Building an Open Source Data Science Platform

https://goo.gl/ZnrR98
Agenda

- Distributed TF, Horovod, Rendezvous Architecture, Serving
- Model Optimization, Feature Store, Hyperparameter Opt
- MLFlow, Jenkins
- Spark, KubeFlow, TF Serving
- TensorFlow, Jupyter
- What is Machine Learning?
Continuous Integration

Monitoring & Operations

Data & Streaming → Model Engineering → Model Training → Model Management → Model Serving

- Distributed Data Storage and Streaming
- Data Preparation and Analysis
- Distributed Training using Machine Learning Frameworks
- Storage of trained Models and Metadata
- Use trained Model for Inference

Feature Catalogue

- Jenkins
- Continuous Integration

- TensorFlow Hub
- Model Library

- DC/OS
- Resource and Service Management

- kubernetes
- MESOS
Challenge...

Sunday, October 21 • 13:30 - 17:15

Workshop: Building and Operating an OSS Data Science Platform - Jörg Schad, Mesosphere

Click here to remove from My Sched.
Who are you?

What is your ML experience?

What do you expect to learn today?

- Distributed TF, Horovod, Rendezvous Architecture, Serving
- Model Optimization, Feature Store, Hyperparameter Opt
- MLFlow, Jenkins
- Spark, KubeFlow, TF Serving
- TensorFlow, Jupyter
- What is Machine Learning?
Jörg Schad

Technical Lead/Engineer
Deep Learning

- Core Mesos developer at Mesosphere

- Twitter: @joerg_schad
Machine Learning

- Why do we care
- What is Machine Learning
- Data Science Principles
- Different Personas

- Distributed TF, Horovod, Rendezvous Architecture, Serving
- Model Optimization, TensorFlow Hub, Feature Store
- MLFlow, Jenkins
- Spark, KubeFlow, TF Serving
- TensorFlow, Jupyter

What is Machine Learning?
Why is machine learning taking off?
AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery

Izhar Wallach  
Atomwise, Inc.  
izhar@atomwise.com

Michael Dzamba  
Atomwise, Inc.  
misko@atomwise.com

Abraham Heifets  
Atomwise, Inc.  
abe@atomwise.com
DEEPBACH: A STEERABLE MODEL FOR BACH CHORALES
GENERATION
What you want to be doing

Get Data -> Write intelligent machine learning code -> Train Model -> Run Model

Repeat
What you’re actually doing

- Configuration
- Data Collection
- Data Verification
- ML Code
- Feature Extraction
- Machine Resource Management
- Analysis Tools
- Process Management Tools
- Serving Infrastructure
- Monitoring

SOFTWARE ENGINEERING

Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968

Chairman: Professor Dr. F. L. Bauer
Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell
Do we need Data Science Engineering Principles?
I have a different controversial opinion: ML development is a different kind of software development and has a different set of best practices.
Challenge: Requirements Engineering

• Do I need Machine Learning? *
• Do I need {Neural Networks, Regression,...}* 

• What dataset(s)?
  – Quality?
• What target/serving environment?
• What model architecture?
• Pre-trained model available?
• How many training resources?
• Required Model Freshness?

* Can I actually use ...
Machine Learning

Fast.ai
Deep Learning: The Promise

TRADITIONAL MACHINE LEARNING

Input → Feature extraction → Classification → Output

DEEP LEARNING

Input → Feature extraction + classification → Output
Deep Learning: Insights

Deep Learning: The Process

**Step 1: Training**
(In Data Center - Over Hours/Days/Weeks)

- **Input:** Lots of Labeled Data
- Deep neural network model
- **Output:** Trained Model

**Step 2: Inference**
(Endpoint or Data Center - Instantaneous)

- **New Input from Camera or Sensor**
- **Trained Model**
- **Output:** Classification

Dog

Input:
Lots of Labeled Data

Output:
Trained Model

Deep neural network model

97% Dog
3% Panda
Deep Learning: The Process

Step 1: Training
(In Data Center - Over Hours/Days/Weeks)

Input: Lots of Labeled Data

Deep neural network model

Output: Trained Model

Step 2: Inference
(Endpoint or Data Center - Instantaneous)

