tf.data 简介

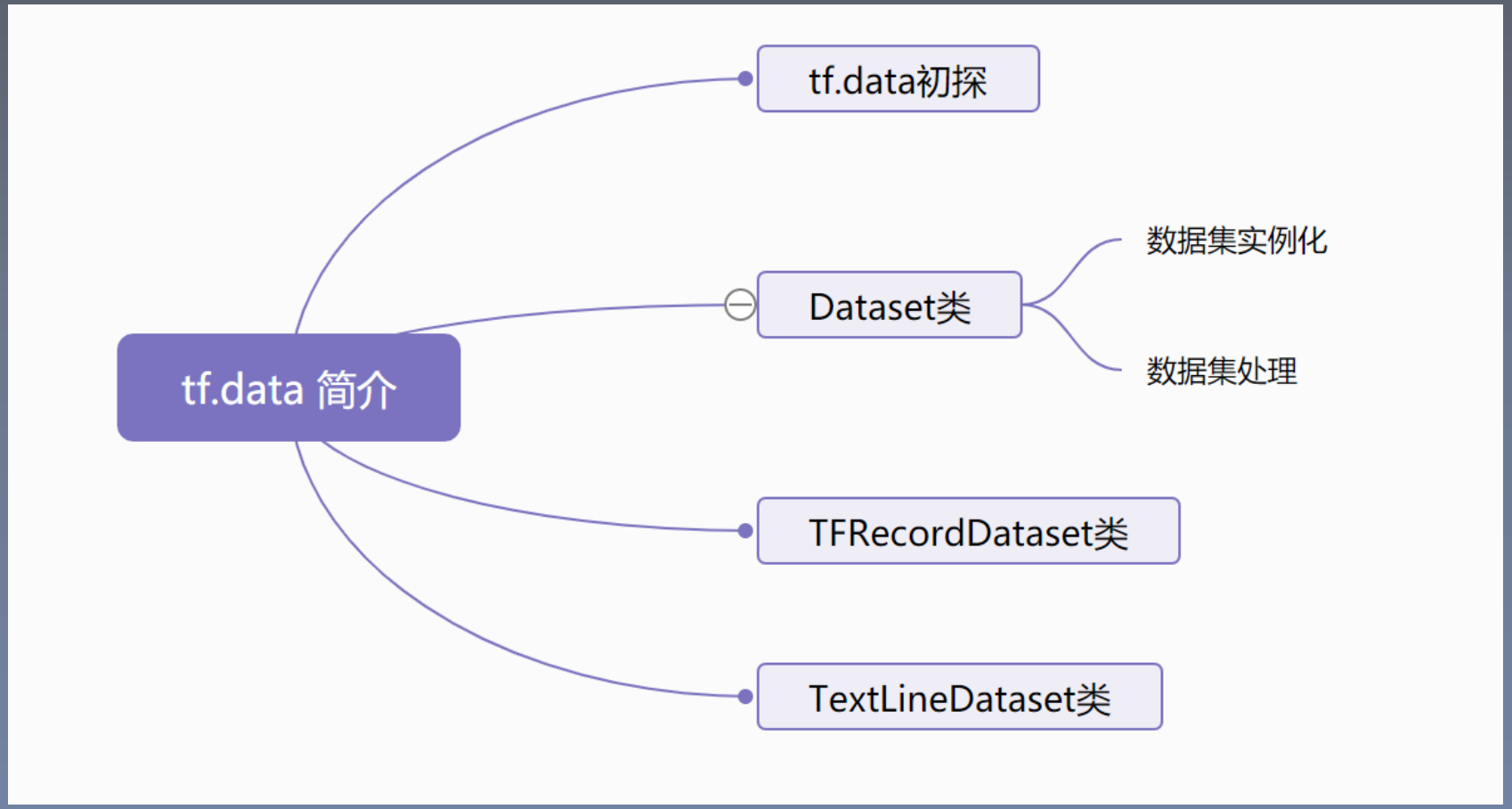
导师: GAUSS

深度之眼
deepshare.net

目录

知识树

Knowledge tree



深度之眼
deepshare.net

tf.data初探

tf.data简介

面对一堆格式不一的原始数据文件？

读入程序的过程往往十分繁琐？

运行的效率上不尽如人意？

TensorFlow 提供了 **tf.data** 这一模块，包括了一套灵活的数据集构建 API，能够帮助我们**快速、高效地构建数据输入的流水线**，尤其适用于**数据量巨大的场景**。

tf.data简介

- tf.data.Dataset类
- tf.data.TFRecordDataset类
- tf.data.TextLineDataset类

更多参考：

https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data

深度之眼
deepshare.net

Dataset类

tf.data.Dataset简介

tf.data 的核心是 **tf.data.Dataset** 类，提供了对数据集的高层封装。tf.data.Dataset 由一系列的**可迭代访问的元素（element）组成**，每个元素包含一个或多个张量。Dataset可以看作是相同类型“元素”的有序列表。

