고차원 입력 데이터에 대한 클러스터링 알고리즘의 성능 분석

성 유진, 박 종수[1)]
성신여자대학교 전산학과
{yjsung, jpark}@cs.sungshin.ac.kr

요약

컴퓨터로 처리되어 저장 관리되고 있는 대용량의 데이터에는 유용하고 가치 있는 정보들이 많이 내재되어 있을 수 있다. 최근 들어 데이터 마이닝 분야에서는 기존의 방법으로 쉽게 찾아내지 못하는 귀중한 정보나 패턴을 탐사하는 시도가 이루어지고 있다. 클러스터링은 대용량 데이터로부터 흥미로운 데이터 분포나 패턴을 찾기 위한 데이터 마이닝 방법 중 하나다.

본 논문에서는 대용량의 데이터에서 흥미로운 데이터 분포나 패턴을 찾아내는 기존의 클러스터링 알고리즘인 MST, CLARANS, CURE 및 PROCLUS를 구현하여 여러 입력 데이터에 대해 결과를 분석하였다. 고차원 데이터 입력에 대해서는 CLARANS와 PROCLUS 알고리즘의 결과를 분석하였다. PROCLUS는 클러스터링에서 데이터 희소성 문제를 해결하기 위해 발견된 클러스터와 이 클러스터에 관련된 차원을 찾아내는 알고리즘이다.

사용자가 결과를 보다 쉽게 이해 할 수 있도록 하기 위해서 자바 애플릿을 이용하여 결과로 나온 클러스터들의 분포를 칼라로 구분하여 보여줄 수 있도록 하였다. 고차원 데이터에 대한 클러스터링 알고리즘의 결과를 디스플레이하는 방법으로 각 클러스터의 분포 상황을 데이터 입력 차원 중에서 원하는 2차원이나 3차원을 선택하여 해당 차원에 대한 데이터 분포를 화면상에서 볼 수 있게 하였다. 부가적으로 고차원 데이터에 대한 성능 분석을 위해서 confusion matrix를 사용하여 전체적인 클러스터들에 대하여 정확하게 클러스터링이 되었는 지를 파악하게 하였다.

1. 서론

최근 몇 년간 데이터를 생성하고 저장할 수 있는 능력이 급격히 증가함에 따라 더 이상 기존의 데이터를 분석하던 방법들로는 데이터 내에 존재하는 유용한 정보들을 찾아내기 힘들게 되었다. 따라서 이러한 방대한 양의 데이터로부터 함축적이며 잠재적 유용성이 있는 정보를 발견할 수 있는 지식 탐사 방법인 데이터 마이닝에 대한 중요도가 더욱 증대되고 있다[8, 13, 14].

본 논문에서는 이러한 마이닝 방법 중 유사도가 높은 데이터들을 같은 그룹으로 분류하여 데이터의 분포나 패턴을 찾아내는 클러스터링 알고리즘들을 구현하여 각 알고리즘의 특성과 성능에 대해 비교, 분석하여 보았다. 구현된 알고리즘은 모든 차원을 고려하는 기존 알고리즘인 MST, CLARANS[2], CURE[1]와 효과적인 고차원 데이터 클러스터링을 위해 관련 차원을 찾아내어 클러스터링을 하는 알고리즘인 PROCLUS[3]이다. 고차원 데이터 클러스터링의 성능 평가를 위해 PROCLUS와 CLARANS를 적용하여 실험하였다.

고차원 데이터에 대한 클러스터링 결과를 쉽게 이해할 수 있도록 자바 애플릿을 이용해서 결과를 클러스터 별로 칼라를 구분하여 최대 3차원까지 선택된 차원에 투사하여 볼 수 있게 하였다. 또한 이러한 시각화 작업은 2차원 데이터의 경우 각 클러스터링의 정확성을 시각적으로 보여줌으로써 클러스터링의 성능을 평가할 수 있으며, 결과에 대한 사용자의 이해를 도울 수 있다.

---

1) 서울 성북구 동선동 3가 249-1, 성신여자대학교

본 논문의 구성은 다음과 같다. 제 2장의 관련연구에서는 클러스터링 알고리즘들에 관해 살펴보고, 제 3장에서는 알고리즘 구현 및 실험결과에 관해 알아보고, 제 4장에서는 시각화에 대해 알아본다. 마지막으로 제 5 장은 결론 및 향후 과제로 구성되어 있다.

