데이터 클러스터링에서 클러스터 수 결정방안

이병수*, 홍지원*, 김상욱*
*한양대학교 컴퓨터공학과
e-mail:leebs1986@hanyang.ac.kr {nowiz, wook}@agape.hanyang.ac.kr

A Method for Determining the Number of Clusters in Data Clustering

Byung-Soo Lee* Jiwon Hong* Sang-Wook Kim*
*Department of Computer Engineering, Hanyang University

요 약

데이터마이닝 분야에서는 주어진 공간상에 분포되어있는 데이터들을 분석위해 다양한 클러스터링 알고리즘이 존재한다. 그러나 대부분의 클러스터링 알고리즘에서는 클러스터 전체 개수를 미리 요구한다. 이 때문에 클러스터링 알고리즘에서 클러스터 전체개수를 미리 알아내는 것은 매우 중요하다. 본 논문에서는 데이터에 분포하는 클러스터들의 개수를 데이터의 그래프 모델을 이용한 분석으로 찾아내는 방법을 제안한다.

1. 서론

최근 거대하게 쌓이는 데이터를 분석하는 작업이 중요해지면서 클러스터링 분야에 많은 연구가 이루어지고 있다. 그러나 클러스터링을 효과적으로 수행하기 위해서는 해당 데이터에 존재하는 클러스터의 총 개수를 미리 아는 것이 매주 중요하다. 본 논문에서는 주어진 데이터에 존재하는 클러스터의 총 개수를 파악하는 방안을 제안한다.

2. 제안 알고리즘

주어지는 데이터는 n-차원 공간상에 위치한 객체들의 집합으로 구성된다. 클러스터란 공간상 비슷한 위치에 모여 있는 객체들을 묶은 것으로, 본 논문에서 제안하는 방법에서는 데이터 내의 클러스터들의 개수를 파악하기 위해 k-최인접 그래프(k-NNG)[1]와 connected component [2]가 사용된다.

제안하는 알고리즘은 반복적으로 수행되며 k번째 단계에서 모든 데이터 객체에 대해 그의 k-최인접 객체로 단방향 간선을 부여한다[3]. 이 중 상호간에 연결을 지닌 객체들 간의 간선(reciprocated edge)[1]만을 남기고 나머지를 제거한다. 결과적으로 k번째 단계마다 데이터 객체와 간선으로 구성된 무향 그래프(undirected graph)가 형성되고 이것이 k번째 반복의 k-NNG가 된다.

k-NNG를 형성한 뒤에는 이 그래프 안의 connected component들을 파악한다. 모든 connected component들 중에서 전체 데이터 객체 수의 1% 이상이 되는 데이터 객체수를 가지는 것들을 하나의 클러스터인 것으로 간주한다. 알고리즘이 반복되면서 클러스터로 취급되는 connected component들의 개수가 이전 단계와 같을 경우 알고리즘을 종료하고 클러스터의 개수를 산출한다.

제안하는 방법은 상호 인접한 connected component들을 하나로 합치는 확장을 반복하고, 더 이상 주변에 상호 인접한 connected component가 없을 때 알고리즘을 종료하고, 이 때 존재하는 일정 크기 이상의 connected component들의 수를 클러스터의 수로 사용한다. 그러나 이 방법은 데이터의 특성에 크게 영향을 받아 다음과 같은 한계점을 지닌다.

한계점1. 모든 데이터 객체를 대상으로 알고리즘을 적용하게 되면, 데이터 객체들이 밀집된 지역과 그렇지 않은 지역에 형성되는 connected component간에 확장속도 차이가 발생함

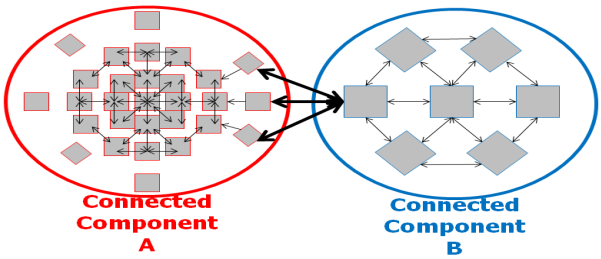
한계점2. 특정 connected component의 주변에 노이즈 객체가 존재할 경우 해당 connected component는 주변 노이즈 객체로도 확장을 계속하는 현상이 발생함

제안하는 알고리즘에서는 독립적으로 존재할 두 클러스터가 한계점 1과 한계점 2에 의해 하나의 connected component에 포함되는 문제점이 발생할 수 있다. 따라서 각각의 클러스터의 의미가 상실되어 정확한 결과를 얻을 수 없다.

3. 개선 방안

한계점 1은 제안하는 알고리즘이 각 데이터 객체 주변의 밀집도에 관계없이 모든 데이터 객체에 대해 그들의 k-최인접 객체로 단방향 간선을 연결하면서 발생한다. k번째 단계에서 데이터 객체 중 일부는 k-최인접 객체와 상호간 연결을 가지지 못하지만 이런 데이터 객체들도 다음 $k+1$번째 단계 다시 $(k+1)$-최인접 객체로 단방향 간선을 추가하여 상호 연결될 경우 그대로 간선으로 사용한다. 이 과정으로 k-최인접 객체와는 상호연결이 아니지만,

(k+x)-최인접 객체와는 상호연결이 되는 현상이 발생한다.



