Final Project

머신러닝입문

2016121150

윤준영

머신러닝입문 기말 프로젝트

2016121150 윤준영

1. Fully Connected NN Design for Image Classification

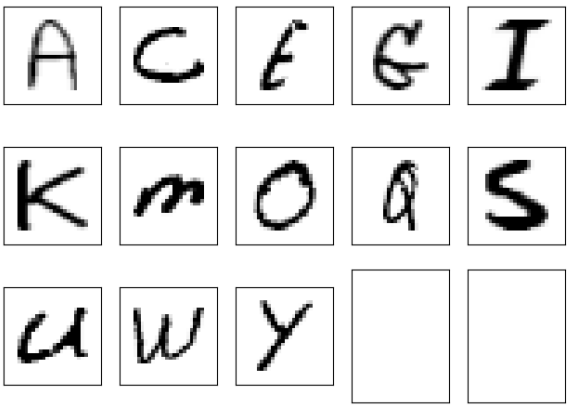
Fully Connected Neural Network Architecture는 input layer와 hidden layer, output layer가 연결되어 있으며, hidden layer는 input layer와 완전히 연결되어 있으며, output layer는 hidden layer와 완전히 연결되어 있는 구조이다. 본 과제에서는 Multilayer Perceptron(MLP) 알고리즘을 사용하였다.

주어진 데이터는 28x28 픽셀이 0~255의 값을 가지는 데이터로 0은 밝은 부분을 의미하고 255는 어두운 부분을 의미한다. 머리부분은 다음과 같다.

	0	1	2	3	4	5	6	...	778	779	780	781	782	783	784
0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0

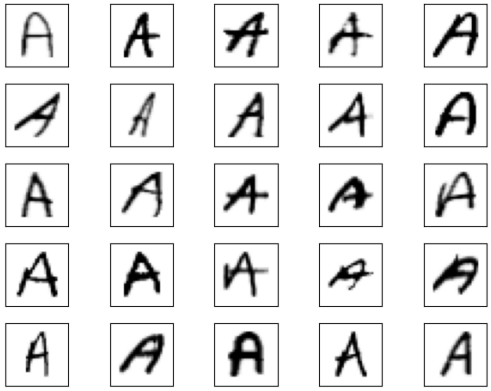
[5 rows x 785 columns]

클래스는 0부터 12까지 13개의 클래스가 있으며, 각 클래스마다 10000개의 데이터가 있다. 각 클래스의 첫번째 샘플을 나타내면 다음과 같다.



따라서 0부터 12의 클래스는 각각 A, C, E, G, I, K, M, O, Q, S, U, W, Y를 의미한다.

A의 앞의 샘플 25개를 나타내면 다음과 같으며, A를 나타낸다는 것을 확인할 수 있다.



MLP 알고리즘을 사용하여 130000개의 알파벳 data를 학습시켰으며, hidden layer의 크기를 10, 50, 100으로 설정하여 학습시켰다. L2 penalty는 0.01, learning rate는 0.0005, 최대 반복 횟수는 100회로 설정하였다.

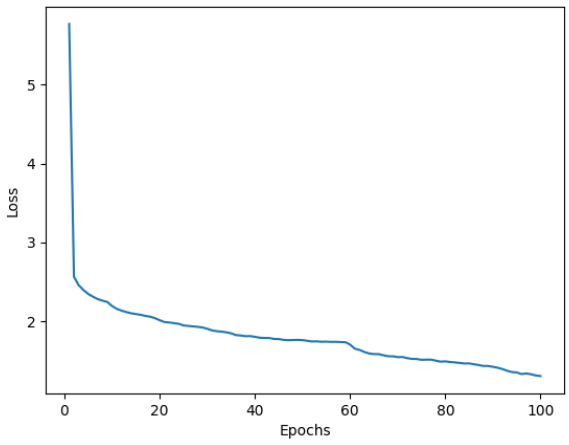
1-(a) 784-10-10 MLP (hidden layer size = 10)

MLP 분류기의 hidden layer size를 10으로 설정하여 학습한 결과는 다음과 같다.

```
₩arning (from warnings module):
  File "C:₩Users₩Blu₩AppData₩Local₩Programs₩Python₩Python39₩lib₩site-packages₩sklearn₩neural_network₩_multilayer_perceptron.py", line 692
    warnings.warn(
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reached and the optimization hasn't converged yet.
Hidden layer size : 10
784-10-10 MLP Train Acc: 0.501
784-10-10 MLP Test Acc: 0.477
Converged at 100 iteration
[5.771 2.564 2.455 2.395 2.346 2.311 2.28  2.26  2.244 2.192 2.156 2.134
 2.116 2.101 2.09  2.081 2.067 2.057 2.039 2.013 1.989 1.984 1.975 1.967
 1.947 1.94  1.934 1.929 1.92  1.904 1.883 1.873 1.868 1.859 1.846 1.824
 1.819 1.81  1.811 1.802 1.79  1.787 1.786 1.775 1.774 1.763 1.759 1.761
 1.763 1.76  1.752 1.742 1.744 1.739 1.741 1.737 1.738 1.735 1.734 1.704
 1.651 1.635 1.609 1.59  1.583 1.582 1.568 1.555 1.555 1.544 1.546 1.531
 1.522 1.52  1.51  1.512 1.512 1.5   1.487 1.49  1.482 1.477 1.47  1.463
 1.464 1.453 1.444 1.432 1.432 1.421 1.41  1.393 1.371 1.355 1.349 1.329
 1.337 1.327 1.312 1.304]
```

