뉴스 기사 분류 : 다중 분류 문제

각 데이터 포인트가 여러 개의 범주에 속할 수 있다면 이것은 **다중 레이블 다중 분류** 문제가 된다.

로이터 데이터셋

텍스트 분류를 위해 널리 사용되는 간단한 데이터셋이다.

```
from keras.datasets import reuters

(train_data, train_labels),
(test_data, test_labels) = reuters.load_data(num_words=10000)
```

데이터에서 가장 자주 등장하는 단어 1만 개로 제한합니다.

8982개의 훈련 샘플과 2246개의 테스트 샘플이 있다. 각 샘플은 정수 리스트이다.

로이터 데이터셋을 텍스트로 디코딩하기

```
word_index = reuters.get_word_index()
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
decoded_newswire = ' '.join([reverse_word_index.get(i - 3, '?') for i in train_data[0]])
```

데이터 준비

데이터 인코딩하기

```
import numpy as np

def vectorize_sequences(sequences, dimension=10000):
  results = np.zeros((len(sequences), dimension))
  for i, sequence in enumerate(sequences):
    results[i, sequence] = 1.
  return results

x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

레이블을 벡터로 바꾸는 방법은 레이블의 리스트를 정수 텐서로 변환하는 것과 원-핫 인코딩을 사용하는 것이다. 원-핫 인코딩이 범주형 데이터에 널리 사용되기 때문에 **범주형 인코딩**이라고도 부른다.

모델 구성

모델 정의하기

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(46, activation='softmax'))
```

출력 클래스의 개수가 46개이다. 차원이 훨씬 크다. 46개의 클래스를 구분하기에는 규모가 큰 중간 층이 필요하다. 64개의 유닛을 사용할 것이다.

마지막 Dense의 크기가 46이다. 각 입력 샘플에 대해 46차원의 벡터를 출력한다는 뜻이다. 이 벡터의 각 원소는 각기 다른 출력 클래스가 인코딩된 것이다.

또한 마지막 층에 softmax 활성화 함수가 사용되었다. 각 입력 샘플마다 46개의 출력 클래스에 대한 확률 분포를 출력한다. 즉 46차원의 출력 벡터를 만들며 output[i]는 어떤 샘플이 클래스 i에 속할 확률이다.

모델 컴파일하기

```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

이런 문제에 사용할 최선의 손실 함수는 categorical_crossentropy이다. 이 함수는 두 확률 분포 사이의 거리를 측정한다. 여기에서는 네트워크가 출력한 확률 분포와 진짜 레이블의 분포 사이의 거리이다. 두 분포 사이의 거리를 최소화하면 진짜 레이블에 가능한 가까운 출력을 내도록 모델을 훈련하게 된다.

훈련 검증

훈련 데이터에서 1,000개의 샘플을 따로 떼어서 검증 세트로 사용하겠다.

```
x_val = x_train[:1000]
partial_x_train = x_train[1000:]

y_val = one_hot_train_labels[:1000]
partial_y_train = one_hot_train_labels[1000:]
```

20번의 에포크로 모델을 훈련시킨다.

