

딥러닝 모델 구현해 보기

학습 내용

- 자전거 공유 업체 시간대별 데이터를 이용한 딥러닝 모델 만들기
- URL : <https://www.kaggle.com/competitions/bike-sharing-demand>
(<https://www.kaggle.com/competitions/bike-sharing-demand>)

목차

- [01. 라이브러리 импорт](#)
- [02. 데이터 셋 로드 및 데이터 탐색](#)
- [03. 입력\(input\)과 출력\(output\) 지정](#)
- [04. 딥러닝 모델 만들기 및 학습](#)

01. 라이브러리 импорт

[목차로 이동하기](#)

In [1]:

```
import tensorflow as tf
import keras

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

In [2]:

```
print("tf version : {}".format(tf.__version__))
print("keras version : {}".format(keras.__version__))
print("numpy version : {}".format(np.__version__))
print("matplotlib version : {}".format(matplotlib.__version__))
print("pandas version : {}".format(pd.__version__))
```

```
tf version : 2.9.1
keras version : 2.9.0
numpy version : 1.23.1
matplotlib version : 3.5.2
pandas version : 1.4.3
```

02. 데이터 셋 로드 및 데이터 탐색

[목차로 이동하기](#)

In [3]:

```
## train 데이터 셋 , test 데이터 셋
## train 은 학습을 위한 입력 데이터 셋
## test 은 예측을 위한 새로운 데이터 셋(평가)
## parse_dates : datetime 컬럼을 시간형으로 불러올 수 있음
train = pd.read_csv("./bike/bike_mod_tr.csv", parse_dates=['datetime'])
test = pd.read_csv("./bike/bike_mod_test.csv", parse_dates=['datetime'])
```

데이터 탐색

In [4]:

```
train.columns
```

Out[4]:

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count',
       'year', 'month', 'day', 'hour', 'minute', 'second', 'dayofweek'],
      dtype='object')
```

In [5]:

```
test.columns
```

Out[5]:

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'year', 'month', 'day', 'dayofweek',
       'hour', 'minute', 'second'],
      dtype='object')
```

In [6]:

```
print(train.info())
print()
print(test.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 19 columns):
#   Column          Non-Null Count  Dtype
---  -
0   datetime        10886 non-null  datetime64[ns]
1   season          10886 non-null  int64
2   holiday         10886 non-null  int64
3   workingday      10886 non-null  int64
4   weather         10886 non-null  int64
5   temp            10886 non-null  float64
6   atemp           10886 non-null  float64
7   humidity        10886 non-null  int64
8   windspeed       10886 non-null  float64
9   casual          10886 non-null  int64
10  registered      10886 non-null  int64
11  count           10886 non-null  int64
12  year            10886 non-null  int64
13  month           10886 non-null  int64
14  day             10886 non-null  int64
15  hour            10886 non-null  int64
16  minute          10886 non-null  int64
17  second          10886 non-null  int64
18  dayofweek       10886 non-null  int64
dtypes: datetime64[ns](1), float64(3), int64(15)
memory usage: 1.6 MB
None
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6493 entries, 0 to 6492
Data columns (total 16 columns):
#   Column          Non-Null Count  Dtype
---  -
0   datetime        6493 non-null  datetime64[ns]
1   season          6493 non-null  int64
2   holiday         6493 non-null  int64
3   workingday      6493 non-null  int64
4   weather         6493 non-null  int64
5   temp            6493 non-null  float64
6   atemp           6493 non-null  float64
7   humidity        6493 non-null  int64
8   windspeed       6493 non-null  float64
9   year            6493 non-null  int64
10  month           6493 non-null  int64
11  day             6493 non-null  int64
12  dayofweek       6493 non-null  int64
13  hour            6493 non-null  int64
14  minute          6493 non-null  int64
15  second          6493 non-null  int64
dtypes: datetime64[ns](1), float64(3), int64(12)
memory usage: 811.8 KB
None
```

03. 입력(input)과 출력(output) 지정

[목차로 이동하기](#)

X : hour, temp (시간, 온도)

y : count - 자전거 시간대별 렌탈 대수

In [8]:

```
input_col = [ 'weather', 'temp' ]  
labeled_col = [ 'count' ]
```

In [9]:

```
X = train[ input_col ]  
y = train[ labeled_col ]  
X_val = test[ input_col ]
```

데이터 나누기

In [11]:

```
from sklearn.model_selection import train_test_split
```

In [12]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,  
                                                    random_state=0)
```

In [13]:

```
print(X_train.shape)  
print(X_test.shape)
```

```
(8164, 2)  
(2722, 2)
```

In [14]:

```
### 난수 발생 패턴 결정 0  
seed = 0  
np.random.seed(seed)
```

04. 딥러닝 모델 만들기 및 학습

[목차로 이동하기](#)

- 케라스 라이브러리 중에서 Sequential 함수는 딥러닝의 구조를 한층 한층 쉽게 쌓아올릴 수 있다.
- Sequential() 함수 선언 후, 신경망의 층을 쌓기 위해 model.add() 함수를 사용한다
- input_dim 입력층 노드의 수
- activation - 활성화 함수 선언 (relu, sigmoid)
- Dense() 함수를 이용하여 각 층에 세부 내용을 설정해 준다.

