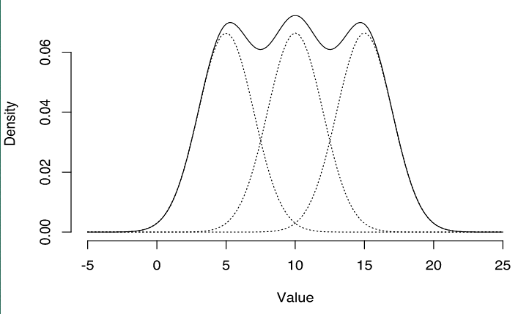
**Mixture of Gaussian의 EM Algorithm 적용**

바이오 인공지능융합학과 2021192861 김가연

**1. Introduction**

**1.1 Mixture of Gaussian**

Gaussian Mixture Model, GMM은 Gaussian 분포가 여러 개 혼합된 clustering 알고리즘이다.

왼쪽 그림처럼 현실에 존재하는 복잡한 형태의 확률 분포를 여러 개의 Gaussian 분포를 혼합하여 표현하자는 것이 GMM의 기본 아이디어다.

Figure 1 Gaussian Mixture Model

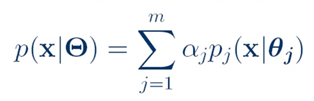


Figure 2 GMM의 x가 발생할 확률

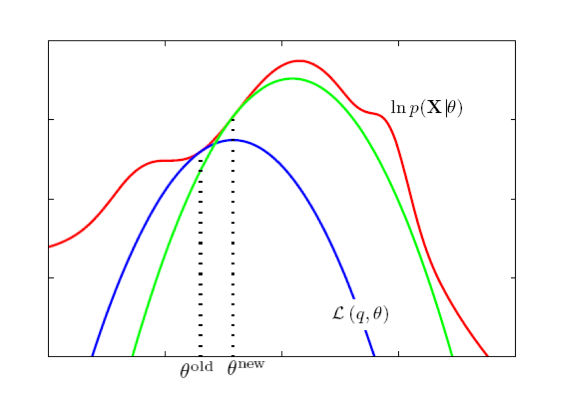
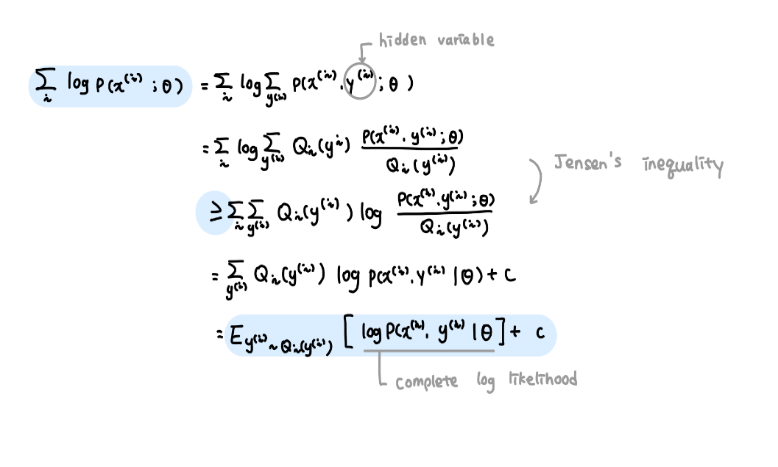
GMM의 x가 발생할 확률은 여러 Gaussian probability density function의 합으로 표현된다. 위 식에서는 총 m개의 components가 있다. 여기서 는 mixing probability로 j번째 component가 선택될 확률을 나타내고 는 0~1 사이의 값을 가지고 모든 의 합은 1이 된다. 는 component density로 j번째 sample의 likelihood이다.

**1.2 EM Algorithm (Expectation-Maximization Algorithm)**

1.2.1 EM 알고리즘의 개념

EM알고리즘은 관측되지 않은 잠재변수(latent variables)에 의존하는 확률 모델에서 maximum likelihood를 갖는 최적의 매개변수 를 찾는 반복적인 알고리즘이다. EM 알고리즘은 2가지 step, E-step, M-step을 반복하면서 이루어지는데 먼저 E-step은 주어진 임의의 파라미터 초기값에서 likelihood와 최대한 근사한 likelihood값을 계산한다. 그리고 M-step은 E-step에서 계산된 likelihood를 최대화하는 새로운 파라미터 를 계산한다.

