Parameterized, Automated and Distributed Machine Learning

Ashwin Nimhan Student of Data Science Indiana University Bloomington

animhan@indiana.edu

Manashree Rao Student of Data Science Indiana University Bloomington

manarao@indiana.edu

Abstract

Keywords: Data Pipeline, Auto ML, Black-box ML, predictive modelling

I. INTRODUCTION

Machine learning has achieved considerable successes in recent years and an ever-growing number of disciplines rely on it. However, this success crucially relies on human machine learning experts, who select appropriate features, workflows, machine learning paradigms, algorithms, and their hyperparameters. The research area that targets progressive automation of machine learning is AutoML. The goal is to design the perfect machine learning black box capable of performing all model selection and hyper-parameter tuning without any human intervention. The current approaches in the AutoML field are heavily dependent on underlying platform and supported languages, like auto-sklearn or auto-weka. What we want to achieve is try to scale this across multiple programming languages, i.e., use python, R, Spark ML, etc. together for ML tasks like choosing a machine learning model, tuning hyper-parameters, avoiding overfitting and optimization for a provided evaluation metric.

II. SURVEY OF EXISTING SYSTEMS

- A. AutoCompete
- B. Auto-Sklearn
- C. Auto-WEKA
- D. Hyperopt-Sklearn

III. WORKFLOW MANAGEMENT SYSTEMS

A. Typical stages of Workflow Management

- Create Jobs to interact with systems that operate on Data Hive/Presto/HDFS/Postgres/S3 etc
- (Dynamic) Workflow creation based on the number of sources, size of data, business logic, variety of data, changes in the schema, etc.
- Manage Dependencies between Operations like Upstream, Downstream, Cross Pipeline dependencies, Previous Job state, etc.
- Schedule the Jobs/Operations like Calendar schedule, Event Driven, Cron Expression etc.
- Keep track of the Operations and the metrics of the workflow, monitor the current/historic state of the jobs, the results of the jobs etc.
- Ensure Fault tolerance of the pipelines and have the capability to back fill any missing data, etc.

B. Survey of Data Pipelines and Workflow Management Systems

There are different workflow management systems like:

- Oozie Oozie is a workflow scheduler system to manage Apache Hadoop jobs
- Luigi Luigi is a Python module that helps you build complex pipelines of batch jobs.
- Airflow Airflow is a platform to programmatically author, schedule and monitor workflows.
- Azkaban Azkaban is a batch workflow job scheduler created at LinkedIn to run Hadoop jobs.
- Pinball Pinball is a scalable workflow manager developed at Pinterest.

We compare these systems as per our requirements.

