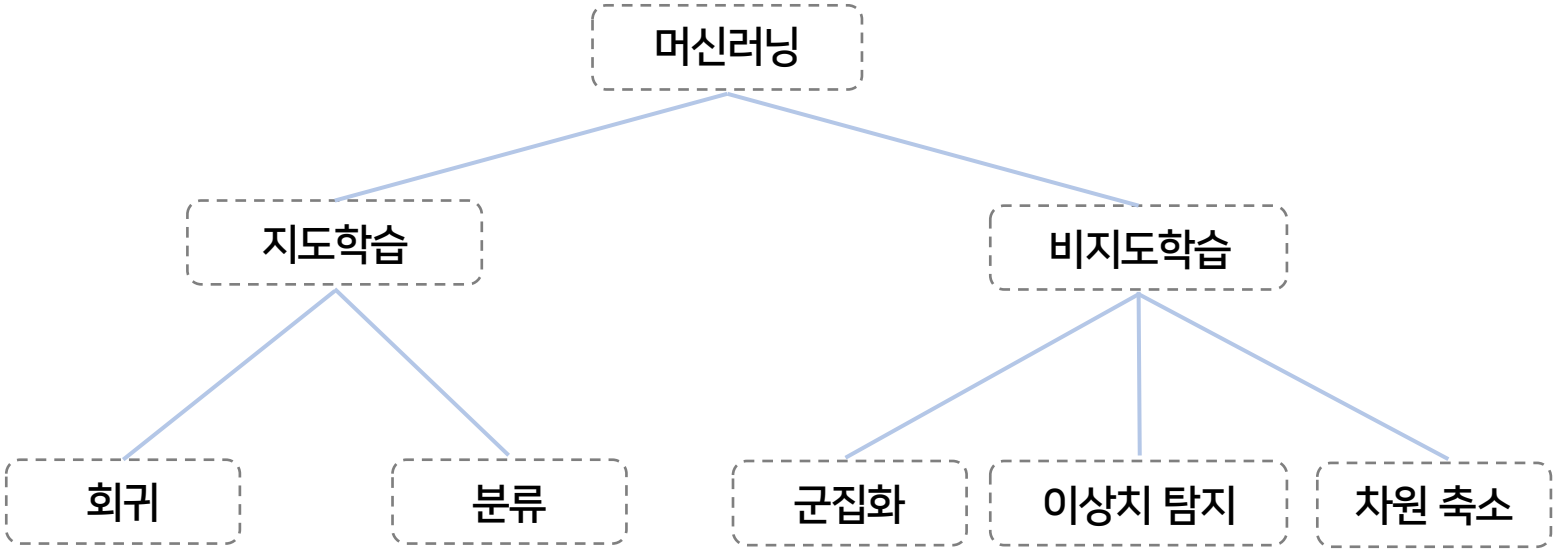
데이터마이닝팀

4팀
오주원
이동기
김형석
이경미
최종혁

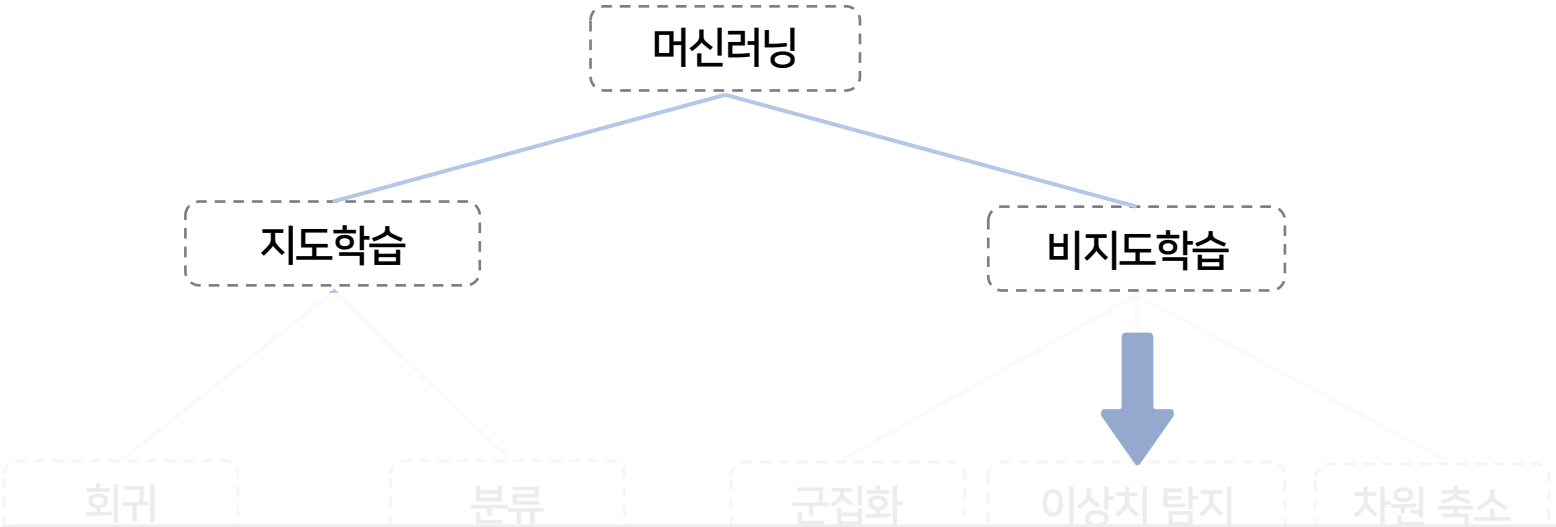
INDEX

비지도학습



1 클러스터링

비지도학습



비지도학습은 **정답(Y) 없이 입력(X)의 내재적 특징을 찾아**내는 것이 목표

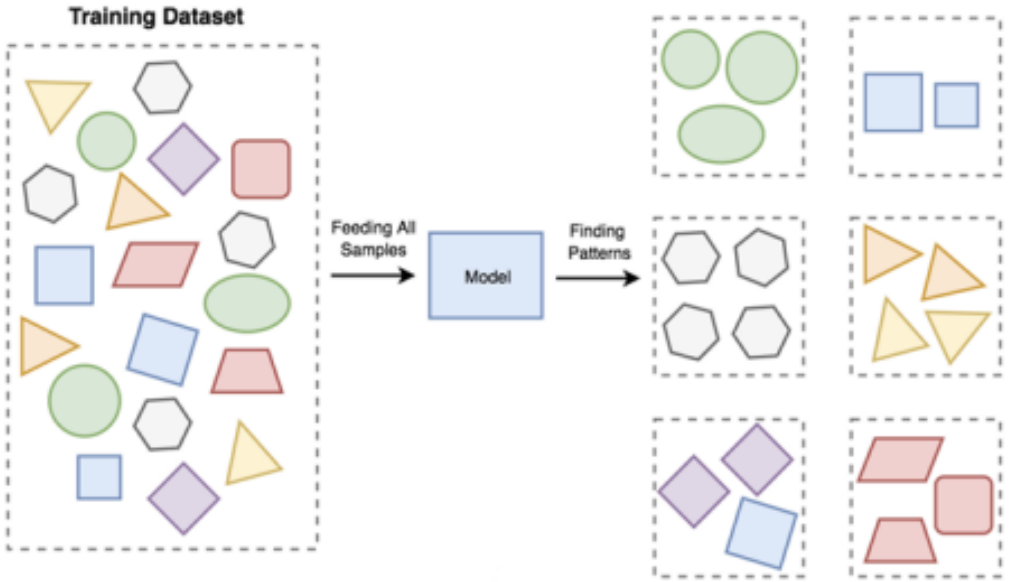
이중 **클러스터링(군집화)**은 대표적인 **비지도학습** 모델

클러스터링

클러스터링

비지도학습 모델의 종류 중 하나로

비슷한 특성을 갖는 데이터들끼리 같은 그룹으로 묶어가는 방식

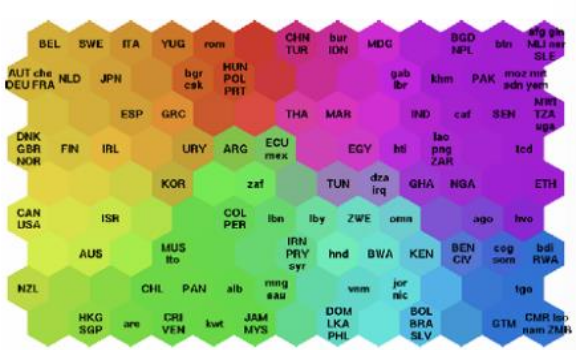


클러스터링

클러스터링

비지도학습 모델의 종류 중 하나로

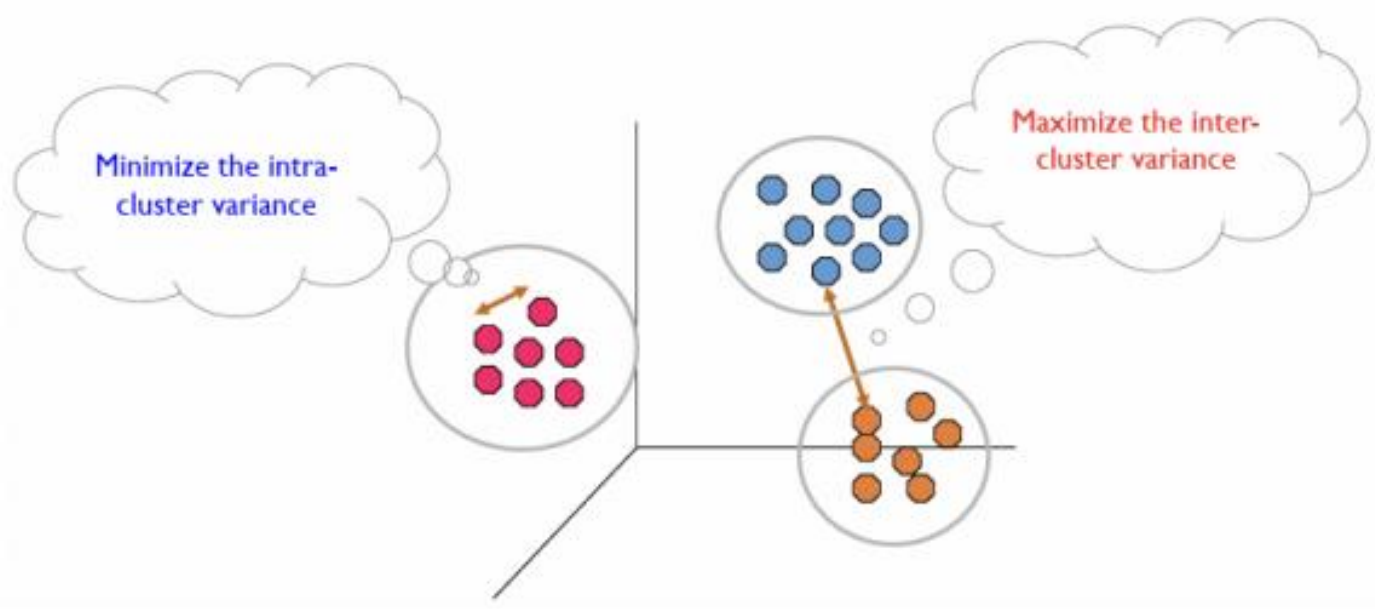
비슷한 특성을 갖는 데이터들끼리 같은 그룹으로 묶어가는 방식



군집화 시 데이터를 보다 **명확하게 이해**할 수 있고,

군집화 결과에 따라 **군집별로 적절한 전략**을 수립할 수도 있음

군집 타당성 지표 (Clustering Validity Measure)



군집화의 핵심 아이디어는 **군집 내부의 분산은 최소화** 시키고,
서로 다른 **군집 간의 분산은 최대화** 시키는 것

군집 타당성 지표 (Clustering Validity Measure)

External Measure

Internal Measure

Relative Measure

군집 타당성 지표의 종류는 위처럼 총 3가지가 존재함

군집 타당성 지표 (Clustering Validity Measure)

External Measure

Internal Measure

Relative Measure

External Measure

클러스터 개수 및 멤버가 정해져 있는 상황에서 평가하는 방식

새로운 알고리즘을 개발하거나 연구할 때 확인용으로 사용

우리가 사용 X !

군집 타당성 지표 (Clustering Validity Measure)

External Measure

Internal Measure

Relative Measure

Internal Measure

군집 내 분산(Intra-cluster variance)에 집중한 방식

군집 타당성 지표 (Clustering Validity Measure)

External Measure

Internal Measure

Relative Measure

Relative Measure

군집 내 분산과 군집 간 분산 모두에 집중한 방식

일반적으로 가장 많이 사용됨

1 클러스터링

Internal Measure

Elbow Method

적절한 클러스터 개수를 쉽게 확인할 수 있는 방식

WSS(Within Sum of Square)를 계산하여 군집의 타당성을 판단함

$$WSS = \sum_{i=1}^{K} \sum_{x_j \in C_i} \|X_j - C_i\|^2$$

C_i : i 번째 클러스터의 중심

K: 클러스터 개수

j: i번째 클러스터의 데이터 포인트 수

군집 내 중심과 객체들 간의 거리를

제곱시켜 합한 값으로

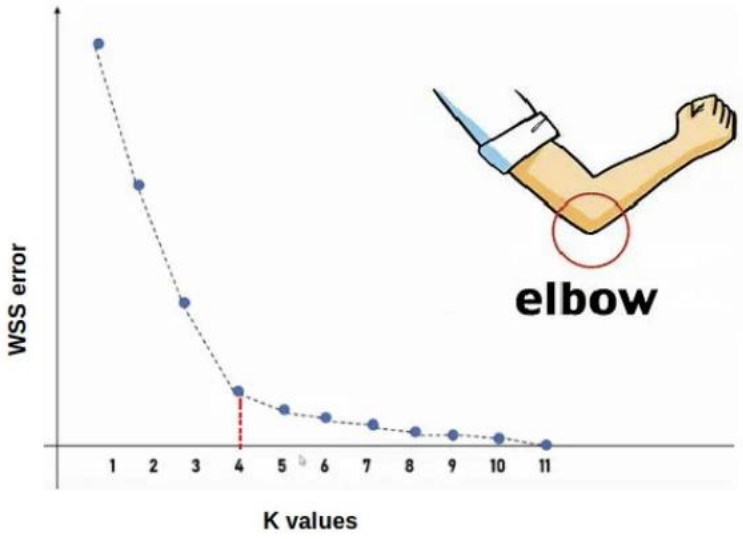
값이 작을수록 군집화가 잘 됐다고 판단

Internal Measure

Elbow Method

적절한 클러스터 개수를 쉽게 확인할 수 있는 방식

WSS(Within Sum of Square)를 계산하여 군집의 타당성을 판단함



WSS값은 **클러스터 수와 반비례** 관계이므로,

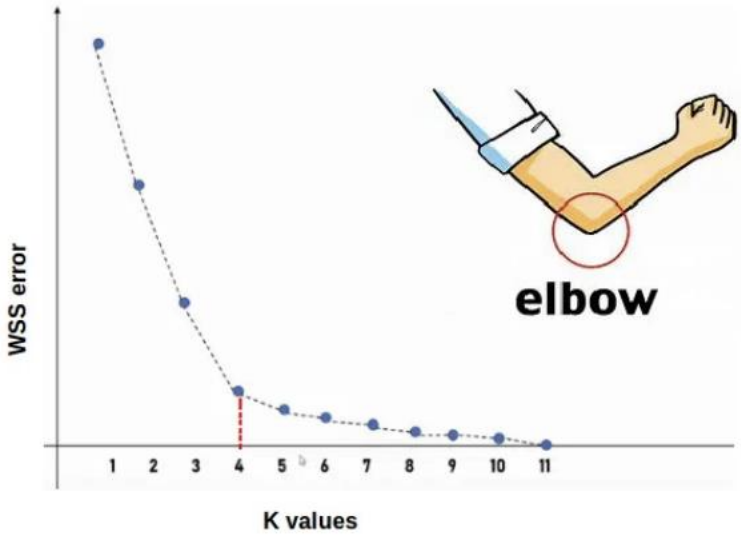
Elbow Method를 이용해 WSS 감소폭이 급격하게 줄어드는 지점을 찾아 **최적의 클러스터 수를 결정**

Internal Measure

Elbow Method

적절한 클러스터 개수를 쉽게 확인할 수 있는 방식

WSS(Within Sum of Square)를 계산하여 군집의 타당성을 판단함



그러나 실제 데이터셋에서는

식별할 수 있는 **명확한 변곡점이 존재하지 않아**

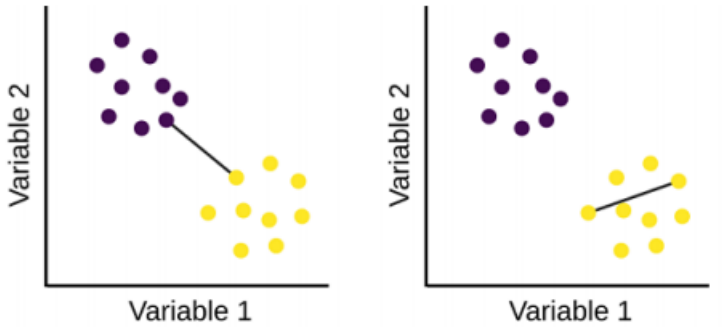
잘 사용하지 않는 방법!

