Clustering

Hyunjoong Kim

soy.lovit@gmail.com

github.com/lovit

Clustering

- 군집화는 데이터에서 비슷한 객체들을 하나의 그룹으로 묶습니다.
  - 각 객체들이 어떤 군집으로 할당되어야 하는지에 대한 정답 정보 (y) 가 없기 때문에 unsupervised 알고리즘으로 분류됩니다.
  - 군집화 방법들은 각 객체들의 유사도(거리) 정보를 이용합니다.
    유사한 객체를 하나의 군집으로 묶습니다.

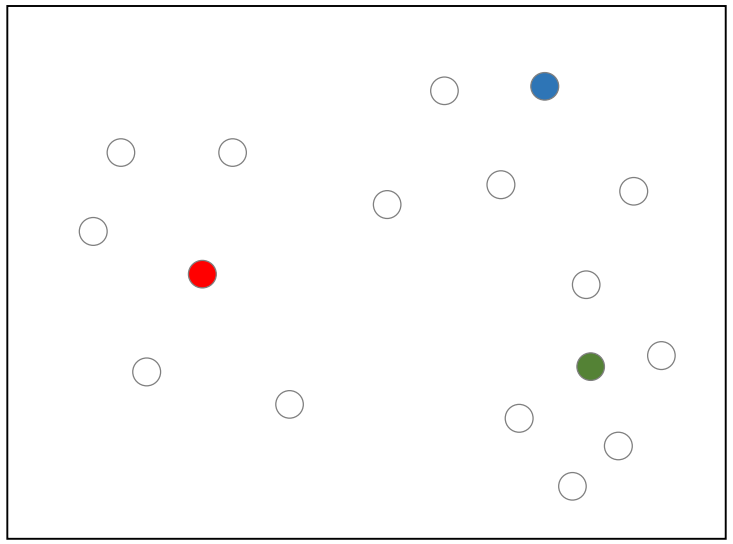
Clustering

- 좋은 군집에 대한 기준은 다양하지만 공통적으로 “군집 내 객체들은 비슷하며, 군집 간 객체들은 이질적”임을 추구합니다.
- 그 전에 근본적으로 데이터의 representation 과 dissimilarity measure 가 잘 정의되어야 합니다.

k-means clustering

- n개의 데이터 X에 대하여 두 데이터 x_i, x_j간에 정의되는 Cosine 이나 Euclidean 혹은 임의의 거리 $d(x_i, x_j)$ 를 이용할 수 있습니다.
- 모든 데이터는 k 개의 부분집합으로 나뉘어진다 가정하며, 각 부분집합은 대표값 벡터 (centroid) 를 지닙니다. 각 점은 가장 가까운 대표값 벡터로 재할당 (reassign) 하는 업데이트 과정을 반복합니다.

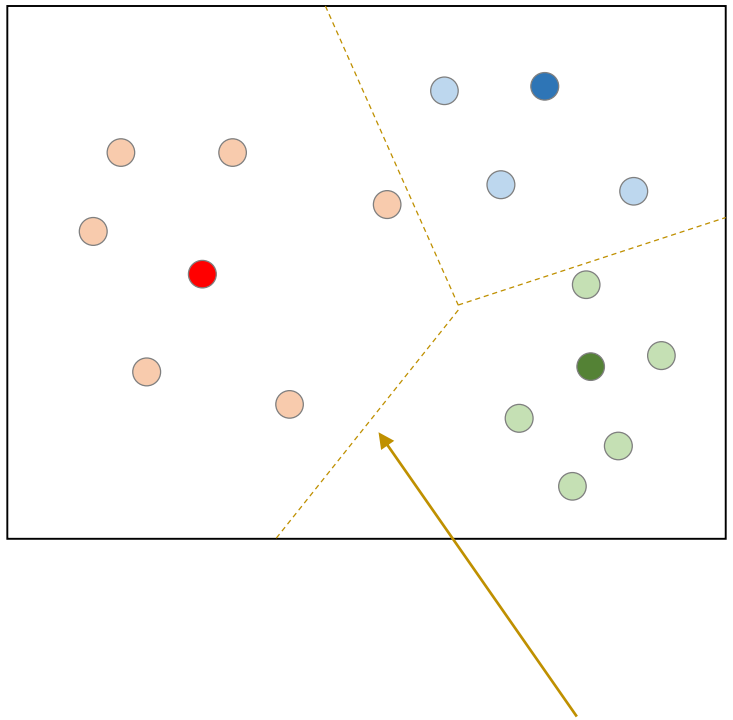
k-means clustering



1. Initialize

k=3 이라 가정하면 3개의 점을 임의로 선택

k-means clustering

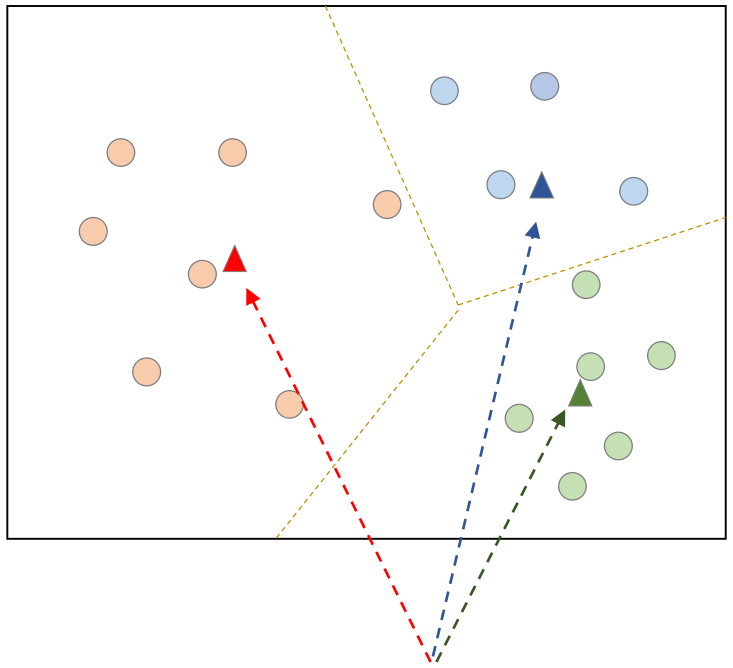


1. Initialize
k=3 이라 가정하면 3개의 점을 임의로 선택

2. Assign (epoch=0)
모든 점을 *k*개의 centroid 중 가장 가까운 점의 색깔(label)로 할당

*k*개의 centroids에 의하여 분할된 공간의 경계면으로, Voronoi partition, Voronoi diagram이라 부름

k-means clustering



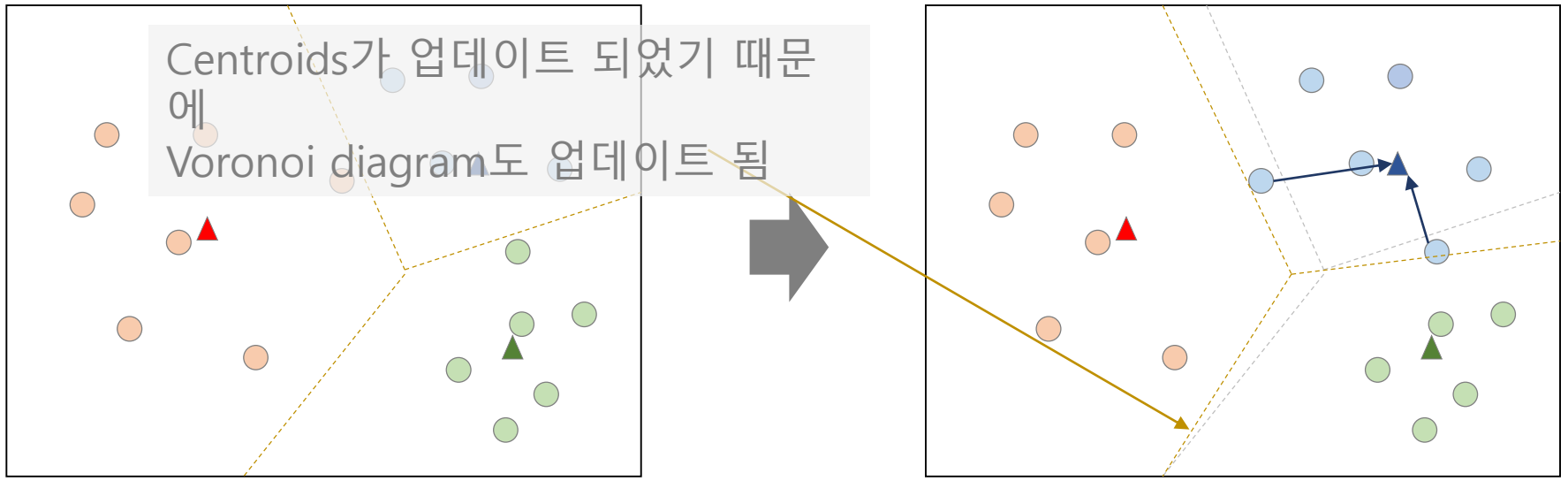
데이터에는 존재하지 않는 가상의 centroids

1. Initialize
k=3 이라 가정하면 3개의 점을 임의로 선택

2. Assign (epoch=0)
모든 점을 *k*개의 centroid 중 가장 가까운 점의 색깔(label)로 할당

3. Update centroid (epoch=0)
같은 색깔(label) 점들의 평균값을 가상의 centroids로 설정

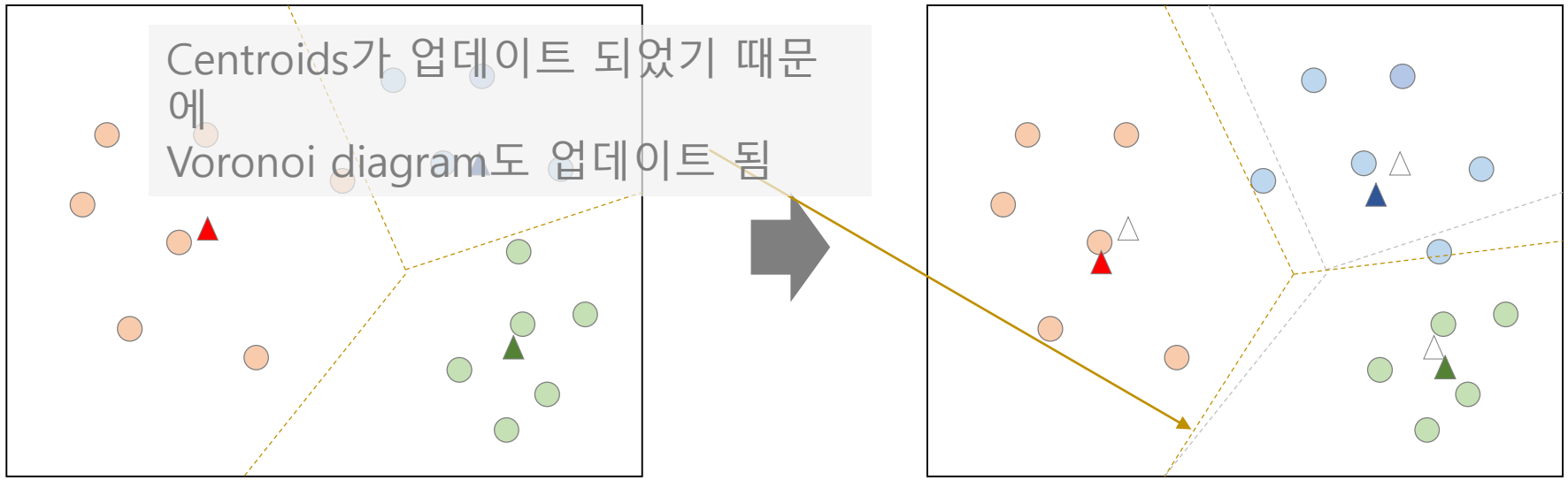
k-means clustering



1. Initialize
k=3 이라 가정하면 3개의 점을 임의로 선택

2. Assign (epoch=1)
모든 점을 업데이트 된 centroids 중 가장 가까운 점으로 할당

k-means clustering



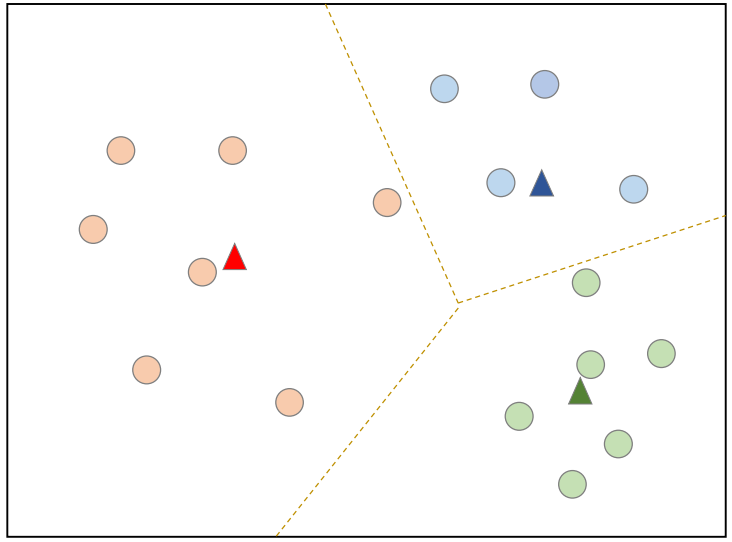
1. Initialize
k=3 이라 가정하면 3개의 점을 임의로 선택

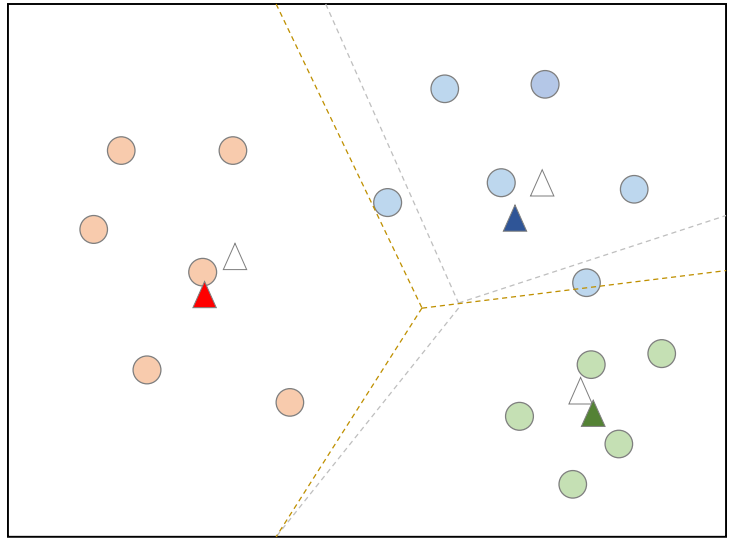
2. Assign (epoch=1)
모든 점을 업데이트 된 centroids 중 가장 가까운 점으로 할당

