8.3 텐서플로 신경망 분석 – 스마트카 이상 징후

딥러닝 학습 및 평가 (1/12)

01. 학습할 데이터셋을 "C://training_data", "C://test_data" 폴더를 만들어 복사해 놓는다.

- C://예제소스/bigdata2nd-master/CH08/CarDrivingIncidentInfo.csv" 파일을 "C://training_data" 디렉터리를 만들어 복사해 놓는다.
- C://예제소스/bigdata2nd-master/CH08/CarDrivingIncidentInfo_Test.csv" 파일을 "C://test_data" 디렉터리를 만들어 복사해 놓는다.

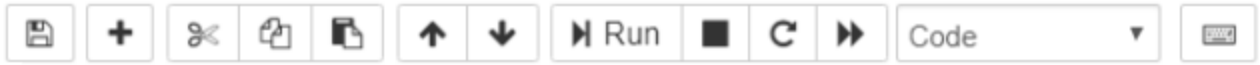
02. DNN 모델을 저장해 놓기 위한 "C://models" 폴더를 만들어 놓는다.

C://models

8.3 텐서플로 신경망 분석 – 스마트카 이상 징후

딥러닝 학습 및 평가 (2/12)

02. 주피터 노트북의 첫 번째 셀에서 개발에 필요한 라이브러리를 임포트하고, 학습 데이터를 판다스의 데이터프레임으로 로드하는 코드를 입력한 후 Shift + Enter를 누르거나 상단의 [Run] 버튼을 클릭한다. 전체 소스코드는 "C://예제소스/bigdata2nd-master/CH08/smartcar_dnn_model.py"에 있으므로 참고한다.




```
import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt

from time import time
from tensorflow.python.keras.callbacks import TensorBoard
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, roc_auc_score, auc
from keras.utils import to_categorical

df = pd.read_csv('C:/training_data/CarDrivingIncidentInfo.csv')
```

8.3 텐서플로 신경망 분석 – 스마트카 이상 징후

딥러닝 학습 및 평가 (3/12)

03. 다음 코드를 입력해 데이터프레임에 로드된 학습 데이터를 배열 구조의 입력값(X)과 결괏값(Y)로 분리 구성하고 범주화와 스케일 작업으로 데이터 전처리를 진행한다.

```
X = df.iloc[:, :-1].values
Y = df.iloc[:, -1].values

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=1)

Y_train = to_categorical(Y_train)
Y_test = to_categorical(Y_test)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

8.3 텐서플로 신경망 분석 – 스마트카 이상 징후

딥러닝 학습 및 평가 (4/12)

04. DNN 모델을 케라스의 Sequential로 구성한다. 입력 레이어(Input: 10, Output: 10), 은닉1 레이어(Input: 10, Output: 20), 은닉2 레이어(Input: 20, Output: 10), 은닉3 레이어(Input: 10, Output: 10), 출력 레이어(Input: 10, Output: 3)를 정의하고 활성화 함수는 Relu와 Softmax로 설정한다. 모델을 컴파일하고 요약 정보를 출력한다.

