
View on TensorFlow.org 
Run in Google Colab 
View source on GitHub

In [1]:
```
#@title Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

Preprocess data with TensorFlow Transform

The Feature Engineering Component of TensorFlow Extended (TFX)

This example colab notebook provides a very simple example of how TensorFlow Transform (`tf.Transform`) can be used to preprocess data using exactly the same code for both training a model and serving inferences in production.

TensorFlow Transform is a library for preprocessing input data for TensorFlow, including creating features that require a full pass over the training dataset. For example, using TensorFlow Transform you could:

- Normalize an input value by using the mean and standard deviation
- Convert strings to integers by generating a vocabulary over all of the input values
- Convert floats to integers by assigning them to buckets, based on the observed data distribution

TensorFlow has built-in support for manipulations on a single example or a batch of examples. `tf.Transform` extends these capabilities to support full passes over the entire training dataset.

The output of `tf.Transform` is exported as a TensorFlow graph which you can use for both training and serving. Using the same graph for both training and serving can prevent skew, since the same transformations are applied in both stages.

Upgrade Pip

To avoid upgrading Pip in a system when running locally, check to make sure that we're running in Colab. Local systems can of course be upgraded separately.

In [2]:
```
try:
  import colab
  !pip install --upgrade pip
except:
  pass
```

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
warnings.r   esourceWarning
```

Install TensorFlow Transform

Note: In Google Colab, because of package updates, the first time you run this cell you may need to restart the runtime (Runtime > Restart runtime ...).

In [3]:
```
#!pip install -q -U tensorflow_transform==0.24.1
```

Did you restart the runtime?

If you are using Google Colab, the first time that you run the cell above, you must restart the runtime (Runtime > Restart runtime ...). This is because of the way that Colab loads packages.

Imports

In [2]:
```
import pprint
import tempfile

import tensorflow as tf
import tensorflow_transform as tft

import tensorflow_transform.beam as tft_beam
from tensorflow_transform.tf_metadata import dataset_metadata
from tensorflow_transform.tf_metadata import schema_utils
```

Data: Create some dummy data

We'll create some simple dummy data for our simple example:

- `raw_data` is the initial raw data that we're going to preprocess
- `raw_data_metadata` contains the schema that tells us the types of each of the columns in `raw_data`. In this case, it's very simple.

In [3]:
```
raw_data = [
      {'x': 1, 'y': 1, 's': 'hello'},
      {'x': 2, 'y': 2, 's': 'world'},
      {'x': 3, 'y': 3, 's': 'hello'}
  ]

raw_data_metadata = dataset_metadata.DatasetMetadata(
    schema_utils.schema_from_feature_spec({
        'y': tf.io.FixedLenFeature([], tf.float32),
        'x': tf.io.FixedLenFeature([], tf.float32),
        's': tf.io.FixedLenFeature([], tf.string),
    }))
```

Transform: Create a preprocessing function

The *preprocessing function* is the most important concept of tf.Transform. A preprocessing function is where the transformation of the dataset really happens. It accepts and returns a dictionary of tensors, where a tensor means a `Tensor` or `SparseTensor`. There are two main groups of API calls that typically form the heart of a preprocessing function:

1. **TensorFlow Ops:** Any function that accepts and returns tensors, which usually means TensorFlow ops. These add TensorFlow operations to the graph that transforms raw data into transformed data one feature vector at a time. These will run for every example, during both training and serving.
2. **Tensorflow Transform Analyzers/Mappers:** Any of the analyzers/mappers provided by tf.Transform. These also accept and return tensors, and typically contain a combination of Tensorflow ops and Beam computation, but unlike TensorFlow ops they only run in the Beam pipeline during analysis requiring a full pass over the entire training dataset. The Beam computation runs only once, during training, and typically make a full pass over the entire training dataset. They create tensor constants, which are added to your graph. For example, tft.min computes the minimum of a tensor over the training dataset while tft.scale_by_min_max first computes the min and max of a tensor over the training dataset and then scales the tensor to be within a user-specified range, [output_min, output_max]. tf.Transform provides a fixed set of such analyzers/mappers, but this will be extended in future versions.

