박 종 수 교수지도

석사학위 청구논문

부분 차원 클러스터링을 위한 개선된 알고리즘

2004

성신여자대학교 대학원

전산학과

김 연 호

부분 차원 클러스터링을 위한 개선된 알고리즘

박 종 수 교수지도

이 논문을 석사학위 논문으로 제출함.

2004년 5월

성신여자대학교 대학원

전산학과

김 연 호

인 준 서

김 연 호의 석사학위 논문으로 인준함

심사위원 (인)

심사위원 (인)

심사위원 (인)

성신여자대학교 대학원

논문개요

고차원 데이터에서 클러스터를 찾아내는 문제는 그 중요성으로 인해 데이터 마이닝 분야에 잘 알려져 있다. 클러스터링은 similarity search, customer segmentation, pattern recognition, trend analysis, classification 등 데이터를 처리하는 많은 분야에서 기본 알고리즘으로 사용되어질 수 있고 대량의 데이터를 처리하는 경영정보시스템에서도 그 활용가치가 높아 좋은 클러스터링 알고리즘의 연구에 대한 관심이 증대되고 있다. 본 논문에서는 고차원 데이터를 부분 차원 클러스터링 하는 새로운 알고리즘을 제안한다. 알고리즘은 먼저 최대 표준편차를 갖는 차원을 기준으로 점들을 분할하여 후보 클러스터들을 생성하고, 다음으로 후보 클러스터들 중 친밀도가 가장 큰 두 개의 클러스터를 찾아내어 미리 정해진 개수의 클러스터가 될 때까지 병합한다. 마지막으로 클러스터의 질을 향상시키기 위해 데이터를 한 번 더 정제 시킨다. 알고리즘의 성능 평가를 위해 실험 데이터를 만든 후 k-Means, PROCLUS, DOC과 같은 여러 알고리즘과 실험한 결과 제안된 알고리즘이 성능이 향상됨을 알 수 있었다.

목 차

그림 목차

표 목차

Ⅰ. 서론

최근 들어 인터넷이 급속도로 발전하고 웹(Web)을 이용하는 사용자들이 크게 늘면서 데이터들이 방대해지고 다양해지고 있으며 또한 이러한 사용자들의 정보를 이용하여 마케팅에 활용하려는 CRM분야에서 데이터 마이닝이 크게 이용되고 있다. 또한 점점 더 방대해지는 자료의 홍수 속에서 유용한 정보와 지식으로 어떻게 찾아내고 이용할 것인가가 주된 이슈가 된 지금 데이터 마이닝을 이용한 분야는 많은 기업에서 이용하고 있으며 앞으로도 더욱더 활용하게 될 것이다. 이러한 데이터 마이닝 분야에서 새롭게 연구방향을 잡아가는 한 분야가 클러스터링(Clustering)이다.

클러스터링은 유사 탐색(similarity search), 고객 세분화 (customer segmentation), 패턴 인식 (pattern recognition), 경향 분석 (trend analysis), 분류(classification) 등 데이터를 처리하는 많은 분야에서 기본 알고리즘으로 사용되어질 수 있고 대량의 데이터를 처리하는 경영정보시스템에서도 그 활용가치가 높아 좋은 클러스터링 알고리즘의 연구에 대한 관심이 증대되고 있다[8, 15, 16]. 특히, 인터넷과 정보통신 기술의 발달로 인해 쏟아지고 있는 고차원 데이터를 클러스터링 하는 알고리즘에 관한 관심이 높아지고 있다.

본 논문에서는 고차원 데이터를 저차원인 부분 차원으로 줄여서 데이터를 클러스터링하는 방법을 연구하였다. 데이터를 각 클러스터에 배정할 때, 전체 d차원으로 거리를 계산하는 것이 아니라 각 클러스터에 따라 다르게 선택된 부분 차원들만 고려된 거리를 계산하여 가까운 클러스터에 데이터를 배정하게 된다. 제안된 알고리즘은 클러스터링의 방법론인 분할 방식과 계층 방식을 차례로 적용한 것으로. 먼저 점(point)들을 여러 개의 클러스터로

나눈 후에, 이들을 원하는 개수의 클러스터로 병합하도록 하였다. 본 논문에서는 대부분의 대량의 데이터는 중심점(centroid)을 기준으로 $\pm 3*\sigma$ (σ : standard deviation)의 범위 내에 포함된다는 정규분포(Gaussian Distribution)의 성질을 이용하여 bound 개념을 사용하였다. 또한 합병단계에서 두 클러스터의 similarity를 계산할 때 DOC 알고리즘에서의 $\mu(|C|, |D|)$를 개선하여 새로운 similarity함수를 사용하였다.

본 논문의 구성은 다음과 같다. 제 Ⅱ장에서는 관련 연구에 대하여 살펴보고, 제 Ⅲ장에서는 새로운 알고리즘을 제안한다. 제 Ⅳ장에서는 실험결과에 대해 알아보고 마지막으로 제 Ⅴ장은 결론 및 향후 과제로 끝을 맺는다.

Ⅱ. 관련 연구

1. 클러스터링(Clustering)

1) 클러스터링의 정의

성격이 비슷한 데이터들을 같은 그룹으로 분류하는 과정을 클러스터링(clustering)이라고 하며 이는 주어진 n개의 d차원 데이터 포인트들을 유사한 성격을 갖는 k개의 클러스터로 나누는 것을 말한다. 클러스터링은 방대한 데이터 집합으로부터 적절한 거리 측정 방법을 이용, 데이터의 밀도가 높은 지역(클러스터)을 찾아낸다. 데이터가 밀집된 지역과 희소한 지역을 찾아냄으로써 데이터 집합의 전반적인 분포 패턴을 찾아낼 수 있다[8, 12].

예를 들어 고객 트랜잭션으로 이루어진 시장바구니(market basket) 데이터를 보면, 각 트랜잭션들은 고객이 구매한 항목들을 포함하고 있는데 클러스터링은 구매 항목들을 이용해 구매 패턴이 유사한 고객들을 같은 그룹(클러스터)으로 나누게 된다. 결과 클러스터의 예로 기저귀나 장난감 등의 유아, 아동 용품을 사는 결혼한 고객들의 그룹, 프랑스산 와인, 스위스 치즈, 벨기에산 초콜릿 등의 고가의 수입품을 구매하는 고수입 고객들로 이루어진 그룹 등을 찾을 수 있다[12].

클러스터링이 이용될 수 있는 분야는 상품이나 서비스 거래의 유사성 혹은 관련성에 대한 정보를 토대로 잠재적인 고객이 흥미를 느낄 상품이나 서비스가 무엇인지 찾아내는 유사도 검색(similarity search), 고객을 세그먼트화하여 각 고객들의 구매 패턴을 이해하여 적절한 마케팅 전략을 세울

수 있도록 하는 고객 세그멘테이션(customer segmentation), 고객의 속성 데이터, 재무 데이터, 거래내역 데이터를 토대로 채산성 있는 고객의 패턴을 도출해 내는 패턴인식(pattern recognition) 등 매우 다양한 도메인에 적용될 수 있다[2, 12].

2) 클러스터링 방법의 분류

클러스터링 알고리즘은 크게 Partitional, Hierarchical, Density-based, Grid-based 클러스터링 알고리즘의 4가지로 분류된다. 특정한 응용에 적당한 클러스터링 알고리즘을 선택하기 위해서는 아래와 같은 여러 요인들이 고려되어야 한다[6].

① 응용목적

응용목적은 종종 사용되는 클러스터링 알고리즘의 유형에 영향을 미칠 수 있다. 예를 들어, 슈퍼마켓 체인을 세우기 위한 좋은 장소를 발견하고자 할 경우 클러스터 중심으로부터의 거리의 합이 최소화되는 고객을 클러스터하고자 할 것이다. 클러스터 중심으로부터의 거리가 짧은 것이 바람직한 이러한 응용에 대해서는 K-Means나 K-Medoids와 같은 partitional 알고리즘이 주로 사용된다. 반면 점방식(raster) 데이터분석과 이미지 인식과 같은 응용에서는 density-based 알고리즘이 적합하다.

② 클러스터링의 질과 속도 사이의 균형

일반적으로 응용에 적합한 클러스터링 알고리즘은 질과 속도 모두를 만족해야 한다. 종종 클러스터되어지는 데이터의 크기는 클러스터링 알고리즘

의 실행시간에 중요한 요인으로 작용한다. 양질의 클러스터를 생산하는 클러스터링 알고리즘은 소량의 데이터베이스를 가진 응용에 대해서만 단지 적합하고 대량의 데이터를 다루는데 있어서는 그렇지 않을 수 있다. 대량의 데이터베이스를 다루기 위한 일반적인 방법은 초기 데이터베이스를 압축된 형태로 사용하는 것이다. 그러나 이러한 방법은 데이터의 손실을 가져오는 경우가 많기 때문에 어떻게 하면 클러스터의 질을 최소화하면서 클러스터링의 속도를 증가시키는 방향으로 데이터를 압축시키느냐 하는 것이 가장 중요한 관건이라 할 수 있다.

③ 데이터의 특징

클러스터되어지는 데이터의 특징은 적용할 클러스터링 알고리즘을 결정하는데 있어 또 다른 중요한 요인으로 작용한다. 이러한 특징은 numeric, binary, categorical 데이터와 같은 데이터 속성의 유형, 데이터의 차원의 수, 데이터에 있는 outlier의 양과 같은 것들을 포함한다.

(1) Partitional 클러스터링 알고리즘

Partitional 클러스터링 알고리즘은 고전적인 클러스터링 알고리즘이다. *d*개의 차원을 가진 *n*개의 점에 집합 N이 주어졌을 때 partitional 알고리즘은 N을 하나의 클러스터 분포 또는 그 클러스터의 중심으로부터 각 점의 전체 편차를 최소화하는 k개의 클러스터로 구성한다. 점의 편차는 알고리즘마다 다르게 계산될 수 있는데 보통 유사함수 또는 거리함수를 이용하며 하나의 클러스터 안에 있는 점은 유사하다고 말한다.

