Manage Experiments

TensorFlow for Deep Learning Research
Lecture 5

Agenda

More word2vec

tf.train.Saver

tf.summary

Randomization

Data Readers

Where are the gradients?

Reverse mode automatic differentiation

Reverse mode automatic differentiation

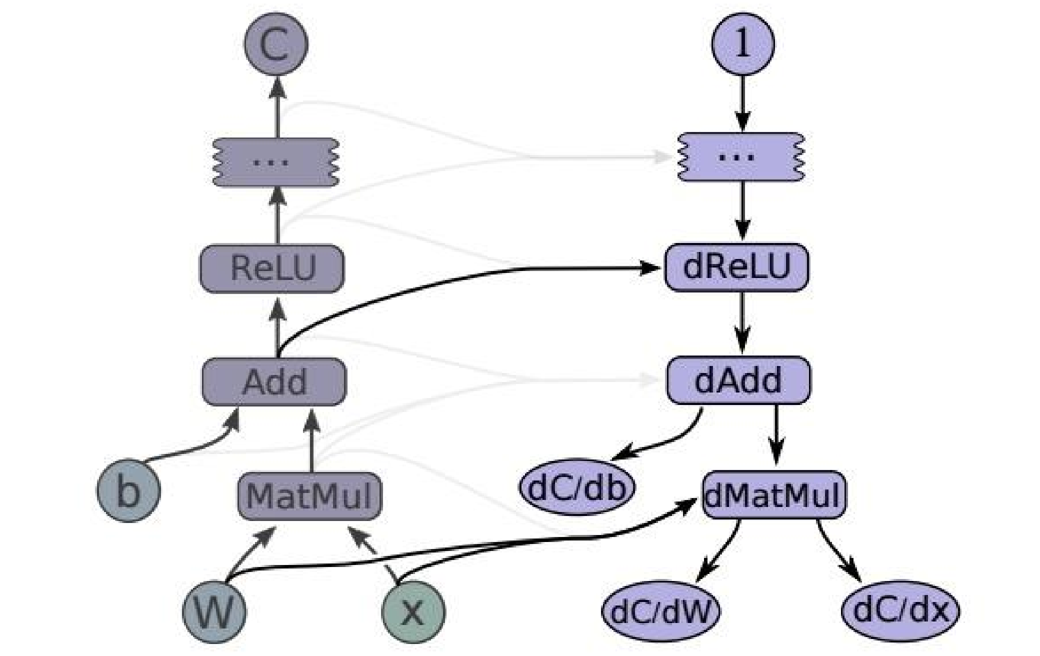


Figure 5: Gradients computed for graph in Figure 2

tf.gradients(y, [xs])

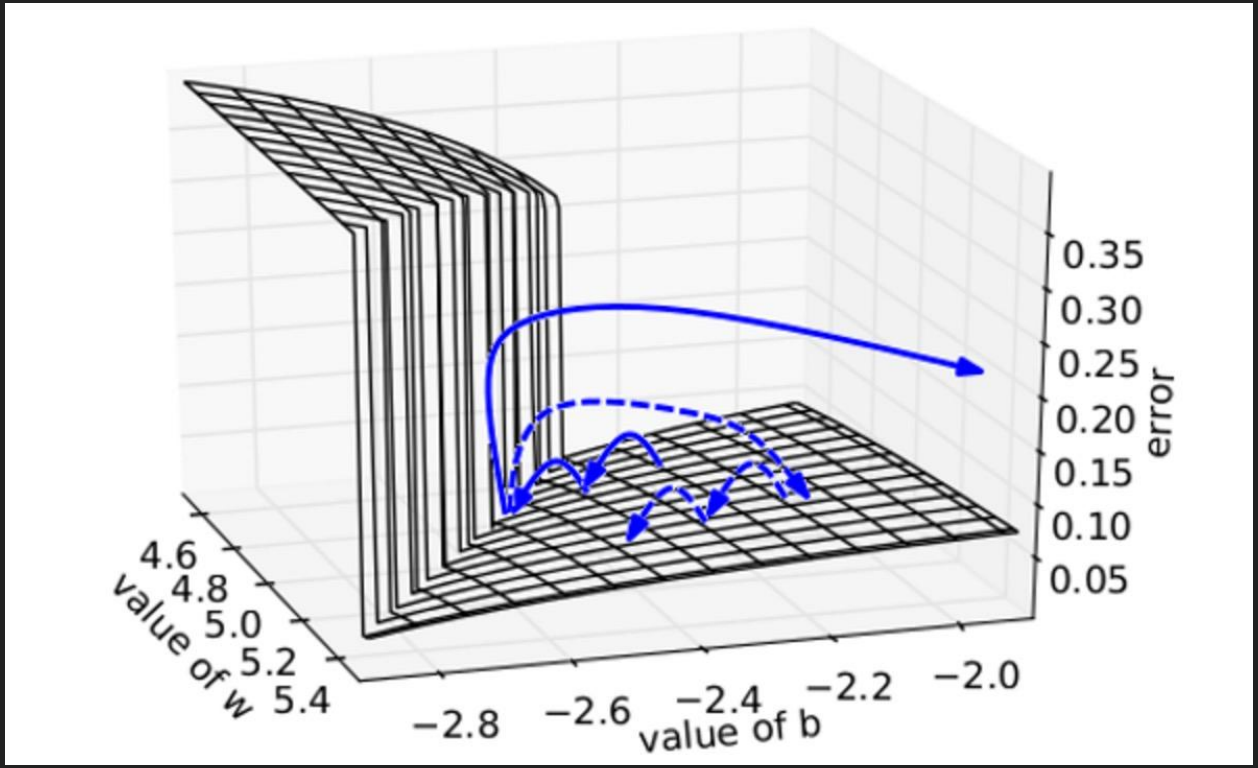
Take derivative of y with respect to each tensor in the list [xs]

