예측 데이터를 이용한 빠른 K-Means 알고리즘

Fast K-Means Clustering Algorithm using Prediction Data

지태창*, 이현진**, 이일병*
연세대학교 컴퓨터과학과*, 한국사이버대학교 컴퓨터정보통신학부**

Tae-Chang Jee(garura@csai.yonsei.ac.kr)*, Hyun-Jin Lee(hjlee@mail.kcu.ac)**,
Yill-Byung Lee(yblee@csai.yonsei.ac.kr)*

요약

본 논문에서 K-Means 군집화 알고리즘을 빠르게 적용하는 방법을 제안했다. 제안하는 알고리즘의 특징은 속도 향상을 위해 변화될 가능성이 있는 데이터를 예측하는 것이다. 군집화 알고리즘의 각 단계에서 군집이 변경될 가능성이 있는 데이터만 선택하여 군집 중심과의 거리를 계산함으로써 전체 군집 계산 시간을 줄일 수 있었다. 군집이 변화될 예측 데이터를 계산할 때는 K-Means 알고리즘을 적용하면서 생성되는 거리 정보를 사용함으로써 추가되는 계산 시간이 적고, 특히, 거리 정보를 이용하기 때문에 차원의 개수에는 영향을 덜 받는 알고리즘을 제안할 수 있었다. 제안하는 알고리즘의 성능 비교를 위해서 원래의 K-Means인 Lloyd's와 이를 개선한 KMHybrid와 비교했다. 제안하는 알고리즘은 대용량 데이터(입력 데이터의 크기가 크고, 데이터의 차원이 크며, 군집의 개수가 많은 경우)의 경우에 Lloyd's와 KMHybrid 보다 높은 속도 향상을 보였다.

■ 중심어 : | 패턴 인식 | 데이터 마이닝 | K-Means 군집화 |

Abstract

In this paper we proposed a fast method for a K-Means Clustering algorithm. The main characteristic of this method is that it uses precalculated data which possibility of change is high in order to speed up the algorithm. When calculating distance to cluster centre at each stage to assign nearest prototype in the clustering algorithm, it could reduce overall computation time by selecting only those data with possibility of change in cluster is high. Calculation time is reduced by using the distance information produced by K-Means algorithm when computing expected input data whose cluster may change, and by using such distance information the algorithm could be less affected by the number of dimensions. The proposed method was compared with original K-Means method - Lloyd's and the improved method KMHybrid. We show that our proposed method significantly outperforms in computation speed than Lloyd's and KMHybrid when using large size data which has large amount of data, great many dimensions and large number of clusters.

■ keyword : | Pattern Recognition | Data Mining | K-Means Clustering |

접수번호 : #080929-006
접수일자 : 2008년 09월 29일
심사완료일 : 2008년 11월 18일
교신저자 : 지태창, e-mail : garura@csai.yonsei.ac.kr

I. 서 론

군집화는 주어진 입력 데이터를 유사한 특징을 가지는 부분집합으로 구분하는 계산 작업이다. 일반적으로 이 부분집합들을 군집이라고 부른다. 군집들은 군집을 이루는 데이터들끼리는 서로 유사한 성질을 가지고, 다른 군집들과는 다른 성질을 가지게 된다. 군집화를 수행함으로써 원 데이터 집합의 정확도는 잃어버리지만, 단일성을 얻게 된다[8].

군집화는 데이터의 구조 등과 같은 사전 정보 없이 분석을 시작한다는 점에서 '자료의 탐험' 또는 '자료의 발굴' 과정으로 볼 수 있다. 그리고 양적으로 많은 데이터를 훨씬 작은 개수의 동질적 집단으로 분류함으로써 최소의 정보손실을 통한 '자료의 집약' 내지 '자료의 단순화'의 과정으로 볼 수 있다. 군집화의 결과는 모집단의 구조적 특성에 관한 정보를 도출함으로써 '가설의 형성' 단계와도 연계된다[13].

대상들을 군집화하는 방법은 매우 다양하지만 모든 방법이 공통적으로 가지고 있는 기본전제는 군집 내의 객체들 간의 유사성을 극대화 하고, 군집간의 유사성은 극소화하는 것이다. 군집화를 위한 알고리즘들은 다음과 같이 다양하게 제안되어 왔다. ISODATA[13], CLARA[17], CLARANS[23], Focusing Techniques[5], P-CLUSTER[14], DBSCAN[4], Ejcluster[11], BIRCH[27], GRIDCLUS[26]. 이런 군집화 방법들을 분류하면 계층적 기법 (hierarchical algorithms)과 분할 기법 (partitioning algorithms)으로 나눌 수 있다[12].

가장 많이 사용되는 분할 기법은 K-Means라고 보통 불리는 Lloyd's algorithm이다[7][19][20]. 이 알고리즘은 d-차원 공간상의 점으로 이루어진 입력 집합을 필요로 한다. 목표는 k개의 군집 중심을 찾고, 입력 집합을 나누는 것이다. K-Means의 주요 장점은 단순하고, 데이터 분석의 기초가 되며 일반적으로 효율적이라는데 있다. 단점은 사용자가 미리 군집의 개수를 설정해야 하고, 예외 값을 다루기 어렵다.