New Input from Camera or Sensor

Trained Model

97% Dog
3% Panda

Output: Classification
Challenge: Persona(s)
The Rise of the *DataOps Engineer*

Combines two key skills:
- Data science
- Distributed systems engineering

The equivalent of *DevOps* for *Data Science*
TensorFlow & Jupyter

- First Hands-On Machine Learning
- **Open Source Technologies**
  - TensorFlow
  - Jupyter
- **Labs**
  - Mnist with Google Colab
  - Deploy and use Jupyter

- Distributed TF, Horovod, Rendezvous Architecture, Serving
- Model Optimization, Feature Store, Hyperparameter Opt
- MLFlow, Jenkins
- Spark, KubeFlow, TF Serving
- **TensorFlow, Jupyter**
- What is Machine Learning?
TensorFlow Overview

“An open-source software library for Machine Intelligence” - tensorflow.org

- Tensorflow is a software library that makes it easy for developers to construct artificial neural networks to analyze their data of interest

Diagram:

- Python
- TensorFlow Library
  - Dataflow Executor, Compute Kernel Implementations, Networking, etc.
  - GPUs
  - CPUs
ML Frameworks Overview
import tensorflow as tf
import numpy as np

X = tf.placeholder("float")
Y = tf.placeholder("float")
W = tf.Variable(np.random.rand(), name="weight")
pred = tf.multiply(X, W)
cost = tf.reduce_sum(tf.pow(pred-Y, 2))

optimizer = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)

    for t in range(10000):
        x = np.array(np.random.rand()).reshape((1, 1, 1, 1))
        y = x * 3
        (_ , c) = sess.run([optimizer, cost], feed_dict={X: x, Y: y})
    print c

import numpy as np
import torch

from torch.autograd import Variable
model = torch.nn.Linear(1, 1)
loss_fn = torch.nn.MSELoss(size_average=False)
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

for t in range(10000):
    x = Variable(torch.from_numpy(np.random.rand((1,1)).astype(np.float32)))
    y = x * 3
    y_pred = model(x)
    loss = loss_fn(y_pred, y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    print(loss.data[0])
Alternatives

import tensorflow as tf
import numpy as np
X = tf.placeholder("float")
Y = tf.placeholder("float")
W = tf.Variable(np.random.random(), name="weight")
pred = tf.multiply(X, W)
cost = tf.reduce_sum(tf.pow(pred - Y, 2))
optimizer = tf.train.GradientDescentOptimizer
init = tf.global_variables_initializer()
with tf.Session() as sess:
    sess.run(init)
    for t in range(10000):
        x = np.array(np.random.random()).reshape((1, 1, 1, 1))
y = x * 3
(_, c) = sess.run([optimizer, cost], feed_dict={X: x, Y: y})
print c

import numpy as np
import torch
from torch.autograd import Variable
model = torch.nn.SequentialLinear(1, 1)
loss_fn = torch.nn.MSELoss(size_average=False)
optimizer = optim.SGD(model.parameters(), lr=0.01)

for t in range(10000):
    x = torch.from_numpy(np.random.random((1, 1)).astype(np.float32)).
    y_pred = model(x)
    loss = loss_fn(y_pred, y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    print loss.data[0]
Challenge: Writing Distributed Model Functions
TensorFlow Estimator & Keras APIs

• Prefered APIs

```python
return tf.estimator.Estimator(
    model_fn=model_fn,  # First-class function
    params=params,      # HParams
    config=run_config  # RunConfig
)
```
Lab 0: Fashion Mnist

Train your first neural network: basic classification

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It’s okay if you don’t understand all the details, this is a fast-paced overview of a complete TensorFlow program with the details explained as we go.

