AI 프로그래밍

- 2024

10주차

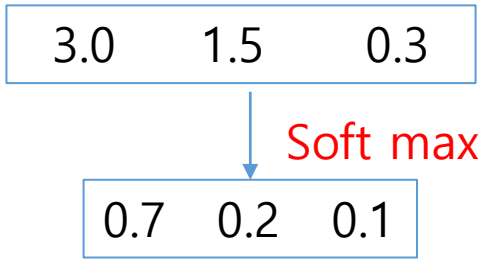
오늘 수업 내용

- **Soft max**
- **패션 MNIST 과제1**
- **타이타닉 생존자 예측 과제2**
  - 판다스 명령어
- **과적합 (Overfitting)**
- **와인 종류 예측**
  - 검증 셋, early stopping 과제 3
- **IMDB 영화 리뷰**
  - DROP OUT 과제 4
- **집값 예측**
  - 회귀 예측 과제 5

케라스 딥러닝 – 소프트 맥스

- 총합이 1인 형태로 변환

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



출력

```
Model.add(Dense(10,input_dim=4, activate='relu'))
Model.add(Dense(3,activation='softmax')
```

1.0 0.0 0.0 (one hot label) 정답

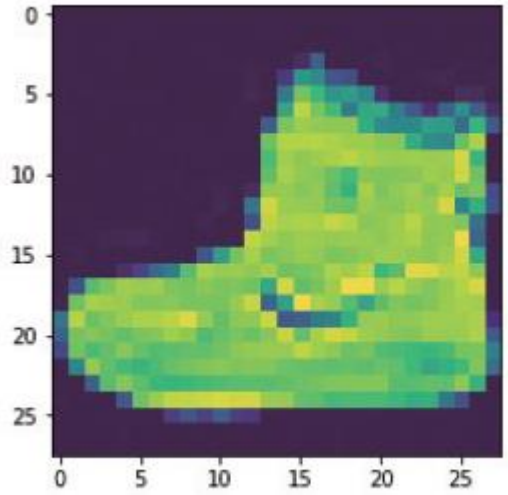
케라스 신경망 실습 – 패션 아이템 분류

- 이미지는 28x28 크기이고
- 픽셀 값은 0과 255 사이
- 레이블(label)은 0에서 9까지의

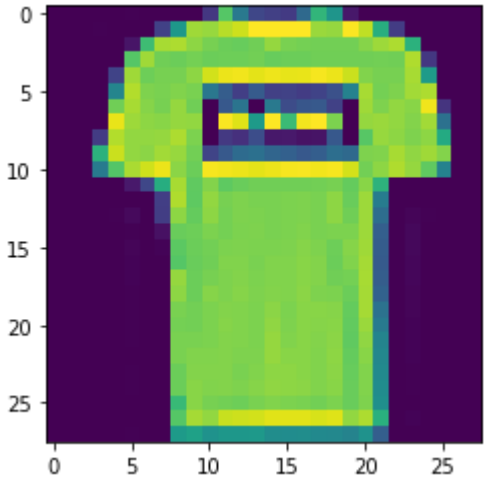
레이블	범주
0	T-shirt/top
1	trouser
2	pullover
3	dress
4	coat
5	sandal
6	shirt
7	sneaker
8	bag
9	Ankle boot

케라스 신경망 실습 – 패션 아이템 분류

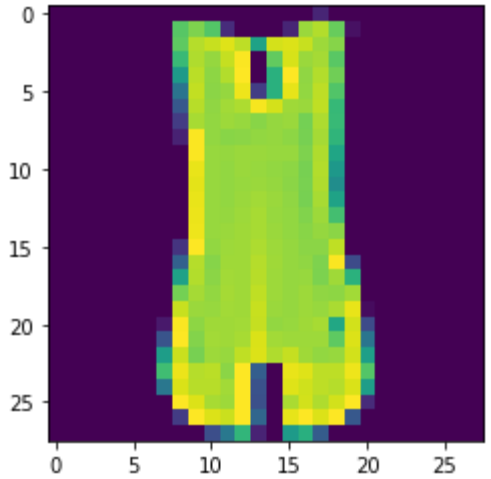
- **plt.imshow(train_images[0])**

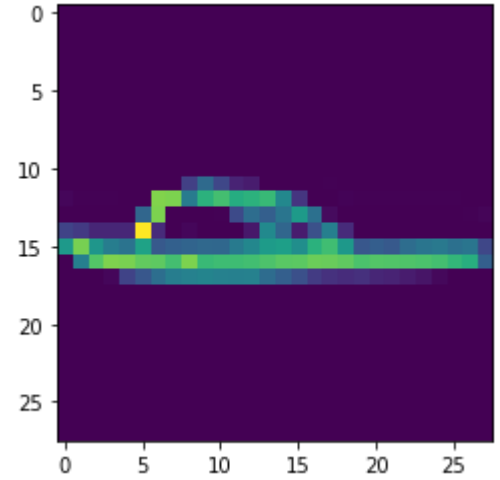


두번째



네번째





28x28 gray image

케라스 신경망 실습 – 패션 아이템 분류


```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, layers, models

fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

plt.imshow(train_images[0])

train_images = train_images / 255.0
test_images = test_images / 255.0
```

케라스 신경망 실습 – 패션 아이템 분류


```
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=5)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('정확도:', test_acc)
```

케라스 신경망 실습 – 패션 아이템 분류

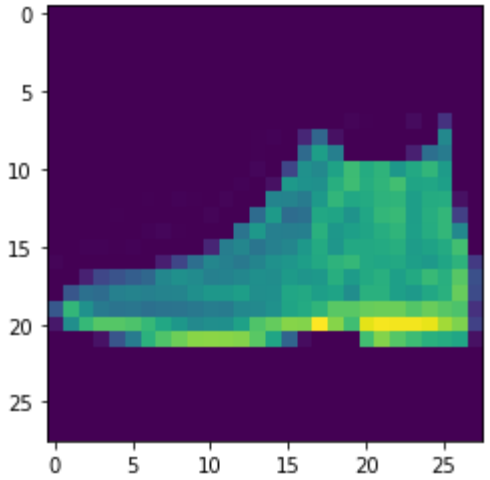

```
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(train_images, train_labels, epochs=5)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('정확도:', test_acc)
```

```
10000/10000 [==============================] - 0s 32us/sample - loss: 0.3560 -
acc: 0.8701
정확도: 0.8701
```

케라스 신경망 실습 – 패션 아이템 분류


```
test_pred=model.predict(test_images)
plt.imshow(test_images[0])
print(np.round(test_pred[0],2))
```

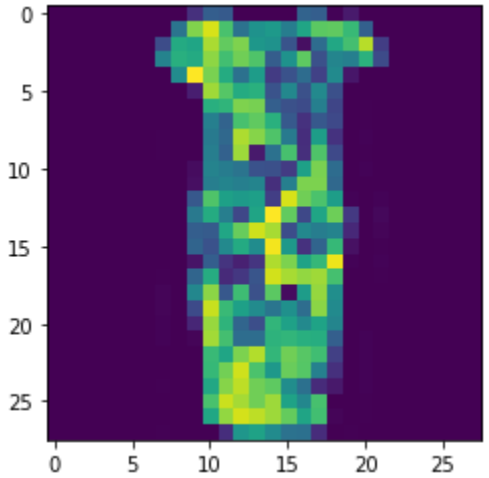
```
[0.  0.  0.  0.  0.  0.  0.  0.04  0.  0.96]
```



케라스 신경망 실습 – 패션 아이템 분류


```
plt.imshow(test_images[100])
print(np.round(test_pred[100],2))
```

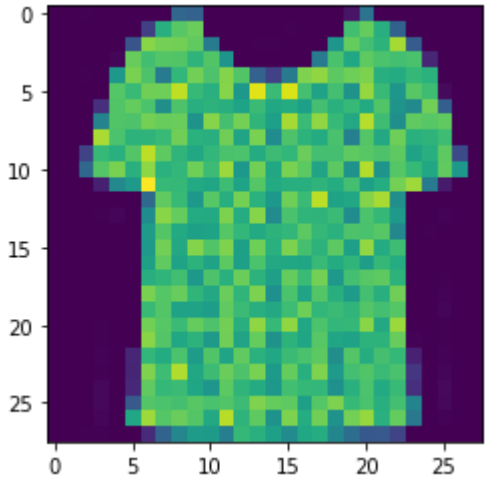
```
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
```



케라스 신경망 실습 – 패션 아이템 분류


```
plt.imshow(test_images[1000])
print(np.round(test_pred[1000],2))
```

