베스트 모델 만들기

모두의 딥러닝 6회차 발표

1팀

데이터의 확인과 실행

메소드 함수 sample(frac=1): 무작위 열람
메소드 함수 info(): 전체 정보 출력

```
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import ModelCheckpoint, EarlyStopping

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

```
# seed 값 설정
seed = 0
numpy.random.seed(seed)
tf.random.set_seed(3)

# 데이터 입력
df_pre = pd.read_csv('../dataset/wine.csv', header=None)
df = df_pre.sample(frac=1)
dataset = df.values
X = dataset[:,0:12]
Y = dataset[:,12]
```

```
# 모델 설정
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'))

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

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

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

```
Epoch 200/200
6497/6497 [==============================] - 0s 6us/step - loss: 0.0493
- accuracy: 0.9865
6497/6497 [==============================] - 0s 18us/step

 Accuracy: 0.9858
```

정확도가 98.58%인 딥러닝 프레임워크를 완성하였습니다.

모델 업데이트

13장 -> save(), load_model() 함수 -> 최종 모델 저장 및 재사용

14장 -> **epoch마다** 모델의 정확도와 함께 모델 저장

1. 모델이 저장될 폴더 지정 + 파일 이름 설정

```
MODEL_DIR = './model/' # 여기부터
if not os.path.exists(MODEL_DIR):
  os.mkdir(MODEL_DIR)
```

(1)

```
modelpath = "./model/{epoch:02d}-{val_loss:.4f}.hdf5"
```

(2)

에포크 횟수 테스트셋 오차값 확장자

2. 모니터할 값 지정

```
checkpointer = ModelCheckpoint(filepath=modelpath, monitor='val_loss', verbose=1, save_best_only=True)
```

ModelCheckpoint()
모델 저장하기 위한 keras의 콜백 함수

filepath
모델이 저장될 곳

model
모니터할 값 지정

verbose=1 : 함수 진행 사항 출력
verbose=0 : 출력X

* keras 내부
loss : 학습셋 오차
val_loss : 테스트 오차
acc : 학습 정확도
val_acc : 테스트 정확도

save_best_only=True
현 모델이 앞선 모델보다 나아졌을 때만 저장

3. epoch마다 지정된 곳에 모델 저장

```
model.fit(X, Y, validation_split=0.2, epochs=200, batch_size=200, verbose=0, callbacks=[checkpointer])
```

model.fit()
모델 실행

validation_split=0.2
테스트 데이터 20%
학습 데이터 80%

모델을 학습할 때마다
checkpointer 값을 받아 저장

최종 코드

```
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import ModelCheckpoint # 여기

import pandas as pd
import numpy as np
import os # 여기
import tensorflow as tf
import matplotlib.pyplot as plt

seed=0
numpy.random.seed(seed)
tf.random.set_seed(3)

df_pre = pd.read_csv(' ', header=None)
df = df_pre.sample(frac=1)
dataset = df.values
X = dataset[:, 0:12]
Y = dataset[:, 12]

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

MODEL_DIR = './model/' # 여기부터
if not os.path.exists(MODEL_DIR):
  os.mkdir(MODEL_DIR)

modelpath = "./model/{epoch:02d}-{val_loss:.4f}.hdf5"

checkpointer = ModelCheckpoint(filepath=modelpath, monitor='val_loss', verbose=1, save_best_only=True)

model.fit(X, Y, validation_split=0.2, epochs=200, batch_size=200, verbose=0, callbacks=[checkpointer]) # 여기까지
```

그래프로 확인하기

preview

에포크를 진행하면서
테스트 오차를 실행한 결과 값이
향상되었을 때만 저장

1. **model fit()** 이용하여 학습시키기
2. **matplotlib** 으로 학습 진행에 따른 오차값과 정확도 그래프 표현하기

1. model fit() 이용하여 학습시키기

```
df = df_pre.sample(frac=0.15)

history = model.fit(X, Y, validation_split=0.33, batch_size=500)
```