가장 대표적인 partitional 클러스터링 알고리즘에는 클러스터의 중심(center)으로써 클러스터내에 있는 점의 중간값(평균)을 사용하는

K-Means, 클러스터내에서 가장 중간에 위치하는 점을 medoid라 하고 이것을 중심으로 사용하는 K-Medoid가 있다[5, 6].

(2) Hierarchical 클러스터링 알고리즘

Hierarchical 클러스터링 알고리즘은 입력으로 들어오는 n개의 데이터 포인트 각각을 개별적인 n개의 클러스터로 보고 k개의 클러스터가 될 때까지 계속 합병해 나가는 방법이다. 클러스터를 합병할 때는 거리가 가장 가까운 두 클러스터를 합병하게 된다. 여기에는 각각의 점을 하나의 클러스터로 보고 하나의 클러스터 또는 종료조건을 만족할 때까지 합병해 나가는 agglomerative hierarchical 클러스터링과 모든 점을 가진 하나의 클러스터로 시작하여 종료조건을 만족할 때까지 더 작은 부분으로 나누는 divisive hierarchical 클러스터링이 있다.

가장 대표적인 hierarchical 클러스터링 알고리즘에는 데이터 점을 수많은 작은 부분클러스터로 압축하여 이것을 가지고 클러스터링하는 BIRCH(Balanced Iterative Reducing and Clustering using Hierarchies)[14], 각 클러스터마다 대표값으로 하나가 아닌 여러 개의 잘 분포된 점이 선택된 후 대표값을 0~1사이의 범위를 가지는 shrinking factor에 의해 클러스터의 중심으로 줄인 후 클러스터링하는 CURE(Clustering Using REpresentatives)가 있다[5, 6, 7, 14].

(3) Density-based 클러스터링 알고리즘

대부분의 partitional 클러스터링 알고리즘은 점사이의 거리를 기준으로 한다. 그러한 방법은 단지 구형의 클러스터를 찾을 수 있을 뿐이고 임의의

모양을 가진 클러스터를 발견하는 데 있어서는 많은 어려움이 있다. 그래서 밀도(density)의 개념에 기반을 둔 새로운 클러스터링 방법이 개발되고 있다. 이것은 일반적으로 outlier라고 표현되는 낮은 밀도를 가진 지역에 의해 구분되는 데이터 공간에서 밀집된 지역의 데이터를 클러스터로 간주한다. density-based 방법은 outlier를 제거하는데 사용될 수 있고 임의의 모양을 가진 클러스터를 발견할 수 있다.

density-based 클러스터링 알고리즘에는 DBSCAN(Density-Based Spatial Clustering of Applications with Noise), OPTICS(Ordering Points To Identify the Clustering Structure), DENCLUE (DENsity-based CLUstEring)가 있다[22, 23, 24].

(4) Grid-based 클러스터링 알고리즘

DBSCAN, OPTICS와 같은 density-based 클러스터링 알고리즘은 고차원일 경우 효율성이 떨어지는 색인기반(index-based) 방법이다. 이러한 클러스터링의 효율성을 향상시키기 위해 grid-based 클러스터링 알고리즘은 격자(grid) 데이터 구조를 사용한다. 클러스터링이 수행됨에 따라 모든 작업은 격자구조 위에서 형성되는 유한개의 cell로 공간을 나눈다. 이러한 접근의 주요 장점은 데이터의 수에 독립적인 빠른 처리시간이다. 그러나 나누어지는 공간에 있는 각 차원이 단지 cell의 수에 의존한다.

grid-based 알고리즘에는 격자 cell안에 저장되는 통계학적 정보를 조사하는 STING(STatistical INformation Grid), wavelet 변형방법을 이용하는 WaveCluster, 고차원 공간에서의 클러스터링을 위해 격자와 밀도를 기반으로 한 방법을 표현하는 CLIQUE(CLustering In QUEst)가 있다[2, 5, 6].

2. 고차원 데이터에 있어서의 클러스터링의 문제점

고차원 공간에서의 데이터에 대한 클러스터링에 있어 문제점은 데이터 자체의 희박성(sparsity)으로 인한 클러스터링 성능의 저하이다. 예를 들어 고차원 응용에서 주어진 한 쌍의 두 점이 있다고 하면 두 점이 대부분의 차원에서는 매우 가깝다 할지라도 거리가 먼 차원도 있을 수 있다. 그림 2.1과 같이 3차원 공간의 포인트 집합을 그림 2.2, 2.3과 같이 x-y, x-z 평면에 따로 투사시켜 보면 각 평면에 따른 패턴들을 볼 수가 있다. 하지만 3차원 모두를 고려해서 이러한 패턴들을 찾기는 매우 힘들다. 왜냐하면 패턴을 가지고 있는 2차원 외의 나머지 한 차원의 데이터가 전체 공간에 분포되어 있다고 할 때 3차원 모두를 고려했을 경우 포인트 간의 거리가 그 나머지 한 차원에 의해 영향을 받기 때문이다.

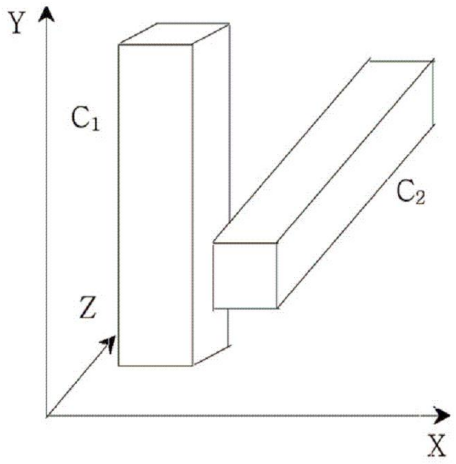
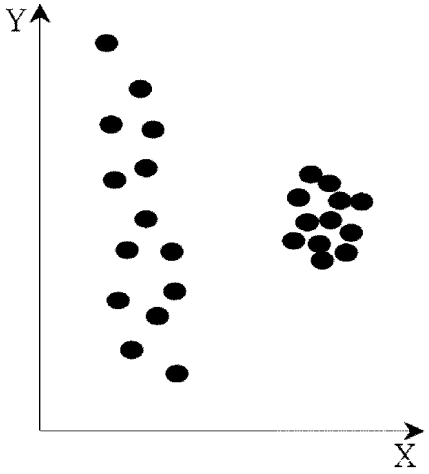


그림 2.1 3차원 공간에서 패턴



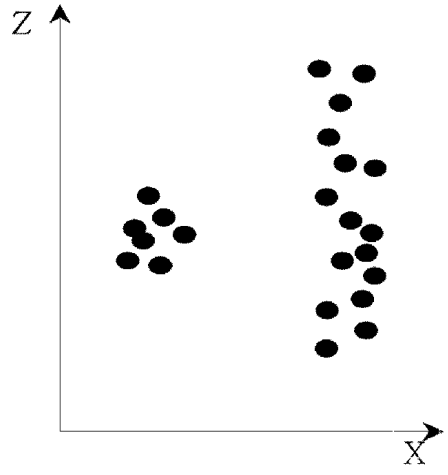


그림 2.2 X-Y 평면으로의 투영　그림 2.3 X-Z 평면으로의 투영

이러한 고차원 데이터 내에서 패턴을 찾기 위해 시도되는 방법에는 서로 관련이 적은 차원들을 클러스터링 시 미리 제거함으로써 잡음을 줄이는 특징 추출 방법과 사용자가 특정 차원을 골라내는 방법이 있다. 전자의 방법을 이용할 경우 클러스터링 전에 미리 특정 차원들을 선택하기 때문에 정보의 손실을 가져올 수 있을 뿐만 아니라 많은 실제 데이터의 경우에서 보면 어떤 포인트들은 선택된 차원들과 상관관계가 있지만 어떤 포인트들은 선택되지 않은 다른 차원들과 더 많은 상관관계를 가진 경우가 발생하게 됨으로써 정보 손실의 문제가 생기게 된다. 사용자가 특정 차원을 선택하는 방법 역시 오류를 발생시킬 확률이 높다.

효과적인 고차원 데이터에 대한 마이닝을 하기 위해서는 의미 있는 차원들을 정보의 손실을 최소화시키면서 잘 추출해 낼 수 있어야 하고, 결과에 대한 해석 능력뿐만 아니라 사용자가 이해할 수 있도록 표현할 수 있어야 하며 차원 수와 데이터의 사이즈가 커지더라도 빠르게 처리할 수 있어야만 한다[1].

1) K-Means

모든 차원을 고려한 클러스터링 알고리즘으로 대표적인 것이 K-Means이다. K-Means는 클러스터의 중심(center)으로써 클러스터내에 있는 점의 중간값(평균)을 사용하는 partitional 클러스터링 알고리즘이다. K-Means 알고리즘은 다음과 같다. 첫째, k-클러스터의 초기 대표값으로 사용될 k-point(mean)를 임의로 선택한다. 둘째, 각 mean과 가장 가까운 클러스터에 점을 할당하고 새로운 클러스터의 평균값으로 각 mean을 변경하는 과정을 변화가 없을 때까지 반복하여 k-클러스터를 구한다. 본 논문에서는 제안된 알고리즘과 비교하기 위해 K-Means를 사용한다.

3. 고차원 데이터에 적합한 클러스터링

앞 장에서 언급한 고차원 공간 내에서 데이터의 희박성으로 인하여 최적의 패턴을 찾을 수 없었다. 이러한 해결책으로 몇 개의 주요 차원에 의해 원래 데이터에 내재하는 전체 변이 중 가능한 많은 부분이 보유 되도록 변환시키므로서 정보의 손실을 최소화하는 차원의 축소(Dimensionality reduction)와 전체 차원 중 일부의 차원만을 고려하여 클러스터링 하는 subspace clustering에 관하여 알아본다.

1) 차원의 축소(Dimensionality reduction)

차원 축소의 일반적인 목적은 속성의 변환(attributes transformation)과 도메인의 분해(domain decomposition)이다.

