리즘

Abstract The projected clustering seeks to find clusters in different subspaces within a high dimensional dataset. We propose an algorithm to discover near optimal projected clusters without user specified parameters such as the number of output clusters and the average cardinality of subspaces of projected clusters. The objective function of the algorithm computes projected energy, quality, and the number of outliers in each process of clustering. In order to minimize the projected energy and to maximize the quality in clustering, we start to find best subspace of each cluster on the density of input points by comparing standard deviations of the full dimension. The weighting factor for each dimension of the subspace is used to get rid of probable error in measuring projected distances. Our extensive experiments show that our algorithm discovers projected clusters accurately and it is scalable to large volume of data sets.

Key words : split, merge, clustering, projected clustering, weighting, algorithm

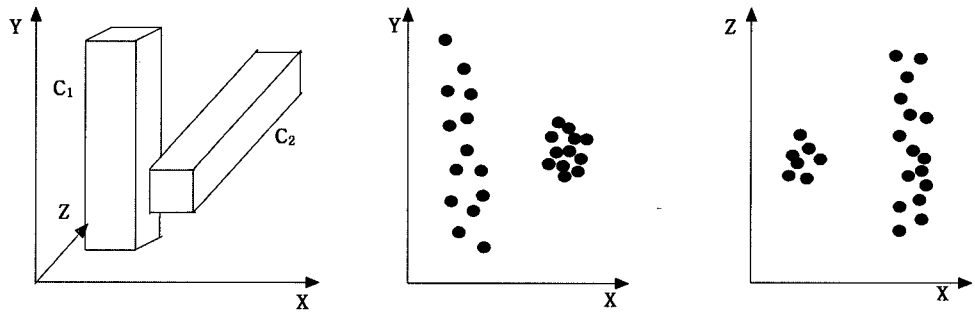
1. 서 론

대 용량의 데이타를 분석하는 방법으로 연관성(association) 분석, 분류(classification), 클러스터링(clustering) 등이 있다[1]. 데이타를 구성하는 객체들에 대한 어떤 분류 정보도 없이 데이타를 분석하는 방법으로 클러스터링을 사용하고 있다. 클러스터링의 입력은 보통 어떤 측정치들의 벡터나 또는 다차원 공간상의 점들을 나타내는 객체들(objects)이고, 출력은 몇 개의 클러스터들(clusters)로 나누어진다[2,3]. 한 클러스터에 속한 객체들 사이에는 높은 유사성을 가지게 되고, 각기 다른 클러스터에 속한 두 객체들 사이에는 낮은 유사성 또는 큰 상이성을 가지게 된다. 유사성은 객체를 구성하는 속성 값에 근거하여 계산되고, 보통 거리(distance) 척도를 사용한다. 본 논문에서 다루는 투영 클러스터링에서, 각 클러스터에 속한 객체들 사이의 유사성 계산은 그 클러스터의 특정한 속성들에만 근거하여 거리를 계산하게 된다. 이런 속성들을 전체 차원에 비교하여 그 클러스터의 부분차원이라 부른다. 보통 대 용량의 데이타를 클러스터링으로 분석하면 각 클러스터는 특유의 부분차원에서만 그 특징을 가지는 것이 일반적이다. 클러스터링 연구의 기여 분야는 데이타 마이닝, 통계학, 기계 학습, 공간 데이타베이스 기술, 생물정보학, 그리고 마케팅

· 이 논문은 2004년도 성신여자대학교 학술연구조성비 지원에 의하여 연구되었음

† 종신회원 : 성신여자대학교 컴퓨터정보학부 교수
jpark@sungshin.ac.kr

논문접수 : 2004년 10월 28일
심사완료 : 2005년 3월 17일



(a) 3차원 공간에서 패턴 (b) X-Y 평면으로의 투영 (c) X-Z 평면으로의 투영

그림 1 투영 클러스터링(projected clustering)의 특성

등이 포함된다[1].

관계없는 속성들은 고 차원 클러스터들에 잡음을 추가하여 전통적인 클러스터링 기법을 부적절하게 만든다. 투영 클러스터링 알고리즘은 숨겨진 부분차원들에서 클러스터들을 찾으려고 제안되었다[3,4]. 투영 클러스터링의 특성을 설명하는 그림 1은 3차원 공간에서 정의된 점들을 X-Y와 X-Z 2차원 평면에 투영한 분포를 보여주고 있다[4,5]. 점들은 크게 두 그룹으로 이루어지고 있다. 왼쪽 그룹 C_1의 점들은 Y축으로 넓게 퍼져 있고, X-Z 평면에서는 서로 가까이 있음을 알 수 있다. 비슷한 관점에서, 오른쪽 그룹 C_2에 속하는 점들은 3차원의 부분차원인 X-Y 평면에서 밀집되어 있다. 이 그림에서 전체 차원인 3차원으로 클러스터링을 하게 되면 바른 결과가 나오지 않게 되고, 각 클러스터에 속하는 부분차원으로 클러스터링을 하게 되면 최적의 결과인 두 클러스터들 C_1과 C_2를 얻을 수 있다. 이러한 고 차원의 데이타에서 부분차원으로 클러스터를 발견하는 알고리즘들이 많이 연구되어져왔고, 이런 방법에 대한 특성 분류와 검토가 최근에 이루어지고 있다[3].

이제까지 제안된 투영 클러스터링 알고리즘들은 각 클러스터의 부분차원을 결정하는 문제와 클러스터들의 개수를 정하는데 있어서 가장 큰 문제점을 가지고 있다[3,6,7]. 본 논문에서 이런 문제점을 해결하려고 최적의 투영 클러스터링 문제를 정의하고 하나의 해결책으로 휴리스틱 알고리즘을 제안한다. 제안된 알고리즘의 주요 특징은 다음과 같다:

첫째, 출력 클러스터들의 개수를 사용자가 지정하지 않고 알고리즘 자체에서 얻어내도록 하였다. 클러스터링하는 과정에서 세 개의 서로 다른 측정치로 투영 에너지, 품질, 이상치의 변화율을 적용하여 투영 클러스터들의 개수를 결정해낸다.

둘째, 클러스터링의 투영 에너지와 품질 계산에서 각 클러스터의 부분차원이 중요한 요소가 되며, 그 결과로 각 클러스터의 부분차원으로 선택되는 차원들이 클러스터링의 정확도에 크게 영향을 미친다. 각 클러스터의 부분차원에 속한 차원들의 평균 개수를 사용자가 미리 지정하지 않고서, 오름차순으로 정렬된 표준 편차에 근거하여 부분차원을 결정하였다. 선택된 부분차원의 확률 오차를 줄이기 위하여, 두 점 사이의 투영 거리를 계산할 때 부분차원에 속한 차원의 표준 편차에 따라 가중치를 부여하여 계산한다.

셋째, 제안된 알고리즘은 집괴적 계층적(agglomerative hierarchical) 클러스터링 방법에 속한다[1,3]. 이 방법은 한 단계에 하나씩의 클러스터를 줄이면서 일정한 개수의 클러스터들이 될 때 까지 합병을 계속해간다. 이 방법에서 두 개의 클러스터를 합병할 때에 두 클러스터의 유사성(similarity)을 투영 에너지와 품질을 사용하여 새로 정의하였다.

그리고, 새로운 알고리즘에서 클러스터링의 결과의 정확도를 측정하기 위하여 입력 클러스터와 출력 클러스터의 관계를 나타내는 혼돈 행렬(confusion matrix)을 기반으로 하여 우세 비율(dominant ratio)을 계산하였다[4,5,8]. 한 점이 어느 클러스터에도 속하지 않는 경우에 이를 이상치(outlier)라고 부르고, 우세 비율 계산에 이 상치들도 포함시켜 계산하였다.

본 논문의 구성은 다음과 같다. 2절에서는 투영 클러스터링에 관한 연구의 흐름을 살펴보고, 이 연구에서 정의되었던 클러스터링 과정에서 성능 평가를 위해 측정되는 주요 용어인 투영 에너지, 품질, 이상치에 대하여 식으로 기술한다. 3절에서는 고차원 데이타에 대한 최적의 투영 클러스터링 문제를 정의하고 그 해결책에 대하여 고찰한다. 그리고, 제안된 알고리즘 SAM에 관해 설명하고 주요 함수들을 상세히 설명한다. 알고리즘 SAM의 타당성과 성능을 평가하기 위하여 4절에서 수행되었던 실험 결과를 보여준다. 마지막 절에서는 알고리즘의 특징을 기술하고 앞으로 연구 방향을 제시한다.

2. 기존 알고리즘의 검토와 성능 평가를 위한 척도

2.1 투영 클러스터링 알고리즘들의 검토

고 차원 데이타의 투영 클러스터링에 관한 연구가 많이 이루어지고 있다. CLIQUE[9]는 데이타 집합의 부분 공간 내에서 클러스터들을 찾으려고 시도하기 위하여 제안된 초기 알고리즘들 중의 하나다. 최근에 고 차원 데이타에서 부분공간 클러스터들을 찾아내는 알고리즘들을 검토하여 하향식 탐색 반복 방법(top down search iterative methods)와 상향식 탐색 그리드 기반 방법(bottom up search grid based methods)으로 분류하여 설명하고 있다[3]. 본 논문에서 비교 분석하는 알고리즘들인 PROCLUS[4]와 DOC[7]은 두 가지 방법론에 각각 속하고 있다.