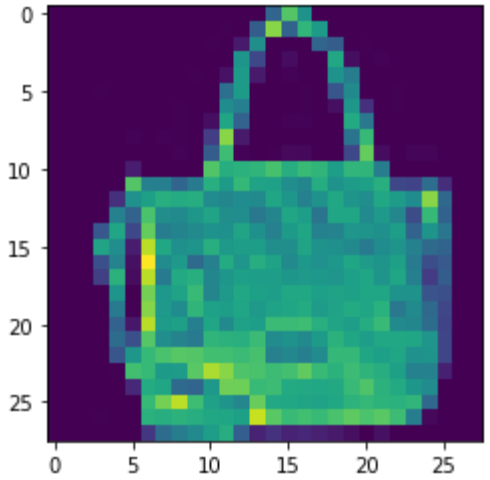
```
[0.43 0.   0.05 0.02 0.   0.   0.51 0.   0.   0.  ]
```



케라스 신경망 실습 – 패션 아이템 분류


```
plt.imshow(test_images[2000])
print(np.round(test_pred[2000],2))
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
```



케라스 신경망 실습 – 패션 아이템 분류

인하공전 컴퓨터 정보공학과

과제 1 fasion_org_exam.ipynb

모델을 다음과 같이 변경 하고 code (모델 생성 부분만)과 정확도를 제출 하시오.

1)

입력 층 : 변동 없음
은닉층 1 : 32 (relu)
출력 층 변동 없음.

2)

입력 층 : 변동 없음
은닉층 1 : 64 (relu)
은닉측 2 : 16 (relu)
출력 층 변동 없음.

3) 1050번째 test이미지의 사진을 붙이고 카테고리 예측 결과를 제출 하시오
4) 615번째 test 이미지와 사진을 붙이고 카테고리 예측 결과를 제출 하시오.

Part 1. 판다스 (pandas)

- 데이터를 머신러닝이나 시각화에 적절하도록 전처리 해주는 라이브러리
- 자료 구조: 데이터 프레임(DataFrame)

Import pandas as pd

판다스 명령어 - 1

	기능	code
기본 명령어	행 인덱스 변경	df.index=['학생1', '학생2']
	열이름 변경/지정	df.columns=['연령', '남녀', '소속']
	인덱스, 열이름 선택 변경	df.rename(columns={'나이':'연령', '성별':'남녀'}, inplace=True)
	행/열 삭제	df.drop(['영어', '음악'], axis=1, inplace=True)
	행 선택	df.loc['서준'], df.iloc[0], df.loc[['서준','우현']], df.iloc[0:2]
	열 선택	df['수학'], df.수학, df[['음악','체육']]
	원소 선택	df.iloc[0, 2], df.loc['서준', '음악'], df.loc['서준', '음악':'체육']
	행 추가	df.loc[3] = 0, df.loc[4] = ['동규', 90, 80, 70, 60]
	열 추가	df['국어'] = 80
	원소 값 변경	df.loc ['서준']['체육'] = 90, df.loc['서준',['음악','체육']] = 100, 50
	행, 열 위치 바꾸기	df.transpose(), df.T
	특정 열을 행 인덱스로 설정	df.set_index('음악'), df.set_index(['이름'])
	행 인덱스 재배열	df.reindex(new_index, fill_value=0)
	행 인덱스 초기화	df.reset_index()

판다스 명령어 - 2

	기능	code
기본 명령어	행 인덱스를 기준으로 정렬	df.sort_index(ascending=False)
	특정 열을 기준으로 정렬	df.sort_values(by='c1', ascending=False)
입출력	Csv 파일-> 데이터 프레임	pd.read_csv(file, header=None), pd.read_csv(file,header=0)
데이터 살펴보기	앞부분 뒷부분 미리보기	df.head(), df.tail(5)
	데이터 프레임의 크기(행, 열)	df.shape
	데이터프레임의 요약 정보(내용)	df.info()
	데이터프레임 자료형	df.dtypes
	데이터프레임 의 통계 정보	df.describe()
	각 열이 가지고 있는 원소 개수	df.count()
	고유값 개수	df['origin'].value_counts()
	평균값, 중간값, 최대값, 표준편차	df.mean(), df.median(), df.max(), df.std()
	상관 계수	df.corr(), df[['mpg','weight']].corr()
	고유값	df['horsepower'].unique()

판다스 명령어 - 3

	기능	code
데이터 전처리	NaN 개수 구하기	df['deck'].value_counts(dropna=False)
	NaN 찾기	df.isnull(), df.notnull()
	NaN 개수 구하기	df.isnull().sum(axis=0)
	NaN이 있는 모든 행 삭제	df.dropna(), df.dropna(how='any')
	행에 있는 모든 값이 NaN일 때 해당 행을 삭제	df.dropna(how='all')
	NaN이 있는 행 삭제시 데이터 개수 기준으로 삭제	df.dropna(axis=0, thresh=1)
	NaN 채우기 (값 지정)	df.fillna(0), df['age'].fillna(mean_age)
	NaN 채우기 (앞, 뒤 값으로 채우기)	df. fillna(method = 'ffill'), df. fillna(method = 'bfill')
	중복 데이터 확인	df.duplicated(), df['c2'].duplicated()
	중복 행 제거	df.drop_duplicates(), df.drop_duplicates(subset=['c2', 'c3'])
	데이터 타입 변경	df['horsepower'] = df['horsepower'].astype('float')
응용	함수 매핑 (열)	df.apply(myfunc, axis=0)
	함수 매핑 (행)	df['add'] = df.apply(lambda x: add_two (x['age'], x['ten']), axis=1)
	필터링 (boolean indexing)	mask1 = (titanic.age >= 10) titanic.loc[mask1, :]
	필터링 (isin)	isin_filter = titanic['sibsp'].isin([3, 4, 5])

Part 3. 데이터 살펴보기

1. 데이터프레임의 구조

```
import pandas as pd

# read_csv() 함수로 df 생성
df = pd.read_csv('./auto-mpg.csv', header=None)

# 열 이름을 지정
df.columns =
['mpg','cylinders','displacement','horsepower','weight',
                'acceleration','model year','origin','name']

# 데이터프레임 df의 내용을 일부 확인
print(df.head())      # 처음 5개의 행
print('₩n')
print(df.tail())      # 마지막 5개의 행
```

```
mpg  cylinders  displacement horsepower  weight  acceleration  model year  ₩
0  18.0          8          307.0       130.0  3504.0          12.0          70
1  15.0          8          350.0       165.0  3693.0          11.5          70
2  18.0          8          318.0       150.0  3436.0          11.0          70
3  16.0          8          304.0       150.0  3433.0          12.0          70
4  17.0          8          302.0       140.0  3449.0          10.5          70

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino


      mpg  cylinders  displacement horsepower  weight  acceleration  ₩
393  27.0          4         140.0       86.00  2790.0          15.6
394  44.0          4          97.0       52.00  2130.0          24.6
395  32.0          4         135.0       84.00  2295.0          11.6
396  28.0          4         120.0       79.00  2625.0          18.6
397  31.0          4         119.0       82.00  2720.0          19.4

     model year  origin                name
393          82       1  ford mustang gl
394          82       2        vw pickup
395          82       1    dodge rampage
396          82       1      ford ranger
397          82       1       chevy s-10
```

2-2. 데이터프레임

```
수학  영어   음악   체육
서준  90  98   85  100
우현  80  89   95   90
인아  70  95  100   90


수학     90
영어     98
음악     85
체육    100
Name: 서준, dtype: int64


수학     90
영어     98
음악     85
체육    100
Name: 서준, dtype: int64
```

2-2. 데이터프레임 ② 여러 개의 행을 선택(인덱스 리스트 활용)

- **행 선택 *(계속)***

```
# 행 인덱스를 사용하여 2개 이상의 행 선택
label2 = df.loc[['서준', '우현']]
position2 = df.iloc[[0, 1]]
print(label2)
print('₩n')
print(position2)
print('₩n')
```

```
    수학  영어  음악  체육
서준  90  98  85  100
우현  80  89  95   90


    수학  영어  음악  체육
서준  90  98  85  100
우현  80  89  95   90
```

1.1 드롭

- 드롭(drop) : 결측치가 나온 열이나 행을 삭제
- dropna 사용하여 NaN이 있는 모든 데이터의 행을 없앰

	first_name	last_name	age	sex	preTestScore	postTestScore
0	Jason	Miller	42.0	m	4.0	25.0
1	NaN	NaN	NaN	NaN	NaN	NaN
2	Tina	Ali	36.0	f	NaN	NaN
3	Jake	Milner	24.0	m	2.0	62.0
4	Amy	Cooze	73.0	f	3.0	70.0