This guide uses tf.keras, a high-level API to build and train models in TensorFlow.

```python
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
```

https://www.tensorflow.org/tutorials/keras/basic_classification
Lab 1: JupyterLab

DataOps: Setup Environment

Data Scientist: Use distributed resources

1. Install HDFS
2. Install Marathon-LB (Proxy)
3. Install Jupyter

https://github.com/dcos/demos/tree/master/jupyterlab/1.11
Lab 1: Connect to Cluster

USER: bootstrapuser
Password: deleteme
Lab 1: HDFS

https://github.com/dcos/demos/tree/master/jupyterlab/1.11
Lab 1: Marathon-LB

Description
HAProxy configured using Marathon state

Preinstall Notes: We recommend at least 2 CPUs and 1GiB of RAM for each Marathon-LB instance. NOTE: For additional Enterprise Edition DC/OS instructions, see https://docs.mesosphere.com/administration/id-and-access-mgt/service-auth/mlb-auth/
# Lab 1: Jupyter

---

<table>
<thead>
<tr>
<th>Edit Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jupyterlab 1.2.0-0.33.7</td>
</tr>
</tbody>
</table>

### Networking
DC/OS JupyterLab networking configuration properties

### Cni Support
Enable Container Networking Interface (CNI) Support.

- enabled ✅

### External Access
Enable access from outside the cluster through Marathon-LB. NOTE: this connection is unencrypted.

- enabled ✅

**external public agent hostname** ✪

- `ioerg-tflts-pub-agt-elb-1895779527.us-west`

---

[https://github.com/dcos/demos/tree/master/jupyterlab/1.11](https://github.com/dcos/demos/tree/master/jupyterlab/1.11)
Lab 1: Jupyter

http://api.hdfs.marathon.l4lb.thisdcos.directory/v1/endpoints

https://github.com/dcos/demos/tree/master/jupyterlab/1.11
Lab 1: Jupyter

Storage

Networking

Environment

warn

- start dask distributed
- start ray head node
- start spark history server
- start tensorboard

https://github.com/dcos/demos/tree/master/jupyterlab/1.11
Lab 1: Jupyter

https://github.com/dcos/demos/tree/master/jupyterlab/1.11
Lab 1: Jupyter

```scala
val NUM_SAMPLES = 10000000

val count2 =
spark.sparkContext.parallelize(1 to NUM_SAMPLES).map{i =>
    val x = Math.random()
    val y = Math.random()
    if (x*x + y*y < 1) 1 else 0
}.reduce(_ + _)

println("Pi is roughly " + 4.0 * count2 / NUM_SAMPLES)
```

https://github.com/dcos/demos/tree/master/jupyterlab/1.11
Lab 1: Jupyter

```bash
eval \
    spark-submit \n    ${SPARK_OPTS} \n    --verbose \n    --class \n    org.apache.spark.examples.SparkPi \n
/opt/spark/examples/jars/spark-example\n\ns_2.11-2.2.1.jar 100
```

https://github.com/dcos/demos/tree/master/jupyterlab/1.11
Lab 1: Jupyter

<table>
<thead>
<tr>
<th>Framework ID</th>
<th>Task ID</th>
<th>Task Name</th>
<th>Role</th>
<th>State</th>
<th>Health</th>
<th>Started</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>...e512ac82c5c4-0007</td>
<td>4</td>
<td>Spark PI 4</td>
<td>*</td>
<td>RUNNING</td>
<td>-</td>
<td>just now</td>
<td>10.0.4.225 Sandbox</td>
</tr>
<tr>
<td>...e512ac82c5c4-0007</td>
<td>2</td>
<td>Spark PI 2</td>
<td>*</td>
<td>RUNNING</td>
<td>-</td>
<td>just now</td>
<td>10.0.4.225 Sandbox</td>
</tr>
<tr>
<td>...e512ac82c5c4-0007</td>
<td>0</td>
<td>Spark PI 0</td>
<td>*</td>
<td>RUNNING</td>
<td>-</td>
<td>just now</td>
<td>10.0.4.225 Sandbox</td>
</tr>
<tr>
<td>...e512ac82c5c4-0007</td>
<td>3</td>
<td>Spark PI 3</td>
<td>*</td>
<td>RUNNING</td>
<td>-</td>
<td>just now</td>
<td>10.0.4.239 Sandbox</td>
</tr>
<tr>
<td>...e512ac82c5c4-0007</td>
<td>1</td>
<td>Spark PI 1</td>
<td>*</td>
<td>RUNNING</td>
<td>-</td>
<td>just now</td>
<td>10.0.4.239 Sandbox</td>
</tr>
<tr>
<td>...e512ac82c5c4-0001</td>
<td>jupyterlab-notebook.b6df434b-9a5f-11e8-a8c2-461a4e765579</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...e512ac82c5c4-0001</td>
<td>jupyterlab-notebook</td>
<td>slave_public</td>
<td>RUNNING</td>
<td>Healthy</td>
<td>8 hours ago</td>
<td>10.0.4.225 Sandbox</td>
<td></td>
</tr>
</tbody>
</table>

