군집

Hands-On

경제학과 2015231035 하지민

군집

비지도 학습

많은 데이터는 대부분 레이블이 없음
레이블을 부여하는 작업은 비용이 크며 오래 걸림
=> 레이블이 없는 데이터를 바로 사용하기 위한 **비지도 학습**

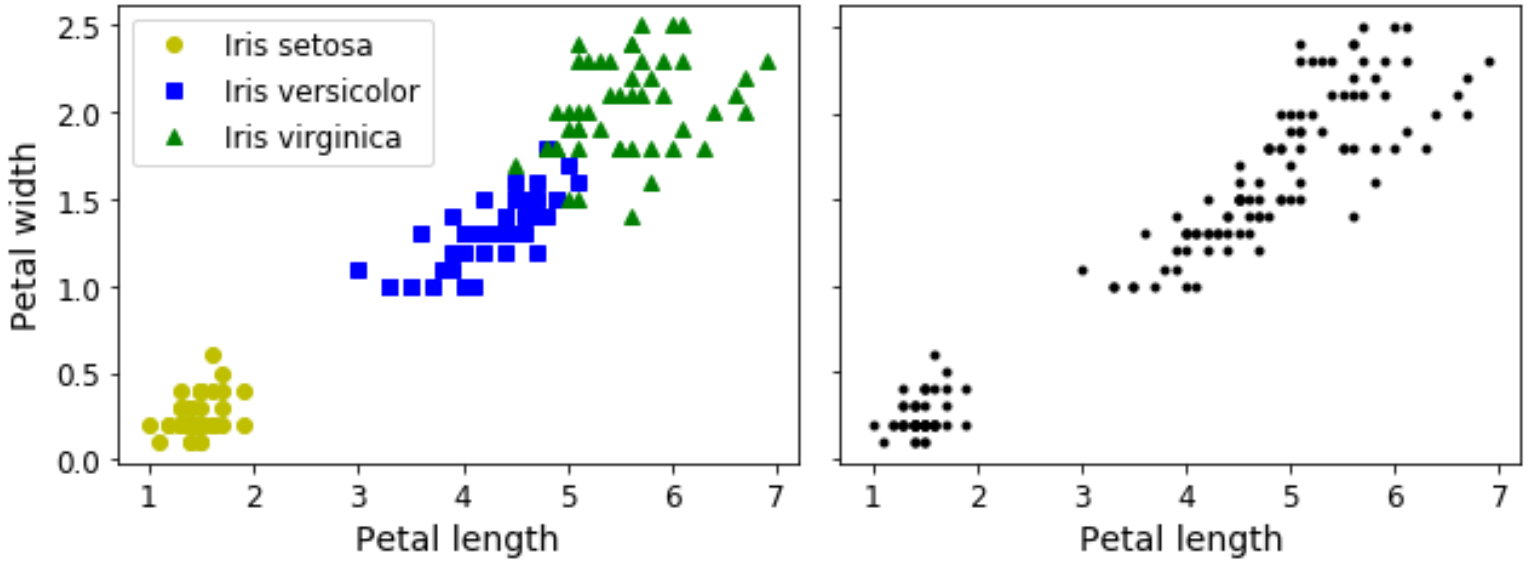
군집

비슷한 샘플을 구별해 하나의 클러스터(cluster) 또는 비슷한 샘플의 그룹으로 할당하는 작업

클러스터

클러스터에 대한 보편적 정의 없음
알고리즘이 다르면 다른 종류의 클러스터를 감지
어떤 알고리즘은 센트로이드라 부르는 특정 포인트를 중심으로 모인 샘플을 탐색

분류 vs 군집



분류	군집
데이터셋이 레이블 되어 있음 로지스틱 회귀, 랜덤 포레스트 분류기 알고리즘이 잘 맞음	레이블이 없음 대부분의 군집 알고리즘은 왼쪽 아래 클러스터를 쉽게 감지하지만, 오른쪽 위의 클러스터는 두 개의 하위 클러스터로 구성되어 있는지 확실하지 않음 모든 특성을 사용하면 세 개의 클러스터 구분 가능

군집이 사용되는 어플리케이션

k-평균

k-평균

레이블이 없는 데이터셋 샘플 중, 덩어리로 잘 묶어지는 듯한 종류의 데이터셋을 빠르고 효율적으로 클러스터로 묶을 수 있는 알고리즘

각 클러스터의 중심을 찾고 가장 가까운 클러스터에 샘플을 할당

군집에서 각 샘플의 레이블은 알고리즘이 샘플에 할당한 클러스터의 인덱스(분류의 레이블과 혼동X)

K-평균 알고리즘은 클러스터의 크기가 많이 다르면 잘 작동하지 않음
샘플을 클러스터에 할당할 때 센트로이드까지 거리를 고려하는 것이 전부이기 때문

