다차원 데이타 공간에서 시퀀스 데이타 세트를 위한 클러스터링 기법

(Clustering Technique for Sequence Data Sets in Multidimensional Data Space)

이 석 룡 † 임 동 혁 †† 정 진 완 †††
(Seok-Lyong Lee) (Tong-Hyeok Lim) (Chin-Wan Chung)

요 약 비디오 스트림이나 음성 아날로그 신호와 같은 연속된 데이타는 특징 공간(feature space)에서 다차원 데이타 시퀀스(multidimensional data sequence)로 모델링될 수 있다. 본 논문에서는 이러한 다차원 데이타 시퀀스의 효과적인 클러스터링 기법에 대하여 연구한다. 각 시퀀스는 차후의 저장 및 유사성 검색(similarity search)을 효율적으로 실행하기 위하여 소수 개의 하이퍼 사각형 (hyper-rectangle) 형태의 클러스터로 표현된다. 본 논문에서는 사전에 정의된 수준의 클러스터링 품질을 보장하는 선형 복잡도를 갖는 클러스터링 알고리즘을 제시하고, 다양한 비디오 데이타에 관한 실험을 통하여 알고리즘의 적합성을 보여준다.

Abstract The continuous data such as video streams and voice analog signals can be modeled as multidimensional data sequences (MDS's) in the feature space. In this paper, we investigate the clustering technique for multidimensional data sequences. Each sequence is represented by a small number of hyper-rectangular clusters for subsequent storage and similarity search processing. We present a linear clustering algorithm that guarantees a predefined level of clustering quality, and show its effectiveness via experiments on various video data sets.

1. 서 론

클러스터링 문제는 고객 구분(customer segmentation), 판매 분석(sales analysis), 패턴 인식(pattern recognition), 유사성 검색(similarity search) 등의 데이타베이스 응용 분야에서 많이 연구되어 왔으며, *'다차원 공간의 주어진 데이타 점들에 대하여, 유사한 특성을 갖는 점들은 같은 클러스터에, 상이한 특성을 갖는 점들은 다른 클러스터에 군집화하는 프로세스'*로 정의된다. 이때, 어느 클러스터에도 속하지 않으며 공간 내의 다른 점들과는 상이한 특성을 갖는 점들을 주변점(outlier) 혹은 소음(noise)이라 한다. 본 논문에서는 차후의 클러스터의 저장과 유사성 검색을 위하여 다차원 데이타 시퀀스(multidimensional data sequence: MDS) 내에 존재하는 데이타 점들의 클러스터링에 대한 문제를 연구한다.

이전의 연구[1]에서, 우리는 n 차원 공간에서 k 개의 점들로 이루어진 MDS S를 다음과 같이 정의하였다.

$$S = S[1], S[2], ..., S[k]$$

여기에서, 각 벡터 요소 $S[j]$ ($1 \le j \le k$)는 n 개의 스칼라 항목으로 구성되어 있으며, 다음의 식으로 표시된다: $S[j] = (S^1[j], S^2[j], , S^n[j])$. 이러한 다차원 시퀀스의 대표적인 예로써 비디오 스트림을 들 수 있다. 비디오 스트림은 복수 개의 프레임이 시간 순으로 연속되어 있으며, 각 프레임은 RGB나 YCbCr 칼라 공간과 같은 특징 공간(feature space)에서 다차원 벡터로 표시될 수 있다. 따라서, 비디오 스트림은 다차원 공간에서 각 프레임을 하나의 점으로 사상시킴으로써 연속된 점들의 시퀀스로 모델링될 수 있다. 비디오 스트림을 MDS로 모델링함으로써, 비디오 스트림의 프레임들을 클러스터링하는 문제는 다차원 공간에서 MDS에 속해있는 점들을 클러스터링하는 문제로 변형될 수 있다.

데이타베이스 분야에서 다양한 클러스터링 기법들이 연구되어 왔지만, 다차원 시퀀스의 클러스터링은 기존의

· 이 논문은 2000년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2000-041-E00262)
† 비 회 원 : 한국과학기술원 정보및통신공학과
sllee@islab.kaist.ac.kr
†† 학생회원 : 한국과학기술원 전산학과
tong@islab.kaist.ac.kr
††† 종신회원 : 한국과학기술원 전산학과 교수
chungcw@islab.kaist.ac.kr
논문접수 : 2000년 11월 8일
심사완료 : 2001년 7월 14일

클러스터링과는 여러 측면에서 다르게 접근해야 한다. 첫째, 기존의 기법에서는 클러스터링의 대상이 되는 한 객체가 하나의 점으로 표현되어 하나의 클러스터링에 속하였으나, 다차원 시퀀스의 클러스터링에서는 한 객체(비디오 스트림)가 여러 개의 점들로 이루어져 있고, 따라서 여러 개의 클러스터에 나뉘어져 포함된다. 둘째, 클러스터의 형태에 있어서도 차이가 발생할 수 있다. 기존의 방법들은 클러스터가 추후 어떻게 사용될 것인가 보다는 클러스터 자체의 정량적인 특성(quantitative property)에 초점을 맞추는 경향이 있다. 즉, MSE (mean square error)와 같은 주어진 조건을 최적화하는 클러스터를 구하는 문제에 집중하고 있다. 그러므로 클러스터의 형태는 데이타 공간에서 점들의 분포에 따라 대개 임의의 모습을 띠게 된다. 반면에, 본 연구에서는 클러스터링이 *주어진 뉴스 비디오와 유사한 비디오 스트림을 데이타베이스에서 찾으시오* 와 같은 유사성 검색을 위한 사전 단계로써 수행된다. 이것은 생성된 클러스터의 형태가 저장과 검색을 위한 인덱싱 메커니즘에 적합해야 한다는 것을 의미한다. 현재 널리 사용되고 있고 효율성이 검증된 R-트리[2]나 R-트리의 변형된 형태인 인덱스 구조들[3,4,5]이 노드 형태로써 MBR(minimum bounding rectangle)을 채택하고 있고, 따라서 이러한 인덱스 구조들을 수정 없이 사용할 수 있게 하기 위하여, 본 연구에서는 클러스터의 형태를 하이퍼 사각형으로 제한한다.

다차원 시퀀스의 유사성 검색을 효과적으로 지원하기 위하여 클러스터링 기법은 여러 가지의 요구 사항을 만족해야 하며, 이러한 요구 사항으로써 우선 클러스터의 기하학적인 특성을 고려할 수 있다. 즉, 클러스터는 효율적인 검색을 위하여 클러스터의 볼륨(volume)와 에지(edge)에 대하여 밀도(density)가 높아야 한다. 이것은 (1) 한 점 당 클러스터의 볼륨(volume per point: VPP)을 최소화하고, (2) 한 점 당 클러스터의 에지의 길이(edge per point: EPP)를 최소화하며, (3) 클러스터 당 점의 개수(number of points per cluster: PPC)를 최대화함으로써 달성될 수 있다. 밀도가 높은 클러스터는 밀도가 낮은 클러스터보다 상대적으로 작은 공간을 차지하게 되며, 검색 공간에서 클러스터들을 인덱스하여 검색할 때, 질의 영역(query region)과 겹칠 확률이 상대적으로 낮아지게 된다. 따라서 밀도가 높은 클러스터를 생성하는 알고리즘은 검색 효율을 높일 수 있다. 다음으로, 클러스터링 기법은 (4) 주변점들을 적절히 처리할 수 있어야 하고, (5) 알고리즘의 입력 인수의 개수를 최소화해야 한다. 이러한 요구 사항들을 고려하여 본 논문에서의 클러스터링 문제는 다음과 같이 정형화될 수 있다:

