PART 2 TensorFlow

2. TensorBoard - 學習過程可視化技術

> Ref : " TensorBoard: Visualizing Learning "
> https://www.tensorflow.org/guide/summaries_and_tensorboard (https://www.tensorflow.org/guide/summaries_and_tensorboard)
>
> - [Youtube] : " Hands-on TensorBoard (TensorFlow Dev Summit 2017) "
>   https://youtu.be/eBbEDRsCmv4 (https://youtu.be/eBbEDRsCmv4)
> - [Code] : https://goo.gl/ZwGnPE (https://goo.gl/ZwGnPE)

In [ ]:

```
import os
import os.path
import shutil

import tensorflow as tf
# for the old-version usage of TensorFlow, such as tensorflow.examp
old_v = tf.logging.get_verbosity()
tf.logging.set_verbosity(tf.logging.ERROR)
```

Loading MNIST Dataset...

In [ ]:

```
LOGDIR = "./mnist_tutorial/"
LABELS = os.path.join(os.getcwd(), "labels_1024.tsv")
SPRITES = os.path.join(os.getcwd(), "sprite_1024.png")

### MNIST EMBEDDINGS ###
mnist = tf.contrib.learn.datasets.mnist.read_data_sets(train_dir=LO
### Get a sprite and labels file for the embedding projector ###

if not (os.path.isfile(LABELS) and os.path.isfile(SPRITES)):
  print("Necessary data files were not found. Run this command from
    "repo provided at "
    "https://github.com/dandelionmane/tf-dev-summit-tensorboard-tut
  exit(1)

# shutil.copyfile(LABELS, os.path.join(LOGDIR, LABELS))
# shutil.copyfile(SPRITES, os.path.join(LOGDIR, SPRITES))
```

Building the Computation Graph for TensorBoard

In [ ]:

```
def conv_layer(input, size_in, size_out, name="conv"):
  with tf.name_scope(name):
    w = tf.Variable(tf.truncated_normal([5, 5, size_in, size_out],
    b = tf.Variable(tf.constant(0.1, shape=[size_out]), name="B")
    conv = tf.nn.conv2d(input, w, strides=[1, 1, 1, 1], padding="SA
    act = tf.nn.relu(conv + b)
    tf.summary.histogram("weights", w)
    tf.summary.histogram("biases", b)
    tf.summary.histogram("activations", act)
    return tf.nn.max_pool(act, ksize=[1, 2, 2, 1], strides=[1, 2, 2


def fc_layer(input, size_in, size_out, name="fc"):
  with tf.name_scope(name):
    w = tf.Variable(tf.truncated_normal([size_in, size_out], stddev
    b = tf.Variable(tf.constant(0.1, shape=[size_out]), name="B")
    act = tf.matmul(input, w) + b
    tf.summary.histogram("weights", w)
    tf.summary.histogram("biases", b)
    tf.summary.histogram("activations", act)
    return act


def mnist_model(learning_rate, use_two_fc, use_two_conv, hparam):
  tf.reset_default_graph()
  sess = tf.Session()

  # Setup placeholders, and reshape the data
  x = tf.placeholder(tf.float32, shape=[None, 784], name="x")
  x_image = tf.reshape(x, [-1, 28, 28, 1])
  tf.summary.image('input', x_image, 3)
  y = tf.placeholder(tf.float32, shape=[None, 10], name="labels")

  if use_two_conv:
    conv1 = conv_layer(x_image, 1, 32, "conv1")
    conv_out = conv_layer(conv1, 32, 64, "conv2")
  else:
    conv_out = conv_layer(x_image, 1, 16, "conv")

  flattened = tf.reshape(conv_out, [-1, 7 * 7 * 64])


  if use_two_fc:
    fc1 = fc_layer(flattened, 7 * 7 * 64, 1024, "fc1")
    relu = tf.nn.relu(fc1)
    embedding_input = relu
    tf.summary.histogram("fc1/relu", relu)
    embedding_size = 1024
    logits = fc_layer(relu, 1024, 10, "fc2")
  else:
    embedding_input = flattened
    embedding_size = 7*7*64
    logits = fc_layer(flattened, 7*7*64, 10, "fc")

  with tf.name_scope("xent"):
    xent = tf.reduce_mean(
        tf.nn.softmax_cross_entropy_with_logits(
```

```
            logits=logits, labels=y), name="xent")
    tf.summary.scalar("xent", xent)

  with tf.name_scope("train"):
    train_step = tf.train.AdamOptimizer(learning_rate).minimize(xen

  with tf.name_scope("accuracy"):
    correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(y
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float3
    tf.summary.scalar("accuracy", accuracy)

  summ = tf.summary.merge_all()


  embedding = tf.Variable(tf.zeros([1024, embedding_size]), name="t
  assignment = embedding.assign(embedding_input)
  saver = tf.train.Saver()

  sess.run(tf.global_variables_initializer())
  writer = tf.summary.FileWriter(LOGDIR + hparam)
  writer.add_graph(sess.graph)

  config = tf.contrib.tensorboard.plugins.projector.ProjectorConfig
  embedding_config = config.embeddings.add()
  embedding_config.tensor_name = embedding.name
  embedding_config.sprite.image_path = SPRITES
  embedding_config.metadata_path = LABELS
  # Specify the width and height of a single thumbnail.
  embedding_config.sprite.single_image_dim.extend([28, 28])
  tf.contrib.tensorboard.plugins.projector.visualize_embeddings(wri

  for i in range(2001):
    batch = mnist.train.next_batch(100)
    if i % 5 == 0:
      [train_accuracy, s] = sess.run([accuracy, summ], feed_dict={x
      writer.add_summary(s, i)
    if i % 500 == 0:
      sess.run(assignment, feed_dict={x: mnist.test.images[:1024],
      saver.save(sess, os.path.join(LOGDIR, "model.ckpt"), i)
    sess.run(train_step, feed_dict={x: batch[0], y: batch[1]})

def make_hparam_string(learning_rate, use_two_fc, use_two_conv):
  conv_param = "conv=2" if use_two_conv else "conv=1"
  fc_param = "fc=2" if use_two_fc else "fc=1"
  return "lr_%.0E,%s,%s" % (learning_rate, conv_param, fc_param)
```

