Clustering Analysis

Dr. Saerom Park
Statistical Learning and Computational Finance Lab.
Department of Industrial Engineering
psr6275@snu.ac.kr
http://slcf.snu.ac.kr

Table of Contents

1. Introduction

2. K-means clustering using scikit-learn

3. Hard vs. Soft clustering

4. Cluster Assessment

6. Hierarchical Clustering

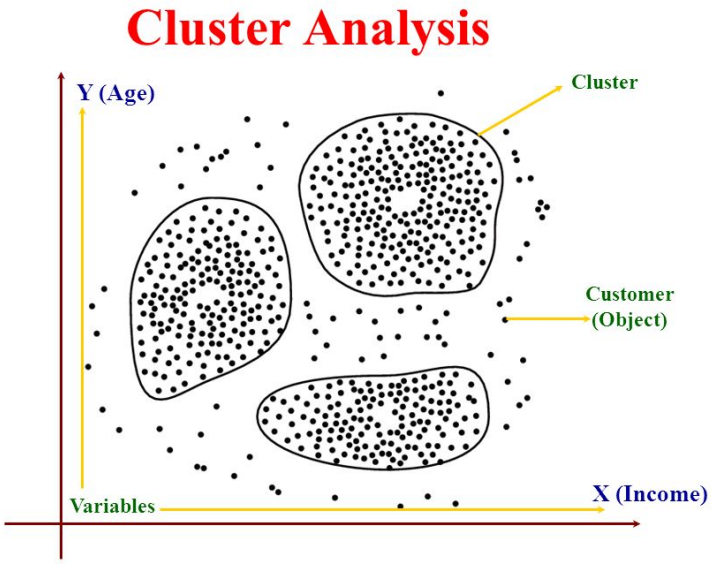
7. DBSCAN

Reference

- **Reading:** [Raschka. (2017), chapter 11], [Müller. (2016), chapter 3].

Introduction

- 군집화(Clustering)
  - 군집화란 데이터셋을 여러 군집(cluster)로 나누는 작업이다.
  - 한 군집 안의 데이터 포인트끼리는 매우 비슷하고, 다른 군집의 데이터 포인트와는 구분되도록 데이터를 나누는 것이 가장 큰 목표이다.
  - 분류 알고리즘과 비슷하게 군집화 알고리즘은 각 데이터 포인트가 어느 군집에 속하는지 할당 또는 예측하는 과정이다.



K-MEANS CLUSTERING

K-means clustering

K-means clustering

- 데이터 샘플 간의 거리를 측정하는 방법에는 여러가지가 있는데, 흔히 사용되는 것은 **squared Euclidean distance** 이다.
- m-차원 공간의 두 점 x, y 의 거리는,

$$d(x,y)^2 = \sum_{j=1}^{m}(x_j - y_j)^2 = \|x-y\|_2^2 \quad (j\text{는 } j\text{번째 feature를 의미})$$

- Squared Euclidean distance를 사용할 때, k-means clustering은 with-in cluster Sum of Squared Errors(SSE) 를 최소화하는 최적화 문제가 된다. (이를 cluster inertia라고도 함)

$$SSE = \sum_{i=1}^{n}\sum_{j=1}^{k} w^{(i,j)}\left\|x^{(i)} - \mu^{(j)}\right\|_2^2$$

이 때 $\mu^{(j)}$는 $cluster\ j$의 centroid,

샘플$x^{(i)}$가 $cluster\ j$에 포함 되면 $w^{(i,j)} = 1$ 아니면 $w^{(i,j)} = 0$

K-means clustering

- 다양한 거리 metric들(참고용)
  - 맨해튼 거리(시티 블록거리)

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

  - 최대좌표거리

$$d(x,y) = max_{\ j=1,2,\ldots m} |x_j - y_j|$$

  - 통계적 거리(마할라노비스 거리)

$$d(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}$$

이 때, S는 공분산행렬

K-means clustering using scikit-learn

Prototype-based clustering

- Prototype-based clustering 이란 각각의 클러스터가 클러스터 내부 점들의 centroid(평균)이나 medoid(관찰치)와 같은 프로토타입으로 표현되는 것을 말한다.
- Ex) K-means clustering

K-means clustering의 한계

- 정확한 해를 구하는 문제는 너무 어려운 문제 (NP-hard)
  - 알고리듬을 통해 근사해를 구함
- K-means clustering 알고리즘은 몇 개의 클러스터로 분석할지 사용자가 직접 k를 선택하여야 한다. 결정된 k에 따라 군집화 성능이 크게 달라질 수 있으므로, 신중하게 k를 선택하여야 한다.
- 초기 centroid들을 어디에 위치시키는지에 따라 모델의 성능이 달라질 수 있다.

K-means clustering using scikit-learn

- K-means clustering 과정

① 데이터에서 k개의 임의의 centroid를 뽑아 initial cluster center로 사용한다. (다른 방법도 가능)

② 각각의 데이터 샘플을 가장 가까운 centroid $\mu^{(j)}, j \in \{1, \ldots, k\}$ 에 할당한다.

