http://www.jsebs.org
ISSN: 2288-3908

The Journal of Society for e-Business Studies
Vol.23, No.3, August 2018, pp.225-237
https://doi.org/10.7838/jsebs.2018.23.3.225

# 그룹특징기반 슬라이딩 윈도우 클러스터링에서의 k-means와 k-medoids 비교 평가

## Comparison between k-means and k-medoids Algorithms for a Group-Feature based Sliding Window Clustering

양주연(Ju-Yon Yang)*, 심준호(Junho Shim)**

### 초 록

대용량 데이터의 발생과 처리가 대중화되면서 대용량 데이터 스트림 처리에 대한 수요가 급격하게 증가하고 있다. 이 수요에 따라 다양한 대용량 데이터 처리 기술이 개발되고 있다. 한 분야로 주목받고 있는 방식은 슬라이딩 윈도우를 사용한 데이터 스트림 클러스터링이다. 슬라이딩 윈도우를 사용한 데이터 스트림 클러스터링은 윈도우가 이동할 때마다 새로운 클러스터를 생성한다. 기존의 슬라이딩 윈도우 상의 클러스터링 기법은 코어셋(Coreset)을 기반으로 데이터 스트림 클러스터링을 구현하고 있다. 이 연구에서는 코어셋을 활용한 그룹특징을 이용한 알고리즘 내에서 이용하는 클러스터링 알고리즘을 변경하였다. 그리고 이를 통해 제안 알고리즘과 기존 알고리즘의 파라미터 값 변화에 따른 성능 비교 실험을 진행하였다. 개선된 사항에 대해 논하여 두 알고리즘을 비교하고 실험자에게 파라미터에 따른 이용 방향을 제시한다.

### ABSTRACT

The demand for processing large data streams is growing rapidly as the generation and processing of large volumes of data become more popular. A variety of large data processing technologies are being developed to suit the increasing demand. One of the technologies that researchers have particularly observed is the data stream clustering with sliding windows. Data stream clustering with sliding windows may create a new set of clusters whenever the window moves. Previous data stream clustering techniques with sliding windows exploit the coresets, also known as group features that summarize the data. In this paper, we present some reformable elements of a group-feature based algorithm, and propose our algorithm that modified the clustering algorithm of the original one. We conduct a performance comparison between two algorithms by using different parameter values. Finally, we provide some guideline for the selective use of those algorithms with regard to the parameter values and their impacts on the performance.

**키워드** : 클러스터링, 데이터 스트림, 슬라이딩 윈도우, K-중간점
Clustering, Data Streams, Sliding Window, K-medoids

이 논문은 2017-18년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2017R1E1A1A03070004).
본 연구의 알고리즘 구현 및 실험 평가를 위해, 비교 코드 제공 및 코멘트를 주신 연종흠 박사님께 깊은 감사를 드립니다.

* First Author, Dept of Computer Science, Sookmyung Women's University(jyyangh92@sookmyung.ac.kr)
** Corresponding Author, Dept of Computer Science, Sookmyung Women's University(jshim@sookmyung.ac.kr)
Received: 2018-08-14, Review completed: 2018-08-22, Accepted: 2018-08-24

## 1. 서 론

다양한 응용 프로그램의 개발과 실시간 정보처리 기술의 대중화로 대형 데이터 스트림의 발생이 이전보다 급격하게 증가하고 있다. 다양한 매체에서 빅데이터(Bigdata)라는 단어를 언급하며 대용량 데이터에서 추측할 수 있는 정보에 대해 집중하고 있으며 이 영향으로 실시간 정보처리 기술에 대한 관심이 더 높아지고 있다.

빅데이터[6, 8]는 정보, 기술, 효과, 방법론의 총 집합체로서 정보 기술의 최신 동향으로 주시되고 있다. 빅데이터는 이름 그대로 대용량 데이터를 처리하고 이를 통해 추출한 정보를 이용하여 기술적, 혹은 사회적 적용을 하는 것까지 포함하는 매우 포괄적인 정의를 가진다.

빅데이터에서 사용되는 대용량 데이터는 실시간으로 수집되는 정보를 기반으로 한다. 따라서 빅데이터를 제대로 활용하기 위해서는 대형 데이터 스트림의 처리가 필수적이다. 결국 데이터 스트림에 포함된 다양한 정보를 얼마나 정확하게 분석해내는 지가 빅데이터의 활용 성공 여부를 판단한다고 볼 수 있다.

대형 데이터 스트림을 처리하는 방식에는 다양한 방식이 존재한다[11]. 데이터 스트림은 실시간으로 정보 값이 변화하는 특수성을 가지고 있고, 이를 통해 변화하는 추이를 확인하기 위해야 한다는 특수성을 가지고 있기 때문에 데이터 스트림을 실시간으로 클러스터링 처리하는 기법이 제시된다.

본 연구에서는 대형 데이터 스트림을 실시간으로 처리하는 클러스터링 기법 중 실시간 처리에 중점을 두는 알고리즘에 대해 이야기하고자 한다. 데이터 값을 그룹화[2]하고 클러스터링을 진행하는 과정을 구분하는 두 단계로 진행하는 클러스터링 방식[9]을 기반으로 하여 그룹화 하는 과정에서 사용하는 클러스터링 기법의 변화가 데이터 클러스터링 결과에 어떠한 영향을 미치는 지 확인하였다.

