군집화 (2)

GMM / EM 알고리즘 / 군집화 / 모형 선택 방법론

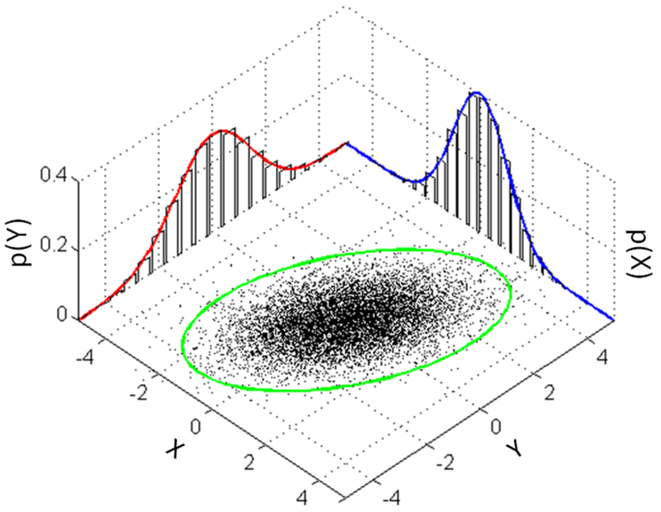
7기 교육부 홍승기

GMM (Gaussian Mixture Model)

GMM

다변량 정규분포 (revisited)

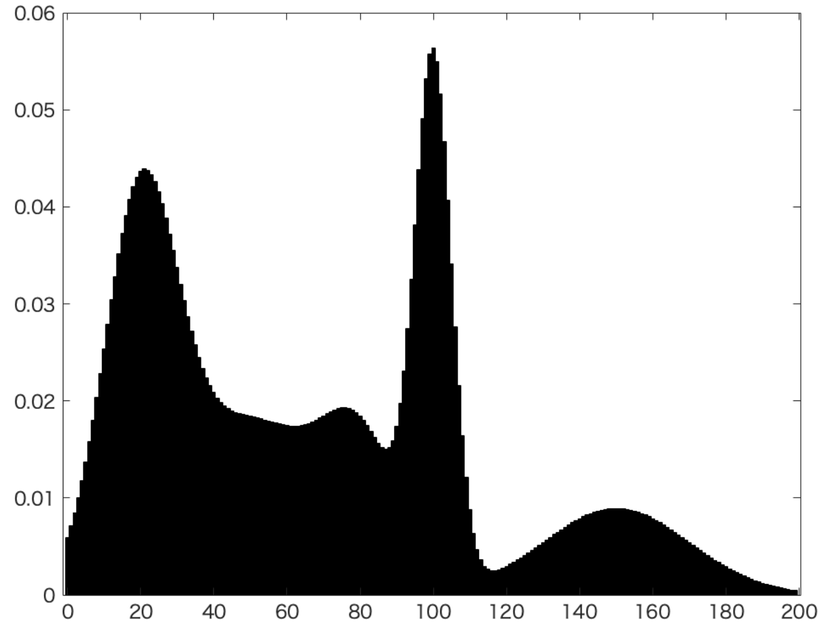
- $Z_1, \cdots, Z_n \overset{iid}{\sim} N(0,1)$ (iid: identically and independently distributed의 약자)이고 상수 행렬과 벡터 $A = (a_{ij})_{1 \le i,j \le n}, \mu = (\mu_1, \cdots, \mu_n)^T$ 에 대해 확률변수 $X = (X_1, \cdots, X_n)^T$ 를 다음과 같이 정의하자:
- $X = AZ + \mu,\ Z = (Z_1, \cdots, Z_n)^T$
- 이 때 A 의 역행렬이 존재한다면, X 의 확률밀도함수는 다음과 같다:
- $f(x) = |2\pi\Sigma|^{-1} \exp\left[\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right], \quad x \in \mathbb{R}^n, \Sigma = AA^T$
- 이러한 분포를 다변량 정규분포라 하며, 기호로는 다음과 같이 표기한다:
- $X \sim N(\mu, \Sigma)$
- 위의 정의에 따르면 $Z = (Z_1, \cdots, Z_n)^T$ 의 분포는 자연스럽게 $Z \sim N(0, I_n)$ 이 된다.
- 따라서 다변량 정규분포는 다음과 같이 정의할 수도 있다:
- $X \sim N(\mu, \Sigma) \Leftrightarrow X = AZ + \mu,\ Z \sim N(0, I_n),\ AA^T = \Sigma$



GMM

가우시안 혼합모형의 정의

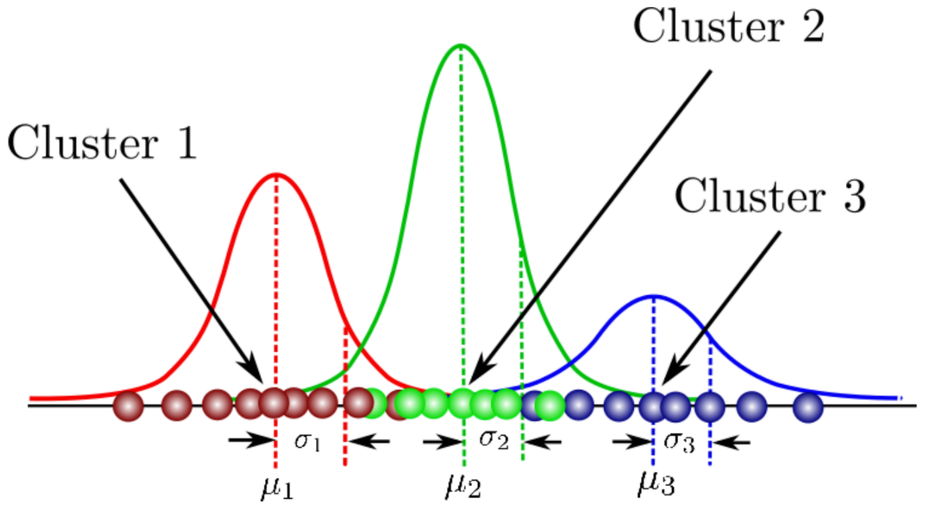
- $\pi_1, \cdots, \pi_k \geq 0$ 이고 $\pi_1 + \cdots + \pi_k = 1$ 라 하자.
- $f_i = f(\cdot \mid \mu_i, \Sigma_i) \sim N(\mu_i, \Sigma_i), \quad i = 1, \cdots, k$ 는 i번째 다변량 정규분포의 확률밀도함수.
- 이 때 가우시안 혼합모형(GMM)의 밀도함수 f 는 다음과 같이 정의된다:
- $f = \pi_1 f_1 + \cdots + \pi_k f_k$
- 즉, 다변량 정규분포의 밀도함수를 적절히 합친 것을 GMM이라 한다.



GMM

GMM을 이용한 군집화

- 오른쪽 그림처럼 3개의 정규분포가 겹쳐져 있는 GMM을 생각해보자.
- 빨간색 데이터들은 첫 번째 정규분포의 밀도값이 높은 곳 위주로 분포하고
- 녹색, 파란색 데이터도 마찬가지로 각 색깔의 정규분포 밀도값이 높은 곳 위주로 분포한다.
- GMM은 데이터들이 타원형으로 겹쳐진 형태로 분포할 때, 이를 확률적으로 모델링하여 구분을 하는데에 특화가 되어있다.
- K-means 는 특정 중심을 기준으로 같은 거리에 있는 것들을 묶는다는 점에서, GMM의 특수 버전이라 생각할 수 있다. (뒤에서 자세하게 서술)



GMM

최대가능도 추정법 (revisited)

- 고정된 관측 결과 $x = (x_1, \cdots, x_n)^T$ 에 대해 가능도함수(likelihood function)는 다음과 같이 정의한다:
- $L(\theta; x) = \prod_{i=1}^{n} f(x_i; \theta),\ \theta \in \Omega$
- 가능도함수는 모수가 θ 일 때 주어진 관측결과가 나올 가능성을 나타내준다고 볼 수 있다. 이러한 관점에서 가능도함수가 더 높은 값을 가지는 모수를 찾아내는 것이 좋다. 따라서 관측 결과가 x 일 때 가능도함수가 최대가 되는 모수를 찾는 것이 목적이 될 수 있다.
- 최대가능도 추정량(maximum likelihood estimator : MLE) 의 정의는 다음과 같다:
- $X_1, \cdots, X_n \overset{iid}{\sim} f(x; \theta), \theta \in \Omega$ 일 때, $\hat{\theta}^{MLE} = \arg\max_{\theta \in \Omega} L(\theta, X) = \arg\max_{\theta \in \Omega} \prod_{i=1}^{n} f(X_i, \theta)$
- 일반적으로 최대가능도 추정량을 구하기 위해 미분을 이용하기 때문에, 가능도함수에 로그를 취해 이용하는 경우가 많다. 이를 로그가능도함수(log-likelihood function)이라 하고, 다음과 같이 정의한다:
- $l(\theta; x) = \log L(\theta; x) = \sum_{i=1}^{n} \log f(x_i; \theta),\ \theta \in \Omega$