[https://github.com/dcos/demos/tree/master/jupyterlab/1.11](https://github.com/dcos/demos/tree/master/jupyterlab/1.11)
First Pipeline

- Pipeline overview
  - KubeFlow
  - TFX
  - Michelangelo

- Open Source Technologies
  - Kubeflow
  - TF Serving

- Labs
  - [opt] Serving
  - [opt] KubeFlow

---

Distributed TF, Horovod, Rendezvous Architecture, Serving

Model Optimization, TensorFlow Hub, Feature Store

MLFlow, Jenkins

Kubeflow, TF Serving

TensorFlow, Jupyter, Spark

What is Machine Learning?
Challenge: Serving

- How to Deploy Models?
  - Zero Downtime
  - Canary
- Multiple Models?
  - Testing

- TensorFlow Serving
Challenge: Serving Environment

- How to Deploy Models?
  - Zero Downtime
  - Canary
- Multiple Models?
  - Testing

https://ai.googleblog.com/2016/02/running-your-models-in-production-with.html
# Download the TensorFlow Serving Docker image and repo

docker pull tensorflow/serving

git clone https://github.com/tensorflow/serving

# Location of demo models

TESTDATA="$(pwd)/serving/tensorflow_serving/servables/tensorflow/testdata"

# Start TensorFlow Serving container and open the REST API port

docker run -t --rm -p 8501:8501 \
    -v "$TESTDATA/saved_model_half_plus_two_cpu:/models/half_plus_two" \
    -e MODEL_NAME=half_plus_two \
    tensorflow/serving &

# Query the model using the predict API

curl -d '{"instances": [1.0, 2.0, 5.0]}' \
    -X POST http://localhost:8501/v1/models/half_plus_two:predict

# Returns => { "predictions": [2.5, 3.0, 4.5] }
@log(labels=_labels, logger=_logger)
def predict(request: bytes) -> bytes:

https://community.cloud.pipeline.ai/admin/app
1. Data Preparation & Model Engineering
2. Model Training
3. Monitoring
4. Debugging
5. Model Serving
Public Cloud Pipeline

1. Data Preparation using Spark

2. Model Training

3. Monitoring

4. Debugging

5. Model Serving

6. 7. Streaming of requests
DIY Open Source Pipeline

1. Data Preparation using Spark
2. Model Training
3. Monitoring
4. Debugging
5. Model Serving
6. HDFS
7. Kafka stream of requests
Data Science Platforms

- **AWS Sagemaker**
  + Spark, MXNet, TF
  + Serving/AB
  - Cloud Only

- **Google Datalab/ML-Engine**
  + TF, Keras, Scikit, XGBoost
  + Serving/AB
  - Cloud Only
  - No control of docker images

- **KubeFlow**
  + TF Everywhere
  - TF only

https://medium.com/intuitionmachine/google-and-ubers-best-practices-for-deep-learning-58488a8899b6
TFX: A TensorFlow-Based Production-Scale Machine Learning Platform

Figure 1: High-level component overview of a machine learning platform.

https://www.youtube.com/watch?v=fPTwLVCq00U
“..there were no systems in place to build reliable, uniform, and reproducible pipelines for creating and managing training and prediction data at scale.”