Launching the Computation Graph...

In [ ]:

```
def main():
  # You can try adding some more learning rates
  for learning_rate in [1E-3, 1E-4]:

    # Include "False" as a value to try different model architectur
    for use_two_fc in [True]:
      for use_two_conv in [False, True]:
        # Construct a hyperparameter string for each one (example:
        hparam = make_hparam_string(learning_rate, use_two_fc, use_
        print('Starting run for %s' % hparam)

        # Actually run with the new settings
        mnist_model(learning_rate, use_two_fc, use_two_conv, hparam
  print('Done training!')
  print('Run `tensorboard --logdir=%s` to see the results.' % LOGDI
  print('Running on mac? If you want to get rid of the dialogue ask
        'network permissions to TensorBoard, you can provide this f
        '--host=localhost')

if __name__ == '__main__':
  main()
```

To run TensorBoard, run the following command on `Anaconda Prompt` :

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

- For instance, **`tensorboard --logdir=/Users/macmini1/Documents/ipynb-AI_DL-TensorFlow/mnist_tutorial/`**

Connecting to **`http://localhost:6006`**

How to Save and Restore Variables

- Ref: https://www.tensorflow.org/guide/saved_model#save_and_restore_variables (https://www.tensorflow.org/guide/saved_model#save_and_restore_variables)

In [3]:

```
import tensorflow as tf
print(tf.__version__)
```

```
1.12.0
```

1. Save variables

- Create a Saver with `tf.train.Saver()` to manage all variables in the model.
- For example, the following snippet demonstrates how to call the `tf.train.Saver.save` method to save variables to checkpoint files:

In [4]:

```
# Create some variables.
v1 = tf.get_variable("v1", shape=[3], initializer = tf.zeros_initia
v2 = tf.get_variable("v2", shape=[5], initializer = tf.zeros_initia

inc_v1 = v1.assign(v1+1)
dec_v2 = v2.assign(v2-1)
```