1

C. Comparison of existing systems

Tasks are grouped to gether in Most of the corde treats Tasks as the main unit of work	Feature	Luigi	Airflow	Pinball
Pipeline together into a DAG to the round work of the code treats Tasks as the main unit of work				
to be run. Most of the code treats Tasks as the main unit of work. Class processing the main unit of work. Ul Overview of Tasks only meta-data/job and factorial status of the main to Airflow, but fewer details. Create multiple Tasks and the main for the meta-data/job and the main to Airflow, but fewer details. Scaling Create multiple Tasks DAGs can be constructed with multiple operators. Scale out by adding Celery workers Subprocess Operators can be constructed with multiple operators. Scale out by adding Celery workers Subprocess Subprocess Subprocess Subprocess Subprocess Subprocess Code Tasks can be operators can be constructed with depends_on_past parameter dependency ment Gode State Uness SQLAlchemy frou don't use a starting, eg child_job requires parent_job. Tircky persistence tracks Yes to db Tasks on by Celery and RabbitMQ/Reddis Tircky persiss fence of insurports any store that is supported by fest of the database (lonly log file). The status of a task is stored to database. Yes to db Yes to db Yes to db To betailed, looks like Sidekiq Workers Detailed, looks like Sidekiq Workers 'claim' messages from the queue with an database. Workers 'claim' messages from the queue with an database. Operators mark jobs as passed or failed. Last updated is refreshed frequently with a heartheath function. kill; Jobs as passed or failed. Last updated is refreshed frequently with a heartheath function. kill; Jobs as passed or failed. Last updated is refreshed frequently with a heartheath function. kill; Jobs as passed or failed. Last updated is refreshed frequently with a heartheath function. kill; Jobs as passed or failed. Last updated is refreshed frequently with a heartheath function. kill; Jobs as passed or failed. Last updated is refreshed frequently with a heartheath function. kill; Jobs as passed or failed. Last updated is refreshed frequently with a heart				WOLKHOW
the code treats Tasks as the main unit of work. Class processing the main unit of work UI Overview of Tasks only metadata/job status is stored atata/sob status is other details. Tasks status is stored atata/sob status is other details. Tasks status is stored atata/sob status is other details. Tasks status is stored atata/sob status is other details. Tasks status is stored atata/sob status is other details. Tasks can be dency execution Tasks can be constructed with multiple Operators. Scale out by adding Celery workers. Tasks can be constructed with multiple operators. Scale out by adding Celery workers. Tasks can be constructed with depends on past parameter Tasks can be constructed with depends on the existence of its output. Tasks can be constructed with depends on the existence of its output. Tasks can be constructed with depends on the existence of its output. Tasks can be constructed with multiple constructed with multiple care and	Pipeline		to define Jobs.	
as the main unit of work Class processing the main unit of work UI Overview of Tasks only meta-data/job data/job data/job data/job status is stored in a database. Task status is stored data/job data/j				
Class pro- cessing the main in unit of work UI Overview of Tasks of tasks a passed or failed. UI Overview of Verview		the code treats Tasks		
Class Processing the main unit of work Ul Overview of Tasks only Job status is stored in a database. Overview of Task status is stored data/job status in a database. Operators mark jobs as passed or failed. Last updated is refreshed frequently with a heartheats. Operators called to clear all jobs with older heartheats. Operators can be constructed with multiple Operators. Scale out by adding Celery workers Operators can be constructed with depends_on_past parameter Operators can be constructed with requires) method age		as the main unit of		
pro- cessing the main unit of work UI Overview of Tasks UI Task status is stored of adatabase. UI Task status is stored of adatabase. Detailed, looks like Sidekiq Overview of Tasks only Task status is stored of a database. Detailed, looks like Sidekiq Operators mark jobs as passed or failed, to clear ull jobs with older bearbeats. Scaling Create multiple Tasks DAGs can be constructed with multiple Operators. Scale out by adding Celery workers Database out by adding Celery workers Subprocess Execution dependency ment Code Code Code State Uses SQLAlchemy for abstracting away the choice of and querying the database(Mysql/Postgresqlbways checked based on the existence of its output. Taraks Yes to db Sistory Morkers 'Claim' messages from the queue with an ownership timestamp on the message. This lease claim get required to 425 file system using Secor. Job status is stored to database. Add Workers Add Workers Threading		work.		
pro- cessing the main unit of work UI Overview of Tasks UI Task status is stored of adatabase. UI Task status is stored of adatabase. Detailed, looks like Sidekiq Overview of Tasks only Task status is stored of a database. Detailed, looks like Sidekiq Operators mark jobs as passed or failed, to clear ull jobs with older bearbeats. Scaling Create multiple Tasks DAGs can be constructed with multiple Operators. Scale out by adding Celery workers Database out by adding Celery workers Subprocess Execution dependency ment Code Code Code State Uses SQLAlchemy for abstracting away the choice of and querying the database(Mysql/Postgresqlbways checked based on the existence of its output. Taraks Yes to db Sistory Morkers 'Claim' messages from the queue with an ownership timestamp on the message. This lease claim get required to 425 file system using Secor. Job status is stored to database. Add Workers Add Workers Threading	Class	Tasks/Workers	Operators	Jobs/Workers
