



Keras 是深度学习神经网络最常用的框架之一，它是一个高级神经网络

API，用 Python 编写，能够在 TensorFlow 等工具库之上运行。

因搭建神经网络时的简单易用性，Keras API

打包为 `tf.keras` 封装在 TensorFlow 中。



示例

```
import numpy as np

from keras.models import Sequential # 顺序模型
from keras.layers import Dense # 全连接层
data = np.random.random((1000, 100)) # 数据
labels = np.random.randint(2, size=(1000, 1)) # 标签
model = Sequential() # 初始化顺序模型

# 添加全连接层
model.add(Dense(32, activation='relu', input_dim=100))

# 添加二分类全连接层
model.add(Dense(1, activation='sigmoid'))

# 编译模型
model.compile(optimizer='rmsprop', \
              loss='binary_crossentropy', metrics=['accuracy'])

model.fit(data, labels, epochs=10, batch_size=32) # 拟合数据
predictions = model.predict(data) # 预估数据
```

1.1 Keras 数据集

```
from keras.datasets import boston_housing, mnist, cifar10, imdb

# 手写数字数据集
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# 波士顿房价数据集
(x_train2, y_train2), (x_test2, y_test2) = boston_housing.load_data()

# cifar 图像分类数据集
(x_train3, y_train3), (x_test3, y_test3) = cifar10.load_data()

# imdb 评论数据集
(x_train4, y_train4), (x_test4, y_test4) = imdb.load_data(num_words=20000)
num_classes = 10
```

2. 数据预处理

2.1 序列填充

```
from keras.preprocessing import sequence

# 填充为固定长度 80 的序列
x_train4 = sequence.pad_sequences(x_train4, maxlen=80)
x_test4 = sequence.pad_sequences(x_test4, maxlen=80)
```

2.2 训练与测试集

```
from sklearn.model_selection import train_test_split
X_train5, X_test5, y_train5, y_test5 = train_test_split( \
    X, y, test_size=0.33, random_state=42)
```

2.3 独热编码

```
from keras.utils import to_categorical

# 类别标签独热编码转换
Y_train = to_categorical(y_train, num_classes)
Y_test = to_categorical(y_test, num_classes)
Y_train3 = to_categorical(y_train3, num_classes)
Y_test3 = to_categorical(y_test3, num_classes)
```

2.4 标准化 / 归一化

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(x_train2)
standardized_X = scaler.transform(x_train2)
standardized_X_test = scaler.transform(x_test2)
```

1. 数据加载

数据要存为 NumPy 数组或数组列表，使用 `sklearn.cross_validation` 的 `train_test_split` 模块进行分割将数据分割为训练集与测试集。

其它

```
from urllib.request import urlopen

data = np.loadtxt(urlopen("http://"), delimiter=",")
X = data[:, 0:8]
y = data[:, 8]
```

<http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data>

3. 模型架构

```
from keras.models import Sequential

model = Sequential()
model2 = Sequential()
model3 = Sequential()
```

3.1 顺序模型

3.3 卷积神经网络 (CNN)

```
from keras.layers import \
    Activation, Conv2D, MaxPooling2D, Flatten

model2.add(Conv2D(32, (3, 3), padding='same', \
                  input_shape=x_train.shape[1:])) # 2D 卷积层
model2.add(Activation('relu')) # ReLU 激活函数
model2.add(Conv2D(32, (3, 3))) # 2D 卷积层
model2.add(Activation('relu')) # ReLU 激活函数
model2.add(MaxPooling2D(pool_size=(2, 2))) # 池化层
model2.add(Dropout(0.25)) # 添加随机失活层

model2.add(Conv2D(64, (3, 3), padding='same'))
model2.add(Activation('relu'))
model2.add(Conv2D(64, (3, 3)))
model2.add(Activation('relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(Dropout(0.25))

model2.add(Flatten()) # 展平成 vector
model2.add(Dense(512)) # 全连接层
model2.add(Activation('relu')) # ReLU 激活函数
model2.add(Dropout(0.5)) # 添加随机失活层
# 类别数个神经元的全连接层
model2.add(Dense(num_classes))

model2.add(Activation('softmax')) # softmax 多分类
```

3.2 多层感知器 (MLP)

二进制分类

```
from keras.layers import Dense

model.add(Dense(12, input_dim=8, kernel_initializer='uniform', activation='relu')) # 添加 12 个神经元的全连接层
model.add(Dense(8, kernel_initializer='uniform', activation='relu')) # 添加 8 个神经元的全连接层
model.add(Dense(1, kernel_initializer='uniform', activation='sigmoid')) # 二分类
```

多级分类

```
from keras.layers import Dropout

model.add(Dense(512, activation='relu', input_shape=(784,))) # 添加 512 个神经元的全连接层
model.add(Dropout(0.2)) # 添加随机失活层
model.add(Dense(512, activation='relu')) # 添加 512 个神经元的全连接层
model.add(Dropout(0.2)) # 添加随机失活层
model.add(Dense(10, activation='softmax')) # 10 分类的全连接层
```

回归

```
model.add(Dense(64, activation='relu', input_dim=train_data.shape[1])) # 添加 64 个神经元的全连接层
model.add(Dense(1))
```

3.4 递归神经网络 (RNN)

```
from keras.layers import Embedding, LSTM

# 嵌入层
model3.add(Embedding(20000, 128))

# LSTM 层
model3.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))

# 二分类全连接
model3.add(Dense(1, activation='sigmoid'))
```

5. 编译模型

多层感知器: 二进制分类

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

多层感知器: 多级分类

