클러스터링 군집화 개요(수고 측정 후보 1)

2020.7.20~2020.7.26

클러스터링(군집화)는 개체들이 주어졌을 때, 개체들을 몇 개의 클러스터(부분 그룹)으로 나누는 과정 의미. Unsupervised learning(비지도 학습)의 대표적인 예이다.

개체들을 그룹으로 나누는 과정을 통해 클러스터 내부 멤버들 사이는 서로 가깝거나 비슷하게, 서로 다른 두 클러스터 사이의 멤버 간에는 서로 멀거나 비슷하지 않게 하는 것이 클러스터링의 목표

* 왜 수고 측정의 방법 후보로 설정했는가?
* 나무의 3차원 포인트 클라우드를 구성 했을 때 겹치는 부분이 없으면 각 개체목의 최대 높이를 구하면 되지만 나무의 모양은 제각각 이므로 겹치는 부분이 생긴다면 각 개체목의 최대 높이를 구별하기 힘들다.

위와 같은 이유로 각 개체목의 구성 포인트들을 군집화를 한 뒤 최대한 개체목을 구분을 한 후 높이를 구하는 방식으로 해결해 보려는 시도를 계획.

* 개체목의 좌표는 3차원 좌표이지만 군집화 개념을 설명 하기 위해 2차원 좌표로 개념 설명 및 이해.

만약 개체목들을 거리공간 안에 나타낼 수 있다면 개체와 개체 사이의 거리를 정의 가능하며 예를들어 유클리드 공간안에 개체목을 표현 가능하다면 유클리드 거리를 정의 가능. 이러한 경우 클러스터링의 목표는 같은 클러스터 내의 두 멤버들 사이의 거리를 최소화하고 서로 다른 두 클러스터 사이의 멤버 간의 거리를 최대화.

거리 공간은 공간을 나타내는 집합 X와 거리 함수 d:X×X→[0,∞)로 이루어져서, 두 점 사이의 거리가 정의된 공간을 의미합니다. d가 거리 함수가 되려면, 아래와 같은 조건들을 만족해야 합니다.

1. 임의의 x,y∈X에 대하여, d(x,y)=0과 x=y는 동치여야 합니다.
2. 임의의 x,y∈X에 대하여, d(x,y)=d(y,x)를 만족하여야 합니다.
3. 임의의 x,y,z∈X에 대하여, d(x,y)+d(y,z)≥d(z,x)를 만족하여야 합니다. 이 부등식은 주로 삼각부등식으로 잘 알려져 있습니다.

**점인 경우에는 유클리드 거리, Lp norm 등의 거리를 정의**할 수 있고, 집합의 경우에는 [자카드 지수](https://en.wikipedia.org/wiki/Jaccard_index) (교집합의 크기 / 합집합의 크기)를, 그리고 문자열의 경우에는 [편집 거리](https://en.wikipedia.org/wiki/Edit_distance)나 LCS(최대 부분 공통 문자열)의 길이를 이용하여 거리를 표현할 수 있습니다. <출처: 삼성 소프트웨어 멤버십 클러스터링(군집화) 개요>

\*유클리드 거리

**유클리드 거리**(Euclidean distance)는 [두 점 사이의 거리](https://ko.wikipedia.org/wiki/%EB%91%90_%EC%A0%90_%EC%82%AC%EC%9D%B4%EC%9D%98_%EA%B1%B0%EB%A6%AC)를 계산할 때 흔히 쓰는 방법이다. 이 거리를 사용하여 [유클리드 공간](https://ko.wikipedia.org/wiki/%EC%9C%A0%ED%81%B4%EB%A6%AC%EB%93%9C_%EA%B3%B5%EA%B0%84)을 정의할 수 있으며, 이 거리에 대응하는 [노름](https://ko.wikipedia.org/wiki/%EB%85%B8%EB%A6%84)을 유클리드 노름(Euclidean norm)이라고 부른다.

[직교 좌표계](https://ko.wikipedia.org/wiki/%EC%A7%81%EA%B5%90_%EC%A2%8C%ED%91%9C%EA%B3%84)로 나타낸 점 **p** = (*p*1, *p*2,..., *pn*)와 **q** = (*q*1, *q*2,..., *qn*)가 있을때, 두 유클리드 노름을 이용하여 두 점 **p**, **q**의 거리를 계산하면 다음과 같다.

<출처: 나무 위키>

‖p−q‖=(p−q)⋅(p−q)=‖p‖2+‖q‖2−2p⋅q.

\*유클리드 노름

n차원 유클리드 공간(=좌표평면)에서의 노름은 **유클리드 노름**(Euclidean norm)이라고 부르고 다음과 같이 정의한다.

∥x∥2=x12+x22+⋯+xn2=∑k=1nxk2∥**x**∥2​=*x*1​2+*x*2​2+⋯+*xn*​2​=*k*=1∑*n*​*xk*​2​

lp 노름에서 p=2의 꼴이다.  
  