Relative Measure

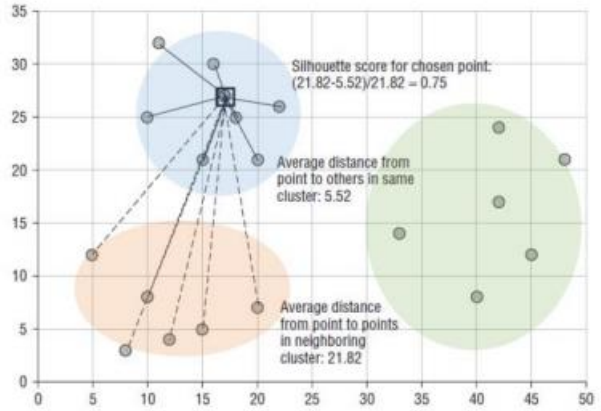
Relative measure

군집 내 분산 및 군집 간 분산 모두 고려한 방식으로

Dunn Index와 Silhouette Index를 이용해 군집의 타당성을 판단



Dunn Index



Silhouette Index

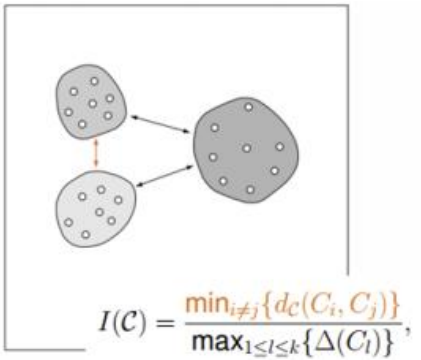
1 클러스터링

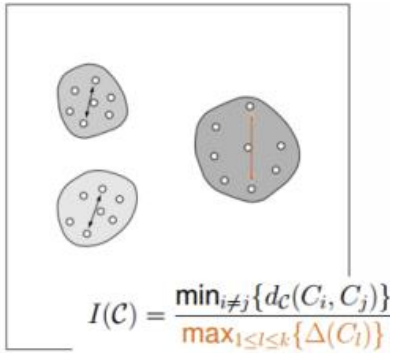
Relative Measure

Dunn Index

클러스터 내 최대거리에 대한 클러스터 간 최소거리의 비율

값이 클수록 클러스터링이 잘 된 것으로 판단





$$I(C) = \frac{min_{i \neq j}\{dc(C_i, C_j)\}}{max_{1 \leq l \leq k}\{\Delta(C_l)\}}$$

C_n : n 번째 클러스터의 중심

ΔC_n: n 번째 클러스터 내 거리

dc: 두 클러스터 중심 간의 거리

k: 클러스터 개수

1 클러스터링

Relative Measure

Dunn Index

클러스터 내 최대거리에 대한 클러스터 간 최소거리의 비율

값이 클수록 클러스터링이 잘 된 것으로 판단

$$I(C) = \frac{min_{i \neq j}\{dc(C_i, C_j)\}}{max_{1 \leq l \leq k}\{\Delta(C_l)\}}$$

- $min_{i \neq j}\{dc(C_i, C_j)\}$ → 군집 간 거리 최소화
- $max_{1 \leq l \leq k}\{\Delta(C_l)\}$ → 군집 내 거리 최대화

이상치의 영향을 상쇄시키기 위해 min, max 대신

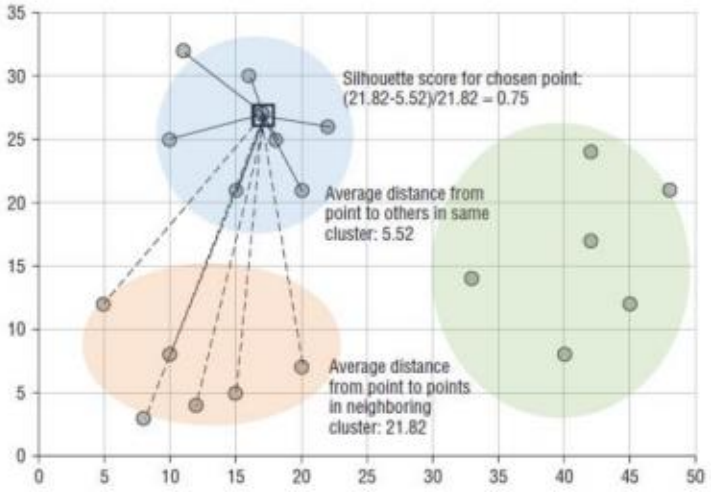
average를 사용하기도 함

Relative Measure

Silhouette Index

각 데이터 포인트마다 **군집 내외의 거리**를 비교한 지표

이론적으로 -1과 1 사이 값을 가지며, **1에 가까울수록 클러스터링이 잘 된 것**으로 판단



$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

$\boldsymbol{a(i)}$: **객체 i와 같은 군집 내 객체들** 사이 평균 거리

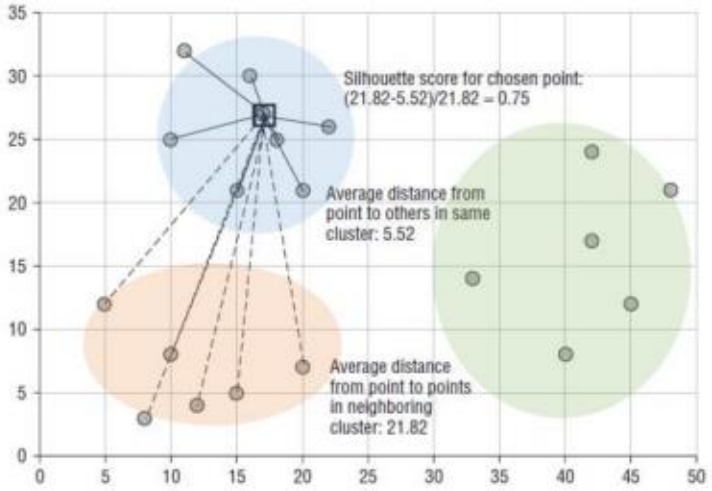
$\boldsymbol{b(i)}$: **객체 i와 다른 군집 객체들** 사이 평균 거리의 최솟값

Relative Measure

Silhouette Index

각 데이터 포인트마다 **군집 내외의 거리**를 비교한 지표

이론적으로 -1과 1 사이 값을 가지며, **1에 가까울수록 클러스터링이 잘 된 것**으로 판단



모든 데이터 포인트마다 $s(i)$ 값들을 계산한 뒤

평균을 구해 **전체** Silhouette Score를 도출

Silhouette Index > 0.5: 잘 묶인 군집

Silhouette Index > 0.7: 정말 잘 묶인 군집

Silhouette Index < 0: 아예 다른 군집에 할당됨

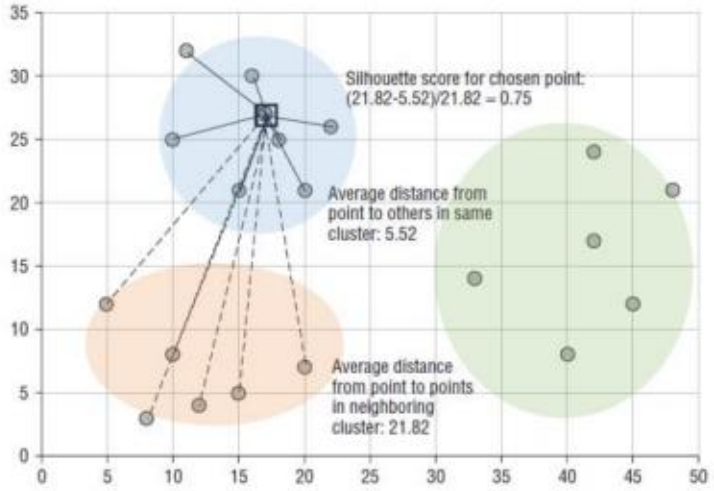
1 클러스터링

Relative Measure

Silhouette Index

각 데이터 포인트마다 **군집 내외의 거리**를 비교한 지표

이론적으로 -1과 1 사이 값을 가지며, **1에 가까울수록 클러스터링이 잘 된 것**으로 판단



모든 데이터 포인트마다 $s(i)$ 값들을 계산한 뒤 평균을 구해 최종 Silhouette Score를 도출

그러나

단순히 전체 Silhouette Score가 높다고 해서

군집화가 잘 됐다고 판단할 수는 없음

Silhouette Index > 0.5: 잘 묶인 군집

Silhouette Index > 0.7: 정말 잘 묶인 군집

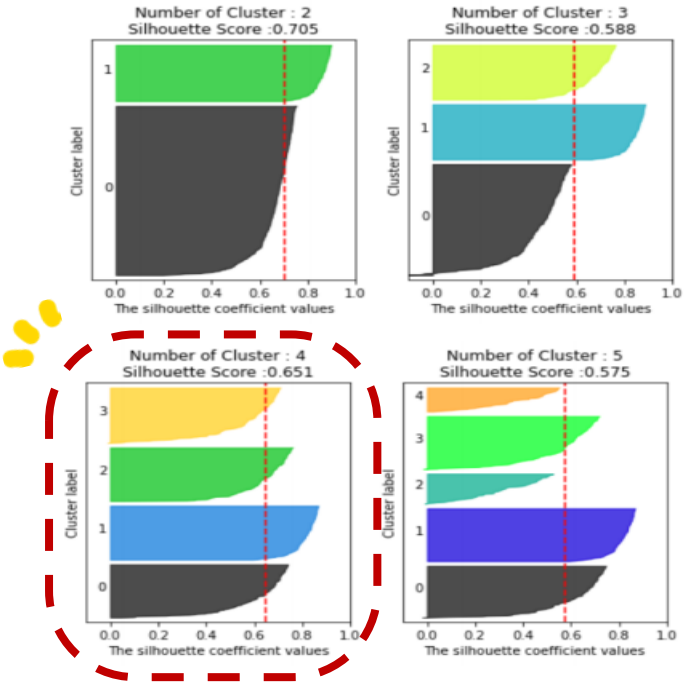
Silhouette Index < 0: 아예 다른 군집에 할당됨

Relative Measure

Silhouette Index

각 데이터 포인트마다 **군집 내외의 거리**를 비교한 지표

이론적으로 -1과 1 사이 값을 가지며, **1에 가까울수록 클러스터링이 잘 된 것**으로 판단



특정 군집의 실루엣 계수가 높은 경우에도

전체 score가 높아질 수 있음

↓

전체 실루엣 계수가 최댓값이 아니더라도

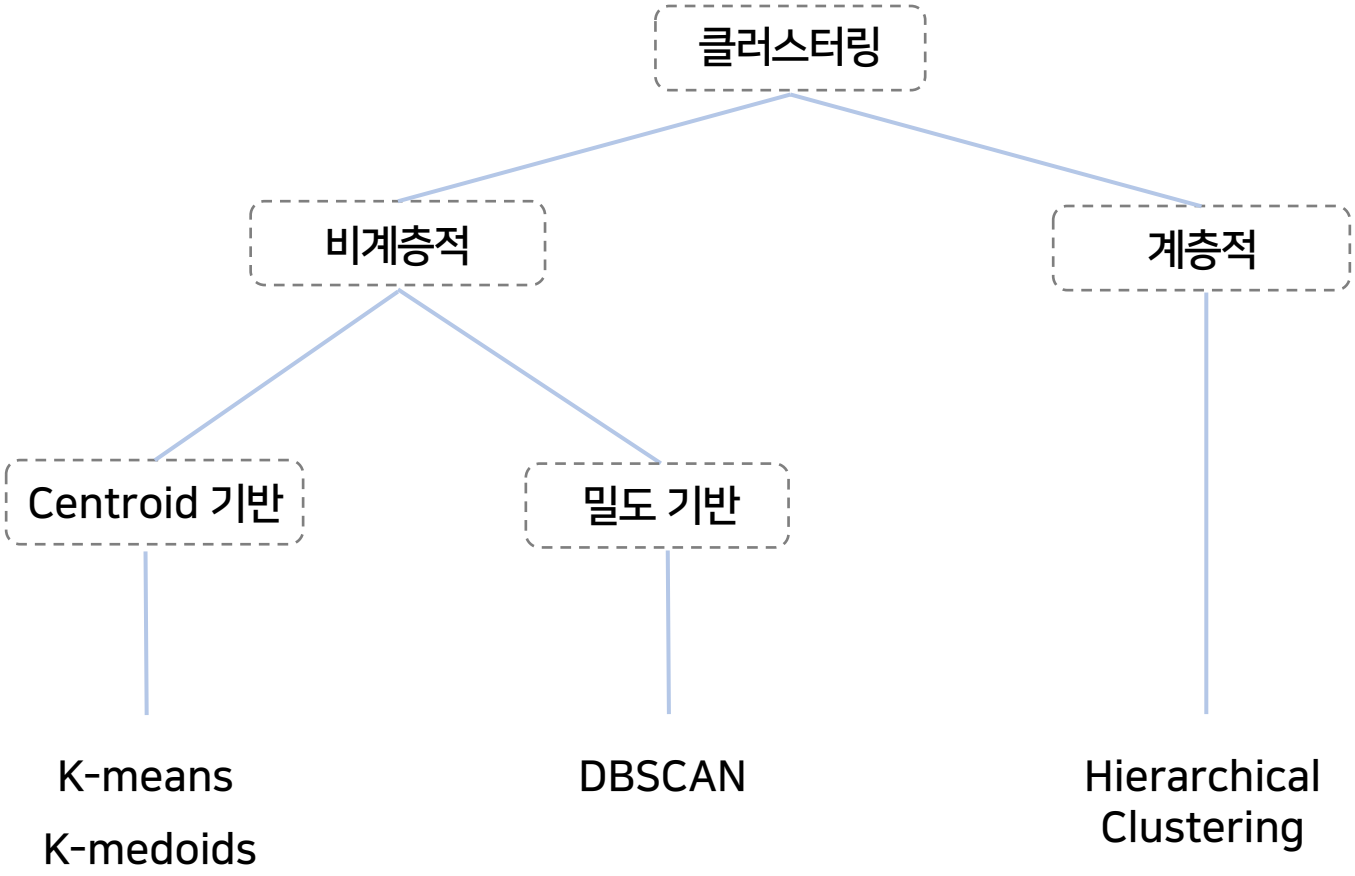
개별 군집의 평균값의 편차가 크지 않은 경우

전체 군집화 성능이 좋다고 판단

왼쪽 예시의 경우 클러스터 수를 4로 결정 !

1 클러스터링

클러스터링의 분류



비계층적 클러스터링 | K-means Clustering

K-means Clustering

데이터들의 **평균 지점을 Centroid로** 활용하는 클러스터링 방식

클러스터 수(K)를 미리 설정해줘야 하고, 평균 지점은 실제 좌표 값이 아닐 수 있음

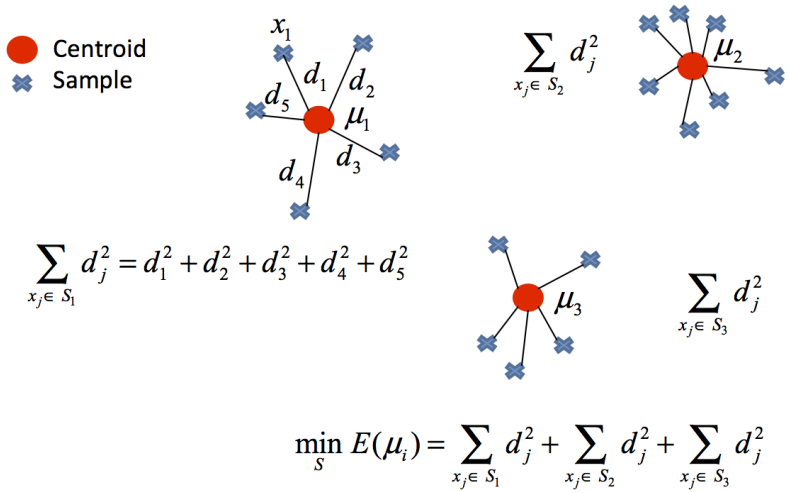
$$C = arg\min_{C} \sum_{i=1}^{K} \sum_{x_j \in C_i} \|X_j - C_i\|^2$$

