HW2: Neurons/Learning Rate

Alex Shah

10/20/17

0.1 Code

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import time

startTime = time.process_time()

from tensorflow.examples.tutorials.mnist import
    input_data
mnist = input_data.read_data_sets(”.”, one_hot=True)

trainimgs = mnist.train.images
trainlabels = mnist.train.labels
testimgs = mnist.test.images
testlabels = mnist.test.labels

ntrain = trainimgs.shape[0]
ntest = testimgs.shape[0]
dim = trainimgs.shape[1]
nclasses = trainlabels.shape[1]

## Change for HW2
n_hidden = 4096
learning_rate = 1.0
training_iters = 100000
batch_size = 100
##

n_input = 28 # MNIST data input (img shape: 28*28)
n_steps = 28 # timesteps
n_classes = 10 # MNIST total classes (0−9 digits)
display_step = 10

x = tf.placeholder(dtype=”float”, shape=[None, n_steps,
    n_input], name=”x”) # Current data input shape: (
    batch_size, n_steps, n_input) [100x28x28]
y = tf.placeholder(dtype=”float”, shape=[None, n_classes
    ], name=”y”)


#different way of writing out a dictionary, or variable
    as a dictionary
weights = {
’out’: tf.Variable(tf.random_normal([n_hidden, n_classes
    ]))
}
biases = {
```

```
'out': tf.Variable(tf.random_normal([n_classes]))
}

lstm_cell = tf.contrib.rnn.BasicLSTMCell(n_hidden,
    forget_bias=1.0)

outputs, states = tf.nn.dynamic_rnn(lstm_cell, inputs=x,
    dtype=tf.float32)

output = tf.reshape(tf.split(outputs, 28, axis=1, num=
    None, name='split')[-1],[-1,n_hidden])
pred = tf.matmul(output, weights['out']) + biases['out']

cost = tf.reduce_mean(tf.nn.
    softmax_cross_entropy_with_logits(labels=y, logits=
    pred ))
optimizer = tf.train.AdamOptimizer(learning_rate=
    learning_rate).minimize(cost)

#define accuracy for learning
correct_pred = tf.equal(tf.argmax(pred,1), tf.argmax(y,1)
    )
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.
    float32))

init = tf.global_variables_initializer()

##Out of memory tweaks, didn't work for 4096
config = tf.ConfigProto()
#config.gpu_options.per_process_gpu_memory_fraction = 0.2
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    sess.run(init)
    step = 1
    # Keep training until reach max iterations
    while step * batch_size < training_iters:

        # We will read a batch of 100 images [100 x 784]
            as batch_x
        # batch_y is a matrix of [100x10]
        batch_x, batch_y = mnist.train.next_batch(
            batch_size)

        # We consider each row of the image as one
            sequence
        # Reshape data to get 28 seq of 28 elements, so
            that, batxh_x is [100x28x28]
        batch_x = batch_x.reshape((batch_size, n_steps,
            n_input))
        # Run optimization op (backprop)
        sess.run(optimizer, feed_dict={x: batch_x, y:
            batch_y})


        if step % display_step == 0:
            # Calculate batch accuracy
            acc = sess.run(accuracy, feed_dict={x:
                batch_x, y: batch_y})
            # Calculate batch loss
            loss = sess.run(cost, feed_dict={x: batch_x,
                y: batch_y})
        step += 1

    # Calculate accuracy for all mnist test images
    test_data = mnist.test.images.reshape((-1, n_steps,
        n_input))
    test_label = mnist.test.labels
    print("Testing Accuracy:", \
        sess.run(accuracy, feed_dict={x: test_data, y:
            test_label}))

sess.close()
currentTime = time.process_time()-startTime
print("Number of hidden layers: ", n_hidden)
print("Learning Rate: ", learning_rate)
print("Time from start: ", currentTime)
print("====================")
```

0.2 Results

Table 1: Neurons and Learning Rate Test Accuracy

Neurons/Learning Rate	0.00001	0.0001	0.001	0.01	0.1	1.0
16	0.1083	0.3873	0.8545	0.9487	0.8425	0.3734
32	0.2175	0.6249	0.9438	0.9703	0.8836	0.0892
64	0.2156	0.8204	0.9619	0.9752	0.8127	0.0958
128	0.4487	0.903	0.9688	0.9776	0.4733	0.1009
256	0.6379	0.9276	0.9705	0.9786	0.098	0.1028
512	0.764	0.9398	0.9661	0.9768	0.4154	0.098
1024	0.8381	0.945	0.9711	0.3739	0.1032	0.1135
2048	0.8541	0.9435	0.9707	0.0892	0.0982	0.1135
4096	DNF	DNF	DNF	DNF	DNF	DNF

0.2.1 Explanation

what is this telling you about the training of an LSTM with respect to the number of neurons or learning rate or both? What can you learn from this?