基于Kubeflow的机器学习服务组件化开发方法

# 引言

随着数字化时代的到来，人类已经迈入人工智能时代，机器学习已经渗透到城市管理、环境保护等各个领域。机器学习受到各行各业从业者的广泛关注。

目前，大部分的机器学习都开始以服务化的方式向外提供机器学习计算，即机器学习即服务MLaaS(Machine Learning as a Service)[1],。采用服务化方式提供机器学习功能的平台大多分为以下几种。第一种是使用训练好的模型，以REST API的方式提供模型的预测服务，如华为的文本识别、文档识别和在线文本翻译等。第二种是指定模型，用户自定义训练集来训练模型。以上两种方法开发的模型自定义程度较低，但开发的速度较快。第三种方式是使用代码开发的方式。但其开发所需时间成本较大，开发较为复杂。当需要使用一种更加简单并可视化的方式开发一个新的机器学习模型时，便需要进行更加细粒度的定义机器学习流程。此时，如果使用以上的三种方式，前两种不能够对模型进行自定义，第三种的虽然可以开发新的机器学习模型，但其复杂的较高。

本文提出一种针对非计算机行业从业者使用的机器学习服务组件化开发平台，该平台可以可视化的对机器学习全流程进行编排，并将编排好的Pipeline进行模型训练。机器学习的整个流程可以划分为不同的模块，每一个模块都有特定的功能意义，在模块内可以指定其具体实现方法。在平台中，所有的具体实现方法都是已经编写好的组件，并以可调用服务的方式向外提供计算功能。因此，在对一个模块的具体实现进行编排时，只需要指定其具体实方法和必要的参数即可。

当提交的Pipeline在服务器上进行训练的时候，需要调度器对Pipeline的运算节点进行调度、元数据管理器对Pipeline执行中的元数据信息进行管理，在编排的Pipeline中随着组件的不同，需要有不同的环境（python环境）等。在市场上有许多包含这些功能的开源流程编排引擎，其中有Airflow、Kubeflow等开源的流程编排引擎。Airflow 是一个使用 python 语言编写的 data pipeline 调度和监控工作流的平台，但Airflow主要是使用定时任务执行Pipeline，并且Airflow不能同时存在多种python环境。Kubeflow是Kubernetes的机器学习工具包，整套Kubeflow可以部署到Kubernetes集群当中，并且Kubeflow能够有多种可用的方式动态的加载组件到Pipeline中，并且在运行时Pipeline中每一个组件都是一个单独的docker容器，一条Pipeline中可以有多种环境并存，正好符合Pipeline的需要。因此，本平台采用Kubeflow作为载体对Pipeline进行调度和运行。

# 机器学习服务的组件化建模方法

## 组件化与模块化

组件：组件的最初的目的是代码重用，功能相对单一或者独立。在整个系统的代码层次上位于最底层，被其他代码所依赖，所以说组件化是纵向分层。同时组件可以理解为是最小单元的复用代码的一种封装。

模块：最初的目的是将同一类型的代码整合在一起，所以模块的功能相对复杂，但都同属于一个业务。不同模块之间也会存在依赖关系，但大部分都是业务性的互相跳转，从地位上来说它们都是平级的。在当前组件化平台上对于模块化还有另一层定义：

* + - * 1. 模块是组件的流程化：一些特定的组件的特定组合，这种组合经常是一起出现在pipeline中并且改变较小，这一系列的组件便可以构成一个模块。
        2. 模块是代码和组件的组合：一些特定组合的组件和一些特定的代码可以完成一类特定的工作，在这类特定的工作中，其组成组件是可同类替换的，并且代码是不变的，但代码的参数可调，这一类特定的组合可以构成一个模块。
        3. 模块是一部分复杂代码的集合：一部分复杂代码，其可以完成特定的功能，且代码不变、参数可调。这一类代码可以构成一个模块。

总体而言，组件化开发和模块化开发实际上有各自的适用领域。组件化开发更多被应用于技术底层的实际实现，而模块化开发则需要结合实际的业务功能。组件化和模块化的具体区别如表1。在项目的实际开发过程中，两种开发模式往往是并存的，结合使用的。

表1 组件化和模块化的具体区别

| **类别** | **目的** | **特点** | **接口** | **成果** | **架构定位** |
| --- | --- | --- | --- | --- | --- |
| 组件 | 重用、解耦 | 高重用、松耦合 | 无统一接口 | 基础库、基础组件 | 纵向分层 |
| 模块 | 隔离/封装 | 高内聚、松耦合 | 统一接口 | 业务框架、业务模块 | 横向分块 |

## 服务的组件化

当机器学习平台需要向外提供服务的时候，为了划分不同的功能对外部提供服务，需要采用组件化的方式对不同的功能进行低耦合的开发。组件化的开发可以使得机器学习流程中的服务都是以单独的方式向其上下游提供服务，不会依赖于其他组件，组件之间的运行互不影响。

机器学习可以在三个层次上进行组件化：

* 1. 应用层：应用层可以提供高度封装好的功能接口，使用者无需机器学习背景知识，只是在Web UI上使用可视化拖拽的形式构建Pipeline，并提交训练数据进行训练。
  2. 中间层：在中间层可以提供训练好的模型、已经编辑好的Pipeline段和编辑好只需要提交训练数据的模型等，使用快速化和高度自定义化的方式实现颗星化需求。
  3. 框架层：框架层可以提供梯度下降算法、softmax、卷积运算等等，这些都是构建深度学习必要的基础组件，但开发者必须设计模型，需要有深厚的深度学习背景知识，以及大量的数据。

