TensorBoard：可视化学习

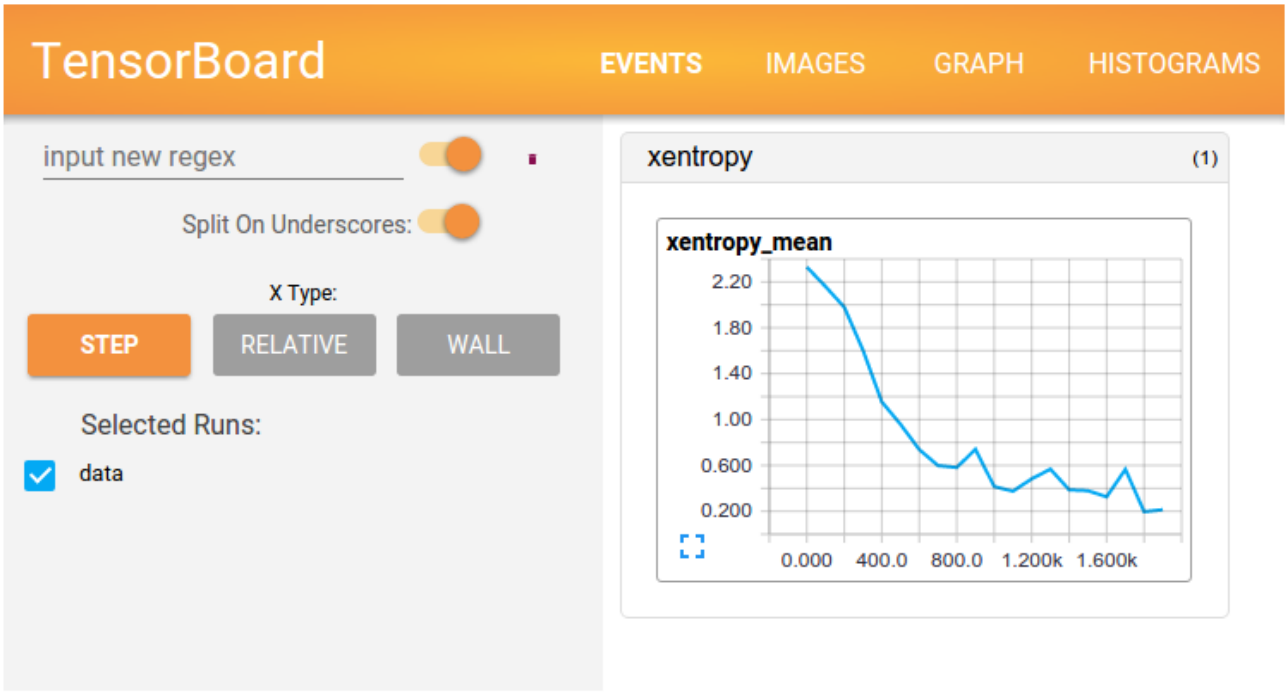
原文链接：https://www.tensorflow.org/get_started/summaries_and_tensorboard

译文链接：http://www.apache.wiki/pages/viewpage.action?pageId=10029491

贡献者：片刻 ApacheCN Apache中文网

校对：Ngxin

您使用TensorFlow的计算 ，比如训练一个庞大的深层神经网络
，这可能是复杂和混乱的。为了更容易理解、调试和优化TensorFlow程序，我们提供了一套名为TensorBoard的可视化工具。您可以使用TensorBoard可视化您的TensorFlow图形，绘制关于图形执行的quantitative
metrics，通过图像的形式，显示其他数据。当TensorBoard完全配置后，它看起来像这样：



本教程旨在让您使用简单的TensorBoard用法。也有其他文档可以教会您这些！TensorBoard README 有大量关于TensorBoard使用的信息
，包括提示和技巧，和调试信息。

序列化数据

TensorBoard通过读取TensorFlow事件文件来运行操作，该文件包含在运行TensorFlow时生成的主要数据。以下内容是TensorBoard中汇总数据的大体生命周期。

首先，创建要收集汇总（summary)数据的TensorFlow图，并确定要在哪些节点使用汇总操作（summary operations)。

例如，假设您正在训练用于识别MNIST数字的卷积神经网络。您想记录学习率随时间变化的方式，以及目标函数的变化。通过将`tf.summary.scalar`操作附加到分别输出学习速率和丢失的节点来收集这些数据。然后，给每`scalar_summary`一个`tag`，`'learning rate'`或`'loss function'`。

也许您也希望可视化特定层次的激活分布，或梯度、权重的分布。通过向梯度输出和权重变量添加`tf.summary.histogram`操作收集数据。

有关可用的所有summary操作的详细信息，请查看 summary operations的文档 。

在TensorFlow中，所有的操作只有当你主动运行，或者依赖于它的输出操作时才会运行。而我们刚刚创建的汇总（summary)节点围绕着你的图形：您当前运行的操作不依赖于它们。因此，为了生成summaries，我们需要运行所有这些summary节点。手工管理它们将是乏味的，所以使用 `tf.summary.merge_all` 将它们组合成一个单一的操作，生成所有的summary数据。

然后，您可以运行合并的summary操作，这将`Summary`在给定步骤中生成一个序列化的protobuf对象，其中 包含所有summary数据。最后，要将此summary数据写入磁盘，将summary protobuf传递给 `tf.summary.FileWriter`。

