데이터 스트림 환경에서 임의 시간 구간에 대한 효율적 클러스터링 알고리즘*

장주현O ,문양세, 노희영
강원대학교 컴퓨터과학과
{jhjangO,ysmoon, rohhy}kangwon.ac.kr

Effective Time Interval Clustering Algorithm of Data Stream Environment

Joo-Hyun JangO, Yang-Sae Moon, Hi-Young Roh
Dept. of CS, Kangwon Nat'l Univ.

요 약

최근에 데이터의 양이 방대하게 늘어남에 따라 이러한 데이터의 처리를 위한 여러 연구들이 진행되어 지고 있다. 이 중에 데이터들 간의 군집 관계를 파악하기 위하여 사용되는 클러스터링에 관한 연구 가 많이 수행되었는데, 이중 BIRCH는 대용량의 데이터를 처리하는데 있어서 적합한 모델로 제시되고 있다. 하지만 BIRCH는 데이터 스트림 환경에서 클러스터링은 효과적이지 못한 단점을 가지고 있다. 본 논문은 데이터 스트림 환경에서 과거의 임의 시간구간에 대한 클러스터링을 수행하는 방법을 제안한다. 이를 위하여 CF-트리를 일정 시간 마다 생성 및 저장하고 이를 이용하여 사용자가 원하는 시간 구간에 대해 동안의 클러스터링을 수행한다. 본 논문에서는 임의 시간구간에 대한 효율적인 클러스터링을 위해 기존의 CF-트리 노드 구조에 추가 데이터를 사용하는 CF^{δ}-트리를 제안한다. 그리고 δ에 대한 연구를 통해, 근사적 접근법을 제안하였다.

1. 서 론

최근에 네트워크와 하드웨어의 발전으로 인해, 대용량의 스트림 데이터가 발생 하고, 이러한 데이터로는 통신 데이터, 은행, 증권 시장 분 석등이 있다.

스트림 데이터의 특징은 데이터의 흐름이 끊임 없고 대용량이라는 점이며, 순차적인 접근만이 가능하여, 기존의 데이터베이스에 저장되어 있는 데이터와 같이 랜덤 접근이 불가능하고, 한번 지나간 데이터는 다시 한 번 읽어 들이지 못하는 특징이 있다[1][2]. 이러한 특징을 고려하여 현재 데이터 스트림을 효과적으로 처리하기 위한 여러 연구들이 활발히 진행되고 있다[3][4].

본 논문에서는 데이터 스트림에 대한 클러스터링 문제를 다룬다. 클러스터링이란 데이터들의 유사성을 척도로 하여 군집 관계를 파악하고, 이를 이용하기 위한 마이닝 기법의 하나이다. 기존 클러스터링 연구 중 대용량의 데이터를 대상으로 하는 알고리즘은 BIRCH가 있으며, BIRCH는 모든 원본 데이터를 저장하지 않고 data의 특징(Feature)을 CF-트리로 구성하고, 이를 사용하여 클러스터링을 하는 one-pass 알고리즘이다[5]. 하지만 BIRCH는 과거의 임의 시간구간에 대한 클러스터링은 할 수 없고, 대용량의 데이터를 처리할 때 내부 노드가 증가됨에 따라 불필요한 메모리를 사용하며, 깊이가 깊어짐에 따라 관리가 어려운 단점을 가지고 있다.

본 논문에서 제안하고자 하는 방법은 기존의 CF-트리를 데이터 스트림 환경에 적합하게 하기 위해 수정한 CF^{δ}-트리를 제안한다. CF^{δ}-트리를 사용하는 프레임워크는 다음과 같다. 먼저, 일정 기간 동안의 데이터로 CF^{δ}-트리를 구축하여 저장한다. 저장된 데이터를 이용하여 클러스터링을 하는 법은 저장된 CF^{δ}-트리들의 임계값이 다를 수 있기 때문에 CF^{δ}-트리 단말 노드에 추가된 데이터를 사용하여, 일정한 임계값으로 복원 한다. 클러스터링은 저장한 다수의 CF^{δ}-트리 단말 노드를 이용하여 수행한다. 이 때 사용되는 추가 데이터 δ는 복원에 필요한 데이터로 본 논문에서는 우선 모든 데이터를 사용하는 방법을 제안한다. 하지만 이는 합병이 되는 횟수에 따라 많은 데이터를 필요로 함으로 δ의 크기를 최소화한 근사적 접근 방법을 제안한다.

