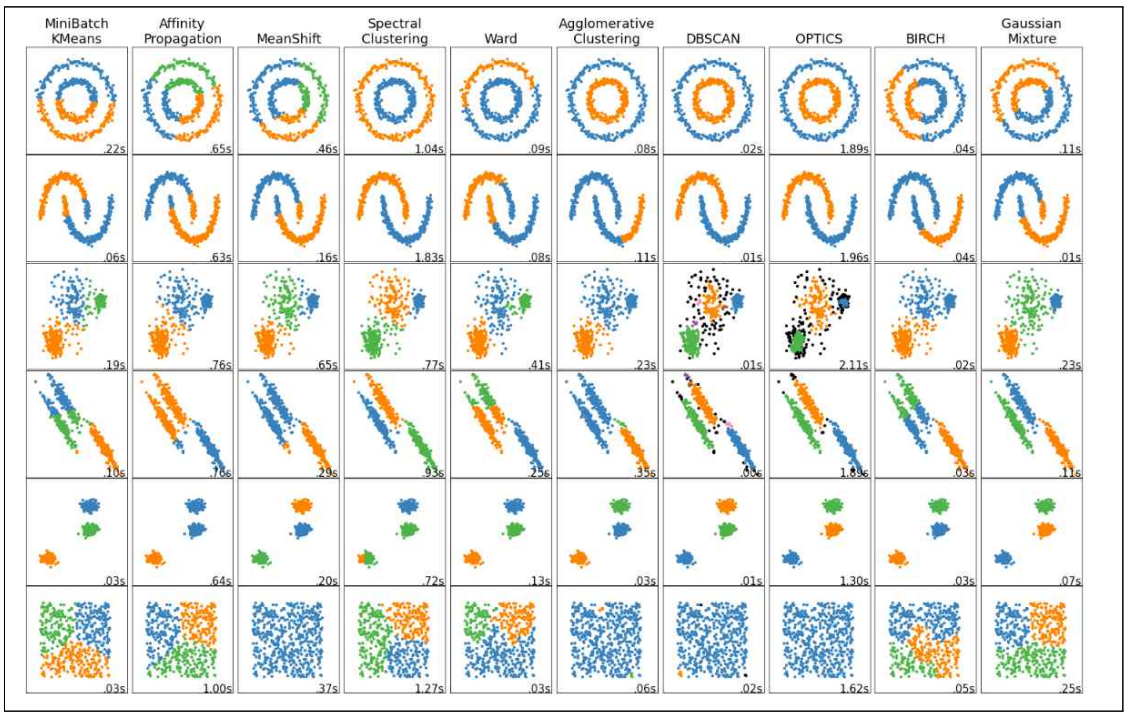
<CSE 302 Assignment 3>

202011212 진호진

1. Run the code below and arrange the result



우리가 앞으로 수행해야 할 4가지의 clustering 방법이 잘 작동하고 있다는 것을 알 수 있었다.

2. Sample 100 images randomly for each class (total 1000 images) from the MNIST training data set.

```
def sampling(num):
  image = []
  label = []

  for i in range(0, len(y_train)):
    if y_train[i] == num:
      image.append(xtrain[i])
      label.append(y_train[i])

  return zip(random.sample(image, 100), random.sample(label, 100))
```

MNIST training data set을 불러 온 후 간단한 sampling 함수에서 random.sample을 이용하여 100개의 image를 랜덤하게 추출하였다. sampling의 num input은 class 번호이다. 따라서 함수를 0부터 9까지 순서대로 실행하여 10개 class에 각각 100개씩 1000개의의 random image를 가져오게 되었다.

3. For 1000 images, perform Agglomerative clustering, k-means clustering, Gaussian mixture model, Spectral clustering. (i.e. k = 10)

```
img_ran, label_ran = zip(*S)
# Training
Kmeans = KMeans(n_clusters = 10, random_state = 42)
Kmeans.fit(img_ran)
```

```
AC = AgglomerativeClustering(n_clusters = 10).fit(img_ran)
prediction = AC.fit_predict(img_ran)
```

```
GM = GaussianMixture(n_components = 10, random_state = 42)
GM = GM.fit(img_ran)
```

```
SC = SpectralClustering(n_clusters = 10, n_init = 10)
SC.fit(img_ran)
```

다음과 같이 4가지의 clustering을 사용하여 1000개의 data에 적용하였다.

4. Based on the clustering results and the labels we know, compute “Rand index” and “mutual information based score”. Explain your findings.

