Keras

点击一键运行代码→ 获取更多速查表→ 使用说明书 [视频 & 图文]→ bilibili Show Me AI

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。

因其搭建神经网络时的简单易用性，Keras API 打包为 **tf.keras** 封装在 google 的 TensorFlow 中。

作者 | 韩信子 @ShowMeAI
设计 | 南 乔 @ShowMeAI
参考 | datacamp cheatsheet

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
> model.add(Dense(32, activation='relu', input_dim=100)) # 添加全连接层
> model.add(Dense(1, activation='sigmoid'))              # 添加二分类全连接层
> model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy']) # 编译模型
> model.fit(data,labels,epochs=10,batch_size=32) # 拟合数据
> predictions = model.predict(data)              # 预估数据
```


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。

Keras 数据集

```
> from keras.datasets import boston_housing, mnist, cifar10, imdb
  (x_train,y_train),(x_test,y_test) = mnist.load_data()     # 手写数字数据集
  (x_train2,y_train2),(x_test2,y_test2) = boston_housing.load_data()  # 波士顿房价数据集
  (x_train3,y_train3),(x_test3,y_test3) = cifar10.load_data() #cifar 图像分类数据集
  (x_train4,y_train4),(x_test4,y_test4) = imdb.load_data(num_words=20000) #imdb 评论数据集
  num_classes = 10
```

其它

```
> from urllib.request import urlopen

  data = np.loadtxt(urlopen("http://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data"),delimiter=",")

  X = data[:,0:8]
  y = data [:,8]
```

2. 数据预处理

序列填充

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

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)
```

独热编码

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

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)
```

3. 模型架构

顺序模型

```
> from keras.models import Sequential
> model = Sequential()
> model2 = Sequential()
> model3 = Sequential()
```

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))) #2D 池化层
> model2.add(Dropout(0.25)) # 添加随机失活层

> model2.add(Conv2D(64,(3,3), padding='same')) #2D 卷积层
> model2.add(Activation('relu')) #ReLU 激活函数
> model2.add(Conv2D(64,(3, 3))) #2D 卷积层
> model2.add(Activation('relu')) #ReLU 激活函数
> model2.add(MaxPooling2D(pool_size=(2,2))) #2D 池化层
> model2.add(Dropout(0.25)) # 添加随机失活层

> 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
> model2.add(Dense(num_classes)) # 类别数个神经元的全连接层
> model2.add(Activation('softmax')) #softmax 多分类
```

多层感知器 (MLP)

二进制分类

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

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))
```

递归神经网络 (RNN)

```
> from keras.layers import Embedding,LSTM
> model3.add(Embedding(20000,128)) # 嵌入层
> model3.add(LSTM(128,dropout=0.2,recurrent_dropout=0.2)) #LSTM 层
> model3.add(Dense(1,activation='sigmoid')) # 二分类全连接
```

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
```

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'])
```

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))
```

7. 评估模型性能

在测试集评估

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

8. 预测

预测标签与概率

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

9. 保存 / 加载模型

```
> from keras.models import load_model
  model3.save('model_file.h5')
  my_model = load_model('model_file.h5')
```

10. 模型微调

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'])
```

早停法

```
> from keras.callbacks import EarlyStopping
  early_stopping_monitor = EarlyStopping(patience=2)  # 最多等待 2 轮，如果效果不提升，就停止
  model3.fit(x_train4, y_train4, batch_size=32, epochs=15, validation_data=(x_test4,y_test4), callbacks=[early_stopping_monitor])
```