KU KONKUK UNIVERSITY

Clustering

자연어처리연구실

기계 학습 문제

- 분류 (classification)
  - 미리 정의된 범주에 입력 데이터를 할당하는 문제

- 군집화 (clustering)
  - 미리 정의된 규칙에 따라 데이터를 그룹화하는 문제

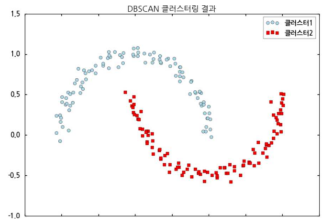


그림 출처: 블로그 참스터디-GodGo

What is Clustering?

- 군집화 (clustering)
  - 정해진 척도(measure)에 따라 유사성을 판단하고, 임계치 이상의 유사성을 가진 데이터를 그룹화하는 것
  - 유사성을 가진 데이터가 하나의 집합을 이루고, 비유사성을 가진 데이터가 다른 집합을 이룸

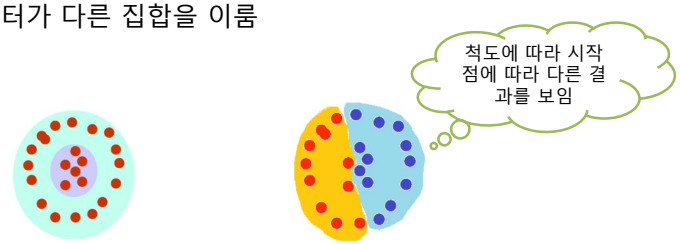


그림 출처: Ullman et al. 강의자료, Center for Brains in MIT, 2014

Prerequisite for Clustering

- 근접성 척도 (proximity measure)
  - 유사도 (similarity)
    - 두 벡터가 유사하면 커지는 척도 → Cosine similarity
  - 거리 (distance)
    - 두 벡터가 유사하면 작아지는 척도 → Euclidean distance
- 평가 기준 함수 (criterion function for evaluation)
  - 좋은 군집화 결과와 그렇지 못한 결과를 구분하는 척도
- 군집화 알고리즘 (clustering algorithm)
  - 평가 기준 함수를 최적화하기 위한 알고리즘

Proximity Measure

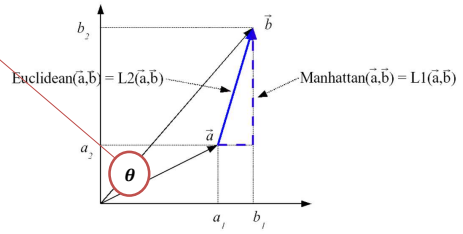
$$\text{sim}_{\text{cosine}}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

$$\text{sim}_{\text{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} \max(v_i,w_i)}$$

$$\text{sim}_{\text{Dice}}(\vec{v},\vec{w}) = \frac{2\times\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} (v_i+w_i)}$$

Similarity

Distance



Criterion Function

Hard & still evolving!

- Intra-cluster cohesion (compactness)
  - 클러스터 내 데이터 포인트들이 중심에 얼마나 가까이 모여 있는지 측정하는 척도
  - 중심과의 각 데이터 포인트 사이의 SSE(Sum of Squared Errors)를 측정하여 계산
- Inter-cluster separation (isolation)
  - 클러스터의 중심들이 서로 얼마나 떨어져 있는지 측정하는 척도
  - 클러스터 중심들 사이의 평균 거리를 측정하여 계산

Clustering Algorithms

- 분할적 군집화 (partitional clustering)
  - K-Means, DBSCAN, K-SOM
- 계층적 군집화 (hierarchical clustering)
  - Agglomerative algorithm (bottom-up)
  - Divisive algorithm (top-down)

K-Means Clustering

- K-Means
  - 1967년 MacQueen에 의해 개발된 가장 대표적이며 인기있는 분할적 군집화 알고리즘

Given *k*, the *k-means* algorithm works as follows:

