삼각 부등식을 활용한 공군집화 알고리즘의 가속화

정상원[1], 황지영[2†]
[1]성균관대학교 수학과
[2]성균관대학교 컴퓨터공학과
{s.jung, jjwhang} @skku.edu

Work-Efficient Co-Clustering Algorithm via Triangle Inequality

Sangwon Jung[1], Joyce Jiyoung Whang[2‡]
[1]Department of mathematics, Sungkyunkwan University
[2]Department of Computer Science and Engineering, Sungkyunkwan University

요 약

공군집화(Co-Clustering)의 목적은 이차원 데이터 행렬의 행 군집화와 열 군집화를 동시에 진행하는 것이다. 기존의 공군집화 알고리즘은 반복이 일어날 때마다 각각의 점(data point)과 군집 사이의 거리를 모두 계산해야 한다. 하지만 알고리즘이 어느 정도의 반복을 거치고 나면 대부분의 점들은 한 군집에서 다른 군집으로 이동하지 않게 되므로 점과 군집 사이의 거리 계산 중 많은 부분을 생략할 수 있다. 이 논문에서는 삼각부등식을 활용하여 기존의 공군집화 알고리즘보다 개선된 공군집화 알고리즘을 제시한다. 또한 실험적 결과를 통해서 개선된 알고리즘이 공군집화 알고리즘의 질적 성능을 저해하지 않으면서 계산량을 대폭 감소시키는 것을 보여준다.

1. 서 론

군집화(Clustering)는 레이블(Label)이 없는 데이터에 대해서 데이터의 특성을 고려해서 데이터를 분류하는 비지도 학습 중 하나이다. 이 때, 데이터의 형태, 크기, 의미 등 여러 가지 상황에 따라 다른 군집화 알고리즘을 적용한다. 그 중 공군집화(Co-Clustering)는 군집화의 대상을 나타내는 행과 그들의 특성이나 특징을 나타내는 열을 가진 2차원 데이터 행렬에 적용할 수 있다. 이러한 형태의 데이터에 대해서 k-means 알고리즘과 같이 행이나 열 방향에서만 군집화를 할 수도 있지만, 행과 열이 관련이 있다면 행과 열에서 동시에 군집화를 할 때 더 의미 있는 결과를 도출할 수 있다. 실제로 공군집화는 유전자-표현 데이터[1], 영화-평점 데이터나 단어-문서 데이터 분석에서 높게 활용되고 있다. 이에 따라 좀 더 효율적이고 좋은 의미를 내는 많은 공군집화 알고리즘이 제시되었는데 공군집화 알고리즘의 속도를 향상시키는 부분에서는 연구의 발전정도가 미약하다. 따라서 이 논문에서는 공군집화 알고리즘의 속도를 향상시키는 방법을 제시하고자 한다.

기존의 k-means 군집화 알고리즘은 데이터마다 자신의 군집 중심과의 거리의 합이 최소가 [1]되도록 하기 위해서 Lloyd 알고리즘을 사용한다. 이 알고리즘은 데이터 점들을 자신과 가장 가까운 군집에 할당하고 군집의 중심을 재계산하는 과정을 반복한다. 이 과정에서 매번 모든 점들과 모든 군집 사이에 거리를 계산하게 되는데 이 계산 중 많은 부분들이 불필요하다. 왜냐하면 한 점과 어떤 군집 사이의 거리가 명확하게 멀다면 그 둘 사이의 거리는 계산할 필요가 없기 때문이다. 또한 한 점과 어떤 군집 사이의 거리가 다른 군집들에 비해 매우 가깝다면 그 점은 그 군집에 속하는 것이 명확하고 그 점에 대한 계산은 불필요하다. k-means 알고리즘은 이러한 특징을 활용해서 다양한 방법으로 성능의 향상이 가능하고 특히 [2]에서는 삼각부등식을 활용하여 계산량을 효율적으로 줄였다.

이 논문에서는 앞서 말한 특징을 공군집화 알고리즘에 적용하여 공군집화 알고리즘의 속도를 향상시킨다. 즉, 공군집화 알고리즘 또한 앞서 말한 k-means 알고리즘의 특성을 가지고 있기 때문에, norm의 삼각부등식을 이용해서 알고리즘의 결과는 그대로 유지하면서 속도를 향상시키는 것이다. 그리고 이렇게 제시한 알고리즘이 기존의 공군집화 알고리즘보다 거리를 구하는 횟수와 실행 시간 면에서 월등히 개선되었음을 확인한다.