2. 관련연구

2.1 모든 차원을 고려하는 알고리즘

2.1.1 MST (Minimum Spanning Tree)

Hierarchical 클러스터링 방법으로 처음의 모든 입력 데이터 포인트들이 개별적인 클러스터를 구성한다. 가장 짧은 거리를 갖는 두 데이터 포인트가 속한 클러스터끼리 합병한다. 같은 방식으로 k개의 클러스터가 될 때까지 계속해서 합병해 나간다. Ouliter에 상당히 민감하며 또한 outlier가 줄지어 있을 경우 chaining effect가 발생하게 된다. 또한 클러스터 합병 정보로 클러스터 내의 모든 데이터 정보를 이용하기 때문에 수행 속도도 매우 느리다.

2.1.2 CLARANS (Clustering Large Application based on Randomized Search)

CLARANS 알고리즘은 PAM과 CLARA를 결합한 알고리즘으로 CLARA가 검색의 모든 단계에서 정해진 일정 샘플만을 이용하는 반면 CLARANS는 전체 데이터 집합에서 임의의 샘플을 뽑는다. 클러스터링 과정은 모든 k-medoid들의 집합인 그래프를 검색하는 것으로 생각될 수 있다. 두 개의 k-medoid 집합에서 하나의 medoid만 다를 경우 이웃(neighbor)이라고 하고, 임의로 선택되는 이웃들의 개수는 변수 maxneighbor에 의해 결정된다. 만약 더 좋은 이웃이 발견되면 CLARANS는 그 이웃 노드로 옮겨가서 위의 과정을 다시 반복하고, 발견되지 않는다면 현재의 k-medoid를 이용한 클러스터링을 지역적 최적(local optimal) 해결책으로 선택한다. 지역적 최적 해결책이 찾아지면 CLARANS는 새로운 지역적 최적 해결책의 검색을 위해 새로운 임의의 노드를 선택한다. 검색될 지역적 최적 해결책들의 개수는 매개변수 numlocal에 의해 결정된다[2].

2.1.3 CURE (Clustering Using Representatives)

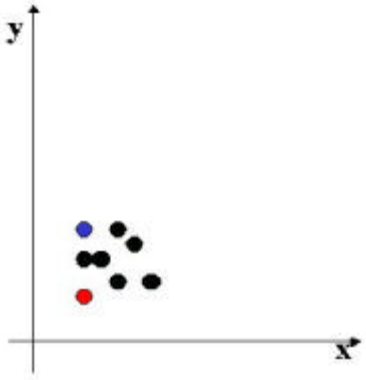
입력으로 들어온 모든 데이터를 개별적 클러스터로 시작하여 k개의 클러스터가 될 때까지 합병해 나가는 hierarchical 방식이지만 클러스터 내의 포인트들을 모두 계산하지 않고 c개의 클러스터의 대표값을 이용한다. 이 대표값들은 각 클러스터로부터 클러스터의 모양을 가장 잘 표현할 수 있는 c개의 적절히 분포된 포인트를 선택하여 클러스터의 중심점으로 입력으로 주어진 α shrinking factor만큼 축소시켜 구해진 점들이다. 클러스터의 합병을 위해서 거리를 계산 할 때 c개의 대표값을 이용함으로써 outlier에 민감하고 chaining effect의 단점을 보여주던 MST를 개선하였다. 그리고, 대표값 하나를 사용함으로써 발생했던 centroid-based approach의 단점인 하나의 클러스터가 두 개 혹은 그 이상으로 분리되는 문제점을 개선했다.

2.2 관련 차원만을 고려하는 알고리즘

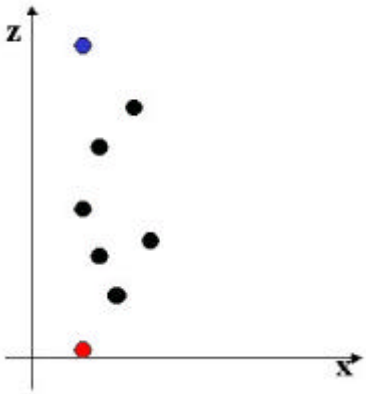
고차원 데이터에 대한 클러스터링의 문제점은 데이터 자체의 희소성(sparsity)으로 인한 클러스터링 성능의 저하이다. 예를 들어 고차원 응용에서 주어진 한 쌍의 두 포인트가 있다고 하면 두 포인트가 대부분의 차원에서도 매우 가깝다 할지라도 거리가 먼 차원도 있을 수 있다. [그림 1]에서 보면 3차원 공간의 포인트 집합을 x-y 평면에 따로 투사시켜 보면 포인트들이 밀집되어 있는 패턴을 찾을 수 있다. 하지만 3차원 모두를 고려해서 이러한 패턴들을 찾기는 매우 힘들다. 왜냐하면 패턴을 가지고 있는 2차원 외에 나머지 한 차원의 데이터가 [그림 2]에서처럼 전체 공간에 분포되어 있다고 한다면 3차원 모두를 고려했을 때 포인트 간의 거리가 그 나머지 한 차원에 의해 영향을 받기 때문이다. 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 있으나 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 의미있는 차원들을 정보의 손실을 최소화시키면서 잘 추출해 낼 수 있어야 하고, 결과에 대한 해석 능력 뿐만 아니라 사용자가 이해할 수 있도록 표현할 수 있어야 하며 차원 수와 데이터의 사이즈가 커지더라도 빠르게 처리할 수 있어야만 한다[3].