기존에 이미 제안된 알고리즘에서는 k-means 알고리즘을 이용하여 그룹화 한 데이터를 클러스터링 하였으나 제안 알고리즘에는 k-medoids 알고리즘을 적용하여 기존 알고리즘과 제안 알고리즘의 성능 차이와 데이터 셋에 따른 추이를 비교하였다. 그리고 이를 통해 파라미터에 따른 두 알고리즘의 선택 방향에 대해서 제안하였다.

## 2. 관련 연구

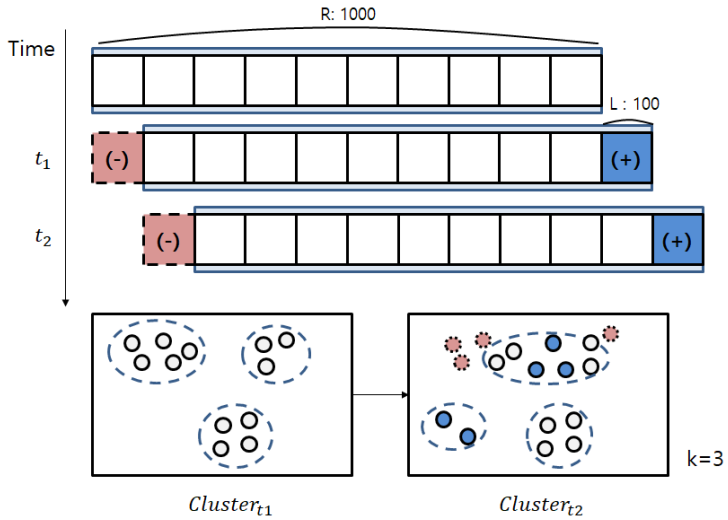
본 연구에서는 슬라이딩 윈도우 모델을 기반으로 하여 데이터 스트림을 처리하는 알고리즘을 활용하였다. 해당 알고리즘의 수행 과정은 두 단계로 나누어 진행된다. 각 단계는 슬라이딩 윈도우와 LSH 알고리즘을 이용하여 클러스터링을 진행하여 시놉시스, 즉 그룹특징을 생성하는 부분과 그룹특징을 활용하여 클러스터링을 진행하는 부분으로 나뉜다. 하지만 이 연구에서 중점적으로 보기 위한 부분은 그룹특징을 활용하여 클러스터링을 진행하는 두 번째 부분의 그룹화 클러스터링 알고리즘을 수정하고 성능을 비교하였다. 따라서 두 단계로 진행되는 알고리즘의 기본 원리와 중점적으로 사용되는 슬라이딩 윈도우에 대한 정의, 그리고 기존에 사용된 그룹화 클러스터링 알고리즘에 대해 기술하였다.

실시간으로 변화하는 데이터 값을 반영하기 위한 방법으로 윈도우 모델이 제안되었다.

윈도우 모델로는 랜드마크(landmark) 윈도우[1]와 감소형(Damped) 윈도우[5], 슬라이딩(sliding) 윈도우[9, 4], 각 세 가지가 널리 사용되고 있다. 랜드마크 윈도우는 데이터 스트림을 중복되지 않은 고정된 크기로 나누어 랜드마크 이후에 도착한 튜플을 포함한다. 이러한 형태는 일반적으로 주기적인 결과가 필요할 때 사용된다. 감소형 윈도우는 흔히 흐름형 윈도우 모델로 알려져 있으며 각 튜플이 가중치 값을 포함하고 있다. 그리고 이 가중치는 튜플의 생성시간이 오래될수록 감소한다. 그러나 이 두 윈도우 모델은 일부 데이터 스트림에서만 효과적인 결과를 낸다. 슬라이딩 윈도우 모델의 경우 윈도우마다 타임 스탬프가 각기 다른 튜플이 포함된다. 따라서 시간이 지나면 타임 스탬프가 만료된 튜플을 윈도우 내에서 제거한다. 이러한 특징은 클러스터의 시놉시스와 결과의 삽입, 삭제를 고려할 수 있게 한다.

슬라이딩 윈도우는 타임 스탬프가 현재 타임 스탬프와 윈도우의 시작 타임 스탬프 범위 내에 있는 튜플만을 포함한다. 데이터 스트림 관리 시스템의 질의에서 슬라이딩 윈도우의 길이를 범위(RANGE)로, 윈도우의 이동 간격에 대해 슬라이드(SLIDE)로 호출하여 각 값을 실험자가 직접 지정할 수 있다. <Figure 1>을 참고하면 슬라이딩의 방식에 대해 그림으로 설명하고 있는데, 이 그림에서는 슬라이드 범위 R을 1,000으로, 이동 간격 L을 100으로 구성하여, 시간 $t_1$, $t_2$에 구성되는 클러스터 $Cluster_{t1}$, $Cluster_{t2}$를 예시로 들고 있다. 예에서는 3개의 클러스터를 구성하고 있으므로, k-means, k-medoids 등 클러스터링 기법의 k값이 3으로 생각할 수 있다.