Given: 다차원 시퀀스 데이타 세트, 클러스터당 최소의 점 수 *minPts*

Goal: 사전에 정의된 측정 요건을 최적화하는 클러스터의 집합 및 주변점들의 집합을 구함

입력 인수 *minPts*는 주변점을 결정하기 위하여 사용된다. 4.3 절에 이를 위한 자세한 사항이 기술된다. 제안된 기법은 저차원 (3 혹은 5차원) 시퀀스뿐만 아니라 고차원 (50 혹은 70차원) 시퀀스에서도 실행될 수 있도록 고안되었다. 그러나 실제 환경에서의 적용에는 제한이 있다. 일반적으로 고차원 데이타의 처리는 심각한 프로세스 오버헤드를 야기하므로 현실적으로 사용되기 어렵다. 이 문제는 *'dimensionality curse problem'*으로 알려져 있으며, 이 문제를 피하기 위해 고차원의 데이타는 저차원으로의 차원 축소(dimensionality reduction) 과정을 거친다. 이러한 의미에서 제안된 기법은 저차원에서 적합한 성능을 발휘한다.

제안된 기법이 클러스터의 볼륨과 에지에 대하여 높은 밀도를 가진 클러스터를 생성하고 주변점들을 적절히 식별해 낼 수 있다는 장점 외에, 또 하나의 중요한 특성은 클러스터링에 필요한 대부분의 인수가 사용자에 의해 외부 입력 변수로써 제공되지 않고 다차원 시퀀스 자체의 정보를 사용하여 결정된다는 점이다. 일반적으로, 생성할 클러스터의 수, 클러스터간의 최소 거리 등의 클러스터링 인수를 사전에 결정하기 위한 도메인 지식을 얻기란 쉽지 않으며, 따라서 클러스터링 인수를 결정하는 것은 복잡한 문제이고 주로 경험상(heuristic)의 수치로 결정된다. 이러한 복잡성을 피하기 위하여 알고리즘의 입력 인수를 최소한으로 제한할 필요가 있으며, 제안된 기법은 하나의 입력 인수(*minPts*) 만을 필요로 한다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구와 함께 데이타 시퀀스의 클러스터링에 대하여 간단하게 언급한다. 3 장에서는 클러스터링에 관련된 기본 정의와 클러스터링의 특성 및 클러스터링의 품질을 측정하기 위한 여러 가지 기준에 관하여 논의한다. 클러스터링 과정에서 필요한 조건의 정의와 알고리즘 및 알고리즘의 복잡도가 4 장에서 기술되고, 5 장에서는 비디오 스트림을 사용한 실험 결과 및 분석을 제공하며, 6 장에서 결론을 맺는다.

2. 관련 연구

다차원 공간에서 데이타 포인트들을 클러스터링하기 위하여 CLARANS[6], BIRCH[7], DBSCAN[8], *CLIQUE*[9] 및 *CURE*[10] 등과 같은 다양한 기법들이 제

안되었다. CLARANS는 임의 검색(randomized search)에 기반하고 있고, 두 개의 사용자 입력 인수를 사용하여 검색 공간을 줄임으로써 효율성을 높이도록 설계한 클러스터링 기법이다. 알고리즘 BIRCH는 데이타베이스를 스캔하여 *CF-tree*라 불리는 계층 자료 구조를 생성하는 다단계 클러스터링 기법이며, *CF-tree*의 단말 노드를 클러스터링하기 위하여 임의의 클러스터링 알고리즘을 사용한다. 이 기법은 데이타베이스 분야에서 주변점을 효과적으로 처리한 최초의 기법이다. DBSCAN에서는 입력 인수를 결정하기 위해 필요한 도메인 지식을 최소화하도록 시도하였으며, 데이타 점의 분포에 따라 임의의 모양의 클러스터를 생성한다. 기본 아이디어는 클러스터내의 각 점에 대하여 주어진 반경 안에 정해진 최소 수의 점들을 포함하게 하는 것이다. 따라서 이 기법은 두 개의 입력 인수 (반경과 최소 점 수) 만을 필요로 한다. CLIQUE는 주어진 고차원 데이타 공간의 서브 공간에서 밀도가 높은 클러스터를 자동으로 식별할 수 있는 기법이다. 즉, 주어진 공간에서는 클러스터가 발견되지 않아도 그 서브 공간에서는 클러스터가 존재할 수 있으며 이 경우에 적합한 기법이다. 입력 인수로써 공간을 나누는 그리드(grid)의 크기와 클러스터를 위한 전역 밀도 한계값을 필요로 한다. 부속 공간에서의 클러스터링에 관한 아이디어는 [11]에서 사상된 클러스터링(projected clustering)의 개념으로 확장되었으며, 이것은 데이타 점이 밀접하게 연관되어 있는 특정 부속 차원을 선택하고 이 차원에 대한 클러스터를 찾아내는 기법이다.

클러스터링에 관한 최근의 연구로써 CURE는 구형이 아니면서 다양한 크기를 갖는 클러스터를 생성한다. 이 기법에서는 각 클러스터를 잘 산재하여 분포하고 있는 복수 개의 포인트들로 나타낸다. 구형이 아닌 클러스터는 여러 개의 포인트를 사용하면 그 모양을 보다 더 잘 표현할 수 있다. 이 알고리즘은 생성된 클러스터의 개수가 입력 인수로 주어진 값에 도달하게 되면 종료하게 된다. 그러나 위에 언급된 기법들은 차후의 검색을 염두에 두지 않고 설계되었고, 여러 개의 입력 인수를 필요로 하며, 하나의 시퀀스가 여러 개의 클러스터에 나누어지는 데이타 시퀀스의 클러스터링에는 적용하기가 어렵다.

데이타 시퀀스에 관한 최초의 알고리즘은 [12]에서 제안되었으며, 이 기법은 한 시퀀스를 형태가 하이퍼 사각형인 복수 개의 서브 시퀀스로 군집화한다. 이 알고리즘은 [13]에서 비디오 샷 경계를 검출하기 위해 알고리즘의 수행이 전방향과 후방향의 두 방향으로 이루어지도록 약간 수정되었으며, 또한 다차원의 하이퍼 사각형 질의를 지원하기 위하여 [1]에서 약간 수정되었다. 이 알고리즘들은 평균 디스크 접근 수를 클러스터 내의 점수로 나눈 값인 한계 비용(marginal cost: MCOST)에 근거하여 클러스터링을 수행한다. 즉, MCOST를 결정하기 위하여 이 기법들은 한 점이 클러스터에 포함될 때 포함되는 점으로 인한 클러스터의 볼륨의 증가를 중요한 클러스터링 요소로 고려한다. 그러나 볼륨 요소만 고려하는 것은 부족하며, 클러스터의 에지의 길이도 고려되어야 한다. 이 문제는 몇 가지의 예와 함께 다음 절에서 다루어진다.

