11주차 과제

과목명 딥러닝 실제
담당교수 전명근 교수님
학과 산업인공지능학과
학번 2021254009
이름 정원용

1. 프로그램 4-4를 수행하여 결과를 정리하고, 프로그램의 동작을 설명하시오.

(1) 수행 결과

```
from sklearn.datasets import fetch_openml
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt
import numpy as np

# MNIST 데이터셋을 읽고 훈련 집합과 테스트 집합으로 분할
mnist=fetch_openml('mnist_784')
# sklearn 라이브러리에서 제공하는mnist 데이터셋70,000 건을 사용한다.
# 학습 데이터로60,000 건, 테스트 데이터10,000 건을 사용한다.
# 0~255 범위의 값을 가진mnist 데이터를0~1로 정규화한다.
# 혼동행렬(confusion matrix) 사용을 위해int 형으로 변환한다.
mnist.data=mnist.data/255.0
x_train=mnist.data[:60000]; x_test=mnist.data[60000:]
y_train=np.int16(mnist.target[:60000]); y_test=np.int16(mnist.target[60000:])

# MLP 분류기 모델을 학습
# 은닉층=100,학습률 초기값=0.001,미니배치=512,에포크횟수=300,알고리즘종류=Adam
mlp=MLPClassifier(hidden_layer_sizes=(100),learning_rate_init=0.001,batch_size=512,max_iter=300,solver='adam',verbose=True)
mlp.fit(x_train,y_train)

# 테스트 집합으로 예측
res=mlp.predict(x_test)

# 예측 데이터로 혼동행렬을 수행하고 시각화한다.
conf=np.zeros((10,10),dtype=np.int16)
for i in range(len(res)):
conf[res[i]][y_test[i]]+=1
print(conf)

# 정확률 계산한다.
no_correct=0
for i in range(10):
no_correct+=conf[i][i]
accuracy=no_correct/len(res)
print("테스트 집합에 대한 정확률은", accuracy*100, "%입니다.")
```

```
Iteration 1, loss = 0.61869145
Iteration 2, loss = 0.27311149
Iteration 3, loss = 0.21658838
Iteration 4, loss = 0.17986618
Iteration 5, loss = 0.15372141
Iteration 6, loss = 0.13446038
Iteration 7, loss = 0.11931440
Iteration 8, loss = 0.10668082
Iteration 9, loss = 0.09627495
Iteration 10, loss = 0.08746653
Iteration 11, loss = 0.08046424
Iteration 12, loss = 0.07375436
Iteration 13, loss = 0.06748929
Iteration 14, loss = 0.06231906
Iteration 15, loss = 0.05723496
Iteration 16, loss = 0.05430583
Iteration 17, loss = 0.05035027
Iteration 18, loss = 0.04580140
Iteration 19, loss = 0.04298031
Iteration 20, loss = 0.04046481
Iteration 21, loss = 0.03697035
Iteration 22, loss = 0.03489111
Iteration 23, loss = 0.03255138
Iteration 24, loss = 0.03089301
Iteration 25, loss = 0.02869538
Iteration 26, loss = 0.02685998
Iteration 27, loss = 0.02489361
Iteration 28, loss = 0.02282457
Iteration 29, loss = 0.02213260
Iteration 30, loss = 0.02053330
Iteration 31, loss = 0.01849150
Iteration 32, loss = 0.01736802
Iteration 33, loss = 0.01625689
Iteration 34, loss = 0.01536532
Iteration 35, loss = 0.01418604
Iteration 36, loss = 0.01337904
Iteration 37, loss = 0.01255261
Iteration 38, loss = 0.01180737
Iteration 39, loss = 0.01095037
Iteration 40, loss = 0.01023534
Iteration 41, loss = 0.01023404
Iteration 42, loss = 0.00874556
Iteration 43, loss = 0.00841599
Iteration 44, loss = 0.00762603
Iteration 45, loss = 0.00725492
Iteration 46, loss = 0.00659978
Iteration 47, loss = 0.00617619
Iteration 48, loss = 0.00590602
Iteration 49, loss = 0.00530728
Iteration 50, loss = 0.00488251
Iteration 51, loss = 0.00481302
Iteration 52, loss = 0.00458059
Iteration 53, loss = 0.00436427
Iteration 54, loss = 0.00392399
Iteration 55, loss = 0.00370021
Iteration 56, loss = 0.00341059
Iteration 57, loss = 0.00322765
Iteration 58, loss = 0.00305502
Iteration 59, loss = 0.00285552
Iteration 60, loss = 0.00255736
Iteration 61, loss = 0.00245688
Iteration 62, loss = 0.00233571
Iteration 63, loss = 0.00220671
Iteration 64, loss = 0.00214245
Iteration 65, loss = 0.00210939
Iteration 66, loss = 0.00189860
Iteration 67, loss = 0.00172919
Iteration 68, loss = 0.00165288
Iteration 69, loss = 0.00157545
Iteration 70, loss = 0.00151083
Iteration 71, loss = 0.00138083
Iteration 72, loss = 0.00149018
Iteration 73, loss = 0.00124626
Iteration 74, loss = 0.00134709
Iteration 75, loss = 0.00117805
Iteration 76, loss = 0.00110989
Iteration 77, loss = 0.00108824
Iteration 78, loss = 0.00100094
Iteration 79, loss = 0.00093561
Iteration 80, loss = 0.00087920
Iteration 81, loss = 0.00084042
Iteration 82, loss = 0.00080395
Iteration 83, loss = 0.00076736
Iteration 84, loss = 0.00078878
```

```
[[ 969    0    4    0    1    1    4    0    5    2]
 [   0 1122    2    0    0    0    3    4    0    2]
 [   2    4 1006    6    4    0    0    9    2    0]
 [   1    1    3  985    0    9    1    6    7    5]
 [   1    0    2    0  962    2    5    0    5   10]
 [   1    1    0    4    0  862    3    0    3    3]
 [   2    1    3    0    3    9  941    0    0    1]
 [   1    1    6    6    1    2    0 1000    5    5]
 [   2    5    5    2    0    4    1    3  943    4]
 [   1    0    1    7   11    3    0    6    4  977]]
테스트 집합에 대한 정확률은 97.67 %입니다.
```