[그림 1] Cross Section on X-Y



[그림 2] Cross Section on X-Z

2.2.1 PROCLUS (Projected Clustering)

기존의 알고리즘들은 데이터의 모든 차원을 고려한 알고리즘들이다. 고차원 데이터에 대한 클러스터링에서 언급되었듯이 데이터의 차원이 증가할수록 데이터의 희소성(sparsity) 문제 때문에 모든 차원을 고려하여 클러스터링을 하게되면 성능이 현저히 떨어짐을 볼 수 있다. PROCLUS에서는 전체 차원이 아닌 일부 차원에서만 클러스터를 찾게된다. Projected cluster란 전체 차원 D 중 부분 차원에 존재하는 클러스터를 의미한다.

PROCLUS는 CLARANS에 관련 차원을 찾는 과정을 더한 알고리즘으로 CLARANS 알고리즘에서처럼 k개의 medoid를 선택한 후, 각 medoid와 관련이 높은 차원을 찾아내 클러스터링을 한다. 주어진 iteration 회수만큼 이 과정을 반복, 결과의 dissilmilarity가 가장 낮은 k-medoid를 갖는 클러스터들을 결과로 한다[3].

3. 알고리즘 구현 및 실험결과

본 논문에서 알고리즘의 성능 평가를 위한 데이터를 생성하는 방법은 다음과 같다. 2차원 데이터 생성을 위해서는 특정한 데이터 분포 함수를 적용하지 않고, 각 클러스터링 알고리즘의 특성을 반영 할 수 있도록 사각형 혹은 원 모양의 클러스터를 만들어주도록 하였다. 고차원 데이터 생성을 위해서는 [3]에서 제시된 데이터 생성 알고리즘을 이용하였다.

3.1 2차원 데이터 생성

2차원 데이터의 생성은 입력 데이터의 개수와 클러스터 개수인 k를 입력 파라미터로 하였고, 각 클러스터의 밀도는 동일하게 하여 생성하였으며 데이터의 범위는 [0.0, 1.0)사이의 랜덤 값으로 지정된 크기의 사각형 혹은 원 모양의 클러스터를 만들 수 있도록 하였으며 outlier의 최대 비율은 전체 데이터의 5%로 하였다.

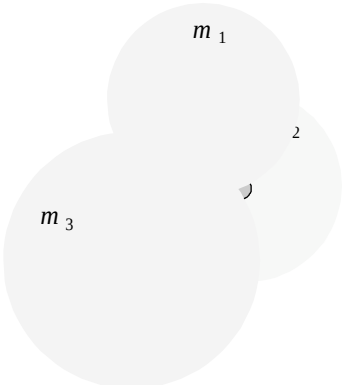
3.1.1 고차원 데이터 생성

고차원 데이터 생성은 [3]에서 제시한 데이터 생성 방법을 이용하였다.

3.2 알고리즘 구현

알고리즘 구현에서는 MST, CLARANS, CURE 알고리즘에 관한 자세한 구현 설명은 생략하고 PROCLUS 알고리즘에 관해서만 설명한다.