GMM

최대가능도 추정법의 한계

- 그러나 GMM의 경우에는 기존의 최대가능도 추정법을 적용하기엔 여러 한계가 있다.
- 예를 들어, 근사적으로 미분을 이용하여 파라미터를 추정하는 방법인 gradient descent 를 이용하는 것을 고려할 수 있다.
- 그러나 π_i 또한 추정해야하는 파라미터로, 미분을 통해서 이를 추정하는 것은 매우 어렵거나 불가능하다.
  ($\because \partial f / \partial \pi_i$ 는 π_i에 대해 상수함수.)
- 따라서 π_i 를 추정하기 위해선 새로운 시각의 접근이 필요하다.

EM 알고리즘

EM 알고리즘

잠재변수가 있는 상황 상정

- 다음 기호와 논리는 Dempster, et.al (1977) 를 참고하였다.
- 현재 우리가 관측한 데이터를 Y_0, 그리고 존재하지만 관측할 수 없는(또는 미처 관측하지 못한) 데이터를 Y_m 이라 하고, 이 둘을 합친 전체 데이터를 $Y_c = (Y_0, Y_m)$ 이라 하자.
- Y_m 를 보통 잠재변수(latent variable) 또는 결측변수(missing data)이라 한다.
- 이때 전체 파라미터를 θ 라 하고, $f(Y_c \mid \theta)$ 를 전체 데이터의 밀도함수라고 하자.
- $g(Y_0 \mid \theta) = \int f(Y_c \mid \theta) dY_m$ 은 관측 데이터 Y_0 의 주변밀도함수라 하자.
- $k(Y_m \mid Y_0, \theta) = \dfrac{f(Y_c \mid \theta)}{g(Y_0 \mid \theta)}$ 는 잠재변수 Y_m 의 조건부밀도함수라 하자.
- 사실 $g(Y_0 \mid \theta)$ 는 관측 데이터 Y_0에 의한 θ 의 가능도함수라고 생각할 수 있다.
- 이 때 우리의 목표는 가능도함수 $g(Y_0 \mid \theta)$ 또는 로그가능도함수 $l(\theta) = \log g(Y_0 \mid \theta)$ 를 극대화하는 것이다.

EM 알고리즘

EM 알고리즘의 정의

- EM 알고리즘의 목표는 $p+1$ 번째 스텝에서 $l(\theta) \geq l(\theta^{(p)})$ 를 만족하는 θ 를 찾고, 이러한 θ 를 $\theta^{(p+1)}$ 이라 한 후, 이 과정을 수렴할 때까지 하는 것이다.
- 일단 함수 $Q(\theta \mid \theta^{(p)})$ 를 다음과 같이 정의하자:
- $Q(\theta \mid \theta^{(p)}) := \int \log f(Y_c \mid \theta) \cdot k(Y_m \mid Y_0, \theta) dY_m = \mathbb{E}_{Y_m \mid Y_0, \theta}(\log f(Y_c \mid \theta))$
- 즉, $Q(\theta \mid \theta^{(p)})$ 는 주어진 Y_0, θ 에 대해 $\log f(Y_c \mid \theta)$ 의 조건부 기대값을 뜻한다.
- **EM(Expectation-Maximization)** 알고리즘은 이에 따라 다음과 같이 정의한다:
- 초기값 $\theta^{(0)}$ 에서 시작한다.
- Expectation-step : p 번째 스텝에서 $Q(\theta \mid \theta^{(p)}) = \mathbb{E}_{Y_m \mid Y_0, \theta}(\log f(Y_c \mid \theta))$ 를 계산한다.
- Maximization-step : $\theta^{(p+1)} = \arg\max_{\theta} Q(\theta \mid \theta^{(p)})$ 를 구한다.
- E-step 과 M-step 을 $Q(\theta^{(p+1)} \mid \theta^{(p)}) \approx Q(\theta^{(p)} \mid \theta^{(p)})$ 일 때까지 반복한다.

EM 알고리즘

젠센부등식과 베이즈 정리 (revisited)

- EM 알고리즘은 전적으로 $Q(\theta|\theta^{(p)})$에 의존한 알고리즘이다.
- 그렇다면 $Q(\theta|\theta^{(p)})$ 는 왜 나타났으며, 이러한 알고리즘이 가능도함수 극대화를 달성할 수 있을까?
- $Q(\theta|\theta^{(p)})$ 의 등장을 살펴보기 전에, 젠센부등식과 베이즈 정리를 기억해보자.
- (**젠센 부등식; Jensen's inequality**) : 수직선 위의 구간 I 에서의 값을 갖는 확률변수 X 의 기댓값이 존재하면, 구간 I 에서 볼록한 함수 ϕ 에 대해 다음이 성립한다:
- $\phi(E[X]) \leq E[\phi(X)]$
- (**베이즈 정리; Bayes' Theorem**) : 사건 열 $\{A_n\}_{n\geq 1}$ 이 표본공간 S 를 공통부분이 없게 분할하고 $P(A_i) > 0$ 일 때, $P(B) > 0$ 이면 다음 비례식이 성립한다:
- $P(A_j|B) \propto P(B|A_j)P(A_j) \quad (j = 1,2,\cdots)$
- 이제 $Q(\theta|\theta^{(p)})$ 가 어디서 나왔는지 알아보자.

EM 알고리즘

함수 Q 의 유도

- $l(\theta) = \log g(Y_0 \mid \theta) = \log \mathbb{E}_{Y_m \mid Y_0, \theta^{(p)}} \left[\dfrac{f(Y_c \mid \theta)}{k(Y_m \mid Y_0, \theta^{(p)})} \right] \geq \mathbb{E}_{Y_m \mid Y_0, \theta^{(p)}} \log \left[\dfrac{f(Y_c \mid \theta)}{k(Y_m \mid Y_0, \theta^{(p)})} \right] =: h(\theta \mid \theta^{(p)})$
- 위의 부등식은 젠센 부등식에 의해 성립한다. ($\because -\log x,\ x > 0$ 는 볼록함수)
- 이 때 마찬가지로 젠센 부등식에 의해 $l(\theta^{(p)}) \geq \mathbb{E}_{Y_m \mid Y_0, \theta^{(p)}} \log \left[\dfrac{f(Y_c \mid \theta^{(p)})}{k(Y_m \mid Y_0, \theta^{(p)})} \right]$.
- 한 편, 베이즈 정리에 의해 $f(Y_c \mid \theta^{(p)}) \propto k(Y_m \mid Y_0, \theta^{(p)}) g(Y_0 \mid \theta^{(p)})$ 가 성립한다.
- 따라서 $l(\theta^{(p)})$ 의 조건부 기댓값 부분의 안쪽은 상수항이 되어, $l(\theta^{(p)}) = \mathbb{E}_{Y_m \mid Y_0, \theta^{(p)}} \log \left[\dfrac{f(Y_c \mid \theta^{(p)})}{k(Y_m \mid Y_0, \theta^{(p)})} \right]$ 가 성립한다.
- 이는 곧 $h(\theta^{(p)} \mid \theta^{(p)}) = l(\theta^{(p)})$ 를 의미하며, 따라서 $h(\theta \mid \theta^{(p)}) \geq h(\theta^{(p)} \mid \theta^{(p)})$ 를 만족하는 θ 를 찾으면 $l(\theta) \geq l(\theta^{(p)})$ 가 성립한다.
- 이 때 $h(\theta \mid \theta^{(p)}) = Q(\theta \mid \theta^{(p)}) - \mathbb{E}_{Y_m \mid Y_0, \theta^{(p)}} \log[k(Y_m \mid Y_0, \theta^{(p)})]$ 이므로, h 를 최대로 하는 것은 Q 를 최대로 하는 것과 같다.
- 즉, Q 가 h보다 계산하기 쉽기 때문에 Q 를 이용하여 EM 알고리즘을 구성한 것이다. 한 편, h 를 구성하는 두 번째 항은 엔트로피이다.

EM 알고리즘

이론적 성질들과 장점

- **(단조성 정리; Monotonicity Theorem, Dempster, et al.(1977))** : 모든 EM 알고리즘은 반복될수록 $l(\theta \mid Y_0)$ 가 단조증가한다. 즉, 임의의 p 에 대해 $l(\theta^{(p+1)} \mid Y_0) \geq l(\theta^{(p)} \mid Y_0)$ 이며, 등호는 $Q(\theta^{(p+1)} \mid \theta^{(p)}) = Q(\theta^{(p)} \mid \theta^{(p)})$ 일 때 성립한다.

- **(수렴성 정리; Convergence Theorem, Wu(1983))** : 가능도함수 $l(\theta \mid Y_0)$ 가 유계상한(bounded above)을 가질 때, EM 알고리즘은 극댓값(local maximum)으로 수렴한다. 특히, 가능도함수가 미분가능하다면 $l(\theta^*) = 0$ 인 어떤 θ^*로 수렴한다.

- 위 정리들은 EM 알고리즘이 최대가능도 추정법을 효과적으로 수행함을 보장한다.

- 특히, 만약 가능도함수가 오목함수라면 유일한 최대가능도 추정량을 근사적으로 구해낼 수 있다.

- EM 알고리즘은 잠재변수 또는 결측값이 있을 때를 상정하여 만들어진 최대가능도 기반 알고리즘이므로
  결측 데이터가 있을 때, 잠재변수가 있다고 가정할 때, 또는 관측할 수 없는 변수가 존재할 때 유용하게 쓸 수 있다.

- 결측 데이터는 의학 데이터, 특히 X-ray나 MRI와 같은 데이터를 다룰 때 주로 나타나는데, 이 때 유용하게 쓰인다.

- 잠재변수를 상정하는 방법론은 클러스터링에서 주로 이용되며, 한 편으로는 재무학에서 포트폴리오의 분산 최적화를 통한 위험관리나 가치평가에서 주로 이용된다.

- 사회과학의 인과추론 영역에서도 EM 알고리즘의 아이디어를 주로 이용한다. 특히 계량경제학에서 외생성, 내생성을 이용한 모델링을 할 때 EM알고리즘 아이디어를 이용하여 모델 최적화를 하며, 이러한 아이디어는 최근 머신러닝 학계에서도 활발히 이용되고 있다.

EM 알고리즘

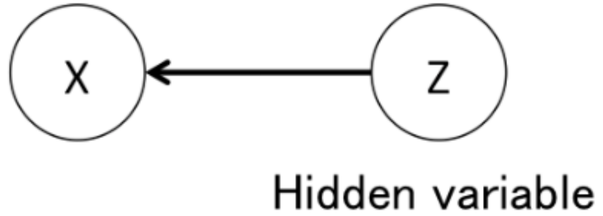
단점

- 하지만 단조적으로 증가하는 만큼 알고리즘의 수렴 속도는 느리다.
- 가능도함수가 오목함수가 아니라 여러 봉우리를 가진 모양이라면, 잘못된 값으로 수렴할 여지가 있다.
- 이는 곧 초기값 설정이 추정량의 성능을 좌우한다는 단점으로도 해석된다.
- 다행인 점은, 이러한 한계점들을 어느정도 극복한 방법론들이 존재한다.

EM 알고리즘

GMM의 잠재변수를 이용한 재해석

- GMM 을 다시 기억해보자 : $f = \pi_1 f_1 + \cdots + \pi_k f_k, \;\; f_i \sim N(\mu_i, \Sigma_i), \quad i = 1, \cdots, k$
- 이 때 잠재변수 $Z = (Z_1, \cdots, Z_k)^T \sim Multi(1, (\pi_1, \cdots, \pi_k)^T)$ 라 하자. 즉, Z 는 k 차원 다항분포를 따르며, 모든 원소의 합이 1인 확률변수이다.
- 그리고 확률변수 X 가 다음을 만족한다고 하자:
- $X|_{Z_i=1} \sim N(\mu_i, \Sigma_i), \quad i = 1, \cdots, k$
- 즉, 잠재변수 Z 의 i 번째 원소가 1이면 확률변수 X는 i 번째 정규분포를 따르는 것이다.



- 이 때 전확률공식 (머신러닝을 위한 통계학 p.5 참조)에 의해 다음이 성립한다:
- $X \sim f \Leftrightarrow f(x) = \sum_{i=1}^{k} f_i(x) P(Z_i = 1) = \sum_{i=1}^{k} \pi_i f_i(x)$
- 이는 다음과 같이 해석할 수 있다: 주어진 데이터 X 가 GMM을 따른다면, 이 데이터가 i 번째 정규분포에서 생성되었을 확률은 π_i 이다.
- 즉, X 가 i 번째 정규분포에서 생성되었는지는 관측할 수 없지만(Z_i 가 1인지 확인할 수 없지만), 그러할 확률을 모델링한 것이 GMM이라고 볼 수 있는 것이다.

EM 알고리즘

GMM에의 적용: 사전 준비

- 이제 본격적으로 GMM에 EM 알고리즘을 적용해보자.
- 관측 데이터가 $x_1, \cdots, x_n \overset{iid}{\sim} f$ 라 하고, 각각에 대응되는 잠재변수가 $z_1, \cdots, z_n \overset{iid}{\sim} Multi(1,(\pi_1, \cdots, \pi_k)^T)$ 라 하자.
- 그리고 앞의 슬라이드와 마찬가지로 관측 데이터 x_i 가 다음을 만족한다고 하자:
- $x_i \,|\, z_{ij} = 1 \sim N(\mu_j, \Sigma_j), \quad i = 1,\cdots,n,\ j = 1,\cdots,k$
- 이 때 θ 가 모든 파라미터를 의미한다고 하면, 관측 데이터 $x_1, \cdots, x_n$ 에 대한 가능도함수 $L_o(\theta) = g(x \,|\, \theta)$ 는 다음과 같다:
- $$L_o(\theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} \pi_j f_j(x_i)$$
- 또한 전체 데이터 $x_1, \cdots, x_n$ 와 $z_1, \cdots, z_n$ 에 대한 가능도함수 $L_c(\theta) = f(x, z \,|\, \theta)$ 는 다음과 같다:
- $$L_c(\theta) = \prod_{i=1}^{n} \prod_{j=1}^{k} (\pi_j f_j(x_i))^{z_{ij}} = \exp\left[\sum_{i=1}^{n} \sum_{j=1}^{k} z_{ij} \left(\log \pi_j - \frac{1}{2} \log |\Sigma_j| - \frac{1}{2}(x_i - \mu_j)^T \Sigma^{-1} (x_i - \mu_j) - \frac{d}{2} \log(2\pi) \right) \right]$$

EM 알고리즘

GMM에의 적용: E-step

- 우선, 현재 $\theta^{(p)}$가 주어진 상황일 때 베이즈 정리에 의해 다음 사실이 성립한다:
- $$P_{j,i}^{(p)} := \mathbb{P}(z_{ij} = 1 \mid X_i = x_i, \theta^{(p)}) = \frac{\pi_j^{(p)} f(x_i \mid \mu_j^{(p)}, \Sigma_j^{(p)})}{\sum_{j=1}^{k} \pi_j^{(p)} f(x_i \mid \mu_j^{(p)}, \Sigma_j^{(p)})}$$
- 따라서 E-step 작업인 Q 를 계산해보면
- $$Q(\theta \mid \theta^{(p)}) = \mathbb{E}_{Z|X,\theta^{(p)}} \log L_c(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{k} P_{j,i}^{(p)} \left(\log \pi_j - \frac{1}{2} \log |\Sigma_j| - \frac{1}{2} (x_i - \mu_j)^T \Sigma^{-1} (x_i - \mu_j) - \frac{d}{2} \log(2\pi) \right)$$

