# **用 Keras 构建神经网络**

幸运的是，每次我们需要使用神经网络时，都不需要编写激活函数、梯度下降等。有很多包可以帮助我们，建议你了解这些包，包括以下包：

* **[Keras](https://keras.io/" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/7c042ed0-08e7-4138-ad78-1a8d671d5da5/concepts/_blank)**
* **[TensorFlow](https://www.tensorflow.org/" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/7c042ed0-08e7-4138-ad78-1a8d671d5da5/concepts/_blank)**
* **[Caffe](http://caffe.berkeleyvision.org/" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/7c042ed0-08e7-4138-ad78-1a8d671d5da5/concepts/_blank)**
* **[Theano](http://deeplearning.net/software/theano/" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/7c042ed0-08e7-4138-ad78-1a8d671d5da5/concepts/_blank)**
* **[Scikit-learn](http://scikit-learn.org/" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/7c042ed0-08e7-4138-ad78-1a8d671d5da5/concepts/_blank)**
* 以及很多其他包！

在这门课程中，我们将学习 **[Keras](https://keras.io/" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/7c042ed0-08e7-4138-ad78-1a8d671d5da5/concepts/_blank)**。Keras 使神经网络的编写过程更简单。为了展示有多简单，你将用几行代码构建一个完全连接的简单网络。

我们会将在前几课学习的概念与 Keras 提供的方法关联起来。

该示例的一般流程是首先加载数据，然后定义网络，最后训练网络。

## **用 Keras 构建神经网络**

要使用 Keras，你需要知道以下几个核心概念。

## **序列模型**

**from** keras.models **import** Sequential

*#Create the Sequential model*

model = Sequential()

**[keras.models.Sequential](https://keras.io/models/sequential/" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/7c042ed0-08e7-4138-ad78-1a8d671d5da5/concepts/_blank)** 类是神经网络模型的封装容器。它会提供常见的函数，例如 fit()、evaluate() 和 compile()。我们将介绍这些函数（在碰到这些函数的时候）。我们开始研究模型的层吧。

## **层**

Keras 层就像神经网络层。有全连接层、最大池化层和激活层。你可以使用模型的 add()函数添加层。例如，简单的模型可以如下所示：

**from** keras.models **import** Sequential

**from** keras.layers.core **import** Dense, Activation, Flatten

*#创建序列模型*

model = Sequential()

*#第一层 - 添加有128个节点的全连接层以及32个节点的输入层*

model.add(Dense(128, input\_dim=32))

*#第二层 - 添加 softmax 激活层*

model.add(Activation('softmax'))

*#第三层 - 添加全连接层*

model.add(Dense(10))

*#第四层 - 添加 Sigmoid 激活层*

model.add(Activation('sigmoid'))

Keras 将根据第一层自动推断后续所有层的形状。这意味着，你只需为第一层设置输入维度。

上面的第一层 model.add(Dense(input\_dim=32)) 将维度设为 32（表示数据来自 32 维空间）。第二层级获取第一层级的输出，并将输出维度设为 128 个节点。这种将输出传递给下一层级的链继续下去，直到最后一个层级（即模型的输出）。可以看出输出维度是 10。

构建好模型后，我们就可以用以下命令对其进行编译。我们将损失函数指定为我们一直处理的 categorical\_crossentropy。我们还可以指定优化程序，稍后我们将了解这一概念，暂时将使用 adam。最后，我们可以指定评估模型用到的指标。我们将使用准确率。

model.compile(loss="categorical\_crossentropy", optimizer="adam", metrics = ['accuracy'])

我们可以使用以下命令来查看模型架构：

model.summary()

然后使用以下命令对其进行拟合，指定 epoch 次数和我们希望在屏幕上显示的信息详细程度。

然后使用fit命令训练模型并通过 epoch 参数来指定训练轮数（周期），每 epoch 完成对整数据集的一次遍历。 verbose 参数可以指定显示训练过程信息类型，这里定义为 0 表示不显示信息。

model.fit(X, y, nb\_epoch=1000, verbose=0)

*注意：在 Keras 1 中，nb\_epoch 会设置 epoch 次数，但是在 Keras 2 中，变成了 epochs。*

最后，我们可以使用以下命令来评估模型：

model.evaluate()

# **Keras 优化程序**

Keras 中有很多优化程序，建议你访问此**[链接](https://keras.io/optimizers/" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/7c042ed0-08e7-4138-ad78-1a8d671d5da5/concepts/_blank)**或这篇精彩**[博文](http://sebastianruder.com/optimizing-gradient-descent/index.html" \l "rmsprop" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/7c042ed0-08e7-4138-ad78-1a8d671d5da5/concepts/_blank)**（此链接来自外网，国内网络可能打不开），详细了解这些优化程序。这些优化程序结合使用了上述技巧，以及其他一些技巧。最常见的包括：

#### **SGD**

这是随机梯度下降。它使用了以下参数：

* 学习速率。
* 动量（获取前几步的加权平均值，以便获得动量而不至于陷在局部最低点）。
* Nesterov 动量（当最接近解决方案时，它会减缓梯度）。

#### **Adam**

Adam (Adaptive Moment Estimation) 使用更复杂的指数衰减，不仅仅会考虑平均值（第一个动量），并且会考虑前几步的方差（第二个动量）。

#### **RMSProp**

RMSProp (RMS 表示均方根误差）通过除以按指数衰减的平方梯度均值来减小学习速率。