12장 다중 분류 문제 해결하기

목 차

1 다중 분류 문제

2 상관도 그래프

3 원-핫 인코딩

4 소프트맥스

5 아이리스 품종 예측 실행

다중 분류 문제 해결하기

- **실습 데이터** 아이리스 품종 예측
  - dataset/iris.csv

1 다중 분류 문제

- 아이리스는 꽃잎의 모양과 길이에 따라 여러 가지 품종으로 나뉨
- 사진을 보면 품종마다 비슷해 보이는데 과연 딥러닝을 사용하여 이들을 구별해 낼 수 있을까?

그림 12-1 아이리스의 품종

1 다중 분류 문제

- 아이리스 품종 예측 데이터는 책과 함께 제공하는 예제 파일의 dataset 폴더에서 찾을 수 있음(dataset/iris.csv)

속성 / 클래스

	정보 1	정보 2	정보 3	정보 4	품종
샘플	1번째 아이리스	5.1	3.5	4.0	0.2	Iris-setosa
	2번째 아이리스	4.9	3.0	1.4	0.2	Iris-setosa
	3번째 아이리스	4.7	3.2	1.3	0.3	Iris-setosa
	...	...	...	...	...	...
	150번째 아이리스	5.9	3.0	5.1	1.8	Iris-virginica

표 12-1 아이리스 데이터의 샘플, 속성, 클래스 구분

1 다중 분류 문제

- 샘플 수: 150
- 속성 수: 4
  - 정보 1: 꽃받침 길이 (sepal length, 단위: cm)
  - 정보 2: 꽃받침 너비 (sepal width, 단위: cm)
  - 정보 3: 꽃잎 길이 (petal length, 단위: cm)
  - 정보 4: 꽃잎 너비 (petal width, 단위: cm)
- 클래스: Iris-setosa, Iris-versicolor, Iris-virginica

1 다중 분류 문제

- 속성을 보니 우리가 앞서 다루었던 것과 중요한 차이는 바로 클래스가 2개가 아니라 3개임
- 즉, 참(1)과 거짓(0)으로 해결하는 것이 아니라, 여러개 중에 어떤 것이 답인지를 예측하는 문제임
- 다중 분류 (multi classification) :

  여러 개의 답 중 하나를 고르는 분류 문제
- 다중 분류 문제는 둘 중에 하나를 고르는 이항 분류(binary classification)와는 접근 방식이 조금 다름

2 상관도 그래프

- 먼저 데이터의 일부를 불러와 내용을 보자

```
import pandas as pd
df = pd.read_csv('../dataset/iris.csv', names = ["sepal_length",
"sepal_width", "petal_length", "petal_width", "species"])
print(df.head( ))
```