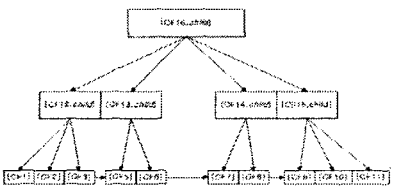
본 논문의 구성은 다음과 같다. 제 2장에서는 관련연구로 기존에 연구된 클러스터링 알고리즘을 설명한다. 제 3장에서는 제안하고자 하는 스트림 환경에서의 임의 시간구간에 대한 클러스터링 프레임워크와 이에 대한 접근방법을 제안한다. 마지막으로 제 4장에서 결론과 향후 연구과제에 대해 서술한다.

2. 관련 연구

BIRCH는 대용량의 데이터를 최소한의 메모리로 클러스터링 문제를 해결하고자 하는 클러스터링 알고리즘이다. 이를 위해서 BIRCH에서는 CF-트리를 구성하고, 이를 데이터의 요약 정보로 삼아 원본 데이터가 필요 없이 요약 정보만으로 제한된 메모리 내에서 클러스터링을 수행한다. 또한 데이터의 양이 많아 CF-트리를 메모리 내에 구성할 수 없는 경우에는 CF-트리의 임계값(threshold)을 크게 하여 CF-트리를 재구성함으로 제한된 메모리 내에서 효율적인 클러스터링을 할 수 있다[5].

BIRCH의 CF-트리 노드는 다음과 같이 구성된다. CF-트리는 높이 균형 트리로 분기갯수(branching factor) B와 임계값 T의 파라매터를 가진다. 각 각의 내부(internal) 노드는 최대한 B개의 엔트리를 $[CF_i, child_i]$로 가진다. 여기에서 i=1,2, ..., B를 나타내며, $child_i$ 는 i번째 자식(child) 노드의 포인터를 의미한다. 단말(leaf) 노드는 최대 L개의 엔트리를 가지고 있으며 $[CF_i]$와 같은 형태를 지닌다. 여기에서 i=1,2, ... L 을 나타내며, 각 단말노드는 전자(prev)와 후자(next)의 포인터를 가지는데 이는 모든 단말 노드를 연결하여 빠른 데이터 검색에 사용하기 위해서다. 아래 [그림 1]은 B=3, L=4를 가지는 CF-트리의 예제를 나타낸다.

* 본 연구는 첨단정보기술연구센터를 통해 한국과학재단의 지원을 받았음



[그림 1.] CF-트리의 예제.

클러스터링 대상이 되는 데이터 객체가 d-차원을 갖는 경우, BIRCH에서 CF 노드는 [그림 2]와 같이 세 가지로 정의된다.

$CF=(N, \vec{LS}, SS)$ N= 클러스터의 데이터 개수,

$\vec{LS}$ = N개의 데이터의 합, $\sum_{i=1}^{n} \vec{X_i}$

SS= N개의 데이터의 제곱의 합 $\sum_{i=1}^{n} \vec{X_i}^2$

[그림 2.] CF 노드의 정의.

CF-트리를 구축하는 과정에서 메모리가 부족한 경우나 사용자에 의해서 CF-트리의 임계값을 증가시켜 CF-트리를 재구성 하는 경우가 존재하면 두 개의 클러스터 CF1과 CF2를 합병하며, 합병 식은 다음과 같다.

$$CF1 + CF2 = (N1 + N2, \vec{LS}_1 + \vec{LS}_2, SS1 + SS2)$$

[그림 3.] CF의 합병식.

구성된 CF-트리를 이용하여 클러스터링을 한 결과는 기존의 연구된 CLARANS[6]에 비해 시간이나, 입출력 면에서 뛰어난 성능을 나타낸다[5]. 이는 대용량의 데이터를 클러스터링 하는 경우에 기존 연구에 비해서 뛰어난 성능을 나타냄을 의미한다.

