딥러닝 실습

출처: 조태호, 모두의딥러닝, 길벗

1 | 딥러닝 실행을 위한 준비 사항

내 컴퓨터의 시스템 정보 확인하기

- 텐서플로는 64비트 윈도만을 지원하므로 사용하는 PC가 64비트인지 확인
- **시작** > **검색** > **제어판** > **시스템** 순서로 시스템 패널을 열어 확인

그림 1-2 내 컴퓨터의 시스템 정보 확인

2 | 딥러닝 작업 환경 만들기

아나콘다 설치하기

① 아나콘다 사이트에서 아나콘다3 64비트 인스톨러를 내려받은 후 설치
(파이썬 3.5 이상 버전을 선택)

- **https://www.continuum.io/downloads**

2 | 딥러닝 작업 환경 만들기

그림 1-3 아나콘다 설치

2 | 딥러닝 작업 환경 만들기

모두의 딥러닝

② 윈도 버튼을 누른 후 방금 전에 설치한 아나콘다 프로그램 폴더 중 Anaconda Prompt 선택

프로그램 폴더가 안 보이면 검색( )버튼을 누르고 Anaconda Prompt 입력

그림 1-4 Anaconda Prompt를 선택

2 | 딥러닝 작업 환경 만들기

② pip install tensorflow를 입력하고 **Enter**를 눌러 텐서플로를 설치

그림 1-7 텐서플로 설치

2 | 딥러닝 작업 환경 만들기

③ 텐서플로가 제대로 설치됐는지 확인하려면 python을 실행한 다음 import tensorflow as tf를 입력

그리고 print(tf.__version__)을 입력했을 때 텐서플로의 버전이 출력되면 설치가 완료된 것

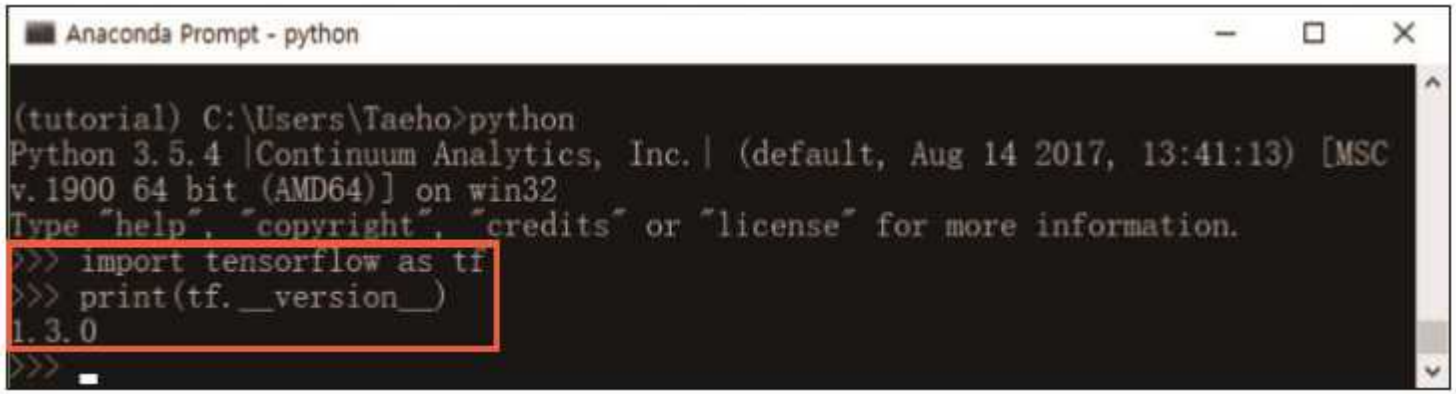

그림 1-8 텐서플로 설치 여부 확인

2 | 딥러닝 작업 환경 만들기

케라스 설치하기

① exit() 명령으로 대화형 셸을 빠져나간 다음 pip install keras라고 입력하여 케라스를 설치

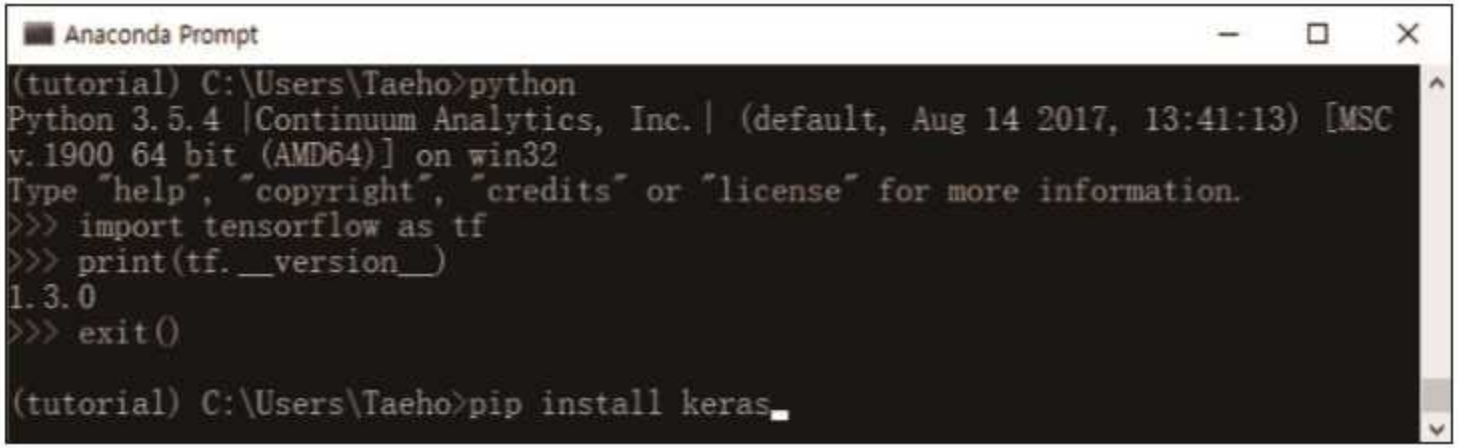

그림 1-9 케라스 설치

11장 데이터 다루기

[실습] 피마 인디언 당뇨병 예측