```
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

훈련과 검증 손실 그리기

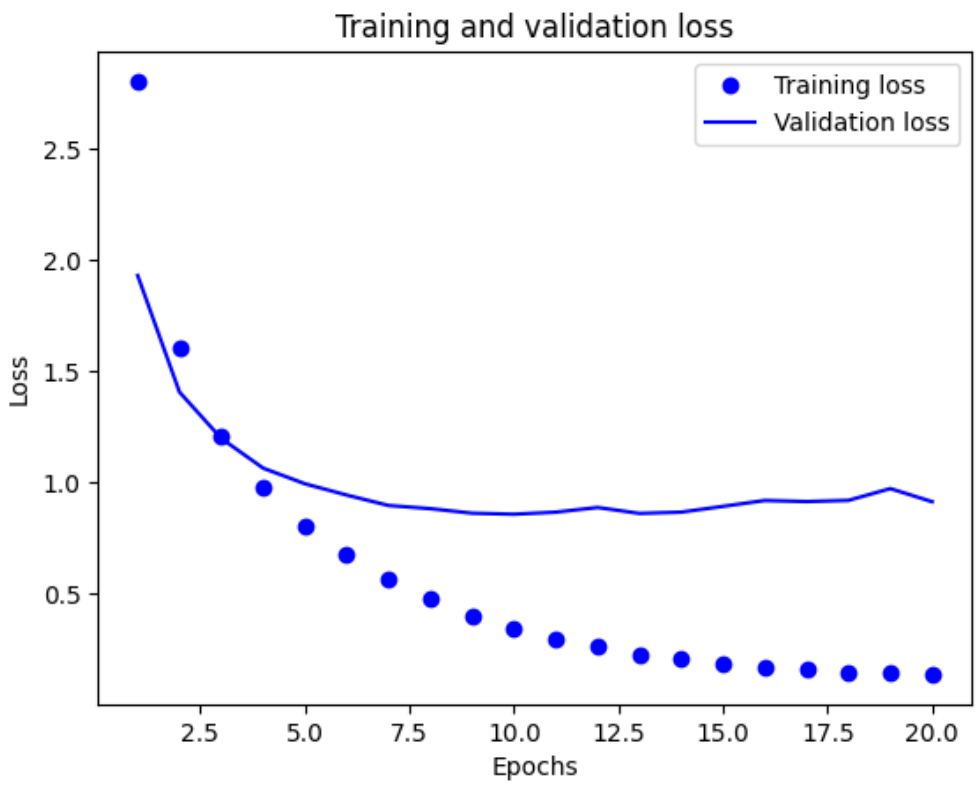
```
import matplotlib.pyplot as plt

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(loss) + 1)

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

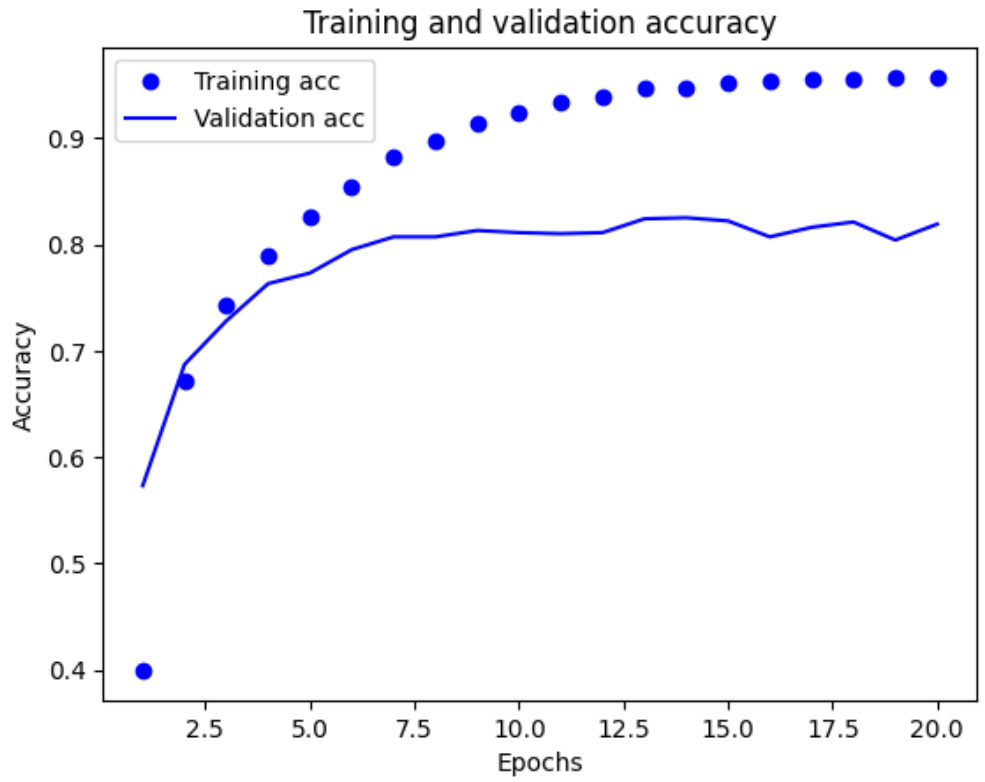


훈련과 검증 정확도 그리기

```
plt.clf()

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



💡 아홉 번째 에포크에서 과대적합이 시작되기 때문에, 아홉 번의 에포크로 새로운 모델을 훈련하고 테스트 세트에서 평가한다.

모델을 처음부터 다시 훈련하기

```
model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(46, activation='softmax'))
```

```
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(partial_x_train, partial_y_train, epochs=9, batch_size=512, validation_data=(x_val, y_val))
results = model.evaluate(x_test, one_hot_test_labels)

최종 결과는 약 78센트의 정확도를 가진다.
>>> results
[0.973463773727417, 0.7849510312080383]
```

정리

- N개의 클래스로 데이터 포인트를 분류하려면 네트워크의 마지막 Dense 층의 크기는 N이어야 한다.
- 단일 레이블, 다중 분류 문제에서는 N개의 클래스에 대한 확률 분포를 출력하기 위해 softmax 활성화 함수를 사용해야 한다.
    - 활성화 함수가 필요한 이유 ?
    - relu와 같은 활성화 함수가 없다면 Dense 층은 선형적인 연산인 점곱과 덧셈 2개로 구성된다. 가설 공간을 풍부하게 만들어 층을 깊게 만드는 장점을 살리기 위해서는 비선형성 또는 활성화 함수를 추가해야 한다. relu는 딥러닝에서 가장 인기 있는 활성화 함수이다.
- 이런 문제에는 항상 범주형 크로스엔트로피를 사용해야 한다. 이 함수는 모델이 출력한 확률 분포와 타깃 분포 사이의 거리를 최소화한다.
- 다중 분류에서 레이블을 다루는 두 가지 방법이 있다.
    - 레이블을 범주형 인코딩(또는 원-핫-인코딩)으로 인코딩하고 categorical_crossentropy 손실함수를 사용해야 한다.
    - 레이블을 정수로 인코딩하고 sparse_categorical_crossentropy 손실 함수를 사용해야 한다.
- 많은 수의 범주를 분류할 때 중간층의 크기가 너무 작아 네트워크에 정보의 병목이 생기지 않도록 해야 한다.