1. Choose *k* (random) data points (seeds) to be the initial centroids, cluster centers
2. Assign each data point to the closest centroid
3. Re-compute the centroids using the current cluster memberships
4. If a convergence criterion is not met, repeat steps 2 and 3

그림 출처: Ullman et al. 강의자료, Center for Brains in MIT, 2014

K-Means Convergence Criterion

- K-Means 수렴 기준
  - 다른 클러스터로 소속을 변경하는 데이터 포인트가 더 이상 없을 때 (특정 기준 이하일 때)
  - 각 클러스터의 중심이 더 이상 변경되지 않을 때 (특정 기준 이하일 때)
  - SSE(Sum of Squared Errors)의 감소가 특정 기준 이하일 때

유클리디언 거리

$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} d(\mathbf{x}, \mathbf{m}_j)^2$$

클러스터 별 데이터 포인트 / 각 클러스터의 중심

K-Means Clustering Example

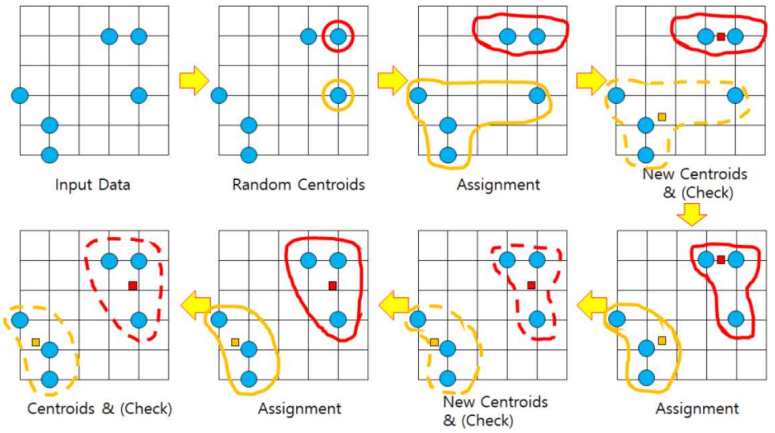
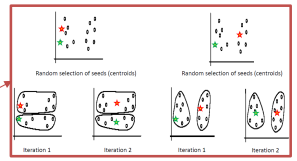
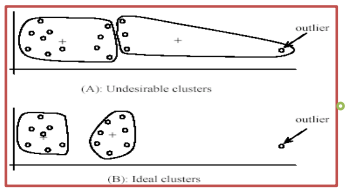


그림 출처: https://blog.naver.com/rla_rkdnjs/222321845904

Strength and Weakness of K-Means

- 장점
  - 단순하고 구현하기 쉬움
  - 시간 복잡도가 *O(tkn)*으로 효율적임
    - *t*: 반복 횟수, *k*: 클러스터의 수, *n*: 데이터 수
- 단점
  - 군집의 개수 k를 미리 정해줘야 함
  - 초기값에 영향을 많이 받음
  - 아웃라이어(outlier)에 취약함





데이터 포인트 일부를 제거하거나 랜덤 샘플링 하는 방법으로 완화

그림 출처: Ullman et al. 강의자료, Center for Brains in MIT, 2014

실습

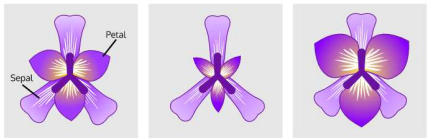
실습 코드 다운로드: https://github.com/KUNLP/Lecture

- K-Means 알고리즘을 이용하여 입력 데이터셋을 군집화 하는 프로그램을 작성하시오.

