Mini-batching

In this section, you'll go over what mini-batching is and how to apply it in TensorFlow.

Mini-batching is a technique for training on subsets of the dataset instead of all the data at one time. This provides the ability to train a model, even if a computer lacks the memory to store the entire dataset.

Mini-batching is computationally inefficient, since you can't calculate the loss simultaneously across all samples. However, this is a small price to pay in order to be able to run the model at all.

It's also quite useful combined with SGD. The idea is to randomly shuffle the data at the start of each epoch, then create the mini-batches. For each mini-batch, you train the network weights with gradient descent. Since these batches are random, you're performing SGD with each batch.

Let's look at the MNIST dataset with weights and a bias to see if your machine can handle it.

```
from tensorflow.examples.tutorials.mnist import input_data
import tensorflow as tf

n_input = 784  # MNIST data input (img shape: 28*28)
n_classes = 10  # MNIST total classes (0-9 digits)

# Import MNIST data
mnist = input_data.read_data_sets('/datasets/ud730/mnist', one_hot=True)

# The features are already scaled and the data is shuffled
train_features = mnist.train.images
test_features = mnist.test.images

train_labels = mnist.train.labels.astype(np.float32)
test_labels = mnist.test.labels.astype(np.float32)

# Weights & bias
weights = tf.Variable(tf.random_normal([n_input, n_classes]))
bias = tf.Variable(tf.random_normal([n_classes]))
```

Question 1

Calculate the memory size of `train_features`, `train_labels`, `weights`, and `bias` in bytes. Ignore memory for overhead, just calculate the memory required for the stored data.

You may have to look up how much memory a float32 requires, using **this link**.

train_features Shape: (55000, 784) Type: float32

train_labels Shape: (55000, 10) Type: float32

weights Shape: (784, 10) Type: float32

bias Shape: (10,) Type: float32

How many bytes of memory does `train_features` need?

172480000 RESET

How many bytes of memory does `train_labels` need?

2200000 RESET

How many bytes of memory does `weights` need?

31360 RESET

How many bytes of memory does `bias` need?

40 RESET

The total memory space required for the inputs, weights and bias is around 174 megabytes, which isn't that much memory. You could train this whole dataset on most CPUs and GPUs.

But larger datasets that you'll use in the future measured in gigabytes or more. It's possible to purchase more memory, but it's expensive. A Titan X GPU with 12 GB of memory costs over $1,000.

Instead, in order to run large models on your machine, you'll learn how to use mini-batching.

Let's look at how you implement mini-batching in TensorFlow.

TensorFlow Mini-batching

In order to use mini-batching, you must first divide your data into batches.

Unfortunately, it's sometimes impossible to divide the data into batches of exactly equal size. For example, imagine you'd like to create batches of 128 samples each from a dataset of 1000 samples. Since 128 does not evenly divide into 1000, you'd wind up with 7 batches of 128 samples, and 1 batch of 104 samples. (7*128 + 1*104 = 1000)

In that case, the size of the batches would vary, so you need to take advantage of TensorFlow's **`tf.placeholder()`** function to receive the varying batch sizes.

Continuing the example, if each sample had `n_input = 784`features and `n_classes = 10` possible labels, the dimensions for `features` would be `[None, n_input]` and `labels` would be `[None, n_classes]`.

```
# Features and Labels
features = tf.placeholder(tf.float32, [None, n_input])
```

```
labels = tf.placeholder(tf.float32, [None, n_classes])
```

What does None do here?

The None dimension is a placeholder for the batch size. At runtime, TensorFlow will accept any batch size greater than 0.

Going back to our earlier example, this setup allows you to feed `features` and `labels` into the model as either the batches of 128 samples or the single batch of 104 samples.

Question 2

Use the parameters below, how many batches are there, and what is the last batch size?

features is (50000, 400)

labels is (50000, 10)

batch_size is 128

How many batches are there?

391

RESET

What is the last batch size?

80

RESET

Now that you know the basics, let's learn how to implement mini-batching.

Question 3

Implement the `batches` function to batch `features` and `labels`. The function should return each batch with a maximum size of `batch_size`. To help you with the quiz, look at the following example output of a working `batches` function.

```
# 4 Samples of features
example_features = [
    ['F11','F12','F13','F14'],
    ['F21','F22','F23','F24'],
    ['F31','F32','F33','F34'],
    ['F41','F42','F43','F44']]
# 4 Samples of labels
example_labels = [
    ['L11','L12'],
    ['L21','L22'],
    ['L31','L32'],
    ['L41','L42']]

example_batches = batches(3, example_features, example_labels)
```

The `example_batches` variable would be the following:

```
[
    # 2 batches:
    #   First is a batch of size 3.
    #   Second is a batch of size 1
    [
        # First Batch is size 3
        [
            # 3 samples of features.
            # There are 4 features per sample.
            ['F11', 'F12', 'F13', 'F14'],
            ['F21', 'F22', 'F23', 'F24'],
            ['F31', 'F32', 'F33', 'F34']
        ], [
            # 3 samples of labels.
            # There are 2 labels per sample.
```

```
            ['L11', 'L12'],
            ['L21', 'L22'],
            ['L31', 'L32']
        ]
    ], [
        # Second Batch is size 1.
        # Since batch size is 3, there is only one sample left from the 4 s
        [
            # 1 sample of features.
            ['F41', 'F42', 'F43', 'F44']
        ], [
            # 1 sample of labels.
            ['L41', 'L42']
        ]
    ]
]
```

Implement the `batches` function in the "quiz.py" file below.