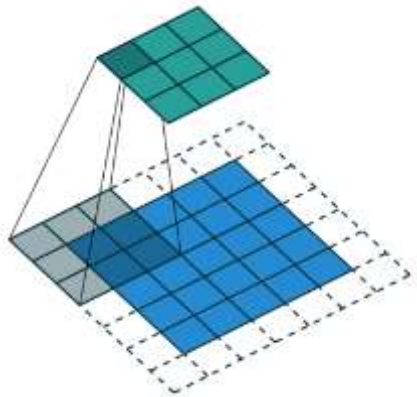
오토인코더

```
(x_train, _), (x_test, _) = cifar10.load_data()
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.

encoder = models.Sequential([
    layers.Input(shape=(32, 32, 3)),
    layers.Conv2D(32, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2), padding='same'),
    layers.Conv2D(16, (3, 3), activation='relu', padding='same'),
    layers.MaxPooling2D((2, 2), padding='same'),
])
decoder = models.Sequential([
    layers.Conv2DTranspose(16, (3, 3), activation='relu', padding='same', strides=(2, 2)),
    layers.Conv2DTranspose(32, (3, 3), activation='relu', padding='same', strides=(2, 2)),
    layers.Conv2D(3, (3, 3), activation='sigmoid', padding='same')
])

autoencoder = models.Sequential([encoder, decoder])
autoencoder.compile(optimizer='adam', loss='mse',metrics=['acc'])
autoencoder.fit(x_train, x_train,epochs=10,batch_size=16,
    validation_data=(x_test, x_test))

encoded_imgs = encoder.predict(x_test)
decoded_imgs = decoder.predict(encoded_imgs)
```

특징 추출

이미지 복원



결과

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_1 (Conv2D)	(None, 16, 16, 16)	4,624
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 16)	0

Total params: 5,520 (21.56 KB)
Trainable params: 5,520 (21.56 KB)
Non-trainable params: 0 (0.00 B)
None

→ 인코더

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_transpose (Conv2DTranspose)	(None, 16, 16, 16)	2,320
conv2d_transpose_1 (Conv2DTranspose)	(None, 32, 32, 32)	4,640
conv2d_2 (Conv2D)	(None, 32, 32, 3)	867

Total params: 7,827 (30.57 KB)
Trainable params: 7,827 (30.57 KB)
Non-trainable params: 0 (0.00 B)
None
(10000, 8, 8, 16)
(10000, 32, 32, 3)