BIRCH이외에 기존에 클러스터링에 관련된 연구는 CURE[7], ROCK[8], CLARANS[6]등이 있으며 이 경우에는 샘플링을 사용하는 방법으로, 스트림환경에서는 샘플링 자체의 정확도를 높이기 어렵기 때문에 스트림환경에서는 사용하기 힘들고, DBSCAN[9]과 DP[10]는 밀도 기반 접근방법으로, 밀도 계산을 위해서는 모든 원본 데이터가 필요하고, 스트림 환경에서는 모든 데이터의 저장을 하는 것이 비효율적이므로 적용이 힘들다.

스트림 환경에 적용한 클러스터링 연구로 LOCALSEARCH[3]가 있으며, 이 연구에서는 데이터 집합을 저장하고, 저장된 데이터의 출현횟수에 따른 가중치를 주어, 클러스터링을 한다. 하지만 LOCALSEARCH는 시간구간에 대한 문제는 언급하지를 않고, 단지 클러스터링의 효율 면을 생각하였다. 또한 CluStream[4]의 경우에는 일정한 시간간격으로 CF를 저장하고, 이를 사용하는 연구가 진행되었다. 이 연구에서는 현재 저장되어 있는 CF와 임의 시간에 저장되어 있는 CF의 차이를 비교하는 연구로, 임의 시간 간격에 대한 연구가 아닌 현재와의 차이를 중점으로 둔 연구이며, 시간차이가 적은 것에 비해, 시간 차이가 많이 나는 경우에는 클러스터링의 정확성이 상대적으로 떨어진다.

3 데이터 스트림 클러스터링 프레임워크

본 논문에서는 임의 시간구간에 대한 클러스터링을 하기 위해서는 임의 시간 간격 마다 CF^6-트리를 저장하고, 저장된 여러 개의 CF^6-트리의 단말 노드를 사용하여 클러스터링을 수행하는 프레임워크를 제시한다. CF^6-트리란 기존의 CF-트리의 단말 노드를 수정한 트리이고, 저장되어 있는 단말 노드들을 이용하여 클러스터링 할 때 단말 노드들의 임계값을 동일한 수준으로 맞추기 위해 제안된 트리이다.

3.1 단순 접근 방법

임의 시간구간에 대한 클러스터링을 수행하는 단순한 방법으로는 저장되어 있는 다수의 CF-트리 단말 노드들을 가지고 클러스터링 하는 방법이 있다. 하지만 일반적으로 CF-트리들은 임계값이 다르며, 임계값이 다른 CF-트리의 단말 노드를 가지고 클러스터링을 수행하면 결과가 매우 부정확해 진다.

3.2 완전 복원 방법

본 논문에서 제안하는 완전 복원 방법은 CF-트리의 단말 노드에 추가 정보를 삽입하여, 합병되기 이전의 임계값으로 CF-트리를 복원하여, 동일한 임계값을 가진 CF-트리의 단말 노드들을 이용하여 클러스터링을 수행한다. 또한 CF-트리를 복원하기 위한 수정된 CF^6-트리를 제안하며, 완전 복원에 대한 추가 정보와 추가 정보를 줄여 근사적으로 복원하는 방안에 대해 제안한다.

임계값의 복원을 위하여 본 논문에서는 CF-트리의 단말 노드에 추가 정보를 넣어 트리를 구성하는데 이 트리를 CF^6-트리라 한다. CF^6-트리의 내부 노드는 CF-트리와 동일한 구조를 가지나, 단말 노드는 다음과 같은 구조를 가진다.

$CF^6=(N, \vec{LS}, SS, \delta)$

N= 클러스터의 데이터 개수,

$\vec{LS}$ = N개의 데이터의 합, $\sum_{i=1}^{n} \vec{X_i}$

SS= N개의 데이터의 제곱의 합 $\sum_{i=1}^{n} \vec{X_i}^2$

δ = CF^6-트리에서 이전 임계값으로 복원하기 위해 추가된 정보

[그림 4.] CF^6-트리의 구조.