C_i : i 번째 클러스터의 중심

C: 클러스터 중심 값들을 모아둔 벡터

K: 클러스터 개수

j: i번째 클러스터의 데이터 포인트 수

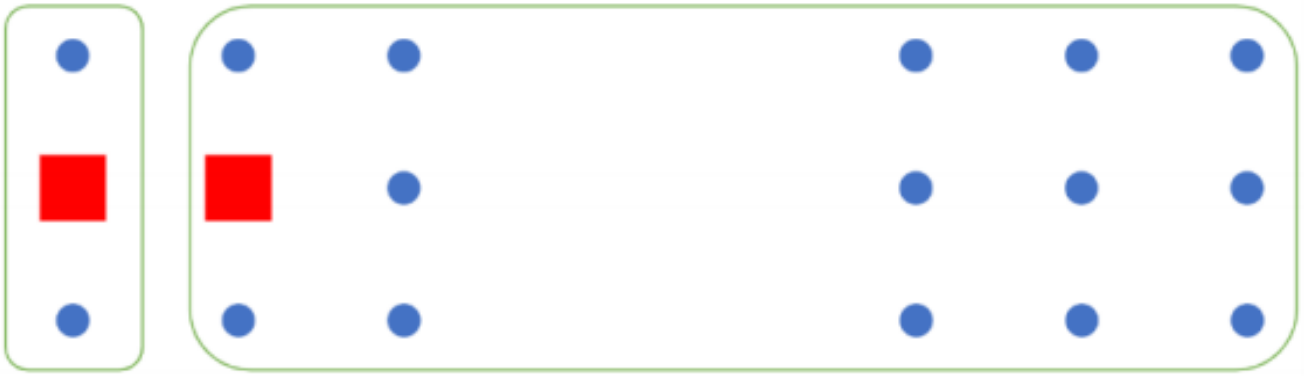


K-means Clustering 학습 과정

① K 개의 Centroid를 랜덤하게 생성

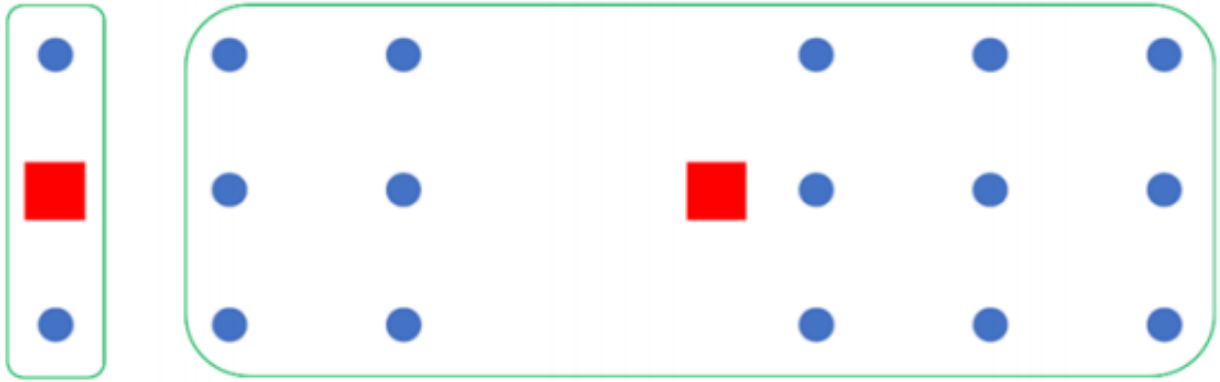
여기서 K = 2

K-means Clustering 학습 과정



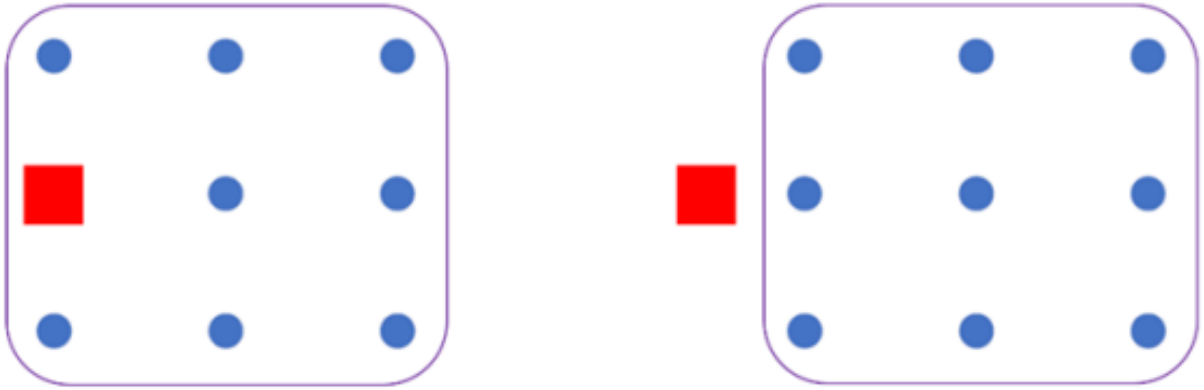
② 관측치들(파란 점)을 가장 가까운 Centroid에 맞게
1차 군집(초록 박스)으로 할당함

K-means Clustering 학습 과정



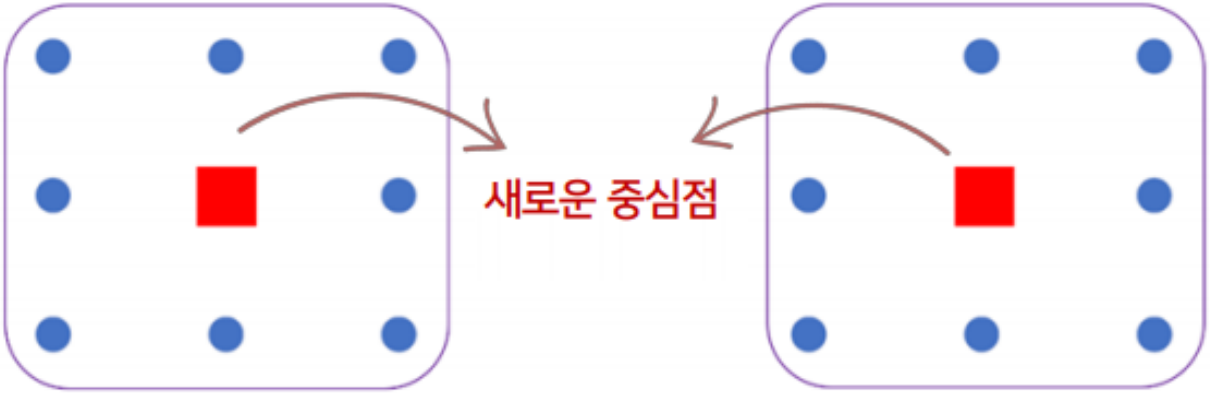
③ 1차 군집의 관측치들을 기준으로 Centroid 업데이트

K-means Clustering 학습 과정



④ 관측치들을 수정된 Centroid에 맞게 2차 군집으로 할당

K-means Clustering 학습 과정



⑤ 2차 군집의 관측치들을 기준으로 Centroid를 다시 업데이트

이러한 과정을 계속 반복 후 Centroid 및 군집에 변화가 없으면 학습을 종료

K-means Clustering 학습 과정

직관적이고 구현이 쉬우나,

초기값에 민감하고, **이상치의 영향**을 많이 받음

또한 **범주형 데이터**에는 적용할 수 없으며,

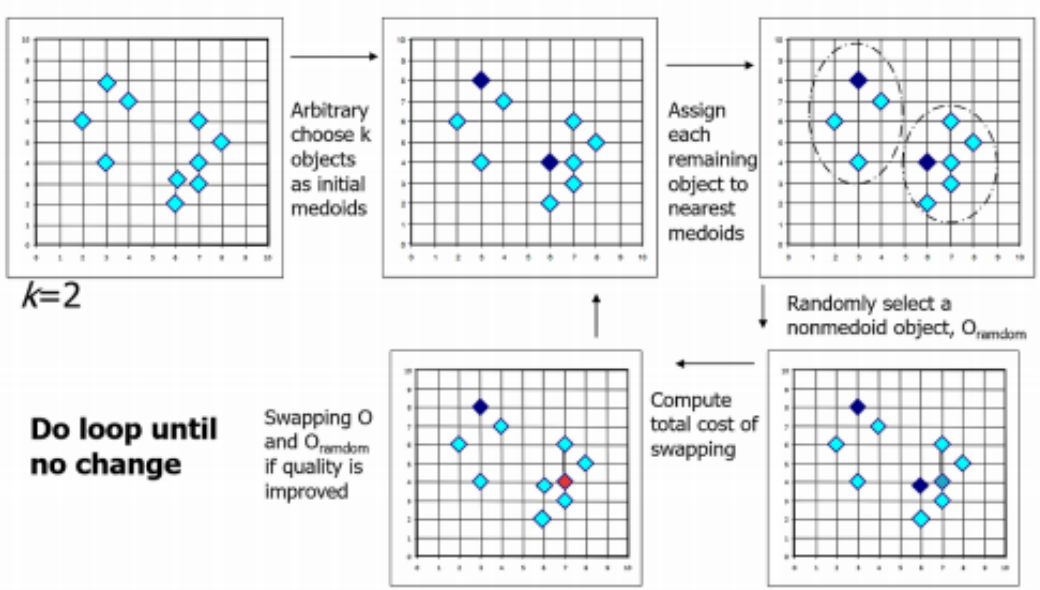
그룹 내 **분산 구조가 원(구)형**인 경우에만 특화되어 있음

비계층적 클러스터링 | K-medoids Clustering

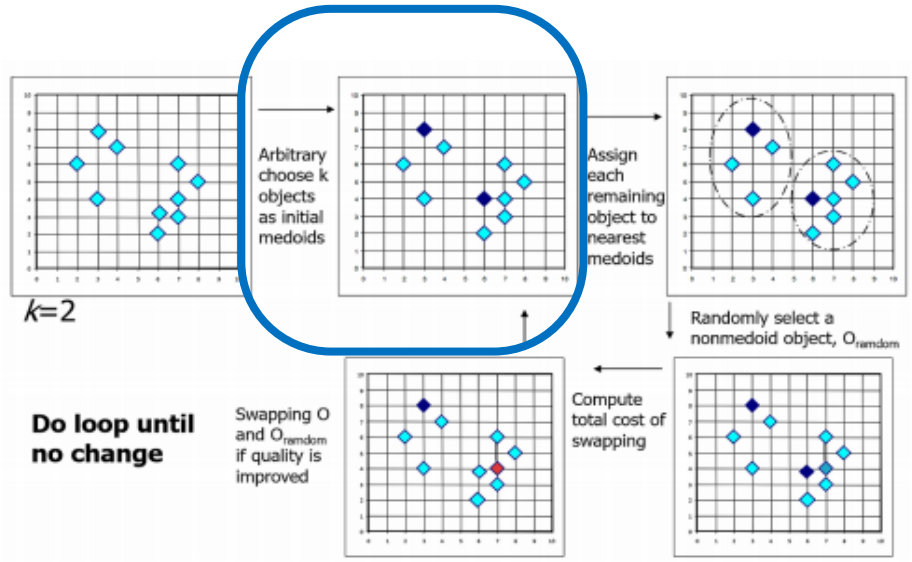
K-medoids Clustering

데이터들의 **중앙값을 Centroid로** 활용하는 클러스터링

K-means와는 달리 **실제 포인트만을** Centroid로 활용



K-medoids Clustering 학습 과정

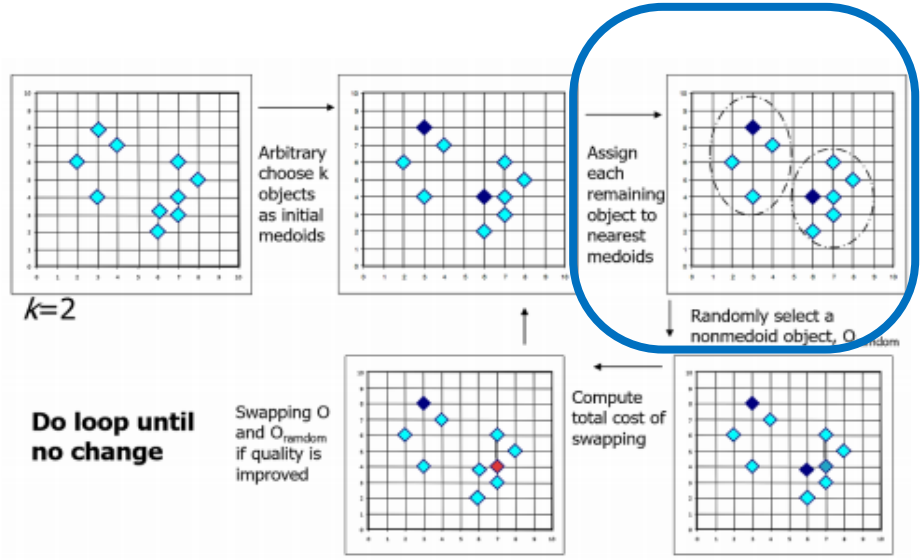


①

n개의 데이터 포인트 중에서

임의로 K개의 Centroid를 지정

K-medoids Clustering 학습 과정

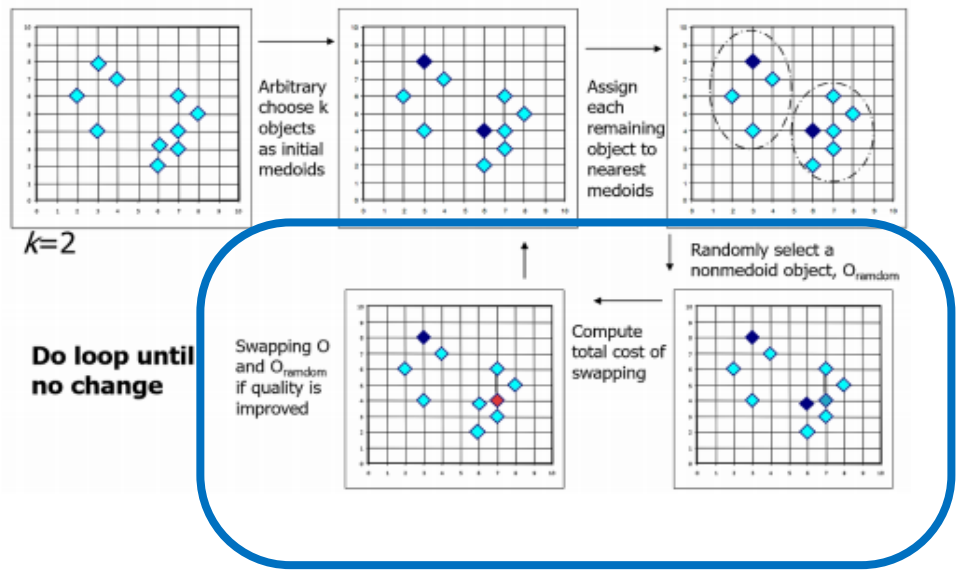


②

가장 가까운 Centroid에

각각의 데이터 포인트를 할당

K-medoids Clustering 학습 과정



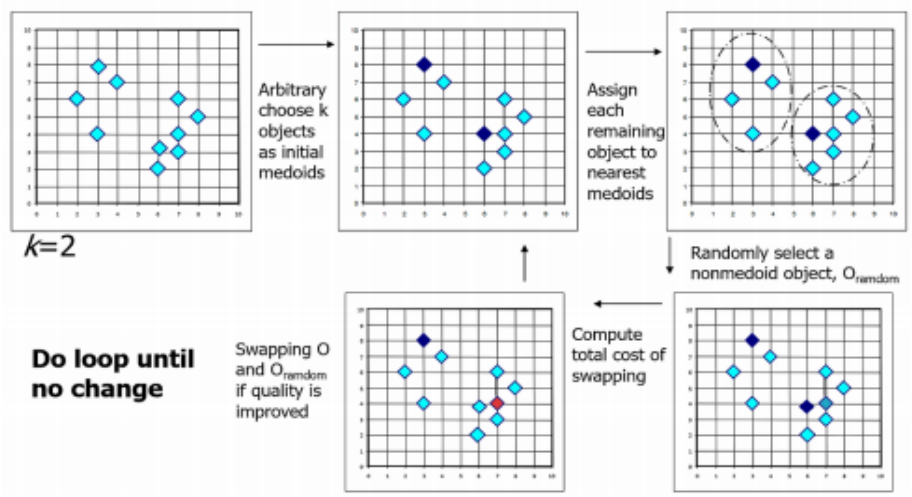
③

각 Centroid에 대해

거리(비용) 합이 가장 작은 데이터 포인트를

새로운 Centroid로 선택

K-medoids Clustering 학습 과정



④

2, 3 단계를 반복하면서

더 이상 Centroid가 변하지 않거나

미리 정한 최대 반복 횟수에 도달하면 종료

K-medoids Clustering 학습 과정

이상치에 강건하고 군집 거리 측정방식이 유연한 편이지만,

실제 포인트 간 거리를 전부 계산해야 하므로

계산 복잡도가 커 속도가 느리다는 단점이 존재

1 클러스터링

거리 기반 클러스터링의 한계

K-Means Clustering

데이터의 **평균 지점**을
Centroids로 활용

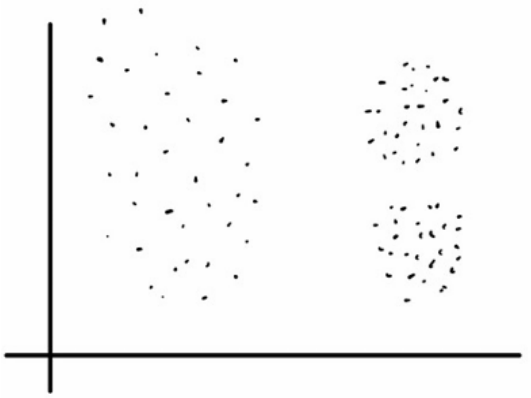
K-Medoids Clustering

데이터의 **중앙값**을
Centroids로 활용

거리 기반 클러스터링은
밀도의 차이나 **특수한 형태**를 반영하지 못하는 한계를 가짐

1 클러스터링

거리 기반 클러스터링의 한계



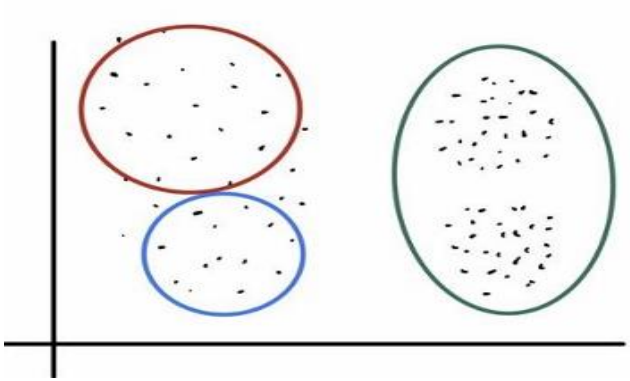
다음과 같은 데이터를 **군집**으로 묶는다면,

직관적으로 왼쪽 1개, 오른쪽 2개를 생각할 것임

1 클러스터링

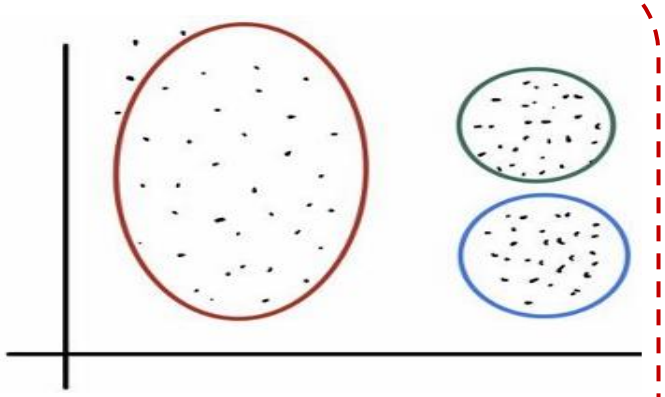
거리 기반 클러스터링의 한계

< 거리 기반 >



(직관에 부합하지 않음)

