Paddle Paddle 源码阅读报告

组长：叶钰楠

组员：唐作其，李佳晖，肖少宁，李星辰，张凤达

1. 代码总结与梳理：

要求：文字分条目列出，描述不少于200字，模块不少于7个

|  |  |
| --- | --- |
| **代码整体认识描述** | **代码模块或主要组成总结** |
| nn 模块定义了常用的神经网络的层和操作，其中包括全连接层fc()、卷积层conv2d()、激活函数relu()以及softmax()等等，另外常用操作包括矩阵乘法matmul()、求均值mean()、转置transpose()以及切片slice()等等。  learning\_rate\_scheduler模块可以调整训练过程中的学习速率。  **metric\_op**提供Accuracy和Auc两种operator，在实际使用中需要通过输入的参数，先建立LayerHelper，后在LayerHelper中append\_op。  使用RNN时，首先定义cell类，再通过rnn函数进行逐步计算。  使用BeamSearch时，首先定义BeamSearchDecoder，再通过dynamic\_decode逐步计算。  Helper用来定义decoding策略，dynamic\_rnn仅支持LoDTensor作为输入，返回LoDTensor，用于对一层的rnn\_unit一步步计算。  ParamAttr定义了参数属性相关的信息，例如参数的名称、初始化方式、学习率、正则化规则、是否需要训练、梯度裁剪方式、是否做模型平均等属性，内含ParamAttr与WeightNormParamAttr两种类，后者额外提供了归一化选项，可以进行参数归一化  executor定义了运行设备（执行器）的相关信息，以及执行范围（scope）。  paddle.fluid.io定义数据接口相关的类与方法，如数据，模型，变量的读取与储存等  unique\_name内含'generate', 'switch', 'guard'三种方法，其中generate生成独有命名，switch和guard改变命名空间。  Initializer是数据初始化模块，主要为数据的初始化提供多种方式。  Regularizer在模型优化时为参数惩罚项提供多种惩罚函数。  Optimizer为模型优化提供多种优化器。  Compiler将输入的Program或Graph进行转换和优化。  Framework模块主要包括：  （1）一些基础数据结构的定义；  （2）定义了Paddle的执行系统环境、CPU和GPU支持情况；  （3）执行流程，使用一种编程式的执行流程，分为编译和运行两部分。  Core模块主要包括：  （1）对导入系统路径异常的判断及处理；  （2）针对不种操作系统平台（Linux、MasOS和Windows）对AVX（Advanced Vector Extensions）支持情况；  （3）设置Paddle 库的路径。 | 1. paddle.fluid.layers.nn |
| 1. paddle.fluid.layers.learning\_rate\_scheduler |
| 1. paddle.fluid.layers.loss |
| 1. paddle.fluid.layers.metric\_op |
| 1. paddle.fluid.layers.rnn |
| 1. paddle.fluid.layers.io |
| 1. paddle.fluid.layers.ops |
| 1. paddle.fluid.layers.tensor |
| 1. paddle.fluid.layers.sequence\_lod |
| 1. paddle.fluid.param\_attr |
| 1. paddle.fluid.executor |
| 1. paddle.fluid.io |
| 1. paddle.fluid.unique\_name |
| 1. paddle.fluid.compiler   paddle.fluid.compiler将输入的Program或Graph通过配置BuildStrategy进行转换和优化。主要由CompiledProgram类来实现。该API仅支持【静态图】模式。可以使用**with\_data\_parallel**将程序转化为数据并行程序，使其能够运行在多个设备上。 |
| 1. paddle.fluid.initializer   fluid.initializer数据初始化接口，主要为参数初始化提供各种函数 |
| 1. paddle.fluid.regularizer   fluid.regularizer 参数正则化接口 |
| 1. paddle.fluid.optimizer   paddle.fluid.optimizer优化器接口，模型训练时为模型优化提供多种优化方法 |
| 1. paddle.fluid.framework |
| 1. paddle.fluid.core |