PROCLUS 알고리즘은 초기화 단계, 반복 단계, 정제 단계의 세 단계로 구성되어 있다. 초기화 단계는 알고리즘 수행 중 사용하게 될 medoid 샘플을 구하는 단계로 먼저 전체 데이터 집합으로부터 A • k개의 샘플을 구한 후 greedy 방법을 이용하여 B • k개 medoid 샘플을 다시 구한다. 이 경우 A > B이다. 두 번에 걸쳐 샘플링을 하는 이유는 greedy 방법을 전체 데이터 집합에 바로 적용할 경우 greedy 방법에서 사용하는 거리 계산의 특성상 outlier가 많이 선택될 수 있기 때문이고 또한 탐색 범위를 A • k로 줄임으로써 초기화 단계의 수행 시간을 줄이고자 함이다. 반복 단계에서는 B • k 개 medoid 샘플로부터 임의의 k개의 medoid 집합을 고른 뒤 현재 medoid 집합에서 클러스터에 속한 포인트들의 개수가 제일 작은 medoid를 계속해서 바꿔 줌으로써 클러스터링의 질을 향상 시켜나간다. $M_{current}$가 선택되고 나면 $M_{current}$의 locality를 이용해 각 medoid와 관련된 차원들을 선택하게 된다. PROCLUS에서 가장 중요한 차원을 찾는 방법은 일단 주어진 k medoid 집합 $M = \{m_1, \ldots, m_k\}$의 각 m_i에 대해 $\delta_i = \min_{j=i} d(m_i, m_j)$인 δ_i를 구한다. [그림 3]은 k가 3인 medoid 집합, $M = m_1, m_2, m_3$에 대해 δ_i을 2차원 평면에서 나타낸 것이다. 각 medoid의 locality L_i는 다음과 같이 정의된다. L_i는 m_i로부터 δ_i만큼의 거리 안에 있는 포인트들의 집합이다.($L_i, \ldots L_k$가 반드시 disjoint할 필요는 없다.) 각 L_i에 대해 차원별로 m_i과 δ_i사이의 평균 거리를 구한다. 각 medoid m_i에 대해서 통계학적 기대치가 가장 작은 j차원들을 선택한다. 이 때 medoid와 관련된 전체 차원의 수는 $k \cdot l$과 같아야만 한다. Medoid와 관련 차원이 구해지면 입력 포인트들을 medoid 중 가장 가까운 medoid에 할당한다. Medoid와의 거리를 계산할 때는 medoid에 관련된 차원만 고려한다. 이렇게 $M_{current}$를 이용하여 클러스터가 형성이 되면 $M_{current}$에 대한 평가를 한다. M_{best}과 비교하여 $M_{current}$가 더 나을 경우 $M_{current}$를 M_{best}로 바꿔준다. 이 과정을 계속해서 termination_criterion이 만족될 때까지 반복한다. 마지막 정제 단계에서는 M_{best}의 클러스터들을 이용해 locality를 구한 후 관련 차원을 찾아서 클러스터를 형성하는 확인 단계이다.



[그림 3] Locality

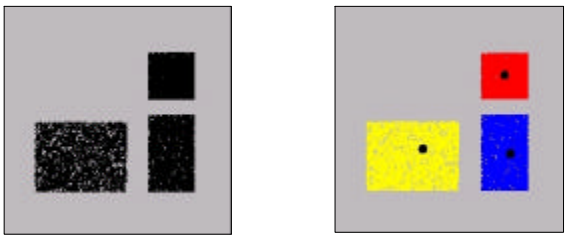
4. 성능 분석

구현한 알고리즘들의 성능을 분석하기 위하여 여러 실험 데이터를 가지고 실험을 하였다. 입력 클러스터의 분포, outlier의 존재 여부, 차원의 수 등을 달리하여 실험하였다.

클러스터링의 정확도를 평가하기 위해서는 동일한 데이터에 대한 각 알고리즘의 결과들을 시각화하여 평가하는 방법과 confusion matrix를 통하여 수치적으로 클러스터링의 정확도를 평가하는 방법을 이용하였다.

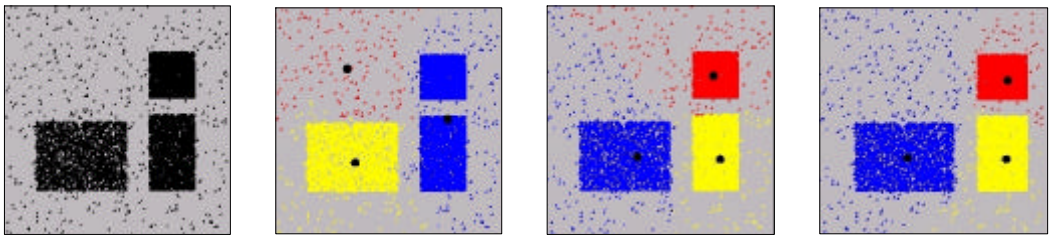
4.1 Outlier에 민감한 정도

클러스터의 사이즈가 비슷하고 잘 분리되어 있는 입력에 대해 각 알고리즘을 outlier가 존재하는 데이터와 존재하지 않는 데이터로 나누어 outlier에 의한 영향을 살펴보았다. 전체 입력 데이터 포인트는 10000개이고, outlier는 전체 입력 데이터의 5%이다. 결과적으로 MST가 outlier에 가장 큰 영향을 받는 것으로 나타났으며, CLARANS와 CURE는 잘 분리된 입력 데이터에 대해서는 outlier에 상관없이 거의 정확한 클러스터를 찾아냈다. [그림 4]와 [그림 5]는 MST, CLARANS, CURE 각각에서 outlier가 존재하지 않는 데이터와 존재하는 데이터에 대한 클러스터링 결과를 보여준다.



