클러스터 타당성 평가기준을 이용한 최적의 클러스터 수 결정을 위한 고속 탐색 알고리즘

Fast Search Algorithm for Determining the Optimal Number of Clusters using Cluster Validity Index

이상욱
목원대학교 정보통신공학과

Sang-Wook Lee(slee@mokwon.ac.kr)

요약

클러스터링 알고리즘에서 최적의 클러스터 수를 결정하기 위한 효율적인 고속 탐색 알고리즘을 소개한다. 제안하는 방법은 클러스터링 적합도의 척도로 사용되는 클러스터 타당성 평가기준을 토대로 한다. 데이터 집합에 클러스터링 프로세스를 진행하여 최적의 클러스터 형상에 도달하게 되면 클러스터 타당성 평가기준은 최대 혹은 최소값을 가질 것으로 기대한다. 본 논문에서는 최적의 클러스터 개수를 찾기 위한 고속의 비소모적 탐색 방법을 설계하고 실제 클러스터링과 접목한다. 제안하는 알고리즘은 k-means++ 클러스터링 알고리즘에 적용하였고, 클러스터 타당성 평가기준으로써 CB 및 PBM 타당성 평가기준 방법을 사용하였다. 몇몇의 가상 데이터 집합과 실제 데이터 집합에 실험한 결과, 제안하는 방법은 정확도의 손실 없이 계산 효율을 획기적으로 증가시킴을 보여주었다.

■ 중심어 : | 클러스터링 | 최적의 클러스터 수 | 클러스터링 타당성 평가기준 | 고속 탐색 알고리즘 |

Abstract

A fast and efficient search algorithm to determine an optimal number of clusters in clustering algorithms is presented. The method is based on cluster validity index which is a measure for clustering optimality. As the clustering procedure progresses and reaches an optimal cluster configuration, the cluster validity index is expected to be minimized or maximized. In this paper, a fast non-exhaustive search method for finding the optimal number of clusters is designed and shown to work well in clustering. The proposed algorithm is implemented with the k-mean++ algorithm as underlying clustering techniques using CB and PBM as a cluster validity index. Experimental results show that the proposed method provides the computation time efficiency without loss of accuracy on several artificial and real-life data sets.

■ keyword : | Clustering | Optimal Number of Clusters | Cluster Validity Index | Fast Search Algorithm |

I. 서 론

클러스터링이란 패턴 형태에 대한 사전 정보 없이 분류하는 unsupervised grouping 기법이다 [1]. 클러스터링의 목표는 알려지지 않은 데이터 구조의 라벨이 붙지 않은 데이터들을 몇 개의 의미 있는 그룹으로 분리시켜

* 이 논문은 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (과제번호: KRF-2007-357-D00236).
접수번호 : #090807-001
접수일자 : 2009년 08월 07일
심사완료일 : 2009년 09월 18일
교신저자 : 이상욱, e-mail : slee@mokwon.ac.kr

주는 것이다 [2]. 이러한 클러스터링은 패턴 인식, 기계 학습 및 데이터 마이닝 분야에서 매우 중요한 기술로 널리 사용되고 있다 [3].

클러스터링 과정에서 그룹핑 결과의 좋고 나쁨은 몇 개의 그룹으로 묶느냐에 따라 크게 좌우된다. 따라서 적당한 수의 그룹을 결정하는 것은 매우 중요한 이슈이다. 너무 많은 그룹으로 묶으면 복잡한 결과를 야기하여 해석하고 분석하는 것이 어렵다. 반면에, 너무 적은 그룹으로 묶으면 정보 손실을 야기하여 마지막 결론을 잘못 내릴 수가 있다 [4]. Dubes는 클러스터의 수를 결정하는 문제를 "클러스터의 유효성을 결정할 수 있는 기본적인 문제"라고 정의하였다 [5]. 몇 개의 그룹으로 클러스터링을 하는 것이 최선인가를 결정하는데 있어서 클러스터 적합성에 대한 측정은 필요불가결하다. 따라서 최적의 클러스터링 측정을 공식화하기 위한 수많은 노력들이 오랜 과거부터 현재까지 시도되어왔다 [6]. 그 결과 '클러스터 타당성 평가기준 (cluster validity index)' 라는 이름으로 다양한 종류의 클러스터 적합성 판단 기준들이 제안되었다.

클러스터링의 기본 목적은 입력 데이터들을 클러스터 내부 유사성을 (intra-cluster similarity) 최대화하고 클러스터 사이 유사성을 (inter-cluster similarity) 최소화하도록 특정한 척도 공간에서 몇 개의 그룹으로 분류하는 것이다 [7]. 유사성을 측정하기 위해 사용되는 가장 유명한 척도 방법은 데이터 간의 유클리디안 거리 (Euclidean distance)이고 가장 자주 사용되는 평가 함수는 제곱오차평가 (squared error criterion)이다 [6]. 이를 토대로 많은 연구자들이 최적의 클러스터 구조를 평가하기 위한 클러스터 타당성 평가 기준들을 제시하였다. 그러한 클러스터 타당성 평가 기준들은 최적의 클러스터 구조에 도달하게 되면 그 값이 최대화 또는 최소화 값을 가지도록 설계되었다.

그러나 클러스터 타당성 평가기준을 사용하여 최적의 클러스터 수를 찾고자 할 때, 클러스터링 문제의 unsupervised 특성 때문에 모든 가능한 클러스터 수에 대해 클러스터링 프로세스를 수행하여 클러스터 타당성 평가기준 값을 계산한 다음 그 값을 비교하여 최대 혹은 최소값을 찾는 소모적인 탐색을 수행한다. 본 논문에서는 클러스터 타당성 평가기준 값의 특성을 이용하여 소모적인 탐색 방법을 대치할 수 있는 고속 탐색 알고리즘을 제안한다.

