TensorFlow2教程-自訂回檔

自訂回檔是一個很好用的工具，可以在訓練，評估和推理期間自訂模型的行為，包括讀取/更改keras模型等。

In [1]:
```
from __future__ import absolute_import, division,print_function, unicode_literals
import tensorflow as tf
```

```
/home/doit/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWar
ning: Conversion of the second argument of issubdtype from `float` to `np.float
ing` is deprecated. In future, it will be treated as `np.float64 == np.dtype(fl
oat).type`.
  from ._conv import register_converters as _register_converters
```

1 Keras回檔簡介

在Kreas中，Callback是一個python類，旨在被子類化以提供特定功能，並在訓練的各階段（包括每個batch/epoch的開始和結束），以及測試中調用一組方法。

我們可以通過回檔列表，傳遞回檔方法，在訓練/評估/推斷的不同階段調用回檔方法。

構建一個模型

In [2]:
```
def get_model():
    model = tf.keras.Sequential()
    model.add(tf.keras.layers.Dense(1, activation = 'linear', input_dim = 784))
    model.compile(optimizer=tf.keras.optimizers.RMSprop(lr=0.1), loss='mean_squar
    return model
```

導入數據

In [3]:
```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train = x_train.reshape(60000, 784).astype('float32') / 255
x_test = x_test.reshape(10000, 784).astype('float32') / 255
```

定義一個簡單的自訂回檔，以跟蹤每批資料的開始和結束。

In [4]:
```
import datetime

class MyCustomCallback(tf.keras.callbacks.Callback):

    def on_train_batch_begin(self, batch, logs=None):
        print('Training: batch {} begins at {}'.format(batch, datetime.datetime.r

    def on_train_batch_end(self, batch, logs=None):
        print('Training: batch {} ends at {}'.format(batch, datetime.datetime.nov

    def on_test_batch_begin(self, batch, logs=None):
        print('Evaluating: batch {} begins at {}'.format(batch, datetime.datetime

    def on_test_batch_end(self, batch, logs=None):
        print('Evaluating: batch {} ends at {}'.format(batch, datetime.datetime.r
```

在訓練時傳入回呼函數

In [5]:
```
model = get_model()
_ = model.fit(x_train, y_train,
          batch_size=64,
          epochs=1,
          steps_per_epoch=5,
          verbose=0,
          callbacks=[MyCustomCallback()])
```

```
Training: batch 0 begins at 14:47:16.854006
Training: batch 0 ends at 14:47:19.215169
Training: batch 1 begins at 14:47:19.215812
Training: batch 1 ends at 14:47:19.219226
Training: batch 2 begins at 14:47:19.219737
Training: batch 2 ends at 14:47:19.222817
Training: batch 3 begins at 14:47:19.223307
Training: batch 3 ends at 14:47:19.226198
Training: batch 4 begins at 14:47:19.226991
Training: batch 4 ends at 14:47:19.229930
```

1.1 以下方法會調用回呼函數

fit(), fit_generator() 訓練或使用反覆運算資料進行訓練。

evaluate(), evaluate_generator() 評估或使用反覆運算資料進行評估。

predict(), predict_generator() 預測或使用反覆運算資料進行預測。

In [6]:
```
_ = model.evaluate(x_test, y_test, batch_size=128, verbose=0, steps=5,
          callbacks=[MyCustomCallback()])
```

```
Evaluating: batch 0 begins at 14:52:23.452381
Evaluating: batch 0 ends at 14:52:23.492404
Evaluating: batch 1 begins at 14:52:23.492663
Evaluating: batch 1 ends at 14:52:23.493493
Evaluating: batch 2 begins at 14:52:23.493606
Evaluating: batch 2 ends at 14:52:23.494279
Evaluating: batch 3 begins at 14:52:23.494388
Evaluating: batch 3 ends at 14:52:23.495022
Evaluating: batch 4 begins at 14:52:23.495128
Evaluating: batch 4 ends at 14:52:23.495710
```