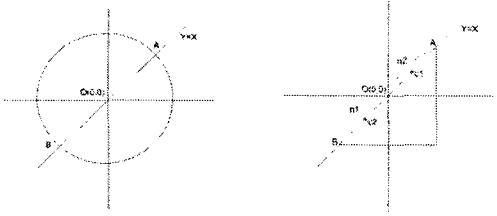
CF 값과 δ값으로 이전의 임계값으로 CF-트리를 복원하기 위한 방법은 다음과 같다. CF_{now}는 기존의 CF인 CF_{prev1}과 CF_{prev2}로 구성되다. $CF_{now}=(N, \vec{LS}, SS) = (N_{prev1}, \vec{LS}_{prev1}, SS_{prev1}) + (N_{prev2}, \vec{LS}_{prev2}, SS_{prev2})$의 관계가 성립된다. 그러면, 하나의 CF_{prev1}을 알 수 있다면, 다른 하나는 $CF_{now}-CF_{prev1}$로 구성이 가능하다. 즉, 완전한 복원을 위해서는 CF_{prev1}, 즉, N_{prev1}, $\vec{LS}_{prev1}$, SS_{prev1}을 δ로 구성해야 한다. 그런데 이때 δ에 포함되는 $\vec{LS}_{prev1}$가 벡터로 구성되어 있어, CF의 합병이 반복될수록 그 크기가 커지게 된다. 즉, 합병이 이루어진 횟수에 따라 최대 $\sum_{i=1}^{m}(2+d)*2^{i-1}$ (d= 차원, m=합병 횟수)개의 추가 데이터가 필요하다. δ의 값을 합병하기 이전 모든 데이터를 사용하면 추가되는 데이터의 개수가 너무 많고, 비효율적인 문제점이 발생한다.

3.3 근사적 접근 방법

본 논문에서 제안하고자 하는 근사적 접근 방법은 합병되기 이전의 중점 간을 이은 선분의 비율을 데이터의 개수 비율로 하여 합병되기 이전의 두 클러스터의 중점을 근사적으로 찾는 방법이다. 이를 위해서 합병된 클러스터의 중점과 합병되기 이전의 두 클러스터의 중점이 항상 y=x선분 위에 고정되어 있다고 가정하였다.[그림 5.]에서와 같이 클러스터 A와 클러스터B가 합병되어 클러스터 C를 생성했다고 가정하고, [그림 6.]과 같이 이 클러스터 C의 중점이 원점이 된다고 가정 한다 또한 합병되기 이전의 클러스터 A와 B의 중점은 y=x를 지나는 선분 위에 있다고 가정하고 이 선분과 클러스터C의 교점의 좌표를 점 A, 점 B라 하면 합병되기 전의 클러스터 A와 클러스터 B의 중점은 [그림 7.]에서 점 c1, c2를 나타낸다고 할 때, 데

이터의 개수 n1,n2의 비율을 이용하여 근사적으로 점c1과 점 c2를 찾는 방법은 아래와 같다.

[그림 5.] 클러스터의 합병.



[그림 6.]합병된 클러스터. [그림 7.] 합병 전, 후의 중점 관계.

[그림 7.]은 [그림 6]의 선분 AB 와 y=x와의 관계와 선분 BO와 선분 OA의 중점인 c1,c2를 나타낸 그림으로 n1과 n2는 선분 AB에서의 비율을 나타낸다. 이를 이용하여 이전 클러스터 의 중점 c2를 구하는 식은 다음과 같다.

점 c2의 x 좌표 = x - r*n1 / (n1+n2)*$2^{1/2}$
점 c2의 y 좌표 = y - r*n1 / (n1+n2)*$2^{1/2}$
x, y : x, y 좌표
r : 반지름의 길이
n1 : 클러스터 A의 데이터의 개수
n2 : 클러스터 B의 데이터의 개수

[그림 8.] (x, y)에서의 점 c2의 좌표.

위에서 구한 식을 n차원에 적용하여 점 c2의 좌표를 일반화한 식은 다음과 같다. [그림 9.]의 일반화 식을 이용하면, 합병되기 이전의 클러스터로 복원이 가능하다.

이를 일반화하여 n차원에 적용하는 경우에는 분면 bit 좌표를 이용하여 n차원의 di값을 계산할 수 있으므로, 아래 그림과 같이 정의될 수 있다.