1.2.2 EM 알고리즘 프로세스

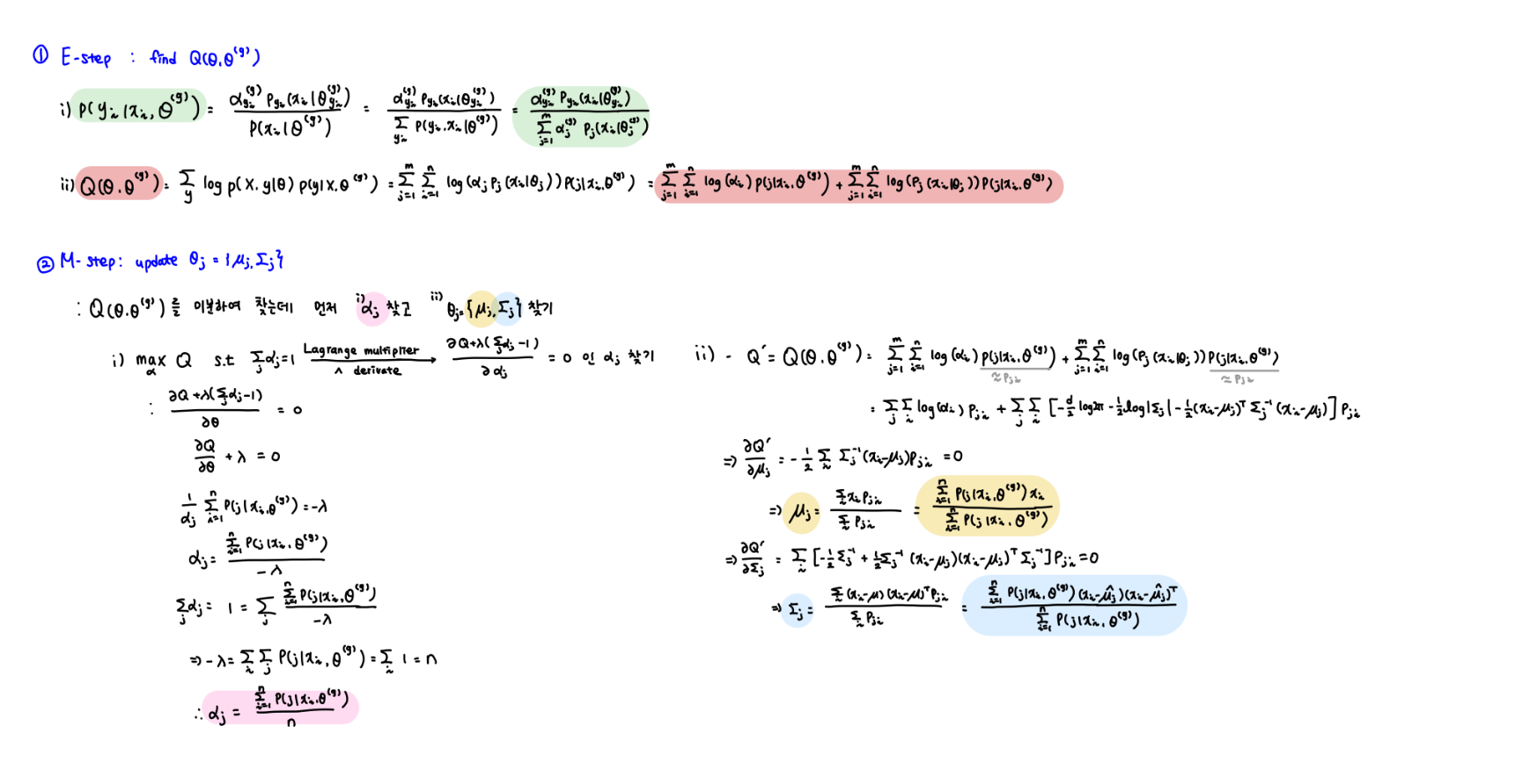


왼쪽 위의 그림처럼 우리가 maximization하고싶은 likelihood 는 ‘Jensen’s inequality’에 의해 complete log likelihood  **이상으로** 정의된다. 따라서 complete log likelihood를 높이게 되면 이를 lower bound로 가지는maximization하고싶은 likelihood를 높일 수 있을 것이다. 즉 오른쪽 위 그림에서 볼 수 있듯이 EM 알고리즘은 실제 cost function 의 tight lower bound 인 를 최대화하는 것이다.

위에서는 두 식의 관계가 이상으로 정의되었는데 만약 이라면 두 식은 같아지게 된다. 즉 EM 알고리즘의 E-step은 인 를 찾고,

M-step에서 로 를 업데이트 하는 것이다.

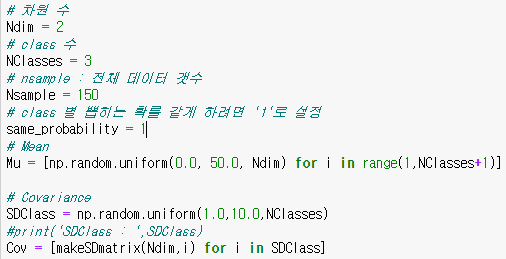
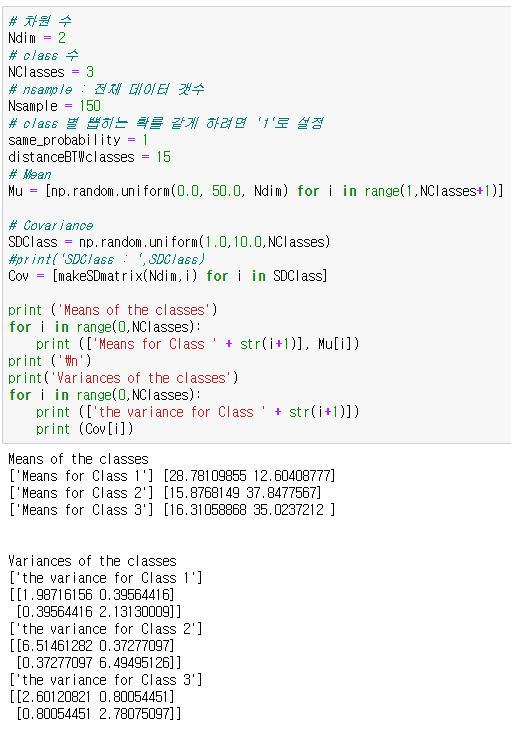
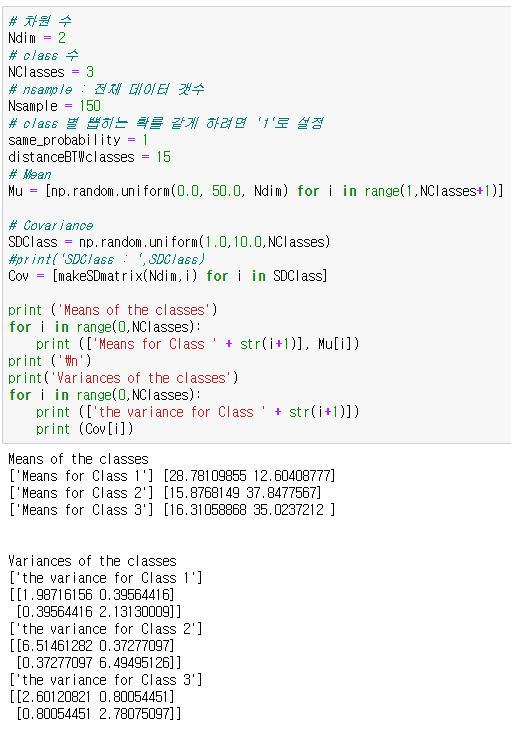
**1.3 MoG(Mixture of Gaussian)의 EM Algorithm 적용**

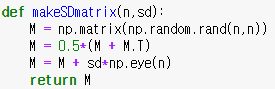
****

**2. Method**

**2.1 Data**

2.1.1 평균과 분산

 ****



2차원 Gaussian Mixture Model의 data를 뽑기 위해 class수 (NClasses)와 총 데이터 sample 수(Nsample)을 설정한다. 각 component에서 sample이 뽑히는 확률을 같게 하려면 same\_probability를 ‘1’로 설정한다.

3개의 components를 사용한다고 하면 평균 Mu는 0에서 50 사이에서 random하게 3개의 class마다 평균 2개, 총 6개의 값을 뽑는다. 분산은 2차원이므로 공분산의 형태일 것이고 마찬가지로 1에서 10 사이에서 하나의 값을 random하게 뽑고 그 값을 makeSDmatrix 함수에 넣어서 symmetric positive definite 행렬로 변환해 covariance 형태의 분산을 만든다.

2.1.2 각 component에서 sample을 뽑을 확률 과 sample 개수 뽑기

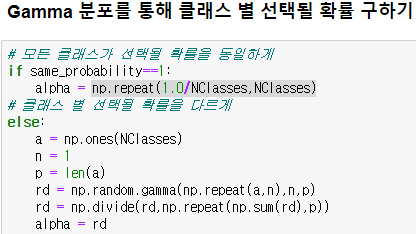


Figure 4 확률이 다를 경우 예시



Figure 5 확률이 같을 경우 예시

Figure 3 component별 선택될 확률

모든 component에서 다른 확률로 sample을 뽑는다면 gamma분포를 통해 합이 1이도록 확률을 class 개수만큼 분배하고, 같은 확률로 sample을 뽑는다면 1을 각 class 개수로 나누어 사용한다.