1. 关键方法、算法、数据摘取与深入分析（Python或C++）：

要求：方法/公式/数据 + 文字，深入解读代码方法不少于7个

|  |  |
| --- | --- |
| **代码/算法/数据片段** | **输入输出及功能描述分析** |
| 1. lod\_reset() | LoD(Level-of-Detail) Tensor是Fluid中特有的概念，它在Tensor基础上附加了序列信息。Fluid中可传输的数据包括：输入、输出、网络中的可学习参数，全部统一使用LoD-Tensor表示。  Lod\_reset的输入是x和y/目标lod，输出out是将x的lod变为与y的lod或者目标lod一致。 |
| 1. LayerHelper() | 关于Layerhelper的设计，在nn中，以fc为例，fluid.layers.fc()方法的源码最终是调用 helper.create\_parameter()方法来创建可学习参数的，其中helper是LayerHelper的实例。  LayerHelper在Paddle中起着关键作用，它的作用主要是在各个layers函数之间共享代码，相较于LayerHelper，开发全局辅助函数有以下缺点：  1.需要提供这些方法的命名空间，方便开发人员快速定位并使用  2.全局函数迫使图层开发人员需要逐个传递参数  LayerHelper的定义可以避免上述问题，但它通常在layers开发中使用，对应创建模型结构的搭建不会使用到LayerHelper |
| 1. paddle.fluid.layers.sequence\_pool() | input (Variable) - 类型为LoDTensor的输入序列，仅支持lod\_level不超过2的LoDTensor。  pool\_type (str) - 池化类型，支持average，sum，sqrt，max，last和first池化操作。  is\_test (bool) - 仅在pool\_type取值为max时生效。当is\_test为False时，则在池化操作过程中会创建maxIndex临时Tenosr，以记录最大特征值对应的索引信息，用于训练阶段的反向梯度计算。默认为False。  pad\_value (float) - 用于填充输入序列为空时的池化结果，默认为0.0。  返回：经过指定类型池化后的LoDTensor，数据类型为float32。  返回类型：Variable |
| #首先创建执行引擎  place = fluid.CPUPlace() # fluid.CUDAPlace(0)  exe = fluid.Executor(place)  data = fluid.layers.data(name='X', shape=[1], dtype='float32')  hidden = fluid.layers.fc(input=data, size=10)  loss = fluid.layers.mean(hidden)  adam = fluid.optimizer.Adam()  adam.minimize(loss)  #仅运行startup程序一次  exe.run(fluid.default\_startup\_program())  x = numpy.random.random(size=(10, 1)).astype('float32')  outs = exe.run(feed={'X': x}, fetch\_list=[loss.name]) | Executor.run()根绝给定的输入执行指定的program，获取对应的输出，类似tf的session.run()。 |
| 1. save\_vars、save\_params、save\_persistables 以及 save\_inference\_model的区别 | 1) save\_inference\_model 会根据用户配置的 feeded\_var\_names 和 target\_vars 进行网络裁剪，保存下裁剪后的网络结构的 \_\_model\_\_ 以及裁剪后网络中的长期变量  2) save\_persistables 不会保存网络结构，会保存网络中的全部长期变量到指定位置。  3) save\_params 不会保存网络结构，会保存网络中的全部模型参数到指定位置。  4） save\_vars 不会保存网络结构，会根据用户指定的 fluid.framework.Parameter 列表进行保存。  Note: save\_persistables 保存的网络参数是最全面的，如果是增量训练或者恢复训练， 请选择 save\_persistables 进行变量保存。 save\_inference\_model 会保存网络参数及裁剪后的模型，如果后续要做预测相关的工作， 请选择 save\_inference\_model 进行变量和网络的保存。 save\_vars 和 save\_params 仅在用户了解清楚用途及特殊目的情况下使用， 一般不建议使用。 |
| with fluid.unique\_name.guard('A'):  name\_1 = fluid.unique\_name.generate('fc')  with fluid.unique\_name.guard('B'):  name\_2 = fluid.unique\_name.generate('fc')  print(name\_1, name\_2) # Afc\_0, Bfc\_0 | 以guard()函数为例，配合’with’方法使用，在with语句的关联语句中使用新的命名空间，调用generate接口时相同前缀的名称将从0开始重新编号。 |