③ Centroid를 추가된 샘플과의 중심으로 이동시킨다.

④ ②, ③ 과정을 더 이상 centroid가 움직이지 않거나 사용자가 지정한 tolerance나 maximum number of iteration까지 반복한다.

K-means clustering using scikit-learn

- Scikit-learn의 KMeans class (cluster 모듈)
  - sklearn.cluster.Kmeans(n_clusters, init, n_init, max_iter, tol, ….)

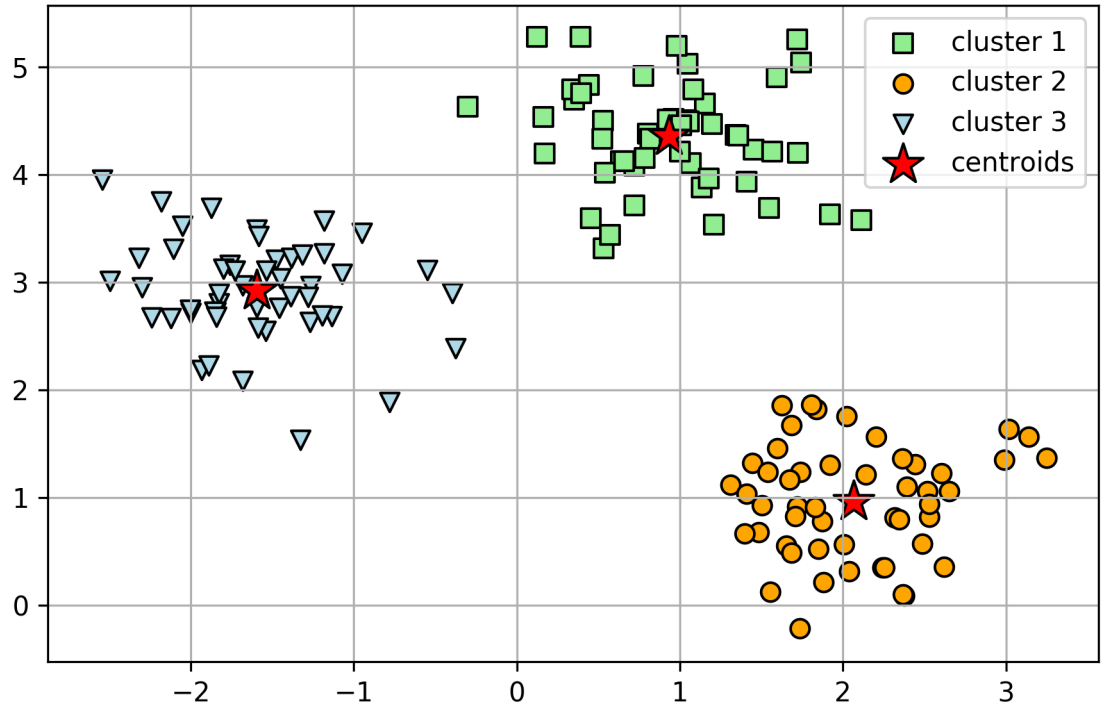
Parameters	
n_clusters	몇 개의 cluster를 만들지 지정
init	Centroid 초기화 {'k-means++', 'random' or an ndarray} 중 선택
n_init	초기 centroid 지정 횟수
max_iter	최대 iteration 횟수
tol	SSE값이 tol 이하로 떨어지면 중지

Attributes	
Cluster_centers_	Cluster centroid들의 좌표(array)
Labels_	각 점의 label
Inertia_	실제 SSE 값

  - http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

K-means clustering using scikit-learn

- Scikit-learn의 KMeans class (cluster 모듈)