EM 알고리즘

GMM에의 적용: M-step

- 이제 M-step 작업인 $Q(\theta|\theta^{(p)})$ 를 최대로 만드는 $\theta^{(p+1)}$ 을 구해보면 다음과 같다:
- $\pi_j^{(p+1)} = \frac{1}{n}\sum_{i=1}^{n} P_{j,i}^{(p)}, \quad j = 1, \cdots, k$
- $\mu_j^{(p+1)} = \frac{\sum_{i=1}^{n} P_{j,i}^{(p)} x_i}{\sum_{i=1}^{n} P_{j,i}^{(p)}}, \quad \Sigma_j^{(p+1)} = \frac{\sum_{i=1}^{n} P_{j,i}^{(p)} (x_i - \mu_j^{(p+1)})(x_i - \mu_j^{(p+1)})^T}{\sum_{i=1}^{n} P_{j,i}^{(p)}}$
- (참고: 위 계산을 하는 데에 필요한 연산은 다음과 같다.)
- 적당한 행렬 A, B에 있어서 $\frac{\partial}{\partial A} tr(BA) = B^T, \quad \frac{\partial}{\partial A} \log|A| = (A^{-1})^T$

EM 알고리즘

GMM에의 적용: 정리

- 정리하자면, GMM의 파라미터를 추정하는 EM 알고리즘은 다음과 같다:
- 1. 파라미터 초기값 $\theta^{(0)}$ 을 설정한다.
- 2. $p = 0$ 부터 다음을 반복한다:
- 2-1. $P_{j,i}^{(p)} = \dfrac{\pi_j^{(p)} f(x_i \mid \mu_j^{(p)}, \Sigma_j^{(p)})}{\sum_{j=1}^{k} \pi_j^{(p)} f(x_i \mid \mu_j^{(p)}, \Sigma_j^{(p)})}$ 를 계산한다.
- 2-2. $\pi_j^{(p+1)} = \dfrac{1}{n} \sum_{i=1}^{n} P_{j,i}^{(p)}, \quad j = 1, \cdots, k$ 를 계산한다.
- 2-3. $\mu_j^{(p+1)} = \dfrac{\sum_{i=1}^{n} P_{j,i}^{(p)} x_i}{\sum_{i=1}^{n} P_{j,i}^{(p)}}, \quad \Sigma_j^{(p+1)} = \dfrac{\sum_{i=1}^{n} P_{j,i}^{(p)} (x_i - \mu_j^{(p+1)})(x_i - \mu_j^{(p+1)})^T}{\sum_{i=1}^{n} P_{j,i}^{(p)}}$ 를 계산한다.
- 위 값들의 차이가 충분히 작아지면 반복을 종료한다.

군집화

군집화

GMM을 이용한 군집화

- k 개의 정규분포를 합친 GMM을 이용하여, k 개의 군집을 만드는 상황을 생각해보자.
- k 개의 군집으로 이뤄져있다고 가정하면, 주어진 데이터 X 가 속할 확률이 가장 높은 군집에 X 를 할당하는 것이 합당할 것이다. 이를 수식으로 쓰면 다음과 같다:
- X 가 속한 군집을 $Y \in \{1, \cdots, k\}$라 하면 $Y = \arg \max_{j=1,\cdots,k} \mathbb{P}(Z_j = 1 \mid X)$
- 한 편, 위의 확률값은 앞선 EM 알고리즘에서 $P_{j,i}$라는 값으로 구하였다.
- 따라서, 관측된 데이터 $x_1, \cdots, x_n$가 각각 속하는 군집 $y_1, \cdots, y_n$ 은 다음과 같이 구해진다:
- $y_i = \arg \max_{j=1,\cdots,k} P_{j,i} = \arg \max_{j=1,\cdots,k} \mathbb{P}(z_{ij} = 1 \mid X_i = x_i, \theta)$

군집화

Python 에서의 GMM

class `sklearn.mixture.` **`GaussianMixture`**(*n_components=1, *, covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0, verbose_interval=10*) [source]

- Python에선 sklearn.mixture.GaussianMixture 를 이용하여 GMM을 진행할 수 있다.
- n_components : 앞선 슬라이드에서 k 를 의미한다. 즉, 혼합하는 정규분포의 개수를 지정한다.
- covariance_type : 앞선 슬라이드에서 Σ_j 들의 형태를 지정해준다.
  -'full' 이면 일반적인 symmetric matrix. 즉 별다른 조건없는 Σ_j이다.
  - 'tied' 이면 $\Sigma_1 = \cdots = \Sigma_k$, 즉 모든 정규분포의 분산이 같은 형태임을 가정한다.
  - 'diag' 이면 $\Sigma_j = diag(\sigma_l^2)$, 즉 모든 정규분포의 분산행렬이 대각행렬임을 가정한다.
  - 'spherical' 이면 $\Sigma_j = \sigma_j^2 I$, 즉 모든 정규분포의 분산행렬이 항등행렬의 실수배 형태임을 가정한다.
- 따라서, 'spherical' < 'diag' < 'tied' < 'full' 순서로 추정해야하는 파라미터의 개수가 많아지며, 이는 곧 EM 알고리즘의 수렴속도와 연관이 있다.
- 또한, 각 가정들은 군집의 기하학적 형태를 결정한다. 결론부터 말하자면, 'full'의 경우엔 k 개의 타원, 'tied'는 k 개의 같은 모양의 타원, 'diag'이면 k 개의 각 축에 평행한 형태의 타원, 'spherical'이면 k 개의 원의 형태로 군집을 이루게 된다.

class `sklearn.mixture.` **`GaussianMixture`**(*n_components=1, *, covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0, verbose_interval=10*) [source]

- tol: 어느정도를 수렴했다고 볼지 정한다. 즉, $Q(\theta^{(p+1)} \mid \theta^{(p)}) - Q(\theta^{(p)} \mid \theta^{(p)}) < tol$ 을 이용한다.
- reg_covar : Σ_j 의 역행렬이 존재하지 않을 때 이용하는 $\Sigma_j + \lambda I$ 의 λ 값을 정하는 변수이다.
- max_iter : EM 알고리즘의 반복 횟수를 정한다.
- n_init : 초기값의 개수를 지정한다. 확실하지 않지만 AIC나 BIC (뒤에서 설명 예정)를 이용하여 최종 파라미터를 선택하는 것으로 보인다.
- init_params : 초기값을 설정하는 방법을 정한다. ‘kmeans’ 는 k-means 클러스터링을 이용하여 초기값을 설정한다. ‘random’ 은 말 그대로 랜덤하게 설정한다. 참고로 ‘random’ 보다는 ‘kmeans’가 더 좋은 초기값을 일반적으로 주지만, k-means 클러스터링 특성상 초기값에 민감하게 정해지기 때문에, k-means의 초기값 불안정성이 GMM에도 전달되는 단점이 있다. Python에는 구현이 되어있지 않지만, 일반적으로 k-means 보다는 hierarchical clustering을 이용하는 것이 더 선호된다.
- weights_init, means_init, precisions_init : 순서대로 π_j, μ_j, Σ_j의 초기값을 직접 선택해주는 변수이다.

EM 알고리즘

GMM 실습

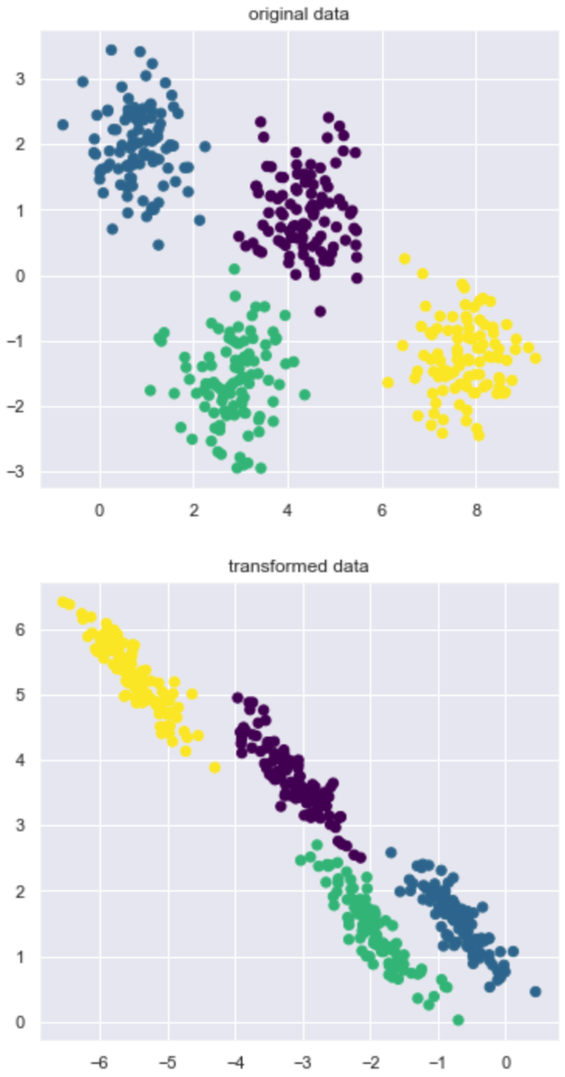
- 시뮬레이션 데이터 생성

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
import numpy as np
from sklearn.mixture import GaussianMixture