II. 배경 연구

1987년, Dubes [8]가 클러스터 수를 결정하는 문제를 이슈로 표명한 이후 지금까지 최적의 클러스터 수를 결정하기 위해 많은 클러스터 타당성 평가기준들이 제안되었다. 본 논문에서는 기존에 많이 사용되던 DB (Davies-Bouldin), DD (Dunn's), XB (Xie-Beni) 클러스터 타당성 평가기준 보다 우수함이 입증된 PBM 평가기준 [9]과 CB 평가기준을 대표로 다룬다. 그러나 본 논문에서 적용하지 않은 다른 클러스터 평가기준을 사용하더라도 제안하는 고속 탐색 알고리즘이 사용될 수 있음을 미리 밝혀둔다.

이 논문에서는 다음과 같은 notation을 사용한다. m 차원 공간 벡터인 데이터 아이템 i 는 p_i 로 표현하고 전체 n개의 데이터 집합은 $P=\{p_1, p_2, \ldots, p_n\}$으로 표시한다. 클러스터 C_j는 클러스터링 알고리즘에 의해 함께 묶여진 데이터 아이템들의 집합을 나타내고 $C_j=\{p_1^{(j)}, p_2^{(j)}, \ldots, p_{n_j}^{(j)}\}$로 표시한다. 전체 클러스터의 개수를 k라 가정하면 $\sum_{j=1}^{k} n_j = n$이다. 클러스터 j의 중심을 $p_0^{(i)}$로 표시하면 $p_0^{(i)} = \frac{1}{n_j}\sum_{i=1}^{n_j} p_i^{(j)}$이고 클러스터 중심들의 집합은 $C=\{p_0^{(1)}, p_0^{(2)}, \ldots, p_{n_0}^{(k)}\}$로 표시한다. 데이터 전체 중심은 p_0로 표시하고 $p_0 = \frac{1}{n}\sum_{i=1}^{n} p_i$로 구할 수 있다.

1. 클러스터링 균형 평가기준 (Clustering Valance index - CB)

클러스터 내부 오차의 합 (intra-cluster error sum) Λ와 클러스터 사이 오차의 합 (inter-cluster error sum) Γ를 다음과 같이 정의한다.

$$\Lambda = \sum_{j=1}^{k}\sum_{i=1}^{n_j} \|p_i^{(j)} - p_0^{(j)}\|_2^2,$$

$$\Gamma = \sum_{j=1}^{k} \|p_0^{(j)} - p_0\|_2^2.$$

집괴적 군집화 (agglomerative clustering)에서, 클러스터 내부 에러는 클러스터링 과정에서 점차 증가하고 반대로 클러스터 사이 에러는 감소한다. 반면에 분할적 군집화 기법 (divisive clustering)에서는 위와는 반대의 성향을 보인다. 이를 토대로, Jung은 특정한 클러스터링 구조 χ에 대한 클러스터링 균형 (CB)을 다음과 같이 정의하였다 [10].

$$CB(k),\ \epsilon(\chi) = \alpha\Lambda + (1-\alpha)\Gamma, \quad (1)$$

여기서 $0 \le \alpha \le 1$는 두 합의 관계를 나타내는 가중치이며 일반적으로 0.5의 값을 사용한다. CB평가기준 $\epsilon(\chi)$는 두 에러 합들이 평행상태에 도달했을 때에 최적의 클러스터링 구조를 이루었다고 볼 수 있다는 직관적인 아이디어에서 유도되었다. CB 평가기준 값이 최소가 되었을 때 최적의 클러스터링을 이루었다고 생각한다.

2. PBM 평가기준 (PBM index – PBM)

PBM 평가기준은 다음과 같이 계산된다 [9].

$$PBM(k) = \left(\frac{1}{k} \times \frac{E_1}{E_k} \times D_k\right)^2, \quad (2)$$

여기서 E_k 와 D_k는 다음과 같다.

$$E_k = \sum_{j=1}^{k}\sum_{i=1}^{n_j} \|p_i^{(j)} - p_0^{(j)}\|_2,$$

$$D_k = \max_{i,j=1}^{k} \|p_0^{(i)} - p_0^{(j)}\|_2.$$

최적의 클러스터링 구조에 도달하면 PBM 평가기준 값은 최대가 될 것으로 생각된다.

III. 탐색 공간 축소 알고리즘

배경연구에서 살펴본 클러스터 타당성 평가기준들로부터, 우리는 어떠한 클러스터링 알고리즘을 사용하더라도 클러스터링 프로세스를 적당한 범위의 클러스터 개수에서 반복하면서 클러스터 타당성 평가기준 값을 관찰하여 최대 혹은 최소값에 이르는 점을 찾음으로써 최적의 클러스터 수를 유도할 수 있다는 사실을 알 수 있다. 일반적으로 '적당한 범위의 클러스터 수'라는 것은 명확하게 정의되어 있지 않다. 그러나 클러스터링 분야의 다양한 연구 경험에서 나온 추측은 최대 클러스터의 수는 데이터 수가 n개 일 때 $\sqrt{n}$을 초과하지 않는다고 발표된 바 있다 [11]. 이 연구 결과를 사실로 받아들여 적용한다 하더라도 최적의 클러스터 개수를 찾기 위해서는 클러스터의 개수가 2개 일 때 부터 $\sqrt{n}$ 까지 $\sqrt{n}$ -1번 ($O(\sqrt{n})$)의 클러스터링 프로세스를 반복해야만 한다.