```
model = Sequential([
    Dense(10, input_dim=10, activation='relu'),
    Dense(20, activation='relu'),
    Dropout(0.25),
    Dense(10, activation='relu'),
    Dense(3, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 10)                110
_________________________________________________________________
dense_1 (Dense)              (None, 20)                220
_________________________________________________________________
dropout (Dropout)            (None, 20)                0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                210
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 33
=================================================================
Total params: 573
Trainable params: 573
Non-trainable params: 0
_________________________________________________________________
```

8.3 텐서플로 신경망 분석 – 스마트카 이상 징후

딥러닝 학습 및 평가 (5/12)

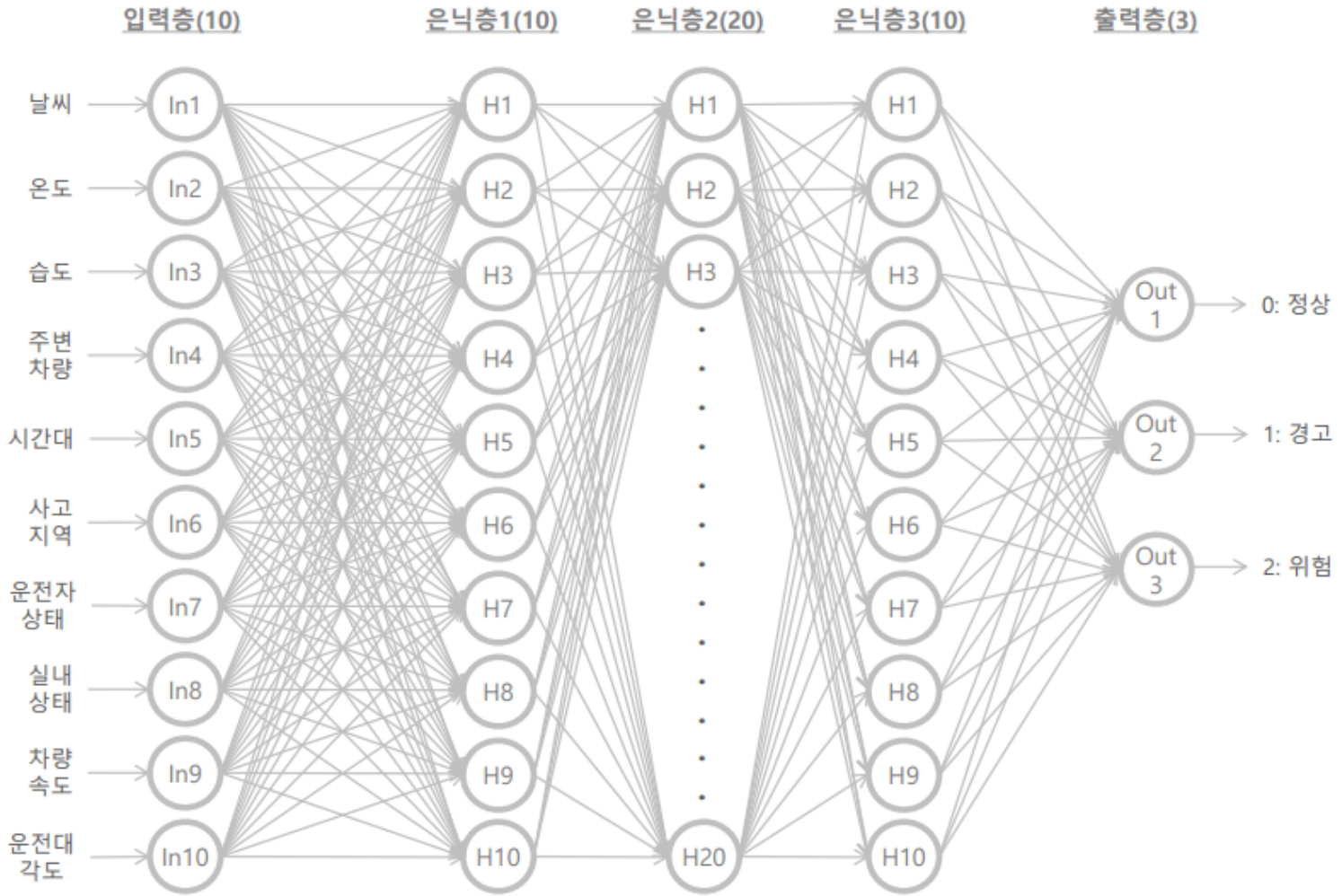


그림 8.24 스마트카 위험 징후를 판별하는 신경망 구조

8.3 텐서플로 신경망 분석 – 스마트카 이상 징후

딥러닝 학습 및 평가 (6/12)

04. 정의한 모델을 학습시킨다. 배치 사이즈 2000으로, 에폭은 50번을 반복하며 학습을 진행한다. 에폭이 진행될 때마다 loss 값은 줄고 정확도는 증가하는 모습을 볼 수 있다. 추가로 진행 과정을 텐서보드 이벤트 로그로 지정했다.

```
tensorboard = keras.callbacks.TensorBoard(log_dir='./graph', histogram_freq=0,
                                          write_graph=True, write_images=True)

hist  = model.fit(X_train, Y_train, batch_size=2000, epochs=50,
                  callbacks=[tensorboard], validation_data=(X_test, Y_test))
```

```
Train on 160000 samples, validate on 40000 samples
Epoch 1/50
160000/160000 [==============================] - 1s 6us/step - loss: 0.8718 - acc: 0.6004 - val_loss: 0.6790 - val_acc: 0.7321
Epoch 2/50
160000/160000 [==============================] - 1s 4us/step - loss: 0.6023 - acc: 0.7618 - val_loss: 0.4541 - val_acc: 0.8324
Epoch 3/50
160000/160000 [==============================] - 1s 4us/step - loss: 0.4258 - acc: 0.8307 - val_loss: 0.3136 - val_acc: 0.8668
Epoch 4/50
160000/160000 [==============================] - 1s 4us/step - loss: 0.3027 - acc: 0.8636 - val_loss: 0.2150 - val_acc: 0.8892
Epoch 5/50
160000/160000 [==============================] - 1s 4us/step - loss: 0.2111 - acc: 0.9210 - val_loss: 0.1396 - val_acc: 0.9634
```

8.3 텐서플로 신경망 분석 – 스마트카 이상 징후

딥러닝 학습 및 평가 (7/12)

05. 다음 코드를 실행해 학습된 DNN 모델의 최종 결괏값을 확인해 본다. 정확도가 0.9997(99%)까지 향상됐다.