- 입력 데이터셋
  - IRIS dataset

  [자질 구성]
  - sepal length(꽃받침 길이)
  - sepal width(꽃받침 너비)
  - petal length(꽃잎 길이)
  - petal width(꽃잎 너비)

- 문제
  - 자질을 바탕으로 3종류의 분꽃을 군집화



실습

Clustering

```
from sklearn import datasets
from sklearn.cluster import KMeans
from matplotlib import pyplot as plt

# IRIS 데이터셋에서 데이터 부분만 읽어오기
iris = datasets.load_iris()
input_data = iris.data
print(input_data[0:3])

# 3개 군집으로 클러스터링하기 위한 모델 생성
model = KMeans(n_clusters=3)

# 모델에 데이터 입력 및 클러스터링 수행
model.fit(input_data)

# 군집 결과 받아오기
labels = model.predict(input_data)
```

초기값 10개에 대해 수행 후 최선 선택

n_init : *int, default=10*
Number of time the k-means algorithm will be run with different centroid seeds. The final results will be the best output of n_init consecutive runs in terms of inertia.

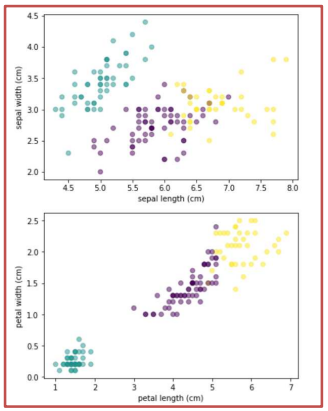
max_iter : *int, default=300*
Maximum number of iterations of the k-means algorithm for a single run.

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]]
```


실습

Visualization

```
# 꽃받침 기준 시각화
x = input_data[:, 0]
y = input_data[:, 1]
plt.scatter(x, y, c=labels, alpha=0.5)
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.show()

# 꽃잎 기준 시각화
x = input_data[:, 2]
y = input_data[:, 3]
plt.scatter(x, y, c=labels, alpha=0.5)
plt.xlabel('petal length (cm)')
plt.ylabel('petal width (cm)')
plt.show()
```



DBSCAN Clustering

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
  - 지역적 밀도를 고려하여 군집을 확장하는 방법으로 임의 형태의 군집을 찾아내는 알고리즘
  - Hyper-ellipsoid 형태의 군집만 가능한 K-Means의 단점인 극복

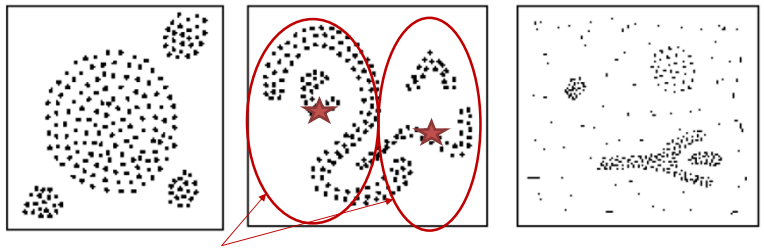
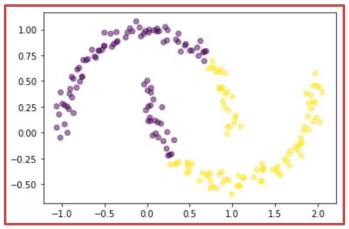


그림 출처: Ullman et al. 강의자료, Center for Brains in MIT, 2014

Hyper-ellipsoid Clustering of K-Means

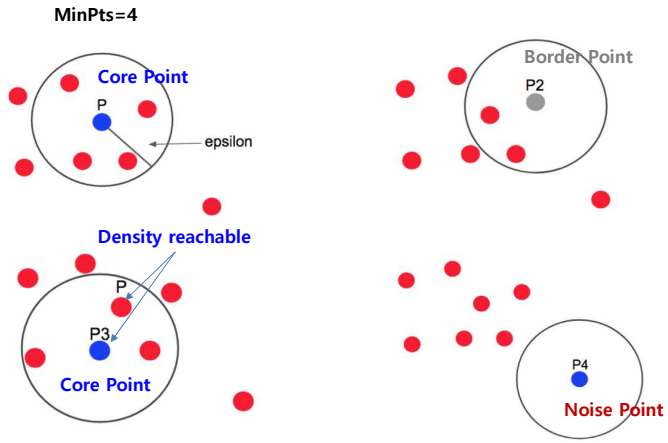
```
from sklearn import datasets
from sklearn.cluster import KMeans
from matplotlib import pyplot as plt