- Feature store (later)

https://eng.uber.com/michelangelo/
Challenge: Data Quality

• Data is typically not ready to be consumed by ML job*
  – Data Cleaning
    • Missing/incorrect labels
  – Data Preparation
    • Same Format
    • Same Distribution

* Demo datasets are a fortunate exception :)
Challenge: Data Quality

- Data is typically not ready to be consumed by ML job*
  - Data Cleaning
    - Missing/incorrect labels
  - Data Preparation
    - Same Format
    - Same Distribution

* Demo datasets are a fortunate exception :)

Don’t forget about the serving environment!!
Spark Overview

- Unified Analytics Engine
  - Started as batch system
  - < 2.0: Spark Streaming
    - Micro-Batches
  - > 2.0 Structure Streaming
    - Native Streaming
Batch vs Streaming

Batch:
- Sensors
- Devices
- Clients

Streaming:
- Sensors
- Devices
- Clients

Batch Stream:
- Bounded Data
- Batches

Streaming Stream:
- Unbounded Stream

ANALYZE

ACT
Lab 3: Spark Data Cleaning

$ git clone https://github.com/yahoo/TensorFlowOnSpark

$ cd $MESOS_SANDBOX
$ curl -fsSL -O https://s3.amazonaws.com/vishnu-mohan/tensorflow/mnist/mnist.zip
$ unzip mnist.zip

$ // Should return error
$ hdfs dfs -ls mnist/

https://github.com/dcos-labs/dcos-jupyterlab-service/blob/master/notebooks/TFoS.ipynb
Lab 1: Spark Data Cleaning

```
eval \
    spark-submit \
    ${SPARK_OPTS} \
    --verbose \
$(pwd)/TensorFlowOnSpark/examples/mnist/mnist_data_setup.py \
    --output mnist/csv \
    --format csv

$ // Should not return an error
$ hdfs dfs -ls mnist/
```

https://github.com/dcos/demos/tree/master/jupyterlab/1.11
Lab: (Optional) KubeFlow

https://www.kubeflow.org/docs/started/getting-started-multipass/
Automation

- Need for reproducibility
  - MLFlow
  - Jenkins
- Open Source Technologies
  - MLFlow
  - Jenkins
- Labs
  - Jenkins
  - [opt] MLFlow

- Distributed TF, Horovod, Rendezvous Architecture, Serving
- Model Optimization, TensorFlow Hub, Feature Store
- MLFlow, Jenkins
- Kubeflow, TF Serving
- TensorFlow, Jupyter, Spark
- What is Machine Learning?
Challenge: Reproducible Builds

- Many adhocs model/training runs
- Regulatory Requirements
- Dependencies
- CI/CD
- Git

Step 1: Training
(In Data Center - Over Hours/Days/Weeks)

Input: Lots of Labeled Data

Deep neural network model

Output: Trained Model
MFlow

An open source platform for the machine learning lifecycle

https://mlflow.org/
MFlow

Tracking
Record and query experiments: code, data, config, results

Projects
Packaging format for reproducible runs on any platform

Models
General format for sending models to diverse deploy tools
import mlflow

# Log parameters (key-value pairs)
mlflow.log_param("num_dimensions", 8)
mlflow.log_param("regularization", 0.1)

# Log a metric;
mlflow.log_metric("accuracy", 0.1)
...
mlflow.log_metric("accuracy", 0.45)

# Log artifacts (output files)
mlflow.log_artifact("roc.png")
mlflow.log_artifact("model.pkl")
MFlow Project

name: My Project
conda_env: conda.yaml
entry_points:
  main:
    parameters:
      data_file: path
      regularization: {type: float, default: 0.1}
    command: "python train.py -r {regularization} {data_file}"
validate:
  parameters:
    data_file: path
    command: "python validate.py {data_file}"
MFlow Model

time_created: 2018-02-21T13:21:34.12
flavors:
  sklearn:
    sklearn_version: 0.19.1
    pickled_model: model.pkl
python_function:
  loader_module: mlflow.sklearn
  pickled_model: model.pkl

$mlflow run example/project -P alpha=0.5
$mlflow run git@github.com:databricks/mlflow-example.git
Lab 2: MLflow

```
$ git clone https://github.com/mlflow/mlflow
$ cd mlflow/
$ python examples/sklearn_elasticnet_wine/train.py
$ mlflow run https://github.com/mlflow/mlflow-example.git -P alpha=0.42
$ mlflow sklearn serve -m ./mlruns/0/024e295715d64b3fb0730008a07c1f75/artifacts/model -p 1234
```

https://www.mlflow.org/docs/latest/quickstart.html
Lab 2: MLflow

curl -X POST -H "Content-Type:application/json" --data '[{"fixed acidity": 6.2, "volatile acidity": 0.66, "citric acid": 0.48, "residual sugar": 1.2, "chlorides": 0.029, "free sulfur dioxide": 29, "total sulfur dioxide": 75, "density": 0.98, "pH": 3.33, "sulphates": 0.39, "alcohol": 12.8}]' http://127.0.0.1:1234/invocations