최근에 K-Means를 효율적으로 수행하기 위한 알고리즘들이 연구되어 왔다[1][15][16][22][24][25]. 이 알고리즘들은 가장 시간이 오래 걸리는 과정인 각 단계에서 최인접 이웃을 계산하는 시간을 줄이는 것을 목적으로 한다. 유클리디언 공간(Euclidean space) 상에서 K-Means 군집 문제를 해결하기 위한 $(1+\varepsilon)$-추정 알고리즘들도 연구되어 왔다[2][6][9][18]. 또한, 메트릭 공간(metric space)상에서 K-Means를 위한 효율적인 추정 알고리즘들도 있다[15][21].

본 논문에서는 군집 중심이 변화되지 않을 데이터를 예측해서 계산 시간을 단축하는 효율적인 K-Menas 알고리즘을 제안한다. 중심이 변하지 않을 데이터를 예측해서 계산을 수행하지 않음으로써 군집화 계산 시간을 단축시키는 것이다. 군집화를 수행할 때 입력 데이터와 군집 중심간의 거리를 계산하는 단계가 반복되는데, 이 과정이 가장 시간이 오래 걸리는 과정이다. 따라서, 각 단계에서 실제로 군집 중심이 변할 데이터만 예측해서 일부 데이터와 군집 중심간의 거리를 계산함으로써 군집화의 계산 시간을 단축한다. 예측 데이터 즉, 각 단계에서 거리를 계산할 데이터는 이전 단계에서의 군집 중심과 현재 단계에서의 군집 중심의 변화량을 이용해서 계산함으로써, 계산시간을 줄일 수 있었다.

K-Means 군집화 알고리즘의 시험대(testbed)가 [22]에 존재한다. [22]에서 제시하는 4개의 알고리즘 중 가장 기본적인 Lloyd's 알고리즘과 가장 빠른 KMHybrid 알고리즘과 실험 결과를 비교했다.

제안하는 방법과 Lloyd's와 KMHybrid [15][22]와 비교했다. KMHybrid는 Lloyd's에 임의의 swap과 simulated annealing을 결합한 것이다. 제안하는 방법은 Lloyd's와 KMHybrid와 비교하여 좋은 성능을 보였다. 입력 데이터의 차원이 작은 경우에는 Lloyd's 보다 성능이 좋지 않았지만, 차원이 15이상으로 올라갈 경우에는 성능이 좋았고, 고차원이 될수록 더 좋은 성능을 보였다.

본 논문의 구성은 다음과 같다. 먼저, 2장에서는 기존 방법에 대해서 기술하고 3장에서는 제안하는 방법을 설명한다. 그리고 4장에서는 실험결과를 분석하고, 마지막 5장에서 결론을 보인다.

II. Preliminaries

1. 기본적인 K-Means 알고리즘

기본적인 K-Means 알고리즘[3]에 대해서 간단히 살펴본다. [그림 1]에서 K-Means 군집화 알고리즘(Lloyd's)에 대해 설명하고 있다.

Lloyd's( )
Initialize k prototypes $(w_1,\ldots,w_k)$
Each cluster C_j is associated with prototype w_j
Repeat
For each input vector x_i, where $i \in \{1,\ldots,n\}$, do
Assign x_i to the cluster C_{j*} with nearest prototype w_{j*}
(i.e., $|x_i - w_{j*}| \le |x_i - w_j|, j \in \{1,\ldots,k\}$)
For each cluster C_j, where $j \in \{1,\ldots,k\}$, do
Update the prototype w_j to be the centroid of all samples currently in C_j, so that

$$w_j = \sum_{x_i \in C_j} \frac{x_i}{|C_j|}$$

Until cluster membership no longer changes

그림 1. 기본적인 K-Means 알고리즘

k의 적절한 개수는 문제와 분야에 의존적이고, 일반적으로 사용자가 k를 변화시키면서 테스트한다. 입력데이터로 n개가 있고, 각 데이터는 d 차원으로 구성된다고 할 때, Lloyd's의 각 반복 단계([그림 1]의 Repeat 구문)에서의 계산 시간은 다음과 같다.

1. [그림 1]의 첫 번째 **For** 문의 계산 시간은 $O(knd)$이다.
2. 중심점을 계산하는데 필요한 계산 시간([그림 1]의 두 번째 For 문)은 $O(nd)$이다.

각 단계의 반복 횟수는 입력 데이터의 수, 군집의 수, 입력 데이터의 차원의 수에 의존하여 수번에서 수백번으로 매우 다양하다. 따라서, K-Means를 직접 적용하면 계산시간이 $O(lnkd)$로 매우 증가하게 된다(여기서, l은 단계의 반복 횟수).