| 1. paddle.fluid.CompiledProgram.with\_data\_parallel (loss\_name=None, build\_strategy=None, exec\_strategy=None, share\_vars\_from=None, places=None) | 该接口用于将输入的Program或Graph进行转换，以便通过数据并行模式运行该模型。用户可以通过 build\_strategy 和 exec\_strategy 设置计算图构建和计算图执行过程中可以进行的一些优化，例如：将梯度聚合的AllReduce操作进行融合、指定计算图运行过程中使用的线程池大小等。  **参数**：  **loss\_name** （str） - 该参数为模型最后得到的损失变量的名字。  **build\_strategy** （BuildStrategy）: 通过配置build\_strategy，对计算图进行转换和优化，例如：计算图中算子融合、计算图执行过程中开启内存/显存优化等。  exec\_strategy （ExecutionStrategy） - 通过**exec\_strategy**指定执行计算图过程可以调整的选项，例如线程池大小等。  **share\_vars\_from** （CompiledProgram） - 如果设置了share\_vars\_from，当前的CompiledProgram将与share\_vars\_from指定的CompiledProgram共享参数值。需要设置该参数的情况：模型训练过程中需要进行模型测试，并且训练和测试都是采用数据并行模式，那么测试对应的CompiledProgram在调用with\_data\_parallel时，需要将share\_vars\_from设置为训练对应的CompiledProgram。由于CompiledProgram只有在第一次执行时才会将变量分发到其他设备上，因此share\_vars\_from指定的CompiledProgram必须在当前CompiledProgram之前运行。  **places** （list(CUDAPlace)|list(CPUPlace)） - 该参数指定模型运行所在的设备。如果希望在GPU0和GPU1上运行，places为[fluid.CUDAPlace(0), fluid.CUDAPlace(1)]；如果希望使用2个CPU运行，places为[fluid.CPUPlace()] \* 2。 如果没有设置该参数，即该参数为None，模型执行时，将从环境变量中获取可用的设备：如果使用GPU，模型执行时，从环境变量FLAGS\_selected\_gpus或CUDA\_VISIBLE\_DEVICES中获取当前可用的设备ID；如果使用CPU，模型执行时，从环境变量CPU\_NUM中获取当前可利用的CPU个数。  **返回：**配置之后的 CompiledProgram 对象 |
| 1. paddle.fluid.optimizer.AdamOptimizer (learning\_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08, parameter\_list=None, regularization=None, name=None, lazy\_mode=False)   参数更新公式为： | Adam优化器，能够利用梯度的一阶矩估计和二阶矩估计动态调整每个参数的学习率。  **参数:**  **learning\_rate** (float|Variable) - 学习率，用于参数更新的计算。可以是一个浮点型值或者一个值为浮点型的Variable  **parameter\_list** (list) - 指定优化器需要优化的参数。在动态图模式下必须提供该参数；在静态图模式下默认值为None，这时所有的参数都将被优化。  **beta1** (float|Variable) - 一阶矩估计的指数衰减率，是一个float类型或者一个shape为[1]，数据类型为float32的Variable类型。默认值为0.9  **beta2** (float|Variable) - 二阶矩估计的指数衰减率，是一个float类型或者一个shape为[1]，数据类型为float32的Variable类型。默认值为0.999  **epsilon** (float) - 保持数值稳定性的短浮点类型值，默认值为1e-08  **regularization** (WeightDecayRegularizer) - 正则化函数，用于减少泛化误差。例如可以是 L2DecayRegularizer  **name** (str)- 该参数供开发人员打印调试信息时使用，具体用法请参见 Name  **lazy\_mode** （bool） - 设为True时，仅更新当前具有梯度的元素。官方Adam算法有两个移动平均累加器（moving-average accumulators）。累加器在每一步都会更新。在密集模式和稀疏模式下，两条移动平均线的每个元素都会更新。如果参数非常大，那么更新可能很慢。 lazy mode仅更新当前具有梯度的元素，所以它会更快。但是这种模式与原始的算法有不同的描述，可能会导致不同的结果. |
| 1. paddle.fluid.regularizer.L2Decay   具体实现中，L2权重衰减正则化的计算公式如下： | L2Decay实现L2权重衰减正则化，用于模型训练，有助于防止模型对训练数据过拟合。  **参数:**  regularization\_coeff (float) – 正则化系数，默认值为0.0。 |
|  |  |
|  |  |
|  |  |
|  |  |

1. 实验结果呈现与说明（如调试模型的某个中间环节）：

要求：结果截图 + 文字，能证明以上解读的实验不少于3项

|  |  |
| --- | --- |
| **实验结果截图** | **描述说明** |
|  | 利用fc层搭建网络实现拟合函数y=2x的目的，输入为1, 2, 3, 4，可以看到训练好的模型的预测性能较好。 |
|  | 测试lod\_reset函数的效果，输入是x和不同lod\_level的y，可以看到lod\_reset函数的输出out的lod\_level与y一致。 |
|  | dynamic\_lstm测试 |
|  | 该操作创建一个shape为[6,3]，value为7的向量data，并将data的batch\_size（第一维）设为和input（向量like）一致，输出一个[4,3]值为7的结果。 |
| 1.3 | 对输入的LoDTensor (a=[[1.], [3.], [2.], [4.], [6.], [5.], [1.]])进行指定方式的池化操作。这里a的lod为[2,3.2,0]，对于sum pooling来说，结果是[[4.],[12.],[6],[0]]计算方式为4.=1. + 3., 12.=2. + 4. + 6., 6.=5. + 1. |
|  | 构建了一个全连接层，并将隐藏层的均值输出。 |
|  | 调用两次guard()函数，并输入同样的关键词，name\_3还是从0开始命名。 |
|  |  |
|  |  |
|  |  |
|  |  |

1. 疑惑问题点整理：

要求：文字分条目列出，问题点不少于3点

|  |
| --- |
| **问题点** |
| 1. paddle paddle的nn中神经网络的层与各种操作仅被定义为函数，为什么不像pytorch一样补充定义class？ |
| 1. metric和loss模块中通过layer\_helper添加operator来实现，那么operator的具体实现方式在哪里，并不清楚。 |
| 1. rnn.py中的功能只支持静态图，如何在动态图使用rnn还不清楚。 |
| 1. rnn.py operator函数中 包含lstm和dynamic\_lstm，前者可实现多层lstm，但是二者具体关系不能完全理解，以及其和rnn operator的关系不能完全理解 |
| 1. 源码中LayerHelper的作用和设计思路是什么？ |
| 1. 在静态图和动态图下，tensor和numpy数据格式能否相互转换，分别是怎么转换的。 |
|  |
|  |

1. 可改进功能或可开展工作整理：

要求：文字分条目列出，可改进功能功能不少于2点

|  |  |
| --- | --- |
| **可改进功能** | **拟开展工作** |
| 1. pytorch中同时有torch.nn和torch.nn.functional，这两个模块功能非常相似。如果我们只保留nn.functional下的函数的话，在训练或者使用时，我们就要手动去维护weight, bias, stride这些中间量的值，这显然是给用户带来了不便。而如果我们只保留nn下的类的话，其实就牺牲了一部分灵活性，因为做一些简单的计算都需要创造一个类。因此，pytorch同时保留了这两个模块。而paddle paddle的nn中神经网络的层与各种操作仅被定义为函数。 | 仿照pytorch，在paddle中增加常用的神经网络层的class的定义。 |
| 1. Dataloader的返回值全都是generator()，打包效率不高，生成自己的dataset时使用成本过高。 |  |
| 1. Unique name中的switch()函数在逻辑上过于简单，功能上是guard()的子集，通过with内外区分即可实现，可以考虑弃用。 |  |
| 1. 如果在构建CompiledProgram和调用with\_data\_parallel时都指定了build\_strategy，在CompiledProgram中的build\_strategy会被复写，可以设计一个函数只在需要的时候调用来更新build\_strategy。 |  |
| 1. CompiledProgram支持静态图可以多GPU多CPU并行处理，而动态图也有自己多GPU训练的实现方法，但没有看到动态图多CPU多进程处理的接口，能否实现这一功能。 |  |

1. 可参考其他技术资料及文献：

要求：网页链接或文件整理，参考外部资料不少于3处

|  |  |
| --- | --- |
| **资料地址** | **资料涉及内容简述** |
| 1. https://zhuanlan.zhihu.com/p/52706973 | Paddle中的关键概念的简介，包括LoD-Tensor(可变长tensor)以及动态图与静态图机制等等，有助于加深对paddle的理解。 |
| 1. https://www.cs.toronto.edu/~graves/icml\_2006.pdf | CTC 损失 |
| 1. http://www.jmlr.org/proceedings/papers/v9/gutmann10a/gutmann10a.pdf | nce |
| 1. http://www.iro.umontreal.ca/~lisa/pointeurs/hierarchical-nnlm-aistats05.pdf | 层次sigmoid |
| 1. http://arxiv.org/abs/1412.2007 | 采样交叉熵损失 |
| 1. http://icml.cc/2015/wp-content/uploads/2015/06/icml\_ranking.pdf | RankNet |
| 1. http://www.nec-labs.com/uploads/images/Department-Images/MediaAnalytics/papers/nips16\_npairmetriclearning.pdf | Npair损失 |
| 1. https://blog.csdn.net/qq\_33200967/column/info/28685 | PaddlePaddle从入门到炼丹 CSDN博客 |