2 回檔方法概述

2.1 訓練/測試/預測的常用方法

為了進行訓練，測試和預測，提供了以下方法來替代。

on_(train|test|predict)_begin(self, logs=None) 在fit/ evaluate/ predict開始時調用。

on_(train|test|predict)_end(self, logs=None) 在fit/ evaluate/ predict結束時調用。

on_(train|test|predict)_batch_begin(self, batch, logs=None) 在培訓/測試/預測期間處理批次之前立即調用。在此方法中，logs是帶有batch和size可用鍵的字典，代表當前批次號和批次大小。

on_(train|test|predict)_batch_end(self, batch, logs=None) 在培訓/測試/預測批次結束時調用。在此方法中，logs是一個包含狀態指標結果的字典。

2.2 訓練時特定方法

另外，為了進行培訓，提供以下內容。

on_epoch_begin（self，epoch，logs = None） 在訓練期間的開始時調用。

on_epoch_end（self，epoch，logs = None） 在訓練期間的末尾調用。

2.3 logsdict的用法

該logs字典包含損loss，已經每個batch和epoch的結束時的所有指標。示例包括loss和平均絕對誤差。

In [7]:
```
class LossAndErrorPrintingCallback(tf.keras.callbacks.Callback):

    def on_train_batch_end(self, batch, logs=None):
        print('For batch {}, loss is {:7.2f}.'.format(batch, logs['loss']))

    def on_test_batch_end(self, batch, logs=None):
        print('For batch {}, loss is {:7.2f}.'.format(batch, logs['loss']))

    def on_epoch_end(self, epoch, logs=None):
        print('The average loss for epoch {} is {:7.2f} and mean absolute error i

model = get_model()
_ = model.fit(x_train, y_train,
          batch_size=64,
          steps_per_epoch=5,
          epochs=3,
          verbose=0,
          callbacks=[LossAndErrorPrintingCallback()])
```

```
For batch 0, loss is   31.99.
For batch 1, loss is  888.84.
For batch 2, loss is   20.06.
For batch 3, loss is    8.30.
For batch 4, loss is   10.34.
The average loss for epoch 0 is  191.91 and mean absolute error is    8.17.
For batch 0, loss is    6.29.
For batch 1, loss is    7.78.
For batch 2, loss is    5.60.
For batch 3, loss is    6.66.
For batch 4, loss is    5.95.
The average loss for epoch 1 is    6.45 and mean absolute error is    2.06.
For batch 0, loss is    6.19.
For batch 1, loss is    5.54.
For batch 2, loss is    7.04.
For batch 3, loss is   10.74.
For batch 4, loss is   15.87.
The average loss for epoch 2 is    9.07 and mean absolute error is    2.46.
```

同樣，可以在evaluate時調用回檔。

In [9]:
```
_ = model.evaluate(x_test, y_test, batch_size=128, verbose=0, steps=20,
          callbacks=[LossAndErrorPrintingCallback()])
```

```
For batch 0, loss is   23.48.
For batch 1, loss is   20.85.
For batch 2, loss is   22.11.
For batch 3, loss is   24.52.
For batch 4, loss is   26.07.
For batch 5, loss is   23.15.
For batch 6, loss is   23.08.
For batch 7, loss is   21.37.
For batch 8, loss is   22.78.
For batch 9, loss is   22.38.
For batch 10, loss is   24.63.
For batch 11, loss is   26.51.
For batch 12, loss is   23.29.
For batch 13, loss is   25.12.
For batch 14, loss is   25.20.
For batch 15, loss is   22.27.
For batch 16, loss is   26.29.
For batch 17, loss is   25.76.
For batch 18, loss is   24.94.
For batch 19, loss is   23.94.
```

3 keras回檔示例

3.1 以最小的損失儘早停止

第一個示例展示了Callback通過達到最小損失時更改屬性model.stop_training（布林值），停止Keras訓練。用戶可以提供一個參數patience來指定訓練最終停止之前應該等待多少個時期。

注：tf.keras.callbacks.EarlyStopping 提供了更完整，更通用的實現。

In [17]:
```
import numpy as np
class EarlyStoppingAtMinLoss(tf.keras.callbacks.Callback):
    def __init__(self, patience=0):
        super(EarlyStoppingAtMinLoss, self).__init__()
        self.patience = patience
        self.best_weights = None  # loss最低時的權重
    def on_train_begin(self, logs=None):
        # loss不再下降時等待的輪數
        self.wait = 0
        # 停止時的輪數
        self.stopped_epoch = 0
        # 開始時的最優loss
        self.best = np.Inf

    def on_epoch_end(self, epoch, logs=None):
        current = logs.get('loss')
        if np.less(current, self.best):
            self.best = current
            self.wait = 0
            # 最佳權重
            self.best_weights = self.model.get_weights()
        else:
            self.wait += 1
            if self.wait >= self.patience:
                self.stopped_epoch = epoch
                self.model.stop_training = True
                print('導入當前最佳模型')
                self.model.set_weights(self.best_weights)
    def on_train_end(self, logs=None):
        if self.stopped_epoch > 0:
            print('在%05d: 提前停止訓練'% (self.stopped_epoch+1))
```