In [3]:
```
df.dropna()
```

Out [3]:

	first_name	last_name	age	sex	preTestScore	postTestScore
0	Jason	Miller	42.0	m	4.0	25.0
3	Jake	Milner	24.0	m	2.0	62.0
4	Amy	Cooze	73.0	f	3.0	70.0

케라스 신경망 실습 – 타이타닉 생존자 예측

인하공전 컴퓨터 정보공학과

PassengerId : 각 승객의 고유 번호
Survived : 생존 여부(종속 변수)
- 0 = 사망
- 1 = 생존

Pclass : 객실 등급 - 승객의 사회적, 경제적 지위
- 1st = Upper
- 2nd = Middle
- 3rd = Lower

Name : 이름
Sex : 성별
Age : 나이
SibSp : 동반한 Sibling(형제자매)와 Spouse(배우자)의 수
Parch : 동반한 Parent(부모) Child(자식)의 수
Ticket : 티켓의 고유넘버
Fare : 티켓의 요금
Cabin : 객실 번호
Embarked : 승선한 항
- C = Cherbourg
- Q = Queenstown
- S = Southampton

케라스 신경망 실습 – 타이타닉 생존자 예측


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

```
# 데이터 세트를 읽어들인다.
train = pd.read_csv("train.csv", sep=',')
test = pd.read_csv("test.csv", sep=',')

# 필요없는 컬럼을 삭제한다.
train.drop(['SibSp', 'Parch', 'Ticket', 'Embarked', 'Name',\
        'Cabin', 'PassengerId', 'Fare', 'Age'], inplace=True, axis=1)

# 결손치가 있는 데이터 행은 삭제한다.
train.dropna(inplace=True)
```

케라스 신경망 실습 – 타이타닉 생존자 예측


```
>>> train.drop(['SibSp', 'Parch', 'Ticket', 'Embarked', 'Name',\
        'Cabin', 'PassengerId', 'Fare', 'Age'], inplace=True, axis=1)


>>> train.head()
   Survived  Pclass     Sex
0         0       3    male
1         1       1  female
2         1       3  female
3         1       1  female
4         0       3    male
```

케라스 신경망 실습 – 타이타닉 생존자 예측


```
# 기호를 수치로 변환한다.
for ix in train.index:
    if train.loc[ix, 'Sex']=="male":
        train.loc[ix, 'Sex']=1
    else:
        train.loc[ix, 'Sex']=0

# 2차원 배열을 1차원 배열로 평탄화한다.
target = np.ravel(train.Survived)

# 생존여부를 학습 데이터에서 삭제한다.
train.drop(['Survived'], inplace=True, axis=1)
train = train.astype(float)     # 최근 소스에서는 float형태로 형변환하여야
```

컴퓨터는 숫자만 처리할 수 있다. 딥러닝은 0부터 1 사이의 실수만 처리 가능

교과서 소스에 추가

케라스 신경망 실습 – 타이타닉 생존자 예측


```
# 케라스 모델을 생성한다.
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(16, activation='relu', input_shape=(2,)))
model.add(tf.keras.layers.Dense(8, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

# 케라스 모델을 컴파일한다.
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
# 케라스 모델을 학습시킨다.
model.fit(train, target, epochs=30, batch_size=1, verbose=1)
```

케라스 신경망 실습 – 타이타닉 생존자 예측


```
...
Epoch 29/30
891/891 [==============================] - 1s 753us/sample - loss: 0.4591 - acc:
0.7677
Epoch 30/30
891/891 [==============================] - 1s 753us/sample - loss: 0.4547 - acc:
0.7789
```

케라스 신경망 실습 – 타이타닉 생존자 예측

인하공전 컴퓨터 정보공학과

```
idx=test.shape[0]
pred=model.predict(test)
test_np=test.to_numpy()
for i in range(idx):
    print(test_np[i,:],pred[i])
```

객실 등급. 성별, 생존 확률

```
[3. 1.] [0.11963955]
[3. 0.] [0.42622244]
[2. 1.] [0.20062831]
[3. 1.] [0.11963955]
[3. 0.] [0.42622244]
[3. 1.] [0.11963955]
[3. 0.] [0.42622244]
[2. 1.] [0.20062831]
[3. 0.] [0.42622244]
[3. 1.] [0.11963955]
[3. 1.] [0.11963955]
[1. 1.] [0.35509673]
[1. 0.] [0.9667859]
[2. 1.] [0.20062831]
[1. 0.] [0.9667859]
[2. 0.] [0.84050065]
[2. 1.] [0.20062831]
[3. 1.] [0.11963955]
[3. 0.] [0.42622244]
[3. 0.] [0.42622244]
[1. 1.] [0.35509673]
[3. 1.] [0.11963955]
[1. 0.] [0.9667859]
[1. 1.] [0.35509673]
[1. 0.] [0.9667859]
[3. 1.] [0.11963955]
```

케라스 신경망 실습 – 타이타닉 생존자 예측

인하공전 컴퓨터 정보공학과

과제 2)　　titanic_test.ipynb　　test.csv, train.csv

모델을 다음과 같이 변경 하고 code (모델 생성 부분만)과 정확도(accuracy)를 제출 하시오.

1)

입력 층 : 변동 없음
은닉층 1 : 16 (relu)
은닉측 2 : 8 (relu)
은닉측 3 : 8 (relu)
출력 층 변동 없음.

2)

입력 층 : 변동 없음
은닉층 1 : 16 (relu)
은닉측 2 : 16 (relu)
은닉측 3 : 8 (relu)
출력 층 변동 없음.

피마 인디언 데이터 분석

인하공전 컴퓨터 정보공학과

샘플 수 : 768

•Pregnancies: 임신 횟수
•Glucose: 포도당 부하 검사 수치
•Pressure: 혈압(mm Hg)
•Thickness: 팔 삼두근 뒤쪽의 피하지방 측정값(mm)
•Insulin: 혈청 인슐린(mu U/ml)
•BMI: 체질량지수(체중(kg)/키(m))^2
•DiabetesPedigreeFunction: 당뇨 내력 가중치 값
•Age: 나이
•Class : 당뇨(1), 당뇨 아님

피마 인디언 데이터 분석


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

# pandas 라이브러리를 불러옵니다.
import pandas as pd

# 깃허브에 준비된 데이터를 가져옵니다. 이미 앞에서 가져왔으므로 주석
처리합니다. 5번 예제만 별도 실행 시 주석 해제 후 실습하세요.
# !git clone https://github.com/taehojo/data.git