본 논문에서 기반으로 하는 연구[9]에서는 두 단계의 시놉시스 값 계산을 통해 클러스터링을 진행한다. 이는 단일 패스로 진행하는 클러스터링 방법[10]에 비해 상대적으로 시간 측면에서 더 효율성을 보이는 방법이기 때문이다. 해당 연구의 그룹특징에 대한 설명인 <Figure 2>를 바탕으로, 그 단계별 진행방법을 구체적으로 설명하고자 한다.



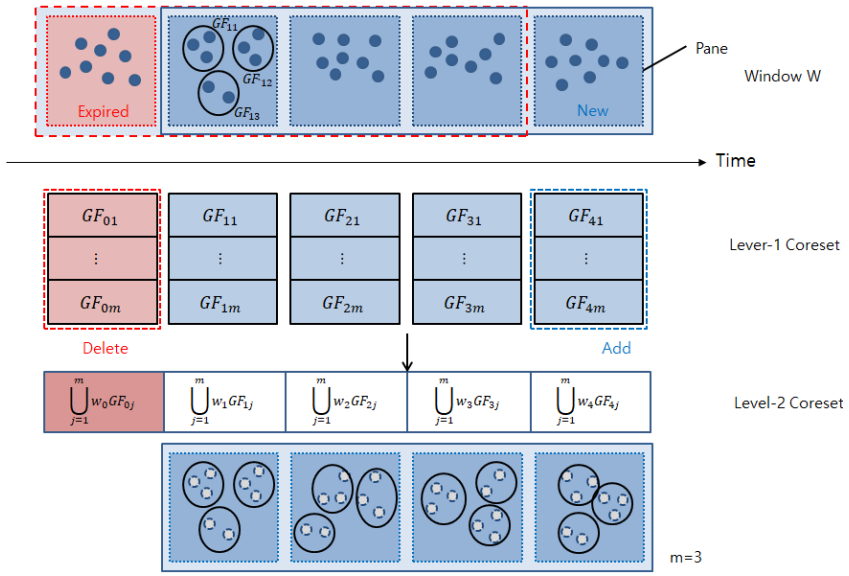

〈Figure 1〉 Sliding Window Process

<Figure 2>을 참고하면, 클러스터링의 결과값이라고 볼 수 있는 코어셋(Coreset)의 생성과정이 1차 단계와 2차 단계로 나뉘는 것을 알 수 있다. 1차 단계에서는 윈도우에 입력된 데이터 값을 슬라이드(SLIDE) 별로 나누고 이를 페인(PANE)이라는 별칭으로 명명한다. 그리고 페인 별로 내부의 값을 그룹특징이라는 명칭으로 요약하여 시놉시스를 생성한다. 그룹특징은 기본적으로 k-means 알고리즘을 활용하여 생성되며 이 그룹특징은 하나의 페인 당 여러 개가 존재할 수 있다. 각각의 그룹특징에는 그룹특징 내의 튜플 값의 선형 합 LS, 튜플의 제곱합 SS, 튜플의 수 N, 튜플의 최근 타임 스탬프 T가 포함되어 있다. 그룹특징은 클러스터링의 변화에 따라 업데이트 될 수 있다. 그룹특징의 중심점은 그룹특징의 요소를 활용하여 쉽게 계산할 수 있으며 이렇게 생성된 그룹특징들의 모임, 곧 클러스터링 결과값을 코어셋이라고 부르는 것이다.

2차 단계에서는 첫 번째 과정을 통해 생성된 1차 코어셋을 합하여 각 페인마다 하나의 코어셋을 새롭게 생성한다. 1차 코어셋 간의 거리를 비교하여 가까운 그룹특징 만을 코어셋 안에 합성하며 최종 거리가 전체 코어셋의 크기보다 작으면 해당 과정이 종료된다. <Figure 2>에서는 슬라이딩 윈도우가 4개의 페인으로 구성되고, 각 페인에는 3개의 그룹특징(m = 3)을 구성하여 코어셋을 구성하는 예시이다.

## 3. 제안 알고리즘

기존의 두 단계 시놉시스 생성 알고리즘 중, 2차 단계 시놉시스 생성 알고리즘을 변경한다. 기존에는 k-means 알고리즘을 활용하여 그룹특징을 처리하여 2차 시놉시스의 결과 값을 추출한다. 기존 알고리즘에서 그룹특징을 생성하는 알고리즘은 k-means 알고리즘[1]이다.




