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.

C. Comparison of existing systems

Feature	Luigi	Airflow	Pinball
Data Pipeline	Tasks are grouped together into a DAG to be run. Most of the code treats Tasks as the main unit of work.	DAG (Directed Acyclic Graph) is used to define Jobs.	Workflow
Class processing the main unit of work	Tasks/Workers	Operators	Jobs/Workers
UI	Overview of Tasks only	Comprehensive, with multiple screens	Detailed, looks like Sidekiq
meta-data/job status	Task status is stored in database. Similar to Airflow, but fewer details.	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.	Workers 'claim' messages from the queue with an ownership timestamp on the message. This lease claim gets renewed frequently. Messages with older lease claims are requeued. Messages successfully processed are archived to S3 file system using Secor. Job status is stored to database.
scaling	Create multiple Tasks	DAGs can be constructed with multiple Operators. Scale out by adding Celery workers	Add Workers
parallel execution	Subprocess	Subprocess	Threading
dependency management	Tasks can be constructed with requires() method	Operators can be constructed with depends_on_past parameter	Jobs can require other jobs to finish first before starting, eg child_job requires parent_job.
Code	Code	Code	Python dict+code
state persistence	uses SQLAlchemy for abstracting away the choice of and querying the database(Mysql/Postgresql)	It supports any store that is supported by SQL Alchemy. If you don't use a external store, the state is not saved (only log file). The status of a task is always checked based on the existence of its output.	Yes to db
tracks history	Yes to db	Yes	Yes to db
messaging queue/message broker	No	Celery and RabbitMQ/Reddis	No
fault tolerance	No	Yes	Yes
hadoop	yes	yes	yes
pig	yes	yes	no
hive	yes	yes	yes
pgsql	yes	yes	no
mysql	yes	yes	no
redshift	no	yes	no
s3	yes	yes	yes

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

Installing and configuring Apache Airflow

```
Installing and configuring Apache Airflow

Install Dependencies
apt-get update
apt-get install unzip
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 RabbitMQ
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]