13강. 과적합 피하기

※오차역전파 정확도

〈실습〉

```python
from keras.models import Sequential
from keras.layers.core import Dense
from sklearn.preprocessing import LabelEncoder

import pandas as pd
import numpy
import tensorflow as tf

# seed 값 설정
numpy.random.seed(3)
tf.random.set_seed(3)

# 데이터 입력
df = pd.read_csv('sonar.csv', header=None)

# 데이터 개괄 보기
print(df.info())

# 데이터의 일부분 미리 보기
print(df.head())
```

〈데이터 확인〉

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 208 entries, 0 to 207
Data columns (total 61 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       208 non-null    float64
 1   1       208 non-null    float64
 2   2       208 non-null    float64
 3   3       208 non-null    float64
 4   4       208 non-null    float64
 5   5       208 non-null    float64
 6   6       208 non-null    float64
 7   7       208 non-null    float64
 8   8       208 non-null    float64
 9   9       208 non-null    float64
 10  10      208 non-null    float64
 11  11      208 non-null    float64
 12  12      208 non-null    float64
 13  13      208 non-null    float64
 14  14      208 non-null    float64
 15  15      208 non-null    float64
 16  16      208 non-null    float64
 17  17      208 non-null    float64
 18  18      208 non-null    float64
 19  19      208 non-null    float64
dtypes: float64(60), object(1)
memory usage: 99.2+ KB
None
      0       1       2       3       4   ...      56      57      58      59  60
0  0.0200  0.0371  0.0428  0.0207  0.0954  ...  0.0180  0.0084  0.0090  0.0032   R
1  0.0453  0.0523  0.0843  0.0689  0.1183  ...  0.0140  0.0049  0.0052  0.0044   R
2  0.0262  0.0582  0.1099  0.1083  0.0974  ...  0.0316  0.0164  0.0095  0.0078   R
3  0.0100  0.0171  0.0623  0.0205  0.0205  ...  0.0050  0.0044  0.0040  0.0117   R
4  0.0762  0.0666  0.0481  0.0394  0.0590  ...  0.0072  0.0048  0.0107  0.0094   R

[5 rows x 61 columns]
```

```python
dataset = df.values
X = dataset[:,0:60].astype(float) #astype 추가해야함. 버전때문,,
Y_obj = dataset[:,60]

# 문자열 변환
e = LabelEncoder()
e.fit(Y_obj)
Y = e.transform(Y_obj)

# 모델 설정
model = Sequential()
model.add(Dense(24,  input_dim=60, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# 모델 컴파일
model.compile(loss='mean_squared_error',
            optimizer='adam',
            metrics=['accuracy'])

# 모델 실행
model.fit(X, Y, epochs=200, batch_size=5)

# 결과 출력
print("\n Accuracy: %.4f" % (model.evaluate(X, Y)[1]))
```

(결과)

```
Epoch 189/200
42/42 [==============================] - 0s 1ms/step - loss: 5.9846e-04 - accuracy:
Epoch 190/200
42/42 [==============================] - 0s 1ms/step - loss: 5.9450e-04 - accuracy:
Epoch 191/200
42/42 [==============================] - 0s 1ms/step - loss: 5.3336e-04 - accuracy:
Epoch 192/200
42/42 [==============================] - 0s 1ms/step - loss: 5.6500e-04 - accuracy:
Epoch 193/200
42/42 [==============================] - 0s 1ms/step - loss: 5.4481e-04 - accuracy:
Epoch 194/200
42/42 [==============================] - 0s 1ms/step - loss: 5.9656e-04 - accuracy:
Epoch 195/200
42/42 [==============================] - 0s 1ms/step - loss: 7.1917e-04 - accuracy:
Epoch 196/200
42/42 [==============================] - 0s 1ms/step - loss: 5.4751e-04 - accuracy:
Epoch 197/200
42/42 [==============================] - 0s 1ms/step - loss: 4.3103e-04 - accuracy:
Epoch 198/200
42/42 [==============================] - 0s 1ms/step - loss: 3.8134e-04 - accuracy:
Epoch 199/200
42/42 [==============================] - 0s 1ms/step - loss: 4.9843e-04 - accuracy:
Epoch 200/200
42/42 [==============================] - 0s 1ms/step - loss: 4.2226e-04 - accuracy:
7/7 [==============================] - 0s 2ms/step - loss: 4.7309e-04 - accuracy: 1