(2) 동작 설명

- sklearn 라이브러리에서 제공하는 mnist 데이터셋 70,000 건을 사용한다.
- 학습 데이터로 60,000 건, 테스트 데이터 10,000 건을 사용한다.
- 0~255 범위의 값을 가진 mnist 데이터를 0~1로 정규화한다.
- 혼동행렬(confusion matrix) 사용을 위해 int 형으로 변환한다.
- MLP 분류기 모델을 생성한다.
  (은닉층=100,학습률 초기값=0.001,미니배치=512,에포크횟수=300,알고리즘종류=Adam)
- 학습을 수행하고 테스트 데이터로 예측한다.
- 예측 데이터로 혼동행렬을 수행하고 시각화한다.
- 정확도를 계산한다.

2. batch size를 128로 하고, 은닉층 사이즈를 50인 경우에 수행하여 결과를 비교하시오.

hidden_layer_sizes=100 batch_size=512	hidden_layer_sizes=50 batch_size=128
Iteration 91, loss = 0.00067412 Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping. [[969 0 4 1 2 3 4 0 4 1] [0 1124 1 0 0 1 3 5 2 2] [2 5 1009 6 2 0 2 9 2 0] [1 0 3 986 1 8 1 2 5 4] [1 0 2 0 960 1 5 0 6 9] [0 1 0 5 0 866 4 0 2 3] [2 2 1 0 4 4 938 0 2 0] [1 1 5 3 3 1 0 1004 2 2] [3 2 6 5 1 4 1 4 943 3] [1 0 1 4 9 4 0 4 6 985]] 테스트 집합에 대한 정확률은 97.84 %입니다.	Iteration 86, loss = 0.00399856 Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping. [[968 0 5 1 2 4 8 1 7 5] [0 1124 6 0 1 1 1 4 0 2] [1 4 987 5 3 1 1 11 4 0] [0 1 8 985 1 17 1 5 11 3] [1 0 7 1 949 2 3 1 5 7] [0 0 0 3 4 850 5 0 3 5] [4 2 4 0 5 6 937 0 7 0] [2 1 8 6 2 3 0 998 4 7] [3 3 6 6 2 5 2 2 928 4] [1 0 1 3 13 3 0 6 5 976]] 테스트 집합에 대한 정확률은 97.02 %입니다.

batch size를 증가시킬수록 정확도는 높아지나, 지난치게 클 경우 정확도가 떨어지는 현상이 발생함.