과제 #2

- [프로그램 4-5] **코드 수정 + 가중치 시각화**

프로그램 4-5 validation_curve 함수로 최적의 은닉 노드 개수 찾기

```
01  from sklearn import datasets
02  from sklearn.neural_network import MLPClassifier
03  from sklearn.model_selection import train_test_split,validation_curve
04  import numpy as np
05  import matplotlib.pyplot as plt
06  import time
07
08  # 데이터셋을 읽고 훈련 집합과 테스트 집합으로 분할
09  digit=datasets.load_digits()
10  x_train,x_test,y_train,y_test=train_test_split(digit.data,digit.target,train_size=0.6)
11
```

훈련:테스트를 6:4로 분할

과제 #2

20에서 시작하여 10씩 증가시키면서 81까지 조사

Epoch = 50 (빠른 학습 위함)

```
12  # 다층 퍼셉트론을 교차 검증으로 성능 평가 (소요 시간 측정 포함)
13  start=time.time()    # 시작 시각
14  mlp=MLPClassifier(learning_rate_init=0.001,batch_size=32,max_iter=300,solver='sgd')
15  prange=range(50,1001,50)
16  train_score,test_score=validation_curve(mlp,x_train,y_train,param_name="hidden_
    layer_sizes",param_range=prange,cv=10,scoring="accuracy",n_jobs=4)
17  end=time.time()      # 끝난 시각
18  print("하이퍼 매개변수 최적화에 걸린 시간은",end-start,"초입니다.")
19
20  # 교차 검증 결과의 평균과 분산 구하기
21  train_mean = np.mean(train_score,axis=1)
22  train_std = np.std(train_score,axis=1)
23  test_mean = np.mean(test_score,axis=1)
24  test_std = np.std(test_score,axis=1)
25
26  # 성능 그래프 그리기
27  plt.plot(prange,train_mean,label="Train score",color="r")
28  plt.plot(prange,test_mean,label="Test score",color="b")
29  plt.fill_between(prange,train_mean-train_std,train_mean+train_std,alpha=0.2,color="r")
30  plt.fill_between(prange,test_mean-test_std,test_mean+test_std,alpha=0.2,color="b")
31  plt.legend(loc="best")
32  plt.title("Validation Curve with MLP")
33  plt.xlabel("Number of hidden nodes"); plt.ylabel("Accuracy")
34  plt.ylim(0.9,1.01)
35  plt.grid(axis='both')
36  plt.show()
```

코어개수는 자유롭게

5-겹 교차 검증으로 성능 측정 (빠른 학습 위함)

과제 #2

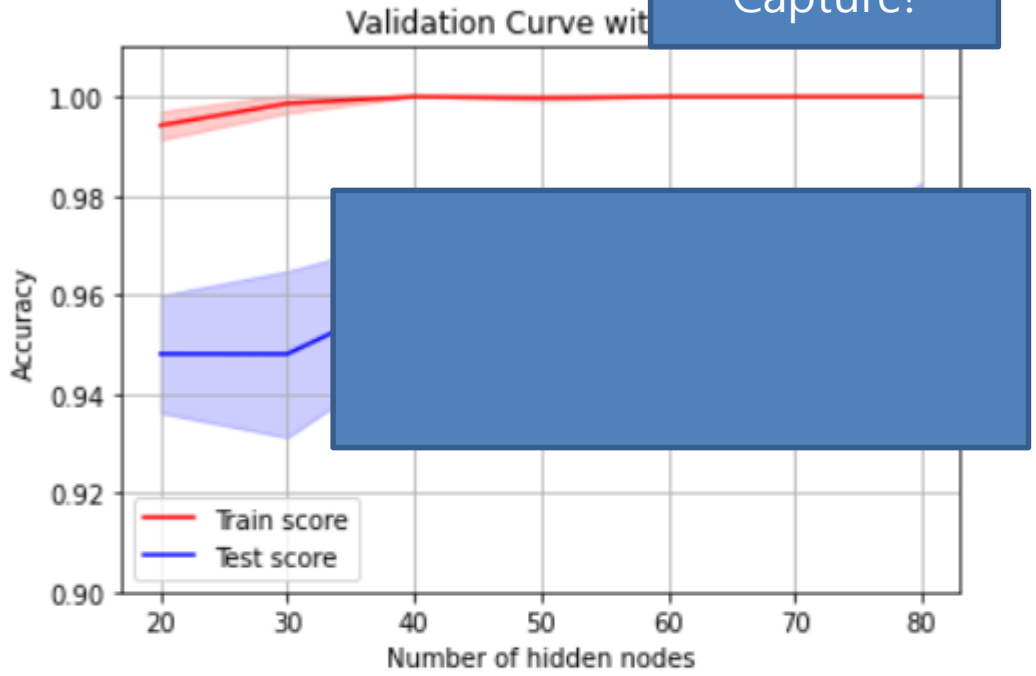
```
37
38  best_number_nodes=prange[np.argmax(test_mean)]     # 최적의 은닉 노드 개수
39  print("\n최적의 은닉층의 노드 개수는",best_number_nodes,"개입니다.\n")
40
41  # 최적의 은닉 노드 개수로 모델링
42  mlp_test=MLPClassifier(hidden_layer_sizes=(best_number_nodes),learning_rate_
    init=0.001,batch_size=32,max_iter=300,solver='sgd')
43  mlp_test.fit(x_train,y_train)
44
45  # 테스트 집합으로 예측
46  res=mlp_test.predict(x_test)
47
48  # 혼동 행렬
49  conf=np.zeros((10,10))
50  for i in range(len(res)):
51      conf[res[i]][y_test[i]]+=1
52  print(conf)
53
54  # 정확률 계산
55  no_correct=0
56  for i in range(10):
57      no_correct+=conf[i][i]
58  accuracy=no_correct/len(res)
59  print("테스트 집합에 대한 정확률은", accuracy*100, "%입니다.")
```

