**2022311499 고우빈**  
Chart, bubble chart

Description automatically generated with medium confidenceChart, scatter chart

Description automatically generatedFigure 1.a는 p1 train dataset을 좌표평면에 나타낸 것이며, label이 0인 경우 파란 점, 1인 경우 빨간 점으로 표시했다.  
빨간 점과 파란 점이 위치를 특정할 수 없을 정도로 뒤섞인 것을 볼 수 있다. 여기서 예상할 수 있는 것은, 보통의 예시처럼 Gaussian Component를 label의 가짓수만큼 설정하는 것(현재의 경우 2개)으로는 좋은 결과를 얻을 수 없다는 것이다.  
Figure 1.b는 component의 개수가 2개이고, covariance\_type을 full로 설정한 Gaussian Mixture Model을 fit시킨 뒤, 2개의 component를 Figure1.a위에 그린 것이다. 예상한 대로 데이터의 분포를 제대로 반영하지 못하고 있는 모습이다. 또한, Train Accuracy도 53.3%에 불과하다.  
 **Figure1.a (p1\_train\_dataset) Figure1.b (p1\_train\_dataset with gaussian component)**

Chart, bubble chart

Description automatically generated**Method**  
정확성을 높이기 위해 p1의 데이터들을 같은 크기의 4x4 섹션으로 나누고, 각 섹션마다 다른 label을 부여할 것이다. 가장 왼쪽 위 섹션부터 가장 오른쪽 아래 섹션까지 0 ~ 15의 새로운 class를 새로 부여한다. 0, 2, 5, 7, 8, 10, 13, 15번 클래스는 빨간 점(원래 label이 1)이 대부분을 차지하고, 나머지 8개의 클래스는 파란 점(원래 label이 0)이 대부분을 차지할 것이다.  
Gaussian Component의 개수를 n개로 설정하면, predict했을 때의 결과도 n개의 class가 나와 기존 label인 0, 1 외에도 2, 3, n과 같은 class로 예측할 것이기 때문에 이와 같은 방법을 선택했다.  
섹션의 개수만큼16개의 component를 생성하고, predict 결과 또한 0~15 범위 내에서 나오도록 한 뒤, predict 결과가 0, 2, 5, 7, 8, 10, 13, 15에 속한다면 빨간 점(원래 label 1), 1, 3, 4, 6, 9, 11, 12, 14에 속한다면 파란 점(원래 label 0)으로 간주할 것이다. 예상되는 출력은 Figure 2.a이고, 실제 출력은 Figure2.b이다.

A picture containing text

Description automatically generated **Figure2.a Figure2.b**

**Result**

초기 파라미터들은 다음과 같다.  
n\_components=16, tol=0.0001, covariance\_type=”spherical”, max\_iter=100  
실행한 결과, 바랐던 결과와는 다르게 각 covariance들의 위치가 제각각이고, 각 클래스에 일치하는 data들을 포함하지 못하는 모습을 보인다. 이 내용은 뒤에서 다시 다룬다.  
Train Accuracy는 55.1%, Test dataset에 대한 Accuracy는 49.8%에 불과하고, iteration의 최댓값을 100으로 설정했음에도 불구하고 40번의 iteration만을 수행했다. (convergence threshold = 0.0001보다 작은 변화를 보였기 때문)

**Chart, bubble chart

Description automatically generatedChart, bubble chart

Description automatically generatedChart, bubble chart

Description automatically generated**Chart, bubble chart

Description automatically generatedFigure 3.a ~ d는 spherical, diagonal, tied, full 4개의 covariance 타입에 따른 출력, Table1은 그에 따른 train/test accuracy이다.  
 **Figure3.a(Spherical) Figure3.b(Diagonal) Figure3.c(tied) Figure3.d(full)**

|  |  |  |
| --- | --- | --- |
| **covariance\_type** | **Train Accuracy** | **Test Accuracy** |
| spherical | 55.1% | 49.8% |
| diagonal | 50.2% | 48.9% |
| tied | 54.7% | 49.8% |
| full | 51.1% | 48.9% |

**Table1**

4가지 경우 모두 test accuracy가 50%를 넘기지 못했다. 별개로, 각 covariance type에 따른 iteration 횟수는 spherical 40회, diagonal 30회, tied 20회, full 20회에 그쳤다. 더 많은 iteration을 거치며 covariance를 수정해도 유의미한 개선점을 얻을 수 없다는 의미로 해석할 수 있다.