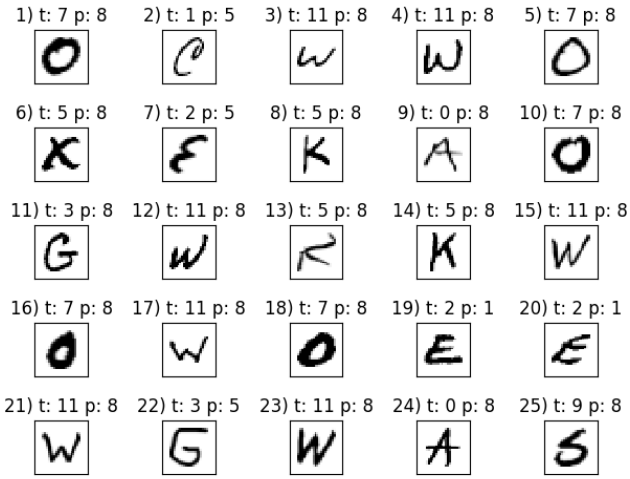
10개의 hidden layer를 사용하였을 때에는 100회를 반복하여도 수렴하지 않았음을 확인할 수 있었고, 정확도 또한 0.477로 상당히 낮은 결과를 얻을 수 있었다. 또한 학습 시간도 오래 걸렸다.

다음은 각 반복마다 측정된 loss를 나타낸 그래프이다.



그래프를 살펴보면 100회 이후에도 loss가 계속 줄어드는 것을 확인할 수 있었으며, 반복횟수를 증가시킨다면, 수렴하기 전까지 더 높은 정확도를 얻을 수 있음을 기대할 수 있다.

다음은 잘못 분류된 data를 25가지 나타낸 것이다.



대표적으로 O, W, E를 분류하는데 어려움이 있었다는 것을 확인할 수 있었다.

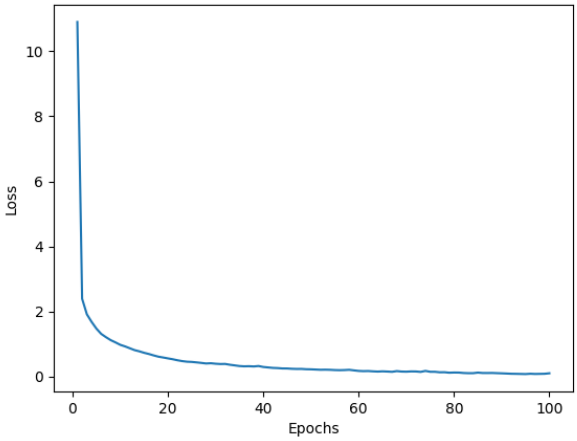
1-(b) 784-50-10 MLP (hidden layer size = 50)

다음은 MLP 분류기의 hidden layer size를 50으로 설정하여 학습한 결과는 다음과 같다.

```
₩arning (from warnings module):
  File "C:₩Users₩Blu₩AppData₩Local₩Programs₩Python₩Python39₩lib₩site-packages₩s
klearn₩neural_network₩_multilayer_perceptron.py", line 692
    warnings.warn(
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reached and
the optimization hasn't converged yet.
Hidden layer size : 50
784-50-10 MLP Train Acc: 0.970
784-50-10 MLP Test Acc: 0.899
Converged at 100 iteration
[10.899  2.394  1.915  1.681  1.475  1.314  1.213  1.122  1.053  0.978
  0.929  0.873  0.815  0.778  0.733  0.696  0.653  0.613  0.588  0.561
  0.535  0.503  0.477  0.459  0.453  0.438  0.422  0.404  0.412  0.397
  0.388  0.391  0.365  0.346  0.326  0.317  0.321  0.313  0.327  0.297
  0.283  0.269  0.263  0.252  0.251  0.241  0.237  0.238  0.228  0.225
  0.218  0.212  0.214  0.21   0.202  0.199  0.202  0.211  0.193  0.176
  0.169  0.173  0.163  0.156  0.163  0.157  0.149  0.17   0.156  0.153
  0.161  0.159  0.146  0.176  0.149  0.149  0.134  0.136  0.121  0.127
  0.125  0.112  0.106  0.107  0.124  0.112  0.111  0.113  0.107  0.102
  0.096  0.089  0.086  0.083  0.079  0.089  0.082  0.085  0.088  0.103]
```