어디서 많이 본 듯한 모양이라면 정답이다. 평면좌표(n=2)나 공간좌표(n=3)에서 원점과 점 x=(x1,x2,⋯ ,xn)**x**=(*x*1​,*x*2​,⋯,*xn*​) 사이의 거리이다. 이 노름에서 유도되는 거리 함수가 유클리드 공간에서 일반적으로 정해지는 거리 함수이며, **유클리드 거리 함수(Euclidean metric)**라고 칭한다. 옛 문헌에는 **피타고라스 거리 함수(Pythagorean metric)**라고 표기된 문헌도 존재하지만 지금은 사장된 표현.

<출처: 나무 위키>

클러스터링의 종류에는 두가지가 있다. Hirearchical Clustering과 Point Assignment Clustering이다.

쉽게 말해 각각 계층적군집분석과 비계층적군집분석이라고 말 할 수 있다.

일단 이 두 방식의 차이점은 계층적 군집분석 같은 경우 개별 대상 간의 거리에 의하여 가장 가까이 있는 대상들로 부터 시작하여 결합해 감으로써 탐욕적 방식을 이용해 군집을 최소단위로 쪼갠후에 거리기반으로 다시 합쳐주면서 군집이 형성되는 과정을 정확하게 파악 할 수 있지만 자료의 크기가 크면 분석하기 어렵다는 단점이있다. agglomerative clustering(병합 군집)의 방식의 예가 있다.

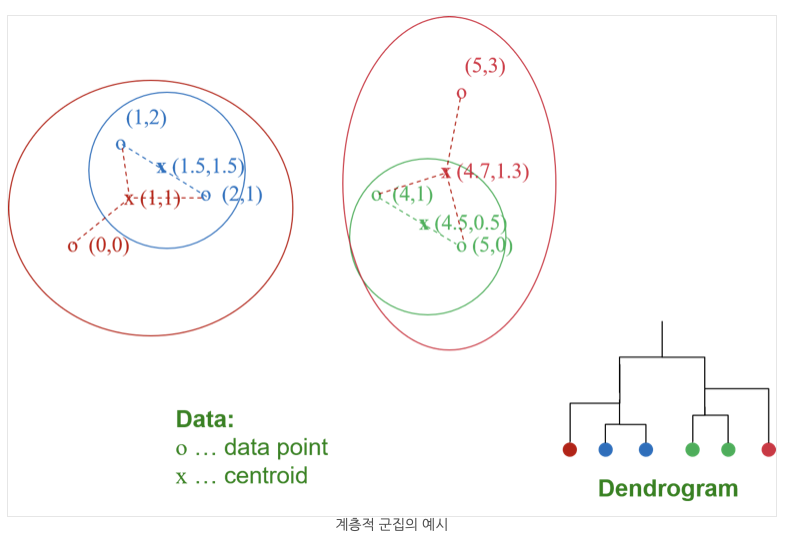
비계층적군집분석의 경우 군집의 수를 정한 상태에서 설정된 군집의 중심에서 가장 가까운 개체를 하나씩 포함하면서 빠르게 군집을 형성 할 수 있는 방식으로 군집의 수를 미리 정하고 초기값을 미리 정해주면서 구하는 방식이다. 미리 training 데이터를 통해 labeling을 해줘야 하는 단점이 있다. 대표적으로 k-means 알고리즘이 있다.

더 자세하게 각각의 방식에 대해서 알아보자.

Hierarchical Clustering<병합군집법>

병합군집은 반복적으로 두개의 가까운 클러스터를 찾는 것이다.

거리 공간에 두점 사이의 거리를 구하기 위해서 거리를 구할 두 점을 구한다. 클러스터 내부에 있는 점의 좌표값의 평균을 이용하는 방법으로 Centroid의 좌표를 구한다. 이 방식으로 하나의 점이 추가 되면 다시 3개의 점의 평균으로 Centroid를 정해 주면서 군집화를 진행한다.



<출처: <https://www.secmem.org/blog/2019/05/17/clustering/>>

위와 같은 방식으로 두개의 가까운 클러스터를 반복적으로 찾는 것이다.

1. 처음에는 (1,2)와 (2,1)의 원소를 갖는 파란색 원의 클러스터를 만들고 centroid를 (1.5,1.5)로 정한다.
2. (4,1)과 (5,0)을 원소로 갖는 녹색원의 클러스터을 만들고, centroid를 (4.5, 0.5)로 정한다.
3. 그 다음은 가장 가까운 두개의 클러스터인 0,0을 원소로 가지는 클러스터, 파란색 원 모양의 클러스터를 합쳐서 왼쪽 빨간색 클러스터를 만들어 낸다.
4. 마찬가지로 오른쪽 빨간색 클러스터 또한 같은 방식으로 만들어 진다.

시간 복잡도는 집합의 원소 N개라 가정할 때 매번 두 클러스터 사이의 거리를 모두 계산하고 합칠 때 마다 O(n^2)의 시간이 소요 되고 총 합치는 횟수가 N-1번 이므로 O(N^3)이 걸려 매우 큰 복잡도를 가진다.

우선순위 큐를 사용한다면 O(N^2log N)이 걸린다.

Point Assignment Clustering

각 클러스터가 어떤 개체를 가지는 지 기록하는 방식으로 진행되는 클러스터링 방법.

대표적으로 K-means알고리즘이 존재.

K-means알고리즘은 임의로 선택한 K개의 점을 이용해 초기의 클러스터 K개를 만들고 클러스터를 계속 맞춰서 변화하며 군집화를 시키는 방식으로써

1. 임의의 k개의 점은 각 나무의 개수만큼 해당 나무의 중심점이나 기타 다른 방법으로 설정 후에 구해준다.
2. 선택한 k개의 점을 centorid로 설정을 한 후 초기 클러스터 k개를 만든다.
3. 각 클러스터 마다 centroid를 새로 찾으며 다시 각 점마다 가장 가까운 centroid를 찾아서 클러스터를 만들고 다시 centroid를 조정하는 작업을 반복한다.
4. 클러스터링한 결과가 달라지지 않을 때 까지 반복.

시계, 그리기이(가) 표시된 사진

자동 생성된 설명

시계, 그리기이(가) 표시된 사진

자동 생성된 설명

<출처: <https://www.secmem.org/blog/2019/05/17/clustering/>>

위와 같은 방식으로 점 여덟개에서 K-means 클러스터링을 진행하면 K = 2로 설정 했을 때 centroid를 주황색점과 파란점으로 설정을 하고 그 centroid를 기반으로 가장 가까운 점을 찾고 클러스터를 만든다. 그 후 다시 새로운 centroid인 네모점을 구해준다.

이런 방식으로 계속해서 centroid를 설정을 해주고 가장 가까운 점을 기준으로 클러스터링을 시켜줌 으로써 사진의 가장 마지막과 같이 더이상 클러스터의 변화가 없을 때 까지 군집화를 완료한다.

k-means 알고리즘은 처음에 k를 선택을 하고 클러스터링을 진행하는데 k를 작게 한다면 다른 클러스터에 위치해야 하는 점이 한 클러스터에 묶이게 되는 문제가 발생 하고 품질이 하락함.

또한 K를 너무 크게 한다면 거리가 충분히 가까워서 한클러스터에 묶일 수 있는 부분이 다른 클러스터에 묶일 가능 성이 있다. 수고의 경우 개체목의 개수를 k로 설정하고 그만큼 군집화를 해야하는데 이런 경우 생기는 문제점에 대해 테스트 할 필요성이 있다.

참고로 최적의 k를 구하는 방법에는 k를 1로 놓고 시작하여 2배씩 늘리면서 평균거리가 적게 줄어드는 k = 2^i ~ 2^i+1사이의 i를 찾아 2^i-1 ~ 2^i 사이의 k값이 최적이 k값이다. 좀 더 정확한 값을 얻으려면 2^i-1 ~ 2^i사이에 이분탐색을 사용하여 구한다. 예를 들어, 현재 최적의 k가 x 이상 y 이하의 닫힌 구간에 존재한다는 사실을 안다면, 먼저 z=(x+y)/2로 놓고, k=z로 놓고 k-means 클러스터링을 진행한다. 이 때, k=z와 k=y 사이에서 평균 거리가 적게 줄어들었다면, 최적의 k는 닫힌 구간 [x,z]에 존재한다고 생각할 수 있다. 반대의 경우에는 최적의 k가 닫힌 구간 [z,y]에 존재한다고 생각할 수 있다.

시간 복잡도는 k 점을 찾는 과정에서 O(k) 그 후 각 점에서 가까운 centroid를 찾는 과정이 O(Nk), centroid를 조정 할 때 O(N), 이러한 과정을 x번 반복시 O(Nkx)

만일 최적의 k를 찾는 방법을 적용한다면 O(NkxlogN)이 된다. (이분탐색의 시간 복잡도: O(log N))

최적의 클러스터를 형성할 때 제대로 클러스터가 형성 되지 않는 문제점이 발생할 수 있으므로 추후 이에 대해 논의 해봐야하며 위에서 설명한 un-supervised 방식과는 다른 종류의 supervised 러닝인 KNN알고리즘에 대해서 논의 해봐야 한다.

\*K-means 사용 예

<https://eunsukimme.github.io/ml/2019/12/16/K-Means/>

참고 : <https://www.secmem.org/blog/2019/05/17/clustering/>

[J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, Chapter. 7. Clustering](J.%20Leskovec,%20A.%20Rajaraman,%20J.%20Ullman:%20Mining%20of%20Massive%20Datasets,%20Chapter.%20%20%207.%20Clustering) (<http://www.mmds.org>)