ALIBABA CLOUD

阿里云

E-MapReduce

Data Science集群

文档版本：20201120

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.概述

Data Science节点底层基于EMR最新软件栈，支持阿里云机器学习平台团队提供的人工智能，包含了机器学习算法平台Alink、向量计算引擎FAISS、深度学习框架TensorFlow和PyTorch等。

适用对象

DataScience节点适用对象：

- 基于开源大数据体系的用户。
- 基于阿里云人工智能技术完成智能推荐和智能风控等解决方案的用户。

集群创建

在E-MapReduce控制台，创建Data Science集群，详情请参见创建集群。

1. 在创建集群的软件配置阶段，选择正确的可用区和EMR版本。
   - 可用区：目前仅支持部分可用区，具体以实际购买页面为准。
   - 产品版本：目前仅支持最新EMR的版本。
   - 可选服务：建议勾选TensorFlow。

   
2. 在创建集群的硬件配置阶段，需要创建VPC、交换机和安全组。新建安全组时需要跳转到ECS控制台上创建。

   

   需要开启8443端口，以便于访问相关组件的UI页面，详细步骤请参见设置安全组访问。
3. 在基础配置阶段，需要添加Knox账号，用于登录Knox服务。

   

   添加的Knox账号即阿里云RAM子账号，添加Knox详细步骤请参见用户管理。

集群查看

集群创建成功后，可以在**集群管理**页面，查看集群服务运行状态。

2.PAI-Alink

Alink是基于Flink或Blink的通用算法平台。本文介绍如何通过在E-MapReduce控制台上配置及使用PAI-Alink。

前提条件

- 已创建E-MapReduce的Data Science集群。详情请参见创建集群。
- 创建Knox账号，详情请参见用户管理。
- 已开启8443端口，详情请参见设置安全组访问。

访问PAI-Alink

1. 登录阿里云E-MapReduce控制台。
2. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
3. 单击上方的**集群管理**页签。
4. 在**集群管理**页面，单击相应集群所在行的**详情**。
5. 配置Flink-VVP。
   i. 在左侧导航栏中，单击**访问链接与端口**。
   ii. 单击**Flink-Vvp UI**所在行的链接。
   iii. 在登录页面，输入已创建的Knox账号的**用户名**和**密码**，单击**登录**。
   iv. 单击**Administration > Deployment Targets**。
   v. 单击**Add Deployment Target**。
   vi. 在**Add Deployment Target**对话框中，设置**Deployment Target Name**、设置**Yarn Queue**为default，单击**OK**。
6. 配置Alink。
   i. 在E-MapReduce控制台左侧导航栏中，单击**访问链接与端口**。
   ii. 单击**PAI-Alink UI**所在行的链接。
   iii. 在左侧导航栏，单击**设置**。
   iv. 从**切换运行的Deployment Target**列表，选择已创建的Deployment Target，单击**Submit**。

使用Alink

在Alink产品首页，可以单击对应模板下方的**从模板创建**，体验首页模板。

使用过程中，您可以选择需要的组件并拖拽到画布上。以首页的评分卡功能模板为例，读取本地HDFS数据，您只需要在读取CSV组件处配置好本地HDFS数据的路径即可。

3.Alink调度

本文介绍如何使用Alink调度作业。

前提条件

- 已新建Data Science集群，详情请参见创建集群。
- 已创建项目，详情请参见项目管理。
- 已配置Alink，详情请参见PAI-Alink。
- 本地安装了PuTTY和文件传输工具（SSH Secure File Transfer Client）。

获取任务脚本

1. 登录阿里云E-MapReduce控制台并进入Alink产品首页，详情请参见PAI-Alink。
2. 在Alink产品首页，单击对应模板下方的**从模板创建**。本示例以评分卡功能为例。
3. 在**新建示例实验**对话框中，输入**实验名**，单击**确定**。
4. 单击下面的**部署 > 生成部署脚本**。**部署脚本**对话框中，展示了详细的脚本信息。

   > **说明** 生成的脚本即为该实验的可执行脚本。

5. 拷贝脚本信息至本地，保存为*script.py*文件。

部署脚本配置

1. 新建配置文件*config.txt*。配置内容如下。

```
userId=default
alinkServerEndpoint=http://127.0.0.1:9301
hadoopHome=/usr/lib/hadoop-current
hadoopUserName=hadoop
token=ZSHTIeEkwrtZJJsN1ZZmCJJmr5jaj1wO
```

2. 使用文件传输工具，上传*config.txt*和*script.py*至Data Science集群Master节点的根目录下。
3. 登录Master节点，详情请参见使用SSH连接主节点。
4. 执行如下命令，查看文件。

配置调度任务

1. 登录阿里云E-MapReduce控制台。
2. 在顶部菜单栏处，根据实际情况选择地域（Region）和资源组。
3. 单击上方的**数据开发**页签。
4. 新建任务执行集群。
   i. 单击新建项目所在行的**作业编辑**。
   ii. 单击上方的**项目管理**页签。

iii. 在左侧导航栏，单击**集群设置**。

iv. 单击右上角的**添加集群**。

v. 在**添加集群**对话框，从**选择集群**列表中，选择已创建的Data Science集群，单击**确定**。

5. 新建作业。

i. 单击上方的**数据开发**页签。

ii. 新建Shell类型作业。新建作业详情请参见Shell作业配置。

iii. 输入作业内容。本示例作业内容如下。

```
sudo alinkcmd run -c /root/config.txt -f /root/script.py
```