3. Update centroid (epoch=1)
색깔이 바뀐 점이 있기 때문에 Centroid를 다시 업데이트

알고리즘이 종료 될 때까지 2, 3을 반복

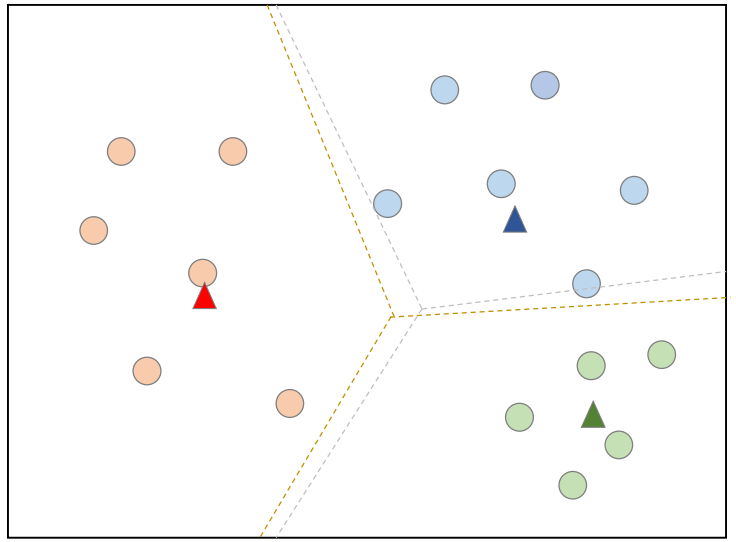
k-means clustering





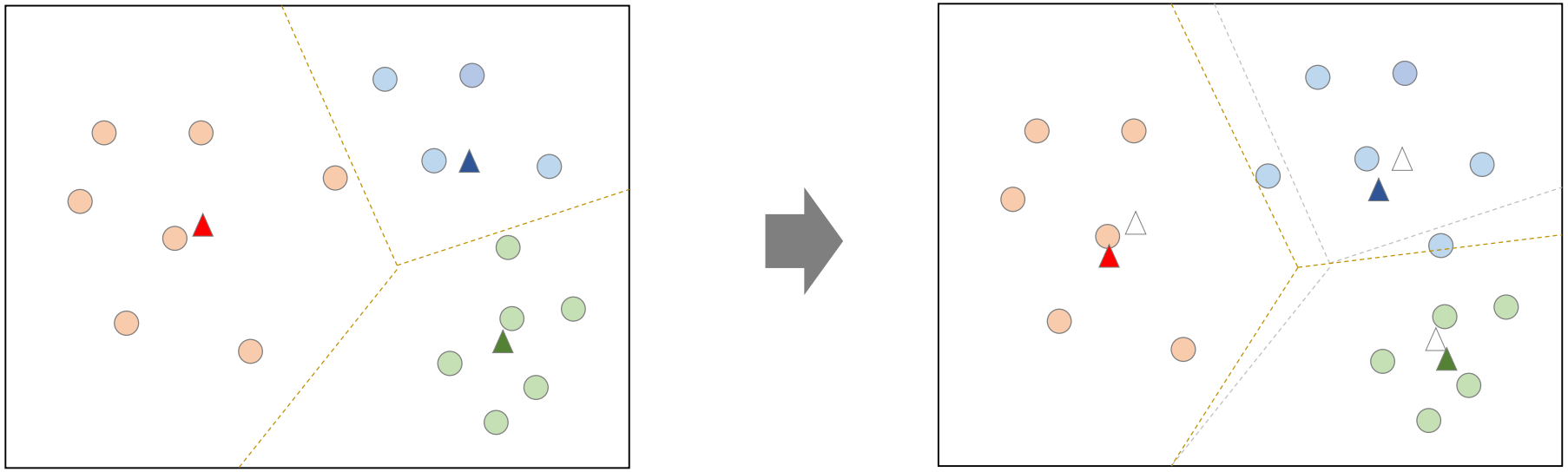
1. Initialize
k=3 이라 가정하면 3개의 점을 임의로 선택

2. Assign (epoch=2)
모든 점을 가장 가까운 centroids로 할당하여도 색깔이 변하지 않으므로 알고리즘 종료



Centroids based clustering

- 우리에게 익숙한 *k*-means 은 local optimal 을 찾는 heuristic 입니다 [1].
  - Lloyd *k*-means 는 수렴할 때까지 반복적으로 centroids를 업데이트 합니다.



[1] Lloyd, S. (1982). Least squares quantization in PCM. *IEEE transactions on information theory*, *28*(2), 129-137.

k-means

- (Lloyd) *k*-means 는 빠릅니다.
  - 계산복잡도가 작습니다. $O(n * i * k)$
  - pairwise distance 를 요구하지 않기 때문에 대량의 데이터에 적합하며,
  - row 단위로 학습하기 때문에 mini-batch / 분산환경의 구현이 쉽습니다.

[1] David Sculley. Web-scale k-means clustering. In Proceedings of the 19th international conference on World wide web, pages 1177–1178. ACM, 2010

k-means

- Representation 과 거리 척도를 잘 설정해야 합니다.
  - 일반적으로 *k*-means 는 Euclidean distance 를 이용합니다.
  - 추천 문제에 이용되는 고차원 sparse vectors 간의 유사도에서는 공통으로 등장한 아이템의 유무가 가장 중요하지만, Euclidean 은 이를 고려하지 않습니다.
  - Sparse high dimensional vector 에는 Jaccard, Pearson, Cosine 을 쓸 수 있지만, Euclidean 만은 쓰지 말아야 합니다 [1].
  - Jaccard, Pearson, Cosine 모두 벡터의 방향성에 관련된 척도입니다.

[1] Anna Huang. Similarity measures for text document clustering. In Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand, pages 49–56, 2008

Magnitude of vector

참고

- p-norm 은 벡터의 크기를 정의하는 방식입니다.
  - $|X|_p = \sqrt[p]{|X_1|^P + \ ... + \left|X_q\right|^P}$ 로 정의됩니다.
  - 2 norm : $|(3,0,4)|_{p=2} = \sqrt[2]{|3|^2 + 0^2 + |4|^2} = 5$
  - 1 norm : $|(3,0,-4)|_{p=1} = \sqrt[1]{|3|^1 + |0|^1 + |-4|^1} = 7$
  - 0 norm : $|(3,0,4)|_{p=0} = |3|^0 + \ |4|^0 = 2$

Inner product

참고

- 내적 (inner product) 은 두 벡터 간의 관계를 정의하는 방식입니다.
  - 같은 위치의 값들끼리 서로 곱하여 그 값을 더합니다.
  - $(3, 0, 4)^T(1, 2, 3) = 3 \times 1 + 0 \times 2 + 4 \times 3 = 3 + 0 + 12 = 15$
  - 둘 중 하나라도 0 일 경우 그 위치는 무시됩니다.

Distance measure: Euclidean

참고

- 두 벡터 간의 거리를 두 벡터의 차이에 대한 L2 norm 으로 정의합니다.
  - $euc\big((3,0,4),(1,2,3)\big) = \big((3-1)^2 + (0-2)^2 + (4-3)^2\big)^{\frac{1}{2}} = (4+4+1)^{\frac{1}{2}} = 3$

Distance measure: Cosine

참고

- 두 벡터 간의 거리를 유닛 벡터 간의 내적으로 정의합니다.
  - $cos(u, v) = \frac{u \cdot v}{|u|_2 \cdot |v|_2} = \left(\frac{u}{|u|_2}\right) \cdot \left(\frac{v}{|v|_2}\right)$
  - $cos\big((3, 0, 4), (4, 3, 0)\big) = \frac{12}{5 \times 5} = 0.48$

Cosine & Euclidean distance

참고

- 유닛 벡터의 Euclidean 거리의 순서는 Cosine 거리의 순서와 같습니다.
  - $|u - v|_2^2$

    $= (u - v)^T (u - v)$

    $= |u|_2^2 - 2 \cdot u \cdot v + |v|_2^2$

    $= 2(1 - u \cdot v)$

    $= 2\,(1 - cos(u, v))$

Spherical *k*-means

- Cosine distance 를 이용하는 *k*-means 를 Spherical *k*-means 라 합니다 [1].
  - Distance measure 외의 학습 방법은 Lloyd 와 같습니다만,
  - 이 차이로 결과는 확연히 다릅니다.

- scikit-learn 에는 metric 이 Euclidean 으로 고정되어 있습니다.

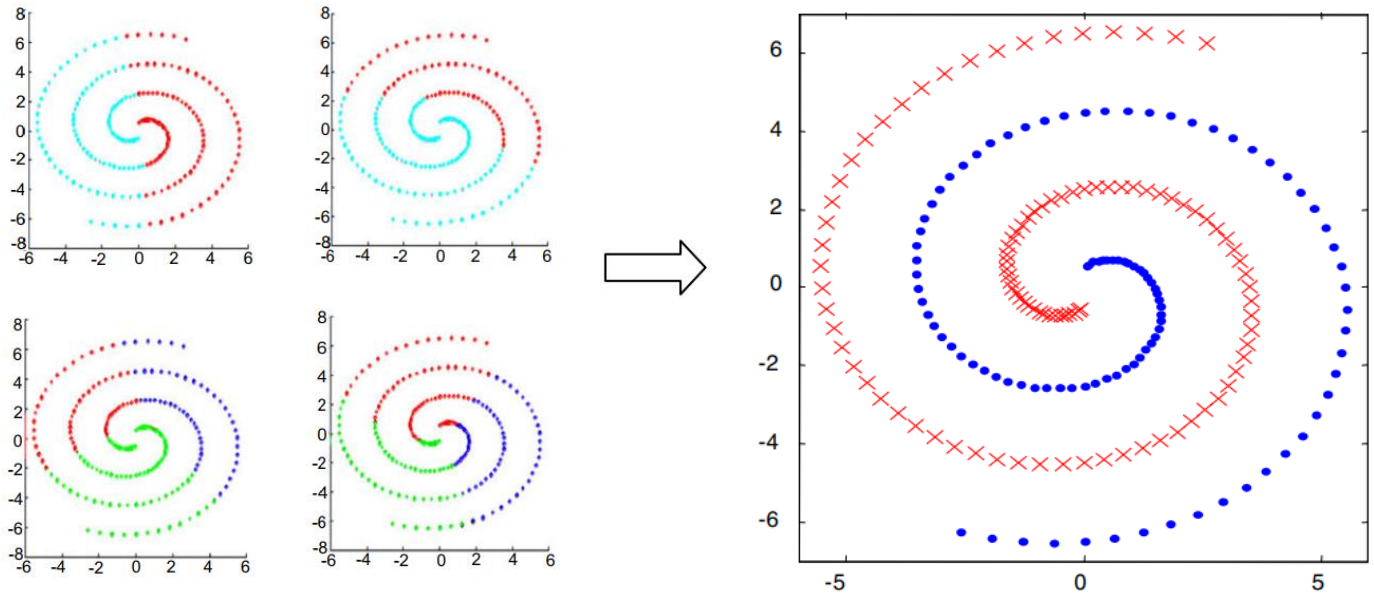
[1] Inderjit S Dhillon and Dharmendra S Modha. Concept decompositions for large sparse text data 120 using clustering. Machine learning, 42(1):143–175, 2001.

k-means

- *k*-means 계열 알고리즘들은 알려진 단점들이 있습니다.
  1. 군집의 모양은 centroid 를 중심으로 한 구형을 가정합니다 (Voronoi diagram)
  2. Initial points 에 따라 군집의 모양이 달라질 수 있습니다.
  3. 적절한 군집의 개수는 사용자가 정의해야 합니다.
  4. 노이즈 데이터에 민감할 수 있습니다.

Limitations 1. Ball-shape

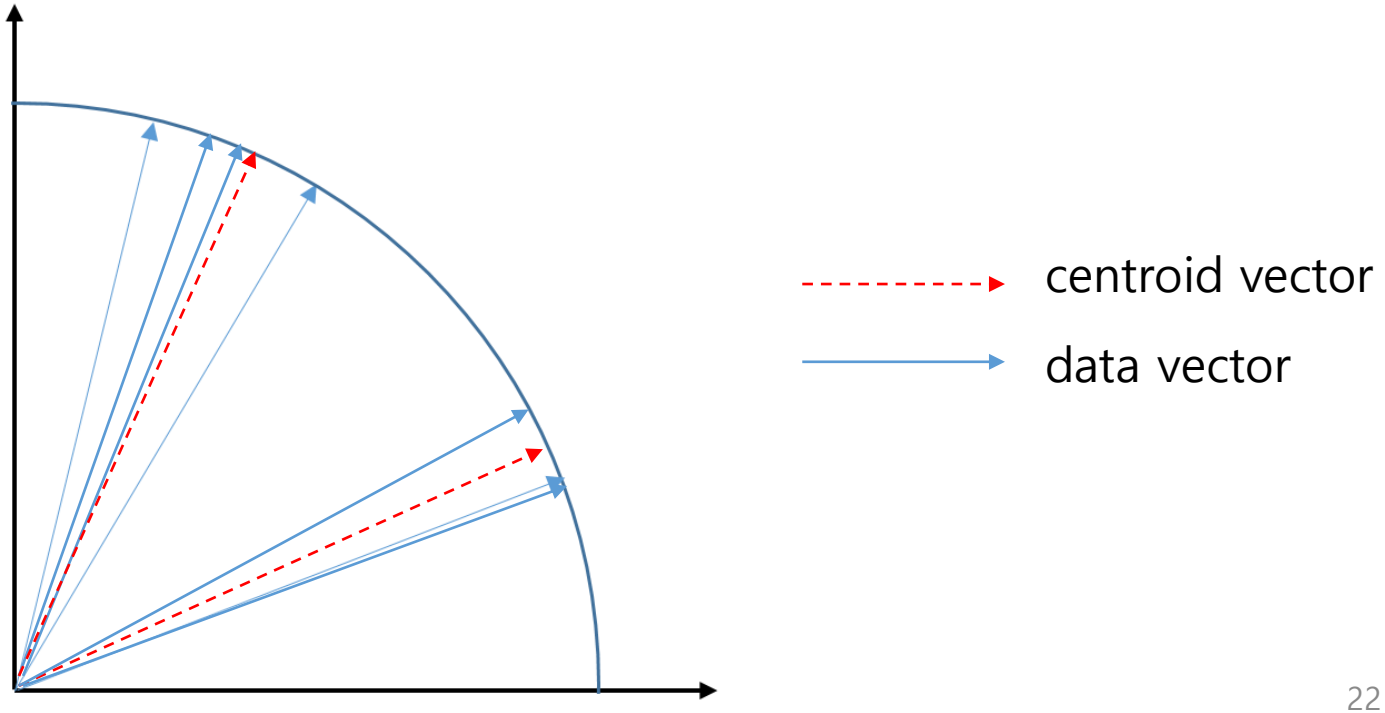
- *k*-means 는 Voronoi diagram 을 만듭니다.
  - 군집의 모습이 구형이 아닌 경우에 취약합니다.
  - *k*-means ensembles 은 이를 해결할 수 있습니다 [1]



[1] Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern recognition letters*, *31*(8), 651-666.

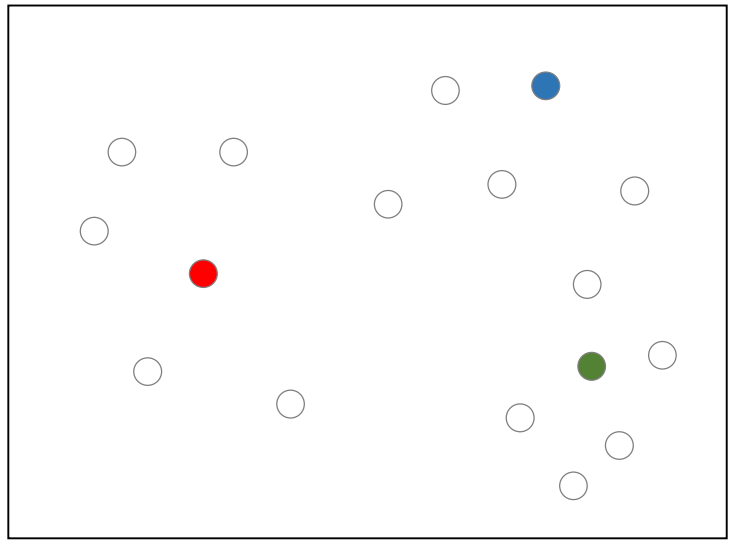
Solution 1. Ball-shape

- High dimensional sparse vector + Cosine distance 를 이용하면,
  각도 기준으로의 partitioning 입니다.

