

# DNN\_TF-3

December 17, 2020

Create TensorFlow DNN model

This notebook illustrates:

Creating a model using the high-level Estimator API

```
[1]: !sudo chown -R jupyter:jupyter /home/jupyter/training-data-analyst
```

```
[2]: # Ensure the right version of Tensorflow is installed.
!pip freeze | grep tensorflow==2.1
```

```
[3]: # change these to try this notebook out
BUCKET = 'qwiklabs-gcp-02-2449df839737'
PROJECT = 'qwiklabs-gcp-02-2449df839737'
REGION = 'us-central1'
```

```
[4]: import os
os.environ['BUCKET'] = BUCKET
os.environ['PROJECT'] = PROJECT
os.environ['REGION'] = REGION
```

```
[5]: %%bash
if ! gsutil ls | grep -q gs://${BUCKET}/; then
    gsutil mb -l ${REGION} gs://${BUCKET}
fi
```

```
[6]: %%bash
ls *.csv
```

eval.csv

train.csv

Create TensorFlow model using TensorFlow's Estimator API

First, write an input\_fn to read the data.

```
[7]: import shutil
import numpy as np
import tensorflow as tf
print(tf.__version__)
```

### 2.3.1

```
[8]: # Determine CSV, label, and key columns
CSV_COLUMNS = 'weight_pounds,is_male,mother_age,plurality,gestation_weeks,key'.
    ↪split(',')
LABEL_COLUMN = 'weight_pounds'
KEY_COLUMN = 'key'

# Set default values for each CSV column
DEFAULTS = [[0.0], ['null'], [0.0], ['null'], [0.0], ['nokey']]
TRAIN_STEPS = 1000
```

```
[9]: # Create an input function reading a file using the Dataset API
# Then provide the results to the Estimator API
def read_dataset(filename, mode, batch_size = 512):
    def _input_fn():
        def decode_csv(value_column):
            columns = tf.compat.v1.decode_csv(value_column, record_defaults=DEFAULTS)
            features = dict(zip(CSV_COLUMNS, columns))
            label = features.pop(LABEL_COLUMN)
            return features, label

        # Create list of files that match pattern
        file_list = tf.compat.v1.gfile.Glob(filename)

        # Create dataset from file list
        dataset = (tf.compat.v1.data.TextLineDataset(file_list) # Read text file
                    .map(decode_csv)) # Transform each elem by applying ↪
    ↪decode_csv fn

        if mode == tf.estimator.ModeKeys.TRAIN:
            num_epochs = None # indefinitely
            dataset = dataset.shuffle(buffer_size=10*batch_size)
        else:
            num_epochs = 1 # end-of-input after this

        dataset = dataset.repeat(num_epochs).batch(batch_size)
        return dataset
    return _input_fn
```

Next, define the feature columns

```
[10]: # Define feature columns
def get_categorical(name, values):
    return tf.feature_column.indicator_column(
        tf.feature_column.categorical_column_with_vocabulary_list(name, values))
```

```
def get_cols():
    # Define column types
    return [
        get_categorical('is_male', ['True', 'False', 'Unknown']),
        tf.feature_column.numeric_column('mother_age'),
        get_categorical('plurality',
                        ['Single(1)', 'Twins(2)', 'Triplets(3)',
                         'Quadruplets(4)', 'Quintuplets(5)', 'Multiple(2+)']),
        tf.feature_column.numeric_column('gestation_weeks')
    ]
```