1 | 딥러닝과 데이터

2 | 피마 인디언 데이터 분석하기

3 | panda를 활용한 데이터 조사

4 | 데이터 가공하기

5 | matplotlib를 이용해 그래프로 표현하기

6 | 피마 인디언의 당뇨병 예측 실행

1 | 딥러닝과 데이터

모두의 딥러닝

- **실습 데이터** 피마 인디언 당뇨병 예측 : datadset/pima-indians-diabetes.csv

- 머신러닝과 딥러닝을 통해 프로젝트를 시작할 때 성공과 실패 여부는 ?
  - → 얼마나 좋은 데이터를 확보하느냐에 달려 있다
  - → 좋은 데이터란 실제 세상을 담고 있는 정보!

- 딥러닝 기술을 공부할 때는 좋은 데이터를 먼저 구해놓고 이를 통해 여러 가지 테크닉을 적용해 보는 연습이 필요

2 | 피마 인디언 데이터 분석하기

모두의 딥러닝

- 비만은 유전일까? 아니면 식습관 조절에 실패한 자신의 탓일까?
- 비만이 유전 및 환경, 모두의 탓이라는 것을 증명하는 좋은 사례가 바로 미국 남서부에 살고 있는 피마 인디언의 사례
- 피마 인디언은 1950년대까지만 해도 비만인 사람이 단 한 명도 없는 민족이었음
- 그런데 지금은 전체 부족의 60%가 당뇨, 80%가 비만으로 고통받고 있음
- 이는 생존하기 위해 영양분을 체내에 저장하는 뛰어난 능력을 물려받은 인디언들이 미국의 기름진 패스트푸드 문화를 만나면서 벌어진 일
- 피마 인디언을 대상으로 당뇨병 여부를 측정한 데이터를 딥러닝 저장소에서 내려받아 보자

2 | 피마 인디언 데이터 분석하기

모두의 딥러닝

- 데이터의 내용을 들여다 보기

- 파이참을 통해 열어 보면 모두 768명의 인디언으로부터 8개의 정보와 1개의 클래스를 추출한 데이터임을 알 수 있음

	속성					클래스
	정보 1	정보 2	정보 3	…	정보 8	당뇨병 여부
샘플 1번째 인디언	6	148	72	…	50	1
2번째 인디언	1	85	66	…	31	0
3번째 인디언	8	183	64	…	32	1
…	…	…	…	…	…	…
768번째 인디언	1	93	70	…	23	0