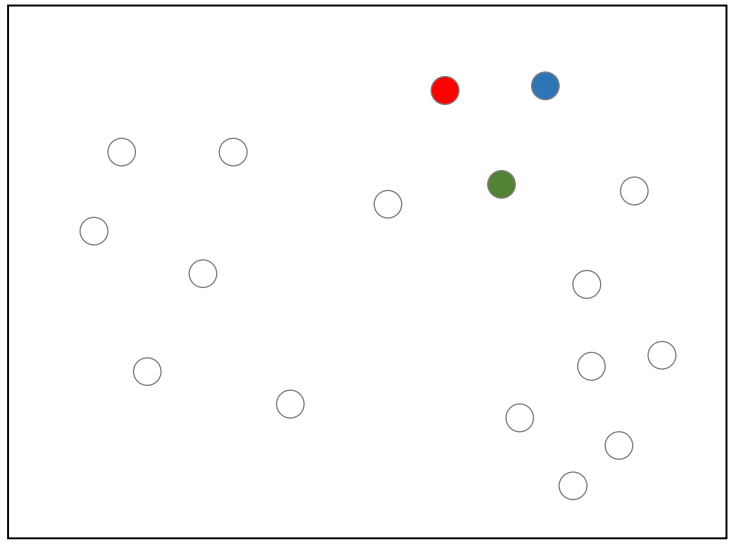


Limitations 2. Unstable initial points

- *k*-means 는 initial points 만 잘 뽑아도 수렴이 빠르다고 알려져 있으며,
  - 사실, 최악의 initialization 만 아니면 수렴은 원래 빠릅니다.
  - 특정 지역에 initial points 가 모여있지만 않아도 됩니다.



Ideal initialization



Worst initialization

Solution 2. Unstable initial points

- *k*-means++ [1] 은 가장 널리 알려진 initialization method 입니다.
  - 지금까지 선택된 점들과의 최소 거리를 다음 점의 확률로 이용합니다.

1. Select a point c_0 randomly
2. Select next point c_t with prob. $\frac{\mathrm{d}(x)^2}{\sum_{x \in D} \mathrm{d}(x)^2}$

   $\mathrm{d}(x)^2$: minimal distance between all points and already chosen points
3. Repeat step 2 until choosing k points

[1] Arthur, D., & Vassilvitskii, S. (2007, January). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1027-1035). Society for Industrial and Applied Mathematics.

Solution 2. Unstable initial points

- *k*-means++ [1] 은 가장 널리 알려진 initialization method 입니다.
  - scikit-learn 에 구현되어 있습니다.

sklearn.cluster.KMeans¶

class sklearn.cluster. **KMeans** (*n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=1, algorithm='auto'*) [source]

Limitations 3. Defining k

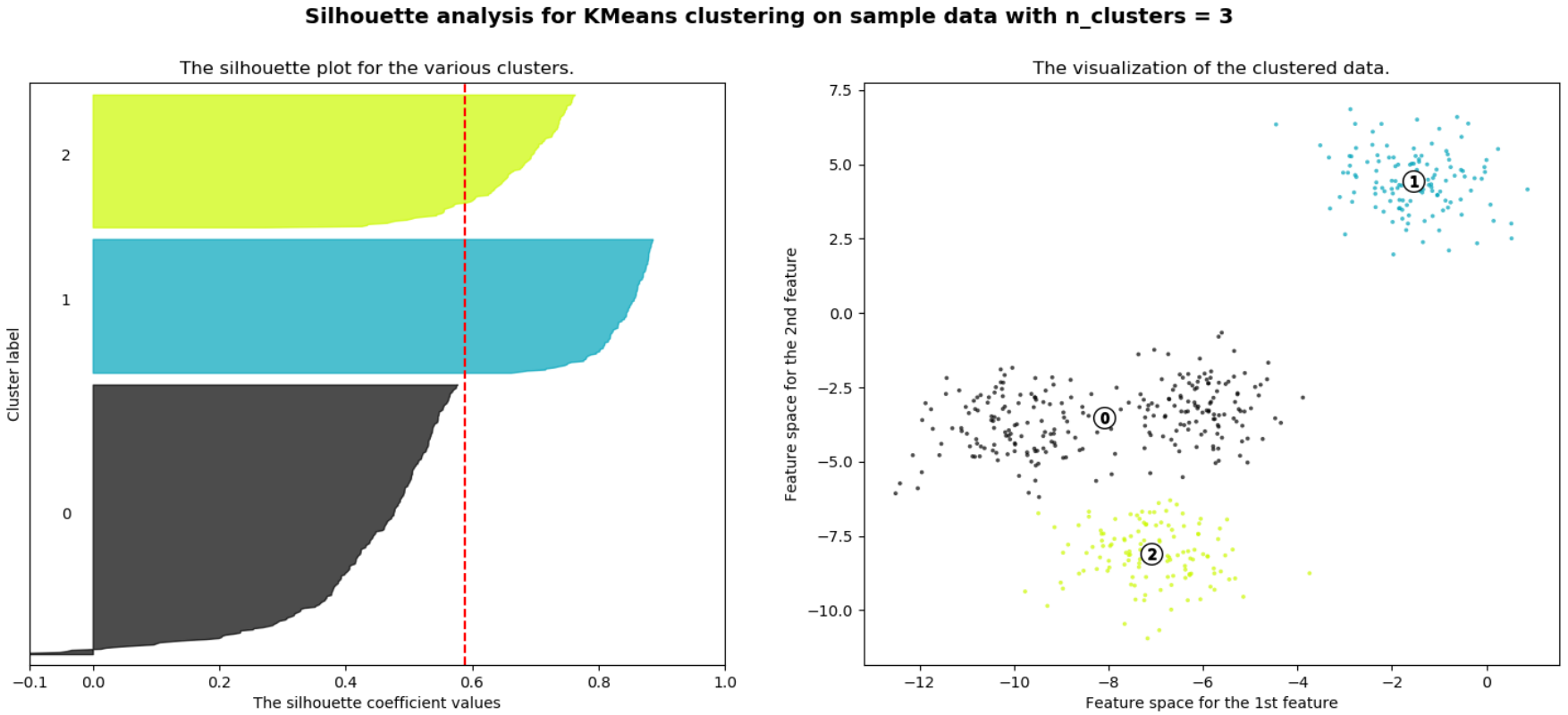
- k-means 는 군집의 개수를 사용자가 직접 정의하여야 합니다.
- Silhouette 은 군집화 품질의 척도로, 적절한 k 를 정하는데 이용됩니다.

$$s(x) = \frac{b - a}{\max(a, b)}, \qquad s(X) = mean\big(s(x)\big)$$

a: mean distance between a sample and all other points in the same class

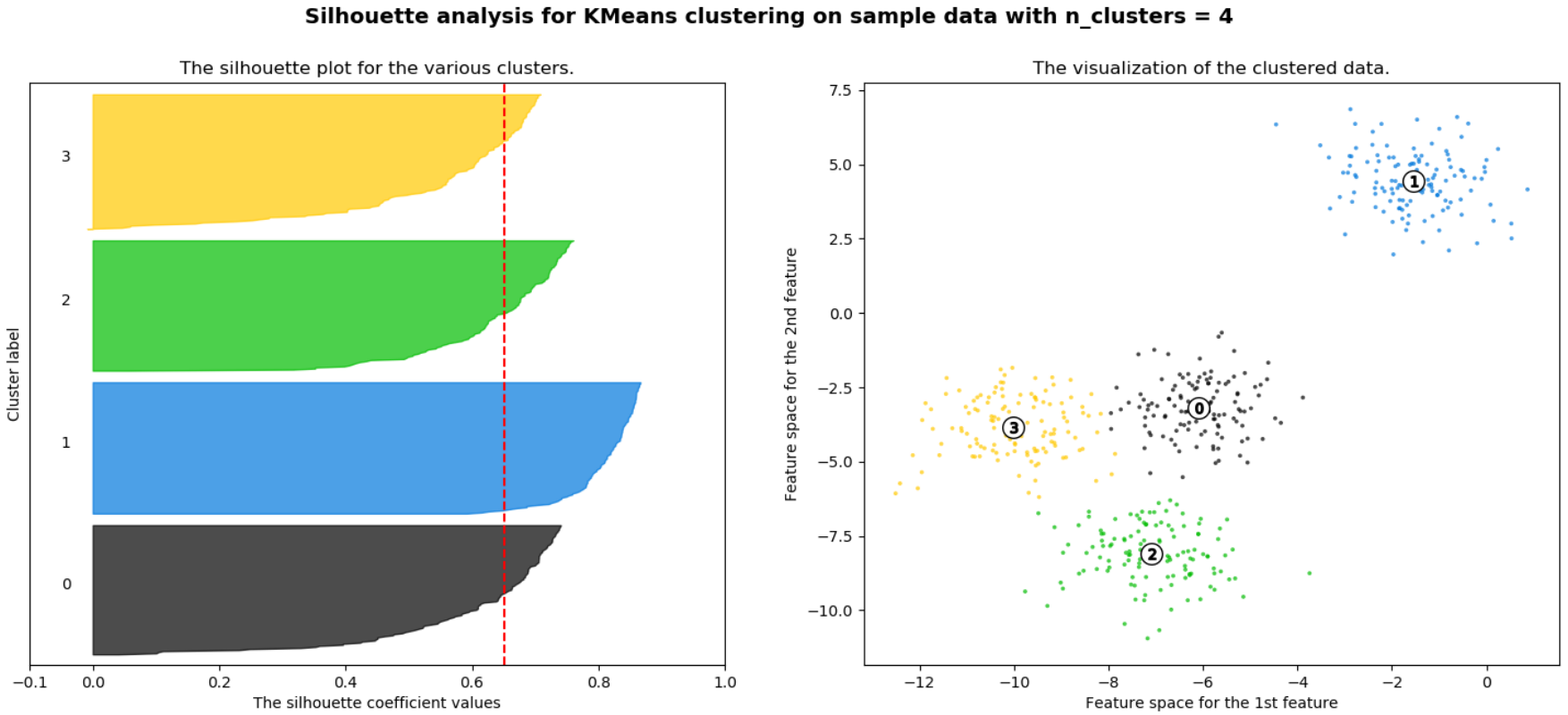
b: mean distance between a sample and all other points in the next nearest class

Limitations 3. Defining *k*



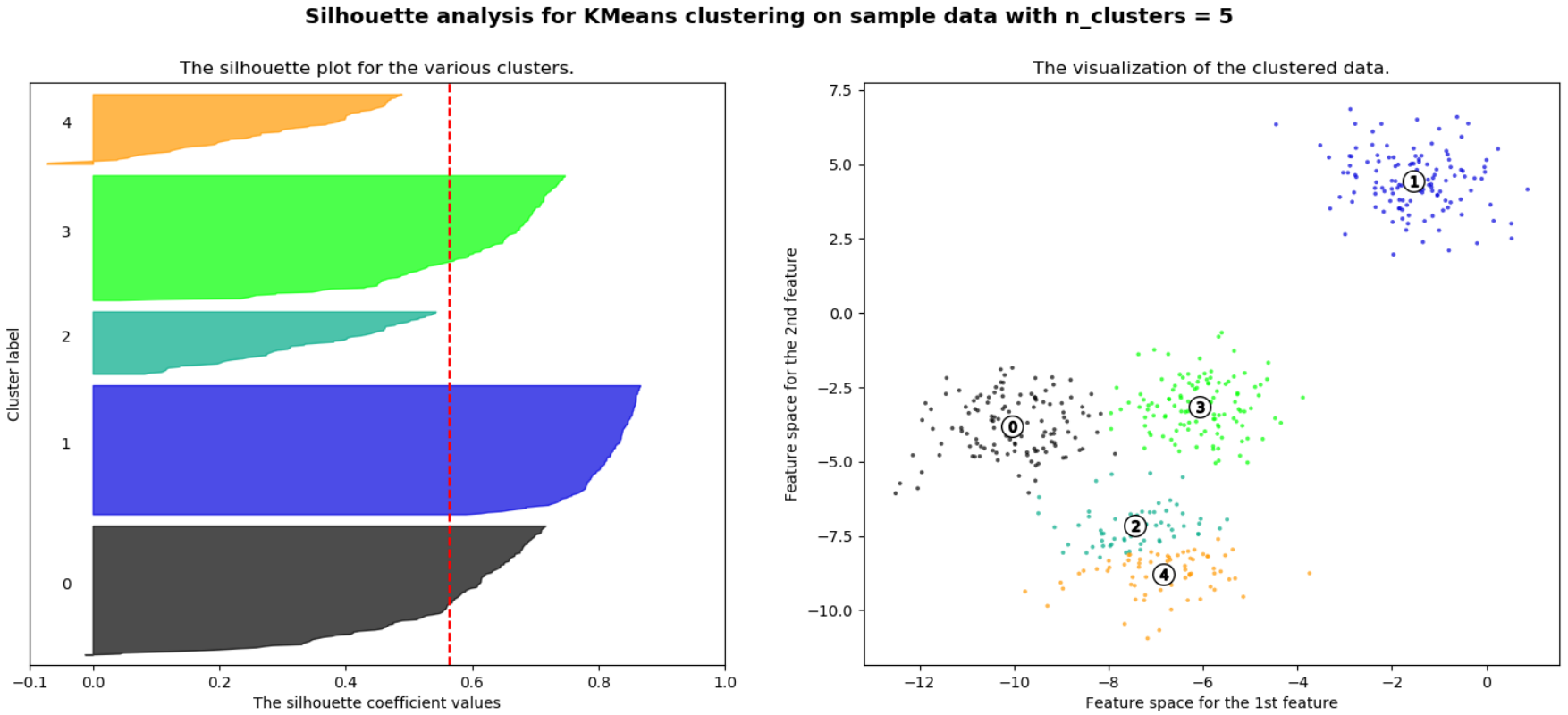
[1] http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Limitations 3. Defining *k*



[1] http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Limitations 3. Defining *k*



[1] http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Limitations 3. Defining *k*

- Silhouette 은 사후 평가 방법입니다.
  - 적절한 k 를 찾았더라도, 그 군집이 최선이라고는 말하지 못합니다.
  - 모든 k 에 대하여 테스트 할 비용도 만만치 않습니다.

Limitations 3. Defining *k*

- Silhouette 은 model fitness measure 입니다.
  - 높은 값이 "우리가 예상하는" 좋은 군집화 결과를 의미하지는 않습니다.
  - 수리적으로는 설명력이 있지만, 현실적이지 않은 measure 라는 주장도 있습니다 [1].

[1] David Newman, Youn Noh, Edmund Talley, Sarvnaz Karimi, and Timothy Baldwin. Evaluating topic 220 models for digital libraries. In Proceedings of the 10th annual joint conference on Digital libraries, pages 215–224. ACM, 2010.

Limitations 3. Defining k

- 고차원 데이터는 잘 작동하지 않을 수 있습니다.
  - 각 군집의 크기가 크다면, 멀리 떨어진 (=거리가 무의미한) 점들이 존재합니다.