{"predictions": [6.379428821398614]}
Challenge: CI/CD

- Automatic Build, Test, Deploy
- Quality Barrier
- Options
  - Jenkins
  - Gitlab
  - TravisCI
  - Skaffold
  - Spinnaker
CI/CD for Data Science

Train Model(s) → Optimize Model(s) → Test Model(s) → Build Serving Container(s) → Deploy

Continuous Integration

Model Engineering

Git

Jenkins
Lab 3: Jenkins
Lab 3: Jenkins

### Services

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>hdf5</td>
<td>Running</td>
<td>1</td>
</tr>
<tr>
<td>jenkins</td>
<td>Deploying</td>
<td>1</td>
</tr>
<tr>
<td>jupyterlab-notebook</td>
<td>Running</td>
<td>1</td>
</tr>
<tr>
<td>marathon-lb</td>
<td>Running</td>
<td>1</td>
</tr>
</tbody>
</table>
Lab 3: Jenkins

**Jenkins**

Enter an item name

DataSciencePipeline

- **Freestyle project**
  - This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

- **Maven project**
  - Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

- **Pipeline**
  - Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

- **External Job**
  - This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.
Lab 3: Jenkins

https://github.com/mesosphere/data-science-cicd
Lab 3: Jenkins

Pipeline DataSciencePipeline

Stage View

Average stage times:
(Average [all] run time: ~1min 33s)

<table>
<thead>
<tr>
<th></th>
<th>Checkout</th>
<th>Train Model(s)</th>
<th>Optimize Model(s)</th>
<th>Test</th>
<th>Build Serving container</th>
<th>Deploy</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>16s</td>
<td>10s</td>
<td>5s</td>
<td>17ms</td>
<td>80ms</td>
<td>626ms</td>
</tr>
</tbody>
</table>

Permalinks
Model Management

- How to manage Models
- **Open Source Technologies**
  - TensorFlow Hub
  - Dask
- **Labs**
  - Dask Hyperparameter Optimization

- Distributed TF, Horovod,
  - Model Optimization, Feature Store, Hyperparameter Opt
  - MLFlow, Jenkins
  - Kubeflow, TF Serving
  - TensorFlow, Jupyter, Spark
  - What is Machine Learning?
Challenge: Data (Preprocessing) Sharing

- Preprocessed Data Sets valuable
  - Sharing
  - Automatic Refresh

- Feature Catalogue ≈ Preprocessing Cache + Discovery
Feature Store

ML Feature Data Warehouse
Reduce the cost of generating and storing the feature data

Just-in-time Feature Transforms
Allow the research teams to experiment with new features and feature engineering techniques

“..there were no systems in place to build reliable, uniform, and reproducible pipelines for creating and managing training and prediction data at scale.”

- **Feature store**

https://eng.uber.com/michelangelo/
Feature Engineering

Featuretools
An open source python framework for automated feature engineering

https://www.featuretools.com/
Hyperparameter Optimization

Hyperparameter tuning vs. model training

Step 1: Training
(In Data Center - Over Hours/Days/Weeks)

- Input: Lots of Labeled Data
- Deep neural network model
- Output: Trained Model

● Networks Shape
● Learning Rate
● ...

https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debb07568
Hyperparameter Search

https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debb07568
Dask-ML for Hyperparameter Search

```python
# use a full grid over all parameters
param_grid = {
    "C": [1e-5, 1e-3, 1e-1, 1],
    "fit_intercept": [True, False],
    "penalty": ["l1", "l2"]
}

clf = LogisticRegression()

# run grid search
dk_grid_search = GridSearchCV(clf, param_grid=param_grid, n_jobs=-1)
sk_grid_search = ms.GridSearchCV(clf, param_grid=param_grid, n_jobs=-1)
```

Lab [Optional] Dask

Dask-ML

Dask-ML provides scalable machine learning in Python using Dask alongside popular machine learning libraries like Scikit-Learn.