(a) 입력 (b) MST, CLARANS, CURE

[그림 4] Outlier가 없는 입력에 대한 클러스터링 결과



(a) 입력 (b) MST (c) CLARANS (4) CURE

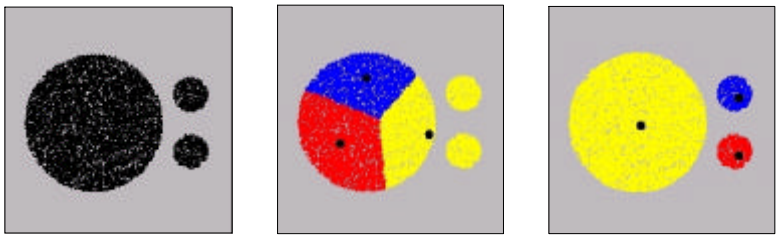
[그림 5] 5% outlier를 가진 입력에 대한 클러스터링 결과

4.2 입력 데이터 분포에 따른 클러스터링 성능

입력 클러스터의 분포와 사이즈를 아래의 [그림 6]과 [그림 7]의 입력 데이터처럼 한 클러스터의 사이즈를 다른 클러스터보다 크게 하고(왼쪽의 큰 클러스터는 작은 클러스터 면적의 4배이다.), 클러스터 간의 간격이 가까운 입력 데이터에 대해서 알고리즘 CLARANS와 CURE의 클러스터의 성능을 알아보았다[1]. 전체 입력 데이터 포인트는 10000개이고, outlier는 전체 입력 데이터의 5%이다.

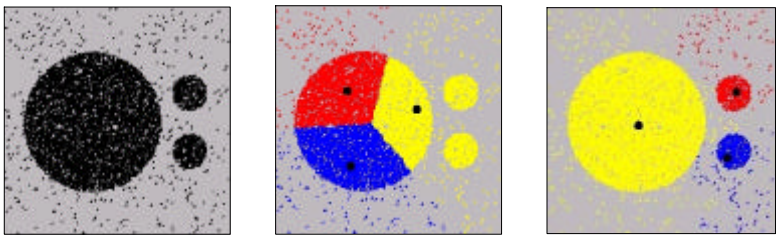
결과를 보면 [그림 6]의 결과 중 CLARANS 경우 outlier가 존재하지 않는 데이터에 대

해서도 바른 결과를 내지 못하고 있으며, 전형적인 centroid-based 알고리즘 방식의 문제점인 하나의 큰 클러스터가 여러 개의 클러스터로 나누어지는 결과를 보여 주고 있다. CURE의 경우는 outlier가 존재하지 않는 입력에 대해서는 대표값 c = 10, shrinking factor = 0.5에서 정확한 결과를 보여준다. [그림 7]은 동일한 분포에 대해 5% outlier를 준 입력의 결과이다.



(a) 입력 (b) CLARANS (c) CURE

[그림 6] 큰 클러스터를 입력으로 가진 클러스터링 결과



(a) 입력 (b) CLARANS (c) CURE

[그림 7] 큰 클러스터를 입력으로 가진 클러스터링 결과
(with 5% outlier)

4.3 고차원 데이터에 대한 성능 평가

고차원 데이터에 대해서는 CLARANS와 PROCLUS 두 알고리즘을 가지고 성능을 평가해 보았다. [표 1]의 입력 데이터는 20차원의 입력 포인트 10000개, outlier는 전체 입력 데이터의 5%, 클러스터 당 관련 차원은 4인 데이터이다.

CLARANS 알고리즘의 성능을 분석하여 보면 [표 2]의 CLARANS 알고리즘 클러스터링 confusion matrix에서 보듯이 결과로 나온 클러스터가 여러 개의 입력 클러스터로 나누어져 있음을 볼 수 있다. 이는 특정 차원에서만 클러스터가 존재하는 입력 데이터를 전체 차원을 고려하여 클러스터링 하였기 때문에 제대로 된 클러스터를 찾기 힘들기 때문이다. [표 4]에서 PROCLUS의 클러스터링 결과를 보면 입력으로 들어간 클러스터 당 포인트 개수가 거의 차이가 없음을 볼 수 있고, 또한 [표 5]의 confusion matrix를 보면 outlier에 의해 입력 클러스터의 포인트 수 보다 클러스터 내의 포인트 수가 증가하긴 했으나 outlier를 제외한 다른 클러스터의 포인트가 해당 클러스터의 포인트로 인식된 경우는 없음을 볼 수 있다.

