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
TensorFlow
Caffe
Theano
Scikit-learn
```

以及很多其他包！
在这门课程中，我们将学习 Keras。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 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级吧。

层级

Keras 层级就像神经网络层级。有完全连接的层级、最大池化层级和激活层级。你可以使用模型的 add() 函数添加层级。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()

#第一层级 - 添加有 32 个节点的输入层
model.add(Dense, input_dim=32)

#第二层级 - 添加有 128 个节点的完全连接层级
model.add(Dense(128))

#第三层级 - 添加 softmax 激活层级
model.add(Activation('softmax'))

#第四层级 - 添加完全连接的层级
model.add(Dense(10))

#第五层级 - 添加 Sigmoid 激活层级
model.add(Activation('sigmoid'))
```

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

上面的第一层级 model.add(Flatten(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。

```
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()
```

很简单，对吧？我们实践操作下。

练习

我们从最简单的示例开始。在此测验中，你将构建一个简单的多层前向反馈神经网络以解决 XOR 问题。

将第一层级设为 Flatten() 层级，并将 input_dim 设为 2。
将第二层级设为 Dense() 层级，并将输出宽度设为 8。
在第二层级之后使用 softmax 激活函数。
将输出层级宽度设为 2，因为输出只有 2 个类别。
在输出层级之后使用 softmax 激活函数。
对模型运行 10 个 epoch。
准确度应该为 50%。可以接受，当然肯定不是太理想！在 4 个点中，只有 2 个点分类正确？我们试着修改某些参数，以改变这一状况。例如，你可以增加 epoch 次数。如果准确率达到 75%，你将通过这道测验。能尝试达到 100% 吗？

首先，查看关于模型和层级的 Keras 文档。 Keras 多层感知器网络示例和你要构建的类似。请将该示例当做指南，但是注意有很多不同之处。

- https://github.com/keras-team/keras/blob/master/examples/mnist_mlp.py

```
import numpy as np
from keras.utils import np_utils
import tensorflow as tf
# Using TensorFlow 1.0.0; use tf.python_io in later versions
tf.python.control_flow_ops = tf

# Set random seed
np.random.seed(42)

# Our data
X = np.array([[0,0],[0,1],[1,0],[1,1]]).astype('float32')
y = np.array([[0],[1],[1],[0]]).astype('float32')

# Initial Setup for Keras
from keras.models import Sequential
from keras.layers.core import Dense, Activation

y = np_utils.to_categorical(y)
# Building the model
xor = Sequential()

# Add required layers
xor.add(Dense(32,input_dim=2))
xor.add(Activation("sigmoid"))
xor.add(Dense(2))
xor.add(Activation("softmax"))
# Specify loss as "binary_crossentropy", optimizer as "adam",
# and add the accuracy metric
xor.compile(loss="categorical_crossentropy", optimizer="adam", metrics = ['accuracy'])

# Uncomment this line to print the model architecture
# xor.summary()

# Fitting the model
history = xor.fit(X, y, nb_epoch=2000, verbose=0)

# Scoring the model
score = xor.evaluate(X, y)
print("\nAccuracy: ", score[-1])

# Checking the predictions
print("\nPredictions:")
print(xor.predict_proba(X))
```

> 输出

```
Using TensorFlow backend.

4/4 [==============================] - 0s

Accuracy:  1.0

Predictions:

4/4 [==============================] - 0s
[[0.769951   0.23004903]
 [0.28393105 0.7160689 ]
 [0.27931508 0.7206849 ]
 [0.6884088  0.3115912 ]]
```