Create Keras Wide-and-Deep model

This notebook illustrates:

1. Creating a model using Keras. This requires TensorFlow 2.1

In [1]:

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

In [2]:

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

In [3]:

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

In [4]:

```
%%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-4-6b1d45d375e6> 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 Keras model

First, write an input_fn to read the data.

In [7]:

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

```
2.3.3
```

In [8]:

```
# 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. Treat is_male and plurality as strings
DEFAULTS = [[0.0], ['null'], [0.0], ['null'], [0.0], ['nokey']]
```

In [9]:

```
def features_and_labels(row_data):
    for unwanted_col in ['key']:
        row_data.pop(unwanted_col)
    label = row_data.pop(LABEL_COLUMN)
    return row_data, label  # features, label

# load the training data
def load_dataset(pattern, batch_size=1, mode=tf.estimator.ModeKeys.EVAL):
  dataset = (tf.data.experimental.make_csv_dataset(pattern, batch_size, CSV_COLU
             .map(features_and_labels) # features, label
             )
  if mode == tf.estimator.ModeKeys.TRAIN:
        dataset = dataset.shuffle(1000).repeat()
  dataset = dataset.prefetch(1) # take advantage of multi-threading; 1=AUTOTUNE
  return dataset
```

Next, define the feature columns. mother_age and gestation_weeks should be numeric. The others (is_male, plurality) should be categorical.

In [11]:

```
## Build a Keras wide-and-deep model using its Functional API
def rmse(y_true, y_pred):
    return tf.sqrt(tf.reduce_mean(tf.square(y_pred - y_true)))

# Helper function to handle categorical columns
def categorical_fc(name, values):
    orig = tf.feature_column.categorical_column_with_vocabulary_list(name, value
    wrapped = tf.feature_column.indicator_column(orig)
    return orig, wrapped

def build_wd_model(dnn_hidden_units = [64, 32], nembeds = 3):
    # input layer
    deep_inputs = {
        colname : tf.keras.layers.Input(name=colname, shape=(), dtype='float32')
           for colname in ['mother_age', 'gestation_weeks']
    }
    wide_inputs = {
        colname : tf.keras.layers.Input(name=colname, shape=(), dtype='string')
           for colname in ['is_male', 'plurality']
    }
    inputs = {**wide_inputs, **deep_inputs}

    # feature columns from inputs
    deep_fc = {
        colname : tf.feature_column.numeric_column(colname)
           for colname in ['mother_age', 'gestation_weeks']
    }
    wide_fc = {}
    is_male, wide_fc['is_male'] = categorical_fc('is_male', ['True', 'False', 'U
    plurality, wide_fc['plurality'] = categorical_fc('plurality',
                      ['Single(1)', 'Twins(2)', 'Triplets(3)',
                       'Quadruplets(4)', 'Quintuplets(5)','Multiple(2+)'])

    # bucketize the float fields. This makes them wide
    age_buckets = tf.feature_column.bucketized_column(deep_fc['mother_age'],
                                                       boundaries=np.arange(15,45,
    wide_fc['age_buckets'] = tf.feature_column.indicator_column(age_buckets)
    gestation_buckets = tf.feature_column.bucketized_column(deep_fc['gestation_w
                                                       boundaries=np.arange(17,47,
    wide_fc['gestation_buckets'] = tf.feature_column.indicator_column(gestation_

    # cross all the wide columns. We have to do the crossing before we one-hot e
    crossed = tf.feature_column.crossed_column(
        [is_male, plurality, age_buckets, gestation_buckets], hash_bucket_size=2
    deep_fc['crossed_embeds'] = tf.feature_column.embedding_column(crossed, nemb

    # the constructor for DenseFeatures takes a list of numeric columns
    # The Functional API in Keras requires that you specify: LayerConstructor()(
    wide_inputs = tf.keras.layers.DenseFeatures(wide_fc.values(), name='wide_inp
    deep_inputs = tf.keras.layers.DenseFeatures(deep_fc.values(), name='deep_inp

    # hidden layers for the deep side
    layers = [int(x) for x in dnn_hidden_units]
    deep = deep_inputs
    for layerno, numnodes in enumerate(layers):
        deep = tf.keras.layers.Dense(numnodes, activation='relu', name='dnn_{}'.
    deep_out = deep

