AI 라이브러리 활용

15장 선형 회귀 적용하기

이 찬 우

학습내용 : 베스트 모델 만들기

4 | 학습의 자동 중단

4 | 학습의 자동 중단

- 학습이 진행될수록 학습셋의 정확도는 올라가지만 과적합 때문에 테스트셋의 실험 결과는 점점 나빠지게 됨
- EarlyStopping( ) 함수 :

  케라스에는 이렇게 학습이 진행되어도 테스트셋 오차가 줄지 않으면

  학습을 멈추게 하는 함수

```
from keras.callbacks import EarlyStopping
```

4 | 학습의 자동 중단

- EarlyStopping( ) 함수에 모니터할 값과 테스트 오차가 좋아지지 않아도 몇 번까지 기다릴지를 정함
- 이를 early_stopping_callback에 저장함

```
early_stopping_callback = EarlyStopping(monitor='val_loss',
patience=100)
```

4 | 학습의 자동 중단

- 이제 앞서 정한 그대로 에포크 횟수와 배치 크기 등을 설정하고 early_stopping_callback 값을 불러옴

```
model.fit(X, Y, validation_split=0.33, epochs=2000, batch_
size=500, callbacks=[early_stopping_callback])
```

4 | 학습의 자동 중단

코드 14-4 와인의 종류 예측하기: 학습의 자동 중단

- 예제 소스: run_project/11_Wine_Early_Stop.ipynb

```
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import EarlyStopping

import pandas as pd
import numpy
import tensorflow as tf

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

4 | 학습의 자동 중단

```
df_pre = pd.read_csv('../dataset/wine.csv', header=None)
df = df_pre.sample(frac=0.15)

dataset = df.values
X = dataset[:,0:12]
Y = dataset[:,12]

model = Sequential( )
model.add(Dense(30, input_dim=12, activation='relu'))
model.add(Dense(12, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

4 | 학습의 자동 중단

```
# 학습 자동 중단 설정
early_stopping_callback = EarlyStopping(monitor='val_loss',
patience=100)

# 모델 실행
model.fit(X, Y, validation_split=0.2, epochs=2000, batch_
size=500, callbacks=[early_stopping_callback])

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

4 | 학습의 자동 중단

실행 결과

(중략)

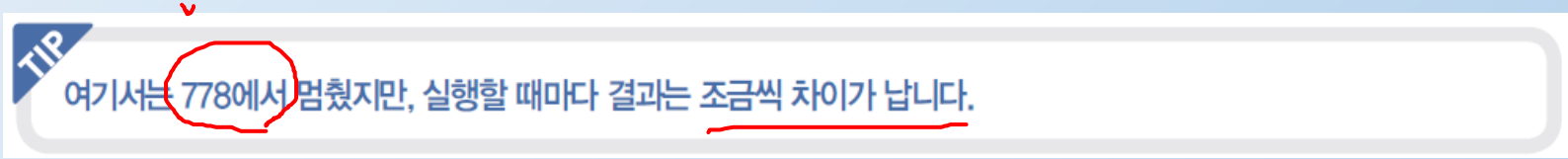
```
Epoch 775/2000
780/780 [==============================] - 0s - loss: 0.0186 - acc:
0.9962 - val_loss: 0.0662 - val_acc: 0.9795
Epoch 776/2000
780/780 [==============================] - 0s - loss: 0.0193 - acc:
0.9936 - val_loss: 0.0743 - val_acc: 0.9795
Epoch 777/2000
```

4 | 학습의 자동 중단

```
780/780 [==============================] - 0s - loss: 0.0198 - acc:
0.9936 - val_loss: 0.0660 - val_acc: 0.9795
Epoch 778/2000
780/780 [==============================] - 0s - loss: 0.0236 - acc:
0.9936 - val_loss: 0.0699 - val_acc: 0.9846
 32/975 [.....] - ETA: 0s
 Accuracy: 0.9918
```

4 | 학습의 자동 중단

- 에포크를 2,000으로 설정하였지만, 도중에 계산이 멈추는 것을 확인할 수 있음



4 | 학습의 자동 중단

코드 14-5 와인의 종류 예측하기: 전체 코드

- 예제 소스: run_project/12_Wine_Check_and_Stop.ipynb

```
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import ModelCheckpoint, EarlyStopping

import pandas as pd
import numpy
import os
import tensorflow as tf
```

4 | 학습의 자동 중단

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

df_pre = pd.read_csv('../dataset/wine.csv', header=None)
df = df_pre.sample(frac=0.15)
dataset = df.values
X = dataset[:,0:12]
Y = dataset[:,12]
```

4 | 학습의 자동 중단

```
model = Sequential( )
model.add(Dense(30, input_dim=12, activation='relu'))
model.add(Dense(12, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

4 | 학습의 자동 중단

```
# 모델 저장 폴더 만들기
MODEL_DIR = './model/'
if not os.path.exists(MODEL_DIR):
    os.mkdir(MODEL_DIR)

modelpath="./model/{epoch:02d}-{val_loss:.4f}.hdf5"

# 모델 업데이트 및 저장
checkpointer = ModelCheckpoint(filepath=modelpath, monitor='val_
loss', verbose=1, save_best_only=True)
```

4 | 학습의 자동 중단

```
# 학습 자동 중단 설정
early_stopping_callback = EarlyStopping(monitor='val_loss',
patience=100)

model.fit(X, Y, validation_split=0.2, epochs=3500, batch_size=500,
verbose=0, callbacks=[early_stopping_callback,checkpointer])
```

4 | 학습의 자동 중단

실행 결과

(중략)

Epoch 00680: val_loss did not improve

Epoch 00681: val_loss did not improve

Epoch 00682: val_loss improved from 0.05415 to 0.05286, saving model to ./model/682-0.0529.hdf5

Epoch 00683: val_loss did not improve

Epoch 00684: val_loss did not improve

Epoch 00685: val_loss did not improve

4 | 학습의 자동 중단

```
(중략)


Epoch 00781: val_loss did not improve
Epoch 00782: val_loss did not improve
Epoch 00783: val_loss did not improve
```

15장 선형 회귀 적용하기

1 | 데이터 확인하기

2 | 선형 회귀 실행

선형 회귀 적용하기

- **실습 데이터** 보스턴 집값 예측
  - dataset/housing.csv

선형 회귀 적용하기

- 1978년, 집값에 가장 큰 영향을 미치는 것이 '깨끗한 공기'라는 연구 결과가
  하버드대학교 도시개발학과에서 발표됨
- 이들은 자신의 주장을 뒷받침하기 위해 집값의 변동에 영향을 미치는
  여러 가지 요인을 모아서 환경과 집값의 변동을 보여주는 데이터셋을 만듦
- 그로부터 수십 년 후,
  이 데이터셋은 머신러닝의 선형 회귀를 테스트하는 가장 유명한 데이터로 쓰이고 있음

1 | 데이터 확인하기

```
import pandas as pd

df = pd.read_csv("../dataset/housing.csv", delim_whitespace=True,
header=None)

print(df.info( ))
```

1 | 데이터 확인하기

Range Index:506 entries,0 to 505			
Data columns (total 14 columns):			
0	506	non-null	float64
1	506	non-null	float64
2	506	non-null	float64
3	506	non-null	int64
...	...	...	...
11	506	non-null	float64
12	506	non-null	float64
13	506	non-null	float64
Dtypes: float64(12), int64(2)			
memory usage: 55.4 KB			

1 | 데이터 확인하기

- Index가 506개이므로 총 샘플의 수는 506개이고,

  컬럼 수는 14개이므로 13개의 속성과 1개의 클래스로 이루어졌음을 짐작할 수 있음

- 일부를 출력해서 확인해 보겠음

```
print(df.head( ))
```

1 | 데이터 확인하기

	0	1	2	3	···	12	13
0	0.00632	18.0	2.31	0	···	4.98	24.0
1	0.02731	0	7.07	0	···	9.14	21.6
2	0.02729	0	7.07	0	···	4.03	34.7
3	0.03237	0	2.18	0	···	2.94	33.4
4	0.06905	0	2.18	0	···	5.33	36.2

1 | 데이터 확인하기

- 특히 마지막 컬럼을 보면 지금까지와는 다름
- 클래스로 구분된 게 아니라 가격이 나와 있음

0	CRIM: 인구 1인당 범죄 발생 수	7	DIS: 5가지 보스턴 시 고용 시설까지의 거리
1	ZN: 25,000평방 피트 이상의 주거 구역 비중	8	RAD: 순환고속도로의 접근 용이성
2	INDUS: 소매업 외 상업이 차지하는 면적 비율	9	TAX: $10,000당 부동산 세율 총계
3	CHAS: 찰스강 위치 변수(1: 강 주변, 0: 이외)	10	PTRATIO: 지역별 학생과 교사 비율
4	NOX: 일산화질소 농도	11	B: 지역별 흑인 비율
5	RM: 집의 평균 방 수	12	LSTAT: 급여가 낮은 직업에 종사하는 인구 비율(%)
6	AGE: 1940년 이전에 지어진 비율	13	가격(단위: $1,000)

2 | 선형 회귀 실행

- 선형 회귀 데이터는 마지막에 참과 거짓을 구분할 필요가 없음
- 출력층에 활성화 함수를 지정할 필요도 없음

```
model = Sequential( )
model.add(Dense(30, input_dim=13, activation='relu'))
model.add(Dense(6, activation='relu'))
model.add(Dense(1))
```

2 | 선형 회귀 실행

- 모델의 학습이 어느 정도 되었는지 확인하기 위해 예측 값과 실제 값을 비교하는 부분을 추가함

```
Y_prediction = model.predict(X_test).flatten( )
for i in range(10):
    label = Y_test[i]
    prediction = Y_prediction[i]
    print("실제가격: {:.3f}, 예상가격: {:.3f}".format(label, prediction))
```

2 | 선형 회귀 실행

- flatten( ) 함수 :

  데이터 배열이 몇 차원이든 모두 1차원으로 바꿔 읽기 쉽게 해 주는 함수

- range('숫자')는 0부터 '숫자-1'만큼 차례대로 증가하며 반복되는 값을 만듦
- 즉, range(10)은 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]를 말함
- for in 구문을 사용해서 10번 반복하게 함

2 | 선형 회귀 실행

코드 15-1 보스턴 집값 예측하기

- 예제 소스: run_project/13_Boston.ipynb

```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split

import numpy
import pandas as pd
import tensorflow as tf
```

2 | 선형 회귀 실행

```
# seed 값 설정
seed = 0
numpy.random.seed(seed)
tf.random.set_seed(3)

df = pd.read_csv("../dataset/housing.csv", delim_whitespace=True,
header=None)

dataset = df.values
X = dataset[:,0:13]
Y = dataset[:,13]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_
size=0.3, random_state=seed)
```

2 | 선형 회귀 실행

```
model = Sequential()
model.add(Dense(30, input_dim=13, activation='relu'))
model.add(Dense(6, activation='relu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error',
              optimizer='adam')
model.fit(X_train, Y_train, epochs=200, batch_size=10)

# 예측 값과 실제 값의 비교
Y_prediction = model.predict(X_test).flatten()
for i in range(10):
    label = Y_test[i]
    prediction = Y_prediction[i]
    print("실제가격: {:.3f}, 예상가격: {:.3f}".format(label, prediction))
```

2 | 선형 회귀 실행

실행 결과

실제가격: 22.600, 예상가격: 20.701

실제가격: 50.000, 예상가격: 26.329

실제가격: 23.000, 예상가격: 21.918

실제가격: 8.300, 예상가격: 12.454

실제가격: 21.200, 예상가격: 18.575

실제가격: 19.900, 예상가격: 21.978

실제가격: 20.600, 예상가격: 19.141

실제가격: 18.700, 예상가격: 24.095

실제가격: 16.100, 예상가격: 19.012

실제가격: 18.600, 예상가격: 13.672

정리 학습 : 베스트 모델 만들기

4 | 학습의 자동 중단

정리 학습 : 선형 회귀 적용하기

1 | 데이터 확인하기

2 | 선형 회귀 실행

다음 수업

CNN 익히기