→ 디코더

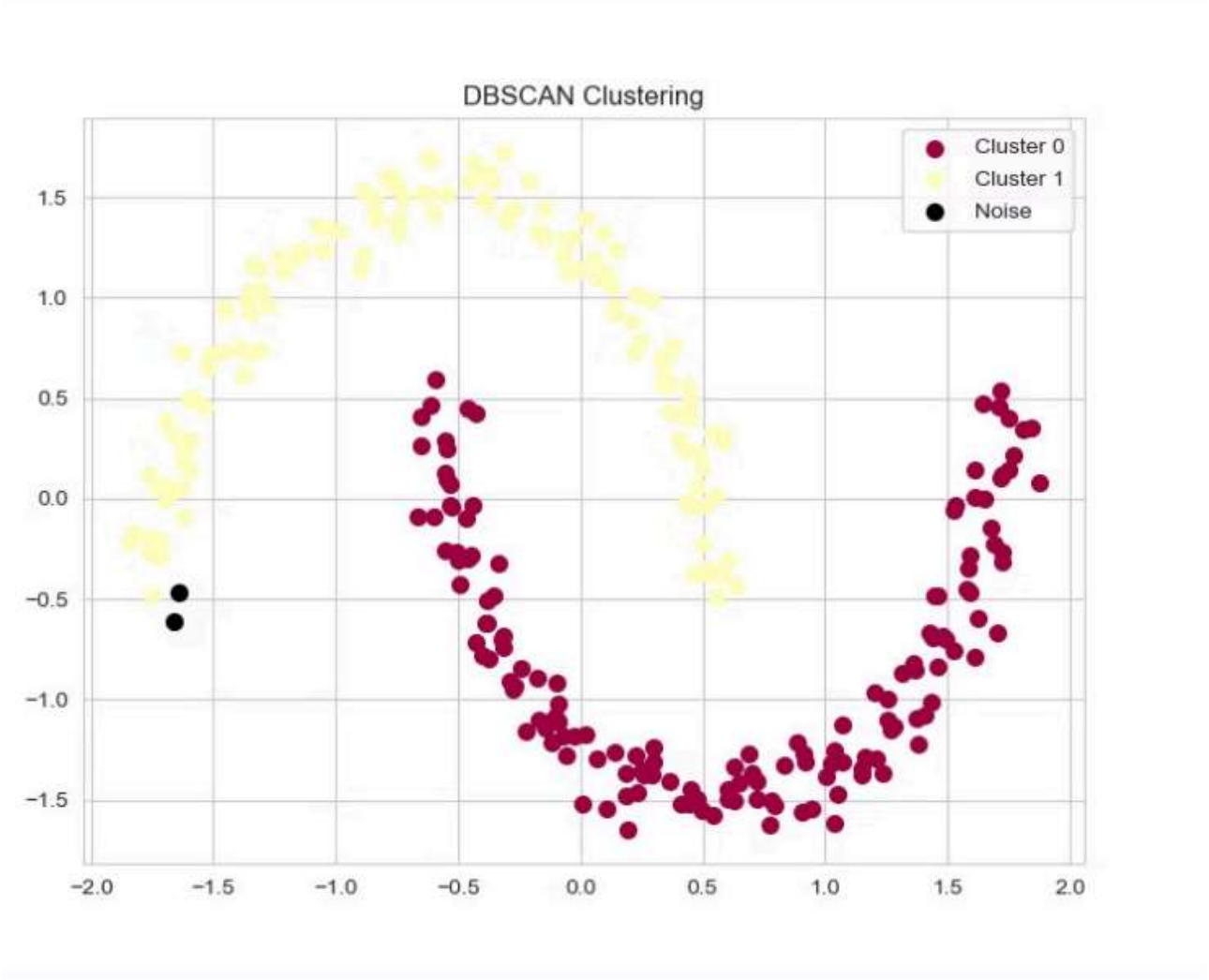
→ 원본

→ 복원 결과

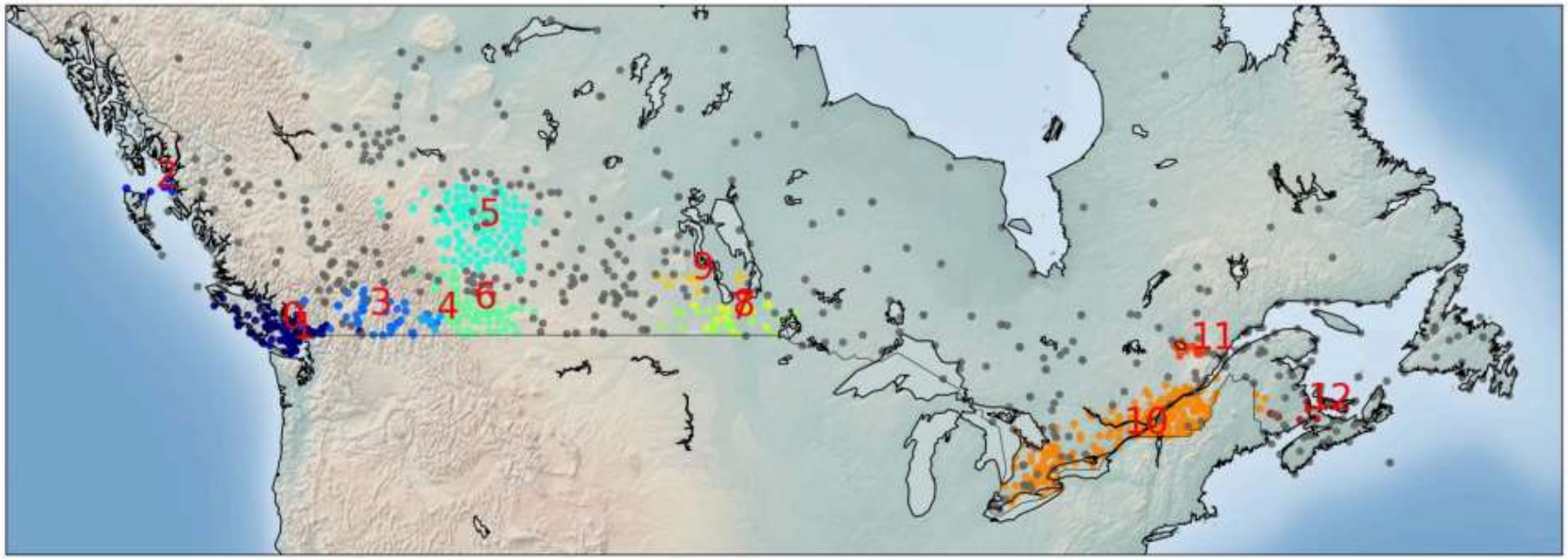
DBSCAN 실습 Task1 두 개의 반달 모양의 데이터 셋 Clustering

