인공신경망과딥러닝심화

Lecture 10. 딥러닝 모델 설계하기

동덕여자대학교
데이터사이언스 전공
권 범

목차

01. 모델의 정의

02. 입력층, 은닉층, 출력층
03. 모델 컴파일
04. 모델 실행하기

01. 모델의 정의

❖ **지금까지 살펴본 것**

- 가끔 어려운 수식이 나오기도 했지만, 개념들이 머릿속에 정리되었다면
  지금부터는 텐서플로, 케라스와 함께 딥러닝의 세계로 나아갈 것임
- 지금부터는 딥러닝의 기본 개념들이 **어떤 방식으로 구현**되는지,
  왜 우리가 그 어려운 개념들을 익혀야 했는지 공부하겠음

- Lecture 02에서 '폐암 수술 환자의 생존율 예측하기' 예제를 소개한 적이 있었음
  당시에는 딥러닝 모델 부분을 자세히 설명할 수 없었지만, 이제 설명할 수 있음

딥러닝의 개념들이 짧은 코드 안에 모두 들어 있음.
어떻게 활용되는지 지금부터 함께 알아보자!

01. 모델의 정의

❖ '폐암 수술 환자의 생존율 예측하기'의 딥러닝 코드 (1/3)

- Lecture 02에서 살펴봤던 딥러닝 코드를 다시 한 번 옮겨 보면 다음과 같음

```
1  # 텐서플로의 케라스 API에서 필요한 함수들을 불러옵니다.
2  # 데이터를 다루는데 필요한 라이브러리를 불러옵니다.
3  from tensorflow import keras
4  from keras import Sequential, Input
5  from keras.layers import Dense
6  import numpy as np
7
8  # 준비된 수술 환자 데이터를 불러옵니다.
9  dataset = np.loadtxt("./data/ThoracicSurgery3.csv", delimiter=',')
10 x = dataset[:, 0:16]    # 환자의 진찰 기록을 변수 x에 저장합니다.
11 y = dataset[:, 16]      # 수술 후 사망/생존 여부를 변수 y에 저장합니다.
12
13
14
15
16
17
```

1. 환경 준비

2. 데이터 준비

01. 모델의 정의

❖ '폐암 수술 환자의 생존율 예측하기'의 딥러닝 코드 (2/3)

```
18 # 딥러닝 모델의 구조를 결정합니다.
19 model = Sequential()
20 model.add(Input(shape=(16,)))
21 model.add(Dense(30, activation="relu"))
22 model.add(Dense(1, activation="sigmoid"))
23
24 # 딥러닝 모델을 실행합니다.
25 model.compile(loss="binary_crossentropy",
26               optimizer="adam",
27               metrics=["accuracy"])
28 history = model.fit(x, y, epochs=5, batch_size=16)
```

3. 구조 결정

4. 모델 실행

01. 모델의 정의

❖ '폐암 수술 환자의 생존율 예측하기'의 딥러닝 코드 (3/3)

- 이 코드에서 데이터를 불러오고 다루는 부분은 Lecture 02에서 이미 살펴보았으므로, 여기서는 실제로 딥러닝이 수행되는 부분을 더 자세히 알아보자

```
# 딥러닝 모델의 구조를 결정합니다.
model = Sequential()
model.add(Input(shape=(16,)))
model.add(Dense(30, activation="relu"))
model.add(Dense(1, activation="sigmoid"))
```

```
# 딥러닝 모델을 실행합니다.
model.compile(loss="binary_crossentropy",
              optimizer="adam",
              metrics=["accuracy"])
history = model.fit(x, y, epochs=5, batch_size=16)
```

✓ 딥러닝의 모델을 설정하고 구동하는 부분은 모두 model이라는 Sequential 구조체를 선언하며 시작
✓ 먼저 model = Sequential( )은 딥러닝 모델의 구조를 만들고 층을 설정
✓ 이어서 model.compile( )은 앞에서 정한 모델을 컴퓨터가 알아들을 수 있게끔 컴파일
✓ 그다음 model.fit( )은 모델을 학습시킴

02. 입력층, 은닉층, 출력층

01. 모델의 정의
03. 모델 컴파일
04. 모델 실행하기

02. 입력층, 은닉층, 출력층

❖ 코드 살펴보기 (1/6)

- 먼저 딥러닝 모델의 구조를 만들고, 층을 설정하는 부분을 살펴보면 다음과 같음

```
# 딥러닝 모델의 구조를 결정합니다.
model = Sequential()
model.add(Input(shape=(16,)))
model.add(Dense(30, activation="relu"))
model.add(Dense(1, activation="sigmoid"))
```