첫 번째로 속성의 변환이라는 것은 실재하는 데이터에 대해 가장 간단한 방법이라고 할 수 있다. 예를 들어서 영업 분석표(sales profiles)나 OLAP-type 데이터에 대해서, 특정기간 동안의(예; 월별, 분기별) 평균이나 합과 같은 roll-up 연산을 하는 것이다. 다변량 통계학에서는 주성분 분석(PCA - Principal Components Analysis)[10,11]방법이 가장 많이 사용된다. 하지만, 이는 클러스터를 해석하기가 쉽지 않은 단점이 있다. 특이치 분해(SVC - Singular Value Decomposition) 방법은 정보 검색(information retrieval)[10]과 통계학[3, 25]에서 많이 사용되어지고 있다.

두 번째로 도메인의 분해는 데이터를 부분 집합(subsets)과 canopies[19]로 분할한 후, 간단한 유사도 수치(similarity measure)을 이용하여, 고차원의 계산을 좀 더 작은 데이터 집합에 적용하는 방법이다. 차원은 동일하나, 데이터의 집합이 적어 소요되는 비용은 줄어들게 된다. 이러한 접근방식은 고차원의 대용량 그리고 많은 클러스터 존재할 경우 적합한 방식이다.

2) 부분 공간 클러스터링(Subspace Clustering)

부분 공간 클러스터링은 본래의 속성 공간에서 전체 차원이 아닌 적절한 부분차원 내에 클러스터를 찾음으로써, 차원의 저주를 해결하고자 하는 방식이다. 이러한 접근방식은 몇 개의 원소에 의해서 데이터를 기술하는 경우, 매우 적절한 방식이다.

본 논문에서는 이와 관련된 투영(projected) 클러스터링 알고리즘인 PROCLUS, DOC에 대해서 알아본다.

(1)PROCLUS(PROjected CLUStering)

K-Means를 포함한 기존의 알고리즘들은 데이터의 모든 차원을 고려한 알고리즘들이다. 앞에서도 언급했듯이 데이터의 차원이 증가할수록 데이터의 희박성(sparsity) 문제 때문에 모든 차원을 고려하여 클러스터링을 하게 되면 성능이 현저히 떨어짐을 볼 수 있다. PROCLUS에서는 전체 차원이 아닌 일부 차원에서만 클러스터를 찾게 된다. Projected cluster란 전체 차원 d중 부분 차원에 존재하는 클러스터를 의미한다.

PROCLUS는 후보medoid 집합을 생성하기 위하여 greedy방법과 CLARANS의 지역탐색접근(local search approach)을 결합하고 연관된 클러스터에 대해 적당한 차원을 찾기 위하여 몇 가지 독창적인 아이디어를 사용한다[3, 4]. 후보 medoid로부터 k개의 medoid를 선택한 후, 각 medoid와 관련이 높은 차원을 찾아내 클러스터링을 한다. termination criterion을 만족할 때까지 이 과정을 반복, 클러스터링의 결과 값이 가장 낮은 k-medoid를 갖는 클러스터들을 결과로 한다.

알고리즘은 초기화 단계, 반복 단계, 정제 단계의 세 단계로 구성되어 있다. 초기화 단계는 데이터 집합의 크기를 줄이고 동시에 이 집합에 있는 각각의 클러스터로부터 대표 점들을 선택한다. 반복단계는 좋은 medoid 집합을 발견하기 위해 hill climbing 방법을 사용한다. 또한 medoid에 할당된 점이 관련차원에 의해 결정된 공간에서 좋은 클러스터를 형성하도록 하기 위해 각 medoid에 대응하는 차원의 집합을 구한다. 정제단계는 클러스터링의 질을 향상시키기 위해 주어진 과정에 의해 데이터를 한 번 더 정제한다.

(2) DOC (Density-based Optimal projective Clustering)

DOC 역시 PROCLUS와 같이 projected clustering 방식을 취하고 있다. 하지만 위에서 언급한 알고리즘들은 결과로 산출된 클러스터의 관련 차원의 수가 동일한 것과 달리 DOC은 클러스터마다 관련된 차원의 수가 다르다. 최적의 투영된 클러스터를 찾기 위해서 간단한 Monte Carlo 알고리즘[18]을 사용하고 있다. 전체 데이터 집합 중에서 점 p를 임의 추출하여, 관련된 차원 D가 정해지면, 그림 2.4와 같이 점 p를 중심으로 관련된 차원 내에서 간격이2w인 box $B_{p,D}$를 형성하여, 전체 데이터 집합에서 $B_{p,D}$에 속하는 점들을 클러스터로 간주한다. 또한, 한 클러스터 내에 포함될 점의 수를 결정하는 인수 α와 한 클러스터 내에서 관련된 차원의 수와 k점들의 수에 대한 중요도를 반영하는 β를 사용자가 정의 할 수 있다.

DOC 내에서 빈번한 전체 데이터 스캔으로 인하여 많은 시간을 필요로 하므로, 좀 더 빠른 실행 속도를 가지는 FASTDOC이라는 알고리즘을 제안하였다.[13]

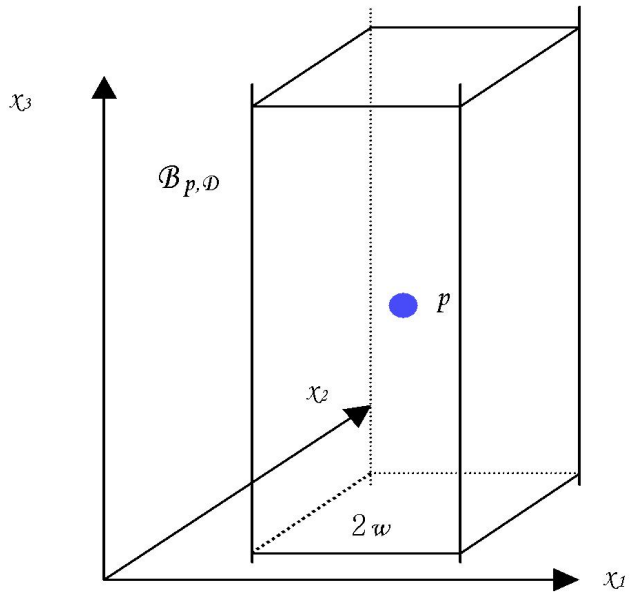


그림 2.4 점 p에 의해 형성된 $B_{D,D}$

Ⅲ. 클러스터링 알고리즘

제안된 알고리즘 DAM(Divided And Merge)은 클러스터링의 방법론인 분할방식과 계층방식을 차례로 적용한 것으로, 먼저 점(point)들을 실제 구하려는 클러스터의 개수보다 많은 클러스터로 나눈 후에 이들을 원하는 개수의 클러스터로 병합하였다. 입력으로는 n개의 d차원을 가진 점들이고, 사용자가 지정하는 변수는 원하는 클러스터의 개수 k와 한 클러스터 당 평균적으로 투영된 차원의 개수이다. 구체적인 알고리즘은 그림 3.1과 같다.

Algorithm DAM(k, l_{avg})
begin
  $k_0 = k_c = A \cdot k$;
  $(s,...,s_{kc},C_1,...,C_{kc}) = DivideData(k_0)$;
  while $(k_c > k)$ **do begin**
    $(D_1,...,D_{kc}) = FindSubdimensions(k_c, l_{avg}, C_1,...,C_{kc})$;
    $(s,...,s_{kc}, C_1,...,C_{kc}) = AssignPoints(s,...,s_{kc}, D_1,...,D_{kc})$;
    $(s,...,s_{kc-1}, C_1,...,C_{kc-1}) = MergeCluster(C_1,...,C_{kc}, k_c)$;
    k_c--;
  end
  $(D_1,...,D_k) = FindSubdimensions(k, l_{avg}, C_1,...,C_k)$;
  $(s,...,s_k, C_1,...,C_k) = AssignPoints(s,...,s_k, D_1,...,D_k)$;
  return $(C_1,...,C_k)$;
end

그림 3.1 DAM 알고리즘

크게 분할 단계, 병합 단계, 그리고 정제 단계, 세 단계로 구성 되어진

다. 첫 번째 분할 단계에서는 최대 표준편차를 갖는 차원을 기준으로 점들을 분할하여 두개의 후보 클러스터들을 생성하고, 다음으로 점이 가장 많은 클러스터를 대상으로 같은 방법으로 초기 분할 개수 $k_0 = A \cdot k$가 될 때까지 분할한다. A의 범위는 입력 데이터베이스의 크기에 따라 결정한다. 초기 단계인 분할 단계를 거치면, 각 클러스터는 자신에게 소속된 점들과 이 점들의 중심점을 가지게 된다.

두 번째 병합 단계에서는 각 클러스터에 대해서 부분 차원을 찾고 부분 차원을 고려하여, 모든 점들을 가장 가까운 중심점을 갖는 클러스터에 배정한다. 이때에 가장 가까운 중심점과의 거리가 $B*\sigma$ (B : bound, σ : standard deviation)이상이면 고려대상에서 제외된다. 다음으로 개수가 가장 적은 클러스터를 찾아 전체 클러스터 개수의 평균의 10분의 1보다 크거나 같으면 모든 클러스터에 대해 similarity가 가장 큰 클러스터를 찾아내어 병합한다. 그러나 그 이하이면 그 클러스터의 모든 점들을 가장 가까운 클러스터에 재배정한다. 가장 가까운 클러스터의 중심점과의 거리가 $B*\sigma$ 이내이면 그 클러스터에 배정하고 그렇지 않으면 outlier로 둔다.

세 번째 단계에서는 k개 클러스터들의 부분차원과 중심점을 재계산하고 모든 점들을 재배정하여 최종적으로 클러스터링의 결과가 더 좋아지도록 한다.

