第四章 深度学习编程框架

1 深度学习编程框架概念

必须性：

- 深度学习算法具有多层结构
- 基本操作存在大量共性运算
- 可以针对性的优化

深度学习框架：将深度学习算法中的基本操作封装成一系列组件

2.TensorFlow 概述

基本历史

3.TensorFlow编程模型及基本概念

1.命令式编程与声明式编程

命令式：关注程序执行的具体步骤，优化困难

声明式：不指定具体的实现步骤，优化容易

TensorFlow的基本概念

计算图-->有向图 （一组节点和边） 对应了神经网络结构 支持通过多种高级语言构建

会话（session）

-->只是描述了计算过程

```
tf.Session(target='',graph=None,config=None)
```

`sess.run()`

`sess.close()`和`with`

求解张量值方法

1. 会话中使用run()函数
2. `tensor.eval()`

`session.run与tensor.eval区别：`

1. tensor.eval()输入参数：

参数名	说明
feed_dict	指定需要填充的张量或者操作
session	指定求解此tensor或操作的会话

tensor.eval()函数使用前，均需要显式指定求解张量或操作的会话，如用with语句定义会话

处理单个**tensor**时，**tensor.eval()**与**session.run()**等价

session.run可以一次进行多个tensor或操作的计算

- 操作（operation）每个计算节点代表一个操作
- 张量（tensor）是计算图上的数据载体

tensor.eval()每次只能计算一个

变量（**variable**）

为计算图中的一个有状态节点。

创建变量：将一个tensor传递给Variable()构造函数，创建时需指定变量的形状与数据类型

- 采用`tf.Variable()`直接定义
- 使用`TensorFlow`内置的函数来定义变量的初值，可以是常量或随机值
- 用其他变量初始值来定义新的

TensorFlow操作	说明
tf.zeros()	产生一个全为0的张量
tf.ones()	产生一个全为1的张量
tf.random_normal()	产生正态分布的随机数
tf.truncated_normal()	从截断的正态分布中输出随机数
tf.random_uniform()	产生满足平均分布的随机数
tf.random_gamma()	产生满足Gamma分布的随机数
tf.fill()	产生一个全为给定值的张量
tf.constant()	产生常量
variable.initialized_value()	产生一个变量的初值

初始化变量

创建完变量后，还需要初始化

最简单的初始化方法：`tf.global_variables_initializer()`

更新变量

变量更新可以通过优化器自动更新完成，也可以通过自定义方法强制赋值更新

更新函数	说明
强制赋值更新	
tf.assign()	更新变量的值
tf. assign_add()	加法赋值
tf. assign_sub()	减法赋值
自动更新	
tf.train.**Optimizer	使用多种优化方法自动更新参数

占位符(placeholder)

计算图表达的是拓扑结构，在向计算图填充数据前，计算图并没有真正执行运算

用占位符构建计算图中的样本输入节点，不需要实际分配数据，占位符本身没有初值，只是在程序中分配了内存

`tf.placeholder()`输入参数：

输入参数	说明
name	在计算图中的名字
dtype	填充数据的数据类型
shape	填充数据的shape值

使用时需与feed_dict参数配合，用feed_dict提交数据

队列(Queue)

queue机制：通过多线程将读取数据与计算数据分开（有状态机制）

加开速度训练：采用多个线程读取数据，一个线程消耗数据

queue包含入队、出队操作

FIFOQueue

TensorFlow 1.x

静态图，方便对整个计算图做全局优化，性能高，调试困难，影响开发效率

TensorFlow 2.x

动态图，调试简单，更适合快速开发；但运行效率低于静态图

数据位宽与算法精度

每层数据都有其保持网络收敛的最低位宽需求

每层数据的位宽需求与数据分布之间存在关系

训练时不需要高位宽

tensor的**device**属性

tf.device(device_name)指定计算出此tensor所用的设备名

TensorFlow不区分CPU，所有的CPU均使用 /cpu:0作为设备名

用/gpu:n表示第几个GPU设备，用/mlu:n表示第n个深度学习处理器

Tensor常用的**op**

tf.shape
tf.to_double(x,name='ToDouble')（float,int32,int64）
tf.cast(x,dtype)
tf.reshape(tensor,shape)
tf.slice(input,begin,size)
tf.split(value,num_or_size_splits,axis)
tf.concat(values,axis)