알고리즘 PROCLUS[4]는 고 차원 데이타를 효과적으로 클러스터링하는 새로운 접근 방식을 제안하였다. 이전에 발표되었던 CLIQUE[9] 알고리즘에 비해서 보다 작은 개수의 클러스터들을 찾아내고 정확도 면에서 우수함을 보여주고 있다. 그러나, 알고리즘 PROCLUS도 무작위로 선택되는 초기 medoid(discrete median or representative)들의 위치에 따라서 클러스터링 결과가 많이 달라질 수 있다. 알고리즘 PROCLUS를 개선하여 DOC[7]와 SUBCLUS[5]가 발표되었다. DOC 알고리즘은 무작위로 medoid에 해당하는 점을 찾아서 일정한 영역안의 점들을 하나의 클러스터에 속하게 하여 클러스터들의 품질을 계산하는 공식을 적용하여 일정한 값 이상의 클러스터링이 될 때까지 계속하여 무작위로 medoid를 선택한다. DOC의 실행 시간이 길어진다는 단점을 개선하기 위하여, 품질 저하를 감수한 FASTDOC도 제안하였다. DOC과 FASTDOC에서 부분차원을 결정하는 중요한 매개변수인 경계 상자(bounding box)의 넓이를 계산해내는 방법이 실제적이지 못하다. 4.4절에서 경계 상자의 넓이에 따른 성능 평가를 수행하였다. DOC를 개선한 알고리즘으로 MineClus[8]가 제안되었고, 이 알고리즘은 빈발 항목집합을 찾아내는 알고리즘을 응용하여 보다 빠르게 클러스터들을 찾아내고 있다. SUBCLUS는 PROCLUS와 ORCLUS[6]의 장점을 적용하여 작은 클러스터들을 삭제하고 새로운 밀접도를 사용한 합병 방법을 제안하여 클러스터링을 효과적으로 개선하였다. 그러나, 기존의 투영 알고리즘들은 부분차원을 결정하는 문제와 최종적인 클러스터들의 개수를 사전에 미리 매개변수로 지정하는 문제가 있다.

2.2 기호와 성능 평가를 위한 척도

먼저 부분차원과 투영 거리를 정의하고, 투영 클러스터링의 성능을 평가할 수 있는 척도로 세 종류의 척도들(measurements)인 투영 에너지, 품질, 그리고 이상치들의 집합에 대해서 기술한다. 원래의 d-차원 공간에서 두 점들을 $\overline{y_1}=(y_{1,1},\cdots,y_{1,d})$ 과 $\overline{y_2}=(y_{2,1},\cdots,y_{2,d})$ 라 하자. 전체 차원을 D로 표기하고, 전체 차원수 $d=|D|$로 나타낸다. 그러면 부분차원 $D_0(D_0\subseteq D)$ 상에서 두 점 $\overline{y_1}$과 $\overline{y_2}$ 사이의 투영 거리(projected distance)는 $P_{dist}(\overline{y_1},\overline{y_2},D_0)$로 표기되고 이것은 D_0에 의해 대표되는 $|D_0|$-차원 공간상에서 투영 $P(\overline{y_1},D_0)$와 $P(\overline{y_2},D_0)$ 사이의 거리를 나타낸다[6]. 부분차원 D_0에 속한 차원들의 개수를 부분차원수라 부르고, 기호는 $|D_0|$로 표기한다. 본 논문에서는 투영 거리를 다시 부분공간의 차원수로 나누어진 투영 가중 유클리디언 단편 거리(Projected, weighted euclidean segmental distance)는 다음과 같이 정의한다:

$$P_{wesd}(\overline{y_1},\overline{y_2},D_0)=\frac{\sqrt{\sum_{i\in D_0} w_i(y_{1,i}-y_{2,i})^2/W}}{|D_0|},$$

여기서 $W=\sum_{i\in D_0} w_i$.

이 식에서 w_i는 부분차원 D_0에 속한 i번째 차원의 가중치이고, W는 가중치의 전체 합이다. 선택된 부분공간이 많은 클러스터에서 투영 거리가 커지는 것을 억제하여, 다음에 정의되는 투영 에너지에서 부분공간이 많은 클러스터들이 더 좋은 클러스터링이 되도록 유도하고 있다. 이것은 DOC[7]에서 각 클러스터의 품질을 결정하는 공식에서 차원수를 지수 승으로 하여 차원수를 중요하게 한 것과 같이 식 $P_{wesd}(\overline{y_1},\overline{y_2},D_0)$도 차원수가 많아짐에 따라 거리가 늘어날 수 있는 소지를 없애주기 위하여 유클리디언 거리에서 차원수로 나누어주어서 차원수가 증가함에 따른 증가된 거리의 영향을 없애도록 하였다.

부분차원 $D_i(D_i\subseteq D)$에서 클러스터 $C_i=\{\overline{x_1},\overline{x_2},\cdots,\overline{x_t}\}$의 투영 에너지(projected energy)는 $E(C_i,D_i)$로 표시되고, C_i에 속한 모든 점들이 부분차원 D_i에 투영될 때 $E(C_i,D_i)$는 그 클러스터의 중심점(centroid)과 점들의 사이의 투영 가중 유클리디언 단편 거리를 평균 제곱한 거리로 다음과 같이 정의된다:

$$E(C_i,D_i)=\sum_{j=1}^{t}\{P_{wesd}(\overline{x_j},\overline{X}(C_i),D_i)\}^2/t,$$

여기서 중심점 $\overline{X}(C_i)=\sum_{j=1}^{t}\overline{x_j}/t$.

이 식의 의미는 투영시키는 부분공간에서 중심점과 각 점 사이의 거리의 곱을 더해서 점들의 개수로 나누었으므로, 통계치 중의 하나인 분산(variance)의 의미를

내포하고 있다. k 개의 클러스터들의 투영 에너지를 다음과 같이 정의한다:

$$E_k = \sum_{i=1}^{k} E(C_i, D_i) = \sum_{i=1}^{k} \sum_{j=1}^{t_i} \{ P_{wesd}(\overline{x_j}, \overline{X}(C_i), D_i)\}^2 / t_i,$$

여기서 $t_i = |C_i|$.

좋은 클러스터링은 전체적으로 투영 에너지를 작아지도록 하고 있지만, 다음의 경우는 최소의 E_k가 항상 최적의 클러스터링을 나타내는 것이 아니라는 것을 알 수 있게 한다. 하나의 클러스터가 대략 반으로 나누어지는 경우의 투영 에너지를 고찰해보자. 하나의 클러스터가 두 개의 클러스터로 되면, 각 점과 중심점 사이의 거리가 줄어들게 된다. 따라서, $P_{wesd}(\overline{x_i}, \overline{X}(C_i), D_i)$는 작아지게 된다. 결과적으로 한 개의 클러스터의 투영 에너지 E_1은 두 개의 클러스터들의 투영 에너지 E_2 보다 더 큰 값을 갖게 된다. 우리가 원하는 클러스터는 두 개로 나누어진 클러스터들이 아니라 한 개로 합병된 클러스터를 원하고 있다. 최소의 값을 갖는 E_k를 갖는 클러스터들을 찾을 경우에는 분할된 클러스터들을 얻을 수도 있다.

클러스터의 품질(quality)은 [7]에서 제시된 것과 같이 많은 점들을 포함하고 부분차원에 속한 차원들을 많이 포함할수록 더 좋은 클러스터로 여겨지고 있다. 투영 클러스터 C_i의 품질 Q는 점들의 개수 $|C_i|$와 부분차원수 $|D_i|$로 다음과 같이 정의된다[7]:

$$Q(C_i, D_i) = |C_i| \cdot (1/\beta)^{|D_i|}.$$

여기서 $\beta \in (0, 1]$는 그 클러스터의 크기(size)에 우선하여 부분차원수의 중요성을 반영한다. 작은 β의 값은 큰 β 값에 비해 부분차원수를 더 크게 품질 값에 반영한다. 큰 β는 높은 부분차원수를 갖는 작은 클러스터보다 더 적은 부분차원수를 가진 큰 클러스터를 유리하게 한다. 그 역도 성립한다. k 개의 클러스터들의 품질은 다음과 같이 정의된다:

$$Q_k = \sum_{i=1}^{k} Q(C_i, D_i) = \sum_{i=1}^{k} |C_i| \cdot (1/\beta)^{|D_i|}.$$

클러스터링에서 품질 Q를 크게 하는 것이 좋다. 부분차원에 속한 원소들의 개수가 클러스터링 품질에 지대한 영향을 끼치므로 각 클러스터의 부분차원을 결정하는 데 정교한 작업이 필요하다.

어느 클러스터에도 속하지 않는 점을 이상치(outlier)라 부르고, 이상치들의 집합을 O로 표기한다. 한 점 p와 k개의 클러스터들의 중심점 사이의 투영 가중 유클리디언 단편 거리 중에서 최소 거리를 δ_p라 하면, 이상치들의 집합 O_k의 정의는 다음과 같다:

$$O_k = \{ p \mid \delta_p > B\sigma \text{ where } \delta_p = \min_i \{ P_{wesd}(p, s_i, D_i) \} \text{ for } 1 \le i \le k \}.$$

여기서 B는 상수이고 σ는 최소 투영 거리를 가지는 클러스터에 속한 점들의 분포에서 표준 편차를 나타낸다. 한 점에서 중심점까지 최소의 투영 거리가 일정 배수의 표준 편차(예를 들면, 4σ)를 벗어나는 점을 이상치로 두고, 이런 점들의 집합이 O로 표기한다. 이와 같은 정의는 [4,6]에서도 비슷하게 정의하였지만, 실제 실험 결과에서는 적용하지 않았다. 보통 부분차원수 $|D_i|$가 1 이상인 다차원을 고려하므로 제안된 알고리즘에서는 표준 편차 σ 대신에 가중 표준 편차(weighted standard deviation)를 적용한다.