比如说，对于一个由图像组成的数据集，每个元素可以是一个形状为 **长×宽×通道数** 的图片张量，也可以是由图片张量和图片标签张量组成的元组（Tuple）。

更多阅读：

https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset

tf.data.Dataset创建数据集

tf.data.Dataset 类创建数据集，对数据集实例化。最常用的如：

tf.data.Dataset.from_tensors() ： 创建Dataset对象， 合并输入并返回具有单个元素的数据集。

tf.data.Dataset.from_tensor_slices() ： 创建一个Dataset对象，输入可以是一个或者多个 tensor，若是多个 tensor，需要以元组或者字典等形式组装起来。

tf.data.Dataset. from_generator() ： 迭代生成所需的数据集，一般数据量较大时使用。

注：Dataset可以看作是相同类型“元素”的有序列表。在实际使用时，单个“元素”可以是向量，也可以是字符串、图片，甚至是tuple或者dict。

from_tensors和from_tensor_slices区别

from_tensors() 函数会把传入的tensor当做一个元素，但是from_tensor_slices() 会把传入的tensor**除开第0维之后的大小**当做元素个数。

```
dataset1=tf.data.Dataset.from_tensors(np.zeros(shape=(10,5,2),dtype=np.float32))
for line in dataset1:
    print(line.shape)
    break
```

```
(10, 5, 2)
```

```
dataset2=tf.data.Dataset.from_tensor_slices(np.zeros(shape=(10,5,2),dtype=np.float32))
for line in dataset2:
    print(line.shape)
    break
```

```
(5, 2)
```

```
dataset3=tf.data.Dataset.from_tensors({"a":np.zeros(shape=(10,5,2),dtype=np.float32),"b":np.zeros(shape=(10,5,2),dtype=np.float32)})
for line in dataset3:
    print(line['a'].shape,line['b'].shape)
    break
```

```
(10, 5, 2) (10, 5, 2)
```

```
dataset4 = tf.data.Dataset.from_tensor_slices({"a":np.zeros(shape=(10,5,2),dtype=np.float32),"b":np.zeros(shape=(10,5,2),dtype=np.float32)})
for line in dataset4:
    print(line['a'].shape,line['b'].shape)
    break
```

```
(5, 2) (5, 2)
```

tf.data.Dataset示例

Numpy数据：

假设有一个feature数组和相应的标签数组，将两个数组作为元组传递给 tf.data.Dataset.from_tensor_slices 以创建 tf.data.Dataset 。

```
mnist = np.load("mnist.npz")

x_train, y_train = mnist['x_train'],mnist['y_train']
```

```
x_train.shape,y_train.shape
```

```
((60000, 28, 28), (60000,))
```

```
x_train = np.expand_dims(x_train, axis=-1)
```

```
x_train.shape
```

```
(60000, 28, 28, 1)
```

```
mnist_dataset = tf.data.Dataset.from_tensor_slices((x_train, y_train))
```

tf.data.Dataset示例

Pandas数据：

使用 tf.data.Dataset.from_tensor_slices 从 pandas dataframe 中读取数值。

```
import pandas as pd
df = pd.read_csv('heart.csv')
```

```
df.head()
```

```
df.dtypes
```

```
df['thal'] = pd.Categorical(df['thal'])
df['thal'] = df.thal.cat.codes
```

```
target = df.pop('target')
```

```
dataset = tf.data.Dataset.from_tensor_slices((df.values, target.values))
```

```
for feat, targ in dataset.take(5):
    print ('Features: {}, Target: {}'.format(feat, targ))
```

tf.data.Dataset示例


```
img_gen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255, rotation_range=20)
```

```
flowers = './flower_photos/flower_photos/'
```

```
def Gen():
    gen = img_gen.flow_from_directory(flowers)
    for (x,y) in gen:
        yield (x,y)
```

```
ds = tf.data.Dataset.from_generator(
    Gen,
    output_types=(tf.float32, tf.float32)
#     output_shapes=([32,256,256,3], [32,5])
)
```

```
for image,label in ds:
    print(image.shape,label.shape)
    break
```

```
Found 3670 images belonging to 5 classes.
(32, 256, 256, 3) (32, 5)
```

tf.data.Dataset数据集处理

tf.data.Dataset 类为我们提供了多种数据集预处理方法。最常用的如：

tf.data.Dataset.map(f) ： 对数据集中的每个元素应用函数 f ，得到一个新的数据集（这部分往往结合 tf.io 进行读写和解码文件， tf.image 进行图像处理）；

tf.data.Dataset.shuffle(buffer_size) ： 将数据集打乱（设定一个固定大小的缓冲区（Buffer），取出前 buffer_size 个元素放入，并从缓冲区中随机采样，采样后的数据用后续数据替换）；

tf.data.Dataset.batch(batch_size) ： 将数据集分成批次，即对每 batch_size 个元素，使用 tf.stack() 在第 0 维合并，成为一个元素；