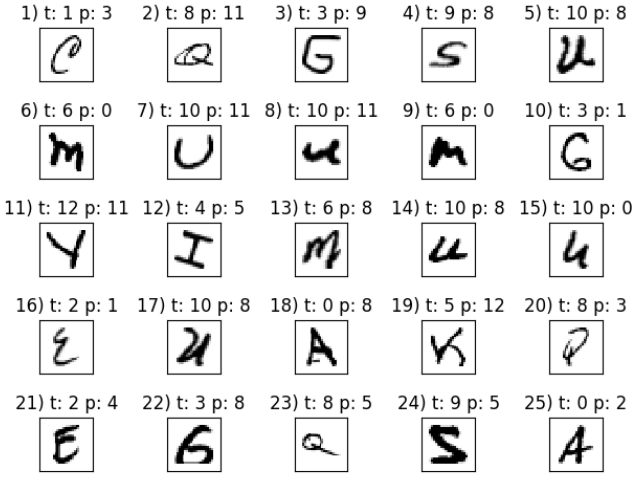
이번의 경우에도 100회의 반복횟수를 모두 사용하였지만 수렴하지 않았다는 경고를 확인할 수 있었다. 하지만 정확도는 hidden layer size가 10일 때보다 매우 높은 값인 0.9에 가까운 값을 가지는 것을 확인할 수 있었다. 학습 시간은 이전 학습보다 상당히 오래걸리는 것을 체감할 수 있었다.

다음은 반복횟수에 따른 loss 그래프이다.



이전 학습과 비교하면 초반 학습률이 매우 좋았던 것을 확인할 수 있었고, 100이후 얼마 지나지 않아 수렴할 것이라는 것을 기대할 수 있다.

다음은 잘못 분류된 25가지의 data이다.



이전과는 달리 골고루 분류하지 못했다는 것을 확인할 수 있었으며, 몇몇 잘못 분류된 예시들은 사람의 경우에도 쉽게 분류하지 못하는 알파벳들도 있었다.

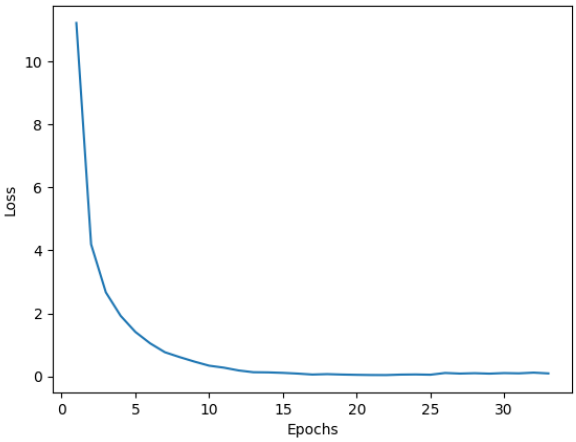
1-(c) 784-100-10 MLP (hidden layer size = 100)

마지막으로, MLP 분류기의 hidden layer size를 100으로 설정하여 학습한 결과는 다음과 같다.

```
Hidden layer size : 100
784-100-10 MLP Train Acc: 0.993
784-100-10 MLP Test Acc: 0.911
Converged at 33 iteration
[11.217  4.192  2.668  1.924  1.412  1.05   0.767  0.613  0.47   0.34
  0.277  0.189  0.132  0.127  0.112  0.089  0.059  0.072  0.058  0.05
  0.044  0.042  0.057  0.061  0.053  0.109  0.09   0.102  0.088  0.105
  0.097  0.119  0.095]
```

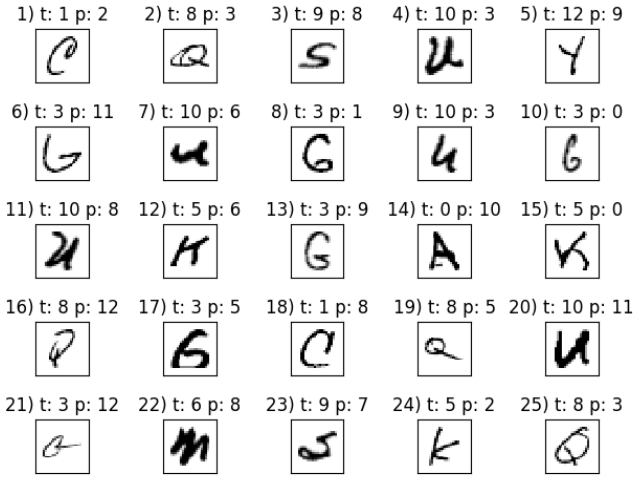
Hidden layer size를 100으로 늘리니 33회의 반복만으로도 수렴하였다는 것을 알 수 있었다. 정확도는 0.911의 정확도를 가졌으며, 학습 시간은 100회의 반복횟수

를 모두 채웠던 이전 학습보다 적게 걸렸다. Loss 행렬을 살펴보면, 22번째의 반복에서 가장 loss가 적었던 것을 확인할 수 있었다.



