

目录	1
----	---

目录

1 简介	2
1.1 推荐配置	2
1.2 计算图	2
1.3 从文件读取数据	4
2 回归	5
2.1 损失函数	5
2.2 正则化	5
2.3 优化器	5
3 神经网络	6
3.1 激活函数	6
3.2 构建网络	6
3.3 卷积神经网络	7

Tensorflow

LukeAlanLee

2019 年 5 月 10 日

1 简介

1.1 推荐配置

- windows: 10-1809
- python:3.7.0
- cuda:10.0
- cudnn:7.4.2
- tensorflow-gpu:1.31.1

1.2 计算图

- 定义常量: `tf.constant()`
- 定义会话: `sess=tf.Session()`, 关闭会话: `sess.close()`, 可以使用with 管理会话(`with tf.Session() as sess`)
- 运行会话: `sess.run()`
- 默认会话: `tf.InteractiveSession()`, 此时可以使用`eval()`直接运行张量而不用显示调用会话
- 指定运行设备: `tf.device()`, CPU/GPU
- 张量类型:
 - 常量:`tf.constant()`
 - 变量:`tf.Variable()`

占位符:tf.placeholder(),通常和feed_dict一起使用来输入数据

举例: tf.zeros(),tf.zeros_like(),tf.ones(),tf.ones_like(), tf.eye(n)

等差数列: tf.linspace(start,stop,nums),tf.range(start,limit,delta)

正态分布: tf.random_normal([rows,cols],mean,stddev,seed)

截尾正态分布: tf.truncated_normal([rows,cols],mean,stddev,seed)

伽马分布: tf.random_uniform([rows,cols],maxval,seed)

裁剪张量: tf.random_crop(t_radam,[rows,cols],seed)

随机重排: tf.random_shuffle(t_radam)沿t_radam第一维随机排列张量

设置随机种子: tf.set_random_seed()相同种子保证多次运行获得相同随机数(种子只能设为整数)

变量别名: tf.Variable(tf.zeros[100],name='biases')

初始化变量的方法: 使用常量、已定义的变量都可以初始化新的变量;必须初始化所有变量tf.global_variables_initializer()

数据类型转换: tf.cast(var,dtype)

使用Saver类来保存变量: saver=tf.Saver()

定义placeholder并使用feed_dict输入: 举例如下:

```
1 x=tf.placeholder("float")
2 y=2*x
3 data=tf.random_uniform([4,5],10)
4 with tf.Session() as sess:
5     x_data=sess.run(data)
6     print(sess.run(y,feed_dict={x:x_data}))
```

tfTest.py

tensorflow数据类型	
数据类型	tensorflow类型
DT_FLOAT	tf.float32
DT_DOUBLE	tf.float64
DT_int8	tf.int8
DT_UINT8	tf.uint8
DT_STRING	tf.string
DT_BOOL	tf.bool
DT_COMPLEX64	tf.complex64
DT_QINT32	tf.qint32

- 运算

加法运算: `tf.add()`

矩阵乘法: `tf.matmul()`

按元素相除 *

矩阵乘标量 `tf.scalar_mul(2,A)`

按元素相除 `tf.div()`

按元素取余 `tf.mod()`

整数张量除法 `tf.truediv(a,b)`, 先将整数转为浮点类, 然后再执行按位除

- 数据流图-TensorBoard

捕获时间序列变化: `tf.summary.scalar()`

捕获输出分布: `tf.summary.histogram()`

捕获所有摘要: `tf.merge_all_summaries()`

保存摘要: `tf.summary.FileWriter('logdir',sess.graph)`

tensorBoard: `tensorboard -logdir='logdir'`

1.3 从文件读取数据

1. 创建文件名列表:使用字符串张量保存或使用
`files=tf.train.match_filenames_once('filename')`
2. 创建文件队列: 之后使用`tf.train.string_input_producer(files)`创建文件队列
3. 创建由换行符分隔的文件行的读取器: `tf.TextLineReader()`
4. 从tensor列表中取出tensor放入文件名队列:
`input_queue=tf.train.slice_input_producer()`
5. 从文件名队列中提取tensor放入文件队列: `image_batch, label_batch = tf.train.batch(input_queue, batch_size=batch_size, num_threads=2, capacity=64)`
6. 线程协调器: `coord=tf.train.Coordinator()` ,用来管理之后在Session中启动的所有线程;