표 11-1 피마 인디언 데이터의 샘플, 속성, 클래스 구분

2 | 피마 인디언 데이터 분석하기

- '정보 1~8'과 '클래스'는 데이터와 함께 제공되는 Data Set Description(데이터셋 상세 설명)을 통해 알아냄

  - 샘플 수: 768
  - 속성: 8
    - 정보 1 (pregnant): 과거 임신 횟수
    - 정보 2 (plasma): 포도당 부하 검사 2시간 후 공복 혈당 농도(mm Hg)
    - 정보 3 (pressure): 확장기 혈압(mm Hg)
    - 정보 4 (thickness): 삼두근 피부 주름 두께(mm)
    - 정보 5 (insulin): 혈청 인슐린(2-hour, mu U/ml)
    - 정보 6 (BMI): 체질량 지수(BMI, weight in kg/(height in $m)^2$)
    - 정보 7 (pedigree): 당뇨병 가족력
    - 정보 8 (age): 나이
  - 클래스: 당뇨(1), 당뇨 아님(0)

3 | panda를 활용한 데이터 조사

- 데이터의 각 정보가 의미하는 의학, 생리학 배경 지식을 모두 알 필요는 없지만, 딥러닝을 구동하려면 반드시 속성과 클래스를 먼저 구분해야 함
- 모델의 정확도를 향상시키기 위해서는 데이터의 추가 및 재가공이 필요할 수도 있으므로 딥러닝의 구동에 앞서 데이터의 내용과 구조를 잘 파악하는 것이 중요
- 그런데 데이터의 크기가 커지고 정보량이 많아지면 데이터를 불러오고 내용을 파악할 수 있는 효과적인 방법이 필요함
- 이때 가장 유용한 방법이 데이터를 시각화해서 눈으로 직접 확인해 보는 것

3 | pandas를 활용한 데이터 조사

- pandas를 사용해 데이터를 불러와 보자

```
import pandas as pd
df = pd.read_csv('../dataset/pima-indians-diabetes.csv',
                 names = ["pregnant", "plasma", "pressure", "thickness",
                 "insulin", "BMI", "pedigree", "age", "class"])
```

- read_csv() 함수로 csv 데이터를 불러왔음
- csv 파일에는 데이터를 설명하는 한 줄의 라인, 즉 헤더(header)가 맨 처음에 나옴
- 그런데 우리가 가진 csv 파일에는 헤더가 없음
- 이에 names라는 함수를 통해 각 속성별 키워드를 지정해 주었음

3 | pandas를 활용한 데이터 조사

모두의 딥러닝

- 불러온 데이터의 내용을 간단히 확인하고자 head() 함수를 이용하여 데이터의 첫 5줄을 불러와 보자

```
print(df.head(5))
```

- 다음과 같이 출력됨, 정보마다 이름이 잘 지정된 것을 볼 수 있음

	pregnant	plasma	pressure	thickness	insulin	BMI	pedigree	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

3 | pandas를 활용한 데이터 조사

- 이제 이 데이터의 전반적인 정보를 확인해 보자
- 다음과 같이 df.info()를 출력

```
print(df.info())
```

3 | pandas를 활용한 데이터 조사

- 그러면 다음과 같은 정보가 화면에 출력됨
- 항목별로 768개가 빠짐없이 들어있음을 나타내고, 각 정보의 형식을 알려 줌

Range Index: 768 entries, 0 to 767

Data	Columns (total 9)		
pregnant	768	non-null	int64
plasma	768	non-null	int64
pressure	768	non-null	int64
thickness	768	non-null	int64
insulin	768	non-null	int64
BMI	768	non-null	float64
pedigree	768	non-null	float64
age	768	non-null	int64
class	768	non-null	int64
Dtypes: float64(2) int64(7)			
Memory usage: 54.1 KB			

3 | pandas를 활용한 데이터 조사

- 정보별 특징을 좀 더 자세히 알고 싶으면 describe() 함수를 이용

```
print(df.describe())
```

3 | pandas를 활용한 데이터 조사

- 다음과 같은 내용이 출력됨
- 정보별 샘플 수(count), 평균(mean), 표준편차 (std), 최솟값(min), 백분위 수로 25%, 50%, 75%에 해당하는 값 그리고 최댓값 (max)이 정리되어 보임

	pregnant	plasma	pressure	thickness	insulin	BMI	pedigree	age	class
count	768	768	768	768	768	768	768	768	768
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.88416	0.331329	11.760232	0.476951
min	0	0	0	0	0	0	0.078	21	0
25%	1	99	62	0	0	27.3	0.24375	24	0
50%	3	117	72	23	30.5	32	0.3725	29	0
75%	6	140.25	80	32	127.25	36.6	0.62625	41	1
max	17	199	122	99	846	67.1	2.42	81	1

3 | pandas를 활용한 데이터 조사

- 데이터의 일부 컬럼만 보고 싶을 때, 예를 들어 임신 횟수(pregnant)와 당뇨병 발병 여부(class)만 확인해 보고 싶다면 다음과 같이 입력

```
print(df[['pregnant', 'class']])
```

3 | pandas를 활용한 데이터 조사

- 그러면 다음과 같이 출력됨

	pregnant	class
0	6	1
1	1	0
2	8	1
3	1	0
4	0	1
5	5	0
...	...	...
765	5	0
766	1	1
777	1	0

4 | 데이터 가공하기

- 그런데 많은 값을 단순히 나열하는 것은 한눈에 들어오지 않으므로 큰 의미가 없다
  - → 데이터를 잘 다루려면 데이터를 한 번 더 가공해야 함

- 데이터를 가공할 때의 주의점
  - → 우리가 무엇을 위해 작업을 하는지 그 목적을 잊어서는 안 됨
  - → 이 프로젝트의 목적 : 당뇨병 발병을 예측하는 것
  - → 그렇다면 모든 정보는 당뇨병 발병과 어떤 관계가 있는지를 중점에 놓아야 함

4 | 데이터 가공하기

- 앞서 살펴본 임신 횟수와 당뇨병 발병 확률을 계산하는 방법

```
print(df[['pregnant','class']].groupby(['pregnant'],
as_index=False).mean().sort_values(by='pregnant', ascending=True))
```

→ 모두 세 가지 함수가 사용되었음

→ groupby 함수를 사용해 'pregnant' 정보를 기준으로 하는 새 그룹을 만들었음

→ as_index=False는 pregnant 정보 옆에 새로운 index를 만들어 줌

→ mean 함수를 사용해 평균을 구하고 sort_values 함수를 써서 pregnant 칼럼을 오름차순(ascending)으로 정리하게끔 설정

4 | 데이터 가공하기

모두의 딥러닝

- 이를 출력하면 다음과 같이 임신 횟수당 당뇨병 발병 확률을 구할 수 있음

	pregnant	class
0	0	0.342342
1	1	0.214815
2	2	0.184466
3	3	0.36
4	4	0.338235
5	5	0.368421
6	6	0.32
7	7	0.555556
8	8	0.578947
9	9	0.642857
10	10	0.416667
11	11	0.636364
12	12	0.444444
13	13	0.5
14	14	1
15	15	1
16	17	1

5 | matplotlib를 이용해 그래프로 표현하기

- 아무리 잘 정리된 데이터 테이블이라 하여도 그래프로 표현하지 않으면 정확한 성격을 파악하기 어려움

  → matplotlib는 파이썬에서 그래프를 그릴 때 가장 많이 사용되는 라이브러리

- matplotlib 라이브러리와 이를 기반으로 좀 더 정교한 그래프를 그리게끔 도와주는 seaborn 라이브러리를 사용해 각 정보끼리 어떤 상관관계가 있는지를 알아보자