1. 분할 단계

첫 번째 단계인 분할 단계에서는 n개의 입력 점들을 k_0개의 클러스터로 나누는데 각 클러스터에 속한 점들의 개수가 가능하면 균일하도록 한다. 이 함수에서는 BIRCH[14]와 ORCLUS[9]에서 사용하는 clustering feature인 , 각 클러스터에 속한 점들의 선형 합(liner sum)과 제곱 합(squared sum)인 LS와 SS를 사용한다. LS와 SS를 사용하여 각 클러스터의 점들이 각 차원에서 분포된 정도를 나타내는 표준 편차를 계산하여, 표준편차가 큰 차원을 기준으로 분할해 나간다. 표준편차가 크다는 것은 중심점으로부터 점들이 많이 흩어져 있어 클러스터의 밀집도가 낮다는 것을 의미하므로 표준편차가 제일 큰 차원을 분리하는 것이 클러스터링에 좋다. 클러스터 C_i의 선형 합은 하나의 점당 d개의 차원 즉, 원소로 구성되어 있고, 각 원소는 $LS_{i,j} = \sum p_{x,j}$ 로 계산된다. 여기서 $|C_i|$는 클러스터 C_i의 점의 개수이고 p_x는 C_i에 포함된 점이다. 그리고 제곱 합도 점의 위치의 제곱을 계산 한다는 것 이외는 비슷한 방식으로 계산된다. C_i에서 d개로 구성된 제곱 합의 j번째 원소는 $SS_{i,j} = \sum p^2_{x,j}$ 로 계산된다. 클러스터 C_i의 전체 d차원들 중에서 j번째 차원에 분포된 점들의 평균값(mean)은 $\mu_{i,j} = LS_{i,j}/|C_i|$로 계산되고, 그 표준 편차(standard deviation)는 $\sigma_{i,j} =$ 로 계산된다. 알고리즘은 그림 3.2과 같다.

클러스터를 분할할 때 기준이 되는 median point는 중심점과는 다르다. median은 자료를 크기 순으로 늘어놓았을 때 가운데 위치한 값을 말한다. 따라서 median을 기준으로 전체 자료의 50%는 median보다 작고 나머지 50%는 median보다 크게 되므로 균일하게 분할하여진다.

DivideData(k_0)

begin

cluster number = 1;

repeat

find the cluster A, which has the maximum number of points;

find the dimension B, which has the maximum standard deviation of the cluster A;

find the median point for dimension B in the cluster A;

if points in the cluster A is larger than the median point, the points will be splitted up into the cluster B or if not, the points will be splitted up into the cluster C;

cluster number++;

until(*cluster number* < k_c)

end

그림 3.2 DivideData 함수

이 분할 단계에서의 결과는 k_0개의 클러스터들이 균일하게 퍼져 있도록 하는 것이 효과적이다.

2. 병합 단계

이전 분할 단계에서 생성된 많은 클러스터들을 실제 구하고자 하는 k개의 클러스터들로 병합하는 단계이다. 이 단계에서는 크게 세 가지 함수 : FindSubdimensions(), AssignPoints(), MergeCluster()를 반복 사용하여 k개의 클러스터를 구한다. 각각의 함수에 대해 자세히 살펴보자.

1) 부분차원 찾기

부분차원 찾기 함수에서도 앞서 사용한 clustering feature인 LS와 SS를 사용하여 각 클러스터의 점들이 각 차원에서 분포된 정도를 나타내는 표준 편차를 계산하여, 이 값을 기준으로 각 차원의 부분차원을 결정한다. 그림 3.3은 이 함수를 설명하고 있다.

모든 클러스터들의 전체 d차원에서 점의 분포를 계산한 표준 편차를 오름차순으로 정렬한 후에, 작은 값을 갖는 차원을 선택하여 해당 클러스터에 부분 차원으로 지정하게 된다. 전체적으로는 $k \cdot l$개의 차원을 선택하고, 선행 조건으로 먼저 각 클러스터에서 제일 작은 표준 편차를 갖는 두 개의 차원을 선택하도록 한다. 그 후에 $k \cdot (l - 2)$개의 표준 편차 값을 크기 순으로 선택하여 그것을 포함하는 클러스터의 부분 차원으로 정한다. 그림2.1에서 2개의 차원이 선택된다면, C_1에 속하는 3차원 점들의 부분차원은 {X, Z}가 되고, C_2에 의해 선택되는 부분차원은 {X, Y}가 된다.

FindSubdimensions(*k*, *l*, C_1, ..., C_k)

begin

for each cluster *i* **do begin**

// linear sum, squared sum and standard deviation of C_i

$LS_{i,j}$ = $\sum p_{x,j}$ and $SS_{i,j}$ = $\sum p^2_{x,j}$ for $p_x \in C_i$ and $1 \le j \le d$;

$\mu_{i,j} = LS_{i,j}/|C_i|$ and $\sigma_{i,j}$ = $\sqrt{\frac{SS_{i,j} - LS_{i,j}^2/|C_i|}{|C_i|-1}}$ for $1 \le j \le d$;

$D_i = \varnothing$;

end

Pick the $k \cdot l$ numbers with the least values of $\sigma_{i,j}$ subject to the constraint that there are at least 2 dimensions for each cluster;

if $\sigma_{i,j}$ is picked **then** add dimension *j* to D_i;

return (D_1,...,D_k);

end

그림 3.3 FindSubdimensions 함수

2) 점 배정

AssignPoints 함수는 각 클러스터의 중심점 s_i과 부분차원 D_i를 고려하여, 각 점을 가장 가까운 중심점을 갖는 클러스터로 배정하게 된다. 한 점 p_x와 한 클러스터의 중심점 사이의 거리를 계산하는 공식 $WSegPdis(p_x, s_i, D_i)$로 표기하고, Weighted segmental distance인 $WSegPdis(p_x, s_i, D_i)$ = $\frac{\sum_{j \in D_i} (\text{weight} \cdot |p_{x,j} - s_{i,j}|)}{\sum_{j \in D_i} \text{weight}}$ 로 계산한다. Weight는 앞서 계산되어진 부분차원의 표준편차가 작을수록 크게 주어, 관련된 부분차원의 중심점과의 거리가 가까울수록 더 높은 weight값을 갖도록 하였다. Bound개념을 사용한 NBS(normal bound sigma)를 벗어난 점들은 outlier로 둔다. NBS는 대부분의 대량의 데이터는 중심점을 기준으로 $\pm 3 * \sigma$ 의 범위 내에

포함된다는 정규분포(Gaussian Distribution)의 성질을 이용하여 NBS = $(B * W * \sigma) / W_t$ (B : *Bound*, W : *weight*, W_t : *total weight*)로 계산된다. PROCLUS는 medoid를 중심으로 가장 가까운 클러스터에 점을 배정하고, DAM은 중심점을 기준으로 가장 가까운 클러스터에 점을 배정하는 차이점이 있다. 모든 점들을 가장 가까운 클러스터에 배정한 후에, MergeCluster함수에서 필요한 선형 합, 제곱 합, 그리고 중심점을 다시 계산한다.

AssignPoints($s,\ldots,s_k,D_1,\ldots,D_k$)

begin
  for each cluster *i* **do begin**
    Determine NBS_i;
    $C_i = \varnothing$;
    end
  for each data point *i* **do begin**
    for each cluster *j* **do**
      Determine $WSegPdis_j$;
    if min($WSegPdis_j$) >= NBS_j **then** set outlier;
    else add p_i to C_j;
    end
  for each cluster *i* **do begin**
  // linear sum, squared sum and centroid
    $LS_{i,j} = \sum p_{x,j}$ and $SS_{i,j} = \sum p^2_{x,j}$ for $p_x \in C_i$ and $1 \le j \le d$;
    $s_i = LS_{i,j}/|C_i|$ for $1 \le j \le d$;
    Determine NBS_i;
    end
  return($s,\ldots,s_k, D_1,\ldots,D_k$);
end

그림 3.4 AssignPoints 함수

3) 병합

병합 함수에서는 중요한 두 가지 함수가 호출된다. 기준에 미치지 못하는 클러스터를 제거하는 ScatterPoints함수와 합병할 두 클러스터간의 similarity를 계산하는 similarity함수이다.

MergeCluster함수는 우선 개수가 가정 적은 클러스터들을 찾아 그 클러스터의 개수가 평균의 10분의 1보다 작으면 ScatterPoints함수에 의해 삭제한다. 다음으로 모든 클러스터에 대해 similarity를 계산하여 가장 큰 값을 갖는 두개의 클러스터를 합병한다. MergeCluster함수는 그림3.5와 같고 두개의 함수에 대해 자세히 설명하겠다.

```
MergeCluster(C1 ,..., Ckc, kc)
begin
    index = 0;
    value = max number;
    foreach i ∈ {1,...,k} do begin
        index = smallest number of cluster index;
        value = smallest number of cluster number;
        end
    if (value < average of cluster number/10.0) then
        ScatterPoints(index);
    else do begin
        max_similar = Max Similarity(Ci,Cj) for each i,j ∈ {1,...,k};
        if (max_similar > 0.0) then merge clusters;
        else ScatterPoints(index);
        end
end
```

그림 3.5 MergeCluster 함수

(1) ScatterPoints 함수

삭제되는 클러스터의 모든 점들을 가장 가까운 다른 클러스터에 배정하거나 outlier로 둔다. 자기 자신을 제외한 모든 클러스터에 대해서 Weighted segmental distance를 구하여 가장 가까운 클러스터를 찾아내고 그 클러스터의 중심점과의 거리가 NBS이내이면 그 클러스터로 배정한다. NBS보다 크거나 같은 점들은 outlier로 둔다.

(2) Similarity 함수

입력으로 두개의 클러스터에 인덱스가 주어진다. 먼저 두 클러스터의 부분차원의 교집합의 개수를 구하여 부분집합의 개수가 2이상일 때만 similarity를 구하고 아니면 0.0을 return한다. 교집합인 부분차원에 대해서 두 중심점간의 거리를 표준편차를 기준으로 $B*\sigma$ 내에 있으면 weight 값을 하나씩 올려준다. $B*\sigma$ 내에 있는 부분 차원이 많을수록 weight값은 커진다. similarity는 DOC 알고리즘에서의 $\mu(|C|, |D|)$를 개선하여 새로운 similarity 공식을 사용하였다.

$$similarity = ((|C_i|+|C_j|) * (\sigma_i + \sigma_j))/(f * \max(|D_i|, |D_j|))$$

여기서 f는 두 클러스터에서 공통인 부분 차원들의 중심점 사이간의 거리이다. 공통인 부분 차원들의 중심점 사이의 거리가 좁을수록, 클러스터의 크기가 클수록, 두 클러스터간의 similarity값은 커진다.

3. 정제 단계

이 단계는 부분차원 찾기 함수와 점 배정 함수, 두 가지 함수가 호출되어진다. 먼저 두 번째 단계에서 k개의 클러스터들을 병합한 결과를 넘겨받으면, 부분차원 찾기 함수에서 각 클러스터에 속한 점들의 분포를 나타내는 각 차원에서 표준 편차를 계산하여 각 클러스터의 부분차원을 구한다. 그리고 점 배정 함수에서는 각 클러스터의 중심점과 부분차원에 따라 점들을 가까운 클러스터에 배정하게 된다. 점 배정 함수의 호출에 의한 결과는 n개의 점들을 우리가 원하는 k개의 클러스터로 분할한 것이다. 결과적으로, 한 점은 한 클러스터에 속하게 되고, 한 클러스터 내의 점들 사이의 거기는 다른 클러스터에 있는 점과의 거리보다 근접해 있게 된다.

Ⅳ. 실험 결과 및 분석

1절에서는 실험 데이터의 생성과 결과의 평가를 위한 혼돈행렬에 대해 설명하고 2절에서는 새로운 알고리즘에 대한 실험 및 결과 분석을 한다.

1. 실험 데이터의 생성 및 혼돈 행렬

본 논문에서는 알고리즘의 성능 평가를 위한 고차원 데이터를 생성하기 위해 [1]에서 제시된 데이터 생성 알고리즘을 이용하였다.

우선 각 점 X는 $0.0 \leq X < 1.0$ 인 범위 내에 있는 실수형 좌표이다. outlier의 최대 비율은 전체 데이터의 5%이며, outlier의 생성은 전체 데이터 공간에서 임의의 균등 분포를 따른다. 클러스터에 속하는 점을 생성하기 위해서 클러스터의 개수인 k와 각 클러스터와 관련된 차원의 개수를 결정하기 위한 포아송 파라미터인 μ를 입력 파라미터로 이용한다. 데이터 점을 생성하기 위해서 각 클러스터가 분포하게 될 anchor point와 클러스터의 관련 차원을 정한다. 그리고 나서 각 클러스터의 점 개수를 정하게 된다. anchor point들은 d차원 공간에서 k개의 균등 분포를 따르는 점을 생성함으로써 얻어지게 된다.

각 클러스터와 관련된 차원의 개수는 평균값 μ를 갖는 포아송 확률 분포 함수에 의해 결정되는데 제약 조건은 이 개수가 최소 2에서 최대 d 사이의 값이어야 한다는 것이다.

일단 클러스터 i에 대한 차원의 개수 d_i가 생성이 됐으면 각 클러스터의

차원은 다음과 같은 방법으로 정해진다. 첫 번째 클러스터의 차원들은 임의의 차원으로 정한다. i번째 클러스터의 차원들은, $(i-1)$번째 클러스터로부터 $\{d_{i-1}, d_i/2\}$ 중 더 작은 수만큼은 $(i-1)$번째 클러스터의 차원과 동일하게 만들고 나머지는 전체 차원 중 임의의 차원을 선택한다. 이것은 서로 다른 클러스터들이 상호 관련된 차원을 갖도록 하기 위해서이다.

각 클러스터의 점 개수를 정하기 위해서 평균값 1을 갖는 k 차수인 확률변수를 생성한다.

마지막으로 클러스터 i의 점들은 다음과 같이 생성된다. 클러스터와 관련되지 않은 차원들의 점 좌표는 uniform distribution을 따르는 임의의 값이 선택된다. 관련 차원 j의 점 좌표는 anchor point의 j차원 좌표를 평균값 μ로 갖는 정규분포와 spread parameter r 값, 그리고 입력인 s를 이용 균일하게 임의로 scale factor $s_{ij} \in [1, s]$를 선택한다. 차원 j에 대한 Normal distribution의 분산은 $(s_{ij} \cdot r)^2$이다.

입력 클러스터	관련차원	점 갯수
1	3, 14, 17	1,867
2	3, 4, 6, 9,14, 16, 17, 21	2,032
3	5, 17	1,592
4	16, 17, 24	1,373
5	7, 14, 17 ,18, 24	1,428
6	16, 17, 20	1,208
outlier	-	500

표 4.1 생성된 실험데이터

표 4.1은 생성된 실험데이터를 나타낸다. 본 논문에서는 서로 다른 관련 차원으로 생성된 클러스터를 포함하는 입력파일을 사용한다. 입력파일은 25 차원 공간에서 N = 10,000개인 데이터 점을 가진 6개의 클러스터로 구성된다.

클러스터의 정확도를 평가하기 위하여 혼돈행렬(Confusion Matrix)을 사용하였다. [14, 25]. 표4.1의 입력 데이터에 대한 PROCLUS 알고리즘의 결과는 표4.2의 혼돈 행렬로 표시하였다. 혼돈 행렬의 (i, j) 항목은 입력 클러스터 j에 속한 점들 중에서 출력 클러스터 i에 속한 점들의 개수를 표시한다.

표4.2의(3, A) 항목은 입력 클러스터 A에 속한 점 1,867개 중에서 1,036개가 출력 클러스터 3으로 대응되어짐을 보여주고 있다. 만약 클러스터링 알고리즘의 성능이 뛰어나다면, 각 행과 열은 다른 셀에 비해 월등히 높은 수치를 가진 한 항목을 가질 가능성이 높아진다. 표4.2에서 입력 클러스터 B, C, D, E열은 특정 출력 클러스터에 점들이 집중되어 있어 클러스터링이 잘되었다는 것을 보여준다.

입력 출력	A	B	C	D	E	F	outlier	Sum
1	15	0	0	0	1428	94	49	1586
2	450	0	0	1	0	804	87	1342
3	1306	0	0	0	0	135	70	1511
4	0	0	1592	0	0	0	167	1759
5	96	0	0	1372	0	175	87	1730
6	0	2032	0	0	0	0	40	2072
Sum	1867	2032	1592	1373	1428	1208	500	10000

표 4.2 PROCLUS 결과 Confusion Matrix

반면에 입력 클러스터 F에 속하는 열의 숫자들은 여러 행으로 분산 되어 있다. 이것은 입력 클러스터 F에 속한 점들이 클러스터링이 잘 되지 못하여 여러 출력 클러스터들에 분산되어 있음을 보여준다. 그리고, outlier에 속한 점들도 역시 여러 출력 클러스터들에 걸쳐 분산되어 있다.

혼돈 행렬의 값에서 클러스터링 알고리즘의 실제 우수성은 우세 비율(dominant ratio)이라는 측정치로 평가한다.[9] 이것은 가장 우세한 입력 클러스터에 의해 차지하게 된 각 클러스터의 평균 부분 값이다. 이것을 구하는 과정은 다음과 같다. (1) 각 출력 클러스터에서 가장 높은 숫자를 갖는 입력 열의 숫자를 계산한다. (2) 각 행에서 가장 높은 값을 갖는 열을 정해서 행과 열의 짝을 맞춘다. (3) 그리고 짝을 이루는 값을 입력 클러스터에 속한 전체 점의 개수로 나누면 우세 비율을 얻을 수 있다.

표4.2에서 짝을 이루는 (출력 클러스터 행, 입력 클러스터 열)은 (6, B), (4, C), (1, E), (5, D), (3, A), (2, F)가 되고 ,각 항의 값을 더하면 8,534가 된다. Outlier를 포함한 입력 클러스터에 속한 전체 점의 개수는 10,000이므로 표 2.2의 혼돈 행력의 우세 비율은 0.85가 된다. 우세 비율이 1에 가까울수록 클러스터링은 깨끗하게 잘 된 것이다.

2. FASTDOC의 w에 따른 성능 평가

그림 4.1에서 FASTDOC알고리즘은 그림 2.4와 같이 box $B_{p,D}$의 크기를 결정하는 w에 의해 영향을 받는다. 실험에 의해, [1]에 제시된 입력 데이터에 대해서는 w값이 0.05일 때에 제일 좋은 성능을 보였다. w는 점 p와 가장 가까이 위치한 점과의 평균 거리에 상수 C를 곱하여 아래의 식과 같이 구하였다. 점 p_i와 가장 가까이 위치한 점 q_i라고 할 때,

$w = \frac{\sum_{i=1}^{n} \|p_i - q_i\|}{n} \times C$

(단, C 는 상수) , $O(n^2 d)$ 이다. w를 구하는 시간 복잡도는 $O(n^2 d)$이고 상수 C에 의해서 클러스터링의 정확도가 달라지게 된다. 그러나 상수 C는 실험에 의한 경험치 중 성능이 우수한 것을 선택한 것으로 입력 값이 달라지게 되면 또다시 상수 C를 조정하면서 실험을 할 수 밖에 없는 단점이 있다. 비록 FASTDOC이 성능 면에서는 우수한 것으로 나타났지만 상수 C를