To predict with the TensorFlow model, we also need a serving input function. We will want all the inputs from our user.

```
[11]: # Create serving input function to be able to serve predictions later using
      ↪ provided inputs
def serving_input_fn():
    feature_placeholders = {
        'is_male': tf.compat.v1.placeholder(tf.string, [None]),
        'mother_age': tf.compat.v1.placeholder(tf.float32, [None]),
        'plurality': tf.compat.v1.placeholder(tf.string, [None]),
        'gestation_weeks': tf.compat.v1.placeholder(tf.float32, [None])
    }
    features = {
        key: tf.expand_dims(tensor, -1)
        for key, tensor in feature_placeholders.items()
    }
    return tf.estimator.export.ServingInputReceiver(features,
      ↪ feature_placeholders)
```

```
[12]: # Create estimator to train and evaluate
def train_and_evaluate(output_dir):
    EVAL_INTERVAL = 300
    run_config = tf.estimator.RunConfig(save_checkpoints_secs = EVAL_INTERVAL,
                                         keep_checkpoint_max = 3)
    estimator = tf.estimator.DNNRegressor(
        model_dir = output_dir,
        feature_columns = get_cols(),
        hidden_units = [64, 32],
        config = run_config)
    train_spec = tf.estimator.TrainSpec(
        input_fn = read_dataset('train.csv', mode = tf.estimator.
      ↪ ModeKeys.TRAIN),
        max_steps = TRAIN_STEPS)
    exporter = tf.estimator.LatestExporter('exporter', serving_input_fn)
    eval_spec = tf.estimator.EvalSpec(
```

```

        input_fn = read_dataset('eval.csv', mode = tf.estimator.
↪ModeKeys.EVAL),

        steps = None,
        start_delay_secs = 60, # start evaluating after N seconds
        throttle_secs = EVAL_INTERVAL, # evaluate every N
↪seconds

        exporters = exporter)
    tf.estimator.train_and_evaluate(estimator, train_spec, eval_spec)

```

Finally, train!

```

[13]: # Run the model
shutil.rmtree('babyweight_trained', ignore_errors = True) # start fresh each
↪time
tf.compat.v1.summary.FileWriterCache.clear()
train_and_evaluate('babyweight_trained')

```

```

INFO:tensorflow:Using config: {'_model_dir': 'babyweight_trained',
'_tf_random_seed': None, '_save_summary_steps': 100, '_save_checkpoints_steps':
None, '_save_checkpoints_secs': 300, '_session_config': allow_soft_placement:
true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 3, '_keep_checkpoint_every_n_hours': 10000,
'_log_step_count_steps': 100, '_train_distribute': None, '_device_fn': None,
'_protocol': None, '_eval_distribute': None, '_experimental_distribute': None,
'_experimental_max_worker_delay_secs': None, '_session_creation_timeout_secs':
7200, '_service': None, '_cluster_spec': ClusterSpec({}), '_task_type':
'worker', '_task_id': 0, '_global_id_in_cluster': 0, '_master': '',
'_evaluation_master': '', '_is_chief': True, '_num_ps_replicas': 0,
'_num_worker_replicas': 1}
INFO:tensorflow:Not using Distribute Coordinator.
INFO:tensorflow:Running training and evaluation locally (non-distributed).
INFO:tensorflow:Start train and evaluate loop. The evaluate will happen after
every checkpoint. Checkpoint frequency is determined based on RunConfig
arguments: save_checkpoints_steps None or save_checkpoints_secs 300.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-
packages/tensorflow/python/training/training_util.py:236:
Variable.initialized_value (from tensorflow.python.ops.variables) is deprecated
and will be removed in a future version.
Instructions for updating:
Use Variable.read_value. Variables in 2.X are initialized automatically both in
eager and graph (inside tf.defun) contexts.
INFO:tensorflow:Calling model_fn.