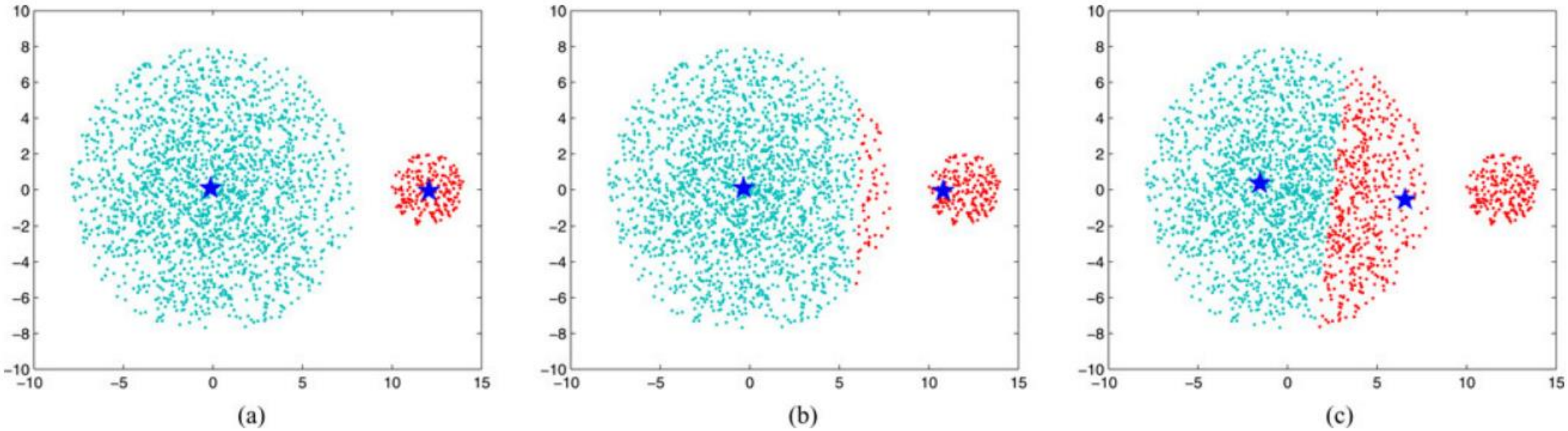
$$s(x) = \frac{b - a}{\max(a, b)}, \qquad s(X) = mean\big(s(x)\big)$$

a: mean distance between a sample and all other points in the same class

b: mean distance between a sample and all other points in the next nearest class

Limitations 3. Uniform effect

- Imbalanced class data 일 때, class distribution 이 잘 반영되지 않고, 모든 군집의 크기가 균일해지는 현상 [1]



[1] Hui Xiong, Junjie Wu, and Jian Chen. K-means clustering versus validation measures: a data distribution perspective. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(2):318–331, 2009.

Limitations 3. Uniform effect

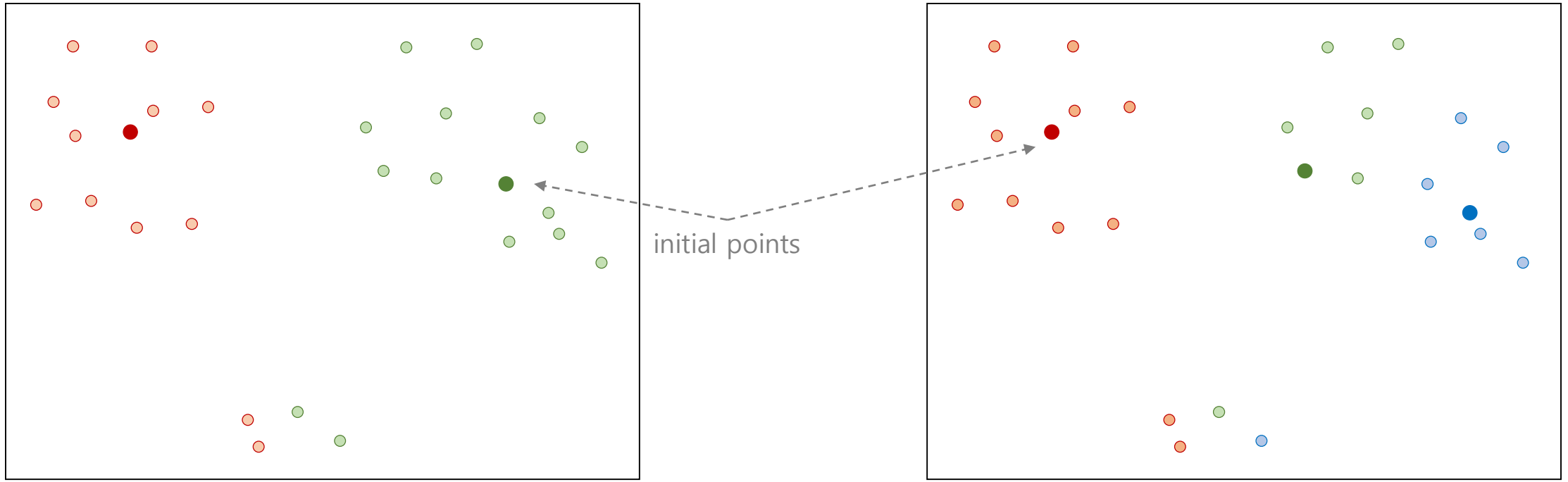
- *k*-means type 은 몇 번의 반복으로 거의 수렴합니다.
  - 그렇다면 “repeat until converged” 동안의 학습은 더 좋은 걸까요?
  - Uniform effect 가 일어나는 과정일 수도 있습니다.

Limitations 3. Defining k

- 현실적인 미봉책은 예상하는 군집의 개수보다 크게 k 설정한 뒤, 후처리로 비슷한 군집을 병합합니다.
  - 학습 후, 하나의 군집이 여러개로 나뉘어 졌는지를 확인하기는 쉽습니다.
  - 하나의 군집에서 잘못된 점을 찾는 것이 더 어려우며, 그 점들의 후처리도 어렵습니다.

Solutions 3. Defining *k*

- 작은 군집은 독립된 군집으로 선택되기가 어렵습니다.
  - Initial points 가 선택되지 않으면 작은 군집은 큰 군집으로 나눠집니다.
  - 한 번 나눠진 작은 군집은 독립된 군집이 될 가능성이 거의 없습니다.

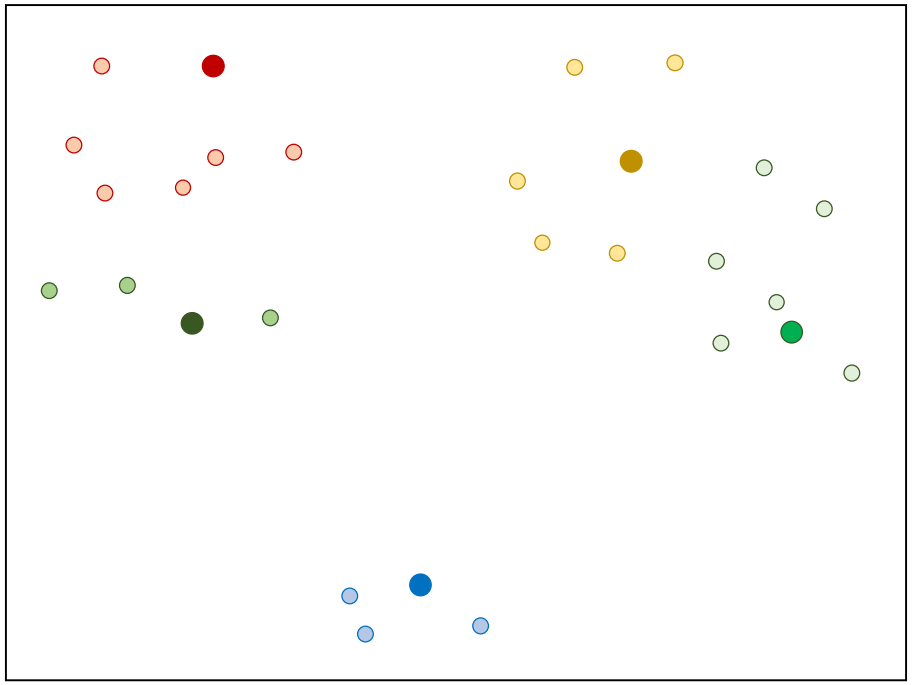


Solutions 3. Defining k

- 현실적인 미봉책은 예상하는 군집의 개수보다 크게 k 설정한 뒤, 후처리로 비슷한 군집을 병합합니다.
  - 실제 군집의 개수보다 k 가 크면 major 군집들이 여러 개로 나뉘어집니다.
  - k 가 작으면 minor 군집이 찢어질 가능성이 높습니다.
    - 이 때 centroids 는 major 편입니다.

Solutions 3. Defining *k*

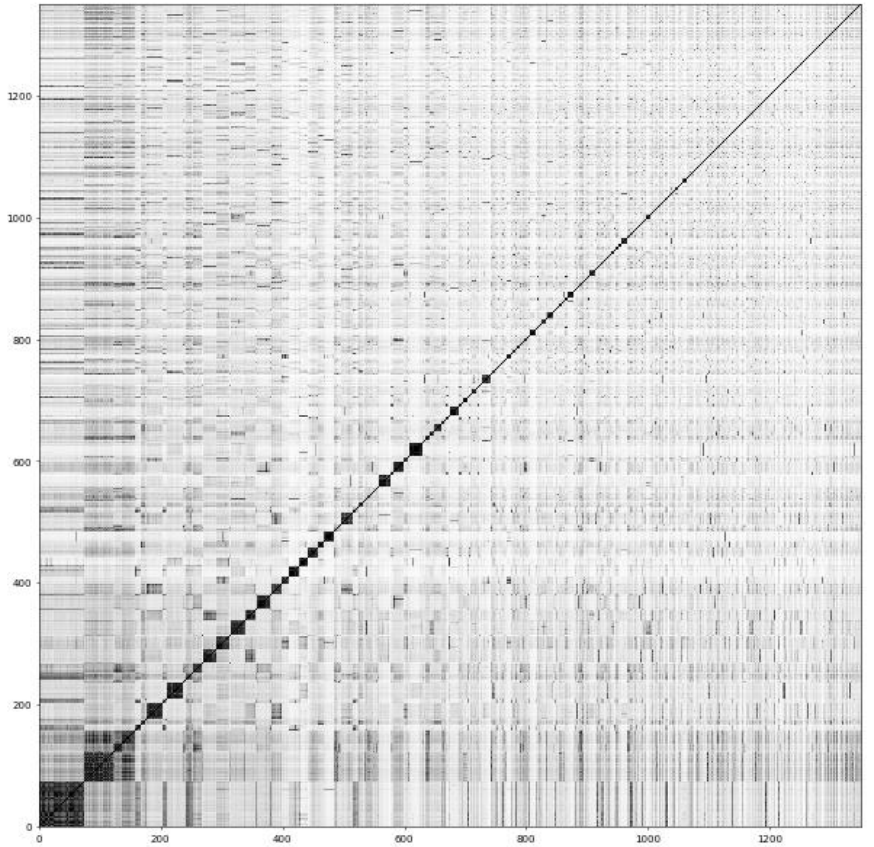
- 큰 군집이 여러 개의 중복된 군집으로 나눠지더라도 *k* 를 크게 설정하여 작은 군집들이 독립적으로 존재하도록 해야 합니다.



Solutions 3. Defining *k*

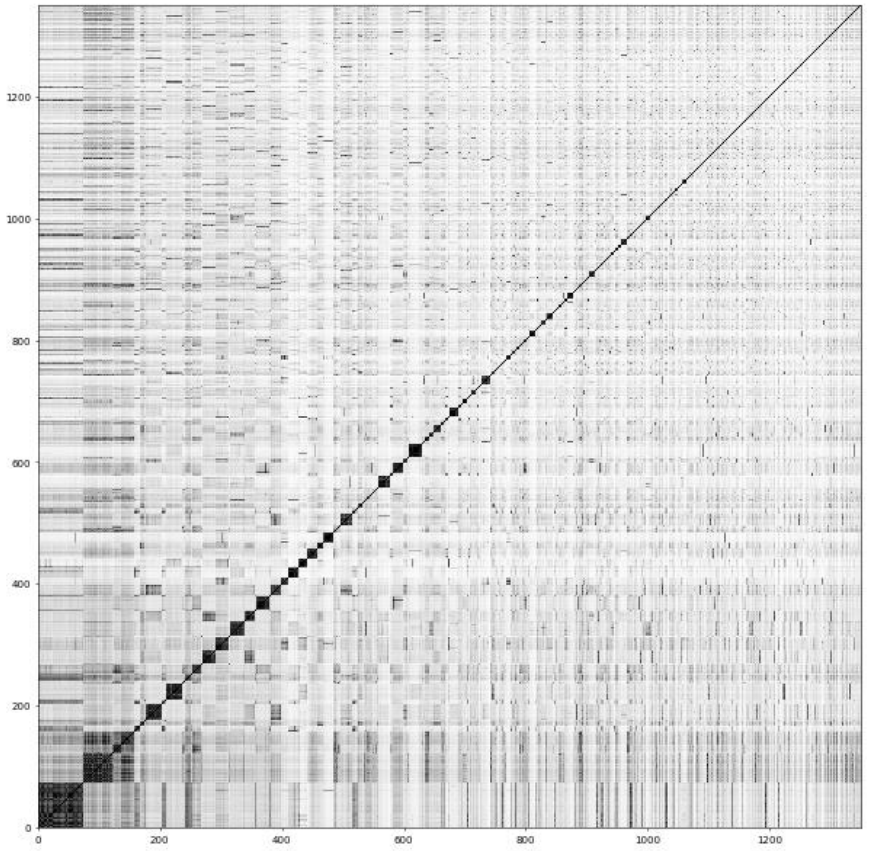
- Centroid 의 pairwise distance matrix 는 군집화 결과의 직관적인 이해를 도와줍니다.

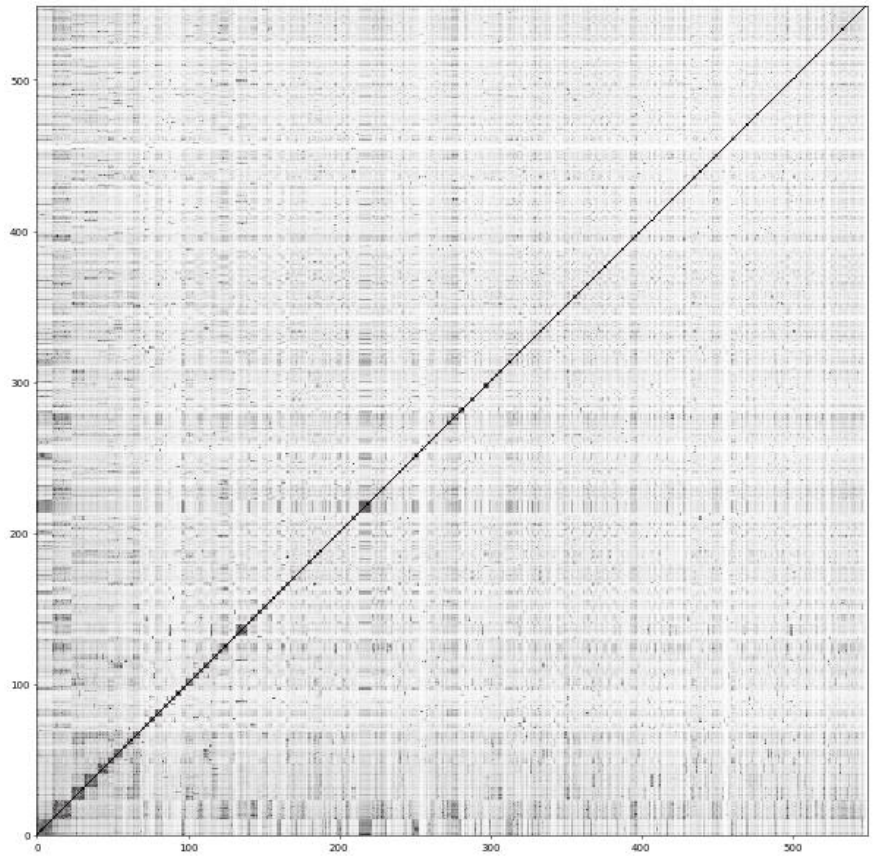
  - Diagonal 만 진할수록, 각 cluster 는 separation 이 잘 이뤄진 것입니다.
  - 진한 square 를 하나의 군집으로 묶을 수 있습니다.