3. 알고리즘 SAM

이 절에서, 최적의 투영 클러스터링의 문제에 대해서 고찰하고, 그리고 그 문제를 해결하는 휴리스틱의 하나로 최적의 투영 클러스터들을 위한 필요 조건을 고려하여 새로운 알고리즘의 기본 아이디어로 삼는다. 3.2절에서는 제안된 알고리즘을 설명하고, 그 알고리즘의 각 함수에 대한 설명이 뒤따른다.

3.1 최적의 투영 클러스터링을 위한 직관 및 필요 조건

계층적인 클러스터링에서, 초기 분할 개수인 k_0(보통 입력 객체들의 개수인 n보다 훨씬 작고 결과로 얻어지는 클러스터들의 개수보다 몇 배수인 값)개의 클러스터들은 최종 출력 클러스터들에 비해 충분히 작아서 각 출력 클러스터의 특성을 가지는 종자 클러스터(seed cluster)가 존재한다고 가정한다. 그러면, k_0개의 투영 클러스터들에서 합병 함수를 통하여 마지막 한 개의 클러스터가 남을 때 까지 합병을 계속한다. 최적의 투영 클러스터링 문제는 앞에서 정의된 용어를 사용하여 직관적으로 다음과 같이 정의할 수 있다: n개의 점들(또는 객체들)로 이루어진 고 차원 데이타 집합에서 클러스터들의 개수 k가 초기 값 k_0에서 1까지 감소하는 과정에서, 두 척도인 투영 에너지 E_k와 이상치들의 집합인 O_k를 변수로 하는 함수 $f(E_k, O_k)$의 값을 최소화하는 조건으로 클러스터링의 품질 Q_k를 최대화하는 k개의 투영 클러스터들을 찾는다. 이를 다시 식으로 표현하면 다음과 같다:

$$\text{optimal clustering quality} = \textit{maximum}_{k_0 \le k \le 1} Q_k \text{ subject to } \textit{minimum } f(E_k, O_k).$$

여기서, 함수 $f(E_k, O_k)$는 O_k에 속한 점들이 투영 에너지를 계산하는 비슷한 방법으로 계산된 후에 투영 에너지 E_k와 결합하는 연산으로 계산되게 한다. 분류의 한 방법인 의사결정 트리 귀납에서 엔트로피 감소가 최대가 되는 속성에서 분할 노드를 택하는 것처럼[1], 함

수 $f(E_k, O_k)$는 투영 클러스터링에서 합병 단계의 엔트로피(entropy) 개념이 되도록 한다. 각 척도를 계산하는 기본 과정에서 각 클러스터의 부분차원이 고려되는데, 이 부분차원을 결정하는 문제도 전체 클러스터링의 품질 문제와 연관된다. k가 감소함에 따라 두 클러스터가 하나의 클러스터로 합병하는 과정이 필요한데, 합병 결과에 따라 클러스터링의 품질이 달라질 수 있으므로 최적의 솔루션을 위해서 되추적(backtracking)이나 분기한정(branch-and-bound) 기법으로 해결을 시도할 수도 있다. 이와 같은 이유와 더불어, 주어진 고차원 데이타에 대해서 최적의 투영 클러스터링을 수행하는 작업은 차원의 저주(curse of dimensionality)와 실제 데이타에서의 객체들의 고유한 희박성(inherent sparsity of objects)으로 인하여 아주 어려운 문제에 속한다[1,3,4].

본 논문에서는 최적의 투영 클러스터링 문제를 해결하는데 기여하기 위하여 실용적인 문제 해결 접근 방법으로 휴리스틱 알고리즘(heuristic algorithm)을 제안한다. 계층적 방식으로 투영 클러스터링을 수행할 때, 두 클러스터가 하나의 클러스터로 되는 합병 과정에서 각 클러스터의 크기나 그것에 속한 점들의 분포에 따라 클러스터링의 품질이 달라져서 계산상의 최대 품질이 사용자가 원하는 결과와 항상 일치하는 것은 아니다. 최소화되어야 하는 투영 에너지와 이상치들의 집합의 크기에 영향을 받아서, 실험 데이타에서 최대 품질이 최대 우세 비율이 되지 않는 경우도 간혹 있다. 여러 가지 실험의 분석과 최적의 클러스터링 품질 문제의 정의에 근거하여, 본 논문에서는 클러스터링의 품질이 최대점이면서 동시에 최대 우세 비율이 되도록 합병 함수를 만들어 최적의 클러스터들을 찾아내는 방법론을 연구하였다. 클러스터링의 품질이 최대점이 되게 하는 필요조건을 설정하기 위하여 품질, 투영 에너지, 그리고 이상치들의 집합의 변화율을 분석하여 알고리즘의 기본 아이디어로 하였다. 변화율의 분석은 표 1과 그림 2를 통하여 설명하고, 알고리즘의 구체적인 단계는 그림 3에서 기술한다.

계층적인 클러스터링의 합병 단계에서 측정되는 값들의 한 예제를 보기로 하자. 표 1과 그림 2는 2.2절에서 언급한 세 개의 척도들을 합병 단계별로 나타내고 있다. k는 각 단계에서 클러스터들의 개수를 나타낸다. 이 예제에서 초기 분할 개수 $k_0=24$이므로, k는 24에서 한 개의 클러스터인 1이 될 때 까지 감소한다. 각 합병 단계에서, 투영 에너지는 E_k로, 클러스터들의 품질의 합은 Q_k로, 이상치들의 집합에 속한 점들의 개수는 $|O_k|$로 나타내고 있다. 각 단계에서 클러스터링 결과를 입력 점들과 비교하여 얻어지는 값인 우세 비율(dominant ratio)은 DR로 표시하고 있다. 그림 2는 표 1의 값을

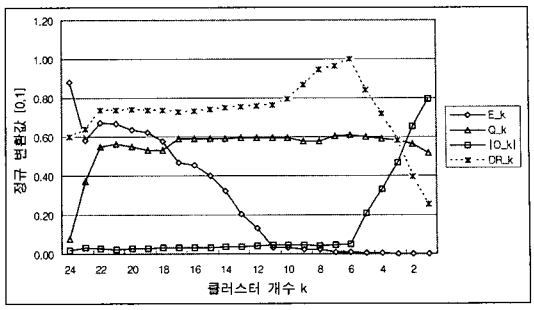


그림 2 합병 단계별로 0과 1 사이의 값으로 정규화된 관련 값의 변화

[0, 1] 사이의 값으로 정규화하여 나타내주고 있다. E_k는 10^{-2}로 나누었고, Q_k는 1,000으로 나누었고, $|O_k|$는 10,000으로 나누어 정규화하였다. 우세 비율 DR_k는 결과적으로 얻어진 퍼센티지를 [0, 1] 사이의 값으로 보여주고 있다.

그림 2에서 클러스터링의 각 단계의 성능 평가치는 점선인 DR_k의 값을 살펴보면, 주어진 데이타에서는 클러스터들의 개수가 6일 때 최적의 결과를 얻는 것을 알 수 있다. 그림 2와 이 단계에서 최적의 의미는 표 1에서 우세 비율 중에서 최대값을 갖는 합병 단계의 상태를 나타낸다. 그러나, 이 성능 평가치는 입력 데이타의 클러스터들을 알고 있는 경우에만 계산이 가능하다. 클러스터링을 수행하고 있는 각 단계에서의 측정치인 투영 에너지 E_k, 품질 Q_k, 그리고 이상치들의 개수인 $|O_k|$ 중에서 어느 한 가지 값으로는 k의 어느 값에서 최적의 클러스터들을 얻을 수 있는지 알 수 없다. 세 개의 서로 다른 값들의 변화율을 깊이 분석하여, 최적의 클러스터들을 탐지해내는 필요 조건들을 설정할 수 있다. 이런 상황은 세 개의 변화율이 동시에 만족해야만 한다. 첫째, 이상치들의 개수가 급격하게 증가한다. 최적의 클러스터들의 개수에서 한 클러스터가 줄어들면, 해당되는 클러스터에 속한 점들은 이상치들의 집합인 O에 속하게 되어, 이전 합병 단계의 이상치들과 비교하면 그 변화율이 급격하게 커진다. 표 1에서 합병 단계 19에서 k가 6에서 5로 변하고 $|O_5|/|O_6| = 4.23$이 된다. 둘째, 최적의 클러스터링의 다음 단계인 합병 단계 19에서 한 클러스터가 O에 속하게 되므로, 클러스터링의 품질 Q_k는 나빠지게 된다. 가장 단순한 방법으로 품질이 최대가 되는 단계를 최적의 클러스터링 결과로 정하는 것이 좋지만, 합병하는 전 단계와 후 단계에서의 품질이 미세하게 변화를 일으켜서 최적의 클러스터링 바로 전 단계에서 최대의 품질 값을 얻는 경우도 발생한다. 이런 경우를 배제하기 위하여, 다른 조건들도 고려한다. 세 번째

조건으로는 클러스터가 줄어들면서 투영 에너지 E_k도 작아지게 된다. 그림 2에서 E_k는 초기에 값이 진동하다가 지속적으로 작아지고 있다. 그 이유는 클러스터들의 개수가 작아지면 2,2절의 식 E_k도 작아짐을 알 수 있다. 세 가지 조건들을 종합해보면, 그림 3의 do-while 문장 내부의 if-조건 문장에서 보여주는 이상치들의 크기 변화율, 품질의 변화율, 그리고 투영 에너지의 변화율을 동시에 고려하여 최적의 클러스터링이 되도록 하였다. 이상치들의 변화율이 최대이면서 동시에 품질과 투영 에너지가 떨어지는 직전의 합병 단계를 최적의 클러스터링 상태로 저장하여, 정제 단계에서 이 정보를 다시 불러 클러스터링을 완료하게 한다. 표 1에서 $k=6$인 경우의 우세 비율은 99.91%이지만, 정제 단계를 마친 후의 마지막 우세 비율은 99.93%로 조금 개선된다.