`FileWriter`构造函数包含logdir -这个logdir是非常重要的，所有的事件都将被写在该目录下。此外，`FileWriter`可以在其构造函数中选择`Graph`。如果它接收到一个`Graph`对象，那么TensorBoard会将您的图形与张量形状信息一起显示。这将使您更好地了解流经图形的内容：请参阅 张量形状信息。

现在您已经修改了图表，并且也有`FileWriter`，准备好开始运行网络了！如果需要，您可以每一步运行合并summary，并记录大量的训练数据。尽管如此，这可能比您需要的数据更多。因此，考虑在每n步中运行合并summary。

下面的代码示例是 简单的MNIST教程的修改，其中我们添加了一些summary操作，并且每十步运行它们。如果你运行这个然后启动`tensorboard --logdir=/tmp/mnist_logs`，你将能够可视化的统计数据，比如在训练过程中权重或准确率如何变化。下面的代码是摘录; 全部来源在 这里。

```
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)
    with tf.name_scope('stddev'):
      stddev = tf.sqrt(tf.reduce_mean(tf.square(var - mean)))
    tf.summary.scalar('stddev', stddev)
    tf.summary.scalar('max', tf.reduce_max(var))
    tf.summary.scalar('min', tf.reduce_min(var))
    tf.summary.histogram('histogram', var)

def nn_layer(input_tensor, input_dim, output_dim, layer_name,
act=tf.nn.relu):
  """Reusable code for making a simple neural net layer.

  It does a matrix multiply, bias add, and then uses relu to
nonlinearize.
  It also sets up name scoping so that the resultant graph is easy to
read,
  and adds a number of summary ops.
  """
  # Adding a name scope ensures logical grouping of the layers in the
graph.
  with tf.name_scope(layer_name):
    # This Variable will hold the state of the weights for the layer
    with tf.name_scope('weights'):
      weights = weight_variable([input_dim, output_dim])
      variable_summaries(weights)
    with tf.name_scope('biases'):
      biases = bias_variable([output_dim])
      variable_summaries(biases)
    with tf.name_scope('Wx_plus_b'):
      preactivate = tf.matmul(input_tensor, weights) + biases
      tf.summary.histogram('pre_activations', preactivate)
    activations = act(preactivate, name='activation')
    tf.summary.histogram('activations', activations)
    return activations

hidden1 = nn_layer(x, 784, 500, 'layer1')

with tf.name_scope('dropout'):
  keep_prob = tf.placeholder(tf.float32)
```

```
  tf.summary.scalar('dropout_keep_probability', keep_prob)
  dropped = tf.nn.dropout(hidden1, keep_prob)

# Do not apply softmax activation yet, see below.
y = nn_layer(dropped, 500, 10, 'layer2', act=tf.identity)

with tf.name_scope('cross_entropy'):
  # The raw formulation of cross-entropy,
  #
  # tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(tf.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 the nn_layer above, and then average across
  # the batch.
  diff = tf.nn.softmax_cross_entropy_with_logits(targets=y_, logits=y)
  with tf.name_scope('total'):
    cross_entropy = tf.reduce_mean(diff)
tf.summary.scalar('cross_entropy', cross_entropy)

with tf.name_scope('train'):
  train_step = tf.train.AdamOptimizer(FLAGS.learning_rate).minimize(
      cross_entropy)

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)

# Merge all the summaries and write them out to /tmp/mnist_logs (by
default)
merged = tf.summary.merge_all()
train_writer = tf.summary.FileWriter(FLAGS.summaries_dir + '/train',
```

```
                                        sess.graph)
test_writer = tf.summary.FileWriter(FLAGS.summaries_dir + '/test')
tf.global_variables_initializer().run()
```

FileWritersFileWriterssummary

```
# Train the model, and also write summaries.
# Every 10th step, measure test-set accuracy, and write test summaries
# All other steps, run train_step on training data, & add training
summaries

def feed_dict(train):
  """Make a TensorFlow feed_dict: maps data onto Tensor placeholders."""
  if train or FLAGS.fake_data:
    xs, ys = mnist.train.next_batch(100, fake_data=FLAGS.fake_data)
    k = FLAGS.dropout
  else:
    xs, ys = mnist.test.images, mnist.test.labels
    k = 1.0
  return {x: xs, y_: ys, keep_prob: k}

for i in range(FLAGS.max_steps):
  if i % 10 == 0:  # Record summaries and test-set accuracy
    summary, acc = sess.run([merged, accuracy],
feed_dict=feed_dict(False))
    test_writer.add_summary(summary, i)
    print('Accuracy at step %s: %s' % (i, acc))
  else:  # Record train set summaries, and train
    summary, _ = sess.run([merged, train_step],
feed_dict=feed_dict(True))
    train_writer.add_summary(summary, i)
```

TensorBoard

启动TensorBoard

要运行TensorBoard，请使用以下命令（可选`python -m tensorflow.tensorboard`）

```
tensorboard --logdir=path/to/log-directory
```

logdirFileWriterlogdirTensorBoard
TensorBoardlocalhost:6006TensorBoard

查看TensorBoard时，您会看到右上角的导航标签。每个选项卡表示可以可视化的一组序列化数据。

有关如何使用图形选项卡可视化图形的更详细信息，请参阅TensorBoard：图形可视化。

有关TensorBoard的更多使用信息，请参阅TensorBoard README。