Figure 6 확률에 따라 component별로 sample 개수선택

random.multinomial을 통해 alpha의 확률을 사용해 총 Nsample개의 데이터를 클래스 별로 나누어 준다.

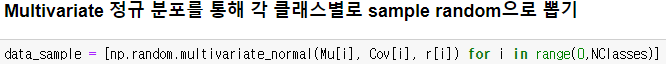
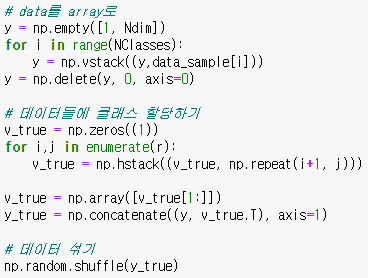


Figure 7 multivariate 정규 분포를 이용하여 각 클래스별로 sample을 random으로 뽑기

random.multivariate\_normal을 이용해 위에서 선택된 평균, 분산, 데이터 개수에 따라 2차원 데이터를 class수 만큼 뽑는다. Class가 3개이고 각 class별로 44, 50, 56개의 데이터가 있다면 이런 식의 data가 각 클래스별로 44, 50, 56 개 존재한다.

2.1.3 EM algorithm test를 위해 데이터 섞기

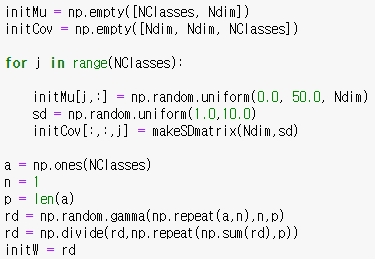
위에서 뽑은 data\_sample을 (150,2) 형태의 array로 만든다.

다음으로 정확도 계산을 위해 data\_sample array에 열을 하나 추가하여 class 번호도 붙여준다. 예를 들어 는 1번 class에 속하는 (x,y) 2차원 데이터를 의미한다.

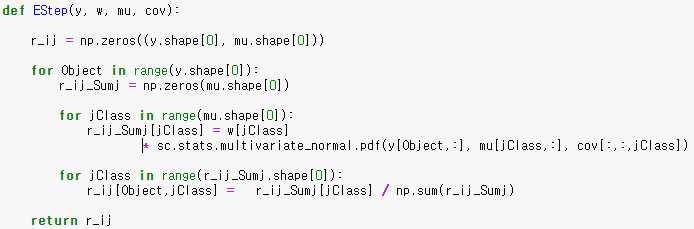
마지막으로 EM algorithm 실행 전 데이터를 random한 순서로 섞어서 데이터를 저장해 준다.

**2.2 EM Algorithm**

2.2.1 초기 평균, 분산, 확률 설정하기

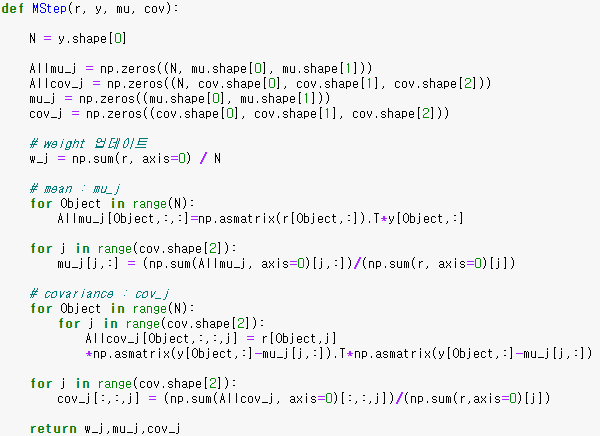
초기 평균(initMu), 분산(initCov), 각 component에서 sample이 뽑힐 확률(initW)을 위해 앞에서 data를 뽑을 때와 동일하게 평균과 분산은 크기만큼 random하게 만들어주고 확률은 Gamma분포를 통해 합이 1이 되도록 확률 값을 나누어 준다.

2.2.2 E-step



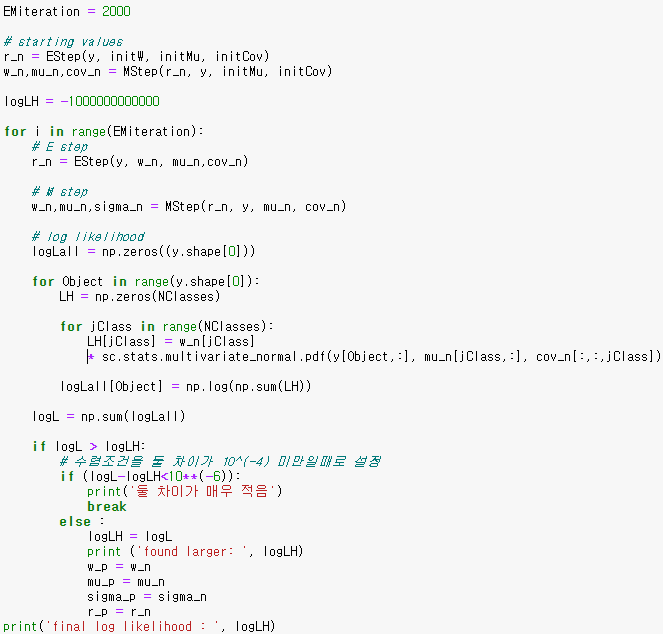
계산을 위해서 먼저 인 r\_ij\_Sumj[jClass]를 보면 가 w[jClass]이고 현재 우리는 multivariate\_normal을 통해 sampling을 하였으므로 는 multivariate\_normal.pdf를 따를 것이다. 따라서 두 식의 곱인 가 분포가 되고 이것을 분자로, class 개수만큼 j에 대해 모두 더한 np.sum(r\_ij\_Sumj)를 분모에 주면 우리가 구할 가 정의된다.

2.2.3 M-step



1.3에서 보았듯이 **, , 로 업데이트**한다.

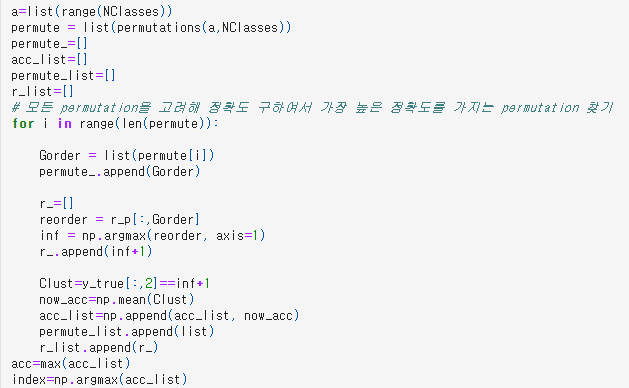
2.2.4 E-step, M-step 반복적으로 적용



EM Algorithm은 E-step과 M-step을 반복적으로 실행하는데 stop condition이 필요하다. 총 2가지 ①현재 log likelihood값과 직전의 log likelihood값의 차이가 설정된 보다 작을 때, 혹은 ② iteration의 수를 정해주어 stop condition을 정의한다. 본 실험에서는 ① 현재 log likelihood값과 직전의 log likelihood값의 차이가 설정된 보다 작을 때에 EM 알고리즘이 종료된다. 두 log likelihood값의 차이가 보다 작지 않지만 log likelihood값이 더 큰 것이 발견되었다면 log likelihood, , 결과 r\_p를 모두 현재 값으로 업데이트 해준다.

2.2.5 정확도 계산

EM 알고리즘을 통해 clustering을 하게 되면 원래의 class 번호에 따라 분류되는 것이 아니기 때문에 r\_p의 결과 (150\*NClasses)에서 NClasses인 열을 순열으로 모든 경우의 수를 따져 보아야 한다. 예를 들어 5개의 class를 사용한다면 결과 r\_p는 150\*5의 형태이고 이때 열 5개를 모든 순열 5\*4\*3\*2\*1=120가지의 경우의 수를 모두 따져 보아야한다.



먼저 class의 개수만큼 0에서 NClasses-1까지의 list를 만들고 이 숫자들로 가능한 모든 순열을 만든다. 총 순열의 수만큼 반복하며 해당 permutation을 이용해 r\_p를 재정렬(reorder)하고 np.argmax를 통해 가장 높은 확률을 가지는 것을 찾는다(inf). 후에 inf에 1을 더하고 원래의 class가 같은지 비교하여(Clust) np.mean을 통해 accuracy를 찾아낸다.

예를 들어 class가 5개라면 a={0,1,2,3,4}가 되고 permute={(0,1,2,3,4), (0,1,2,4,3), … }이며 permute는 총 120개의 원소가 있다. 이중 하나의 순열 {0,1,3,4,2} 이용해 r\_p를 재정렬하여 reoder을 만들고 np.argmax(reorder)을 하여 재정렬한 후 확률이 가장 높은 값을 가지는 index를 150개 모두 찾아낸 후 값에 1을 더해주면 이것이 우리가 예측한 class번호 inf+1이 된다. 마지막으로 원래의 답인 y\_true[:,2]와 inf+1를 비교하여 Clust를 만들고 np.mean(Clust)를 통해 정확도를 계산한다.

for문을 이용해 가능한 모든 순열의 정확도를 acc\_list에 저장해 두고 list에서 가장 큰 값이 우리가 사용할 정확도이고 그때 사용된 순열이 가장 바르게 재정렬한 경우이다. 정확도를 acc로 그때의 순열을 index로 저장한다.

**3. Result**

비교를 위해 총 150개의 데이터를 class 3개, 5개, 7개로 나누어 실험을 진행하였다.

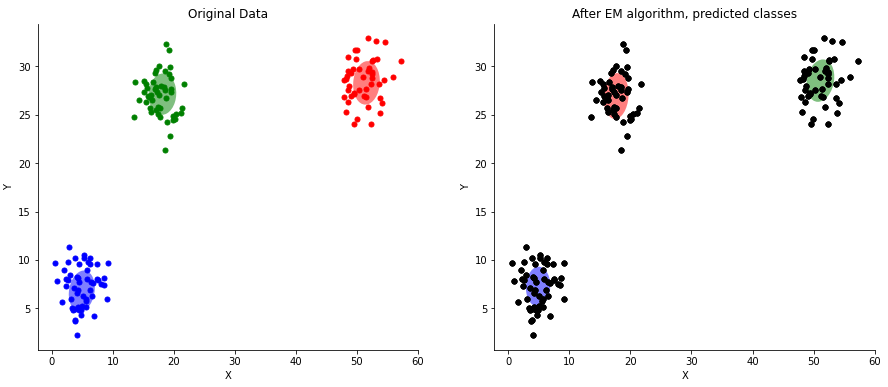
**3.1 class 3개**

3.1.1 각 component 별 선택될 확률 가 동일할 때 (=0.333)



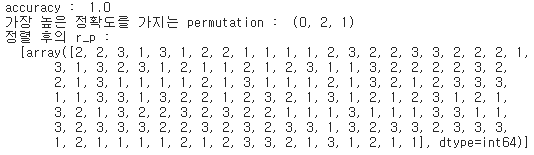
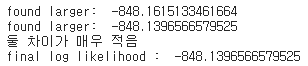
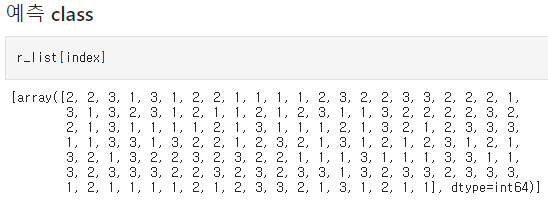
각 component별로 선택될 확률 가 동일하므로 각각의 component들은 모두 33.33%의 확률로 데이터가 선택될 것이다. 위 확률을 바탕으로 데이터 150개는 각각 class에 59, 49, 47 개로 나누어 졌다.

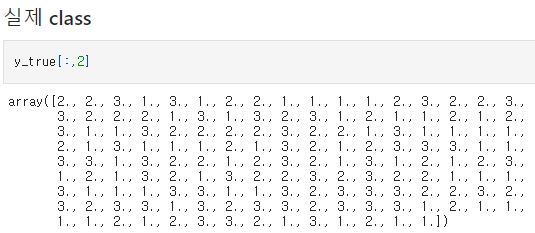
① 원래 데이터 분포와 clustering 이후의 분포 비교



실험 데이터를 확인해 보면 왼쪽 위의 그림과 같이 분포하는 것을 확인할 수 있다. 다음으로 EM 알고리즘을 반복적으로 실행하여 clustering 한 결과 오른쪽 위의 그림과 같이 데이터가 잘 clustering 된 것을 확인할 수 있다.

② log likelihood값과 정확도 비교





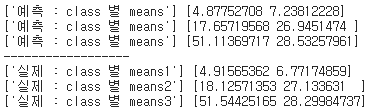
EM 알고리즘은 이전과 현재의 log likelihood값의 차이가 보다 작을 때 종료된다. 본 실험에서 2번의 iteration 후 최종 log likelihood값은 -848.1396이고 100%의 정확도를 가지며 그때 사용된 순열은 (0, 2, 1)이다. 실제 class번호와 예측된 class 번호를 비교해 보면 모든 데이터가 동일한 것을 확인할 수 있다.

③ 각 component가 선택될 확률 , 평균, 분산 비교

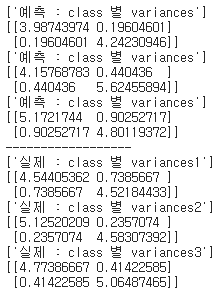
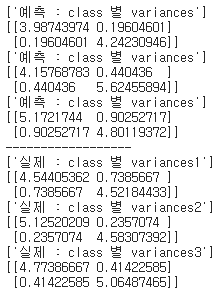
- 각 component가 선택될 확률



- 평균



- 분산

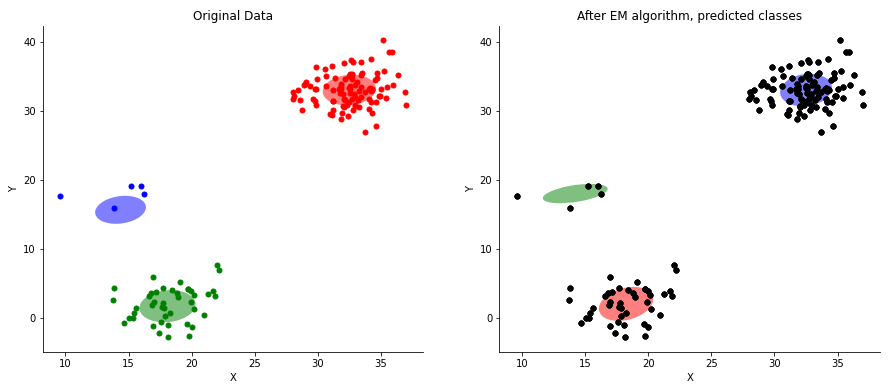


마지막으로 원래 data의 분산, 평균, 각 component가 선택될 확률 을 EM알고리즘이 종료된 후 우리가 구한 값들과 비교해 보면 모두 비슷한 값을 가지는 것을 확인할 수 있다.

3.1.2 각 component 별 선택될 확률 가 다를 때

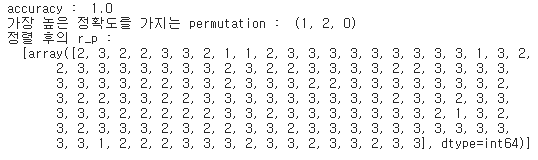
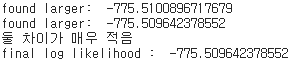
각 component별로 선택될 확률 가 모두 다르게 설정된다. 본 실험에서는 각각 왼쪽 위의 사진과 같은 확률로 선택되었고 위 확률을 바탕으로 데이터는 각각 class에 5, 43, 102 개로 나누어 졌다.

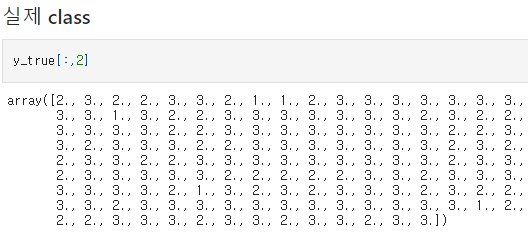
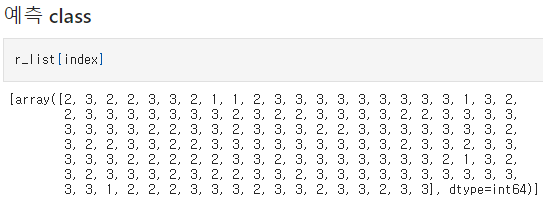
① 원래 데이터 분포와 clustering 이후의 분포 비교



본 실험 데이터를 확인해 보면 왼쪽 위의 그림과 같이 분포하는 것을 확인할 수 있다. 다음으로 데이터를 random한 순서로 섞은 후 EM 알고리즘을 반복적으로 실행하여 clustering 한 결과 오른쪽 위의 그림과 같이 데이터가 잘 clustering 된 것을 확인할 수 있다.

② log likelihood값과 정확도 비교





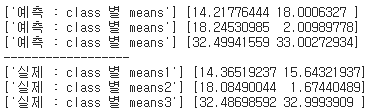
본 실험에서 2번의 iteration 후 최종 log likelihood값은 -775.5096으로 매우 높은 log likelihood값을 가지고 100%의 정확도를 가지며 그때 사용된 순열은 (1, 2, 0)이다. 실제 class 번호와 예측된 class 번호를 비교해 보면 모든 데이터가 동일한 것을 확인할 수 있다.

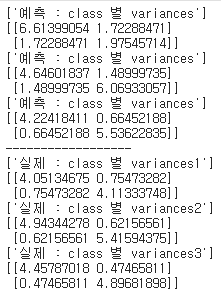
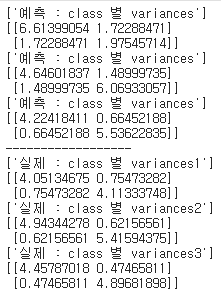
③ 각 component가 선택될 확률 , 평균, 분산 비교

- 각 component가 선택될 확률

****

- 평균

****

****- 분산

마지막으로 원래 data의 분산, 평균, 각 component가 선택될 확률 을 EM알고리즘이 종료된 후 우리가 구한 값들과 비교해 보면 평균과 는 비슷한 값을 가지는 것을 확인할 수 있다. 분산은 첫번째 class에서 분산의 차이가 조금 있지만 분류는 잘 되었음을 확인할 수 있다.

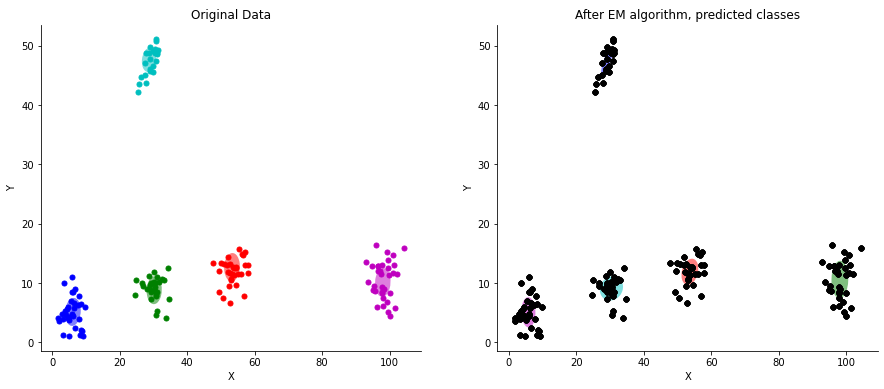
**3.2 class 5개**

3.2.1 각 component 별 선택될 확률 가 동일할 때 (=0.2)



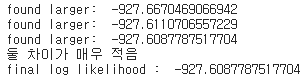
각 component별로 선택될 확률 가 동일하므로 각각의 component들은 모두 20%의 확률로 데이터가 선택될 것이다. 그 확률을 바탕으로 각 component의 데이터는 33, 28, 32, 23, 34개로 나누어 졌다.

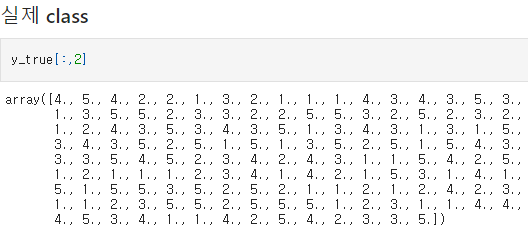
① 원래 데이터 분포와 clustering 이후의 분포 비교



실험 데이터를 확인해 보면 왼쪽 위의 그림과 같이 분포하는 것을 확인할 수 있다. 다음으로 EM 알고리즘을 반복적으로 실행하여 clustering 한 결과 오른쪽 위의 그림과 같이 데이터가 잘 clustering 된 것을 확인할 수 있다.

② log likelihood값과 정확도 비교





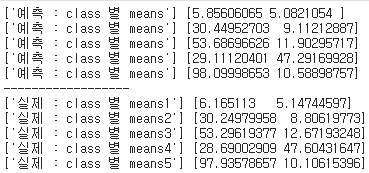
본 실험에서 3번의 iteration 후의 최종 log likelihood값은 -927.6088이고 100%의 정확도를 가지며 그때 사용된 순열은 (4, 3, 2, 0, 1)이다. 실제 class번호와 예측된 class 번호를 비교해 보면 모든 데이터가 동일한 것을 확인할 수 있다.

③ 각 component가 선택될 확률 , 평균, 분산 비교

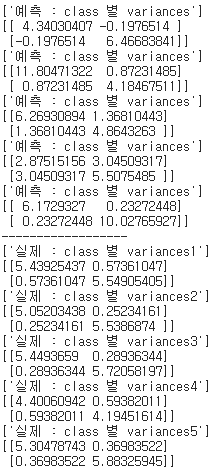
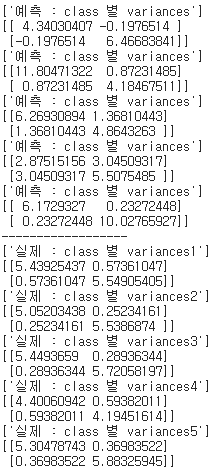
- 각 component가 선택될 확률



- 평균



- 분산

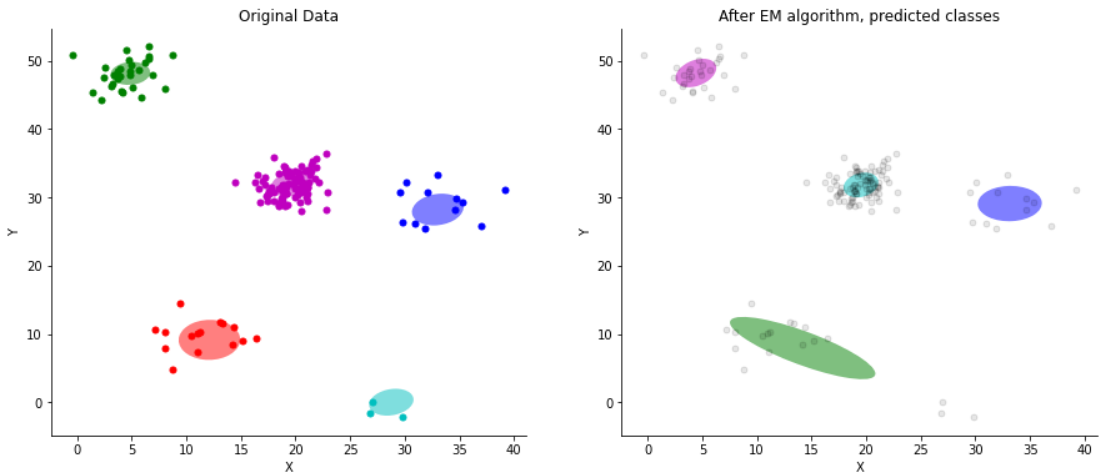


마지막으로 원래 data의 분산, 평균, 각 component가 선택될 확률 을 EM알고리즘이 종료된 후 우리가 구한 값들과 비교해 보면 2번 class의 분산을 제외한 나머지에서 모두 비슷한 값을 가지는 것을 확인할 수 있다.

3.2.2 각 component 별 선택될 확률 가 다를 때



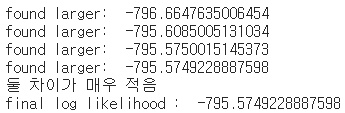
각 component별로 선택될 확률 가 각각 다르게 설정되고 위 확률을 바탕으로 데이터 150개는 각각 class에 12, 29, 15, 3, 91개로 나누어 졌다.

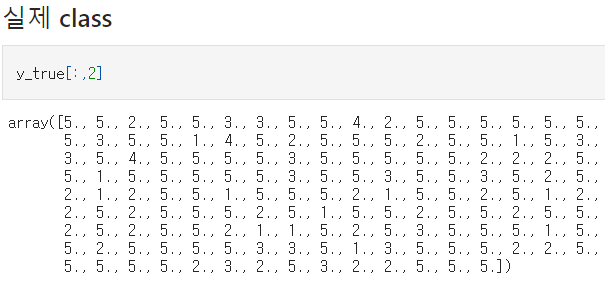


① 원래 데이터 분포와 clustering 이후의 분포 비교

실험 데이터를 확인해 보면 왼쪽 위의 그림과 같이 분포하는 것을 확인할 수 있다. 다음으로 EM 알고리즘을 반복적으로 실행하여 clustering 한 결과 오른쪽 위의 그림과 같이 데이터가 잘 clustering 된 것을 확인할 수 있다.

② log likelihood값과 정확도 비교





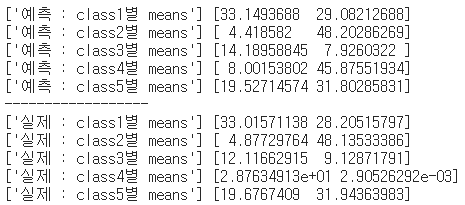
본 실험에서 총 4번의 iteration후의 최종 log likelihood값은 -795.5749이고 97.33%의 정확도를 가지며 그때 사용된 순열은 (0, 4, 1, 2, 3)이다. 실제 class번호와 예측된 class 번호를 비교해 보면 대부분의 데이터가 동일한 것을 확인할 수 있다.