3.2 클러스터링 알고리즘

알고리즘 SAM은 그림 3에서와 같이 다섯 개의 주요 함수들로 이루어져있고, 크게 분할 단계, 합병 단계, 정제 단계인 세 단계로 구성되어 있다. 알고리즘의 이름은 Split-And-Merge에서 유래하였고, 알고리즘의 기술은 C++언어 위주로 기술하였다. 실제 실험에서는 마이크로소프트의 Visual C++ 언어로 구현하여 실험하였다. 주요 변수들은 클래스의 private 변수로 선언되었기 때문에, 각 함수에서 매개변수 전달은 외부적으로 표현되지 않았다.

기본적인 알고리즘의 전개 방식은 ORCLUS[6] 및 SUBCLUS[5]와 비슷한 유형에 속하지만, 제안된 알고리즘 SAM에서는 사용자에 의해 지정되는 클러스터들의 개수와 부분차원에 속한 차원들의 평균 개수가 미리 주어지지 않는다. 알고리즘 SAM의 첫 단계인 분할 단계는 n개의 d차원 점들로 이루어진 입력을 k_0개의 클러스터들로 나눈다. 초기 분할 개수 k_0는 결과로 예상되는 클러스터 개수의 3배 이상의 값을 사용한다. 즉, 작은 개수의 점들을 가진 여러 개의 클러스터들로 분할한다. 분할하는 방법은 SUBCLUS[5]에서 사용한 알고리즘 K-Means[1]를 사용하던지, 차원들 중에서 표준편차가 가장 큰 차원을 반으로 나누는 과정을 거쳐서 k_0개의 여러 클러스터들로 나누면 된다. 본 논문에서는 간편한 방법으로 후자를 택하여 구현하였다. 합병 단계에서는 초기 단계에서 만들어진 k_0개의 클러스터들을 합병을 통하여 하나씩 줄여가는 단계로, 그림 3에서 do-while 문장에 해당된다. 마지막 하나의 클러스터와 이상치 집합이 남을 때 까지 합병을 계속한다. 이 단계에서 앞에서 설명한 세 척도의 값들을 사용하여 최선의 클러스터링을 구한다. 클러스터들의 개수를 결정하는 방

표 1 합병단계에 따른 클러스터에 따른 관련 값들의 변화

합병단계	k	$E_k(\times 10^{-5})$	Q_k	$\lvert O_k \rvert$	$DR_k(\%)$
0	24	881.81	78,160	183	60.14
1	23	583.88	372,313	326	64.22
2	22	672.76	551,275	252	73.73
3	21	666.68	562,238	247	73.56
4	20	634.11	549,695	261	73.90
5	19	624.58	533,247	275	73.47
6	18	575.49	529,734	299	73.75
7	17	467.80	590,767	317	72.71
8	16	455.62	590,890	318	73.25
9	15	401.47	591,653	333	74.30
10	14	324.07	591,195	351	74.94
11	13	204.66	593,741	378	75.39
12	12	130.86	594,262	409	75.81
13	11	31.30	594,216	444	76.21
14	10	30.86	593,997	443	79.59
15	9	22.15	579,483	436	86.95
16	8	23.85	578,725	427	94.66
17	7	9.23	605,554	476	96.41
18	6	8.43	607,852	495	99.91
19	5	3.77	601,472	2,092	84.08
20	4	2.57	591,808	3,300	72.00
21	3	1.51	580,824	4,673	58.27
22	2	0.69	565,888	6,540	39.60
23	1	0.13	520,192	7,968	25.32

```
algorithm SAM(int k0)      // 초기 분할 개수 k0는 예상되는 클러스터들의 개수의 대략 3 배 이상의 값
begin
    k = k0;
    C current = SplitData( k);      // 분할 단계: n개의 입력 점들을 k0개의 클러스터들로 나눈다
    FindSubdimensions();
    C previous = C optimal = AssignPoints();
    ObjectiveFunction();                    // 품질 Qk와 투영 에너지 Ek를 계산
    o previous = |Ok|;                      // outlier의 개수
    q previous = Qk;                        // 현재 clustering의 quality
    e previous = Ek;                        // 현재 clustering의 projected energy
    max_ratio = 0.0;

    do {                            // 합병 단계
        MergeCluster( k);
        k--;                                // 합병 함수에서 k가 1 감소한 것을 표시한다
        FindSubdimensions();
        C current = AssignPoints();         // 현재 클러스터 집합
        ObjectiveFunction();

        if ( |Ok|/o previous > max_ratio && Qk < q previous && Ek < e previous) {
            max_ratio = |Ok|/o previous;    // outlier의 변화율
            C optimal = C previous;         // 한 단계 이전 클러스터 집합, |C previous| = k+1
        } else if ( |Ok| < o previous || Qk > q previous || Ek > e previous)
            max_ratio = 0.0;

        o previous = |Ok|;                  // outlier의 개수
        q previous = Qk;                    // clustering의 quality
        e previous = Ek;                    // projected energy
        C previous = C current;
    } while ( k > 1);

    C current = C optimal;          // 정제 단계
    C current = AssignPoints();
    ObjectiveFunction();
end
```

그림 3 알고리즘 SAM

법과 여러 내부 함수에 대한 상세한 설명은 다음 세부 절들에서 이루어진다. 마지막 단계인 정제 단계에서는 최선의 클러스터들의 집합을 불러내어 최종적으로 다시 모든 점들을 재배정하여 클러스터링을 마무리하게 된다. 각 단계의 목적 함수 *ObjectiveFunction()*에서는 클러스터링의 투영 에너지 E_k와 품질 Q_k를 2.2절의 식에 따라 계산한다.

3.3 부분차원 계산

본 논문의 특징들 중의 하나가 각 클러스터의 부분차원에 속한 차원들의 평균 개수를 사용자가 미리 지정한 매개변수를 사용하는 것[4-6]이 아니라 알고리즘 내부에서 각 클러스터의 부분차원에 속한 차원들의 개수인 부분차원수를 결정하도록 하는 것이다. 부분차원수를 결정하는 방법은 두 단계로 구성된다. 첫 번째 단계에서는 각 클러스터에 속한 점들을 사용하여 모든 차원들의 표준 편차들(standard deviations)을 구하고, 이 표준편차들을 오름차순으로 정렬한다. 다음 단계에서, 연속적인 두 표준 편차들의 변화율을 계산하여, 변화율이 일정한 값 이상인 경우에 변화율을 가장 크게 일으키는 순서의 바로 앞 인덱스까지 주어진 클러스터의 부분차원으로 결정한다. 각 점에 가장 가까운 중심점을 찾는 투영 거리를 계산할 때, 부분차원에 속한 차원들 중에서 제일

작은 표준 편차를 갖는 차원이 가장 큰 가중치를 갖고, 마지막 순서의 표준 편차를 갖는 차원의 가중치는 제일 작게 부여하여 결과적으로 가장자리에 있는 차원이 포함되는 영향을 줄이도록 한다. 각 클러스터의 부분집합은 클러스터링의 품질과 투영 에너지의 계산 그리고 이상치의 결정에 기본적인 변수로 사용되기 때문에, 적절한 부분집합의 선정은 전체 클러스터링을 위해서 아주 중대하다.