과제 #2

- 실행 결과

Colab GPU 스탠다드 기준 :
1분 이내 소요

하이퍼 매개변수 최적화에 걸린 시간은 Capture! 초입니다.



최적의 은닉층의 노드 개수는 Capture!

```
[[66.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. 63.  0.  0.  0.  0.  1.  0.  3.  0.]
 [ 0.  0. 69.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  0.  1. 61.  0.  1.  0.  1.  3.  1.]
 [ 1.  0.  0.  0. 79.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0. 60.  0.  0.  0.  1.]
 [ 0.  0.  0.  0.  0.  1. 75.  0.  1.  0.]
 [ 0.  0.  0.  0.  0.  1.  0. 66.  1.  0.]
 [ 0.  2.  0.  3.  1.  0.  0.  0. 72.  3.]
 [ 0.  0.  0.  0.  0.  3.
```

테스트 집합에 대한 정확률은 Capture! %입니다.

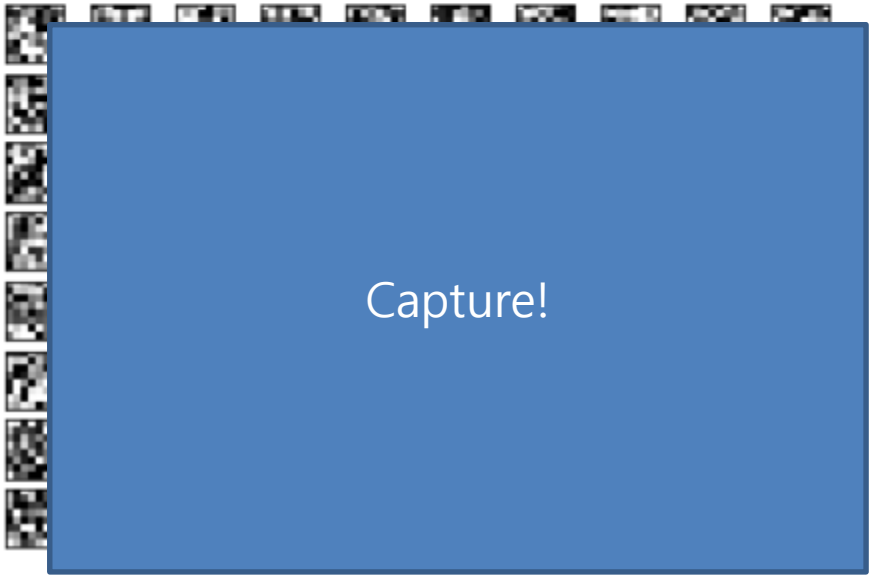
과제 #2

가중치 시각화 코드 추가 ➔

```
row = int(best_number_nodes / 10)
col = 10
fig, axes = plt.subplots(row, col)

# use global min / max to ensure all weights are shown on the same scale
vmin, vmax = mlp_test.coefs_[0].min(), mlp_test.coefs_[0].max()
for coef, ax in zip(mlp_test.coefs_[0].T, axes.ravel()):
    ax.matshow(coef.reshape(8, 8),
               cmap=plt.cm.gray, vmin=0.5 * vmin, vmax=0.5 * vmax)
    ax.set_xticks(())
    ax.set_yticks(())

plt.show()
```



*source: https://scikit-learn.org/stable/auto_examples/neural_networks/plot_mnist_filters.html?highlight=mlp

과제 #2

- Requirements
  - **TWO FILES:**
    - 1) **Code** (or ipynb) file
    - 2) **Report (MS Word or PDF)**
  - In the report:
    - Three **results** with captured figure
      - 1) the best **hidden node number** + **accuracy**,
      - 2) **Visualization** of MLP weights
      - 3) Running time
    - **Runtime environment**
      - Versions
      - Cloud or local

※ 본 과제는 랭킹반영 안함