2 상관도 그래프

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5	3.6	1.4	0.2	Iris-setosa

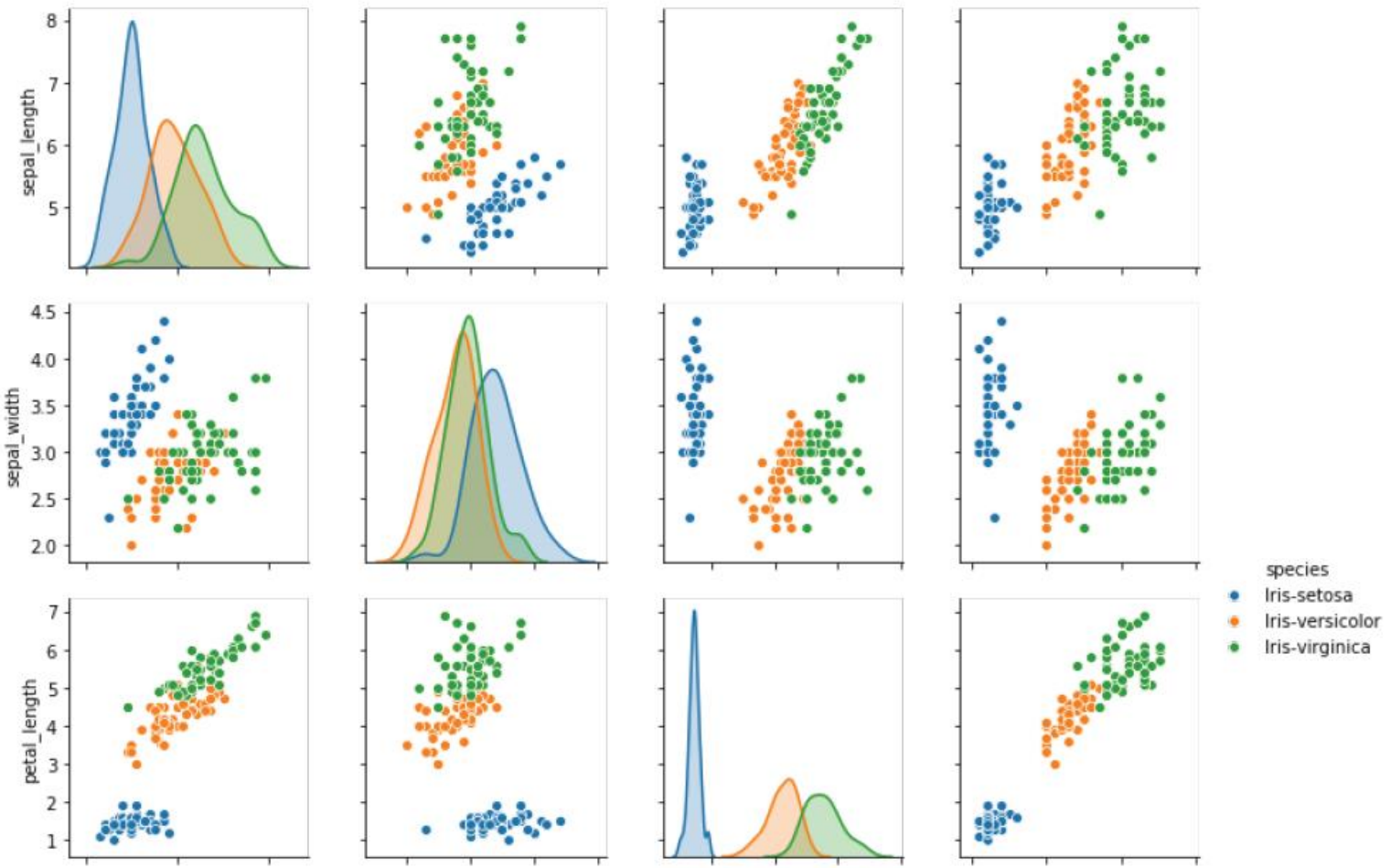
2 상관도 그래프

- 이번에는 pairplot( ) 함수를 써서 데이터 전체를 한번에 보는 그래프를 다음과 같이 출력

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(df, hue='species');
plt.show( )
```

2 상관도 그래프


sepal_length
sepal_width
petal_length
species
Iris-setosa
Iris-versicolor
Iris-virginica

2 상관도 그래프

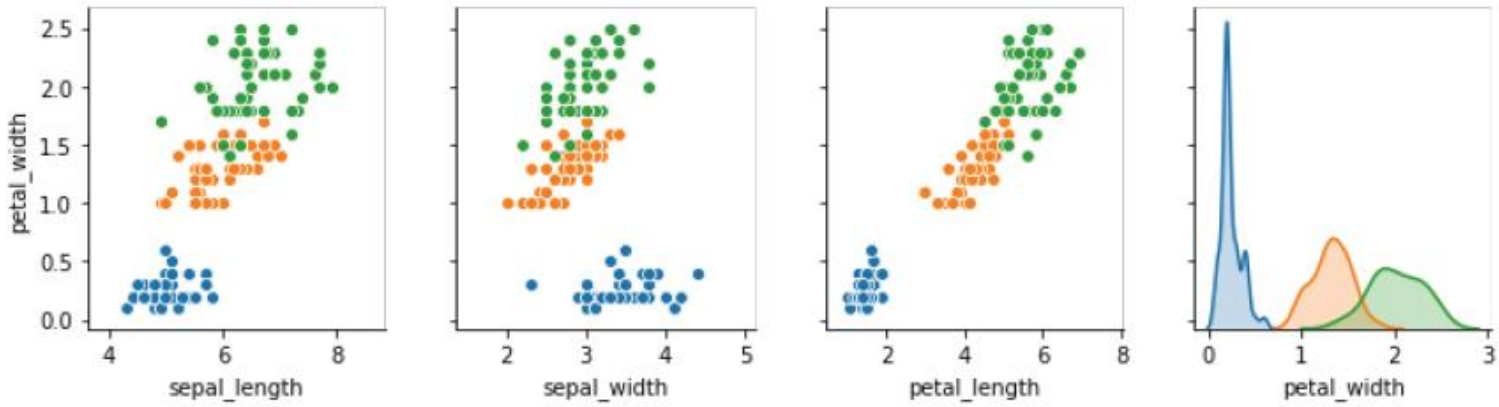


그림 12-2 pairplot 함수로 데이터 한번에 보기

2 상관도 그래프

- 그래프를 보니, 사진으로 볼 때는 비슷해 보이던 꽃잎과 꽃받침의 크기와 너비가 품종별로 차이가 있음을 알 수 있음
- 속성별로 어떤 연관이 있는지를 보여 주는 상관도 그래프를 통해 프로젝트의 감을 잡고 프로그램 전략을 세울 수 있음

3 원-핫 인코딩

- 이제 케라스를 이용해 아이리스의 품종을 예측해 보자
- Iris-setosa, Iris-virginica 등 데이터 안에 문자열이 포함되어 있음
- numpy보다는 pandas로 데이터를 불러와 X와 Y값을 구분하는 것이

```
df = pd.read_csv('../dataset/iris.csv', names = ["sepal_length",
"sepal_width", "petal_length", "petal_width", "species"])


dataset = df.values
X = dataset[:,0:4].astype(float)
Y_obj = dataset[:,4]
```

3 원-핫 인코딩

- 또한, Y값이 이번에는 숫자가 아니라 문자열임
- 문자열을 숫자로 바꿔 주려면 클래스 이름을 숫자 형태로 바꿔 주어야 함
- 이를 가능하게 하는 함수가 sklearn 라이브러리의 LabelEncoder( ) 함

```
from sklearn.preprocessing import LabelEncoder


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

3 원-핫 인코딩

- array(['Iris-setosa', 'Iris-versicolor','Iris-virginica'])가 array([1,2,3])로 바뀜
- 활성화 함수를 적용하려면 Y 값이 숫자 0과 1로 이루어져 있어야 함
- 이 조건을 만족시키려면 tf.keras.utils.categorical( ) 함수를 적용해야 함
- 이에 따라 Y 값의 형태는 다음과 같이 변형됨

```
from tensorflow.keras.utils import np_utils


Y_encoded = tf.keras.utils.to_categorical(Y)
```

3 원-핫 인코딩

- array([1,2,3])가 다시 array([[1., 0., 0.], [0., 1., 0.],[0., 0., 1.]])로 바뀜
- 원-핫 인코딩(one-hot-encoding) :

  여러 개의 Y 값을 0과 1로만 이루어진 형태로 바꿔 주는 기법

4 소프트맥스

- 이제 모델을 만들어 보자

```
model = Sequential( )
model.add(Dense(16, input_dim=4, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

4 소프트맥스

- 소프트맥스 :

  그림 12-3에서와 같이 총합이 1인 형태로 바꿔서 계산해 주는 함

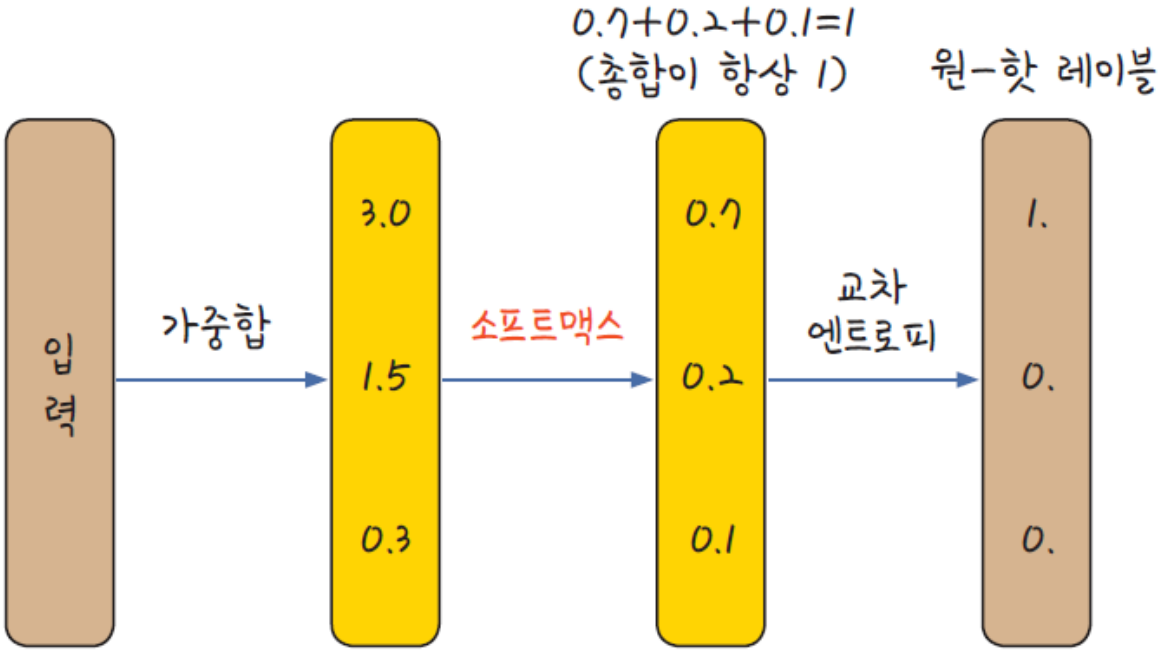


그림 12-3 소프트맥스 함수의 원리

5 아이리스 품종 예측 실행

- 다중 분류에 적절한 오차 함수인 categorical_crossentropy를 사용하고, 최적화 함수로 adam을 사용함
- 전체 샘플이 50회 반복될 때까지 실험을 진행하되 한 번에 입력되는 값은 1개로 함

코드 12-1 아이리스 품종 예측하기

- 예제 소스: run_project/03_Iris_Multi_Classfication.ipynb

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

5 아이리스 품종 예측 실행

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf

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

# 데이터 입력
df = pd.read_csv('../dataset/iris.csv', names = ["sepal_length",
"sepal_width", "petal_length", "petal_width", "species"])
```

5 아이리스 품종 예측 실행

```
# 그래프로 확인
sns.pairplot(df, hue='species');
plt.show( )

# 데이터 분류
dataset = df.values
X = dataset[:,0:4].astype(float)
Y_obj = dataset[:,4]

# 문자열을 숫자로 변환
e = LabelEncoder( )
e.fit(Y_obj)
Y = e.transform(Y_obj)
Y_encoded = tf.keras.utils.to_categorical(Y)
```

5 아이리스 품종 예측 실행

```
# 모델의 설정
model = Sequential( )
model.add(Dense(16, input_dim=4, activation='relu'))
model.add(Dense(3, activation='softmax'))

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

5 아이리스 품종 예측 실행

```
# 모델 실행
model.fit(X, Y_encoded, epochs=50, batch_size=1)

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

5 아이리스 품종 예측 실행

실행 결과

```
Train on 150 samples
Epoch 1/50
150/150 [==============================] - 0s 3ms/sample - loss: 1.2541
- accuracy: 0.3733
Epoch 2/50
150/150 [==============================] - 0s 1ms/sample - loss: 0.9932
- accuracy: 0.4200
Epoch 3/50
150/150 [==============================] - 0s 1ms/sample - loss: 0.7919
- accuracy: 0.6400
(중략)
```

5 아이리스 품종 예측 실행

```
Epoch 47/50
150/150 [==============================] - 0s 1ms/sample - loss: 0.0926
- accuracy: 0.9867
Epoch 48/50
150/150 [==============================] - 0s 993us/sample - loss:
0.0924 - accuracy: 0.9733
Epoch 49/50
150/150 [==============================] - 0s 1ms/sample - loss: 0.0884
- accuracy: 0.9667
Epoch 50/50
150/150 [==============================] - 0s 999us/sample - loss:
0.0894 - accuracy: 0.9733
...
 Accuracy: 0.9867
```