부분차원을 계산하는 함수는 그림 4에 설명되어 있다. 표준 편차의 변화율을 검사한 결과, 부분차원수가 하나인 경우에는 차원을 추가하도록 하였다. 클러스터의 부분차원수가 1인 경우는 클러스터링의 결과를 나쁘게 하는 원인이 된다. 그렇다고, 클러스터의 부분차원의 원소가 원래 하나인 경우도 있으므로 이를 견지하는 것이 바람직하다는 것도 염두에 두고서 다음 두 가지 경우를 고려하였다. 첫 번째 경우는 한 클러스터의 부분차원에 속한 첫 번째 차원의 표준 편차가 너무 작아서 최대 변화율이 되는 경우로, 이 경우에는 다음 두 번째 큰 변화율이 일정 값 이상인 표준 편차 까지 부분차원으로 정하였다. 두 번째 경우는 표준 편차의 값이 서서히 단조적으로 증가하는 경우로 연속된 두 표준 편차의 변화율이 일정한 값 이상으로 되지 않는 것이다. 이 경우에,

```
FindSubdimensions()
begin
    각 클러스터에 대하여, 모든 차원들의 표준 편차들을 구한 후에 오름차순으로 표준
    편차들을 정렬한다;
    // 정렬된 표준 편차는 클러스터 i와 부분차원에 속한 차원 loc(j)에 대한 기호로 σ_{i,loc(j)}로 표기된다
    // σ_{i,loc(j)} < σ_{i,loc(j+1)} for 1≤j<|D|, loc(j)는 j번째로 작은 표준 편차를 갖는 차원 번호, loc(j) ∈ {1,2,....,|D|}

    for ( i = 1; i <= k; i++ ) {          // k 개의 클러스터의 각각에 대해서 부분차원을 구한다
        n_s = 1;                          // 부분차원에 속한 차원들의 개수를 지정하는 변수, |D_i|
        max_ratio = σ_{i,loc(2)}/σ_{i,loc(1)};   // loc(1)은 최소의 표준 편차를 갖는 차원 번호
        for ( j = 3; j <= |D|; j++ ) {
            ratio = σ_{i,loc(j)}/σ_{i,loc(j-1)};
            if (ratio >= 2.0 && max_ratio < ratio) {
                max_ratio = ratio;
                n_s = j;
            }
        }

        if ( n_s == 1) {    // 부분차원에 속한 차원들의 개수가 하나인 경우
            if (max_ratio >= 2.0) {     // 부분차원에 속한 첫 번째 차원의 표준 편차가 아주 큰 경우
                for ( j = 3; j <= |D|; j++ ) {
                    ratio = σ_{i,loc(j)}/σ_{i,loc(j-1)};
                    if (ratio >= 2.0) {
                        n_s = j;
                        break;
                    }
                }
            } else {                    // 부분차원에 속한 차원들의 표준 편차가 서서히 증가하는 경우
                for ( j = 3; j <= |D|; j++ ) {
                    ratio = σ_{i,loc(j)}/σ_{i,loc(1)};      // 최소의 표준 편차와 비교한다
                    if (ratio >= 2.0) {
                        n_s = (1 + j + 1)/2;    // 중간 인덱스까지로 결정
                        break;
                    }
                }
            }
        }

        D_i = {loc(1),loc(2),···,loc(n_s)};   // 부분차원, σ_{i,loc(j)} < σ_{i,loc(j+1)} for 1 ≤ j < n_s
    }
end
```

그림 4 각 클러스터에서 부분차원을 찾는 함수

부분차원의 첫 번째 차원의 표준 편차와 순서대로 다른 차원들의 표준 편차와 비교하여 일정한 값 이상인 경우에는 그 중간 인덱스 값으로 부분차원수를 정하였다.

3.4 점 배정

점 배정 함수는 한 점을 가장 가까운 클러스터에 배정하거나 또는 어느 클러스터에도 속하지 않는 이상치 집합에 배정한다. ORCLUS[6]에서 이상치를 결정하는 방법은 클러스터들의 중심점 사이의 최소 거리를 사용했으므로, 중심점 간의 거리가 아주 크게 떨어져 있는 경우에는 별로 효과적이지 못하다. 수리 통계학에서 데이타의 크기 n이 커지면 중심 극한 정리(central limit theorem)에 의하여 점차 정규 분포(normal distribution)에 따른다는 것을 설명하고 있다[10]. 이 정리에 근거하여, 점 p가 가장 가까운 클러스터의 중심점까지의 투영 거리가 그 부분차원의 표준 편차의 일정 배수 이상을 벗어나는 경우에만 이상치로 두는 것이 더 합리적일 것이다. 본 논문에서는 중심 극한 이론과 각 부분차원에 속한 차원들의 가중치를 고려한 투영 거리를 계산하여, 각 점을 가까운 클러스터에 배정하거나 또는 이상치로 배정하였다.

알고리즘 SAM의 점 배정 함수는 그림 5에서 설명되고 있다. 한 점 p_i에서 현재 클러스터들의 중심점들에서 가장 가까운 중심점 사이의 투영 거리를 계산한다. 이 투영 거리가 미리 주어진 상수 B와 가중 표준 편차의 곱을 벗어나면, 그 점은 이상치(outlier)로 둔다. 그렇지 않으면, 그 점은 가장 가까운 투영 거리를 가지는 클러스터로 배정하게 된다. 한 점 p_i와 가장 가까운 중심점을 갖는 클러스터를 $C_{j_{min}}$이라하면, 최소 투영 가중 유클리디언 단편 거리는 $P_{wesd}(p_i, s_{j_{min}}, D_{j_{min}})$으로 표기되고 클러스터 $C_{j_{min}}$의 가중 표준 편차(weighted standard deviation)는 σ_{weight}로 나타내고 식은 다음과 같다:

$$P_{wesd}(p_i, s_{j_{min}}, D_{j_{min}}) = \frac{\sqrt{\sum_{j=1}^{|D_{j_{min}}|} w_j (p_{i,loc(j)} - s_{j_{min},loc(j)})^2 / W}}{|D_{j_{min}}|},$$

그리고 $\sigma_{weight} = \left(\sum_{j=1}^{|D_{j_{min}}|} w_j \sigma_{loc(j)} \right) / \left(W \cdot |D_{j_{min}}| \right)$.

여기서 $w_j = (|D_{j_{min}}| - j - 1)$로 두고, $W = \sum_{j=1}^{|D_{j_{min}}|} w_j$.

$P_{wesd}(p_i, s_{j_{min}}, D_{j_{min}})$과 σ_{weight}의 계산에서, 클러스터 $D_{j_{min}}$에 속한 차원의 표준 편차들은 오름차순으로 정렬되어 $\sigma_{loc(j)} < \sigma_{loc(j+1)}$가 되며, w_j는 j 번째 차원의 가중치이고 W는 가중치들의 합이다. $loc(j)$는 $D_{j_{min}}$에 속한 j 번째로 작은 표준 편차를 갖는 차원으로 전체 차원 D에 속한 차원 번호이다. 예를 들면, $D = \{1, 2, \cdots, |D|\}$에서 $loc(1) = 14$는 최소의 표준 편차를 갖는 차원 번호가 14이다.

모든 점들을 가장 가까운 클러스터나 또는 이상치 집

```
AssignPoints()
begin
    O = ∅;                                          // 이상치들의 집합
    for ( i = 1; i <= k; i++)
        C_i = ∅;
    for ( i = 1; i <= n; i++) {                     // 각각의 점 p_i에 대해서
        dist_min = ∞;   j_min = -1;
        for ( j = 1; j <= k; j++)     // 중심점과 최소의 투영 거리를 갖는 클러스터를 찾는다
            if ( dist_min < P_wesd(p_i, s_j, D_j) {
                dist_min = P_wesd(p_i, s_j, D_j);
                j_min = j;
            }
        if ( P_wesd(p_i, s_{j_min}, D_{j_min}) < B · σ_weight )     // B의 설정은 4.3절 참조
                C_{j_min} = C_{j_min} ∪ { p_i };                    // p_i를 클러스터 C_{j_min}에 속하게 한다
        else
                O = O ∪ {p_i};                                      // p_i를 이상치 집합 O에 속하게 한다
    }

    for ( i = 1; i <= k; i++)          // 참고문헌 [5, 11] 참조
        클러스터 C_i의 선형 합, 제곱 합, 그리고 중심점 s_i를 구한다;
end
```

그림 5 모든 점들을 클러스터에 배정하는 AssignPoints 함수

합에 배정한 후에, 다음 단계의 *MergeCluster()* 함수에서 필요한 각 클러스터의 선형 합, 제곱 합, 그리고 중심점이 다시 계산된다[5,11].

3.5 합병 함수

알고리즘 SAM은 초기 분할 개수 k_0 개의 클러스터들에서 한 개의 클러스터가 남을 때 까지 클러스터들을 합병해가는 집괴적 계층 클러스터링으로 분류된다[1,3]. 3.1절의 표 1에서와 같이 합병 함수 *MergeCluster()*가 한번 호출될 때 마다 하나의 클러스터가 줄어든다. 합병 함수에서는 최적의 투영 클러스터링에 기여가 가장 적은 클러스터를 찾아내어 삭제하거나 또는 합병하고, 다음 합병 단계로 넘어간다. 첫 단계로, 한 클러스터에 속한 점들의 개수가 극도로 작은 경우에 그 클러스터를 삭제하도록 한다. 아주 작은 개수의 점들로 구성된 클러스터의 중심점과 부분차원에 속한 차원들의 표준 편차들은 왜곡 현상을 일으켜서 전체 클러스터링 결과에 나쁜 영향을 일으킬 가능성이 아주 높기 때문이다. 최악의 경우에서는 2.2절에서 언급한 클러스터의 투영 에너지와 품질 계산에서 아주 좋은 값을 생성하여 끝까지 살아남을 가능성도 있다. 두 번째 단계에서, 일반적인 접근 방법으로 가능한 모든 쌍의 클러스터들에 대하여 유사성(similarity)을 검사하여, 그 중에서 가장 유사한 한 쌍의 두 클러스터들을 하나의 클러스터로 합병한다. 이 단계에서 제대로 합병이 이루어지지 않으면, 세 번째 단계로 남아있는 모든 클러스터들 중에서 품질이 가장 나쁜 클러스터를 삭제하도록 한다.

설명한 사항들을 정리하면, 합병 함수는 다음과 같은 세 단계로 이루어진다:

단계 1: 극도로 작은 개수의 점들을 갖는 클러스터를 삭제한다. 예를 들면, 현재 클러스터들의 점들의 개수를 평균하여, 평균 5% 이하의 개수의 점들을 갖는 클러스터들 중에서 최소의 점들을 갖는 클러스터를 삭제한다.