# 초승달 데이터 만들기
input_data, _ = datasets.make_moons(n_samples=200, noise=0.05)

# 2개 군집으로 클러스터링하기 위한 모델 생성
model = KMeans(n_clusters=2)

# 모델에 데이터 입력 및 클러스터링 수행
model.fit(input_data)

# 군집 결과 받아오기
labels = model.predict(input_data)

# 시각화
x = input_data[:, 0]
y = input_data[:, 1]
plt.scatter(x, y, c=labels, alpha=0.5)
plt.show()
```



DBSCAN Terms

- Eps
  - 밀도를 측정할 공간(원)의 반지름 (사용자가 지정)
- Density
  - Eps 공간 안에 존재하는 데이터 포인트의 수
- Core point
  - Density가 MinPts(최소 밀도)보다 큰 데이터 포인트
- Border point
  - Core point의 이웃이지만(core point와 같은 Eps 공간에 존재하지만) density가 MinPts보다 작은 데이터 포인트
- Noise point
  - Core point와 border point가 아닌 데이터 포인트

DBSCAN Terms



DBSCAN Algorithm

Let ClusterCount=0. For every point *p*:

1. If *p* it is not a core point, assign a null label to it [e.g., zero]
2. If *p* is a core point, a new cluster is formed
   [with label ClusterCount:= ClusterCount+1]

   Then find all points density-reachable from p and classify them in the cluster.
   [Reassign the zero labels but not the others]

Repeat this process until all of the points have been visited.

Since all the zero labels of border points have been reassigned in 2, the remaining points with zero label are noise.

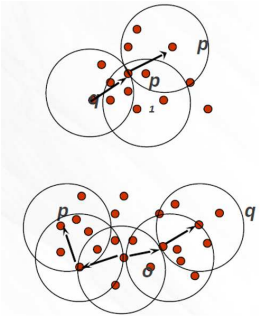


그림 출처: Ullman et al. 강의자료, Center for Brains in MIT, 2014

그림 출처: Garrett Poppe

DBSCAN Example

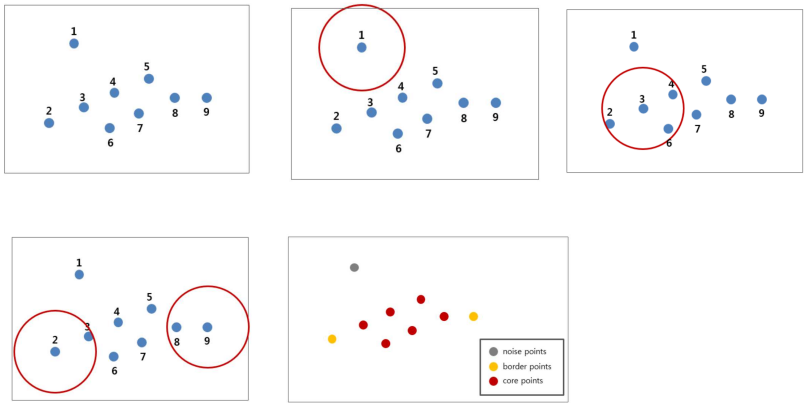
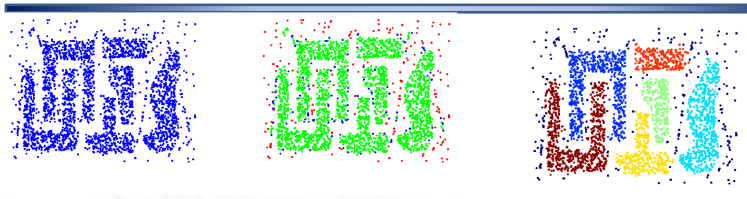


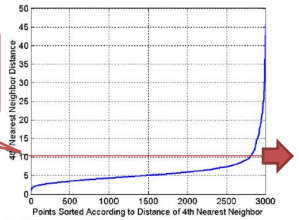
그림 출처: 참스터디 GodGo 블로그

DBSCAN: Large Eps vs. Optimal Eps



Large Eps **Optimal Eps**

k번째 가까운 포인트들의 거리를 오름차순으로 정렬



Eps=10

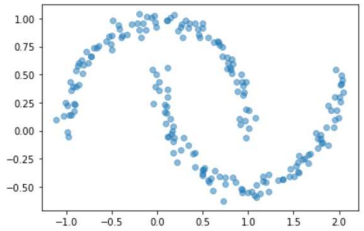
그림 출처: Garrett Poppe

실습

실습 코드 다운로드: https://github.com/KUNLP/Lecture

- DBSCAN 알고리즘을 이용하여 초승달 모양의 데이터셋(moon dataset)을 군집화하는 프로그램을 작성하시오.

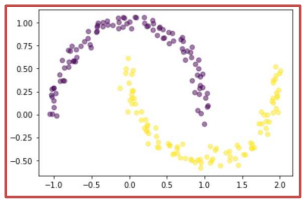
```
from sklearn import datasets
from matplotlib import pyplot as plt