[그림 8]과 [그림 9]는 클러스터링 결과를 시각화해서 보여준다. 입력 데이터는 4차원 상에 만들어진 클러스터들로 구성되지만, 그림에서는 3차원 상에 투사시켰을 때 클러스터링 결과를 보여주고 있다. 위의 confusion matrix나 클러스터링 결과표에서 성능을 확인할 수 있었지만 [그림 8]과 [그림 9]의 (b)PROCLUS를 보면 4개의 클러스터가 서로 다른 칼라로 확연히 구분되어 있음을 볼 수 있고, [그림 8]과 [그림 9]의 (a)CLARANS를 보면 하나의 클러스터에 여러 가지 칼라가 섞여 있음을 볼 수 있다.

Input	Dimensions	Points
A	2, 4, 11, 14	2584
B	2, 3, 7, 14	2812
C	2, 12, 14, 17	2204
D	2, 3, 13, 14	1900
Outliers		500

[표 1] n = 10000, k = 4, **평균차원 = 4인 입력 데이터**

Found	Dimensions	Points
1	Full dimensions	2701
2	Full dimensions	1736
3	Full dimensions	1965
4	Full dimensions	3598
Outliers		

[표 2] CLARANS **알고리즘의 클러스터링 결과**

Input / Output	A	B	C	D	Out.
1	834	1062	21	649	135
2	283	276	156	953	68
3	196	5	1602	25	137
4	1271	1469	425	273	160
Outliers					182

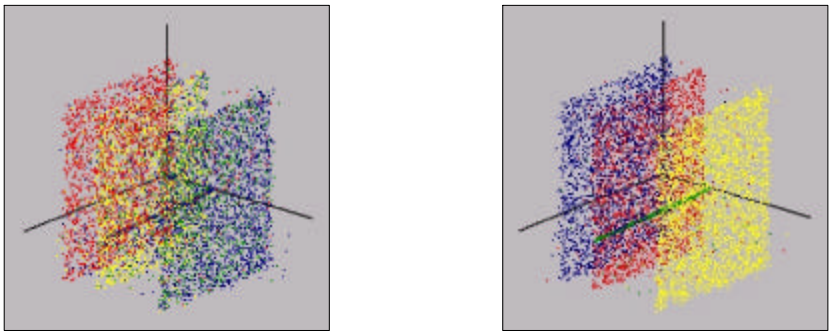
[표 3] CLARANS: Confusion Matrix

Found	Dimensions	Points
1	2, 4, 11, 14	2659
2	2, 3, 7, 14	2859
3	2, 12, 14, 17	2360
4	2, 3, 13, 14	1940
Outliers		182

[표 4] PROCLUS 알고리즘의 클러스터링 결과

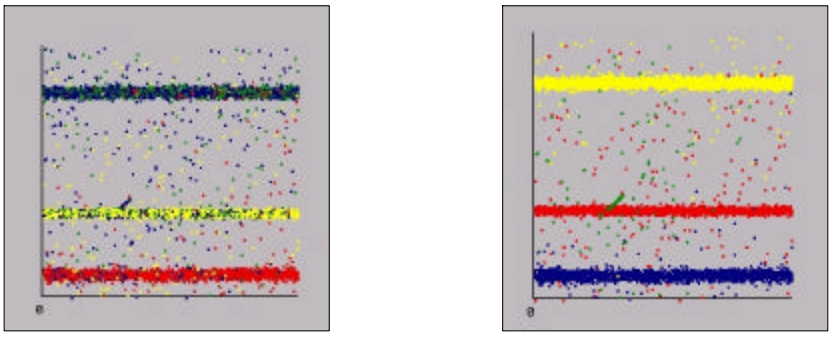
Input / Output	A	B	C	D	Out.
1	0	0	0	1900	135
2	0	2812	0	0	47
3	2584	0	0	0	75
4	0	0	2204	0	156
Outliers					182

[표 5] PROCLUS: Confusion Matrix



(a) CLARANS (b) PROCLUS

[그림 8] 20차원 데이터에 대한 클러스터링 결과의 3차원 공간에의 투사



(a) CLARANS (b) PROCLUS

[그림 9] 20차원 데이터에 대한 클러스터링 결과의 2차원 공간에의 투사

5. 시각화

통계학이나 패턴 인식 등 데이터 분석이 중요한 역할을 하는 분야에서 시각화는 중요한 작업이 되어 왔다. 시각화 작업은 데이터 마이닝에서도 사용자가 결과를 명확히 이해하도록 돕기 위해 매우 중요하다. 그러나 대용량의 데이터를 다룸에 있어 데이터의 차원이 점차적으로 증가하고 있기 때문에 이를 시각화하는데 많은 어려움이 따르고 있다[9, 15].