在三个层次的组件化中，其组件化的细粒度逐步变小，其自定义程度逐步变高。组件所在的层级越高，接口越简单，使用越方便，但灵活度越低，只适合非常通用的场景。层级越低，灵活性越高，但编写和训练模型就越复杂。高级层次包含了低级层次，从结构上来说，高级层次是对低级层次的使用，低级层次能够有很高的灵活性来组装成高级层次，但其方式比较复杂。高级层次虽然比较简单，但其灵活性不如低级层次。所以处在中间层级的模型组件化能够在复杂性和灵活性之间取得很好的平衡。

## 组件化的开发方法

组件化的开发主要是将一定细粒度的机器学习方法封装为一个组件，封装好的组件能够直接的向外提供服务。将机器学习中需要执行和调用的方法封装为各个组件，每一个组件有其特定的功能。组件是最基本的能够重用的功能封装，不需要统一的接口，因此在开发上没有过多的要求。

在组件化的基础之上可以通过模块化的方式对组件进行一层封装，此时可以将模块定义为一个复杂的代码和组件的集合和功能堆叠，在一个模块中可以存在多个组件和功能代码的组合，一个模块强调的是一个功能的具体实现，模块内部的组件是可替换和可修改的，模块配置中可提供对模块参数进行修改的接口，以此使得整个模块的可修改行更强，在具体pipeline编排的过程中，一个模块可以被当作是一个组件去编排，在编排的同时模块的内部实现是可以在一定程度上进行修改的。

## 组件化服务平台

组件化的服务平台指的是将以组件化开发的功能组件/功能模块以组件化的方式向外提供服务，各个组件之间互不影响、独立运行。在平台上能够动态的去编排组件并形成一条由组件组成的pipeline, pipeline中的每一个组件都是一个单独的服务。在调度时会将每一个组件看作是一个单独的服务，同时也是pipeline最基本的调度单元，调度器向组件中输入需要计算的数据，组件返回计算结果。

# 机器学习服务组件化开发框架

机器学习服务组件化开发框架采用组件化的形式将机器学习服务进行打包，并使其向外提供计算服务。

服务组件体系结构（SCA）[2]是根据面向服务体系结构（SOA）[3]的原则组合系统的建模规范[1]。SAC将实现关注点分为三个构件：(a)：组件实现其业务功能；(b)：将各种组件组装在一起形成一个模块以创建业务解决方案；(c)：服务创建用于远程访问组件和复合功能的接口。

传统的机器学习服务组件化开发框架一般采用SCA的方式，通过Web远程访问API的方式将系统内的算法以服务的方式向外提供，在Pipeline的编排端能够直接组合API调用顺序的方式来构建不同的机器学习模型。此时，一个API便是机器学习流程中的一个组件，每一个组件都是一个单独的或者是集成的Web服务。当调度器中传入Pipeline进行调度时，调度器需要维护整个机器学习流程中的全部输入数据和过程数据。在调用指定API的时候需要将这些数据以HTTP的方式在各个组件之间传输。此时，数据的维护便是调度器中一项任务量不小的工作，并且，当机器学习数据量达到一定规模的时候，Web之间的数据传输就会成为整个机器学习流程中花费时间最多的过程。机器学习组件在服务器中是一直处于部署状态的，当有请求过来时，组件开始运行。这种方式会导致大量的组件服务都是处于上线状态。

本文所提出的机器学习组件化开发框架同样也是将机器学习的不同的步骤封装成为不同的组件，再让不同的组件以独立的方式去执行，并返回执行的结果。与传统机器学习服务组件化开发方法不同的是，当前组件并不是以Web API的方式向外提供服务，并且组件的运行方式也不是在同一个工程项目中。当前组件均是以描述文件的方式存储在云上。当需要对Pipeline进行调度时，只需要将指定需要调用组件的描述文件和输入数据的元数据信息便可。此时，组件会以一个docker容器的方式启动，并到元数据指定的地方加载所需要的数据进行计算，计算完成时，将执行结果存储到元数据指定的结果存储路径中并关闭该容器。在当前的框架中，如果是在分布式的环境场景之下，可以采取移动程序的方式，将数据处理程序传送到数据所在的服务器中运行，避免了使用网络的方式传输大量的数据，节约了大量的数据传输时间。

# 基于开源框架（同构代码）的机器学习服务组件化建模标准与开发方法

同构代码指的是在同一个工程项目中的代码，它们对特定的数据有着相同的数据结构，当一个工程的子模块被组件化为多个不同的组件时，这些组件之间相同数据的数据结构不需要再进行转换便可以直接在组件之间进行传输。

在面对基于开源框架（同构代码）进行组件化和建模时有两种方式对其整个项目进行组件化。一种是扫描整个开源框架的源码，直接对源码进行组件化。一种是在开源项目向外暴露的API方法上添加一层调用该方法的方法，并对外层方法进行组件化。

[1] Ribeiro M , Grolinger K , Capretz M . MLaaS: Machine Learning as a Service[C]// IEEE International Conference on Machine Learning & Applications. IEEE, 2016.

[2] ServiceComponentArchitectureAssemblyModelSpecificationVersion 1.1. Accessed: 30-04-2015. [Online]. Available: <http://docs.oasis-open>.

[3]叶钰, 应时, 李伟斋,等. 面向服务体系结构及其系统构建研究[J]. 计算机应用研究, 2005, 022(002):32-34.