Solutions 3. Defining *k*

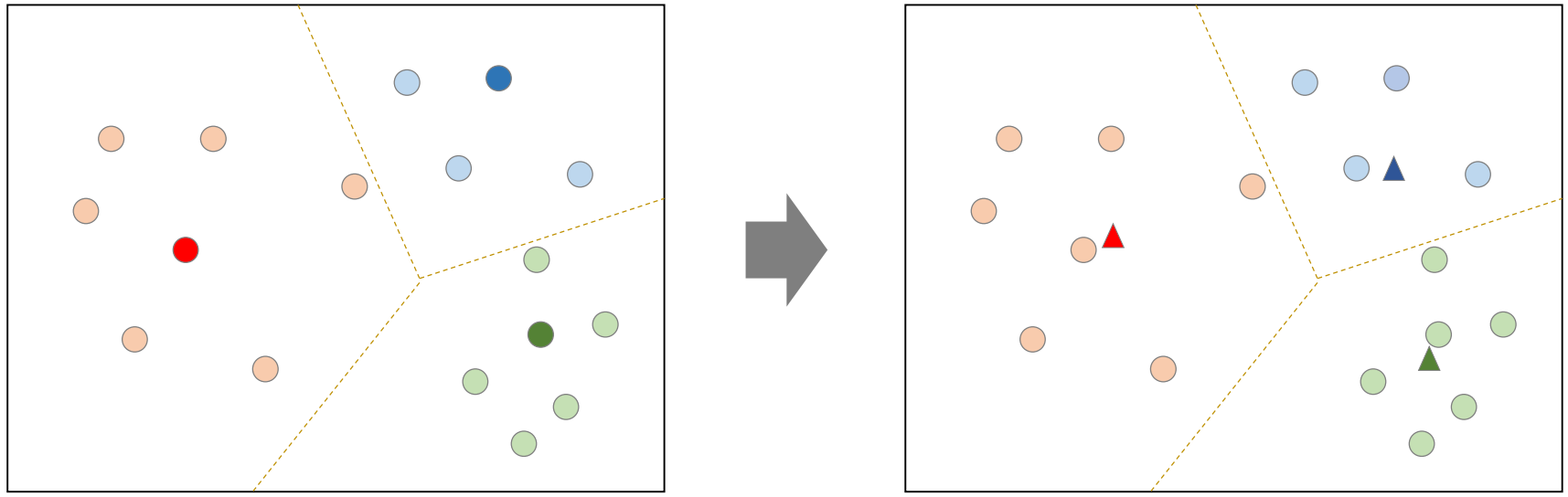
- Centroids 간의 거리가 가까운 군집들을 하나의 군집으로 병합할 수 있습니다.





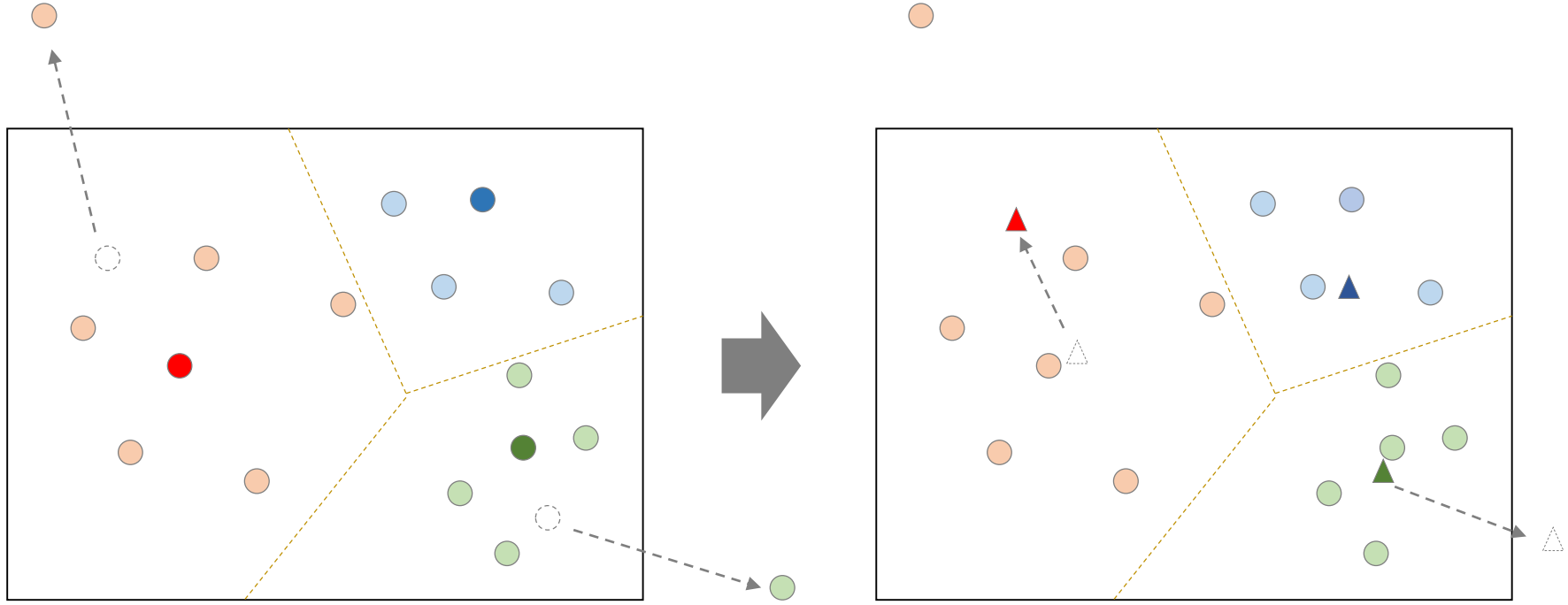
Limitations 4. Sensitive to noise points

- 모든 점을 반드시 한 개 이상의 군집으로 assign 하기 때문에, 일단 가장 가까운 군집에 할당되어 centroid 를 크게 움직입니다.



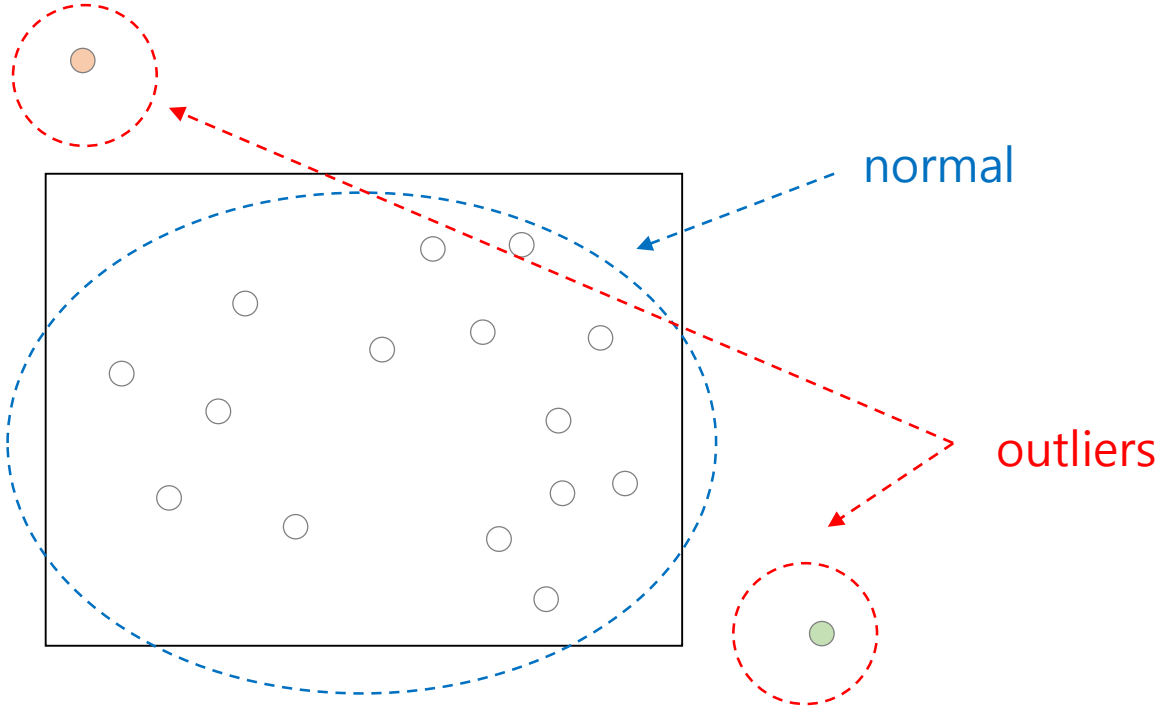
Limitations 4. Sensitive to noise points

- 모든 점을 반드시 한 개 이상의 군집으로 assign 하기 때문에, 일단 가장 가까운 군집에 할당되어 centroid 를 크게 움직입니다.

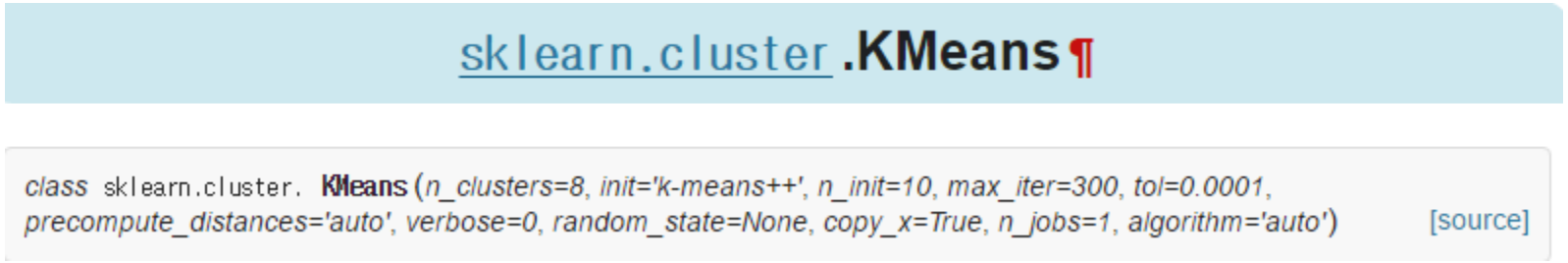


Solutions 4. Sensitive to noise points

- 데이터의 노이즈를 미리 제거하는 것이 좋습니다.
  - 앞서 살펴본 Isolation Forest 등의 방법이 이용될 수 있습니다.



Defining max iter



- *k*-means 는 수렴 속도가 매우 빠릅니다. max_iter=300 으로 설정할 필요가 없습니다.
  - *k*-means 는 근사알고리즘으로, 반복 학습으로의 성능향상에 한계가 있습니다.
  - 일반적으로 20 ~ 30 반복이면 거의 수렴합니다.

Defining max iter

- *k*-means type 은 몇 번의 반복으로 거의 수렴합니다.
  - 그렇다면 “repeat until converged” 동안의 학습은 더 좋은 걸까요?
  - Uniform effect 가 일어나는 과정일 수도 있습니다.

k=100, n = 30,091

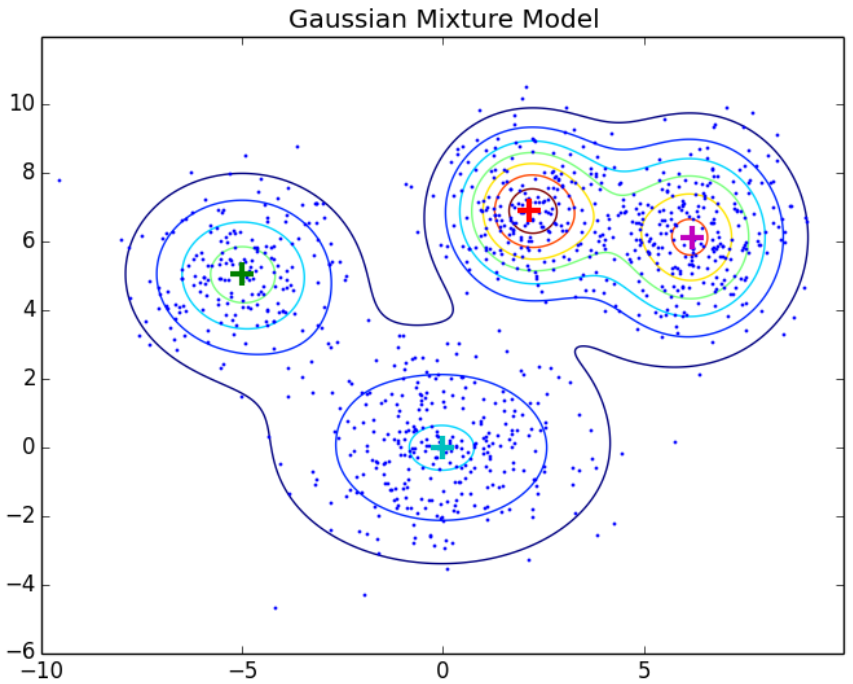
iteration	num_changed	inertia	times [sec]
1	29969	15323.44	4.435
2	5062	11127.62	4.466
3	2179	10675.31	4.463
4	1040	10491.64	4.449
5	487	10423.5	4.437
6	297	10392.49	4.483
7	178	10373.65	4.442
8	119	10362.63	4.449
9	78	10355.91	4.438
10	80	10350.7	4.452

Gaussian Mixture Model (GMM)

- 데이터와 centroids 간의 유사도를 정규분포의 확률값 $P(x_i \mid G_j)$ 으로 정의합니다.
  - $P(x_i \mid G_j) = c \cdot \exp\left(-\frac{(x_i - \mu_j)^2}{\Sigma_j^2}\right)$
  - 데이터의 분포 (밀도)를 고려한 k-means 라 생각할 수 있습니다.
  - Gaussian 을 이용하기 때문에 Euclidean distance 에 대해서만 정의됩니다.

Gaussian Mixture Model (GMM)

- 데이터가 Centroids 를 중심으로 Gaussian 을 따른다고 가정합니다.
  - 군집 사이에 밀도 차이가 있을 경우에 적합합니다.

