Assignment 3: Hyperparameter Tuning for Neural Networks

2016314364 박수헌

Spambase 데이터셋의 instance 개수는 4601개였으며, 그 중 spam은 1813개로 39.4%를, non-spam 은 2788개로 39.4%를 차지했다. 각 instance는 57개의 실수형 attribute들과 0(non-spam)과 1(spam)으로 표현되는 1개의 class를 가지고 있다. 'spambase.data' 파일에는 각 줄 당 57개의 실수 attribute와 마지막에는 1개의 0,1의 class 정보가 나열되어 있다. 따라서, 각 줄을 읽어 들이며 57개의 실수를 x list에 append했고, class 정보인 마지막 수는 y list에 append했다. 그 후, x_train, x_test, y_train, y_test로 train_test_split을 이용해 test set의 비율이 0.25%를 차지하게 분할했다. Scaling으로는 StandardScaler와 MinMaxscaler를 사용해 x_train과 x_test를 scaling해 사용했고 그에 대한 성능을 비교해 보았다.

위의 데이터셋은 classification이 필요한 데이터셋이므로 MLPClassifier를 사용했다. MLPClassifier의 주요 hyper parameter인 hidden_layer, solver, activation, alpha를 변화시켜가며 최적의 hyperparameter를 찾아내었다.

hidden_layer	450	500	550	600
train accuracy	0.9947826086956522	0.9968115942028986	0.996231884057971	0.9953623188405797
test accuracy	0.94874022589053	0.950477845351868	0.9435276375065161	0.945264986967854

Solver	Lbfgs	sgd	adam
train accuracy	0.9994202898550725	0.9542028985507246	0.9968115942028986
test accuracy	0.930495221546813	0.9374456993918332	0.950477845351868

* max_iter = 600

Activation	Relu	tanh
train accuracy	0.9968115942028986	0.9927536231884058
test accuracy	0.950477845351868	0.9365768896611643

Alpha	0.0001	0.001	0.01	0.1
train accuracy	0.9968115942028986	0.9965217391304347	0.995072463768116	0.9892753623188406
test accuracy	0.950477845351868	0.952215464813206	0.94874022589053	0.952215464813206

Scaler	StandardScaler	MinMaxScaler
train accuracy	0.9965217391304347	0.9910144927536232
test accuracy	0.952215464813206	0.950477845351868

* alpha=0.001

종합해보면, StandardScaler를 사용해 scaling한 dataset을 hidden_layer = 500, max_iter = 600, solver = 'adam', activation = 'relu', alpha = 0.001를 사용해 training한 경우에 가장 좋은 test accuracy인 0.9522를 보였다.

만약 spambase 데이터셋보다 더 복잡하고 큰 데이터셋이 주어진다면, 더 높은 max_iter의 값과, hidden_layer의 수가 필요할 것으로 예상된다. 그만큼 더 복잡한 모델을 구성해야 accuracy가 올라갈 것이기 때문이다.

```
x = list()
y = list()
f = open('spambase.data', 'r')
while True:
  data = f.readline()
  if data == '' :
    break
  data_split = data.split(',')
  if data_split[-1][0] == '1':
    y.append(1)
  else:
    y.append(0)
  data_split.pop()
  int_data = list(map(float, data_split))
  x.append(int_data)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, stratify = y, random_stat
e = 2021)
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(x_train)
x_train_sc = scaler.transform(x_train)
x_test_sc = scaler.transform(x_test)
```

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
hidden_layer = [450, 500, 550, 600]
for hl in hidden_layer:
  print('hidden layer : ', hl)
  mlp = MLPClassifier(hidden_layer_sizes=hl, random_state=1016, max_iter = 600)
  mlp.fit(x_train_sc, y_train)

  y_train_hat = mlp.predict(x_train_sc)
  y_test_hat = mlp.predict(x_test_sc)

  train_accuracy = accuracy_score(y_train, y_train_hat)
  test_accuracy = accuracy_score(y_test, y_test_hat)

  print('train acc : ', train_accuracy)
  print('test acc : ', test_accuracy)
```

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
solver = ['lbfgs', 'sgd', 'adam']
for sv in solver:
  print('solver : ', sv)
```

```
  mlp = MLPClassifier(hidden_layer_sizes=500, solver=sv, random_state=1016, max_ite
r=600)
  mlp.fit(x_train_sc, y_train)

  y_train_hat = mlp.predict(x_train_sc)
  y_test_hat = mlp.predict(x_test_sc)

  train_accuracy = accuracy_score(y_train, y_train_hat)
  test_accuracy = accuracy_score(y_test, y_test_hat)

  print('train acc : ', train_accuracy)
  print('test acc : ', test_accuracy)
```

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
activation = ['relu', 'tanh']
for ac in activation:
  print('activation : ', ac)
  mlp = MLPClassifier(hidden_layer_sizes=500, solver='adam', random_state=1016, max
_iter=600, activation=ac)
  mlp.fit(x_train_sc, y_train)

  y_train_hat = mlp.predict(x_train_sc)
  y_test_hat = mlp.predict(x_test_sc)

  train_accuracy = accuracy_score(y_train, y_train_hat)
  test_accuracy = accuracy_score(y_test, y_test_hat)

  print('train acc : ', train_accuracy)
  print('test acc : ', test_accuracy)
```

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
alpha = [0.0001, 0.001, 0.01, 0.1]
for ap in alpha:
  print('alpha : ', ap)
  mlp = MLPClassifier(hidden_layer_sizes=500, solver='adam', random_state=1016, max
_iter=600, activation='relu', alpha=ap)
  mlp.fit(x_train_sc, y_train)

  y_train_hat = mlp.predict(x_train_sc)
  y_test_hat = mlp.predict(x_test_sc)

  train_accuracy = accuracy_score(y_train, y_train_hat)
  test_accuracy = accuracy_score(y_test, y_test_hat)

  print('train acc : ', train_accuracy)
  print('test acc : ', test_accuracy)
```

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score


mmscaler = MinMaxScaler()
mmscaler.fit(x_train)
x_train_sc = mmscaler.transform(x_train)
x_test_sc = mmscaler.transform(x_test)

mlp = MLPClassifier(hidden_layer_sizes=500, solver='adam', random_state=1016, max_i
ter=1000, activation='relu', alpha=0.001)
mlp.fit(x_train_sc, y_train)

y_train_hat = mlp.predict(x_train_sc)
y_test_hat = mlp.predict(x_test_sc)

train_accuracy = accuracy_score(y_train, y_train_hat)
test_accuracy = accuracy_score(y_test, y_test_hat)

print('train acc : ', train_accuracy)
print('test acc : ', test_accuracy)
```