3. 다차원 시퀀스의 클러스터링 특성

3.1 클러스터의 표현

이 절에서는 클러스터를 표현하기 위해 사용되는 하이퍼 사각형의 여러 가지 특성들에 대하여 논의한다. 클러스터는 1절에서 언급한 대로 하이퍼 사각형 형태로 표현되며, 다음과 같이 정의된다.

정의 1 (클러스터 *CL*): 단위 공간 $[0,1]^n$에서 k 개의 점 P_t (t = 1, 2, ..., k)를 포함하고 있는 하이퍼 사각형 형태의 클러스터 CL은 두 개의 끝점 L(low point)과 H(high point) 및 클러스터 안의 점의 개수로 정의되며 다음과 같이 나타낸다: $CL = \langle L, H, k \rangle$. 여기에서 $L = \{(L^1, L^2, ..., L^n) \mid L^i = min_{1 \le t \le k}(P_t^i)\}$이며, $H = \{(H^1, H^2, ..., H^n) \mid H^i = max_{1 \le t \le k}(P_t^i)\}$이다. 단, i = 1, 2, ..., n. ■

점 P_t를 L=H=P_t와 k=1로 대치함으로써 클러스터 형식으로 표현할 수 있다. 즉, P_t, P_t, 1로 나타낼 수 있으며 $CL(P_t)$로 표기한다. 클러스터 CL의 하이퍼 볼륨 $Vol(CL)$과 에지 $Edge(CL)$는 다음과 같이 계산된다.

$$Vol(CL) = \prod_{1 \le i \le n}(H^i - L^i)$$

$$Edge(CL) = 2^{n-1} \cdot \sum_{1 \le i \le n}(H^i - L^i)$$

따라서, 한 점 당 클러스터 CL의 볼륨 $VPP(CL)$과 에지 $EPP(CL)$는 각각 다음과 같이 나타낼 수 있다.

$$VPP(CL) = \frac{Vol(CL)}{k} = \frac{\prod_{1 \le i \le n}(H^i - L^i)}{k}$$

$$EPP(CL) = \frac{Edge(CL)}{k} = \frac{2^{n-1} \cdot \sum_{1 \le i \le n}(H^i - L^i)}{k} \quad (1)$$

클러스터링의 진행 과정은 두 개의 클러스터를 계속적으로 병합함으로써 진행된다. 병합을 표현하기 위한 병합 연산자(merging operator)를 다음과 같이 정의한다.

정의 2 (병합 연산자 ⊕): CL_1과 CL_2가 클러스터이고, $CL_1 = \langle L_1, H_1, k_1 \rangle$, $CL_2 = \langle L_2, H_2, k_2 \rangle$일 때, 병합 연산자 ⊕는 다음과 같이 정의된다: $CL_1 \oplus CL_2 = CL_3$, 여기에서, $CL_3 = \langle L_3, H_3, k_3 \rangle$, $L_3 = \{(L_3^1,$

L_3^2, ..., L_3^n) | $L_3^i = min(L_1^i, L_2^i)$}, $H_3 = \{(H_3^1, H_3^2, ..., H_3^n) | H_3^i = max(H_1^i, H_2^i)\}$ (i = 1, 2, ..., n)이고, $k_3 = k_1 + k_2$. ■

단위 공간 $[0,1]^n$에서 클러스터 $CL_t = <L_t, H_t, k_t>$가 클러스터 $CL_s = <L_s, H_s, k_s>$에 병합되는 경우를 고려해 보자. 이때, CL_t와 CL_s는 클러스터가 될 수도 있고 한 점이 될 수도 있다. 일반적으로 두 개의 클러스터를 병합하면 볼륨과 에지가 큰 클러스터가 생성되며, 점의 수도 두 개의 클러스터의 점의 수를 합한 수가 된다. 이 병합 과정에서 발생하는 변화의 양이 클러스터링의 품질을 결정하는 중요한 요소가 된다. (이것은 이후의 절에서 자세히 언급된다.) 클러스터 CL_t가 클러스터 CL_s에 병합되는 경우의 볼륨과 에지의 증가량은 다음의 식으로 나타낼 수 있다.

$\Delta Vol(CL_s, CL_t) = Vol(CL_s \oplus CL_t) - Vol(CL_s)$

$\Delta Edge(CL_s, CL_t) = Edge(CL_s \oplus CL_t) - Edge(CL_s)$

위의 식을 사용하여 한 점 당 볼륨과 에지의 평균 증가율, ΔVPP 및 ΔEPP를 각각 구하면 다음의 식으로 표현된다.

$$\Delta VPP(CL_s, CL_t) = \frac{\Delta Vol(CL_s, CL_t)}{k_t} = \frac{Vol(CL_s \oplus CL_t) - Vol(CL_s)}{k_t}$$

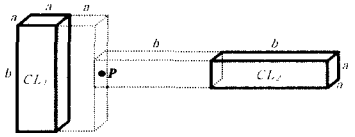
$$\Delta EPP(CL_s, CL_t) = \frac{\Delta Edge(CL_s, CL_t)}{k_t} = \frac{Edge(CL_s \oplus CL_t) - Edge(CL_s)}{k_t}$$

위의 식에서 가율의 함수, ΔVPP 및 ΔEPP는 비대칭적(non-symmetric)인 함수임을 알 수 있다. 즉, $\Delta VPP(CL_s, CL_t) \neq \Delta VPP(CL_t, CL_s)$, $\Delta EPP(CL_s, CL_t) \neq \Delta EPP(CL_t, CL_s)$이다. 위의 두 증가율은 제안한 알고리즘에서 한 클러스터가 병합될 대상 클러스터를 찾을 때 중요한 기준으로 사용된다.

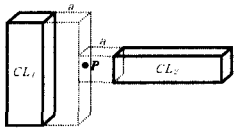
3.2 클러스터링 요소

시퀀스 데이타를 위한 클러스터링 알고리즘은 [12]에서 최초로 제안되었으며, 시퀀스를 각각이 MBR 형태를 갖는 여러 개의 부속 시퀀스로 분할한다. 이 알고리즘은 한계 비용(marginal cost: MCOST)을 이용하여 클러스터링을 실행하는 데, 이 한계 비용은 평균 디스크 접근 수를 클러스터의 점들의 수로 나눈 값으로 정의된다. 즉, 클러스터를 생성하기 위한 점들의 군집화는 다음과 같은 방식으로 진행된다: 시퀀스의 첫 번째 점을 한 점을 갖는 클러스터로 나타내고 연속되는 각각의 점들을 클러스터에 계속 포함시킨다. 특정 점을 현재의 클러스터에 포함시킬 때 MCOST 값이 증가하지 않으면 그 클러스터에 점을 포함시키고, 증가하면 그 점으로부터 시작하여 새로운 클러스터를 생성한다. MCOST 값의 판정 기준은 특정 점이 현재의 클러스터에 포함될 때 증가하는 볼륨 요소를 고려하여 결정된다. 그러나 많은 경우에 볼륨 요소만 고려하는 것은 충분치 못하다. 우리의 주장은 볼륨 요소 뿐만 아니라 에지 요소도 중요하게 고려되어야 한다는 것이다. 다음의 두 예는 이러한 주장을 뒷받침해 준다.