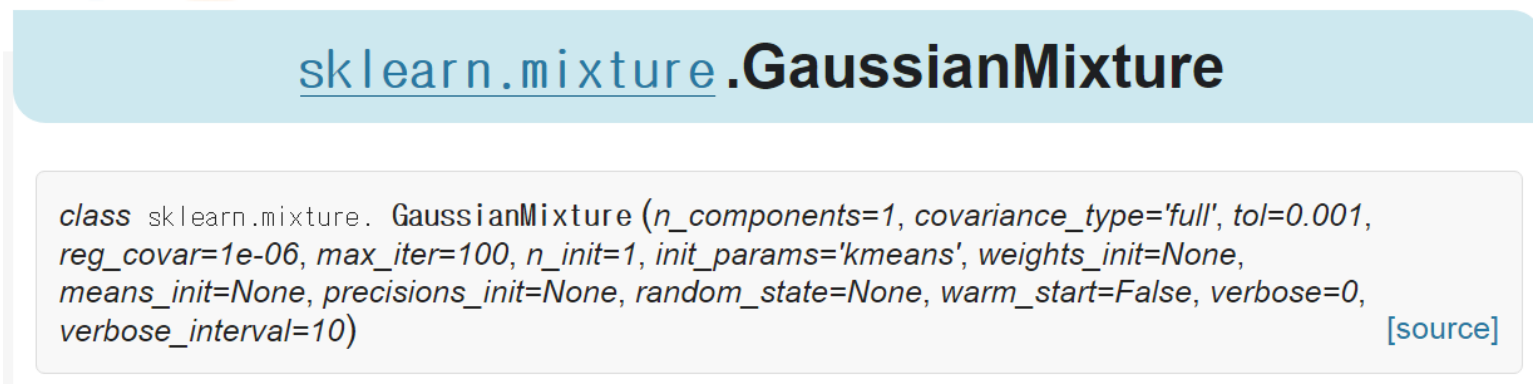


< 같은 확률값을 가지는 점들의 등고선 >

http://yulearning.blogspot.kr/2014/11/einsteins-most-famous-equation-is-emc2.html

Gaussian Mixture Model (GMM)

- scikit-learn 에 GMM 이 구현되어 있습니다.
  - *k*-means 처럼 n_components 를 사용자가 정의합니다.
  - 원형이 아닌 데이터의 분포를 학습하기 위해 분산행렬의 모양을 선택합니다.
    - covariance_type : {'full', 'tied', 'diag', 'spherical'}



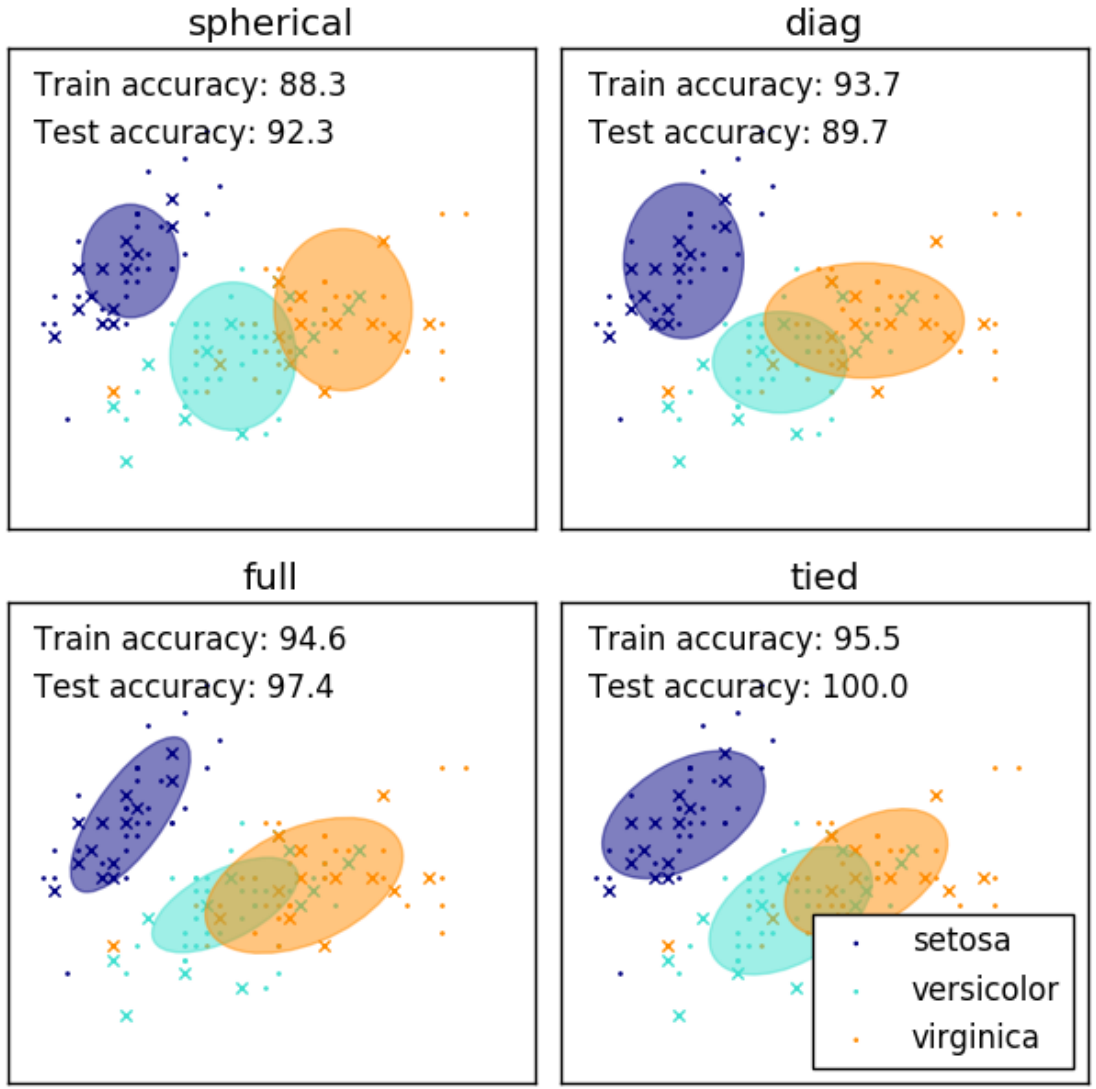
Gaussian Mixture Model (GMM)

'full': its own general covariance matrix,

'tied': share the same general covariance matrix,

'diag': its own diagonal covariance matrix,

'spherical': its own single variance

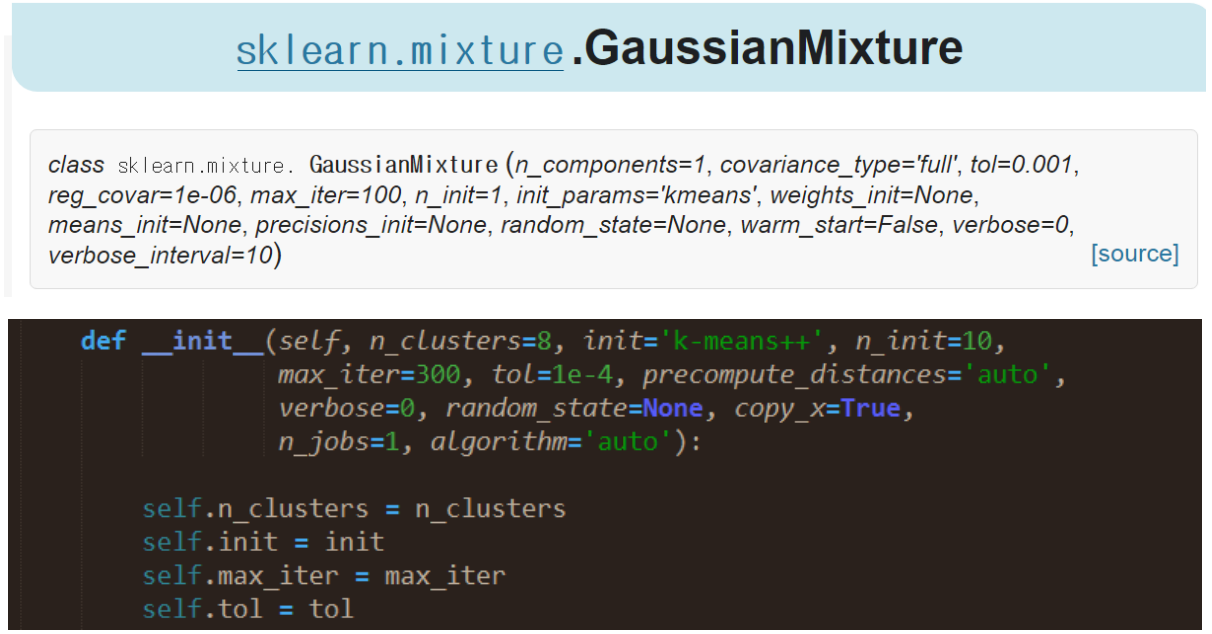


http://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture

Gaussian Mixture Model (GMM)

- Parameter initialization 을 위하여 *k*-means 이용합니다.*k*-means 의 기본 설정값이 지나치게 커서 초기화가 오래걸립니다. k_means.py 파일을 직접 열어서 기본값을 수정합니다.

(예시) .../anaconda/envs/YOURENVS/lib/python3.5/site-packages/sklearn/cluster/k_means_.py

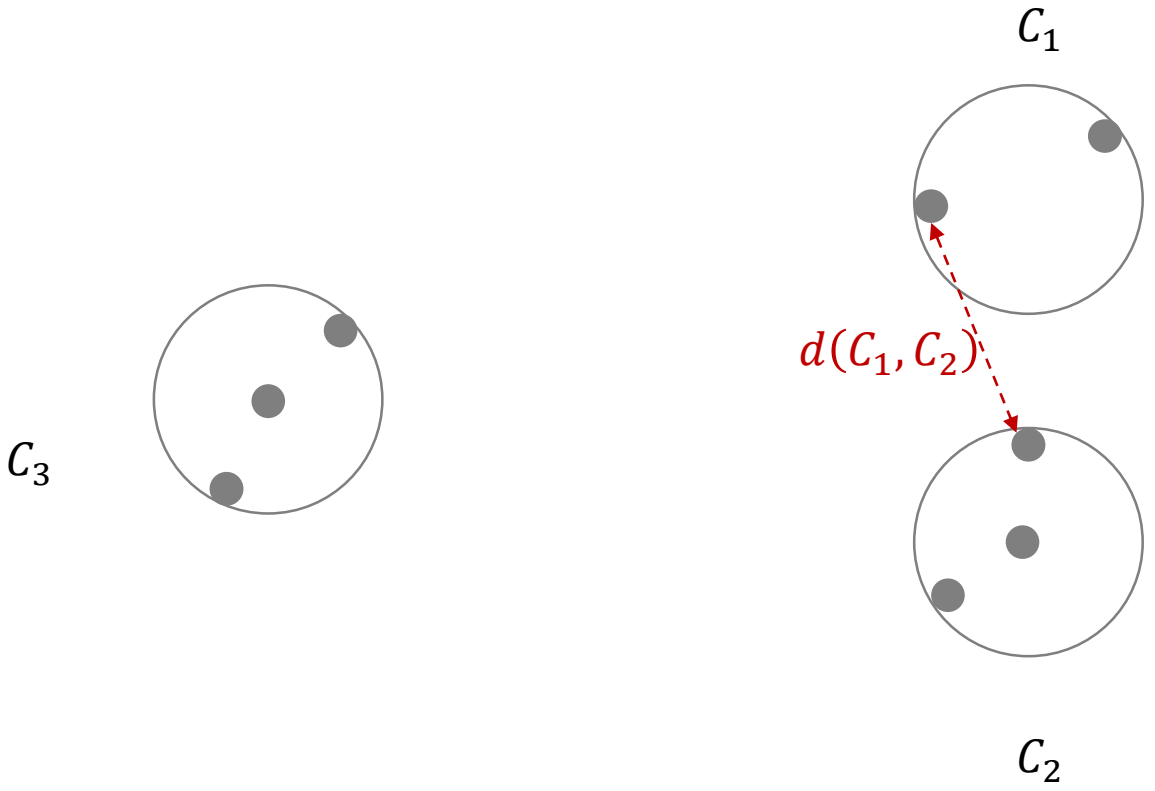


Hierarchical (agglomeration) clustering

- 두 데이터 x_i, x_j간에 정의되는 임의의 거리 $d(x_i, x_j)$ 를 이용할 수 있으며, 그룹 간 거리는 그룹 내 점들간의 거리를 이용한 다양한 방법으로 정의됩니다.
  - 최소 (single linkage)
  - 최대 (complete linkage)
  - 평균 (average linkage)
  - 분포 기반 거리 (Ward)
- 원하는 개수의 군집으로 묶일 때까지 가장 가까운 군집들을 병합합니다.