반복횟수에 따른 loss 그래프를 보면 15에서 20부근에서 어느정도 수렴하는 것까지 확인할 수 있었다.

잘못 분류된 25가지의 data는 다음과 같았다.



Hidden layer size가 100인 경우 50인 경우보다 육안으로 구분하기 어려운 sample들이 잘못 분류된 경우가 많았다. 이는 hidden layer size가 늘어날수록 대체적으로 더 높은 정확도를 가지며, 빠르게 수렴한다는 것을 확인할 수 있는 결과이다.

1-(d) Perceptron과 비교

Midterm project에서 준수한 성능과 계산시간을 보였던 perceptron을 사용하여 학습한 결과는 다음과 같다.

```
Perceptron Train Acc: 0.900
Perceptron Test Acc: 0.853
```

0.853의 정확도를 보였으며, 이는 hidden layer size가 10일 때보다 높으며 50일 때보다는 낮은 정확도를 가졌다. 하지만 계산시간을 고려하면, 큰 정확도를 필요로 하지 않는다면, 작은 hidden layer size를 택하여 수렴하는 시간을 기다리는 것 보다, 단일 perceptron을 사용하는 것 또한 합리적인 선택으로 보인다.

2. Multiple Linear Regression Problem

Multiple linear regression은 n개의 feature로 이루어진 data를 n개의 linear regression으로 조합하는 방법으로 target을 구분할 수 있는 n차원의 hyperplane을 생성한다.

먼저 주어진 데이터의 머리 부분을 살펴보면 다음과 같다.

	0	1	2	3	4
0	118474.03	A	76253.86	113867.30	298664.47
1	111313.02	C	78389.47	153773.43	299737.29
2	110352.25	B	73994.56	122782.75	303319.26
3	108733.99	B	67532.53	105751.03	304768.73
4	108552.04	C	77044.01	99281.34	140574.81

주어진 R_table data는 5가지의 feature을 가지는 50개의 데이터로 이루어져 있으며 첫번째의 feature가 target data가 되고 나머지 4개의 특성이 feature data로 분석하여야 한다.

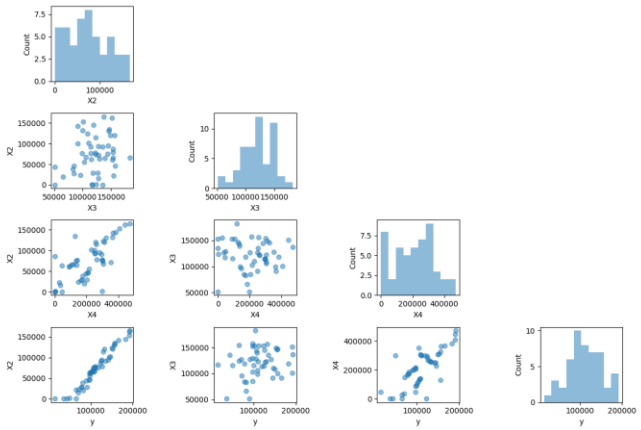
2-(a) Data preprocessing: One-hot encoding

먼저 Label을 y, X1, X2, X3, X4로 바꾼 뒤 문자 A, B, C로 이루어진 X1 feature에 대해 one-hot encoding을 적용하여 새로운 3개의 feature를 만들면 다음과 같다.

	y	X2	X3	X4	X1_A	X1_B	X1_C
0	118474.03	76253.86	113867.30	298664.47	1	0	0
1	111313.02	78389.47	153773.43	299737.29	0	0	1
2	110352.25	73994.56	122782.75	303319.26	0	1	0
3	108733.99	67532.53	105751.03	304768.73	0	1	0
4	108552.04	77044.01	99281.34	140574.81	0	0	1

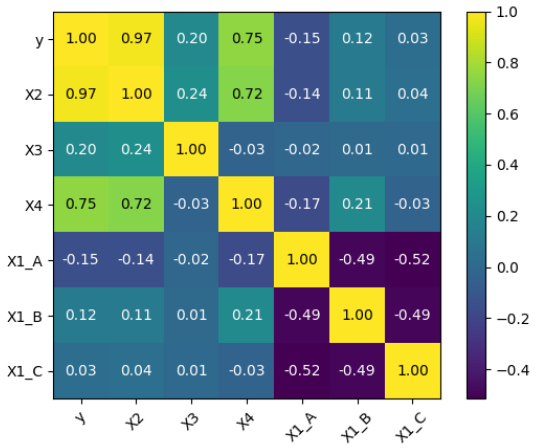
2-(b) Scatterplot Matrix and covariance matrix

One-hot encoding으로 만들어진 X1데이터를 제외한 X2, X3, X4, y를 scatterplot matrix를 만들어 각 feature 사이의 관계를 분석하였다.