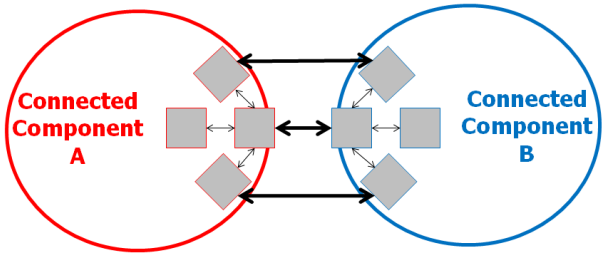
(그림 1) 한계점 1에 의한 악영향

결과적으로 그림 1과 같이 데이터 객체의 밀집도가 서로 다른 두 지역은 알고리즘을 반복적으로 수행하는 과정에서 외곽 데이터 객체들 중 일부가 서로 다른 connected component와 연결되어 서로 다른 두 클러스터가 하나로 합쳐질 수 있다.

이러한 현상을 막기 위해서는 알고리즘의 k번째 단계에 k-최인접 객체와 상호간 연결을 가지지 못하는 객체는 다음 k+1번째 단계에서는 (k+1)-최인접 객체와의 간선을 만들지 않고 k-최인접 객체 상호연결을 기다려야 한다. 이렇게 함으로써 밀집도가 서로 다른 두 클러스터가 병합되는 문제를 해결할 수 있다.

한계점 2는 데이터에 노이즈 객체가 존재할 경우 특정 connected component에 속한 데이터 객체와 노이즈 객체 간에 상호연결이 발생했을 때, 해당 connected component가 노이즈를 통해 지속적으로 확장되어 다른 connected component와 상호 연결되는 문제를 낳는다.

한계점 2에 의한 악영향을 막기 위해서 서로 다른 connected component를 상호 연결하는 간선을 삽입할 때에는 제약을 두어 조건을 만족하지 못하면 간선을 삽입하지 않는 방법을 사용한다.



(그림 2) 두 connected component간의 Bridge-Edge

정의 1. Component의 평균 간선 값
Connected component의 (내부간선개수)/(내부객체개수)

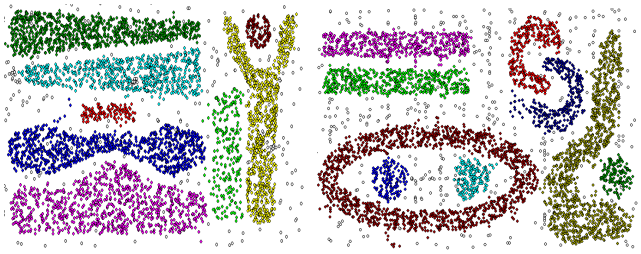
정의 2. Bridge-Edge
해당 간선을 삽입 하면, 서로 다른 connected component에 포함된 두 객체를 연결 하게 될 간선 (그림 2)

조건 1. Bridge-Edge를 삽입할 때, 해당 Bridge-Edge가 연결하는 두 개의 connected component 사이에 존재하는 총 Bridge-Edge의 개수가 두개의 connected component 평균 간선 값 이상이 되어야 허용

조건 1을 이용하면 Bridge-Edge가 충분히 형성되기 전에 두 connected component가 상호 연결되어 하나의 connected component가 되는 것을 막을 수 있기 때문에 노이즈 객체에 의한 확장 문제가 발생하지 않는다.

4. 실험

제안 알고리즘을 실험 해보기 위해 CHAMELEON에서 사용된 데이터 중 8000개와 10000개로 이루어진 두 종류의 데이터를 사용했다. 실험의 수행 속도를 고려하여 임의 샘플링 을 통해 5000개의 데이터 객체만을 추출하여 실험을 수행하였다.



(그림 3) 제안 알고리즘 결과 화면

실험을 통해 그림 3과 같이 제안하는 방법과 그 개선 방안을 이용하여 각각의 데이터에 존재하는 8개와 9개의 클러스터 개수를 찾아낼 수 있음을 확인할 수 있다.

5. 결론

본 논문에서는 k-NNG를 이용하여 주어진 데이터에 존재하는 클러스터의 개수를 찾는 알고리즘을 제시 하였으며, 이 알고리즘이 가지고 있는 한계점을 극복 하기위해 2가지의 개선방안도 제시하였다. 최종적인 실험과 그 결과를 통해 제안하는 알고리즘과 그 개선 방안을 이용하여 존재하는 클러스터의 총 개수를 파악할 수 있음을 보였다.

감사의 글

본 연구는 지식경제부 및 정보통신산업진흥원의 IT융합 고급인력과정 지원사업의 연구결과로 수행되었음(NIPA-2011-C6150-1101-0001)

참고문헌

[1] M. R. Britoa, E. L. Chávezb, A. J. Quiroz, and J. E. Yukichd, "Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection," *Statistics & Probability Letters*, 1997.

[2] D. B. West, *Introduction to graph theory*, 1996.

[3] Ravi Kumar and Jure Leskovec and Andrew Tomkins, "ShatterPlots: Fast Tools for Mining Large Graphs," *SIAM*, 2009.