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

5 | matplotlib를 이용해 그래프로 표현하기

- 먼저 그래프의 크기를 결정

```
plt.figure(figsize=(12,12))
```

- seaborn 라이브러리 중 각 항목 간의 상관관계를 나타내 주는 heatmap 함수를 통해 그래프를 표시해 보자
- heatmap 함수는 두 항목씩 짝을 지은 뒤 각각 어떤 패턴으로 변화하는지를 관찰하는 함수
- 두 항목이 전혀 다른 패턴으로 변화하고 있으면 0을, 서로 비슷한 패턴으로 변할수록 1에 가까운 값을 출력함

```
sns.heatmap(df.corr(), linewidths=0.1, vmax=0.5, cmap=plt.cm.gist_heat,
linecolor='white', annot=True)
```

5 | matplotlib를 이용해 그래프로 표현하기

모두의 딥러닝

- 이제 그래프를 다음과 같이 표시해 보자

```
plt.show()
```

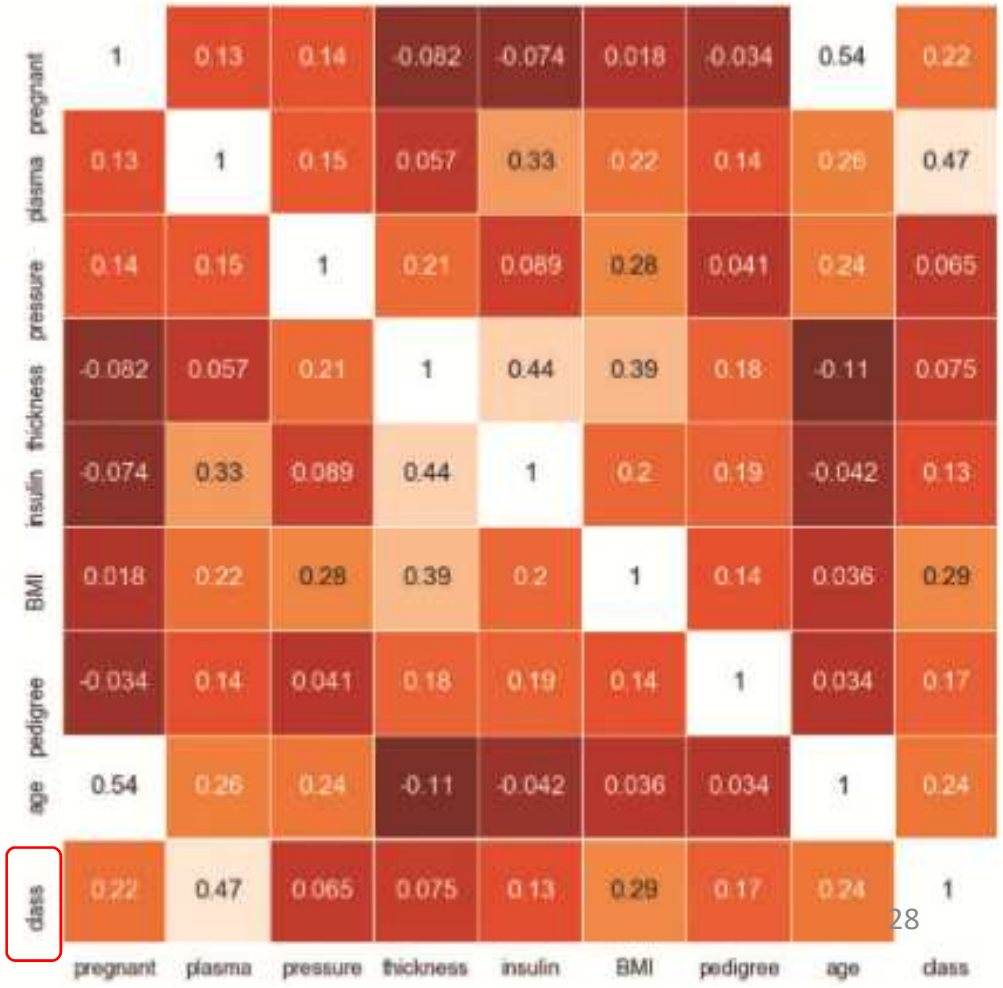

그림 11-6 정보 간 상관관계 그래프

5 | matplotlib를 이용해 그래프로 표현하기

- 그림 11-6에서 가장 눈여겨 봐야 할 부분은 당뇨병 발병 여부를 가리키는 class 항목
  - → class 항목을 보면 pregnant부터 age까지 상관도가 숫자로 표시되어 있고, 숫자가 높을수록 밝은 색상으로 채워져 있음
  - → 그래프를 통해 plasma 항목(공복 혈당 농도)이 class 항목과 가장 상관관계가 높다는 것을 알 수 있음
  - → 즉, 이 항목이 결론을 만드는 데 가장 중요한 역할을 한다는 것을 예측할 수 있음