Caution: When you apply your preprocessing function to serving inferences, the constants that were created by analyzers during training do not change. If your data has trend or seasonality components, plan accordingly.

Note: The `preprocessing_fn` is not directly callable. This means that calling `preprocessing_fn(raw_data)` will not work. Instead, it must be passed to the Transform Beam API as shown in the following cells.

In [4]:
```
def preprocessing_fn(inputs):
    """Preprocess input columns into transformed columns."""
    x = inputs['x']
    y = inputs['y']
    s = inputs['s']
    x_centered = x - tft.mean(x)
    y_normalized = tft.scale_to_0_1(y)
    s_integerized = tft.compute_and_apply_vocabulary(s)
    x_centered_times_y_normalized = (x_centered * y_normalized)
    return {
        'x_centered': x_centered,
        'y_normalized': y_normalized,
        's_integerized': s_integerized,
        'x_centered_times_y_normalized': x_centered_times_y_normalized,
    }
```

Putting it all together

Now we're ready to transform our data. We'll use Apache Beam with a direct runner, and supply three inputs:

1. `raw_data` - The raw input data that we created above
2. `raw_data_metadata` - The schema for the raw data
3. `preprocessing_fn` - The function that we created to do our transformation

Key Term: Apache Beam uses a special syntax to define and invoke transforms. For example, in this line:

> `result = pass_this | 'name this step' >> to_this_call`

The method `to_this_call` is being invoked and passed the object called `pass_this`, and this operation will be referred to as `name this step` in a stack trace. The result of the call to `to_this_call` is returned in `result`. You will often see stages of a pipeline chained together like this:

> `result = apache_beam.Pipeline() | 'first step' >> do_this_first() | 'second step' >> do_this_last()`

and since that started with a new pipeline, you can continue like this:

> `next_result = result | 'doing more stuff' >> another_function()`

In [5]:
```
def main():
  # Ignore the warnings
  with tft_beam.Context(temp_dir=tempfile.mkdtemp()):
    transformed_dataset, transform_fn = (  # pylint: disable=unused-variable
        (raw_data, raw_data_metadata) | tft_beam.AnalyzeAndTransformDataset(
            preprocessing_fn))

  transformed_data, transformed_metadata = transformed_dataset  # pylint: disable=unused-variable

  print('\nRaw data:\n{}\n'.format(pprint.pformat(raw_data)))
  print('Transformed data:\n{}'.format(pprint.pformat(transformed_data)))

if __name__ == '__main__':
  main()
```

```
WARNING:tensorflow:Tensorflow version (2.3.3) found. Note that Tensorflow Transform support for TF 2.0 is currently in beta, and features such as tf.function may not work as intended.
WARNING:apache_beam.runners.interactive.interactive_environment:Dependencies required for Interactive Beam PCollection visualization are not available, please use: `pip install apache-beam[interactive]` to install necessary dependencies to enable all data visualization features.

WARNING:tensorflow:Tensorflow version (2.3.3) found. Note that Tensorflow Transform support for TF 2.0 is currently in beta, and features such as tf.function may not work as intended.
WARNING:tensorflow:Tensorflow version (2.3.3) found. Note that Tensorflow Transform support for TF 2.0 is currently in beta, and features such as tf.function may not work as intended.
WARNING:tensorflow:You are passing instance dicts and DatasetMetadata to TFT which will not provide optimal performance. Consider following the TFT guide to upgrade to the TFXIO format (Apache Arrow RecordBatch).
WARNING:tensorflow:You are passing instance dicts and DatasetMetadata to TFT which will not provide optimal performance. Consider following the TFT guide to upgrade to the TFXIO format (Apache A
```

```
arow.RecordBatch)
WARNING:tensorflow:From C:\Miniconda3\envs\tensorflowgpu\lib\site-packages\tensorflow_transform\tf_utils.py:218: Tensor.experimental_ref (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use ref() instead.
WARNING:tensorflow:From C:\Miniconda3\envs\tensorflowgpu\lib\site-packages\tensorflow_transform\tf_utils.py:218: Tensor.experimental_ref (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use ref() instead.
WARNING:tensorflow:From C:\Miniconda3\envs\tensorflowgpu\lib\site-packages\tensorflow\python\saved_model\signature_def_utils_impl.py:201: build_tensor_info (from tensorflow.python.saved_model.utils_impl) is deprecated and will be removed in a future version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.compat.v1.saved_model.utils.build_tensor_info or tf.compat.v1.saved_model.build_tensor_info.
WARNING:tensorflow:From C:\Miniconda3\envs\tensorflowgpu\lib\site-packages\tensorflow\python\saved_model\signature_def_utils_impl.py:201: build_tensor_info (from tensorflow.python.saved_model.utils_impl) is deprecated and will be removed in a future version.
Instructions for updating:
This function will only be available through the v1 compatibility library as tf.compat.v1.saved_model.utils.build_tensor_info or tf.compat.v1.saved_model.build_tensor_info.
INFO:tensorflow:Assets added to graph.
INFO:tensorflow:Assets added to graph.
INFO:tensorflow:No assets to write.
INFO:tensorflow:No assets to write.
WARNING:tensorflow:Issue encountered when serializing tft_analyzer_use.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Counter' object has no attribute 'name'
WARNING:tensorflow:Issue encountered when serializing tft_analyzer_use.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Counter' object has no attribute 'name'
WARNING:tensorflow:Issue encountered when serializing tft_mapper_use.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Counter' object has no attribute 'name'
WARNING:tensorflow:Issue encountered when serializing tft_mapper_use.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Counter' object has no attribute 'name'
INFO:tensorflow:SavedModel written to: C:\Users\StaFaka\AppData\Local\Temp\tmpswg93an_\tftransform_tmp\133dccadf2e54b6ea19ab54aefcb3676\saved_model.pb
INFO:tensorflow:SavedModel written to: C:\Users\StaFaka\AppData\Local\Temp\tmpswg93an_\tftransform_tmp\133dccadf2e54b6ea19ab54aefcb3676\saved_model.pb
INFO:tensorflow:Assets added to graph.
INFO:tensorflow:Assets added to graph.
INFO:tensorflow:No assets to write.
INFO:tensorflow:No assets to write.
WARNING:tensorflow:Issue encountered when serializing tft_analyzer_use.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Counter' object has no attribute 'name'
WARNING:tensorflow:Issue encountered when serializing tft_analyzer_use.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Counter' object has no attribute 'name'
WARNING:tensorflow:Issue encountered when serializing tft_mapper_use.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Counter' object has no attribute 'name'