단계 2: 단계 1에서 삭제할 클러스터가 없으면, 이번 단계가 적용된다. 가능한 모든 조합을 구성하여 두 개의 클러스터들이 합병할 수 있는지를 검사한다. 가능한 합병 쌍들 중에서 최소의 투영 에너지를 갖는 쌍을 하나의 클러스터로 합병한다. 한 쌍의 클러스터들의 유사성은 그림 6의 *ClusterSimilarity()* 함수로 계산한다.

단계 3: 단계 2에서 합병할 클러스터들이 없는 경우에만, 남아있는 클러스터들 중에서 가장 작은 품질 값을 갖는 클러스터를 삭제한다.

3.6 유사성 함수

합병 함수의 두 번째 단계에서 두 클러스터들인 C_x와 C_y 사이의 유사성을 계산하는 함수는 그림 6에 기술되어 있다. 먼저 두 클러스터들의 부분차원들의 교집합 비율이 일정한 값 이상인 경우에만, 합병의 여부를 가늠할 한 클러스터 C_z를 가정한다. 이 클러스터의 부분차원, 중심점, 그리고 표준 편차를 계산해내어, 클러스터인 C_z가 두 클러스터들 C_x와 C_y와 비슷한 분포를 갖는지를 검사한다. 유사한 분포를 가지면, 품질에 비해서 데이타의 분포에 대한 의미가 더 큰 투영 에너지를 반환한다. 그림 6에서와 같이 비교 클러스터들인 C_x와 C_y의 투영 에너지를 합하여 반환한다. 그렇지 않으면, 무한대 값을 반환하여 두 클러스터들이 유사하지 않음

```
ClusterSimilarity(x, y)
begin
    if ( |Dx∩Dy|/|Dx∪Dy| > 0.65) {      // 교집합의 비율이 일정한 값 이상이면
        Cz = Cx∪Cy;
        Dx∩Dy에서 범위 1≤i≤|Dx∩Dy|인 중심점 s_{z,loc(i)}와 표준 편차 σ_{z,loc(i)}를 구한다;
        Dz = ∅;
        for ( i = 1; i <= |Dx∩Dy|; i++) {
            if ( σ_{z,loc(i)} > σ_{x,loc(i)} + σ_{y,loc(i)}) continue;      // 표준 편차가 너무 크면 제외한다
            if ( |s_{z,loc(i)} − s_{x,loc(i)}| < 2σ_{z,loc(i)} && |s_{z,loc(i)} − s_{y,loc(i)}| < 2σ_{z,loc(i)})
                Dz = Dz ∪ {loc(i)};          // Dx∩Dy에서 i 번째 차원 번호를 포함한다
        }

        if ( |Dz|/|Dx∪Dy| > 0.65)
            return R(Cx, Dx) + R(Cy, Dy);          // 클러스터 Cx와 Cy의 합산 투영 에너지를 반환
        else
            return ∞;                          // 두 클러스터가 전혀 유사하지 않다
    } else
        return ∞;
end
```

그림 6 클러스터 C_x와 클러스터 C_y 사이의 유사성 계산 함수

을 보여준다.

두 클러스터들 C_x와 C_y의 공통집합 $D_x \cap D_y$는 몇 가지 방법으로 계산할 수 있다. 가장 정확한 계산 방법은 두 클러스터들에 속한 모든 점들에 대하여 부분차원을 계산해내는 방법이다. 그러나, 이 방법은 시간이 많이 소비된다. 두 번째 방법으로 근사적인 방법은 두 클러스터들이 가지고 있는 통계치인 선형 합과 제곱 합을 이용하여 합병될 클러스터의 각 차원의 중심점과 표준 편차를 구하고, 그런 후에 부분차원을 구한다. 두 클러스터들의 중심점이 계산된 차원들의 중심점에서 일정 배수의 표준 편차 이내에 들어오면 공통집합에 속하도록 한다. 어느 방법으로든지 공통집합을 계산하게 되면, $|D_x \cap D_y|/|D_x \cup D_y|$로 정의된 교집합의 비율을 계산해낼 수 있다. 본 논문에서는 계산의 단순성을 선호하여 두 번째 방법을 구현하였고, 그 세부적인 과정은 그림 6에서 설명되고 있다. 교집합의 비율 계산에서, 각 클러스터의 부분차원에 속한 세 개의 차원들 중에서 최소한 2개 이상이 공통집합에 속해야 된다는 의미에서 교집합의 비율을 0.65 이상이 되도록 하였다.

4. 실험 결과

실험 데이타를 생성하여 제안된 알고리즘의 성능을 측정하였다. 실험은 3.2 GHz P-IV CPU, 2 GB DDR2 메모리, 그리고 74 GB S-ATA Hard Disk를 가진 PC에서 이루어졌다. 운영체제는 Windows XP Professional이고, 사용된 언어는 MS Visual C++이다. 실험은 제안된 알고리즘인 SAM이 최적의 투영 클러스터들을 찾아내는지와 알고리즘의 성능 평가에 역점을 두었다. 알고리즘 SAM의 초기 분할 개수 k_0 값의 변화에 따른 결과 분석, 점 배정 함수에서 사용되는 범위 B의 값에 따른 성능 평가, 그리고 다른 클러스터링 알고리즘들인 PROCLUS[4]와 FASTDOC[7]과의 성능 비교를 수행하였다. FASTDOC은 알고리즘 DOC의 느린 실행시간을 개선한 알고리즘으로, 투영 알고리즘의 대표적인 알고리즘들 중의 하나다. 이 FASTDOC의 성능이 경계 박스의 넓이의 값에 아주 종속적이라는 것을 보여준다. 마지막 세부 절에서, 입력 데이타베이스 크기에 따른 실행 시간을 측정하여 알고리즘 SAM의 실행 시간이 입력 데이타 개수에 선형적으로 비례한다는 것을 보여준다.

4.1 실험 데이타와 합병 단계별 혼돈 행렬의 예제

실험 데이타를 생성하기 위해서 [4]에서 제시된 방법으로 데이타를 만들었다. 표 2는 입력 데이타의 하나의 예로 입력 클러스터와 각 클러스터의 부분차원과 소속된 점들의 개수를 보여주고 있다. 이 데이타에서 클러스터들의 부분차원에 속한 평균 차원들의 개수는 4이다. 입력되는 점들의 개수는 10,000개(n = 10,000)이고 각 점은 25차원(d = 25)으로 이루어지고 있다[5]. 10,000개의 점들 중에서 5%인 500개는 어느 특정 부분차원에 소속시키지 않고 전체 차원 D에 골고루 분포시켜 놓아 이상치(outlier) 또는 잡음의 특성을 가지도록 하였다.

표 2 입력 데이타의 클러스터와 부분차원

클러스터	부분차원	Points
A	3 14 17	1867
B	3 4 6 9 14 16 17 21	2032
C	5 17	1592
D	16 17 24	1373
E	7 14 17 18 24	1428
F	16 17 20	1208
O	*D*	500

클러스터링의 정확도를 평가하기 위하여 혼돈 행렬(confusion matrix)을 기반으로 한 우세 비율(dominant ratio)을 사용하였다[4,5,7]. 표 2의 입력 데이타를 SAM에 적용하면, 3.1절의 표 1과 그림 2의 결과가 나오는데, 이 절에서 중간 과정을 상세히 설명하고 결과적으로 최적의 클러스터들을 발견해내는 것을 보여준다. 표 1에서 k = 9인 경우에 알고리즘 SAM의 중간 단계의 결과는 표 3에 주어졌다. 표 3의 마지막 열을 제외하게 되면, 혼돈 행렬이 된다. 마지막 열은 출력 클러스터의 부분차원을 표시하고, 부분차원에 속한 차원들의 순서는 표준 편차의 오름차순이다. 표 3의 5번째 열에서, 입력 클러스터 D의 점 1,373개 중에서 출력 클러스터 2에 6개, 클러스터 5에 97개, 클러스터 9에 1269, 그리고 출력 클러스터 *O*에 1개씩 배정된 것을 나타낸다. 표 3에서, 우세 비율은 각 입력 클러스터에 대해서 가장 많은 점들로 배정된 출력 클러스터로 쌍을 이루는 점들을 합한 값 8,695에 전체 입력 점들의 수 10,000으로 나누면 0.8695가 된다. 표 3에서 굵은 글자체(bold font)의 숫자가 우세 비율에 포함된다. 따라서, 표 3의 우세 비율은 86.95%로 나타낸다. 표 3에서 출력 클러스터 *O*에 속한 이상치들은 모두 436개가 되고, 이것은 표 1에서도 확인할 수 있다.