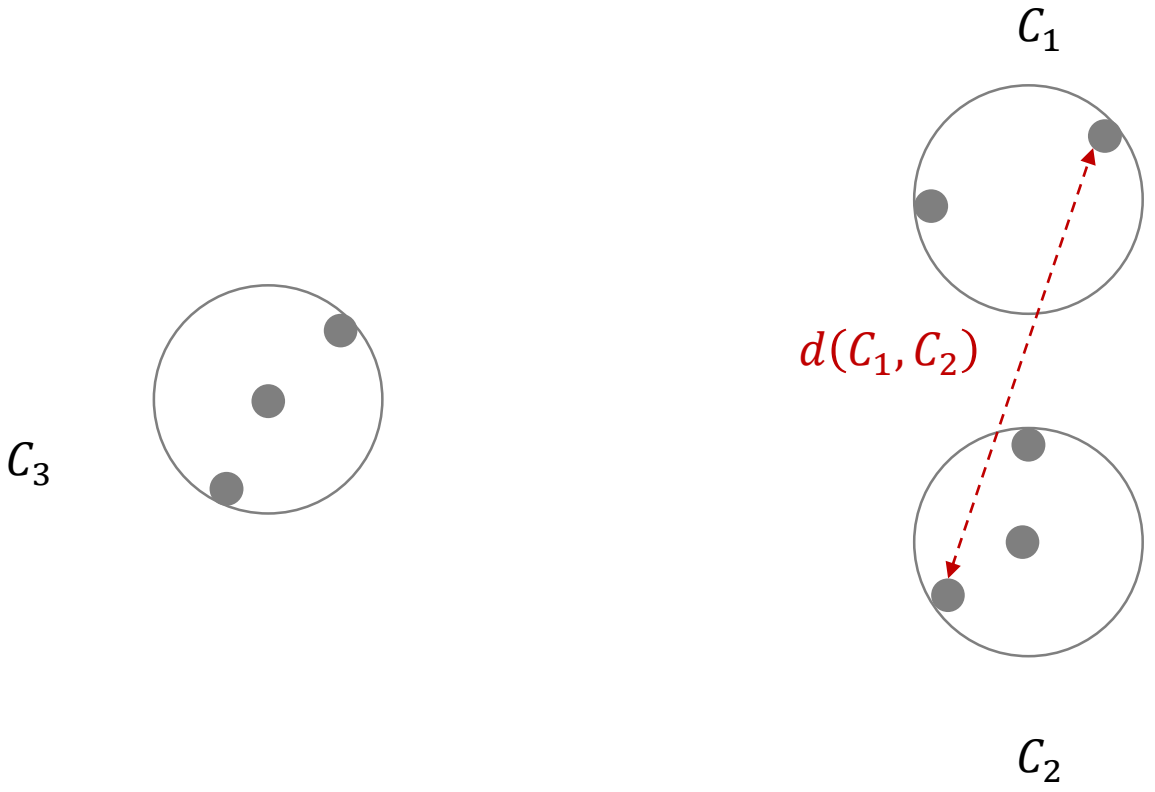
Hierarchical (agglomeration) clustering

- Single linkage



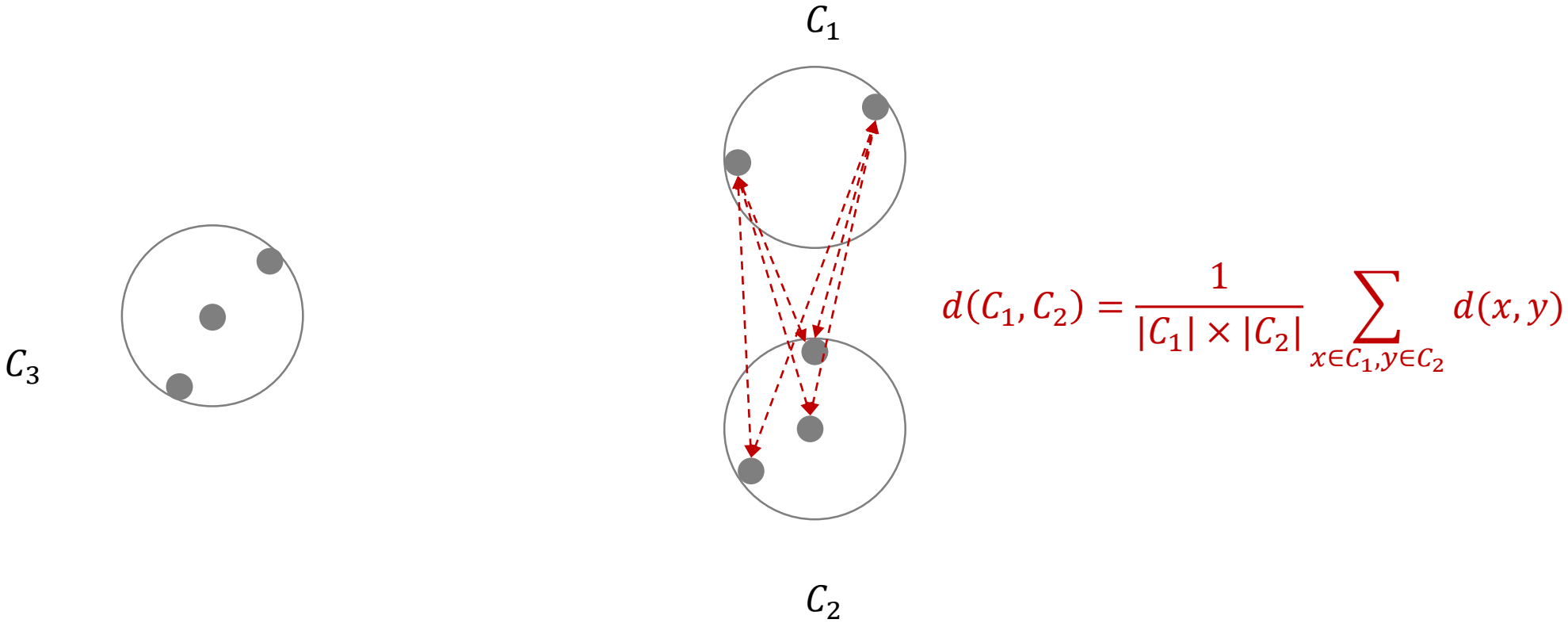
Hierarchical (agglomeration) clustering

- Complete linkage



Hierarchical (agglomeration) clustering

- Average linkage

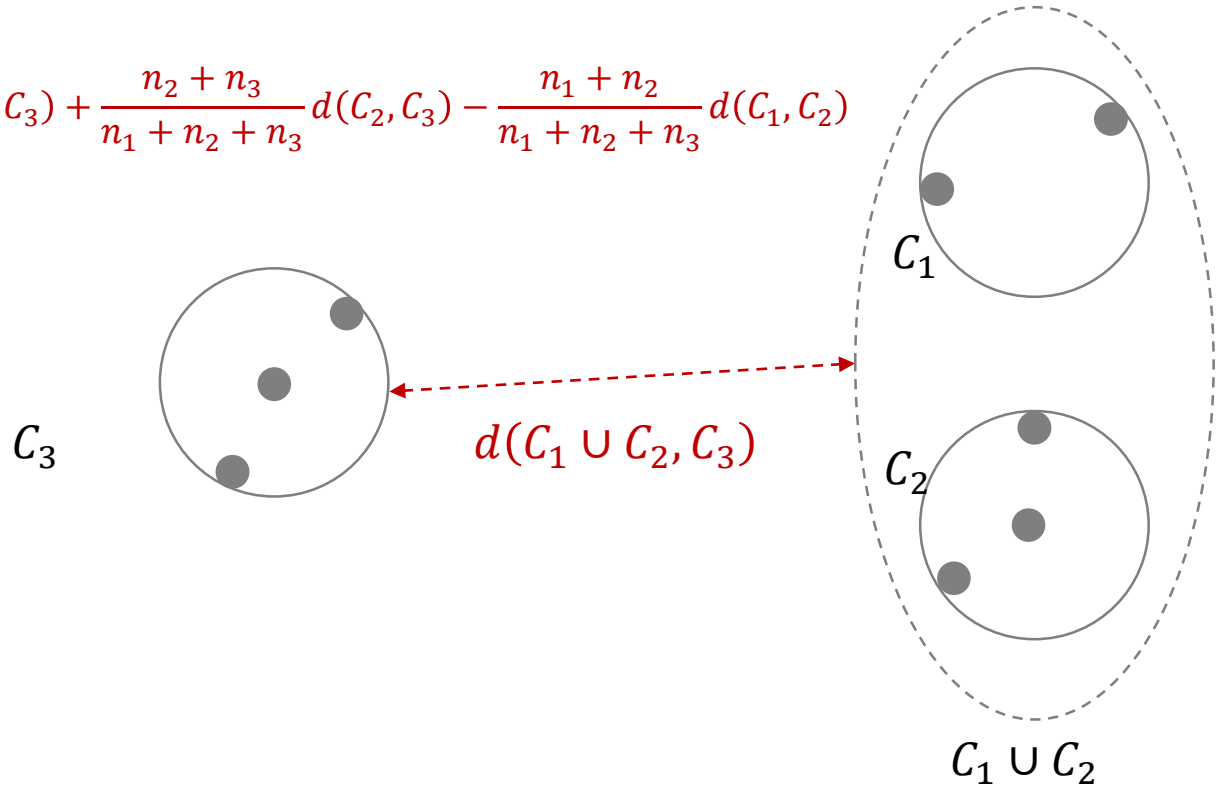


$$d(C_1, C_2) = \frac{1}{|C_1| \times |C_2|} \sum_{x \in C_1, y \in C_2} d(x, y)$$

Hierarchical (agglomeration) clustering

- Ward linkage 는 군집이 병합 된 이후의 거리를 계산합니다.

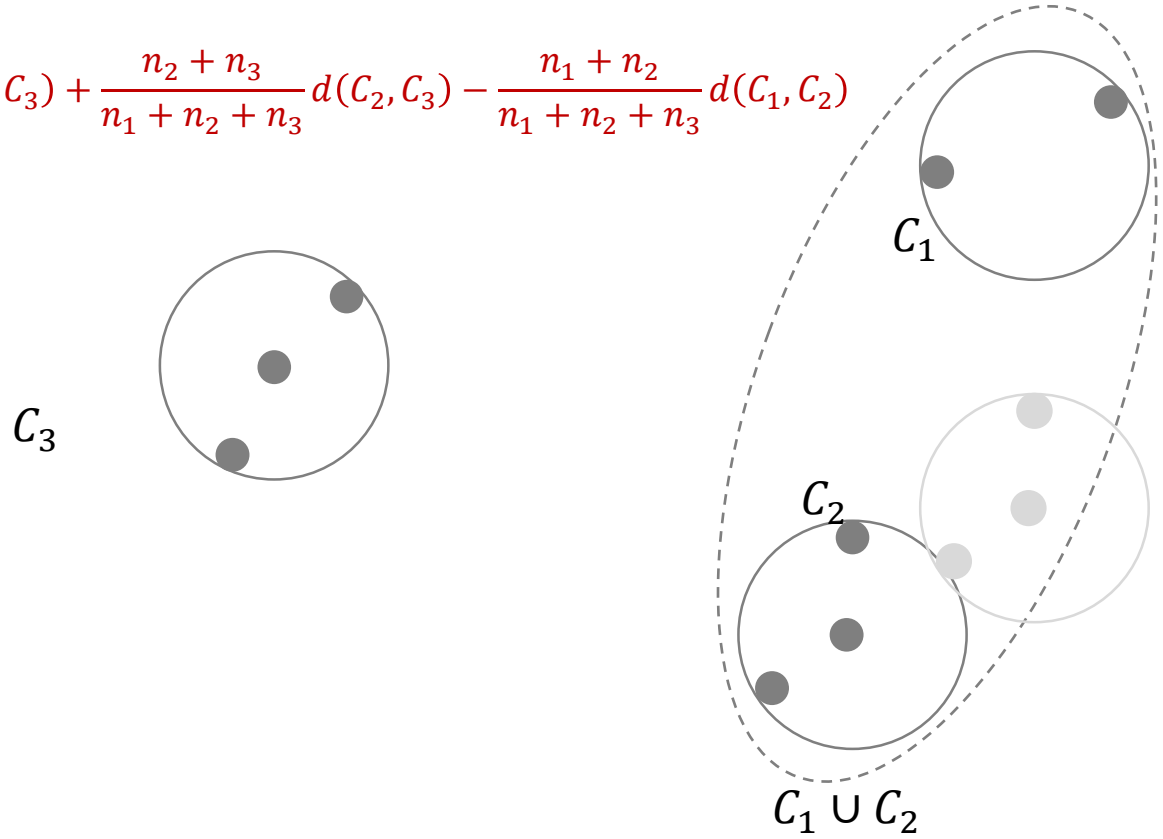
$$= \frac{n_1 + n_3}{n_1 + n_2 + n_3} d(C_1, C_3) + \frac{n_2 + n_3}{n_1 + n_2 + n_3} d(C_2, C_3) - \frac{n_1 + n_2}{n_1 + n_2 + n_3} d(C_1, C_2)$$



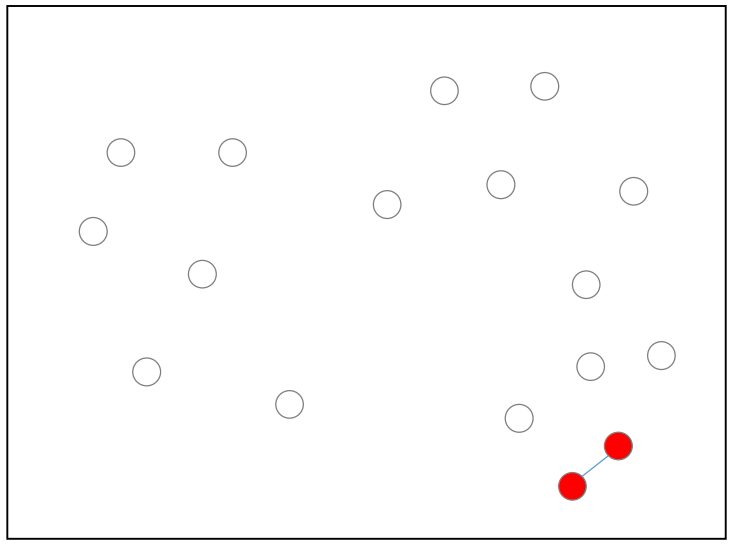
Hierarchical (agglomeration) clustering

- Ward linkage 는 C_1, C_2 가 멀리 떨어져 있으면 병합된 이후 $d(C_1 \cup C_2, C_3)$ 가 작아집니다.

$$= \frac{n_1 + n_3}{n_1 + n_2 + n_3} d(C_1, C_3) + \frac{n_2 + n_3}{n_1 + n_2 + n_3} d(C_2, C_3) - \frac{n_1 + n_2}{n_1 + n_2 + n_3} d(C_1, C_2)$$

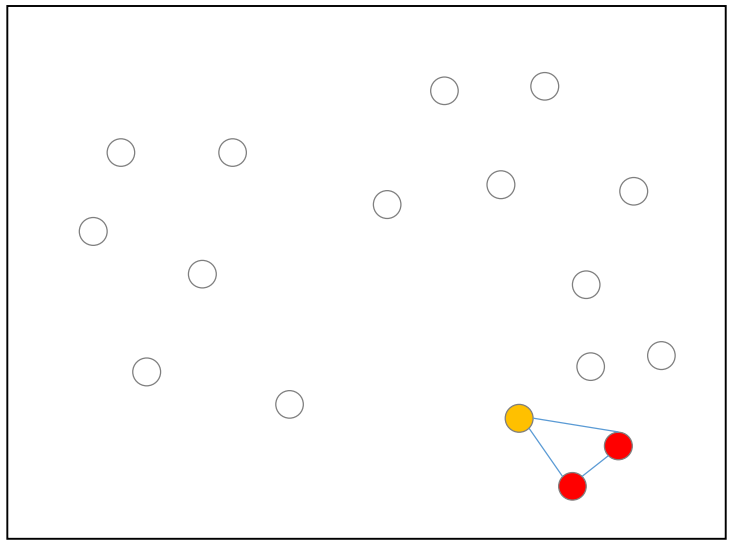


Hierarchical (agglomeration) clustering



<u>Iter = 1</u>
가장 가까운 두 점을 연결

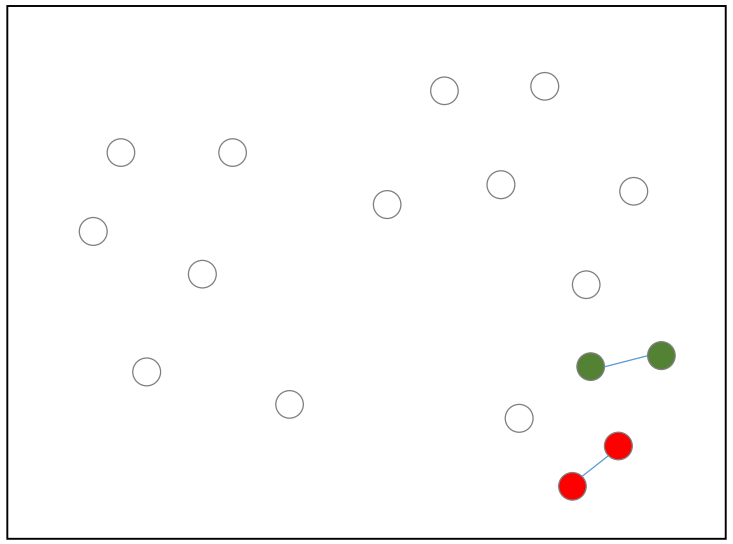
Hierarchical (agglomeration) clustering



<u>Iter = 1</u>
가장 가까운 두 점을 연결

<u>Iter = 2</u>
$d(C_i, C_j)$를 $d(x_p, x_q)$의 평균으로 정의한다면
두 빨간색점들과의 거리 평균이 다른 점들보다
가까우므로 주황색 점이 연결
(average linkage)