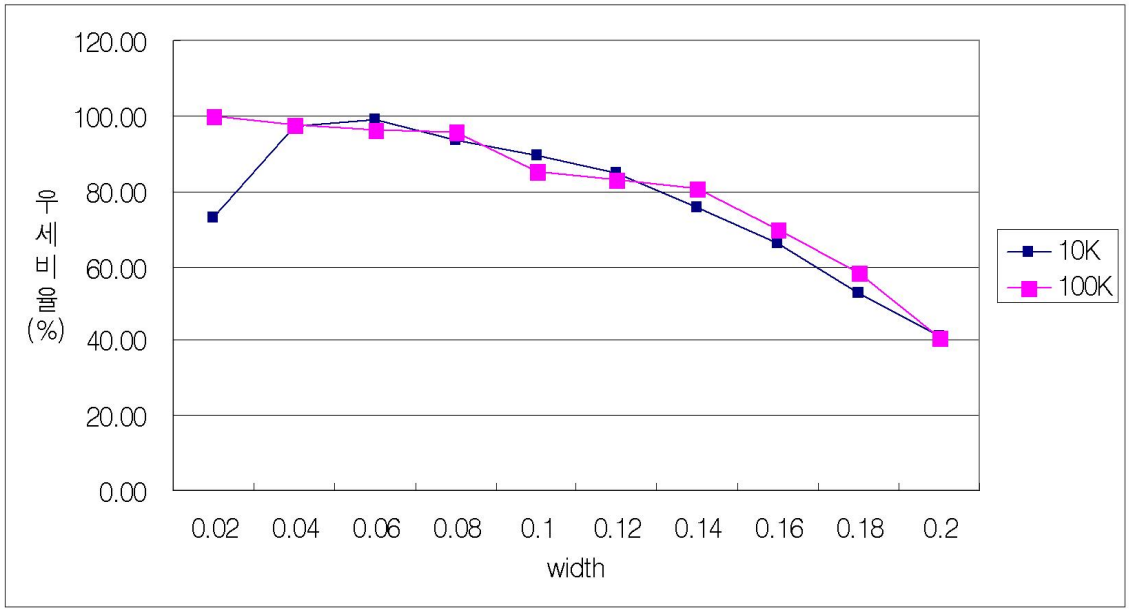


그림 4.1 FASTDOC의 w에 따른 우세 비율

정하는 심각한 문제[27]로 인해 실제 데이터를 기반으로 한 실험 결과가 요구될 때에는 적용하기 어려운 문제가 있다. 그림 4.2의 w에 따른 수행 시간은 우세 비율에 비례하여 변하고 있다. FASTDOC 알고리즘에서는 최대 관련된 부분 차원을 찾는 내부 반복에서 관련된 부분 차원의 개수가 평균치에 도달하게 되면 내부 반복을 빠져나오게 된다. width가 크다는 것은 많은 데이터를 포함하게 되어 수행시간이 더 소요 될 것처럼 보이지만, 그림 4.2의 FASTDOC에서는 관련된 차원의 개수가 빨리 평균치에 도달하게 되어 수행시간이 더 빨라지게 되었다.

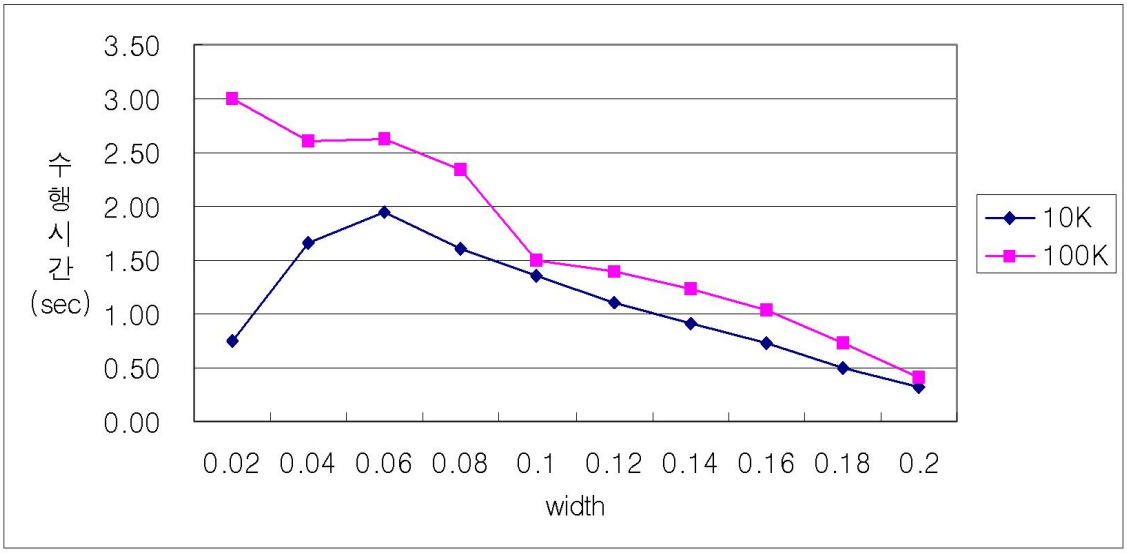


그림 4.2 FASTDOC의 w에 따른 수행 시간(sec)

3. 실험 결과

새로운 알고리즘의 성능을 평가하기 위하여 인텔 펜티엄Ⅳ 2.4GHz CPU와 2GB의 메인 메모리를 장착한 개인용 컴퓨터에서 실험하였다. 이 컴퓨터의 운영체제는 Windows XP Professional이고, 사용된 언어는 MS

Visual C++ 6.0이다. 1절에서 설명한 방법으로 실험데이터를 생성하여 알고리즘의 성능을 평가하였다.

실험은 제안된 알고리즘 DAM의 타당성과 자체 성능평가에 주안점을 두었다. 제안된 알고리즘에서는 두개의 변수의 값을 설정해야 하는데 이러한 변수의 값을 설정하는 실험결과에 대해서 설명한다. 첫 번째 bound 값에 따른 클러스터링의 정확도 측정하였고, 두 번째는 초기 분할 단계에서 클러스터의 개수에 따른 수행성능을 분석하였다. 세 번째는 앞의 분석을 기초로 하여 알려진 다른 클러스터링 알고리즘과 비교하였고, 마지막으로 입력 데이터의 차원을 달리 하여 FASTDOC 알고리즘과 성능비교를 하였다.

1) bound에 따른 클러스터링의 우세 비율

대부분의 대량의 데이터는 중심점을 기준으로 $\pm 3*\sigma$의 범위 내에 포함된다는 정규분포(Gaussian Distribution)의 성질을 이용하여 $B*\sigma$라는 한계를 두었다. 이 범위를 벗어난 점들에 대해서는 outlier로 두거나 다시 재배정하는 방법을 택했다. 실제 우리의 실험 데이터에 대해서는 bound 변수에 따라서 클러스터링의 정확도가 어떻게 변하는지 그림 4.3과 같이 실험해 보았다. [1]에서 제시된 방법으로 만들어진 100,000개와 10,000개의 입력 점들에 차원은 25차원으로 이루어지고 있다. bound의 범위는 2.0에서 5.0까지 0.5씩 늘려가며 실험하였다. 그림4.2에서 보면 결과는 예상했던 되로 정규분포의 성질에 따라 bound가 2에서 3사이 즉, $2*\sigma$에서 $3*\sigma$사이에서는 우세 비율이 상대적으로 낮았고 $3*\sigma$이상에서는 좋은 우세 비율을 보이고 있다. 10,000개의 데이터는 bound가 4.0일 때, 100,000개 데이터의 대해서는 3.5일 때 가장 좋은 성능을 보였다.

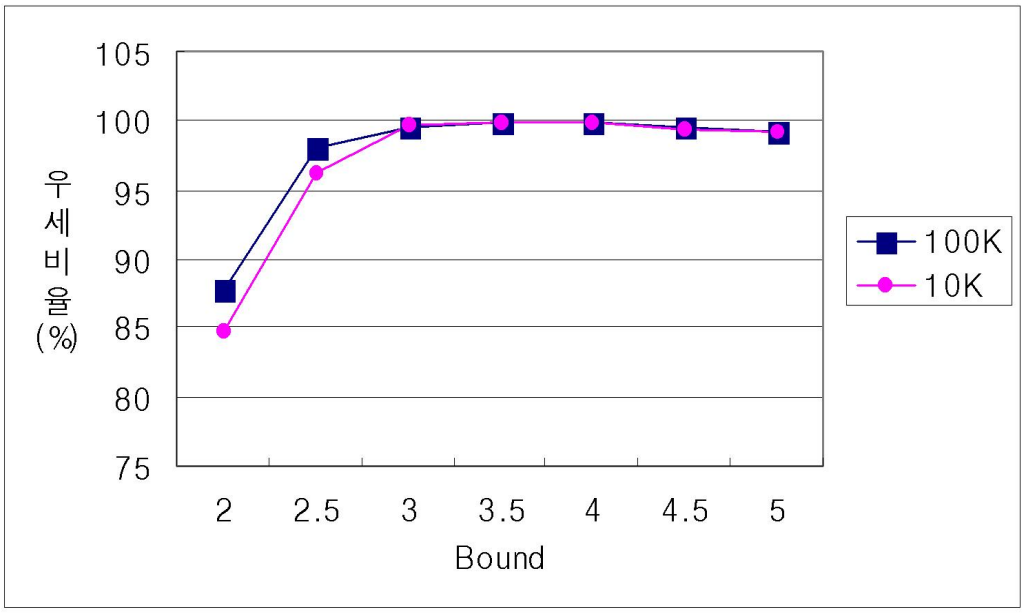


그림 4.3 bound에 따른 우세 비율

2) 초기 분할에 따른 클러스터링의 우세 비율

알고리즘 DAM의 초기 단계에서 클러스터의 개수는 $k_0 = A \cdot k$에 의해서 결정되어 전체 알고리즘의 성능에 영향을 미치므로 A의 크기에 따른 클러스터링의 결과 분석이 필요하다. 그림 4.4에서는 A의 변화에 따른 입력 데이터베이스의 크기를 10,000(10K)에서 100,000(100K)까지 다양하게 우세 비율을 보여주고 있다. 여기서 bound 변수는 4.0으로 두었다. 초기 분할이 너무 작게 되면 입력 데이터의 크기에 따라 고르지 못한 결과를 보여주고 있고, 그 외의 크기에는 고른 분포를 보이고 그 중에 A가 3일 때 모든 입력 데이터에 대해 가장 좋은 결과를 보이고 있다.

그림 4.5에서는 그에 따른 수행시간을 나타내고 있는데 A가 커짐에 따라 수행시간도 비례하여 커지고 있다.