하드 군집

각 샘플에 대해 가장 가까운 클러스터 선택하는 것

소프트 군집

클러스터마다 **점수**를 부여하는 것

고차원 데이터셋을 변환할 때,
매우 효율적인 비선형 차원 축소 기법

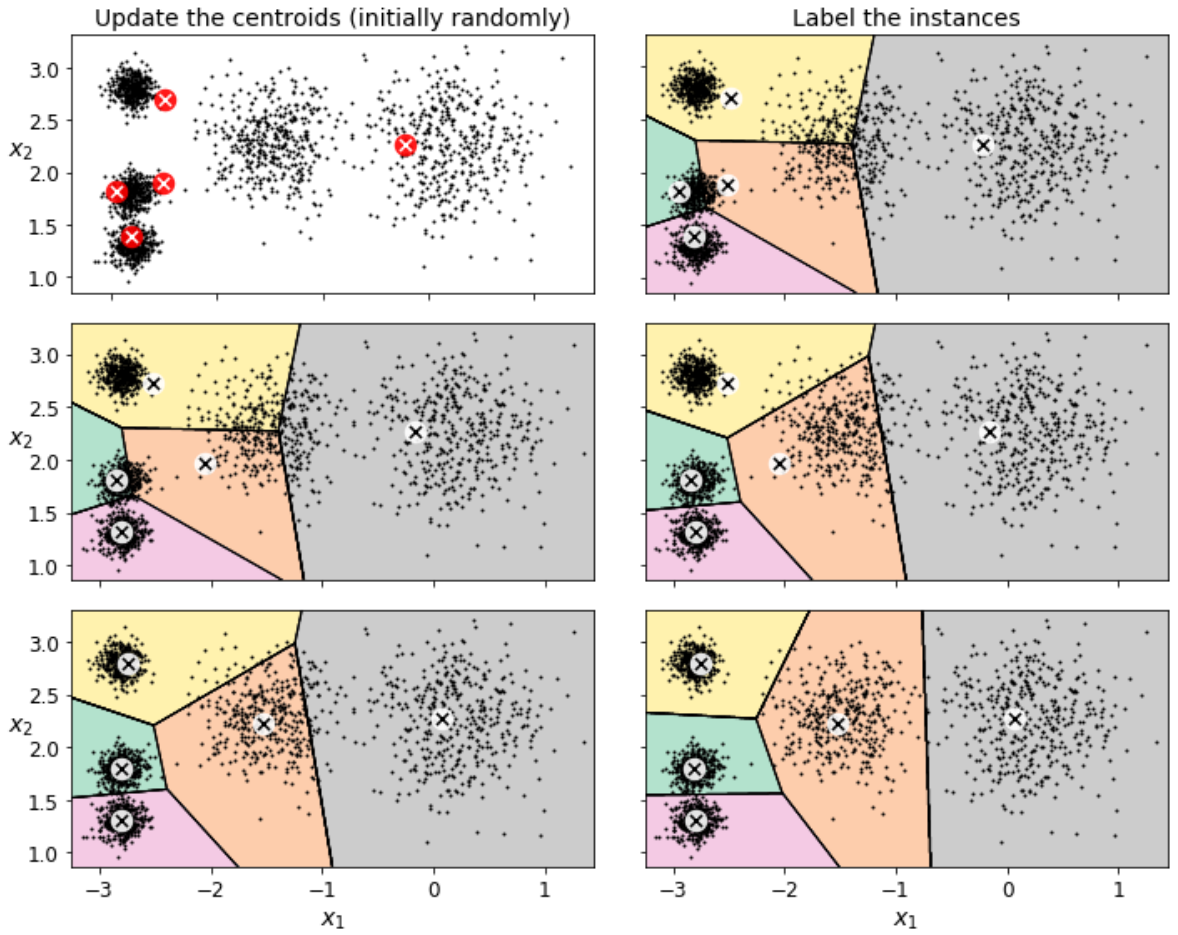
※ 여기서 말하는 점수란?

샘플과 센트로이드 사이의 거리
(Kmeans 클래스의 transform() 메서드 통해 구할 수 있음)