5 | matplotlib를 이용해 그래프로 표현하기

- 이제 plasma와 class 항목만 따로 떼어 두 항목 간의 관계를 그래프로 다시 한번 확인해 보자

```
grid = sns.FacetGrid(df, col='class')
grid.map(plt.hist, 'plasma', bins=10)
plt.show()
```

5 | matplotlib를 이용해 그래프로 표현하기

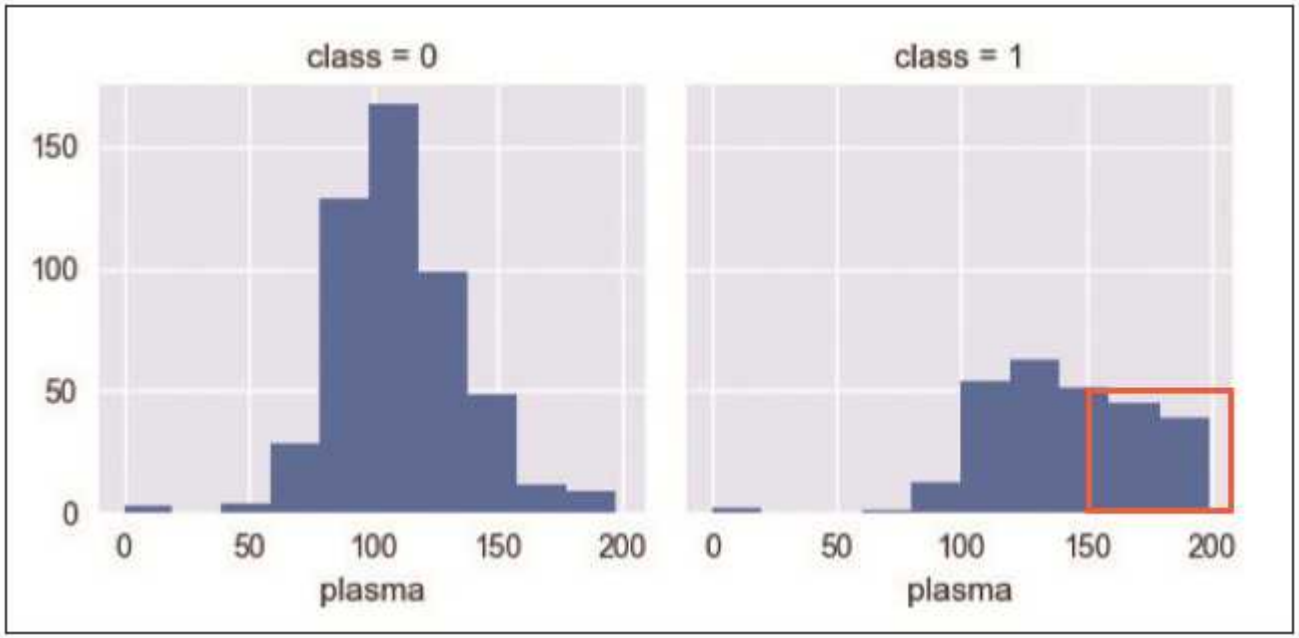

그림 11-7 plasma 정보와 class

5 | matplotlib를 이용해 그래프로 표현하기

- 이 그래프를 통해 당뇨병 환자의 경우(class=1) plasma 항목의 수치가 150 이상인 경우가 많다는 것을 알 수 있음
- 이렇게 결과에 미치는 영향이 큰 항목을 발견하는 것이 데이터 전처리 과정의 한 예
- 데이터 전처리 과정은 딥러닝을 비롯하여 모든 머신러닝의 성능 향상에 중요한 역할을 함

