腾讯云 TI 平台 TI-ONE

操作指南

产品文档

腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100。

文档目录

自动建模（AutoML）

操作指南
操作指南总览

最近更新时间：2021-12-22 14:08:44

本文将从交互方式、功能模块、算法手册等方面提供 TI-ONE 更多介绍，供您参考。您在使用过程中遇到问题，可定位至相关内容查看。

交互方式

TI-ONE 支持多样化的交互方式，本模块为您提供详细的操作指引。在此之前，您可以通过 快速入门 了解三种交互方式的操作流程。

- 可视化建模使用指南
- Notebook 使用指南
- TI SDK 使用指南

功能模块

您可以在这里了解 TI-ONE 更多功能与特性。

- 任务列表
- 模型仓库
- 离线批量服务
- 自动建模（AutoML）

算法手册

TI-ONE 为您提供上百种机器学习和深度学习算子与框架，在使用过程中，如果您需要了解算法与框架详情，请参考以下文档。

- 框架及算法说明

Angel 是由腾讯自研并开源的高性能分布式机器学习和图计算平台，TI-ONE 内置了多种 Angel 相关算法与框架，您可以参考以下文档。

- Angel 算法指南

TI-ONE 中的图像分类、目标检测和 BERT 类算法支持迁移学习功能，您可以在预训练模型的基础上，在特定目标任务上进行微调，更多详情请参考以下文档。

- 内置算法的预训练模型

可视化建模使用指南
可视化建模简介

最近更新时间：2021-12-22 11:55:22

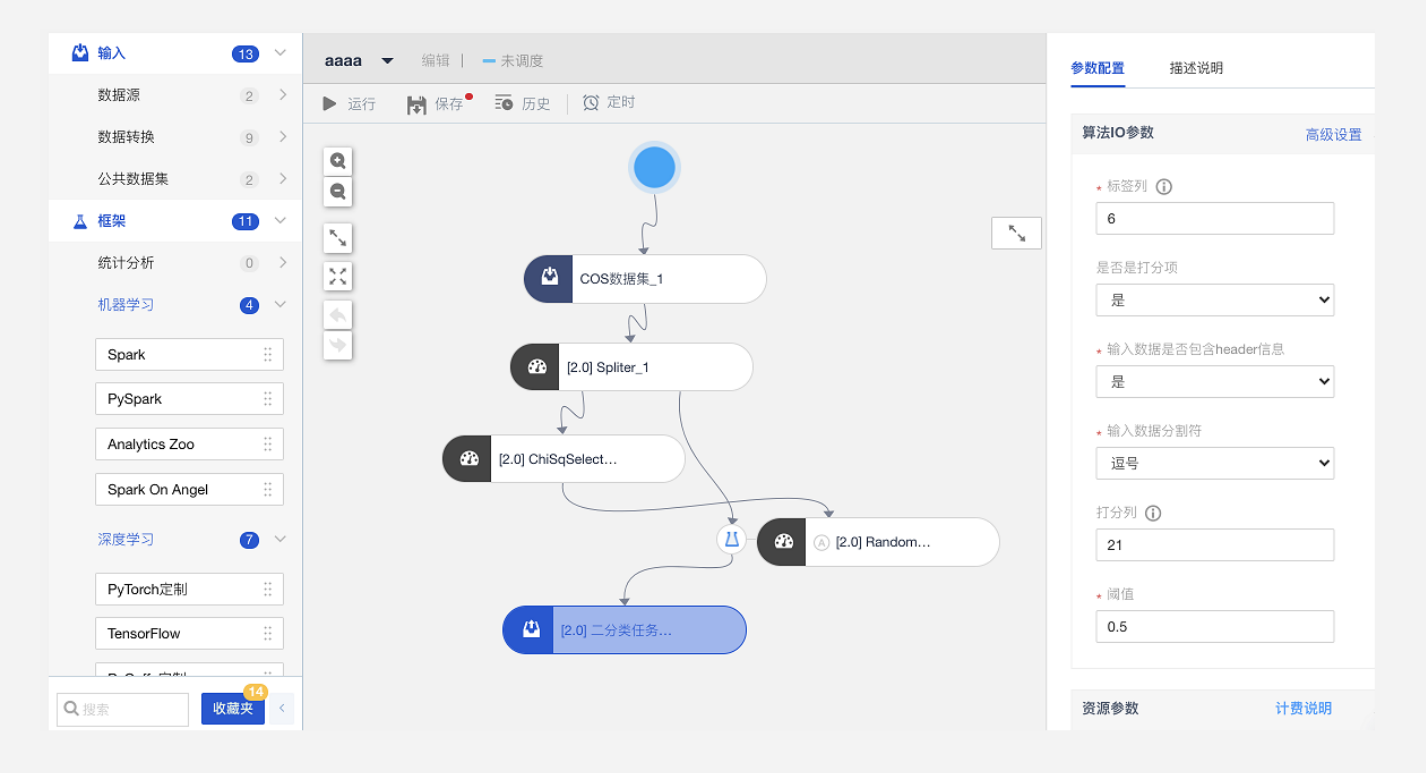
为了方便您有效地使用腾讯云 TI 平台 TI-ONE ，本文档带您进行可视化建模的入门指导。

概述

腾讯云 TI 平台 TI-ONE 通过可视化的拖拽布局，组合各种数据源、组件、算法、模型及评估模块，为 AI 工程师打造从数据预处理、模型训练、到模型评估的全流程开发支持。

操作界面

腾讯云 TI 平台 TI-ONE 可视化建模的整体操作界面如下。左侧为算法框架栏，TI-ONE 为您提供上百种机器学习和深度学习算子与框架，您可以按需选择；中间是工作流画布，您可以将各种数据源、组件和评估模块等在此自由组合；右侧是参数配置区域，您可以在这里配置算法参数和资源参数。



核心特性

- **拖拽式任务流**：TI-ONE 良好的交互体验和易用的功能设计，能够极大地降低机器学习的技术门槛。

- **多种学习框架**：TI-ONE 囊括多种学习框架：PySpark、Spark、Pycaffe、PyTorch、Tensorflow 等，满足不同开发者的使用需求与习惯。
- **丰富算法支持**：TI-ONE 内置丰富算法，从传统的机器学习算法到深度学习，图片分类、目标检测、NLP 满足各类细分场景与应用方向。
- **便捷的效果可视化**：TI-ONE 对源数据底强大可视化交互数据解析，让用户高效直观地了解数据的全貌。
- **全自动建模**：只需要拖动自动建模组件、输入数据 TI-ONE 即可自动完成建模的全流程，无基础的 AI 初学者也可毫无障碍地完成整个训练流程。自动调参工具也可大幅提升 AI 工程师的调参效率。
- **模型训练的完整闭环**：TI-ONE 为用户提供一站式机器学习平台体验，从数据预处理、模型构建、模型训练到模型评估，覆盖全工作流程，形成机器学习训练的完整闭环。
- **灵活的资源调度**：TI-ONE 支持多种的 CPU/GPU 资源，符合用户对差异化算力的场景需求。采用灵活的计费方式，真正帮助用户降本增效。

资源规格

腾讯云 TI 平台可视化建模提供的算力类型和配额默认请参考 配额限制，计费详情请参考 购买指南。

新建工程与任务流

最近更新时间：2022-01-11 09:24:20

使用腾讯云 TI 平台 TI-ONE 建模，首先需要完成新建工程和工作流。在使用之前，请确保您已经完成了 注册与开通服务。

新建工程

登录 腾讯云 TI 平台 TI-ONE 控制台，将平台地域切换为您开通服务时所选地域。在工程列表页面，单击**我的工程>新建工程**，根据提示填写信息：

- **工程名称**：您可按需填写。
- **工程描述**：您可按需填写。
- **COS Bucket**：COS 对象存储服务-存储桶（Bucket）用于读写工程中的训练数据、中间结果等内容，请在下拉列表处选择您的Bucket。注意 COS Bucket 需要与平台处于**相同区域**，若无 COS Bucket 可选，请前往 COS 控制台 新建，详情请参考 创建存储桶。

新建任务流

- **自定义任务流**：在工程中单击“+”号，输入任务流名称，即可新建**自定义任务流**。
- **从模板创建**：腾讯云 TI 平台为您准备了很多案例，在工程中单击“+”号，勾选**从模板中选择新建**，选择您需要的典型任务流。

创建完成后，单击即可进入画布，您还可以对任务流进行更名/复制/启动/删除等操作。

基础操作说明

最近更新时间：2021-12-22 11:55:35

操作场景

本文档将向您介绍可视化建模的基础操作，包括创建节点、配置节点、节点右键菜单、运行工作流、常见状态与查看日志等。

腾讯云 TI 平台 TI-ONE 提供输入、框架、算法、模型、输出、自动建模等大类节点。

- 输入：包含数据源、数据转换和公共数据集，直接拖拽即可使用。
- 框架：包含常用机器学习与深度学习框架。
- 算法：包含机器学习算法、图算法、深度学习算法。
- 模型：包含个人模型。
- 输出：包含可视化和模型评估。
- 自动建模：包含全自动 AutoML 组件。

创建节点

- 从算法区选择组件，拖拽至画布，即可新增节点并自动连线，自动连线后，数据输入输出路径会根据连线自动生成。
- 模型节点具有两个输入桩时，需要手动进行连线。手动连线时，从上一节点的输出桩按住鼠标左键移动至当前节点的输入桩即可完成连线操作。
- 若需删除连线，将鼠标悬停连线上，右键即可删除连线。

配置节点

单击算法节点，页面右侧会出现参数配置框。

配置算法参数

参数配置包含三部分：算法 IO 参数、算法参数和资源参数。
在参数配置中，输入框右边会悬浮展示参数说明，用户可以根据需要调节参数，各算法详情请参考 算法手册。

配置组件参数

腾讯云 TI 平台 TI-ONE 包含机器学习框架和深度学习框架。以机器学习框架中的 Spark 为例，简要说明组件参数的配置。
组件参数的配置包括两部分：组件参数和资源参数。资源参数的配置与算法的资源参数配置类似，不再赘述。组件参数的配置步骤如下：

1. 准备 Spark Jar 包。
2. 单击“作业 Jar 包”右侧输入框，通过本地上传作业 Jar 包或脚本。

节点右键菜单

右键单击画布中的节点，会出现一列工具栏，包括重命名、删除节点等，用户可根据需要进行操作。不同节点的右键菜单栏有所不同。

- **起点运行**：以当前节点为起始点运行工作流。
- **停止任务**：终止任务流的执行。
- **重命名**：更改当前节点的名称（直接双击节点也可进行重命名操作）。
- **运行到该节点**：运行工作流到该节点。
- **运行该节点**：运行当前节点。
- **复制节点**：复制当前节点。
- **删除节点**：删除当前节点。
- **执行设置**：当有节点是不想执行，但是又不想删除时，可以用该功能暂时屏蔽。如果选择"Yes"则执行该节点，如果选择"No"则不执行该节点，常用于调试。
- **收藏**：收藏本节点，放入画布左侧收藏夹，使用时可直接在收藏夹中拖拽出来使用。
- **查看数据**：可进行中间结果查看。

运行工作流

- 保存工作流：单击工具栏**保存**。
- 运行工作流：单击工具栏**运行**。
- 从指定环节运行工作流：右键单击要运行的环节，选择**起点运行**，从该环节开始向下执行。

常见状态

- **就绪**：任务配置成功，已经在 TI 平台后台生成实例，等待计算集群调度。
- **运行中**：任务已经提交集群，并在计算集群上运行。
- **成功**：节点运行成功。
- **失败**：运行失败，通常指计算集群上执行失败。
- **被终止**：运行过程中用户强行停止执行。

特殊状态说明：

- **终止失败**：强行停止执行操作本身失败。
- **强制终止**：终止失败以后，节点的右键菜单会有一个**强制终止**，执行后状态就是“强制终止”。

查看日志

您可以右键相关节点，在右键菜单中，选择“日志信息 > 查看日志”或“ Spark 日志”查看该节点的运行日志。

工具菜单说明

最近更新时间：2021-12-22 11:55:48

腾讯云 TI 平台的画布上方为工具菜单栏，说明如下：

- **运行：**手动运行画布上当前实例。
- **保存：**工作流（包括节点参数）修改后会有小红点提示保存。
- **历史：**查看已经运行完的实例清单及相关信息。您可以选择续跑/重跑操作，或单击详情进入实例快照页面。

- **定时：**添加/取消定时设置，用户可以自行设置任务的开始时间和调度周期。

外部 API 启动工作流

最近更新时间：2021-12-22 11:56:48

用户可通过外部 API 调用，启动运行 TI-ONE 中的工作流。如使用场景之一：用户需等待外部数据清洗等操作完成后，才开始启动 TI-ONE 工作流运行算法模型，此时“定时任务”的功能已无法满足用户需求，则可以通过外部 API 调用来自定义工作流启动时间。

具体操作步骤如下：

1. 下载由 TI-ONE 平台提供的满足该功能的 外部调用工作流 jar 包。
2. 该 jar 包的运行环境：用户需在客户端安装 Java 环境。
3. 用户在 TI-ONE 中搭建工作流，并可通过网页链接找到该工作流的相关信息：流 ID 信息（folwId）和工作流所在地域信息（ap-guangzhou）。
4. 用户在 TI-ONE 提供的 外部调用工作流 jar 包 的客户端安装路径下，运行命令：
   java -cp runflow.jar tione.demo.Demo ${secret_id} ${secret_key} ${region} ${flow_id}
   例如：java -cp runflow.jar tione.demo.Demo xxx yyy ap-shanghai 3035其中，
   - ${secret_id} 和 ${secret_key} 为个人帐户信息，可在 腾讯云访问管理 中查看。
   - ${region} 和 ${flow_id} 为用户通过工作流网页链接得到的 TI-ONE 工作流的“流 ID 信息”和“工作流所在地域信息”。
5. 刷新 TI-ONE 对应工作流页面，即可看到已成功开始运行工作流。

使用内置案例

最近更新时间：2022-01-10 16:07:39

操作场景

腾讯云 TI 平台 TI-ONE 内置了多个案例模板及配套文档，模板中包含完整的搭建及运行工作流的流程，您无需自己搭建工作流，也可以快速体验平台功能。所有算法/组件 Demo 及典型案例，您可以在**工程列表>Demo工程**中进行查看。

前提条件

您需要完成 注册与开通服务。

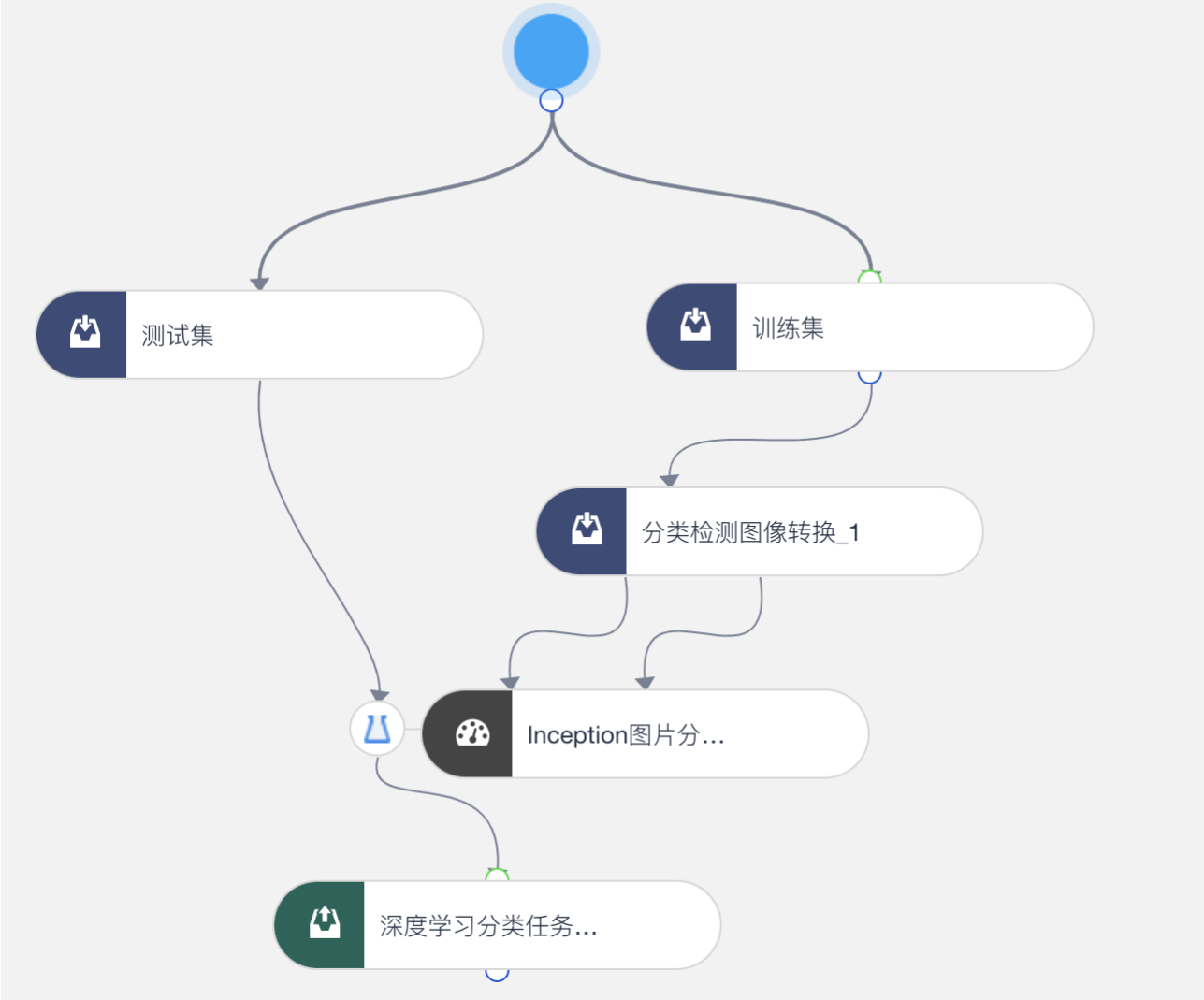
操作步骤

从模板创建工作流

1. 单击页面上的**新建工程**，在弹出框中填入**工程名称**、**工程描述**，并选择**存储桶**。
2. 在新建的工程名下，单击 + 开始创建工作流，从模板中选择您想要创建的工作流，我们以典型案例花朵图像分类为例，创建工作流。