X2, X3, X4, y의 feature data를 이용하여 scatterplot matrix를 만들어 분석한 결과 X2와 y는 확연하게 양의 상관관계를 가지는 것을 확인할 수 있었으며, X4와 y 또한 어느정도 양의 상관관계를 가지는 것을 확인할 수 있었다. X3와 y의 관계에선 scatterplot matrix에서 상관관계를 확인하기 힘들었다.

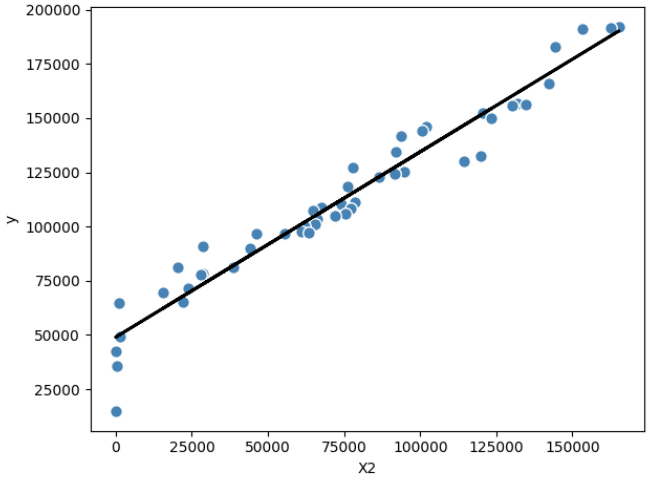
다음으로 X1을 포함하여 covariance matrix를 만들어 분석하였다.



scatterplot matrix에서도 확인할 수 있듯, X2와 y의 경우 1에 가까운 양의 상관관계를 가지고 있었으며, X4는 y와 0.75, X3는 0.2의 관계를 가졌다. X1의 특성과 y는 A, B, C 각각 -0.15, 0.12, 0.03으로 낮은 수준의 관계를 가지는 것을 확인할 수 있었다.

2-(c) Single Linear Regression for X2

Scatterplot matrix와 covariance matrix를 통해 y와 가장 높은 수치의 관계를 가지는 X2를 이용하여 single linear regression을 수행한 결과는 다음과 같았다.

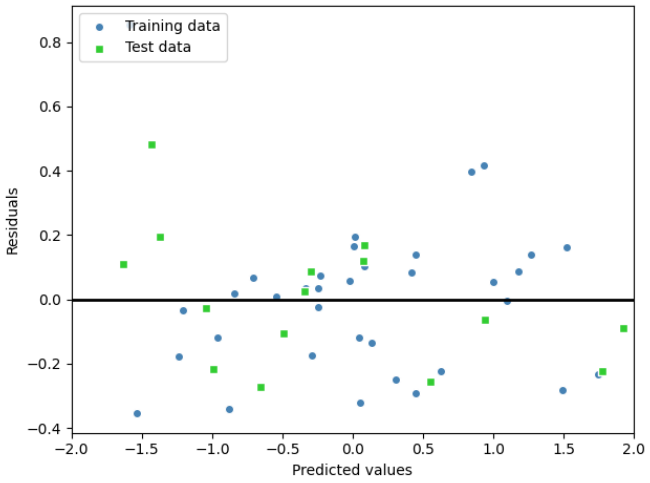



```
Linear Regression for X2
Slope(X2-y): 0.854
Intercept(X2-y): 49032.899
```

그래프를 살펴보면, X2와 y는 linear하게 연관되어 있음을 확인할 수 있으며, outlier의 수도 적다는 것을 알 수 있었다.

2-(d) Multiple Linear Regression

마지막으로 모든 feature을 사용하여 multiple linear regression을 하였다. 먼저 X1, X2, X3, X4의 값이 모두 수의 자리수가 다르므로, 표준화를 진행한 후 시행하였다.




```
Multiple Linear Regression
MSE train: 0.058, test: 0.040
R^2 train: 0.930, test: 0.971
```

Multiple Linear Regression을 시행한 결과 residual은 최대 0.5, 최소 -0.4 정도의 값을 가지는 것을 알 수 있었고, MSE는 0.040, R^2 score는 0.971의 값을 가졌다. 표준화하였기 때문에 MSE의 수치가 작은 것은 큰 의미가 없으며, 다만, train data의 MSE보다 작은 것으로

regression이 적절했다는 것을 알 수 있다. R^2 score의 경우 0.971로 1의 가깝게 나타났으며, train data의 R^2 score와 비슷하면서 조금 더 높게 나타난 것으로 보아 regression이 잘 수행되었다는 것을 알 수 있었다.

