넷째마당
딥러닝 기본기 다지기

12장 다중 분류 문제 해결하기

1 다중 분류 문제

2 상관도 그래프

3 원-핫 인코딩

4 소프트맥스

5 아이리스 품종 예측의 실행

1 다중 분류 문제

1 다중 분류 문제

- **다중 분류 문제**
  - 아이리스는 그 꽃봉오리가 마치 먹물을 머금은 붓과 같다 하여 우리나라에서는 '붓꽃'이라고 부르는 아름다운 꽃
  - 아이리스는 **꽃잎의 모양**과 **길이**에 따라 여러 가지 품종으로 나뉨
  - 사진을 보면 품종마다 비슷해 보이는데요. 과연 딥러닝을 사용해서 이들을 구별해 낼 수 있을까?

▼ **그림 12-1 | 아이리스의 품종**

Iris-virginica　　Iris-setosa　　Iris-versicolor

iris setosa

iris versicolor

iris virginica

1 다중 분류 문제

- **다중 분류 문제**
  - 아이리스 품종 예측 데이터는 예제 파일의 data 폴더에서 찾을 수 있음(data/iris3.csv)
  - 데이터의 구조는 다음과 같음

▼ **그림 12-2 | 아이리스 데이터의 샘플, 속성, 클래스 구분**

속성 X(입력데이터) 클래스 y(정답)

	정보 1	정보 2	정보 3	정보 4	품종
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

샘플

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

- **다중 분류 문제**
  - 속성을 보니 우리가 앞서 다루었던 것과 중요한 차이가 있음
  - 바로 클래스가 두 개가 아니라 세 개
  - 즉, 참(1)과 거짓(0)으로 해결하는 것이 아니라, 여러 개 중에 어떤 것이 답인지 예측하는 문제
  - 이렇게 여러 개의 답 중 하나를 고르는 분류 문제를 **다중 분류(multi classification)**라고 함
  - 다중 분류 문제는 둘 중에 하나를 고르는 **이항 분류(binary classification)**와는 접근 방식이 조금 다름
  - 지금부터 아이리스 품종을 예측하는 실습을 통해 **다중 분류 문제를** 해결해 보자

2 상관도 그래프

2 상관도 그래프

- 상관도 그래프
  - 먼저 데이터의 일부를 불러와 내용을 보자

```
import pandas as pd

# 깃허브에 준비된 데이터를 가져옵니다.
!git clone https://github.com/taehojo/data.git

# 아이리스 데이터를 불러옵니다.
df = pd.read_csv('./data/iris3.csv')

df.head() # 첫 다섯 줄을 봅니다.
```

2 상관도 그래프

- 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	3.0	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	5.0	3.6	1.4	0.2	Iris-setosa

X = df.iloc[:, 0:4]

입력데이터

y = df.iloc[:, 4]

정답

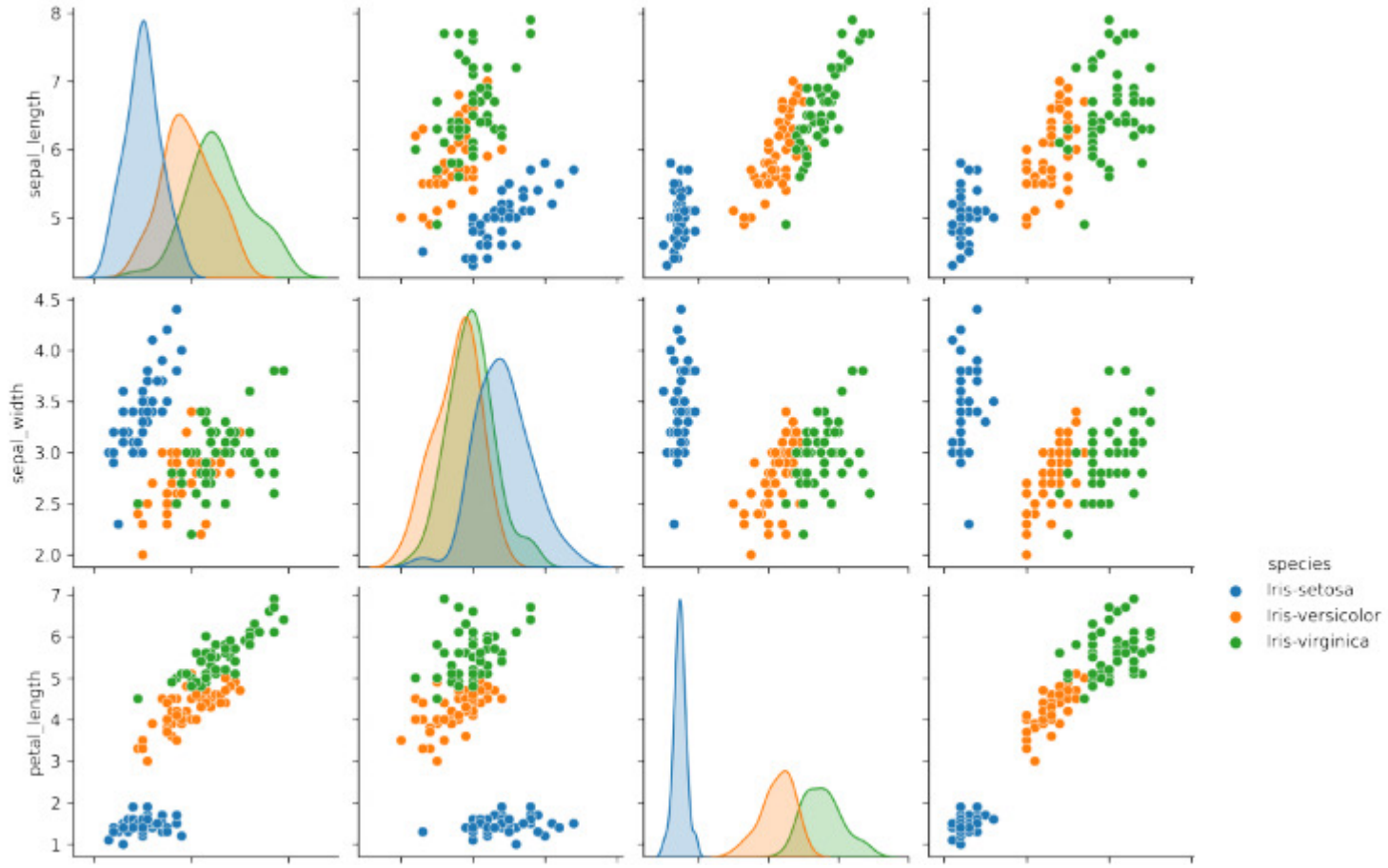
2 상관도 그래프

- **상관도 그래프**
  - 이번에는 시본(seaborn) 라이브러리에 있는 pairplot() 함수를 써서 전체 상관도를 볼 수 있는 그래프를 출력해 보자

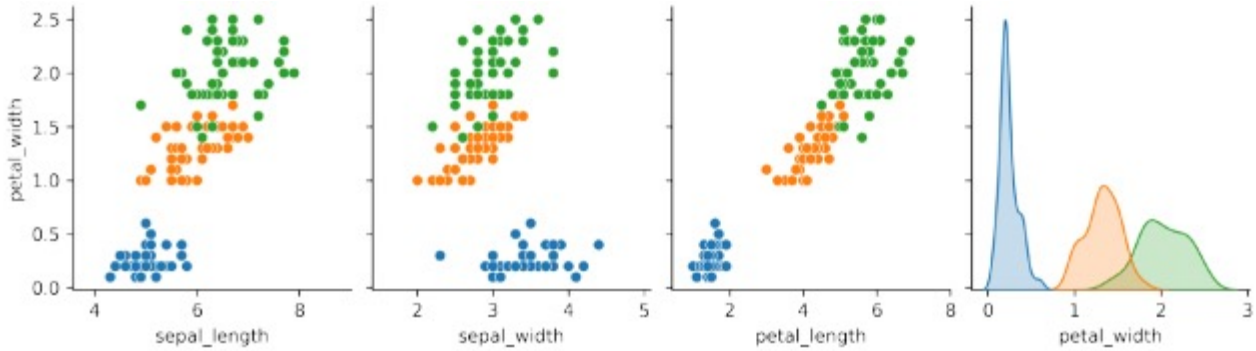
```
import seaborn as sns
import matplotlib.pyplot as plt

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

2 상관도 그래프

▼ 그림 12-3 | pairplot 함수로 데이터 한번에 보기



2 상관도 그래프


petal_width
sepal_length
sepal_width
petal_length
petal_width

2 상관도 그래프

- **상관도 그래프**
  - 이 그림을 상관도 그래프라고 함
  - 이를 통해 각 속성별 데이터 분포와 속성 간의 관계를 한눈에 볼 수 있음
  - pairplot() 함수 설정 중 hue 옵션은 주어진 데이터 중 어떤 카테고리를 중심으로 그래프를 그릴지 정해 주게 되는데, 우리는 품종(❶ species)에 따라 보여지게끔 지정
  - 그래프 각각의 가로축과 세로축은 서로 다른 속성을 나타내며, 이러한 속성에 따라 품종이 어떻게 분포되는지 알 수 있음
  - 가운데 대각선 위치에 있는 그림은 가로축과 세로축이 같으므로 단순히 해당 속성에 따라 각 품종들이 어떻게 분포하는지 보여 줌
  - 이러한 분석을 통해 사진상으로 비슷해 보이던 꽃잎과 꽃받침의 크기와 너비가 품종별로 어떤 차이가 있는지 알 수 있음

3 원-핫 인코딩

3 원-핫 인코딩

- **원-핫 인코딩**
  - 이제 케라스를 이용해 아이리스의 품종을 예측해 보자
  - Iris-setosa, Iris-virginica 등 데이터 안에 문자열이 포함되어 있음
  - 먼저 조금 전 불러온 데이터 프레임을 X와 y로 나누겠음

```
X = df.iloc[:,0:4]
y = df.iloc[:,4]
```

3 원-핫 인코딩

- **원-핫 인코딩**
  - X와 y의 첫 다섯 줄을 출력해 보자

```
print(X[0:5])
print(y[0:5])
```

3 원-핫 인코딩

- 원-핫 인코딩

실행 결과

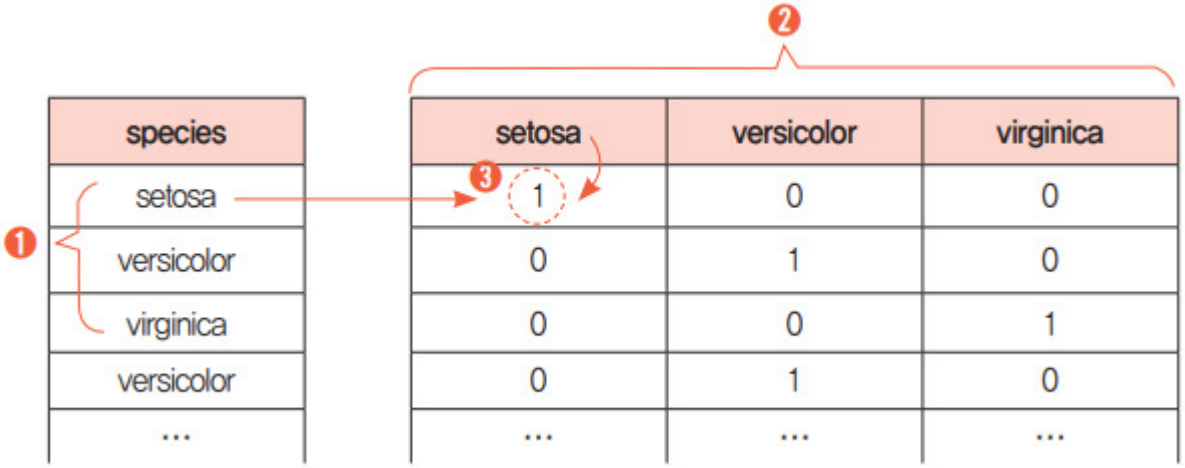
```
   sepal_length  sepal_width  petal_length  petal_width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
3    Iris-setosa
4    Iris-setosa
Name: species, dtype: object
```

3 원-핫 인코딩

- **원-핫 인코딩**
  - 우리가 저장한 y의 값이 숫자가 아닌 문자
  - 딥러닝에서는 계산을 위해 **문자를 모두 숫자형**으로 바꾸어 주어야 함
  - 이를 위해서는 다음과 같이 처리
  - 먼저 아이리스 꽃의 종류는 ❶ 처럼 세 종류
  - ❷ 처럼 각각의 이름으로 세 개의 열을 만든 후 ❸ 처럼 자신의 이름이 일치하는 경우 1로, 나머지는 0으로 바꾸어 줌

3 원-핫 인코딩

▼ 그림 12-4 | 원-핫 인코딩



3 원-핫 인코딩

- **원-핫 인코딩**
  - 여러 개의 값으로 된 문자열을 0과 1로만 이루어진 형태로 만들어 주는 과정을 **원-핫 인코딩(one-hot encoding)**이라고 함
  - 원-핫 인코딩은 판다스가 제공하는 `get_dummies()` 함수를 사용하면 간단하게 해낼 수 있음

```
# 원-핫 인코딩 처리를 합니다.
y = pd.get_dummies(y)

# 원-핫 인코딩 결과를 확인합니다.
print(y[0:5])
```

3 원-핫 인코딩

- 원-핫 인코딩

실행 결과

	Iris-setosa	Iris-versicolor	Iris-virginica
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0

4 소프트맥스

4 소프트맥스

- **소프트맥스**
  - 이제 모델을 만들어 줄 차례
  - 다음 코드를 보면서 이전에 실행했던 피마 인디언의 당뇨병 예측과 무엇이 달라졌는지 찾아보기 바람

```
# 모델 설정
model = Sequential()
model.add(Dense(12, input_dim=4, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()

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

4 소프트맥스

- **소프트맥스**
  - 세 가지가 달라졌음
  - 첫째 출력층의 노드 수가 3으로 바뀜
  - 활성화 함수가 **softmax**로 바뀜
  - 마지막으로 컴파일 부분에서 손실 함수 부분이 **categorical_crossentropy**로 바뀜

4 소프트맥스

- **소프트맥스**
  - 먼저 출력 부분에 대해 알아보자
  - 이전까지 우리는 출력이 0~1 중 하나의 값으로 나왔음
  - 예를 들어 당뇨인지 아닌지에 대한 예측 값이 시그모이드 함수를 거치며 0~1 사이의 값 중 하나로 변환되어 0.5 이상이면 당뇨로, 이하이면 정상으로 판단
  - 이항 분류의 경우 출력 값이 하나면 됨
  - 이번 예제에서는 예측해야 할 값이 세 가지로 늘었음
  - 즉, 각 샘플마다 이것이 setosa일 확률, versicolor일 확률, 그리고 virginica일 확률을 따로따로 구해야 한다는 것
  - 예를 들어 예측 결과는 그림 12-5와 같은 형태로 나타남