예 1. 클러스터 CL_1과 CL_2가 그림 1에 도시한 것처럼 3차원 공간에서 변의 길이가 각각 a, a, b ($a<b$)인 직육면체 형태이고, 점 P가 이 두 클러스터 중 하나에 병합되는 경우를 고려해 보자. 어느 클러스터에 병합되는 것이 유리한가를 결정하는 문제이다. 그림 1-(a)에서, $\Delta VPP(CL_1, CL(P)) = \Delta VPP(CL_2, CL(P)) = a^2 \cdot b$, $\Delta EPP(CL_1, CL(P)) = 4 \cdot a$, $\Delta EPP(CL_2, CL(P)) = 4 \cdot b$임을 알 수 있다. 클러스터링 요소로써 볼륨만을 고려한다면 두 클러스터에 대하여 점 P가 병합될 때 증가하는 볼륨이 같기 때문에 CL_1과 CL_2 모두 후보가 될 수 있다. 한편, 그림 1-(b)에서 $\Delta EPP(CL_1, CL(P)) = \Delta EPP(CL_2, CL(P)) = 4 \cdot a$, $\Delta VPP(CL_1, CL(P)) = a^2 \cdot b$, $\Delta VPP(CL_2, CL(P)) = a^3$가 되므로 에지만을 클러스터링 인수로써 고려한다면, 또한 CL_1과 CL_2 모두 후보가 될 수 있다. 그러나, 직관적으로 관찰할 때 $a < b$이므로, 병합 대상 클러스터로써 전자의 경우에는 CL_1이 적당하고 후자의 경우에는 CL_2가 적당함을 알 수 있다. 이것은 시퀀스를 클러스터링할 때 클러스터링 요소로써 볼륨과 에지 모두 중요하다는 것을 의미한다. ■



(a) 같은 볼륨, 다른 에지의 경우



(b) 다른 볼륨, 같은 에지의 경우

그림 1 클러스터링 요소 : 볼륨과 에지

예 2. 그림 2에 도시한 것처럼 3차원 공간에서 점 P가 두 클러스터 CL_1과 CL_2 중 하나에 병합되는 경우, 어느 클러스터에 병합되는 것이 유리한가를 고려해 보자. 클러스터링 요소로써 볼륨만을 고려한다면 $\Delta VPP(CL_1, CL(P)) = 0$, $\Delta VPP(CL_2, CL(P)) > 0$ 이

므로 CL_1이 더 적당할 것이다. 그러나 볼륨은 0일지라도 넓은 면적에 펼쳐져 있는 클러스터(CL_1)는 작은 직육면체(CL_2)의 클러스터보다 영역 질의에서 겹칠 확률이 높다. 그러므로 점 P가 클러스터 CL_2에 병합되는 것이 보다 나은 클러스터링이 될 것이다. 이러한 경우는 고차원에서 클러스터의 에지가 어느 한 차원에서 0이 되는 경우에 자주 발생한다. ■

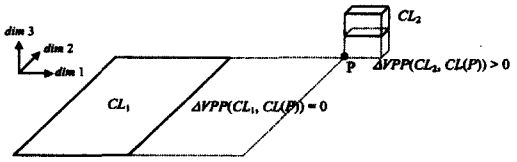


그림 2 볼륨이 0인 넓은 클러스터 vs. 볼륨이 작은 직육면체 형태의 클러스터

3.3 클러스터링의 품질 측정 기준

1절에서 논의한 클러스터링 요구 사항에 적합한 클러스터를 생성하기 위하여 적절한 클러스터의 품질 측정 기준을 사용할 필요가 있다. 이를 위하여 클러스터의 정량적인 특성을 나타내는 VPP, EPP, PPC의 3가지 평가 기준을 소개한다. k개의 점을 포함하고 있는 다차원 시퀀스 S에 대하여 클러스터링한 결과 p개의 클러스터를 생성했다고 하면, 이때 시퀀스 S에 대한 VPP, EPP, PPC는 다음과 같이 정의된다:

$$VPP = \frac{\sum_{1\le j\le p} Vol(CL_j)}{\sum_{1\le j\le p} k_j},\quad EPP = \frac{\sum_{1\le j\le p} Edge(CL_j)}{\sum_{1\le j\le p} k_j},$$

$$PPC = \frac{\sum_{1\le j\le p} k_j}{p} \qquad (3)$$

4. 클러스터링 알고리즘

주어진 MDS의 집합과 클러스터 당 최소 점의 수 $minPts$에 대하여, 제안한 알고리즘은 식 (3)에서 정의된 클러스터링 측정 기준을 최적화하는 클러스터의 집합 C와 주변점들의 집합 O를 찾으려고 시도한다. 클러스터링 과정 중, 두 개의 클러스터의 병합 여부를 결정하기 위하여 제안한 알고리즘은 특별히 정의된 조건을 사용한다. 즉, 두 클러스터가 이 조건을 만족시키는 경우에만 병합이 허용된다. 이 절에서는 이러한 조건과 제안한 알고리즘에 대하여 논의한다.

4.1 τ -바운딩 조건

한 점만을 포함하고 있는 클러스터가 다른 클러스터에 병합되는 경우, 이 병합을 허용할 지의 여부를 결정하기 위하여, 클러스터의 볼륨과 에지에 대하여 바운딩 조건을 사용한다. 이 조건을 자세하게 논의하기 전에 먼저 단위 하이퍼 큐브(unit hyper-cube)의 개념에 대하여 다음과 같이 정의한다.

정의 3 (단위 하이퍼 큐브 *uCUBE*): MDS S의 모든 k개의 점들을 내포(tightly bound)하는 최소의 하이퍼 사각형을 갖는 클러스터를 CL_{mds}라고 하자. 이때, 단위 하이퍼 큐브 $uCUBE$는 MDS S의 모든 k개의 점들이 CL_{mds}의 하이퍼 사각형 내에서 균일하게 분포된다고 가정했을 때, 한 점이 차지하는 큐브로써 정의된다. 이 큐브의 한 변의 길이가 e라 하면 볼륨과 에지의 길이는 다음의 식으로 표현된다:

$$Vol(uCUBE) = e^n = \frac{Vol(CL_{mds})}{k}$$

$$Edge(uCUBE) = 2^{n-1}\cdot n\cdot e = 2^{n-1}\cdot n\cdot \sqrt[n]{\frac{Vol(CL_{mds})}{k}} \quad ■$$

MDS S의 모든 점들이 CL_{mds}의 하이퍼 사각형 내에서 균일하게 분포한다면, 한 점이 하나의 단위 하이퍼 큐브에 할당된다고 생각할 수 있다. 즉, 직관적으로 MDS S의 각 점이 형태가 단위 하이퍼 큐브인 하나의 클러스터를 형성한다고 볼 수 있다. 그러나 여기에서 가정한 균일 분포는 일반적으로 실세계에서는 거의 존재하지 않는다. 예를 들면, 동일한 비디오 샷 내에 있는 프레임들은 상당한 유사성을 보이고 있으며, 따라서 각 프레임을 한 점으로 나타냈을 때 이 점들은 군집화되는 특성을 보인다. 이 균일 분포가 두 클러스터의 병합을 허용할 지의 여부를 결정하는 바운딩 조건을 제공한다. MDS S를 클러스터링할 때 볼륨과 에지에 대한 바운딩 한계 값(bounding threshold) - τ_{vol}과 τ_{edge} - 은 다음의 식으로 표현한다.