```
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

多层感知器: 回归

```
model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
```

递归神经网络

```
model3.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

4. 审视模型

获取模型信息

```
model.output_shape # 模型输出形状
model.summary() # 模型摘要展示
model.get_config() # 模型配置
model.get_weights() # 列出模型的所有权重张量
```



扫码回复“工具库”

下载最新全套资料

6. 模型训练

在数据上拟合

```
model3.fit(x_train4, y_train4, batch_size=32, \
          epochs=15, verbose=1, validation_data=(x_test4, y_test4))
```

在测试集评估

```
score = model3.evaluate(x_test4, y_test4, batch_size=32)
```

7. 评估模型性能

预测标签与概率

```
model3.predict(x_test4, batch_size=32)
model3.predict_classes(x_test4, batch_size=32)
```

8. 预测

```
from keras.models import load_model
```

```
model3.save('model_file.h5')
```

```
my_model = load_model('model_file.h5')
```

9. 保存 / 加载模型

10. 模型微调

10.1 参数优化

```
from keras.optimizers import RMSprop

opt = RMSprop(lr=0.0001, decay=1e-6)

model2.compile(loss='categorical_crossentropy', \
               optimizer=opt, metrics=['accuracy'])
```

10.2 早停法

```
from keras.callbacks import EarlyStopping
```

最多等待 2 轮, 如果效果不提升, 就停止

```
early_stopping_monitor = EarlyStopping(patience=2)
```

```
model3.fit(x_train4, y_train4, batch_size=32, epochs=15, \
          validation_data=(x_test4, y_test4), \
          callbacks=[early_stopping_monitor])
```

数据科学工具库速查表



NumPy 是 Python 数据科学计算的核心库，提供了高性能多维数组对象及处理数组的工具。使用以下语句导入 NumPy 库：

```
import numpy as np
```



SciPy 是基于 NumPy 创建的 Python 科学计算核心库，提供了众多数学算法与函数。



Pandas 是基于 NumPy 创建的 Python 库，为 Python 提供了易于使用的的数据结构和数据分析工具。使用以下语句导入：

```
import pandas as pd
```



Matplotlib 是 Python 的二维绘图库，用于生成符合出版质量或跨平台交互环境的各类图形。

```
import matplotlib.pyplot as plt
```



Seaborn 是基于 matplotlib 开发的高阶 Python 数据可视图库，用于绘制优雅、美观的统计图形。使用下列别名导入该库：

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```



Bokeh 是 Python 的交互式可视图库，用于生成在浏览器里显示的大规模数据集高性能可视图。Bokeh 的中间层通用 **bokeh.plotting** 界面主要为两个组件：数据与图示例。

```
from bokeh.plotting import figure
```

```
from bokeh.io import output_file, show
```



PySpark 是 Spark 的 Python API，允许 Python 调用 Spark 编程模型。Spark SQL 是 Apache Spark 处理结构化数据模块。

AI 垂直领域工具库速查表



Scikit-learn 是开源的 Python 库，通过统一的界面实现机器学习、预处理、交叉验证及可视化算法。



Keras 是强大、易用的深度学习库，基于 Theano 和 TensorFlow 提供了高阶神经网络 API，用于开发和评估深度学习模型。



“TensorFlow™ is an open source software library for numerical computation using data flow graphs.” **TensorFlow** 是 Google 公司开发的机器学习架构，兼顾灵活性和扩展性，既适合用于工业生产也适合用于科学研究。



PyTorch 是 Facebook 团队 2017 年初发布的深度学习框架，有利于研究人员、爱好者、小规模项目等快速搞出原型。**PyTorch** 也是 Python 程序员最容易上手的深度学习框架。



Hugging Face 以开源的 NLP 预训练模型库 **Transformers** 而广为人知，目前 GitHub Star 已超过 54000+。**Transformers** 提供 100+ 种语言的 32 种预训练语言模型，简单，强大，高性能，是新手入门的不二选择。



OpenCV 是一个跨平台计算机视觉库，由 C 函数 /C++ 类构成，提供了 Python、MATLAB 等语言的接口。**OpenCV** 实现了图像处理和计算机视觉领域的很多通用算法。

编程语言速查表



SQL 是管理关系数据库的结构化查询语言，包括数据的增删查改等。作为数据分析的必备技能、岗位 JD 的重要关键词，SQL 是技术及相关岗位同学一定要掌握的语言。



Python 编程语言简洁快速、入门简单且功能强大，拥有丰富的第三方库，已经成为大数据和人工智能领域的主流编程语言。

More...

AI 知识技能速查表



Jupyter Notebook 交互式计算环境，支持运行 40+ 种编程语言，可以用来编写漂亮的交互式文档。这个教程把常用的基础功能讲解得很清楚，对新手非常友好。



正则表达式 非常强大，能匹配很多规则的文本，常用于文本提取和爬虫处理。这也是一门令人难以捉摸的语言，字母、数字和符号堆在一起，像极了“火星文”。

More...



ShowMeAI 速查表 (©2021)

获取最新版 | <http://www.showmeai.tech/>

作者 | 韩信子 @ShowMeAI

设计 | 南乔 @ShowMeAI

数据科学工具库速查表

扫码回复“数据科学”

获取最新全套速查表

AI 垂直领域工具库速查表

扫码回复“工具库”

获取最新全套速查表

编程语言速查表

扫码回复“编程语言”

获取最新全套速查表

AI 知识技能速查表

扫码回复“知识技能”

获取最新全套速查表