< 밀도 기반 >



(직관에 부합함)

밀도 기반 클러스터링은

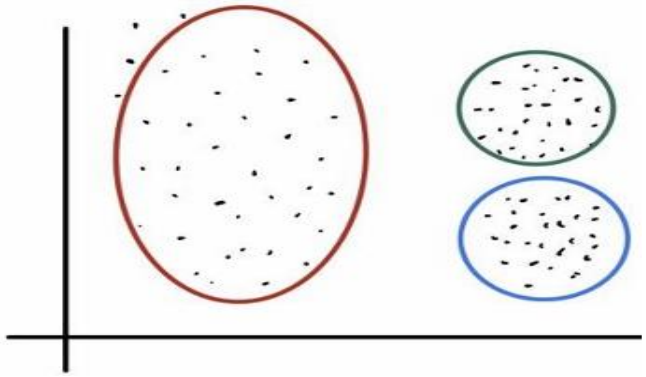
데이터의 **분포(뭉쳐있는 정도)를 고려**하므로 직관에 부합하는 결과

거리 기반 클러스터링의 한계

< 거리 기반 >

(직관에 부합하지 않음)

< 밀도 기반 >



(직관에 부합함)

밀도 기반의 대표적인 알고리즘인 DBSCAN에 대해 알아보자!

1 클러스터링

DBSCAN

DBSCAN

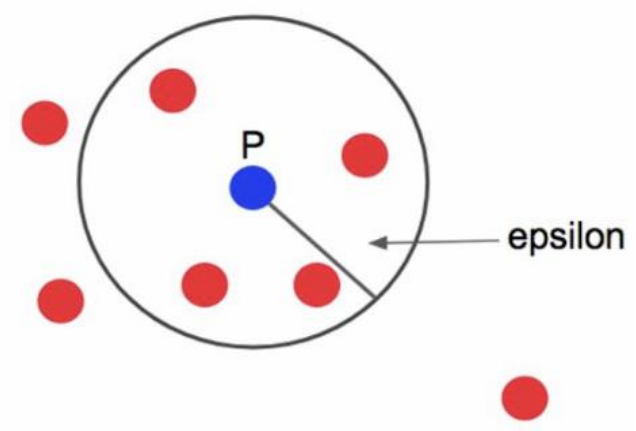
Density-Based Spatial Clustering of Application with Noise의 준말

서로 인접한 데이터들은 같은 클러스터일 것이라는 아이디어에서 착안

밀도가 높은 곳에 포함된 데이터에는 클러스터를 할당, 굉장히 낮다면 노이즈 취급

DBSCAN 용어 정리

ε-Neighborhood (of a point p)



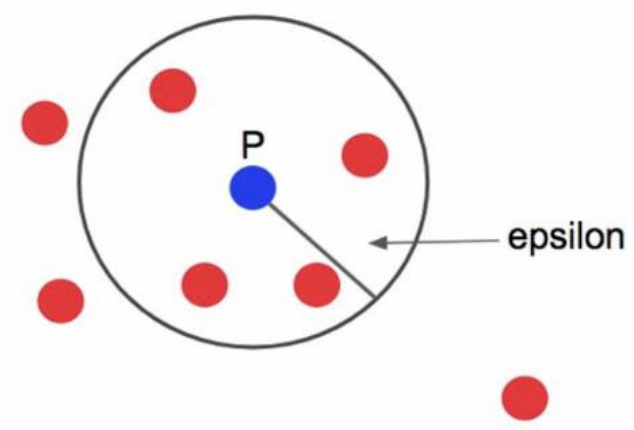
$$N_{\epsilon}(p) = \{q \in D \mid dist(p, q) \leq \epsilon\}$$

P와의 거리가 ε보다 작은 점들

즉, p를 기준으로 반경 ε 내에 있는 모든 점들

DBSCAN 용어 정리

ε-Neighborhood (of a point p)

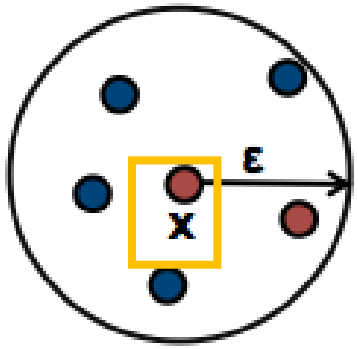


$$N_{\epsilon}(p) = \{q \in D \mid dist(p, q) \leq \epsilon\}$$

반경 내 점의 최소 개수인 minPts와 점 p와의 거리인 엡실론(ε)은
사용자가 지정

DBSCAN 용어 정리

Core Point

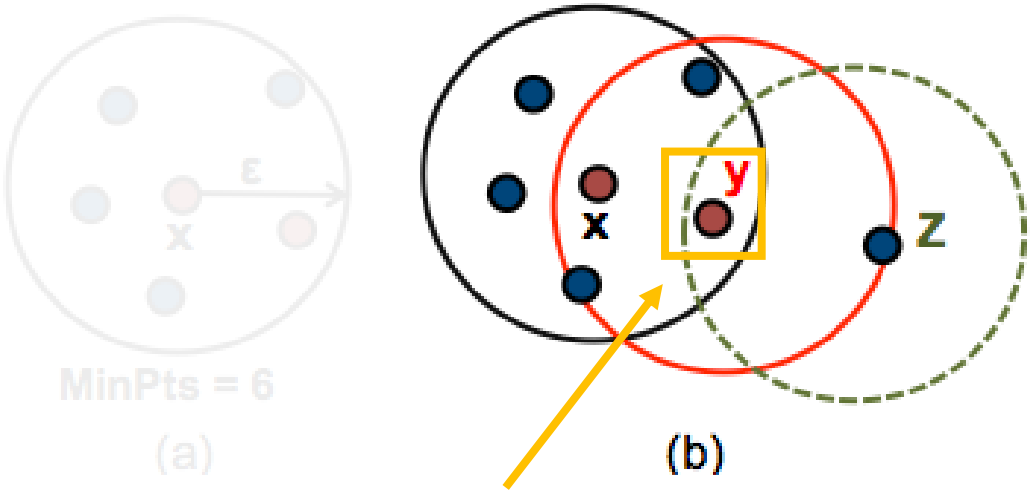


엡실론 반경 내에 **minPts개 이상의 점이** 존재하는 점

위 그림(a)의 경우 minPts가 6일 때 자기 자신 포함 6개 존재

DBSCAN 용어 정리

Border Point



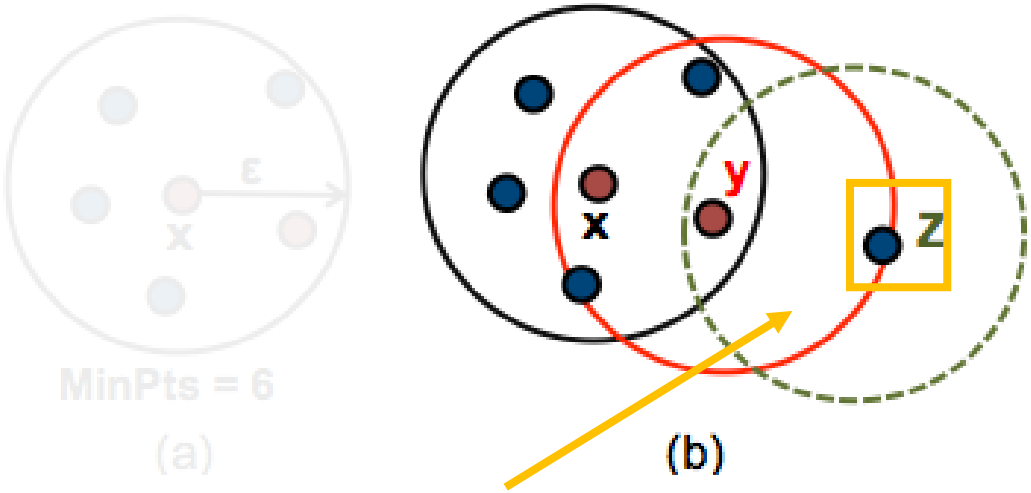
엡실론 반경 내에 **minPts개 미만의 점이** 존재하는 점들 중,

Core Point의 **엡실론 반경 내에 포함**되는 점

위 그림(b)의 경우 y는 minPts개 미만의 점을 가지지만, core point인 x의 반경 내에 존재

DBSCAN 용어 정리

Noise Point

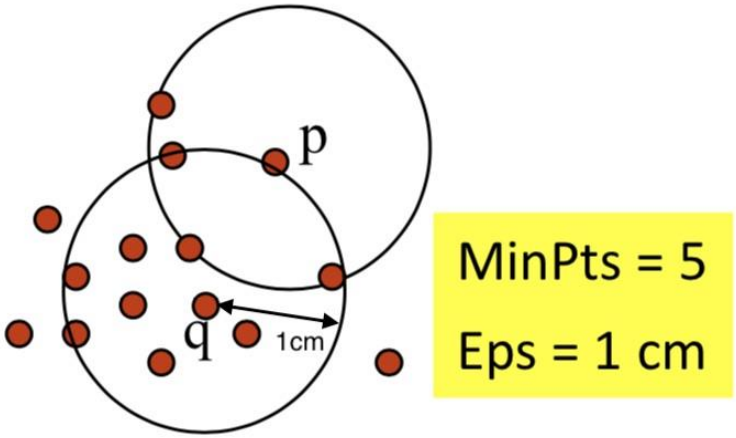


Core도 아니고 Border도 아닌 점

위 그림(b)에서 z의 경우 minPts개 이하의 점을 가지고, 주변에 Core point도 없음

DBSCAN 용어 정리

Directly Density-reachable



밀도 관점에서 직접적으로 접근 가능

이를 위해서는 2가지 조건 모두 만족 필요

DBSCAN 용어 정리

Directly Density-reachable

Reachability

$$p \in N_\epsilon(q)$$

p가 q의 ε-Neighborhood에 속해야 함

Core point condition

$$|N_\epsilon(q)| \geq minPts$$

q의 ε-Neighborhood 개수는 $minPts$ 이상이어야 함

두 가지 모두 만족할 때, q가 Core point이며

점 p가 q로부터 직접적으로 밀도 접근 가능한 관계에 있음

DBSCAN 용어 정리

Directly Density-reachable

Reachability

$p \in N_\epsilon(q)$

p가 q의 ε-Neighborhood에 속해야 함

Core point condition

$|N_\epsilon($ $\geq$ s

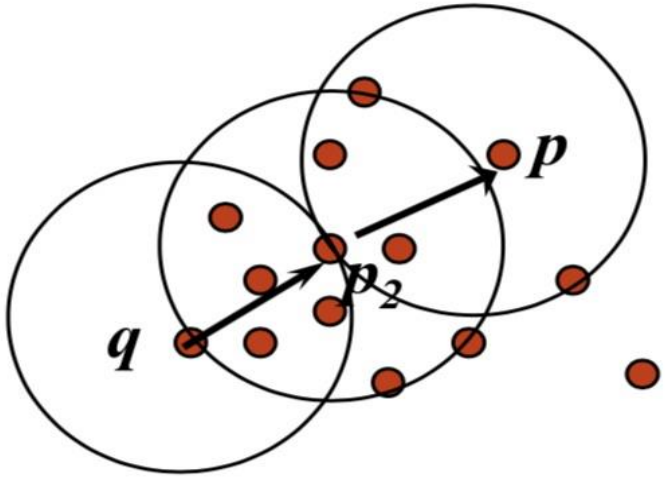
q의 ε-N 개수는

s 이상이어

하지만, DDR한 점들끼리만 한 군집으로 묶는 다면,
비교적 바깥쪽에 있는 점들을 놓치게 될 수 있음
(∵ **바깥 점은 core point condition을 만족하지 못 할 수도** 있음)

DBSCAN 용어 정리

Directly-reachable



P가 q의 반경 밖에 있어도

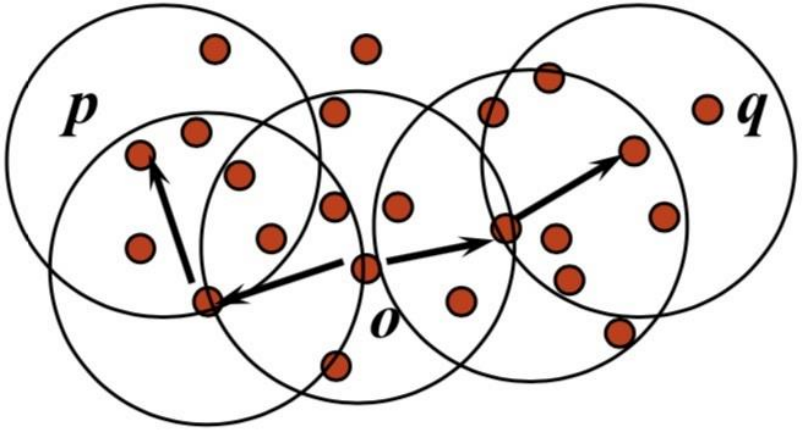
그 사이의 점들을 D.D.R하게 연결 가능

= p가 q로부터 D.R함

점 p가 q로부터 **밀도 관점에서 접근 가능**

DBSCAN 용어 정리

Directly-connected



p가 o로부터 D.R하고
q도 o로부터 D.R함
=p가 q로부터 D.C함

점 p가 q로부터 **밀도 관점에서 연결되어 있음**
즉, **같은 군집에 할당된 점들**은 모두 Density-connected

1 클러스터링

DBSCAN 학습 과정

1. 임의의 데이터 포인트 선택

2. p로부터 Density-reachable한 포인트 탐색

i) p가 Core point면 클러스터에 할당

ii) p가 Border point면 다른 point 선택

3. 모든 데이터 포인트가 탐색될 때까지 반복

1 클러스터링

DBSCAN

DBSCAN의 장점

① **특정한 모양**의 클러스터를 찾아낼 수 있음

거리 기반 모델의 경우 유클리드 거리를 기반으로 해서 원형의 군집만 찾아내지만 DBSCAN은 제한 없음

② 군집에 **할당되지 않는 객체**도 허용함

다른 모델들은 모든 데이터를 군집에 할당하지만, DBSCAN은 이상치를 굳이 할당하지 않음

③ 클러스터링의 **랜덤성**이 작음

거리 기반 모델의 경우 초기값에 굉장히 민감

④ 클러스터 **개수(K)**를 지정하지 않아도 됨

1 클러스터링

DBSCAN

DBSCAN의 한계점

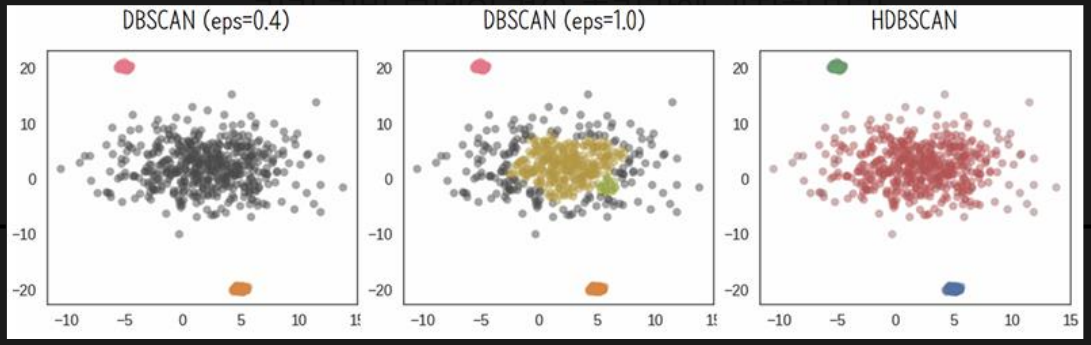
적절한 엡실론과 minPts를 설정하는 것이 어려움

데이터셋 내에서 밀도가 제각각이라면 난이도는 더욱 올라감

이를 해결하기 위해 HDBSCAN 사용!

② 군집에 **할당되지 않는 객체**도 허용함

HDBSCAN(Hierarchical DBSCAN)



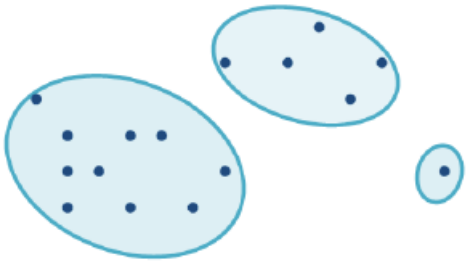
1 클러스터링

계층적 클러스터링

계층적 클러스터링 (Hierarchical Clustering)

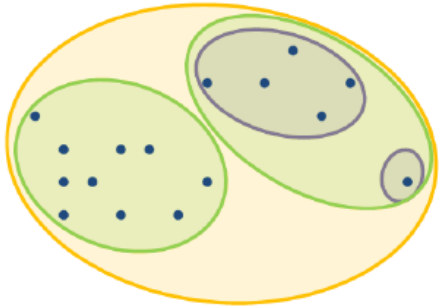
계층이 존재하는 트리를 이용해서 개별 개체들을

순차적으로 묶어나가는 클러스터링 방식, 내포 관계 (nested cluster)가 생길 수 있음



분리형 클러스터링

(Partitional Clustering)

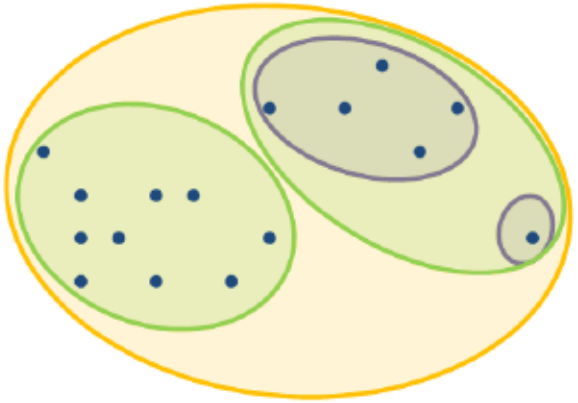


계층형 클러스터링

(Hierarchical Clustering)

1 클러스터링

계층적 클러스터링



사용자가 순차적인 부분 군집들 중에
원하는 정도를 선택 가능

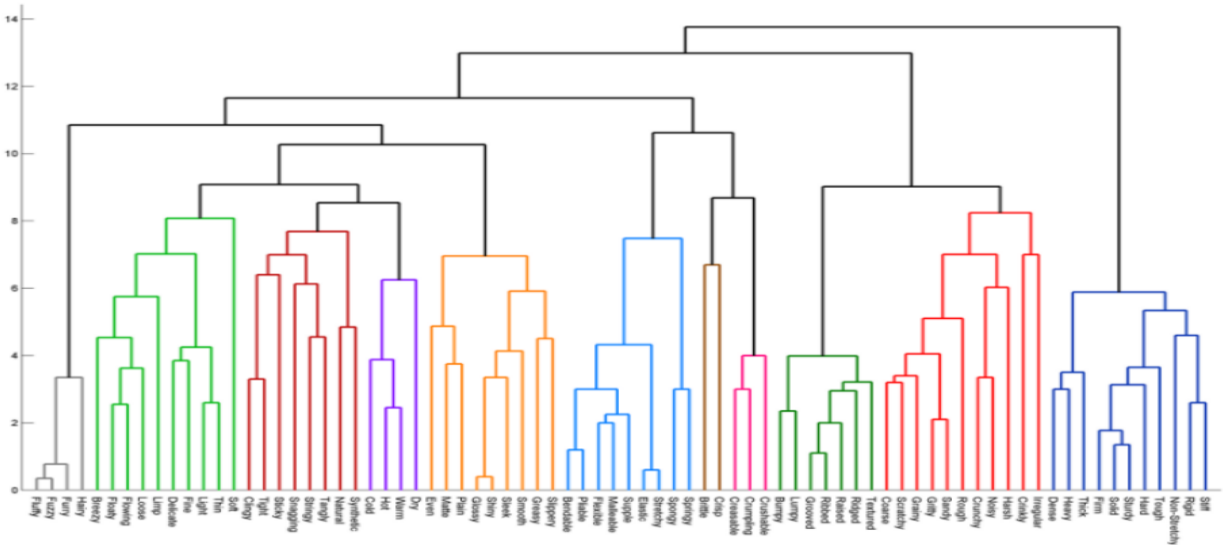
ex) 노란색 군집, 보라색 군집, 초록색 군집

계층적 클러스터링의 장점 ①

클러스터의 개수를 **사전에 정하지 않고** 학습이 가능함

1 클러스터링

계층적 클러스터링



계층적 클러스터링의 장점 ②

클러스터링 과정을 **덴드로그램**으로 시각화 가능

1 클러스터링

계층적 클러스터링

계층적 클러스터링은 **가장 가까운 객체나 집단을 순차적으로 병합**해나가는 방식

즉, 거리가 계산되어 있어야 함

개체 간 거리 측정 방식

Affirnity

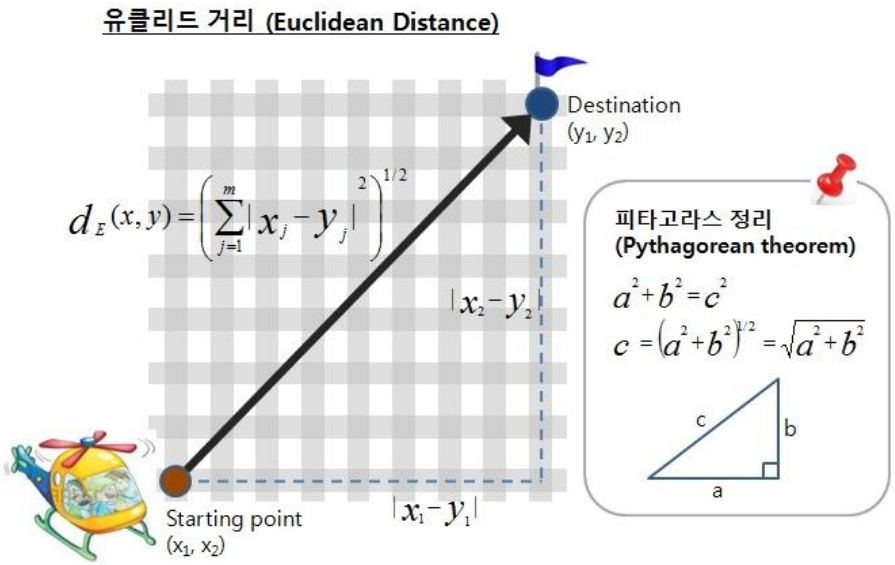
군집 간 거리 측정 방식

Linkage

1 클러스터링

계층적 클러스터링

Affinity (개체 간 거리 측정)



유클리드 (Euclidean) 거리

두 관측치 사이의 **직선** 최단 거리

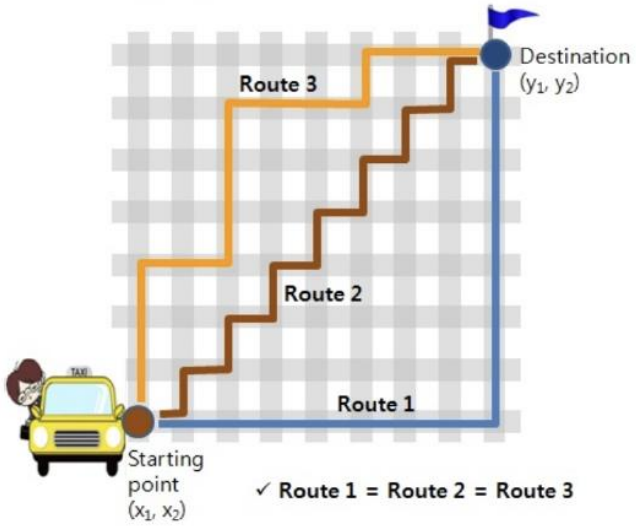
1 클러스터링

계층적 클러스터링

Affinity (개체 간 거리 측정)

맨하탄 거리
(Manhattan Distance)

$$d_M(x,y) = \sum_{j=1}^{m} | x_j - y_j |$$



맨하탄 (Manhattan) 거리

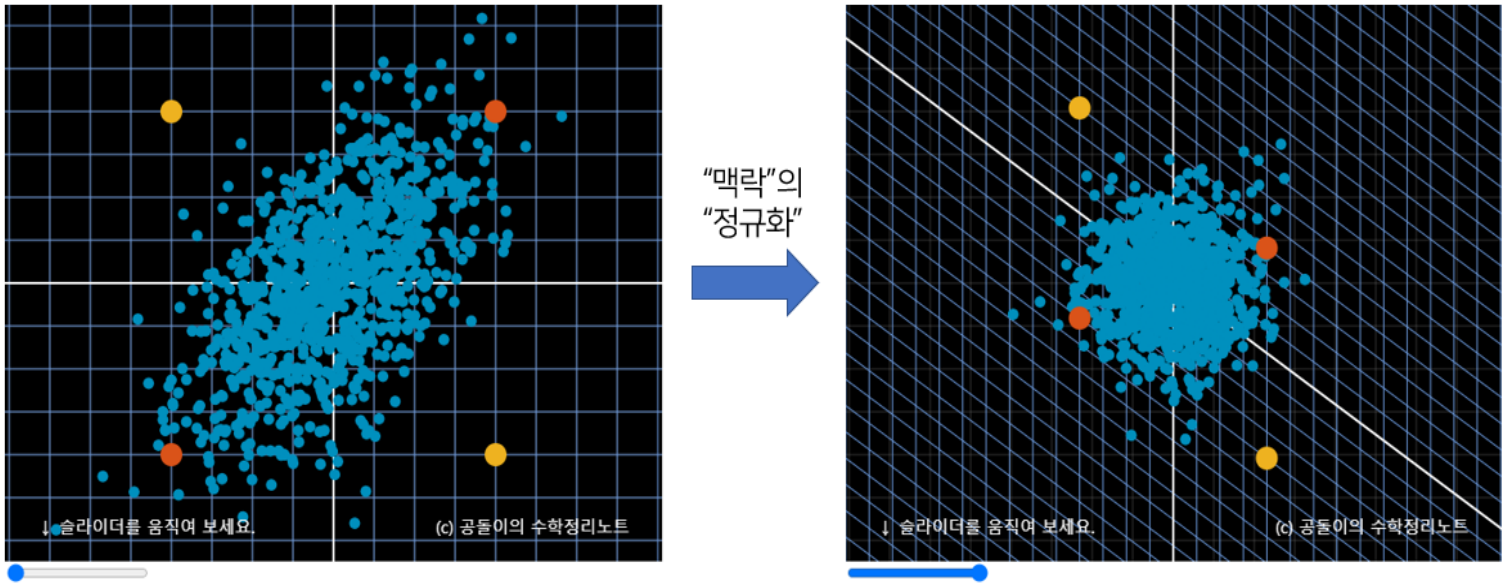
두 점 사이를 직선으로 이동하지 않고,

각 좌표축의 방향으로만 이동할 경우 계산되는 거리

1 클러스터링

계층적 클러스터링

Affinity (개체 간 거리 측정)



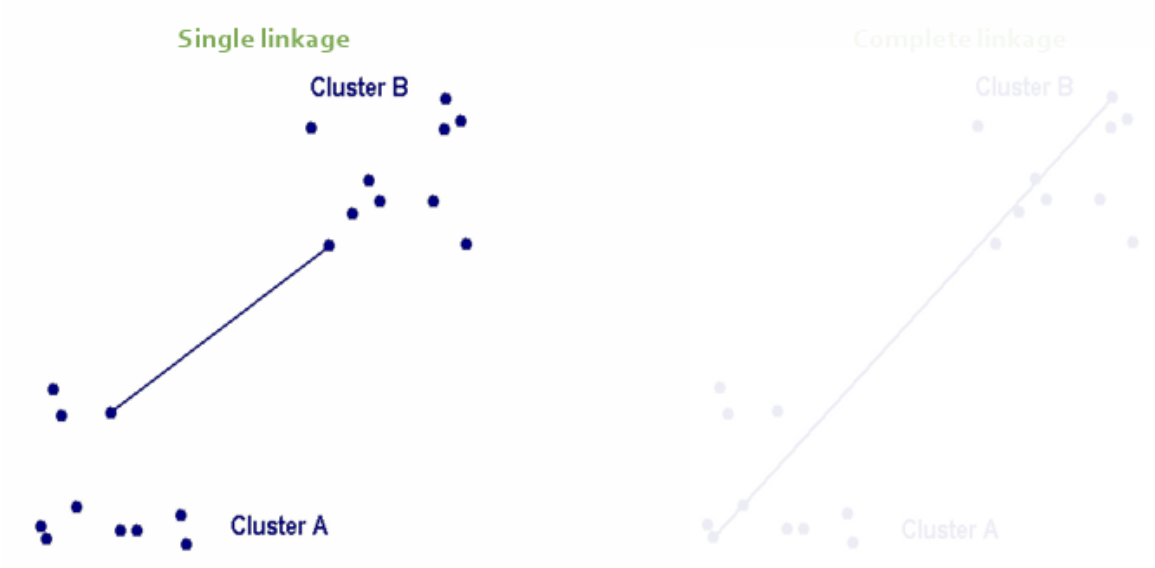
마할라노비스 거리

주변 데이터의 맥락 (변수 간 공분산, 주변 데이터 분포 등)을 반영하여 계산되는 거리

1 클러스터링

계층적 클러스터링

Linkage (군집 간 거리 측정)



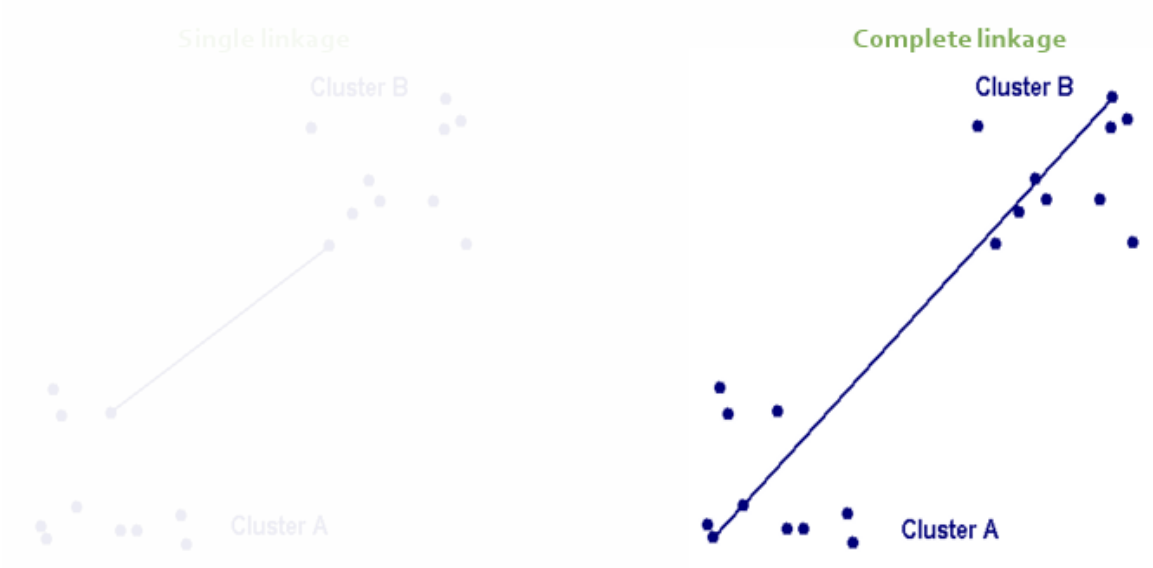
Single linkage (min)

서로 다른 군집의 **모든 데이터 간 거리 중 최소값**

1 클러스터링

계층적 클러스터링

Linkage (군집 간 거리 측정)



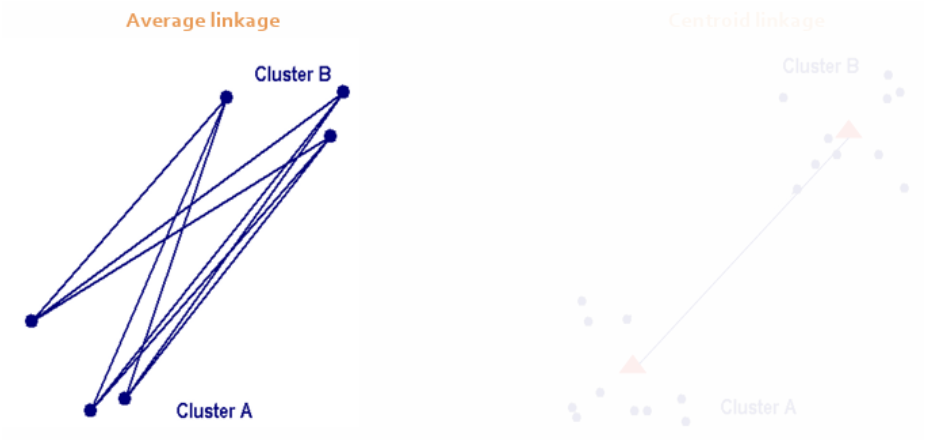
Complete linkage (max)

서로 다른 군집의 **모든 데이터 간 거리 중 최대값**

1 클러스터링

계층적 클러스터링

Linkage (군집 간 거리 측정)

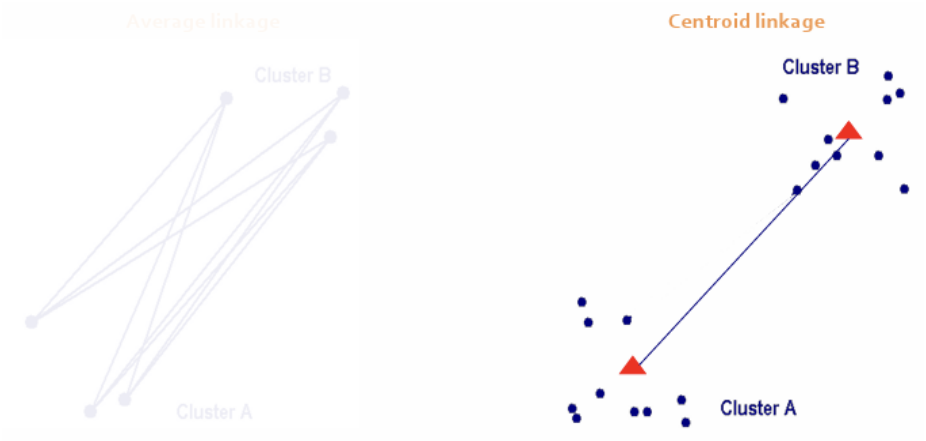


Average linkage

서로 다른 군집의 **모든 데이터 간 거리들의 평균**

계층적 클러스터링

Linkage (군집 간 거리 측정)



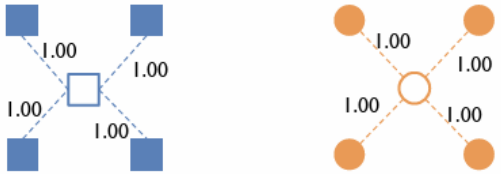
Centroid linkage

서로 다른 군집의 **중심점(Centroid) 간 거리**

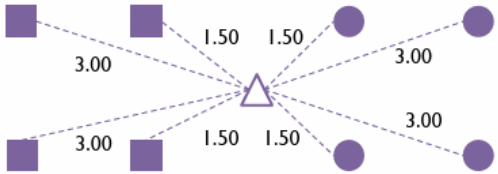
계층적 클러스터링

Linkage (군집 간 거리 측정)

- SSE before merge: $1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 = 8$



- SSE after merge: $4 \times 1.5^2 + 4 \times 3^2 = 45$



- Ward distance: 45-8 = 37

Ward Method

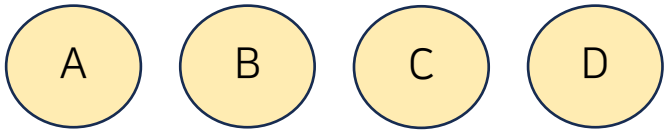
병합했을 때 SSE가 **적게 늘어날수록** 더 유사한 군집

위 예시의 경우, 파란 군집과 주황 군집 간 거리는 45-8=37

계층적 클러스터링

Agglomerative Clustering의 과정

	A	B	C	D
A		20	7	2
B	20		10	25
C	7	10		3
D	2	25	3	



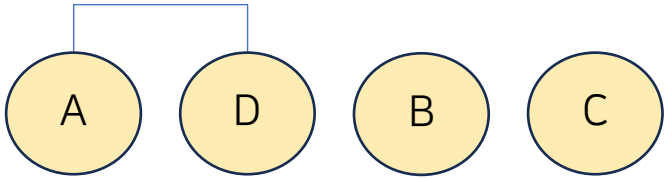
① 각 관측치 (또는 군집) 간의 거리를 모두 계산

주로 거리 행렬 (Distance Matrix) 사용

계층적 클러스터링

Agglomerative Clustering의 과정

	AD	B	C
AD			
B			
C			

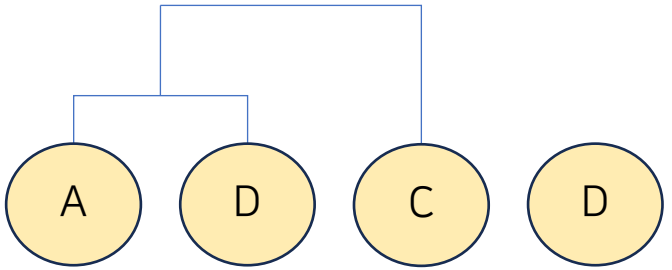


② 계산된 거리가 가장 가까운 두 관측치(또는 군집)을 병합

계층적 클러스터링

Agglomerative Clustering의 과정

	AD	B	C
AD		20	3
B	20		10
C	3	10	



③ 병합 이후 Distance Matrix를 업데이트

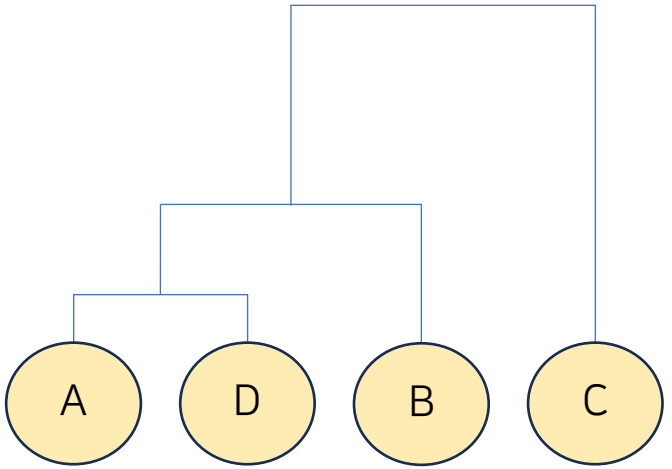
이 경우는 single linkage 사용

1 클러스터링

계층적 클러스터링

Agglomerative Clustering의 과정

	AD	B
AD		20
B	20	



④ 클러스터가 1개만 남을 때까지(= 전체가 다 묶일 때까지) 위 과정 반복

1 클러스터링

계층적 클러스터링

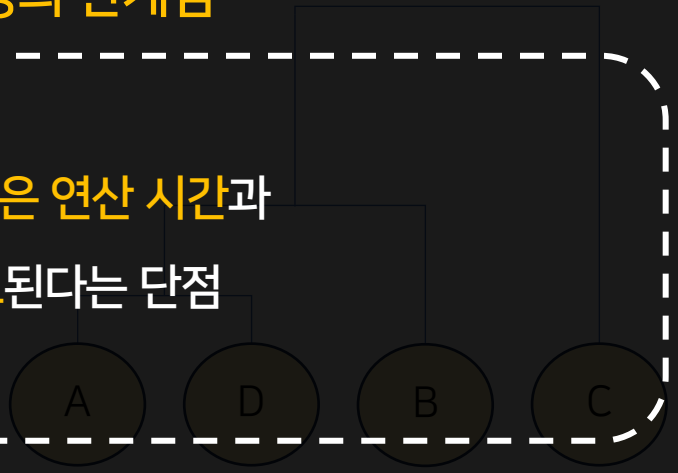
Agglemerative Clustering의 과정

상향식 클러스터링의 한계점

	AD	B
AD		20
B	20	

계산량이 많으므로 많은 연산 시간과

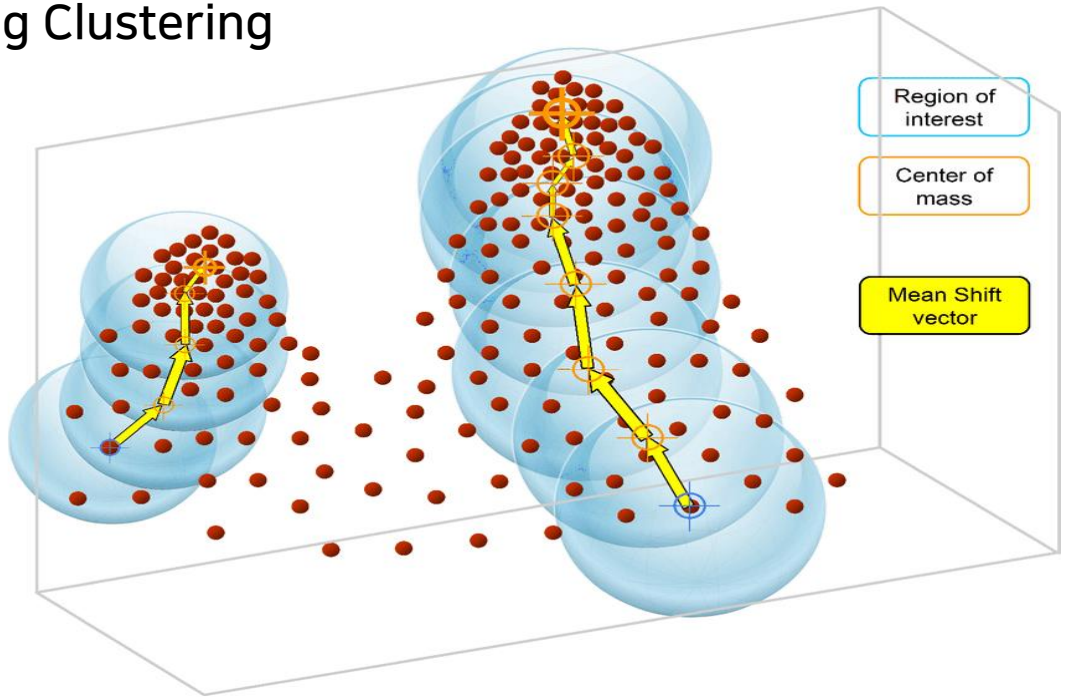
컴퓨팅 파워가 소모된다는 단점



④ 클러스터가 1개만 남을 때까지(=전체가 다 묶일 때까지) 위 과정 반복

그 외 클러스터링

Mean Shifting Clustering

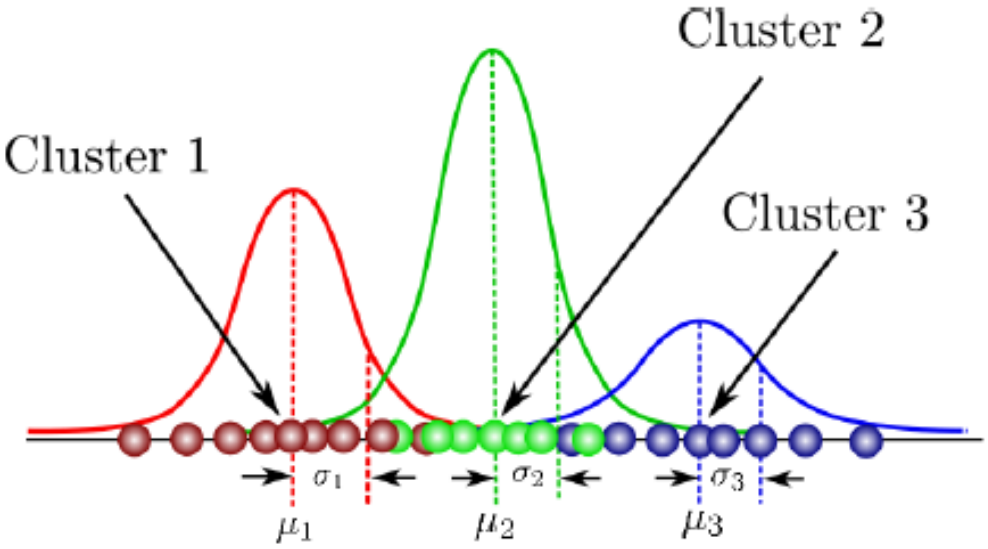


Centroid를 갱신할 때 데이터 분포를 활용하는 Mean Shifting Clustering

데이터의 **확률 밀도**가 더 높은 곳으로 중심을 이동

그 외 클러스터링

GMM (Gaussian Mixture Model)

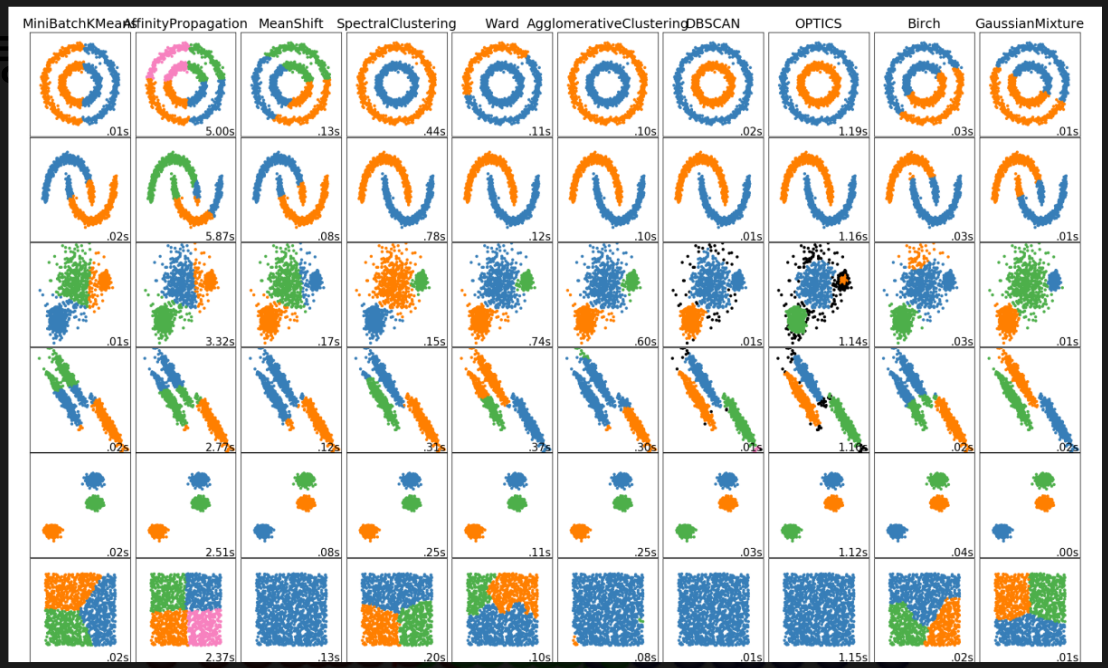


데이터가 여러 다른 모양의 가우시안 분포로 결합되어 있다는 가정 하에

개별 데이터를 동일한 분포로 묶어주는 GMM

1 클러스터링

그 외 클러스터링



이 외에도 데이터를 더 낮은 차원으로 사영하여 (= 차원축소)

클러스터링을 진행하는 Spectral Clustering 등

데이터가 여러 다른 모양의 가우시안 분포로 결합되어 있다는 가정 하에 개별 데이터를 동일한 분포로 묶어주는 GMM(Gaussian Mixture Model)