Cose Code	pro-		- r · · · · · ·	
metamain unit of work UI Overview of Tasks only metamatada obstatus status at the status of the status of the status of the status of the choice of a database. Similar to Airflow, but for clearly metamata to Airflow, but for clear of the choice of an experiment of the choice of the				
main unit of work UI Overview of Tasks only metal- data/ob status is stored in a database. metal- data/ob status is stored in a database. Job status is stored in a database. Workers 'claim' messages from the queue with an ownership timestamp on the message. This lease claim gets renewed frequently. Messages with older lease claim gets renewed frequently. Messages are renewed frequently. Messages from the question of the gets renewed frequently. Messages are				
unit of work UI Overview of Tasks only only Task status is stored only only only only only only only only				
Work UT				
Detailed, looks like Sidekiq				
meta- data/job status Task status is stored in database. Similar to Airflow, but fewer details. Task status is stored in database. Operators mark jobs as passed or failed. Last updated is refreshed frequently with a details. Scaling Create multiple Tasks Scaling Subprocess Scaling Create multiple Tasks Subprocess Threading Yes to db Subprocess Yes to db Subprocess Yes to db Subprocess Yes to db Subprocess Yes to db Subprocess Yes to db Yes to db Subprocess Yes to db Yes to db Subprocess Subprocess Subprocess Subprocess Subprocess Subprocess Subprocess Subproces				
Task status is stored Job status is stored in a database. Operators mark jobs as passed or failed. Last updated is refreshed frequently with a heartbeat function. kill_zombies() is called to clear all jobs with older heartbeats. DAGs can be constructed with multiple Operators. Scale out by adding Celery workers	UI	Overview of Tasks	Comprehensive, with multiple screens	Detailed, looks like Sidekiq
data/job status In database. Similar to Airflow, but fewer details. Care to Airflow, but details. Ca		only		
data/job status In database. Similar to Airflow, but fewer details. Care to Airflow, but details. Ca	meta-	Task status is stored	Job status is stored in a database.	Workers 'claim' messages from the queue with an
Status to Airflow, but fewer details.				
details. a hear/beat function. kill_zombies() is called to clear all jobs with older heartbeats.	-	I .		
Create multiple Tasks DAGs can be constructed with multiple Operators. Scale out by adding Celery workers	status			
Scaling Create multiple Tasks DAGs can be constructed with multiple Operators. Scale out by adding Celery workers Add Workers		uctails.		
Create multiple Tasks DAGs can be constructed with multiple Operators. Scale out by adding Celery workers				
parallel execution parallel execution Tasks can be depend constructed with requires() method agement Code Code Code State uses SQLAlchemy persistence the database(Mysql/Postgrtsqdbways checked based on the existence of its output. Tracks history Tracks history Threading Operators. Scale out by adding Celery workers Subprocess Threading Jobs can require other jobs to finish first before starting, eg child_job requires parent_job. Python dict+code Yes to db Tracks history Tracks history Threading Threading Jobs can require other jobs to finish first before starting, eg child_job requires parent_job. Python dict+code Yes to db Tracks history Tracks history Threading Threading				
parallel Subprocess Threading	scaling	Create multiple Tasks		Add Workers
Description Subprocess Subprocess Subprocess Threading			Operators. Scale out by adding Celery	
Code Code Code It supports any store that is supported by the choice of and querying the database(Mysql/Postgressqlbways checked based on the existence of its output.			workers	
Code Code Code It supports any store that is supported by the choice of and querying the database(Mysql/Postgressqlbways checked based on the existence of its output.	parallel	Subprocess	Subprocess	Threading
tion dependency Tasks can be constructed with requires() method agement Code Code Code State persistence of the constructed of a state persistence of the database(Mysql/Postgressaging queue/message broker fault toler-ance ladoop pig yes pig yes pes pig yes pes pig hive yes pessil depends on past parameter of the depends on past parameter of the constructed with depends on past parameter of the constructed with depends on past parameter of the persistence of the depends on past parameter of the pends of the pends on past parameter of the pends of the pends on past parameter of the pends of the pends of the pends on past parameter of the pends	_	1		
Tasks can be constructed with requires() method age-ment				
dency manage		Tasks can be	Operators can be constructed with	Jobs can require other jobs to finish first before
man- age- ment Code State persis- tence Tracks history mes- saging queue/message broker fault toler- ance hadoop pig pes pes per pes per pes per				
age- ment Code Code State state persis- tence Tracks history mes- saging queue/message broker fault toler- ance hadoop pig pig pyes pig pig pyes pig pyes pig pig pyes pyes pyes pig pyes pyes pyes pyes pyes pyes pyes pyes			depends_on_past parameter	starting, eg enna_joo requires parent_joo.