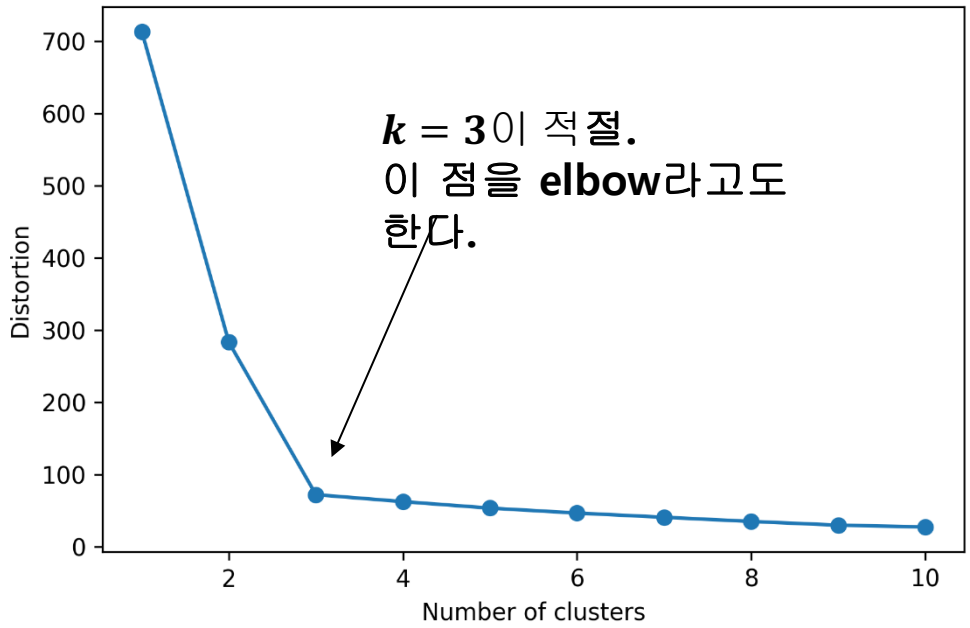
K means++을 활용한 초기 centroid 설정

- K-means ++
  - K-means++은 초기 centroid 설정을 좀 더 **합리적**으로 하기 위해 사용된다.
    - 다양한 초기 centroid 설정
  - 과정은 다음과 같다.
    ① 비어있는 집합 $\boldsymbol{M}$을 만든다. (여기에 선택된 k개의 centroid 저장)
    ② 임의로 첫번째 centroid $\mu^{(j)}$를 데이터 샘플에서 고르고, $\boldsymbol{M}$에 저장한다.
    ③ $\boldsymbol{M}$에 포함되지 않은 다른 샘플 $x^{(i)}$ 들에 대하여, $\boldsymbol{M}$에 포함되어 있는 centroid들과의 minimum squared distance $d(x^{(i)}, \boldsymbol{M})^{\,2}$ 을 구한다.
    ④ 다음 centroid $\mu^{(p)}$를 $x^{(i)}$들 중 선택하기 위해, 가중확률분포 $\frac{d(\mu^{(p)}, \boldsymbol{M})^{\,2}}{\sum_i d(x^{(i)}, \boldsymbol{M})^{\,2}}$ 를 사용한다.
    ⑤ 위의 과정을 k개의 centroid가 선택될 때까지 반복한다.
    ⑥ 기존의 k-means 알고리즘 진행

CLUSTER ASSESSMENT

Elbow method를 이용한 적절한 클러스터 수 찾기

- Elbow method
  - 얼마나 클러스터링이 잘 되었는지를 나타내주는 척도인 within-cluster SSE가 k가 변함에 따라 어떻게 달라지는지를 관측하여 적절한 k를 찾아본다.
  - Scikit-learn을 사용할 경우 inertia_에 자동적으로 within-cluster SSE가 계산되기 때문에 편리하게 사용할 수 있다.



Silhouette plot을 통한 클러스터링의 quality 측정

- Silhouette Analysis
  - Silhouette analysis는 클러스터링의 quality를 측정하기 위해 사용된다.
  - Silhouette coefficient을 계산한 뒤, 이를 플로팅하여 얼마나 클러스터링이 잘 되었는지 한 눈에 알아볼 수 있다.
  - Silhouette coefficient 계산하는 법:
    ① 한 클러스터 내의 샘플 $x^{(i)}$와 모든 다른 점들의 평균 거리 $a^{(i)}$를 구한다. (Cluster cohesion)
    ② 샘플 $x^{(i)}$와 가장 근접해 있는 다른 클러스터의 모든 점들과 $x^{(i)}$의 평균 거리 $b^{(i)}$를 구한다. (Cluster separation) (이 때, '가장 근접해 있는 클러스터'의 정의는 $x^{(i)}$와 클러스터의 모든 점들의 평균 거리가 최소인 클러스터)
    ③ Silhouette coefficient $s^{(i)}$는 다음과 같다.

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

Silhouette plot을 통한 클러스터링의 quality 측정

- Silhouette Analysis
  - scikit-learn에 내장된 silhouette_samples를 사용하면 편리하다.
  - sklearn.metrics.silhouette_samples(X, labels, metric,..)