2. KMHybrid 알고리즘

실험에서 제안하는 알고리즘은 Lloyd's와 KMHybrid [15][22] 알고리즘과 비교했다. KMHybrid는 K-Means의 중심을 swap하는 방법과 다양한 simulated annealing 기법을 결합한 방법이다. 이 알고리즘은 Lloyd's의 각 단계에서 군집의 중심을 swap하고 용인 실험을 한다. 변경된 솔루션이 실험을 통과하면, 변경된 솔루션으로 계속 진행하고, 그렇지 않으면, 이전 솔루션을 복원한다. 용인 실험은 simulated annealing을 이용하여 수행한다. 게다가 kD-tree를 사용하여 최인접 이웃을 찾는 시간을 단축했다. 자세한 알고리즘은 [15][22]에서 살펴볼 수 있다.

Efficient *K*-Means ( )
Initialize k prototypes $(w_1,\ldots,w_k)$
Each cluster C_j is associated with prototype w_j
Copy w_j *to* w_j'
Repeat **Lloyd's( )** for one iteration
Compute distance matrix D_{ij}
For each input vector
For each prototype, do
$D_{ij} = |x_i - w_j|$
Repeat
Call **PredictInput( )** to predict input vector
For each predict input vector
Assign $\hat{x}_i$ to the cluster C_{j*} with nearest prototype w_{j*}, and update distance matrix D_{ij}
(i.e., $|x_i - w_{j*}| \le |\hat{x}_i - w_j|, j \in \{1,\ldots,k\}$)
Copy w_j *to* w_j'
For each cluster C_j, where $j \in \{1,\ldots,k\}$, do
Update the prototype w_j to be the centroid of all samples currently in C_j, so that

$$w_j = \sum_{x_i \in C_j} \frac{x_i}{|C_j|}$$

Until cluster membership no longer changes

그림 2. 제안하는 효율적인 K-Means 알고리즘

III. 제안하는 방법

제안하는 방법은 [그림 1]과 매우 유사하다. 따라서, 이해하기도 쉽고, 구현하기도 쉬운 장점이 있다. [그림 1]의 일반적인 K-Means와 다른 점은 첫 번째 **For** 문에서 일반적인 K-Means는 전체 입력 데이터를 사용해서 군집의 중심(prototype)까지의 거리를 계산하지만, 제안하는 방법에서는 군집이 변경될 가능성이 있는 입력 데이터(Predict Input)에 대해서만 거리를 계산한다.

K-Means를 수행할 때 최인접 이웃을 찾는 단계를 보면, 이전 군집 소속이 변경되는 경우는 초기 단계에서는 많아도 단계가 진행 될수록 점점 적어지게 된다. 따라서, 군집의 소속이 변경될 데이터 만 인접 군집을 다시 계산하고, 소속이 변경되지 않을 데이터는 인접 군집을 계산하지 않으면 전체 계산 시간을 단축 시킬 수 있다.

제안하는 방법은 [그림 2]와 같다. 제안하는 방법을 수행하는 과정은 일반적인 K-Means의 단계와 동일하다. 단, 최인접 군집을 계산하는 과정에서 전체 입력 데이터에 대해서 계산을 수행하는 것이 아니라, 변경될 가능성이 있는 데이터에 대해서만 계산을 수행함으로써 계산 수행 시간을 단축하는 것이다.

계산 시간을 줄이기 위해서 데이터 저장 공간이 추가로 필요한데, $w_j^{'}$와 D_{ij}가 그것이다. $w_j^{'}$는 이전 단계에서의 군집의 중심값을 저장하기 위한 벡터로 $O(kd)$의 저장 공간을 필요로 한다. D_{ij}는 전체 입력 데이터와 군집의 중심간의 거리를 저장하기 위한 행렬로 $O(kn)$의 저장 공간을 필요로 한다. $w_j^{'}$는 현 단계의 군집 중심값을 저장하는 벡터인 w_j를 그대로 복사한 것이기 때문에 계산 시간은 상수가 된다. D_{ij}를 구하기 위해 따로 계산을 수행하면 시간이 오래 걸리지만, 입력 데이터와 군집 중심간의 거리를 구하면서 D_{ij}의 값을 수정하면 되기 때문에 마찬가지로 계산 시간은 거의 걸리지 않게 된다.

$w_j^{'}$와 D_{ij}를 계산하는 과정을 제외하면, [그림 1]과 [그림 2]의 차이점은 **PredictInput( )** 함수를 수행해서 소속 군집이 변화될 예측 데이터 집합인 $\hat{x}_i$를 구하고, 이 예측 데이터에 대해서 거리 계산을 다시하고 소속 군집을 변경하는 것이다. 예측 데이터 집합을 구하는 **PredictInput( )** 함수의 알고리즘은 [그림 3]과 같다.

입력 데이터 벡터 X에 대해, 각 단계에서 소속이 변하는 데이터는 $X^{'}$이 된다. $X^{'}$을 정확하게 찾기는 어렵기 때문에 이의 예측치인 $\hat{X}$ 을 찾아서, 이에 대해서만, 최인접 군집을 계산하게 된다.

