

1. TensorFlow

安装 installation

要在本地计算机上安装 TensorFlow, 可以使用 pip

```
pip install tensorflow
```

安装 TensorFlow 的 GPU 版本, 需要安装一些其他软件

<https://www.tensorflow.org/install/gpu>

```
pip install tensorflow-gpu
```

For google collab users to use 2.x version, add this line

```
% tensorflow_version 2.x
```

导入 importing

```
import tensorflow as tf
```

Make sure the version is 2.x

```
print(tf.version)
```

张量 tensors

Tensors 是向量和矩阵向更高维度的推广, 每个张量都有数据类型和维度

张量类型: 变量、常量、占位符、SparseTensor

创建一个张量

```
tf.Variable("sample string", tf.string)
```

```
tf.Variable(32, tf.int16)
```

张量中涉及的维数

```
tf.rank(tf.Variable([[1, 2], [3, 4]], tf.int16))
```

Rank - 2

更改张量的形状 (shape)

```
t1 = tf.ones(original_shape)
```

```
t2 = tf.reshape(t1, new_shape)
```

2. TF 基本的机器学习算法

线性回归 Linear Regression

线性回归是机器学习的最基本形式之一, 用于预测数值。

```
lr = tf.estimator.LinearClassifier(feature_columns) # 创建模型
```

```
lr.train(train_function) # 训练模型
```

```
lr.evaluate(test_function) # 获取测试集的 model metrics
```

```
lr.predict(test_function) # 获取数据集的预测结果
```

用于分类的深度学习神经网络

在一些分类任务中, 我们的数据和标签并不存在线性关系, 使用 DNN 这种非线性结构可以更好地拟合和建模。

创建模型

```
c = tf.estimator.DNNClassifier(feature_columns, \
                               hidden_units, n_classes)
```

隐藏单元 -[# 第一隐藏层的神经元, # 第二隐藏层的神经元, ...]

训练模型

```
c.train(train_function, steps)
```

隐马尔可夫模型 HMM

隐马尔可夫模型 (Hidden Markov Model, HMM) 是典型的概率模型, 它使用概率来预测未来事件或状态。

- 在 tensorflow 的新版本中, 需要使用 `tf.compat.v1.Session()` 而不是 `tf.Session()`

输入 tensorflow probability 记为 `tfp` (HMM 为统计模型)

```
tfd = tfp.distributions
```

创建模型

```
model = tfd.HiddenMarkovModel(initial_distribution, \
                               transition_distribution, \
                               observation_distribution, \
                               num_steps)
```

i TensorFlow 是一个由谷歌开发和维护的开源机器学习平台, 用于科学计算、神经网络、图像分类、聚类、回归、强化学习、自然语言处理等。

它有两个主要组件: **graph** (计算图) 和 **session** (会话)。它定义操作的方式是为计算构建计算图 (graph) 并在会话 (session) 中执行部分计算图。

人工智能 (Artificial Intelligence)

由机器智能自动化完成本由人类完成的复杂任务

机器学习 (Machine Learning)

通过利用数据, 训练出模型, 然后使用模型预测的一种方法

神经网络 (Neural Networks)

一类特殊的模型结构, 通常由多个层次的结构构成; 应用这个多层结构建模的过程, 叫深度学习 (Deep Learning)

7. 各种超参数 hyperparameters

内置优化器 built-in optimizers

	SGD (随机梯度下降)
<code>tf.keras.optimizers</code>	Adagrad
	Adam
	RMSProp

内置损失函数 built-in loss functions

	BinaryCrossentropy (二元交叉熵)
<code>tf.keras.losses</code>	CategoricalCrossentropy (分类交叉熵)
	MeanAbsoluteError (平均绝对误差)
	MeanSquaredError (均方差)

内置指标 built-in metrics

	Accuracy (准确度)
<code>tf.keras.metrics</code>	AUC
	False Positive (假阳)
	Precision (精度)

3. Keras

Keras 是一种高级神经网络 API，用 Python 编写，并合并并在 tensorflow 的高版本中。

tensorflow.keras 对用户友好、可模块化、可扩展性，能够轻松快速地创建原型，支持各种类型的网络 (CNN、RNN、transformer 等)，可以在 CPU 和 GPU 上无缝运行。

导入 Keras

```
from tensorflow import keras
```

Keras 模型的基本工作流程

① 定义模型 define a model

② 编译模型 compile a model

③ 训练模型 fit a model

④ 评估模型 evaluate a model

⑤ 预测 make predictions

创建模型后

- 使用 `model.compile()` 编译模型 (计算损失函数值和评估准则值)
- 使用 `model.fit()` 训练模型
- 使用 `model.evaluate()` 对测试集的 loss 和评估准则进行计算
- 使用 `model.predict()` 应用模型进行预测

① 定义模型 define a model

```
model = keras.Model(inputs, outputs, ...)
model.summary()
# Groups a linear stack of layers into a model
keras.Sequential(layers, ...)
# For multi - GPU data parallelism
keras.utils.multi_gpu_model(model, gpus, ...)
```

② 编译模型 compile a model

为训练配置模型

```
model.compile(optimizer, loss, metrics, loss_weights, \
              weighted_metrics, ...) # 参数调整
```

③ 训练模型 fit a model

固定迭代次数的训练模型

```
model.fit(x, y, batch_size, epochs, verbose, callbacks, ...)
model.fit_generator() # 在生成器逐批生成的数据上拟合模型
model.train_on_batch() # 在一批特定训练数据上更新梯度
```

④ 评估模型 evaluate a model

返回测试模式下模型的损失值和度量值

```
model.evaluate(x, y, batch_size, steps, ...)
# 在数据生成器上计算模型
model.evaluate_generator(generator, ...)
```

⑤ 预测 make predictions

```
model.predict() # 从模型生成预测
model.predict_on_batch(x) # 返回单个批次样本的预测
# 从数据生成器生成输入样本的预测
model.predict_generator(generator, steps, ...)
model.predict_step(data) # 一个推理步骤的逻辑
```

4. 可选高级功能

输出模型信息

```
model.summary()
```

在图层堆栈顶部添加图层

```
model.add(layer)
```

使用名称 / 索引

```
model.get_layer()
```

保存模型并在将来随时重新加载

```
model.save()
keras.models.load_model()
```

要使用的各种数据集

```
keras.datasets
```

例如，下方模型包含 60000 个 32x32 彩色图像，以及 10 个不同日常对象的 6000 个图像

```
keras.datasets.cifar10.load_data()
```

从分类分布中提取样本

```
tf.random.categorical()
```

通过指定要加载的确切文件来加载任何检查点

```
tf.train.load_checkpoint()
```



TensorFlow2 建模应用速查表

获取最新版 | <http://www.showmeai.tech/>

作者 | 韩信子 @ShowMeAI

设计 | 南乔 @ShowMeAI

参考 | Jayant Uppal

扫码回复“工具库”

下载最新全套资料



4.1 keras.layers 不同模型层

① 卷积层 convolutional layers

尺寸 (1D、2D、3D) 的选择取决于输入的尺寸。

Conv1D 通常用于与语音类似的输入信号, Conv2D

用于图像, Conv3D 用于每个时间跨度都有一帧的视频

`layers.Conv1D()`

`layers.Conv2D()`

`layers.Conv3D()`

转置卷积, 即与正常卷积方向相反

`keras.layers.Conv1DTranspose()`

`keras.layers.ZeroPadding1D()` # 零填充层

`keras.layers.Cropping1D()` # 裁剪层

`keras.layers.UpSampling1D()` # 上采样层

② 池化层 pooling layers

最大池化层

`keras.layers.MaxPool1D()`

`keras.layers.MaxPool2D()`

`keras.layers.MaxPool3D()`

平均池化层

`keras.layers.AveragePooling1D()`

全局平均池化操作和全局最大池化操作

`keras.layers.GlobalAveragePooling1D()`

`keras.layers.GlobalMaxPool2D()`

③ 激活层 activation layers

`keras.layers.Activation('relu')` # 激活函数

不同版本的 ReLU 激活函数

`keras.layers.ReLU()`

`keras.layers.LeakyReLU()`

`keras.layers.PReLU()`

④ dropout 层

`keras.layers.Dropout()` # Dropout 随机失活

Dropout 的 1D、2D、3D 版本

`Spatial 1D version of Dropout`

⑤ 嵌入层 embedding layers

将索引转换为固定大小的 embedding 密集向量

`keras.layers.Embedding()`

⑥ RNN 层 recurrent layers

`keras.layers.LSTM()` # Long Short Term Memory

`keras.layers.RNN()`

`keras.layers.GRU()` # Gated Recurrent Unit

⑦ 展平层 flatten layers

`keras.layers.Flatten()` # 展平输入, 不影响批次大小

⑧ 稠密层 dense layers

稠密层 / 全连接层 dense layers

`keras.layers.Dense(32, activation='relu')`

⑨ 激活层 activation layers

局部连接层工作方式与 Conv 层类似, 不同之处在于权重是非共享的, 即在输入的每个不同面上应用一组不同的过滤器

`keras.layers.LocallyConnected1D`

`keras.layers.LocallyConnected2D`

4.2 回调 callbacks

回调是在训练过程中的一组函数, 可以在训练期间使用回调获取模型内部状态和统计信息的视图

`keras.callbacks`

`keras.callbacks.ModelCheckpoint()` # 回调以某种频率保存 Keras 模型或模型权重

`keras.callbacks.EarlyStopping()` # 当监控指标停止改善时停止训练

4.3 预处理 pre-processing

① 图像预处理 image preprocessing

`keras.preprocessing.image` # 图像数据实时数据扩充工具集

`keras.preprocessing.image.load_img()` # 加载图像, 将图像加载到数组, 或将数组保存为图像

`keras.preprocessing.image.img_to_array()`

`keras.preprocessing.image.array_to_img()`

`keras.preprocessing.image.ImageGenerator` # 通过实时数据增强生成批量张量图像数据

② 文本预处理 text preprocessing

`keras.preprocessing.text.Tokenizer()` # Text tokenization

`keras.preprocessing.text.one_hot()` # One-hot 将文本编码到单词索引列表中

③ 序列预处理 sequence preprocessing

`keras.preprocessing.sequence.pad_sequences` # 将序列填充到相同长度

`keras.preprocessing.sequence.skipgrams()` # 生成 skipgram 字对

4.4 预训练模型 pre-trained models

如果数据集与原始数据集分布没有显著差异, 则预训练模型的 Transfer learning 和 fine-tuning 可节省时间。

`tensorflow.keras` 应用程序是深度学习模型, 可与预先训练的权重一起使用, 用于预测、特征提取和微调。

`keras.applications`

举例, 该模型在 140 万张图像上进行训练, 有 1000 个不同的类

`tensorflow.keras.applications.MobileNetV2()`

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。



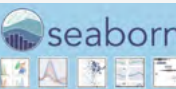
Pandas 是基于 NumPy 创建的 Python 库，为 Python 提供了易于使用的的数据结构和数据分析工具。使用以下语句导入：

```
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
```

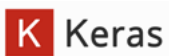


PySpark 是 Spark 的 Python API，允许 Python 调用 Spark 编程模型。Spark SQL 是 Apache Spark 处理结构化数据模块。

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