02. 입력층, 은닉층, 출력층

❖ 코드 살펴보기 (2/6)

- Sequential 모델

```
# 딥러닝 모델의 구조를 결정합니다.
model = Sequential()
model.add(Input(shape=(16,)))
model.add(Dense(30, activation="relu"))
model.add(Dense(1, activation="sigmoid"))
```

✓ 딥러닝이란 입력층과 출력층 사이에 은닉층들을 차곡차곡 추가하면서 학습시키는 것임을 배웠음
✓ 이 층들이 케라스에서는 Sequential( ) 모델을 통해 쉽게 구현
✓ Sequential( ) 모델을 model로 선언해 놓고 model.add( )라는 라인을 추가하면 새로운 층이 만들어짐
✓ 코드에는 model.add( )로 시작되는 라인이 세 개 있으므로 층을 세 개 가진 모델을 만든 것
✓ 첫 번째 층은 입력층, 두 번째 층은 은닉층, 맨 마지막 층은 결과를 출력하는 출력층이 됨
✓ 은닉층과 출력층은 Dense라는 함수를 통해 구체적으로 그 구조가 결정

02. 입력층, 은닉층, 출력층

❖ 코드 살펴보기 (3/6)

- 입력층과 shape

```
# 딥러닝 모델의 구조를 결정합니다.
model = Sequential()
model.add(Input(shape=(16,)))
model.add(Dense(30, activation="relu"))
model.add(Dense(1, activation="sigmoid"))
```

✓ Input( ) 함수에는 shape이라는 인자(Argument)가 나옴
✓ 이는 입력 데이터의 차원 즉, 입력 데이터의 모양(Shape)을 model에게 알려주는 것
✓ 우리가 다루고 있는 폐암 수술 환자의 생존 여부 데이터에는 입력 값이 16개 있음
✓ 첫 번째 입력층에서 shape=16을 적어 줌으로써, 입력 값 16개를 전달받아 이를 다음 은닉층으로 보낸다는 의미

02. 입력층, 은닉층, 출력층

❖ 코드 살펴보기 (4/6)

- Dense Layer(Densely-Connected Neural Network Layer)

```
# 딥러닝 모델의 구조를 결정합니다.
model = Sequential()
model.add(Input(shape=(16,)))
model.add(Dense(30, activation="relu"))
model.add(Dense(1, activation="sigmoid"))
```

✓ 이제 model.add(Dense(30, activation="relu")) 부분을 더 살펴보자
✓ model.add( ) 메서드를 통해 새로운 층을 만들고 나면,
Dense( ) 함수의 첫 번째 인자에 몇 개의 노드(Node)를 이 층에 만들 것인지 숫자를 적어 줌
✓ 노드란 앞서 소개된 가중합(Weighted Sum)에 해당하는 것으로
이전 층에서 전달된 변수와 가중치(Weight), 바이어스(Bias)가 하나로 모이게 되는 곳
✓ 하나의 층에 여러 개의 노드를 적절히 만들어 주어야 하는데,
30이라고 되어 있는 것은 이 층에 노드를 30개 만들겠다는 것

02. 입력층, 은닉층, 출력층

❖ 코드 살펴보기 (5/6)

- 출력층과 활성화 함수

```
# 딥러닝 모델의 구조를 결정합니다.
model = Sequential()
model.add(Input(shape=(16,)))
model.add(Dense(30, activation="relu"))
model.add(Dense(1, activation="sigmoid"))
```

✓ 이제 세 번째 나오는 model.add(Dense(1, activation="sigmoid"))를 보겠음
✓ 마지막 층이므로 이 층이 곧 출력층이 됨
✓ 출력 값을 하나로 정해서 보여 주어야 하므로 출력층의 노드 수는 한 개
✓ 이 노드에서 입력받은 값은 활성화 함수를 거쳐 최종 출력 값으로 나와야 함
✓ 여기서는 시그모이드(Sigmoid) 함수를 활성화 함수로 사용

