Create TensorFlow wide-and-deep model

This notebook illustrates:

1. Creating a model using the high-level Estimator API

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

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

In [3]:
```
# change these to try this notebook out
BUCKET = 'cloud-training-demos-ml'
PROJECT = 'cloud-training-demos'
REGION = 'us-central1'
```

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

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

```
Creating gs://cloud-training-demos-ml/...
ServiceException: 409 A Cloud Storage bucket named 'cloud-training-demos-ml' alr
eady exists. Try another name. Bucket names must be globally unique across all G
oogle Cloud projects, including those outside of your organization.
```

```
---------------------------------------------------------------------------
CalledProcessError                        Traceback (most recent call last)
<ipython-input-5-6b1d45d375e6> in <module>
----> 1 get_ipython().run_cell_magic('bash', '', 'if ! gsutil ls | grep -q gs://
${BUCKET}/; then\n  gsutil mb -l ${REGION} gs://${BUCKET}\nfi\n')

/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py in run_c
ell_magic(self, magic_name, line, cell)
   2401             with self.builtin_trap:
   2402                 args = (magic_arg_s, cell)
-> 2403                 result = fn(*args, **kwargs)
   2404             return result
   2405

/opt/conda/lib/python3.7/site-packages/IPython/core/magics/script.py in named_sc
ript_magic(line, cell)
    140             else:
    141                 line = script
--> 142             return self.shebang(line, cell)
    143
```

```
    144         # write a basic docstring:

/opt/conda/lib/python3.7/site-packages/decorator.py in fun(*args, **kw)
    230             if not kwsyntax:
    231                 args, kw = fix(args, kw, sig)
--> 232             return caller(func, *(extras + args), **kw)
    233     fun.__name__ = func.__name__
    234     fun.__doc__ = func.__doc__

/opt/conda/lib/python3.7/site-packages/IPython/core/magic.py in <lambda>(f, *a,
 **k)
    185     # but it's overkill for just that one bit of state.
    186     def magic_deco(arg):
--> 187         call = lambda f, *a, **k: f(*a, **k)
    188
    189         if callable(arg):

/opt/conda/lib/python3.7/site-packages/IPython/core/magics/script.py in shebang
(self, line, cell)
    243             sys.stderr.flush()
    244         if args.raise_error and p.returncode!=0:
--> 245             raise CalledProcessError(p.returncode, cell, output=out, std
err=err)
    246
    247     def _run_script(self, p, cell, to_close):

CalledProcessError: Command 'b'if ! gsutil ls | grep -q gs://${BUCKET}/; then\n
gsutil mb -l ${REGION} gs://${BUCKET}\nfi\n'' returned non-zero exit status 1.
```

In [ ]:

```
%%bash
ls *.csv
```

Create TensorFlow model using TensorFlow's Estimator API

First, write an input_fn to read the data.

In [6]:

```
import shutil
import numpy as np
import tensorflow as tf
print(tf.__version__)
```

```
2.3.3
```

In [7]:

```
# Determine CSV, label, and key columns
CSV_COLUMNS = 'weight_pounds,is_male,mother_age,plurality,gestation_weeks,key'.s
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
```

In [8]:

```
# 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

    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

In [9]:

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

  # Discretize
  age_buckets = tf.feature_column.bucketized_column(mother_age,
                      boundaries=np.arange(15,45,1).tolist())
  gestation_buckets = tf.feature_column.bucketized_column(gestation_weeks,
                      boundaries=np.arange(17,47,1).tolist())

  # Sparse columns are wide, have a linear relationship with the output
  wide = [is_male,
          plurality,
          age_buckets,
          gestation_buckets]

  # Feature cross all the wide columns and embed into a lower dimension
  crossed = tf.feature_column.crossed_column(wide, hash_bucket_size=20000)
  embed = tf.feature_column.embedding_column(crossed, 3)

  # Continuous columns are deep, have a complex relationship with the output
  deep = [mother_age,
```

```
          gestation_weeks,
          embed]
  return wide, deep
```

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

In [10]:
```
# Create serving input function to be able to serve predictions later using prov
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_placeholde
```

In [11]:
```
# Create estimator to train and evaluate
def train_and_evaluate(output_dir):
  wide, deep = get_wide_deep()
  EVAL_INTERVAL = 300
  run_config = tf.estimator.RunConfig(save_checkpoints_secs = EVAL_INTERVAL,
                                      keep_checkpoint_max = 3)
  estimator = tf.estimator.DNNLinearCombinedRegressor(
                       model_dir = output_dir,
                       linear_feature_columns = wide,
                       dnn_feature_columns = deep,
                       dnn_hidden_units = [64, 32],
                       config = run_config)
  train_spec = tf.estimator.TrainSpec(
                       input_fn = read_dataset('train.csv', mode = tf.estimator.
                       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.M
                       steps = None,
                       start_delay_secs = 60, # start evaluating after N seconds
                       throttle_secs = EVAL_INTERVAL,  # evaluate every N second
                       exporters = exporter)
  tf.estimator.train_and_evaluate(estimator, train_spec, eval_spec)
```

Finally, train!

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

```
INFO:tensorflow:Using config: {'_model_dir': 'babyweight_trained', '_tf_random_s
eed': 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, '_servic
e': None, '_cluster_spec': ClusterSpec({}), '_task_type': 'worker', '_task_id':
0, '_global_id_in_cluster': 0, '_master': '', '_evaluation_master': '', '_is_chi
ef': 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 ev
ery 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/pytho
n/training/training_util.py:236: Variable.initialized_value (from tensorflow.pyt
hon.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_estima
tor/python/estimator/canned/linear.py:1481: Layer.add_variable (from tensorflow.
python.keras.engine.base_layer_v1) is deprecated and will be removed in a future
version.
Instructions for updating:
Please use `layer.add_weight` method instead.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow/pytho
n/keras/optimizer_v2/adagrad.py:83: calling Constant.__init__ (from tensorflow.p
ython.ops.init_ops) with dtype is deprecated and will be removed in a future ver
sion.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the c
onstructor
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 = 81.848, step = 0
INFO:tensorflow:global_step/sec: 24.4236
INFO:tensorflow:loss = 2.6031132, step = 100 (4.097 sec)
INFO:tensorflow:global_step/sec: 41.0972
INFO:tensorflow:loss = 2.3955927, step = 200 (2.434 sec)
INFO:tensorflow:global_step/sec: 34.5411
INFO:tensorflow:loss = 2.040091, step = 300 (2.894 sec)
```

The exporter directory contains the final model.