2. 삼각부등식 활용 방법

2.1 공군집화 목적 함수

이 논문에서는 가장 보편적으로 많이 쓰이는 MSSR 공군집화 목적 함수[3]를 사용한다. 공군집화 알고리즘에서 데이터 행렬 $X \in R^{n \times m}$가 주어졌을 때 X^r을 행 군집을 위한 점의 집합, X^c를 열 군집을 위한 점의 집합이라고 하자. 그러면 공군집화의 목표는 $X^r = \{x_1, \cdots, x_n\}$ ($x_i \in R^m$ for $i = 1, \cdots, n$)를 k개의 행 군집 $C_1^r, C_2^r, \cdots, C_k^r$와 $X^c = \{x_1, \cdots, x_m\}$ ($x_j \in R^n$ for $j = 1, \cdots, m$)를 l개의 열 군집 $C_1^c, C_2^c, \cdots, C_l^c$로 군집화하는 것이다. 이 때 행의 중

† 교신저자 (Corresponding author)

이 논문은 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2016R1D1A1B03934766, NRF-2010-0020210).
또한, 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학지원사업의 연구결과로 수행되었음(2015-0-00914).

심 행렬을 $M \in R^{k \times m}$, M의 i번째 행을 $M_i \in R^m$, 열의 한 점 x_j이 속한 군집을 $C^c_{c(x_j)}$라고 한다면 목적함수는 다음과 같다.

$$\min_{C^r, C^c} \sum_{i=1}^{k} \| x_i - M_i \|_2 \quad where\ M_{ij} = \frac{\sum_{x_s \in C^r_i} \sum_{x_t \in C^c_{c(x_j)}} X_{st}}{\left| C^r_i \right| \times \left| C^c_{c(x_j)} \right|}$$

2.2 상계(Upper Bound)를 활용한 계산량 감소

행과 열의 각 점들은 알고리즘이 t번 반복했을 때, 자신이 속해있는 군집과의 거리에 대해 상계를 가지고 있고 그 값을 가지는 벡터를 $U^t_r \in R^n, U^t_c \in R^m$ 라고 하자. $c(x_i)$를 행의 점 x_i가 현재 속해있는 군집번호라고 하면 상계는 다음과 같이 갱신된다.

$$U^{t+1}_r = U^t_r + \| M^t_{c(x_i)} - M^{t+1}_{c(x_i)} \|$$

우리는 점과 군집 사이에 거리를 구하기 위해 norm을 사용하는데 norm은 삼각부등식을 만족한다. norm의 삼각부등식을 사용하면 다음과 같은 식이 성립한다.

$$If\ \| x_i - M^{t+1}_j \| \le \frac{1}{2} \| M^{t+1}_s - M^{t+1}_t \| \text{ for } s, t = 1, \cdots, k$$

$$M^{t+1}_j = \arg min_{M^{t+1}_s} \| x_i - M^{t+1}_s \| \qquad (1)$$

그러면 각 점 x_i에 대해서 $\min_j \| x_i - M^{t+1}_j \| \le (U^{t+1}_r)_i$가 성립하므로 (1) 식과 결합하면 우리는 다음과 같은 식을 만족하는 점 x_i는 해당 iteration에서 거리를 계산하지 않고 할당되어 있던 군집에 계속 머문다.

For each iteration t and each point x_i

$$(U^{t+1}_r)_i \le \min_{i,j} \frac{1}{2} \| M^{t+1}_j - M^{t+1}_i \|$$

2.3 하계(Lower Bound)를 활용한 계산량 감소

행과 열의 각 점들은 각 군집과의 거리에 대해 하계를 가지고 있고 그 값을 가지는 벡터를 $L_r \in R^{n \times k}, L_c \in R^{m \times l}$라고 하자. 행의 점 x_i와 어떤 군집 C_j에 대해 하계는 다음과 같이 업데이트된다.

$$(L^{t+1}_r)_{ij} = \max \left\{ 0, (L^t_r)_{ij} - \| M^t_j - M^{t+1}_j \| \right\}$$

또한, 만약 $(L^{t+1}_r)_{ij} \neq 0$이라면 삼각부등식을 사용하면 다음과 같은 식을 얻는다.

$$\| x_i - M^{t+1}_j \| \ge \| x_i - M^t_j \| - \| M^{t+1}_j - M^t_j \| \ge (L^{t+1}_r)_{ij}$$

따라서 c(x_i)를 현재 x_i가 속해있는 군집 번호라고 하면 다음 두 개의 부등식 중 하나라도 만족하면 해당 iteration에서 x_i와 C_j사이의 거리는 계산할 필요가 없다.

$$(L^{t+1}_r)_{ij} \ge \| x_i - M^{t+1}_{c(x_i)} \|$$

$$\frac{\| M^{t+1}_{c(x_i)} - M^{t+1}_j \|}{2} \ge \| x_i - M^{t+1}_{c(x_i)} \|$$

3. 계산 효율적인 공군집화 알고리즘 (Work-Efficient Co-Clustering Algorithm)

알고리즘

1. Initialize row assignment c(x_i) for all i and column assignment c(x_j) for all j. Set U^0_r, U^0_c to and set L^0_r, L^0_c to 0.
2. For each iteration t, update row clusters
   1) Calculate row center matrix M^t.
   2) Update U^t_r and L^t_r for each x_i and C_j such that
   $$(U^t_r)_i = (U^{t-1}_r)_i + \| M^{t-1}_{c(x_i)} - M^t_{c(x_i)} \|$$
   $$(L^t_r)_{ij} = \max \left\{ 0, (L^{t-1}_r)_{ij} - \| M^t_j - M^{t-1}_j \| \right\}$$
   3) Calculate the distance $\| x_i - M^t_{c(x_i)} \|$ for all x_i
   4) Identify all row points x_i such that
   $$(U^t_r)_i \le \frac{1}{2} min_{i \neq j} \| M^t_i - M^t_j \|$$
   5) For each cluster C_j and all remaining x_i,
      If $\| x_i - M^t_{c(x_i)} \| > (L^t_r)_{ij}$ and
      $\frac{\| M^t_{c(x_i)} - M^t_j \|}{2} < \| x_i - M^t_{c(x_i)} \|$,
      Calculate $\| x_i - M^t_j \|$
      Update $(L^t_r)_{ij} = \| x_i - M^t_j \|$
      If $\| x_i - M^t_j \| < \| x_i - M^t_{c(x_i)} \|$,
         Update $(U^t_r)_i = \| x_i - M^t_j \|$ and $c(x_i) = j$
3. Update column clusters in the same way
4. Repeat 2-4 until converged

우리는 기존의 공군집화 알고리즘에서 상계와 하계 값을 구하고 이를 통해서 거리 계산이 필요 없는 부분을 생략할 수 있었다. 이 알고리즘이 기존의 것보다 효율적인 이유는 초기 몇 번의 반복을 거치고 나면 상계와 하계가 대부분의 x_i와 C_j의 거리에 대해서 타이트하기 때문이다.

단계1에서 행 데이터 점들과 열 데이터 점들을 초기화하는 것은 행과 열 각각에 대해 k-means++알고리즘을 실행하여 진행하였다. [2]에서는 단계2.1 ~ 2.2가 단계 2.5 이후에 진행되었는데 공군집화에서는 행 군집화가 끝나면 열 군집화가 시작되기 때문에 다음번 행 군집의 중심을 구하는 것은 열 군집화가 끝난 다음이 되어야 한다.

위에 제시한 알고리즘은 기존의 공군집화 알고리즘에서 상계와 하계를 더해서 계산량만 줄인 것이다. 즉, 기존의 공군집화 알고리즘은 MSSR을 목적함수로 하여 행과 열에 각각 Lloyd 알고리즘을 적용하는 방식으로 진행되는데, 제시된 알고리즘에서는 그 과정에서 삼각부등식

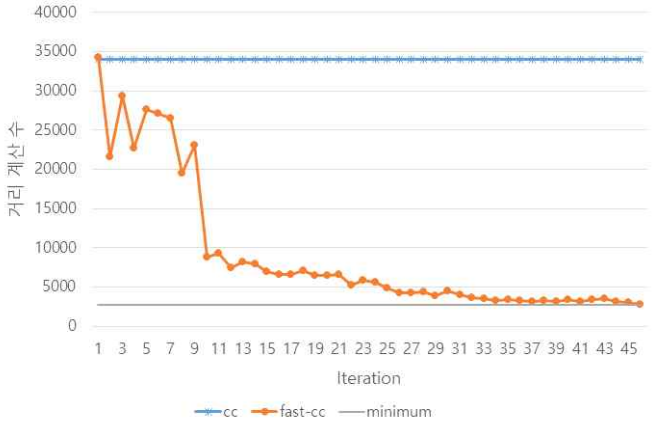

그림 1 (k, l) = (14, 2)일 때, 거리 계산 수 비교

을 통해 점과 군집사이의 거리 계산이 생략될 수 있는 부분을 탐색하고 제거하는 부분이 추가되었다.

4. 실험 결과

이 연구에서는 새로 제시한 알고리즘을 [4]에서 제공한 yeast gene expression 데이터에 적용한다. 이 데이터에서 행은 유전자를 열은 특수한 생물학적 환경에서 나타나는 표현 정도를 나타낸다. 데이터는 총 2,417 행과 103개의 열이 있으며 행은 14개로 분류되어진 ground-truth가 있다.

실험은 기존의 공군집화 알고리즘과 이 논문에서 제시된 공군집화 알고리즘을 데이터에 각각 적용했을 때, 매 반복에서 점과 점 사이의 거리를 구하는 수가 얼마나 되는지 확인하였다. 또한, 두 알고리즘의 실행시간을 비교해서 계산량을 줄이는 것뿐만 아니라 실제 실행시간도 줄일 수 있는지 확인하였다. 그리고 k, l 값을 바꿔가면서 이 알고리즘이 k, l 값에 민감하지 않게 향상된 결과를 보이는지 확인하였다.

그림 1과 같이 기존의 공군집화는 알고리즘이 반복할 때 마다 상당한 량의 거리를 계산해야 하는 반면, 삼각부등식을 이용한 공군집화 알고리즘은 10번 반복하고 나면 거리 계산 수가 빠르게 떨어지는 것을 볼 수 있다 (그림 1에서 cc는 기존의 공군집화 알고리즘을 나타내고, fast-cc는 본 논문에서 제안하는 알고리즘을 나타낸다). 매번 목적함수 계산을 위해 약 2,500번의 계산은 불가피한 것을 감안했을 때, 알고리즘 후반에는 불필요한 계산이 거의 없음을 확인할 수 있다.

그림 2는 행과 열의 군집 개수를 다르게 해줄 때 두 알고리즘의 실행속도를 비교한 그래프이다. 기존의 공군집화 알고리즘은 클러스터 개수가 많아지면 매 반복에서 구해야 하는 거리의 개수가 많아진다. 하지만 삼각부등식을 사용한 공군집화 알고리즘은 군집의 개수에 민감하게 반응하지 않는 것을 확인할 수 있다.

표 2에서는 cc와 fast-cc 알고리즘의 최종 목적 함수 값을 나타낸다. 두 알고리즘의 목적함수 값이 같은 것을 확인할 수 있으며, 이는 두 알고리즘의 공군집화 형태가

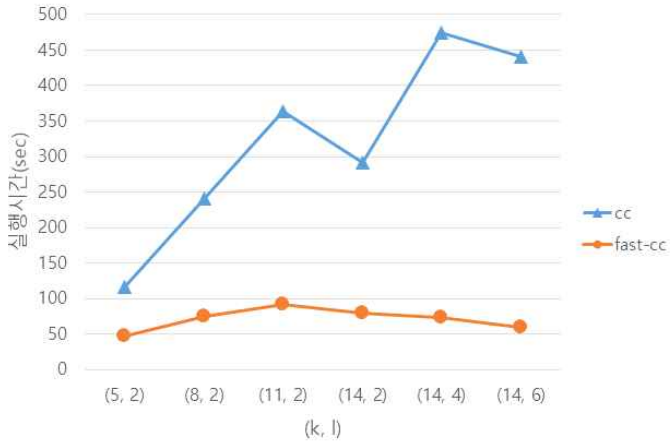

그림 2 군집 개수 별 실행시간 비교

표 2 군집 개수 별 목적 함수 값 비교

	(5,2)	(8,2)	(11,2)	(14,2)	(14,4)	(14,6)
cc	4482.0	4467.8	4462.5	4459.3	4205.3	4040.9
fast-cc	4482.0	4467.8	4462.5	4459.3	4205.3	4040.9

같음을 보여준다. 즉, fast-cc 알고리즘이 기존 공군집화 알고리즘의 질적 성능을 저해하지 않으면서 상당량의 계산을 줄일 수 있다는 것을 보여준다.

5. 결론 및 향후연구

삼각부등식을 활용한 공군집화 알고리즘은 기존의 공군집화 알고리즘의 군집 형태를 그대로 유지하면서 계산량을 월등히 줄일 수 있다. 또한 군집의 개수와 같은 파라미터의 변화에 민감하지 않고 비교적 일정하게 좋은 성능을 유지한다. 앞에서 언급했던 기존의 공군집화 알고리즘은 exhaustive하고 non-overlapping 군집화이다. 하지만 실제 데이터에는 [5]와 같이 군집이 겹치는 것을 허용하고 outlier 데이터를 고려해야 좀 더 좋은 군집화가 이루어진다. 따라서 향후 우리는 NEO-CC에 대해서도 삼각부등식 등을 적용하여 성능을 향상시킬 것이다.

6. 참고문헌

[1] G. Pio, M. Ceci, D. Malerba, and D. D' Elia. "ComiRNet: A web-based system for the analysis of miRNA-gene regulatory networks," In *BMC Bioinformatics,* 2015.

[2] C. Elkan. "Using the triangle inequality to accelerate k-means," In *ICML,* 2003.

[3] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. "Minimum Sum-Squared Residue Co-clustering of Gene Expression Data," In *SDM,* 114-125, 2004.

[4] A. Elissee. and J. Weston. "A Kernel Method for Multi-Labelled Classification," In *NIPS,* 681-687, 2001.

[5] J. J. Whang and I. S. Dhillon. "Non-Exhaustive, Overlapping Co-Clustering," In *CIKM,* 2017.