You can try Dask-ML on a small cloud instance by clicking the following button:

```python
import dask.dataframe as dd
df = dd.read_parquet('...')
data = df[['age', 'income', 'married']]
labels = df['outcome']

from dask_ml.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(data, labels)
```

Challenge: Serving Environment

- Different Serving Environments
  - Mobile
  - GPU
  - CPU
- Small/Fast model without losing too much performance
- 500 KB models....
Model Optimization

```bash
transform_graph \
  --in_graph=unoptimized_cpu_graph.pb \  \< Original Graph
  --out_graph=optimized_cpu_graph.pb \ \< Transformed Graph
  --inputs='x_observed:0' \ \< Feed (Input)
  --outputs='Add:0' \ \< Fetch (Output)
  --transforms='\n    strip_unused_nodes
    remove_nodes(op=Identity, op=CheckNumerics)
    fold_constants(ignore_errors=true)
    fold_batch_norms
    fold_old_batch_norms
    quantize_weights
    quantize_nodes'
```
Model Optimization

<table>
<thead>
<tr>
<th></th>
<th>Dynamic Range</th>
<th>Min Pos Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP32</td>
<td>$-3.4 \times 10^{38}$ – $+3.4 \times 10^{38}$</td>
<td>$1.4 \times 10^{-45}$</td>
</tr>
<tr>
<td>FP16</td>
<td>$-65504$ – $+65504$</td>
<td>$5.96 \times 10^{-8}$</td>
</tr>
<tr>
<td>INT8</td>
<td>$-128$ – $+127$</td>
<td>1</td>
</tr>
</tbody>
</table>
Distributed TensorFlow

- How to distribute TensorFlow
  - {TF, Horovod} on Spark
- Open Source Technologies
  - TensorFlow
  - Spark
- Labs
  - TFonSpark

Model Optimization, TensorFlow Hub, Feature Store
MLFlow, Jenkins
Kubeflow, TF Serving
TensorFlow, Jupyter, Spark
What is Machine Learning?
TensorFlow Overview
Challenge: Distributed TensorFlow
Challenge: Distributed TensorFlow

https://eng.uber.com/horovod/
Horovod

- **All-Reduce** to update Parameter
  - Bandwidth Optimal
- Uber Horovod is MPI based
  - Difficult to set up
  - Other Spark based implementations
- **Wait for TensorFlow 2.0 ;)**

https://eng.uber.com/horovod/
TF Distribution Strategy

- **MirroredStrategy**: This does in-graph replication with synchronous training on many GPUs on one machine. Essentially, we create copies of all variables in the model's layers on each device. We then use all-reduce to combine gradients across the devices before applying them to the variables to keep them in sync.

- **CollectiveAllReduceStrategy**: This is a version of MirroredStrategy for multi-working training. It uses a collective op to do all-reduce. This supports between-graph communication and synchronization, and delegates the specifics of the all-reduce implementation to the runtime (as opposed to encoding it in the graph). This allows it to perform optimizations like batching and switch between plugins that support different hardware or algorithms. In the future, this strategy will implement fault-tolerance to allow training to continue when there is worker failure.

- **ParameterServerStrategy**: This strategy supports using parameter servers either for multi-GPU local training or asynchronous multi-machine training. When used to train locally, variables are not mirrored, instead they placed on the CPU and operations are replicated across all local GPUs. In a multi-machine setting, some are designated as workers and some as parameter servers. Each variable is placed on one parameter server. Computation operations are replicated across all GPUs of the workers.

https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/distribute
Challenge: “Libraries”