ment Code Code Code State state persis- tence Lit supports any store that is supported by for abstracting away the choice of and querying the database(Mysql/Postgresqdlways checked based on the existence of its output. Litracks history mes- saging queue/message broker fault toler- ance hadoop pig yes pig yes pig hive yes pgsql mysql yes pressland Code Code Code Python dict+code Pyes to db Pyes to db Yes to db Yes to db Yes to db Yes Ves to db Yes Yes to db Yes Yes to db Yes Yes to db Yes Yes Yes Yes Mo Celery and RabbitMQ/Reddis No Yes Yes Yes The code Python dict+code Python dict+code Python dict+code Python dict+code Python dict+code Yes to db Yes Yes The code The code Yes The code Yes The code Yes The code The code Yes The code The co		requires() method		
Code Code Code State Uses SQLAlchemy It supports any store that is supported by Yes to db	_			
state persistence where the choice of and querying the database(Mysql/Postgresqd) ways checked based on the existence of its output. tracks history messaging queue/message broker fault tolerance hadoop pig yes pig live where yes pegal yes pesql yes mosqued yes pesql yes yes pesql yes yes mog mysql yes yes yes mog redshift no own yes yes yes no mog yes yes yes mog redshift no own yes yes yes no mog yes yes yes no no yes yes yes no mog yes yes yes no no yes				
persistence of and the choice of and querying the database(Mysql/Postgresqll)ways checked based on the existence of its output. tracks history messaging queue/message broker fault tolerance hadoop yes pig yes pig pig yes pgsql yes yes pgsql yes yes mo mo mysql yes yes yes mo no mysql yes yes yes mo no mysql redshift no mo yes makes task is not saved (only log file). The status of a task is not sa	Code			
tence the choice of and querying the database(Mysql/Postgresqllways checked based on the existence of its output. Tracks Yes to db history messaging queue/message broker fault No Yes tolerance hadoop yes yes pig yes pig yes pig yes pig gallow yes yes pgsql yes pgsql yes pgsql yes pgsql yes yes pgsql yes yes yes mo mysql yes yes yes mo mo mysql yes yes yes no mo mo mysql yes yes yes no mo mysql yes yes no mo mysql yes yes yes no no mysql yes yes yes no mo mysql yes yes yes yes no mysql yes	state		It supports any store that is supported by	Yes to db
querying the database(Mysql/Postgresql)ways checked based on the existence of its output. tracks history messaging queue/message broker fault tolerance hadoop yes yes pig yes pig yes yes pig yes pig yes yes pgsql yes yes yes pssql yes yes yes mo mysql yes yes yes no mo mysql yes yes yes no mo mysql yes yes yes no mo no mysql yes yes yes no no no no mysql yes yes yes no no no no mysql yes yes yes no no no no no mysql yes yes yes no no no no mysql yes yes yes no no no no no mysql yes yes yes no	persis-	for abstracting away	SQL Alchemy. If you don't use a	
querying the database(Mysql/Postgresql)ways checked based on the existence of its output. tracks history mes-saging queue/message broker fault toler-ance hadoop yes yes pig yes yes pig yes yes pgsql yes yes pgsql yes yes yes moo mysql yes yes yes moo mysql yes yes yes no mo no mysql yes yes yes no mo no mysql yes yes yes no mo no moo mysql yes yes yes no mo no mysql yes yes yes no mo no mysql yes yes yes no no no no mysql yes yes yes yes no no no mysql yes	tence	the choice of and	external store, the state is not saved	
database(Mysql/Postgresql]ways checked based on the existence of its output. tracks		querying the		
tracks history mes- saging queue/message broker fault toler- ance hadoop yes yes yes pig yes yes yes pgsql yes yes yes pgsql yes yes yes pgsql yes yes yes yes no mysql yes yes yes no mo yes yes yes no mo mysql yes yes yes no no redshift no yes yes no no no yes no				
tracks history mes- saging queue/message broker fault No Yes hadoop yes yes yes no hive yes yes pgsql yes redshift no yes mysql yes yes yes no yes no no no mysql redshift no yes Yes to db Yes no yes yes yes no				
history mes- saging queue/message broker fault No Yes hadoop yes yes pig yes yes hive yes yes pgsql yes yes mysql yes yes redshift no yes No Celery and RabbitMQ/Reddis No Yes Yes Yes Yes Yes Yes Yes Yes	tracks	Ves to dh		Ves to dh
mes- saging queue/message broker fault No Yes Yes toler- ance hadoop yes yes yes pig yes yes yes hive yes yes yes pgsql yes yes mysql yes yes redshift no yes No Celery and RabbitMQ/Reddis No Yes Yes Yes Yes Yes Yes Yes Ye		103 10 40	103	103 10 40
saging queue/message broker fault No Yes Yes toler- ance hadoop yes yes yes pig yes yes no hive yes yes yes pgsql yes yes mysql yes yes no redshift no yes no	- ·	N.T.	Colomo and Dalli (MO/D 11)	N-
queue/message broker Yes fault toler- ance No Yes hadoop pig hive pig hive yes pgsql mysql redshift yes yes yes yes yes no yes yes no no no mysql redshift yes no no no no no no no no no no no no no		INO	Celery and KaddiliviQ/Reddis	INO
broker Yes fault toler-ance No Yes hadoop pig yes pig yes pig yes yes pig yes yes pig yes yes yes pgsql yes yes pgsql yes yes pgsql yes yes no mo mysql yes yes no mo no no mysql yes yes yes no mo mysql yes yes no no no no				
fault toler-ance No Yes hadoop pig yes pig yes pig yes yes pig yes yes pig yes yes yes pgsql yes yes yes pgsql yes yes yes no mo mysql yes yes no mo no mysql yes yes no no no no no		ssage		
toler- ance hadoop yes yes yes pig yes yes no hive yes yes yes pgsql yes yes no mysql yes yes no redshift no yes no				
ance yes yes hadoop yes yes pig yes yes hive yes yes pgsql yes yes mysql yes no redshift no yes no	fault	No	Yes	Yes
ance yes yes hadoop yes yes pig yes yes hive yes yes pgsql yes yes mysql yes no redshift no yes no	toler-			
hadoop yes yes pig yes yes piyes yes no hive yes yes pgsql yes yes pgsql yes no mysql yes no redshift no yes no				
pig yes yes no hive yes yes yes pgsql yes yes no mysql yes yes no redshift no yes no		ves	ves	ves
hive yes yes pgsql yes yes mysql yes no redshift no yes no		•	-	1 *
pgsqlyesyesnomysqlyesyesnoredshiftnoyesno		•	-	
mysqlyesyesnoredshiftnoyesno			-	•
redshift no yes no		•	-	
		<u>-</u>	-	
s3 ves ves ves			-	
J J J		Vec	ves	l ves

