COSC4337_NEURAL_NETWORK_API

[1]:
```
# Neural Network with Eager API.

#A 2-Hidden Layers Fully Connected Neural Network (a.k.a Multilayer Perceptron)
#implementation with TensorFlow's Eager API. This example is using the MNIST␣
 ↪database
#of handwritten digits (http://yann.lecun.com/exdb/mnist/).


from __future__ import print_function

import tensorflow as tf

# Set Eager API
tf.enable_eager_execution()
tfe = tf.contrib.eager

# Import MNIST data
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("/tmp/data/", one_hot=False)

# Parameters
learning_rate = 0.001
num_steps = 1000
batch_size = 128
display_step = 100

# Network Parameters
n_hidden_1 = 256 # 1st layer number of neurons
n_hidden_2 = 256 # 2nd layer number of neurons
num_input = 784 # MNIST data input (img shape: 28*28)
num_classes = 10 # MNIST total classes (0-9 digits)

# Using TF Dataset to split data into batches
dataset = tf.data.Dataset.from_tensor_slices(
    (mnist.train.images, mnist.train.labels))
dataset = dataset.repeat().batch(batch_size).prefetch(batch_size)
dataset_iter = tfe.Iterator(dataset)
```

```
# Define the neural network. To use eager API and tf.layers API together,
# we must instantiate a tfe.Network class as follow:
class NeuralNet(tfe.Network):
    def __init__(self):
        # Define each layer
        super(NeuralNet, self).__init__()
        # Hidden fully connected layer with 256 neurons
        self.layer1 = self.track_layer(
            tf.layers.Dense(n_hidden_1, activation=tf.nn.relu))
        # Hidden fully connected layer with 256 neurons
        self.layer2 = self.track_layer(
            tf.layers.Dense(n_hidden_2, activation=tf.nn.relu))
        # Output fully connected layer with a neuron for each class
        self.out_layer = self.track_layer(tf.layers.Dense(num_classes))

    def call(self, x):
        x = self.layer1(x)
        x = self.layer2(x)
        return self.out_layer(x)


neural_net = NeuralNet()


# Cross-Entropy loss function
def loss_fn(inference_fn, inputs, labels):
    # Using sparse_softmax cross entropy
    return tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(
        logits=inference_fn(inputs), labels=labels))


# Calculate accuracy
def accuracy_fn(inference_fn, inputs, labels):
    prediction = tf.nn.softmax(inference_fn(inputs))
    correct_pred = tf.equal(tf.argmax(prediction, 1), labels)
    return tf.reduce_mean(tf.cast(correct_pred, tf.float32))


# SGD Optimizer
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
# Compute gradients
grad = tfe.implicit_gradients(loss_fn)

# Training
average_loss = 0.
```

```
average_acc = 0.
for step in range(num_steps):

    # Iterate through the dataset
    d = dataset_iter.next()

    # Images
    x_batch = d[0]
    # Labels
    y_batch = tf.cast(d[1], dtype=tf.int64)

    # Compute the batch loss
    batch_loss = loss_fn(neural_net, x_batch, y_batch)
    average_loss += batch_loss
    # Compute the batch accuracy
    batch_accuracy = accuracy_fn(neural_net, x_batch, y_batch)
    average_acc += batch_accuracy

    if step == 0:
        # Display the initial cost, before optimizing
        print("Initial loss= {:.9f}".format(average_loss))

    # Update the variables following gradients info
    optimizer.apply_gradients(grad(neural_net, x_batch, y_batch))

    # Display info
    if (step + 1) % display_step == 0 or step == 0:
        if step > 0:
            average_loss /= display_step
            average_acc /= display_step
        print("Step:", '%04d' % (step + 1), " loss=",
              "{:.9f}".format(average_loss), " accuracy=",
              "{:.4f}".format(average_acc))
        average_loss = 0.
        average_acc = 0.

# Evaluate model on the test image set
testX = mnist.test.images
testY = mnist.test.labels

test_acc = accuracy_fn(neural_net, testX, testY)
print("Testset Accuracy: {:.4f}".format(test_acc))
```

```
WARNING:tensorflow:
The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-
sunset.md
```

```
  * https://github.com/tensorflow/addons
  * https://github.com/tensorflow/io (for I/O related ops)
If you depend on functionality not listed there, please file an issue.

WARNING:tensorflow:From <ipython-input-1-8dabc7e25a1f>:18: read_data_sets (from
tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be
removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from
tensorflow/models.
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:260:
maybe_download (from tensorflow.contrib.learn.python.learn.datasets.base) is
deprecated and will be removed in a future version.
Instructions for updating:
Please write your own downloading logic.
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:262:
extract_images (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting /tmp/data/train-images-idx3-ubyte.gz
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:267:
extract_labels (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:290:
DataSet.__init__ (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from
tensorflow/models.
WARNING:tensorflow:From <ipython-input-1-8dabc7e25a1f>:44: Network.__init__
(from tensorflow.contrib.eager.python.network) is deprecated and will be removed
in a future version.
Instructions for updating:
Please inherit from `tf.keras.Model`, and see its documentation for details.
`tf.keras.Model` should be a drop-in replacement for `tfe.Network` in most
cases, but note that `track_layer` is no longer necessary or supported. Instead,
`Layer` instances are tracked on attribute assignment (see the section of
`tf.keras.Model`'s documentation on subclassing). Since the output of
```

```
`track_layer` is often assigned to an attribute anyway, most code can be ported
by simply removing the `track_layer` calls.

`tf.keras.Model` works with all TensorFlow `Layer` instances, including those
from `tf.layers`, but switching to the `tf.keras.layers` versions along with the
migration to `tf.keras.Model` is recommended, since it will preserve variable
names. Feel free to import it with an alias to avoid excess typing :).
WARNING:tensorflow:** tfe.Network is deprecated and will be removed in a future
version.

Please inherit from `tf.keras.Model`, and see its documentation for details.
`tf.keras.Model` should be a drop-in replacement for `tfe.Network` in most
cases, but note that `track_layer` is no longer necessary or supported. Instead,
`Layer` instances are tracked on attribute assignment (see the section of
`tf.keras.Model`'s documentation on subclassing). Since the output of
`track_layer` is often assigned to an attribute anyway, most code can be ported
by simply removing the `track_layer` calls.

`tf.keras.Model` works with all TensorFlow `Layer` instances, including those
from `tf.layers`, but switching to the `tf.keras.layers` versions along with the
migration to `tf.keras.Model` is recommended, since it will preserve variable
names. Feel free to import it with an alias to avoid excess typing :).
Initial loss= 2.318457127
Step: 0001  loss= 2.318457127  accuracy= 0.1250
Step: 0100  loss= 0.576966524  accuracy= 0.8330
Step: 0200  loss= 0.253465354  accuracy= 0.9264
Step: 0300  loss= 0.219562486  accuracy= 0.9338
Step: 0400  loss= 0.182450980  accuracy= 0.9461
Step: 0500  loss= 0.141979888  accuracy= 0.9584
Step: 0600  loss= 0.120840818  accuracy= 0.9633
Step: 0700  loss= 0.117536522  accuracy= 0.9654
Step: 0800  loss= 0.111080579  accuracy= 0.9652
Step: 0900  loss= 0.085732564  accuracy= 0.9733
Step: 1000  loss= 0.083394840  accuracy= 0.9738
Testset Accuracy: 0.9693
```

[ ]: