2c. Refactoring to add batching and feature-creation

In this notebook, we continue reading the same small dataset, but refactor our ML pipeline in two small, but significant, ways:

1. Refactor the input to read data in batches.
2. Refactor the feature creation so that it is not one-to-one with inputs. The Pandas function in the previous notebook also batched, only after it had read the whole data into memory -- on a large dataset, this won't be an option.

In [1]:

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

```
1.15.0
```

1. Refactor the input

Read data created in Lab1a, but this time make it more general and performant. Instead of using Pandas, we will use TensorFlow's Dataset API.

In [2]:

```
CSV_COLUMNS = ['fare_amount', 'pickuplon','pickuplat','dropofflon','dropofflat','passengers', 'key']
LABEL_COLUMN = 'fare_amount'
DEFAULTS = [[0.0], [-74.0], [40.0], [-74.0], [40.7], [1.0], ['nokey']]

def read_dataset(filename, mode, batch_size = 512):
  def _input_fn():
    def decode_csv(value_column):
      columns = tf.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.gfile.Glob(filename)

    # Create dataset from file list
    dataset = tf.data.TextLineDataset(file_list).map(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.make_one_shot_iterator().get_next()
  return _input_fn


def get_train():
  return read_dataset('./taxi-train.csv', mode = tf.estimator.ModeKeys.TRAIN)

def get_valid():
  return read_dataset('./taxi-valid.csv', mode = tf.estimator.ModeKeys.EVAL)

def get_test():
  return read_dataset('./taxi-test.csv', mode = tf.estimator.ModeKeys.EVAL)
```

2. Refactor the way features are created.

For now, pass these through (same as previous lab). However, refactoring this way will enable us to break the one-to-one relationship between inputs and features.

In [3]:

```
INPUT_COLUMNS = [
    tf.feature_column.numeric_column('pickuplon'),
    tf.feature_column.numeric_column('pickuplat'),
    tf.feature_column.numeric_column('dropofflat'),
    tf.feature_column.numeric_column('dropofflon'),
    tf.feature_column.numeric_column('passengers'),
]

def add_more_features(feats):
  # Nothing to add (yet!)
  return feats

feature_cols = add_more_features(INPUT_COLUMNS)
```

Create and train the model

Note that we train for num_steps * batch_size examples.

In [4]:

```
tf.logging.set_verbosity(tf.logging.INFO)
OUTDIR = 'taxi_trained'
shutil.rmtree(OUTDIR, ignore_errors = True) # start fresh each time
model = tf.estimator.LinearRegressor(
      feature_columns = feature_cols, model_dir = OUTDIR)
model.train(input_fn = get_train(), steps = 100);  # TODO: change the name of input_fn as needed
```

```
INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_save_summary_steps': 100, '_protocol': None, '_save_checkpoints_sec
s': 600, '_task_id': 0, '_experimental_max_worker_delay_secs': None, '_cluster_spec': <tensorflow.py
thon.training.server_lib.ClusterSpec object at 0x7fa5580df320>, '_session_config': allow_soft_placem
ent: true
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_tf_random_seed': None, '_service': None, '_experimental_distribute': None, '_keep_checkpoint_ma
x': 5, '_global_id_in_cluster': 0, '_evaluation_master': '', '_save_checkpoints_steps': None, '_keep
_checkpoint_every_n_hours': 10000, '_master': '', '_device_fn': None, '_session_creation_timeout_sec
s': 7200, '_model_dir': 'taxi_trained', '_task_type': 'worker', '_train_distribute': None, '_num_ps_
replicas': 0, '_log_step_count_steps': 100, '_is_chief': True, '_eval_distribute': None, '_num_worke
r_replicas': 1}
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflow_core/python/training/train
ing_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 (ins
ide tf.defun) contexts.
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflow_core/python/autograph/conv
erters/directives.py:119: The name tf.decode_csv is deprecated. Please use tf.io.decode_csv instead.

WARNING:tensorflow:From <ipython-input-2-84f279a11ca0>:25: DatasetV1.make_one_shot_iterator (from te
nsorflow.python.data.ops.dataset_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use `for ... in dataset:` to iterate over a dataset. If using `tf.estimator`, return the `Dataset` o
bject directly from your input function. As a last resort, you can use `tf.compat.v1.data.make_one_s
hot_iterator(dataset)`.
INFO:tensorflow:Calling model_fn.
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflow_core/python/feature_colum
n/feature_column_v2.py:305: Layer.add_variable (from tensorflow.python.keras.engine.base_layer) is d
eprecated and will be removed in a future version.
Instructions for updating:
Please use `layer.add_weight` method instead.
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflow_core/python/ops/resource_v
ariable_ops.py:1630: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_vari
able_ops) with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflow_estimator/python/estimato
```

```
r/canned/linear.py:308: to_float (from tensorflow.python.ops.math_ops) is deprecated and will be rem
oved in a future version.
Instructions for updating:
Use `tf.cast` instead.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflow_core/python/ops/array_ops.
py:1475: where (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 0 into taxi_trained/model.ckpt.
INFO:tensorflow:loss = 142184.22, step = 1
INFO:tensorflow:Saving checkpoints for 100 into taxi_trained/model.ckpt.
INFO:tensorflow:Loss for final step: 57468.258.
```

Evaluate model

As before, evaluate on the validation data. We'll do the third refactoring (to move the evaluation into the training loop) in the next lab.

In [5]:

```
def print_rmse(model, name, input_fn):
  metrics = model.evaluate(input_fn = input_fn, steps = 1)
  print('RMSE on {} dataset = {}'.format(name, np.sqrt(metrics['average_loss'])))
print_rmse(model, 'validation', get_valid())
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2020-01-17T20:20:08Z
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from taxi_trained/model.ckpt-100
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Evaluation [1/1]
INFO:tensorflow:Finished evaluation at 2020-01-17-20:20:09
INFO:tensorflow:Saving dict for global step 100: average_loss = 81.33217, global_step = 100, label/m
ean = 10.920898, loss = 41642.07, prediction/mean = 11.729969
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 100: taxi_trained/model.ckpt-100
RMSE on validation dataset = 9.018434524536133
```