클러스터링에서 시각화 작업은 사용자가 결과를 쉽게 이해할 수 있도록 하기 위한 방법일 뿐만 아니라 결과의 정확도를 평가하기 위한 방법의 하나이며 고차원 데이터의 경우 선택한 차원에 투사시켜 봄으로써 데이터의 결과와 분포에 대한 이해를 도울 수 있다.

본 논문에서는 클러스터링 결과를 자바 애플릿을 이용해 k개의 칼라로 구분하여 보여주도록 하였다.

5.1 2차원 데이터 결과에 대한 시각화

클러스터 개수인 k와 클러스터링 결과를 입력으로 하여 각 결과로 나온 클러스터를 칼라로 구분하여 결과를 보여준다. Centroid-based 알고리즘의 경우는 center 혹은 medoid 값도 표시함으로써 결과의 이해를 도왔다.

[그림 10]은 2차원 데이터를 클러스터링 한 결과를 보여주는 애플릿으로 Display Data 버튼을 클릭하게 되면 결과 파일을 읽어 클러스터 별로 칼라를 구분하여 보여준다.

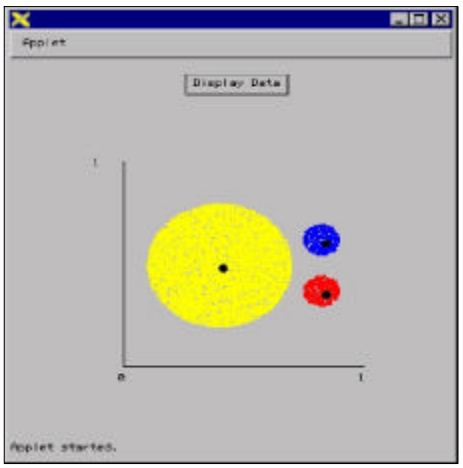
5.2 고차원 데이터 결과에 대한 시각화

3차원 이상의 데이터에 대해서는 시각화 작업이 어렵기 때문에 고차원 데이터의 결과를 시각화 할 때는 입력으로 들어온 d개의 차원 중 선택된 2차원 혹은 3차원에 투사시켜 볼 수 있도록 하였다.

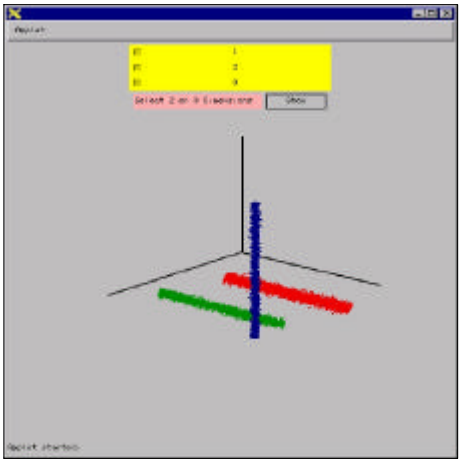
애플릿에 d개의 차원을 체크 박스로 만들었으며, 그 중 2개 혹은 3개 차원을 선택 버튼 중에서 선택하여 Display 버튼을 누르면 선택된 차원에 투사된 결과 데이터를 k개의 칼라로 보여주게 된다.

3차원이 선택되는 경우 3차원 변형 작업을 하여 입체적으로 보여줄 수 있도록 하였다.

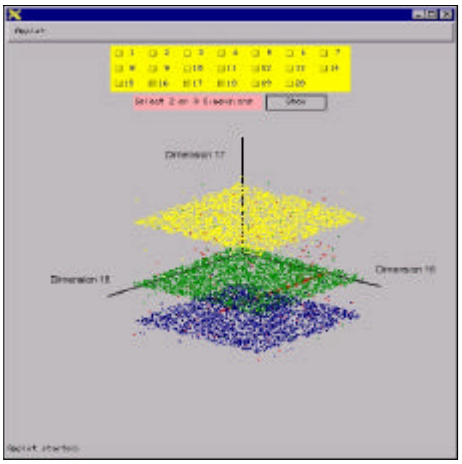
[그림 11]은 3차원 데이터에 대한 클러스터링 결과를 보여주는 애플릿이고 [그림 12]는 20차원 데이터에 대한 클러스터링 결과를 보여주는 애플릿이다. [그림 13]은 [그림 12]의 선택된 3차원 중 다시 2차원을 선택하여 2차원 평면상에서 결과를 보여주고 있다.