유사도점수

K-평균 알고리즘 작동 원리

1. Centroid를 랜덤하게 초기화
2. 샘플에 레이블을 할당
3. Centroid 업데이트
4. 샘플에 다시 레이블을 할당
5. 최적으로 보이는 cluster에 도달할 때까지 위 과정들을 반복

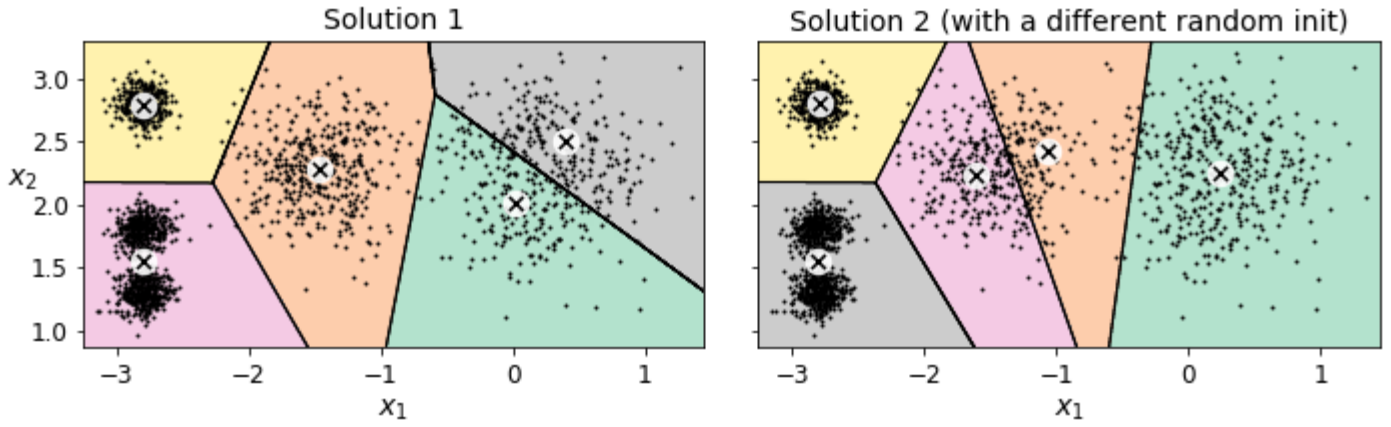


센트로이드 초기화 방법

이 알고리즘이 수렴하는 것이 보장되지만, 적절한 솔루션으로 수렴하지 못할 수도 있음

이는 센트로이드 초기화에 달려 있음(운에 따라 다름)

센트로이드 초기화를 개선하여, 아래와 같은 위험을 줄일 수 있음



센트로이드 초기화 방법

다른 군집 알고리즘을 먼저 실행하여 센트로이드 위치를 근사하게 파악

또는 랜덤 초기화를 다르게 하여 여러 번 알고리즘을 실행하고 가장 좋은 솔루션을 선택

- 최선의 솔루션을 얻기 위해 성능 지표로 모델의 이너셔(inertia) 사용
- 이너셔는 각 샘플과 가장 가까운 센트로이드 사이의 평균 제곱 거리
- Kmeans 클래스는 알고리즘을 n_init 번 실행해 이너셔가 가장 높은 모델을 반환

센트로이드 초기화 방법

k-평균++ 알고리즘

다른 센트로이드와 거리가 먼 센트로이드를 선택하는 똑똑한 초기화 단계를 거침

- k-평균 알고리즘이 최적이 아닌 솔루션으로 수렴할 가능성을 크게 낮춤
- Kmeans 클래스는 기본적으로 이 초기화 방법 사용

1. 가지고 있는 데이터 포인트 중 랜덤하게 1개를 선택하여 첫 번째 센트로이드로 지정
2. 나머지 데이터 포인트들에 대해 그 첫 번째 센트로이드까지의 거리 계산
3. 두번째 센트로이드는 각 점들로부터 거리비례 확률에 따라 선택
   즉, 두 번째 센트로이드은 첫 번째 센트로이드으로부터 최대한 먼 곳에 배치된 데이터 포인트로 지정한다는 뜻
4. 센트로이드가 k개가 될 때까지 2,3번을 반복

초기 중심점을 더욱 전략적으로 배치하기 때문에 전통적인 K-Means보다 더 최적의 군집화 가능

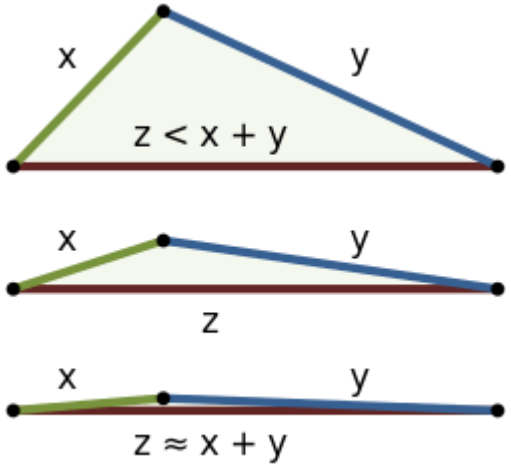
K-Means보다 알고리즘이 수렴하는 속도가 빠름
전통적인 K-Means를 사용하면 초기 중심점이 (재수 없게) 잘못 지정되는 경우 알고리즘이 수렴하는 데 시간이 오래 걸릴 수 있기 때문

k-평균 속도 개선과 미니배치 k-평균

k-평균 속도 개선

불필요한 거리 계산을 피함으로써 알고리즘의 속도를 높임

- 삼각 부등식 사용(두 점사이의 직선은 항상 가장 짧은 거리가 됨)
- 샘플과 센트로이드 사이의 거리를 위한 하한선과 상한선을 유지
- KMeans 클래스에서 기본으로 사용



