Keras fit_generator的steps_per_epoch

张树刚
谨言慎行

fit_generator教程太多，这里只专注于steps_per_epoch。

先看个官方的例子

```
def generate_arrays_from_file(path):
    while True:
    with open(path) as f:
    for line in f:
    # create numpy arrays of input data
    # and labels, from each line in the file
                        x1, x2, y = process_line(line)
    yield ({'input_1': x1, 'input_2': x2}, {'output': y})

model.fit_generator(generate_arrays_from_file('./my_folder'),steps_per_epoch=10000, e
```

这个demo大致实现的功能是构造一个能从文件中不断生成数据的generator，这也是fit_generator最简单的用法。这里与fit的区别之一在于，model.fit()需要传递的参数是batch_size，而model.fit_generator()则需要传递一个叫steps_per_epoch的参数，而并没有指定batch_size。这是为什么呢？

不同于fit()一次性加载所有的train数据集，遍历一遍就可以作为一轮epoch的结束，generator是可以从给定的数据集中“无限”生成数据的，并且因为一次只加载数据集的一部分（generator就是为了解决数据集过大无法一次性加载到内存的问题），所以他并不知道什么时候才是一轮epoch的结束。同样的，batch_size也没有作为参数传递给fit_generator()，所以必须有机制来判断：(1)什么时候结束一轮epoch (2)batch_size是多少。

这时候steps_per_epoch就顺理成章的出现了。这个参数实际上就是指定了每一轮epoch需要执行多少steps，也就是多少steps，才能认为一轮epoch结束。那么衍生问题就是，一个step是怎么度量？其实就是规定每个step加载多少数据，也就是batch_size。他们的关系如下：

steps_per_epoch=len(x_train)/batch_size

一句话概括，就是对于整个训练数据集，generator要在多少步内完成一轮遍历（epoch），从而也就规定了每步要加载多少数据（batch_size）。

进阶

用keras Sequence实例作为输入。这个方法在batch_size问题上有点反其道而行之，他重新回归直接设置batch_size，而不再间接通过steps_per_epoch设置。看代码：

```
class MnistSequence(Sequence):

    def __init__(self, x_set, y_set, batch_size):
        self.x, self.y = x_set, y_set
        self.batch_size = batch_size

    def __len__(self):
        return int(np.ceil(len(self.x) / float(self.batch_size)))

    def __getitem__(self, idx):
        batch_x = self.x[idx * self.batch_size:(idx + 1) * self.batch_size]
        batch_y = self.y[idx * self.batch_size:(idx + 1) * self.batch_size]

        return np.array([
                resize(imread(file_name), (200, 200))
                for file_name in batch_x]), np.array(batch_y)

model.fit_generator(MnistSequence(training_x, training_y, 32), epochs=10000)
```

先看最后一行，这个demo和上面的demo的主要区别就是定义和实现了一个Sequence子类的实例MnistSequence，作为一种customized generator，相比上一个generator，这个稍微复杂一些。具体的按下不表，这里仍关注steps_per_epoch的问题。注意到这里的fit_generator并没有传递steps_per_epoch?

刚才提到了Sequence方法有点反其道而行之，他实际上在初始化MnistSequence子类实例对象的时候，就需要指定batch_size并传递给__init__()进行实例化。那么没有steps_per_epoch，如何知道一个epoch结束？答案是__len__()这个函数。可以看到他实现的功能也就是上部分列出的公式steps_per_epoch=len(x_train)/batch_size。至于为什么__len__()对应的就是steps_per_epoch，这里没有再细究了，但是至此为止，理解上已经比较通透，不影响使用了。

后话

不出意外，之后一段时间应该不会再用fit_generator了，COCO数据集大小高达25G，但minist才几十M，自己目前在做的更小了，甚至不到10 M，加载进内存完全没有问题。而对于小数据集，fit_generator因为仍然每个batch都要从硬盘加载（不确定？），导致程序运行起来特别慢。另外，昨天简单查了下资料，pytorch似乎没有直接对应generator的机制。至于pytorch如何应对大数据集，以后用到了再去细查吧。

So，byebye generator！

参考文献

[1] blog.csdn.net/leviopku/...

[2] blog.csdn.net/Trent1985... 数据加载

[3] blog.csdn.net/mlp750303...

[4] zhihu.com/question/2659...

[5] blog.csdn.net/qq_399386...

发布于 2020-07-30

Keras 深度学习（Deep Learning）