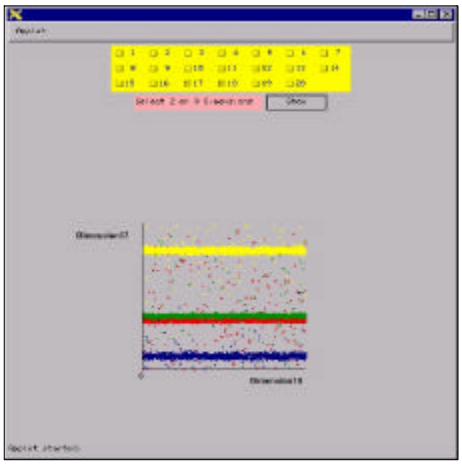
[그림 10] 2차원 데이터에 대한 시각화



[그림 11] 전체 3차원인 데이터에 대한 시각화



[그림12] 전체 20차원 데이터에 중 3차원에 대한 시각화



[그림 13] 전체 20차원 데이터에 중 2차원에 대한 시각화

6. 결론 및 향후 과제

본 논문에서는 유사도가 높은 성격을 가진 데이터들을 같은 그룹으로 분류하여 데이터의 분포나 패턴을 찾아내는 클러스터링 알고리즘들을 구현하여 각 알고리즘의 특성과 성능에 대해 비교. 분석하여 보았다. 입력 데이터와 여러 가지 알고리즘에 따른 매개 변수들을 변화시켜 실험해 본 결과 모든 차원을 고려한 클러스터링 알고리즘들(MST, CLARANS, CURE) 중에서는 CURE가 정확도 면에서는 성능이 가장 우수함을 알 수 있었다. 또한 고차원 데이터로 실험 해본 결과 PROCLUS의 클러스터링의 성능이 우수함을 알 수 있었다. 기존 알고리즘들은 고차원 데이터에서 클러스터를 제대로 찾아내지 못했다.

사용자가 결과를 쉽게 이해할 수 있도록 자바 애플릿을 이용해 각 클러스터의 칼라를 구분해 보여주었으며 시각화 작업은 고차원 데이터의 결과를 해석하는데 도움을 주었고 각 알고리즘 성능을 시각적으로 평가 할 수 있도록 하였다.

향후 과제로는 구현된 알고리즘을 실제 데이터에 적용시켜 보는 것과 포인트 간의 거리를 이용하던 전통적인 방식 외에 boolean 타입의 데이터나 categorical 타입의 데이터에 대한 클러스터링 알고리즘의 구현이다. 또한 결과를 보여주는 시각화 작업을 좀 더 향상시키는 것이다.

참고 문헌

[1] S. Guha, R. Rastogi and K. Shim, CURE: "A Efficient Clustering Algorithm for Large Databases", In *Proceedings of ACM SIGMOD, pages 73-84, 1998.*

[2] 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.

[3] C. C. Aggarwal, C. 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.

[4] 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.

[5] S. Guha, R. Rastogi, and K. Shim, "ROCK: A Roust Clustering Algorithm for Categorical Attributes", the 15th International Conference on IEEE Data Engineering, 1999.

[6] M. Ester, H.-P. Kreigel, J. Sander, "A Density Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", In *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining, 1995.*

[7] 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.

[8] R. Agrawal, M. Mehta, J. Shafer, and R. Srikant, "The Quest Data Mining System", In *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining, 1996.*

[9] T. Fukuta, Y. Morimoto, and S. Morishita, "Data Mining Using Two-Dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization", In *Proceedings of ACM SIGMOD*, pages 13-23, 1996.

[10] R. Agrawal, T. Imielinski and A. Swami, "Mining Association Rules between Sets of Items in Large Databases", In *Proceedings of ACM SIGMOD*, pages 207-216, 1993.

[11] R. Agrawal and R. Srikant, "Mining Sequential Patterns", In *Proceedings of the 11th International Conference on Data Engineering*, pages 3-14, 1995.

[12] M.-S. Chen, J. S. Park, and P. S. Yu, "Data Mining for Path Traversal Patterns in a Web Environment", In *Proceedings of the 16th International Conference on Distributed Computing Systmes*, pages 385-392, 1996.

[13] 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.

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

[15] S. M. Weiss and N. Indurkhya, "Predictive Data Mining", Morgan Kaufmann Publishers, Inc, 1998.

[16] **김지현**, "**연관 규칙 기반 패턴 마이너의 설계 및 구현**", **성신여자 대학교 석사 학위 논문**, 1999.

[17] **장성원**, "**교양과목에 관한 수강 신청 데이터에서 연관 규칙 탐사와 순차 패턴의 추출**", **성신여자 대학교 석사 학위 논문**, 1997.