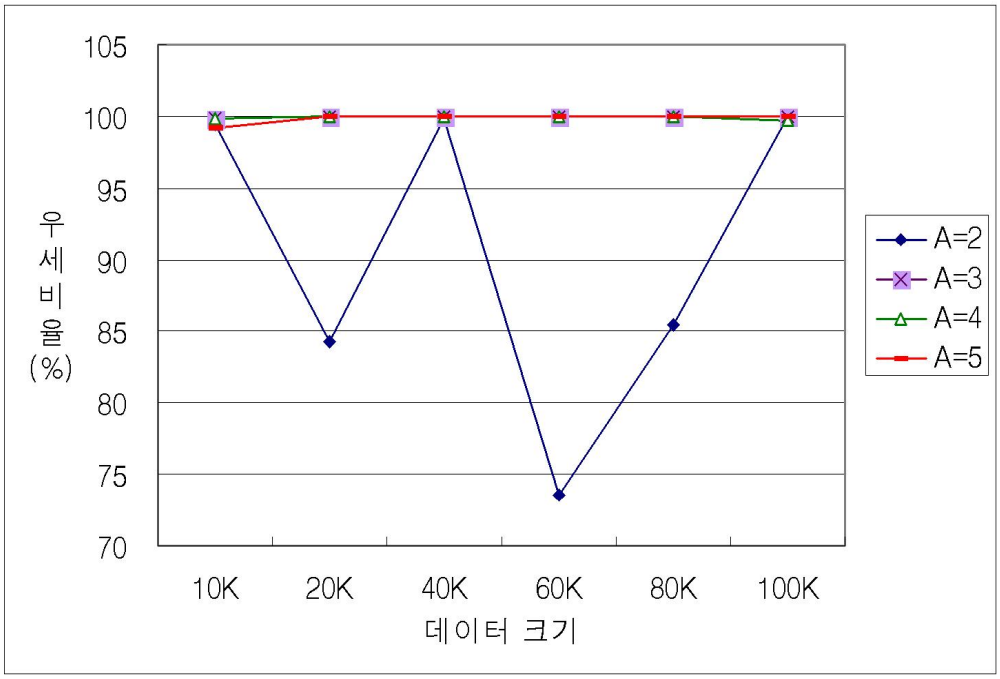


그림 4.4 초기분할 크기에 따른 우세 비율

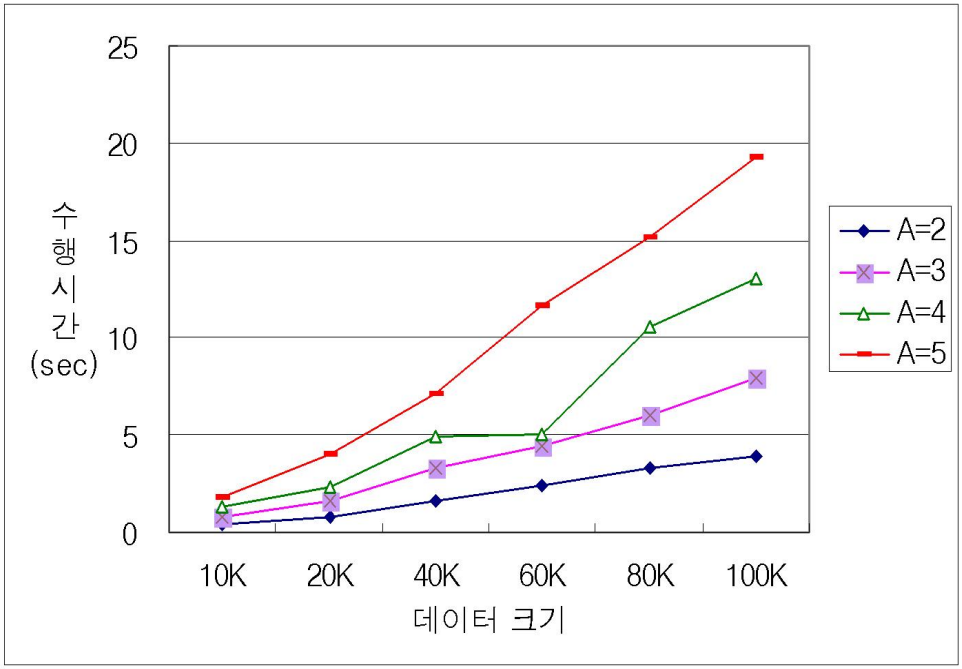


그림 4.5 초기분할 크기에 따른 수행 시간

3) 클러스터링 알고리즘들의 성능 비교

제안된 알고리즘과 성능을 비교하기 위하여 기존의 알고리즘 중 2장에서 설명되어진 3개의 알고리즘을 사용하였다. 고차원 데이터에 대해 성능이 좋은 FASTDOC과 PROCLUS 알고리즘, 널리 알려진 K-Means알고리즘이다. DOC 알고리즘은 내부 반복 단계에서 전체 데이터 집합을 계속적으로 스캐닝 하여 많은 시간을 소모하는 단점이 있다. 이에 내부반복단계에서는 차원 D만을 계산하고 횟수를 MAXITER로 제한한 FASTDOC을 구현하여 실험에 사용하였다. 또 본 연구실에서 구현되어진 PROCLUS와 K-Means 알고리즘으로 DAM과 비교 실험하였다.

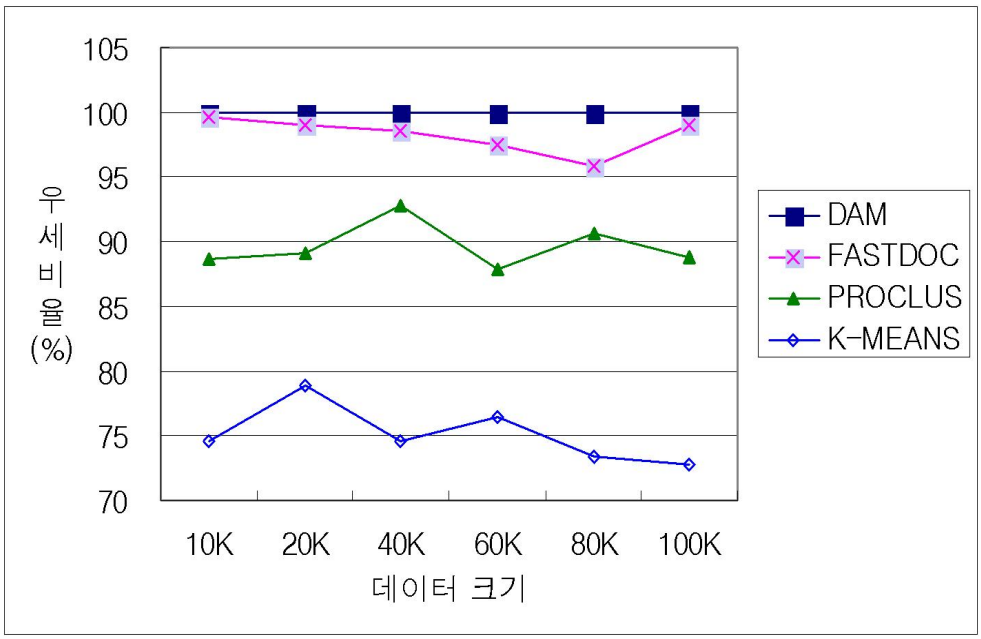


그림 4.6 클러스터링 알고리즘들의 우세비율 비교

그림 4.6은 표 4.1의 데이터를 입력으로 하여 20번을 실행하여 평균을 낸 실험 결과이다. DAM에서의 변수는 앞 절에서 언급한 것과 같이 A = 3, bound = 4.0으로 실험하였다.

K-Means 알고리즘은 전체 차원을 고려하는 단순한 알고리즘으로 부분

차원을 고려한 알고리즘에 비해 성능은 떨어지나 실행시간은 짧다. 제안된 DAM과 PROCLUS, FASTDOC 알고리즘은 고차원 데이터를 클러스터링 하는 과정에서 각 클러스터에 부분 차원을 선택하여 클러스터링 하는 알고리즘으로 성능이나 실행 시간 면에서 DAM이 우수함을 보이고 있다.

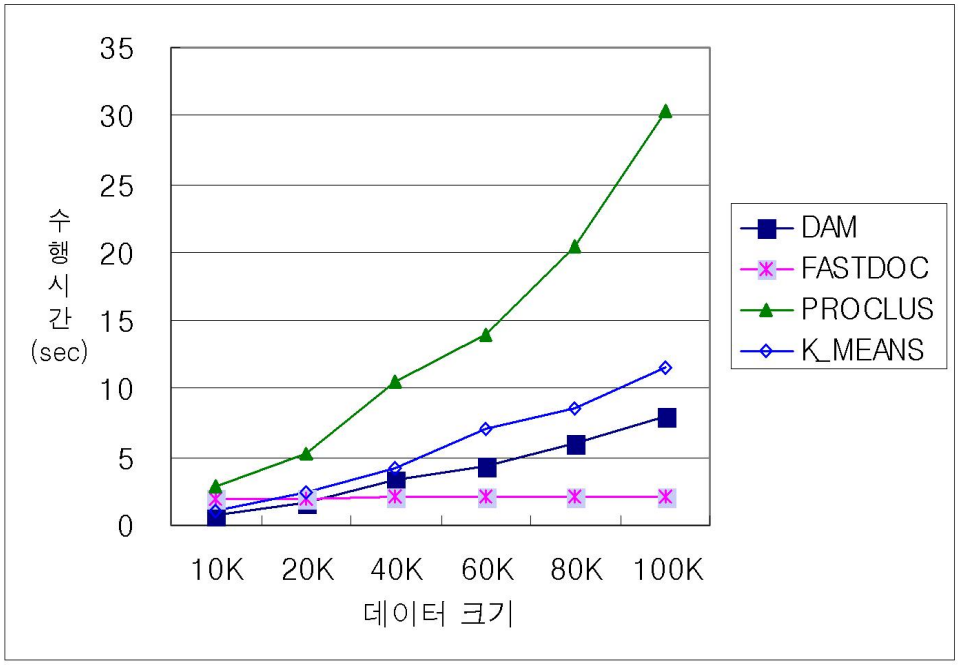


그림 4.7 클러스터링 알고리즘들의 수행시간 비교

그림 4.7에서는 입력 데이터 베이스의 크기를 10,000(10K)에서 100,000(100K)까지 변화 시키면서 4가지 알고리즘의 실행시간을 비교해 보았다. 특이한 점은 FASTDOC의 실행 시간이 입력 데이터의 크기에 거의 변화가 없다. 이는 앞에서도 말한 것처럼 내부 반복 단계에서의 횟수를 MAXITER로 제한하는 것과 관련이 있다. MAXITER는 차원에 비례하는 변수로 차원에는 변화가 없으므로 모든 입력 데이터 크기에 같은 회수의 내부 반복이 이루어진다. 나머지 알고리즘들은 입력 데이터에 크기에 비례하여 수행 시간이 길어진다.

4) 차원에 따른 알고리즘의 성능 비교

그림 4.8은 차원을 10차원에서 100차원까지 변화 시키면서 두 클러스터링 알고리즘인 DAM과 FASDOC의 우세 비율을 측정한 결과이다. 알고리즘 DAM은 초기 분할 변수 A를 3과 5로 두어 실험하였다. DAM은 차원의 수가 커짐에 따라 초기 분할 클러스터가 많을수록 우세 비율이 높았고, 차원에 따라 외부 반복이 횟수의 제곱에 비례하는 FASTDOC도 우세 비율이

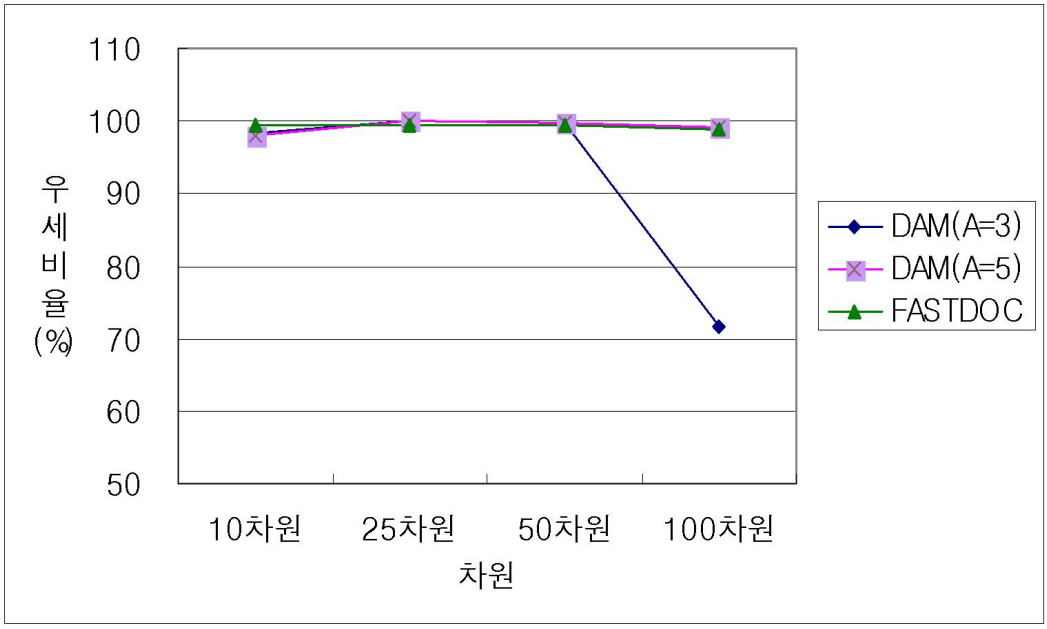


그림 4.8 차원에 따른 알고리즘의 성능 비교

높았다. 하지만 표 4.3에서의 수행 시간 면에서는 현저히 느려지는 것을 볼 수 있다.

알고리즘	DAM(A=3)	DAM(A=5)	FASTDOC
10차원	0.60sec	1.09sec	0.21sec
25차원	0.69sec	1.65sec	1.81sec
50차원	1.14sec	2.60sec	10.33sec
100차원	1.80sec	3.89sec	61.66sec

표 4.3 차원에 따른 알고리즘 수행 시간

V. 결론 및 향후 과제

본 논문에서는 고차원 데이터를 클러스터링 하는 알고리즘을 제안하였다. 제안된 알고리즘 DAM은 먼저 최대 표준 편차를 갖는 차원을 기준으로 원하는 개수보다 많은 클러스터들로 분할하고, 각 클러스터에 대해서 부분 차원을 찾았다. 찾아진 부분 차원을 고려하여 거리 함수를 계산하고, 입력 점들을 가장 가까운 중심점을 갖는 클러스터에 배정하였다. 다음 단계에서 조건에 미치지 못하는 클러스터를 찾아 재배정 하거나 outlier로 둔다. 마지막으로 클러스터간의 similarity가 큰 두 클러스터를 찾아내어 병합하고, 최종으로 원하는 개수의 클러스터가 될 때까지 과정을 반복한다.

클러스터링 알고리즘의 성능은 입력 클러스터와 출력 클러스터 사이의 점의 분포를 표시하는 혼돈 행렬(confusion matrix)을 사용하여 우세 비율(dominant ratio)로 계산하였다.

실험 데이터에서는 DAM이 기존 알고리즘인 K-Means, PROCLUS, 그리고 FASTDOC에 비해 실행 시간이나 우세 비율에서 성능이 좋음을 보여주고 있다.

향후 과제로는 알고리즘 DAM의 초기 단계인 분할 과정에서 각 클러스터의 균일한 개수의 점들을 갖도록 분할 함수를 개선하는 것이고, 각 클러스터에서 데이터의 분포가 밀집된 부분차원을 선택하는 과정, 그리고 두 클러스터 사이의 similarity를 계산하는 함수를 개선하는 연구가 필요하다. 또한 알고리즘들의 성능 평가에서 실제 데이터를 기반으로 한 실험 결과가 요구된다.

참고 문헌

[1] Charu C. Aggarwal, Ceilia Procopiuc, Joel L. Wolf, Philip S. Yu, and Jong Soo 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.

[2] R. Agrawal, J. Gehrke, D. Cunoploes, 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, 1998.

[3] R. Ng and J. Han, "Efficient and Effective Clustering Method for Spatial Data Mining", In Proceedings of the 20th VLDB Conference, pages 144-155, 1994.

[4] T. Gonzalez, "Clustering to minimize the maximum intercluster distance", Theoretical Computer Science, Vol.38, pp. 293-306, 1998.

[5] Jiawei Han, Micheline Kamber, "Data Mining: Concepts and Techniques", Morgan Kaufmann Publishers, San Francisco, CA, 2001

[6] J. Han, M. Kamber, and A. K. H. Tung, "Spatial Clustering Methods in Data Mining: A Survey", H. Miller and J. Han (eds.), {\sf

Geographic Data Mining and Knowledge Discovery}, Taylor and Francis, 2001.

[7] Sudipto Guha, Rajeev Rastogi and Kyuseok Shim, "CURE: An Efficient Clustering Algorithm for Large Databases", In Proceedings of ACM SIGMOD, pages 73-84, 1998.

[8] M. S. Chen, J. Han, and P.S. Yu, "Data Mining: An Overview from Database Perspective", IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, pp. 866-883, Dec. 1996.

[9] 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 (also, IEEE TKDE Vol 14, No 2, pp. 210-225, 2002)

[10] K.Mardia, J. Kent and J. Bibby, Multivariate Analysis, Academic Press, 1980.

[11] I.Joliffe, "principal Component Analysis", Springer-Verlag, New Your, NY, 1986.

[12] Sudipto Guha, Rajeev Rastogi, Kyuseok Shim, "ROCK: A Roust Clustering Algorithm for Categorical Attributes", the 15th International Conference on IEEE Data Engineering, 1999.

[13] Cecilia M.Procopiuc, Michael Jones, "A Monte Carlo Algorithm for Fast Projective Clustering", In the proceedings of the SIGMOD International Conference on Management of Data June 3-6, 2002.

[14] T. Zhang, R. Ramakrishnan, M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases", In Proceedings of the ACM SIGMOD International Conference on Management of Data, 1996.

[15] Rakesh Agrawal, Manish Mehta, John Shafer, Ramakrishnan Srikant, "The Quest Data Mining System", In Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining, 1996.

[16] R. Agrawal, T. Imielinski and A. Swami, "Database Mining: A Performance Perspective", IEEE Transformation on Knowledge and Data Engineering, pages 914-925, 1993.

[17] R. Kohavi, D. Sommerfield, "Feature Subset Selection Using the Wrapper Method: Overfitting and Dynamic Search Space Topology", In Proceedings of the First International Conference on Knowledge Discovery and Data Mining, 1995.

[18] Richard Neapolitan, Kumarss Naimipour, " Foundation of Algorithms" Jones and Bartlett Publisher, 1998.

[19] M.W. Berry and M.Browne, "Understanding Search Engines : Mathematical Modeling and Test Retrieval", SAM, 1999.

[20] Charu C.Aggarwal, Philip S.Yu, "Redefining Clustering for High-Dimensional Applications", IEEE Transactions on Knowledge and Data engineering, Vol.14, No.2, March/April, 2002.

[21] Jong Soo Park and Do-Hyung Kim, "An Effective Algorithm for Subdimensional Clustering of High Dimensional Data," The KIPS(Korea Information Processing Society) Transactions: Part D, Vol. 10-D, No 3, pp. 417-426, June 2003. (in Korean)

[22] M. Ester, H-P. Kriegel, J. Sander, X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, 1996.

[23] Ankerst, M., Breunig, M.M., Kriegel, H.-P. and Sander, J., OPTICS: Ordering Points To Identify the Clustering Structure., Proc. ACM SIGMOD99, Philadelphia PA, 1999.

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

[25] 성유진, “클러스터링 알고리즘의 성능분석 및 시각화”, 성신여자 대학교 석

사 학위 논문, 1999.

[26] 최혜선, "Projected Clustering 을 위한 개선된 알고리즘", 성신여자 대학교 석사 학위 논문, 2002.

[27] 조민정, “Projected Clustering Algorithms의 성능 비교 및 분석”, 성신여자 대학교 석사 학위 논문, 2003.

ABSTRACT

An Improved Algorithm for Subset Clustering

Kim, Younho
Department of Computer Science
Graduated School of Sungshin Women's University

The issue of finding cluster from high dimensional data has well known in the field of data mining due to its importance, because the clustering as the basic algorithm had been used in fields of processing data such as similarity search, customer segmentation, pattern recognition, trend analysis and classification. In addition, researches for clustering algorithms have increased in management information systems, because the clustering can process enormous data to get information.

In this research, a new algorithm is proposed for subdimensional clustering of high dimensional data. In the new algorithm, candidate clusters are created through partition, which input points are divided by dimension that has maximum variation. Next, two cluster are found, which have a maximum closeness among candidate clusters, and them merges until these cluster are the predefined number of clusters. Lastly, the data is refined one more time for improving quality of clusters. The result of extensive experiments with k-Means, PROCLUS and FASTDOC shows that the proposed algorithm is better than other algorithm.