PredictInput( )
  Compute difference of cluster
    $\hat{w}_j = |w_j - w_j^{'}|$
  Compute predict input
    For each input vector
      For each prototype, update distance matrix
        If $x_i \in C_j$ **then**
          $D_{ij} = D_{ij} + \hat{w}_j$
        Else
          $D_{ij} = D_{ij} - \hat{w}_j$
        Assign x_i to the cluster $\hat{C}_{j^*}$ with smallest D_{ij}
        (i.e., $D_{ij^*} \le D_{ij}, j \in \{1,\ldots,k\}$)
        If $j^* \neq j$ **then**
          Assign x_i to the predict input $\hat{x}_i$
          Deassign x_i from the cluster C_j

그림 3. PredictInput 알고리즘

$\hat{X}$ 은 예측 거리 $\hat{D}$ 를 계산해서 구한다. $\hat{D}$ 은 다음과 같이 계산하다.

$$\hat{D}_{ij} = \begin{cases} D_{ij} + \hat{w}_j \text{ if } X_i \in C_j \\ D_{ij} - \hat{w}_j \text{ if } X_i \notin C_j \end{cases} \quad (1)$$

D는 현재의 데이터와 군집간의 거리 행렬이고, $\hat{w}$은 이전 단계에서 계산된 군집의 중심과 현재 단계에서 계산된 군집의 중심과의 거리의 차로 그 식은 [그림 3]의 첫 번째 식인 $\hat{w}_j = |w_j - w_j'|$ 이다. w_j'은 이전 단계에서의 군집의 중심값이고, w_j는 현재 단계에서의 군집의 중심으로 [그림 2]의 마지막 **For** loop 단계에서 구하게 된다. $\hat{D}$ 은 현재 군집에 속한 데이터의 거리는 멀게 하고, 군집에 속하지 않은 데이터의 거리는 더 가깝게 함으로써 항상 $X^{'} \subset \hat{X}$ 이 된다. 그러므로, 각 단계의 수행 결과가 Lloyd's 알고리즘에 의한 군집 결과와 동일해지기 때문에 최종 군집 결과는 동일하다.

Claim 3.1 $X^{'} \subset \hat{X}$

Proof. $x_t \in X^{'}$ 이면서, $x_t \notin \hat{X}$ 인 입력 데이터 x_t가 있다고 가정하면, $X^{'} \not\subset \hat{X}$ 이 된다. 이 x_t 는 현재 속한 군집은 C_{l_1} 이고, 새로 거리 계산을 수행하면, C_{l_2}로 소속 군집이 변경된다. 한 단계 이전의 거리 벡터를

D_{ij}^{*} 라고 하면, $x_t \in X'$ 라는 의미는 $D_{tl_1} > D_{tl_2}$ 이 되고, 이는 $D_{tl_1}^{*} + \delta_{l_1} > D_{tl_2}^{*} + \delta_{l_2}$ 이 되고, 이를 변환하면 $\delta_{l_1} - \delta_{l_2} > D_{tl_2}^{*} - D_{tl_1}^{*}$ 가 된다. 이전에 소속된 군집이 C_{l_1} 이기 때문에 $D_{tl_1}^{*} < D_{tl_2}^{*}$ 이 된다.

$x_t \notin \hat{X}$ 이라는 의미는 $D_{tl_1}^{*} + \widehat{w_{l_1}} < D_{tl_2}^{*} - \widehat{w_{l_2}}$ 가 된다. 이를 변환하면, $\widehat{w_{l_1}} + \widehat{w_{l_2}} < D_{tl_2}^{*} - D_{tl_1}^{*}$ 이 된다.

두 식을 결합하면, $\delta_{l_1} - \delta_{l_2} > D_{tl_2}^{*} - D_{tl_1}^{*} > \widehat{w_{l_1}} + \widehat{w_{l_2}}$ 이고, 이는 $\delta_{l_1} - \delta_{l_2} > \widehat{w_{l_1}} + \widehat{w_{l_2}}$ 이 된다. $\hat{w}$ 은 이전 군집 중심과 현재 군집 중심의 차의 절대 값이기 때문에 항상 양이 된다. δ_l 은 실제로 변화된 군집 중심을 이용하여 구하기 때문에 $D_{tl}^{*} - D_{tl}$ 이 되고, 항상 $|\delta_l| \le \widehat{w_l}$ 이 된다.

$\delta_{l_1} - \delta_{l_2} > \widehat{w_{l_1}} + \widehat{w_{l_2}}$ 은 $|\delta_{l_1}| + |\delta_{l_2}| > \delta_{l_1} - \delta_{l_2} > \widehat{w_{l_1}} + \widehat{w_{l_2}}$ 이 되며, 이는 $|\delta_l| \le \widehat{w_l}$ 에 모순이 된다. 따라서 $x_t \in X'$ 이면, 항상 $x_t \in \hat{X}$ 이 되므로, $X' \subset \hat{X}$ 이 된다. 즉, 군집 중심이 변화될 데이터는 항상 군집 중심이 변화될 가능성이 있는 데이터로 예측하게 된다.