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

X, _ = make_moons(n_samples=300, noise=0.05, random_state=42) #반달 모양 데이터 생성

scaler = StandardScaler()#데이터 정규화
X_scaled = scaler.fit_transform(X)

dbscan = DBSCAN(eps=0.2, min_samples=5)#eps: min_samples: 클러스터를 형성하기 위해 최소 5개의 데이터 필요
labels = dbscan.fit_predict(X_scaled)#예측값 labels에 저장

unique_labels = set(labels)#labels에 있는 값들 추출 (-1(이상치),0(0번 클래스),1(1전 클래스))
colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))]#각 label 별 색깔 지정 0 - 빨강 , 1 -노랑
print(colors)
plt.figure(figsize=(8, 6))
for k, col in zip(unique_labels, colors):
    if k == -1:#이상치는 검정색으로 지정
        col = [0, 0, 0, 1]
    class_member_mask = (labels == k)
    xy = X_scaled[class_member_mask]
    plt.scatter(xy[:, 0], xy[:, 1], c=[col], label=f"Cluster {k}" if k != -1 else "Noise", s=50)

plt.title("DBSCAN Clustering")
plt.legend()
plt.show()

```

DBSCAN 실습 결과


DBSCAN Clustering
Cluster 0
Cluster 1
Noise
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

- 특정 지역의 날씨 정보 클러스터링



1. 밀도(반경 내의 데이터의 개수) 낮은 곳은 이상치로 분류하여 어떤 클래스에도 분류하지 않음
2. 다양한 형태의 클러스터를 발견 가능하며, 복잡한 데이터 구조 분석에 적합

DBSCAN 실습 코드

```
import numpy as np
import pandas as pd
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

file = r'C:/Users/dahae/machine learning/ch12_code/weather-stations20140101-20141231.csv'
pdf = pd.read_csv(file)
pdf.head(5)

#데이터 필터링 : 지도에서 long(경도) -140 ~ 40 , lat(위도) -50 ~ 60 사이로 데이터 필터링
llon = -140; llat = 40; ulon = -50; ulat = 60
pdf = pdf[(pdf['Long'] > llon) & (pdf['Long'] < ulon) & (pdf['Lat'] > llat) & (pdf['Lat'] < ulat)]

#지도를 그리는 라이브러리(Basemap)
my_map = Basemap(projection='merc',
                 resolution='l',area_thresh=1000.0,
                 llcrnrlon=llon,llcrnrlat=llat, # min longitude, min latitude
                 urcrnrlon=ulon,urcrnrlat=ulat) # max longitude, max latitude
my_map.drawcoastlines() #해안선 그림
my_map.drawcountries() # 국가 경계 그림
my_map.fillcontinents(color='white',alpha=0.3) #대륙을 흰색으로 채움
my_map.shadedrelief() # 지형의 음영 추가

a = np.asarray(pdf.Long)#csv 파일에서 읽은 Long
b = np.asarray(pdf.Lat)
xs, ys = my_map(a,b) #Basemap을 이용해서 지도 좌표로 변환된 x,y 값
#csv 파일 맨끝에 xm,ym 추가
pdf.loc[:,'xm'] = xs
pdf.loc[:,'ym'] = ys

#clustering 데이터
Clus_dataset = pdf[['xm','ym','Tm']]
Clus_dataset = np.nan_to_num(Clus_dataset) # 결측치는 0으로 대체
Clus_dataset = StandardScaler().fit_transform(Clus_dataset)#정규화
```

DBSCAN 실습 코드

```
38
39  db = DBSCAN(eps=0.15, min_samples=10)#eps: min_samples: 클러스터를 형성하기 위해 최소 10개의 데이터 필요
40  db.fit(Clus_dataset)
41
42  core_samples_mask = np.zeros_like(db.labels_,dtype=bool)#labels 개수만큼 배열 만들어 False로 초기화
43  core_samples_mask[db.core_sample_indices_] = True#코어 포인트라고 판단되는 포인트만 배열에서 True로 설정
44
45  labels = db.labels_
46  pdf.loc[:,'Clus_Db'] = labels#csv파일에 Clus_Db열 추가하여 labels 추가
47  realClusterNum = len(set(labels)) - (1 if -1 in labels else 0)#노이즈(-1) 제거
48  clusterNum = len(set(labels)) # 14
49
50  colors = plt.get_cmap('jet')(np.linspace(0.0, 1.0, clusterNum))#14개에 색상 지정
51
52
```

DBSCAN 실습 결과