$$\tau_{vol} = Vol(uCUBE) = e^n,$$

$$\tau_{edge} = Edge(uCUBE) = 2^{n-1}\cdot n\cdot e$$

정의 4 (τ -바운딩 조건): n차원 공간에서 한 점만을 포함하고 있는 클러스터 CL_t가 다른 클러스터 CL_s에 병합되는 경우, τ -바운딩 조건은 두 클러스터가 병합되기 위하여 만족되어야 하는 조건이며 다음의 식으로 정의된다.

$$\Delta VPP(CL_s, CL_t) \le \tau_{vol} \wedge \Delta EPP(CL_s, CL_t) \le \tau_{edge} \quad (4)$$

Lemma 1. *τ -바운딩 조건을 만족하는 클러스터링은 VPP와 EPP에 대하여 균일 분포의 경우 보다 항상 나은 클러스터링 품질을 보장한다.*

증명: k개의 점을 포함하고 있는 MDS S에 대하여

클러스터링한 결과 $p(0 \le p \le k)$개의 클러스터가 생성되었다고 가정하고, 이때 생성된 클러스터의 집합을 $C = \{CL_1, ..., CL_p\}$라 하자. 클러스터 CL_j $(1 \le j \le p)$가 하나의 점으로부터 출발하여 u번의 병합 과정을 거쳐 생성되었다고 가정하고, l번째 병합으로 인한 볼륨, 에지, 점의 수의 증가량을 각각 $\Delta Vol_{j,l}$, $\Delta Edge_{j,l}$, $k_{j,l}$라 하자. 그러면, 클러스터링 시작 시점에서는 $k_j = 1$, $Vol(CL_j) = 0$, $Edge(CL_j) = 0$이므로 u번의 병합 과정 후에는 다음의 결과를 얻는다: $k_j = 1 + \sum_{1 \le l \le u} k_{j,l}$, $Vol(CL_j) = \sum_{1 \le l \le u} \Delta Vol_{j,l}$, $Edge(CL_j) = \sum_{1 \le l \le u} \Delta Edge_{j,l}$. 각 병합 단계에서 식 (4)의 조건이 성립하므로 다음 관계가 성립한다: $\Delta VPP_{j,l} \le e^n$과 $\Delta EPP_{j,l} \le 2^{n-1} \cdot n \cdot e$, 즉, $\Delta Vol_{j,l} \le k_{j,l} \cdot e^n$과 $\Delta Edge_{j,l} \le k_{j,l} \cdot 2^{n-1} \cdot n \cdot e$. 따라서, 다음의 식을 유도할 수 있다.

$$\begin{aligned} Vol(CL_j) &= \sum_{1 \le l \le u} \Delta Vol_{j,l} \le \sum_{1 \le l \le u} (k_{j,l} \cdot e^n) \\ &< (1 + \sum_{1 \le l \le u} k_{j,l}) \cdot e^n \\ &= k_j \cdot e^n \\ Edge(CL_j) &= \sum_{1 \le l \le u} \Delta Edge_{j,l} \le \sum_{1 \le l \le u} (k_{j,l} \cdot 2^{n-1} \cdot n \cdot e) \\ &< (1 + \sum_{1 \le l \le u} k_{j,l}) \cdot 2^{n-1} \cdot n \cdot e \\ &= k_j \cdot 2^{n-1} \cdot n \cdot e \end{aligned} \quad (5)$$

MDS S의 클러스터링 결과 한 점당 볼륨과 에지를 각각 VPP, EPP라 하고, 균일 분포의 경우를 각각 VPP_0, EPP_0라 하면, 식 (3), (5)에 의하여 다음 결과를 얻는다.

$$VPP = \frac{\sum_{1 \le j \le p} Vol(CL_j)}{\sum_{1 \le j \le p} k_j} = \frac{Vol(CL_1) + ... + Vol(CL_p)}{k_1 + ... + k_p}$$
$$< \frac{k_1 \cdot e^n + ... + k_p \cdot e^n}{k_1 + ... + k_p} = e^n = VPP_0$$

$$EPP = \frac{\sum_{1 \le j \le p} Edge(CL_j)}{\sum_{1 \le j \le p} k_j} = \frac{Edge(CL_1) + ... + Edge(CL_p)}{k_1 + ... + k_p}$$
$$< \frac{k_1 \cdot 2^{n-1} \cdot n \cdot e + ... + k_p \cdot 2^{n-1} \cdot n \cdot e}{k_1 + ... + k_p} = 2^{n-1} \cdot n \cdot e = EPP_0$$

그러므로, Lemma 1이 성립한다. ■

4.2 θ-병합 조건

MDS S에 포함되어 있는 점들의 집합으로부터 초기 클러스터가 생성된 후에는 공간적으로 근접한 (즉, 유사한 특성을 가진) 클러스터들은 다시 병합될 필요가 있다. 생성된 클러스터 간의 병합을 허용할 지의 여부를 판정하기 위한 기준으로써, θ-병합 조건을 다음과 같이 정의한다.

정의 5 (θ-병합 조건): 두 클러스터 CL_s와 CL_t가 n차원 공간에서 서로 병합되는 경우, $\theta_{vol} = Vol(CL_s) + Vol(CL_t)$, $\theta_{edge} = Edge(CL_s) + Edge(CL_t)$라 할 때, θ-병합 조건은 두 클러스터를 병합하기 위하여 만족되어야 하는 조건을 나타내고, 다음과 같이 정의된다:

$$Vol(CL_s \oplus CL_t) \le \theta_{vol} \wedge Edge(CL_s \oplus CL_t) \le \theta_{edge} \quad ■ \quad (6)$$

Lemma 2. *θ-병합 조건을 만족시키는 두 클러스터 CL_s와 CL_t의 병합은 다음의 두 경우 보다 항상 나은 클러스터링 품질을 보장한다.*

(i) VPP와 EPP에 대하여 균일 분포의 경우

(ii) VPP, EPP와 PPC에 대하여 병합하기 전의 경우

증명: $CL_m = CL_s \oplus CL_t$, $k_m = k_s + k_t$이라 하고, VPP_m, VPP_s, VPP_t를 각각 CL_m, CL_s, CL_t의 VPP, EPP_m, EPP_s, EPP_t를 각각 CL_m, CL_s, CL_t의 EPP라 하자. CL_s와 CL_t의 병합이 θ-병합 조건을 만족한다면, 식 (1)와 (6)에 의하여 $k_m \cdot VPP_m \le k_s \cdot VPP_s + k_t \cdot VPP_t$와 $k_m \cdot EPP_m \le k_s \cdot EPP_s + k_t \cdot EPP_t$이 성립함을 알 수 있다.