    # linear model for the wide side
    wide_out = tf.keras.layers.Dense(10, activation='relu', name='linear')(wide_

    # concatenate the two sides
    both = tf.keras.layers.concatenate([deep_out, wide_out], name='both')

    # final output is a linear activation because this is regression
    output = tf.keras.layers.Dense(1, activation='linear', name='weight')(both)
    model = tf.keras.models.Model(inputs, output)
    model.compile(optimizer='adam', loss='mse', metrics=[rmse, 'mse'])
    return model

print("Here is our Wide-and-Deep architecture so far:\n")
model = build_wd_model()
print(model.summary())
```

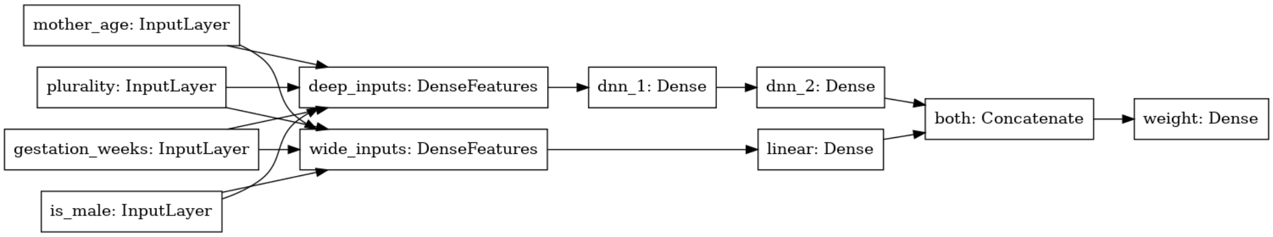
```
Here is our Wide-and-Deep architecture so far:

Model: "functional_3"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
gestation_weeks (InputLayer)    [(None,)]            0
________________________________________________________________________________
__________________
is_male (InputLayer)            [(None,)]            0
________________________________________________________________________________
__________________
mother_age (InputLayer)         [(None,)]            0
________________________________________________________________________________
__________________
plurality (InputLayer)          [(None,)]            0
________________________________________________________________________________
__________________
deep_inputs (DenseFeatures)     (None, 5)            60000       gestation_weeks
[0][0]
                                                                 is_male[0][0]
                                                                 mother_age[0]
[0]
                                                                 plurality[0][0]
________________________________________________________________________________
__________________
dnn_1 (Dense)                   (None, 64)           384         deep_inputs[0]
[0]
________________________________________________________________________________
__________________
wide_inputs (DenseFeatures)     (None, 71)           0           gestation_weeks
[0][0]
                                                                 is_male[0][0]
                                                                 mother_age[0]
[0]
                                                                 plurality[0][0]
_________________________________________________________________________________
__________________
dnn_2 (Dense)                   (None, 32)           2080        dnn_1[0][0]
_________________________________________________________________________________
__________________
linear (Dense)                  (None, 10)           720         wide_inputs[0]
[0]
_________________________________________________________________________________
__________________
both (Concatenate)              (None, 42)           0           dnn_2[0][0]
                                                                 linear[0][0]
_________________________________________________________________________________
__________________
weight (Dense)                  (None, 1)            43          both[0][0]
=================================================================================
==================
Total params: 63,227
Trainable params: 63,227
Non-trainable params: 0
_________________________________________________________________________________
__________________
None
```

We can visualize the DNN using the Keras plot_model utility.

In [12]:

```python
tf.keras.utils.plot_model(model, 'wd_model.png', show_shapes=False, rankdir='LR'
```

Out[12]:



Train and evaluate

In [13]:

```python
TRAIN_BATCH_SIZE = 32
NUM_TRAIN_EXAMPLES = 10000 * 5 # training dataset repeats, so it will wrap aroun
NUM_EVALS = 5  # how many times to evaluate
NUM_EVAL_EXAMPLES = 10000 # enough to get a reasonable sample, but not so much t

trainds = load_dataset('train*', TRAIN_BATCH_SIZE, tf.estimator.ModeKeys.TRAIN)
evalds = load_dataset('eval*', 1000, tf.estimator.ModeKeys.EVAL).take(NUM_EVAL_E

steps_per_epoch = NUM_TRAIN_EXAMPLES // (TRAIN_BATCH_SIZE * NUM_EVALS)