7. 线程协调器方法:

- `coord.should_stop()`来查询是否应该终止所有线程,当文件队列(queue)读取完毕,抛出一个`OutOfRangeError`的异常,此时应该结束左右线程
- 使用`coord.request_stop()`来发出终止所有线程的命令
- 使用`coord.join(threads)`把线程加入主线程,等待threads结束

8. 启动线程: `tf.train.start_queue_runners(coord=coord)`,由多个或单个线程(使用`coord`作为线程管理器),按照设定规则,把文件读入`Filename Queue`中.

9. Example

2 回归

2.1 损失函数

1. 平方损失: `tf.square()`
2. 滑动平均: `tf.reduce(tf.square())`
3. 交叉熵:

```

1          entropy=tf.nn.softmax_cross_entropy_with_logits()
2          loss=tf.reduce_mean(entropy)

```

2.2 正则化

1. L1 `tf.reduce_sum()`
2. L2 `tf.nn.l2_loss()`

2.3 优化器

1. **GD**: `tf.train.GradientDescentOptimizer(learning_rate)`
2. **momentum**: `tf.train.MomentumOptimizer()`

3. **Adadelta**: `tf.train.AdadeltaOptimizer()`
4. **RMSprop**: `tf.train.RMSpropOptimizer()`
5. **Adagrad**: `tf.train.AdagradOptimizer()`
6. **AdagradDA**: `tf.train.AdagradDAOptimizer()`
7. **Ftrl**: `tf.train.FtrlOptimizer()`
8. **ProximalGD**: `tf.train.ProximalGradientDescentOptimizer()`
9. **ProximalAdagrad**: `tf.train.ProximalAdagradOptimizer()`
10. 设置指数衰减学习率`tf.train.exponential_decay()`

```

1 global_step=tf.Variable(0,trainable=False)
2 initial_learning_rate= 0.2
3 learning_rate=tf.train.exponential_decay(initial_learning_rate
4     ,global_step,decay_steps=100000,decay_rate=0.95)
5 '''pass this learning rate to optimizer as before'''

```

exponentialDecayExample.py

3 神经网络

3.1 激活函数

3.2 构建网络

1. threshold:

```

1 def threshold(x):
2     cond = tf.less(x, tf.zeros(tf.shape(x), dtype = x.dtype))
3     out = tf.where(cond, tf.zeros(tf.shape(x)), tf.ones(tf.shape(x)))
4     return out

```

2. Sigmoid:

```

1 out = tf.sigmoid(h)

```

3. tanh:

```
1 out = tf.tanh(h)
```

4. ReLu:

```
1 out = tf.nn.relu(h)
```

5. Softmax:

```
1  $y_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$   
2 out = tf.nn.softmax(h)
```

3.3 卷积神经网络

1. 添加卷积层:

```
1 tf.nn.conv2d(input,filter, strides,padding,  
2             use_cudnn_on_gpu=None,data_format=None,name=None)
```

2. 池化层

- 最大池化

```
1 tf.nn.max_pool(value,kszie, strides,padding,data_format='NHWC',name=None)
```

```
1 out = tf.nn.relu(h)
```

3.4 图片相关

1. tf.gfile.FastGFile(path,decodestyle)

函数功能： 实现对图片的读取。

函数参数： (1)path: 图片所在路径(2)decodestyle:图片的解码方式。(‘r’:UTF-8编码; ‘rb’:非UTF-8编码)