다양한 클러스터링 방법이 존재

2

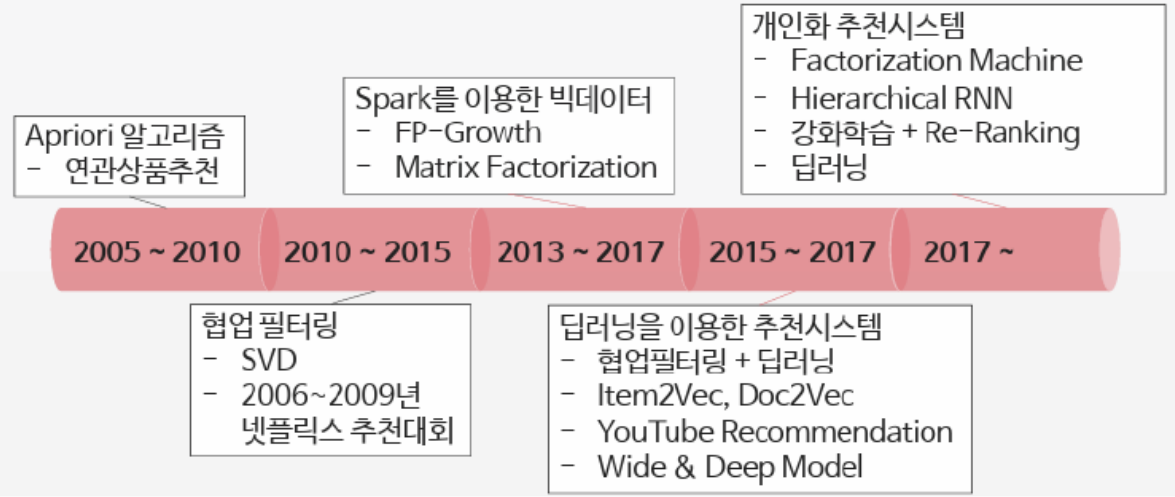
추천 시스템

추천 시스템

추천 시스템

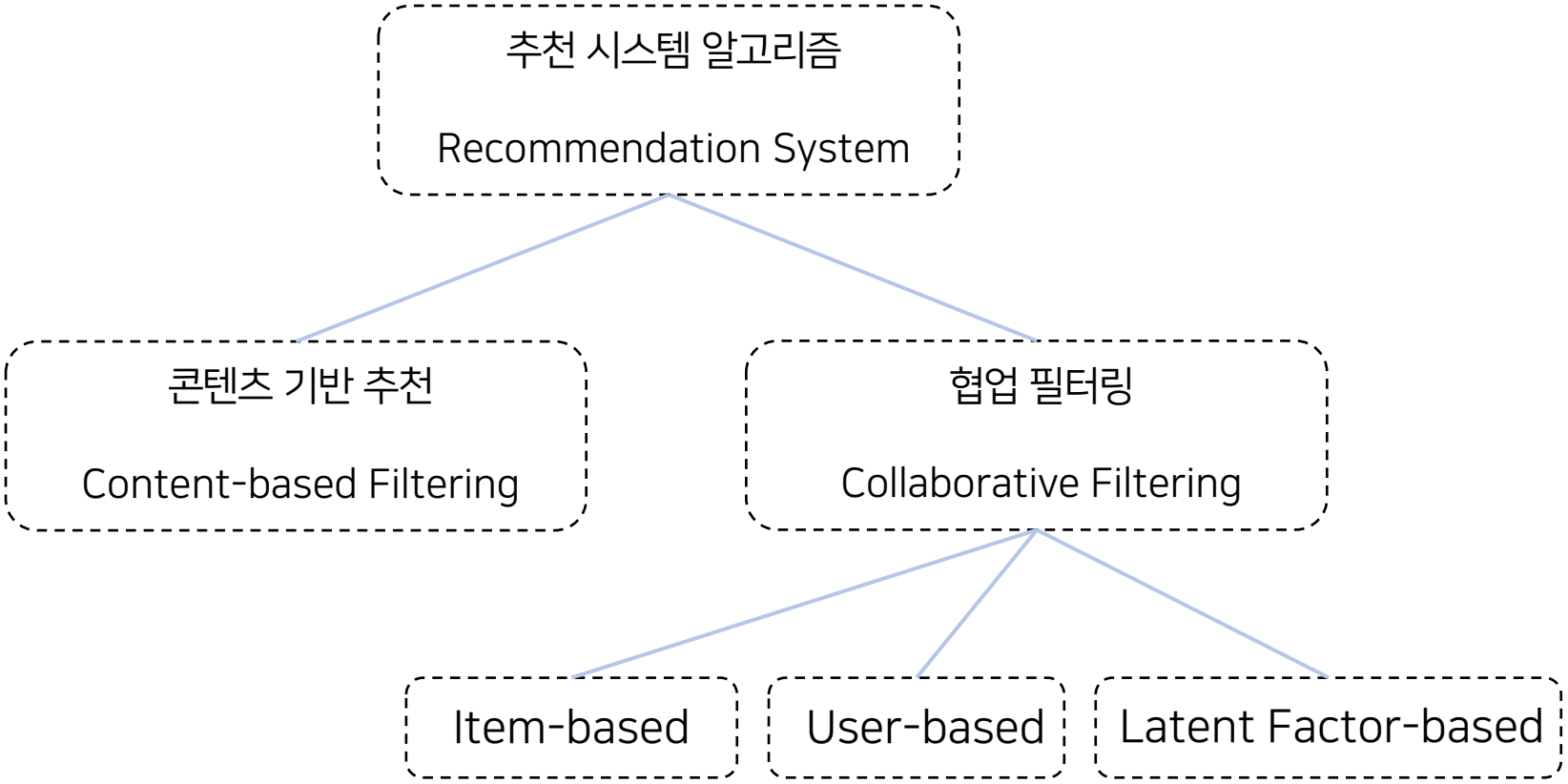
사용자에게 상품 혹은 서비스를 제안하는 소프트웨어 혹은 기술

ex) 다양한 콘텐츠 플랫폼에서 추천 시스템을 통한 개인 취향에 맞는 서비스 제공



➡ 현재 딥러닝, 강화학습과도 연계되며 개인화 추천시스템까지 계속해서 **고도화**되고 있음

추천 시스템


추천 시스템 알고리즘
Recommendation System
콘텐츠 기반 추천
Content-based Filtering
협업 필터링
Collaborative Filtering
Item-based
User-based
Latent Factor-based

2 추천 시스템

컨텐츠 기반 필터링 (Content-based Filtering)

컨텐츠 기반 필터링

사용자가 이전에 구매한 상품 중에서

좋아하는 상품들과 **유사한 상품들을 추천**하는 방법

<눈물의 여왕>을 재미있게 본 고객

비슷한 장르

같은 작가

메타데이터

장르, 작가, 배우, 줄거리 등 고객이

이미 소비한 **컨텐츠를 설명**하는 데이터

메타데이터를 활용해 고객이 소비한

상품의 특성을 분석하면,

유사한 특성을 지닌 상품을 추천 가능

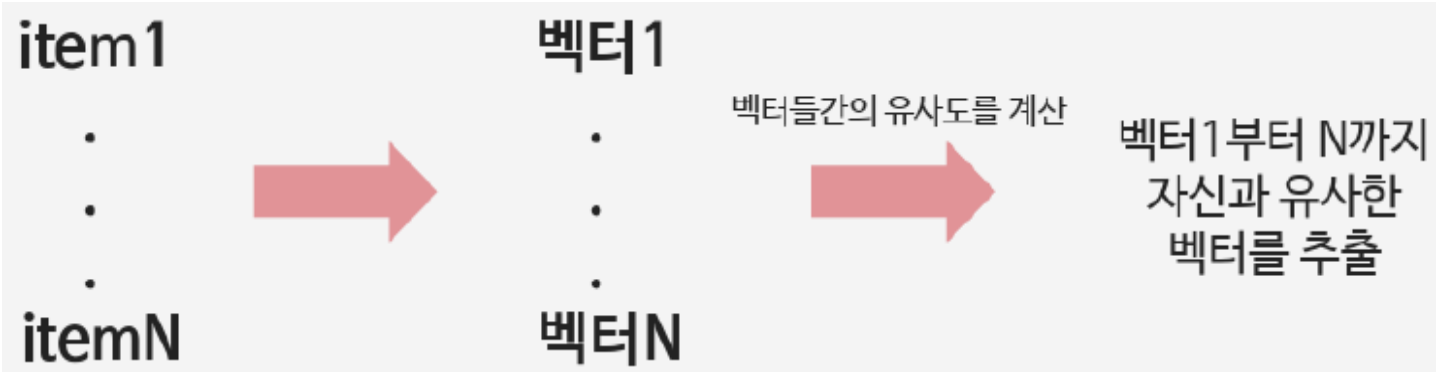
2 추천 시스템

컨텐츠 기반 필터링 (Content-based Filtering)

Key Idea

① 아이템을 **벡터 형태**로 표현

② 벡터들 간의 **유사도**를 계산 = **추천의 근거!**



2 추천 시스템

컨텐츠 기반 필터링 (Content-based Filtering)

Key Idea

① 아이템을 **벡터 형태**로 표현

② 벡터들 간의 **유사도**를 계산 = **추천의 근거!**

앞선 예시를 생각해 보면,
줄거리가 비슷한 드라마를
추천해주는 것이 가장 합리적

드라마의 줄거리와 같은 아이템을 어떻게 벡터로 나타낼 수 있을까?

2 추천 시스템

컨텐츠 기반 필터링 | 벡터 표현

Bag of Words (BoW)

단어들의 순서를 고려하지 않고,

단어의 출현 빈도에만 집중하는 텍스트 데이터의 수치화 표현 방법

① 각 단어에 고유한 정수 인덱스 부여

② 각 인덱스의 위치에 단어 토큰의 등장 횟수를 기록한 벡터 생성

Example

text data : "피셋의 학회장 김보근은 천사같고 피셋의 데마팀은 귀엽다."

vocabulary : {'피셋' : 0, '의' : 1, '학회장' : 2, '김보근' : 3, '은' : 4, '천사' : 5, '같고' : 6, '데마팀' : 7, '귀엽다' : 8}
BoW vector : {2, 2, 1, 1, 2, 1, 1, 1, 1}

2 추천 시스템

컨텐츠 기반 필터링 | 벡터 표현

단순 빈도 기반 방식의 단점

조사 '의'나 '은'처럼 **크게 중요하지 않은 단어** (불용어)들이

자주 등장할 수밖에 없음

불용어와 **중요한 단어**를 제대로 구분하기 위해

이에 **가중치**를 주는 방법을 생각하게 됨

컨텐츠 기반 필터링 | 벡터 표현

TF-IDF

다른 문서에서는 등장하지 않지만 **특정 문서에서만 자주 등장하는 단어**를 찾아
문서 내 단어의 가중치를 계산하는 방법

TF

특정 문서에서 특정 단어가
등장한 빈도

IDF

전체 문서에서 특정 단어가
등장한 빈도의 역수

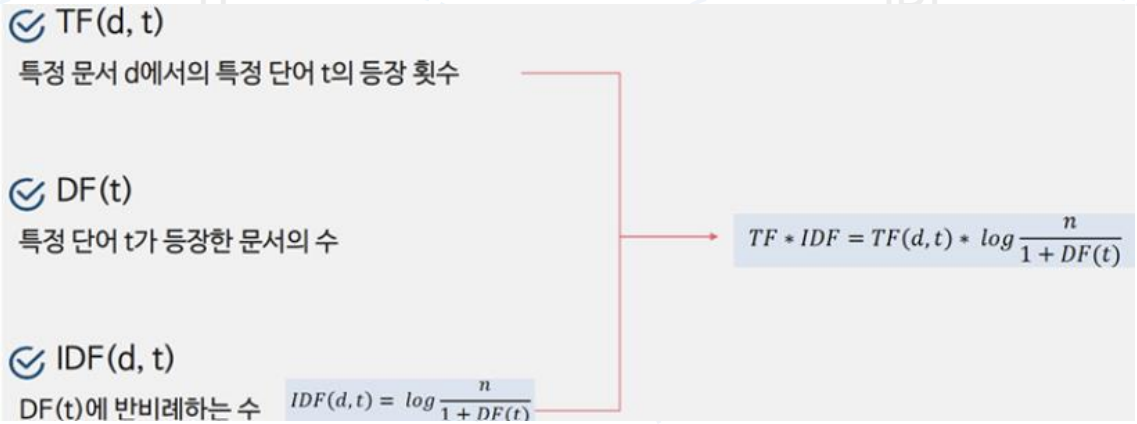
➡ TF-IDF의 기본적인 개념은 위의 단순 빈도수 문제를 개선하기 위해
단어의 빈도수에 역 빈도수를 곱해주는 것

2 추천 시스템

컨텐츠 기반 필터링 | 벡터 표현

TF-IDF

다른 문서에서는 등장하지 않지만 **특정 문서에서만 자주 등장하는 단어**를 찾아
문서 내 단어의 가중치를 계산하는 방법



단순히 모든 문서에서 많이 등장하는 단어에 페널티를 부여할 수 있고,
좀 더 정확하게 **특정 문서를 대표하는 단어**가 무엇인지 알 수 있음

2 추천 시스템

컨텐츠 기반 필터링 | 벡터 표현

그러나, 이러한 TF-IDF 방법은 **아주 높은 차원**을 갖게 되고

매우 sparse한 형태의 matrix를 갖는 **한계**가 있음

예를 들면, 100개의 문서를 비교할 때

문서마다 새로운 단어가 1000개씩만 있어도

100*100,000 matrix가 생성됨

이를 보완하기 위해 Word2Vec과 같은

추론 기반 임베딩을 사용할 수 있음

자세한 내용은 자연어처리팀 2주차 클린업 참고!

2 추천 시스템

컨텐츠 기반 필터링 | 유사도 함수

유클리디안 유사도

$$Euclidean\ Similarity = \frac{1}{Euclidean\ distance + 1e - 5}$$

유클리드 거리에 역수를 취해준 값

거리가 가까울수록 유사한 것

컨텐츠 기반 필터링 | 유사도 함수

Example)

문서	내용
0	먹고 싶은 사과
1	먹고 싶은 바나나
2	길고 노란 바나나 바나나
3	저는 과일이 좋아요

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서0	0	0	0	1	0	1	1	0	0
문서1	0	0	0	1	1	0	1	0	0
문서2	0	1	1	0	2	0	0	0	0
문서3	1	0	0	0	0	0	0	1	1

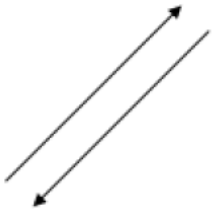
문서0과 문서1의 **유클리디안 유사도** : $\frac{1}{\sqrt{1^2+1^2}+1e-5} = 0.707$

2 추천 시스템

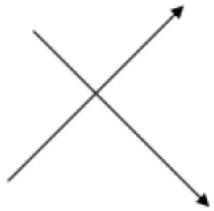
컨텐츠 기반 필터링 | 유사도 함수

코사인 유사도

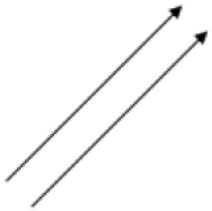
$$Cosine\ Similarity = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_1^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$



코사인 유사도 : -1



코사인 유사도 : 0



코사인 유사도 : 1

두 벡터가 가리키는 **방향이 얼마나 유사한가**를 의미하는 값

2 추천 시스템

컨텐츠 기반 필터링 | 유사도 함수

Example)

문서	내용
0	먹고 싶은 사과
1	먹고 싶은 바나나
2	길고 노란 바나나 바나나
3	저는 과일이 좋아요

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서0	0	0	0	1	0	1	1	0	0
문서1	0	0	0	1	1	0	1	0	0
문서2	0	1	1	0	2	0	0	0	0
문서3	1	0	0	0	0	0	0	1	1

문서0과 문서1의 **코사인 유사도** : $\frac{1+1}{\sqrt{1^2+1^2+1^2}+\sqrt{1^2+1^2+1^2}} = 0.6667$

2 추천 시스템

컨텐츠 기반 필터링 | 유사도 함수

Example)

문서	내용
0	먹고 싶은 사과
1	먹고 싶은 바나나
2	길고 노란 바나나 바나나
3	저는 과일이 좋아요

코사인 유사도의 장점

① 계산 과정에서 0*0 조합이 사라짐 → sparse한 데이터에서 더 정확한 유사도

	노란	먹고	바나나	사과			좋아요
문서0	0	0	1	0	1	1	0	0
문서1	0	0	1	1	0	1	0	0
문서2	0	1	1	0	2	0	0	0
문서3	1	0	0	0	0	0	0	0

② -1과 1 사이의 범위로 정규화된 지표 → 문서의 길이가 다른 상황 (두 벡터의 스케일 차이가 있는 상황)에서도 비교적 공정한 비교

문서0과 문서1의 코사인 유사도: $\frac{1+1}{\sqrt{1^2+1^2+1^2}+\sqrt{1^2+1^2+1^2}} = 0.6667$

컨텐츠 기반 필터링 | 유사도 함수

피어슨 유사도

$$Pearson\ Similarity = \frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\bar{Y})^2}}$$

두 변수 간 **선형 관계의 정도**를 수량화하는 척도

얼마나 직선같은지를 나타냄

컨텐츠 기반 필터링 | 유사도 함수

피어슨 유사도

$$Pearson\ Similarity = \frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\bar{Y})^2}}$$

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서0	0	0	0	1	0	1	1	0	0
문서1	0	0	0	1	1	0	1	0	0
문서2	0	1	1	0	2	0	0	0	0
문서3	1	0	0	0	0	0	0	1	1

문서0과 문서1의 **피어슨 유사도**를 계산해보면

$$\frac{1}{\sqrt{6\left(\frac{1}{3}\right)^2+3\left(\frac{2}{3}\right)^2}\cdot\sqrt{6\left(\frac{1}{3}\right)^2+3\left(\frac{2}{3}\right)^2}} = 0.5$$

2 추천 시스템

컨텐츠 기반 필터링 | 유사도 함수

피어슨 유사도는 다음과 같은 코사인 유사도의 **단점**을 **보완**

유저 A가 모든 영화에 **평점 5점**을 주고, **유저** B는 모든 영화에 **평점 1점**을 준 상황

({5,5,5,5,5}, {1,1,1,1,1})

두 유저의 취향은 완전 반대임에도 불구하고

코사인 유사도(=1)는 두 유저의 취향이 비슷하다고 **잘못 판단**을 내림

⋮

피어슨 유사도는 사용자의 평점값에서 평점의 평균값을 빼줌으로써

위의 문제를 어느 정도 **해결**

컨텐츠 기반 필터링 | 유사도 함수

자카드 유사도

$$Jaccard\ Similarity = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

합집합 대비 교집합의 비율로 두 집합 **A**와 **B**의 유사도를 계산하는 방법

2 추천 시스템

컨텐츠 기반 필터링 | 유사도 함수

자카드 유사도

$$Jaccard\ Similarity = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

자카드 유사도 역시 0*0 조합은 고려하지 않으므로

sparse한 데이터에서 사용하기 좋으며,

앞선 예시의 경우 $\frac{2}{3+3-2}$ = 0.5로 계산됨

2 추천 시스템

컨텐츠 기반 필터링 (Content-based Filtering)

장점

- Cold-start

(서비스 초반 누적 데이터의 부족으로 제대로 된 추천이 어려운 문제)

- 현상에 크게 구애받지 않음

단점

- 새로운 사용자에게는 추천이 불가능

추천시스템의 고질적인 문제

- 메타데이터로부터 주요 feature를 추출하기 어렵다는 한계

신규 사용자 가입 시 선호 아이템을 물어보는 방식을 사용해볼 수 있음

2 추천 시스템

협업 필터링 (Collaborative Filtering)

협업 필터링

다른 사람의 구매 패턴이나 평점 등 **사용자 행동양식**(User Behavior)만을 기반으로 추천

사용자 기반 (User-based Collaborative Filtering)

아이템 기반 (Item-based Collaborative Filtering)

잠재 요인 기반 (Latent-factor Collaborative Filtering)

2 추천 시스템

협업 필터링 (Collaborative Filtering)

협업 필터링

다른 사람의 구매 패턴이나 평점 등 **사용자 행동양식**(User Behavior)만을 기반으로 추천

	오펜하이머	아바타	듄: 파트2	서울의 봄	라라랜드
동기	5	4	4	3	1
형석	1	0	1	3	4
종혁	4	4	2	5	3
경미	4	2	3	2	2
주원	5	3	1	2	???