# 피마 인디언 당뇨병 데이터셋을 불러옵니다.
df = pd.read_csv('./data/pima-indians-diabetes3.csv')


 # 세부 정보를 X로 지정합니다.
 X = df.iloc[:,0:8]
 # 당뇨병 여부를 y로 지정합니다.
 y = df.iloc[:,8]
```

080324/ch11.ipynb at master · gilbutITbook/080324 · GitHub

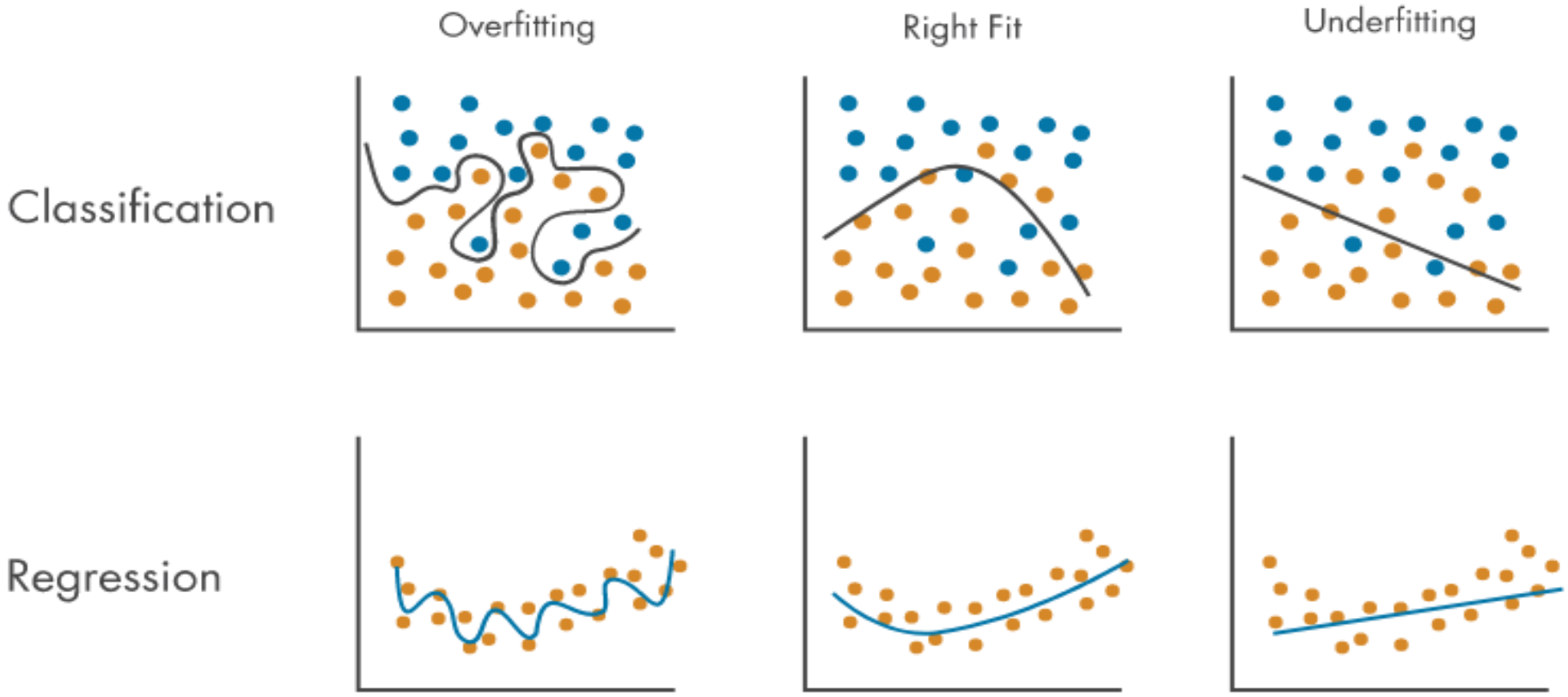
```
model = Sequential()
model.add(Dense(12, input_dim=[redacted], activation='relu', name='Dense_1'))
model.add(Dense(8, activation='relu', name='Dense_2'))
model.add(Dense([redacted], activation='sigmoid',name='Dense_3'))
model.summary()
```

```
# 모델을 컴파일합니다.
model.compile(loss='[redacted]', optimizer='adam',
metrics=['accuracy'])