4 소프트맥스

▼ 그림 12-5 | 소프트맥스

샘플		setosa일 확률	versicolor일 확률	virginica일 확률
1번 샘플	예측 실행	0.2	0.7	0.1
2번 샘플		0.8	0.1	0.1
3번 샘플		0.2	0.2	0.6

4 소프트맥스

- 이렇게 세 가지의 확률을 모두 구해야 하므로 시그모이드 함수가 아닌 다른 함수가 필요함
- 이때 사용되는 함수가 바로 소프트맥스 함수
- 소프트맥스 함수는 그림 12-5와 같이 각 항목당 예측 확률을 0과 1 사이의 값으로 나타내 주는데, 이때 각 샘플당 예측 확률의 총합이 1인 형태로 바꾸어 주게 됨(예를 들어 1번 샘플의 경우 0.2 + 0.7 + 0.1 = 1이 됨)
- activation란에 '**softmax**'라고 적어 주는 것으로 소프트맥스 함수를 바로 적용할수 있음
- 마찬가지로 손실 함수도 이전과는 달라져야 함
- 이항 분류에서 **binary_crossentropy**를 썼다면, 다항 분류에서는 **categorical_crossentropy**를 쓰면 됨

4 Tensorflow 활성화(activation) 함수

- `linear`
  - 디폴트 값으로 입력값과 가중치(w, b)로 계산된 값이 그대로 출력으로 나옴
- `sigmoid`
  - 이진 분류에서 출력층에서 주로 사용됨
- `softmax`
  - 다중 클래스 분류에서 출력층에서 주로 사용됨
- `relu`
  - 은닉층에서 주로 사용됨

5 아이리스 품종 예측의 실행

5 아이리스 품종 예측의 실행

- **아이리스 품종 예측의 실행**
  - 이제 모든 소스 코드를 모아 보면 다음과 같음

실습 | 아이리스 품종 예측하기

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

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

5 아이리스 품종 예측의 실행

- 아이리스 품종 예측의 실행

```
# 깃허브에 준비된 데이터를 가져옵니다.
!git clone https://github.com/taehojo/data.git

# 아이리스 데이터를 불러옵니다.
df = pd.read_csv('./data/iris3.csv')

# 속성을 X, 클래스를 y로 저장합니다.
X = df.iloc[:,0:4]
y = df.iloc[:,4]

# 원-핫 인코딩 처리를 합니다.
y = pd.get_dummies(y)
```

5 아이리스 품종 예측의 실행

- 아이리스 품종 예측의 실행

```
# 모델 설정
model = Sequential()
model.add(Dense(12, input_dim=4, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()

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

# 모델 실행
history = model.fit(X, y, epochs=50, batch_size=5)
```

5 아이리스 품종 예측의 실행

- 아이리스 품종 예측의 실행

실행 결과

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 12)                60
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 104
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 27
=================================================================
```

5 아이리스 품종 예측의 실행

- 아이리스 품종 예측의 실행

```
Total params: 191
Trainable params: 191
Non-trainable params: 0


_________________________________________________________________
Epoch 1/30
30/30 [==============================] - 2s 2ms/step - loss: 1.4888 - accu
racy: 0.3333
... (중략) ...
Epoch 30/30
30/30 [==============================] - 0s 2ms/step - loss: 0.2746 - accu
racy: 0.9667
```

5 아이리스 품종 예측의 실행

- **아이리스 품종 예측의 실행**
  - model.summary()를 사용해 두 개의 은닉층에 각각 12개와 여덟 개의 노드가 만들어졌고, 출력은 세 개임을 확인할 수 있음
  - 결과는 30번 반복했을 때 정확도가 96.0% 나왔음
  - 꽃의 너비와 길이를 담은 150개의 데이터 중 144개의 꽃 종류를 정확히 맞추었다는 의미
  - 이제부터는 이렇게 측정된 정확도를 어떻게 신뢰할 수 있는지, 예측 결과의 신뢰도를 높이는 방법에 대해 알아보자