미니배치 k-평균

전체 데이터셋을 사용해 반복하지 않고 각 반복마다 미니배치를 사용해 센트로이드를 조금씩 이동

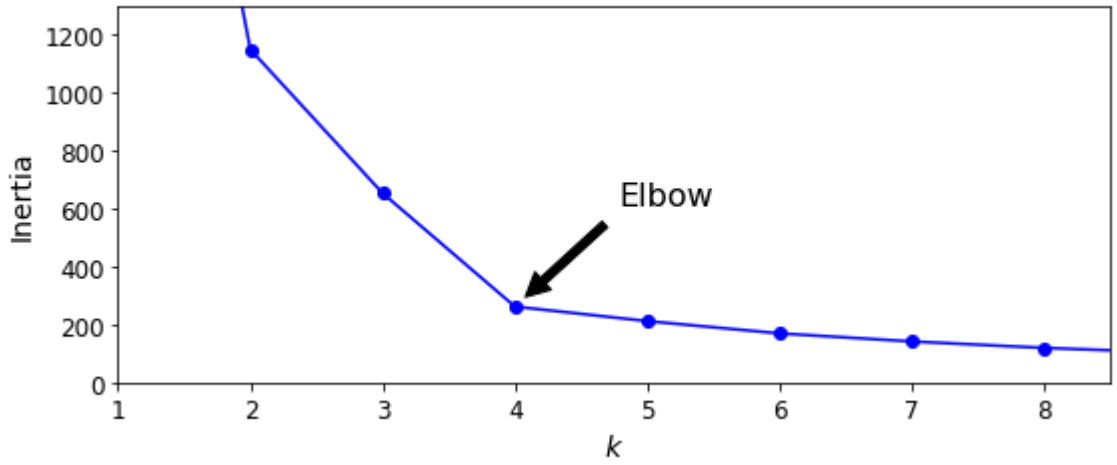
- 알고리즘 속도 3~4배 정도 높임
- 메모리에 들어기지 않는 대량의 데이터셋에 군집 알고리즘을 적용 가능

But 일반 k-평균보다 훨씬 빠르지만, 이너셔는 조금 더 나쁨
(특히 클러스터의 개수가 증가할 때)

최적의 클러스터 개수 찾기

1. 이너셔 그래프를 그래프를 클러스터 개수(k)의 함수로 그렸을 때, 그래프가 꺾이는 지점(Elbow) 찾기

- 아래 그래프를 보면, k가 4까지 증가할 때 이너셔는 빠르게 감소
- 하지만 k가 계속 증가할수록 이너셔의 감소 폭은 크게 줄어드는 것을 알 수 있음
- 따라서 이 그래프가 꺾이는 지점. 즉, Elbow가 위치한 k값을 최적의 클러스터 개수로 지정해주면 됨



최적의 클러스터 개수 찾기

2. 실루엣 점수 및 실루엣 다이어그램

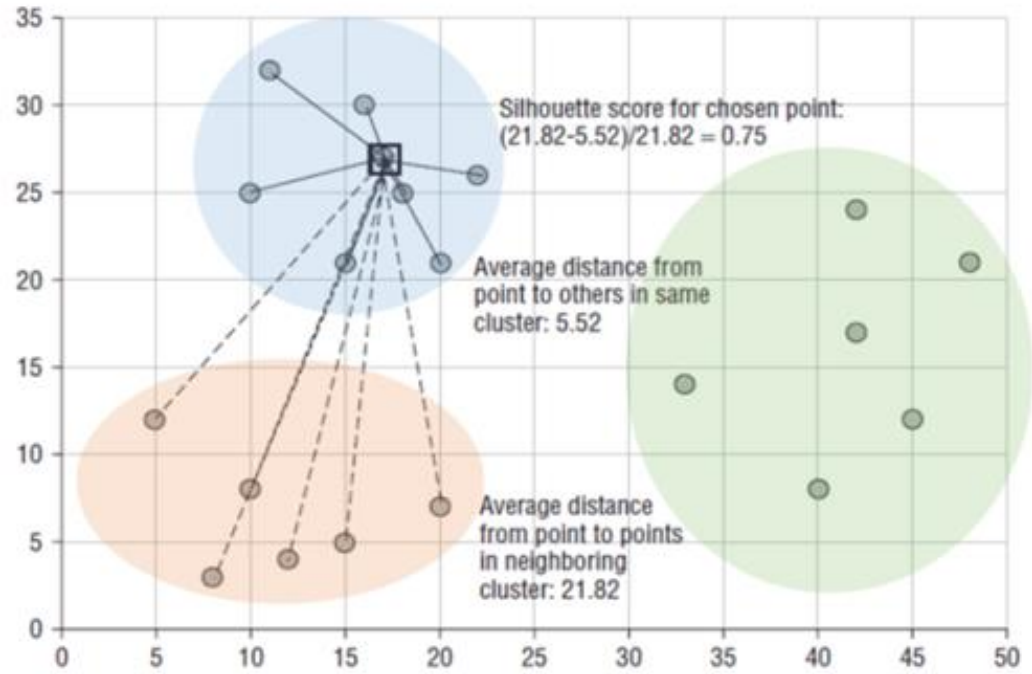
- 각 군집 간의 거리가 얼마나 효율적으로 분리 되어 있는지를 나타냄

- 다른 군집과의 거리는 떨어져 있고, 동일 군집끼리의 데이터는 서로 가깝게 잘 뭉쳐 있는지 확인

- 군집화가 잘 되어 있을 수록 개별 군집은 비슷한 정도의 여유 공간을 가지고 있을 것

- 실루엣 계수

개별 데이터가 가지는 실루엣 계수는 해당 데이터가 같은 군집 내의 데이터와 얼마나 가깝게 군집화 돼있고, 다른 군집에 있는 데이터와는 얼마나 멀리 분리돼 있는지를 나타내는 지표