- df_pre에서 sample 의 15% 무작위 추출 (frac : 추출할 비율)
- validation_split : 테스트에 사용할 비율 지정
- batch_size : 한 번에 처리되는 데이터 수

2. matplotlib 으로 오차값과 정확도 그래프 표현

```
import matplotlib.pyplot as plt

y_vloss = history.history['val_loss']
y_acc = history.history['acc']

x_len = numpy.arange(len(y_acc))
plt.plot(x_len, y_vloss, c="red", markersize=3)
plt.plot(x_len, y_acc, "o", c="blue", markersize=3)
```

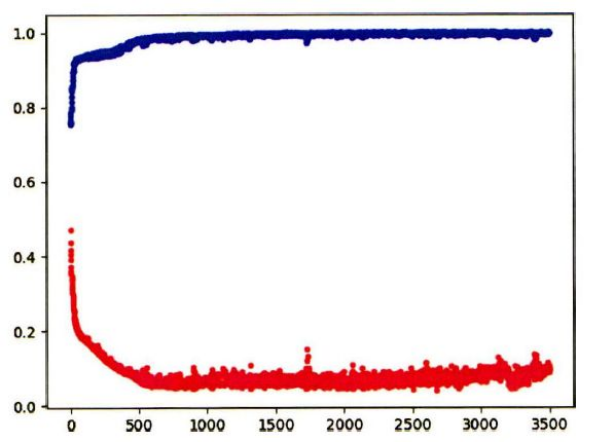
케라스 내부에서

테스트 오차 : val_loss

학습 정확도 : acc

- 테스트 오차를 y_vloss에 저장, 빨간색 선
- 정확도를 y_acc에 저장, 파란색 선으로 표현

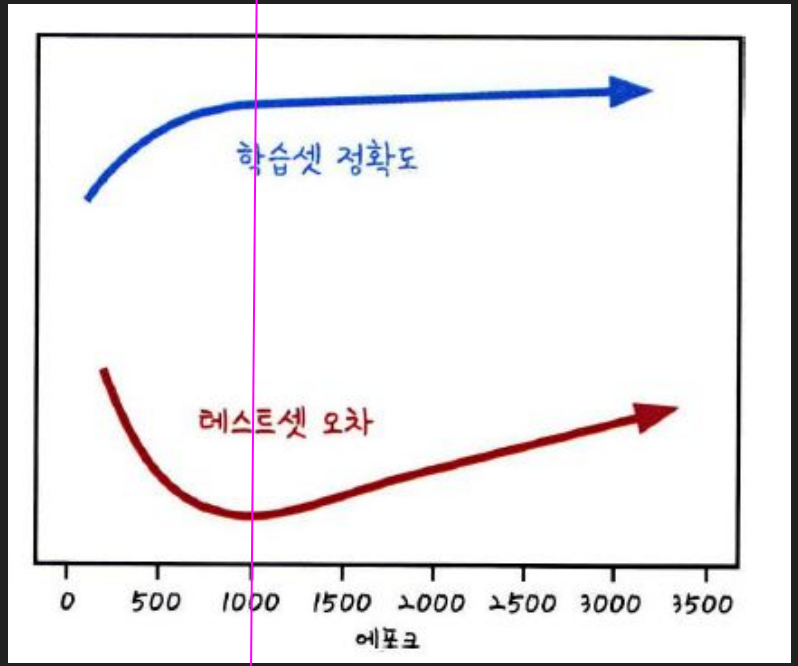
3. 그래프 해석




```
Epoch 3497/3500
653/653 [==============================] - 0s - loss: 0.0119 - acc:
0.9954 - val_loss: 0.0976 - val_acc: 0.9783
Epoch 3498/3500
653/653 [==============================] - 0s - loss: 0.0102 - acc:
0.9985 - val_loss: 0.0885 - val_acc: 0.9783
Epoch 3499/3500
653/653 [==============================] - 0s - loss: 0.0123 - acc:
0.9954 - val_loss: 0.0933 - val_acc: 0.9783
Epoch 3500/3500
653/653 [==============================] - 0s - loss: 0.0097 - acc:
0.9969 - val_loss: 0.0958 - val_acc: 0.9783
```