In [5]:

```
# Add an op to initialize the variables.
init_op = tf.global_variables_initializer()

# Add ops to save and restore all the variables.
saver = tf.train.Saver()

# Later, launch the model, initialize the variables, do some work,
# variables to disk.
with tf.Session() as sess:
  sess.run(init_op)
  # Do some work with the model.
  inc_v1.op.run()
  dec_v2.op.run()
  # Save the variables to disk.
  save_path = saver.save(sess, "./tmp/model.ckpt")
  print("Model saved in path: %s" % save_path)
```

```
Model saved in path: ./tmp/model.ckpt
```

2. Restore variables

- The `tf.train.Saver` object not only saves variables to checkpoint files, it also restores variables. Note that when you restore variables you do not have to initialize them beforehand.
- For example, the following snippet demonstrates how to call the `tf.train.Saver.restore` method to restore variables from the checkpoint files:

In [6]:

```
tf.reset_default_graph()

# Create some variables.
v1 = tf.get_variable("v1", shape=[3])
v2 = tf.get_variable("v2", shape=[5])

# Add ops to save and restore all the variables.
saver = tf.train.Saver()

# Later, launch the model, use the saver to restore variables from
# do some work with the model.
with tf.Session() as sess:
  # Restore variables from disk.
  saver.restore(sess, "tmp/model.ckpt")
  print("Model restored.")
  # Check the values of the variables
  print("v1 : %s" % v1.eval())
  print("v2 : %s" % v2.eval())
```

```
INFO:tensorflow:Restoring parameters from tmp/model.ckpt
Model restored.
v1 : [1. 1. 1.]
v2 : [-1. -1. -1. -1. -1.]
```

3. Choose variables to save and restore

In [7]:

```
tf.reset_default_graph()
# Create some variables.
v1 = tf.get_variable("v1", [3], initializer = tf.zeros_initializer)
v2 = tf.get_variable("v2", [5], initializer = tf.zeros_initializer)

# Add ops to save and restore only `v2` using the name "v2"
saver = tf.train.Saver({"v2": v2})

# Use the saver object normally after that.
with tf.Session() as sess:
  # Initialize v1 since the saver will not.
  v1.initializer.run()
  saver.restore(sess, "tmp/model.ckpt")

  print("v1 : %s" % v1.eval())
  print("v2 : %s" % v2.eval())
```

```
INFO:tensorflow:Restoring parameters from tmp/model.ckpt
v1 : [0. 0. 0.]
v2 : [-1. -1. -1. -1. -1.]
```

4. Inspect variables in a checkpoint

In [8]:

```python
# import the inspect_checkpoint library
from tensorflow.python.tools import inspect_checkpoint as chkp

# print all tensors in checkpoint file
chkp.print_tensors_in_checkpoint_file("tmp/model.ckpt", tensor_name

# tensor_name:  v1
# [ 1.  1.  1.]
# tensor_name:  v2
# [-1. -1. -1. -1. -1.]

# print only tensor v1 in checkpoint file
chkp.print_tensors_in_checkpoint_file("tmp/model.ckpt", tensor_name

# tensor_name:  v1
# [ 1.  1.  1.]

# print only tensor v2 in checkpoint file
chkp.print_tensors_in_checkpoint_file("tmp/model.ckpt", tensor_name

# tensor_name:  v2
# [-1. -1. -1. -1. -1.]
```

```
tensor_name:  v1
[1. 1. 1.]
tensor_name:  v2
[-1. -1. -1. -1. -1.]
tensor_name:  v1
[1. 1. 1.]
tensor_name:  v2
[-1. -1. -1. -1. -1.]
```

Save and Restore Models

[REFERENCE]

- **TensorFlow - "Save and Store"**, https://www.tensorflow.org/guide/saved_model#save_and_restore_models (https://www.tensorflow.org/guide/saved_model#save_and_restore_models)
- **"TensorFlow-7-TensorBoard Embedding可視化"**, https://www.jianshu.com/p/d5339d04aa17 (https://www.jianshu.com/p/d5339d04aa17)
- **"tensorflow保存和恢复模型的两种方法介绍"**, https://zhuanlan.zhihu.com/p/31417693 (https://zhuanlan.zhihu.com/p/31417693)