第 5 章

Keras 简介

Keras 是由谷歌公司的研究员 Francois Chollet 开发的一套高层框架，进一步封装 TensorFlow。TensorFlow 1.0 版本已经集成了 Keras 框架。目前 Keras 版本为 2.0。

5.1 Keras

Keras 是对 TensorFlow、Theano 和 CNTK 的进一步封装，抽取共性，二选一。Keras 设计采用极简主义原则，是一套高度模块化的神经网络架构库。Keras 具有方便使用的特点，如简洁的网络定义方式，常用技巧的封装。同时，Keras 又保留足够的扩展性，可以自行实现各种层(Layers)。

机器学习模型、训练、预测过程如图 5-1 所示。

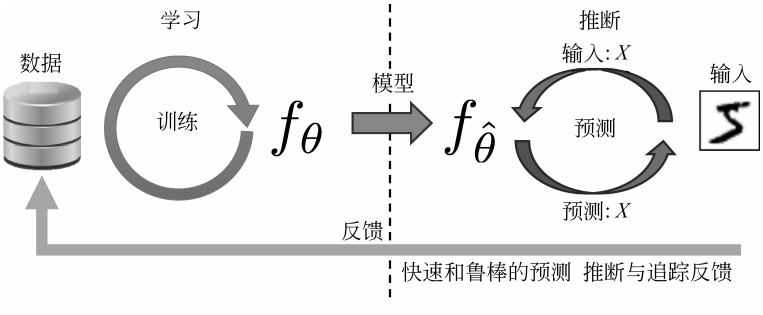

图 5-1　机器学习模型、训练、预测过程

5.2 Keras 组织结构

5.2.1 Models

Keras 核心的数据结构称为 Model，如图 5-2 所示，Model 是组织层的一种方式。最简单的 Model 类型是序列模型(Sequential Model)。序列模型是指串接很多层的管道。

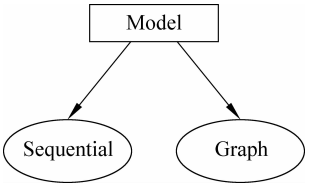

图 5-2 Keras 核心的数据结构 Model

5.2.2 Core Layers

Keras 核心层(Core Layers) 包括 Dense、Activation、Dropout、Flatten、Reshape、Permute、Repeat Vector、Lambda、Activity Regularization、Masking。

5.2.3 Layers

Keras 非核心层包括 Convolutional Layers、Pooling Layers、Locally-connected Layers、Recurrent Layers、Embedding Layers、Merge Layers、Advanced Activations Layers、Normalization Layers、Noise Layers。

5.2.4 Activations

Keras 激活函数(Activations)包括 softmax、elu、softplus、softsign、relu、tanh、sigmoid、hard_sigmoid、linear 等，函数表示式如下：

```
softmax(x), elu(x, alpha=1.0), softplus(x), softsign(x), relu(x, alpha=0.0,
max_value=None), tanh(x), sigmoid(x), hard_sigmoid(x), linear(x)
```

5.2.5 Optimizers

Keras 优化器（Optimizers）包括 SGD、RMSprop、Adagrad、Adadelta、Adam、Adamax、Nadam、Optimizer。

Nadam 采用 Nesterov 加速梯度算法。基本思想是模型的参数更新是基于之前的所有梯度（全局的信息），而不是仅基于当前的梯度（局部信息），训练收敛的速度是可以进一步提升的。

RMSprop、Adagrad、Adam 都是自适应调整学习率的算法。

5.3 Keras 实践

5.3.1 Keras 安装

首先，安装 Keras 要求的后端软件，如 Theano、TensorFlow、CNTK 等。这里以 TensorFlow 为例，可访问 https://www.tensorflow.org/install/。

选择安装 TensorFlow 的操作系统环境，如 Linux、MacOS、Windows 等。

如果需要 GPU 的支持，则不能默认安装，从网页给出的一张列表中，选择合适的 pip 包。

在 Linux 环境下，安装 Keras 命令如下：

```
$ sudo apt-get install libhdf5-10 libhdf5-dev
```

在 Windows 环境下，安装 Keras 命令如下：

```
$ pip install keras numpy scipy pyyaml h5py
```

至此，Keras 安装完毕。

如果在安装运行中，系统报缺失 xxx module 的错误，则说明需要额外的程序包支持的功能。此时，需要自己寻找缺少的包是什么，再逐一安装。

5.3.2 Keras 使用

1. 定义网络

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model =Sequential([
    Dense(32,input_dim=784),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

网络结构是以一个栈的形式给出,一层接一层。

第一层的 input_shape/input_dim 参数指定输入层的大小。例如,这里指定一个 784 维的输入。

input_dim 总是存在于第一层的参数列表,而与第一层是什么类型的网络无关。

最后一层一般是 softmax 作为输出层。此例中给出了 10 个输出,即表示该网络解决 10 类分类的问题。

2. 编译网络

给定网络的训练目标,给定网络的优化算法如下:

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

optimizer 是指使用 rmsprop 算法进行梯度下降。

loss 是指网络使用多类交叉熵作为优化目标。

metrics 是指在训练过程中,我们希望观察到准确度这个指标是如何变化的。

3. 训练网络

```
import numpy as np
data =np.random.random((1000, 784))
```

```
labels =np.random.randint(2, size=(1000, 1))

#train the model, iterating on the data in batches
#of 32 samples
model.fit(data, labels,nb_epoch=10, batch_size=32)
```

上述代码产生 1000 个符合 input_shape 的数据和标签，然后传给 model.fit，从而开始迭代训练过程。

训练过程会动态输出损失(loss)和准确度(accuracy)等参数。

4. 评估网络和数据预测

代码如下：

```
model.evaluate(self, x, y, batch_size=32, verbose=1, sample_weight=None)
model.predict(self, x, batch_size=32, verbose=0)
```

5. 保存/载入网络

将保存/载入网络结构与参数，采用 hdf5 格式保存网络参数数据，代码如下：

```
from keras.models import load_model
model.save('./model.h5')
model2 =load_model('./model.h5')
```

5.4 小结

本章介绍了 Keras 的基本概念及安装和使用。Keras 通过封装 TensorFlow，大大提高了 TensorFlow 的易用性。

参考文献

[1] Keras：Deep Learning Library for Theano and TensorFlow[EB/OL]. http://keras.io.

[2] Keras Source Code[EB/OL]. https://github.com/fchollet/keras.

[3] Keras 中文文档[EB/OL]. https://keras-cn.readthedocs.io.