최적의 클러스터 개수 찾기

2. 실루엣 점수 및 실루엣 다이어그램

- 실루엣 계수

a(i) : 해당 데이터 포인트와 같은 군집 내에 있는 다른 데이터 포인트와의 거리를 평균한 값
b(i) : 해당 데이터 포인트가 속하지 않은 군집 중 가장 가까운 군집과의 평균 거리

두 군집 간의 거리 값은 b(i) - a(i) 이며 이 값을 정규화 하기 위해 Max(a(i),b(i)) 값으로 나눈다. 따라서 i 번째 데이터 포인트의 실루엣 계수 값 s(i) 는 다음과 같이 정의된다.

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

실루엣 계수는 -1 에서 1사이의 값을 가지며, 1로 가까워 질수록 근처의 군집과 더 멀리 떨어져 있다는 것이고, 0에 가까울 수록 근처의 군집과 가까워 진다는 것
- 값은 아예 다른 군집에 데이터 포인트가 할당 되었음을 뜻함

최적의 클러스터 개수 찾기

2. 실루엣 점수 및 실루엣 다이어그램

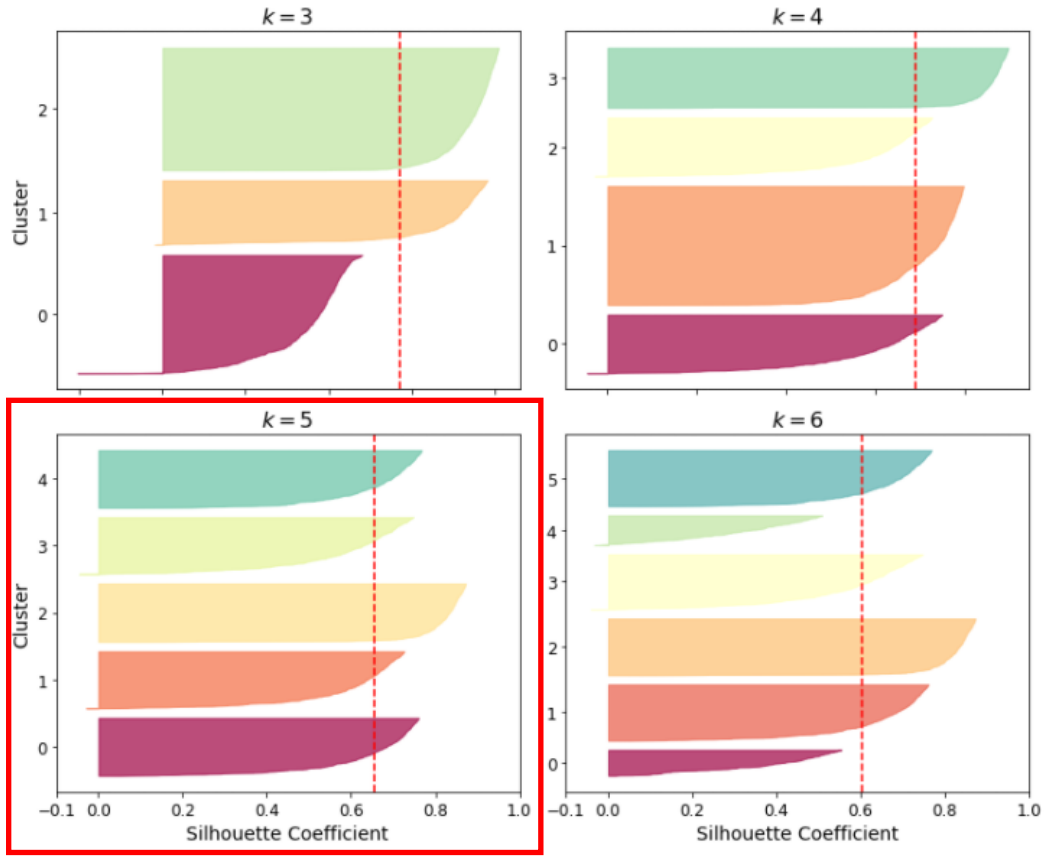
- 좋은 군집화의 조건

1. 전체 실루엣 계수의 평균값, 즉 사이킷런의 silhouette_score() 값은 0~1 사이의 값을 가지며, 1에 가까울 수록 좋음

2. 전체 실루엣 계수의 평균값과 더불어 개별 군집의 평균값의 편차가 작아야 함

즉 개별 군집의 실루엣 계수 평균값이 전체 실루엣 계수의 평균값에서 크게 벗어나지 않는 것이 중요

만약 전체 실루엣 계수의 평균값은 높지만, 특정 군집의 실루엣 계수 평균값만 유난히 높고 다른 군집들의 실루엣 계수 평균값은 낮으면 좋은 군집화가 아님