# 초승달 데이터 만들기
input_data, _ = datasets.make_moons(n_samples=200, noise=0.05)
```



실습

Clustering & Visualization

```
from sklearn import datasets
from sklearn.cluster import DBSCAN
from matplotlib import pyplot as plt

# 초승달 데이터 만들기
input_data, _ = datasets.make_moons(n_samples=200, noise=0.05)

# DBSCAN 모델 생성
model = DBSCAN(eps=0.2, min_samples=5, metric='euclidean')

# DBSCAN 클러스터링 수행
labels = model.fit_predict(input_data)

# 시각화
x = input_data[:, 0]
y = input_data[:, 1]
plt.scatter(x, y, c=labels, alpha=0.5)
plt.show()
```



K-SOM (Kohonen Self Organization Maps)

- K-SOM
  - 출력층의 경쟁 학습(competitive learning)을 통해 가중치를 업데이트하는 방법으로 군집화를 수행하는 인공신경망
    - 경쟁 단계: 가장 큰 값을 가진 노드 선정 (Winner-Takes-All)
    - 협력 및 적응 단계: 승자 노드의 이웃노드의 가중치를 업데이트

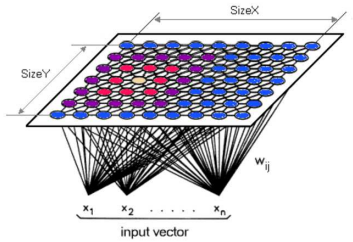


그림 출처: 아이리스님의 블로그

K-SOM Algorithm

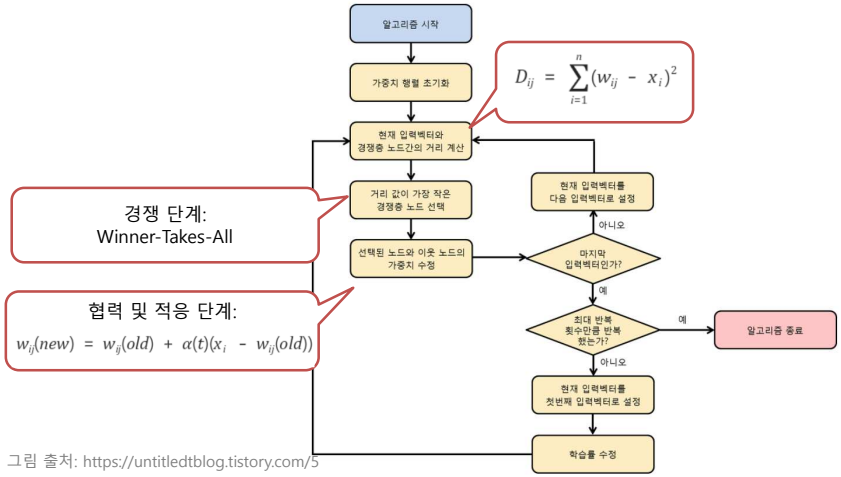


그림 출처: https://untitledtblog.tistory.com/5

확인 예제

- 4차원의 입력 벡터를 2개 그룹으로 클러스터링하는 K-SOM

입력 벡터 (4개)

(1, 0, 0, 1)

(0, 0, 1, 1)

(0, 1, 0, 1)

(0, 0, 1, 0)

학습률: α(0)=0.6

$\alpha(t+1) = 0.4 \times \alpha(t)$

초기 가중치

$$\begin{bmatrix} 0.5 & 0.1 \\ 0.3 & 0.7 \\ 0.6 & 0.8 \\ 0.2 & 0.2 \end{bmatrix}$$