③ 각 component가 선택될 확률 , 평균, 분산 비교

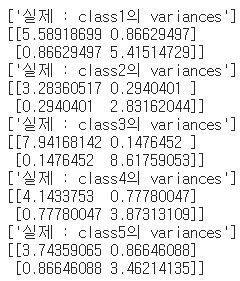
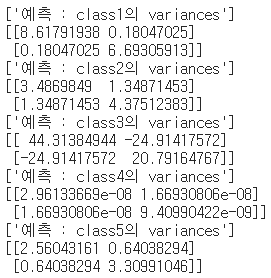
- 각 component가 선택될 확률



- 평균



- 분산



마지막으로 원래 data의 분산, 평균, 각 component가 선택될 확률 을 EM알고리즘이 종료된 후 우리가 구한 값들과 비교해 보면 4번 class의 와 평균, 3, 4번 class의 분산을 제외한 나머지에서는 모두 비슷한 값을 가지는 것을 확인할 수 있다.

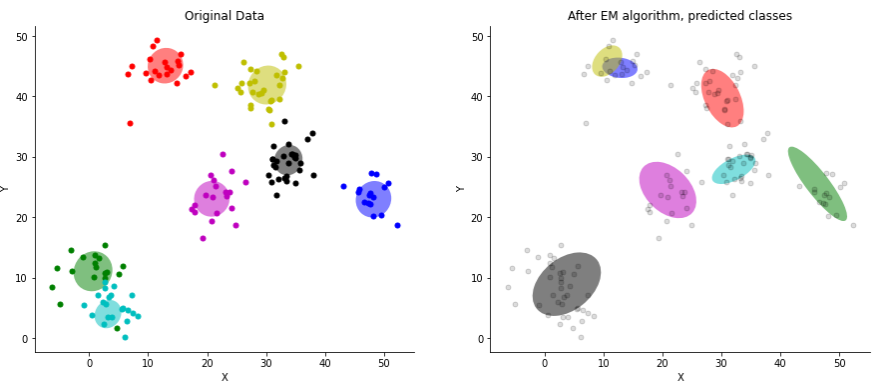
**3.3 class 7개**

3.3.1 각 component 별 선택될 확률 가 동일할 때 (=0.14285714)



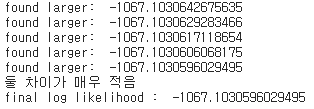
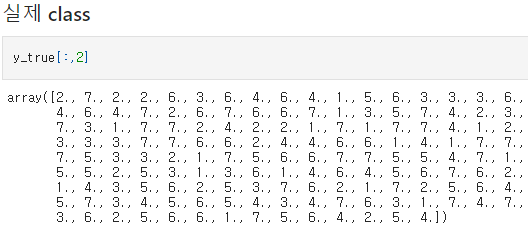
각 component별로 선택될 확률 가 동일하므로 각각의 component들은 모두 14.29%의 확률로 데이터가 선택될 것이다. 위 확률을 바탕으로 데이터 150개는 각각 class에 16, 18, 20, 21, 19, 28, 28 개로 나누어 졌다.

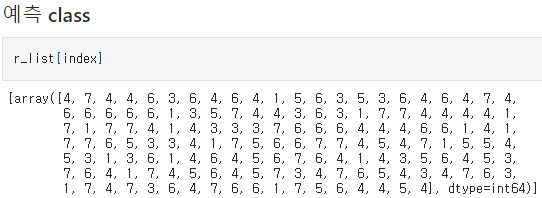
① 원래 데이터 분포와 clustering 이후의 분포 비교



실험 데이터를 확인해 보면 왼쪽 위의 그림과 같이 분포하는 것을 확인할 수 있다. 다음으로 EM 알고리즘을 반복적으로 실행하여 clustering 한 결과 오른쪽 위의 그림과 같이 모여 있는 몇 개의 class를 제외한 4개의 class는 잘 clustering 된 것을 확인할 수 있다.

② log likelihood값과 정확도 비교

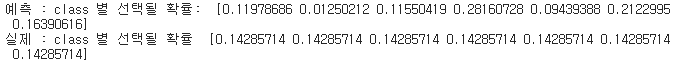




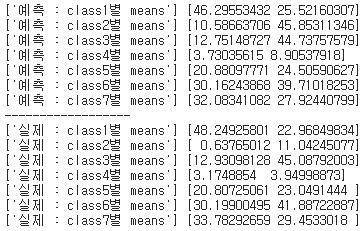
본 실험에서 많은 iteration 후의 최종 log likelihood값은 -1067.1031이고 82%의 정확도를 가지며 그때 사용된 순열은 (1, 5, 0, 6, 4, 2, 3)이다.

③ 각 component가 선택될 확률 , 평균, 분산 비교

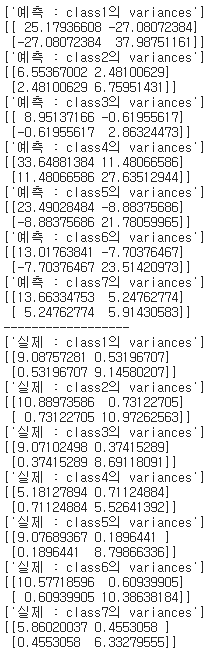
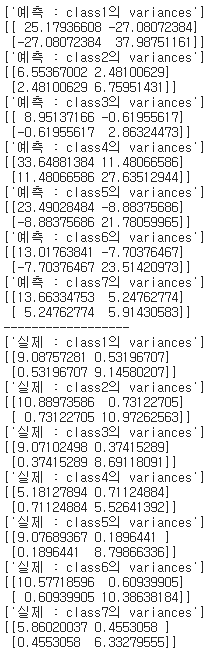
- 각 component가 선택될 확률



- 평균



- 분산

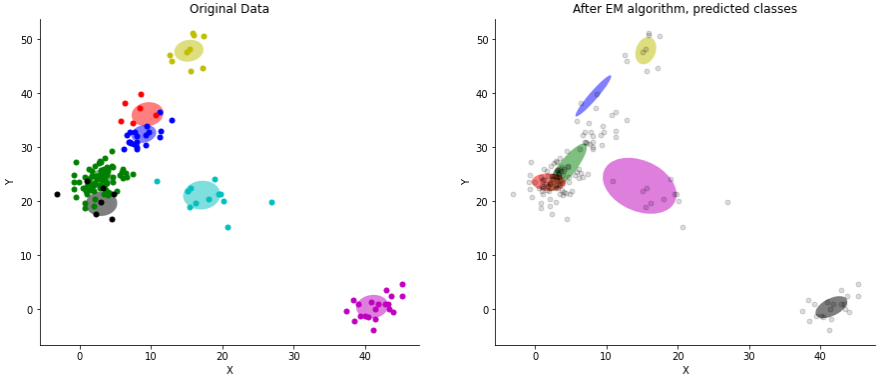
  
마지막으로 원래 data의 분산, 평균, 각 component가 선택될 확률 을 EM알고리즘이 종료된 후 우리가 구한 값들과 비교해 보면 2번 class의 와 평균을 잘 예측하지 못하고, 대부분의 분산을 잘 예측하지 못하는 것을 확인할 수 있다.