Multiple Linear Regression의 경우 선형 조합으로 이루어지기 때문에 한 feature data가 target data와 연관성이 높아, feature이 target의 특성을 잘 나타낼 수 있으면 매우 효과적이다. 본 문제의 경우 X2와 y의 상관관계가 높았고, outlier의 수도 적었기 때문에 multiple linear regression이 잘 수행되었다고 볼 수 있다.

3. Unsupervised Learning: Clustering

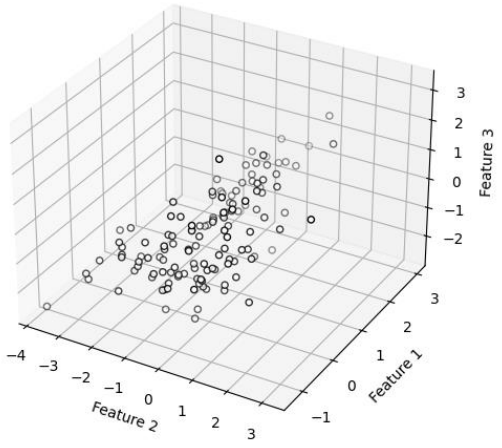
Clustering analysis는 unsupervised learning으로 특정한 target이 주어지지 않았을 때 숨겨진 구조를 찾는 것이 목적으로, 비슷한 특성을 가지는 집합으로 data를 구분하는 방법이다. Clustering 알고리즘에는 K-mean 알고리즘과 DBSCAN 알고리즘, Hierarchical 알고리즘 등이 있다. Clustering을 검증하는 방법에는 Elbow method와 Silhouette analysis가 있다.

C_table data는 target variable이 없는 11개의 표준화된 feature로 이루어진 dataset으로 숨겨진 구조를 찾는 unsupervised learning이 필요하다. C_table의 머리 부분은 다음과 같았다.

	0	1	2	...	8	9	10
0	1.533475	-0.544264	0.218697	...	0.170270	0.319781	1.915054
1	0.158598	-0.481456	-0.851553	...	-0.379938	0.361800	1.208533
2	0.105205	0.038947	1.104422	...	0.187737	0.277762	0.896033
3	1.720352	-0.328924	0.477034	...	1.113485	-0.436563	1.276468
4	0.211991	0.245314	1.842526	...	-0.406139	0.319781	0.569947

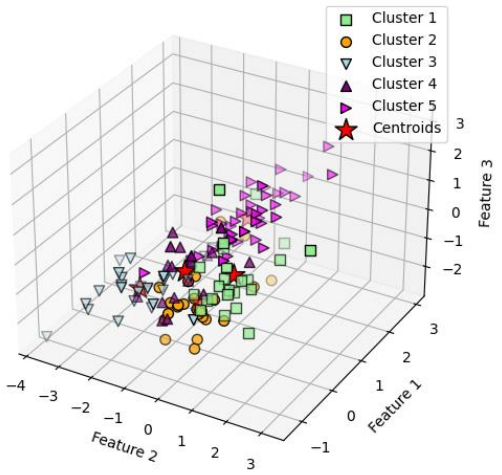
[5 rows x 11 columns]

11개의 feature가 있으므로, 그래프에 나타내기 위하여 feature 1, 2, 3을 사용하여 3차원으로 그래프에 나타내면 다음과 같다.



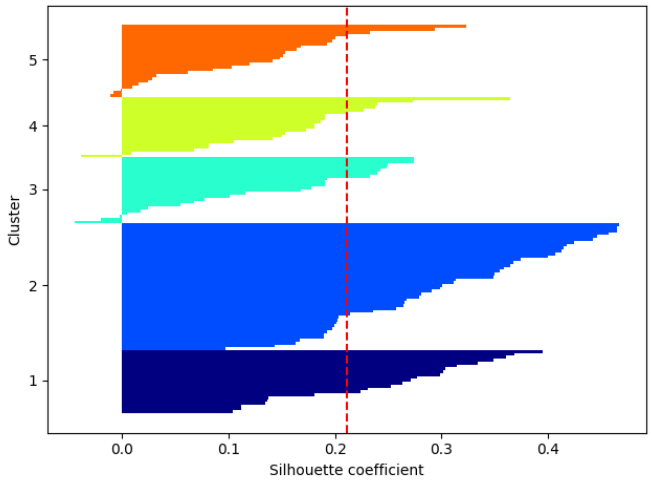
3-(a) K-means 알고리즘 (k=5)

먼저 적절한 k 값을 찾기 전에 k=5정도로 K-means 알고리즘을 사용하여 clustering해 본 결과 다음과 같은 결과를 얻을 수 있었다.