# 모델을 실행합니다.
history=model.fit(X, y, epochs=100, batch_size=5)
```

080324/ch11.ipynb at master · gilbutITbook/080324 · GitHub

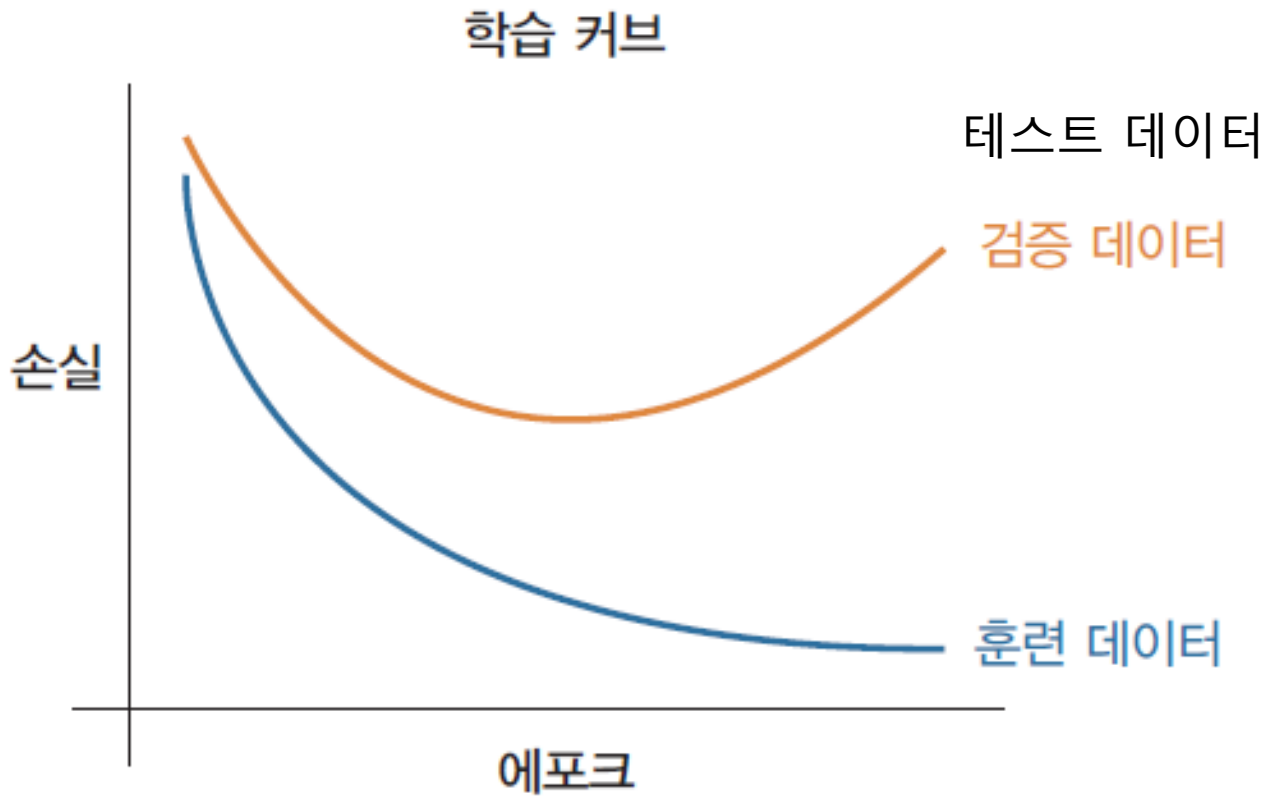
과적합 overfitting

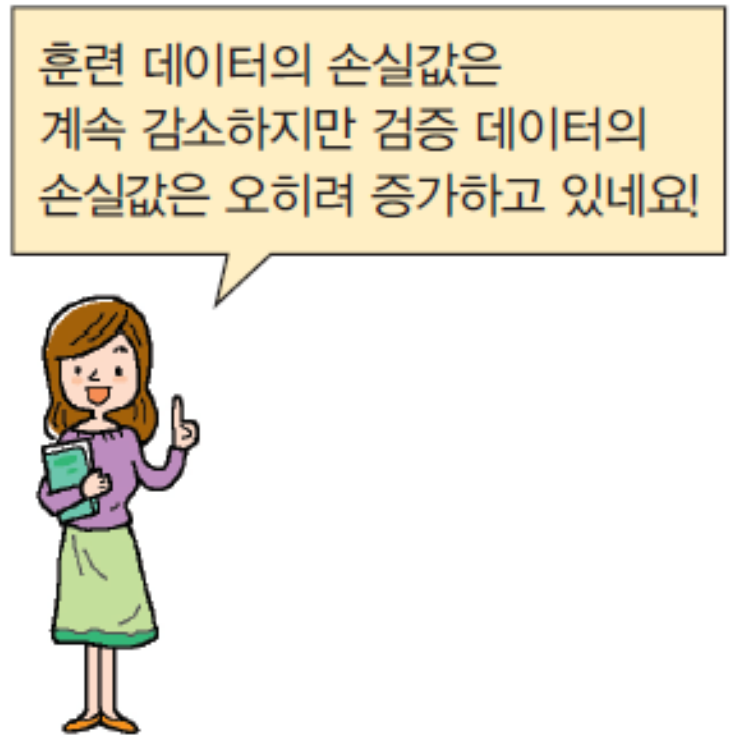
과잉 적합(over fitting)은 학습이 훈련 데이터에 특화돼 데이터의 전체적인 패턴을 학습하지 못하는것



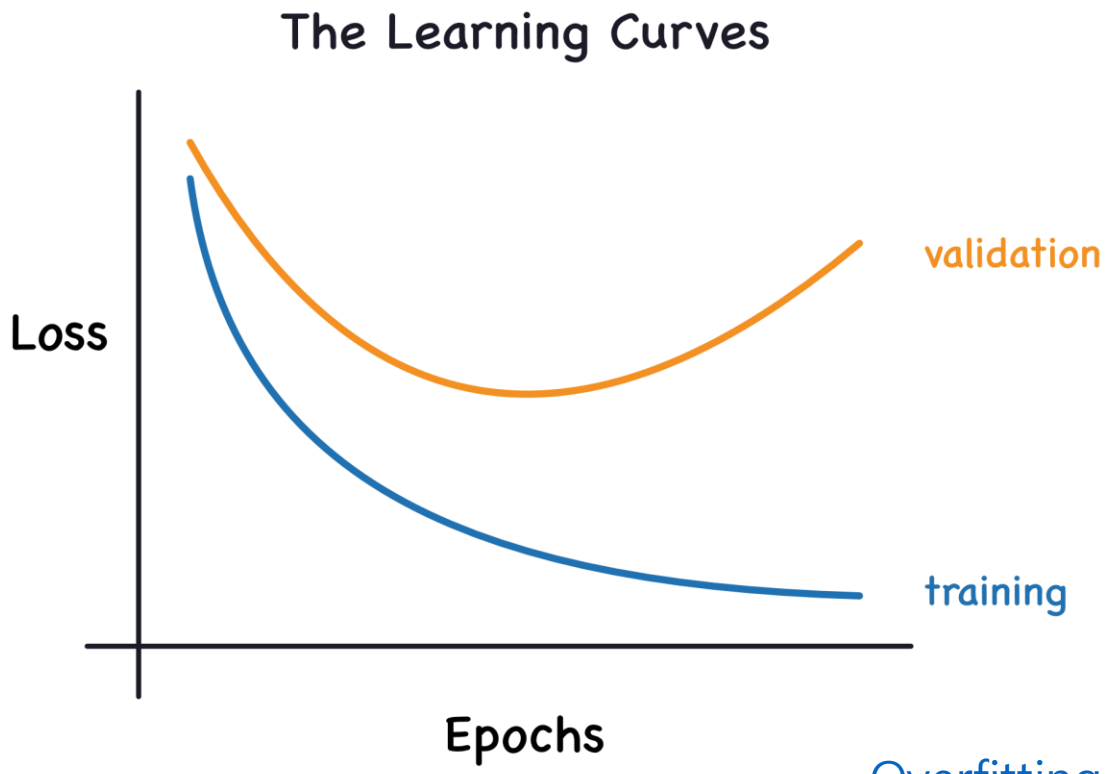
Overfitting - MATLAB & Simulink (mathworks.com)

과잉 적합과 과소 적합


학습 커브
테스트 데이터
검증 데이터
손실
훈련 데이터
에포크


훈련 데이터의 손실값은
계속 감소하지만 검증 데이터의
손실값은 오히려 증가하고 있네요!

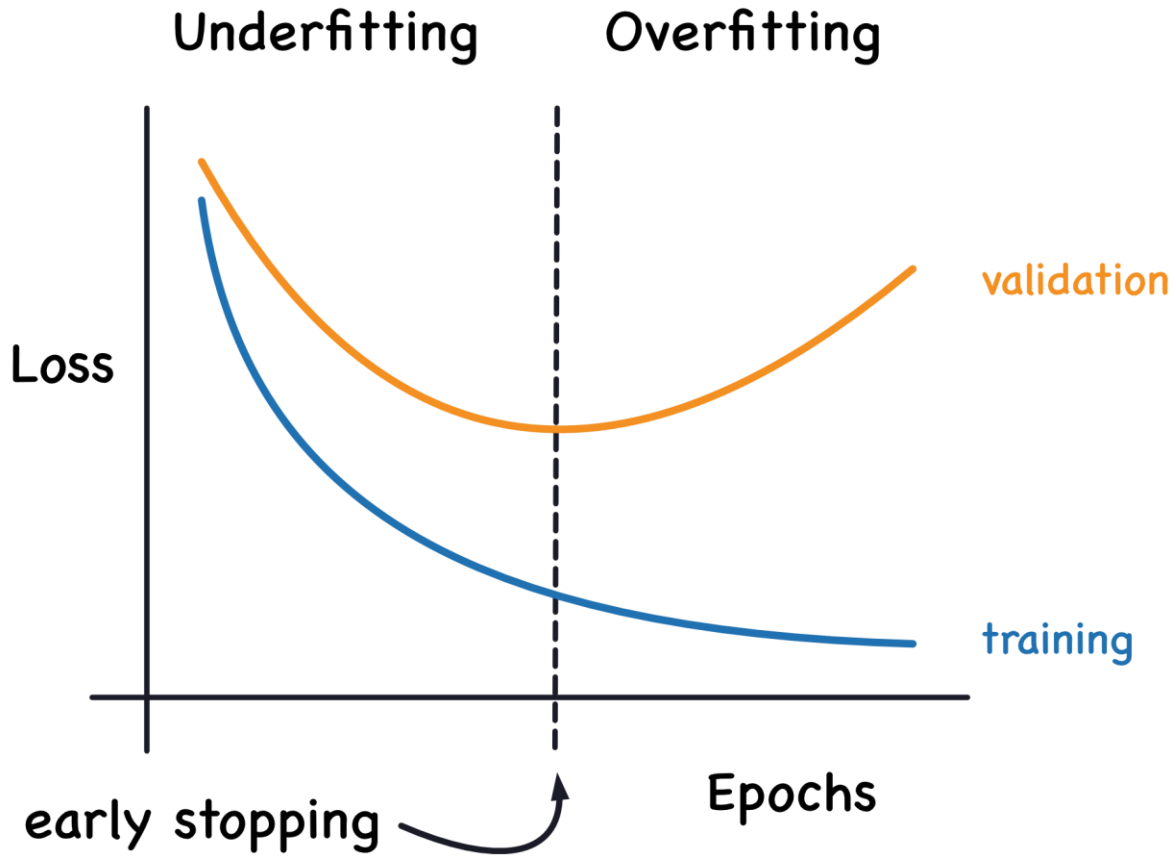
- **학습(training) 데이터 loss는 감소 될 때 검증(validation) 데이터의 loss가 줄어들지 않으면 과적합 (overfitting)**



Overfitting and Underfitting (kaggle.com)

조기종료

- 검증 손실(validation loss)이 더 이상 감소하지않을때 학습을 중단=< (Early Stopping)라고 한다.



모델 저장과 사용

- **학습 끝난 후 model 저장**
  - model.save('./mymodel.hdf5')
- **학습 끝난 후 model 저장**
  - From tensorflow.keras.models import load_model
  - model=load_model('./mymodel.hdf5')

케라스 신경망 실습 – 와인 종류 예측 하기


```
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from keras.callbacks import ModelCheckpoint,
EarlyStopping
import os
import pandas as pd

# 데이터를 입력합니다.
df = pd.read_csv('/content/wine.csv', header=None)

df
```

080324/ch11.ipynb at master · gilbutITbook/080324 · GitHub

케라스 신경망 실습 – 와인 종류 예측 하기

인하공전 컴퓨터 정보공학과

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	1
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5	1
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6	1
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
6492	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	6	0
6493	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	5	0
6494	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	6	0
6495	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	7	0
6496	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	6	0

6497 rows × 13 columns

케라스 신경망 실습 – 와인 종류 예측 하기

- **와인 데이터 셋**
  - **샘플 6497개**
  - **속성(특징) 12개**
  - **13번째 열 class (1:레드 0: 화이트)**
  - 샘플이 전체 6,497개 있음

케라스 신경망 실습 – 와인 종류 예측 하기


```
# 와인의 속성을 X로 와인의 분류를 y로 저장합니다.
X = df.iloc[:,0:12]
y = df.iloc[:,12]
X = X.astype(float)
y = y.astype(float)

#학습셋과 테스트셋으로 나눕니다.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True)
```

케라스 신경망 실습 – 와인 종류 예측 하기


```
# 모델 구조를 설정합니다.
model = Sequential()
model.add(Dense(30,  input_dim=12, activation='relu'))
model.add(Dense(12, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()

#모델을 컴파일합니다.
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

케라스 신경망 실습 – 와인 종류 예측 하기


```
### 학습의 자동 중단 및 최적화 모델 저장
# 학습이 언제 자동 중단 될지를 설정합니다.
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=20)

#최적화 모델이 저장될 폴더와 모델의 이름을 정합니다.
modelpath="./data/model/bestmodel.hdf5"

# 최적화 모델을 업데이트하고 저장합니다.
checkpointer = ModelCheckpoint(filepath=modelpath, monitor='val_loss', verbose=1,
save_best_only=True)

#모델을 실행합니다.
history=model.fit(X_train, y_train, epochs=2000, batch_size=500, validation_split=0.25, verbose=1,
callbacks=[early_stopping_callback,checkpointer])
```

