Simplifying ML Workflows with Apache Beam & TensorFlow

Extended

Tyler Akidau
@takidau

Software Engineer at Google
Apache Beam PMC
Apache Beam
Portable data-processing pipelines
Example pipelines

Python

```
input | WindowInto(FixedWindows(3600),
            trigger=AfterWatermark())
    | Sum.PerKey()
    | Write(BigQuerySink(...))
```

Java

```
input
   .apply(Window.into(FixedWindows.of(...))
   .triggering(
      AfterWatermark.pastEndOfWindow())
   .apply(Sum.integersPerKey())
   .apply(BigQueryIO.Write.to(...))
```
Python compatible runners

Direct runner (local machine): Now

Google Cloud Dataflow: Now

Apache Flink: Q2-Q3

Apache Spark: Q3-Q4
TensorFlow Extended
End-to-end machine learning in production
“Doing ML in production is hard.”

- Everyone who has ever tried
Because, in addition to the actual ML...
...you have to worry about so much more.
In this talk, I will...
In this talk, I will...

Show you how to apply transformations...

TensorFlow
Transform
In this talk, we will...

Show you how to apply transformations... ... consistently between Training and Serving

TensorFlow Transform
TensorFlow Estimators
TensorFlow Serving
In this talk, we will...

Introduce something new...

TensorFlow Transform
TensorFlow Estimators
TensorFlow Model Analysis
TensorFlow Serving
TensorFlow Transform
Consistent In-Graph Transformations in Training and Serving
Typical ML Pipeline

During training

- data
- batch processing

During serving

- request
- “live” processing
Typical ML Pipeline

During training

- Data
  - Batch processing

During serving

- Request
  - "Live" processing
TensorFlow Transform

During training

data

tf.Transform batch processing

During serving

request

transform as tf.Graph
Defining a preprocessing function in TF Transform

```python
def preprocessing_fn(inputs):
    x = inputs['X']
    ...
    return {
        "A": tft.bucketize(
            tft.normalize(x) * y),
        "B": tensorflow_fn(y, z),
        "C": tft.ngrams(z)
    }
```

Many operations available for dealing with text and numeric features, user can define their own.
def preprocessing_fn(inputs):
    x = inputs['X']
    ...
    return {
        "A": tft.bucketize(tft.normalize(x) * y),
        "B": tensorflow_fn(y, z),
        "C": tft.ngrams(z)
    }

Many operations available for dealing with text and numeric features, user can define their own.
Defining a preprocessing function in TF Transform

```python
def preprocessing_fn(inputs):
    x = inputs['X']
    ...
    return {
        "A": tft.bucketize(tft.normalize(x) * y),
        "B": tensorflow_fn(y, z),
        "C": tft.ngrams(z)
    }
```

Many operations available for dealing with text and numeric features, user can define their own.
def preprocessing_fn(inputs):
    x = inputs['X']
    ...
    return {
        "A": tft.bucketize
            tft.normalize(x) * y,
        "B": tensorflow_fn(y, z),
        "C": tft.ngrams(z)
    }

Many operations available for dealing with text and numeric features, user can define their own.
Defining a preprocessing function in TF Transform

```python
def preprocessing_fn(inputs):
    x = inputs['X']
    ...
    return {
        "A": tft.bucketize(tft.normalize(x) * y),
        "B": tensorflow_fn(y, z),
        "C": tft.ngrams(z)
    }
```

Many operations available for dealing with text and numeric features, user can define their own.
Defining a preprocessing function in TF Transform

```python
def preprocessing_fn(inputs):
    x = inputs['X']
    ...
    return {
        "A": tft.bucketize(tft.normalize(x) * y),
        "B": tensorflow_fn(y, z),
        "C": tft.ngrams(z)
    }
```

Many operations available for dealing with text and numeric features, user can define their own.
Analyzers

Reduce (full pass)

Implemented as a distributed data pipeline

Transforms

Instance-to-instance (don’t change batch dimension)

Pure TensorFlow
Analyze

Normalize

Multiply

Bucketize

data

Constant tensors

Mean

Stddev

Normalize

Multiply

Quantiles

Bucketize
What can be done with TF Transform?

Pretty much anything.

tf.Transform batch processing
What can be done with TF Transform?

Anything that can be expressed as a TensorFlow Graph

Pretty much anything.
Some common use-cases...

Scale to ...

- `tft.scale_to_z_score`
- ...

Bucketization

- `tft.quantiles`
- `tft.apply_buckets`

Bag of Words / N-Grams

- `tf.string_split`
- `tft.ngrams`
- `tft.string_to_int`

Feature Crosses

- `tf.string_join`
- `tft.string_to_int`
Some common use-cases...

Scale to ...
- `tft.scale_to_z_score`
- ...

Bag of Words / N-Grams
- `tf.string_split`
- `tft.ngrams`
- `tft.string_to_int`

Bucketization
- `tft.quantiles`
- `tft.apply_buckets`

Feature Crosses
- `tf.string_join`
- `tft.string_to_int`

Apply another TensorFlow Model
- `tft.apply_saved_model`
github.com/tensorflow/transform
Introducing...

TensorFlow Model Analysis

Scaleable, sliced, and full-pass metrics
Let’s Talk about Metrics...

- How accurate?
- Converged model?
- What about my TB sized eval set?
- Slices / subsets?
- Across model versions?
ML Fairness: analyzing model mistakes by subgroup

ROC Curve

Sensitivity (True Positive Rate) vs. Specificity (False Positive Rate)

All groups

Learn more at ml-fairness.com
ML Fairness: analyzing model mistakes by subgroup

ROC Curve

- Sensitivity (True Positive Rate)
- Specificity (False Positive Rate)

- Orange: All groups
- Black: Group A
- Blue: Group B

Learn more at ml-fairness.com
ML Fairness: understand the failure modes of your models
ML Fairness: Learn More
How does it work?

```python
estimator = DNNLinearCombinedClassifier(...)  
estimator.train(...)  
estimator.export_savedmodel(  
    serving_input_receiver_fn=serving_input_fn)

tfma.export.export_eval_savedmodel (  
estimator=estimator,  
eval_input_receiver_fn=eval_input_fn)
```

Inference Graph (SavedModel)
How does it work?

... 
estimator = DNNLinearCombinedClassifier(...) 
estimator.train(...) 
estimator.export_savedmodel( 
    serving_input_receiver_fn=serving_input_fn) 
tfma.export.export_eval_savedmodel( 
    estimator=estimator, 
    eval_input_receiver_fn=eval_input_fn) 
...

Inference Graph (SavedModel)

Eval Graph (SavedModel)

SignatureDef

Eval Metadata

TF + B
Summary

**Apache Beam:** Data-processing framework that runs locally and scales to massive data, in the Cloud (now) and soon on-premise via Flink (Q2-Q3) and Spark (Q3-Q4). Powers large-scale data processing in the TF libraries below.

**tf.Transform:** Consistent in-graph transformations in training and serving.

**tf.ModelAnalysis:** Scalable, sliced, and full-pass metrics.