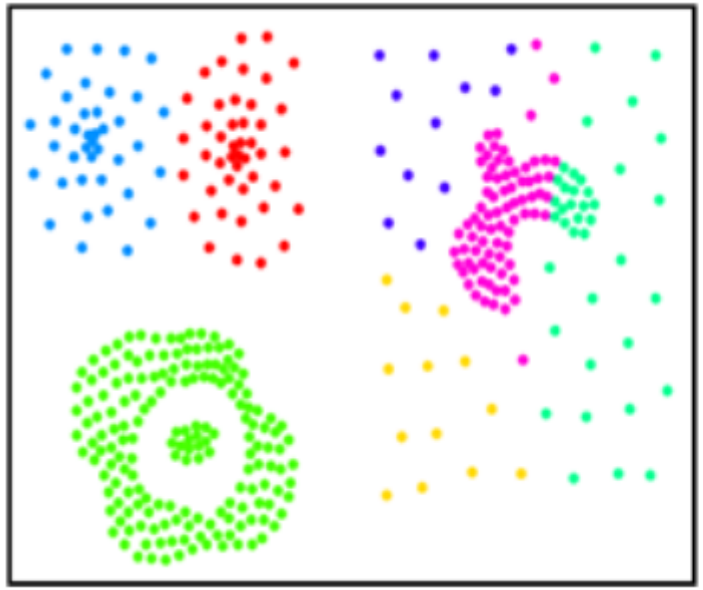
k-평균의 한계

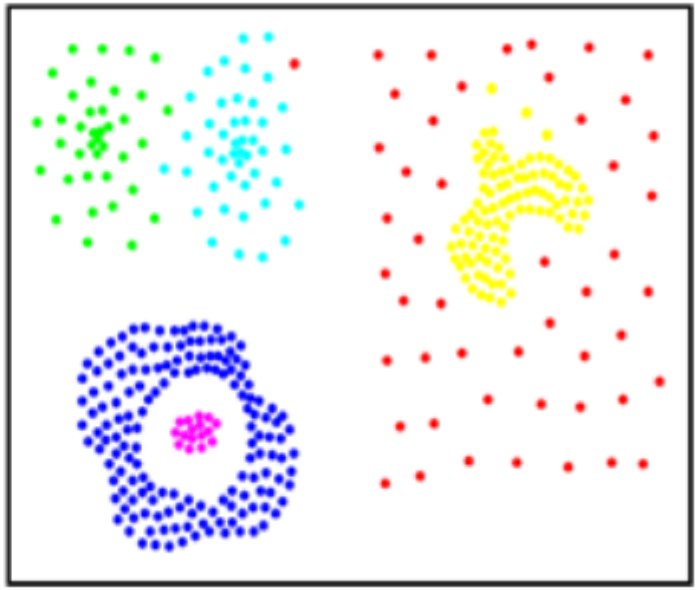
- K-means 알고리즘은 속도가 빠르고 확장이 용이하다는 장점을 갖고 있다.
- 하지만 최적의 cluster 개수(k)를 직접 지정해줘야 한다는 점, 그리고 최적의 솔루션에 도달하지 못하는 문제를 해결하기 위해 알고리즘을 여러 번 실행해주어야 한다는 점이 꽤나 번거롭다.
- 또한 k-means 알고리즘은 cluster의 크기나 밀집도가 서로 다르거나, 원형이 아닐 경우(타원형 등과 같은 경우) 잘 작동하지 않는다.
- 참고로 타원형 cluster에서는 가우시안 혼합 모델(GMM)이 잘 작동한다.
- 마지막으로 k-means 알고리즘을 실행하기 전에 입력 특성의 스케일을 맞춰주는 것은 굉장히 중요하다.