深度之眼
deepshare.net

TFRecordDataset类

tf.data.TFRecordDataset简介

对于特别巨大而无法完整载入内存的数据集，我们可以先将数据集处理为 **TFRecord** 格式，然后使用 tf.data.TFRecordDataset() 进行载入。

TFRecord 是 TensorFlow 中的数据集存储格式。当我们将数据集整理成 TFRecord 格式后，TensorFlow 就可以高效地读取和处理这些数据集，从而帮助我们更高效地进行大规模的模型训练。

先在notebook中看看怎么使用，后面详细介绍！

TFRecordDataset参数详解


```
tf.data.TFRecordDataset(
    filenames, compression_type=None, buffer_size=None, num_parallel_reads=None
)
```

filenames：tf.string张量，值为一个或多个文件名。

compression_type：tf.string标量，值为 “” （不压缩）、"ZLIB"或"GZIP"之一。

buffer_size：tf.int64标量，表示读取缓冲区中的字节数。

num_parallel_reads：tf.int64标量，表示要并行读取的文件数。

TFRecordDataset示例

TFRecordDataset使用方法示例：

```
feature_description = { # 定义Feature结构，告诉解码器每个Feature的类型是什么
    'image': tf.io.FixedLenFeature([], tf.string),
    'label': tf.io.FixedLenFeature([], tf.int64),
}

def _parse_example(example_string): # 将 TFRecord 文件中的每一个序列化的 tf.train.Example 解码
    feature_dict = tf.io.parse_single_example(example_string, feature_description)
    feature_dict['image'] = tf.io.decode_jpeg(feature_dict['image'])    # 解码JPEG图片
    feature_dict['image'] = tf.image.resize(feature_dict['image'], [256, 256]) / 255.0
    return feature_dict['image'], feature_dict['label']

batch_size = 32

train_dataset = tf.data.TFRecordDataset("train.tfrecords")    # 读取 TFRecord 文件
train_dataset = train_dataset.map(_parse_example)
train_dataset = train_dataset.shuffle(buffer_size=23000)
train_dataset = train_dataset.batch(batch_size)
train_dataset = train_dataset.prefetch(tf.data.experimental.AUTOTUNE)
```

深度之眼
deepshare.net

TextLineDataset类

tf.data.TextLineDataset简介

tf.data.TextLineDataset 提供了一种从一个或多个文本文件中提取行的简单方法。

给定一个或多个文件名，TextLineDataset 会为这些文件的每行生成一个字符串值元素。像 TFRecordDataset 一样，TextLineDataset 将 filenames 视为 tf.Tensor。

类中保存的元素：**文中一行，就是一个元素**，是string类型的tensor。

tf.data.TextLineDataset参数详解


```
tf.data.TextLineDataset(
    filenames, compression_type=None, buffer_size=None, num_parallel_reads=None
)
```

filenames：tf.string张量，值为一个或多个文件名。

compression_type：tf.string标量，值为 “” （不压缩）、"ZLIB"或"GZIP"之一。

buffer_size：tf.int64标量，表示读取缓冲区中的字节数。

num_parallel_reads：tf.int64标量，表示要并行读取的文件数。

tf.data.TextLineDataset示例

使用方法：

```
titanic_lines = tf.data.TextLineDataset(['train.csv','eval.csv'])
```

```
def data_func(line):
    line = tf.strings.split(line, sep = ",")
    return line
```

```
titanic_data = titanic_lines.skip(1).map(data_func)
```

```
for line in titanic_data:
    print(line)
    break
```

```
tf.Tensor(
[b'0' b'male' b'22.0' b'1' b'0' b'7.25' b'Third' b'unknown' b'Southampton'
 b'n'], shape=(10,), dtype=string)
```

本节小结

Summary

tf.data 构建 TensorFlow 输入管道	tf.data简介	简单介绍
	Dataset类	可以读取numpy、pandas、文件
	TFRecordDataset类	解决大文件数据集的问题
	TextLineDataset类	提供了一种从一个或多个文本文件中提取行的简单方法

结语

我 说

GAUSS老师个人公众号，主要分享NLP、推荐、比赛实战相关知识！

深度之眼
deepshare.net

联系我们：

电话：18001992849

邮箱：service@deepshare.net

Q Q： 2677693114

公众号

客服微信