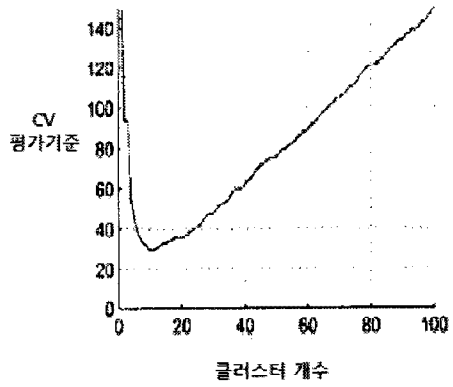


그림 1. 클러스터 개수에 따른 CB 평가기준 값 변화

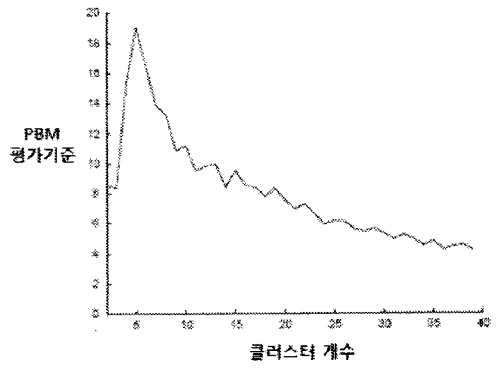


그림 2. 클러스터 개수에 따른 PBM 평가기준 값 변화

[그림 1]과 [그림 2]는 클러스터 수에 따른 CB 및 PBM 평가기준 값 변화를 보여주고 있다 [9][10]. 이 그림들로부터 CB 및 PBM 평가기준 값이 클러스터의 수에 따라 대략적인 컨벡스 (convex) 형태를 가지고 있음을 알 수 있다. 클러스터 타당성 평가기준이 완전하지는 않지만 대략적인 컨벡스 형태를 나타내는 경향은 CB 및 PBM외에 다른 클러스터 타당성 평가기준에서도 볼 수 있다. 이러한 클러스터 타당성 평가기준의 특성을 활용하면 최적의 클러스터 개수를 찾기 위해 반복해야 하는 클러스터링 프로세스 횟수를 획기적으로 줄일 수 있다. 본 논문에서 제안하는 방법은 다음과 같다. CB 평가기준을 바탕으로 설명할 것이며, 다른 평가 기준의 경우도 찾아야 하는 평가 기준 값이 최소이냐 최대이냐의 차이일 뿐 방법은 동일하다.

만약 탐색 공간이 대략적으로 컨벡스 형태를 구성하고 찾아야 할 최소를 포함하고 있다면, 탐색 공간 을 [그림 3]과 같이 5개의 점 (k1, k2, k3, k4, k5, 탐색 공간 - [k1, k5])을 찍어 4개의 구역으로 균등하게 나누고 그 점들에서의 클러스터 타당성 평가기준 값들을 비교함으로써 탐색 공간을 줄여나갈 수 있다. 만약 $CB(k2)$가 5개의 평가 기준 값 중 최소라면, [그림 3] (a)를 참조하면 탐색 공간을 [k1, k3]로 줄일 수 있음을 직관적으로 알 수 있다. 같은 방법으로 $CB(k3)$이 최소라면 [그림 3](b)를 참조하여 탐색 공간을 [k2, k4]로 줄일 수 있음을 알 수 있으며, $CB(k4)$가 최소라면 [그림 3](c)를 참조하여 탐색 공간을 [k3, k5]로 줄일 수 있음을 확인할 수 있다. $CB(k1)$과 $CB(k5)$이 최소가 되는 경우는 각각 $CB(k2)$와 $CB(k4)$가 최소가 되는 경우에 포함되기 때문에 생략이 가능하다. 따라서 $CB(k2)$, $CB(k3)$, $CB(k4)$가 5개 중 최소가 되는 [그림 3](a), (b), (c)의 3가지 경우만 고려하면 된다. 이를 토대로, 다음과 같은 반복적인 탐색 공간 축소를 수행함으로써 클러스터 타당성 평가기준 방법을 이용한 최적의 클러스터 개수 찾는데 있어 고속 탐색 기법을 제안할 수 있다.

(1) 탐색 공간을 5개의 점 k1, k2, k3, k4, k5로 정확히 4등분 하고 각 점에서 클러스터 타당성 평가기준을 계산한다.

(2) [그림 3] (a), (b), (c)의 각 조건에 따라 빗금 친 부분을 버림으로써 탐색 공간을 축소한다.

(3) (1)-(2) 과정을 원하는 탐색 공간만큼 축소 될 때까지 반복한다.

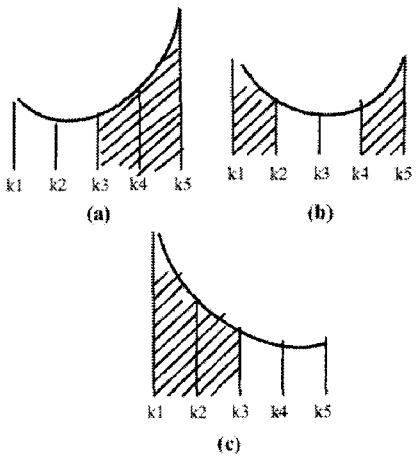


그림 3. 탐색공간을 4개의 균등한 부분으로 나누었을 때, 극점을 포함한 컨벡스 그래프의 3가지 경우들

[표 1]은 반복적인 탐색 공간 축소 방법에 대한 수도코드 (pseudo-code)이다.