history = model.fit(trainds,
                    validation_data=evalds,
                    epochs=NUM_EVALS,
                    steps_per_epoch=steps_per_epoch)
```

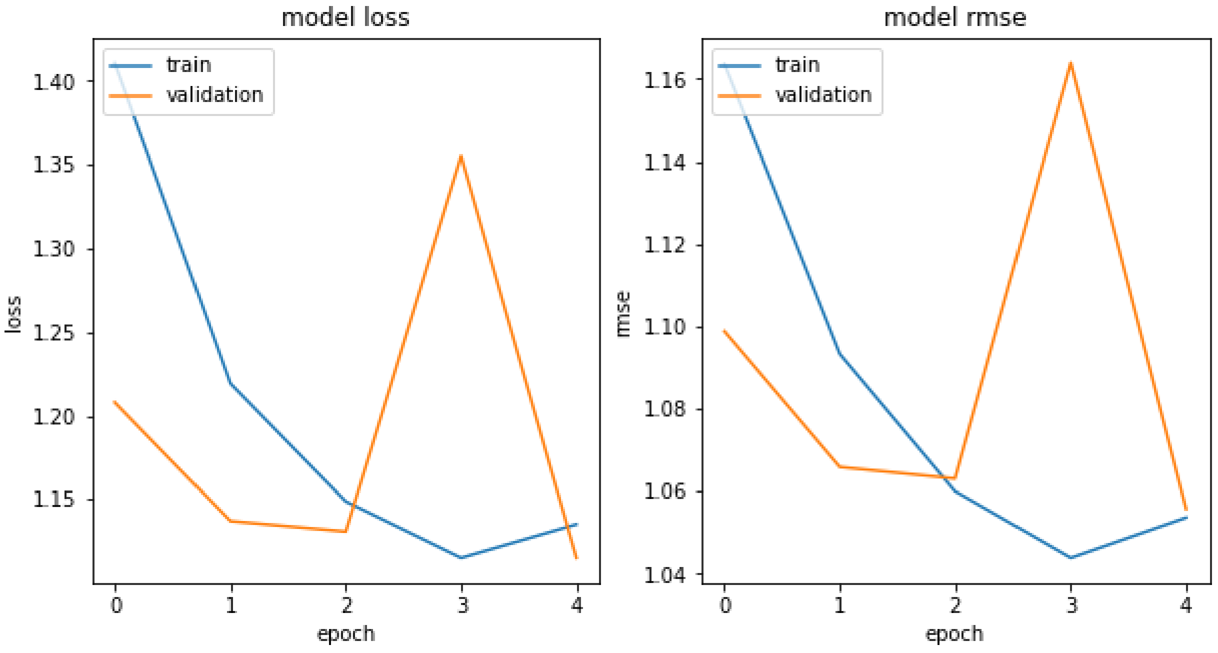
```
Epoch 1/5
312/312 [==============================] - 4s 11ms/step - loss: 1.4109 - rmse:
1.1636 - mse: 1.4109 - val_loss: 1.2079 - val_rmse: 1.0988 - val_mse: 1.2079
```

```
Epoch 2/5
312/312 [==============================] - 2s 8ms/step - loss: 1.2191 - rmse: 1.
0933 - mse: 1.2191 - val_loss: 1.1369 - val_rmse: 1.0659 - val_mse: 1.1369
Epoch 3/5
312/312 [==============================] - 3s 8ms/step - loss: 1.1486 - rmse: 1.
0600 - mse: 1.1486 - val_loss: 1.1307 - val_rmse: 1.0631 - val_mse: 1.1307
Epoch 4/5
312/312 [==============================] - 2s 7ms/step - loss: 1.1150 - rmse: 1.
0439 - mse: 1.1150 - val_loss: 1.3552 - val_rmse: 1.1639 - val_mse: 1.3552
Epoch 5/5
312/312 [==============================] - 2s 6ms/step - loss: 1.1350 - rmse: 1.
0536 - mse: 1.1350 - val_loss: 1.1151 - val_rmse: 1.0557 - val_mse: 1.1151
```

Visualize loss curve

In [14]:

```python
# plot
import matplotlib.pyplot as plt
nrows = 1
ncols = 2
fig = plt.figure(figsize=(10, 5))

for idx, key in enumerate(['loss', 'rmse']):
    ax = fig.add_subplot(nrows, ncols, idx+1)
    plt.plot(history.history[key])
    plt.plot(history.history['val_{}'.format(key)])
    plt.title('model {}'.format(key))
    plt.ylabel(key)
    plt.xlabel('epoch')
    plt.legend(['train', 'validation'], loc='upper left');
```



Save the model

In [15]:

```python
import shutil, os, datetime
OUTPUT_DIR = 'babyweight_trained'
shutil.rmtree(OUTPUT_DIR, ignore_errors=True)
EXPORT_PATH = os.path.join(OUTPUT_DIR, datetime.datetime.now().strftime('%Y%m%d%
tf.saved_model.save(model, EXPORT_PATH) # with default serving function
print("Exported trained model to {}".format(EXPORT_PATH))
```

```
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow/pytho
n/training/tracking/tracking.py:111: Model.state_updates (from tensorflow.pytho
n.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied autom
atically.
WARNING:tensorflow:From /opt/conda/lib/python3.7/site-packages/tensorflow/pytho
n/training/tracking/tracking.py:111: Layer.updates (from tensorflow.python.kera
s.engine.base_layer) is deprecated and will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied autom
atically.
INFO:tensorflow:Assets written to: babyweight_trained/20210818183030/assets
Exported trained model to babyweight_trained/20210818183030
```

In [16]:

```
!ls $EXPORT_PATH
```

```
assets  saved_model.pb  variables
```

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