Accuracy: 1.0000
```

• 과적합 : 모델이 학습 데이터셋 안에서는 일정수준 이상의 예측정확도를 보이지만,
새로운 데이터에 적용하면 잘 맞지 않는것.

※ 과적합을 피하기 위해서는?

- 학습 데이터셋과 테스트 데이터셋을 완전히 구분하여 학습과 테스트를 병행해야 함.

ex) 70개는 학습셋, 30개는 테스트셋

※ 학습이 깊어져서 학습셋 내부에서의 성공률은 높아져도 테스트셋에서 효과가 없다면 과적합이 일어나는 것!

〈실습 : 학습셋과 테스트셋 구분〉

```
dataset = df.values
X = dataset[:,0:60].astype(float)
Y_obj = dataset[:,60]

e = LabelEncoder()
e.fit(Y_obj)
Y = e.transform(Y_obj)

# 학습 셋과 테스트 셋의 구분
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,
                                                    random_state=seed)

model = Sequential()
model.add(Dense(24,  input_dim=60, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=['accuracy'])

model.fit(X_train, Y_train, epochs=130, batch_size=5)

# 테스트셋에 모델 적용
print("\n Test Accuracy: %.4f" % (model.evaluate(X_test, Y_test)[1]))
```

(결과)

```
Epoch 127/130
29/29 [==============================] - 0s 1ms/step - loss: 0.0294 - accuracy: 0.98
Epoch 128/130
29/29 [==============================] - 0s 2ms/step - loss: 0.0225 - accuracy: 0.98
Epoch 129/130
29/29 [==============================] - 0s 1ms/step - loss: 0.0239 - accuracy: 0.97
Epoch 130/130
29/29 [==============================] - 0s 1ms/step - loss: 0.0190 - accuracy: 0.99
2/2 [==============================] - 0s 8ms/step - loss: 0.1386 - accuracy: 0.8254

 Test Accuracy: 0.8254
```

〈모델 저장 및 재사용〉

```
from keras.models import Sequential, load_model
```

```
model.save('my_model.h5')  # 모델을 컴퓨터에 저장
```

```
del model       # 테스트를 위해 메모리 내의 모델을 삭제
model = load_model('my_model.h5') # 모델을 새로 불러옴

# 불러온 모델로 테스트 실행
print("\n Test Accuracy: %.4f" % (model.evaluate(X_test, Y_test)[1]))
```

(결과)

```
2/2 [==============================] - 0s 4ms/step - loss: 0.1557 - accuracy: 0.825

 Test Accuracy: 0.8254
```

- 테스트가 충분치 않음 → k겹 교차 검증
  - 데이터셋을 여러개로 나누어 하나씩 테스트셋으로 사용하고, 나머지를 모두 합해서 학습셋으로 사용하는 방법

〈실습〉

```
from sklearn.model_selection import StratifiedKFold
```

```
# 10개의 파일로 쪼갬
n_fold = 10
skf = StratifiedKFold(n_splits=n_fold, shuffle=True, random_state=seed)

# 빈 accuracy 배열
accuracy = []

# 모델의 설정, 컴파일, 실행
for train, test in skf.split(X, Y):
    model = Sequential()
    model.add(Dense(24, input_dim=60, activation='relu'))
    model.add(Dense(10, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='mean_squared_error',
                  optimizer='adam',
                  metrics=['accuracy'])
    model.fit(X[train], Y[train], epochs=100, batch_size=5)
    k_accuracy = "%.4f" % (model.evaluate(X[test], Y[test])[1])
    accuracy.append(k_accuracy)

# 결과 출력
print("\n %.f fold accuracy:" % n_fold, accuracy)
```

(결과)

```
Epoch 96/100
38/38 [==============================] - 0s 2ms/step - loss: 0.0184 - accuracy: 0.9
Epoch 97/100
38/38 [==============================] - 0s 2ms/step - loss: 0.0323 - accuracy: 0.9
Epoch 98/100
38/38 [==============================] - 0s 2ms/step - loss: 0.0365 - accuracy: 0.9
Epoch 99/100
38/38 [==============================] - 0s 2ms/step - loss: 0.0271 - accuracy: 0.9
Epoch 100/100
38/38 [==============================] - 0s 2ms/step - loss: 0.0270 - accuracy: 0.9
WARNING:tensorflow:11 out of the last 12 calls to <function Model.make_test_functio
1/1 [==============================] - 0s 118ms/step - loss: 0.1398 - accuracy: 0.8

 10 fold accuracy: ['0.7143', '0.7619', '0.7619', '0.9048', '0.8095', '0.8095', '0.
```