```

```

WARNING:tensorflow:From /opt/conda/lib/python3.7/site-
packages/tensorflow/python/keras/optimizer_v2/adagrad.py:83: calling
Constant.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated
and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the
constructor
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Calling checkpoint listeners before saving checkpoint 0...
INFO:tensorflow:Saving checkpoints for 0 into babyweight_trained/model.ckpt.
INFO:tensorflow:Calling checkpoint listeners after saving checkpoint 0...
INFO:tensorflow:loss = 12.427019, step = 0
INFO:tensorflow:global_step/sec: 31.1664
INFO:tensorflow:loss = 1.1931837, step = 100 (3.211 sec)
INFO:tensorflow:global_step/sec: 35.8302
INFO:tensorflow:loss = 1.3298357, step = 200 (2.794 sec)
INFO:tensorflow:global_step/sec: 33.5681
INFO:tensorflow:loss = 1.215979, step = 300 (2.984 sec)
INFO:tensorflow:global_step/sec: 39.407
INFO:tensorflow:loss = 1.2376792, step = 400 (2.540 sec)
INFO:tensorflow:global_step/sec: 39.7111
INFO:tensorflow:loss = 1.4252473, step = 500 (2.516 sec)
INFO:tensorflow:global_step/sec: 34.5566
INFO:tensorflow:loss = 1.3504784, step = 600 (2.889 sec)
INFO:tensorflow:global_step/sec: 43.4632
INFO:tensorflow:loss = 1.1825649, step = 700 (2.307 sec)
INFO:tensorflow:global_step/sec: 38.4056
INFO:tensorflow:loss = 1.1616136, step = 800 (2.599 sec)
INFO:tensorflow:global_step/sec: 36.6707
INFO:tensorflow:loss = 1.1814494, step = 900 (2.723 sec)
INFO:tensorflow:Calling checkpoint listeners before saving checkpoint 1000...
INFO:tensorflow:Saving checkpoints for 1000 into babyweight_trained/model.ckpt.
INFO:tensorflow:Calling checkpoint listeners after saving checkpoint 1000...
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2020-12-17T20:29:01Z
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from babyweight_trained/model.ckpt-1000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Inference Time : 1.04942s
INFO:tensorflow:Finished evaluation at 2020-12-17-20:29:02
INFO:tensorflow:Saving dict for global step 1000: average_loss = 1.2350379,
global_step = 1000, label/mean = 7.2368712, loss = 1.2361832, prediction/mean =

```

```

7.189839
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 1000:
babyweight_trained/model.ckpt-1000
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
WARNING:tensorflow:From /opt/conda/lib/python3.7/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:Signatures INCLUDED in export for Classify: None
INFO:tensorflow:Signatures INCLUDED in export for Regress: None
INFO:tensorflow:Signatures INCLUDED in export for Predict: ['predict']
INFO:tensorflow:Signatures INCLUDED in export for Train: None
INFO:tensorflow:Signatures INCLUDED in export for Eval: None
INFO:tensorflow:Signatures EXCLUDED from export because they cannot be served
via TensorFlow Serving APIs:
INFO:tensorflow:'serving_default' : Regression input must be a single string
Tensor; got {'is_male': <tf.Tensor 'Placeholder:0' shape=(None,) dtype=string>,
'mother_age': <tf.Tensor 'Placeholder_1:0' shape=(None,) dtype=float32>,
'plurality': <tf.Tensor 'Placeholder_2:0' shape=(None,) dtype=string>,
'gestation_weeks': <tf.Tensor 'Placeholder_3:0' shape=(None,) dtype=float32>}}
INFO:tensorflow:'regression' : Regression input must be a single string Tensor;
got {'is_male': <tf.Tensor 'Placeholder:0' shape=(None,) dtype=string>,
'mother_age': <tf.Tensor 'Placeholder_1:0' shape=(None,) dtype=float32>,
'plurality': <tf.Tensor 'Placeholder_2:0' shape=(None,) dtype=string>,
'gestation_weeks': <tf.Tensor 'Placeholder_3:0' shape=(None,) dtype=float32>}}
WARNING:tensorflow:Export includes no default signature!
INFO:tensorflow:Restoring parameters from babyweight_trained/model.ckpt-1000
INFO:tensorflow:Assets added to graph.
INFO:tensorflow:No assets to write.
INFO:tensorflow:SavedModel written to:
babyweight_trained/export/exporter/temp-1608236942/saved_model.pb
INFO:tensorflow:Loss for final step: 1.3176429.

```

The exporter directory contains the final model.

Copyright 2020 Google Inc. 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 <http://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