*(i)*의 경우: 식 (5)에 의하여 $VPP_s < e^n$, $VPP_t < e^n$, $EPP_s < 2^{n-1} \cdot n \cdot e$, $EPP_t < 2^{n-1} \cdot n \cdot e$가 되므로 다음의 결과를 얻을 수 있다:

$$VPP_m \le \frac{k_s \cdot VPP_s + k_t \cdot VPP_t}{k_m} < \frac{k_s \cdot e^n + k_t \cdot e^n}{k_m} = e^n$$

$$EPP_m \le \frac{k_s \cdot EPP_s + k_t \cdot EPP_t}{k_m}$$
$$< \frac{k_s \cdot 2^{n-1} \cdot n \cdot e + k_t \cdot 2^{n-1} \cdot n \cdot e}{k_m} = 2^{n-1} \cdot n \cdot e$$

*(ii)*의 경우: 두 클러스터 CL_s와 CL_t를 병합하기 전의 VPP, EPP를 각각 VPP_n, EPP_n이라 하면, 식 (3)와 (6)에 의하여 다음의 결과를 얻을 수 있다:

$$VPP_m = \frac{Vol(CL_m)}{k_m} = \frac{Vol(CL_s \oplus CL_t)}{k_s + k_t}$$
$$\le \frac{Vol(CL_s) + Vol(CL_t)}{k_s + k_t} = VPP_n$$

$$EPP_m = \frac{Edge(CL_m)}{k_m} = \frac{Edge(CL_s \oplus CL_t)}{k_s + k_t}$$
$$\le \frac{Edge(CL_s) + Edge(CL_t)}{k_s + k_t} = EPP_n$$

PPC에 관해서는, 두 클러스터를 병합하기 전의 PPC가 $(k_s + k_t) / 2$인데 반해 병합 후에는 $k_m = k_s + k_t$가 되므로 두 클러스터의 병합이 PPC에 대하여 더 나은 클러스터를 생성함은 명백하다. 따라서, Lemma 2가 성립한다.

4.3 주변점의 걸정

다차원 시퀀스의 클러스터링에서 전체적인 클러스터링 패턴에 대해 중요하지 않은 주변점들이 있을 수 있다. 제안된 알고리즘에서 클러스터링 시작 시점에서는 시퀀스

내의 각 점들이 하나의 점 만을 갖는 클러스터로 간주되며, 클러스터링이 진행됨에 따라 서로 관련이 있는 유사한 클러스터들이 반복적으로 병합된다. 만일 클러스터링 과정이 종료된 후에 생성된 클러스터 중에서 극히 소수의 점만을 갖는 클러스터가 있다면 그 클러스터에 있는 점들은 주변점으로 간주된다. 예를 들어, 클러스터링 후에 두세 개의 점만을 갖는 클러스터가 다른 클러스터와 떨어진 위치에 존재한다면 그 클러스터는 주변점을 포함하고 있을 확률이 높게 된다. 특정 클러스터가 주변점을 포함하고 있는 클러스터인 지를 판정하기 위하여 과연 몇 개의 점(*minPts*)을 기준으로 할 것인가는 응용 분야에 따라 경험적으로 결정될 수 있다. *minPts* 값이 너무 작으면 중요하지 않은 클러스터가 인덱스에 포함될 수 있으며, 이는 메모리의 효율을 저하시키는 결과를 초래한다. 반면에 너무 크게 되면 의미 있는 정보를 상실하게 된다. 이러한 맥락에서 클러스터링 후에 생성된 클러스터가 주어진 *minPts* 값보다 적은 수의 점을 포함하고 있으면, 그 클러스터에 있는 모든 점들은 주변점으로 간주된다. 이러한 주변점들은 인덱스에 포함되지 않고 차후의 처리를 위해 디스크에 파일로써 수록된다.

4.4. 클러스터링 알고리즘

그림 3의 알고리즘 **CDMDS**(Clustering a Data set of MDS)는 제안된 알고리즘의 전체적인 구조를 나타낸다. 주어진 MDS의 집합과 클러스터 당 최소 점의 수 *minPts*에 대하여 이 알고리즘은 식 (3)에서 정의된 클러스터링 측정 기준을 최적화하는 클러스터의 집합 C와 주변점들의 집합 O를 발견하는 알고리즘이다.

그림 4의 알고리즘 **ChooseCluster**는 주어진 점이 어느 클러스터에 병합될 지를 결정하는 알고리즘이다. 먼저, 각 클러스터가 τ-바운딩 조건을 만족시키는 지를 검사한다. 이 조건을 만족시키는 클러스터는 일단 병합 대상 후보가 되며, 이 후보 클러스터들 중에 가장 적합한 클러스터를 선택한다. 이 선택에 사용될 기준 $join$을 볼륨과 에지에 대한 가중치 w_1과 w_2를 고려하여 다음과 같이 정의한다.

Algorithm CDMDS
Input: a data set of *MDS's*, *minPts*
Output: cluster set C, outlier set O
Step 0: set $C \leftarrow \phi$, set$O \leftarrow \phi$
Step 1: **for each** *MDS* S_i in the data set$(1 \le i \le N)$
C_i, O_i **ClusterMDS**(*MDS* S_i, *minPts*)
$C \leftarrow C \cup C_i$, $O \leftarrow O \cup O_i$
end for
Step 2: **return** set C, set O

그림 3 알고리즘의 전체적인 구조

Algorithm ChooseCluster
Input: point P_m, set C_i, vol, and $edge$
Output: ϕ or cluster CL_c
Step 0: $result \leftarrow \phi$, $\lambda_{current} \leftarrow \infty$, $vol_{current} \leftarrow \infty$
Step 1: /* Choose a cluster to merge with */
if set C_i == ϕ **then**
return *result*
for each cluster CL_c in set C_i
compute $\Delta VPP(CL_c, CL(P_m))$, $\Delta EPP(CL_c, CL(P_m))$
if τ-*bounding condition holds* **then**
compute $\lambda_{join}(CL_c, CL(P_m))$
if ($\lambda_{join}(CL_c, CL(P_m)) < \lambda_{current}$)
$\vee$($\lambda_{join}(CL_c, CL(P_m)) == \lambda_{current}$
$\wedge Vol(CL_c \oplus CL(P_m)) < vol_{current}$) **then**
$\lambda_{current} \leftarrow \lambda_{join}(CL_c, CL(P_m))$
$vol_{current} \leftarrow Vol(CL_c \oplus CL(P_m))$
$result \leftarrow CL_c$
end if
end if
end for
Step 2: **return** *result*

그림 4 알고리즘 ChooseCluster

$$\lambda_{join}(CL_s, CL(P_t)) = \frac{w_1 \cdot \lambda_{vol} + w_2 \cdot \lambda_{edge}}{w_1 + w_2},$$

$$where\ \lambda_{vol} = \frac{\Delta VPP(CL_s, CL(P_t))}{Vol(CL_s \oplus CL(P_t))}\ and$$

$$\lambda_{edge} = \frac{\Delta EPP(CL_s, CL(P_t))}{Edge(CL_s \oplus CL(P_t))}$$

응용 분야의 특성에 따른 볼륨과 에지 인수의 중요성을 고려하여 가중치 w_1과 w_2를 결정한다. 5절의 실험에서는 동일한 가중치(즉, w_1=w_2=1)를 사용하였다. ΔVPP와 ΔEPP의 값의 차이를 보정하기 위하여 정규화된 값인 λ_{vol}과 λ_{edge}을 도입하였으며, 이는 큰 차이가 나는 두 값을 단순히 합하게 되면 작은 값은 거의 무시되는 예기치 않은 결과를 피하기 위함이다. 따라서, τ-바운딩 조건을 만족시키는 후보 클러스터들 중에 가장 작은 λ_{join} 값을 갖는 클러스터를 병합 대상 클러스터로 선택한다. 만일 같은 값을 가지는 클러스터가 복수 개 존재하면 $Vol(CL_s \oplus CL(P_t))$ 값이 가장 작은 클러스터 CL_s를 선택한다.

그림 5의 알고리즘**ClusterMDS**는 각 MDS의 클러스터링을 실행하는 알고리즘이다. Step 1에서 MDS 내의 점들로부터 초기의 클러스터 집합이 생성된다. 이때, 클러스터 간의 순서는 고려하지 않는다. 제안된 알고리즘의 목적이 다차원 시퀀스를 가능한 한 적은 수효의 클러스터로 나타내어 검색 효율을 증진시키는 것이기 때문이다. Step 2에서 각 클러스터 쌍에 대해 τ-병합 조건을 검사한다. 이 조건을 만족시키는 모든 클러스터 쌍은 병합된다. Step 3에서 클러스터링 후에 생성된 각

클러스터가 포함하고 있는 점의 수를 검사한다. 이 점의 수가 주어진 *minPts* 값보다 작은 클러스터의 경우, 4.3 절에서 언급한 대로 그 클러스터 내의 모든 점들은 주변점으로 간주되고, 주변점들의 집합으로 이동된다.

Algorithm ClusterMDS
Input: MDS S_i, *minPts*
Output: set C_i, set O_i
Step 0: set $M \leftarrow$ all points in S_i
set $C_i \leftarrow \phi$, set $O_i \leftarrow \phi$, $numCL \leftarrow 0$
compute τ_{vol} and τ_{edge} for S_i
Step 1: /* Initial cluster generation */
for each point P_m in set M
$CL_c \leftarrow$ **ChooseCluster**(P_m, C_i, $_{vol}$, $_{edge}$)
if $CL_c == \phi$ **then**
represent P_m in the cluster form CL_{numCL}
$C_i \leftarrow C_i \cup \{CL_{numCL}\}$
$numCL \leftarrow numCL + 1$
else
$CL_c \leftarrow CL_c \oplus CL(P_m)$
end if
end for
Step 2: /* Cluster elaboration */
for each pair of clusters (CL_x, CL_y) in C_i
if θ-*merging condition holds* **then**
$CL_x \leftarrow CL_x \oplus CL_y$
remove CL_y from set C_i
end if
end for
Step 3: /* Outlier determination */
for each cluster CL_c in C_i
if $k_c < minPts$ **then**
remove CL_c from C_i
insert all points of CL_c into O_i
end if
end for
Step 4: **return** set C_i, set O_i

그림 5 알고리즘ClusterMDS

4.5 알고리즘의 복잡도

그림 3의 전체 알고리즘 구조 **CDMDS**에서 알고리즘의 복잡도(complexity)는 데이타베이스에 저장된 다차원 시퀀스의 개수 N에 대하여 $O(N)$임을 명백히 알 수 있다. 다음에 각 시퀀스의 점의 개수 m과 클러스터의 개수 c에 대한 복잡도를 구해 보자. 알고리즘 **Cluster MDS**의 Step 1에서 복잡도는 mc가 됨을 알 수 있으며, Step 2에서는 c^2, Step 3에서는 c가 됨을 알 수 있다. 여기에서, c의 값은 m에 비해 무시될 수 있을 정도로 작고 알고리즘 내부에서 변하는 값이므로 N과 m만을 고려한다. 따라서, 결과적으로 알고리즘의 복잡도는 N과 m에 대하여 선형적으로 변하는 $O(Nm)$이 된다.

5. 실험 결과 및 고찰

제안된 알고리즘의 적합성(effectiveness)을 검증하기 위한 실험 데이타로 비디오 스트림을 사용하였다. TV 뉴스, 드라마 그리고 애니메이션 영화 등을 임의의 길이로 분할하여 비디오 스트림을 생성하고, 이 스트림으로부터 다차원 시퀀스를 구성하였다. 실험은 제안한 알고리즘이 생성하는 클러스터링의 품질이 사전에 정의된 측정 기준에 얼마나 부합하느냐에 초점을 맞추었다. 이 절에서는 실험의 진행 과정과 결과에 대한 분석을 기술한다.

5.1 실험 준비

비디오 데이타 스트림을 파싱(parsing)하여 각 프레임으로부터 RGB의 칼라 특징(feature)을 추출한다. 따라서 스트림 내의 각각의 프레임은 다차원 공간에서 한 점으로 나타내지게 된다. 결과적으로 비디오 스트림은 다차원 시퀀스로 사상되게 된다. 제안된 알고리즘 자체는 저차원 뿐만 아니라 고차원에서도 실행 가능하지만, 1절에서 언급한 대로 *'dimensionality curse problem'*을 피하기 위해 본 실험에서는 편의상 각 프레임을 RGB 칼라 속성을 갖는 3차원 벡터로 표현한다. 그림 6은 3025 프레임으로 구성된 뉴스 비디오 스트림으로부터 생성된 3차원 시퀀스를 단위 공간 $[0,1]^3$에서 표현한 예이다.

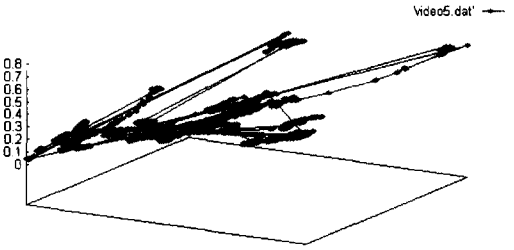


그림 6 TV 뉴스 비디오로부터 생성된 3차원 시퀀스

실험 데이타는 TV 뉴스, 드라마 그리고 애니메이션 영화 등의 비디오 데이타 세트로부터 임의의 길이의 비디오 스트림을 생성하였으며, 생성된 비디오 스트림의 길이와 개수는 표 1에 열거되어 있다.

표 1 테스트 비디오 데이타 세트

데이타세트 이름	시퀀스 길이	시퀀스 개수
v1	L ≤ 64	1,632
v2	64 < L ≤ 128	1,401
v3	128 < L ≤ 256	1,206
v4	256 < L ≤ 512	1,017
v5	512 < L ≤ 1024	912
v6	1024 < L ≤ 2048	813
v7	2048 < L ≤ 4096	651
v8	4096 < L	513
합 계		8,145

5.2 결과 고찰

본 실험은 VPP, EPP 및PPC에 대하여 제안한 알고리즘 CDMDS의 클러스터링 품질을 보여주는 것에 중점을 둔다. 제안한 알고리즘은 [12]에서 제안된 기법인 MCOST 알고리즘과 비교하였으며, 이것은 이후에 제안된 다른 알고리즘이 MCOST를 필요에 맞게 약간 변형했을 뿐이며, 근본적으로 비슷하기 때문이다. MCOST와는 달리, CDMDS는 MDS 내에서 클러스터의 순서를 고려하지 않고 클러스터를 생성한다. 이것은 이전에 언급한 대로 MDS를 가능한 한 적은 수의 클러스터로 표현하여 검색의 효율을 높이는 것이 목적이기 때문이다. 주변점을 판정하기 위하여 *minPts* 값을 5로 정하였으며, 비디오 응용 분야에서는 이 값이 적합하다고 판단된다. 그 이유는 5 프레임 길이를 갖는 비디오를 상연하는 데는 약 1/6초가 소요되며, 5 프레임 이하의 비디오는 인간 지각 능력의 한계 때문에 현실적으로 의미가 없기 때문이다. 실험 결과는 그림 7~10에 나타나 있다.