$\hat{D}$을 스캔해서 이전 단계에서 소속했던 군집이 변경된 데이터를 $\hat{X}$ 에 할당한다. $\hat{D}$ 을 계산할 때는 거리만을 이용하기 때문에 데이터의 차원은 고려할 필요가 없다. 따라서, $\hat{D}$ 을 계산하는 시간은 고차원 데이터의 경우에 상대적으로 작게 된다.

Efficient K-Means( )의 각 반복 단계([그림 2]의 Repeat 구문)에서의 계산 시간은 다음과 같다.

1. [그림 2]의 Initialize부터 Repeat 구문 전까지의 계산 시간은 $O(kn)$이다.
2. PredictInput()의 계산시간은 $O(kn)$이다.
3. [그림 2]의 Repeat 구문의 첫 번째 **For** 문의 계산 시간은 $O(k\hat{n}d)$이다. 여기서 $\hat{n}$은 **PredictInput( )**에 의해서 예측된 입력 데이터의 개수이다.
4. 중심점을 계산하는데 필요한 계산 시간([그림 2]의 두 번째 For 문)은 $O(nd)$이지만, 예측된 입력 데이터에 대해서만 중심점을 계산하면 $O(\hat{n}d)$이다.

따라서, Repeat 구문의 반복 횟수를 l이라고 하면, 전체 시간 복잡도는 $O(lk\hat{n}d + lkn)$이 된다. Lloyd's의 시간복잡도인 $O(lknd)$와 비교하면, $\hat{n} \ll n$이 되면, 제안하는 알고리즘이 더 빨리 수행될 것이다.

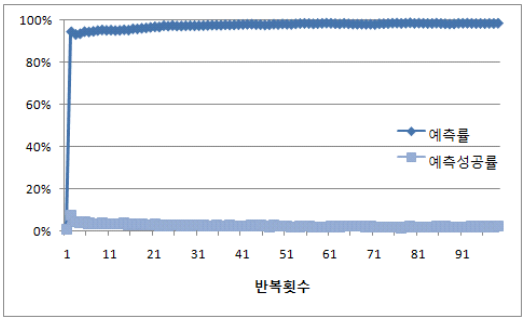

그림 4. 제안하는 방법의 예측 성능

$\hat{X} = X$ 이면, **PredictInput( )** 을 계산하는 시간이 추가되기 때문에 전체 계산 시간은 원래 알고리즘에 의한 계산시간보다 늦어지게 된다. 하지만, 일반적으로는 $\hat{X} < X$ 이기 때문에 원래 알고리즘 보다는 계산 시간은 빨라지게 되는데, 특히 대용량 데이터의 경우에 계산 시간이 빨라지게 된다.

제안하는 알고리즘의 성능 즉, 수행 시간에 영향을 끼치는 요인 중의 하나로 예측의 정확도를 생각할 수 있다. 군집이 변경될 가능성이 있는 데이터만 계산을 수행하면 수행 시간이 가장 빠르겠지만, 실제로는 군집이 변경되지 않은 데이터에 대해서도 계산을 수행할 것이다. 이상적인 상황에서는 $\hat{x} = x'$이지만, 일반적으로는 $\hat{x} > x'$이 된다. 제안하는 방법의 $\hat{x}$ 비율을 살펴보면 [그림 4]와 같다. 입력 데이터는 300,000개, 데이터의 차원의 수는 20개, 군집의 수는 10개이다.

[그림 4]를 보면, 예측률은 전체 데이터 대비 군집 중심이 변하지 않을 것이라고 예측한 데이터의 비이다. 예측성공률은 군집 중심이 변할 것이라고 예측한 데이터 대비 실제로 군집 중심이 변한 데이터이다. 전체적으로 예측률은 97%, 예측성공률은 2%로 예측성공률이 아주 좋다고 볼 수는 없지만, 실제로 군집 중심이 변화

된 데이터는 0.07%이기 때문에, 이를 고려하면, 정확률은 2800%이상 증가하였다. 각 단계마다 전체 데이터의 3% 즉, 90,000개의 데이터에 대해서만 거리 계산을 수행하기 때문에 전체 수행 시간은 단축된다.

IV. 실험 결과

실험은 Intel Core2Duo E6550, 2GB RAM 시스템 상에서 .Net Framework 2.0을 이용해서 수행했다. 실험 데이터는 임의로 구성하였고, 각 데이터에 대한 설명은 실험 내용에서 설명한다. KMHybrid[22]와 Original Lloyd's[22] 와 비교하였는데, 입력 데이터를 읽는 시간은 제외하고, 각 알고리즘의 시작부터 종료시까지의 수행 시간을 비교했다. 그림에 나와 있는 수행 시간의 비교 단위는 초(sec)이다.

