Keras简介

Keras中文文档

https://keras-cn.readthedocs.io/en/latest/for\_beginners/concepts/

Keras的底层库使用Theano或TensorFlow，这两个库也称为Keras的后端。无论是Theano还是TensorFlow，都是一个“符号式”的库。

符号主义的计算首先定义各种变量，然后建立一个“计算图”，计算图规定了各个变量之间的计算关系。建立好的计算图需要编译以确定其内部细节，

然而，此时的计算图还是一个“空壳子”，里面没有任何实际的数据，只有当你把需要运算的输入放进去后，才能在整个模型中形成数据流，从而形成输出值。

就像用管道搭建供水系统，当你在拼水管的时候，里面是没有水的。只有所有的管子都接完了，才能送水。

## 张量

张量，或tensor，是本文档会经常出现的一个词汇，在此稍作解释。

使用这个词汇的目的是为了表述统一，张量可以看作是向量、矩阵的自然推广，我们用张量来表示广泛的数据类型。

规模最小的张量是0阶张量，即标量，也就是一个数。

当我们把一些数有序的排列起来，就形成了1阶张量，也就是一个向量

如果我们继续把一组向量有序的排列起来，就形成了2阶张量，也就是一个矩阵

把矩阵摞起来，就是3阶张量，我们可以称为一个立方体，具有3个颜色通道的彩色图片就是一个这样的立方体

把立方体摞起来，好吧这次我们真的没有给它起别名了，就叫4阶张量了，不要去试图想像4阶张量是什么样子，它就是个数学上的概念。

张量的阶数有时候也称为维度，或者轴，轴这个词翻译自英文axis

Theano模式会把100张RGB三通道的16×32（高为16宽为32）彩色图表示为下面这种张量形式（100,3,16,32）

而TensorFlow，的表达形式是（100,16,32,3），即把通道维放在了最后，这种数据组织方式称为“channels\_last”。

唉，真是蛋疼，你们商量好不行吗？

## 函数式模型

函数式模型算是本文档比较原创的词汇了，所以这里要说一下

在Keras 0.x中，模型其实有两种，一种叫Sequential，称为序贯模型，也就是单输入单输出，一条路通到底，层与层之间只有相邻关系，

跨层连接统统没有。这种模型编译速度快，操作上也比较简单。第二种模型称为Graph，即图模型，这个模型支持多输入多输出，

层与层之间想怎么连怎么连，但是编译速度慢。可以看到，Sequential其实是Graph的一个特殊情况。

在Keras1和Keras2中，图模型被移除，而增加了了“functional model API”，这个东西，更加强调了Sequential是特殊情况这一点。

一般的模型就称为Model，然后如果你要用简单的Sequential，OK，那还有一个快捷方式Sequential。

由于functional model API在使用时利用的是“函数式编程”的风格，我们这里将其译为函数式模型。

总而言之，只要这个东西接收一个或一些张量作为输入，然后输出的也是一个或一些张量，那不管它是什么鬼，统统都称作“模型”。

## batch

这个概念与Keras无关，老实讲不应该出现在这里的，但是因为它频繁出现，而且不了解这个技术的话看函数说明会很头痛，这里还是简单说一下。

深度学习的优化算法，说白了就是梯度下降。每次的参数更新有两种方式。

第一种，遍历全部数据集算一次损失函数，然后算函数对各个参数的梯度，更新梯度。这种方法每更新一次参数都要把数据集里的所有样本都看一遍，

计算量开销大，计算速度慢，不支持在线学习，这称为Batch gradient descent，批梯度下降。

另一种，每看一个数据就算一下损失函数，然后求梯度更新参数，这个称为随机梯度下降，stochastic gradient descent。这个方法速度比较快，

但是收敛性能不太好，可能在最优点附近晃来晃去，hit不到最优点。两次参数的更新也有可能互相抵消掉，造成目标函数震荡的比较剧烈。

为了克服两种方法的缺点，现在一般采用的是一种折中手段，mini-batch gradient decent，小批的梯度下降，这种方法把数据分为若干个批。

按批来更新参数，这样，一个批中的一组数据共同决定了本次梯度的方向，下降起来就不容易跑偏，减少了随机性。另一方面因为批的样本数与整个数据集相比小了很多，计算量也不是很大。

基本上现在的梯度下降都是基于mini-batch的，所以Keras的模块中经常会出现batch\_size，就是指这个。

顺便说一句，Keras中用的优化器SGD是stochastic gradient descent的缩写，但不代表是一个样本就更新一回，还是基于mini-batch的。

iteration：表示1次迭代（也叫training step，完成一次参数迭代运算），每次迭代更新1次网络结构的参数；

batch-size：1次迭代所使用的样本量；

epoch：1个epoch表示过了1遍训练集中的所有样本。

值得注意的是，在深度学习领域中，常用带mini-batch的随机梯度下降算法（Stochastic Gradient Descent, SGD）训练深层结构，

它有一个好处就是并不需要遍历全部的样本，当数据量非常大时十分有效。此时，可根据实际问题来定义epoch，

例如定义10000次迭代为1个epoch，若每次迭代的batch-size设为256，那么1个epoch相当于过了2560000个训练样本。

为了使参数达到收敛我们需要不断训练直到参数趋于收敛，epoch就是训练轮数，

## 指定输入数据的shape

模型需要知道输入数据的shape，因此，Sequential的第一层需要接受一个关于输入数据shape的参数，后面的各个层则可以自动的推导出中间数据的shape，因此不需要为每个层都指定这个参数。有几种方法来为第一层指定输入数据的shape

传递一个input\_shape的关键字参数给第一层，input\_shape是一个tuple类型的数据，其中也可以填入None，如果填入None则表示此位置可能是任何正整数。数据的batch大小不应包含在其中。

有些2D层，如Dense（全连接层），支持通过指定其输入维度input\_dim来隐含的指定输入数据shape,是一个Int类型的数据。一些3D的时域层支持通过参数input\_dim和input\_length来指定输入shape。

如果你需要为输入指定一个固定大小的batch\_size（常用于stateful RNN网络），可以传递batch\_size参数到一个层中，例如你想指定输入张量的batch大小是32，数据shape是（6，8），则你需要传递batch\_size=32和input\_shape=(6,8)。

## 编译

在训练模型之前，我们需要通过compile来对学习过程进行配置。compile接收三个参数：

优化器optimizer：该参数可指定为已预定义的优化器名，如rmsprop、adagrad，或一个Optimizer类的对象，详情见optimizers

损失函数loss：该参数为模型试图最小化的目标函数，它可为预定义的损失函数名，如categorical\_crossentropy、mse，也可以为一个损失函数。详情见losses

指标列表metrics：对分类问题，我们一般将该列表设置为metrics=['accuracy']。指标可以是一个预定义指标的名字,也可以是一个用户定制的函数.指标函数应该返回单个张量,或一个完成metric\_name - > metric\_value映射的字典.请参考性能评估