- Different Frameworks
- Existing architectures
- Pretrained models

```python
import tensorflow as tf
import tensorflow_hub as hub

with tf.Graph().as_default():
    embeddings = embed(["A long sentence.", "single-word", "http://example.com"])

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    sess.run(tf.tables_initializer())

print(sess.run(embeddings))
```
Lab 4: TensorFlow

```bash
eval \
  spark-submit \
  ${SPARK_OPTS} \
  --verbose \
  --conf \
  spark.mesos.executor.docker.image=dcoslabs/dcos-jupyterlab:1.2.0-0.33.7 \
  --py-files $(pwd)/TensorFlowOnSpark/examples/mnist/spark/mnist_dist.py \
  $(pwd)/TensorFlowOnSpark/examples/mnist/spark/mnist_spark.py \
  --cluster_size 5 \
  --images mnist/csv/train/images \ 
  --labels mnist/csv/train/labels \ 
  --format csv \ 
  --mode train \ 
  --model mnist/mnist_csv_model
```
Lab 4: TensorFlow
Further Reading/Watching

https://www.youtube.com/watch?v=tx6HyoUYGL0

Distributed Deep Learning with Apache Spark and TensorFlow
Jim Dowling (Logical Clocks AB)
Challenge: Testing

• Training/Test/Validation Datasets
• Unit Tests?
• Different factors
  – Accuracy
  – Serving performance
  – ….
• A/B Testing with live Data
Challenge: Debugging

https://www.tensorflow.org/programmers_guide/debugger
Challenge: Monitoring

- Understand {...}
- Debug
- Model Quality
  - Accuracy
  - Training Time
  - ...
- Overall Architecture
  - Availability
  - Latencies
  - ...

- TensorBoard

- Traditional Cluster Monitoring Tool
Lab 5: TensorBoard

To CSV mnist
Profiling

- Crucial when using “expensive” devices
- Memory Access Pattern
- “Secret knowledge”
- More is not necessarily better....

https://www.tensorflow.org/performance/performance_guide
Lab 6 (Optional): TFDebug

https://www.tensorflow.org/guide/debugger
Challenge: Resource and Service Management

• Different Distributed Systems
  – Deployment
  – Updates
  – Failure Recovery
  – Scaling
• Resource Efficiency
  – Multiple VM per Service?

Typical Datacenter
siloed, over-provisioned servers,
low utilization

Spark
TensorFlow
Jupyter
Jenkins
HDFS
Apache Mesos

Two-level Scheduling

1. Agents advertise resources to Master
2. Master offers resources to Framework
3. Framework rejects / uses resources
4. Agent reports task status to Master
Example: GPU Isolation

Mesos Agent

Containerizer API

(Unified)
Mesos
Containerizer

Isolator API

CPU

Memory

GPU
Example: GPU Isolation

Mesos Agent

Containerizer API

(Underground) Mesos Containerizer

Isolator API

CPU Memory GPU

Nvidia GPU Isolator

Nvidia GPU Allocator

Nvidia Volume Manager

Linux devices cgroup
Example: GPU Isolation

Mimics functionality of \texttt{nvidia-docker-plugin}

Allocates GPUs to tasks

Isolates Access to GPUs between tasks
Resource Management

Typical Datacenter
- siloed, over-provisioned servers,
  low utilization

DC/OS
- automated schedulers, workload multiplexing onto the same machines

Spark
TensorFlow
Jupyter
Jenkins
Microservices
Service Orchestration

LOB 1

- kubernetes
- Spark
- GitLab
- Jenkins

LOB 2

- kubernetes
- Spark
- GitLab
- Jenkins

Application-Aware Automation
Security & Compliance
Hybrid Cloud Management
Multitenancy

Datacenter and Cloud as a Single Computing Resource
Powered by Apache Mesos
Resource Management

- Application-Aware Automation
- Security & Compliance
- Hybrid Cloud Management
- Multitenancy

Datacenter and Cloud as a Single Computing Resource
Powered by Apache Mesos

REGION 1 (LOCAL)
- DC/OS Master 1
- DC/OS Master 2
- DC/OS Master 3

REGION 2 (AWS-WEST)
- Zone 1: aws-west-a
- Zone 2: aws-west-b
- Zone 3: aws-west-c

REGION 3 (GCP-EAST)

REGION 4 (AZURE-EUROPE)

Site-to-Site VPN
Low Latency
THANK YOU!

ANY QUESTIONS?

@dcos
chat.dcos.io
users@dcos.io
/groups/8295652
/dcos
/dcos/examples
/dcos/demos

https://mesosphere.com/resources/building-data-science-platform/
Make it insanely easy to build and scale world-changing technology

MESOSPHERE