Distortion: 639.53

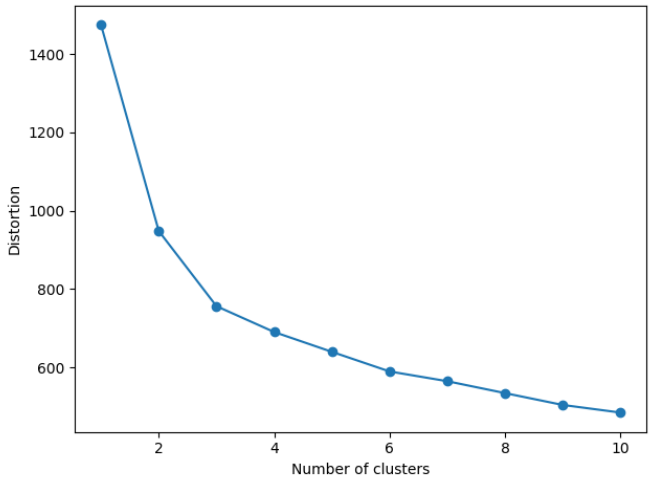
11개의 feature로 구성되어 있기 때문에 3차원으로 차원을 감소시켜서 보게 되면 정확하게 분류되었는지 확인하기 어렵기 때문에 silhouette plot을 그려 분류가 잘 되었는지 확인해 보았다.



Silhouette plot을 보면 cluster 간의 coefficient 크기와 cluster의 요소 너비의 편차가 심한 것으로 보아 상대적으로 clustering이 좋지 않았다는 것을 알 수 있었다.

3-(b) Elbow method

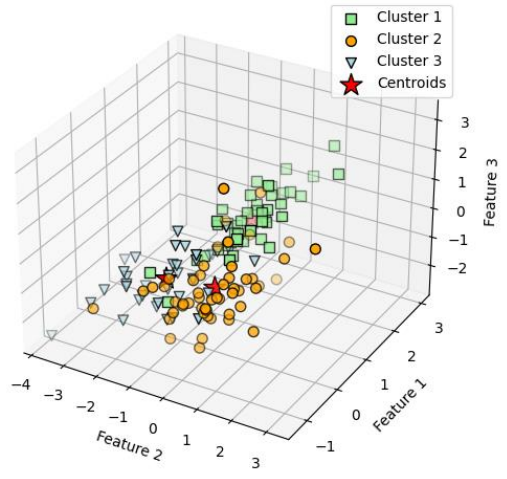
적절한 cluster의 개수를 찾기 위하여 각 k값에 따른 distortion을 plot하여 elbow method를 사용해 보았다.



Distortion 값은 k=3까지는 급하게 줄어들기 시작하여 k=4이후로는 상대적으로 선형적으로 줄어드는 것을 확인할 수 있었다. 따라서 cluster의 개수를 증가시키면 distortion을 줄일 수 있지만, 과도하게 clustering을 시키게 되므로 k=3이 적절한 값이라고 판단했다.

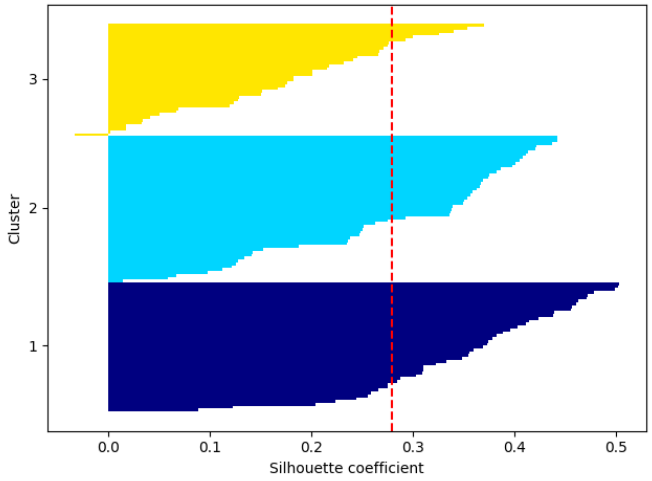
3-(c) K-means 알고리즘 (k=5)

k=3으로 K-means 알고리즘을 사용하여 clustering한 결과는 다음과 같다.




```
Distortion: 756.69
```

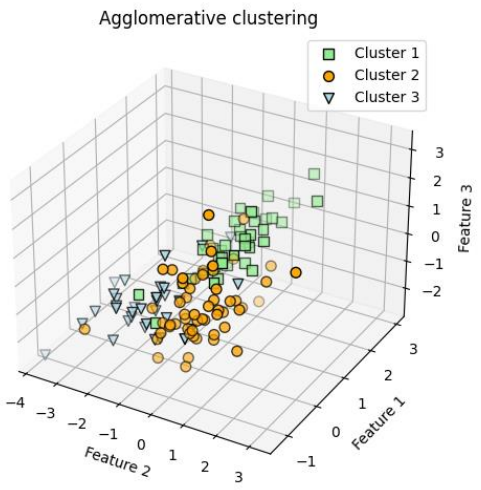
Distortion 값은 증가하였지만, 그래프를 보면 조금 더 깔끔하게 clustering이 완료된 것을 확인할 수 있었다. k=3에 대한 silhouette plot은 다음과 같다.