DBSCAN

비선형 클러스터의 군집이나 다양한 크기를 갖는 공간 데이터를 보다 효과적으로 군집하기 위해 이웃한 개체와의 밀도를 계산하여 군집하는 기법

K-Means와 같이 군집 이전에 클러스터의 개수가 필요하지 않고 잡음에 대한 강인성이 높기 때문에 현재까지도 다양한 분야에서 활용



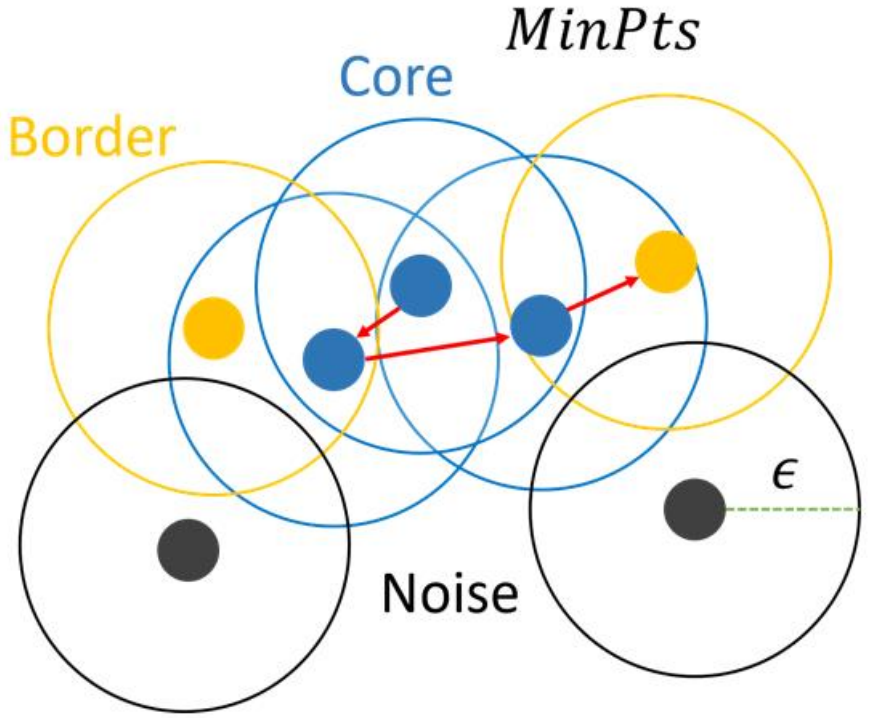


DBSCAN

알고리즘

사실 DBSCAN은 컴퓨팅 알고리즘으로 제안된 기법이기 때문에 특별한 수식 존재하지 않음
따라서 2가지 파라미터만 기억!
이웃과의 거리를 나타내는 최소 이웃 반경 ϵ과 최소 이웃 수 minPts

1. 각각의 객체들은 반경 ϵ을 기준으로 최소 이웃 minPts 이상을 충족하면 군집
2. 최소 이웃 minPts를 충족하지 못하면 잡음으로 판단
3. 군집이 되었다면 군집된 이웃 객체를 대상으로 1번 반복



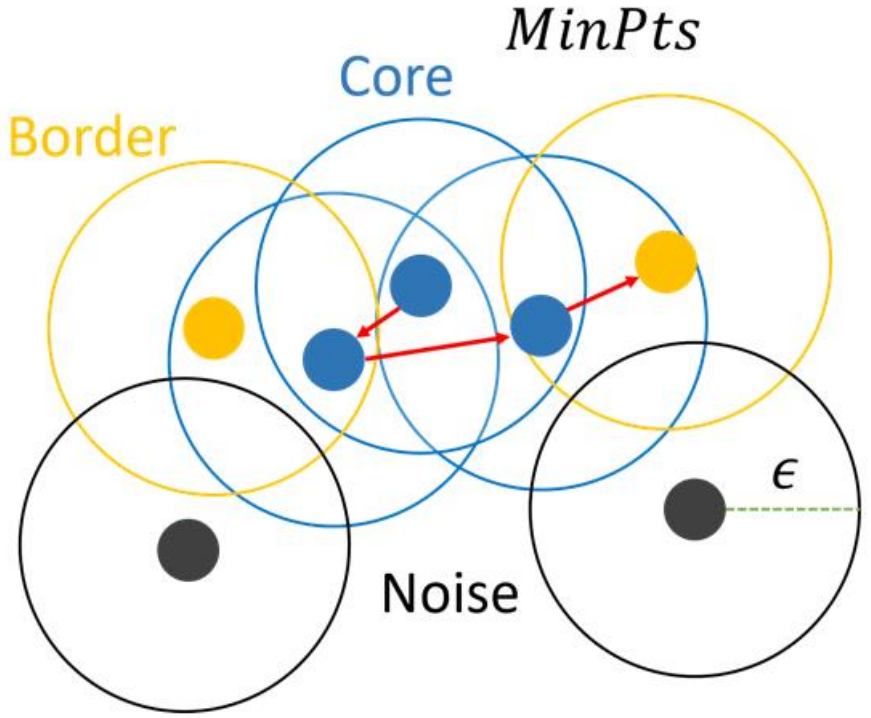
DBSCAN

K-Means처럼 전체 데이터의 분포 특성을 통해 중심점을 추정하는 방식이 아닌, 개별적인 객체 하나하나의 밀도 특성을 1-3번 알고리즘을 반복하며 군집
모든 객체는 한 번 연산된 이후에는 다시 군집 되는 일이 없도록 Flag를 지정해줌
처음 시작하는 객체는 초기화를 통해 무작위로 뽑히게 되며 군집이 완료되면 그림과 같이 군집에 속하는 Core와 Border, 그리고 Outlier로 판단되는 Noise 값으로 분류할 수 있음