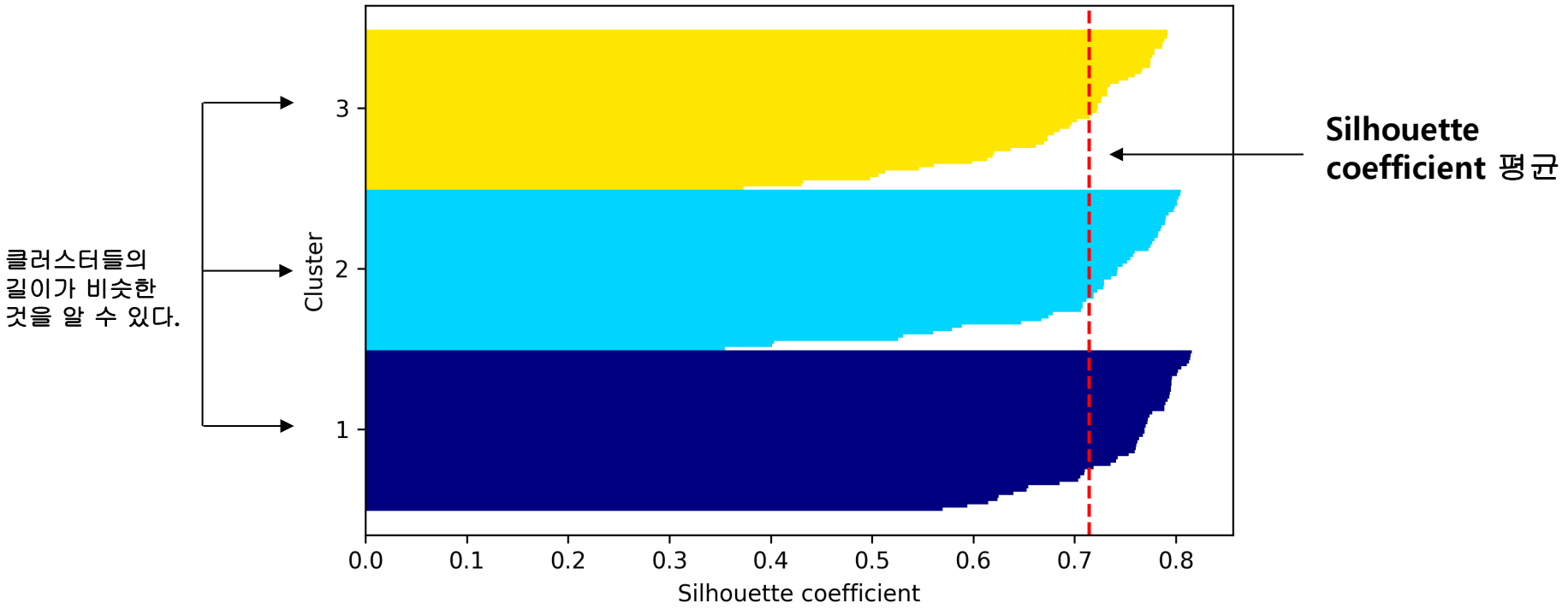
Parameters	
X	샘플 간 거리를 나타낸 array 혹은 sample 수 x features array
labels	각 샘플의 label 값
metric	'Euclidean', 'manhattan' 등

Returns	
silhouette	각 샘플의 silhouette coefficient를 array로 반환

  - http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_samples.html

Silhouette plot을 통한 클러스터링의 quality 측정

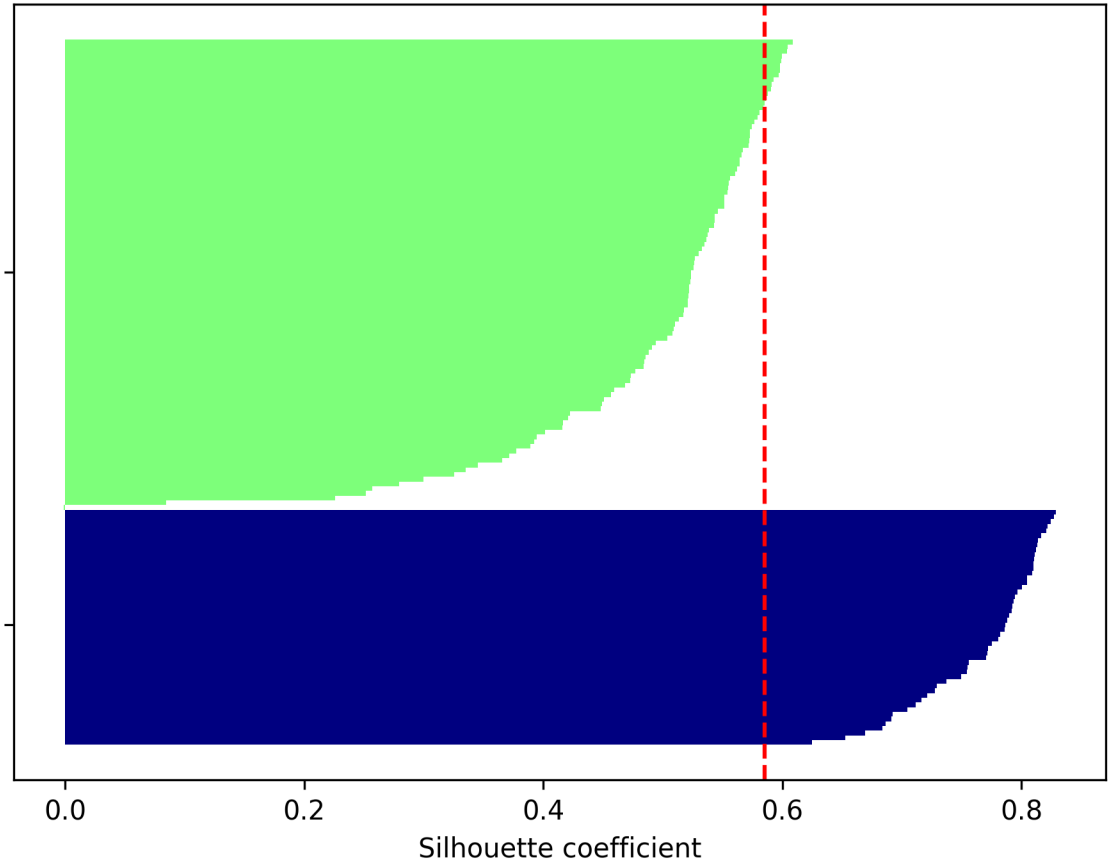
- Silhouette Analysis
  - 실제 데이터로 실루엣 분석 실시(잘 된 경우)



