계층적 clustering (Hierarchy)

↳ 하나의 Data sample을 하나의 cluster로 보고 가장 유사도가 높은 cluster를 합치면서 cluster 갯수를 줄여 나가는 방법.

$K=N \Rightarrow K=N-1 \Rightarrow \cdots \Rightarrow K=1$

⇒ Data 간의 거리가 아니라, cluster간의 거리를 측정.

Hierarchy clustering에는 ① 비귀납적 방법 ② 귀납적 방법이 있다.

① 비귀납적 방법

i) Centroid : 두 cluster의 중심점(centroid)를 정의한 다음, 두 중심점의 거리를 cluster간의 거리로 정의한다.

↳ $d(u,v) = \| c_u - c_v \|_2$ (여기서 c_u과 c_v는 각각 두 cluster u와 v의 중심점)

ii) Single : 클러스터 u의 모든 데이터 i와 클러스터 v의 모든 Data j의 모든 조합에 대한 거리를 측정해서 최소값을 구한다. 최단 거리(Nearest Point) 방법이라고도 한다.

↳ $d(u,v) = \min(dist(u[i], v[j]))$

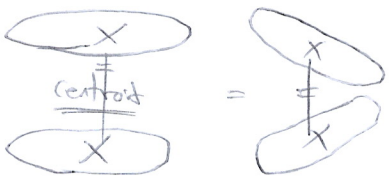
ii) Complete : 클러스터 u의 모든 Data i와 클러스터 v의 모든 Data j의 모든 조합에 대한 거리를 측정해서 최장근 값을 구한다. Farthest Point Algorithm 또는 Voor Hees Algorithm이라고 한다.

↳ $d(u,v) = \max(dist(u[i], v[j]))$

iv) Average : 클러스터 u의 모든 Data i와 클러스터 v의 모든 Data j의 모든 조합에 대한 거리를 측정후, 평균을 구한다. $|u|$와 $|v|$는 각각 두 클러스터의 원소의 갯수를 뜻한다.

↳ $d(u,v) = \sum_{ij} \frac{d(u[i], v[j])}{|u||v|}$

Diagram



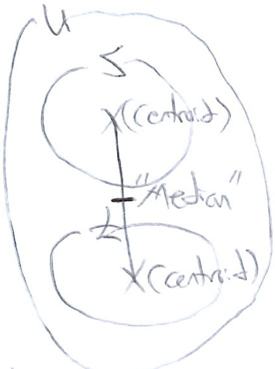
⇒ Centroid로 클러스터간의 거리를 정의한다면 왼쪽그림과 같이 클러스터간의 거리는 같지만, 실제 쭉같지고 보기 힘듦.
따라서, Single, Complete, Average가 사용.

⇒ Single 방식 : 가장 짧은 거리 방법
Complete 방식 : 가장 긴 거리 방법.
Average 방식 : 평균 거리 방법.

✗ 비귀납적 방법의 문제점 : 계산량이 너무 많기 때문에, 많은 Data를 실제 사용하기는 현실적으로 힘듦.
따라서, 보통 귀납적 방법을 사용.

② 귀납적 방법

i) Median : 이 방법은 Agglomerative clustering에서 사용할 수 있는 귀납적 방법으로 Centroid 방법의 변종이다. 만약 cluster "u"가 cluster s와 cluster t가 결합하여 생겼다면, cluster u의 중심점은 새로 계산하지 않고 원래 클러스터의 중심점의 평균을 사용한다.



ii) weighted : 이 방법도 Agglomerative Clustering에서 사용할 수 있는 귀납적 방법이다. 만약 cluster u가 cluster s와 cluster t가 결합해서 생겼다면, 다음과 같이 원래 cluster 까지의 두 거리의 평균을 사용한다.

$$d(u,v) = (dist(s,v)+dist(t,v))/2$$

iii) Ward : 이 방법도 Agglomerative clustering에서 사용할 수 있는 귀납적 방법이다. 만약 cluster u가 cluster s와 cluster t가 결합한 것이라면, 두 cluster 간의 거리의 가중평균에서 원래의 두 클러스터 사이의 거리를 보정한 값을 사용한다.

$$d(u,v) = \sqrt{\frac{|v|+|s|}{|v|+|s|+|t|}d(v,s)^2 + \frac{|v|+|t|}{|v|+|s|+|t|}d(v,t)^2 - \frac{|v|}{|v|+|s|+|t|}d(s,t)^2}$$

여기서 |·|는 클러스터의 원소 갯수를 의미함.

