딥러닝 모델 구현해 보기

- 첫번째 데이터 셋 : 자전거 공유 업체 시간대별 데이터
- 두번째 데이터 셋 : 타이타닉 데이터 셋

In [14]:

```
import tensorflow as tf
import keras
```

```
Using TensorFlow backend.
```

In [15]:

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

In [17]:

```
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 : 1.15.0
keras version : 2.3.1
numpy version : 1.16.4
matplotlib version : 3.1.0
pandas version : 0.24.2
```

데이터 셋 불러오기

In [18]:

```
## 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 [19]:

```
train.columns
```

Out[19]:

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

In [20]:

```
test.columns
```

Out[20]:

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

In [21]:

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 19 columns):
datetime      10886 non-null datetime64[ns]
season        10886 non-null int64
holiday       10886 non-null int64
workingday    10886 non-null int64
weather       10886 non-null int64
temp          10886 non-null float64
atemp         10886 non-null float64
humidity      10886 non-null int64
windspeed     10886 non-null float64
casual        10886 non-null int64
registered    10886 non-null int64
count         10886 non-null int64
year          10886 non-null int64
month         10886 non-null int64
day           10886 non-null int64
hour          10886 non-null int64
minute        10886 non-null int64
second        10886 non-null int64
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):
datetime      6493 non-null datetime64[ns]
season        6493 non-null int64
holiday       6493 non-null int64
workingday    6493 non-null int64
weather       6493 non-null int64
temp          6493 non-null float64
atemp         6493 non-null float64
humidity      6493 non-null int64
windspeed     6493 non-null float64
year          6493 non-null int64
month         6493 non-null int64
day           6493 non-null int64
dayofweek     6493 non-null int64
hour          6493 non-null int64
minute        6493 non-null int64
second        6493 non-null int64
dtypes: datetime64[ns](1), float64(3), int64(12)
memory usage: 811.7 KB
None
```

모델을 위한 데이터 선택

- X : hour, temp : 시간, 온도

- y : count - 자전거 시간대별 렌탈 대수

In [22]:

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

In [51]:

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

In [52]:

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

In [53]:

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

In [54]:

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

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

In [55]:

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

딥러닝 구조 결정

- 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 [56]:

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

In [57]:

```
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))
```

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.
  - **미니배치**라고 한다.

딥러닝 실행

In [58]:

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

```
Epoch 1/20
8164/8164 [==============================] - 1s 172us/step - loss: 30114.0064
Epoch 2/20
8164/8164 [==============================] - 1s 161us/step - loss: 23271.4145
Epoch 3/20
8164/8164 [==============================] - 1s 161us/step - loss: 21343.8217
Epoch 4/20
8164/8164 [==============================] - 1s 141us/step - loss: 19959.9441
Epoch 5/20
8164/8164 [==============================] - 1s 143us/step - loss: 19547.3706
Epoch 6/20
8164/8164 [==============================] - 1s 140us/step - loss: 19369.3852
Epoch 7/20
8164/8164 [==============================] - 1s 144us/step - loss: 19294.2127
Epoch 8/20
8164/8164 [==============================] - 1s 164us/step - loss: 19262.8771
Epoch 9/20
8164/8164 [==============================] - 1s 167us/step - loss: 19183.2823
Epoch 10/20
8164/8164 [==============================] - 1s 180us/step - loss: 19076.4567
Epoch 11/20
8164/8164 [==============================] - 2s 199us/step - loss: 18961.7522
Epoch 12/20
8164/8164 [==============================] - 2s 187us/step - loss: 18902.4526
Epoch 13/20
8164/8164 [==============================] - 1s 182us/step - loss: 18829.4558
Epoch 14/20
8164/8164 [==============================] - 2s 184us/step - loss: 18725.4699
Epoch 15/20
8164/8164 [==============================] - 1s 181us/step - loss: 18617.5444
Epoch 16/20
8164/8164 [==============================] - 1s 178us/step - loss: 18544.2181
Epoch 17/20
8164/8164 [==============================] - 2s 188us/step - loss: 18397.7230
Epoch 18/20
8164/8164 [==============================] - 2s 206us/step - loss: 18270.4018
Epoch 19/20
8164/8164 [==============================] - 2s 211us/step - loss: 18063.0341
Epoch 20/20
8164/8164 [==============================] - 2s 222us/step - loss: 17927.7797
```

Out[58]:

```
<keras.callbacks.callbacks.History at 0x25c523049b0>
```

In [59]:

```
### 평가 확인
model.evaluate(X_test, y_test)
```

```
2722/2722 [==============================] - 0s 67us/step
```

Out[59]:

```
18073.671321064474
```

In [60]:

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

In [61]:

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

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

In [62]:

```
sub.to_csv("nn_sub_0528.csv", index=False)
```

점수 : 1.04514

In [ ]:

```

```