표 1. 반복적 탐색 공간 축소 알고리즘

반복적 탐색 공간 축소 알고리즘

begin
$k1 = 1$;
$k3 = [(\sqrt{n}/2)]$;
$k5 = [\sqrt{n}]$;
while ($k5 - k1 >$ desired search space) do
$k2 = [k1 + (k5 - k1 + 1)/4)]$;
$k4 = [k1 + 3(k5 - k1 + 1)/4]$;
if($CB(k2) < CB(k3)$)
$k5 = k3$;
$k3 = k2$;
else
if($CB(k3) < CB(k4)$)
$k1 = k2$;
$k5 = k4$;
else
$k1 = k3$;
$k3 = k4$;
end while
end

여기서 [x]는 x의 소수점 이하를 버린 정수 값을 뜻한다. 반복적인 탐색 공간 축소를 실행하여 원하는 탐

색 공간에 도달한 후, 줄여진 탐색 공간에서는 클러스터링 타당성 평가기준의 전역 극점을 발견하기 위하여 완전 탐색을 실행한다. 이 방법은 매우 간단하지만 획기적으로 계산량을 줄일 수 있다. 탐색 공간 축소를 한 번 실행할 때 마다 절반의 탐색 공간을 줄일 수 있으며, 한번 축소 할 때 클러스터 타당성 평가기준을 계산하기 위한 클러스터링 프로세스를 단 두 번만 실행하면 된다 (k2와 k4). 따라서 제안하는 고속탐색 알고리즘은 계산량을 $O(\sqrt{n})$ 에서 $O(\log_2\sqrt{n})$으로 줄일 수 있다.

IV. 실 험

본 연구에서 제안한 고속 탐색 알고리즘의 효율성을 증명하기 위하여, 3개의 가상 클러스터 데이터 집합과 4개의 실제 데이터 집합에 알고리즘을 적용하여 실험하였고 소모적인 탐색 알고리즘과 비교분석 하였다. 소모적인 탐색 및 제안하는 고속 탐색 알고리즘은 클러스터링 과정과 최적의 클러스터 수를 찾는 과정이 분리되어있다. 그러나 근래에는 메타 휴리스틱을 사용하여 이 두 과정을 결합하여 한 번에 최적의 클러스터링 구조를 탐색하는 알고리즘들이 많이 제안되고 있다. 본 실험에서는 제안하는 고속 탐색 클러스터링 알고리즘과 휴리스틱을 사용한 클러스터링 알고리즘의 성능 또한 비교분석 하였다. 여기서 비교대상이 되는 알고리즘으로 휴리스틱 진화에 기반한 클러스터링 알고리즘을 사용하였다 [12].

Intel(R) Core(TM)2 Duo CPU 2.66GHz, 2.67GHz 컴퓨터에 Visual C++6.0을 사용하여 알고리즘을 구현하고 성능을 측정하였다. 클러스터링 알고리즘은 가장 유명한 k-means 알고리즘의 성능을 개선하여 만들어진 k-means++ 알고리즘을 사용하였다 [13].

1. 실험 문제

3개의 가상 데이터 집합을 가상1, 가상2, 가상3 으로 명칭하고 데이터 집합은 다음과 같은 방법으로 생성하였다. 여기서 $N(\mu,\sigma)$는 평균 μ와 표준편차 σ인 정규변수를 나타낸다.

4개의 실제 데이터 집합은 *Iris, Wine, Yeast, SCCTS* (Synthetic Control Chart Time Series)이다. 이 데이터 집합들은 UCI Machine Learning Repository 에서 참조하였다 [14].

1.1 *가상1* [그림 4]

250개의 랜덤 변수들이 [그림 4]와 같이 클러스터의 중심들이 직선을 형성하는 5개의 그룹으로 분류되어 있다. 5개의 그룹은 $N([-1,-1],[0.2,0.2])$, $N([0,0],[0.2,0.2])$, $N([1,1],[0.2,0.2])$, $N([2,2],[0.2,0.2])$, $N([3,3],[0.2,0.2])$에 의해 만들어졌으며 각 그룹은 50개의 점들로 이루어져 있다.

1.2 *가상2* [그림 5]

250개의 랜덤 변수들이 [그림 5]와 같이 클러스터의 중심들이 교차하는 형태의 5개의 그룹으로 분류되어 있다. 하나의 그룹은 $N([0,0],[0.2,0.2])$에 의해 만들어졌으며 100개의 점들로 이루어져 있다. 나머지 그룹들은 $N([-2,0],[0.3,0.3])$ $N([2,0],[0.3,0.3])$, $N([0,2],[0.4,0.4])$, $N([0,-2],[0.4,0.4])$에 의해 만들어 졌으며 각 그룹은 50개의 점들로 이루어져 있다.

1.3 *가상3* [그림 6]

1000개의 랜덤 변수들이 [그림 6]과 같이 클러스터의 중심들이 직사각형 격자구조를 형성하는 10개의 그룹으로 분류되어 있다. 10개의 그룹은 $N([0,0],[1,1])$, $N([0,6],[1,1])$, $N([0,12],[1,1])$, $N([0,18],[1,1])$, $N([6,0],[1,1])$, $N([6,6],[1,1])$, $N([6,12],[1,1])$, $N([12,0],[1,1])$, $N([12,6],[1,1])$, $N([12,12],[1,1])$에 의해 만들어 졌으며 각 그룹은 100개의 점들로 이루어져 있다.