```
score = model.evaluate(X_test, Y_test, verbose=0)
print(model.metrics_names)
print(score)
```

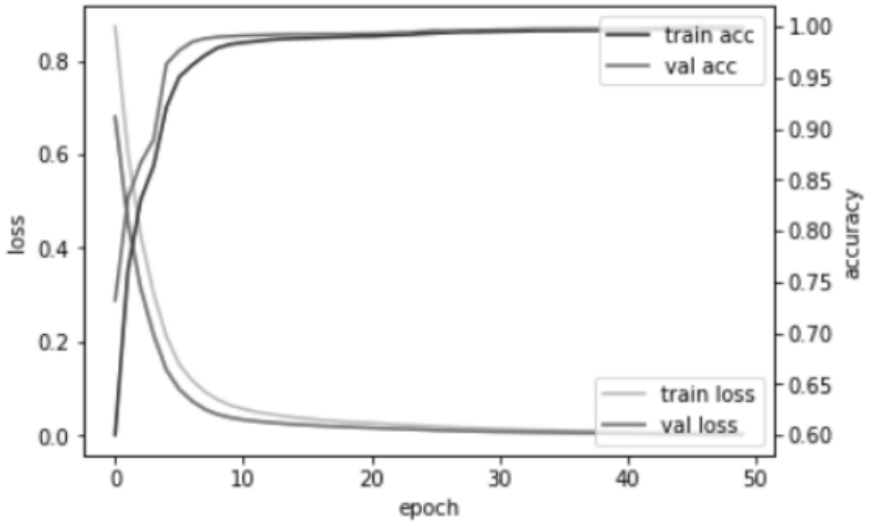
```
['loss', 'acc']
[0.0014955078887650642, 0.9997]
```

8.3 텐서플로 신경망 분석 – 스마트카 이상 징후

딥러닝 학습 및 평가 (8/12)

06. 이어서 학습 결과를 시각화해 출력해 본다. 훈련/검증 정확도(train/val acc)와 훈련/검증 손실 값(train/val loss)이 각각 상승/하향 곡선을 그리다가 에폭 10을 기점으로 완만해진 결과를 얻었다.

```
fig, loss_ax = plt.subplots()
acc_ax = loss_ax.twinx()

loss_ax.plot(hist.history['loss'], 'y', label='train loss')
loss_ax.plot(hist.history['val_loss'], 'r', label='val loss')
acc_ax.plot(hist.history['acc'], 'b', label='train acc')
acc_ax.plot(hist.history['val_acc'], 'g', label='val acc')

loss_ax.set_xlabel('epoch')

loss_ax.set_ylabel('loss')
loss_ax.legend(loc='lower right')

acc_ax.set_ylabel('accuracy')
acc_ax.legend(loc='upper right')