점 c2의 i차원 좌표 = x_i + $(-1)^{di-1}$ * r*n1 / (n1+n2)*$n^{1/2}$
di: bit 좌표의 i자리 수 (i=1,2,...,n)
xi : xi 좌표
r : 반지름의 길이
n1 : 클러스터 A의 데이터의 개수
n2 : 클러스터 B의 데이터의 개수

[그림 9.] n차원에서 점 c2의 좌표.

위 식을 이용하면, 추가 데이터는 기존의 n1값과 di값을 이용하여 근사적으로 이전의 클러스터 c1과 c2의 추정이 가능하다.

4. 결론 및 향후 연구과제

본 논문에서는 데이터 스트림 환경에서 과거의 임의 시간간격에 대한 클러스터링 알고리즘을 제안하였다. 기존에 대용량 데이터의 클러스터링 알고리즘인 BIRCH는 스트림 환경에 적용하는 경우, 과거의 임의 시간 간격에 대한 클러스터링을 하지 못하는 단점이 있다. 본 논문에서는 이러한 단점을 해결하기 위해 임의 시간단위의 CF^6-트리를 저장하고 이들을 사용하여 임의 시간간격에 대한 클러스터링을 하는 프레임워크를 제안하였다. 제안한 프레임 워크를 다수의 CF^6-트리를 사용하는 방법으로 병합되기 이전의 모든 데이터를 추가 데이터로 사용하는 단순 접근법을 먼저 제시하였다. 다음으로 추가 데이터를 줄이는 접근 방법인 근사 접근법을 제안하였다. 본 논문에서 제안한 근사적 접근법은 아주 적은 양의 정보(δ)를 추가함으로써, 과거의 임의 시간 구간에 대한 클러스터링을 보다 효율적으로 수행할 수 있다고 사료된다. 향후 연구에서는 제안한 접근법이 실제로 정확하게 클러스터링을 수행하며, 실용적으로 사용할 수 있는지 실험을 통해 입증한다.

5. 참고문헌

[1] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, Jennifer Widom., "Models and Issues in Data Stream Systems," *Proc. of the 22th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Madison, Wisconsin, USA, pp. 1-16, June 2002.

[2] Yunyue Zhu, Dennis Shasha., "StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time," *Proc. of the 28st Int'l Conf. on Very Large Data Bases*, HongKong China, pp. 358-369, Aug. 2002.

[3] Charu C. Aggarwal., "A Framework for Change Diagnosis of Data Streams," *Proc. of the ACM SIGMOD Int'l Conf.* San Diego, California, pp. 575-586, June 2003.

[4] Liadan O'Callaghan, Adam Meyerson, Rajeev Motwani, Nina Mishra, Sudipto Guha., "Streaming-Data Algorithms for High-Quality Clustering," *Proc. of the 18th Int'l Conf. on Data Engineering*, San Jose, California, USA, pp. 685-697, Feb. 2002

[5] Tian Zhang, Raghu Ramakrishnan, Miron Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," *Proc. of the ACM SIGMOD Int'l Conf.*, Montreal, Canada, pp. 103-114, June, 1996

[6] Raymond T. Ng, Jiawei Han, "CLARANS: A Method for Clustering Objects for Spatial Data Mining," *IEEE Trans. Knowl. Data Eng.*, Vol. 14, No 5, pp. 1003-1016, Sept./Oct. 2002

[7] Sudipto Guha, Rajeev Rastogi, Kyuseok Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," *Proc. of the ACM SIGMOD Int'l Conf.*, Seattle, Washington, pp. 73-84, June, 1998

[8] Sudipto Guha, Rajeev Rastogi, Kyuseok Shim, "ROCK: A Robust Clustering Algorithm for Categorical Attributes," *Proc. of the 15th Int'l Conf. on Data Engineering*, Sydney, Australia, pp. 512-521, March, 1999

[9] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. of the second Int'l Conf. on Knowledge Discovery and Data Mining*, Portland, Oregon, pp. 226-231, March, 1999

[10] Jae-Joon Hwang, Kyu-Young Whang, Yang-Sae Moon, and Byung-Suk Lee, "A Top-down Approach for Density-Based Clustering Using Multidimensional Indexes," *Journal of Systems and Software*, Vol. 73, Issue 1, pp. 196-180, Sept. 2004.