- Epoch 3400대 val_acc 의 값
- 0.0976, 0.0885, 0.0933, 0.0958 증가

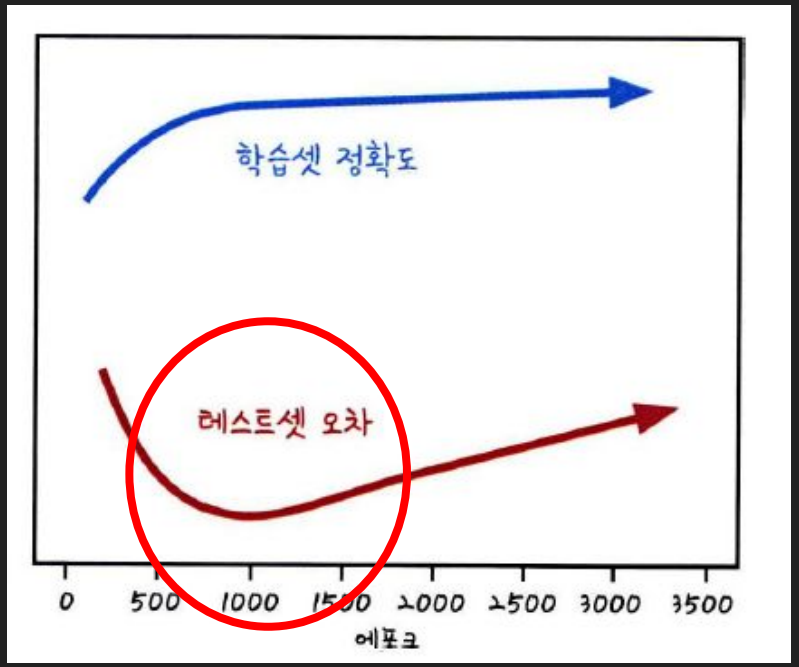
3. 그래프 해석



- 학습셋의 정확도 : 시간과 비례
- 테스트셋 오차 : 과적합 발생

학습의 자동 중단(Early Stopping)

학습의 자동 중단 (Early Stopping)