KMHybrid, Lloyd's와의 수행 시간 비교를 위해서, 입력 데이터의 개수, 차원의 크기, 군집의 개수를 각각 변화시키면서 수행 시간을 비교하였다. 이런 목적을 위해서 인공적으로 데이터를 생성했다. 데이터는 가우시안(Gaussian) 분포를 따르며, 표준 편차는 0.02이다.

초기 중심점([그림 2]의 **Initialize**)은 Random Sampling 방법을 이용하여 계산했다. 1회의 실험은 100번 반복(iteration)을 했고, 각각 10회 실험을 수행한 평균값을 구했다.

1. Dependence on input size

입력 데이터의 크기와의 관계를 비교하기 위하여 군집의 개수는 10, 입력 데이터의 차원은 10으로 고정하고, 입력 데이터의 개수를 100,000 개부터 800,000 개까지 100,000개 단위로 증가시키면서 데이터를 생성했다. 그 결과는 [그림 5]와 같다. 시간은 10번씩 수행한 결과의 평균 값이며, 각 알고리즘은 모두 100번의 반복 횟수를 보였다. 입력 데이터의 개수가 작을 경우에도 제안하는 알고리즘의 수행시간이 더 좋은 것을 알 수 있다. 입력 데이터의 개수가 많아질 수 록 KMHybrid와 제안하는 알고리즘이 더 좋은 성능을 보이고 있다. 특히, 제안하는 알고리즘은 입력 데이터의 개수가 많아져도 시간이 증가하는 비율이 KMHybrid에 비해서 적은 것을 알 수 있다.

군집화 결과의 성능 비교 척도는 Sep(Separation) 값을 사용했다[10]. [그림 6]을 보면 Lloyd's와 KMHybrid는 비슷한 값을 보이고 있지만, 동일한 결과를 보이지는 않는다. 하지만, Lloyd's와 Suggest(제안하는 방법)를 보면, 서로 동일 한 결과를 보이는 것을 알 수 있다. 즉, 제안하는 방법은 Lloyd's의 군집 성능 저하 없이 군집 시간은 단축시키고 있다.

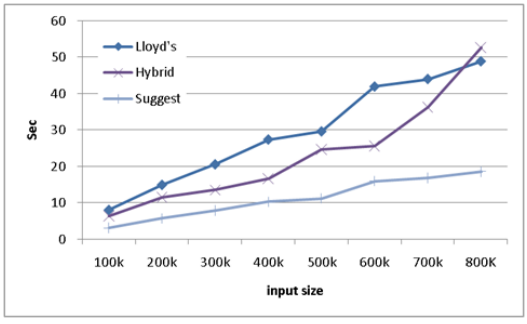

그림 5. 입력 데이터의 변화에 대한 수행 시간 비교

No. of Dim	Lloyd's	KMHybrid	Suggest
5dim	0.85	0.84	0.85
10dim	0.86	0.88	0.86
15dim	0.87	0.83	0.87
20dim	0.89	0.88	0.89
25dim	0.85	0.87	0.85
30dim	0.86	0.86	0.86
35dim	0.87	0.88	0.87
40dim	0.89	0.88	0.89

그림 6. 입력 데이터의 성능 비교

2. Dependence on the dimension

입력 데이터의 차원의 크기와의 관계를 비교하기 위하여 군집의 개수는 20, 입력 데이터의 개수는 300,000개로 고정하고, 입력 데이터의 차원을 5 개부터 40 개까지 5개 단위로 증가시키면서 데이터를 생성했다. 그 결과는 [그림 7]과 같다. 시간은 10번씩 수행한 결과의 평균 값이며, 각 알고리즘은 모두 100번의 반복 횟수를 보였다. 차원이 5인 경우에는 3개의 알고리즘이 거의 동일한 결과를 보이지만, 10이상 부터는 KMHybrid와 제

안하는 방법이 더 좋은 결과를 보이고 있다. 차원의 개수가 많아질 수 록 제안하는 방법이 KMHybrid와 비교하여 더 좋은 결과를 보이고 있다. 차원의 개수가 증가할 수 록 3개의 알고리즘 모두 수행시간이 증가하지만, 제안하는 방법의 증가 기울기가 더 작다. 즉, 제안하는 방법은 차원이 많을 수 록 더 효율적이라고 할 수 있다. 군집화 결과 성능을 비교해 보면 [그림 6]과 같이 Lloyd's와 제안하는 방법은 동일한 결과가 나오고, KMHybrid와는 다른 결과를 보였다.