〈Figure 2〉 Clustering based on Group Features

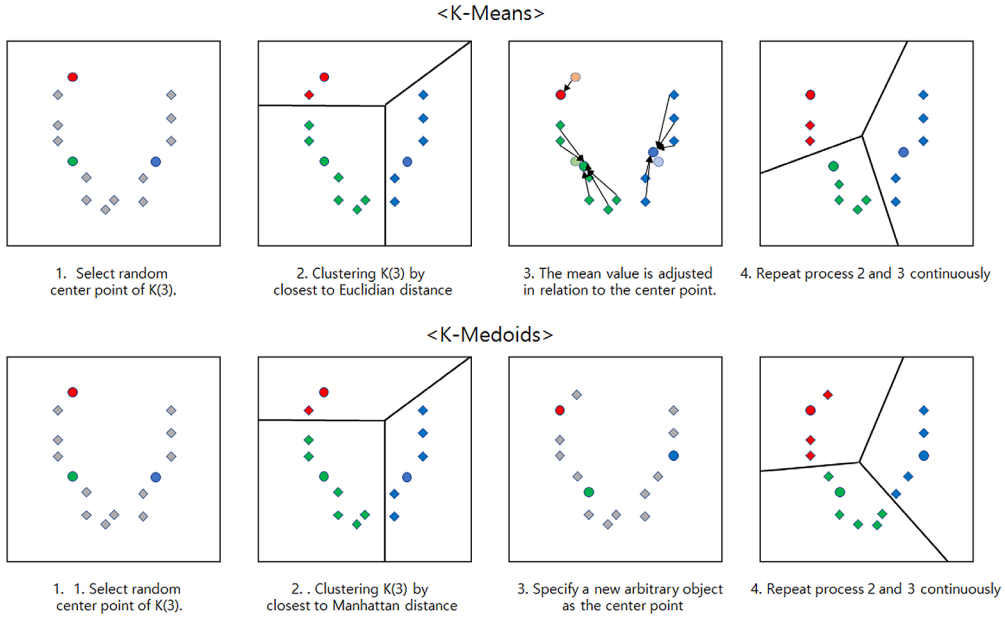
<Figure 3>의 설명을 보면 k-means 알고리즘은 K개의 클러스터를 생성하기 위해 무작위로 중심점을 생성하고 그 중심점을 기반으로 유클리드 거리를 비교하여 최적의 거리 값을 우선으로 클러스터를 생성한다.

유클리드 거리를 비교하기 때문에 데이터가 한 차원에 몰려 있는 등, 이상 값의 데이터 값이 존재하는 경우에는 성능이 급격하게 저하될 수 있으며 이러한 단점에도 통상적으로 일정한 성능 값을 내는 알고리즘이기에 다양한 분야에서 사용되고 있다.

k-means 알고리즘은 데이터 분포 형태에 따라 매우 판이한 결과를 나타낸다. 또한 데이터의 모든 정보의 가중치를 동일하다고 계산하여 데이터의 실질 분별에 있어서 신뢰도가 낮다는 단점을 가지고 있다. 따라서 k-means 알고리즘의 단점을 보강하기 위해 제시된 k-medoids 알고리즘을 적용하여 최종 클러스터링 값에 미치는 영향을 파악하고 개선 방향을 제시해보고자 한다.

k-medoids 알고리즘은 <Figure 3>을 보면 K개의 중간점을 선택하여 그 중간점을 기준으로 데이터 값들 간의 거리 비교를 맨해튼 거리로 측정하여 클러스터링을 진행한다. 이때 k-means 알고리즘처럼 유클리드 기하뿐만 아니라 이외의 기하 환경 내에서도 거리 계산이 가능하다. 하지만 선택되지 않은 각 데이터들에 대해, 해당 데이터를 자신이 포함된 집합의 새로운 중간점으로 가정하여 거리 비용을 계산하기 때문에 거리 계산에 연산 시간이 가장 많이 치중되는 단점이 있다.

그럼에도 k-means 알고리즘이 이상 값에 크게 반응한다는 단점을 해결하기 위해서 가장 우선적으로 k-medoids 알고리즘을 대체하고자 한다. 데이터셋의 형태를 따라 k-means 알고리즘을 적용한 클러스터링과 k-medoids 알고리즘을 적용한 클러스터링의 결과 값을 비교




〈Figure 3〉 Process for K-Means and K-Medoids

하여 클러스터링 성능과 시간 성능을 개선하는 알고리즘의 기반을 제시하고자 한다.

과정은 다음과 같다. 데이터 셋이 입력된 슬라이딩 윈도우가 2차 단계에서 그룹특징을 처리할 때 k-means 알고리즘과 k-medoids 알고리즘 중 어떤 것을 사용할 지를 선택하여 결과값을 출력한다.

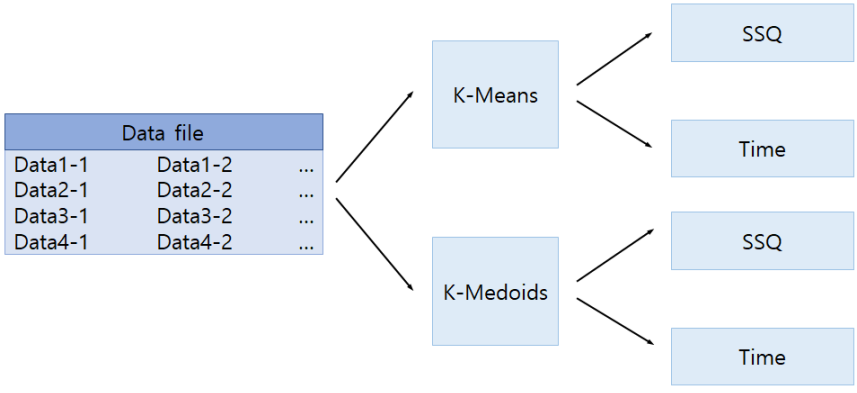
성능 평가는 <Figure 4>와 같이 두 가지로 나누어서 진행한다. 성능평가는 총 두 가지로 진행된다. 그룹특징의 제곱거리의 합(SSQ: Sum of Squared distance)[3] 성능과 총 수행 시간이다. 제곱거리의 합은 $\sum \| S_i - C_i \|^2$로 계산이 가능하다. 제곱거리의 합 성능은 그룹특징마다 값이 모두 다르므로 K개의 그룹특징 값의 평균 제곱거리의 합 성능 값을 비교하여 해당 알고리즘의 성능을 평가한다.