1.4 *Iris*

패턴 인식 문헌에서 가장 많이 사용되는 데이터베이스 중 하나로서 데이터들은 4개의 특징 값들을 가진 서로 다른 iris 카테고리를 나타내고 있다. 각 50개의 아이템으로 구성된 3개의 그룹으로 이루어져 있으며 각 그룹은 iris plant의 종류를 나타낸다. 한 클래스는 다른 두 개의 클래스와 선형적으로 분리되어 있으며, 나머지

두 개의 클래스는 선형적으로 분리가 불가능한 것으로 알려져 있다. 따라서 대부분의 클러스터링 알고리즘은 이 데이터 집합으로부터 2개의 클러스터 개수를 최적의 클러스터 개수로 제공한다.

1.5 *Wine*

3개의 이탈리아의 같은 지역에서 자랐으나 서로 다른 3개의 품종으로부터 재배된 와인의 화학적 분석 결과로부터 얻어진 데이터 집합이다. 분석 결과 13개의 특성들로부터 3개의 와인 종류를 발견하였다. 각 그룹의 아이템 수는 59, 71, 48 이다.

1.6 *Yeast*

단백질의 위치를 예측하는 분야에서 만들어진 데이터 집합으로서 10개의 그룹이 8개의 특성들에 의해 분류되어 있다. 총 데이터의 수는 1484개 이다.

1.7 *SCCTS* (Synthetic Control Chart Time Series)

Alcok과 Manolopoulos 과정으로부터 합성하여 만들어진 제어 차트 600개 아이템으로 구성된 데이터 집합이다. 6개의 그룹이 각 100개의 아이템으로 분류되어 있으며 60개의 특성들로 구성되어 있다.

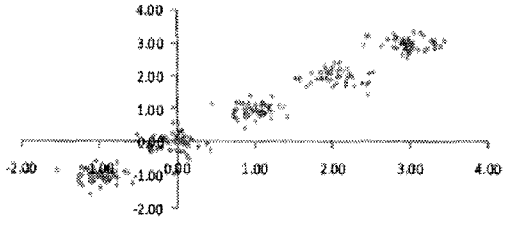


그림 4. 가상1

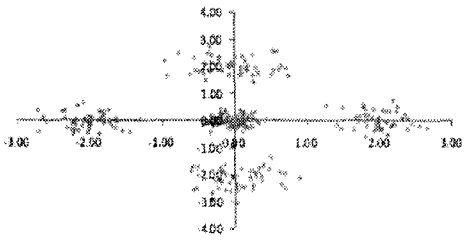


그림 5. 가상2

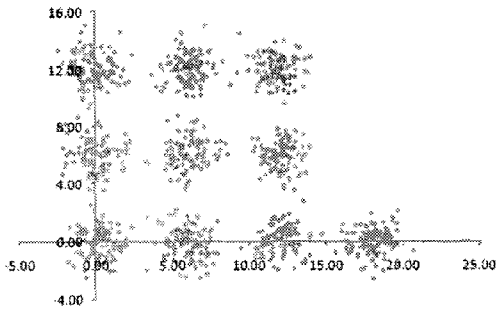


그림 6. 가상3

2. 휴리스틱 진화에 기반한 클러스터링 알고리즘

1975년 존 홀랜드에 의해 제안된 진화알고리즘을 적용한 클러스터링 알고리즘이며 진화속도를 향상시키기 위해 [그림 7]과 같이 교배연산 대신 휴리스틱 연산을 사용하였다 [12]. 각 클러스터 중심의 집합으로 개체를 표현하기 위해 실수로 표현된 가변적 길이를 가지도록 개체를 인코딩 하였고 적합도 평가를 위해 클러스터 타당성 평가기준을 이용하였다. 본 실험에서는 CB와 PBM 평가기준 값을 적합도로 사용하였다. 진화 연산으로는 돌연변이 연산과 합병연산, 분할연산, K-means 연산과 같은 휴리스틱 연산을 사용하였다 [12]. 돌연변이 연산은 현 세대의 중심의 위치를 랜덤하게 이동시키는데, 가우시안 분포 함수를 이용하여 현재 클러스터 중심으로부터 가까운 곳에서 새로운 중심이 선택될 확률을 높게하여 랜덤하게 결정한다. 3가지의 휴리스틱 연산은 다음과 같으며 [12]에 자세히 설명되어 있다.

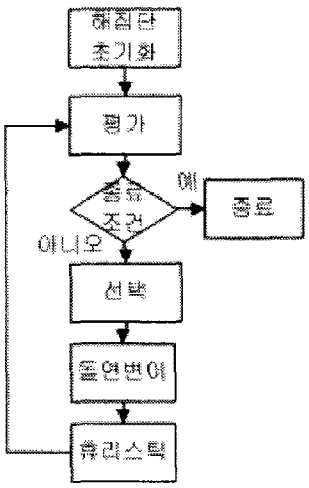


그림 7. 진화기반 클러스터링 알고리즘

2.1 합병연산

두 클러스터 중심간의 거리가 임계값 θ_M보다 작으면 두 클러스터의 중심 $p_0^{(i)}$, $p_0^{(j)}$를 합하여 한 개의 클러스터로 만든다. 생성된 클러스터의 중심 $p_0^{(k)}$는 $p_0^{(k)} = \frac{p_0^{(i)} + p_0^{(j)}}{2}$로 계산된다.

여기서 임계값 θ_M은 다음과 같이 평균 중심거리에 상수 α를 곱한 것으로 계산된다.

$$\theta_M = \alpha \left[\frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sqrt{\sum_{l=1}^{d} (p_0^{(i)} - p_0^{(j)})^2} \right]$$

만약, 한 세대에 두 중심 간의 거리가 임계값 보다 작은 것이 두 개 이상이면 그 중 가장 가까운 중심 두 개만을 합병한다 [12].