In [18]:
```
model = get_model()
_ = model.fit(x_train, y_train,
          batch_size=64,
          steps_per_epoch=5,
          epochs=30,
          verbose=0,
          callbacks=[LossAndErrorPrintingCallback(), EarlyStoppingAtMinLoss()])
```

```
For batch 0, loss is   23.36.
For batch 1, loss is 1043.09.
For batch 2, loss is   25.19.
For batch 3, loss is    8.33.
For batch 4, loss is    6.51.
The average loss for epoch 0 is  221.30 and mean absolute error is    8.41.
For batch 0, loss is   10.12.
For batch 1, loss is    6.01.
For batch 2, loss is    7.86.
For batch 3, loss is    6.52.
For batch 4, loss is    5.53.
The average loss for epoch 1 is    7.21 and mean absolute error is    2.26.
For batch 0, loss is    4.40.
For batch 1, loss is    4.59.
For batch 2, loss is    5.08.
For batch 3, loss is    4.74.
For batch 4, loss is    6.71.
The average loss for epoch 2 is    5.10 and mean absolute error is    1.89.
For batch 0, loss is    5.96.
For batch 1, loss is    4.13.
For batch 2, loss is    5.94.
For batch 3, loss is    7.71.
For batch 4, loss is   16.43.
The average loss for epoch 3 is    8.03 and mean absolute error is    2.30.
导入当前最佳模型
在00004: 提前停止训练
```

自訂學習率

在模型訓練中通常要做的一件事是隨著訓練輪次改變學習率。Keras後端公開了可用於設置變數的get_value API。在此示例中，我們展示了如何使用自訂的回檔來動態更改學習率。

注：這只是示例實現，請參見callbacks.LearningRateScheduler和keras.optimizers.schedules有關更一般的實現。

In [19]:
```
class LearningRateScheduler(tf.keras.callbacks.Callback):
    def __init__(self, schedule):
        super(LearningRateScheduler, self).__init__()
        self.schedule = schedule

    def on_epoch_begin(self, epoch, logs=None):
        if not hasattr(self.model.optimizer, 'lr'):
            raise ValueError('Optimizer沒有lr參數。')
        # 獲取當前lr
        lr = float(tf.keras.backend.get_value(self.model.optimizer.lr))
        # 調整lr
        scheduled_lr = self.schedule(epoch, lr)
        tf.keras.backend.set_value(self.model.optimizer.lr, scheduled_lr)
        print('Epoch %05d: 學習率為%6.4f.'%(epoch, scheduled_lr))
```