그림 7은 MCOST와 CDMDS 경우 모두 비디오 시퀀스의 길이가 증가함에 따라 VPP의 값이 감소됨을 보여 준다. 그러나 CDMDS의 VPP는 MCOST에 비하여 그래프에서 알 수 있듯이 11~23% 정도로 낮다. 그림 8에서 관찰할 수 있듯이 EPP 역시 CDMDS의 EPP는 MCOST에 비하여 38~59% 정도로 낮게 나타난다. 따라서, CDMDS 알고리즘에 의하여 생성된 클러스터들이 MCOST에 의하여 생성된 클러스터들 보다 볼륨과 에지에 대하여 밀도가 상당히 높음을 알 수 있고, 이는 차후의 검색의 효율을 높일 수 있는 근거가 된다.

클러스터링의 적합성을 나타내 주는 다른 척도로써, 그림 9에 CDMDS와 MCOST의 PPC 값을 보여 준다. CDMDS와 MCOST의 PPC 비율CDMDS/MCOST는 0.69-2.67로써, 시퀀스의 길이가 길어질수록 증가함을 알 수 있다. 이것은 CDMDS 알고리즘에 의하여 생성된 클러스터들이 MCOST에 의하여 생성된 클러스터들에 비해 시퀀스의 길이가 길어질수록 밀도가 점진적으로 높아짐을 의미한다. 일반적으로 긴 비디오 스트림은 비슷한 비디오 샷들을 포함할 확률이 높다. 예를 들어, 뉴스 비디오 스트림은 동일한 앵커가 출현하는 유사한 샷들을 여러 개 포함할 수 있으며, 따라서 클러스터당 점수는 증가하게 된다. 그림 10은 시퀀스 안에 포함되어 있는 전체 점의 수에 대한 주변점의 수의 비율을 나타낸다. 시퀀스의 길이가 증가함에 따라 6.28~1.29%로 점차 감소함을 알 수 있다. 이러한 주변점들은 차후의 검색에 대비하여 클러스터와 다르게 취급되며, 인덱스에 포함되지 않고 파일로써 디스크에 저장된다.

6. 결 론

본 논문에서는 비디오 스트림과 같은 다차원 시퀀스의 클러스터링에 대하여 연구하였다. 다차원 시퀀스의 클러

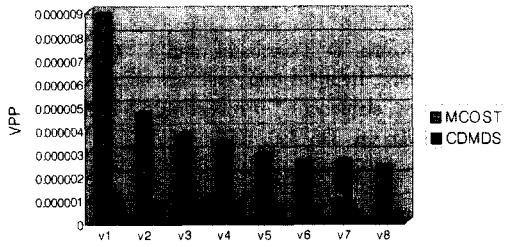


그림 7 VPP 값 비교

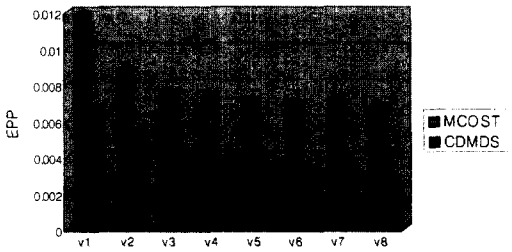


그림 8 EPP 값 비교

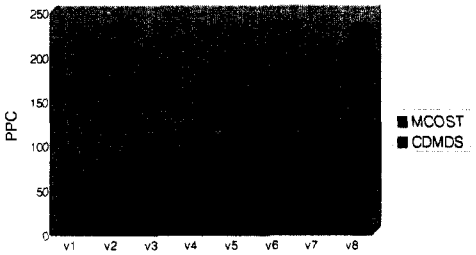


그림 9 PPC 값 비교

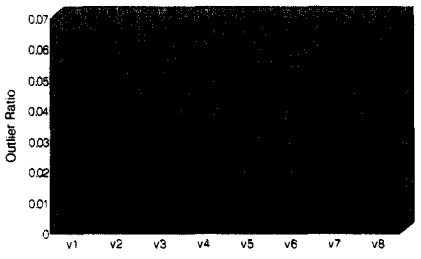


그림 10 MDS의 주변점 비율

스터링은 아직까지 널리 연구되지는 않았지만 데이타베이스 응용 분야에서 많은 잠재력이 있는 분야라고 할 수 있으며, 비디오 스트림과 같은 다차원 시퀀스의 표현, 저장 및 검색 등의 기초 단계로써 중요한 의미를 갖는다. 본 논문에서 제안한 기법은 다음과 같은 장점을 가지고 있다.

1. 클러스터의 볼륨과 에지에 관하여 미리 정의된 요건을 충족하는 비교적 높은 밀도를 가진 클러스터를 생성한다.
2. 클러스터링에 필요한 대부분의 인수가 사용자에 의해 외부 입력 변수로써 제공되지 않고 비디오 클립 자체의 정보를 사용하여 결정되기 때문에 제안한 기법은 하나의 입력 인수(*minPts*) 만을 필요로 한다.
3. 차후의 검색에서 클러스터와 구별하여 처리하여 검색 효율을 높이기 위하여 주변점들을 적절히 식별해 낼 수 있다.

본 논문에서는 다차원 시퀀스의 예로써 비디오 데이타 세트를 중점적으로 언급하였고 실험 데이타로 사용하였지만, 다차원 시퀀스의 클러스터링은 회사의 판매 패턴, 기후 변화, 음성 신호 처리 등 여러 응용 분야에서 사용될 수 있을 것이다. 향후 연구로써, 제안한 클러스터링 기법에 의거하여 다차원 시퀀스의 저장 및 검색에 관하여 연구할 계획이다.

참 고 문 헌

[1] S. L. Lee, S. J. Chun, D. H. Kim, J. H. Lee, and C. W. Chung, "Similarity search for multidimensional data sequences," *Proceedings of IEEE Int'l Conference on Data Engineering*, pp. 599-608, 2000.

[2] A.Guttman, "R-trees: a dynamic index structure for spatial searching," *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pp. 47-57, 1984.

[3] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: an efficient and robust access method for points and rectangles," *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pp. 322-331, 1990.

[4] S. Berchtold, D. Keim, and H. Kriegel, "The X-tree: an index structure for high-dimensional data," *Proceedings of Int'l Conference on Very Large Data Bases*, pp. 28-39, 1996.

[5] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+ tree: a dynamic index for multi-dimensional objects," *Proceedings of Int'l Conference on Very Large Data Bases*, pp. 507-518, 1987.

[6] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," *Proceedings of Int'l Conference on Very Large Data Bases*, pp. 144-155, 1994.

[7] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pp. 103-114, 1996.

[8] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Int'l Conference on Knowledge Discovery in Databases and Data Mining*, pp. 226-231, 1996.

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

[10] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pp. 73-84, 1998.

[11] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park, "Fart algorithms for projected clustering," *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pp. 61-72, 1999.

[12] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," *Proceedings of ACM SIGMOD Int'l Conference on Management of Data*, pp. 419-429, 1994.

[13] V. Kobla, D. Doermann, and C. Faloutsos, Video "Trails: Representing and visualizing structure in video sequences," *Proceedings of ACM Multimedia*, pp. 335-346, 1997.

이 석 룡
정보과학회논문지 : 데이타베이스
제 28 권 제 2 호 참조

임 동 혁
정보과학회논문지 : 데이타베이스
제 28 권 제 2 호 참조

정 진 완
정보과학회논문지 : 데이타베이스
제 28 권 제 2 호 참조