Core Vector(object) : 임의의 벡터로부터 반경 ϵ내에 있는 이웃 벡터의 수가 minPts 보다 클 경우, 하나의 군집을 생성하며 중심이 되는 벡터

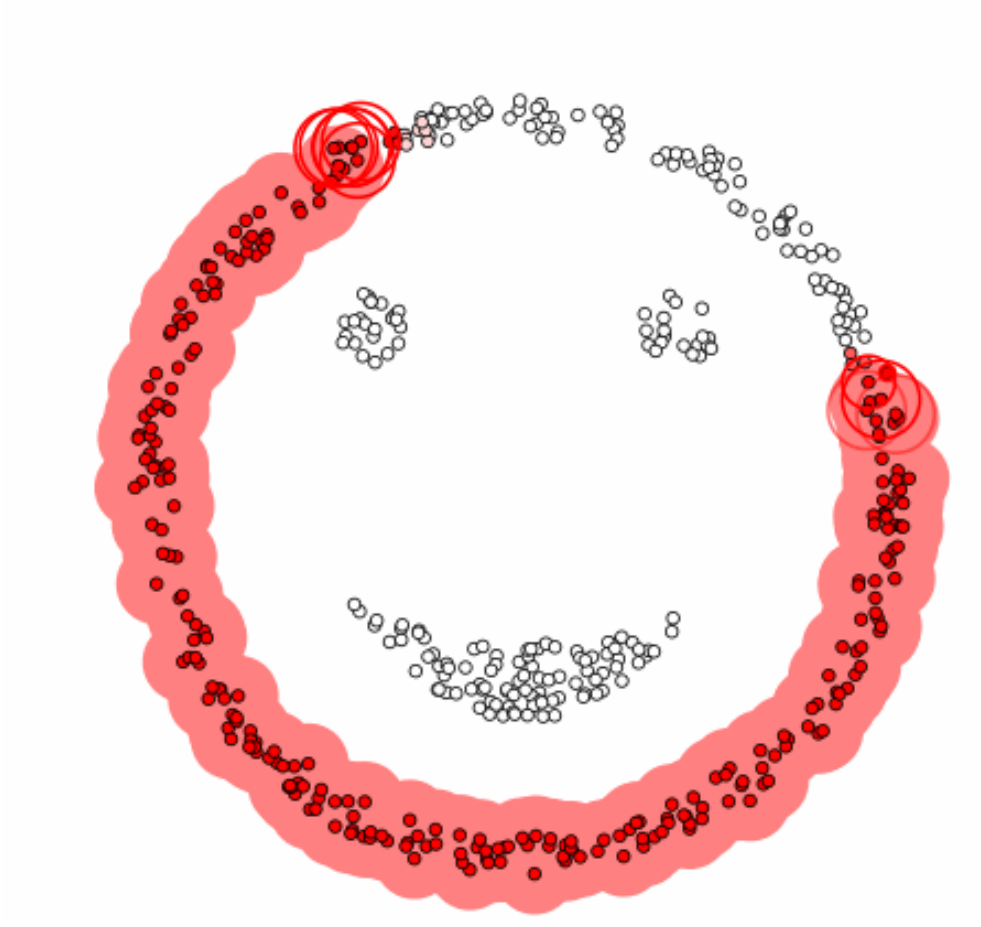
Border Vector(object) : 핵심 벡터로부터 거리 ϵ내에 위치해서 같은 군집으로 분류되나, 그 자체로는 핵심 벡터가 아닌, 군집의 외각에 위치하는 벡터

Noise Vector(object) : 핵심 또는 외각 벡터가 아닌, 즉 ϵ 이내에 minPts개 미만의 벡터가 있으며, 그 벡터들 모두 핵심 벡터가 아닐 경우, 어떠한 군집에도 속하지 않는 벡터



DBSCAN

DBSCAN 군집 과정



DBSCAN

복잡도

일반적인 DBSCAN은 각 객체와 객체 사이의 모든 거리가 필요
군집 과정에서도 하나하나 군집되기 때문에 시간 연산 복잡성은 보통 전체 데이터가 N개 일 때, O($\mathbf{N}\log N$) ~ O(N^2) 정도
그래프 기반 클러스터링 O(N^3)보다는 매우 낮이만 데이터가 방대하다면 무시 못할 수준

다른 군집 알고리즘

병합 군집

인접한 클러스터 쌍을 연결(처음에는 하나에서 시작)

각 샘플을 중심으로 하는 원을 그린 후, 원마다 안에 포함된 모든 샘플의 평균을 구함

평균 - 이동

BRICH

대규모 데이터 셋을 위해 고안

샘플 사이의 유사도 행렬을 받아 저차원 임배딩을 만듦 (차원 축소)

스펙트럼 군집

유사도 전파

샘플은 자신을 대표할 수 있는 비슷한 샘플에 투표하며, 알고리즘이 수렴하면 각 대표와 투표한 샘플이 클러스터를 형성