Silhouette plot을 통한 클러스터링의 quality 측정

- Silhouette Analysis
  - 실제 데이터로 실루엣 분석 실시(잘 안 된 경우)

클러스터들의 길이가 다르다는 것을 알 수 있다.



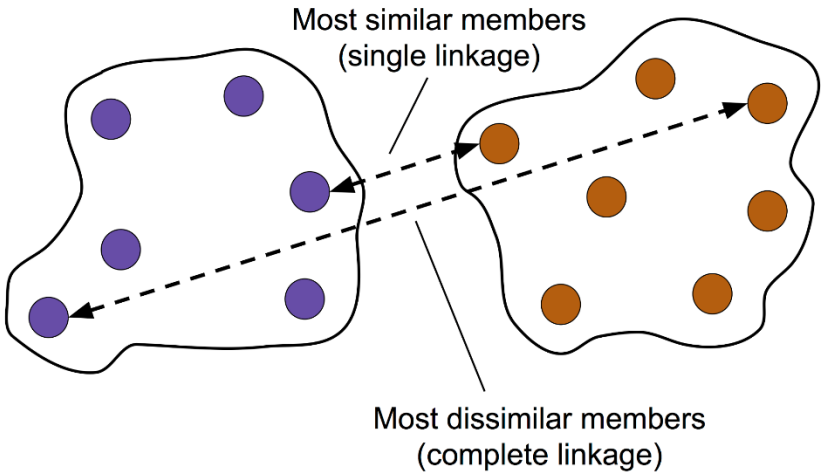
HIERARCHICAL CLUSTERING

Hierarchical Clustering(계층적 군집분석)

- Hierarchical Clustering?
  - 계층적 군집분석은 dendrogram을 그릴 수 있게 해준다는 것과, 클러스터의 수를 초기에 설정해주지 않아도 되는 매력적인 분석방법이다.
  - 하지만 $n \, x \, n$ 거리행렬을 계산해야 되기 때문에 계산 비용이 많이 들 수 있으며 데이터를 한 번만 통과시키기 때문에 분석 초기에 데이터가 잘못된 군집에 속하게 되면 수정이 불가능하다는 단점이 있다.
  - 두 가지 approach:
    ① Agglomerative(응집적): 각각의 샘플들이 하나의 cluster가 되고, 하나의 cluster가 될 때까지 가까운 pair끼리 합치는 방법
    ② Divisive(분할적): 모든 샘플들이 하나의 cluster로 시작하여 하나의 cluster에 하나의 샘플이 있을 때까지 지속적으로 분할하는 방법

Agglomerative Hierarchical Clustering

- Agglomerative Hierarchical Clustering의 두 가지 알고리즘
  - Single Linkage(단일연결법): 두 군집의 샘플 간의 최소 거리를 사용하여, 최소 거리가 가장 작은 군집끼리 합친다.
  - Complete Linkage(완전연결법): Singe Linkage와 비슷하나, 두 군집의 샘플 간의 최대 거리를 사용하여 군집끼리 합친다.



Agglomerative Hierarchical Clustering

- Ward Linkage(참고)
  - 와드 연결법은 군집간의 거리에 따라 데이터들을 연결하기 보다는 군집내 편차들의 제곱함에 근거를 두고 군집들을 병합시키는 방법이다.
  - 군집 G_1, G_2 가 있을 때, 각각의 크기가 n_1, n_2 라면

$$d(G_1, G_2) = \frac{\|\overline{x_1} - \overline{x_2}\|_2^2}{\frac{1}{n_1} + \frac{1}{n_2}}$$

이 때, $\overline{x_1}, \overline{x_2}$ 는 군집의 평균

  - 군집 평균 간의 거리에 가중합을 부여한다는 점에서 average linkage와 구분된다.

Agglomerative Hierarchical Clustering

- Hierarchical Complete Linkage Clustering
  - 과정을 정리해보면 다음과 같다.
    ① 모든 샘플 간의 거리를 계산하여 거리 행렬을 만든다.
    ② 각 데이터 샘플을 하나의 클러스터로 놓는다.
    ③ 최대 거리가 가장 짧은 두 클러스터를 하나로 합친다.
    ④ Similarity 행렬을 새로운 클러스터에 맞춰서 update한다.
    ⑤ ② ~ ④ 과정을 하나의 cluster가 남을 때까지 반복 시행한다.

Agglomerative Hierarchical Clustering

- Hierarchical Clustering with Python
  - Scikit-learn으로도 agglomerative clustering을 할 수 있다.
  - sklearn.cluster.AgglomeartiveClustering(n_clusters,affinity,..,linkage,..)
  - http://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

Parameters	
n_clusters	원하는 cluster의 개수
affinity	연결 시 사용할 거리 metric
linkage	연결 방법. Default는 ward method이다.

큰 장점!

Methods	
fit(X[, y])	Data에 hierarchical clustering fittting
fit predict(X[, y])	X 에 대한 Clustering을 실시하고 , label값을 return

DBSCAN

DBSCAN 알고리즘

- DBSCAN
  - Density-based Spatial Clustering of Applications with Noise(DBSCAN)은 k-means처럼 구형 cluster를 가정하거나, cluster 수를 정해주지 않아도 된다.
  - 이상치가 클러스터링 시 큰 문제가 되는 경우가 많은데, 이상치에 대한 해법으로 등장하였다. (Noise point)

    => 꼭 데이터 내의 모든 점이 cluster에 할당되지 않는다.
  - 클러스터의 밀도에 따라 클러스터를 서로 연결하기 때문에, 기하학적 모양을 갖는 군집도 잘 찾아낼 수 있다.

DBSCAN 알고리즘

DBSCAN

- DBSCAN 에서 각 점들은 3 가지 유형으로 분류된다:
  - 정해진 반지름 ε 내에 이웃하는 점이 MinPts(정해준다)개 이상이면, core point로 간주한다.
  - Core point의 반지름 ε 안에 있으나, 이웃하는 점의 개수가 MinPts 보다는 작다면 neighboring point로 간주한다.
  - Core/Neighboring point가 아닌 점들은 noise point로 간주한다.

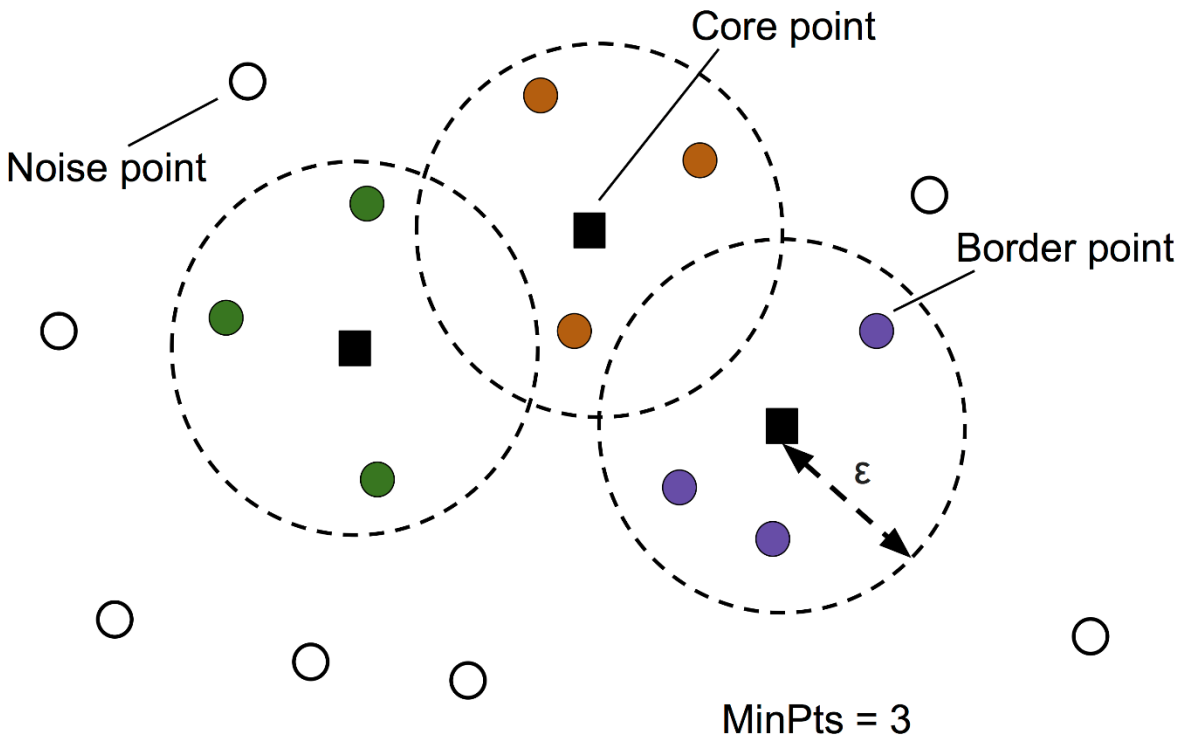
- 이렇게 각 점에 label을 부여한 뒤에는 아래의 과정을 거친다.

① Core point 별로 독립된 cluster를 형성하고, ε 안에 여러 개의 core point 가 존재하면 이를 연결하여 군집을 형성한다.

② 각 border point를 core point에 맞는 cluster로 할당한다.

DBSCAN 알고리즘

- DBSCAN



DBSCAN 알고리즘

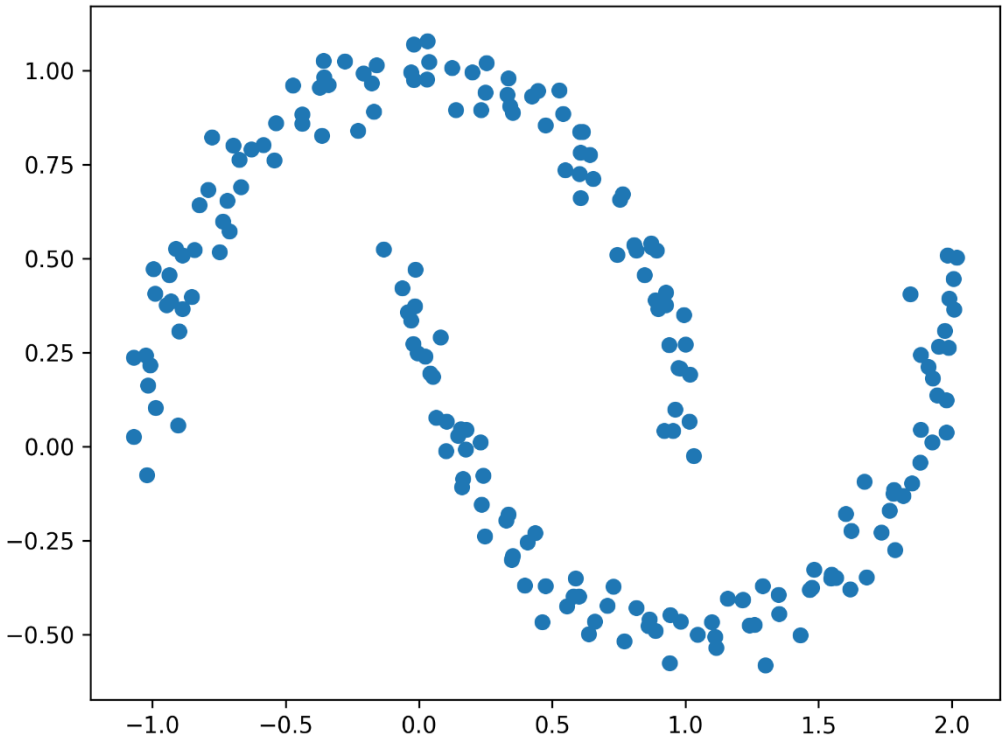
- DBSCAN
  - sklearn.cluster.DBSCAN(eps, min_samples, metric..)
  - http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

Parameters	
eps	Neighbor로 간주되는 두 점 사이의 최대 거리
min_samples	Core point를 찾기 위해 필요한 최소 sample 개수. (이 때 자기자신도 포함)
metric	사용할 거리 metric

Methods	
fit(X[, y, sample_weight])	DBSCAN 실행
fit_predict(X[, y, sample_weight])	X 에 대한 DBSCAN을 실시하고 , label값을 return