IV. ARCHITECTURE OF OUR SYSTEM

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity Magnetization, or Magnetization, M, not just M. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write Magnetization (A/m) or Magnetization A[m(1)], not just A/m. Do not label axes with a ratio of quantities and units. For example, write Temperature (K), not Temperature/K.

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi TIFF or EPS file, with all fonts embedded) because, in an document, this method is somewhat more stable than directly inserting a picture.

Fig. 1. Inductance of oscillation winding on amorphous magnetic core versus DC bias magnetic field

V. INSTALLATION OF AIRFLOW

A. For single node

apt-get update

Install Dependencies

apt-get install unzip

Installing and configuring Apache Airflow

```
Installing and configuring Apache Airflow
```

```
apt-get install build-essential
apt-get install python-dev
apt-get install libsasl2-dev
apt-get install python-pandas
Installing Pip
cd /tmp/
wget https://bootstrap.pypa.io/ez_setup.py
python ez_setup.py
unzip setuptools-X.X.zip
cd setuptools-X.X
easy_install pip
Install MySQL
sudo apt-get install mysql-server
apt-get install libmysqlclient-dev
pip install MySQL-python
Install RabbitMO
apt-get install rabbitmq-server
Install airflow and required libraries
pip install airflow=1.7.0
pip install airflow[mysql]
pip install airflow[rabbitmq]
pip install airflow[celery]
Configuring Airflow
Changes in airflow configuration file at {AIRFLOW_HOME}/airflow.cfg
executor = CeleryExecutor
sql_alchemy_conn = mysql://root:root@localhost:3306/airflow
broker_url = amqp://guest:guest@localhost:5672/
celery_result_backend = db+mysql://root:root@localhost:3306/airflow
On Master execute following initialization commands (Initialize the Airflow database, start th
service rabbitmq-server start
airflow initdb
airflow webserver
airflow scheduler
airflow flower
```

On Worker execute the following commands (Initialize Airflow worker) airflow worker

B. For multi-node setup

VI. CONCLUSIONS

A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

APPENDIX

Appendixes should appear before the acknowledgment.

ACKNOWLEDGMENT

REFERENCES

- [1] https://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf
- [2] http://bytepawn.com/luigi-airflow-pinball.html
- [3] https://www.slideshare.net/r39132/airflow-agari-63072756
- [4] https://www.slideshare.net/erikbern/luigi-presentation-nyc-data-science
- [5] https://www.slideshare.net/growthintel/a-beginners-guide-to-building-data-pipelines-with-luigi
- [6] https://www.michaelcho.me/article/data-pipelines-airflow-vs-pinball-vs-luigi
- [7]
- [8] [9]