학습이 계속 된다면
overfitting 발생

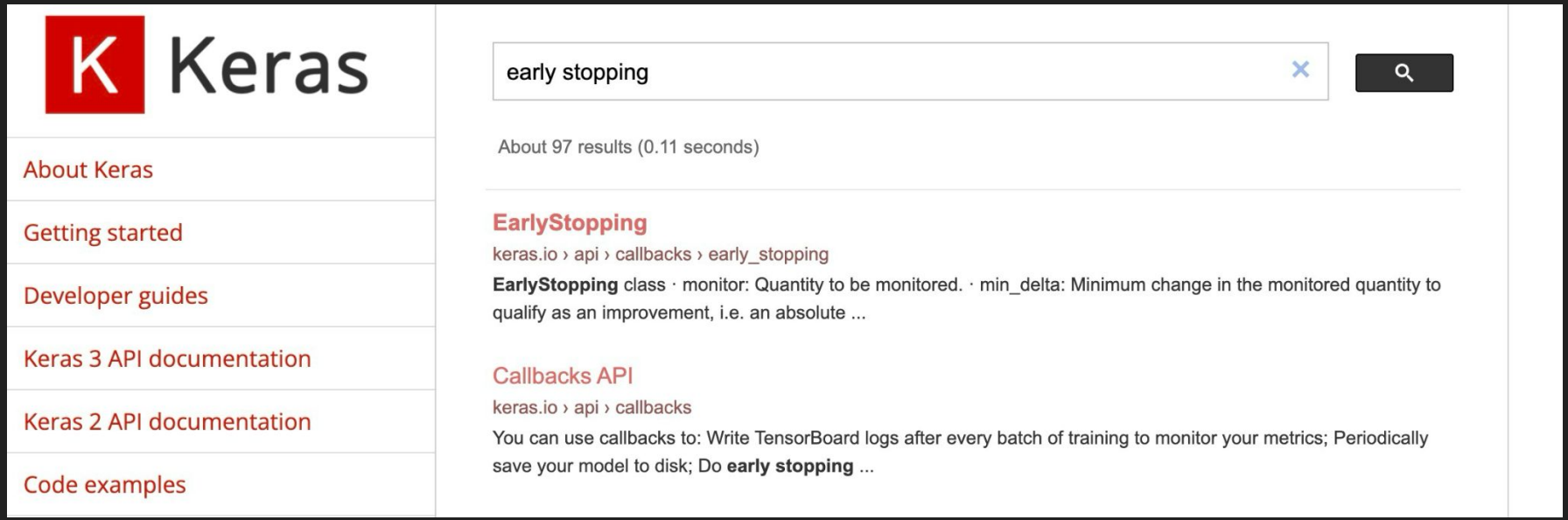
여기서 일찍 (early) 학습을 멈춰보자 (stop)

Early Stopping?

모델 훈련 중 특정 지표가 지정한 횟수 동안 감소하지 않을 때,
훈련을 조기에 중단 시키는 기법

Why?

overfitting 방지
효율적 컴퓨팅 자원 활용



https://keras.io/api/callbacks/early_stopping/

EarlyStopping

EarlyStopping class [source]

```
keras.callbacks.EarlyStopping(
    monitor="val_loss",
    min_delta=0,
    patience=0,
    verbose=0,
    mode="auto",
    baseline=None,
    restore_best_weights=False,
    start_from_epoch=0,
)
```

Stop training when a monitored metric has stopped improving.

Assuming the goal of a training is to minimize the loss. With this, the metric to be monitored would be `'loss'`, and mode would be `'min'`. A `model.fit()` training loop will check at end of every epoch whether the loss is no longer decreasing, considering the `min_delta` and `patience` if applicable. Once it's found no longer decreasing, `model.stop_training` is marked True and the training terminates.

The quantity to be monitored needs to be available in `logs` dict. To make it so, pass the loss or metrics at `model.compile()`.