按輪次調整學習率

In [20]:
```
LR_SCHEDULE = [
    # (epoch to start, learning rate) tuples
    (3, 0.05), (6, 0.01), (9, 0.005), (12, 0.001)
]

def lr_schedule(epoch, lr):
    if epoch < LR_SCHEDULE[0][0] or epoch > LR_SCHEDULE[-1][0]:
        return lr
    for i in range(len(LR_SCHEDULE)):
        if epoch == LR_SCHEDULE[i][0]:
            return LR_SCHEDULE[i][1]
    return lr

model = get_model()
_ = model.fit(x_train, y_train,
          batch_size=64,
          steps_per_epoch=5,
          epochs=15,
          verbose=0,
          callbacks=[LossAndErrorPrintingCallback(), LearningRateScheduler(lr_sch
```

```
Epoch 00000: 学习率为0.1000.
For batch 0, loss is   24.55.
For batch 1, loss is 1134.79.
For batch 2, loss is   26.95.
For batch 3, loss is    8.14.
For batch 4, loss is    6.28.
The average loss for epoch 0 is  240.14 and mean absolute error is    8.91.
Epoch 00001: 学习率为0.1000.
For batch 0, loss is    7.12.
For batch 1, loss is    7.65.
For batch 2, loss is    5.11.
For batch 3, loss is    5.36.
For batch 4, loss is    5.43.
The average loss for epoch 1 is    6.13 and mean absolute error is    2.06.
Epoch 00002: 学习率为0.1000.
For batch 0, loss is    5.18.
For batch 1, loss is    5.11.
For batch 2, loss is    5.08.
For batch 3, loss is    4.34.
For batch 4, loss is    4.21.
The average loss for epoch 2 is    4.78 and mean absolute error is    1.77.
Epoch 00003: 学习率为0.0500.
For batch 0, loss is    5.30.
For batch 1, loss is    3.95.
For batch 2, loss is    4.07.
For batch 3, loss is    3.63.
For batch 4, loss is    4.44.
The average loss for epoch 3 is    4.28 and mean absolute error is    1.69.
Epoch 00004: 学习率为0.0500.
For batch 0, loss is    3.40.
For batch 1, loss is    7.80.
For batch 2, loss is    4.98.
For batch 3, loss is    3.08.
```

```
For batch 4, loss is    4.66.
The average loss for epoch 4 is    4.78 and mean absolute error is    1.75.
Epoch 00005: 学习率为0.0500.
For batch 0, loss is    3.86.
For batch 1, loss is    6.12.
For batch 2, loss is    4.59.
For batch 3, loss is    3.81.
For batch 4, loss is    5.60.
The average loss for epoch 5 is    4.80 and mean absolute error is    1.70.
Epoch 00006: 学习率为0.0100.
For batch 0, loss is    5.00.
For batch 1, loss is    4.14.
For batch 2, loss is    3.43.
For batch 3, loss is    3.61.
For batch 4, loss is    3.96.
The average loss for epoch 6 is    4.03 and mean absolute error is    1.58.
Epoch 00007: 学习率为0.0100.
For batch 0, loss is    5.50.
For batch 1, loss is    4.07.
For batch 2, loss is    4.17.
For batch 3, loss is    3.99.
For batch 4, loss is    3.56.
The average loss for epoch 7 is    4.26 and mean absolute error is    1.60.
Epoch 00008: 学习率为0.0100.
For batch 0, loss is    3.93.
For batch 1, loss is    3.56.
For batch 2, loss is    3.22.
For batch 3, loss is    3.66.
For batch 4, loss is    4.47.
The average loss for epoch 8 is    3.77 and mean absolute error is    1.52.
Epoch 00009: 学习率为0.0050.
For batch 0, loss is    4.83.
For batch 1, loss is    4.33.
For batch 2, loss is    3.32.
For batch 3, loss is    4.38.
For batch 4, loss is    3.41.
The average loss for epoch 9 is    4.05 and mean absolute error is    1.63.
Epoch 00010: 学习率为0.0050.
For batch 0, loss is    3.49.
For batch 1, loss is    3.63.
For batch 2, loss is    3.74.
For batch 3, loss is    3.56.
For batch 4, loss is    4.65.
The average loss for epoch 10 is    3.81 and mean absolute error is    1.54.
Epoch 00011: 学习率为0.0050.
For batch 0, loss is    3.46.
For batch 1, loss is    4.41.
For batch 2, loss is    2.83.
For batch 3, loss is    5.34.
For batch 4, loss is    3.91.
The average loss for epoch 11 is    3.99 and mean absolute error is    1.53.
Epoch 00012: 学习率为0.0010.
For batch 0, loss is    4.89.
For batch 1, loss is    3.13.
For batch 2, loss is    3.65.
For batch 3, loss is    3.14.
For batch 4, loss is    2.93.
```

```
The average loss for epoch 12 is    3.55 and mean absolute error is    1.53.
Epoch 00013: 学习率为0.0010.
For batch 0, loss is    3.93.
For batch 1, loss is    3.33.
For batch 2, loss is    3.01.
For batch 3, loss is    5.31.
For batch 4, loss is    3.33.
The average loss for epoch 13 is    3.78 and mean absolute error is    1.56.
Epoch 00014: 学习率为0.0010.
For batch 0, loss is    4.76.
For batch 1, loss is    3.42.
For batch 2, loss is    3.05.
For batch 3, loss is    4.45.
For batch 4, loss is    4.50.
The average loss for epoch 14 is    4.04 and mean absolute error is    1.57.
```

In [ ]: