MNIST_for_beginners_no_convolution

December 21, 2016

1 MNIST For ML Beginners

A very simple MNIST classifier. See extensive documentation at http://tensorflow.org/tutorials/mnist/beginners/index.md

In [1]:
```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import os.path

import argparse
import sys

from tensorflow.examples.tutorials.mnist import input_data

import tensorflow as tf

flags = tf.app.flags
FLAGS = flags.FLAGS
flags.DEFINE_string('data_dir', './', 'Directory to put the training data.'
```

In [2]:
```
def main(_):
  # Import data
  mnist = input_data.read_data_sets(FLAGS.data_dir, one_hot=True)

  # Create the model
  x = tf.placeholder(tf.float32, [None, 784])
  W = tf.Variable(tf.zeros([784, 10]))
  b = tf.Variable(tf.zeros([10]))
  y = tf.matmul(x, W) + b

  # Define loss and optimizer
  y_ = tf.placeholder(tf.float32, [None, 10])

  # The raw formulation of cross-entropy,
  #
```

```
            #   tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(tf.nn.softmax(y)),
            #                                 reduction_indices=[1]))
            #
            # can be numerically unstable.
            #
            # So here we use tf.nn.softmax_cross_entropy_with_logits on the raw
            # outputs of 'y', and then average across the batch.
            cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(y,
            train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entrop

            #sess = tf.InteractiveSession()
            with tf.Session() as sess:
                #tf.global_variables_initializer().run()
                init = tf.initialize_all_variables()
                sess.run(init)
                # Train
                for _ in range(1000):
                  batch_xs, batch_ys = mnist.train.next_batch(100)
                  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

                # Test trained model
                correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
                accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
                print(">>> Test Accuracy::"+str(sess.run(accuracy, feed_dict={x: mnis
```

```
In [3]: main(_)
```

```
Extracting ./train-images-idx3-ubyte.gz
Extracting ./train-labels-idx1-ubyte.gz
Extracting ./t10k-images-idx3-ubyte.gz
Extracting ./t10k-labels-idx1-ubyte.gz
>>> Test Accuracy::0.9197
```

1.1 TensorBoard: Visualizing Learning

```
In [4]: def variable_summaries(var):
            """Attach a lot of summaries to a Tensor (for TensorBoard visualization).
            with tf.name_scope('summaries'):
              mean = tf.reduce_mean(var)
              #tf.summary.scalar('mean', mean)
              tf.scalar_summary(var.name+'_mean', mean)
              with tf.name_scope('stddev'):
                stddev = tf.sqrt(tf.reduce_mean(tf.square(var - mean)))
              #tf.summary.scalar('stddev', stddev)
              tf.scalar_summary(var.name+'_stddev', stddev)
              #tf.summary.scalar('max', tf.reduce_max(var))
              tf.scalar_summary(var.name+'_max', tf.reduce_max(var))
```

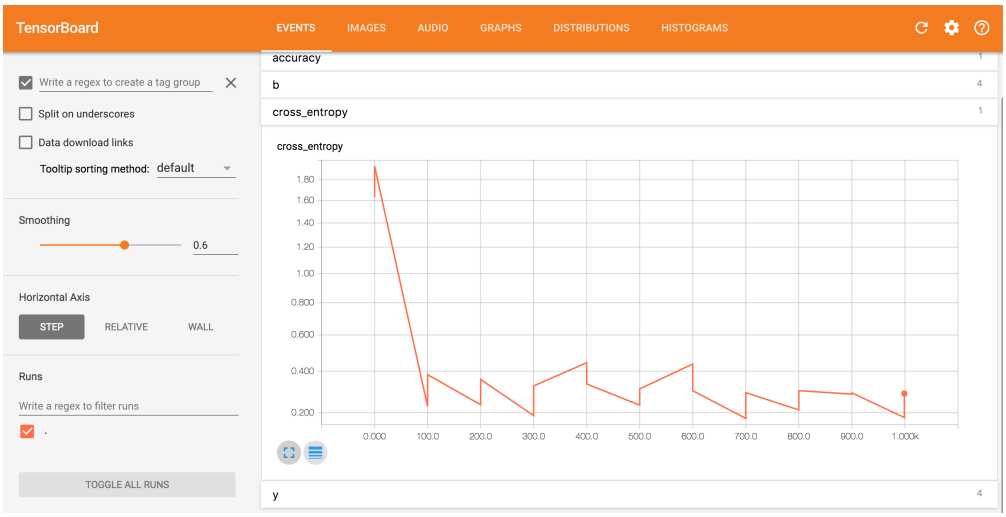
```
    #tf.summary.scalar('min', tf.reduce_min(var))
    tf.scalar_summary(var.name+'_min', tf.reduce_min(var))


def main2(_):
  # Import data
  mnist = input_data.read_data_sets(FLAGS.data_dir, one_hot=True)

  # Create the model
  x = tf.placeholder(tf.float32, [None, 784])

  with tf.name_scope('W'):
    W = tf.Variable(tf.zeros([784, 10]))
    variable_summaries(W)

  with tf.name_scope('b'):
    b = tf.Variable(tf.zeros([10]))
    variable_summaries(b)

  with tf.name_scope('y'):
    y = tf.matmul(x, W) + b
    variable_summaries(y)

  # Define loss and optimizer
  y_ = tf.placeholder(tf.float32, [None, 10])

  with tf.name_scope('cross_entropy'):
    cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits
  #tf.summary.scalar('cross_entropy', cross_entropy)
  tf.scalar_summary('cross_entropy', cross_entropy)

  with tf.name_scope('train_step'):
    train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entr

  # Test trained model
  with tf.name_scope('accuracy'):
    with tf.name_scope('correct_prediction'):
      correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))

    with tf.name_scope('accuracy'):
      accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
      #tf.summary.scalar('accuracy', accuracy)
  tf.scalar_summary('accuracy', accuracy)

  #######
  init = tf.initialize_all_variables()

  # Create a saver for writing training checkpoints.
```

```
    saver = tf.train.Saver()

    #sess = tf.InteractiveSession()
    with tf.Session() as sess:
        #tf.global_variables_initializer().run()

        # Merge all the summaries and write them out to ./logs (by default)
        merged = tf.merge_all_summaries()
        writer = tf.train.SummaryWriter(FLAGS.data_dir + '/_logs',sess.graph)

        sess.run(init)

        # Train
        iterations = 1000
        for i in range(iterations):
          batch_xs, batch_ys = mnist.train.next_batch(100)
          sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

          if i % 100 == 0 or i == (iterations-1):
            summary = sess.run(merged, feed_dict={x: batch_xs, y_: batch_ys})
            writer.add_summary(summary, i)
            summary, acc = sess.run([merged, accuracy], feed_dict={x: mnist.t
            writer.add_summary(summary, i)
            writer.flush()

            checkpoint_file = os.path.join(FLAGS.data_dir + '/_logs', 'checkp
            saver.save(sess, checkpoint_file, global_step=i)

            print('>>> Test Accuracy [%s/%s]: %s' % (i,iterations,acc))
```

```
In [5]: main2(_)

Extracting ./train-images-idx3-ubyte.gz
Extracting ./train-labels-idx1-ubyte.gz
Extracting ./t10k-images-idx3-ubyte.gz
Extracting ./t10k-labels-idx1-ubyte.gz
>>> Test Accuracy [0/1000]: 0.4075
>>> Test Accuracy [100/1000]: 0.8948
>>> Test Accuracy [200/1000]: 0.9031
>>> Test Accuracy [300/1000]: 0.9074
>>> Test Accuracy [400/1000]: 0.9037
>>> Test Accuracy [500/1000]: 0.9125
>>> Test Accuracy [600/1000]: 0.9135
>>> Test Accuracy [700/1000]: 0.918
>>> Test Accuracy [800/1000]: 0.9152
>>> Test Accuracy [900/1000]: 0.9188
>>> Test Accuracy [999/1000]: 0.9193
```

To run TensorBoard, use the following command (alternatively python -m tensorflow.tensorboard) `{c} >>tensorboard --logdir=_logs`

1.1.1 Cross Entropy on training set by step



title

1.1.2 Accuracy on test set by step (learning curve)

title