3.3.2 각 component 별 선택될 확률 가 다를 때



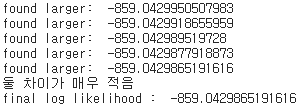
각 component별로 선택될 확률 가 왼쪽 위의 사진과 같은 각각 다른 확률로 선택되었고 위 확률을 바탕으로 데이터는 각각 class에 20, 76, 6, 12, 20, 9, 7개로 나누어 졌다.

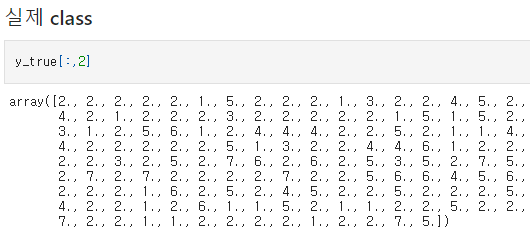
① 원래 데이터 분포와 clustering 이후의 분포 비교



실험 데이터를 확인해 보면 왼쪽 위의 그림과 같이 분포하는 것을 확인할 수 있다. 다음으로 EM 알고리즘을 반복적으로 실행하여 clustering 한 결과 오른쪽 위의 그림과 같이 모여 있지 않은 class에 대한 데이터만 잘 clustering 된 것을 확인할 수 있다.

② log likelihood값과 정확도 비교

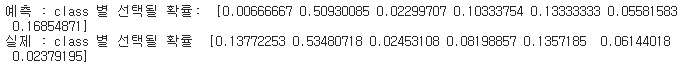




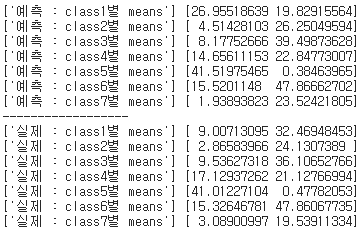
본 실험에서 많은 iteration 후의 최종 log likelihood값은 -859.0430이고 73.3%의 정확도를 가지며 그때 사용된 순열은 (3, 1, 0, 4, 6, 5, 2)이다.

③ 각 component가 선택될 확률 , 평균, 분산 비교

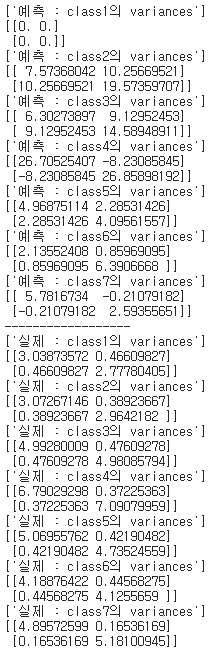
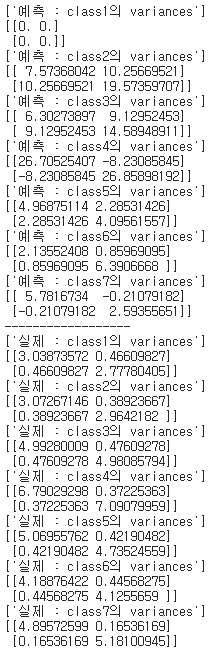
- 각 component가 선택될 확률



- 평균



- 분산

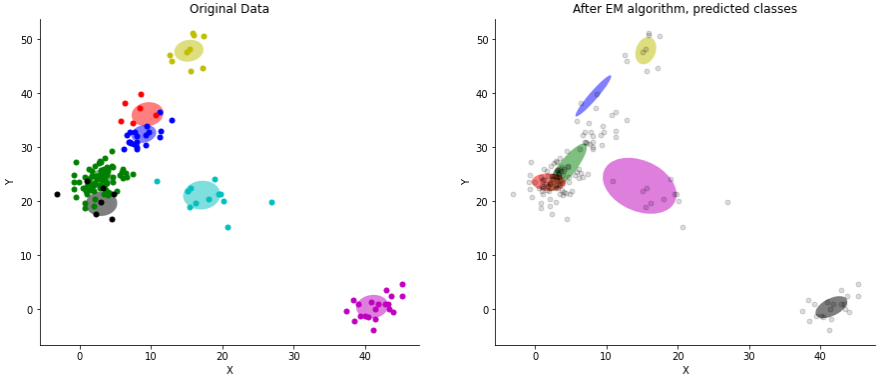


마지막으로 원래 data의 분산, 평균, 각 component가 선택될 확률 을 EM알고리즘이 종료된 후 우리가 구한 값들과 비교해 보면 1, 7번 class의 와 1번 class의 평균을 잘 예측하지 못하고, 3번 class의 분산을 잘 예측하지 못하는 것을 확인할 수 있다.

**4. Conclusion**

EM 알고리즘을 Gaussian Mixture Model에 적용하여 Clustering을 진행해 보았다. 본 실험은 초기 확률과 평균, 분산을 random한 값으로 설정한 후 log likelihood값의 차이가 보다 작을 때까지, 를 구하는 E-step과, 와 각 component가 선택될 확률 를 업데이트하는 M-step을 반복하여 진행한다. 즉 EM 알고리즘을 통해 log likelihood값을 최대로 하는 평균과 분산을 찾아서 clustering을 진행한다.

본 실험에서는 비교를 위하여 같은 수의 데이터, 총 150개의 데이터를 class의 개수를 바꾸어 가며 실행하였다. Class 3, 5, 7개로 실행을 해보며 비교를 해 보았는데 class 3개로 진행하였을 때는 보통 100%의 정확도로, 5개로 진행되었을 때는 평균적으로 95% 이상의 정확도로, 마지막으로 7개로 진행하였을 때는 75% 정도의 정확도를 보였다.



**Figure 7 class 7개중 몇 개의 class가 겹쳐진 경우**

같은 양의 데이터를 사용해 class를 나누다 보니 class가 적을수록 clustering이 진행이 잘 되었지만 class수가 7개일때는 위의 그림처럼 class들이 모여 있는 현상이 많이 나타났다. 따라서 EM 알고리즘을 반복적으로 실행하여도 직관적으로 모여 있는 데이터들의 clustering이 어려웠고 결과적으로 예측된 7개의 class들의 분산과 평균, 각 component가 선택될 확률 값에서 차이가 많이 발견되었다.

즉 결과적으로 데이터의 수가 동일할 때, class의 수가 많아질수록 모여 있는 class가 많이 발견될 확률이 높아지기 때문에 clustering에 어려움이 따른다.

하지만 class수가 적고 (보통 5이하) 각 class들의 평균 거리가 멀고 잘 퍼져 있다면 clustering 결과는 매우 높고 그때의 log likelihood값도 높은 것을 실험을 통해 확인할 수 있었다.