코드가 전부 동일하여 kmeans를 대표로 가져왔다. mutual_info_score는 normalized 된 것을 사용하였는데, 0에서 1로 재한하여 조금더 직관적으로 알아보기 위해서이다.

```
prediction = Kmeans.fit_predict(img_ran)

print("rand_index:",rand_score(label_ran,prediction))
print("mutual_info_score:",normalized_mutual_info_score(label_ran,prediction))
```

rand_index: 0.8774914914914915 mutual_info_score: 0.5066922014129566	kmeans의 rand_index와 mutual_info_score이다. rand_index는 1에 가까울수록 좋은 값이며, mutual_info_score는 0에 가까울수록 확률변수가 독립이라는 의미이다. 이에 근거하여 점수를 비교해 보면 kmeans는 적당히 좋은 성능을 가지고 있다는 것을 알 수 있다.
rand_index 0.8786706706706706 mutual_info_score: 0.5908255313818036	Agglomerative clustering의 rand_index mutual_info_score 이다. 수치가 위의 kmeans와 상당히 유사하다. 이를 통해 Agglomerative clustering 또한 괜찮은 성능을 보여주고 있는 것을 알 수 있다.
rand_index 0.8774914914914915 mutual_info_score: 0.5066922014129566	Gaussian mixture model의 rand_index, mutual_info_score 이다. 신기하게 위의 2개와 거의 똑같은 수치를 보이고 있다. 따라서 위의 2개와 마찬가지로 괜찮은 성능일 것이다.
rand_index 0.13910310310310312 mutual_info_score: 0.03838009941772235	Spectral Clustering의 결과이다. rand_index와 mutual_info_score이 n_init 을 10 일때 비정상적으로 작은 것을 확인 할 수 있었다. 이는 affinity를 변경하면서 확인하면 바뀔 수 있을 것 같다.

5. Based on the clustering results, you can get the center of each cluster. Classify the MNIST test data set using 1-NN classifier and provide accuracy. Explain your findings.

kmeans 를 제외 하고는 center를 구하는 함수는 따로 존재하지 않았다. 하지만 단순하게 각 class의 list의 중앙값을 가져오는 것으로 간단하게 구할 수 있었다. 이는 단순히 중앙값을 구하는 함수로 간단하게 작성하였기 때문에 따로 코드를 가져 오진 않았다. 이후 center를 K-NN의 k 값을 1로 설정하여 classifier를 사용하고 classification report에서 accuracy를 관찰하였다.

```
KNN=KNeighborsClassifier(n_neighbors = 1) #1-NN
KNN.fit(Xt,Yt)
prediction_knn = KNN.predict(xtest)
print("Accuracy:₩n",classification_report(y_test,prediction_knn))
```

Accuracy:

	precision	recall	f1-score	support
0	0.89	0.68	0.77	980
1	0.56	0.98	0.71	1135
2	0.89	0.50	0.64	1032
3	0.66	0.68	0.67	1010
4	0.00	0.00	0.00	982
5	0.36	0.46	0.40	892
6	0.69	0.78	0.73	958
7	0.57	0.69	0.62	1028
8	0.00	0.00	0.00	974
9	0.36	0.78	0.49	1009
accuracy			0.56	10000
macro avg	0.50	0.55	0.50	10000
weighted avg	0.50	0.56	0.51	10000

kmeans의 classification report이다. accuracy는 56퍼센트 정도이다. 절반이 조금 넘는 정확성을 보장해준다. data개수가 너무 적은 것도 영향을 미칠 수 있을 것 같다.

Accuracy:

	precision	recall	f1-score	support
0	0.37	0.69	0.48	980
1	0.00	0.00	0.00	1135
2	0.20	0.85	0.33	1032
3	0.00	0.00	0.00	1010
4	0.00	0.00	0.00	982
5	0.22	0.74	0.34	892
6	0.13	0.06	0.08	958
7	0.32	0.13	0.18	1028
8	0.00	0.00	0.00	974
9	0.00	0.00	0.00	1009
accuracy			0.24	10000
macro avg	0.12	0.25	0.14	10000
weighted avg	0.12	0.24	0.14	10000

Agglomerative clustering 의 classification report이다. 24퍼센트 정도의 accuracy 를 보장해준다. 상당히 낮은 정확도를 보인다. 상당히 높은 정확도를 보일 것에 반하여 낮은 것으로 보아 데이터가 너무 적어 지엽적인 결과가 나온 것으로 예상해 볼 수 있다.

Accuracy:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	980
1	0.00	0.00	0.00	1135
2	0.74	0.19	0.30	1032
3	0.45	0.21	0.29	1010
4	0.53	0.52	0.53	982
5	0.00	0.00	0.00	892
6	0.54	0.47	0.51	958
7	0.58	0.26	0.36	1028
8	0.23	0.75	0.35	974
9	0.21	0.77	0.33	1009
accuracy			0.32	10000
macro avg	0.33	0.32	0.27	10000
weighted avg	0.33	0.32	0.26	10000

Gaussian mixture model 의 accuracy 이다. 32퍼센트 정도의 accuracy 를 보장해준다. 이 또한 data가 좀더 많아지면 clustering이 다양해지므로 정확도가 올라갈 것이다.

Accuracy:

	precision	recall	f1-score	support
0	0,00	0,00	0,00	980
1	0,16	1,00	0,27	1135
2	0,00	0,00	0,00	1032
3	0,00	0,00	0,00	1010
4	0,25	0,67	0,36	982
5	0,00	0,00	0,00	892
6	0,00	0,00	0,00	958
7	0,10	0,02	0,04	1028
8	0,00	0,00	0,00	974
9	0,00	0,00	0,00	1009
accuracy			0,18	10000
macro avg	0,05	0,17	0,07	10000
weighted avg	0,05	0,18	0,07	10000

Spectral clustering 의 accuracy는 18퍼센트로 가장 낮게 나왔다. 예상과 다른 결과였지만, 이는 Spectral clustering의 옵션이 다양하여 발생하는 문제로 생각 된다.