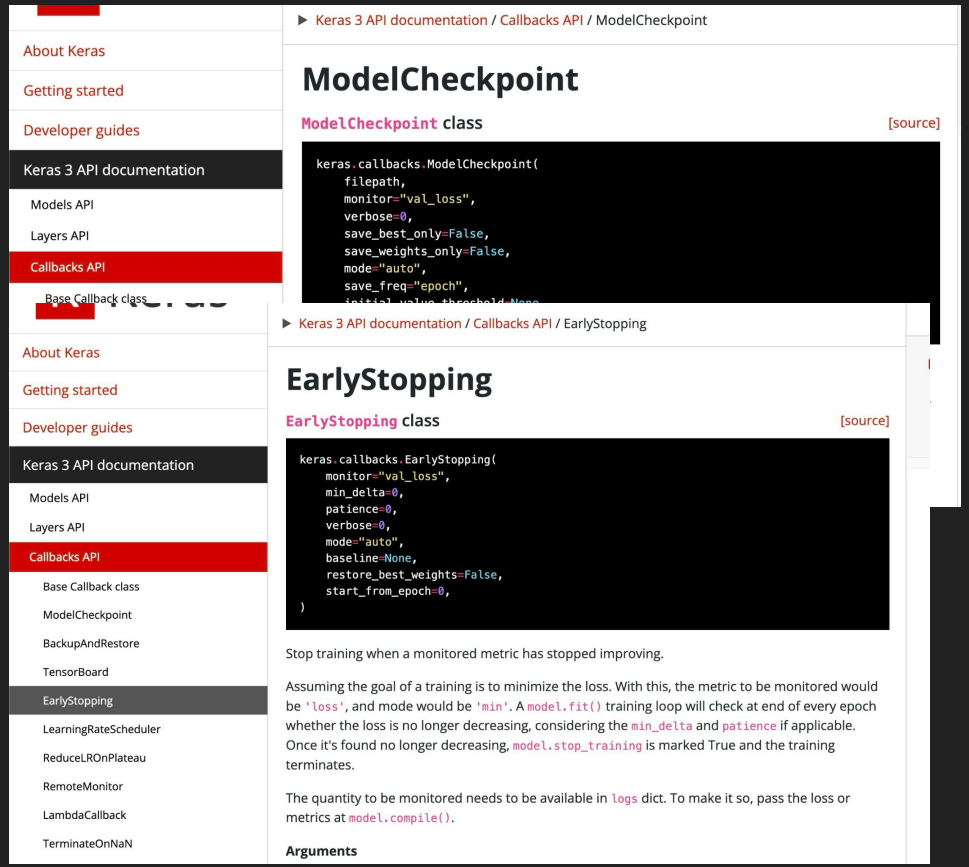
Arguments

- **monitor**: Quantity to be monitored. Defaults to `"val_loss"`.
- **min_delta**: Minimum change in the monitored quantity to qualify as an improvement, i.e. an absolute change of less than min_delta, will count as no improvement. Defaults to `0`.
- **patience**: Number of epochs with no improvement after which training will be stopped. Defaults to `0`.
- **verbose**: Verbosity mode, 0 or 1. Mode 0 is silent, and mode 1 displays messages when the callback takes an action. Defaults to `0`.
- **mode**: One of `{"auto", "min", "max"}`. In `min` mode, training will stop when the quantity monitored has stopped decreasing; in `"max"` mode it will stop when the quantity monitored has stopped increasing; in `"auto"` mode, the direction is automatically inferred from the name of the monitored quantity. Defaults to `"auto"`.
- **baseline**: Baseline value for the monitored quantity. If not `None`, training will stop if the model doesn't show improvement over the baseline. Defaults to `None`.
- **restore_best_weights**: Whether to restore model weights from the epoch with the best value of the monitored quantity. If `False`, the model weights obtained at the last step of training are used. An epoch will be restored regardless of the performance relative to the `baseline`. If no epoch improves on `baseline`, training will run for `patience` epochs and restore weights from the best epoch in that set. Defaults to `False`.
- **start_from_epoch**: Number of epochs to wait before starting to monitor improvement. This allows for a warm-up period in which no improvement is expected and thus training will not be stopped. Defaults to `0`.

https://keras.io/api/callbacks/early_stopping/

@see https://keras.io/api/callbacks/early_stopping/

*Best Model*을 만들기 위한 노력



model check point
early stopping...

Callbacks API?

Callback ?

callback ⋮ 개요 사용 예시 발음 유의어 및 반의어

정의 출처: Oxford Languages · 자세히 알아보기

noun

1. an invitation to return for a second audition or interview.
   "he was one of only twelve applicants to receive a callback"
2. a phone call made to return a call received.

프로그래밍에서 콜백(callback) 또는 콜백 함수(callback function)는 다른 코드의 인수로서 넘겨주는 실행 가능한 코드를 말한다. 콜백을 넘겨받는 코드는 이 콜백을 필요에 따라 즉시 실행할 수도 있고, 아니면 나중에 실행할 수도 있다.

Wikipedia
https://ko.wikipedia.org › wiki › 콜백 ⋮

콜백 - 위키백과, 우리 모두의 백과사전

Keras 에서의 CallbackAPI

Callbacks API

A callback is an object that can perform actions at various stages of training (e.g. at the start or end of an epoch, before or after a single batch, etc).