Cluster간의 길이 차이는 여전히 있었지만 k=5일 때보다 줄어든 것을 확인할 수 있었고, cluster간의 너비차이 또한 k=5에 비해 줄어든 것을 확인할 수 있었다. 따라서 k=3이 k=5일 때보다 적절하다고 볼 수 있다.

3-(d) Agglomerative hierarchical clustering (k=3)

Agglomerative hierarchical clustering은 데이터의 개수만큼의 cluster를 생성한 뒤 지정한 cluster의 개수만큼 남을 때까지 근접한 cluster끼리 서로 합치는 방법이다.

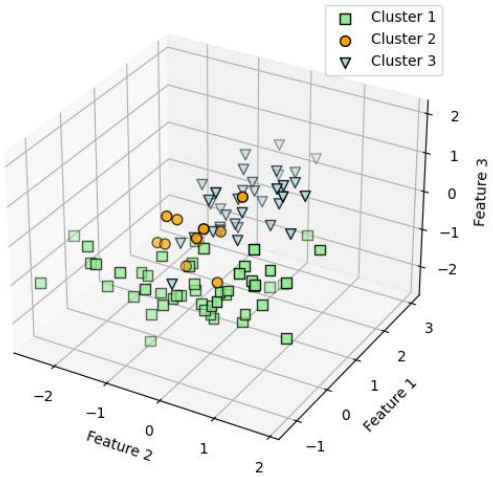


Agglomerative hierarchical clustering을 시행한 결과 K-means 알고리즘을 사용할 때의 결과와 유사하게 나온 것을 확인할 수 있었다. 두 알고리즘 모두 거리를 통해 cluster를 생성하는 방법이기 때문에, 유사한 결과를 나타내는 것이라고 생각할 수 있다.

3-(e) DBSCAN 알고리즘

DBSCAN 알고리즘은 거리를 이용하는 위 두 알고리즘과 다르게 밀도를 이용하여 cluster를 형성한다. 따라서 넓게 퍼져있는 두 종류의 data가 있을 때, 두 세트의 데이터가 가까운 부분이 있을 때, 거리 기반 알고리즘은 그 데이터들을 같은 cluster로 묶기 쉽지만, 밀도 기반 알고리즘은 잘 분류할 수 있다.

C_table의 데이터에 DBSCAN을 적용한 결과는 다음과 같다.



DBSCAN을 이용한 결과 이전의 알고리즘을 이용하였을 때보다 cluster로 분류되지 않은 데이터가 많은 것을 확인할 수 있었는데 이는 여러 개의 sample을 알고리즘이 noisy sample(-1)로 분류하여 clustering 하지 않았다는 것을 알 수 있었다. 분류된 결과를 살펴보니 다음과 같았다. (eps=2, min_samples=5)

```
[ 0  0 -1  0  0  0  0  0  0  0  0  0  0 -1 -1  0  0  0  0  0  0  0  0  0
  0 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0 -1  0 -1 -1 -1
  0 -1  1  0 -1  0 -1 -1  1 -1 -1 -1 -1 -1  0  1 -1 -1  0  0  1  2 -1  0
  1  1  1  1  1  1  1  0 -1 -1 -1  0  0 -1  0 -1 -1 -1  2  2  2  2 -1  2
 -1 -1  2  2  2  2  2  2  2 -1  2  2  2  2  2  2 -1  2 -1  2  2 -1 -1 -1
  2  2  2  2  2  2  2  2 -1  2  2  2  2  2]
```

eps값을 1.5로 낮추게 되면, 다음과 같았다.

```
[ 0 -1 -1 -1 -1 -1  0 -1 -1  0  0 -1 -1 -1 -1 -1 -1 -1 -1  0 -1  0  1
  1 -1 -1 -1 -1  0 -1 -1 -1 -1  1  1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
```

eps값을 3으로 증가시킨 결과는 다음과 같았다.

```
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
```

DBSCAN 알고리즘을 사용할 때에는 eps값을 증가시키면, cluster의 개수가 적어져 매우 높은 eps값에서는 1개의 cluster만 생성하게 되고, eps값을 감소시키면, noisy sample로 판단해 clustering하지 않는 것을 알 수 있었다. 따라서 DBSCAN 알고리즘을 사용하려면 적절한 eps값을 설정하여, 잡음이 많이 들어간 sample은 noisy sample로 분류하고, 잡음이 들어가지 않은 sample은 noisy sample로 분류하지 않도록 하여야 한다.