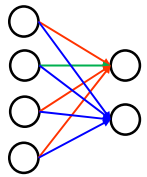
수정 가중치

$$\begin{bmatrix} 0.8 & 0.1 \\ 0.12 & 0.7 \\ 0.24 & 0.8 \\ 0.68 & 0.2 \end{bmatrix}$$

$D_{ij} = \sum_{i=1}^{n}(w_{ij} - x_i)^2$

$w_{ij}(new) = w_{ij}(old) + \alpha(t)(x_i - w_{ij}(old))$

$w_{12} = 0.3 + 0.6 \times (0 - 0.3) = 0.12$



$D_{11} = (0.5 - 1)^2 + (0.3 - 0)^2 + (0.6 - 0)^2 + (0.2 - 1)^2 = 1.34$

$D_{12} = (0.1 - 1)^2 + (0.7 - 0)^2 + (0.8 - 0)^2 + (0.2 - 1)^2 = 2.58$

Clustering by K-SOM

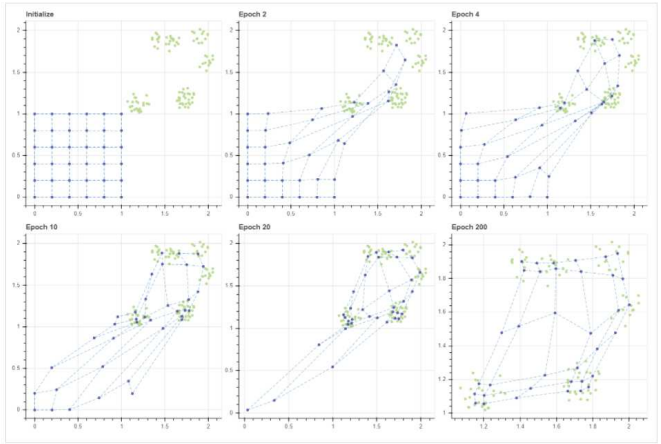


그림 출처: https://lovit.github.io/visualization/2019/12/02/som_part1/

실습

실습 코드 다운로드: https://github.com/KUNLP/Lecture

라이브러리 설치

```
from google.colab import drive
drive.mount("/gdrive", force_remount=True)
```

```
Mounted at /gdrive
```

```
!pip install sklearn-som
```

```
Collecting sklearn-som
  Downloading https://files.pythonhosted.org/pa
Requirement already satisfied: numpy in /usr/lo
Installing collected packages: sklearn-som
Successfully installed sklearn-som-1.0.1
```

Clustering

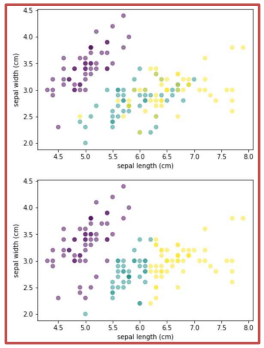
```
from sklearn import datasets
from sklearn_som.som import SOM
from matplotlib import pyplot as plt

# IRIS 데이터셋에서 데이터 부분만 읽어오기
iris = datasets.load_iris()
# 꽃받침 부분만 잘라내기 (2차원 데이터 만들기)
data = iris.data[:, :2]
label = iris.target
print(data[0:3])
print(label[0:3])

# 2차원 데이터를 3*1 행렬(3개로 군집)로 매핑하는 SOM 모델 생성
model = SOM(m=3, n=1, dim=2)
model.fit(data)

# 군집 결과 받아오기
predict = model.predict(data)
```

```
[[5.1 3.5]
 [4.9 3. ]
 [4.7 3.2]]
[0 0 0]
```

실습

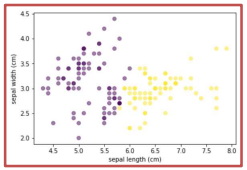
Visualization

```
# 원본 시각화
x = data[:, 0]
y = data[:, 1]
plt.scatter(x, y, c=label[:], alpha=0.5)
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.show()

# SOM 클러스터링 결과 시각화
x = data[:, 0]
y = data[:, 1]
plt.scatter(x, y, c=predict[:], alpha=0.5)
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.show()
```



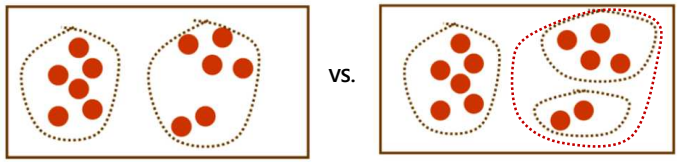
2개 군집으로 실행한 경우