2.2 분할연산

클러스터의 표준편차 벡터 요소들 중 임계값 θ_S보다 크면 분할연산을 적용하여 중심 다음과 같은 계산으로 $p_0^{(i)}$를 $p_0^{(1i)}$, $p_0^{(2i)}$로 분할한다.

$$p_0^{(1i)} = p_0^{(i)} + \frac{1}{2}\sigma_k,\ p_0^{(2i)} = p_0^{(i)} - \frac{1}{2}\sigma_k,\ (\sigma_k = \max_i[\sigma_i])$$

이때 임계값 θ_S는 모든 클러스터 표준편차 벡터 요소 합의 평균에 상수 β를 곱한 값으로 다음과 같다.

$$\theta_S = \beta \left[\frac{1}{k} \sum_{i=1}^{k} \left[\frac{1}{d} \sum_{l=1}^{d} \left(\sqrt{\frac{1}{n_i} \sum_{j=1}^{n_i} (p_j^i - p_0^i)^2} \right) \right] \right]$$

만약, 한 세대에 표준편차 벡터 요소가 임계값 보다 큰 것이 두 개 이상일 경우에는 그 중 가장 큰 벡터 요소를 가지는 클러스터 하나만 분할한다 [12].

2.3 K-means 연산

K-means 알고리즘을 한 단계 적용한 연산으로써 돌연변이연산, 합병연산, 분할연산에 의해 생성된 중심을 정확한 방향으로 이동시키기 위한 연산이다.

위에서 정의한 연산들은 [표 2]와 같은 순서로 한 세대에 적용된다.

표 2. 연산 수행 순서

연산 수행 순서

돌연변이연산();
  if(두 중심간 거리 $< \theta_M$)
  합병연산();
  if(클러스터 표준편차 백터 요소 $> \theta_S$)
  분할연산();
K-means 연산();

3. k-means++

본 실험에서 성능 테스트를 위해 사용한 클러스터링 알고리즘으로써 기본 k-means 알고리즘의 정확도를 향상시킨 버전이다 [13]. 표준 k-means 알고리즘은 다음과 같다.

(1) 데이터 집합을 랜덤하게 k 개의 그룹으로 분할하고 클러스터 중심들 C를 계산한다.
(2) 데이터 집합의 각 요소들을 가장 가까운 클러스터에 할당한다.
(3) 현재 분할 결과에 대해 클러스터 중심들을 다시 계산한다.
(4) (2)-(3)의 과정을 각 클러스터의 변화가 없을때까지 반복한다.

k-means 알고리즘의 성능은 클러스터 중심의 초기 설정에 매우 민감하다. k-means++ 알고리즘은 초기의 클러스터 중심을 다음과 같은 방법으로 최대한 넓게 분포시켜 전역 최적해를 찾는데 도움을 준다. $D(p_i)$를 데이터 점 p_i로부터 가장 가까운 클러스터 센터까지의 최소 거리라고 놓으면 k-means++ 알고리즘은 다음과 같이 정의된다.

(1a) 초기 클러스터 중심 $p_0^{(1)}$을 P로부터 랜덤하게 정한다.

(1b) 다음 클러스터 중심 $p_0^{(i)}$를 $\frac{D(p_i)^2}{\sum_{j=1}^{n} D(p_j)^2}$의 확률로부터 $p_0^{(i)} = p_i \in P$로 정한다.

(1c) 1b의 과정을 k개의 클러스터 중심을 결정할 때까지 반복한다.

(2)-(4) k-means 알고리즘을 수행한다.

k-means++알고리즘은 k-means 알고리즘에 비해 속도와 정확도 측면에서 모두 우수함이 발표된바 있다 [13].

4. 실험 결과

표 3. 가상 데이터 집합에서의 CB 및 PBM 평가기준 값

클러스터 수	가상 데이터 집합					
	가상1		가상2		가상3	
	CB	PBM	CB	PBM	CB	PBM
2	139.9	12.00	283.8	1.506	17287	48.96
3	66.14	16.71	180.3	2.409	10388	50.20
4	42.04	26.40	97.09	5.145	7087	76.23
5	**25.16**	**65.56**	33.58	**11.85**	5561	66.87
6	28.38	50.24	**32.81**	9.482	4709	92.11
7	37.17	43.70	33.85	7.270	3863	91.02
8	37.57	36.16	33.57	7.644	3083	97.86
9	36.40	28.08	33.33	6.172	2301	121.41
10	36.32	24.95	35.34	5.707	**1627**	**171.1**
11	44.54	25.70	34.98	6.076	1657	163.63
12	49.15	23.29	34.72	4.764	1712	141.70
13	45.52	21.47	36.41	4.405	1646	127.51
14	49.39	19.53	34.37	4.544	1632	105.86
15	59.72	18.17	39.07	4.304	1657	107.88
16			38.66	4.233	1707	93.86
17			40.54	3.718	1767	93.04
18					1901	79.50
19					1694	78.31
20					2071	76.86
21					1836	68.63
22					2243	68.41
23					2162	67.43
24					2055	63.18
25					2241	61.99
26					2360	61.71
27					2266	54.43
28					2256	55.74
29					2548	55.07
30					2402	50.37
31					2568	51.54