표 3의 혼돈 행렬에서 합병 함수의 단계 2가 적용되어 출력 클러스터 3과 클러스터 8이 합병되고, k = 8에서 합병 함수의 단계 3이 적용되어 클러스터 2가 삭제되고, k = 7에서 합병 함수의 단계 3이 적용되어 클러스터 5가 삭제되어 그 결과가 표 4에 나타난다. 표 4의 우세 비율은 99.91%가 된다. 알고리즘 SAM은 계속 클러스터 개수를 줄여가므로, 표 4에서 삭제되는 클러스터는 여섯 개의 클러스터들 중에서 품질이 가장 낮은 클

표 3 클러스터링 중간 단계(k=9)의 혼돈 행렬(confusion matrix)과 부분차원

입력 / 출력	A	B	C	D	E	F	*O*	부분차원
1	**1867**	0	0	0	0	0	0	14, 3, 17
2	0	102	7	6	0	6	48	14
3	0	0	783	0	0	0	1	5, 7
4	0	0	0	0	1428	0	0	7, 17, 14, 18, 24
5	0	0	0	97	0	230	19	17
6	0	**1930**	0	0	0	0	0	4, 14, 9, 21, 17, 6, 3, 16
7	0	0	0	0	0	**971**	0	17, 16, 20
8	0	0	**799**	0	0	0	1	5, 17
9	0	0	0	**1269**	0	0	0	16, 24, 17
O	0	0	3	1	0	1	**431**	*D*

표 4 클러스터링 중간 단계(k=6)의 혼돈 행렬과 부분차원

입력 / 출력	A	B	C	D	E	F	*O*	부분차원
1	1867	0	0	0	0	0	0	14, 3, 17
2	0	0	1592	0	0	0	7	5, 7
3	0	0	0	0	1428	0	0	7, 17, 14, 18, 24
4	0	2032	0	0	0	0	0	4, 14, 9, 21, 17, 6, 3, 16
5	0	0	0	0	0	1207	0	17, 16, 20
6	0	0	0	1372	0	0	0	16, 24, 17
O	0	0	0	1	0	1	493	*D*

표 5 클러스터링 중간 단계(k=5)의 혼돈 행렬과 부분차원

입력 / 출력	A	B	C	D	E	F	*O*	부분차원
1	1867	0	0	0	0	0	0	14, 3, 17
2	0	0	0	0	1428	0	0	7, 17, 14, 18, 24
3	0	2032	0	0	0	0	0	4, 14, 9, 21, 17, 6, 3, 16
4	0	0	0	0	0	1208	0	17, 16, 20
5	0	0	0	1373	0	0	0	16, 24, 17
O	0	0	1592	0	0	0	500	*D*

표 6 데이타베이스 크기와 초기 분할 개수 k_0의 변화에 따른 성능 비교

DB 크기 / k_0		12	18	24	30	36	42
10K	우세 비율(%)	99.93	99.93	99.93	99.93	99.93	99.93
	실행시간(sec)	0.563	0.968	1.485	1,906	2.625	3.469
100K	우세 비율(%)	85.70	99.99	99.99	99.99	99.99	99.99
	실행시간(sec)	5.235	9.375	14.109	18.609	24.391	30.328

러스터 2가 합병 함수의 단계 3에서 선정되어 삭제된다. 그 결과가 표 5에서 보여준다. 표 5의 우세 비율은 84.08%가 된다. 표 5에서 계속해서 한 개씩의 클러스터가 삭제되고, 그 삭제 순서는 클러스터 4, 클러스터 5, 클러스터 1, 클러스터 2가 된다. 마지막으로, 표 5의 클러스터 3이 남게 된다. 클러스터 3은 품질 $|C_3| \cdot (1/\beta)^{|D_3|}$에서 그것에 속한 점들의 개수와 부분차원에 속한 원소들의 개수가 크기 때문에 마지막 까지 삭제되지 않는다. 모든 실험에서 β = 0.5로 두었다.

4.2 초기 분할 개수 k_0의 분석

표 6은 데이타베이스 크기와 초기 분할 개수 k_0를 변화시켜서 얻어진 결과를 보여주고 있다. 이 표에서 우리는 적절한 k_0의 값을 설정할 수 있다. k_0의 값이 너무 작으면, 클러스터링의 성능이 나빠지고, k_0의 값이 커지면 좋은 클러스터링 결과를 얻을 수 있지만 실행 시간이 커지는 단점이 있다. 이 표로부터 우리는 k_0의 값이 18과 24 사이의 값이면 적절할 것으로 판단된다. 이 값은 최적의 클러스터링에서 얻어진 클러스터들의 개수의

3 배와 4 배 사이의 값이다.

초기 분할 개수 k_0가 최적의 클러스터 개수에 비하여 너무 작을 때, n개의 데이타를 k_0개의 클러스터들로 분할하는 *SplitData()* 함수의 결과가 전체 클러스터링에 나쁜 영향을 미치고 있다는 것을 실험을 통해 분석하였다. k_0가 크면 별 문제가 없지만, k_0가 작은 경우에 더 효과적인 클러스터링을 위해서는 현재 보다 더 정밀한 *SplitData()* 함수의 연구가 필요하다. 이것과 별도로, 다음과 같은 다른 접근 방법을 상정해볼 수 있다. 임의의 k_{01}로 시작하여 알고리즘 SAM을 수행하여 첫 번째 최선의(best) 클러스터들을 찾아서 투영 에너지, 품질, 이상치의 변화율을 기록한다. 그리고, 초기 분할 개수 k_{02}를 k_{01}의 2 배로 설정하여 다시 SAM을 수행하여 두 번째 최선의 클러스터들을 찾아서 목적 함수의 값들을 구한다. 이런 과정을 몇 번 반복하여 목적 함수의 값들이 더 이상 변하지 않는 최선의 클러스터들을 찾아낼 수 있다. 그러나, 이 방법은 많은 실행 시간이 요구되는 단점도 있다.

4.3 점 배정 함수에서 B의 민감도 분석

점 배정 함수에 사용되는 사용자 지정 상수 B의 값을 결정하기 위하여 실험을 수행하였다. 데이타베이스는 10,000개의 점들이고 초기 분할 개수 k_0 = 24인 경우에, 상수 B의 여러 값에 대하여 실험을 수행하여, 그림 7에 그 결과를 보여주고 있다. 중심 극한 정리에 의하여 대용량의 데이타가 정규 분포를 따른다고 가정하면[10], 한 클러스터의 부분차원에 속한 한 차원의 중심점인 평균치를 μ라 하고, 그 차원의 표준 편차를 σ라고 하자. 그러면, 그 클러스터에 속한 점들이 평균치에서 표준 편차에 B를 곱한 값의 범위까지인 $[\mu - B\sigma,\ \mu + B\sigma]$ 내에 얼마나 많이 포함하는가를 가늠하기 위한 상수가 B이다. B가 2.0에서 4.0까지 변할 때 우세 비율은 점차적으로 증가하였고, B = 4.0인 경우에 우세 비율은 99.93%였다. 그러나, B의 값이 4.0을 넘어서면 우세 비율이 미세하게 감소하였다. 그 원인은 B가 너무 큰 값으로 지정되면, 한 클러스터에 포함되는 점들의 분포가 넓게 되어 다른 클러스터들에 속해야 될 점을 포함하거나 또는 이상치를 포함시켜 정확도를 낮출 수도 있기 때문이다. 정규 분포에서 점들이 평균치에서 $\pm 2\sigma$ 범위 이내에 포함될 확률은 0.9544이고, $\pm 3\sigma$ 이내에 포함될 확률은 0.9974이고, $\pm 4\sigma$ 이내에 들어올 확률은 소수점 4 자리까지 고려하면, 거의 100%이다[12]. 실험 결과와 정규 분포상의 확률에 근거하여, 알고리즘 SAM의 점 배정함수에서 가중 표준 편차에 곱하는 배수 B의 값을 4.0으로 정하였다.

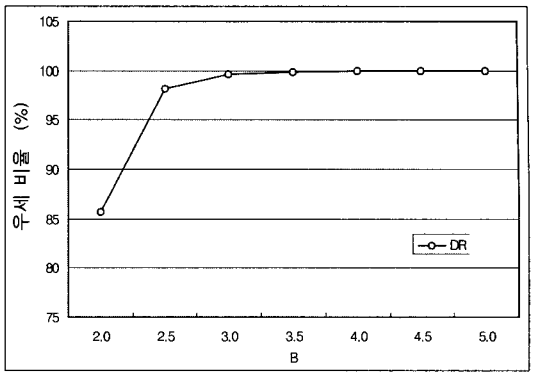


그림 7 B의 값에 따른 우세 비율

4.4 FASTDOC의 넓이 w에 따른 성능 평가

제안된 알고리즘 SAM과 비교 대상 알고리즘들 중의 하나인 알고리즘 FASTDOC[7]은 경계 상자(bounding box)의 크기인 넓이 w의 선택이 결과에 중대한 영향을 미치고 있다. 넓이의 지정 문제는 PROCLUS와 SUBCLUS의 부분차원에 속한 차원들의 평균 개수 지정과 연관된 문제이다. DOC에서 제시하고 있는 w는 점 p와 가장 가까이 위치한 점과의 평균 거리에 상수 C를 곱하여 아래의 식과 같이 구하였다.

$$w = C\sum_{i=1}^{n} w^i/n, \text{ 여기서 } w^i = \sum_{j=1}^{d} |p_{i,j} - q_{i,j}|/d.$$

w^i는 주어진 점 p_i와 가장 가까이 위치한 점을 q_i라고 할 때 두 점간의 거리를 나타내고, d는 전체 차원수를 나타낸다. 이 거리를 합산한 평균치에다 상수 C를 곱하면, 클러스터를 결정할 경계 박스의 넓이 w가 구해진다. n개의 점들로 구성된 데이타에서, 넓이 w를 구하는 시간 복잡도는 $O(n^2d)$이다. 그림 8에서와 같이 경계 상자의 넓이 w의 값의 변화에 따라, 클러스터링의 정확도인 우세 비율이 다르게 나타남을 알 수 있다. 실험에 사용된 데이타에서는 w가 대략 0.05 내외일 때

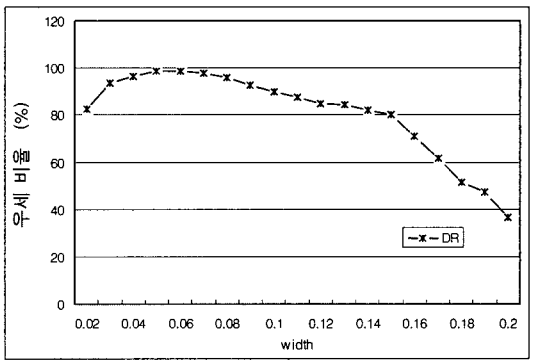


그림 8 FASTDOC의 w에 따른 우세 비율

좋은 결과를 보여주고 있다. 부분차원의 결정에 사용되는 식 w에서 상수 C는 몇 번의 시행을 거쳐 결정해야 하고, w'는 부분차원이 아니라 전체 차원 상에서 계산하므로 투영 클러스터링을 위한 식으로 적절하지 않다고 여겨진다.