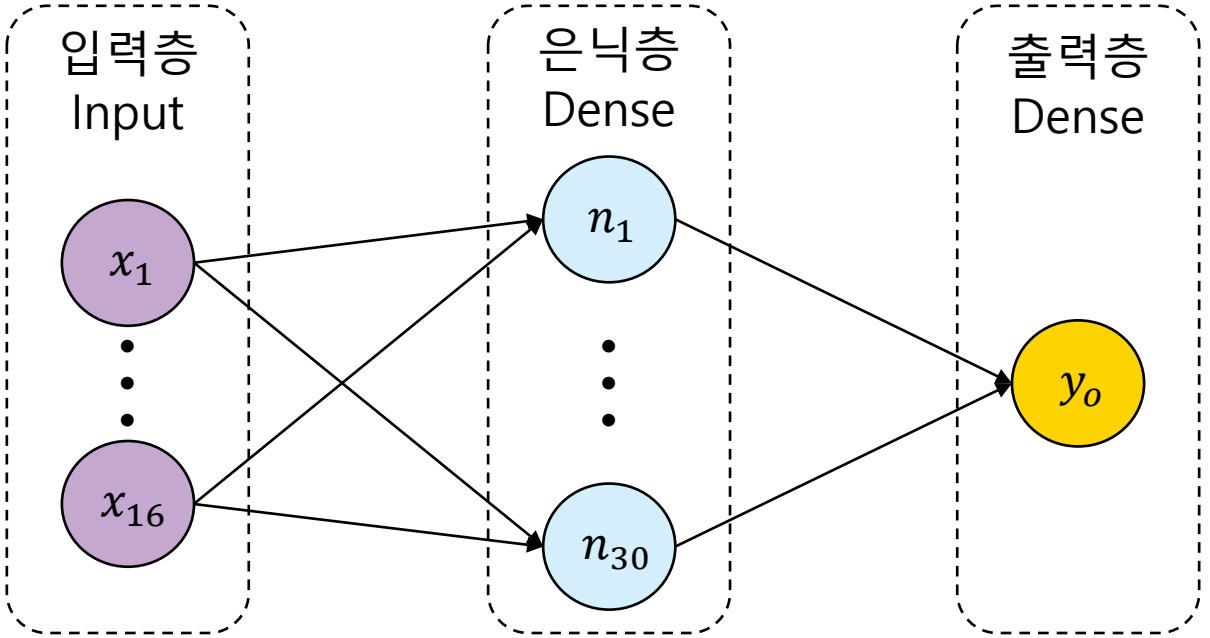
02. 입력층, 은닉층, 출력층

❖ 코드 살펴보기 (6/6)

- 프로그래밍을 통해 만든 인공신경망의 구조

```
# 딥러닝 모델의 구조를 결정합니다.
model = Sequential()
model.add(Input(shape=(16,)))
model.add(Dense(30, activation="relu"))
model.add(Dense(1, activation="sigmoid"))
```

Input은 입력층, 첫 번째 Dense는 은닉층, 그리고 두 번째 Dense는 출력층을 의미



03. 모델 컴파일

01. 모델의 정의
02. 입력층, 은닉층, 출력층
04. 모델 실행하기

03. 모델 컴파일

❖ 코드 살펴보기 (1/4)

- 다음으로 model.compile 부분

```
model.compile(loss="binary_crossentropy",
              optimizer="adam",
              metrics=["accuracy"])
```

03. 모델 컴파일

❖ 코드 살펴보기 (2/4)

- 손실 함수 정하기

```
model.compile(loss="binary_crossentropy",
              optimizer="adam",
              metrics=["accuracy"])
```

✓ model.compile( ) 부분은 앞서 지정한 모델이 효과적으로 구현될 수 있게 여러 가지 환경을 설정해 주면서 컴파일하는 부분
✓ 먼저 **어떤 손실 함수를 사용할지 정해야 함**
✓ Lecture 05에서 손실 함수에는 두 가지 종류가 있음을 배웠음
✓ 바로 ① **선형 회귀**에서 사용한 **평균 제곱 오차**와
② **로지스틱 회귀**에서 사용한 **교차 엔트로피 오차**
✓ 폐암 수술 환자의 생존율 예측은 생존과 사망, 둘 중 하나를 예측하므로 교차 엔트로피 오차 함수를 적용하기 위해 binary_crossentropy를 선택
✓ 손실 함수는 최적의 가중치를 학습하기 위해 필수적인 부분
✓ 올바른 손실 함수를 통해 계산된 오차는 옵티마이저를 적절히 활용하도록 만들어 줌

03. 모델 컴파일

❖ 코드 살펴보기 (3/4)