from sklearn.datasets.samples_generator import make_blobs
X, y_true = make_blobs(n_samples=400, centers=4,
                       cluster_std=0.60, random_state=0)
X = X[:, ::-1]

rng = np.random.RandomState(13)
X_trans = np.dot(X, rng.randn(2, 2))

figure, axes = plt.subplots(2, figsize=(5,10))
axes[0].scatter(X[:, 0], X[:, 1], c=y_true, s=40, cmap='viridis')
axes[0].set_title("original data")
axes[1].scatter(X_trans[:, 0], X_trans[:, 1], c=y_true, s=40, cmap='viridis')
axes[1].set_title("transformed data")
```



EM 알고리즘

GMM 실습

- 정규분포별 contour plot 생성하는 함수코드 & 원본 데이터와 변형 데이터에 GMM 적용

```
from matplotlib.patches import Ellipse

def draw_ellipse(position, covariance, ax = None, **kwargs):
    ax = ax or plt.gca()

    if covariance.shape == (2, 2):
        U, s, Vt = np.linalg.svd(covariance)
        angle = np.degrees(np.arctan2(U[1, 0], U[0, 0]))
        width, height = 2 * np.sqrt(s)
    else:
        angle = 0
        width, height = 2 * np.sqrt(covariance)

    for nsig in range(1, 4):
        ax.add_patch(Ellipse(position, nsig * width, nsig * height,
                             angle, **kwargs))

def plot_gmm(gmm, X, label = True, ax = None):
    ax = ax or plt.gca()
    labels = gmm.fit(X).predict(X)
    if label:
        ax.scatter(X[:, 0], X[:, 1], c=labels, s=40, cmap='viridis', zorder=2)
    else:
        ax.scatter(X[:, 0], X[:, 1], s=40, zorder=2)
    ax.axis('equal')

    w_factor = 0.2 / gmm.weights_.max()
    if gmm.covariances_.shape == (2, 2):
        gmm.covariances_ = np.repeat(gmm.covariances_[np.newaxis, :, :], 4, axis=0)
    for pos, covar, w in zip(gmm.means_, gmm.covariances_, gmm.weights_):
        if covar.size == 1:
            n_features = gmm.means_.shape[1]
            covar = np.repeat(covar, n_features)
        draw_ellipse(pos, covar, ax = ax, alpha=w * w_factor)
```

```
gmm = GaussianMixture(n_components=4)
gmm_trans = GaussianMixture(n_components=4)
labels = gmm.fit(X).predict(X)
labels_trans = gmm_trans.fit(X_trans).predict(X_trans)

figure, axes = plt.subplots(1, 2, figsize = (12,6))
axes[0].scatter(X[:, 0], X[:, 1], c = labels, s = 40, cmap = 'viridis')
axes[0].set_title("original data, fitted")
axes[1].scatter(X_trans[:, 0], X_trans[:, 1], c = labels_trans, s = 40, cmap = 'viridis')
axes[1].set_title("transformed data, fitted")
```

EM 알고리즘

GMM 실습 : 성능평가

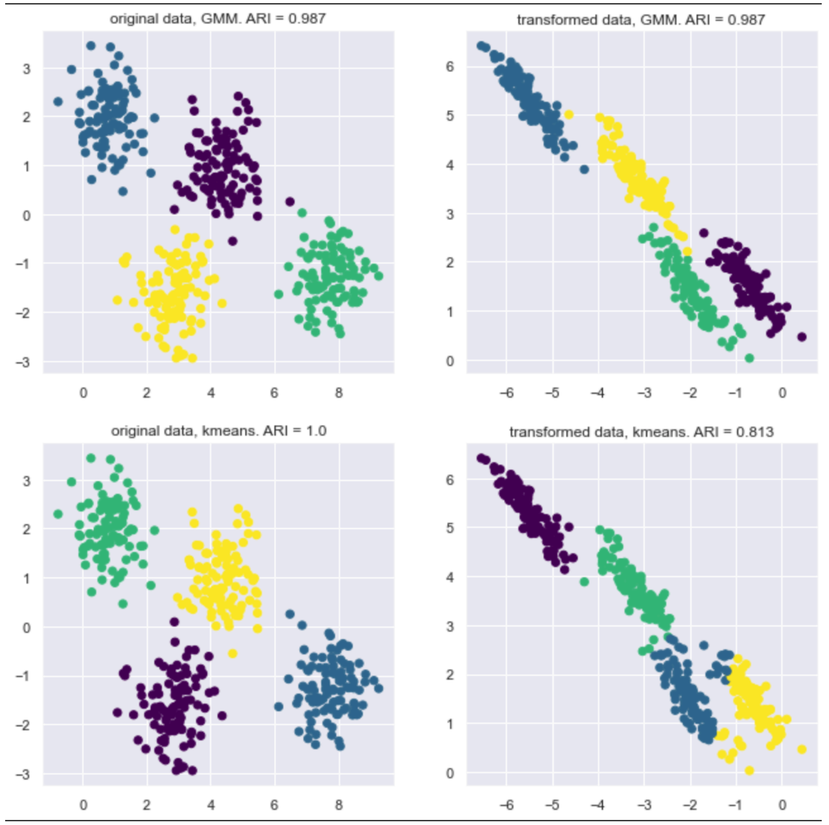
- 타겟값을 아는 경우엔 일반적으로 adjusted random index (ARI) 를 이용하여 군집성능을 평가한다. 1에 가까울수록 군집화가 잘 되었음을 의미한다.

- 구체적인 정의는 다음을 참조: https://en.wikipedia.org/wiki/Rand_index

```
from sklearn.cluster import KMeans
from sklearn.metrics import adjusted_rand_score

kmeans = KMeans(n_clusters=4)
kmeans_trans = KMeans(n_clusters=4)
kmeans_labels = kmeans.fit(X).predict(X)
kmeans_labels_trans = kmeans_trans.fit(X_trans).predict(X_trans)

figure, axes = plt.subplots(2, 2, figsize = (11,11))
axes[0, 0].scatter(X[:, 0], X[:, 1], c = labels, s = 40, cmap = 'viridis')
axes[0, 0].set_title("original data, GMM. ARI = " +
                     str(round(adjusted_rand_score(y_true, labels), 3)))
axes[0, 1].scatter(X_trans[:, 0], X_trans[:, 1], c = labels_trans, s = 40, cmap = 'viridis')
axes[0, 1].set_title("transformed data, GMM. ARI = " +
                     str(round(adjusted_rand_score(y_true, labels_trans), 3)))
axes[1, 0].scatter(X[:, 0], X[:, 1], c = kmeans_labels, s = 40, cmap = 'viridis')
axes[1, 0].set_title("original data, kmeans. ARI = " +
                     str(round(adjusted_rand_score(y_true, kmeans_labels), 3)))
axes[1, 1].scatter(X_trans[:, 0], X_trans[:, 1], c = kmeans_labels_trans, s = 40, cmap = 'viridis')
axes[1, 1].set_title("transformed data, kmeans. ARI = " +
                     str(round(adjusted_rand_score(y_true, kmeans_labels_trans), 3)))
```

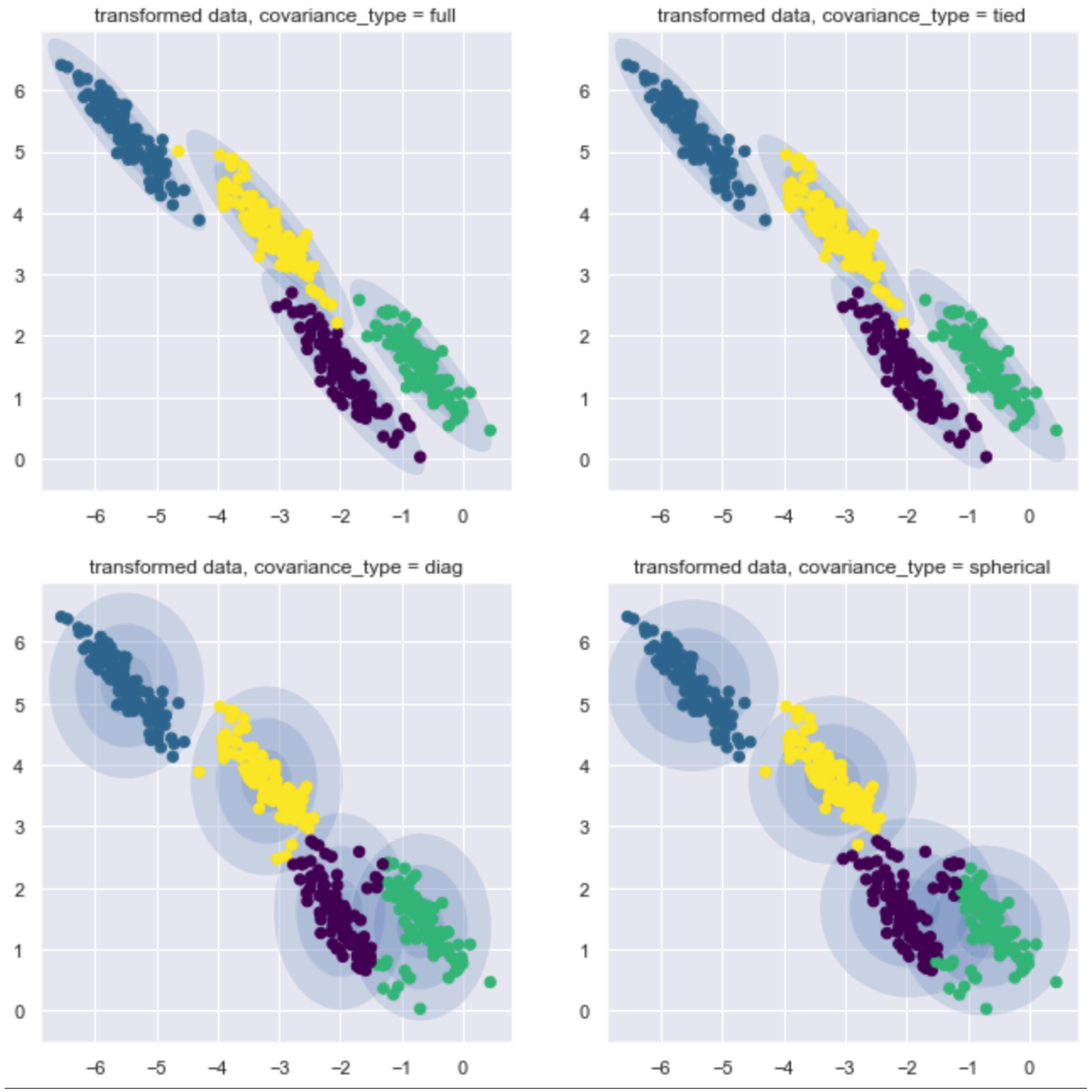


EM 알고리즘

GMM 실습 : 공분산 행렬 조건

- 공분산 행렬에 주어진 조건에 따라
  군집화의 모양이 달라짐을 확인할 수 있다.