표 4. 실제 데이터 집합에서의 CB 및 PBM 평가기준 값

클러스터 수	가상 데이터 집합							
	Iris		Wine		Yeast		SCCTS	
	CB	PBM	CB	PBM	CB	PBM	CB	PBM
2	**1328**	1992	3101	326	49.41	28.22	10303	7436
3	1366	**2517**	**2278**	474	42.63	21.93	7050	9263
4	1585	2013	2371	658	38.71	22.31	6787	5661
5	2021	1911	2609	647	35.88	21.87	6483	5193
6	2180	1689	3505	853	32.86	**39.4**	**6253**	3817
7	2540	1471	3550	1140	31.65	31.46	6344	3032
8	2721	1223	3717	1095	31.27	25.36	6369	3203
9	3162	966	4333	1002	30.86	21.71	6403	2617
10	3329	909	4580	1010	30.75	21.35	6550	2157
11	3801	885	4687	1051	30.64	18.37	6561	2092
12	4388	796	5969	**1244**	**30.5**	15.99	7151	1796
13			6433	1180	30.98	13.94	7317	1668
14					31.75	18.97	7542	1392
15					31.10	17.58	7451	1329
16					32.00	15.62	7698	1118
17					32.09	15.17	7930	1032
18					31.78	8.34	8168	914.2
19					32.36	12.12	8866	874.2
20					32.88	11.24	8735	871.6
21					33.24	7.112	8921	710.8
22					33.79	10.13	8623	724.9
23					34.06	8.994	9399	691.7
24					34.46	8.286	10546	614.8
25					35.64	8.207		
26					35.97	7.777		
27					36.17	7.341		
28					36.55	6.678		
29					37.17	6.196		
30					38.66	6.257		
31					39.21	5.806		
32					39.46	5.584		
33					39.61	5.382		
34					40.65	5.051		
35					41.02	4.874		
36					43.32	5.04		
37					43.72	4.746		
38					42.79	4.251		

클러스터의 수 k 값을 [2, $\sqrt{n}$]의 범위로 변화시키면서 각 k 값에 대해 k-means++ 알고리즘을 10번 반복 실험하여 그 중 클러스터 내부 오차의 합 (k-means 알고리즘의 목적 함수)이 가장 우수한 클러스터 구조에 대해 CB와 PBM 평가기준 값을 계산하였다.

[표 3]과 [표 4]는 3개의 가상 데이터 집합과 4개의 실제 데이터 집합에서 실험한 결과이다. 이 방법은 현재 기본적으로 사용되고 있는 소모적인 방법이며 표에서 볼드체로 표시된 부분은 발견한 최적해를 뜻한다. 제안

하는 고속 탐색 알고리즘의 클러스터링 프로세스는 밑줄 그어진 클러스터 수에서만 진행한다. 표에서 회색으로 그레이 처리된 부분은 반복적 탐색 공간 축소 방법에 의해 제거된 탐색 부분을 나타낸다. 탐색해야 할 공간이 클수록 계산 이득을 더 많이 볼 수 있음을 알 수 있다. 여기서 탐색 공간 축소 방법을 이용할때 원하는 탐색 공간의 크기 (desired search space)를 10으로 정했다. 이것은 줄여진 공간의 크기가 10 이하가 되면 완전 탐색을 한다는 것을 의미한다. 이 변수는 클러스터 타당성 평가기준 값 분포가 불완전한 컨벡스 형태임을 상쇄시킬 수 있는 중요한 역할을 담당한다.

표 5. CB 및 PBM 클러스터 타당성 평가기준을 사용하여 진화기반 클러스터링 알고리즘으로 구한 최적의 평가기준 값 및 클러스터 수

데이터 집합	CB		PBM	
	평가기준 값	발견한 클러스터 수	평가기준 값	발견한 클러스터 수
가상1	25.16	5	65.56	5
가상2	31.50	5	11.85	5
가상3	1615	10	**170.5**	10
Iris	1328	2	2517	3
Wine	2278	3	1244	12
Yeast	**30.28**	**10**	**40.8**	6
SCCTS	**6247**	6	**9286**	3

표 6. CB 및 PBM 클러스터 타당성 평가기준을 사용하여 소모적인 탐색, 고속 탐색 및 진화기반탐색으로 발견한 최적의 클러스터 수 및 계산시간 비교

데이터 집합	실제 클러스터 수	발견한 클러스터 수 (계산 시간 - 초)					
		CB			PBM		
		소모적 탐색	고속 탐색	진화기반탐색	소모적 탐색	고속 탐색	진화기반탐색
가상1	5	5 (1.25)	5 **(0.853)**	5 (1.03)	5 (1.25)	5 **(0.853)**	5 (1.12)
가상2	5	6 (1.188)	6 **(0.812)**	5 (0.987)	5 (1.188)	5 **(0.812)**	5 (0.954)
가상3	10	10 (34.18)	10 **(15.12)**	10 (25.48)	10 (34.19)	10 **(15.13)**	10 (26.58)
Iris	3	2 (0.203)	2 (0.153)	2 **(0.136)**	3 (0.203)	3 **(0.153)**	3 (0.145)
Wine	3	3 (0.515)	3 **(0.389)**	3 (0.51)	12 (0.515)	12 **(0.412)**	12 (0.43)
Yeast	10	12 (137.2)	12 **(42.58)**	10 (85.92)	6 (137.2)	6 **(40.83)**	6 (89.16)
SCCTS	6	6 (29.13)	6 **(13.84)**	6 (23.93)	3 (29.13)	3 **(13.84)**	3 (24.72)

[표 5]는 CB 평가기준과 PBM 평가기준을 이용하여 진화기반 클러스터링 알고리즘을 통해 구한 최적의 평가기준 값 및 클러스터 수를 보여준다. 이 실험에서는 개체집단의 크기를 30, α를 0.5, β를 1.5, 돌연변이 확률을 0.2로 하여 실험하였으며, 20세대 이상 최적해의 변화가 없으면 종료하도록 설정하였다.