- 케라스는 쉽게 사용 가능한 여러 가지 손실 함수를 준비해 놓고 있음

✓ 크게 **평균 제곱 오차 계열**과 **교차 엔트로피 오차 계열**로 나뉘는데, 정리하면 아래 표와 같음
✓ 선형 회귀 모델은 평균 제곱 계열 중 하나를, 이항 분류를 위해서는 binary_crossentropy를, 그리고 다항 분류에서는 categorical_crossentropy를 사용한다는 것을 기억하자

대표적인 손실 함수

* 실제 값을 y_t, 예측 값을 y_o라고 가정할 때

평균 제곱 계열 (선형 회귀 모델)	mean_squared_error	평균 제곱 오차 계산: $mean(square(y_t - y_o))$
	mean_absolute_error	평균 절대 오차(실제 값과 예측 값 차이의 절댓값 평균) 계산: $mean(abs(y_t - y_o))$
	mean_absolute_percentage_error	평균 절대 백분율 오차(절댓값 오차를 절댓값으로 나눈 후 평균) 계산: $mean(abs(y_t - y_o)/abs(y_t))$ (단, 분모 ≠ 0)
	mean_squared_logarithmic_error	평균 제곱 로그 오차(실제 값과 예측 값에 로그를 적용한 값의 차이를 제곱한 값의 평균) 계산: $mean(square((log(y_o) + 1) - (log(y_t) + 1)))$
교차 엔트로피 계열 (다항 분류, 이항 분류)	categorical_crossentropy	범주형 교차 엔트로피(다항 분류, 여럿 중 하나를 예측할 때)
	binary_crossentropy	이항 교차 엔트로피(이항 분류, 둘 중 하나를 예측할 때)

[사진출처] 모두의 딥러닝 (출판사: 길벗, 저자: 조태호)

03. 모델 컴파일

❖ 코드 살펴보기 (4/4)

- 옵티마이저 정하기

```
model.compile(loss="binary_crossentropy",
              optimizer="adam",
              metrics=["accuracy"])
```

✓ 현재 가장 많이 쓰는 옵티마이저는 adam
✓ optimizer란에 adam을 적어 주는 것으로 실행할 준비가 됨 (예: optimizer="adam")
✓ metrics 인자(Argument)는 모델이 컴파일될 때, 모델 수행의 결과를 나타내게끔 설정하는 부분
✓ accuracy라고 설정한 것은 학습셋에 대한 정확도에 기반해 결과를 출력하라는 의미
(예: metrics=["accuracy"])
✓ accuracy 외에 학습셋에 대한 손실 값을 나타내는 loss,
테스트셋을 적용할 경우 테스트셋에 대한 정확도를 나타내는 val_acc,
테스트셋에 대한 손실 값을 나타내는 val_loss 등을 사용할 수 있음

04. 모델 실행하기

01. 모델의 정의
02. 입력층, 은닉층, 출력층
03. 모델 컴파일

04. 모델 실행하기

❖ 코드 살펴보기 (1/3)

- 모델을 정의하고 컴파일하고 나면 이제 학습시킬 차례
- 앞서 컴파일 단계에서 결정한 환경에서 주어진 데이터를 불러 모델을 학습시킬 때는 model.fit( )을 사용함

```
history = model.fit(x, y, epochs=5, batch_size=16)
```

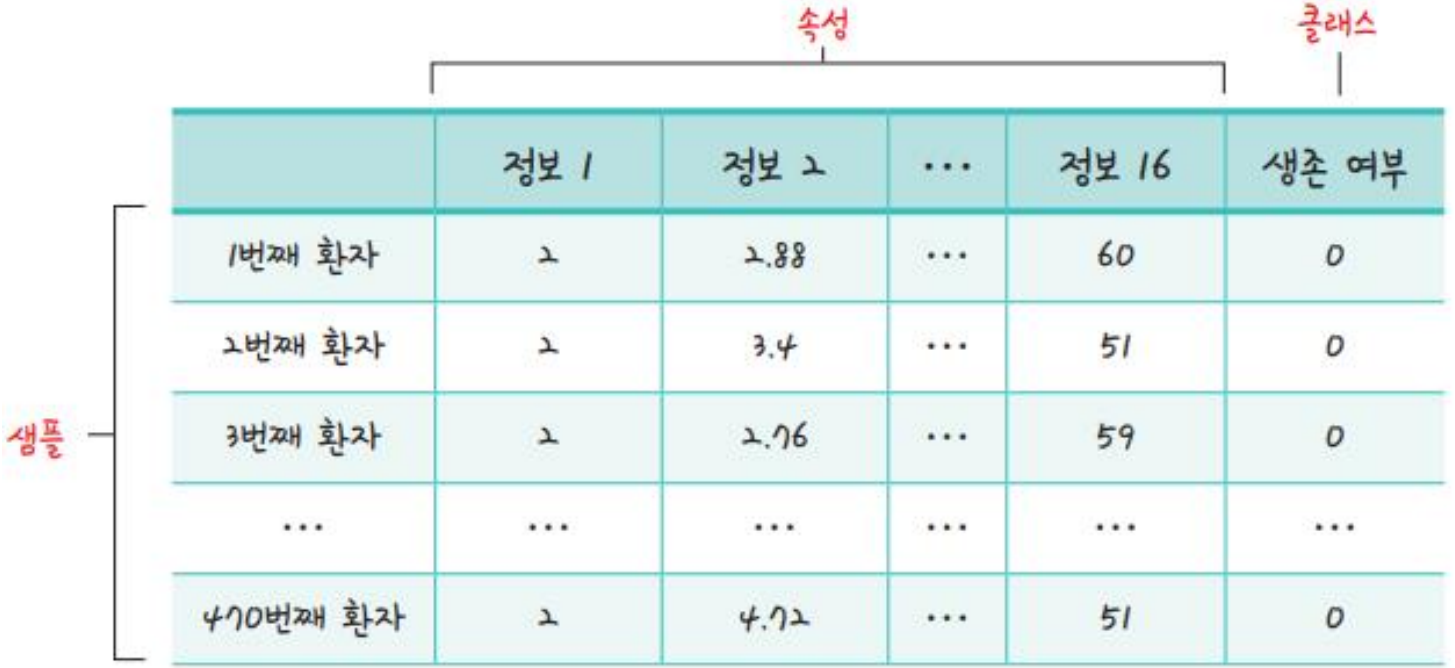
04. 모델 실행하기

❖ 코드 살펴보기 (2/3)

- model.fit( )을 설명하기에 앞서 용어를 다시 한 번 정리해 보자
- 주어진 폐암 수술 환자의 수술 후 생존 여부 데이터는 총 470명의 환자에게서 16개의 정보를 정리한 것

✓ 이때 각 정보를 속성(Attribute) 또는 **특성(Feature)**이라고 함
✓ 가로 한 줄에 해당하는 각 환자의 정보를 **샘플(Sample)**, Instance 또는 Example이라고 함
✓ 생존 여부를 **클래스(Class)** 또는 **레이블(Label)**이라고 함
✓ 즉, 주어진 데이터에는 총 470개의 샘플이 있으며, 각 샘플은 16개의 속성으로 구성되어 있음

폐암 환자 데이터의 샘플, 속성, 클래스 구분



	정보 1	정보 2	···	정보 16	생존 여부
1번째 환자	2	2.88	···	60	0
2번째 환자	2	3.4	···	51	0
3번째 환자	2	2.76	···	59	0
···	···	···	···	···	···
470번째 환자	2	4.12	···	51	0

[사진출처] 모두의 딥러닝 (출판사: 길벗, 저자: 조태호)

04. 모델 실행하기

❖ 코드 살펴보기 (3/3)

- 모델 학습시키기

```
history = model.fit(x, y, epochs=5, batch_size=16)
```

✓ 학습 프로세스가 모든 샘플에 대해 한 번 실행되는 것을 1 epoch('에포크'라고 읽음)라고 함
✓ epochs=5는 각 샘플이 처음부터 끝까지 다섯 번 재사용될 때까지 실행을 반복하라는 의미
✓ batch_size는 샘플을 한 번에 몇 개씩 처리할지 정하는 부분으로
batch_size=16은 전체 470개의 샘플을 16개씩 끊어서 집어넣으라는 의미
✓ batch_size가 너무 크면 학습 속도가 느려지고,
너무 작으면 각 실행 값의 편차가 생겨서 전체 결괏값이 불안정해질 수 있음
✓ 자신의 컴퓨터 메모리가 감당할 만큼의 batch_size를 찾아 설정해 주는 것이 좋음

끝맺음

- ❖ 01. 모델의 정의
- ❖ 02. 입력층, 은닉층, 출력층
- ❖ 03. 모델 컴파일
- ❖ 04. 모델 실행하기

THANK YOU!
Q & A

- Name: 권범
- Office: 동덕여자대학교 인문관 B821호
- Phone: 02-940-4752
- E-mail: bkwon@dongduk.ac.kr