```
figure, axes = plt.subplots(2, 2, figsize = (11,11))
for k, cov_type in enumerate(['full', 'tied', 'diag', 'spherical']):
    j = (k) % 2
    i = (k-j) // 2 % 2
    gmm = GaussianMixture(n_components=4, covariance_type=cov_type, random_state=0)
    plot_gmm(gmm, X_trans, ax = axes[i, j])
    axes[i, j].set_title("transformed data, covariance_type = " + cov_type)
```



군집화

k-means와 GMM

- k-means 알고리즘의 경우, 다음을 최소화 하는 것을 목표로 한다(13주차 자료 참고):
- $$C = \sum_{i=1}^{n}\sum_{j=1}^{k} r_{ij}\|x_i - \mu_j\|^2,\ r_{ij} = \begin{cases} 1, & j = \arg\min_l \|x_i - \mu_l\|^2 \\ 0, & \text{otherwise} \end{cases}, \mu_j = \sum_{i=1}^{n} r_{ij}x_i / \sum_{i=1}^{n} r_{ij}$$
- 한 편, GMM에 적용된 EM 알고리즘에서 다음과 같은 가정을 추가해보자:
- $$\Sigma_j = I,\ \mathbb{P}(z_{ij} = 1 \,|\, X_i = x_i, \theta) = \begin{cases} 1, & j = \arg\max_l f(x_i; \mu_l) \\ 0, & \text{otherwise} \end{cases}, \quad \forall j = 1, \cdots, k$$
- 즉, GMM의 모든 공분산 행렬이 I 이고, 관측 데이터 x_i 가 j 번째 군집에 속할 확률이 0 또는 1이며 각 정규분포의 밀도값에 의해서만 결정된다고 하자.
- 이러한 가정 하에, GMM은 k-means 알고리즘과 동치가 된다. 즉, 슬라이드 17장의 $P_{j,i}$ 가 위의 r_{ij} 와 동치가 된다.
- 따라서, k-means 알고리즘은 EM 알고리즘의 특수 케이스로 볼 수 있으며, 이런 맥락에서 k-means 알고리즘을 hard-thresholded EM algorithm 이라 부르기도 한다.

군집화

k-means와 GMM

- 이미지 특징 클러스터링에 k-means가 자주 쓰이듯, GMM 또한 해당 문제에 좋은 성능을 보인다.
- k-means 의 일반화인 만큼, 단점을 공유한다.
- EM 알고리즘의 단점인 초기값 선정 문제, 지역적 최적값 문제를 가지고 있다.
- 차원의 저주 문제 또한 가지고 있다.
- 그러나 k-means 와 달리, GMM은 모델 선택 방법론이 존재한다.
- 또한 k-means는 특이한 데이터 분포의 경우엔 성능이 매우 안좋지만 GMM은 그러한 문제를 회피할 수 있다.
- 모델 선택 방법론을 통해 확인해보자.

모형 선택 방법론

모형 선택 방법론

일반화 오차(Generalization Error)

- 일반화 오차 또는 표본 외 오차(out-of-sample error) 는 알고리즘이 마주한 적 없는 데이터를 얼마나 정확하게 예측하는가에 대한 척도이다.
- 예를 들어, 회귀 모형 $Y = m(x) + \epsilon$ 에 대해 $\hat{m}(x)$ 를 알고리즘을 통해 얻은 fit 이라 하자.
- 훈련 데이터를 $S = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ 이라 하면, 훈련 평균오차제곱합(training MSE) T 는 다음과 같다.
- $$T = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{m}(x_i))^2$$
- 앞서 cross-validation에서도 살펴봤듯, T 는 모델의 예측 성능을 잘 나타내지 않는다.
- 이와 같은 맥락에서 일반화 오차 G 는 보통 다음과 같이 정의한다:
- $G = \mathbb{E}_{X,Y}(Y - \hat{m}(X))^2$ (X, Y 모두 확률변수로 취급하며 $\hat{m}$ 과 X 는 독립.)
- 즉, 새로운 데이터에 대한 기댓값을 취하는 것으로 정의한다.

모형 선택 방법론

AIC 정의

- 위와 같이 일반화 오차를 정의하면, 일반적으로 다음과 같이 표현이 된다.
- “일반화 오차(generalization error)” = E(“훈련 오차(training error)”) + “패널티”
- (회귀분석의 Mallow’s Cp, Leave-one-out CV, Generalized CV 등등 모두 위와 같이 표현된다.)
- 한 편, 앞서 예시를 든 제곱 손실함수와 같이 negative log-likelihood 또한 손실함수의 역할을 할 수 있으며, 이를 이용하여 일반화 오차를 정의할 수 있다.
- 이 때 Akaike 는 다음을 제안하였다:
- (**Akaike 정보기준; Akaike information criterion**) : L_s 를 모델 S 의 로그가능도함수, $dim(L_s)$ 를 S 의 추정 파라미터 개수라 하면 Akaike 정보기준은 다음과 같이 정의한다.
- $AIC = -2 \cdot L_s + 2dim(L_s)$
- 만약 훈련 데이터의 개수가 충분히 많다면, AIC는 로그가능도 기반의 일반화 오차와 거의 같다.
- 따라서, 여러가지 모델들 중에서 AIC가 가장 작은 모델을 선택하는 것이 바람직하다.

모형 선택 방법론

AIC의 장단점

- AIC의 최대 장점은 훈련 데이터셋 만으로 충분히 모델 성능을 나타내준다는 것이다.
- 훈련, 검증의 단계로 나눠 여러 번 검증하는 cross-validation 과 가장 큰 차이점이다.
- 또한 훈련 데이터가 많은 경우에는 이론적 정확도가 보장되어 있다.
- 하지만 훈련 데이터가 많다는 것이 어느정도의 크기를 의미하는지는 알려져있지 않다.
- 훈련 데이터가 많을수록 복잡한 모델을 선택할 가능성이 높다. 이는 뒤의 BIC와 비교했을 때 바람직하지 못한 면이다.
- 따라서, 데이터가 많더라도 AIC가 좋은 모델을 선택하지 못하는 경우가 존재한다.
- 또한 모델의 파라미터가 제대로 추정되지 않았을 경우(예를 들어 MLE가 아닌 경우)엔 성능이 좋지 못하다.

모형 선택 방법론

BIC 정의

- 한 편, Bayesian 의 시각에선 모델에 대한 사전분포와 파라미터 θ 에 대한 분포가 존재한다.
- 이러한 시각에선, 로그가능도의 역할을 사후분포가 대체하게 된다.
- 따라서 손실함수를 AIC와 다르게 사후분포로 대체한다면, 다음이 유도된다.
- (**Bayesian 정보기준; Bayesian information criterion**) : 훈련 데이터의 개수를 n 이라 하면 BIC는 다음과 같이 정의한다.
- $BIC = -2L_s + n \cdot dim(L_s)$
- 즉, BIC는 AIC의 패널티 항에서 2 대신 훈련 데이터 개수를 대입한 것과 같다.
- 또한 AIC와 마찬가지로 훈련 데이터가 충분히 많다면, BIC는 사후분포 기반의 일반화 오차와 거의 같다.
- 따라서, 여러가지 모델들 중에서 BIC가 가장 작은 모델을 선택하는 것이 바람직하다.

모형 선택 방법론

BIC의 장단점

- 만약 비교하고자 하는 모델들 중에 true model이 있다면, BIC는 점근적으로 true model을 선택함이 알려져있다.
- 즉, 훈련 데이터의 수가 클수록 AIC와 비교하여 더 좋은 모델을 선택할 가능성이 높다.
- 하지만 반대로, 훈련 데이터가 충분히 많지 않다면 AIC와 비교할 때 너무 단순한 모델을 선택하는 경향이 있다.
- 또한 모델의 파라미터 개수가 많은 경우엔 BIC의 선택 성능이 좋지 못함이 알려져 있다.
- 결론적으로, AIC, BIC 모두 훈련 데이터만을 이용한다는 점에서 장단점이 있기 때문에 다양한 기준을 이용하여 모델을 선택하는 것이 필요하다.

모형 선택 방법론

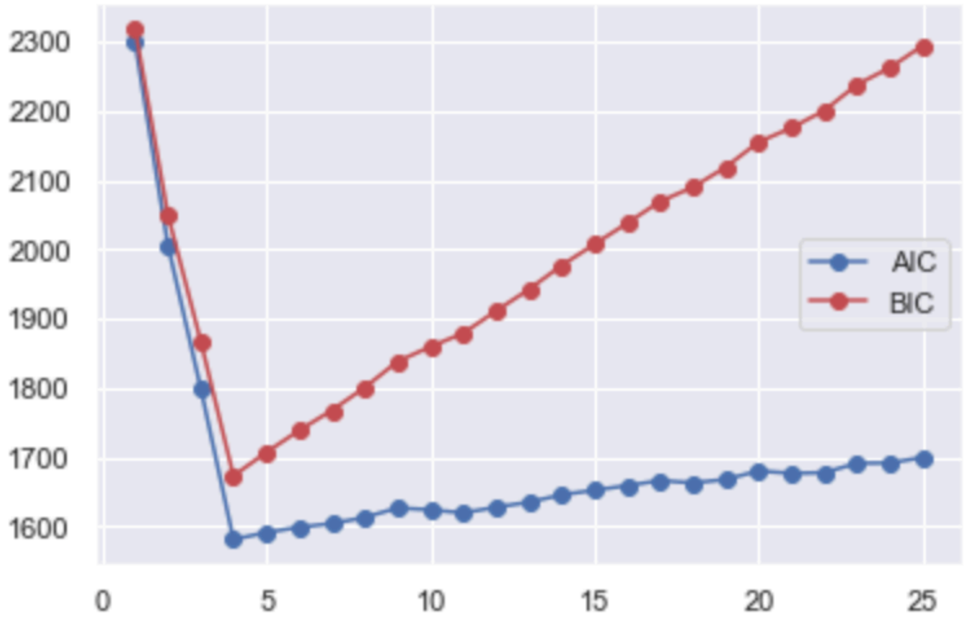
AIC, BIC를 이용한 최적 GMM 선택

- 밑의 그림에서 봤을 때, AIC와 BIC 모두 4개의 component를 가진 모형을 최적 모형으로 선택하였다.

- BIC 가 AIC에 비해 더 가파른 기울기로 증가하는 것 또한 볼 수 있다.

```
AIC = []
BIC = []
max_components = 25
for comp in range(max_components):
    gmm = GaussianMixture(n_components=comp+1, random_state=0).fit(X_trans)
    AIC.append(gmm.aic(X_trans))
    BIC.append(gmm.bic(X_trans))

plt.plot(range(1, 26), AIC, marker = 'o', label = 'AIC', color = 'b')
plt.plot(range(1, 26), BIC, marker = 'o', label = 'BIC', color = 'r')
plt.legend()
```

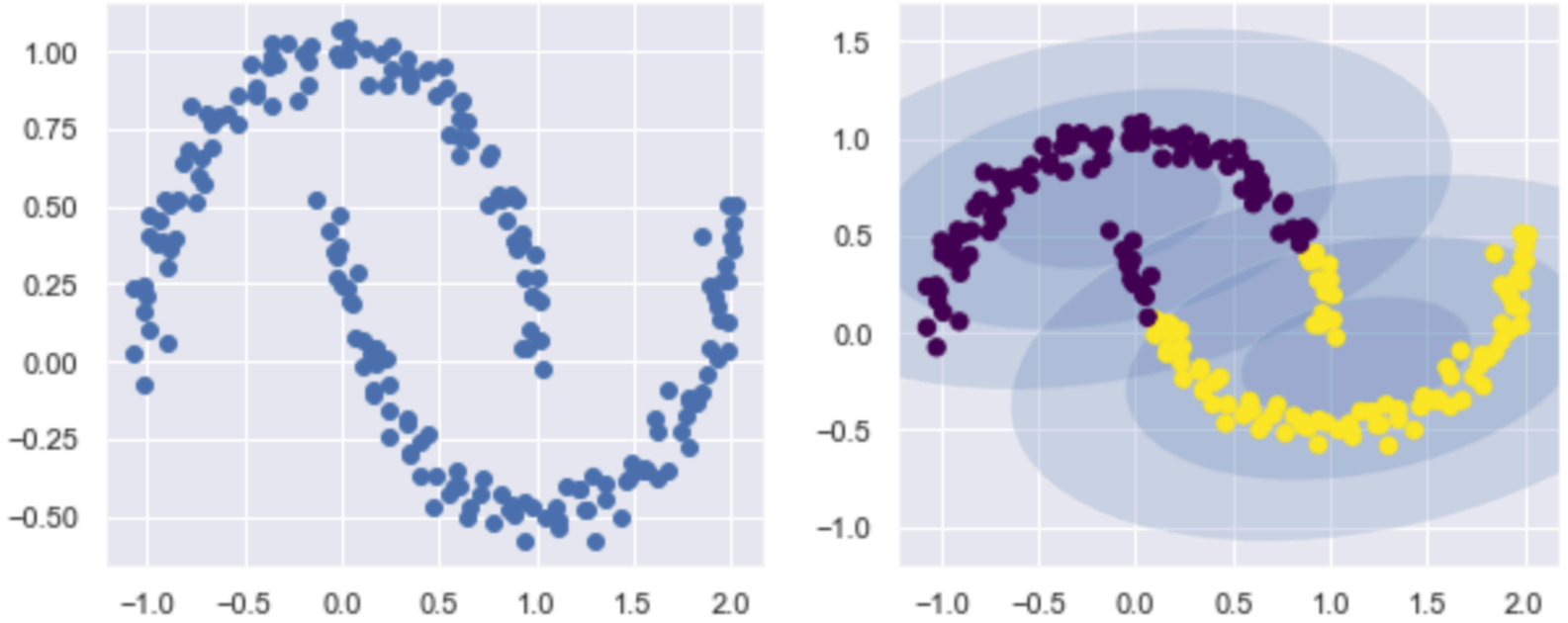


모형 선택 방법론

AIC, BIC를 이용한 최적 GMM 선택 (2)