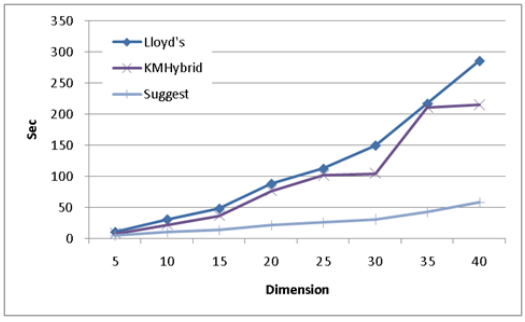

그림 7. 입력 데이터의 차원 변화에 대한 수행 시간 비교

3. Dependence on the number of clusters

군집의 개수와의 관계를 비교하기 위하여 입력 데이터의 차원은 10, 입력 데이터의 개수는 300,000개로 고정하고, 군집의 개수를 5 개부터 40 개까지 5개 단위로 증가시키면서 데이터를 생성했다. 그 결과는 [그림 8]과 같다. 시간은 10번씩 수행한 결과의 평균 값이며, 각 알고리즘은 모두 100번의 반복 횟수를 보였다. 전체적으로 제안하는 방법인 Lloyd's와 KMHybrid와 비교하여 빠른 수행 시간을 보이고 있다. 또한, 군집의 개수가 증가할 수 록 늘어나는 수행 시간의 기울기는 제안하는 방법이 가장 작게 증가하고 있기 때문에 제안하는 알고리즘은 군집의 개수가 많을 수록 더 효율적이라고 할 수 있다. 군집화 결과 성능을 비교해 보면 [그림 6]과 같이 Lloyd's와 제안하는 방법은 동일한 결과가 나오고, KMHybrid와는 다른 결과를 보였다.

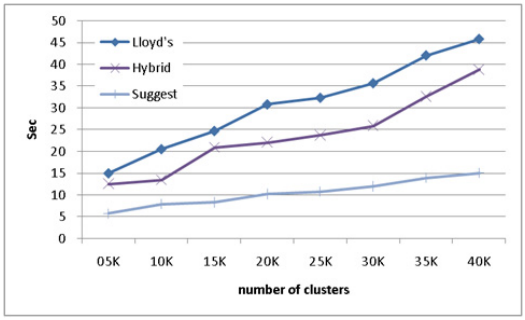

그림 8. 군집의 수의 변화에 대한 수행 시간 비교

4. Summary

실험 결과에서 언급하였지만, 3개의 알고리즘 모두 동일한 반복 횟수인 100회를 보였다. 100회 반복한 후의 결과를 살펴보면, 제안하는 방법은 Lloyd's에 기반하고 있기 때문에, Lloyd's의 결과와 소속 군집이 동일하고, 소속 군집과의 거리도 동일한 결과를 보이고 있다. 제안하는 방법의 주요 강점은 원래의 K-Means인 Lloyd's의 군집 성능에는 영향을 끼치지 않으면서, 효율적으로 동일한 결과를 보일 수 있다는 점이다.

제안하는 방법은 예측 데이터를 계산할 때 거리 정보만을 이용함으로써 차원에 덜 영향을 받으며, 그 결과는 실험 결과로 확인 할 수 있다. 입력 데이터의 수를 변화시킨 경우와 군집의 수를 변화시킨 경우에는 최대 3배 정도의 속도 향상을 보였는데, 차원을 변화 시킨 경우를 살펴보면 최대 6배 정도의 속도 향상을 보이고 있다.

V. 결론

본 논문에서는 예측을 이용해서 K-Means 군집화 알고리즘의 군집 시간을 효율적으로 개선한 알고리즘을 제안했다. 제안하는 알고리즘은 일반적인 K-Means인 Lloyd's[22]와 이를 개선한 KMHybrid[22]와 비교하여 입력 데이터의 차원과 입력 데이터의 개수, 군집의 개수가 큰 경우에 군집 성능에 있어서는 저하되지 않으면서, 군집화 시간은 크게 단축했다. 본 논문에서는 실험 결과를 보이지는 않았지만, 입력 데이터의 차원, 입력 데이터의 크기, 군집의 개수가 작은 경우에는 제안하는

방법이 더 효율적이라고 하기는 힘들다. 향후, 알고리즘의 반복 횟수에 대한 비교를 통해서 입력 데이터가 적은 반복 횟수로 수렴할 경우에 제안하는 방법의 효과를 살펴보고, 대용량 데이터가 아닐 경우에 (데이터의 크기와 차원 및 군집의 개수 관점에서) 성능 개선 방안에 대해 연구할 계획이다.

참 고 문 헌

[1] K. Alsabti, S. Ranka and V. Singh, "An Efficient K-Means Clustering Algorithm," Proc. of the first Workshop on High Performance Data Mining, 1998.

[2] M. Badoiu, S. Har-Peled, and P. Indyk, "Approximate Clustering via Coresets," Proc. 34th Annual ACM Symposium on Theory of Computing, pp.250-257, 2002.

[3] R. C. Dubes and A. K. Jain, "*Algorithms for Clustering Data*", Prentice Hall, 1988.

[4] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining, 1996.

[5] M. Ester, H. Kriegel, and X. Xu, "Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification," Proc. of the Fourth International Symposium on Large Spatial Databases, 1995.