- Scipy에 Hierarchy clustering을 쓸수 있음.

Z = linkage(X, 'ward') # X가 150

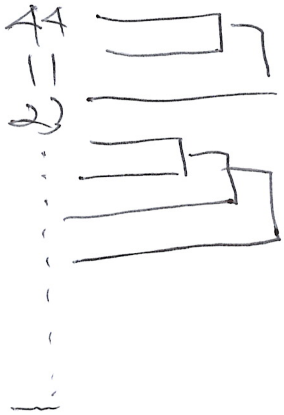
⇓

```
array[[52.,  53.,  0.04151,  2.],    ← 150번 cluster ⇒ "가장 처음 합쳐진 것"
      [14.,  79.,  0.05914,  2.],    ← 151번 cluster
       ⋮
      [62., 152.,  0.1726,   3.],
```

(52. : Index#, 53. : Index#, 0.04151 : 두개의 cluster 간의 거리 (Data간의 거리임), 2. : 원소의 개수)

62 → (33, 68, 62)

- Dendrogram



└▸ 나열 순서: 비슷한 Data 끼리 있도록 재배열된 것임 (원래는 1, 2, 3, ⋯, N)
　　└▸ 일종의 "유사도 Sorting"이 됨.

가우시안 혼합 모형과 EM 방법

가우시안 혼합 모형 (Gaussian Mixture)

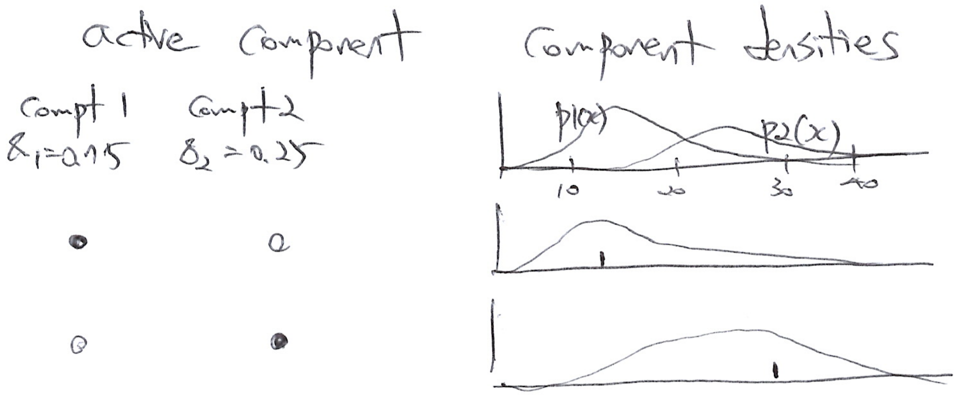
└▸ 실수 값을 출력하는 확률 변수 X가 사실 눈에 보이지 않는 (관측되지 않는) K-class 카테고리 확률 변수 Z의 값에 따라, 다른 기댓값과 분산을 가지는 복수의 가우시안 정규 분포들로 이루어진 모형을 가우시안 혼합 모형이라고 한다.

$$p(x) = \sum_{z} p(z)\cdot p(x|z) = \sum_{k=1}^{K} \theta_k N(x \mid \mu_k, \Sigma_k)$$

- $p(x)$: 전체 분포 · $p(x|z)$: 혼합모형의 각 성분이 되는 개별적인 연속확률 분포
- $p(z)$: 카테고리확률분포 · θ_k: 카테고리 확률 분포의 모수. 모든 성분중에 특정한 z가 선택될 확률.
- $\sum \theta_k = 1$: 카테고리 확률 분포의 모수 제한 조건.