6 | 피마 인디언의 당뇨병 예측 실행

- 케라스를 이용해 당뇨병 예측을 실행하면 다음과 같음
- seed 값 설정 (매번 실행시 같은 결과값이 나올수있게 한다. 단 라이브러리 내부 랜덤 함수가 추가로 작동할 경우 100% 같은 값이 나오는 것을 보장 할 수는 없음)

```
# seed 값 생성
seed = 0
numpy.random.seed(seed)
tf.set_random_seed(seed)
```

6 | 피마 인디언의 당뇨병 예측 실행

- 이번에는 딥러닝 모델에 은닉층을 하나 더 추가해 보자
- 층을 추가할 때는 model.add 함수를 이용해 새로운 줄을 추가하기만 하면 됨

```
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

6 | 피마 인디언의 당뇨병 예측 실행

- 모델을 도식화하면 그림 11-8과 같음

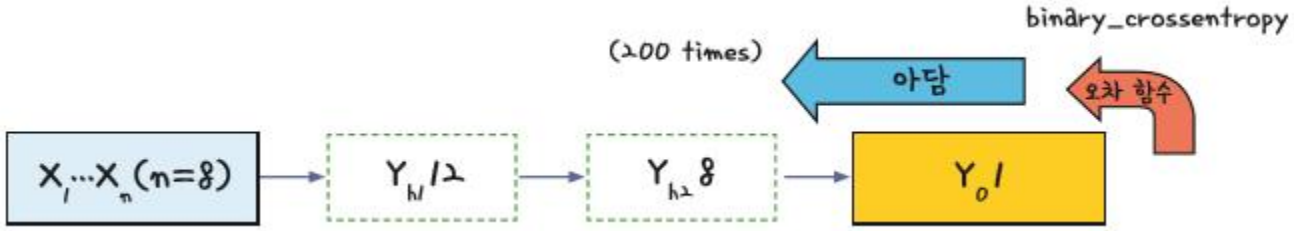

그림 11-8 피마 인디언 신경망 모델의 간단한 도식화

- 둘 중 하나를 결정하는 이항 분류(binary classification) 문제이므로 오차 함수는 binary_crossentropy를 사용하고, 최적화 함수로 adam을 사용
- 전체 샘플이 200번 반복해서 입력될 때까지 실험을 반복함
- 한 번에 입력되는 입력 값을 10개로 하고 종합하면 다음과 같은 프로그램이 완성됨

6 | 피마 인디언의 당뇨병 예측 실행

코드 11-2 피마 인디언의 당뇨병 예측하기

- **예제 소스** deep_code/02_Pima_Indian.py

```
from keras.models import Sequential
from keras.layers import Dense
import numpy
import tensorflow as tf

# seed 값 생성
seed = 0
numpy.random.seed(seed)
tf.set_random_seed(seed)

# 데이터 로드
dataset = numpy.loadtxt("../dataset/pima-indians-diabetes.csv", delimiter=",")
X = dataset[:,0:8]
Y = dataset[:,8]
```

6 | 피마 인디언의 당뇨병 예측 실행

```
# 모델의 설정
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

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

# 모델 실행
model.fit(X, Y, epochs=200, batch_size=10)

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

6 | 피마 인디언의 당뇨병 예측 실행

- **실행 결과**

```
Epoch 1/200
768/768 [==============================] - 0s - loss: 2.4216 - acc: 0.5117
Epoch 2/200
768/768 [==============================] - 0s - loss: 0.9143 - acc: 0.6393
Epoch 3/200
768/768 [==============================] - 0s - loss: 0.7944 - acc: 0.6393
Epoch 4/200
768/768 [==============================] - 0s - loss: 0.7408 - acc: 0.6055

(중략)
```

6 | 피마 인디언의 당뇨병 예측 실행

```
Epoch 197/200
768/768 [==============================] - 0s - loss: 0.4626 - acc: 0.7773
Epoch 198/200
768/768 [==============================] - 0s - loss: 0.4615 - acc: 0.7812
Epoch 199/200
768/768 [==============================] - 0s - loss: 0.4714 - acc: 0.7747
Epoch 200/200
768/768 [==============================] - 0s - loss: 0.4667 - acc: 0.7617
 32/768 [>.............................] - ETA: 0s
 Accuracy: 0.7786
```

- 약 77.86%의 예측 정확도를 보임