평점 행렬을 기반으로 추천을 수행

평점 행렬에 축적된 데이터를 기반으로, 아직 사용자가 평가하지 않은 아이템에 대한 평점을 **예측**

2 추천 시스템

협업 필터링 | 사용자 기반

사용자 기반 협업 필터링

사용자의 구매 패턴과 유사한 '사용자'를 찾아서

추천 리스트를 생성하는 방식

	오펜하이머	아바타	듄: 파트2	서울의 봄	라라랜드
동기	5	4	4	3	1
형석	1	0	1	3	4
종혁	4	4	2	5	3
경미	4	2	3	2	2
주원	5	3	1	2	???

"당신과 비슷한 취향을 가진 고객들이 이 영화도 관람했습니다!"

2 추천 시스템

협업 필터링 | 사용자 기반

	오펜하이머	아바타	듄: 파트2	서울의 봄	라라랜드	Pearson(i,주원)
동기	5	4	4	3	1	0.71713
형석	1	0	1	3	4	-0.27144
종혁	4	4	2	5	3	0.42656
경미	4	2	3	2	2	0.56061
주원	5	3	1	2	???	1.0

피어슨 유사도를 활용하여 유사도 계산

$$Pearson\ Similarity = \frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\bar{Y})^2}}$$

2 추천 시스템

협업 필터링 | 사용자 기반

	오펜하이머	아바타	듄: 파트2	서울의 봄	라라랜드	Pearson(i,주원)
동기	5	4	4	3	1	0.71713
형석	1	0	1	3	4	-0.27144
종혁	4	4	2	5	3	0.42656
경미	4	2	3	2	2	0.56061
주원	5	3	1	2	???	1.0

주원이와 가장 비슷한 Top-2 사용자는 동기와 경미임을 알 수 있음

$$\text{Pearson Similarity} = \frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i-\bar{Y})^2}}$$

동기와 경미가 높은 점수를 준 영화들을 주원이에게도 추천

2 추천 시스템

협업 필터링 | 사용자 기반

	Pearson(i,주원)
동기	0.71713
형석	-0.27144
종혁	0.42656
경미	0.56061
주원	1.0

주원이가 아직 관람하지 않은

<라라랜드>에 대한 평점도 **예측**해볼 수 있음

$$\frac{(0.717 * 1) + (-0.271 * 4) + (0.427 * 3) + (0.561 * 2)}{0.717 - 0.271 + 0.427 + 0.561} = 1.42$$

분석자의 **주관**에 따라 추천 여부 **결정**

Ex) 예측 평점이 **3점**이 넘었을 때만 추천을 해주자!

2 추천 시스템

협업 필터링 | 아이템 기반

아이템 기반 협업 필터링

사용자들이 매긴 평점이 유사한 '상품'을 찾아서
추천 리스트를 생성하는 방식

	오펜하이머	아바타	듄: 파트2	서울의 봄	라라랜드
동기	5	4	4	3	1
형석	1	0	1	3	4
종혁	4	4	2	5	3
경미	4	2	3	2	2
주원	5	3	1	2	???

"그 영화를 관람한 고객들이 이 영화도 관람했습니다!"

2 추천 시스템

협업 필터링 | 아이템 기반

	오펜하이머	아바타	듄: 파트2	서울의 봄	라라랜드
동기	5	4	4	3	1
형석	1	0	1	3	4
종혁	4	4	2	5	3
경미	4	2	3	2	2
주원	5	3	1	2	???
Pearson (오펜하이머,j)	1.0	0.9045	0.8944	0.0765	-0.8944

피어슨 유사도를 활용하여 **유사도** 계산

<오펜하이머>를 재밌게 본 사용자에게 **<아바타>**를 **추천**

2 추천 시스템

협업 필터링 | 아이템 기반

사용자 or 아이템 기반 협업 필터링의 한계

	오펜하이머	아바타	듄: 파트2	서울의 봄	라라랜드
동기	5	4	4	3	1
형석	1	0	1	3	4
종혁	4	4	2	0	3
경미	4	3	2	1	2
주원	5	3	3	2	???
Pearson (오펜하이머,j)	1	0.72	0.72	0.65	-0.894

매우 직관적이고 합리적으로 보이는 이 방법도

메모리나 시간 측면에서는 **매우 비효율적**임

사용자와 아이템이 늘어날수록

평점 행렬의 크기는 기하급수적으로 커짐

피어슨 유사도를 활용하여 **유사도** 계산

<오펜하이머>를 재미있게 봤는데 [illegible] 사용자에게 <듄: 파트2>를 **추천**

잠재요인 협업 필터링으로 해결!

협업 필터링 | 잠재 요인

잠재 요인 협업 필터링

평점 행렬을 **저차원의 행렬들로 분해**하는 방식

사용자와 아이템 간에 어떤 **잠재 요소**가 있다고 가정함

명확하게 정의 할 순 없지만,

사용자가 상황에 맞게 유추 가능함.

ex) 장르, 연기력, 스토리

협업 필터링 | 잠재 요인

잠재 요인 협업 필터링

평점 행렬을 **저차원의 행렬들로 분해**하는 방식

사용자와 아이템 간에 어떤 **잠재 요소**가 있다고 가정함

사용자 – 아이템 행렬은

사용자 – 잠재요인 행렬과 **잠재요인 – 아이템 행렬**의 곱으로 표현

2 추천 시스템

협업 필터링 | 잠재 요인

	킹스맨	노팅힐	다크 나이트	이터널 선샤인	미션 임파서블
동기	6	12	20	12	3
형석	12	10	6	8	10
종혁	14	7	9	4	13
경미	16	4	12	0	16

<사용자 : 영화 > 행렬

장르를 **잠재요인**으로 고려

(사용자 : 영화) = (사용자 : 장르) * (장르 : 영화) 로 분해

4X5 4X2 2X5

2 추천 시스템

협업 필터링 | 잠재 요인

	Romance	Action
동기	3	0
형석	2	2
종혁	1	3
경미	0	4

	킹스맨	노팅힐	다크 나이트	이터널 선샤인	미션 임파서블
Romance	2	4	0	4	1
Action	4	1	3	0	4

특이값 분해를 통해 (사용자 : 장르), (장르 : 영화) 라는 2개의 행렬 생성

2 추천 시스템

협업 필터링 | 잠재 요인

자세한 내용은 리드오프 3주차 후반부 참고!

'특이값 분해'는 특이값을 원소로 갖는 대각행렬 Σ 에서
가장 작은 특이값부터 제외하며 차원을 축소하는 것

저장 공간을 절약함과 동시에
주요한 잠재요인만을 고려하여 좀 **더 정교한 추천**을 가능하게 함

특이값 분해를 통해 (사용자 : 장르), (장르 : 영화) 라는 2개의 행렬 생성

2 추천 시스템

협업 필터링 | 잠재 요인

	Romance	Action
동기	3	0
형섭	2	2
승혁	1	3
경미	0	4

	킹스맨	노팅힐	다크 나이트	이터널 선샤인	미션 임파서블
Romance	2	4	0	4	1
Action	4	1	3	0	4

하지만 여전히 **Cold – Start 문제**나

평점 행렬의 특성으로 인해 발생하는 **협업 필터링의 근본적인 문제점**은 남아 있음

협업 필터링 | 잠재 요인

SVD 협업 필터링의 한계

현실의 대부분의 평점행렬은 대부분의 원소가 비어있는 (결측치인)

희소행렬 (Sparse Matrix)의 형태를 띰

	Romance	Action
동기	3	0
형석	2	2
종혁	1	3
경미	0	4

	킹스맨	노팅힐	다크 나이트	이터널 선샤인	다션 임파서블
			0	4	1
			3	0	4

	아이템1	아이템2	아이템3	아이템4	아이템5
사용자1	4.0			2.0	
사용자2		5.0		3.0	1.0
사용자3			3.0	4.0	4.0
사용자4	5.0	2.0	1.0	2.0	

따라서 SGD, ALS 등 결측치를 원래 행렬과 비슷하도록

예측하는 최적화 과정이 필요함

2 추천 시스템

평가 지표

① Precision@K

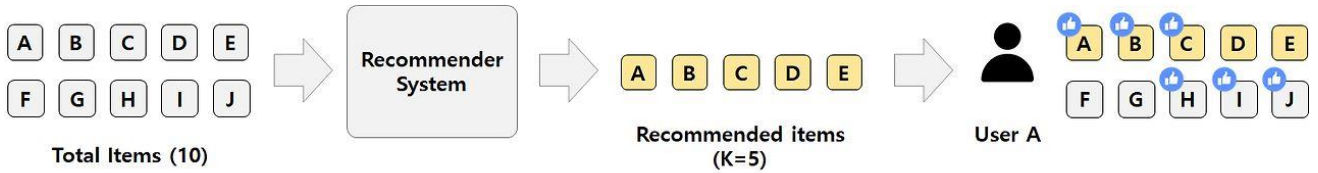
추천한 아이템 K개 중에

실제 사용자가 관심있는 아이템의 비율

② Recall@K

사용자가 관심있는 모든 아이템 중에서

추천한 아이템 K개의 비율



$$Precision@5 = \frac{3}{5} = 0.6 \qquad Recall@5 = \frac{3}{6} = 0.5$$

2 추천 시스템

평가 지표

③ MAP@K

Precision/Recall@K 와 달리 **순서를 고려**하여

관심을 더 가질만한 아이템을 상위에 추천했는지 여부를 반영한 지표

추천 시스템에서는 사용자가 **관심을 더 많이 가질만한 아이템을**

상위에 추천해주는 것이 중요함!

평가 지표

③ MAP@K

AP@K = Precision@K의 평균

Recommendations	Precision@K's	AP@3
[0, 0, 1]	[0, 0, 1/3]	(1/3)(1/3) = 0.11
[0, 1, 1]	[0, 1/2, 2/3]	(1/3)(1/2 + 2/3) = 0.38
[1, 1, 0]	[1/1, 2/2, 2/3]	(1/3)(1/1 + 2/2 + 2/3) = 0.89
[1, 1, 1]	[1/1, 2/2, 3/3]	(1/3)(1/1 + 2/2 + 3/3) = 1

4명의 사용자에게
3개씩 추천한 상황

$$MAP@3 = (0.11 + 0.38 + 0.89 + 1) * \frac{1}{4} = 0.595$$

MAP@K = AP@K의 평균

평가 지표

③ MAP@K

AP@K = Precision@K의 평균

Recommendations	Precision@K's	AP@3
[0, 0, 1]	[0, 0, 1/3]	(1/3)(1/3) = 0.11
[0, 1, 1]	[0, 1/2, 2/3]	(1/3)(1/2 + 2/3) = 0.38
[1, 1, 0]	[1/1, 2/2, 2/3]	(1/3)(1/1 + 2/2 + 2/3) = 0.89
[1, 1, 1]	[1/1, 2/2, 3/3]	(1/3)(1/1 + 2/2 + 3/3) = 1

이때, 고객2와 고객3은 추천 일치 개수가 2개로 같음에도

높은 순위에 추천한 것을 맞혔을수록 점수가 더 높음을 확인!

평가 지표

④ NDCG@K

추천 순서에 가중치를 두어 평가하며,
추천 일치 여부 대신 **관련성 (Relevance)**을 활용

사용자가 특정 아이템과
얼마나 관련이 있는지를 나타내는 값
쉽게 말해 , **'중요도'**

이 값은 사용자가 직접 지정

평가 지표

④ NDCG@K

추천 순서에 가중치를 두어 평가하며,
추천 일치 여부 대신 **관련성 (Relevance)**을 활용

NDCG@K 값은 **CG, DCG**를 거쳐
차례대로 정의될 수 있음

평가 지표

CG

추천한 아이템의
Relavance 합계

$$CG_K = \sum_{i=1}^{K} rel_i$$

DCG

CG에다 순서에 따른
할인을 도입

$$DCG_K = \sum_{i=1}^{K} \frac{rel_i}{\log_2(i+1)}$$

NDCG

DCG를 IDCG로
정규화한 값

$$NDCG_K = \frac{DCG}{IDCG}$$

최선의 추천을 했을 때
받는 DCG 값

2 추천 시스템

평가 지표

CG

추천한 아이템의 Relevance 합계

$$CG_K = \sum_{i=1}^{K} rel_i$$

DCG

CG에다 순위에 따른 할인을 도입

$$DCG_K = \sum_{i=1}^{K} \frac{rel_i}{\log_2(i+1)}$$

결론적으로, NDCG@K는 **가장 이상적인 추천 조합 대비 현재 모델의 추천 성능**을 나타내는 지표

정규화를 해서 0~1 사이의 값을 가지게 되므로, 다른 추천 모델과 비교하기 좋음

NDCG

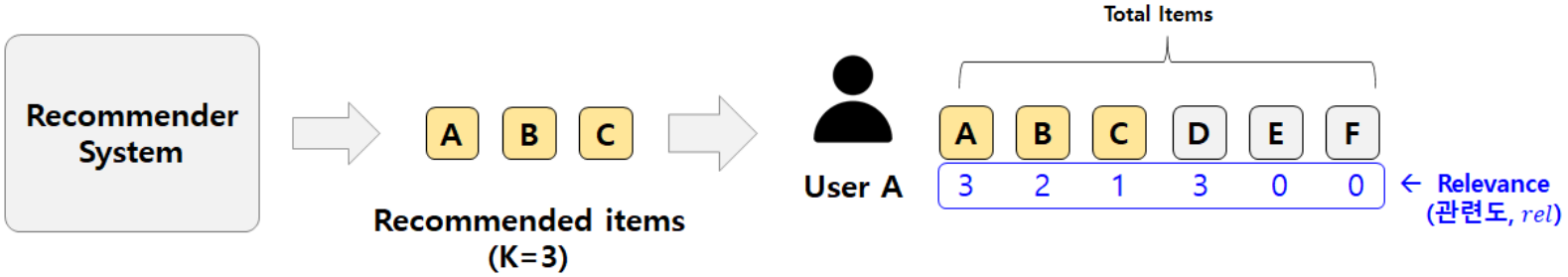
DCG를 IDCG로 정규화한 값

$$NDCG_K = \frac{DCG}{IDCG}$$

최선의 추천을 했을 때 받는 DCG 값

평가 지표

[NDCG 계산 과정]



$$CG_3 = \sum_{i=1}^{K} rel_i = rel_1 + rel_2 + rel_3 = 3 + 2 + 1 = 6$$

$$DCG_3 = \sum_{i=1}^{K} \frac{rel_i}{log_2(i+1)} = \frac{3}{log_2(1+1)} + \frac{2}{log_2(2+1)} + \frac{1}{log_2(3+1)} = \frac{3}{1} + \frac{2}{1.58} + \frac{1}{2} = 4.78$$

$$IDCG_3 = \sum_{i=1}^{K} \frac{rel_i^{opt}}{log_2(i+1)} = \frac{3}{log_2(1+1)} + \frac{3}{log_2(2+1)} + \frac{2}{log_2(3+1)} = \frac{3}{1} + \frac{3}{1.58} + \frac{2}{2} = 5.89$$

$$NDCG_3 = \frac{DCG}{IDCG} = \frac{4.78}{5.89} = 0.81$$

평가 지표

⑤ MAE/RMSE

'평점 예측'에 대한 평가 방법

평균 절댓값 오차인 MAE와 **오차 제곱의 평균의 제곱근**인 RMSE를 사용

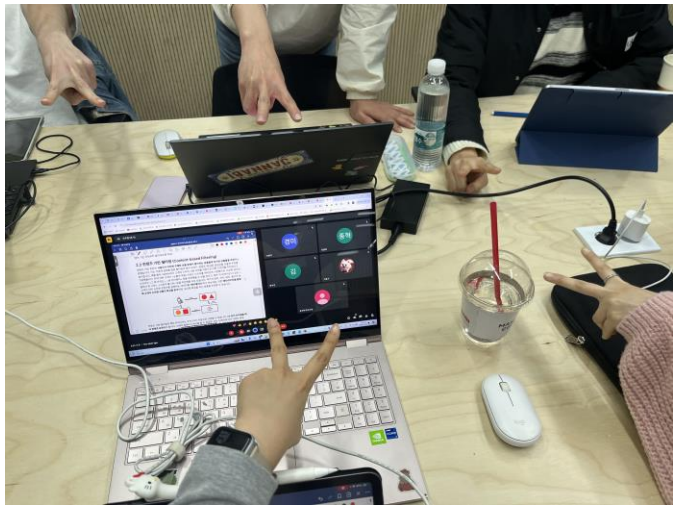
$$MAE = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}|$$

실제 비즈니스 관점에서는 참신성이나

클릭 증가율 등 다른 **다양한 지표들도 함께 고려된다**는 점 참고 !

화목한 르데마핌


photoism
KEEP YOURSELF ALIVE
PHOTOISM
KEEP YOURSELF ALIVE
PHOTOISM



THANK YOU

♡ 르**데마**핌 클린업 3주간 수고 많았습니다 !! ♡