PaddleFL

Release 0.1.0.beta

PaddlePaddle

May 25, 2020

QUICK START

CHAPTER

ONE

QUICK START INSTRUCTIONS

1.1 Install PaddleFL

To install PaddleFL, we need the following packages.

```
paddlepaddle >= 1.6
networkx
```

We can run

```
python setup.py install
or
pip install paddle-fl
```

1.2 Step 1: Define Federated Learning Compile-Time

We define very simple multiple layer perceptron for demonstration. When multiple organizations agree to share data knowledge through PaddleFL, a model can be defined with agreement from these organizations. A FLJob can be generated and saved. Programs needed to be run each node will be generated separately in FLJob.

```
import paddle.fluid as fluid
import paddle_fl as fl
from paddle_fl.core.master.job_generator import JobGenerator
from paddle_fl.core.strategy.fl_strategy_base import FLStrategyFactory

class Model(object):
    def __init__(self):
        pass

    def mlp(self, inputs, label, hidden_size=128):
        self.concat = fluid.layers.concat(inputs, axis=1)
        self.fc1 = fluid.layers.fc(input=self.concat, size=256, act='relu')
        self.fc2 = fluid.layers.fc(input=self.fc1, size=128, act='relu')
    self.predict = fluid.layers.fc(input=self.fc2, size=2, act='softmax')
        self.sum_cost = fluid.layers.cross_entropy(input=self.predict, label=label)
        self.accuracy = fluid.layers.accuracy(input=self.predict, label=label)
        self.loss = fluid.layers.reduce_mean(self.sum_cost)
        self.startup_program = fluid.default_startup_program()

inputs = [fluid.layers.data( \
            name=str(slot_id), shape=[5],
```

(continues on next page)

(continued from previous page)

```
        dtype="float32")
      for slot_id in range(3)]
label = fluid.layers.data( \
            name="label",
            shape=[1],
            dtype='int64')

model = Model()
model.mlp(inputs, label)

job_generator = JobGenerator()
optimizer = fluid.optimizer.SGD(learning_rate=0.1)
job_generator.set_optimizer(optimizer)
job_generator.set_losses([model.loss])
job_generator.set_startup_program(model.startup_program)
job_generator.set_infer_feed_and_target_names(
    [x.name for x in inputs], [model.predict.name])

build_strategy = FLStrategyFactory()
build_strategy.fed_avg = True
build_strategy.inner_step = 1
strategy = build_strategy.create_fl_strategy()

endpoints = ["127.0.0.1:8181"]
output = "fl_job_config"
job_generator.generate_fl_job(
    strategy, server_endpoints=endpoints, worker_num=2, output=output)
```

1.3 Step 2: Issue FL Job to Organizations

We can define a secure service to send programs to each node in FLJob. There are two types of nodes in distributed federated learning job. One is FL Server, the other is FL Trainer. A FL Trainer is owned by individual organization and an organization can have multiple FL Trainers given different amount of data knowledge the organization is willing to share. A FL Server is owned by a secure distributed training cluster. By means of security of the cluster, all organizations participated in the Federated Training Job should agree to trust the cluster is secure.

1.4 Step 3: Start Federated Learning Run-Time

On FL Scheduler Node, number of servers and workers are defined. Besides, the number of workers that participate in each upating cycle is also determined. Finally, the FL Scheduler waits servers and workers to initialize.

```
from paddle_fl.core.scheduler.agent_master import FLScheduler

worker_num = 2
server_num = 1
# Define the number of worker/server and the port for scheduler
scheduler = FLScheduler(worker_num,server_num,port=9091)
scheduler.set_sample_worker_num(worker_num)
scheduler.init_env()
print("init env done.")
scheduler.start_fl_training()
```

On FL Trainer Node, a training script is defined as follows:

```
from paddle_fl.core.trainer.fl_trainer import FLTrainerFactory
from paddle_fl.core.master.fl_job import FLRunTimeJob
import numpy as np
import sys

def reader():
    for i in range(1000):
        data_dict = {}
        for i in range(3):
            data_dict[str(i)] = np.random.rand(1, 5).astype('float32')
        data_dict["label"] = np.random.randint(2, size=(1, 1)).astype('int64')
        yield data_dict

trainer_id = int(sys.argv[1]) # trainer id for each guest
job_path = "fl_job_config"
job = FLRunTimeJob()
job.load_trainer_job(job_path, trainer_id)
job._scheduler_ep = "127.0.0.1:9091" # Inform the scheduler IP to trainer
trainer = FLTrainerFactory().create_fl_trainer(job)
trainer.start()

output_folder = "fl_model"
step_i = 0
while not trainer.stop():
    step_i += 1
    print("batch %d start train" % (step_i))
    trainer.run(feed=data, fetch=[])
    if trainer_id == 0:
        print("start saving model")
        trainer.save_inference_program(output_folder)
    if step_i >= 100:
       break
```

On FL Server Node, a training script is defined as follows:

```
import paddle_fl as fl
import paddle.fluid as fluid
from paddle_fl.core.server.fl_server import FLServer
from paddle_fl.core.master.fl_job import FLRunTimeJob
server = FLServer()
server_id = 0
job_path = "fl_job_config"
job = FLRunTimeJob()
job.load_server_job(job_path, server_id)
job._scheduler_ep = "127.0.0.1:9091" # IP address for scheduler
server.set_server_job(job)
server._current_ep = "127.0.0.1:8181" # IP address for server
server.start()
```

See instruction for quick start.

CHAPTER TWO

PADDLEFL

PaddleFL is an open source federated learning framework based on PaddlePaddle. Researchers can easily replicate and compare different federated learning algorithms with PaddleFL. Developers can also benefit from PaddleFL in that it is easy to deploy a federated learning system in large scale distributed clusters. In PaddleFL, serveral federated learning strategies will be provided with application in computer vision, natural language processing, recommendation and so on. Application of traditional machine learning training strategies such as Multi-task learning, Transfer Learning in Federated Learning settings will be provided. Based on PaddlePaddle's large scale distributed training and elastic scheduling of training job on Kubernetes, PaddleFL can be easily deployed based on full-stack open sourced software.

CHAPTER

THREE

FEDERATED LEARNING

Data is becoming more and more expensive nowadays, and sharing of raw data is very hard across organizations. Federated Learning aims to solve the problem of data isolation and secure sharing of data knowledge among organizations. The concept of federated learning is proposed by researchers in Google [1, 2, 3].

3.1 Overview of PaddleFL

In PaddleFL, horizontal and vertical federated learning strategies will be implemented according to the categorization given in [4]. Application demonstrations in natural language processing, computer vision and recommendation will be provided in PaddleFL.

3.1.1 Federated Learning Strategy

- **Vertical Federated Learning**: Logistic Regression with PrivC, Neural Network with third-party PrivC [5]
- **Horizontal Federated Learning**: Federated Averaging [2], Differential Privacy [6]

3.1.2 Training Strategy

- **Multi Task Learning** [7]
- **Transfer Learning** [8]
- **Active Learning**

3.2 Framework design of PaddleFL

In PaddleFL, components for defining a federated learning task and training a federated learning job are as follows:

3.2.1 Compile Time

- **FL-Strategy**: a user can define federated learning strategies with FL-Strategy such as Fed-Avg[1]
- **User-Defined-Program**: PaddlePaddle's program that defines the machine learning model structure and training strategies such as multi-task learning.
- **Distributed-Config**: In federated learning, a system should be deployed in distributed settings. Distributed Training Config defines distributed training node information.

- **FL-Job-Generator**: Given FL-Strategy, User-Defined Program and Distributed Training Config, FL-Job for federated server and worker will be generated through FL Job Generator. FL-Jobs will be sent to organizations and federated parameter server for run-time execution.

3.2.2 Run Time

- **FL-Server**: federated parameter server that usually runs in cloud or third-party clusters.
- **FL-Worker**: Each organization participates in federated learning will have one or more federated workers that will communicate with the federated parameter server.
- **FL-scheduler**: Decide which set of trainers can join the training before each updating cycle.

3.3 On Going and Future Work

- Experimental benchmark with public datasets in federated learning settings.
- Federated Learning Systems deployment methods in Kubernetes.
- Vertical Federated Learning Strategies and more horizontal federated learning strategies will be open sourced.

CHAPTER

FOUR

PADDLEFL

PaddleFL is an open source federated learning framework based on PaddlePaddle. Researchers can easily replicate and compare different federated learning algorithms with PaddleFL. Developers can also benefit from PaddleFL in that it is easy to deploy a federated learning system in large scale distributed clusters. In PaddleFL, serveral federated learning strategies will be provided with application in computer vision, natural language processing, recommendation and so on. Application of traditional machine learning training strategies such as Multi-task learning, Transfer Learning in Federated Learning settings will be provided. Based on PaddlePaddle's large scale distributed training and elastic scheduling of training job on Kubernetes, PaddleFL can be easily deployed based on full-stack open sourced software.

CHAPTER

FIVE

FEDERATED LEARNING

Data is becoming more and more expensive nowadays, and sharing of raw data is very hard across organizations. Federated Learning aims to solve the problem of data isolation and secure sharing of data knowledge among organizations. The concept of federated learning is proposed by researchers in Google [1, 2, 3].

5.1 Overview of PaddleFL

In PaddleFL, horizontal and vertical federated learning strategies will be implemented according to the categorization given in [4]. Application demonstrations in natural language processing, computer vision and recommendation will be provided in PaddleFL.

5.1.1 Federated Learning Strategy

- **Vertical Federated Learning**: Logistic Regression with PrivC, Neural Network with third-party PrivC [5]
- **Horizontal Federated Learning**: Federated Averaging [2], Differential Privacy [6]

5.1.2 Training Strategy

- **Multi Task Learning** [7]
- **Transfer Learning** [8]
- **Active Learning**

5.2 Framework design of PaddleFL

In PaddleFL, components for defining a federated learning task and training a federated learning job are as follows:

5.2.1 Compile Time

- **FL-Strategy**: a user can define federated learning strategies with FL-Strategy such as Fed-Avg[1]
- **User-Defined-Program**: PaddlePaddle's program that defines the machine learning model structure and training strategies such as multi-task learning.
- **Distributed-Config**: In federated learning, a system should be deployed in distributed settings. Distributed Training Config defines distributed training node information.

- **FL-Job-Generator**: Given FL-Strategy, User-Defined Program and Distributed Training Config, FL-Job for federated server and worker will be generated through FL Job Generator. FL-Jobs will be sent to organizations and federated parameter server for run-time execution.

5.2.2 Run Time

- **FL-Server**: federated parameter server that usually runs in cloud or third-party clusters.
- **FL-Worker**: Each organization participates in federated learning will have one or more federated workers that will communicate with the federated parameter server.
- **FL-scheduler**: Decide which set of trainers can join the training before each updating cycle.

5.3 On Going and Future Work

- Experimental benchmark with public datasets in federated learning settings.
- Federated Learning Systems deployment methods in Kubernetes.
- Vertical Federated Learning Strategies and more horizontal federated learning strategies will be open sourced.

5.3.1 Example in recommendation with FedAvg

This document introduces how to use PaddleFL to train a model with Fl Strategy.

Dependencies

- paddlepaddle>=1.6

How to install PaddleFL

Please use python which has paddlepaddle installed

```
python setup.py install
```

Model

Gru4rec is a classical session-based recommendation model. Detailed implementations with paddlepaddle is here.

Datasets

Public Dataset Rsc15

```
#download data
cd example/gru4rec_demo
sh download.sh
```

How to work in PaddleFL

PaddleFL has two phases , CompileTime and RunTime. In CompileTime, a federated learning task is defined by fl_master. In RunTime, a federated learning job is executed on fl_server and fl_trainer in distributed clusters.

```
sh run.sh
```

How to work in CompileTime

In this example, we implement compile time programs in fl_master.py

```
# please run fl_master to generate fl_job
python fl_master.py
```

In fl_master.py, we first define FL-Strategy, User-Defined-Program and Distributed-Config. Then FL-Job-Generator generate FL-Job for federated server and worker.

```
# define model
model = Model()
model.gru4rec_network()

# define JobGenerator and set model config
# feed_name and target_name are config for save model.
job_generator = JobGenerator()
optimizer = fluid.optimizer.SGD(learning_rate=2.0)
job_generator.set_optimizer(optimizer)
job_generator.set_losses([model.loss])
job_generator.set_startup_program(model.startup_program)
job_generator.set_infer_feed_and_target_names(
    [x.name for x in model.inputs], [model.loss.name, model.recall.name])

# define FL-Strategy , we now support two flstrategy, fed_avg and dpsgd. Inner_step␣
↪means fl_trainer locally train inner_step mini-batch.
build_strategy = FLStrategyFactory()
build_strategy.fed_avg = True
build_strategy.inner_step = 1
strategy = build_strategy.create_fl_strategy()

# define Distributed-Config and generate fl_job
endpoints = ["127.0.0.1:8181"]
output = "fl_job_config"
job_generator.generate_fl_job(
    strategy, server_endpoints=endpoints, worker_num=2, output=output)
```

How to work in RunTime

```
python -u fl_scheduler.py >scheduler.log &
python -u fl_server.py >server0.log &
python -u fl_trainer.py 0 data/ >trainer0.log &
python -u fl_trainer.py 1 data/ >trainer1.log &
```

fl_trainer.py can define own reader according to data.

```
r = Gru4rec_Reader()
train_reader = r.reader(train_file_dir, place, batch_size=10)
```

Simulated experiments on real world dataset

To show the concept and effectiveness of horizontal federated learning with PaddleFL, a simulated experiment is conducted on an open source dataset with a real world task. In horizontal federated learning, a group of organizations are doing similar tasks based on private dataset and they are willing to collaborate on a certain task. The goal of the collaboration is to improve the task accuracy with federated learning.

The simulated experiment suppose all organizations have homogeneous dataset and homogeneous task which is an ideal case. The whole dataset is from small part of [Rsc15] and each organization has a subset as a private dataset. To show the performanc e improvement under federated learning, models based on each organization's private dataset are trained and a model under distributed federated learning is trained. A model based on traditional parameter server training is also trained where the whole dataset is owned by a single organization.

From the table and the figure given below, model evaluation results are similar between federated learning and traditional parameter server training. It is clear that compare with models trained with only private dataset, models' performance for each organization get significant improvement with federated learning.

```
# download code and readme
wget https://paddle-zwh.bj.bcebos.com/gru4rec_paddlefl_benchmark/gru4rec_benchmark.tar
```

Dataset	training methods	FL Strategy	recall@20
the whole dataset	private training	•	0.504
the whole dataset	federated learning	FedAvg	0.504
1/4 of the whole dataset	private training	•	0.286
1/4 of the whole dataset	private training	•	0.277
1/4 of the whole dataset	private training	•	0.269
1/4 of the whole dataset	private training	•	0.282

5.3.2 API Reference

paddle_fl.core.master: PaddleFL Compile-Time

paddle_fl.core.strategy: Federated Learning Strategies

paddle_fl.core.trainer: Trainer Run-Time

paddle_fl.core.server: Server Run-Time

CHAPTER

SIX

THE TEAM

6.1 The Team

PGL is developed and maintained by NLP and Paddle Teams at Baidu

PaddleFL is developed and maintained by Nimitz Team at Baidu

6.2 Reference

[1]. Jakub Konen, H. Brendan McMahan, Daniel Ramage, Peter Richtik. **Federated Optimization: Distributed Machine Learning for On-Device Intelligence.** 2016

[2]. H. Brendan McMahan, Eider Moore, Daniel Ramage, Blaise Agera y Arcas. **Federated Learning of Deep Networks using Model Averaging.** 2017

[3]. Jakub Konen, H. Brendan McMahan, Felix X. Yu, Peter Richtik, Ananda Theertha Suresh, Davepen Bacon. **Federated Learning: Strategies for Improving Communication Efficiency.** 2016

[4]. Qiang Yang, Yang Liu, Tianjian Chen, Yongxin Tong. **Federated Machine Learning: Concept and Applications.** 2019

[5]. Kai He, Liu Yang, Jue Hong, Jinghua Jiang, Jieming Wu, Xu Dong et al. **PrivC - A framework for efficient Secure Two-Party Computation. In Proceedings of 15th EAI International Conference on Security and Privacy in Communication Networks.** SecureComm 2019

[6]. Mart Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, Li Zhang. **Deep Learning with Differential Privacy.** 2016

[7]. Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, Ameet Talwalkar. **Federated Multi-Task Learning** 2016

[8]. Yang Liu, Tianjian Chen, Qiang Yang. **Secure Federated Transfer Learning.** 2018

6.3 Reference

[1]. Jakub Konen, H. Brendan McMahan, Daniel Ramage, Peter Richtik. **Federated Optimization: Distributed Machine Learning for On-Device Intelligence.** 2016

[2]. H. Brendan McMahan, Eider Moore, Daniel Ramage, Blaise Agera y Arcas. **Federated Learning of Deep Networks using Model Averaging.** 2017

[3]. Jakub Konen, H. Brendan McMahan, Felix X. Yu, Peter Richtik, Ananda Theertha Suresh, Davepen Bacon. **Federated Learning: Strategies for Improving Communication Efficiency.** 2016

[4]. Qiang Yang, Yang Liu, Tianjian Chen, Yongxin Tong. **Federated Machine Learning: Concept and Applications.** 2019

[5]. Kai He, Liu Yang, Jue Hong, Jinghua Jiang, Jieming Wu, Xu Dong et al. **PrivC - A framework for efficient Secure Two-Party Computation. In Proceedings of 15th EAI International Conference on Security and Privacy in Communication Networks.** SecureComm 2019

[6]. Mart Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, Li Zhang. **Deep Learning with Differential Privacy.** 2016

[7]. Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, Ameet Talwalkar. **Federated Multi-Task Learning** 2016

[8]. Yang Liu, Tianjian Chen, Qiang Yang. **Secure Federated Transfer Learning.** 2018

CHAPTER SEVEN

LICENSE

PaddleFL uses Apache License 2.0.

PYTHON MODULE INDEX

INDEX