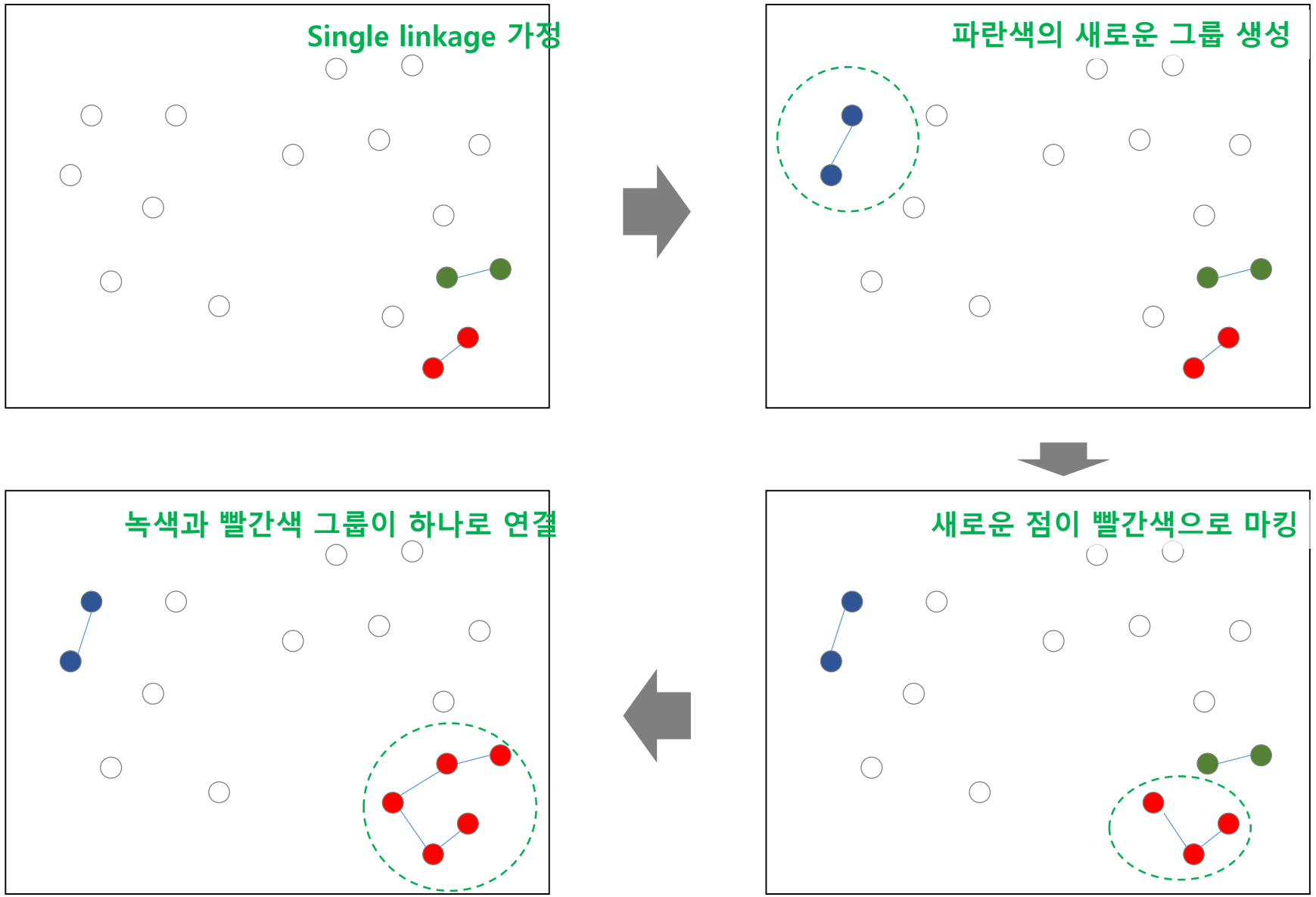
Hierarchical (agglomeration) clustering



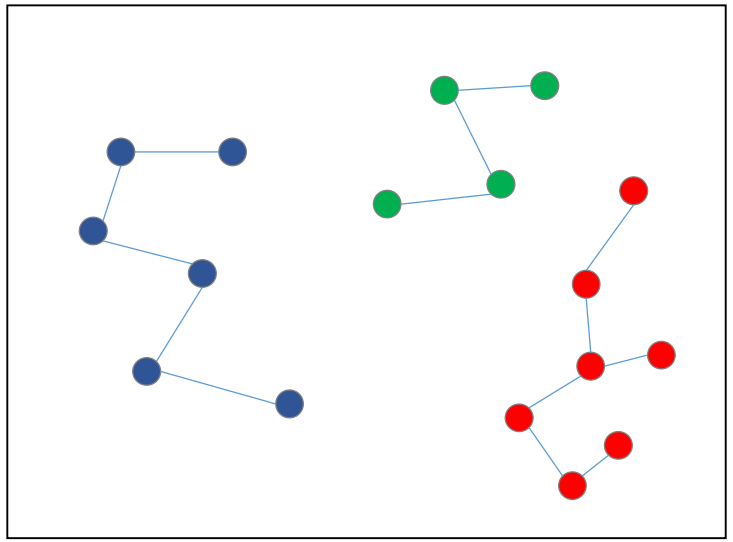
<u>Iter = 1</u>
가장 가까운 두 점을 연결

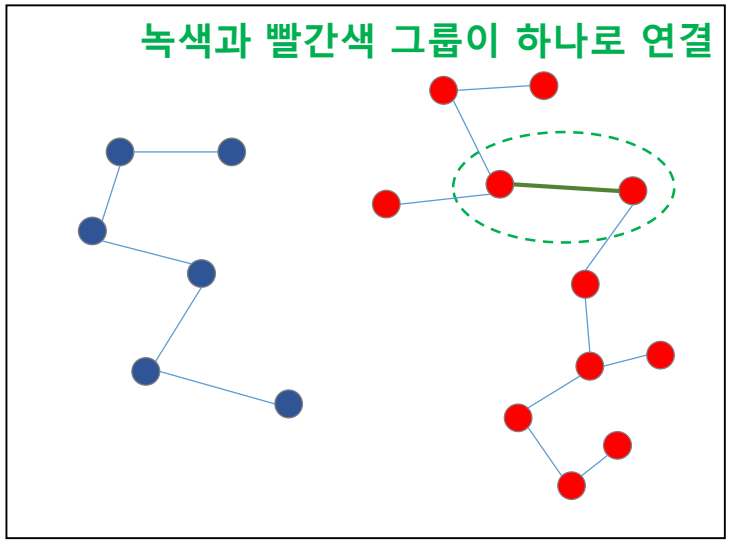
<u>Iter = 2</u>
$d(C_i, C_j)$를 $d(x_p, x_q)$의 min으로 정의한다면
녹색의 점이 하나로 연결
(single linkage)

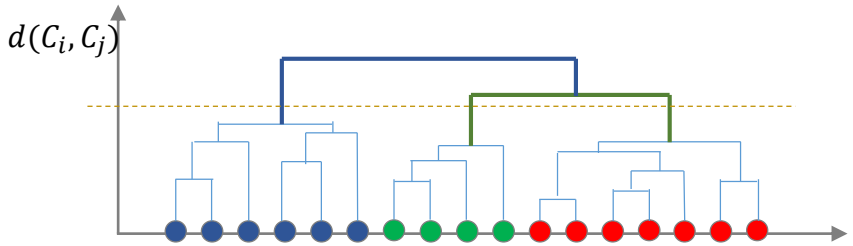
Hierarchical (agglomeration) clustering

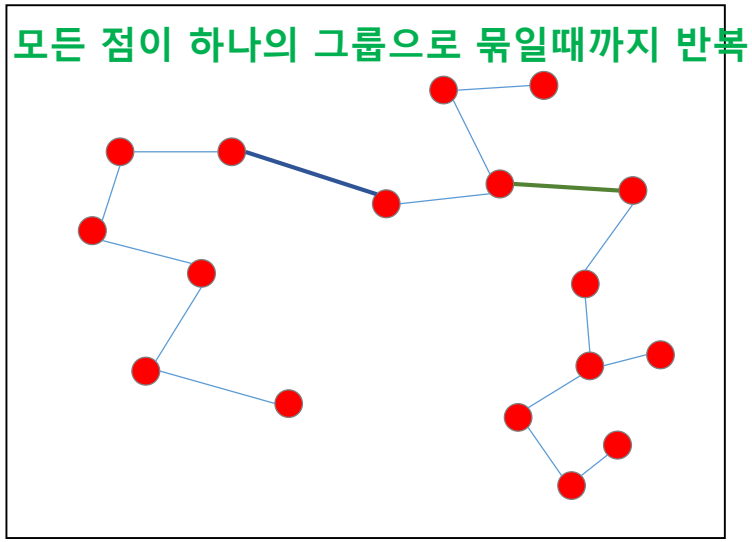


Hierarchical (agglomeration) clustering





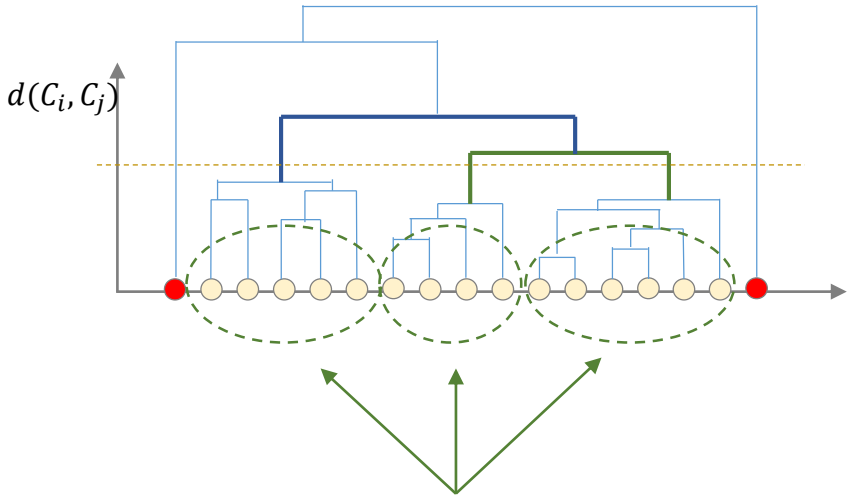




- Dendrogram은 링크가 생성되는 과정을 시각화한것
- 노란선의 distance로 cut한다는 것은 파란/녹색의 링크를 추가하지 않고 3개의 군집으로 묶겠다는 의미

Hierarchical (agglomeration) clustering

- Outliers 의 영향을 덜받습니다.
  - Single linkage 는 가장 가까운 점들을 하나씩 이어나갑니다.
  - 마지막까지 다른 점들과 큰 군집으로 묶이지 않는 점들이 outliers 입니다.



다른 점들은 큰 3개의 그룹으로 묶이지만,
붉은색 점들은 마지막에 큰 군집으로 묶임

Hierarchical (agglomeration) clustering

- 계산 비용이 비쌉니다.
  - 데이터의 개수가 n 개라고 할 때, 모든 점들간의 거리를 계산해야 하기 때문에 $O(n^2)$ 계산 공간과 비용이 필요합니다.
  - 상대적으로 k-means 보다 큰 계산 공간과 계산 시간을 필요로 합니다.

Hierarchical (agglomeration) clustering

- 고차원 벡터에서 잘 작동하지 않습니다.
  - 고차원에서는 최인접이웃들의 거리 외에는 정보력이 없습니다.
  - average linkage 는 두 군집의 모든 점들 간의 거리의 평균을 군집 간의 거리로 이용하기 때문에 대부분의 군집 간 거리가 비슷합니다.
  - 고차원 데이터에서는 최초의 몇 단계 외에는 의미가 없습니다.

Hierarchical clustering

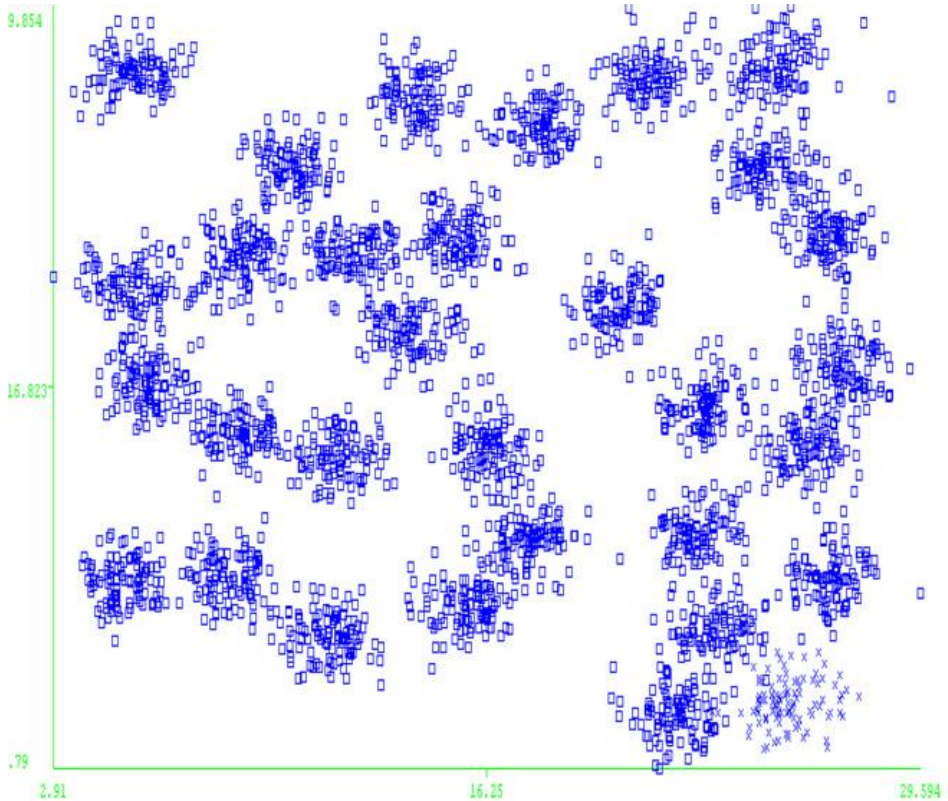
- Clustering ensemble 에서는 hierarchical clustering 을 이용합니다.
  - 고차원에서는 Euclidean 이나 Cosine 을 이용하여 데이터 간 거리를 정의하기 어렵습니다.
  - Clustering ensemble 은 여러 번의 클러스터링 결과를 이용하여 데이터 간의 유사도를 정의합니다. 잘 정의된 유사도가 주어진다면 hierarchical clustering 을 적용할 수 있습니다.

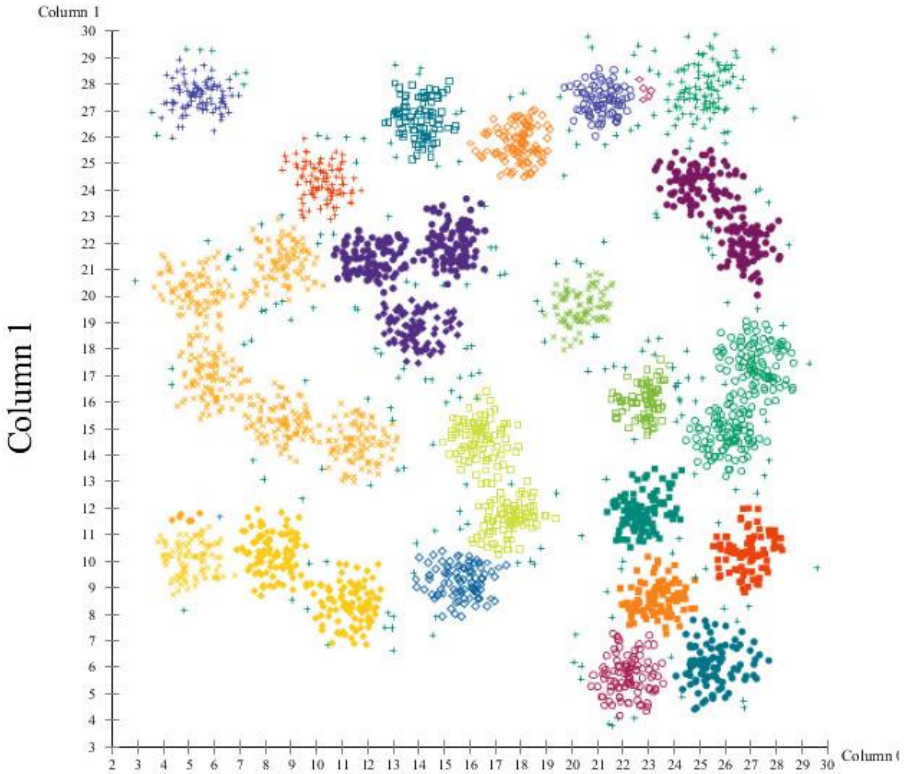
* https://github.com/lovit/soykmeans/blob/master/tutorials/ensemble_and_embedding.ipynb

DBSCAN

- Density-Based Spatial Clustering of Applications with Noise
  - 모든 점이 반드시 그룹에 속하지 않는다고 가정합니다 (노이즈)
  - 유사도
    - n 개의 데이터 X 에 대하여 두 데이터 x_i, x_j 간에 정의되는 임의의 거리 $d(x_i, x_j)$
  - 그룹화의 방식
    - Threshold 이상의 밀도를 지닌 점들을 모두 이어나갑니다.

DBSCAN





DBSCAN

- Parameters 에 민감합니다.
  - 군집을 결정하는 밀도값 threshold 에 의하여 데이터에서의 노이즈 비율이 예민하게 변합니다.
- 계산 비용이 큽니다.
  - DBSCAN 은 모든 점들간의 거리를 한 번 이상 계산해야하기 때문에 $O(N^2)$ 의 계산 비용을 필요로 합니다.