베르누이-가우시안 혼합 모형 (Bern-Gaussian Mixture)

— 카테고리가 2개인 가우시안 혼합 모형은 베르누이-가우시안 혼합모형 이라고 한다.



observations

14.2

29.4

> Bern-Gaussian Mixture 의 문제점은 우리가 아는 것은 observation 뿐이라, 우리가 원하는 것은 어떤 확률모형에서 나왔는가 하는 것임.
> ⇓
> 2개를 섞어서 각각의 모수 값들을 알고싶은 것임
> ⇓
> 우리는 전체 pdf는 아는데 7개의 상수 값을 모르며
> ⇓
> 우리가 미분을 해서 모수값이 가장 큰걸

EM (Expectation – Maximization)

- 혼합 모델의 모수추정 에서 중요한 역할을 하는 것중에 하나가 바로 각 data가 어떤 Category 에 속하는 지를 알려주는 조건확률 $p(z|x)$ 값이다. 이 값을 responsibility라고 한다.

$$r_{ik} = p(z_i = k \mid x_i) = \frac{p(z_i=k)\cdot p(x_i \mid z_i=k)}{p(x_i)}$$

$$= \frac{p(z_i=k)\cdot p(x_i \mid z_i=k)}{\sum_{k=1}^{K} p(x_i, z_i=k)}$$

$$= \frac{p(z_i=k)\cdot p(x_i \mid z_i=k)}{\sum_{k=1}^{K} p(z_i=k)\cdot p(x_i \mid z_i=k)}$$

- 가우시안 혼합모델의 경우 다음과 같이 정리된다.

$$r_{ik} = \frac{\theta_k N(x_i \mid \mu_k, \Sigma_k)}{\sum_{k=1}^{K} \theta_k N(x_i \mid \mu_k, \Sigma_k)}$$

⇒ 이식은 모수로부터 responsibility를 측정한다.

$(\theta_k, \mu_k, \Sigma_k) \Rightarrow r_{ik}$

※ r_{ik}는 i번째 data x_i가 카테고리 k에서 만들어 졌을 확률을 나타낸다.

- 앞에서구한 Log Likelihood 함수를 미분하여 0이 되도록하는 방정식을 만들면 다음과 같다.

$$0 = -\sum_{i=1}^{N} \frac{p(z_i=k)p(x_i \mid z_i=k)}{\sum_{k=1}^{K} p(z_i=k)\cdot p(x_i \mid z_i=k)} \Sigma_k (x_i - \mu_k)$$

- 이를 정리하면 $\sum_{i=1}^{N} r_{ik}(x_i - \mu_k) = 0$, $\mu_k = \frac{1}{N_k}\sum_{i=1}^{N} r_{ik} x_i$

$$\Rightarrow N_k = \sum_{i=1}^{N} r_{ik}$$

- k 카테고리에 속하는 데이터의 수와 비슷한 의미를 가진다. 즉 N_k는 k카테고리에 속하는 데이터의 영향력과 같은 의미이다.
- 마찬가지로 log likelihood를 Σ_k로 미분하여 0이 되도록하는 모수를 구하면 다음과 같다.

$$\Sigma_k = \frac{1}{N_k}\sum_{i=1}^{N} r_{ik}(x_i - \mu_k)(x_i - \mu_k)^T$$

- 마지막으로 log likelihood를 θ_k로 미분하여 0이되는 모수값을 구해야 하는데, 이때 카테고리 별로 모수가 가지는 제한조건을 위해 Lagrange Multiplier를 추가해야 한다.

$$\Pi + \lambda\left(\sum_{k=1}^{K}\theta_k - 1\right)$$

이를 θ_k로 미분하여 0이라는 값을 갖으면 $\theta_k = \frac{N_k}{N}$

- 이 세가지 식은 모두 responsibility로부터 모수를 추정하고 있다.

$$\gamma_{ik} \Rightarrow (\theta_k, \mu_k, \Sigma_k)$$

- 초기 방법은 모수와 responsibility를 번갈아 추정하며 정확도를 높여가는 방법.
  - "E" step 에서는 우리가 현재까지 알고있는 모수가 정확하다고 가정하고 이를 사용하여 각 Data가 어느 카테고리에 속하는지, 즉 responsibility를 추정.

$$(\theta_k, \mu_k, \Sigma_k) \Rightarrow \gamma_{ik}$$

  - "M" step 에서는 우리가 알고있는 responsibility가 정확하다고 가정하고 모수를 추정.

$$\gamma_{ik} \Rightarrow (\theta_k, \mu_k, \Sigma_k)$$

$\Rightarrow$ 이를 반복하면, 모수와 responsibility를 동시에 점진적으로 개선가능

Clustering

- 각각의 Data에 대해 responsibility를 알게되면 responsibility가 가장 큰 카테고리를 찾아내서 그 데이터가 어떤 카테고리에 속하는지 알 수 있다. 즉, clustering 가능.

$$k_i = \arg\max_k \gamma_{ik}$$