결과로 나온 좌표평면을 보면, 16개의 Component들이 의도한 바(4x4)와는 다르게 제각각 아무데나 위치하고 있으며, 파란 점이 분포한 지역에 빨간 covariance가 보이기도 한다.  
이에 대한 원인을 생각해봤는데, GMM은 unsupervised learning technique이다. model을 fit시키는 과정에서 각 data들에 대한 label을 피드백으로서 제공해주지 않기 때문에, 우리가 원하는 label 기준으로 covariance가 이동하지 않는다. (우리에게는 빨간 점/파란 점으로 구분되지만 GMM 입장에서는 모두 같은 검정색 점으로 보일 것이다.)  
또한, 우리가 원하는 대로 covariance가 4x4의 형태로 잘 위치하고 있다고 해도, 우리가 임의로 설정했던 16가지의 클래스와 일치할 것이라는 보장이 없다. 우리가 정한 기준은 왼쪽 위부터 [0, 1, 2, …, 15]지만, GMM predict가 내놓는 클래스의 기준은 [3, 5, 12, 8, …, 2]와 같이 뒤죽박죽일 수 있다.

내가 GMM을 제대로 이해했는가에 의심이 많이 생기지만, 서로 다른 label을 가진 data들이 골고루 퍼져있는 경우 GMM으로 이를 classify하기 적절하지 않다는 결론에 도달했다.  
이후 내용은p2 dataset에 대해 component 개수를 2로 고정하고(개수를 늘리면 예측 결과의 가짓수도 늘어나기 때문에), 그 외에 다른 parameter 값들을 바꿔서 실행했을 때 나오는 결과들의 차이에 집중했다.

Chart

Description automatically generated**Chart, bubble chart

Description automatically generated**Chart, bubble chart

Description automatically generatedFigure 4.a~d는 p2 dataset에 대해 4가지 covariance type을 주었을 때의 결과이고, Table2는 각 결과의 accuracy이다.

Chart, bubble chart

Description automatically generated **Figure4.a (Spherical) Figure4.b (Diagonal) Figure4.c (Tied) Figure4.d (Full)**

|  |  |  |
| --- | --- | --- |
| **covariance\_type** | **Train Accuracy** | **Test Accuracy** |
| spherical | 54.2% | 50.2% |
| diagonal | 50.2% | 56.4% |
| tied | 55.1% | 61.3% |
| full | 56.0% | 62.2% |

**Table2**

이번에는 covariance type에 따른 결과 차이가 약간 보인다.  
p2 dataset은 나선 모양을 띄고 있는 것을 볼 수 있는데, tied와 full type의 covariance처럼 각도가 기울어진 타원이 왼쪽 위의 빨간 점들이 모인 지점과, 오른쪽 아래의 파란 점들이 모인 지점을 찾아가 조금 더 나은 Accuracy를 보일 수 있다고 생각했다.

tol값을default값의 10분의 1인 0.0001로 설정했음에도 불구하고, default max\_iter(100)까지 가는 경우가 한 번도 없었다.spherical과 diagonal의 경우 50번, tied는 10번 이하, full은 10번 실행됐다.  
따라서, 그나마 수행횟수가 높은 diagonal을 선택해 max\_iter를 바꿔가며 실행해봤다.

각각 max \_iter를 10, 20, 40, 50을 주고 실행한 결과이다. 빨간 원이 점점 작아지는 것을 볼 수 있다.  
각 경우에 대해 accuracy를 출력해봤는데, 아래와 같은 결과가 나왔다.

max\_iter=10 -> train 55.1%, test 61.3%  
max\_iter=20 -> train 56.9%, test 60.0%  
max\_iter=40 -> train 55.1%, test 55.6%  
max\_iter=50 -> train 55.1%, test 50.2%

위 수치에서 볼 수 있다시피, test accuracy가 iteration이 증가함에 따라 오히려 감소하는 모습을 보인다.  
overfitting이 일어났다고 하기에는 train accuracy가 그대로인데, 각 epoch마다 피드백이 제공되지 않는 unsupervised learning이기 때문이라고 추측된다.  
(다음 장에 위 내용에 대한 자료 있습니다.)

Chart, pie chart, bubble chart

Description automatically generated**References**1. <https://towardsdatascience.com/gaussian-mixture-models-with-python-36dabed6212a>  
2. <https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture.fit>  
3. <https://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_covariances.html>  
4. <https://jakevdp.github.io/PythonDataScienceHandbook/05.12-gaussian-mixtures.html>

**Environment**python 3.9.0  
scikit-learn 1.0.2  
matplotlib 3.5.1  
numpy 1.21.4