[6] W. Fernandez de la Vega, M. Karpinski, C. Kenyon, and Y. Rabani, "Approximation schemes for clustering problems," Proc. 35th Annual ACM Symposium on Theory of Computing, pp.50-58, 2003.

[7] E. Forgey, "Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classification," Biometrics, Vol.21, No.3, p.768, 1965.

[8] G. Frahling and C. Sohler, "A fast k-means implementation using corsets," Proc. of the twenty-second annual symposium on Computational Geometry (SoCG'06), pp.135-143, 2006.

[9] S. Har-Peled and S. Mazumdar, "Coresets for k-Means and k-Median Clustering and their Applications," Proc. 36th Annual ACM Symposium on Theory of Computing, pp.291-300, 2004.

[10] J. He, A. H. Tan, and S. Y. Sung, "On Quantitative Evaluation of Clustering Systems," Information Retrieval and Clustering, Kluwer Academic Publishers, 2003.

[11] J. Garcia, J. Fedz-Valdivia, F. Cortijo, and R. Molina, "Dynamic Approach for Clustering Data," Signal Processing, Vol.44, No.2, 1994.

[12] J. Hartigan, "*Clustering Algorithms,*" John Wiley & Sons, New York, 1975.

[13] A. Jain and R. Dubes, "*Algorithms for Clustering Data,*" Prentice Hall, New Jersey, 1988.

[14] D. Judd, P. McKinley, and A. Jain, "Large-Scale Parallel Data Clusrering," Proc. of the International Conference on Pattern Recognition, 1996.

[15] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "A Local Search Approximation Algorithm for k-Means Clustering," Proc. of the 18th Annual Symposium on Computational Geometry (SoCG'02), pp.89-112, 2004.

[16] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An Efficient K-Means Clustering Algorithm: Analysis and Implementation," IEEE Trans. Pattern Analysis and Mchine Intelligence, Vol.24, No.7, pp.881-892, 2002.

[17] L. Kaufman and P. J. Rousseeuw, "*Finding Groups in Data: and Introduction to Cluster Analysis,*" John Wiley & Sons, 1990.

[18] A. Kumar, Y. Sabharwal, and S. Sen, "A simple linear time (1+ε)-approximation algorithm for k-means clustering in any dimensions," Proc. 45th IEEE Symposium on Foundations of Computer Science, pp.454-462, 2004.

[19] S. Lloyd, "Least Squares Quantization in PCM", IEEE Transactions on Information Theory, Vol.28, pp.129-137, 1982.

[20] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Vol.1, pp.281-296, 1967.

[21] R. Mettu and G. Plaxton. "Optimal Time Bounds for Approximate Clustering," Machine Learning, Vol.56, No.1-3, pp.35-60, 2004.

[22] D. Mount, "KMlocal: A Testbed for k-menas Clustering Algorithms," Available at http://www.cs.umd.edu/~mount

[23] R. T. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," Proc. of the 20th International Conference on Very Large Databases, pp.144-155, 1994.

[24] D. Pelleg and A. Moore, "x-Means: Extending k-means with efficient estimation of the number of clusters," Proc. 17th International Conference of Machine Learning, 2000.

[25] S. Philips, "Acceleration of k-Means and Related Clustering Problems," Proc. of Algorithms Engineering and Experiments, 2002.

[26] E. Schikuta, "Grid Clustering: An Efficient Hierarchical Clustering Method for Very Large Data Sets," Proc. 13th International Conference on Pattern Recognition, 1996.

[27] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," Proc. of the 1996 ACM SIGMOD International Conference. on Management of Data, pp.103-114, 1996.

저 자 소 개

지 태 창(Tae-Chang Jee) 정회원

- 1997년 2월 : 연세대학교 컴퓨터과학과(공학사)
- 1999년 2월 : 연세대학교 컴퓨터과학과(공학석사)
- 2004년 9월 ~ 현재 : 연세대학교 컴퓨터과학과(공학박사과정)

<관심분야> : 인공지능, 데이터마이닝, 패턴인식

이 현 진(Hyun-Jin Lee) 정회원

- 1996년 8월 : 순천향대학교 전산학과(공학사)
- 1998년 8월 : 연세대학교 컴퓨터과학과(공학석사)
- 2002년 8월 : 연세대학교 컴퓨터과학과(공학박사)
- 2003년 1월 ~ 현재 : 한국사이버대학교 컴퓨터정보통신학부 부교수

<관심분야> : 신경회로망, 데이터마이닝, 이러닝

이 일 병(Yill-Byung Lee) 정회원

- 1976년 2월 : 연세대학교 전자공학과(공학사)
- 1980년 5월 : University of Illinois 전산과학과(공학석사)
- 1985년 2월 : University of Massachusetts 전산정보과학과 (공학박사)
- 1985년 3월 ~ 현재 : 연세대학교 컴퓨터과학과 교수

<관심분야> : 신경회로망, 문서인식, Computer Vision, Data Mining, 필기체 문자 인식, Biometrics