You can use callbacks to:

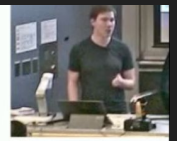
- Write TensorBoard logs after every batch of training to monitor your metrics
- Periodically save your model to disk
- Do early stopping
- Get a view on internal states and statistics of a model during training
- ...and more

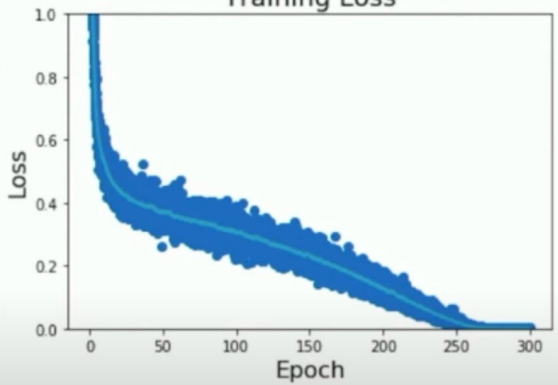
복습) Learning Rate?

Learning Rate도 제어 할 수 있을까?

Learning Rate Decay

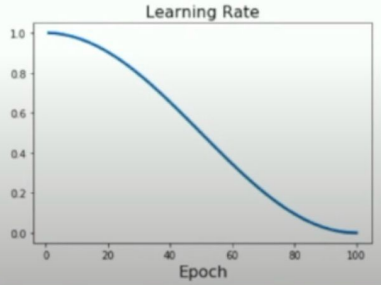
Learning Rate Decay: Cosine





Step: Reduce learning rate at a few fixed points. E.g. for ResNets, multiply LR by 0.1 after epochs 30, 60, and 90.

Cosine: $\alpha_t = \frac{1}{2}\alpha_0 \left(1 + \cos(t\pi/T)\right)$

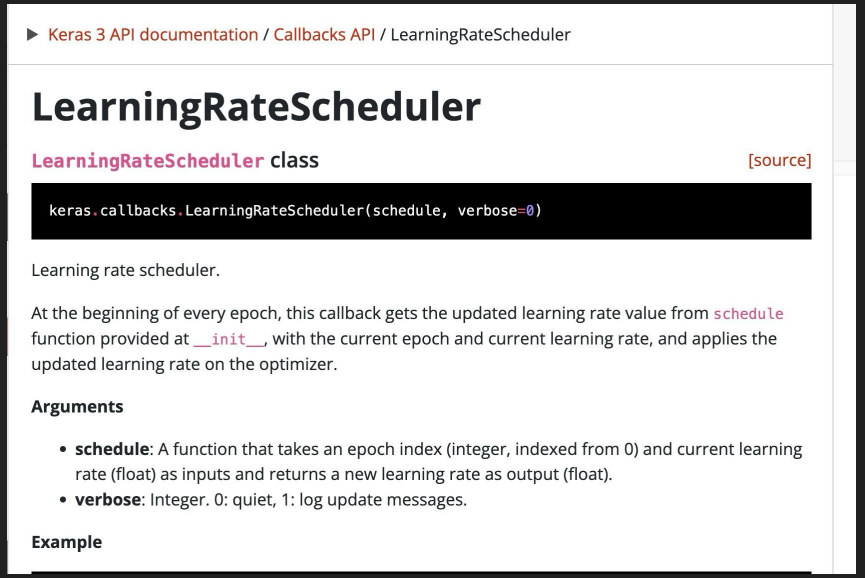


Loshchilov and Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts", ICLR 2017
Radford et al, "Improving Language Understanding by Generative Pre-Training", 2018
Feichtenhofer et al, "SlowFast Networks for Video Recognition", ICCV 2019
Radosavovic et al, "On Network Design Spaces for Visual Recognition", ICCV 2019
Child at al, "Generating Long Sequences with Sparse Transformers", arXiv 2019

https://www.youtube.com/watch?v=WUazOtIti0g

Keras LearningRateScheduler

epoch 에 따라 LR 을 조작해보자



▶ Keras 3 API documentation / Callbacks API / LearningRateScheduler

LearningRateScheduler

`LearningRateScheduler` class [source]

```
keras.callbacks.LearningRateScheduler(schedule, verbose=0)
```

Learning rate scheduler.