6. 设置作业。

i. 单击右上角的**作业设置**。

ii. 在**作业设置**对话框，单击**高级设置**。

iii. 在**模式**区域，从**提交节点**列表中，选择**在Header/Gateway节点提交**。

7. 运行作业。

i. 单击上方的**保存**。

ii. 单击上方的**运行**。

iii. 在**运行作业**对话框中，设置执行集群为已创建的Data Science集群，单击**集群**。

4.Faiss-Server

Faiss-Server基于Faiss的开源实现，是一个使用gRPC协议提供向量检索的服务。

支持加载的向量文件格式

文件格式通过后缀来区分，支持的文件格式如下：

- *.fvecs*（Fvecs格式文件）
- *.svm*（Libsvm格式文件）

  Libsvm要求的格式如下。

  ```
  21187279 1:0.2663046344870018 2:0.36652042181588923 3:1.0686633708024278 4:0.48038935355720136 5:0.25852895390543884 6:0.3548201486996048 7:0.5256999599627583 8:0.6950687558969181 9:0.6783058946157461 10:0.22776047265501018
  21264640 1:0.3939700179792862 2:0.2754414668803289 3:0.9439534329739017 4:0.40100161008614665 5:0.4995636031244379 6:0.013824724791385053 7:0.5587276328436032 8:0.5785080421720411 9:0.2784678162956546 10:0.5387681136371216
  ```

  通用的形式标识为 `tagid 1:xx 2:xx 3:xx` 。其中 `tagid` 可以是任意的字符串，例如在推荐的场景中，通常设置为itemid。数字 `1` 是向量的数据，通常从1开始。 `xx` 是维度的具体值。

- *.index*（Faiss生成的Index文件）

  如果您直接加载数据量大的Index文件，创建索引会非常慢。但是在线下通过Libsvm格式的文件build成Faiss的Index文件，再通过在线Faiss Server加载Index文件，加载速度非常快。

> **说明** 文件命名时，请保持文件格式与后缀一致。在部分场景下生成OSS文件时，OSS会在后缀名添加随机的字符，此时并不影响文件的加载，只需要与原始的后缀保持一致即可。

加载本地的向量文件

示例文件：article_word2vec.svm

加载本地的向量文件代码示例如下。

```
docker run -it -p 9000:9000 -v /home/bruceding.jing:/index registry.cn-beijing.aliyuncs.com/pai-recommend/faiss-server:0.0.2 --index_source=/index/article_word2vec.svm --index_params=Flat
// 以下是docker的输出内容。
2020-05-01 15:18:01,134 INFO src/faiss_parameter.cc:39 index source:/index/article_word2vec.svm,index params:Flat
2020-05-01 15:18:01,134 INFO src/faiss_index.cc:26 init index
2020-05-01 15:18:01,377 INFO src/faiss_index.cc:119 size:10907,dimension:10
2020-05-01 15:18:01,377 INFO src/faiss_index.cc:120 index_params=Flat
2020-05-01 15:18:01,377 INFO src/main.cc:55 create index success
2020-05-01 15:18:01,377 INFO src/faiss_server.cc:23 Server listening on 0.0.0.0:9000
```

运行命令后，输出相关的日志信息，启动gRPC server成功，监听9000的端口。

示例中相关参数描述如下。

参数	描述
-p	使用docker的 `-p` 参数进行端口映射。 本示例中为9000，表示Faiss-Server的默认监听端口。
registry.cn-beijing.aliyuncs.com/pai-recommend/faiss-server:0.0.2	Faiss-Server提供的docker镜像地址。
--index_source	指定向量文件的具体路径。
--index_params	指定Faiss Index构建的参数，本示例中使用了 `Flat` 搜索的方式构建索引。 Faiss-Server使用Faiss的index_factory的形式构建索引。 `--index_params` 参数详情请参见Faiss indexes。

加载OSS中的向量文件

通常，向量文件生成在OSS中，Faiss-Server支持直接从OSS的向量文件进行索引构建。

```
docker run -it -p 9000:9000 registry.cn-beijing.aliyuncs.com/pai-recommend/faiss-server:0.0.2 --index_params=IVF256,Flat --search_params=nprobe=128 --accessid=xxx --accesskey=xxx --endpoint=oss-cn-beijing.aliyuncs.com --bucket_name=faiss-server --object_name=demo_data/article_word2vec.svm
```

示例中相关参数描述如下。

参数	描述
--index_params	使用的是 `IVF256,Flat` ，使用了聚簇索引的方式，分成了256份。
--search_params	查询向量最相近的128个聚簇集合。具体的查询参数设置请参见Index IO, cloning and hyper parameter tuning。
--bucket_name	OSS的bucket名称。
--object_name	OSS的向量文件地址。
--endpoint	OSS的访问地址， 这里是外网的地址。在VPC内网的情况下，一定要使用内网地址，节省流量费用，读取速度也会更快。

> **说明** 因为要读取OSS的数据，所以需要设置阿里云账号的AccessKey ID和AccessKey Secret。

您可以执行以下命令，查看Faiss-Server支持的参数列表。

```
docker run registry.cn-beijing.aliyuncs.com/pai-recommend/faiss-server:0.0.2 --help
```

加载OSS中Version检查机制

通常，Faiss-Server是在线的服务，直接加载向量文件耗时长。因此您可以生成Index索引文件，通过Faiss-Server直接加载Index文件节省时间。

部分场景下，向量文件需要定时加载。

您可以生成多个版本的向量文件，通过Version文件记录当前加载的文件。Faiss-Server会异步检查Version文件内容，一旦有变动，会自动加载最新的Index和Idxmap文件。

对应图中步骤描述如下：

- 1：向量召回算法，生成item的向量文件，存储至OSS文件中。
- 2和3：Faiss-Server读取OSS的向量文件，进行构建索引。

  Version文件包含版本内容，实质是Index和Idxmap两个文件地址：
  - Index文件是构建出来的Faiss Index文件。
  - Idxmap文件是映射表，是tagid和Index具体向量位置的映射。
- 4：在线的Faiss-Server服务一直是启动状态，会监听Version文件内容。

介绍离线流程中，如何构造Index文件。

1. 构建Index索引文件。

   此示例构建索引，构建完成后，进程退出。因为不涉及gRPC服务，所以docker的 `-p` 参数不需要设置。

   ```
   docker run  -it registry.cn-beijing.aliyuncs.com/pai-recommend/faiss-server:0.0.2  --index_params=IVF256,Flat --search_params=nprobe=128 --accessid=xxx --accesskey=xxxx --endpoint=oss-cn-beijing.aliyuncs.com --bucket_name=faiss-server --object_name=demo_data/article_word2vec.svm --version_name=demo_data/versions --action=build_index
   ```

   示例中相关参数描述如下：
   - `--version_name` ： 指定Version文件在OSS上的具体位置。如果Version文件不存在，可以自动生成。
   - `--action` ： build流程设置build_index即可。

   构建成功后，生成以下三个文件。

   

   如果svm的向量文件在本地，执行以下命令，将生成的Index等文件放到OSS中。

   ```
   docker run  -it -v /home/bruceding.jing:/index  registry.cn-beijing.aliyuncs.com/pai-recommend/faiss-server:0.0.2  --index_params=IVF256,Flat --search_params=nprobe=128 --accessid=xxx --accesskey=xxx --endpoint=oss-cn-beijing.aliyuncs.com --bucket_name=faiss-server --index_source=/index/article_word2vec.svm --version_name=demo_data/versions --action=build_index
   ```

> **说明** `-v`：表示把本地的svm文件传给docker。

2. 启动Faiss-Server在线服务。

   您可以重新开启一个终端，模拟启动Faiss-Server的在线服务。

   ```
   docker run -it -p 9000:9000 registry.cn-beijing.aliyuncs.com/pai-recommend/faiss-server:0.0.2 --index_params=IVF256,Flat --search_params=nprobe=128 --accessid=xxxx --accesskey=xxxx --endpoint=oss-cn-beijing.aliyuncs.com --bucket_name=faiss-server --version_name=demo_data/versions --check_version=true
   ```

   示例中相关参数描述如下：

   - `--index_params` 和 `--search_params` 需要和构建的Index文件保持一致。
   - `--version_name`：指定了Version文件在OSS上的具体位置。如果文件不存在，可以自动生成。
   - `--check_version=true`：表示开启Version检查机制。开启Version检查机制，Version文件变更，Faiss-Server才会自动加载。

   启动成功后，回到前一个窗口，再次构建Index索引。构建成功后，如果输出信息类似如下，表示最新的Index加载成功。

   ```
   2020-05-01 17:09:37,478 INFO src/faiss_index.cc:197 start to load
   2020-05-01 17:09:37,567 INFO src/oss_util.cc:30 GetObjectToFile success62
   2020-05-01 17:09:37,568 INFO src/faiss_index.cc:219 localPath=/tmp/20200501170924.index
   2020-05-01 17:09:37,703 INFO src/oss_util.cc:30 GetObjectToFile success535963
   2020-05-01 17:09:37,815 INFO src/oss_util.cc:30 GetObjectToFile success98159
   2020-05-01 17:09:37,817 INFO src/faiss_index.cc:245 load index success
   ```

3. 追加向量文件。

   加载全量的Index文件之后，Faiss-Server还支持加载增量的SVM格式的向量文件。

   ```
   docker run -it -p 9000:9000 registry.cn-beijing.aliyuncs.com/pai-recommend/faiss-server:0.0.2 --index_params=IVF256,Flat --search_params=nprobe=128 --accessid=xxx --accesskey=xxxx --endpoint=oss-cn-beijing.aliyuncs.com --bucket_name=faiss-server --version_name=demo_data/versions --check_version=true --append_dir=demo_data/append
   ```

   `--append_dir`：指定SVM格式增量文件追加目录。

   当您把增量文件放到append_dir后，会看到如下类似的输出信息。

   ```
   2020-05-01 17:24:53,161 INFO src/oss_util.cc:30 GetObjectToFile success2428471
   2020-05-01 17:24:53,357 INFO src/faiss_index.cc:383 append index, file:article_word2vec.svm, size=10907, idx size=10907
   ```

测试向量服务

启动Faiss-Server之后，可以使用如下工具测试。

工具和源码下载：faiss-client-go.tar.gz和faiss-client-java.tar.gz。

```
./faiss-client-go --host "11.158.166.161:9000" --vector "1:0.8254435895816826 2:0.3546776922906237 3:0.149
82954432756015 4:0.4796270382425792 5:0.5478646494350313 6:0.3771617042371068 7:0.5107318036421319
8:0.7005006312566686 9:0.7030542773195687 10:0.692132791047145"  --k 20
```

示例中相关参数描述如下。

参数	描述
--host	指定Faiss-Server的地址，包括IP地址和端口号。
--vector	指定请求的向量数据，和SVM的格式一样，维度也一致。本示例中是10维的数据。
--k	指定返回TopN的大小。 默认是10，本示例中指定为20。

返回信息如下。

> **说明** tagid 指的是SVM文件里的tagid，通常是 itemid 。

在EMR上部署Faiss-server服务

1. 创建EMR的Data Science集群，相关信息请参见概述。默认支持Faiss-Server服务。

   

   > **说明** 镜像*registry.cn-beijing.aliyuncs.com/pai-recommend/faiss-server:0.0.2*已经打包在EMR上，镜像名称为 *faiss-server:1.0.0*。

2. 新建Shell类型的作业来加载向量文件，启动gRPC服务。

   ```
   sudo docker run -d --restart unless-stopped -p 9001:9000  faiss-server:1.0.0  --index_params=IVF256,Flat
   --search_params=nprobe=128 --accessid=xxxx --accesskey=xxx --endpoint=oss-cn-huhehaote-internal.a
   liyuncs.com --bucket_name=faiss-test  --object_name=article_word2vec.svm
   ```

5.AutoML

本文介绍如何使用AutoML，进行超参数（机器学习模型的框架参数）调优。例如超参数，算法的学习率（Learning Rate）和学习率的衰减率（Decay Rate）。

前提条件

- 本地已经准备好可运行的模型训练脚本和AutoML的配置脚本。
- 已安装Python 3和文件传输工具（SSH Secure File Transfer Client）。

操作步骤

本文使用*digits_model.py*（手写自识别的模型训练脚本）和*test.py*（AutoML的配置脚本）。详情请参见digits_model.py和test.py。

1. 在*digits_model.py*文件的 `parser.add_argument` 中，添加待调优参数。

```
# -*- coding: utf-8 -*-
"""local集成测试用例，用于测试algorithm正确性，sklearn手写数字识别。"""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
from __future__ import unicode_literals
import argparse
import logging
import os
import numpy as np
# Import datasets, classifiers and performance metrics
from sklearn import datasets
from sklearn import ensemble
from sklearn import metrics
import joblib
logger = logging.getLogger()
def run_sk_digits(inputs):
  """sklearn手写数字识别。"""
  # Fake: copy
  if inputs.checkpoint_id >= 0:
    logger.info('copy model from {} to {}'.format(inputs.checkpoint_id,
                                                  inputs.trial_id))
  np.random.seed(0)
  # The digits dataset
  digits = datasets.load_digits()
  # To apply a classifier on this data, we need to flatten the image, to
  # turn the data in a (samples, feature) matrix:
  n_samples = len(digits.images)
```

```
  data = digits.images.reshape((n_samples, -1))
  n_samples = int(n_samples * inputs.ratio)
  data = data[:n_samples]
  digits.target = digits.target[:n_samples]
  offset = n_samples // 2
  x_train, y_train = data[:offset], digits.target[:offset]
  x_test, y_test = data[offset:], digits.target[offset:]
  params = {
    'n_estimators': inputs.n_estimators,
    'max_depth': inputs.max_depth,
    'min_samples_split': inputs.min_samples_split,
    'learning_rate': inputs.learning_rate,
    'loss': 'deviance'
  }
  clf = ensemble.GradientBoostingClassifier(**params)
  assert 0.0 < inputs.used_data_percent <= 1.0
  used_data = int(np.ceil(inputs.used_data_percent * len(x_train)))
  clf.fit(x_train, y_train)
  # TODO(weidan): We should allocate dirs (name) for each trial in the TrialManager.
  metric_file = inputs.metric_file
  dump_file = inputs.dump_file
  dirname = os.path.dirname(metric_file)
  if not os.path.exists(dirname):
    os.makedirs(dirname)
  dirname = os.path.dirname(dump_file)
  if not os.path.exists(dirname):
    os.makedirs(dirname)
  logger.info('saving to: {}, {}'.format(metric_file, dump_file))
  joblib.dump(clf, dump_file)
  #{"iteration": 0,"metric":"metrics,string","total_processed_sample":10000}
  with open(metric_file, 'w') as f:
    for _, y_pred in enumerate(clf.staged_predict(x_test)):
      result = metrics.classification_report(y_test, y_pred, digits=4)
      #avg / total     0.76     0.76     0.76     899
      result = result.rstrip().split('\n')[-1].split()[-2]
      f.write('%s\n' % result.strip())
if __name__ == '__main__':
  parser = argparse.ArgumentParser()
  parser.add_argument('--n_estimators', type=int)
  parser.add_argument('--max_depth', type=int)
```

```
parser.add_argument('--min_samples_split', type=int)
parser.add_argument('--learning_rate', type=float)
parser.add_argument('--trial_id', type=str)
parser.add_argument('--used_data_percent', type=float, default=1.0)
parser.add_argument('--checkpoint_id', type=int, default=-1)
parser.add_argument('--dump_file', type=str)
parser.add_argument('--metric_file', type=str)
parser.add_argument('--ratio', type=float, default=1.0)
args = parser.parse_args()
logging.basicConfig(level=logging.INFO)
run_sk_digits(args)
```

2. 在*test.py*中，配置如下参数。

```
from pai.automl.hpo import *
n = hyperparam.create(type='Categorical',
            name='n', candidates=[10, 20, 30, 40, 50, 60, 70, 80, 90])
s = hyperparam.create(type='Categorical',
            name='s', candidates=[4, 3, 2])
d = hyperparam.create(type='Categorical',
            name='d', candidates=[2, 3, 4, 5])
lr = hyperparam.create(type='Categorical',
             name='lr', candidates=[0.01, 0.02, 0.04, 0.08, 0.16, 0.32])
params = [n, s, d, lr]
```

3. 在*test.py*中，设置数据输出路径。

```
local_reader = reader.create(type='local_reader', # local_reader用于读取本地文件。
                location='./sk_log/{{trial.id}}/metric.log',
                parser_pattern='(\\d.\\d+)')
print(local_reader)
```

4. 在*test.py*中，配置待调参的参数和Metric文件。

```
cmd = ['python', './digits_model.py',
    '--n_estimators', "{{ trial.param.n }}",
    '--min_samples_split', "{{ trial.param.s }}",
    '--max_depth', "{{ trial.param.d }}",
    '--learning_rate', "{{ trial.param.lr }}",
    '--trial_id', "{{ trial.id }}",
    '--dump_file', "./sk_log/{{ trial.id }}/model.dump",
    '--metric_file', "./sk_log/{{ trial.id }}/metric.log"]
bashtask = task.create(type='BashTask',
                cmd=cmd,
                metric_reader=local_reader)
tasks = [bashtask]
```

5. 在*test.py*中，配置调参的资源。

```
tuner = AutoTuner(hyperparams=params,
            task_list=tasks,
            max_parallel=4,
            max_trial_num=20,
            mode='local',
            user_id='your_cloud_id'
            )
```

6. 使用文件传输工具，上传*digits_model.py*和*test.py*至同一目录。本文上传至*/usr/etc*路径。
7. 进入文件目录。

```
cd /usr/etc
```

8. 执行*test.py*脚本。

```
python3 test.py
```

结果显示如下。

```
2020-08-07,14:51:22.163 [INFO] results:
  params          results metadata
     d   lr  n s          id
0    2 0.32 50 2 0.9085    17
1    2 0.32 50 4 0.9085    18
2    4 0.16 80 2 0.8965     1
3    4 0.08 70 4 0.8735     3
4    4 0.16 30 3 0.8723     0
5    2 0.32 10 2 0.8723    15
6    4 0.04 80 4 0.8606     2
7    5 0.01 50 4 0.8152    13
8    5 0.01 90 2 0.8110    14
9    5 0.04 10 4 0.8101    22
10   5 0.01 40 4 0.8056    12
11   2 0.01 90 2 0.8011    16
12   2 0.01 90 3 0.8011    19
13   5 0.01 40 2 0.7998     9
14   5 0.01 50 2 0.7998    10
15   2 0.01 60 2 0.7639    11
16   2 0.01 50 2 0.7553     8
17   3 0.01 10 2 0.7511     7
18   2 0.01 10 2 0.7035     4
19   2 0.01 10 4 0.7035     5
20   2 0.01 10 3 0.7035     6
```

6.TensorFlow

Data Science集群内置Python 3的Tensorflow 1.15.0版本，可以直接使用。其中Master节点只支持购买CPU资源计算TensorFlow作业，Core节点支持购买CPU或GPU资源计算TensorFlow作业。本文主要介绍如何查看TensorFlow的版本、切换TensorFlow版本以及如何安装Python包。

使用引导

- 查看版本信息
- 切换TensorFlow版本
- 安装Python包

查看版本信息

1. 使用SSH方式登录到集群主节点，详情请参见使用SSH连接主节点。
2. 使用**pip3 list**命令，查看TensorFlow的版本信息。

切换TensorFlow版本

1. 下载切换TensorFlow版本的压缩包。切换TensorFlow版本压缩包：install_tf_header.tar.gz
   - 111111
   - datascience_tar
2. 使用文件传输工具，上传*install_tf_header.tar.gz*至Data Science集群Master节点的任意目录下。

   > **说明** 本文上传至*/root*目录。

3. 使用SSH方式登录到集群主节点，详情请参见使用SSH连接主节点。
4. 执行如下命令，切换TensorFlow版本。
   i. 解压缩文件。

   ```
   tar -zxvf install_tf_header.tar.gz
   ```

   ii. 切换TensorFlow版本。
   - 命令格式

     ```
     sh install_tf_header.sh <version>
     ```

     `version` 为您需要切换的版本号。
   - 示例：执行如下命令，切换TensorFlow版本为2.0.3。

     ```
     sh install_tf_header.sh 2.0.3
     ```

5. 使用**pip3 list**命令，查看TensorFlow版本号。

   TensorFlow版本已经切换为2.0.3版本。

安装Python包

1. 下载安装Python的压缩包。安装Python的压缩包：install_app_onds.tar.gztest11
2. 使用文件传输工具，上传*install_app_onds.tar.gz*至Data Science集群Master节点的任意目录下。

   > **说明** 本文上传至*/root*目录。

3. 使用SSH方式登录到集群主节点，详情请参见使用SSH连接主节点。
4. 执行如下命令，在Data Science集群所有节点安装Python包。
   i. 解压缩文件。

   ```
   tar -zxvf install_app_onds.tar.gz
   ```

   ii. 安装Python包。
   - 命令格式

     ```
     sh install_app_onds.sh <package_name> <version>
     ```

     其中涉及参数如下：
     - `package_name` 为您需要安装的Python包名。
     - `version` 为您需要安装Python包的版本号。
   - 示例：执行如下命令，在Data Science集群的所有节点上安装GNU Readline包，版本号为8.0.0。

     ```
     sh install_app_onds.sh gnureadline 8.0.0
     ```

7.EasyRec算法库

7.1. EasyRec概述

本文为您介绍阿里云自研的推荐算法库，包含DeepFM、DIN和MultiTower等经典主流推荐算法。E-MapReduce的Data Science集群已经内置了EasyRec算法库，您可以直接使用。

前提条件

已创建E-MapReduce的Data Science集群，并且勾选了EasyRec服务，详情请参见创建集群。

输入数据

本文输入数据以CSV格式（非CSV格式请转换为CSV格式）的文件为例，示例数据如下：

- train：dwd_avazu_ctr_deepmodel_train.csv
- test：dwd_avazu_ctr_deepmodel_test.csv

CSV文件不需要有文件头，输入数据列之间使用英文逗号（,）分隔。

```
1,10,1005,0,85f751fd,c4e18dd6,50e219e0,0e8e4642,b408d42a,09481d60,a99f214a,5deb445a,f4fffcd0,1,0,209
8,32,5,238,0,56,0,5
```

复制数据至HDFS

示例如下：

- `hadoop fs -mkdir -p hdfs:///user/easy_rec/data/`
- `hadoop fs -put dwd_avazu_ctr_deepmodel_train.csv hdfs:///user/easy_rec/data/`
- `hadoop fs -put dwd_avazu_ctr_deepmodel_test.csv hdfs:///user/easy_rec/data/`

训练文件

配置文件示例：dwd_avazu_ctr_deepmodel.config。详细配置请参见配置文件。

使用 `el_submit` 提交训练任务。语法如下。

```
el_submit [-h] [-t APP_TYPE] [-a APP_NAME] [-m MODE] [-m_arg MODE_ARG]
[-interact INTERACT] [-x EXIT]
[-enable_tensorboard ENABLE_TENSORBOARD]
[-log_tensorboard LOG_TENSORBOARD] [-conf CONF] [-f FILES]
[-pn PS_NUM] [-pc PS_CPU] [-pm PS_MEMORY] [-wn WORKER_NUM]
[-wc WORKER_CPU] [-wg WORKER_GPU] [-wm WORKER_MEMORY]
[-wnpg WNPG] [-ppn PPN] [-c COMMAND [COMMAND ...]]
```

> **说明** `el_submit` 相关的参数信息，请参见el_submit参数。

作业提交示例如下。

```
# bash
el_submit -t tensorflow-ps -a easy_rec_train -f dwd_avazu_ctr_deepmodel.config -m local -pn 1 -pc 4 -pm 20
000 -wn 3 -wc 6 -wm 20000 -c "python -m easy_rec.python.train_eval --pipeline_config_path dwd_avazu_ctr
_deepmodel.config --continue_train"
```

评估文件

使用 `el_submit` 提交评估任务。

```
# bash
el_submit -t standalone -a easy_rec_eval -f dwd_avazu_ctr_deepmodel.config -m local -wn 1 -wc 6 -wm 2000
0 -c "python -m easy_rec.python.eval --pipeline_config_path dwd_avazu_ctr_deepmodel.config"
```

导出文件

- 使用 `el_submit` 提交导出任务。

  ```
  # bash
  el_submit -t standalone -a easy_rec_export -f dwd_avazu_ctr_deepmodel.config,export.py -m local -wn 1
  -wc 1 -wm 5000 -c "python -m easy_rec.python.export --pipeline_config_path dwd_avazu_ctr_deepmodel
  .config --export_dir hdfs:///user/easy_rec/experiment/export"
  ```

  - `export_dir` ：导出模型目录。
  - `pipeline_config_path` ：EasyRec配置文件。
  - `checkpoint_path` ：指定checkpoint。默认不指定，不指定时则使用model_dir下面最新的checkpoint。

- 查看导出结果。

  ```
  # bash
  hadoop fs -ls hdfs:///user/easy_rec/experiment/export
  ```

配置文件

- 数据准备
  - 训练表

    ```
    train_input_path: "hdfs://127.0.0.1:9000/user/easy_rec/data/dwd_avazu_ctr_deepmodel_train.csv"
    ```

  - 测试数据

    ```
    eval_input_path: "hdfs://127.0.0.1:9000/user/easy_rec/data/dwd_avazu_ctr_deepmodel_test.csv"
    ```

  - 模型保存路径

    ```
    model_dir: "hdfs://127.0.0.1:9000/user/easy_rec/experiment"
    ```

- 数据相关的描述

```
# protobuf
data_config {
 separator: ","
 input_fields: {
  input_name: "label"
  input_type: FLOAT
  default_val:""
 }
 input_fields: {
  input_name: "hour"
  input_type: STRING
  default_val:""
 }
 input_fields: {
  input_name: "c1"
  input_type: STRING
  default_val:""
 }
 ...
 input_fields: {
  input_name: "c20"
  input_type: STRING
  default_val:""
 }
 input_fields: {
  input_name: "c21"
  input_type: STRING
  default_val:""
 }
 label_fields: "label"
 batch_size: 1024
 num_epochs: 10000
 prefetch_size: 32
 input_type: CSVInput
}
```

- 特征相关的参数

```
# protobuf
feature_configs: {
 input_names: "hour"
 feature_type: IdFeature
 embedding_dim: 16
 hash_bucket_size: 50
}
feature_configs: {
 input_names: "c1"
 feature_type: IdFeature
 embedding_dim: 16
 hash_bucket_size: 10
}
...
feature_configs: {
 input_names: "c20"
 feature_type: IdFeature
 embedding_dim: 16
 hash_bucket_size: 500
}
feature_configs: {
 input_names: "c21"
 feature_type: IdFeature
 embedding_dim: 16
 hash_bucket_size: 500
}
```

- 训练相关的参数

```
# protobuf
train_config {
 # 每200轮打印一行log。
 log_step_count_steps: 200
 # 优化器相关的参数。
 optimizer_config: {
  adam_optimizer: {
   learning_rate: {
    exponential_decay_learning_rate {
     initial_learning_rate: 0.0001
     decay_steps: 100000
     decay_factor: 0.5
     min_learning_rate: 0.0000001
    }
   }
  }
  use_moving_average: false
 }
 # 使用SyncReplicasOptimizer进行分布式训练（同步模式）。
 sync_replicas: true
 num_steps:2000
}
```

- 评估相关的参数

```
# protobuf
eval_config {
 metrics_set: {
  # metric为auc
  auc {}
 }
}
```

- 模型相关的参数

```
# protobuf
model_config:{
 model_class: "MultiTower"
 feature_groups: {
  group_name: "item"
  feature_names: "c1"
  feature_names: "banner_pos"
  feature_names: "site_id"
```

```
  feature_names: "site_id"
  feature_names: "site_domain"
  feature_names: "site_category"
  feature_names: "app_id"
  feature_names: "app_domain"
  feature_names: "app_category"
  wide_deep:DEEP
 }
 feature_groups: {
  group_name: "user"
  feature_names: "device_id"
  feature_names: "device_ip"
  feature_names: "device_model"
  feature_names: "device_type"
  feature_names: "device_conn_type"
  wide_deep:DEEP
 }
 feature_groups: {
  group_name: "user_item"
  feature_names: "hour"
  feature_names: "c14"
  feature_names: "c15"
  feature_names: "c16"
  feature_names: "c17"
  feature_names: "c18"
  feature_names: "c19"
  feature_names: "c20"
  feature_names: "c21"
  wide_deep:DEEP
 }
 multi_tower {
  towers {
   input: "item"
   dnn {
    hidden_units: [384, 320, 256, 192, 128]
   }
  }
  towers {
   input: "user"
   dnn {
    hidden_units: [384, 320, 256, 192, 128]
```

```
    }
  }
  towers {
   input: "user_item"
   dnn {
    hidden_units: [384, 320, 256, 192, 128]
   }
  }
  final_dnn {
   hidden_units: [256, 192, 128, 64]
  }
  l2_regularization: 0.0
 }
 embedding_regularization: 0.0
}
```

el_submit参数

参数	说明
-t APP_TYPE	提交的任务类型： • Tensorflow-ps模式，采用Parameter Server方式通信，该方式为原生TensorFlow PS模式。 • Tensorflow-mpi模式，采用Horovod进行通信。 • Standalone模式，您可以调度任务到YARN集群中启动单机任务。
-a APP_NAME	作业名称。
-m MODE	运行环境： • Local，使用的是emr-worker节点上的Python运行环境。如果您需要使用第三方Python包，请手动在所有节点上安装第三方Python包。 • Docker，使用的是emr-worker节点上的Docker运行环境，TensorFlow运行在Docker Container内。 • Virtual-env，您上传的Python环境。您可以在Python环境中安装不同于emr-worker节点的Python库。
-m_arg MODE_ARG	补充参数。 此参数和 MODE 配合使用，如果运行环境是Docker，则设置为Docker镜像名称。如果运行环境是Virtual-env，则设置为Python环境文件名称。
-x Exit	当所有emr-worker节点成功完成训练后，PS节点自动退出。

参数	说明
-enable_tensorboard	是否在启动训练任务时启动TensorBoard。
-log_tensorboard	指定HDFS中TensorBoard日志的位置。如果训练时启动了TensorBoard，则需要指定TensorBoard日志位置。
-conf CONF Hadoop conf	位置。
–f FILES	运行TensorFlow所有依赖的文件和文件夹，包含执行脚本。如果执行Virtual-env文件，您可以放置所有依赖到一个文件夹中，脚本会自动按照文件夹层次关系上传到HDFS中。
-pn TensorFlow	启动的参数服务器个数。
-pc	每个参数服务器申请的CPU核数。
-pm	每个参数服务器申请的内存大小。
-wn TensorFlow	启动的emr-worker节点个数。
-wc	每个emr-worker节点申请的CPU核数。
-wg	每个emr-worker节点申请的GPU核数。
-wm	每个emr-worker节点申请的内存个数。
-c COMMAND	执行的命令。例如 pythoncensus.py 。
-wnpg	每个GPU核上同时运行的Worker数量（针对tensorflow-ps模式）。
-ppn	每个GPU核上同时运行的Worker数量（针对horovod模式）。

相关内容

- Excel特征配置
- FeatureConfig配置

7.2. Excel特征配置

本文为您介绍如何通过Excel方式生成配置文件。

命令

Excel模板如下：

- multi_tower：multi_tower_template.xls

- deepfm：deepfm_template.xls
- wide_and_deep：同deepfm。

```
# bash
pip install pandas
wget https://pai-vision-data-hz.oss-cn-zhangjiakou.aliyuncs.com/easy-rec/releases/scripts/create_config_from_excel.py
python scripts/create_config_from_excel.py --model_type multi_tower --excel_path multi_tower_template.xls --output_path multi_tower.config
```

参数	说明
model_type	必须和Excel模板匹配。
excel_path	输出Config文件。
output_path	输入输出文件，train_input_path TRAIN_INPUT_PATH和eval_input_path EVAL_INPUT_PATH。 默认数据文件（CSV）列（Column）分割符号是 , ；列（Column）内部里面字符分割符号是 | 。 您可以自定义分隔符： --column_separator $'|' --incol_separator $',' 。

配置说明

模板包含features、global、group、types和basic_types共五个页签。

- features描述特征配置。

```
|**name**|**data_type**|**type**|**user_item_other**|**global**|**hash_bucket_size**|**embedding_dim**|**default_value**|**weights**|**boundaries**|**query**|
|---|---|---|---|---|---|---|---|---|---|---|
|label|double|label|label|||||||
|uid|string|category|user|uid|||||
|own_room|bigint|dense|user|||||10,20,30||
|cate_idx|string|tags|user|cate|||cate_wgt|||
|cate_wgt|string|weights|user|||||||
|**...**|||||||||||
```

参数	说明
name	输入的列名。

参数	说明
data_type	输入的数据类型，包括STRING，BIGINT和DOUBLE。
type	特征类型，引用types中的types列： ○ label：要预测的列 ○ category：离散值特征 ○ dense：连续值特征 ○ tags：标签型特征 ○ weights：tags对应的weight ○ indexes：一串数字，使用in column separator分割。例如：1,2,4,5。 ○ notneed：不需要的，可以忽略的
group	分组设置，引用groupsheet列： ○ multi_tower ■ user：user tower ■ item：item tower ■ user_item：user_item tower ■ label：label tower ○ deepfm ■ wide：特征仅用在wide部分 ■ deep：特征仅用在deep和fm部分 ■ wide_and_deep：特征用在wide, deep, fm部分，默认选wide_and_deep

参数	说明
global	引用global里面的name列。 ○ hash_bucket_size：hash_bucket桶的大小 ○ embedding_dim：embedding的大小 ○ default_value：缺失值填充 ○ weights：如果type是tags，则可以指定weights。weights和tags必须要有相同的列 ○ boundaries：连续值离散化的区间。例如：10,20,30，将会离散成区间(-inf, 10), [10, 20), [20, 30), [30, inf)。 **说明** 配置boundaries时，通常也需要配置embedding_dim。 ○ query：当前未使用，拟用作DIN的target，通常是item_id。 **说明** ■ features必须和CSV文件的列是一一对应的，顺序必须要一致。 ■ features的数目和输入标或者文件的列的数目必须是一致的 ■ 如果某些列不需要，可以设置type为notneed。

- global：描述share embedding里面share embedding的信息。

  其中hash_bucket_size embedding_dim default_value会覆盖features表里面对应的信息。

```
| **name** | **type** | **hash_bucket_size** | **embedding_dim** | **default_value** |
| --- | --- | --- | --- | --- |
| cate | category | 1000 | 10 | 0 |
| uid | category | 100000 | 10 | 0 |
| **...** | | | | |
```

- group
  - - multi_tower模型中的分组设置

```
| **group** |
| --- |
| user |
| item |
| user_item |
| label |
```

- - deep_fm模型中的分组设置

group
wide_and_deep
wide
deep
label

- types：描述数据类型。

types
label
category
tags
weights
dense
indexes
notneed

- basic_types：描述输入表的数据类型， 包括STRING，BIGINT和DOUBLE。

basic_types
string
bigint
double

7.3. FeatureConfig配置

本文为您介绍如何通过特征配置方式生成配置文件。常用的特征有ID类、连续值类、Tag类和Sequense类。

ID类

ID类特征中， `user_id` 和 `itemid` 取值数较多，通常做Hash处理，映射ID值到相对较小的桶中。如下例所示。

```
feature_configs: {
 input_names: "uid"
 feature_type: IdFeature
 embedding_dim: 64
 hash_bucket_size: 100000
}
feature_configs: {
 input_names: "iid"
 feature_type: IdFeature
 embedding_dim: 64
 hash_bucket_size: 100000
}
```

workday和level等ID类特征取值数较少时：

- 取值可以直接写在 `vocab_list` 中，如下例所示。

```
feature_configs: {
 input_names: "workday"
 feature_type: IdFeature
 vocab_list: ["星期一", "星期二", "星期三", "星期四", "星期五", "星期六", "星期日"]
 embedding_dim: 8
}
```

- 取值也可以直接映射到Hash桶中，无需配置 `vocab_list` 。

```
feature_configs: {
 input_names: "workday"
 feature_type: IdFeature
 hash_bucket_size: 70
 embedding_dim: 10
}
```

> 注意 为防止Hash冲突，通常设置Hash桶数（hash_bucket_size）为取值数（embedding_dim）的5~10倍。

连续值类

连续值类特征可以在Config中手动配置离散化的区间，如下例所示。

```
feature_configs: {
 input_names: "ctr"
 feature_type: RawFeature
 boundaries: [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
 embedding_dim: 8
}
```

Tag类

Tag类特征格式一般为“XX|XX|XX”，例如文章标签特征为“娱乐|搞笑|热门”。

对于Tag中的每个值，同样需要配置hash_bucket_size。如下例所示。

```
feature_configs: {
 input_names: "article_tag"
 feature_type: TagFeature
 embedding_dim: 16
 hash_bucket_size: 1000
}
```

Sequense类

Sequense类特征格式通常为“XX|XX|XX”，例如用户行为序列特征为“item_id1|item_id2|item_id3”。

对于序列中的每个值，同样需要配置hash_bucket_size，如下例所示。

```
feature_configs: {
 input_names: "play_sequence"
 feature_type: SequenceFeature
 embedding_dim: 64
 hash_bucket_size: 100000
}
```

8.部署TensorFlow模型至PAI EAS

本文介绍如何在E-MapReduce中部署TensorFlow模型同步至PAI EAS。

前提条件

- 已创建Data Science集群，相关注意信息请参见概述。
- 已开通PAI的在线模型服务EAS，详情请参见EAS开通与购买指南。
- 已将Tensorflow SaveModel格式的模型打包成*tar.gz*文件。

背景信息

PAI-EAS的命令行工具eascmd64已经内置在DataScience集群中，可以直接使用。

部署TensorFlow模型

1.
2. 创建配置。 执行以下命令创建EAS的配置。

   ```
   eascmd64 config -i LTAI******* -k 1Hr5GzqNkc8******** -e pai-eas.cn-shenzhen.aliyuncs.com
   ```

   各参数描述如下。

参数	描述
-i	当前阿里云账号的AccessKey ID。
-k	当前阿里云账号的AccessKey Secret。
-e	EAS的访问地址（Host）。 **说明** 不同区域的Host不同，例如北京的Host是pai-eas.cn-beijing.aliyuncs.com。

3. 上传模型。

   执行以下命令将模型部署至EAS。

   ```
   eascmd64 upload <model_path>
   ```

   说明 `<model_path>` 为模型打包好的文件的名称。

   返回信息如下，请记录好 `oss tartget path` 。

4. 创建*pmml.json*文件并部署。

i. 按照如下示例格式配置*pmml.json*文件。

```
{
"name": "test_3",
"generate_token": "true",
"model_path": "oss://eas-model-hangzhou/107992689699****/resnet_v1_fp32_savedmodel_NCHW.tar.gz",
"processor": "tensorflow_cpu",
"metadata": {
"instance": 1,
"cpu": 1
}
}
```

涉及参数解释如下。

参数	描述	示例
name	表示服务的名称，不支持中文字符。	test_3
model_path	上传模型步骤中返回的oss tartget path。	*oss://eas-model-hangzhou/107992689699****/resnet_v1_fp32_savedmodel_NCHW.tar.gz*

ii. 执行以下命令完成部署。

```
eascmd64 create pmml.json
```

查看服务

1. 登录PAI-EAS控制台。
2. 在左侧导航栏中，单击**模型部署 > EAS-模型在线服务**。 在**EAS-模型在线服务**页面，可查看服务。