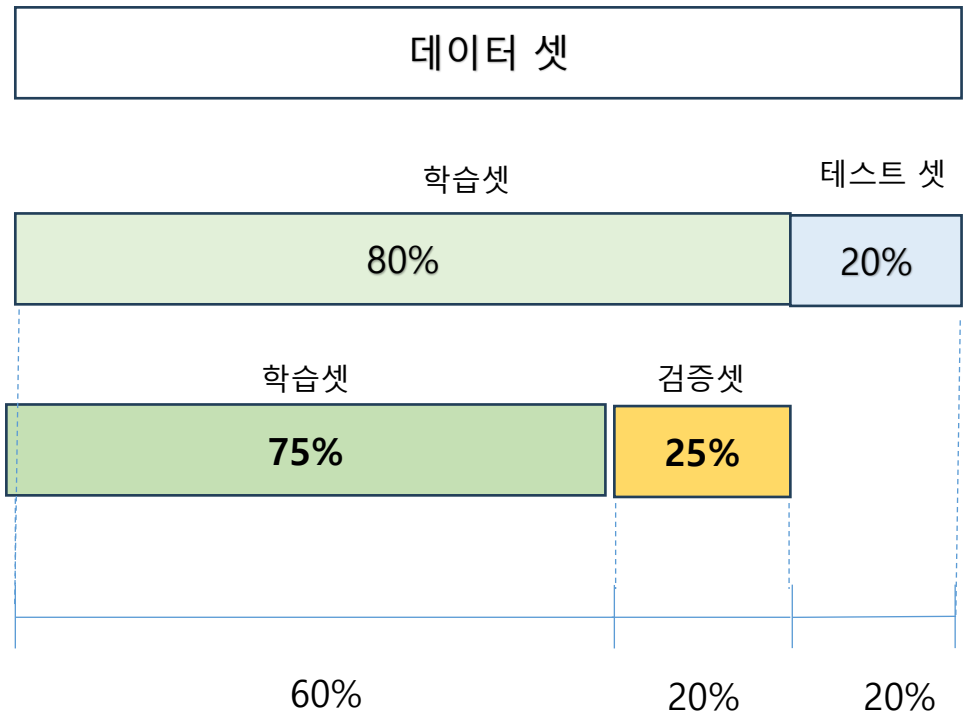
케라스 신경망 실습 – 와인 종류 예측 하기

인하공전 컴퓨터 정보과

```
# 테스트 결과를 출력합니다.
score=model.evaluate(X_test, y_test)
print('Test accuracy:', score[1])
```

케라스 신경망 실습 – 와인 종류 예측 하기

인하공전 컴퓨터 정보공학과



학습 셋 : training set
검증 셋: validation set
테스트 셋: test set

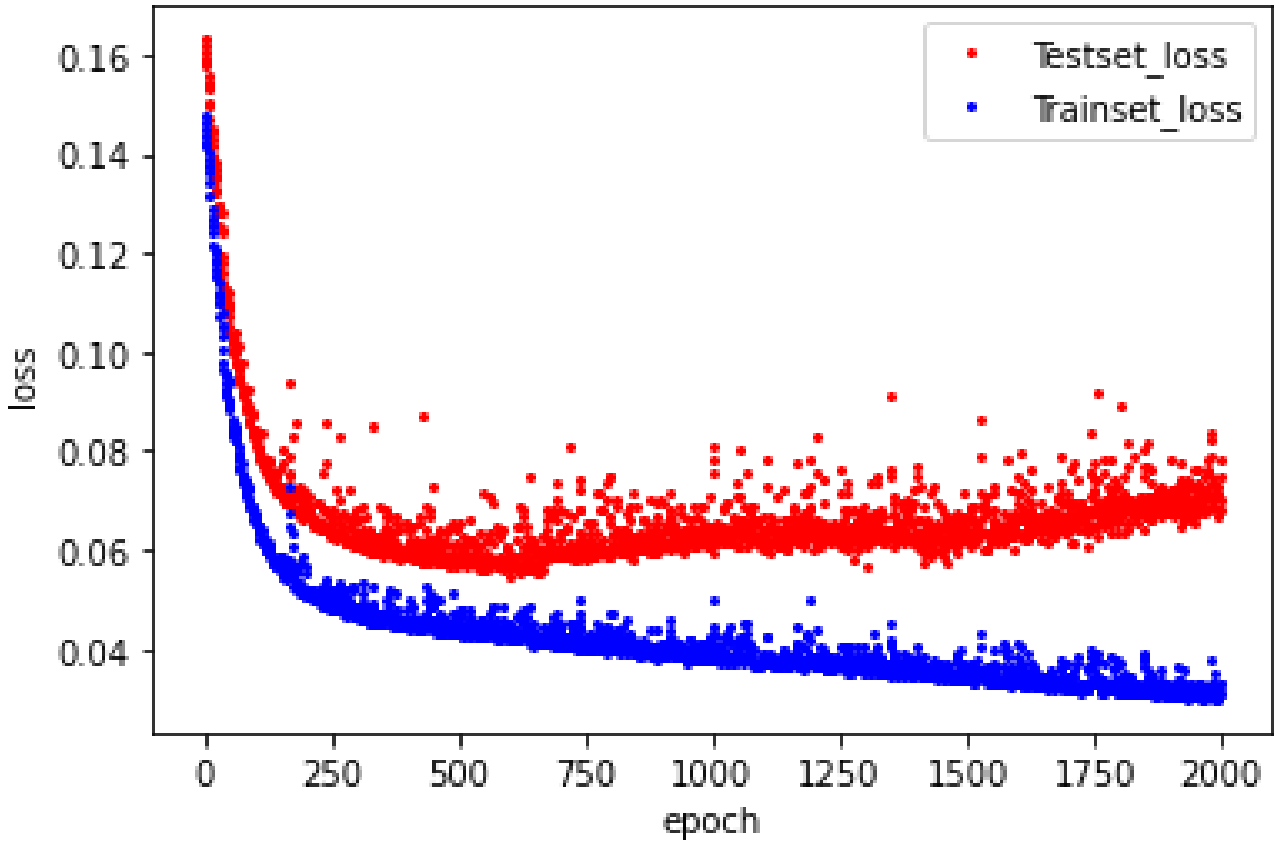
케라스 신경망 실습 – 와인 종류 예측 하기

- 검증 셋 (validation set) : 학습 과정에서 최적의 parameter를 찾기 위해서 사용
  과적합 방지에 도움이 됨.
  fit 함수 안에 validation_split 을 이용하여 만들어 짐

참고 모두의 딥러닝

케라스 신경망 실습 – 와인 종류 예측 하기

인하공전 컴퓨터 정보공학과



과적합 발생

검증셋 오차 (validation loss)가 커지기 전에 학습을 중단 시키는게 필요

케라스 신경망 실습 – 와인 종류 예측 하기

Early stopping 학습 중단 시점 선택

```
### 학습의 자동 중단 및 최적화 모델 저장
# 학습이 언제 자동 중단 될지를 설정합니다.
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=30)
```

monitor='val_loss', patience=20이라고 지정하면 검증셋의 오차가 30번 이상 낮아지지 않을 경우 학습을 종료하라는 의미

케라스 신경망 실습 – 와인 종류 예측 하기

모델 저장 시점 선택

```
#최적화 모델이 저장될 폴더와 모델의 이름을 정합니다.
modelpath="./data/model/bestmodel.hdf5"

# 최적화 모델을 업데이트하고 저장합니다.
checkpointer = ModelCheckpoint(filepath=modelpath, monitor='val_loss', verbose=1,
save_best_only=True)
```

=> 검증 셋 오차(validation loss) 기준으로 제일 성능이 좋을 때 모델 저장

```
#모델을 실행합니다.
history=model.fit(X_train, y_train, epochs=2000, batch_size=500, validation_split=0.25, verbose=1,
callbacks=[early_stopping_callback,checkpointer])
```