데이터 셋은 탭 구분 값 파일 내에 숫자로 구성되어 있으며 각 데이터의 구분은 공백으로 나뉘어져 있다. <Figure 4>를 참고하면 파일의 저장 방식을 확인할 수 있다. 윈도우에 값을 입력하는 Slide Action은 공백을 기준으로 하여 데이터를 구분하여 입력하며 미리 설정한 R과 L을 기준으로 하여 윈도우에 데이터 값을 저장한다. 그리고 해당 윈도우 내에서 그룹특성을 계산하여 생성한다. R에는 K개의 그룹특성이 생성되며, 윈도우 슬라이드 하나가 온전히 다 차야 제곱거리의 합 값을 계산한다.

윈도우 슬라이드에 들어간 데이터들은 LSH 함수를 통해 그룹특징을 생성한다. 그리고 그 그룹특징을 클러스터링 단위로 삼아 k-means 알고리즘, 또는 k-medoids 알고리즘을 이용하여 클러스터링을 진행한다.

k-means 알고리즘의 경우 유클리드 거리를 계산하는 함수를 이용하여 처리되며, k-medoids 알고리즘은 대중적으로 맨해튼 거리를 계산하는 PAM(Partitioning Around Medoids) 함수로 처리되게끔 설정하였다. 두 함수는 JSAT[7]에서 제공하는 오픈 라이브러리를 수정하여 사용하였다. 차이는 <Figure 5>와 <Figure 6>을 통하여 알 수 있다.

이때 사용되는 k-means 알고리즘과 k-medoids 알고리즘에서 초기 무작위 중심점, 중간점을 설정하는 방식은 모두 최대한 멀리에 존재하는 값들이 우선 지정되게 설정하였으며 이를 통해 보다 효과적인 알고리즘 구현을 진행하였다.




〈Figure 4〉 Performance Comparison Procedure

```
1: function K-Means clustering(Range, Slide)
2: Input: Data tsv File
3: Output: SSQ or Time
4: m = Number of Group Feature
5: Dp = Data points in Group Feature
6:        if Windows's size > m:
7:              Windows += Dp
8:              Run K-means with Dp, m
9:        end if
10:end function
```

〈Figure 5〉 How to Process TSV Files as K-Means

```
1: function K-Medoids clustering(Range, Slide)
2: Input: Data tsv File
3: Output: SSQ or Time
4: m = Number of Group Feature
5: Dp = Data points in Group Feature
6:        if Windows's size > m:
7:              Windows += Dp
8:              Run K-medoids with Dp, m
9:        end if
10:end function
```

〈Figure 6〉 How to Process TSV Files as K-Medoids

## 4. 실 험

### 4.1 실험 데이터

실험 환경의 운영체제는 64비트의 Windows 10 Home이다. 8.00GB 메모리(RAM)을 사용하였고, 프로세서는 인텔(Intel(R) Core(TM) i5-4460 CPU 3.20GHz)에서 진행하였다.

실험은 총 세 가지의 데이터 셋을 중심으로 진행되었다. 첫 번째로는 국내 포털 사이트에서 제공되는 뉴스 정보 값을 임의로 추출하여 생성한 'knews.tsv'[9]이다. 해당 파일은 뉴스 정보 값을 모아 숫자로 변경한 값들을 포함하고 있다. 두 번째는 루즈벨트 국제 삼림에서 추출한 지도제작 정보 값을 포함한 'Covtype.tsv'[9]이다. 자연의 값을 기반으로 하고 있기 때문에 첫 번째 파일과는 다르게 인간의 손길이 닿지 않은 분야에 대해 알고리즘 성능 여부를 판단할 수 있다. 세 번째 데이터 셋은 무작위로 구성된 'syn1k30d40.tsv'[9]이다. 일상적으로 생성된 데이터 값 이외에도 해당 실험의 효용성 여부를 판단하기 위하여 사용하였다.

## 4.2 실험 설정 값

실험은 Java 1.8에서 진행하였다. 데이터 셋은 위에서 언급한 세 가지의 각각의 특징을 가진 데이터를 사용하였다. <Figure 1>을 참조하면 K는 그룹특성, 곧 클러스터 설정 수, R은 윈도우 전체 범위의 수, L은 윈도우 슬라이드의 수를 지칭한다. 처음에는 R과 L값을 동일하게 부여한 환경 아래에서 K값을 변경해 가며 실험을 진행하였다.