plt.show()
```



8.3 텐서플로 신경망 분석 – 스마트카 이상 징후

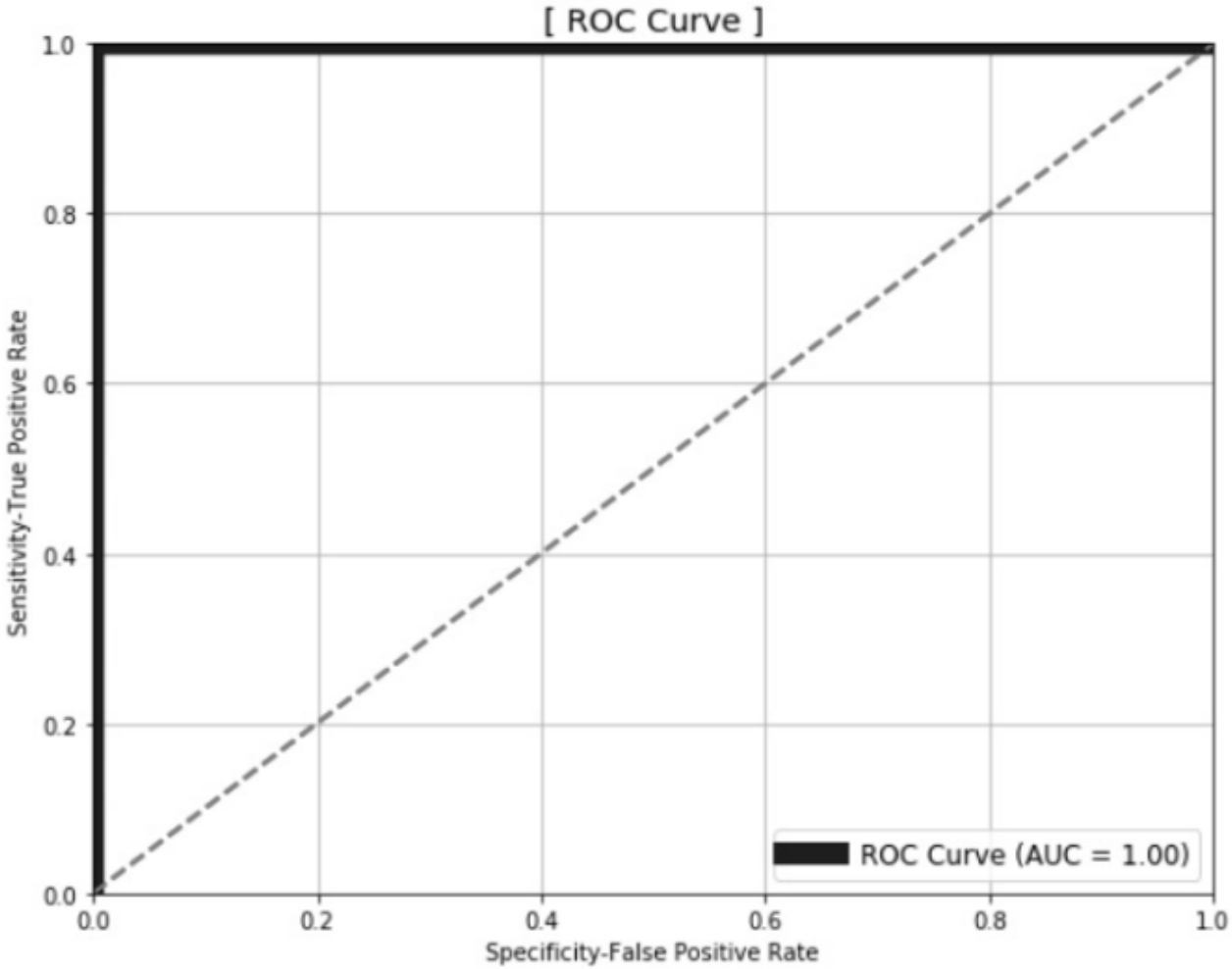
딥러닝 학습 및 평가 (9/12)

07. 학습된 모델을 테스트 데이터로 예측(y_predict_result) 후, 실제값(Y_test)과 비교해 성능을 평가하는 ROC 차트를 출력해 본다. ROC 차트를 해석하자면 대각선 라인을 중심으로 ROC 커브가 상단에 형성되면 좋은 모델이고, 하단에 형성되면 좋지 않은 모델이다. 아래 DNN 모델의 경우 ROC 커브가 민감도(Sensitivity) 1에 매우 근접해서 정확도가 100%에 가까운 결과를 보여준다.

```
y_predict_result = model.predict(X_test)

fpr, tpr, thresholds = roc_curve (Y_test.ravel(), y_predict_result.ravel())
roc_auc = auc(fpr, tpr)

plt.clf()
plt.figure(figsize = (9, 7))
plt.plot(fpr, tpr, color='navy', lw=10, label='ROC Curve (AUC = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.rcParams['font.size'] = 12
plt.title('[ ROC Curve ]')
plt.xlabel('Specificity-False Positive Rate')
plt.ylabel('Sensitivity-True Positive Rate')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

8.3 텐서플로 신경망 분석 – 스마트카 이상 징후

딥러닝 학습 및 평가 (10/12)



8.3 텐서플로 신경망 분석 – 스마트카 이상 징후

딥러닝 학습 및 평가 (11/12)

08. 마지막으로 생성된 모델을 다음 경로를 만들어 저장한다. 이후 단계에서 저장된 모델을 활용해 웹 애플리케이션 서비스를 만들어 보겠다.

```
from keras.models import load_model

model.save('C://models/smartcar_dnn_model.h5')
```

8.3 텐서플로 신경망 분석 – 스마트카 이상 징후

딥러닝 학습 및 평가 (12/12)

실습