总结：

- 计算图：对应神经网络结构
- 操作：对应神经网络具体计算
- 张量：对应神经网络中的数据
- 会话：执行神经网络真正的训练和预测
- 变量：对应神经网络参数
- 占位符：对应神经网络的训练或预测输入
- 队列：对应神经网络训练样本的多线程并行处理

5.基于TensorFlow的训练及预测实现

tf.nn模块

操作	说明
tf.nn.conv2d(input, filter, strides, padding)	在给定的 input与 filter下计算卷积
tf.nn.depthwise_conv2d(input, filter, strides, padding)	卷积核能相互独立的在自己的通道上面进行卷积操作。
tf.nn.separable_conv2d(input, depthwise_filter, pointwise_filter, strides, padding)	在纵深卷积 depthwise filter 之后进行逐点卷积 separable filter
tf.nn,bias_add(value,bias)	对输入加上偏置

激活函数

操作	说明
tf.nn.relu(features)	计算relu函数
tf.nn.elu(features)	计算elu函数
tf.nn.dropout(x, keep_prob)	计算dropout，keep_prob为keep概率,
tf.sigmoid(x)	计算sigmoid函数
tf.tanh(x)	计算tanh函数

池化函数和损失函数

操作	说明
tf.nn.avg_pool(value, ksize, strides, padding)	平均方式池化
tf.nn.max_pool(value, ksize, strides, padding)	最大值方法池化
tf.nn.max_pool_with_argmax(input, ksize, strides,padding)	返回一个二维元组(output,argmax), 最大值pooling，返回最大值及其相应的索引
tf.nn.l2_loss(t)	output = sum(t ** 2) / 2

构建模型

1. 加载数据
2. 定义损失函数
3. 创建优化器
4. 定义模型训练方法

训练模型

1.加载数据

注入（**feeding**）： 利用feed_dict直接传递输入数据

预取（**pre_load**）: 利用Const和Variable直接读取输入数据

基于队列 **API**：基于队列相关的API来构建输入流水线(Pipeline)

tf.data API: 利用tf.data API来构建输入流水线

2.定义损失函数

TensorFlow内置4个损失函数

1. softmax交叉熵
2. 加入稀疏的softmax交叉熵
3. sigmoid交叉熵
4. 带参数的sigmoid交叉熵

3.创建优化器

梯度下降优化器

Adam算法优化器

4.定义模型训练方法

一般采用最小损失函数（minimize）方法

常用的训练操作

操作	功能
tf.train.Optimizer.minimize(loss, global_step=None, var_list=None)	使用最小化损失函数的方法来训练模型。执行该操作时会内部依次调用compute_gradients和apply_gradients操作
tf.train.Optimizer.compute_gradients(loss,var_list=None)	对var_list中列出的模型参数计算梯度，返回（梯度，模型参数）组成的列表
tf.train.Optimizer.apply_gradients(grads_and_vars)	将计算出的梯度更新到模型参数上，返回更新参数的操作

对梯度的处理

梯度爆炸或梯度消失

Solution:1.减少学习率 2.梯度裁剪

- TensorFlow中内置的梯度处理功能

方法	功能
tf.clip_by_value(t,clip_value_min,clip_value_max)	将梯度t裁剪到[clip_value_min,clip_value_max]区间
tf.clip_by_norm(t,clip_norm)	对梯度t的L2范式进行裁剪，clip_norm为裁剪阈值
tf.clip_by_average_norm(t, clip_norm)	对梯度t的平均L2范式进行裁剪，clip_norm为裁剪阈值
tf.clip_by_global_norm(t_list, clip_norm)	对梯度t_list进行全局规范化加和裁剪，clip_norm为裁剪阈值
tf.global_norm(t_list)	计算t_list中所有梯度的全局范式

5.模型保存

在模型训练过程中，使用tf.train.Saver()来保存模型中的所有变量