```
from sklearn.datasets import make_moons
Xmoon, ymoon = make_moons(200, noise=.05, random_state=0)

gmm2 = GaussianMixture(n_components=2, covariance_type='full', random_state=0)
figure, axes = plt.subplots(1, 2, figsize = (10,5))
axes[0].scatter(Xmoon[:, 0], Xmoon[:, 1]);
plot_gmm(gmm2, Xmoon, ax = axes[1])
```



모형 선택 방법론

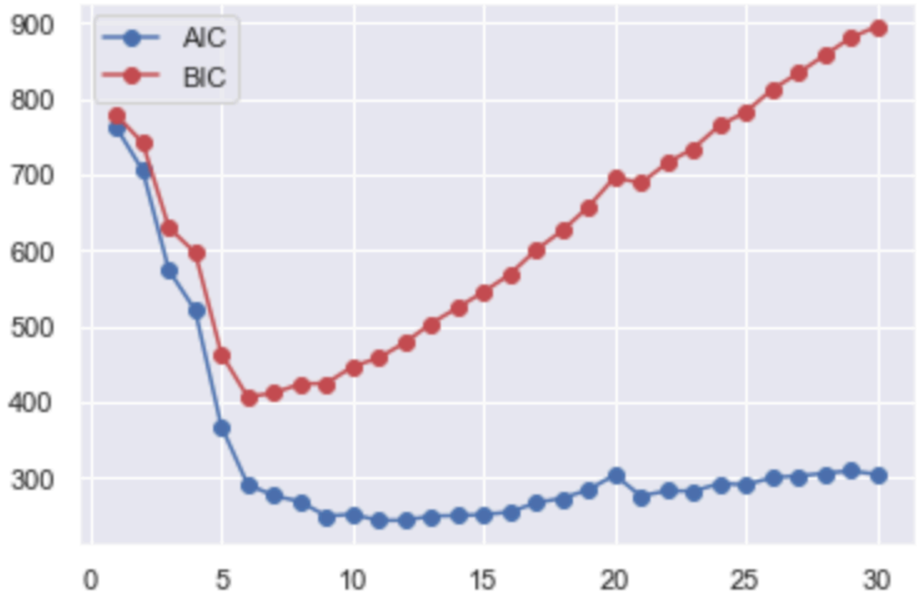
AIC, BIC를 이용한 최적 GMM 선택 (3)

- 마찬가지로 BIC가 AIC보다 간단한 모델을 선택하는 경향을 보인다.

```
AIC = []
BIC = []
max_components = 30
for comp in range(max_components):
    gmm = GaussianMixture(n_components=comp+1, random_state=0).fit(Xmoon)
    AIC.append(gmm.aic(Xmoon))
    BIC.append(gmm.bic(Xmoon))

(AIC.index(min(AIC)), BIC.index(min(BIC)))
```

```
(10, 5)
```



모형 선택 방법론

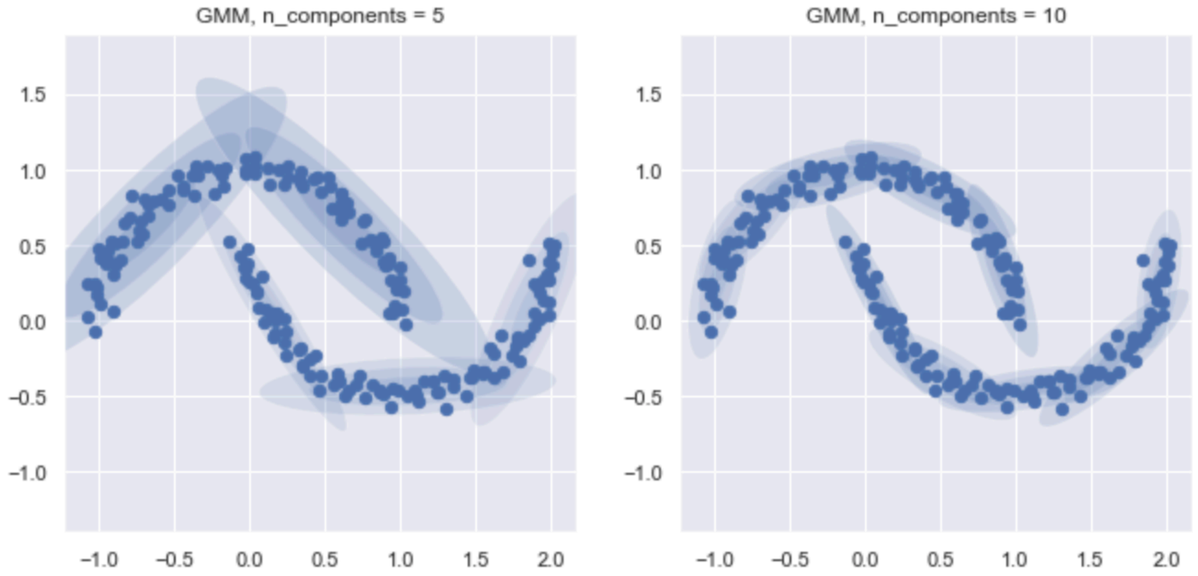
AIC, BIC를 이용한 최적 GMM 선택 (4)

- 만약 겹치는 타원들을 하나의 클러스터로 본다면, GMM으로 오른쪽과 같이 특이한 경우에도 군집화를 진행할 수 있다.

- 이러한 특성으로 미루어볼 때, GMM은 특이한 형태의 데이터를 묘사하는 데에 좋은 성능을 보인다.

- 한 편, GMM은 확률구조를 묘사하므로 역으로 그럴듯한 데이터를 생성해내는 작업에서도 좋은 성능을 보인다. 이렇듯 데이터에 대한 확률구조를 묘사하는 모델을 generative model 이라 한다.

```
gmm5 = GaussianMixture(n_components=5, random_state=0)
gmm10 = GaussianMixture(n_components=10, random_state=0
figure, axes = plt.subplots(1, 2, figsize = (11,5))
plot_gmm(gmm5, Xmoon, label = False, ax = axes[0])
axes[0].set_title("GMM, n_components = 5")
plot_gmm(gmm10, Xmoon, label = False, ax = axes[1])
axes[1].set_title("GMM, n_components = 10")
```



참고자료

- [1] 응용통계 강의자료, 임요한, et al.(2017), 서울대학교 다변량통계, 생물통계, 공간통계 연구실
- [2] “Finite Mixture Models”, G. Mclachlan and D. Peel(2001), John Wiley & Sons Inc
- [3] “Maximum Likelihood from Incomplete Data via the EM Algorithm”, A. Dempster, et al.(1977), Journal of the Royal Statistical Society, Series B
- [4] "On the Convergence Properties of the EM Algorithm”, J. Wu(1983), Annals of Statistics
- [5] “Chapter 13: The Multivariate Gaussian”, UC Berkeley, URL: https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter13.pdf
- [6] “The EM Algorithm”, Martin Haugh(2015), IEOR E4570, Columbia Univ.
- [7] “A new look at the statistical model identification”, H. Akaike(1974), IEEE Transactions on Automatic Control
- [8] “Estimating the dimension of a model”, G. E. Schwarz (1978), Annals of Statistics
- [9] “The Elements of Statistical Learning”, T. Hastie, et al.(2017), Springer
- [10] “Python Data Science Handbook”, J. VanderPlas(2016), O'Reilly Media, Inc