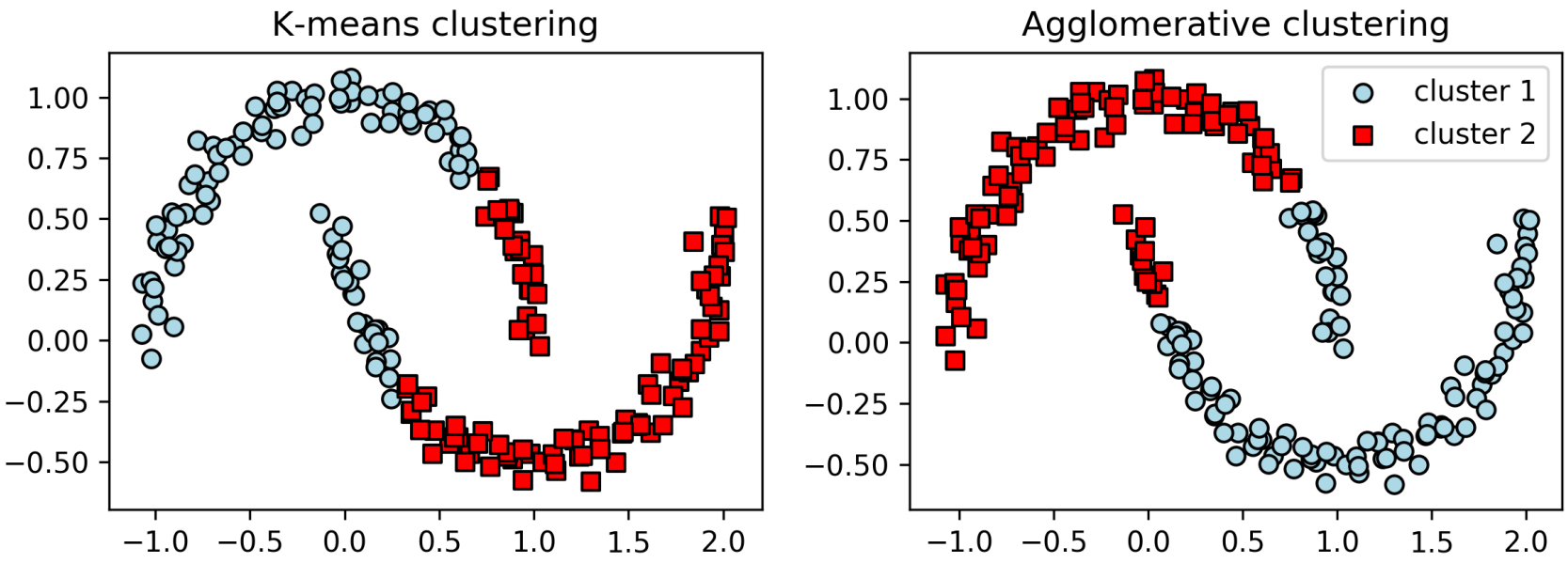
DBSCAN 알고리즘

- DBSCAN



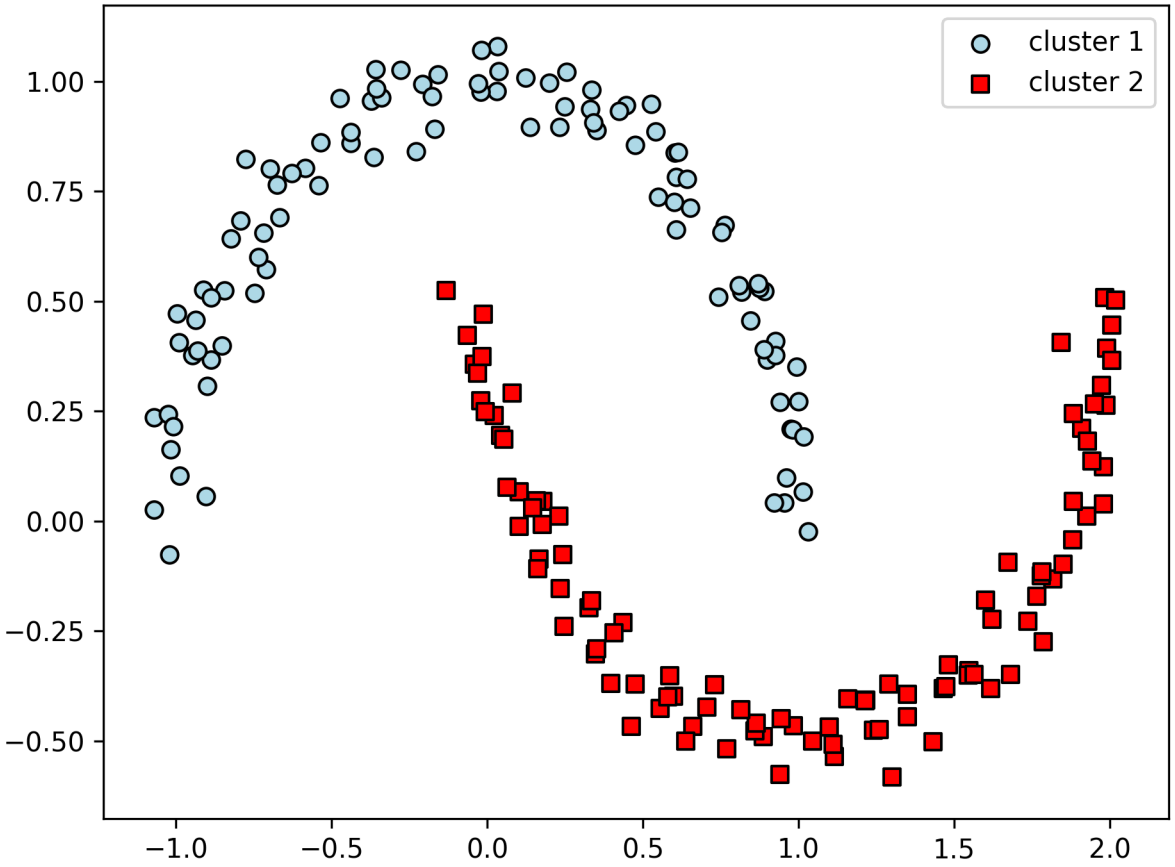
DBSCAN 알고리즘

- DBSCAN



DBSCAN 알고리즘

- DBSCAN
  - K-means나 Agglomerative보다 잘 clustering 한 것을 알 수 있다.



Reference

- [Raschka. (2017)] Raschka, Sebastian, and Vahid Mirjalili. *Python machine learning*. Packt Publishing Ltd, 2017.
- [Müller. (2016)] Müller, Andreas C., and Sarah Guido. *Introduction to machine learning with Python: a guide for data scientists*. 2016.
- [GÉRON. (2017)] GÉRON, Aurélien. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* Sebastopol, CA: O, 2017.