결과 값은 10번 반복 실험한 결과를 평균 내어 기록하였다. 볼드체로 표시된 부분은 소모적인 방법 및 고속탐색 방법으로 구한 결과와 비교해 더 우수한 성능을 나타낸 부분을 뜻한다. 본 실험을 통해 전반적으로 진화기반 알고리즘이 소모적 탐색 및 고속 탐색 방법에 비해 우수한 성능을 보여줌을 알 수 있었다.

[표 6]은 CB 및 PBM 클러스터 타당성 평가기준을 사용하여 소모적인 탐색, 고속 탐색 및 진화기반 탐색으로 발견한 최적의 클러스터 수 및 계산시간을 비교한 결과를 보여주고 있다. 제안하는 고속 탐색 알고리즘은 소모적인 탐색 알고리즘과 동일한 결과를 제공하면서 계산량을 현격하게 줄여주는 반면 진화기반 알고리즘에 비해서는 몇몇 데이터 집합에서 저성능을 보임을 알 수 있다. 그러나 계산 효율 측면에서는 고속 탐색 알고리즘이 매우 뛰어남을 알 수 있으며 문제의 복잡도가 커질수록 계산 효율이 다른 알고리즘에 비해 더 우수함을 알 수 있었다.

V. 결 론

본 논문은 클러스터 타당성 평가기준을 사용하여 클러스터 수를 결정하는데 있어서 반복적 탐색 공간 축소 기법을 사용한 고속 탐색 알고리즘을 제안하였다. 3개의 가상 데이터 집합과 4개의 실제 데이터 집합에서 실험한 결과는 제안하는 방법이 소모적인 탐색 방법으로 구한 결과와 동일한 결과를 제공하면서 이론적으로는 계산량을 $O(\sqrt{n})$ 에서 $O(\log_2\sqrt{n})$으로 줄여줄 수 있음을 보여주었다. 또한 제안하는 고속 탐색 알고리즘을 진화기반 클러스터링 알고리즘과 비교 분석한 결과, 성능적인 측면서는 진화기반 클러스터링 알고리즘이 우수하나, 계산 효율측면에서는 고속 탐색 알고리즘이

우수함을 알 수 있었으며 문제의 복잡도가 커질수록 계산 효율의 차이가 벌어짐을 알 수 있었다.

생체정보학과 같은 현대의 클러스터링 문제는 방대한 양의 데이터를 다루고 있다. 본 논문에서 제안하는 고속 탐색 알고리즘이 방대한 양의 데이터 클러스터링 문제 해결에 계산 효율 측면에서 기여할 수 있을 것으로 생각한다.

참 고 문 헌

[1] B. Everitt, S. Landau, and M. Less, *Cluster Analysis*, Arnold, London, 2001.

[2] V. Cherkassky and F. Mulier, *Learning from Data: Concepts, Theory, and Methods*, Wiley, New York, 1998.

[3] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, *From Data Mining to Knowledge Discovery: An Overview*, The MIT Press, 1996.

[4] T. Ishioka, "An expansion of x-means for automatically determining the optimal number of clusters," In proceedings of fourth IASTED international conference, pp.91-96, 2005.

[5] R. Dubes, "Cluster analysis and related issue," In handbook of pattern recognition and computer vision, pp.3-32, World Scientific, 1993.

[6] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: a review," ACM Computing Surveys, Vol.31, No.3, pp.264-323, 1999.

[7] M. R. Anderberg, Cluster Analysis for Applications, Academic Press, New York, 1973.

[8] R. Dubes, "How many clusters are best? - an Experiment," Pattern Recognition, Vol.20, No.6, pp.645-663, 1987.

[9] M. K. Pakhira, S. Bandyopadhyay, and U. Maulik, "Validity index for crisp and fuzzy clusters," Pattern Recognition, Vol.37, pp.487-501, 2004.

[10] Y. Jung, H. Park, D. Du, and B. L. Drake, "A decision criterion for the optimal number of clusters in hierarchical clustering," Journal of Global Optimization, Vol.25, pp.91-111, 2003.

[11] Y. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model," IEEE Trans. Fuzzy Systems, Vol.3, No.3, pp.370-379, 1995.

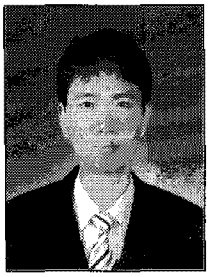
[12] 류정우, 강명구, 김명원, "휴리스틱 진화에 기반한 효율적인 클러스터러링 알고리즘", 한국정보과학회논문지: 소프트웨어 및 응용, 제29권, 제2호, pp.80-90, 2002.

[13] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," In 2007 Symposium on Discrete Algorithms, 2007.

[14] http://archive.ics.uci.edu/ml/

저 자 소 개

이 상 욱(Sang-Wook Lee) 정회원



- 2000년 2월 : 한국과학기술원 기계공학과(공학사)
- 2002년 2월 : 광주과학기술원 기전공학과(공학석사)
- 2007년 8월 : 광주과학기술원 정보기전공학부(공학박사)
- 2007년 9월 ~ 2008년 9월 : 조지아공과대학교 전산학과 박사후연구원
- 2008년 11월 ~ 2009년 2월 : 삼성전자 통신연구소 책임연구원
- 2009년 3월 ~ 현재 : 목원대학교 정보통신공학과 전임강사

<관심분야> : 휴리스틱 알고리즘, 최적화, 데이터 마이닝, 유비쿼터스 컴퓨팅