运行工作流

进入画布，单击画布上方**运行**，即可**保存及运行**工作流。



查看模型效果

运行成功后右击该模型的评估指标，即可查看模型效果。在花朵图像分类案例中，可右击**深度学习分类任务评估**，选择**查看数据**，即可查看花朵图像分类模型的效果。

统计指标(表)

序号	label	precision	recall	f1
1	sunflowers	1.0	0.6666666666666666	0.8
2	daisy	0.8	0.9411764705882353	0.8648648648648648
3	dandelion	0.55	1.0	0.7096774193548387
4	roses	0.45	1.0	0.6206896551724138
5	tulips	1.0	0.6060606060606061	0.7547169811320755

accuracy 0.76

当前仅支持查看1000行数据！如需查看全量数据 (限制1万行)，请点击

Notebook 使用指南
Notebook 简介

最近更新时间：2021-12-22 11:58:24

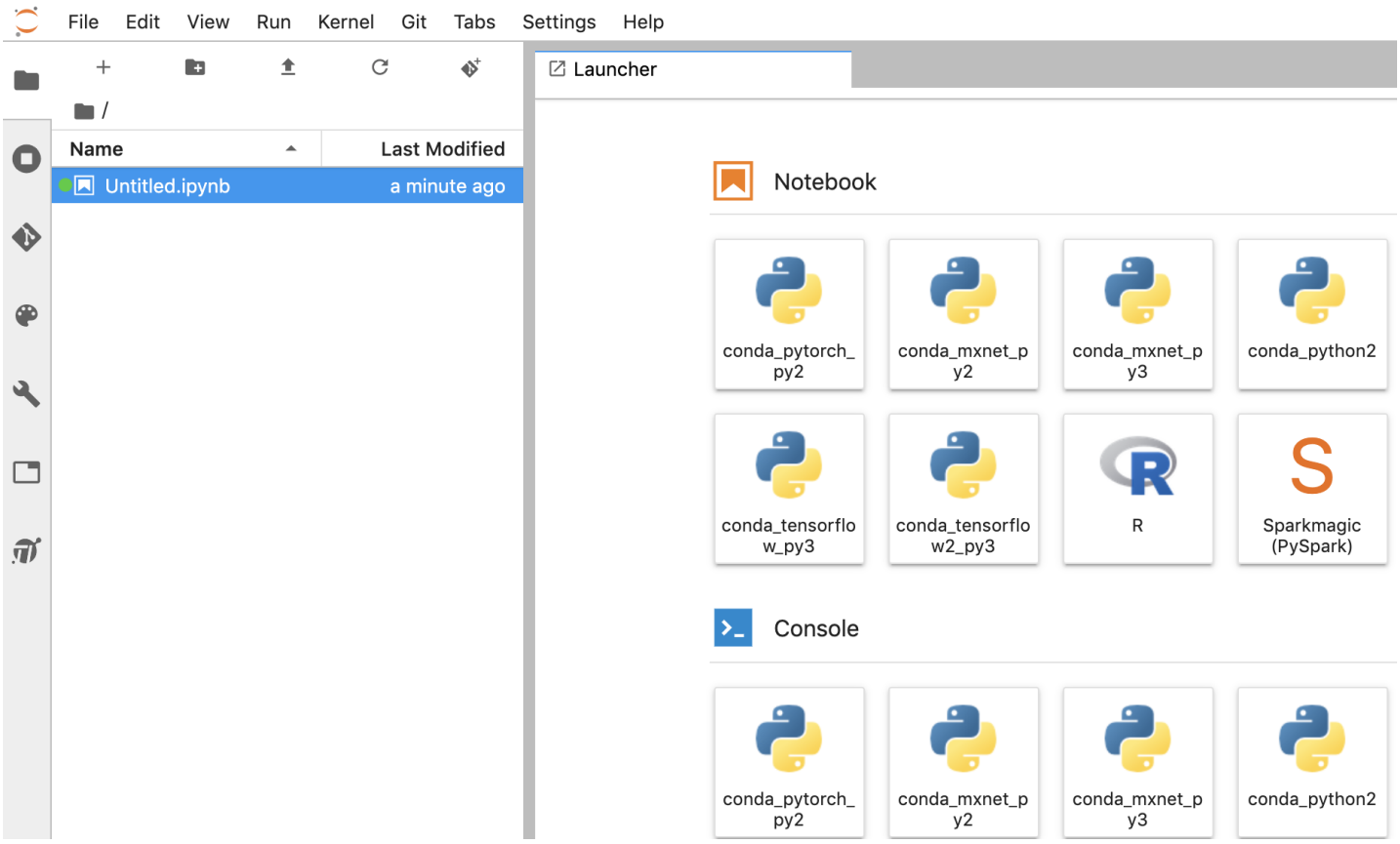
为了方便您有效地使用腾讯云 TI 平台 TI-ONE 的 Notebook，本文档为您提供入门指导。

概述

Notebook 是腾讯云 TI 平台 TI-ONE 为开发者量身打造的灵活的交互式开发工具，您可以在腾讯云 TI 平台 Notebook 中完成数据准备、数据预处理、算法调试与模型训练，无需多平台切换。腾讯云 TI 平台 Notebook 提供了多种内核供您选择，同时也支持自行安装第三方库。使用过程中，您仅需为运行 Notebook 时所消耗的算力及存储资源付费，没有最低费用，也不需要前期承诺。

操作界面

腾讯云 TI 平台 TI-ONE 的 Notebook 整体操作界面如下，右侧为 Notebook 内容展示区域，左栏主要为文件管理和各类功能键。



核心特性

- 提供多种资源规格供用户自由选择。
- 支持各类资源灵活切换，降低使用成本。
- 内置 TI SDK，用户可以在 Notebook 中向 TI 提交训练任务。
- 内置多种内核环境，支持自定义安装第三方库。
- 支持生命周期脚本，用户可以自定义初始化 Notebook，在创建/重启 Notebook 实例时可运行用户预设的 shell 脚本。
- 支持与 Git 存储库对接，避免误删的数据丢失，方便协同开发与版本控制，用户还可以下载公开库里的 Notebook 文件进行学习与编辑。

资源规格

腾讯云 TI 平台 Notebook 提供的算力类型和配额默认请参考 配额限制，计费详情请参考 购买指南。

创建实例

最近更新时间：2021-12-22 11:58:32

操作场景

本文档将向您演示如何在腾讯云 TI 平台 TI-ONE 中创建一个 Notebook 实例。

操作步骤

1. 登录 腾讯云 TI 平台 TI-ONE 控制台 ，单击**菜单栏**的**Notebook**，页面将跳转至 Notebook 的实例列表页面，此页面将罗列用户创建的所有 Notebook 实例。

2. 在 Notebook 实例列表页，单击左上角**新增实例**，跳转至创建 Notebook 实例的设置页面。填写说明如下：
   - **地区**：此字段不可修改，将自动显示平台选择的地区。
   - **Notebook 名称**：设置此 Notebook 实例的名称。
   - **资源选择**：选择此实例需要配置的资源。（**注意**：只要 Notebook 实例处于运行中，都将对配置的资源进行按时收费。）
   - **存储大小**：Notebook 实例的存储大小(以 GB 为单位)，最小值为10GB且为10的倍数，注意：请大于当前硬盘值，最大值为16380GB(约16TB)。
   - 高级设置（默认收起）：
     - **Root 权限**：选择是否赋予 root 权限来访问 Notebook。如果启用 Root 权限，则所有 Notebook 实例用户具有管理员权限，并且可以访问和编辑实例上的所有文件。
     - **生命周期配置**：选择是否使用生命周期脚本。
     - **Git 存储**：此为可选项，用户可以前往 Git 存储库-新增存储库进行配置。
     - **VPC**：用户可以选择配置自有的 VPC 网络。
     - **CLS 日志服务**：用户可以自行选择是否开通 CLS 日志服务。
     - **自动停止**：开启该选项后，该实例将在运行时长超过您选择的时长后自动停止，停止状态计算资源不再收费，存储资源仍会收费，请注意费用产生。自动停止时间以小时为单位，最小为1小时，最大为24小时。
     - **价格**：平台根据您选择的配置显示相关价格。

3. 单击**创建**，Notebook 列表中将新增一条实例记录，用户可单击**状态**列，查看 Notebook 实例创建详细进程（购买资源中、资源准备到位、启动 Notebook 程序等）。当实例“状态”由**创建中**变为**运行中**时，单击**打开**进入 Notebook 实例内部。

管理实例

最近更新时间：2021-12-22 11:58:36

操作场景

本文档将向您演示在腾讯云 TI 平台 TI-ONE 中，如何对 Notebook 实例进行实例管理、资源管理和数据管理。

实例管理

完成**新增实例**后，Notebook 列表中将新增一条实例记录。您可以在此页面查看实例的名称、资源、运行时长、计费项、运行状态、最近更新时间、监控与日志等。您还可以对 Notebook 实例进行打开、停止、编辑、删除等操作。

- 单击**状态**栏，可查看 Notebook 实例详细创建进程。
- 单击**监控与日志**栏，可查看 Notebook 资源监控情况和一键跳转到日志详情。
- 单击**打开**进入 Notebook 实例内部。
- 单击**停止** Notebook 将会停止运行。
- 单击**编辑**可以对部分配置信息进行修改。
- 单击**删除**此实例记录销毁。

实例支持**自动停止**设置，开启该选项后，该实例将在运行时长超过您选择的时长后自动停止，停止状态计算资源不再收费，存储资源仍会收费，请注意费用产生。

- 自动停止时间范围为1-24小时，数值为整数。
- 对于新增实例，您可以设置自动停止时间。
- 对于运行中的实例，您可以单击**小时钟**，开启、禁止或重设自动停止时间。
- 对于已终止的实例，您可以在**启动**或**编辑**页面，设置自动停止时间。

资源管理

腾讯云 TI 平台 TI-ONE 的 Notebook 支持后付费模式下各种资源型号的切换，您可以根据需求进行资源调整，降低使用成本。操作步骤如下：

1. 在 Notebook 列表页面，选定您需要切换资源的实例，单击**停止**，释放当前计算资源。
2. 实例停止后，单击**编辑**，进入设置页面，您可以重新进行资源选择，选定后单击**更新**。
3. 资源更新后，单击**启动**即可重启 Notebook 。

> ⚠ 注意：

后付费的实例按照所选的资源配置与使用时长计费。成功切换资源后，切换前的资源不再计费，平台按照更新后的资源配置与使用时长计费。

数据管理

Notebook 使用 **/home/tione/notebook** 目录作为用户的工作空间。重启 Notebook 时，只有此目录下的数据会被保留，保存在此目录之外的文件和数据将被覆盖。

您可以在新建 Notebook 的时候选择 /home/tione/notebook 的存储大小，对于已运行的实例，您可以根据需求对存储大小进行调整。操作步骤如下：

1. 在 Notebook 列表页面，选定您需要调整存储大小的实例，单击**停止**。
2. 实例停止后，单击**编辑**，进入设置页面，您可以调大存储大小，选定后单击**更新**。
3. 资源更新后，单击**启动**即可重启 Notebook 。

⚠ 注意：

您可以调大存储大小，但无法减小存储大小。如果要减小存储卷的大小，请创建一个具有所需大小的新 Notebook 实例。
在 Notebook 停止运行时，云硬盘也会产生存储费用，该费用在实例删除后停止收取。

使用内置案例

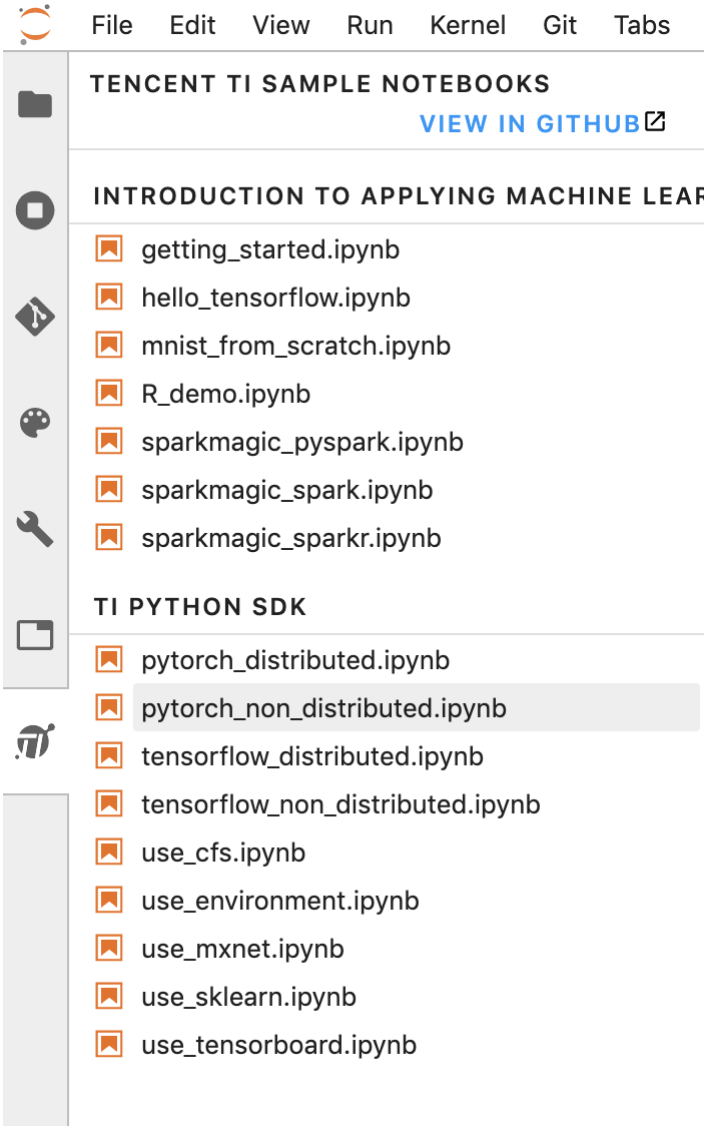
最近更新时间：2021-12-22 11:58:41

操作场景

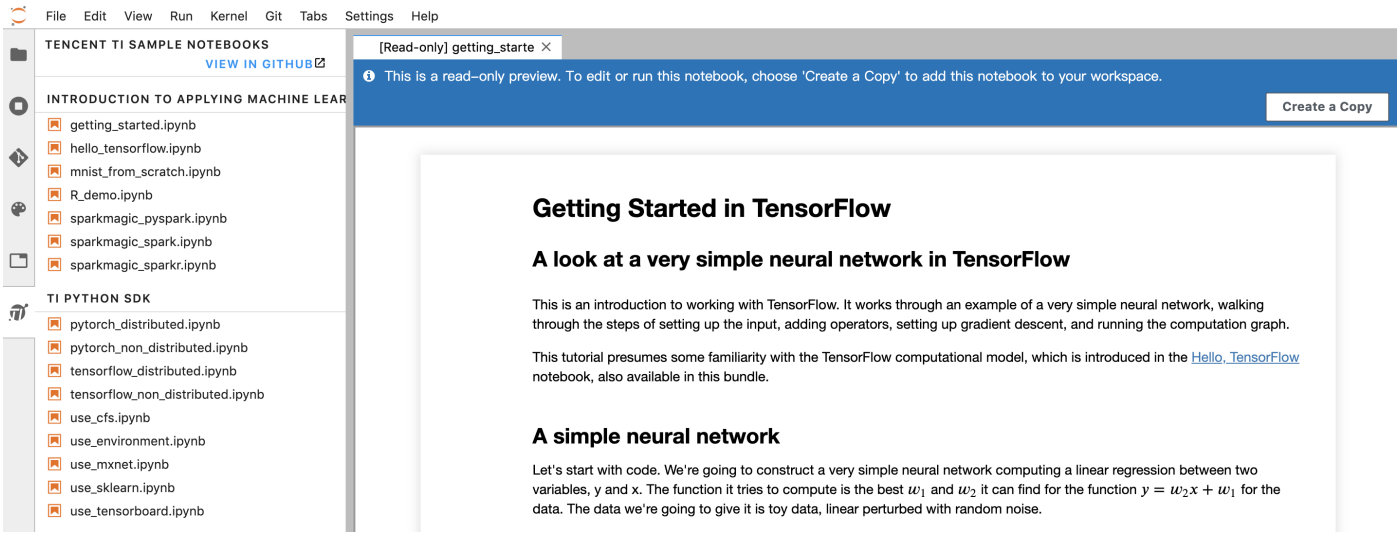
腾讯云 TI 平台 TI-ONE 的 Notebook 内置多个示例，您可以通过内置案例熟悉产品功能。本文档将向您演示如何使用内置案例。