R값과 L값은 증가하면 증가할수록 데이터 처리 속도가 급격하게 증가하는 사안을 이유로 실험환경에서 최상의 상태를 유지할 수 있는 값을 채택하였다. 따라서 R값은 500, L값은 100으로 설정한 뒤 K값을 조정하며 k-means 알고리즘과 k-medoids 알고리즘의 성능 상태를 확인하였다. 예를 들어 제 2장 관련 연구에서 나타난 <Figure 1>에서, R, L, K는 각각 1,000, 100, 3의 값을 나타내고 있다.

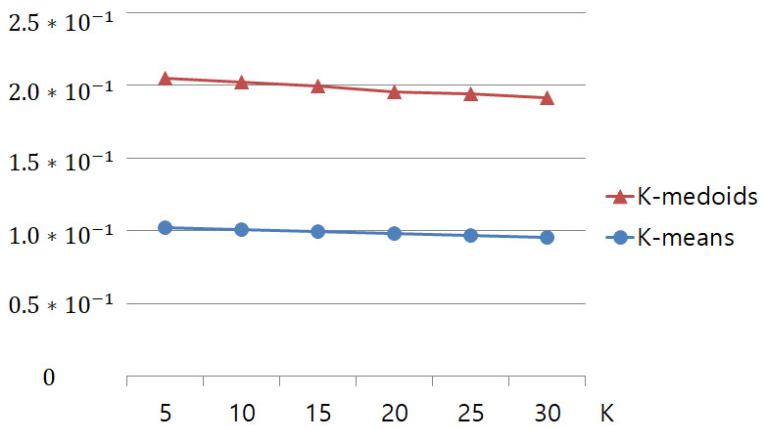
k-means 알고리즘과 k-medoids 알고리즘은 성능이 클러스터의 수인 K값에 가장 영향을 많이 받기 때문에 위와 같은 환경에서 비교하는 것을 중점으로 실험을 진행하였다.

## 4.3 실험 결과

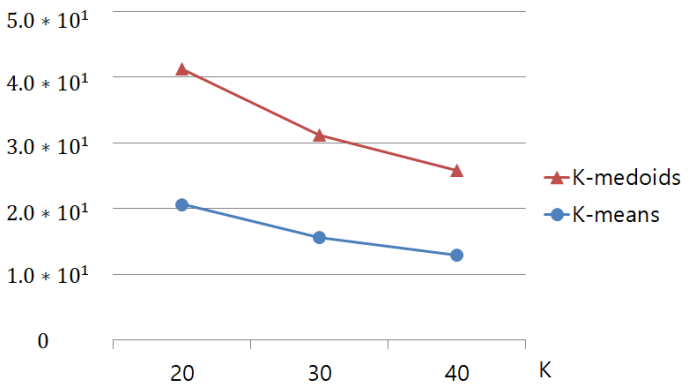
실험은 K를 변경하는 것을 기반으로 진행하였다. K값에 대한 실험을 진행한 결과 값을 표로 그려 성능 비교를 진행하였다. 성능은 평균 제곱거리의 합과 수행 시간을 비교하였으며 평균 제곱거리의 합의 비교 결과로는 얼마나 효율적인 클러스터링이 진행되었는지를 확인할 수 있다. 그리고 수행 시간의 비교 결과로는 어떤 알고리즘이 더욱 시간을 적게 소요하여 효율적인 성능을 냈는지를 확인할 수 있다.

### 4.3.1 K에 따른 평균 제곱거리의 합 비교

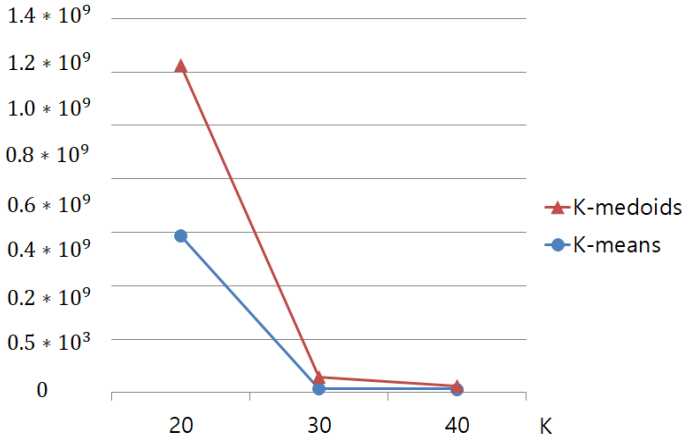
<Figure 7>은 'knews.tsv' 데이터 셋을 기준으로 하여 K값을 변화하여 평균 제곱거리의 합의 값을 비교한 실험 결과를 나타낸 그림이다. 그리고 <Figure 8>은 'covtype.tsv' 파일 값을 기준으로 하여 K값을 변화하여 평균 제곱거리의 합의 값을 비교한 실험 결과를 나타낸 그림이다. <Figure 9>는 'syn1k30d40.tsv' 파일 값을 기준으로 하여 K값을 변화하여 평균 제곱거리의 합의 값을 비교한 실험 결과를 나타낸 그림이다.