```
model = SOM(m=2, n=1, dim=2)
```



Needs of Hierarchical Clustering

- 계층적 군집화의 필요성
  - 밀도를 기준으로 겹겹이 묶는 것이 많은 경우에 이상적



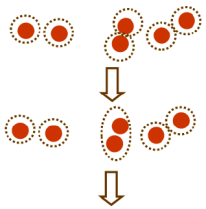
분할적 군집화 계층적 군집화

- Agglomerative algorithm (bottom-up)
- Divisive algorithm (top-down)

그림 출처: Ullman et al. 강의자료, Center for Brains in MIT, 2014

Agglomerative Hierarchical Clustering

initialize with each example in singleton cluster
while there is more than ***1*** cluster
1. find 2 nearest clusters
2. merge them



- Four common ways to measure cluster distance
  1. minimum distance $d_{min}(D_i, D_j) = \min_{x \in D_i, y \in D_j} \| x - y \|$
  2. maximum distance $d_{max}(D_i, D_j) = \max_{x \in D_i, y \in D_j} \| x - y \|$
  3. average distance $d_{avg}(D_i, D_j) = \frac{1}{n_i n_j} \sum_{x \in D_i} \sum_{y \in D_j} \| x - y \|$
  4. mean distance $d_{mean}(D_i, D_j) = \| \mu_i - \mu_j \|$

자료 출처: Ullman et al. 강의자료, Center for Brains in MIT, 2014

Single Linkage or Nearest Neighbor

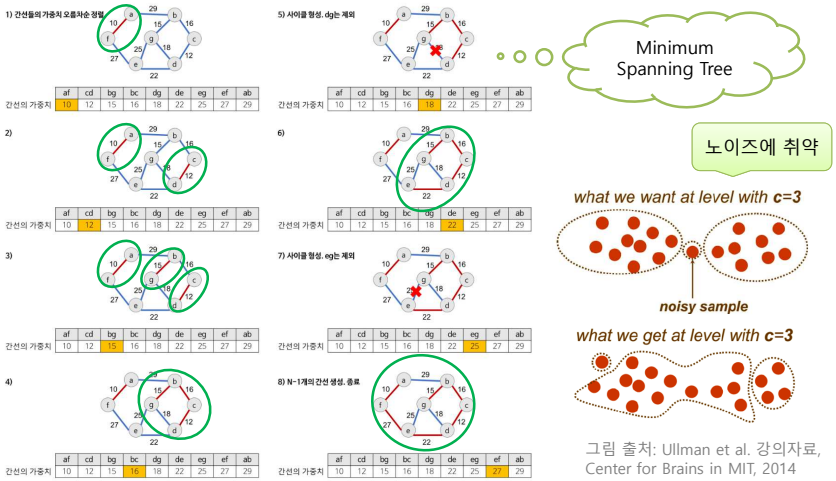


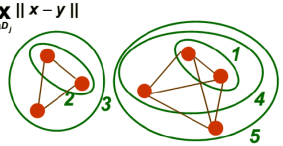
그림 출처: Ullman et al. 강의자료, Center for Brains in MIT, 2014

Complete Linkage or Farthest Neighbor

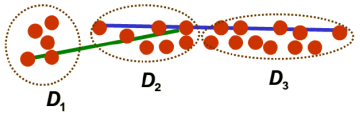
- Agglomerative clustering with maximum distance

$$d_{max}(D_i, D_j) = \max_{x \in D_i, y \in D_j} \| x - y \|$$

- encourages compact clusters



- Does not work well if elongated clusters present



1. 군집 간 거리 계산: 가장 먼 데이터 포인트 사이의 거리
2. 군집화: 군집 간 거리가 가장 가까운 두 군집을 그룹화

- $d_{max}(D_1, D_2) < d_{max}(D_2, D_3)$
- thus D_1 and D_2 are merged instead of D_2 and D_3

자료 출처: Ullman et al. 강의자료, Center for Brains in MIT, 2014

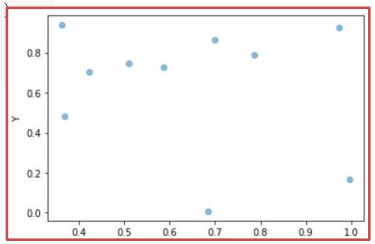
Divisive vs. Agglomerative

- Divisive Algorithm (top-down)
  - 전체 데이터 분포를 고려하여 군집화를 수행할 수 있으므로 최적의 공간 분할이 가능함
  - 실행 속도가 느림
- Agglomerative Algorithm (bottom-up)
  - 수행 속도가 빠름
  - 전체 데이터 분포를 고려하지 못해서 잘못된 군집(local optima)을 수행할 가능성이 있음

실습

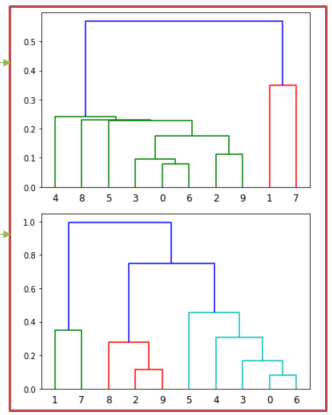
실습 코드 다운로드: https://github.com/KUNLP/Lecture

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.cluster.hierarchy import dendrogram, linkage

# 2차원 랜덤 넘버 생성
data = np.random.rand(10,2)

# 원본 시각화
x = data[:, 0]
y = data[:, 1]
plt.scatter(x, y, alpha=0.5, label='True Position')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```



실습

```
# Single-Linkage 클러스터링 수행
hcluster = linkage(data, method='single')

# 클러스터링 결과 시각화
dendrogram(hcluster)
plt.show()

# Complete-Linkage 클러스터링 수행
hcluster = linkage(data, method='complete')

# 클러스터링 결과 시각화
dendrogram(hcluster)
plt.show()
```



질의응답

Q&A

Homepage: http://nlp.konkuk.ac.kr
E-mail: nlpdrkim@konkuk.ac.kr

강의를 마치며

- 지금까지 기계학습의 개념을 시작으로 전통적인 기계학습 모델들을 소개하고, 최신의 딥러닝 모델들까지 살펴보았습니다.
- 여러분의 연구 분야에 기계학습을 접목하는데 소개 드린 개념과 예제 코드가 도움이 되기를 바랍니다.
- 긴 시간 강의 들으시느라 수고 많으셨습니다!