In [15]:

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

In [16]:

```
model = Sequential()
model.add(Dense(30, input_dim=2, activation='relu'))
model.add(Dense(15, activation='relu'))
model.add(Dense(15, activation='relu'))
model.add(Dense(1))
```

In [17]:

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 30)	90
dense_1 (Dense)	(None, 15)	465
dense_2 (Dense)	(None, 15)	240
dense_3 (Dense)	(None, 1)	16
Total params: 811		
Trainable params: 811		
Non-trainable params: 0		

미니배치의 이해

- 이미지를 하나씩 학습시키는 것보다 여러 개를 한꺼번에 학습시키는 쪽이 효과가 좋다.
- 많은 메모리와 높은 컴퓨터 성능이 필요하므로 일반적으로 데이터를 적당한 크기로 잘라서 학습시킨다.
 - 미니배치라고 한다.

딥러닝 compile과 모델 학습(최적화)

In [18]:

```
model.compile(loss = 'mean_squared_error', optimizer='rmsprop')
model.fit(X_train, y_train, epochs=20, batch_size=10)
```

```
Epoch 1/20
817/817 [=====] - 3s 2ms/step - loss: 38706.3516
Epoch 2/20
817/817 [=====] - 1s 2ms/step - loss: 27885.8184
Epoch 3/20
817/817 [=====] - 1s 2ms/step - loss: 27864.7461
Epoch 4/20
817/817 [=====] - 1s 2ms/step - loss: 27796.6406
Epoch 5/20
817/817 [=====] - 1s 2ms/step - loss: 27740.3535
Epoch 6/20
817/817 [=====] - 1s 2ms/step - loss: 27718.2988
Epoch 7/20
817/817 [=====] - 1s 2ms/step - loss: 27683.9512
Epoch 8/20
817/817 [=====] - 1s 2ms/step - loss: 27673.3262
Epoch 9/20
817/817 [=====] - 1s 2ms/step - loss: 27663.5430
Epoch 10/20
817/817 [=====] - 1s 2ms/step - loss: 27609.6699
Epoch 11/20
817/817 [=====] - 1s 2ms/step - loss: 27616.7930
Epoch 12/20
817/817 [=====] - 1s 2ms/step - loss: 27606.2773
Epoch 13/20
817/817 [=====] - 1s 2ms/step - loss: 27597.0020
Epoch 14/20
817/817 [=====] - 1s 2ms/step - loss: 27539.1953
Epoch 15/20
817/817 [=====] - 1s 2ms/step - loss: 27577.2891
Epoch 16/20
817/817 [=====] - 1s 2ms/step - loss: 27565.6875
Epoch 17/20
817/817 [=====] - 1s 2ms/step - loss: 27551.3359
Epoch 18/20
817/817 [=====] - 1s 2ms/step - loss: 27562.4727
Epoch 19/20
817/817 [=====] - 1s 2ms/step - loss: 27551.9199
Epoch 20/20
817/817 [=====] - 1s 2ms/step - loss: 27542.7266
```

Out [18]:

```
<keras.callbacks.History at 0x12c0f038cd0>
```

모델 평가

In [19]:

```
model.evaluate(X_test, y_test)
```

86/86 [=====] - 0s 1ms/step - loss: 27238.3242

Out[19]:

27238.32421875

예측 수행

In [20]:

```
pred = model.predict(X_val)
```

203/203 [=====] - 0s 2ms/step

In [21]:

```
sub = pd.read_csv("./bike/sampleSubmission.csv")  
sub['count'] = pred
```

```
sub.loc[sub['count'] < 0, 'count'] = 0
```

In [22]:

```
sub.to_csv("nn_sub_2207.csv", index=False)
```

추가 실습

변수를 추가를 통해 성능을 향상시켜보자(5-10분) - epoch수도 증가

- (예) ['hour', 'temp', 'dayofweek', 'workingday', 'season', 'weather']
- (예) 100epoch, ['hour', 'temp', 'dayofweek', 'workingday', 'season', 'weather'] => 0.82071
- (예) 300epoch, ['hour', 'temp', 'dayofweek', 'workingday', 'season', 'weather'] => 0.70710
- input_col = ['hour', 'temp', 'weather', 'season', 'holiday', 'temp', 'workingday', 'windspeed'] 300epoch