WARNING:tensorflow:Issue encountered when serializing tft_mapper_use.
Type is unsupported, or the types of the items don't match field type in CollectionDef. Note this is a warning and probably safe to ignore.
'Counter' object has no attribute 'name'
INFO:tensorflow:SavedModel written to: C:\Users\StaFaka\AppData\Local\Temp\tmpswg93an_\tftransform_tmp\ccd6c388725141b08e296bee7a1fd53d\saved_model.pb
INFO:tensorflow:SavedModel written to: C:\Users\StaFaka\AppData\Local\Temp\tmpswg93an_\tftransform_tmp\ccd6c388725141b08e296bee7a1fd53d\saved_model.pb
WARNING:tensorflow:Tensorflow version (2.3.3) found. Note that Tensorflow Transform support for TF 2.0 is currently in beta, and features such as tf.function may not work as intended.
WARNING:tensorflow:Tensorflow version (2.3.3) found. Note that Tensorflow Transform support for TF 2.0 is currently in beta, and features such as tf.function may not work as intended.
```

```
WARNING:tensorflow:You are passing instance dicts and DatasetMetadata to TFT which will not provi
de optimal performance. Consider following the TFT guide to upgrade to the TFXIO format (Apache A
rrow RecordBatch).
WARNING:tensorflow:You are passing instance dicts and DatasetMetadata to TFT which will not provi
de optimal performance. Consider following the TFT guide to upgrade to the TFXIO format (Apache A
rrow RecordBatch).
WARNING:apache_beam.options.pipeline_options:Discarding unparseable args: ['C:\\Miniconda3\\env
s\\tensorflowgpu\\lib\\site-packages\\ipykernel_launcher.py', '-f', 'C:\\Users\\StaFaka\\AppDat
a\\Roaming\\jupyter\\runtime\\kernel-3fa22702-e4de-4312-bca4-305cdeef9429.json']
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
INFO:tensorflow:Assets added to graph.
INFO:tensorflow:Assets added to graph.
INFO:tensorflow:Assets written to: C:\Users\StaFaka\AppData\Local\Temp\tmpswg93an_\tftransform_tm
p\8c55a3f108d0498fbab4042bafa93929\assets
INFO:tensorflow:Assets written to: C:\Users\StaFaka\AppData\Local\Temp\tmpswg93an_\tftransform_tm
p\8c55a3f108d0498fbab4042bafa93929\assets
INFO:tensorflow:SavedModel written to: C:\Users\StaFaka\AppData\Local\Temp\tmpswg93an_\tftransfor
m_tmp\8c55a3f108d0498fbab4042bafa93929\saved_model.pb
INFO:tensorflow:SavedModel written to: C:\Users\StaFaka\AppData\Local\Temp\tmpswg93an_\tftransfor
m_tmp\8c55a3f108d0498fbab4042bafa93929\saved_model.pb
WARNING:tensorflow:Expected binary or unicode string, got type_url: "type.googleapis.com/tensorfl
ow.AssetFileDef"
value: "\n\013\n\tConst_3:0\022-vocab_compute_and_apply_vocabulary_vocabulary"

WARNING:tensorflow:Expected binary or unicode string, got type_url: "type.googleapis.com/tensorfl
ow.AssetFileDef"
value: "\n\013\n\tConst_3:0\022-vocab_compute_and_apply_vocabulary_vocabulary"

INFO:tensorflow:Saver not created because there are no variables in the graph to restore
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
WARNING:tensorflow:Expected binary or unicode string, got type_url: "type.googleapis.com/tensorfl
ow.AssetFileDef"
value: "\n\013\n\tConst_3:0\022-vocab_compute_and_apply_vocabulary_vocabulary"

WARNING:tensorflow:Expected binary or unicode string, got type_url: "type.googleapis.com/tensorfl
ow.AssetFileDef"
value: "\n\013\n\tConst_3:0\022-vocab_compute_and_apply_vocabulary_vocabulary"

INFO:tensorflow:Saver not created because there are no variables in the graph to restore
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
Raw data:
[{'s': 'hello', 'x': 1, 'y': 1},
 {'s': 'world', 'x': 2, 'y': 2},
 {'s': 'hello', 'x': 3, 'y': 3}]

Transformed data:
[{'s_integerized': 0,
  'x_centered': -1.0,
  'x_centered_times_y_normalized': -0.0,
  'y_normalized': 0.0},
 {'s_integerized': 1,
  'x_centered': 0.0,
  'x_centered_times_y_normalized': 0.0,
  'y_normalized': 0.5},
 {'s_integerized': 0,
  'x_centered': 1.0,
  'x_centered_times_y_normalized': 1.0,
  'y_normalized': 1.0}]
```

Is this the right answer?

Previously, we used `tf.Transform` to do this:

```
x_centered = x - tft.mean(x)
y_normalized = tft.scale_to_0_1(y)
s_integerized = tft.compute_and_apply_vocabulary(s)
x_centered_times_y_normalized = (x_centered * y_normalized)
```

x_centered

With input of `[1, 2, 3]` the mean of x is 2, and we subtract it from x to center our x values at 0. So our result of `[-1.0, 0.0, 1.0]` is correct.

y_normalized

We wanted to scale our y values between 0 and 1. Our input was `[1, 2, 3]` so our result of `[0.0, 0.5, 1.0]` is correct.

s_integerized

We wanted to map our strings to indexes in a vocabulary, and there were only 2 words in our vocabulary ("hello" and "world"). So with input of `["hello", "world", "hello"]` our result of `[0, 1, 0]` is correct. Since "hello" occurs most frequently in this data, it will be the first entry in the vocabulary.

x_centered_times_y_normalized

We wanted to create a new feature by crossing `x_centered` and `y_normalized` using multiplication. Note that this