〈Figure 7〉 SSQ of the Dataset 'knews.tsv'(Comparative group: K)




〈Figure 8〉 SSQ of the Dataset 'covtype.tsv'(Comparative group: K)



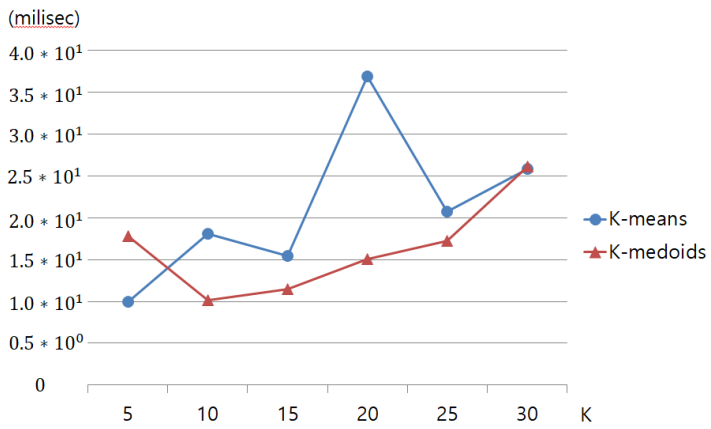

〈Figure 9〉 SSQ of the Dataset 'syn1k30d40.tsv'(Comparative group: K)

위의 세 그래프 모두를 비교하면 k-means 알고리즘이 k-medoids 알고리즘에 비해 상대적으로 클러스터링 결과가 확실한 형태를 보인다. K값이 증가함에 따라 두 알고리즘 간의 격차가 줄어드는 것은 클러스터링 알고리즘의 특성에 의한 것이므로 제할 수 있다.

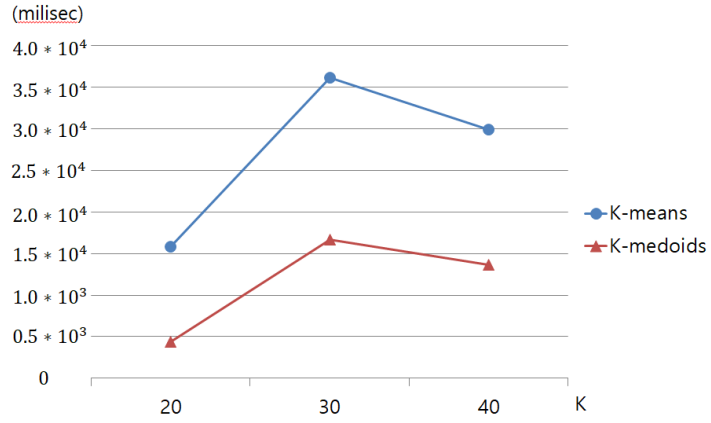
### 4.3.2 K에 따른 수행 시간 비교

<Figure 10>은 'knews.tsv' 데이터 셋을 기준으로 하여 K값을 변화하여 수행 시간의 값을 비교한 실험 결과를 나타낸 그림이다. 그리고 <Figure 11>은 'covtype.tsv' 데이터 셋을 기준으로 하여 K값을 변화하여 수행 시간의 값을 비교한 실험 결과를 나타낸 그림이다. <Figure 12>은 'syn1k30d40.tsv' 데이터 셋을 기준으로 하여 K값을 변화하여 수행 시간의 값을 비교한 실험 결과를 나타낸 그림이다.

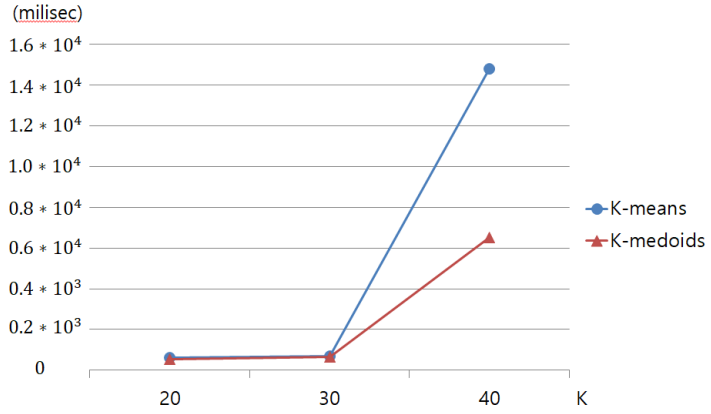
수행 시간 비교에서는 평균 제곱거리의 합의 값을 비교했을 때와 정 반대의 양상이 나타났다. k-medoids 알고리즘의 속도가 평균적으로 추측이 가능한 추이를 나타내며 k-means 알고리즘보다 좋은 성능을 냈음을 알 수 있다.