At the beginning of every epoch, this callback gets the updated learning rate value from `schedule` function provided at `__init__`, with the current epoch and current learning rate, and applies the updated learning rate on the optimizer.

Arguments

- **schedule**: A function that takes an epoch index (integer, indexed from 0) and current learning rate (float) as inputs and returns a new learning rate as output (float).
- **verbose**: Integer. 0: quiet, 1: log update messages.

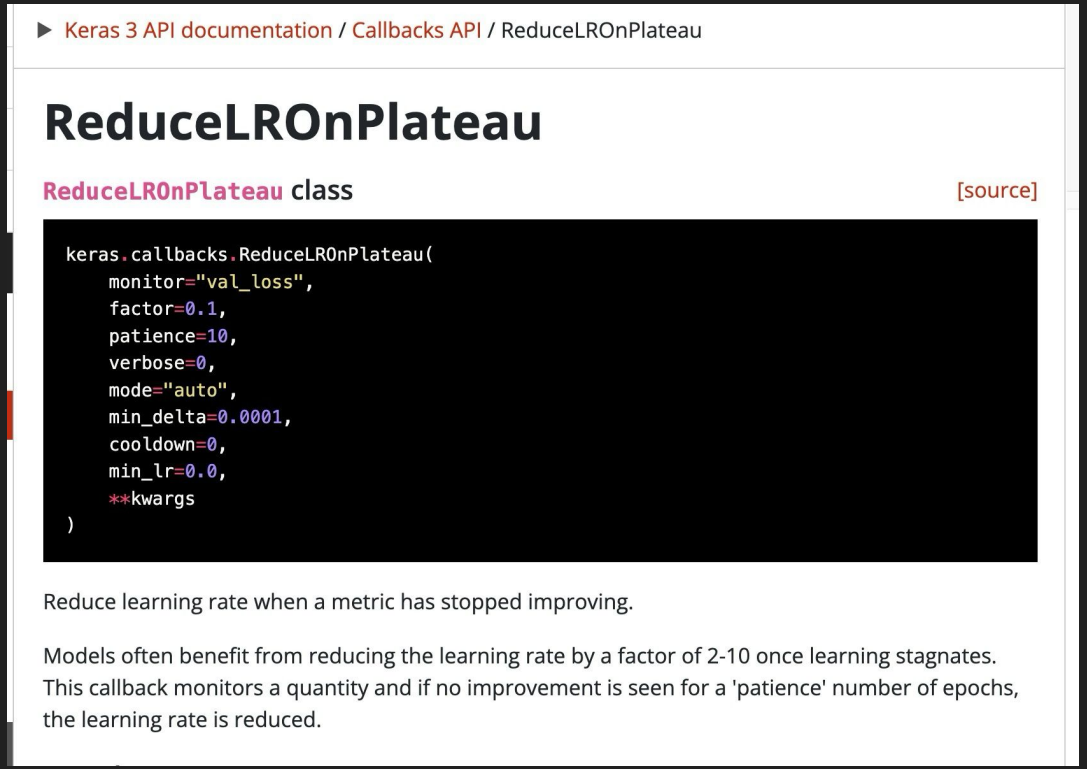
Example

ReduceLROnPlateau

LR(Learning Rate)

지표가 개선되지 않을 때

LR 을 줄여보자!



https://keras.io/api/callbacks/reduce_lr_on_plateau/