# 1. 프로그램 4-4를 실행하여 결과를 정리하고, 코드를 설명하시오.

from sklearn.datasets import fetch\_openml from sklearn.neural\_network import MLPClassifier import matplotlib.pyplot as plt import numpy as np

### #mnist 데이터 가져옴

mnist=fetch\_openml('mnist\_784') mnist\_data=mnist\_data/255.0 #[0.255]의 값을 가진 데이터를 255로 |

mnist.data=mnist.data/255.0 #[0,255]의 값을 가진 데이터를 255로 나눔으로써 [0,1]로 정규화한다.

x\_train=mnist.data[:60000]; #학습데이터를 60000개로 설정한다.

x\_test=mnist.data[60000:] #테스트 데이터는 10000개가 된다.

y\_train=np.int16(mnist.target[:60000]); y\_test=np.int16(mnist.target[60000:])

#np.int16을 적용해서 str형을 16비트 정수형으로 변환(혼동 행렬에서 계산하기 위해 필수로 수행해야하는 부분)

#### # MLP 분류기 모델을 학습

mlp=MLPClassifier(hidden\_layer\_sizes=(100),learning\_rate\_init=0.001,batch\_size=512,max\_iter=30 0,solver='adam',verbose=True) #은닉층 100, 배치크기는 512로 설정한다. mlp.fit(x\_train,y\_train) #학습을 진행한다.

# 테스트 집합으로 예측

res=mlp.predict(x\_test) #테스트 데이터로 예측한다.

## #혼동행렬

conf=np.zeros((10,10),dtype=np.int16) #혼동 행렬 계산한다. for i in range(len(res)):
 conf[res[i]][y test[i]]+=1

conf[res[i]][y\_test[i]]+=1
print(conf)

#정확률계산

no\_correct=0

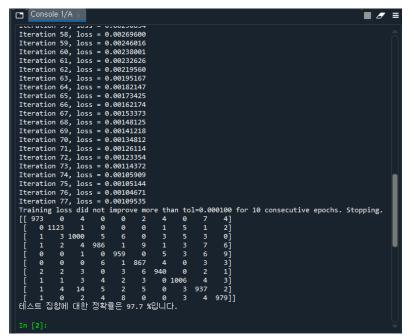
for i in range(10):

no correct+=conf[i][i]

accuracy=no correct/len(res) # 정확도를 계산한다.

print("테스트 집합에 대한 정확률은", accuracy\*100, "%입니다.")

# 결과



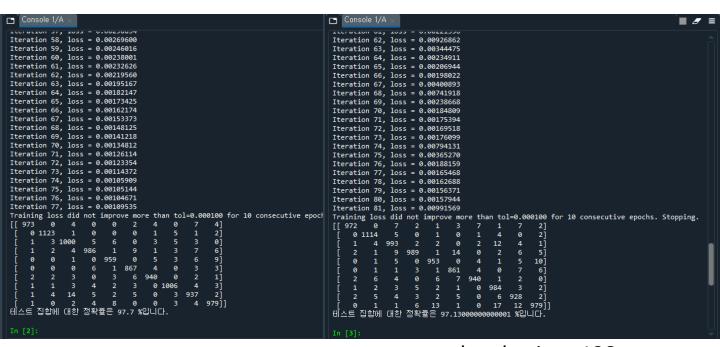
결과 정리

- 77번 학습함.
- 최종 loss는 0.00109535
- 정확도 97.7%

2. btch\_size를 128로 하고, 은닉층 사이즈를 50으로한 수행결과를 비교하시오.

정확도 차이 97.7%에서 97.13%로 약간 감소하였으나, 큰 변화는 아니다. 은닉층과 batch\_size가 줄었으나 횟수는 조금 더 많아 졌다.

따라서 batch\_size와 은닉층의 수는 정확도에 영향을 준다.



batch\_size : 512 은닉층 : 100 batch\_size : 128 은닉층 : 50