〈Figure 10〉 Process Time of the Dataset 'knews.tsv'(Comparative group: K)




〈Figure 11〉 Process Time of the Dataset 'covtype.tsv'(Comparative group: K)




〈Figure 12〉 Process Time of the Dataset 'syn1k30d40.tsv'(Comparative group: K)

## 5. 결 론

실험을 진행하며 예상하기로는 k-medoids 알고리즘의 경우가 k-means 알고리즘보다 더 개선된 알고리즘이라는 판단이 우선적으로 작용하였다. 따라서 k-medoids 알고리즘이 어떠한 환경에서든 더 좋은 결과를 나타낼 것이라고 예상하였다. 그러나 실험 결과 값은 예상과 다른 결과를 보였다.

k-means 알고리즘의 경우 '평균 제곱거리의 합' 측면에서 훨씬 좋은 성능을 나타냈고 k-medoids 알고리즘의 경우 '수행 시간' 측면에서 보다 더 좋은 성능을 나타냈다. 따라서 해당 클러스터링을 진행할 때, 어떠한 파라미터를 우선순위로 두고 클러스터링을 진행하는지에 따라 둘 중 어느 알고리즘을 선택하는 지에 결정적인 역할을 하게 될 것이라는 것을 알 수 있었다.

현장에서 해당 알고리즘을 활용하여 실시간 대형 데이터 스트림을 처리할 때는 실험의 결과 값에서 클러스터의 밀집도와 수행 시간 중 어떤 것을 더 우선으로 여기고 실험을 진행해야 할 지 우선적으로 고려한다.

정확한 클러스터링을 통해 보다 구체적인 정보를 유추해내는 것이 우선이 되는 경우가 있고, 보다 빠른 속도를 우선으로 하여 사용자로 하여금 해당 정보에 대해 대략적인 틀을 제공해주어야 하는 경우가 존재하기 때문이다.

더불어 본 연구에서는 실험 과정을 통해 클러스터 값을 어떻게 설정하는지에 따라 평균 제곱거리의 합 측면에서 차이를 나타낼 수 있음을 알 수 있다. 따라서 이에 대한 추가적인 실험을 통해 k-means 알고리즘과 k-medoids 알고리즘의 알고리즘 사용 방향성을 찾아낼 수 있다는 것을 알았다. 이는 추가적인 실험의 지속과정을 통해 정확한 수치와 비교 분석이 필요하며, 이를 통해 현장에서 이용하는 방향에 K, L, R과 같은 파라미터 측면에서 조금 더 명확한 가이드라인을 제시해 줄 수 있을 것이다.

## References

[1] Ackermann, M. R., Martens, M. Raupach, C., Wsierket, K., Lammersen, C., and Sohler, C., "StreamKm++: A clustering algorithm for data streams," Journal of Experimental Algorithmics, Vol. 17, No. 1, pp. 2.4:2.1-2.4:2.30, 2012.

[2] Aggarwal et al., "A framework for clustering evolving data streams," Proceedings of the 29th international conference on Very large data bases, Vol. 29, pp. 81-92, 2003.

[3] Anderson, M. J., "A new method for non-parametric multivariate analysis of variance," Austral Ecology, Vol. 26, No. 1, pp. 32-46, 2001.

[4] Braverman, V., Lang, H., Levin, K., and Monemizadeh, M., "Clustering problems on sliding windows," Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, pp. 1374-1390, 2016.

[5] Cao, F., Ester, M., Qian W., and Zhou A., "Density-based clustering over an evol-

ving data stream with noise," 2006 SIAM Conference on Data Mining, pp. 328-329, 2006.

[6] De Mauro, A., Greco, M., and Grimaldi, M., "A formal definition of Big Data based on its essential features," Library Review, Vol. 65, No. 3, pp. 122-135, 2016.

[7] Raff, E., "JSAT: Java statistical analysis tool, a library for machine learning," The Journal of Machine Learning Research, Vol. 18, No. 1, pp. 792-796, 2017.

[8] Yang, B. and Shim, J., "Practical Datasets for Similarity Measures and Their Threshold Values," The Journal of Society for e-Business Studies, Vol. 18, No. 1, pp. 97-105, 2013.

[9] Youn, J. H., A Scalable Clustering Algorithm for High-dimensional Data Streams over Sliding Windows, Diss. Seoul National University, 2017.

[10] Zhang, T., Remakrishnan, R., and Livny, M., "Birch, An efficient data clustering method for very large databases," SIGMOD record, Vol. 25, No. 2, pp. 103-114, 1996.

[11] Zhou, X. and Jin, Q., "A heuristic approach to discovering user correlations from organized social stream data," Multimedia Tools and Applications, Vol. 76, No. 9, pp. 11487-11507, 2017.

## 저 자 소 개

양주연 (E-mail: jyyangh92@sookmyung.ac.kr)
2015년 숙명여자대학교 멀티미디어과학과 (학사)
2018년 숙명여자대학교 컴퓨터과학과 (석사)
2018년~현재 한국갤럽 RA
관심분야 데이터베이스, 빅데이터, 웹

심준호 (E-mail: jshim@sookmyung.ac.kr)
1990년 서울대학교 계산통계학과 (학사)
1994년 서울대학교 계산통계학과 전산과학전공 (석사)
1998년 Northwestern University, Electrical & Computer Engineering (박사)
2001년~현재 숙명여자대학교 소프트웨어학부 교수
관심분야 데이터베이스, 빅데이터, e-비즈니스 기반기술, 웹