케라스 신경망 실습 – 와인 종류 예측 하기


```
1/8 [==>.....] - ETA: 0s - loss: 0.1462 - accuracy: 0.9540
Epoch 31: val_loss improved from 0.16765 to 0.16567, saving model to ./data/model/Ch14-4-bestmodel.hdf5
8/8 [==============================] - 0s 9ms/step - loss: 0.1712 - accuracy: 0.9430 - val_loss: 0.1657 - val_accuracy: 0.9392
Epoch 32/2000
1/8 [==>.....] - ETA: 0s - loss: 0.2067 - accuracy: 0.9400
Epoch 32: val_loss did not improve from 0.16567
8/8 [==============================] - 0s 7ms/step - loss: 0.1678 - accuracy: 0.9430 - val_loss: 0.1675 - val_accuracy: 0.9346
Epoch 33/2000
1/8 [==>.....] - ETA: 0s - loss: 0.1330 - accuracy: 0.9560
Epoch 33: val_loss did not improve from 0.16567
8/8 [==============================] - 0s 6ms/step - loss: 0.1661 - accuracy: 0.9438 - val_loss: 0.1691 - val_accuracy: 0.9431
Epoch 34/2000
1/8 [==>.....] - ETA: 0s - loss: 0.1694 - accuracy: 0.9400
Epoch 34: val_loss improved from 0.16567 to 0.15903, saving model to ./data/model/Ch14-4-bestmodel.hdf5
8/8 [==============================] - 0s 9ms/step - loss: 0.1694 - accuracy: 0.9418 - val_loss: 0.1590 - val_accuracy: 0.9362
Epoch 35/2000
1/8 [==>.....] - ETA: 0s - loss: 0.1949 - accuracy: 0.9380
Epoch 35: val_loss improved from 0.15903 to 0.15821, saving model to ./data/model/Ch14-4-bestmodel.hdf5
8/8 [==============================] - 0s 9ms/step - loss: 0.1628 - accuracy: 0.9464 - val_loss: 0.1582 - val_accuracy: 0.9369
Epoch 36/2000
1/8 [==>.....] - ETA: 0s - loss: 0.1485 - accuracy: 0.9360
```

참고 모두의 딥러닝

케라스 신경망 실습 – 와인 종류 예측 하기

과제3. wine_2024.ipynb, wine.csv

wine.ipynb 를 실행 하고 1) model이 마지막으로 저장 되는 부분과
2) 학습이 종료되는 부분의 출력 결과를 붙이시오
3) test결과의 accuracy를 적으시오.

참고 모두의 딥러닝

과잉 적합 방지 –영화 리뷰 분류 drop out

- 영화 리뷰를 분류하는 문제를 생각해보자. IMDB 사이트는 영화에 대한 리뷰가 올라가 있는 사이트이다.
- 영화 리뷰를 입력하면 리뷰가 긍정적인지 부정적인지를 파악하는 문제

과잉 적합 방지 –영화 리뷰 분류 drop out


```
import numpy as numpy
import tensorflow as tf
import matplotlib.pyplot as plt

# 데이터 다운로드
(train_data, train_labels), (test_data, test_labels) = \
    tf.keras.datasets.imdb.load_data( num_words=1000)

# 원-핫 인코딩으로 변환하는 함수
def one_hot_sequences(sequences, dimension=1000):
    results = numpy.zeros((len(sequences), dimension))
    for i, word_index in enumerate(sequences):
        results[i, word_index] = 1.
    return results

train_data = one_hot_sequences(train_data)
test_data = one_hot_sequences(test_data)
```

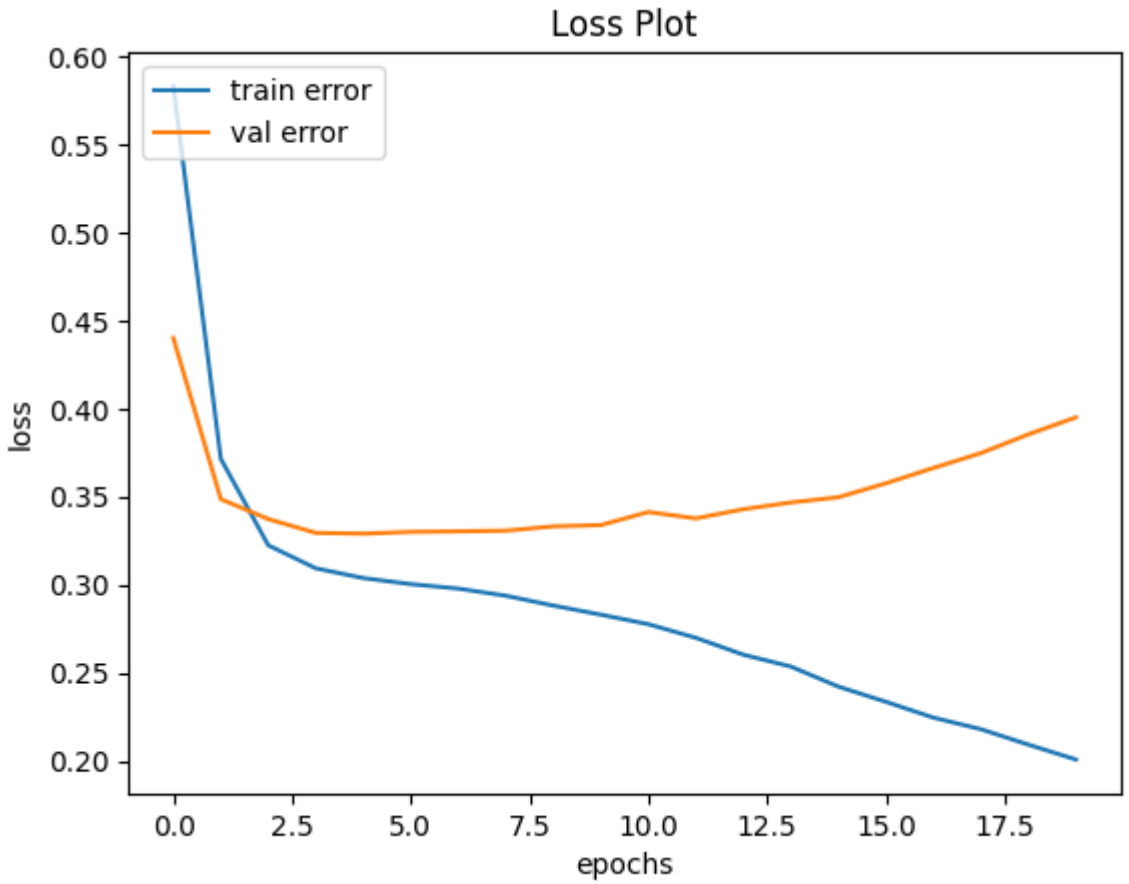
과잉 적합 방지 –영화 리뷰 분류 drop out


```
# 신경망 모델 구축
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(16, activation='relu', input_shape=(1000,)))
model.add(tf.keras.layers.Dense(16, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

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

# 신경망 훈련, 검증 데이터 전달
history = model.fit(train_data,
                    train_labels,
                    epochs=20,
                    batch_size=512,
                    validation_data=(test_data, test_labels),
                    verbose=2)
```

과잉 적합 방지 –영화 리뷰 분류 drop out


```
# 훈련 데이터의 손실값과 검증 데이터의 손실값을 그래프에 출력
history_dict = history.history
loss_values = history_dict['loss']                    # 훈련 데이터 손실값
val_loss_values = history_dict['val_loss']            # 검증 데이터 손실값
acc = history_dict['accuracy']                                # 정확도
epochs = range(1, len(acc) + 1)                       # 에포크 수

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss Plot')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend(['train error', 'val error'], loc='upper left')
plt.show()
```

실행 결과


Loss Plot
train error
val error
loss
epochs
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

드롭 아웃

- 드롭아웃은 학습 과정에서 node들을 랜덤하게 비 활성화 하는것이다.
- 특정 노드에 너무 의존 하지 않도록
- 과적합 방지

Drop out 과제

과제 4)

Imdb.ipynb code에서 아래처럼 mode 생성 부분에 drop out 을 추가 한 후 학습 시킨 후 생성된. loss plot (train err, val err)를 upload하시오

```
# 신경망 모델 구축
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(16, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

케라스 신경망 실습 – 주택 가격 예측

- **주택 가격 예측**
  - 회귀 문제
  - 80개의 속성 (환경, 차고 등등)을 이용하여 집값 예측 을 맞히거나 여러 개의 보기 중 하나를 예측하는 분류 문제

080324/ch11.ipynb at master · gilbutITbook/080324 · GitHub

케라스 신경망 실습 – 주택 가격 예측


```
df.head()
```

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	Bs
0	1	60	65.0	8450	7	5	2003	2003	196.0	
1	2	20	80.0	9600	6	8	1976	1976	0.0	
2	3	60	68.0	11250	7	5	2001	2002	162.0	
3	4	70	60.0	9550	7	5	1915	1970	0.0	
4	5	60	84.0	14260	8	5	2000	2000	350.0	

5 rows × 289 columns

케라스 신경망 실습 – 주택 가격 예측

- **데이터 파악하기**
  - 총 80개의 속성으로 이루어져 있고 마지막 열이 우리의 타깃인 집 값(SalePrice)
  - 모두 1,460개의 샘플이 들어 있음

케라스 신경망 실습 – 주택 가격 예측