4.5 성능 비교

제안된 알고리즘 SAM과 기존의 다른 두 알고리즘들과 성능을 비교하기 위하여 실험을 수행하였다. 투영 클러스터링 알고리즘들 중에서 대표적인 알고리즘인 PROCLUS와 FASTDOC를 비교 알고리즘들로 선택하여 C++언어로 구현하여 실험하였다. 입력 데이타 점들의 개수가 20,000 개에서 100,000 개 까지 그 크기를 변화하면서, 세 알고리즘들을 20번 실행시켜서 평균값으로 실행시간과 결과 클러스터링의 우세 비율을 측정하였다. 먼저, 그림 9에서 알고리즘 SAM의 실행 시간을 살펴보면, 실행 시간이 데이타베이스 크기에 대략 선형적으로 증가함을 보여준다[5,6]. 그림 9와 표 7의 결과는 k_0가 18인 경우에 측정된 클러스터링의 결과이다. 그림 9에서 PROCLUS가 제일 큰 실행 시간을 가지고 FASTDOC이 빠르고 SAM은 PROCLUS에 비해 많이 개선되었음을 보여준다. FASTDOC의 실행 시간은 데이타베이스의 크기에 상관없이 거의 일정하고 데이타베이스가 커지면 미세하게 증가한다. FASTDOC에서 클러스터를 발견할 때, 대부분의 시간이 부분차원을 결정하는 내부 반복 회수 $|D|^2$에 의해 결정되고, 이런 방법은 샘플링을 적용하는 것으로 볼 수 있다. 이 실험에서 경계 상자의 넓이 w는 4.4절의 식에 의해서 구하지 않았고, 그림 8에 근거하여 최적의 값인 w = 0.05로 미리 설정하여 실험 결과를 얻었다.

클러스터링의 정확도는 표 7의 우세 비율로 측정하였다.

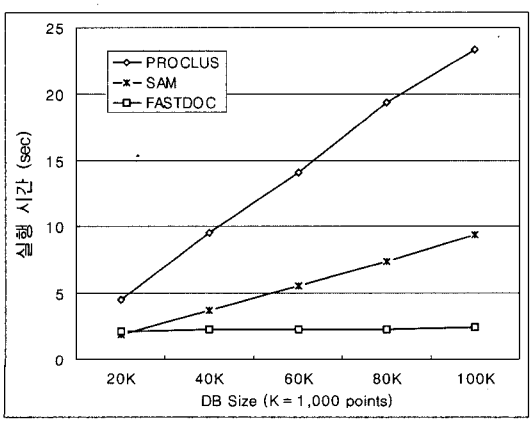


그림 9 데이타베이스 크기에 따른 클러스터링 알고리즘들의 실행 시간 비교

표 7 데이타베이스 크기에 따른 클러스터링 결과의 우세 비율(%)

알고리즘 \ DB 크기	20K	40K	60K	80K	100K
PROCLUS	82.28	87.11	85.76	81.72	87.23
SAM	100	99.99	99.99	99.99	99.99
FASTDOC	98.27	97.65	98.28	97.20	97.61

다. SAM과 FASTDOC의 최대 우세 비율은 100%이지만, PROCLUS는 이상치를 분리하지 않았기 때문에 95%가 최대 우세 비율이 된다. 제안된 알고리즘 SAM이 실험 데이타에서 가장 우수하게 최적의 클러스터들을 발견한다. 그 다음이 알고리즘 FASTDOC이다. 알고리즘 DOC를 적용하면, 더 좋은 정확도를 얻을 수 있지만, 부분차원을 계산하는 내부 반복 횟수가 너무 많아 실행 시간이 아주 길게 된다[7].

5. 결 론

대용량의 고 차원 데이타를 분석하기 위한 투영 클러스터링을 세 가지 척도인 투영 에너지, 품질, 이상치들의 집합을 변수들로 설정하여 투영 클러스터링에 대한 최적 문제로 정의하고, 이를 해결하기 위한 실용적인 해결책인 휴리스틱 알고리즘을 제안하였다. 이 알고리즘은 클러스터들의 개수와 클러스터의 부분차원에 속한 차원들의 평균 개수의 지정이 없어도 자체적으로 최선의 클러스터들을 찾아낸다. 분할과 합병 방식을 사용하는 알고리즘은 계층 클러스터링으로 분류된다. 클러스터들의 합병 과정에서 투영 에너지와 품질의 변화율을 계산하였고 이상치 집합을 분리해내어 이상치들의 개수 변화율도 측정하였다. 투영 에너지와 품질 계산에서 중요한 요소가 각 클러스터의 부분차원인데, 이것을 찾아내기 위하여 표준 편차의 상대적인 비율을 계산하여 부분차원으로 정하였다. 부분차원으로 인한 오차의 가능성을 줄이기 위하여, 각 표준 편차에 가중치를 부여하여 클러스터링에 사용하였다. 제안된 알고리즘은 세 가지 서로 다른 척도들인 투영 클러스터들의 투영 에너지, 품질, 그리고 이상치의 변화율에 근거하여 거의 최적인 투영 클러스터들을 결정하였다.

제안된 알고리즘 SAM과 투영 클러스터링의 대표적인 알고리즘인 PROCLUS와 FASTDOC을 실제로 구현하여 성능을 측정하였다. 주어진 실험 데이타에서 세 알고리즘들의 성능을 검토해보면, SAM은 최적의 클러스터들을 발견해내고 FASTDOC은 결과에서 클러스터링의 정확도가 조금 떨어지지만 빠른 실행 시간에 클러스터들을 찾았다. FASTDOC은 부분차원을 결정하게 하는 매개변수인 경계 상자의 넓이를 계산해내는 식에

사용자 지정 상수가 있어서 항상 최선의 클러스터들을 찾는다는 보장이 없다. FASTDOC의 빠름과 SAM의 정확도를 결합할 수 있다면, 빠르게 최선의 클러스터링 결과를 얻을 수 있을 것이다. 추가적인 성능 평가에서 실제 데이타를 기반으로 한 실험 결과가 요구된다. 고차원 데이타에서 최적의 투영 클러스터링을 찾는 알고리즘의 연구에서 여러 가지 척도들의 분석이 필요하고, 이런 척도들을 사용하여 되추적 기법이나 분기한정 기법을 적용하는 최적의 솔루션에 관한 연구가 필요하다.

참 고 문 헌

[1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques,* Morgan Kaufmann Publishers, San Francisco, CA, 2001.

[2] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, 31(3):264-323, 1999.

[3] L. Parsons, E. Haque, and H. Liu, "Subspace Clustering for High Dimensional Data: A Review," *ACM SIGKDD Explorations*, Vol. 6, Issue 1, pp. 90-105, June 2004.

[4] C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park, "Fast Algorithms for Projected Clustering," In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 61-72, Philadelphia, PA, June 1-3, 1999.

[5] 박종수, 김도형, "고 차원 데이터를 부분차원 클러스터링하는 효과적인 알고리즘", 정보처리학회 논문지 D, 10-D권, 3호, pp.417-426, June 2003.

[6] C. C. Aggarwal and P. S. Yu, "Finding generalized projected clusters in high dimensional spaces," In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 70-81, 2000.

[7] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali, "A monte carlo algorithm for fast procetive clustering," In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2002.

[8] M. L. Yiu and N. Mamoulis, "Frequent-Pattern based Iterative Projected Clustering," In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)*, Melbourn, Florida, USA, November 2003.

[9] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 94-105, 1998.

[10] 심정욱, 손영숙, 백장선 역, *수리통계학, 제4판*, 자유아카데미, 1999년.

[11] T. Zhang, R. Ramakrishnan, and M. Linvy, "BIRCH: An Efficient Data Clustering Method for Large Databases," In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 103-114, 1996.

[12] W. H. Beyer, *CRC Standard Mathematical Tables,* 28th Edition, CRC Press, 1987.

박 종 수

1981년 부산대학교 전기기계공학과(공학사). 1983년 한국과학기술원 전기및전자공학과(공학석사). 1983년~1986년 국방부 군무설계기좌. 1990년 한국과학기술원 전기및전자공학과(공학박사). 1994년~1995년 IBM Watson 연구소 객원 연구원. 1990년~현재 성신여자대학교 컴퓨터정보학부 교수. 관심분야는 데이타 마이닝, 데이타베이스 시스템, 데이타 웨어하우징