操作步骤

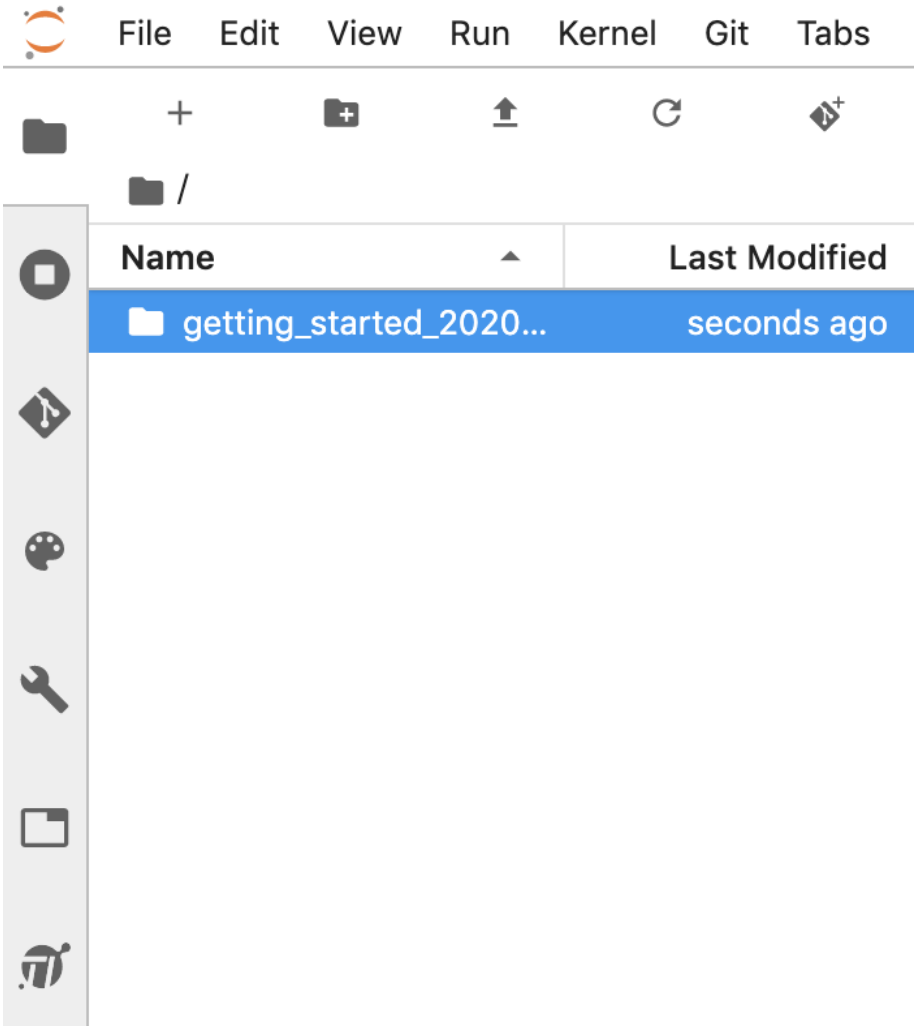
1. 单击左边栏 TI sample notebook 图标，切换到 Notebook 示例。



2. 内置的 Notebook 示例都是只读的，如果需要编辑和运行，单击对应的 Notebook 例子，在打开的页面中，单击右上角"Create a Copy"拷贝至自己的工作空间。



3. 切换到自己的工作空间，会发现工作空间下面多了一个同名的文件夹，这就是第2步拷贝过来的 Notebook 例子。



4. 打开文件夹，单击 ipynb 后缀的文件，可以根据自己的需要编辑和运行 Notebook 示例。

查看日志

最近更新时间：2021-12-22 14:07:25

操作场景

Notebook 日志通过 腾讯云日志服务 CLS 进行存储，本文档将向您介绍如何查看 Notebook 日志。

操作详情

- 创建 Notebook 实例时，您可以在配置框中选择开通 CLS 日志服务或自行保存日志信息。CLS 会对相关服务进行计费，详情请参考 CLS 购买指南。
- 如果选择开通 CLS 的日志服务，CLS 会为您创建名为 Notebook 的 topic ，以保存您所有 Notebook 实例的日志。日志地区将与您的 Notebook 处于同一地区。
- Notebook 在停止状态时，相应 topic 不会删除；当您删除所有 Notebook 实例后，相应 topic 也会随之删除。
- 我们只为您保留最近 7 天的日志数据，请注意日志数据时效。

设置内核

最近更新时间：2021-12-22 11:58:54

操作场景

本文档向您介绍腾讯云 TI 平台 TI-ONE 提供的多种内核环境。

操作详情

设置内核

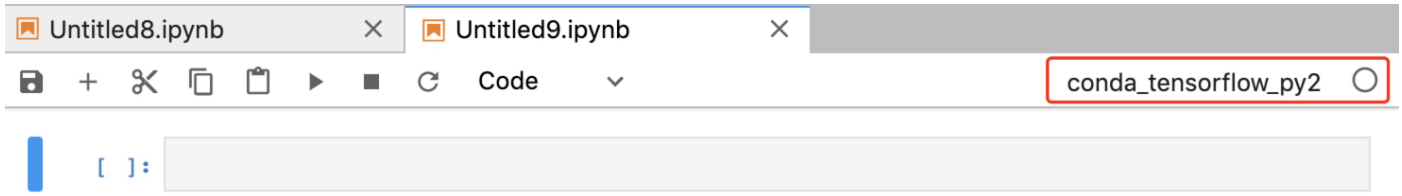
目前腾讯云 TI 平台 TI-ONE 提供了以下13种内核选择，您可以根据需要选择适合自己的 Notebook 多内核环境。

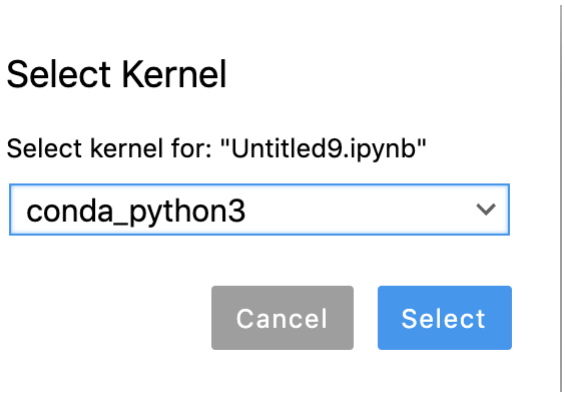
- 纯净的 python 环境
  - python3 : conda_python3
  - python2 : conda_python2
- TensorFlow 环境
  - TensorFlow 1.14 + python3 : conda_tensorflow_py3
  - TensorFlow 1.14 + python2 : conda_tensorflow_py2
  - TensorFlow 2.0.0 + python3 : conda_tensorflow2_py3
- PyTorch 环境
  - Pytorch 1.2.0 + python3 : conda_pytorch_py3
  - Pytorch 1.2.0 + python2 : conda_pytorch_py2
- MxNet 环境
  - MxNet 1.5.0 + python3 : conda_mxnet_py3
  - MxNet 1.5.0 + python2 : conda_mxnet_py2
- R环境
  - R 3.6.1 : R
- Sparkmagic 环境
  - Spark : Sparkmagic(Spark)
  - PySpark : Sparkmagic(PySpark)
  - SparkR : Sparkmagic(SparkR)

切换内核

设置完成后，如需切换内核，您可以参考以下步骤：

1. 单击右上角内核图标。





2. 在弹出框的下拉列表中选择您需要的内核，进行切换。

安装第三方库

最近更新时间：2021-12-30 17:02:50

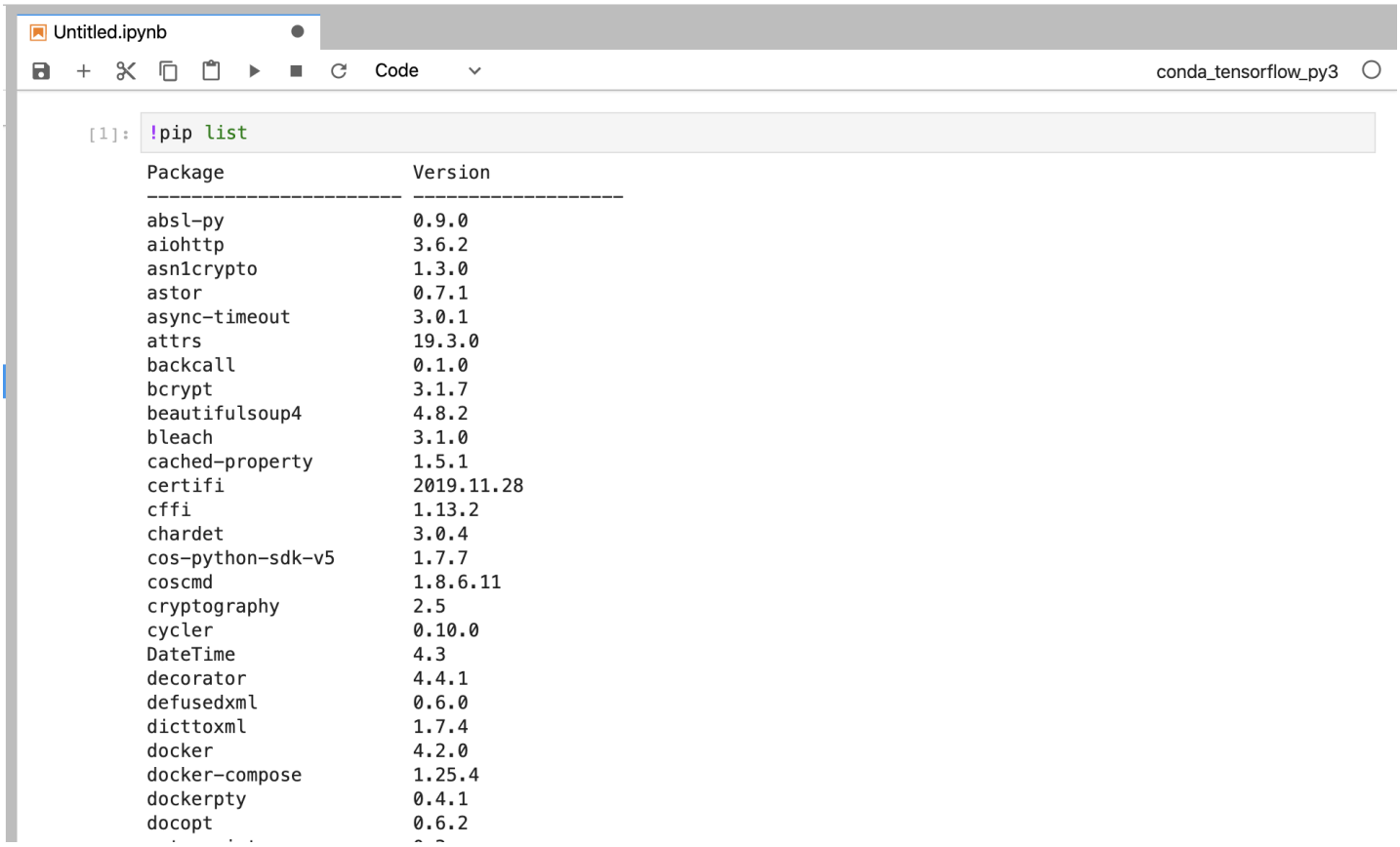
操作场景

本文档将向您演示如何在 Notebook 中**查看已有依赖包**以及**安装第三方库**。

操作步骤

查看已有依赖包

您可以选择适合自己的 Notebook 内核环境，在对应内核的输入框中输入 pip list，查看该内核已有的依赖包：



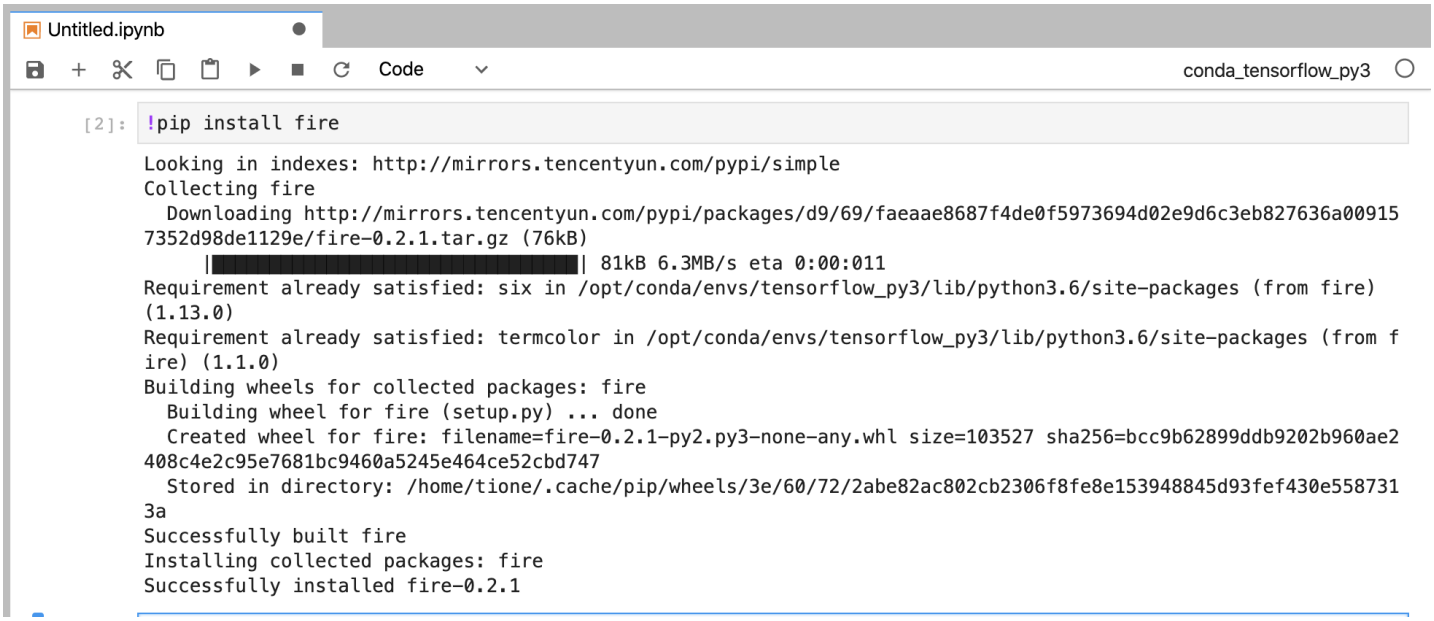
安装外部依赖包

依赖包在腾讯云 pip 源仓库中已有

如果您期望安装的依赖包及其对应版本可以在

https://mirrors.cloud.tencent.com/pypi/simple/

中找到，则可以直接在对应内核的命令框中通过 !pip install 安装。



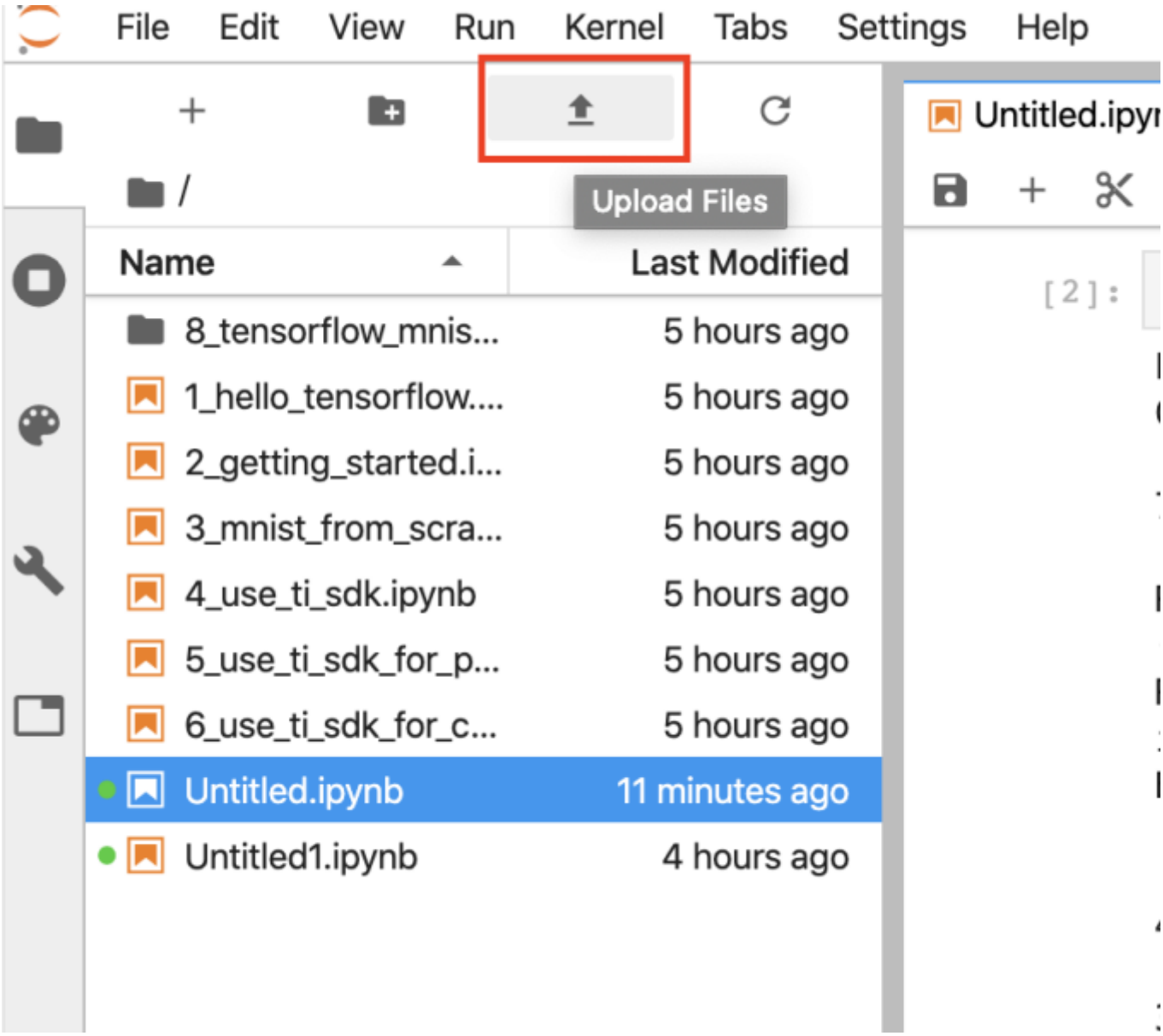
依赖包不在腾讯云 pip 源仓库中

在默认的情况下，Notebook 具有访问外网的权限。您可以通过外网下载第三方依赖包到本地，再在内核的命令框中通过 pip install 安装。

如果您的 Notebook 额外配置了不带有 Internet 访问权限的子网时，可以考虑：

- 首先，将所需的第三方依赖包从外部网络下载。

- 其次，上传依赖包。单击“上传文件”，将依赖包上传。上传大小限制300MB。



- 最后，新建 Notebook 文件，在对应内核的命令框中通过 !pip install 命令安装上传的外部包。
- 上述步骤成功后，您才能通过 import 命令使用。

数据上传与下载

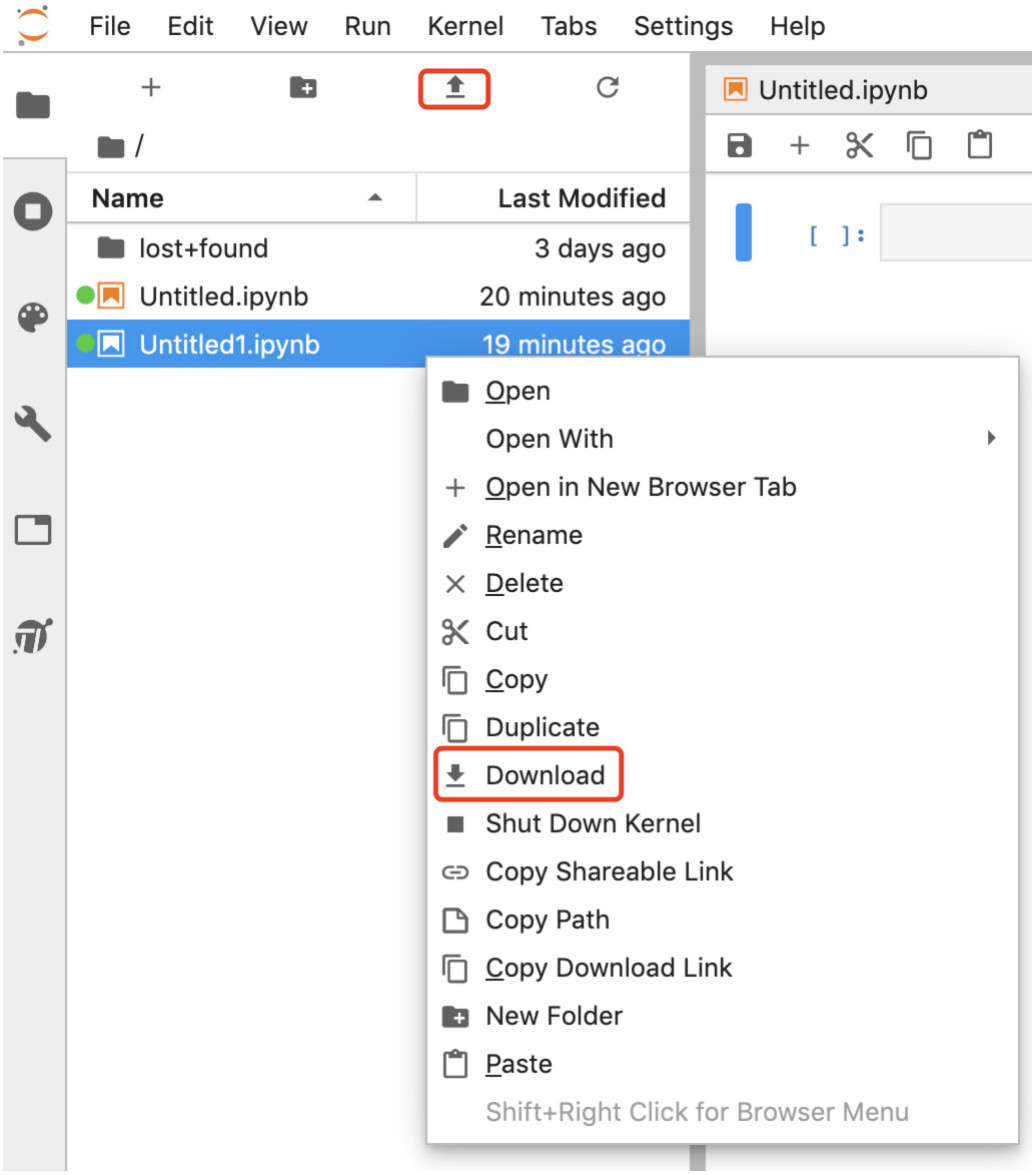
最近更新时间：2021-12-22 12:03:19

操作场景

本文档将向您介绍如何在 Notebook 中进行数据的上传与下载等操作。

小文件的上传与下载

- 如果文件大小不超过 300M，您可以直接通过 Notebook 页面的文件上传工具进行上传。
- 如果文件大小不超过 300M，您可以直接在 Notebook 中右键文件，单击 Download 进行下载。



大文件的上传与下载

对象存储 COS 是腾讯云的分布式存储服务，将会应用于腾讯云 TI 平台 TI-ONE 中的各个环节，包括训练数据、中间结果数据和模型文件的存放与读取等。对于超过 300M 的文件，我们建议您通过 COS 进行上传和下载。

> ⚠ 注意：
> 您的 COS 存储桶需要与腾讯云 TI 平台 TI-ONE 处在同一地区，如平台处于上海地区，COS 存储桶也需要在上海地区。如果不在同一地区，请您参考 COS SDK 文档进行操作。

从个人 COS 导入数据到 Notebook

您可以参照以下示例代码进行鉴权和数据导入。

获取密钥：

```
import os
import requests
cred_url = os.environ["QCLOUD_CONTAINER_INSTANCE_CREDENTIALS_URL"]
r = requests.get(cred_url)
secretId = r.json()["TmpSecretId"]
secretKey = r.json()["TmpSecretKey"]
token = r.json()["Token"]
```

导入数据步骤如下，其中：

- local_file：您指定的本地文件路径，数据将被导入至此路径。
- bucket：您存放数据的存储桶名。
- data_key：您的数据文件路径（注意，您的数据需要是一个文件，不能直接导入文件夹）。


```
import os
from qcloud_cos import CosConfig
from qcloud_cos import CosS3Client
from ti.utils import get_temporary_secret_and_token

#### 指定本地文件路径，可根据需要修改。
local_file = "/home/tione/notebook/mnist_cnn.pt"

#### 用户的存储桶，修改为存放所需数据文件的存储桶，存储桶获取参考腾讯云对象存储
bucket="demo-project-ap-shanghai-1259675134"

#### 用户的数据，修改为对应的数据文件路径，文件路径获取参考腾讯云对象存储
data_key="mnist_cnn.pt"

#### 获取用户临时密钥
secret_id, secret_key, token = get_temporary_secret_and_token()
config = CosConfig(Region=os.environ.get('REGION'), SecretId=secret_id, SecretKey=secret_ke
y, Token=token, Scheme='https')
client = CosS3Client(config)

#### 获取文件到本地
```

```
response = client.get_object(
Bucket=bucket,
Key=data_key,
)
response['Body'].get_stream_to_file(local_file)
```

从公共 COS 导入数据到 Notebook

您可以参照以下示例代码进行数据导入。

```
!pip install wget
import wget, tarfile
filename = wget.download("https://tesla-ap-guangzhou-1256322946.cos.ap-guangzhou.myqcloud.com/cephfs/tesla_common/deeplearning/dataset/contest/demo.zip")
print(filename)
```

```
import zipfile
zFile = zipfile.ZipFile(filename, "r")
for fileM in zFile.namelist():
zFile.extract(fileM, "./")
print(fileM)
zFile.close();
```

从 Notebook 上传数据到 COS

在使用 Notebook 的过程中，您可以自行指定结果保存的 COS 路径，以下是示例代码。

- path：结果文件的路径。
- bucket：指定存储桶。
- key_prefix：存储桶下的 COS 路径。

```
from ti import session
ti_session = session.Session()
inputs = ti_session.upload_data(path="result_file", bucket="demo-project-ap-guangzhou-1259675134", key_prefix="contest")
```

后续您可以在对象存储 COS 中您指定的路径下查看文件。

使用生命周期脚本配置 Notebook 实例

最近更新时间：2021-12-22 12:03:29

生命周期脚本配置规则

生命周期配置提供 SHELL 脚本，在用户创建 Notebook 实例或每次启动 Notebook 实例时运行，可以帮助用户安装自定义依赖，个性化配置 Notebook 环境。

生命周期配置遵循以下规定：

- 创建脚本：第一次新建 Notebook 后启动 Notebook 实例会运行的脚本，只会运行一次。
- 启动脚本：每次启动 Notebook 实例时都会运行的脚本，包括第一次创建时。
- 每个脚本 BASE64 编码后不能超过16384个字符。
- 每个脚本将以 root 用户的角色运行。
- 每个脚本的 $PATH 环境变量为 /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin
- 每个脚本最长运行时间为5分钟，超过5分钟 Notebook 将启动失败，请避免在脚本中安装大型依赖包。可在详情页中查看失败原因如“启动脚本超时”。
- 如果脚本出错，Notebook 也将启动失败，可在详情页中查看具体失败原因。
- 如果脚本是从自己的编辑器复制到 TI-ONE 网页上的，请确保编辑脚本的编辑器使用 Unix 风格的编排。

生命周期脚本最佳实践

以下是使用生命周期配置的一个实践案例：

- 生命周期脚本以 root 用户权限运行，Notebook 进程以 tione 用户运行。如果需要切换用户，可以在脚本中运行 sudo -u tione 切换到 tione 用户。
- Notebook 使用 conda 管理多内核，可以激活 conda env 来为不同的内核安装依赖包。

例如：在 conda_python3 的内核中安装 Python 依赖包 fire，可以编写如下启动脚本：

```
#!/bin/bash
sudo -u tione -i <<'EOF'

# This will affect only the Jupyter kernel called "conda_python3".
source /opt/conda/bin/activate python3

# Replace fire with the name of the package you want to install.
pip install fire
# You can also perform "conda install" here as well.
```

```
source /opt/conda/bin/deactivate

EOF
```

例如：在所有内核中都安装 fire 依赖包，可以这样编写脚本：

```
#!/bin/bash
sudo -u tione -i <<'EOF'

# Note that "base" is special environment name, include it there as well.

for env in base /opt/conda/envs/*; do
source /opt/conda/bin/activate $(basename "$env")

# Installing packages in the Jupyter system environment can affect stability of your tione
# Notebook Instance. You can remove this check if you'd like to install Jupyter extensions, etc.
if [ $env = 'JupyterSystemEnv' ]; then
continue
fi

# Replace myPackage with the name of the package you want to install.
pip install fire
# You can also perform "conda install" here as well.

source /opt/conda/bin/deactivate
done

EOF
```

关联 Git 存储库与 Notebook

最近更新时间：2022-01-10 16:19:48

操作场景

腾讯云 TI 平台 TI-ONE 支持 Notebook 与 Git 存储库关联。您可以将 Notebook 保存到 Git 库中，不必担忧信息因为 Notebook 实例的删除和关闭而丢失。您也可以将一个 Notebook 实例与一个默认存储库和多个其他存储库同时关联，最多关联3个其他存储库。

与存储库关联有以下特性优势：

- **持续存储**
  Notebook 实例中的笔记本存储在腾讯云 CBS 中，与 Git 存储库关联后，您可以在 Notebook 实例之外管理您的笔记本，避免实例误删导致的数据丢失。
- **项目合作**
  当与多个伙伴协同开发机器学习项目时，将 Notebook 数据存储到 Git 存储库将方便您进行数据的共享与版本控制。
- **开源项目的学习**
  目前很多机器学习项目都在 Github 等平台上开源，用户可以轻松关联自己的笔记本实例与公开库，并下载其中的 Jupyter notebooks，基于其进行学习。

操作详情

新增存储库

前往 Notebook Git 存储库 页面，单击【新增存储库】，在弹窗中输入存储库的名称，目标存储库的 URL 以及存储库分支的名称（可选）。如果目标存储库为需要验证的私有存储库，您必须准确输入所需的用户名与密码，用

于凭证验证。

> ⚠ 注意：
>
> - 出于密钥安全考虑，凭证将由腾讯云 KMS 产品进行加密和存储，具体内容请参见 KMS 说明 与 KMS 定价，历史创建的密钥可前往 KMS 控制台 查看。
> - 对于 GitHub 存储库，建议使用个人访问令牌而不是您的账户密码。具体详情请参考 创建用于命令行的个人访问令牌 文档。

新增 Notebook 实例关联 Git 存储库

前往 Notebook 页面，单击【新建实例】，您可以在 Git 存储的下拉列表中看到在第一步创建的存储库名称。单击□图标可以添加除默认存储库以外的其他存储库，最多添加3个其他存储库。

除了选择在第一步创建的存储库，用户还可以选择在此实例中克隆一个公共 Git 存储库，然后在输入框中正确输入此公共存储库的 URL 即可。

进入 Notebook 后，成功关联的存储库将在左栏的 Git 模块下显示。

历史实例关联 Git 存储库

已经创建的 Notebook 实例也可以与 Git 存储库关联。如果是运行中的实例，您需要先单击【停止】，再单击【编辑】，对 Git 存储相关信息进行编辑，重新**启动**后成功关联的存储库将在左栏的 Git 模块下显示。

编辑已创建的 Git 存储库

前往**Notebook-Git 存储库**页面，单击【编辑】，可以对已创建的 Git 存储库的密钥进行编辑。

> ⚠ **注意：**
>
> 一旦创建，存储库的名称和 URL 将**不能**修改，如果输入错误，建议删除重新新建。

使用 Notebook 远程连接腾讯云 EMR

最近更新时间：2021-12-22 12:03:41

操作场景

弹性 MapReduce（EMR）是云端托管的弹性开源 Hadoop 服务，支持 Spark、Hbase、Presto 等大数据框架。本文档将向您介绍如何使用 Notebook 远程连接腾讯云的 EMR。

操作详情

新建 EMR 集群

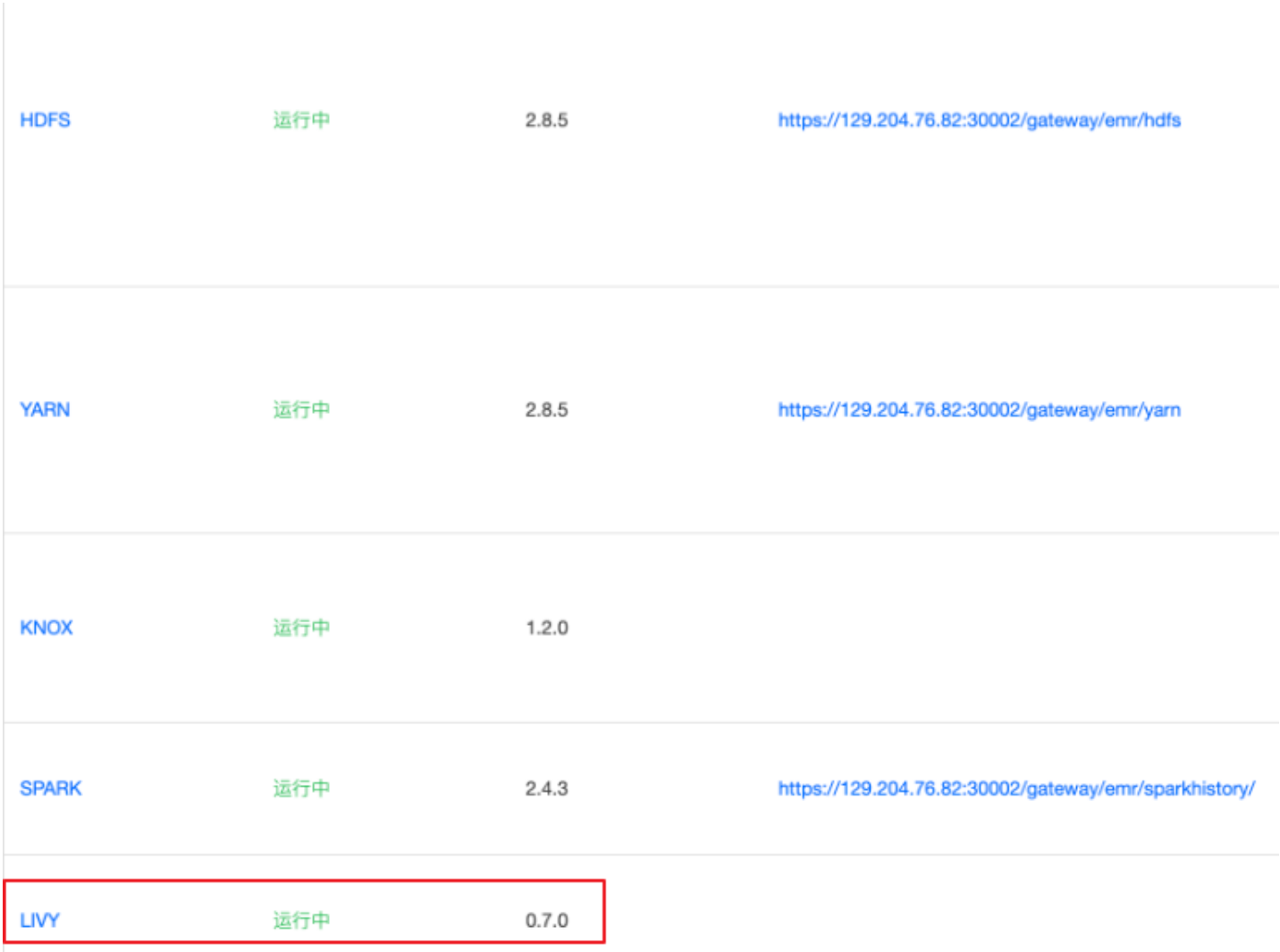
登录 EMR 控制台，单击【创建集群】，选择地域（请选择 TIONE 支持的地域），选择产品版本【EMR-V2.2.0.tlinux】，勾选可选组件【spark_hadoop2.8 2.4.3】和【livy 0.7.0】。

单击下一步：基础配置，进入硬件配置，请选择合适的机型，并且配置网络的 vpc 和子网。

单击下一步配置其他集群信息，完成 EMR 集群的创建。

查看 livy 服务的 IP

单击左侧的【组件管理】，找到 LIVY 服务。



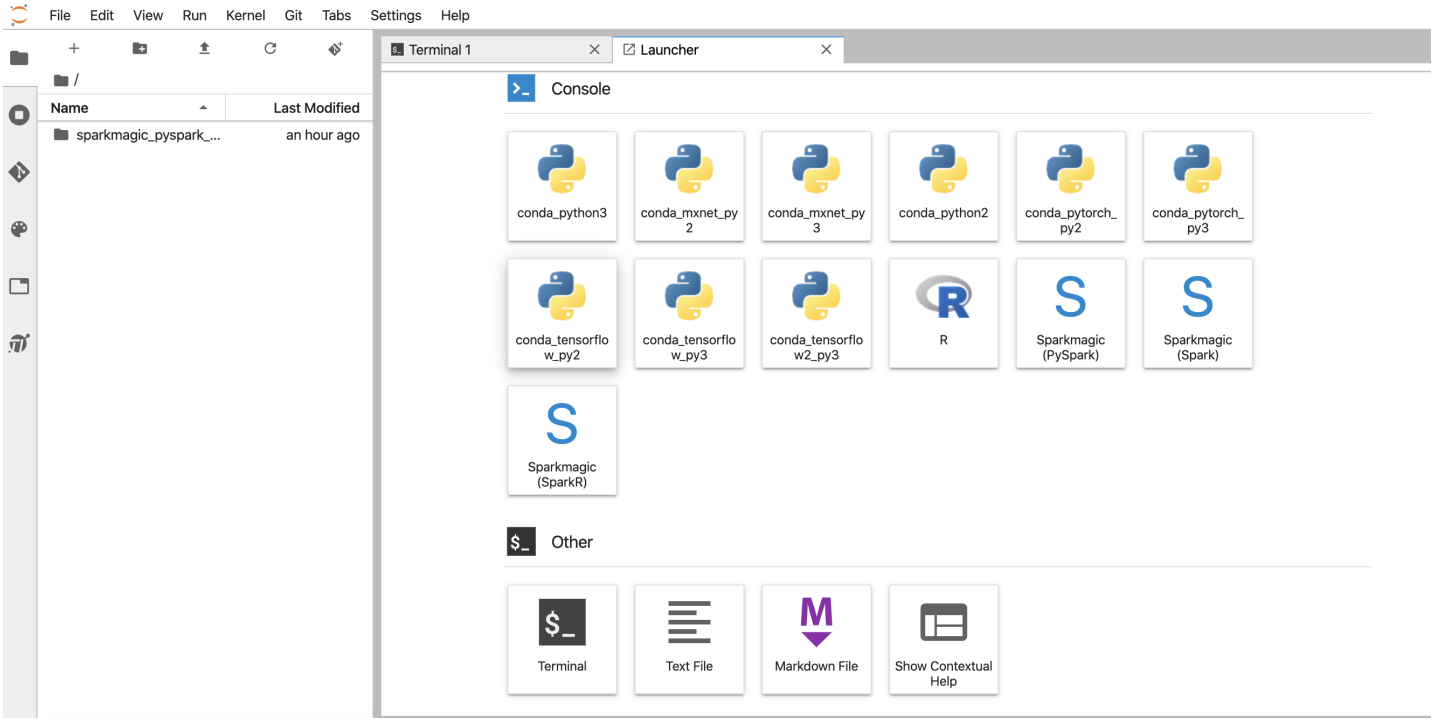
查看 LIVY 服务的 IP，记住其一即可。

新建 Notebook 配置 EMR 连接凭据

登录 腾讯云 TI 平台 TI-ONE 控制台 ，选择与 EMR 集群相同的地域，单击【新增实例】，打开 VPC 选项，请确保此处的 VPC 和子网与新建 EMR 时的选择相同。


新建Notebook实例
地区
广州
Notebook名称 *
请输入Notebook名称
资源选择 *
CPU-2核4G
存储大小(GB) *
10
Notebook实例的存储大小(以 GB 为单位)，最小值为 10GB 且为 10 的倍数，注意：请大于当前硬盘值，最大值为 16380GB(约16TB)。
高级设置
Root 权限
允许
生命周期配置
不使用生命周期脚本
Git存储
无
此为可选选项，用户可以前往 Git存储库-新增存储库进行配置。
VPC
Default-VPC
子网
subnet-hu2hzkka(172.1
直接访问Internet
启用
CLS日志服务
启用
开启该选项后，CLS会为您创建名为Notebook的topic, 以保存您近7天的日志数据。计费详情请参考CLS购买指南
自动停止
禁止
开启该选项后，该实例将在运行时长超过您选择的时长后自动停止，停止状态计算资源不再收费，存储资源仍会收费，请注意费用产生。
计算资源价格
存储资源价格
总价
创建
取消

打开 Notebook，在启动界面的最下面找到 Terminal。



修改 ~/.sparkmagic/config.json 文件，替换 kernel_python_credentials、kernel_scala_credentials、kernel_r_credentials url 里的 localhost 为 LIVY 的服务 IP。

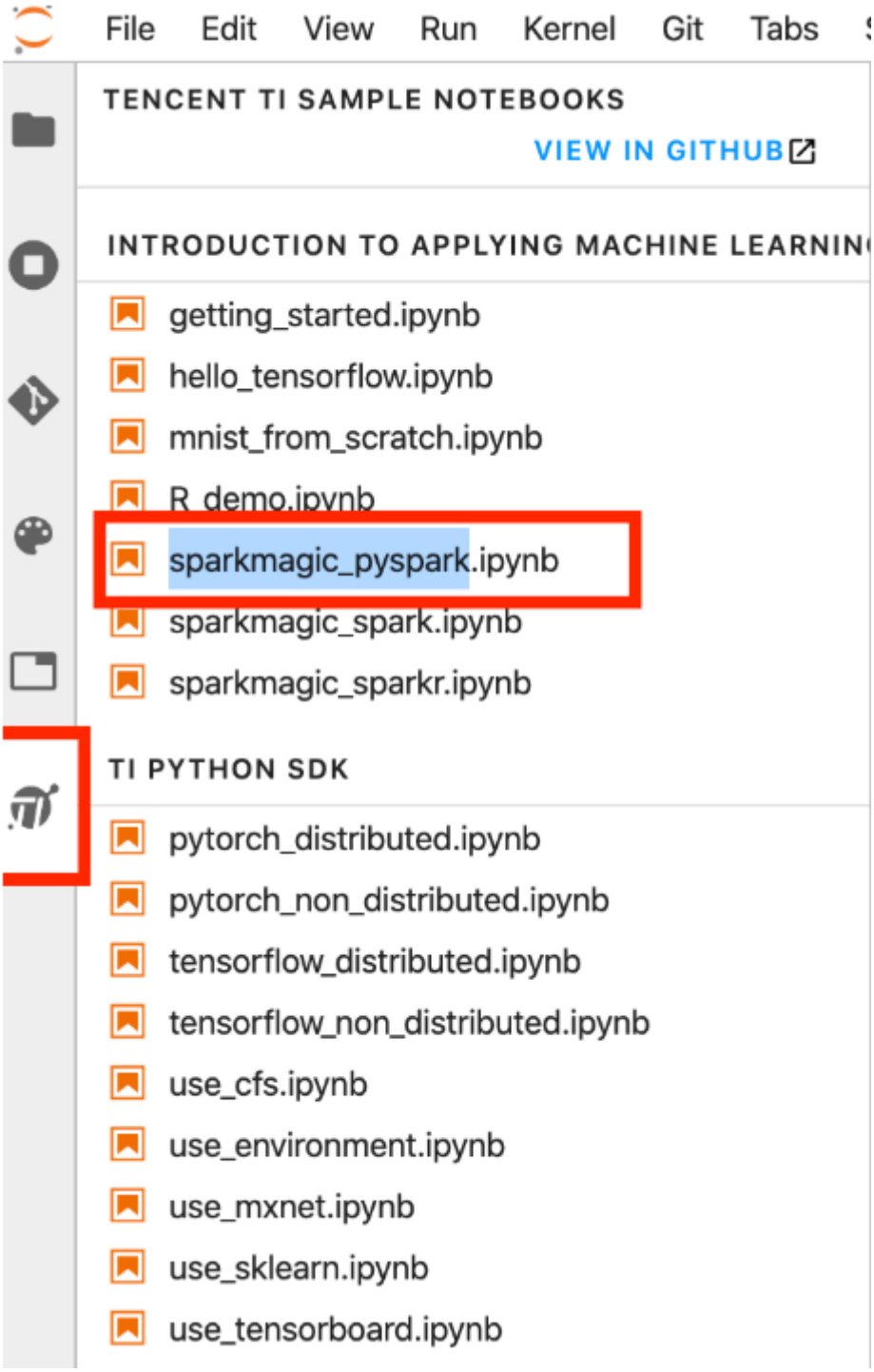
Terminal 1 | Launcher

```
{
  "kernel_python_credentials" : {
    "username": "",
    "password": "",
    "url": "http://localhost:8998",
    "auth": "None"
  },

  "kernel_scala_credentials" : {
    "username": "",
    "password": "",
    "url": "http://localhost:8998",
    "auth": "None"
  },
  "kernel_r_credentials": {
    "username": "",
    "password": "",
    "url": "http://localhost:8998"
  },

  "logging_config": {
    "version": 1,
    "formatters": {
      "magicsFormatter": {
        "format": "%(asctime)s\t%(levelname)s\t%(message)s",
        "datefmt": ""
      }
    },
    "handlers": {
      "magicsHandler": {
        "class": "hdijupyterutils.filehandler.MagicsFileHandler",
        "formatter": "magicsFormatter",
        "home_path": "~/.sparkmagic"
      }
    },
    "loggers": {
      "magicsLogger": {
-- INSERT --
```

如果 EMR 中配置了其他访问相关的设置，例如：auth， kerbersos，您均可在 ~/.sparkmagic/config.json 文件中配置，详情请参见 config.json GitHub文件 。

使用 SparkMagic 远程连接 EMR

打开 Notebook 自带的 Example，找到 sparkmagic_pyspark.ipynb，单击【Create a Copy】，运行示例代码，即将 Spark 程序运行在远程的 EMR 集群中。



您可以在 Notebook 的 Terminal 中手动 curl LIVY 的服务，查看 session 进程。

```
sh-4.4$ curl http://10.0.0.6:8998/sessions
{"from":0,"total":0,"sessions":[]}sh-4.4$
sh-4.4$ curl http://10.0.0.6:8998/sessions
{"from":0,"total":1,"sessions":[{"id":1,"name":null,"appId":"application_1585819262434_0002","owner":null,"proxyUser":null,"state":"sta
rting","kind":"spark","appInfo":{"driverLogUrl":"http://10.0.0.12:5008/node/containerlogs/container_1585819262434_0002_01_000001/hadoop
","sparkUiUrl":"http://10.0.0.6:5004/proxy/application_1585819262434_0002/"},"log":["stdout: ","\nstderr: ","20/04/02 20:14:28 WARN Nat
iveCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable","20/04/02 20:14:2
8 WARN RSCConf: Your hostname, 10.0.0.6, resolves to a loopback address, but we couldn't find any external IP address!","20/04/02 20:14
:28 WARN RSCConf: Set livy.rsc.rpc.server.address if you need to bind to another address.","20/04/02 20:14:33 WARN Client: Neither spar
k.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.","20/04/02 20:14:39 WARN Client: Same
name resource file:///usr/local/service/spark/python/lib/pyspark.zip added multiple times to distributed cache","20/04/02 20:14:39 WARN
 Client: Same name resource file:///usr/local/service/spark/python/lib/py4j-0.10.7-src.zip added multiple times to distributed cache","
\nYARN Diagnostics: "]}]}sh-4.4$
```

TI SDK 使用指南
TI SDK 简介

最近更新时间：2021-12-22 12:03:51

为了方便您有效地使用腾讯云 TI 平台 TI-ONE 的 TI SDK，本文档为您提供入门指导。

概述

TI SDK 是腾讯云 TI 平台 TI-ONE 提供的开源软件包，让用户可以通过代码（SDK/API）向 TI-ONE 提交机器学习和深度学习的训练任务。SDK 简单易用，让用户专注于模型和算法等业务逻辑，无需关心底层硬件和系统配置；同时 SDK 又通过镜像规范和集成云上各种组件服务提供了强大的灵活性，例如，用户可以通过 SDK 提供的训练日志、资源监控和指标监控等特性对训练过程了如指掌。目前您可以在腾讯云 TI 平台的 Notebook 内直接使用 TI SDK；也可以在本地 IDE 环境中使用 TI SDK，直接使用 pip 命令安装：pip install ti-sdk-python 。

核心特性

- 内置了深度优化的 TensorFlow 和 PyTorch 等多种流行的深度学习框架，提供单机和多机多卡训练的能力。
- 对接丰富的云上服务（CLS、CM、COS、CBS、CFS、VPC、CAM、CLB 等），利用对象存储 COS、容器服务 TKE、日志服务 CLS、云监控 Monitor 等腾讯云上成熟的组件作为支撑，帮助用户在云上快速搭建自己的机器学习和深度学习训练任务。
- 通过 API/SDK 方式使用机器学习平台能力，支持 CPU 和 GPU 等多种算力类型。
- 跟用户 VPC 互通、安全访问用户 VPC 内的服务和资源。
- 对接 CFS（Cloud File System），支持超大文件和高速 I/O 的述求。
- 支持自定义镜像（满足腾讯云 TI 平台镜像规范）。

资源规格

TI SDK 提供的算力类型和配额默认如下，计费详情您可以参考 购买指南。如果您需要更多的配额项数量，可通过 配额申请工单 提出配额申请。

算力类型	默认值
TI.SMALL2.1core2g	20
TI.SMALL4.1core4g	20
TI.MEDIUM4.2core4g	20
TI.MEDIUM8.2core8g	20

算力类型	默认值
TI.LARGE8.4core8g	20
TI.LARGE16.4core16g	20
TI.2XLARGE16.8core16g	20
TI.2XLARGE32.8core32g	20
TI.3XLARGE24.12core24g	20
TI.3XLARGE48.12core48g	20
TI.4XLARGE32.16core32g	20
TI.4XLARGE64.16core64g	20
TI.6XLARGE48.24core48g	20
TI.6XLARGE96.24core96g	15
TI.8XLARGE64.32core64g	17
TI.8XLARGE128.32core128g	11
TI.12XLARGE96.48core96g	11
TI.12XLARGE192.48core192g	8
TI.16XLARGE128.64core128g	9
TI.16XLARGE256.64core256g	6
TI.20XLARGE320.80core320g	5
TI.GN10X.2XLARGE40.1xV100	11
TI.GN10X.4XLARGE80.2xV100	5
TI.GN10X.9XLARGE160.4xV100	2
TI.GN10X.18XLARGE320.8xV100	1
总实例数	20

使用 SDK

最近更新时间：2021-03-25 15:26:15

操作场景

本文档向您介绍如何使用 TI SDK 训练模型。

在 Notebook 中使用 TI SDK

我们在 Notebook 中内置了 TI SDK 的案例，您可以通过典型案例快速上手，详情请参考 使用内置案例。

在本地环境使用 TI SDK

若您的 TI SDK 环境为非腾讯云 Jupyter Notebook 环境时，您需要先配置 TI SDK 环境。
TI SDK 配置的环境目录为 ~/.ti/config.yaml，用户需要提供的配置信息如下：

1. region：训练任务提交的腾讯云资源的地域，目前支持 ap-guangzhou，ap-shanghai。
2. uin：腾讯云账号 ID，可在腾讯云控制台 账号信息 中查看。
3. app_id：腾讯云账号 AppID，可在腾讯云控制台 账号信息 中查看。
4. secret_id：腾讯云账号 API 密钥 ID，可在腾讯云控制台 API 密钥管理 中查看。
5. secret_key：腾讯云账号 API 密钥 KEY，可在腾讯云控制台 API 密钥管理 中查看。

示例如下：

```
# ~/.ti/config.yaml的内容格式如下：
basic:
region: 你的腾讯云地域
uin: 你的uin
app_id: 你的appid
secret_id: 你的secret_id
secret_key: 你的secret_key
```

操作步骤

TI SDK 使用以下几个核心类实现 TI 的模型训练

- Estimators： 对训练任务的抽象。
- Session：使用 TI 资源的方法集合。

使用 TI SDK 训练模型需要以下四个步骤：

1. 准备一个训练脚本。
2. 上传训练数据集。
3. 构造一个 Estimator。
4. 调用 Estimator 的 fit 方法。

准备训练脚本

训练脚本必须在 Python2.7 或3.6环境下执行。TI 提供了很高的兼容性，只需要少部分改动就可以将外部环境运行的训练脚本适配到 TI 中，同时 TI 提供了许多环境变量定义训练环境各种资源和参数，在训练脚本中可以直接访问这些环境变量获取相关属性，包括：

- TM_MODEL_DIR：string 类型，表示容器中模型的输出路径，设置为/opt/ml/model。
- TM_NUM_GPUS：整型，表示实例可用的 GPU 数。
- TM_OUTPUT_DATA_DIR：string 类型，表示容器中输出数据（例如 checkpoints、图像或其他文件，但不包括生成的模型）的路径。
- TM_CHANNEL_XXXX：string 类型，表示输入训练数据的路径，XXXX对应fit参数中通道的名字，如 train 和 test 两个通道对应的环境变量是TM_CHANNEL_TRAIN TM_CHANNEL_TEST
- TM_HPS：json 格式的超级参数。

一个典型的训练脚本处理流程如下：

1. 从输入通道加载训练数据。
2. 读取超参数配置。
3. 开始训练模型。
4. 保存模型。

TI 会运行用户的训练脚本，建议将启动训练的入口代码放到 main 方法中（ if__name__== '__main__' ）

上传训练数据集

- TI SDK 任务采用 COS 对象存储数据源或 CFS 文件存储数据源作为训练脚本的输入源。
- 当您采用 CFS 文件存储数据源作为输入源时，您需要提前将数据集拷贝至文件系统目录，详见 使用文件系统提交训练任务。
- 当您采用 COS 对象存储数据源作为输入源时，您需要将本地数据集上传至目标 COS 中。

以下例子展示了一个简单的本地数据上传 COS 的使用：

```
from ti import session
ti_session = session.Session()

bucket = "your bucket"
```

```
key_prefix = "train-data"
path = "train-data"

inputs = ti_session.upload_data(bucket=bucket, path=path, key_prefix=key_prefix)
```

参数

- bucket：str 用户 COS 对象存储桶名称。
- path：str 用户数据集的本地目录路径。
- key_prefix：str 用户数据集 COS 桶下的存储路径。

使用 Estimator 提交训练任务

Estimator 是对一个训练任务的高级抽象，包含训练镜像、算力资源、安全权限、算法参数、输入输出等一次训练依赖的所有参数。TI 针对 Tensorflow、PyTorch 等多种流行的机器学习框架分别封装了 Estimator 的具体实现。

以下例子展示了一个简单的 Tensorflow Estimator 使用：

```
tf_estimator = TensorFlow(role=role,
train_instance_count=1,
train_instance_type='TI.SMALL2.1core2g',
py_version='py3',
script_mode=True,
framework_version='1.14.0',
entry_point='train.py',
source_dir='gpu/code')

tf_estimator.fit(inputs)
```

参数

- role：str 用户在云控制台创建的角色，需要传递角色给 TI，授权 TI 服务访问用户的云资源。
- train_instance_count：int 创建的算力实例数量。
- train_instance_type：str 创建的算力类型，目前支持的类型有：

CPU 算力

类型

类型
TI.SMALL2.1core2g
TI.SMALL4.1core4g
TI.MEDIUM4.2core4g
TI.MEDIUM8.2core8g
TI.LARGE8.4core8g
TI.LARGE16.4core16g
TI.2XLARGE16.8core16g
TI.2XLARGE32.8core32g
TI.3XLARGE24.12core24g
TI.3XLARGE48.12core48g
TI.4XLARGE32.16core32g
TI.4XLARGE64.16core64g
TI.6XLARGE48.24core48g
TI.6XLARGE96.24core96g
TI.8XLARGE64.32core64g
TI.8XLARGE128.32core128g
TI.12XLARGE96.48core96g
TI.12XLARGE192.48core192g
TI.16XLARGE128.64core128g
TI.16XLARGE256.64core256g
TI.20XLARGE320.80core320g

GPU 算力（V100）

类型

类型
TI.GN10X.2XLARGE40.1xV100
TI.GN10X.4XLARGE80.2xV100
TI.GN10X.9XLARGE160.4xV100
TI.GN10X.18XLARGE320.8xV100

- train_volume_size：int 附加的云硬盘大小，单位 GB。
- entry_point：str 训练任务的执行入口点名称。例如下面的代码路径中，mnist.py 为训练任务执行入口点，而 np_convert.py 和 image_convert.py 分别为依赖的代码。

```
├── code
│   ├── np_convert.py
│   ├── image_convert.py
│   └── mnist.py
```

- source_dir：str 训练任务的代码路径，将会统一压缩代码路径上传至用户 COS 中。
- hyperparameters：dict 超级参数，将传递到训练容器中。
- train_max_run：int 最大运行时间，单位秒，超过设定时间若训练未完成，TI 会终止训练任务（默认值：24 * 60 * 60）。
- input_mode：输入类型，默认 File。
- base_job_name：str fit()方法启动的训练任务名称前缀，如果没有指定，会使用镜像名和时间戳生成默认任务名。
- output_path：用于保存模型和输出文件的 COS 路径，如果未指定，会生成默认的存储桶。
- subnet_id：str 子网 ID，如果未指定，将在没有 VPC 配置的情况下创建任务。
- image_name：str 训练任务镜像名称，用户可传入 TKE 镜像仓库中自定义训练镜像。

更多的参数意义请参考 EstimatorBase 类（ ti-python-sdk/src/ti/EstimatorBase.py）。

调用 fit 方法

> ⚠ 注意：
> fit(inputs=None、wait=True、logs=True、job_name=None)
> fit 方法会创建并启动一个训练任务

参数

- inputs： 存储训练数据集的 COS 路径或 CFS 文件系统信息，可以采用以下多种数据结构。
  - str：例如：cos://my-bucket/my-training-data，COS URI，表示数据集的路径。
  - dict[str, str]：例如{'train': 'cos://my-bucket/my-training-data/train', 'test': 'cos://my-bucket/my-training-data/test'}，可以指定多个通道的数据集。
  - FileSystemInput：表示 CFS 数据集的数据结构，详见 使用文件系统提交训练任务。
  - dict[str, FileSystemInput]：例如{'train': TrainFileSystemInput, 'test': TestFileSystemInput}，可以指定多个 CFS 数据集的字典结构。
- logs (bool)：认为 False，是否打印训练任务产生的日志。如果设置为 True，将开通 CLS 日志服务，并输出训练的任务日志。CLS 会为您创建名为 TrainingJob 的 topic ，以保存您所有训练任务的日志。
- wait (bool)：默认为 True，是否等待直到训练完成。如果设置为 False，fit 立即返回，训练任务后台异步执行。
- job_name (str)：训练任务名称。如果未指定，则 Estimator 将根据训练镜像名和时间戳生成默认名字。

查看日志与监控

最近更新时间：2021-12-22 12:04:45

操作场景

本文档将向您介绍如何查看训练日志及监控。

- 通过**日志服务 CLS** 查看日志
- 通过**任务列表**查看监控。

使用 CLS 查看日志

- TI-ONE 会收集训练日志并上传到 腾讯云日志服务 CLS ，支持关键词过滤和更多检索语法。
- CLS 会对相关的服务进行计费，详情请参考 CLS 购买指南。
- 目前 TI-ONE 会默认创建一个日志集（TiOne）和日志主题（TrainingJob），您可以通过以下条件过滤指定任务的日志。

job 对应 训练任务 名称，例如：

```
job: tensorflow-2019-08-25-17-59-28-182
```

更多日志检索语法请参考 开启索引 文档。

> ⚠ **注意：**
> 我们只为您保留最近7天的日志数据，请注意日志数据时效。

使用任务列表查看监控

您可以通过 TI-ONE 平台的任务列表，查看 SDK 任务的资源使用情况。详见 任务列表。

Tensorflow 单机训练任务

最近更新时间：2020-12-14 11:45:46

操作场景

本文档将向您介绍如何使用 TI SDK 训练 Tensorflow 模型。

操作步骤

使用 TI SDK 训练 Tensorflow 模型只需要三步：

1. 准备训练脚本。
2. 构造一个 ti.tensorflow.Tensorflow Estimator。
3. 调用 Estimator 的 fit 方法。

TI 预置了1.14和2.0.0两个版本的 Tensorflow 镜像，用户也可以上传自定义镜像，自定义镜像的版本不受限制，只需要参考 TI 容器规范，参考 使用自定义镜像训练模型。

准备训练脚本

TI 中使用的训练脚本和标准的 Tensorflow 脚本非常相似，只需少量的修改就可以将用户现有的 Tensorflow 训练脚本适配到 TI 中。训练脚本可以直接读取注入的环境变量和超级参数。

可用的环境变量列表：

- TM_MODEL_DIR：string 类型，表示容器中模型的输出路径，设置为/opt/ml/model。
- TM_NUM_GPUS：整型，表示实例可用的 GPU 数。
- TM_OUTPUT_DATA_DIR：string 类型，表示容器中输出数据（例如 checkpoints、图像或其他文件，但不包括生成的模型）的路径。
- TM_CHANNEL_XXXX：string 类型，表示输入训练数据的路径，XXXX对应fit参数中通道的名字，如 train 和 test 两个通道对应的环境变量是TM_CHANNEL_TRAIN TM_CHANNEL_TEST。

一个典型训练脚本的工作流程是

1. 读取环境变量指定的路径加载数据。
2. 读取超级参数用于训练（可选）。
3. 训练完成后，将模型保存到约定的目录中。

TI 会运行用户的训练脚本，建议将启动训练的入口代码放到 main 方法中（ if__name__== '__main__' ）。

构造 Estimator

```
tf_estimator = TensorFlow(role=role,
train_instance_count=1,
train_instance_type='ML.GN8.3XLARGE112',
py_version='py3',
script_mode=True,
framework_version='1.14.0',
entry_point='train.py',
output_path="一个cos的路径",
source_dir='gpu/code')
```

调用 fit 方法

```
tf_estimator.fit('cos://bucket/path/to/training/data')
```

必须参数

- inputs：存储训练数据集的 COS 路径，可以采用以下两种数据结构。
  - str：例如：cos://my-bucket/my-training-data，COS URI，表示数据集的路径。
  - dict[str, str]：例如{'train': 'cos://my-bucket/my-training-data/train', 'test': 'cos://my-bucket/my-training-data/test'}，可以指定多个通道的数据集。

可选参数

- wait (bool)：默认为 True，是否阻塞直到训练完成。如果设置为 False,fit立即返回，训练任务后台异步执行，后面仍可通过attach方法附加。
- logs (bool)：默认为 True，是否打印训练任务产生的日志。只有在wait为 True 时才生效。
- run_tensorboard_locally (bool)：如果设置为 True，将打印 Tensorboard。
- job_name (str)：训练任务名称。如果未指定，则 Estimator 将根据训练镜像名和时间戳生成默认名字。

工作流程

调用 fit 方法启动训练任务后，TI 后台会执行以下操作：

- 启动train_instance_count数量的train_instance_type对应的 CVM 实例在每台实例上：
- 拉取预置 Tensorflow 镜像启动容器。
- 下载训练数据集。
- 设置训练相关的环境变量。
- 开始训练。

如果 fit 方法指定了参数 wait=False，fit 方法立即返回，训练任务在后台异步执行。之后可以通过 attach 方法获得 Tensorflow Estimator，可以继续打印标准输出。

```
tf_estimator.fit(your_input_data, wait=False)
training_job_name = tf_estimator.latest_training_job.name

# after some time, or in a separate Python notebook, we can attach to it again.

tf_estimator = TensorFlow.attach(training_job_name=training_job_name)
```

Tensorflow 分布式训练任务

最近更新时间：2020-06-29 15:23:40

操作场景

本文档将向您介绍如何使用 TI SDK 实现 Tensorflow 分布式训练。

TI 支持**参数服务器**和 **Horovod** 两种类型的分布式训练框架。使用 distributions 参数配置分布式训练策略。

使用参数服务器训练

如果指定 parameter_server 作为 distributions 参数的值，容器会在训练集群上启动参数服务器。

以下例子中创建了一个使用2台实例的参数服务器训练任务

```
tf_estimator = TensorFlow(role=role,
train_instance_count=2,
train_instance_type='TI.SMALL2.1core2g',
py_version='py3',
framework_version='1.14.0',
entry_point='train.py',
source_dir='path/code',
distributions={
'parameter_server': {'enabled': True}
}
)
tf_estimator.fit(inputs)
```

使用 Horovod 训练

Horovod 是一个基于 MPI 的分布式训练框架。 使用 Horovod 需要使用1.12以上版本的 Tensorflow（TI 预置镜像是1.14.0和2.0.0）
容器内设置 MPI 环境并执行 mpirun 命令，可以运行任何 Horovod 训练脚本

distributions 支持的 mpi 参数有

- enabled (bool)：如果设置为 True，则设置 MPI 并执行 mpirun 命令。
- processes_per_host (int)：MPI 应在每台实例上启动的进程数。请注意，这个值需不大于实例上的 GPU 卡数

- custom_mpi_options (str)：可以在此字段中传递任何 mpirun 支持的参数选项，TI 执行的 mpirun 时会附带此参数，以启动分布式 horovod 训练。

以下例子中创建了一个使用2台单V100卡机器的分布式训练任务

```
tf_estimator = TensorFlow(role=role,
train_instance_count=2,
train_instance_type='TI.GN10X.2XLARGE40.1xV100',
py_version='py3',
hyperparameters=hyperparameters,
framework_version='1.14.0',
entry_point='train.py',
source_dir='path/code',
distributions={
'mpi': {
'enabled': True,
'processes_per_host': 1,
'custom_mpi_options': '--NCCL_DEBUG INFO'
}
}
)
tf_estimator.fit(inputs)
```

Pytorch 单机训练任务

最近更新时间：2020-07-28 10:30:19

操作场景

本文档将向您介绍如何使用 TI SDK 训练 Pytorch 模型。

操作步骤

使用 TI SDK 训练 Pytorch 模型只需要三步

1. 准备训练脚本
2. 构造一个 ti.pytorch.Pytorch Estimator
3. 调用 Estimator 的 fit 方法

TI 预置了1.1版本 Pytorch 镜像，用户也可以上传自定义镜像，自定义镜像的版本不受限制，只需要参考 TI 容器规范，参考 使用自定义镜像。

准备训练脚本

TI 中使用的训练脚本和标准的 Pytorch 脚本非常相似，只需少量的修改就可以将用户现有的 Pytorch 训练脚本适配到TI中。训练脚本可以直接读取注入的环境变量和超级参数。

可用的环境变量有：

- TM_MODEL_DIR：string 类型，表示容器中模型的输出路径
- TM_NUM_GPUS：整型，表示实例可用的 GPU 数
- TM_OUTPUT_DATA_DIR：string 类型，表示容器中输出数据（例如 checkpoints、图像或其他文件，但不包括生成的模型）的路径
- TM_CHANNEL_XXXX：string 类型，表示输入训练数据的路径，XXXX对应fit参数中通道的名字，如 train 和 test 两个通道对应的环境变量是TM_CHANNEL_TRAIN TM_CHANNEL_TEST.

一个典型训练脚本的工作流程是

1. 读取环境变量指定的路径加载数据
2. 读取超级参数用于训练（可选）
3. 训练完成后，将模型保存到约定的目录中

TI 会运行用户的训练脚本，建议将启动训练的入口代码放到 main 方法中（ if__name__== '__main__' ）

构造 Estimator

```
estimator = PyTorch(entry_point='train.py',
role=role,
framework_version='1.1.0',
train_instance_count=1,
train_instance_type='TI.SMALL2.1core2g',
source_dir='path/to/code',
# available hyperparameters: emsize, nhid, nlayers, lr, clip, epochs, batch_size,
# bptt, dropout, tied, seed, log_interval
hyperparameters={
'epochs': 1,
'tied': True
})
```

调用 fit 方法

```
estimator.fit({'training': 'cos://path/to/input'})
```

必须参数

- inputs： 存储训练数据集的 COS 路径，可以采用以下两种数据结构
  - str：例如 cos://my-bucket/my-training-data，COS URI，表示数据集的路径
  - dict[str, str]：例如{'train': 'cos://my-bucket/my-training-data/train', 'test': 'cos://my-bucket/my-training-data/test'}，可以指定多个通道的数据集

可选参数

- wait (bool)：默认为 True，是否阻塞直到训练完成。如果设置为 False，fit立即返回，训练任务后台异步执行，后面仍可通过attach方法附加。
- logs (bool)：默认为 True，是否打印训练任务产生的日志。只有在wait为 True 时才生效。
- job_name (str)：训练任务名称。如果未指定，则 Estimator 将根据训练镜像名和时间戳生成一个默认名字。

工作流程

调用 fit 方法启动训练任务后，TI 后台会执行以下操作：

- 启动train_instance_count数量的train_instance_type对应的算力

在每台实例上，执行以下步骤

1. 拉取预置 Pytorch 镜像启动容器

2. 下载训练数据集
3. 设置训练相关的环境变量
4. 开始训练

Pytorch 分布式训练任务

最近更新时间：2020-06-29 15:22:44

操作场景

本文档将向您介绍如何使用 TI SDK 实现 Pytorch 分布式训练。

使用 Horovod 训练

Horovod 是一个基于 MPI 的分布式训练框架，容器内设置 MPI 环境并执行 mpirun 命令，可以运行任何 Horovod 训练脚本。

在 Estimator 中使用 distributions 表示这是一个分布式训练任务，支持的 mpi 参数有：

- enabled (bool)：如果设置为 True，则设置 MPI 并执行 mpirun 命令。
- processes_per_host (int)：MPI 应在每台实例上启动的进程数。请注意，这个值需不大于实例上的 GPU 卡数。
- custom_mpi_options (str)：可以在此字段中传递任何 mpirun 支持的参数选项，TI 执行的 mpirun 时会附带此参数，以启动分布式 horovod 训练。

在下面的示例中，我们创建了一个使用2台单V100卡机器的分布式训练任务。

```
ti_session = session.Session()

role = "TIONE_QCSRole"

inputs = ti_session.upload_data(path='distributed/data', key_prefix="data/pytorch_dist")
print(inputs)

estimator = PyTorch(role=role,
framework_version='1.1.0',
train_instance_count=2,
train_instance_type='TI.GN10X.2XLARGE40.1xV100',
source_dir='distributed/code',
entry_point="horovod_pytorch_example.py",
distributions={
'mpi': {
'enabled': True,
```

```
'processes_per_host': 1,
'custom_mpi_options': '--NCCL_DEBUG INFO'
}
})
estimator.fit({'training': inputs})
```

使用自定义镜像训练模型

最近更新时间：2020-12-14 11:43:05

操作场景

TI 平台预置了 TensorFlow、Pytorch 等流行的机器学习框架，方便用户快速使用 SDK 提交训练任务。除了平台预置的框架和算法外，TI 还允许用户打包自己的算法并上传自定义镜像到平台进行训练。

无论您使用哪种编程语言、系统环境、机器学习框架，及需要安装哪些库依赖，只要您的自定义镜像及训练脚本遵循 TI 的容器规范约束，即可在 TI 平台完成训练。

本文档将向您介绍 TI 自定义镜像需要遵循的一些容器规范约束，再通过几个典型案例向您演示如何制作镜像。

容器规范

启动规范

TI 启动训练容器的方式可以类比为在本地执行 docker run 命令

```
$ docker run {image_name} train
```

其中 train 为 TI 传入的启动参数，容器入口（EntryPoint）需遵循以下启动规范：

- 镜像未指定 EntryPoint 情况
  train 即为启动命令，系统路径$PATH中必须包含 train 的命令
- 镜像指定 EntryPoint 情况
  train 为传入 EntryPoint 命令的第一个参数，EntryPoint 可以选择解析这个参数或忽略

目录规范

TI 在启动训练容器前，会做一些准备工作，包括

1. 在容器内/opt/ml/下新建多个工作目录，详见下表
2. 拉取训练数据到输入数据目录(/opt/ml/input/data/)
3. 落地训练需要的超级参数配置和资源配置到输入配置目录中(/opt/ml/input/config/)

自定义镜像的训练脚本只需要去这些目录下读取训练数据和配置，训练完后将模型文件写入到模型目录，checkpoint 和训练日志写入到输出目录中，TI 后台会自动将模型和输出目录的文件自动上传到用户指定的 COS 存储桶中。

下表是 TI 自动创建的工作目录和文件

名称	容器内目录	备注
输入目录	/opt/ml/input/	存放训练输入相关的目录
输入配置目录	/opt/ml/input/config/	存放训练相关的配置文件
超级参数文件	/opt/ml/input/config/hyperparameters.json	文件为 JSON 格式，读取值是为 string 格式
资源配置文件	/opt/ml/input/config/resourceConfig.json	分布式训练时的网络布局描述
输入数据目录	/opt/ml/input/data/	存放训练数据的目录
模型目录	/opt/ml/model/	模型文件将被自动压缩为 tar 包格式上传到 COS
输出目录	/opt/ml/output/	存放训练过程的 checkpoint 及日志文件，训练失败时将把 FailReason 写进 failure 文件

目录结构：

```
/opt/ml
├── input
│   ├── config
│   │   ├── hyperparameters.json
│   │   └── resourceConfig.json
│   └── data
│       └── user input data
├── model
│   └── user model files
└── output
    └── failure
```

典型案例 Scikit-learn 决策树

下面的例子实现了一个 Scikit-learn 决策树的自定义镜像，例子源码可以在 sdk example 中找到。

```
ti-python-sdk/example/scikit/decision-trees
```

制作镜像

Dockerfile所在目录结构：

```
├── Dockerfile
├── build_and_push.sh       // 构建镜像并 push 镜像到镜像仓库的脚本
├── sources.list            // 建议替换成腾讯云 apt 源
├── pip.conf                // 建议替换成腾讯云 pip 源
└── decision_trees
    └── train               // 训练脚本，EntryPoint，此处内容为 Python 的格式
```

Dockerfile 文件：

```
FROM ubuntu:16.04

COPY sources.list /etc/apt/

RUN apt-get -y update && apt-get install -y --no-install-recommends \
wget \
python \
nginx \
ca-certificates \
&& rm -rf /var/lib/apt/lists/*

COPY pip.conf /etc/

RUN wget https://bootstrap.pypa.io/get-pip.py && python get-pip.py && \
pip install numpy==1.16.2 scipy==1.2.1 scikit-learn==0.20.2 pandas==0.24 flask gevent gun
icorn && \
(cd /usr/local/lib/python2.7/dist-packages/scipy/.libs; rm *; ln ../../numpy/.libs/* .) && \
rm -rf /root/.cache

# Update PATH so that the train program is found when the container is invoked.
```

```
ENV PYTHONUNBUFFERED=TRUE
ENV PYTHONDONTWRITEBYTECODE=TRUE
ENV PATH="/opt/program:${PATH}"

# Set up the program in the image
COPY decision_trees /opt/program
WORKDIR /opt/program
```

执行 build_and_push.sh 文件制作和上传镜像到腾讯云容器服务镜像仓库。

```
$ build_and_push.sh decision_trees
```

提交训练

example.py：

```
from __future__ import absolute_import, print_function

from ti import session
from ti.estimator import Estimator

def test_scikit_decision_trees_training():
try:
print("Set up ti environment")
ti_session = session.Session()

role = "TIONE_QCSRole"

print("Upload the data to a COS bucket")
inputs = ti_session.upload_data(path="data", key_prefix="data/scikit/decision-trees")
print(inputs)

print("Start training on ti")
tree = Estimator(image_name="ccr.ccs.tencentyun.com/ti_private/decision-trees:latest",
role=role,
train_instance_count=1,
```

```
train_instance_type='TI.LARGE8.4core8g')
tree.fit(inputs)
except Exception as e:
print(e)

if __name__ == '__main__':
test_scikit_decision_trees_training()
```

运行 example.py 提交训练任务：

```
$ python example.py
```

典型案例自定义 Tensorflow 镜像

下面的例子实现了一个自定义 Tensorflow 镜像，使用 cifar10 数据集作为训练集，例子源码可以在 sdk example 中找到

```
ti-python-sdk/example/cifar10
```

制作镜像

Dockerfile：

```
FROM tensorflow/tensorflow:1.8.0-py3

# Update PATH so that the train program is found when the container is invoked.
ENV PATH="/opt/program:${PATH}"

COPY /cifar10 /opt/program

WORKDIR /opt/program
```

通过执行 build_and_push.sh 文件制作和上传镜像到腾讯云容器服务镜像仓库

```
$ build_and_push.sh tensorflow1.8-py3-cifar10
```

制作完的镜像名称为：ccr.ccs.tencentyun.com/ti_private/tensorflow1.8-py3-cifar10:latest

准备 CIFAR-10 数据集

执行以下命令下载 CIFAR-10 数据集到 cifar-10-data 目录，并转换成 TFRecords 格式：

```
$ python utils/generate_cifar10_tfrecords.py --data-dir=<your workspace>/cifar-10-data
```

```
cifar-10-data/
├── eval.tfrecords
├── train.tfrecords
└── validation.tfrecords
```

提交训练

example.py：

```
from __future__ import absolute_import, print_function

from ti import session
from ti.estimator import Estimator


def test_cifar10_training():
try:
print("Set up ti environment")
ti_session = session.Session()

role = "TIONE_QCSRole"

print("Upload the data to a COS bucket")
inputs = ti_session.upload_data(path="cifar-10-data/", key_prefix="/data/tensorflow-cifar10")
print(inputs)

print("Start training on ti")
tree = Estimator(image_name="ccr.ccs.tencentyun.com/ti_private/tensorflow1.8-py3-cifar10:la
test",
role=role,
```

```
train_instance_count=1,
train_instance_type='TI.LARGE8.4core8g',
hyperparameters={'train-steps': 100})
tree.fit(inputs)
except Exception as e:
print(e)

if __name__ == '__main__':
test_cifar10_training()
```

运行 example.py 提交训练任务：

```
$ python example.py
```

使用文件系统提交训练任务

最近更新时间：2020-07-30 14:44:37

操作场景

腾讯云的文件系统（Cloud File Storage，CFS）提供了标准的 NFS 及 CIFS/SMB 文件系统访问协议。使用 CFS，您的训练任务可以节省数据从远程下载的时间。 您上传数据至 CFS 文件系统的云服务器需满足以下条件：

1. 云服务器已安装 NFS 客户端。
2. 云服务器与 CFS 处于同一个私有网络环境中。

单击查看 CFS 操作指引。

本文档将向您介绍如何使用文件系统作为输入源提交任务训练。

操作步骤

准备 CFS 训练数据

在 TI 中使用 CFS 训练数据作为输入源，与使用 COS 训练数据作为输入源类似，只需少量的修改训练脚本进行适配。CFS 文件系统主要提供如下参数：

1. 文件系统 ID。
2. 文件系统挂载目录。
3. 读写方式。

文件系统详情信息，单击查看控制台。

```
from ti.session import FileSystemInput

# 文件系统的ID
file_system_id = 'cfs-xxxxxxx'

# 指定文件系统的挂载目录
file_system_directory_path = '/data/input'

# 文件系统类型，目前取值为cfs
file_system_type = "cfs"

# 文件系统读写模式，分为只读和读写方式，取值为rw和ro
```

```
file_system_access_mode = 'rw'
inputs = FileSystemInput(file_system_id=file_system_id,
directory_path=file_system_directory_path,
file_system_type = file_system_type,
file_system_access_mode=file_system_access_mode)
```

构造 Estimator

```
tf_estimator = TensorFlow(role=role,
train_instance_count=1,
train_instance_type='ML.GN8.3XLARGE112',
py_version='py3',
script_mode=True,
framework_version='1.14.0',
entry_point='train.py',
subnet_id = "subnet-yyyyyy",
source_dir='gpu/code')
```

subnet_id 参数

subnet_id：子网 ID。提交的训练任务所处私有网络需与 CFS 文件系统所处私有网络保持一致。

调用 fit 方法

```
tf_estimator.fit(inputs)
```

文件系统参数

- inputs：采用 CFS 文件系统时可以采用以下两种数据结构。
  - FileSystemInput：表示 CFS 数据集数据结构。
  - dict[str, FileSystemInput]：例如{'train': FileSystemInput, 'test': FileSystemInput}，可以支持两个 CFS 数据集的字典结构。

使用 Tensorboard 查看训练模型

最近更新时间：2022-01-10 16:28:50

操作场景

TI 平台预置了 TensorFlow、Pytorch 等流行的机器学习框架，方便用户快速使用 SDK 提交训练任务。除了平台预置的框架和算法外，TI 还允许用户打包自己的算法并上传自定义镜像到平台进行训练。

无论您使用哪种编程语言、系统环境、机器学习框架，及需要安装哪些库依赖，只要您的自定义镜像及训练脚本遵循 TI 的容器规范约束，即可在 TI 平台完成训练。

本文档将向您介绍 TI 自定义镜像需要遵循的一些容器规范约束，再通过几个典型案例向您演示如何制作镜像。

容器规范

启动规范

TI 启动训练容器的方式可以类比为在本地执行 docker run 命令

```
$ docker run {image_name} train
```

其中 train 为 TI 传入的启动参数，容器入口（EntryPoint）需遵循以下启动规范：

- 镜像未指定 EntryPoint 情况
  train 即为启动命令，系统路径$PATH中必须包含 train 的命令
- 镜像指定 EntryPoint 情况
  train 为传入 EntryPoint 命令的第一个参数，EntryPoint 可以选择解析这个参数或忽略

目录规范

TI 在启动训练容器前，会做一些准备工作，包括

1. 在容器内/opt/ml/下新建多个工作目录，详见下表
2. 拉取训练数据到输入数据目录(/opt/ml/input/data/)
3. 落地训练需要的超级参数配置和资源配置到输入配置目录中(/opt/ml/input/config/)

自定义镜像的训练脚本只需要去这些目录下读取训练数据和配置，训练完后将模型文件写入到模型目录，checkpoint 和训练日志写入到输出目录中，TI 后台会自动将模型和输出目录的文件自动上传到用户指定的 COS 存储桶中。

下表是 TI 自动创建的工作目录和文件

名称	容器内目录	备注
输入目录	/opt/ml/input/	存放训练输入相关的目录
输入配置目录	/opt/ml/input/config/	存放训练相关的配置文件
超级参数文件	/opt/ml/input/config/hyperparameters.json	文件为 JSON 格式，读取值是为 string 格式
资源配置文件	/opt/ml/input/config/resourceConfig.json	分布式训练时的网络布局描述
输入数据目录	/opt/ml/input/data/	存放训练数据的目录
模型目录	/opt/ml/model/	模型文件将被自动压缩为 tar 包格式上传到 COS
输出目录	/opt/ml/output/	存放训练过程的 checkpoint 及日志文件，训练失败时将把 FailReason 写进 failure 文件

目录结构：

```
/opt/ml
├── input
│   ├── config
│   │   ├── hyperparameters.json
│   │   └── resourceConfig.json
│   └── data
│       └── user input data
├── model
│   └── user model files
└── output
    └── failure
```

典型案例 Scikit-learn 决策树

下面的例子实现了一个 Scikit-learn 决策树的自定义镜像，例子源码可以在 sdk example 中找到。

```
ti-python-sdk/example/scikit/decision-trees
```

制作镜像

Dockerfile所在目录结构：

```
├── Dockerfile
├── build_and_push.sh       // 构建镜像并 push 镜像到镜像仓库的脚本
├── sources.list            // 建议替换成腾讯云 apt 源
├── pip.conf                // 建议替换成腾讯云 pip 源
└── decision_trees
    └── train               // 训练脚本，EntryPoint，此处内容为 Python 的格式
```

Dockerfile 文件：

```
FROM ubuntu:16.04

COPY sources.list /etc/apt/

RUN apt-get -y update && apt-get install -y --no-install-recommends \
wget \
python \
nginx \
ca-certificates \
&& rm -rf /var/lib/apt/lists/*

COPY pip.conf /etc/

RUN wget https://bootstrap.pypa.io/get-pip.py && python get-pip.py && \
pip install numpy==1.16.2 scipy==1.2.1 scikit-learn==0.20.2 pandas==0.24 flask gevent gun
icorn && \
(cd /usr/local/lib/python2.7/dist-packages/scipy/.libs; rm *; ln ../../numpy/.libs/* .) && \
rm -rf /root/.cache

# Update PATH so that the train program is found when the container is invoked.
```

```
ENV PYTHONUNBUFFERED=TRUE
ENV PYTHONDONTWRITEBYTECODE=TRUE
ENV PATH="/opt/program:${PATH}"

# Set up the program in the image
COPY decision_trees /opt/program
WORKDIR /opt/program
```

执行 build_and_push.sh 文件制作和上传镜像到腾讯云容器服务镜像仓库。

```
$ build_and_push.sh decision_trees
```

提交训练

example.py：

```
from __future__ import absolute_import, print_function

from ti import session
from ti.estimator import Estimator

def test_scikit_decision_trees_training():
try:
print("Set up ti environment")
ti_session = session.Session()

role = "TIONE_QCSRole"

print("Upload the data to a COS bucket")
inputs = ti_session.upload_data(path="data", key_prefix="data/scikit/decision-trees")
print(inputs)

print("Start training on ti")
tree = Estimator(image_name="ccr.ccs.tencentyun.com/ti_private/decision-trees:latest",
role=role,
train_instance_count=1,
```

```
train_instance_type='TI.LARGE8.4core8g')
tree.fit(inputs)
except Exception as e:
print(e)

if __name__ == '__main__':
test_scikit_decision_trees_training()
```

运行 example.py 提交训练任务：

```
$ python example.py
```

典型案例自定义 Tensorflow 镜像

下面的例子实现了一个自定义 Tensorflow 镜像，使用 cifar10 数据集作为训练集，例子源码可以在 sdk example 中找到

```
ti-python-sdk/example/cifar10
```

制作镜像

Dockerfile：

```
FROM tensorflow/tensorflow:1.8.0-py3

# Update PATH so that the train program is found when the container is invoked.
ENV PATH="/opt/program:${PATH}"

COPY /cifar10 /opt/program

WORKDIR /opt/program
```

通过执行 build_and_push.sh 文件制作和上传镜像到腾讯云容器服务镜像仓库

```
$ build_and_push.sh tensorflow1.8-py3-cifar10
```

制作完的镜像名称为：ccr.ccs.tencentyun.com/ti_private/tensorflow1.8-py3-cifar10:latest

准备 CIFAR-10 数据集

执行以下命令下载 CIFAR-10 数据集到 cifar-10-data 目录，并转换成 TFRecords 格式：

```
$ python utils/generate_cifar10_tfrecords.py --data-dir=<your workspace>/cifar-10-data
```

```
cifar-10-data/
├── eval.tfrecords
├── train.tfrecords
└── validation.tfrecords
```

提交训练

example.py：

```
from __future__ import absolute_import, print_function

from ti import session
from ti.estimator import Estimator


def test_cifar10_training():
try:
print("Set up ti environment")
ti_session = session.Session()

role = "TIONE_QCSRole"

print("Upload the data to a COS bucket")
inputs = ti_session.upload_data(path="cifar-10-data/", key_prefix="/data/tensorflow-cifar10")
print(inputs)

print("Start training on ti")
tree = Estimator(image_name="ccr.ccs.tencentyun.com/ti_private/tensorflow1.8-py3-cifar10:la
test",
role=role,
```

```
train_instance_count=1,
train_instance_type='TI.LARGE8.4core8g',
hyperparameters={'train-steps': 100})
tree.fit(inputs)
except Exception as e:
print(e)

if __name__ == '__main__':
test_cifar10_training()
```

运行 example.py 提交训练任务：

```
$ python example.py
```

任务列表

最近更新时间：2020-07-31 17:13:58

操作场景

本文档向您介绍 TI-ONE 任务列表的概念和基本操作。

操作说明

- 任务列表汇集了 TI-ONE 平台内的所有类型的任务，包括工作流任务和 SDK 任务。
- 您可以通过任务列表查看每个任务的运行状态、资源使用情况与训练时长。
- 您可以基于历史运行数据进行查看比对，调整资源配置。
- 对于运行中的任务您可以一键终止。

模型仓库

最近更新时间：2021-12-22 12:10:34

操作场景

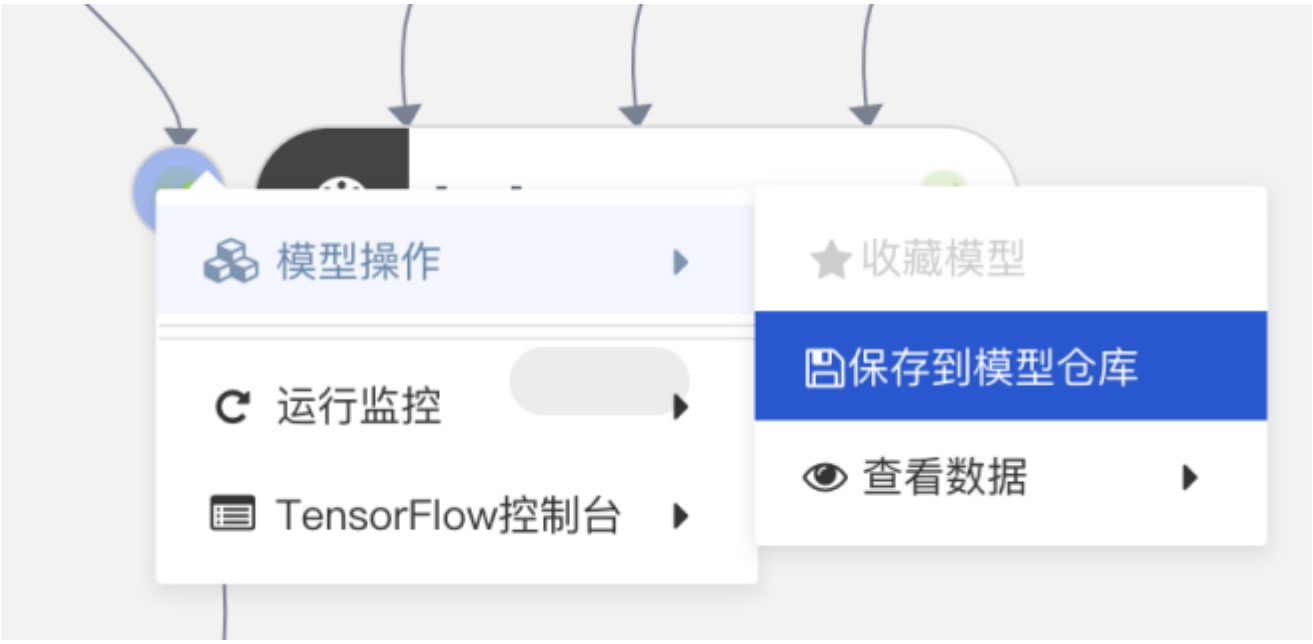
模型仓库页面用于管理所有您保存的模型。您可以：

- 对每个模型进行版本控制和切换。
- 通过自动生成的 TAG 对模型进行筛选。
- 创建基于模型的离线批量预测作业。

操作步骤

保存模型

在工作流页面运行成功后，右键单击**模型**组件，选择**保存**到模型仓库。输入**模型名称**，您的模型将保存至**模型仓库**。您可以对模型进行离线批量预测和下载等操作。



模型仓库

模型仓库用于管理所有您保存的模型，您可以进行如下操作：

- **版本控制和切换**：单击【模型名称】旁边的版本号，即可对每个模型进行版本控制和切换。
- **高级筛选**：单击【高级筛选】，即可通过自动生成的 TAG 对模型进行筛选。
- **离线批量预测**：创建基于模型的离线批量预测作业。更多详情请参考离线批量服务。
- **模型下载**：单击即可前往 COS 下载模型。
- **启动模型服务**：单击即可启动模型服务，更多详情请参考腾讯云 TI 平台 TI-EMS 服务。

离线批量服务

最近更新时间：2021-03-25 14:57:23

启动离线服务

1. 在 模型仓库 页面，选择相应的模型单击**离线批量预测**。
2. 在跳转的离线批量预测页面对**作业类型**等进行设置，并对预处理环节进行编辑。如果删除预处理节点，请重新在组件间完成连线。
3. 成功开启作业后，您可以在**离线批量服务**页面对所有作业进行查看和管理。

管理离线批量服务

- 离线批量服务 页面将展示所有作业的名称、类型、版本、运行时长、创建时间与状态等信息。
- 您可以通过页面上方弹框对作业进行搜索和筛选。
- 作业的运行状态包括：运行中、已完成、失败、已终止：
  - 对于**运行中**的服务，单击状态下的运行中，即可打开任务流视图，查看离线预测作业的运行详情。
  - 对于**已完成**的任务，单击快照即可查看详情。
- 对于周期型服务，可以单击定时周期修改服务运行周期。

自动建模（AutoML）

最近更新时间：2021-12-22 12:10:21

半自动建模

1. 登录 腾讯云 TI 平台 TI-ONE 控制台，新建工程和任务流，进入画布，拖拽所选组件置于画布中。
2. 单击组件，单击画布右侧参数配置区上方工具栏的【自动调参】，启动算法自动调参模式。

3. 配置算法参数，确定模型评估方法。

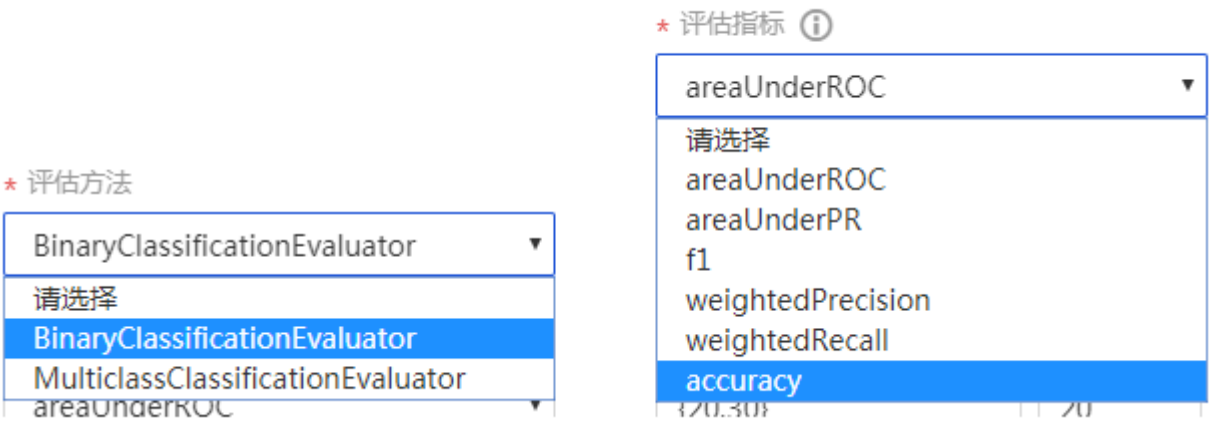
3.1 调参算法目前 TI-ONE 支持贝叶斯调参、网格调参、随机调参三种调参方式：

- 贝叶斯调参：
  - 给定可调参数的所在范围和初始值。
  - 确定代理函数（Gaussian Process）和采集函数（Expected Improvement）。
  - 根据采集函数获得在代理函数上表现最佳的超参数组，作为下一个采样点。
  - 将上个步骤得到的采样点用于模型训练并更新代理函数。
  - 重复以上步骤，直到达到最大迭代次数或时间，根据评估方法对模型进行排序，保存所需模型。
- 网格调参：
  - 给定可调参数的所在范围和初始值。
  - 循环遍历每个可调参数的候选值，获得对应参数组进行模型训练。
  - 根据评估方法进行排序，保存所需模型。
- 随机调参：
  - 给定可调参数的所在范围和初始值。
  - 从每个参数的范围内随机选取一个值。
  - 将随机选取的参数值组成一组参数进行模型训练。
  - 将2、3步重复 m 次，得到 m 个模型，根据评估方法进行排序，保存所需模型。

对于某个算法的可调参数，目前支持离散和连续两种类型，离散参数以集合的形式完整给出该参数的所有候选值，连续参数给出取值范围。以决策树算法为例，这里的 maxDepth、maxBins 和 minInstancesPerNode 为离散参数，minInfoGain 则为连续参数。

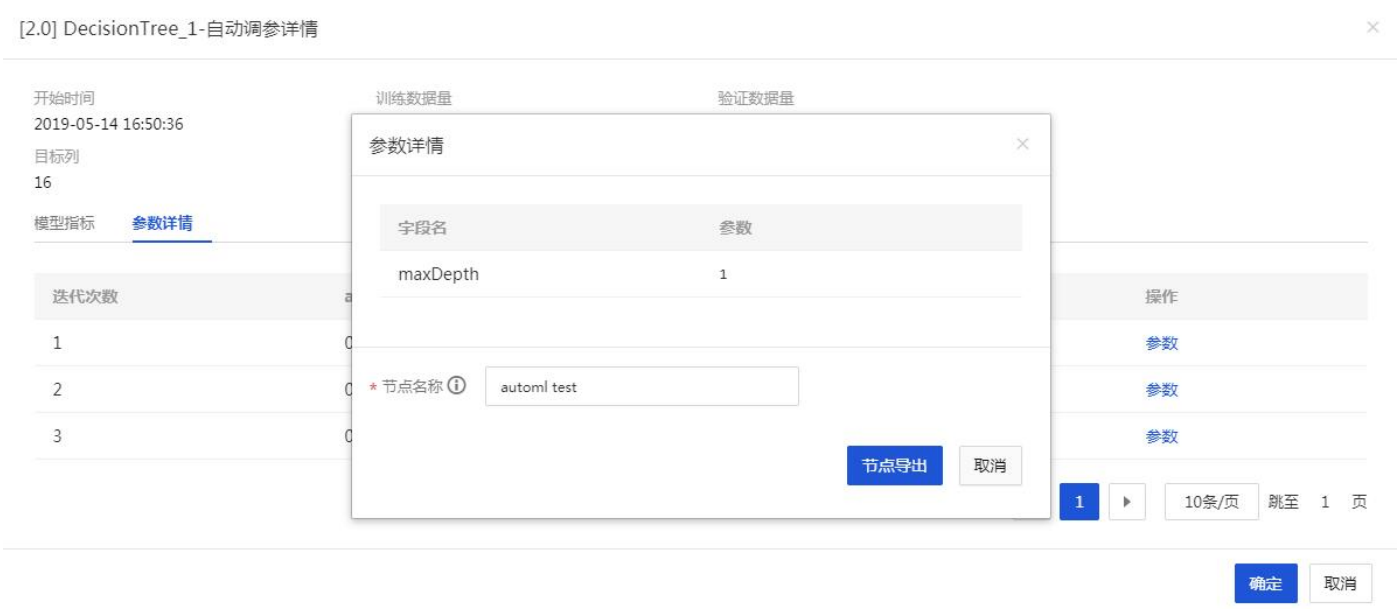
3.2 评估方法和评估指标

评估方法和指标为多次迭代获得的模型提供排序和选择标准。分类、回归和聚类算法有各自的评估方法和指标，具体请依据训练数据和算法类型进行选择。以决策树分类算法为例，评估方法有二分类和多分类两种，评估指标有 AUC 和 Accuracy 等。如果选择Accuracy作为评估指标，则在多个生成的模型中，选择 Accuracy 最高的模型作为最后的输出模型。



配置完所有的算法参数后，单击工具栏【运行】，启动任务运行。

4. 对于运行效果良好的迭代，可将其**作为节点导出**至当前画布，单击【参数】，为**节点命名**并单击【节点导出】即可。



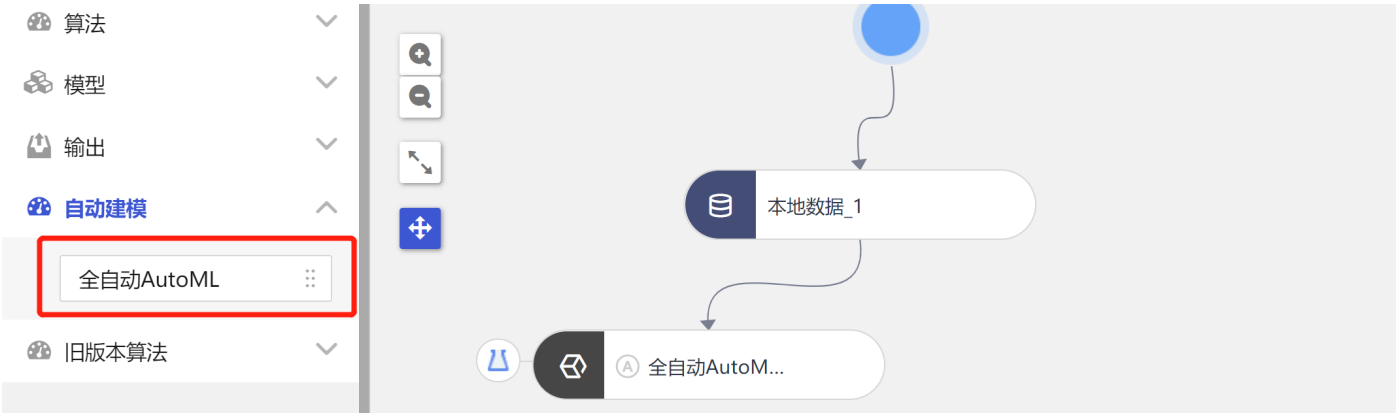
导出后的节点默认直接与数据源进行连接，节点参数将默认填充迭代的具体值。

全自动建模

在不通过人为来设定参数的情况，通过某些学习机制，让系统智能地去调节这些超参数，让整个机器学习流程做到全自动化。

1. 登录【腾讯云 TI 平台 TI-ONE 】，建立一个**全自动调参**的任务。
   此步骤中需要先设置好数据源，将左侧**自动建模（全自动AutoML）节点**拖入至画布产生连接，便可启动一个

AutoML的任务。



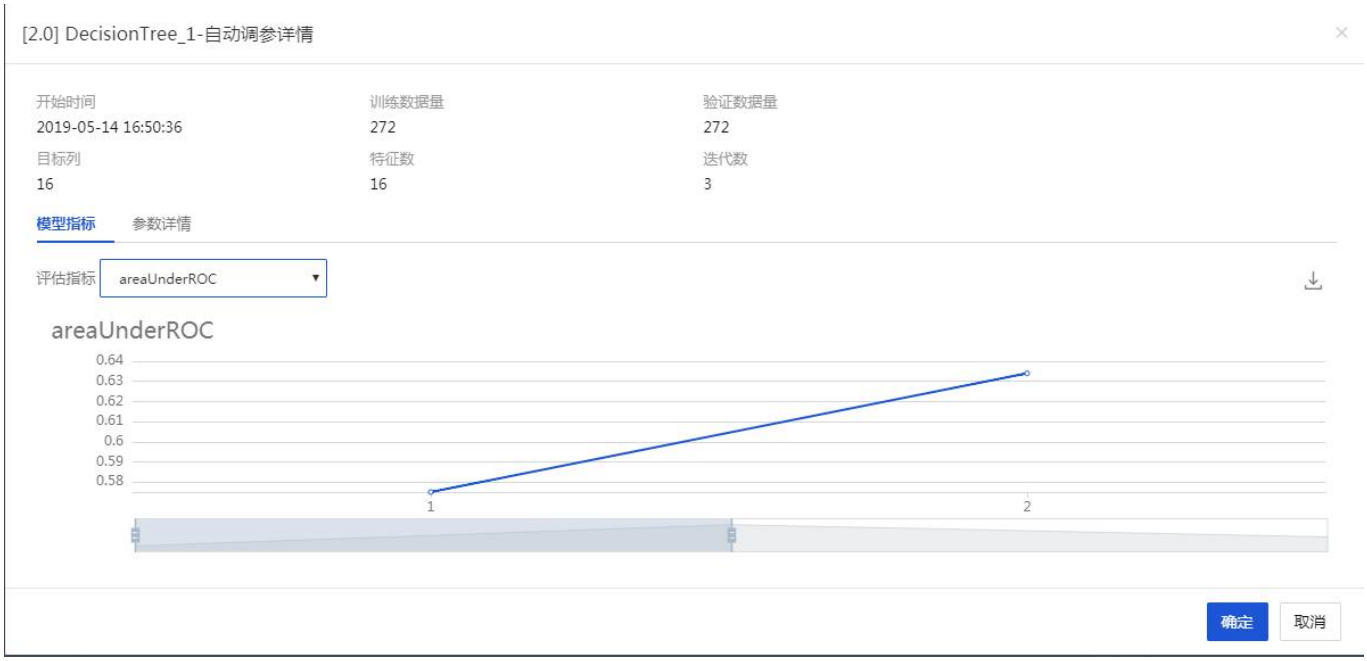
2. AutoML 节点的参数栏中，只需指定输入路径，进行简单的参数设置（迭代次数/迭代时间等）和资源参数设置，便可完成建模流程。

查看自动建模参数详情

用户可进行半自动与全自动建模参数的查看，方法如下：

1. 单击组件，通过右键算法节点中的【自动调参详情】入口进行查看。

2. 自动调参详情页面：
   - 上方展示的是算法的基本运行信息，如训练/验证数据量、开始运行时间、目标列和特征列等。
   - 下方分两个模块，**模型指标**和**参数详情**。用户可以选择不同的评估指标来查看，**模型指标**模块主要以曲线呈现每一迭代的 AUC 效果。



   - 参数详情模块展示的是每一轮迭代的信息、AUC 值、运行状态及对应迭代的参数，单击参数展示被调的参数这一轮的具体值。