```
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt
import seaborn as sns

import pandas as pd
import numpy as np

#데이터를 불러 옵니다.
df = pd.read_csv("/content/house_data.csv")

#카테고리형 변수를 0과 1로 이루어진 변수로 바꾸어 줍니다
df = pd.get_dummies(df)

#결측치를 전체 칼럼의 평균으로 대체하여 채워줍니다.
df = df.fillna(df.mean())
```

080324/ch11.ipynb at master · gilbutITbook/080324 · GitHub

케라스 신경망 실습 – 주택 가격 예측


```
#업데이트된 데이터프레임을 출력해 봅니다.
#df
#집 값을 제외한 나머지 열을 저장합니다.
cols_train=['OverallQual','GrLivArea','GarageCars','GarageArea','TotalBsmtSF']
X_train_pre = df[cols_train]

#집 값을 저장합니다.
y = df['SalePrice'].values
X_train_pre=X_train_pre.astype(float)
y=y.astype(float)
#전체의 80%를 학습셋으로, 20%를 테스트셋으로 지정합니다.
X_train, X_test, y_train, y_test = train_test_split(X_train_pre, y, test_size=0.2)
```

080324/ch11.ipynb at master · gilbutITbook/080324 · GitHub

케라스 신경망 실습 – 주택 가격 예측


```
#모델의 구조를 설정합니다.
model = Sequential()
model.add(Dense(10, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(30, activation='relu'))  # 30
model.add(Dense(40, activation='relu'))
model.add(Dense(1))
model.summary()

#모델을 실행합니다.
model.compile(optimizer ='adam', loss = 'mean_squared_error')

# 20회 이상 결과가 향상되지 않으면 자동으로 중단되게끔 합니다.
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=20)
```

080324/ch11.ipynb at master · gilbutITbook/080324 · GitHub

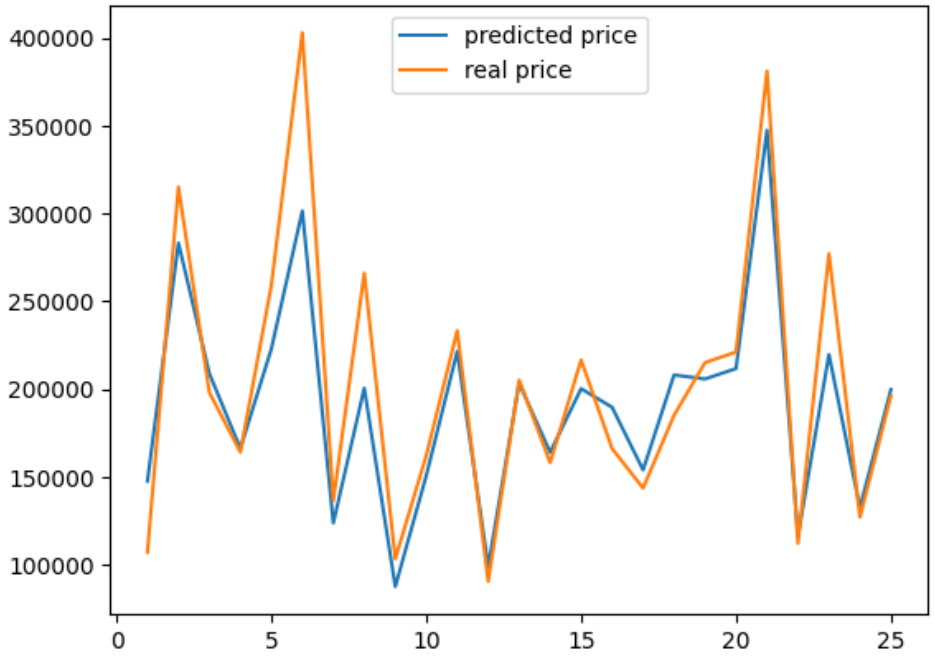
케라스 신경망 실습 – 주택 가격 예측


```
# 모델의 이름을 정합니다.
modelpath="/content/house.hdf5"

# 최적화 모델을 업데이트하고 저장합니다.
checkpointer = ModelCheckpoint(filepath=modelpath, monitor='val_loss', verbose=1,
save_best_only=True)

#실행 관련 설정을 하는 부분입니다. 전체의 20%를 검증셋으로 설정합니다.
history = model.fit(X_train, y_train, validation_split=0.25, epochs=2000, batch_size=32,
callbacks=[early_stopping_callback, checkpointer])
# 예측 값과 실제 값, 실행 번호가 들어갈 빈 리스트를 만듭니다.
real_prices =[]
pred_prices = []
X_num = []
```

https://github.com/gilbutITbook/080324/blob/master/ch15.ipynb

케라스 신경망 실습 – 주택 가격 예측


```
# 25개의 샘플을 뽑아 실제 값, 예측 값을 출력해 봅니다.
n_iter = 0
Y_prediction = model.predict(X_test).flatten()
for i in range(25):
    real = y_test[i]
    prediction = Y_prediction[i]
    print("실제가격: {:.2f}, 예상가격: {:.2f}".format(real, prediction))
    real_prices.append(real)
    pred_prices.append(prediction)
    n_iter = n_iter + 1
    X_num.append(n_iter)
#그래프를 통해 샘플로 뽑은 25개의 값을 비교해 봅니다.

plt.plot(X_num, pred_prices, label='predicted price')
plt.plot(X_num, real_prices, label='real price')
plt.legend()
plt.show()
```



https://github.com/gilbutITbook/080324/blob/master/ch15.ipynb

케라스 신경망 실습 – 주택 가격 예측


```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 10)                60

 dense_13 (Dense)            (None, 30)                330

 dense_14 (Dense)            (None, 40)                1240

 dense_15 (Dense)            (None, 1)                 41

=================================================================
Total params: 1671 (6.53 KB)
Trainable params: 1671 (6.53 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/2000
 1/28 [>.............................] - ETA: 21s - loss: 40451330048.0000
Epoch 1: val_loss improved from inf to 41709600768.00000, saving model to /content/house.hdf5
28/28 [==============================] - 1s 10ms/step - loss: 37837418496.0000 - val_loss: 41709600768.0000
Epoch 2/2000
 1/28 [>.............................] - ETA: 0s - loss: 41037373440.0000
Epoch 2: val_loss improved from 41709600768.00000 to 40783089664.00000, saving model to /content/house.hdf5
28/28 [==============================] - 0s 4ms/step - loss: 37259124736.0000 - val_loss: 40783089664.0000
Epoch 3/2000
 1/28 [>.............................] - ETA: 0s - loss: 33677219840.0000/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are
saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
```

https://github.com/gilbutITbook/080324/blob/master/ch15.ipynb

케라스 신경망 실습 – 주택 가격 예측


```
print('테스트 loss', model.evaluate(X_test,y_test)/10000000)
```

https://github.com/gilbutITbook/080324/blob/master/ch15.ipynb

케라스 신경망 실습 – 주택 가격 예측

- **주택 가격 예측 모델**
  - 실행에서 달라진 점은 손실 함수
  - 선형 회귀이므로 평균 제곱 오차(mean_squared_error)를 적음

```
model.compile(optimizer='adam', loss='mean_squared_error')
```

참고 모두의 딥러닝

케라스 신경망 실습 – 주택 가격 예측

인하공전 컴퓨터 정보공학과

과제5 (house_2024.ipynb, house_data.csv)

1) 아래와 같이 모델과 학습 방법을 변경 한 후 모델을 house1.hdf5로 저장 하시오.

입력 층 : 변동 없음
은닉층 1 : 20 (relu)
은닉측 2 : 20 (relu)
은닉측 3 : 40 (relu)
출력 층 변동 없음.

17회 이상 결과가 향상 되지 않으면 자동으로 학습이 중단되게
Batch size 16

2) 아래와 같이 모델과 학습 방법을 변경 한 후 모델을 house2.hdf5로 저장 하시오.

입력 층 : 변동 없음
은닉층 1 : 10 (relu)
은닉측 2 : 20 (relu)
은닉측 3 : 20 (relu)
출력 층 변동 없음.

15회 이상 결과가 향상 되지 않으면 자동으로 학습이 중단되게
Batch size 20

code (.ppt에 직접 붙여도 되고 .py 파일로 제출 가능) 와 model 제출 (house1.hdf5, hoise2.hdf5) upload

수고하셨습니다

jhmin@inhatc.ac.kr