tf.gradients(y, [xs])

```
x = tf.Variable(2.0)

y = 2.0 * (x ** 3)

z = 3.0 + y ** 2

grad_z = tf.gradients(z, [x, y])

with tf.Session() as sess:

      sess.run(x.initializer)

      print sess.run(grad_z) # >> [768.0, 32.0]

# 768 is the gradient of z with respect to x, 32 with respect to y
```

Should I still learn to take gradients?

Vanishing/exploding gradients



Plot by “Understanding the exploding gradient problem”, Pascanu et al. (2012)

Structure our model

We’ve dumped everything into one giant function word2vec (check minus for style in CS106)

Need models to be reusable

Yay, object oriented programming!!

```
class SkipGramModel:
    """ Build the graph for word2vec model """
    def __init__(self, params):
        pass

    def _create_placeholders(self):
        """ Step 1: define the placeholders for input and output """
        pass

    def _create_embedding(self):
        """ Step 2: define weights. In word2vec, it's actually the weights that we care about """
        pass

    def _create_loss(self):
        """ Step 3 + 4: define the inference + the loss function """
        pass

    def _create_optimizer(self):
        """ Step 5: define optimizer """
        pass
```

Manage experiments

tf.train.Saver

saves graph's variables in binary files

Saves sessions, not graphs!

```
tf.train.Saver.save(sess, save_path,
    global_step=None...)
```

Saves sessions, not graphs!

```
tf.train.Saver.save(sess, save_path,
global_step=None...)
```

Save parameters after 1000 steps

```
# define model

# create a saver object
saver = tf.train.Saver()

# launch a session to compute the graph
with tf.Session() as sess:
    # actual training loop
     for step in range(training_steps):
          sess.run([optimizer])

          if (step + 1) % 1000==0:
               saver.save(sess, 'checkpoint_directory/model_name',
                                global_step=model.global_step)
```

Each saved step is a checkpoint

```
# define model

# create a saver object
saver = tf.train.Saver()

# launch a session to compute the graph
with tf.Session() as sess:
    # actual training loop
     for step in range(training_steps):
          sess.run([train_op])

          if (step + 1) % 1000==0:
               saver.save(sess, 'checkpoint_directory/model_name',
                              global_step=model.global_step)
```

Global step

Very common in TensorFlow program

```
self.global_step = tf.Variable(0, dtype=tf.int32, trainable=False,
                               name='global_step')
```

Global step

Need to tell optimizer to increment global step

```
self.global_step = tf.Variable(0, dtype=tf.int32, trainable=False,
                               name='global_step')

self.optimizer = tf.train.GradientDescentOptimizer(self.lr).minimize(self.loss,

                                                        global_step=self.global_step)
```

tf.train.Saver

Only save variables, not graph

Checkpoints map variable names to tensors

Restore variables

```
saver.restore(sess, 'checkpoints/name_of_the_checkpoint')

e.g. saver.restore(sess, 'checkpoints/skip-gram-99999')
```

Restore the latest checkpoint

```
ckpt = tf.train.get_checkpoint_state(os.path.dirname('checkpoints/checkpoint'))

if ckpt and ckpt.model_checkpoint_path:
     saver.restore(sess, ckpt.model_checkpoint_path)
```

1. checkpoint keeps track of the latest checkpoint
2. Safeguard to restore checkpoints only when there are checkpoints

tf.summary

Why matplotlib when you can summarize?

tf.summary

Visualize our summary statistics during our training

```
tf.summary.scalar
tf.summary.histogram
tf.summary.image
```

Step 1: create summaries

```
with tf.name_scope("summaries"):
    tf.summary.scalar("loss", self.loss
    tf.summary.scalar("accuracy", self.accuracy)
    tf.summary.histogram("histogram loss", self.loss)
    # merge them all
    self.summary_op = tf.summary.merge_all()
```

Step 2: run them

```
loss_batch, _, summary = sess.run([model.loss, model.optimizer,
                                   model.summary_op],
                                   feed_dict=feed_dict)
```

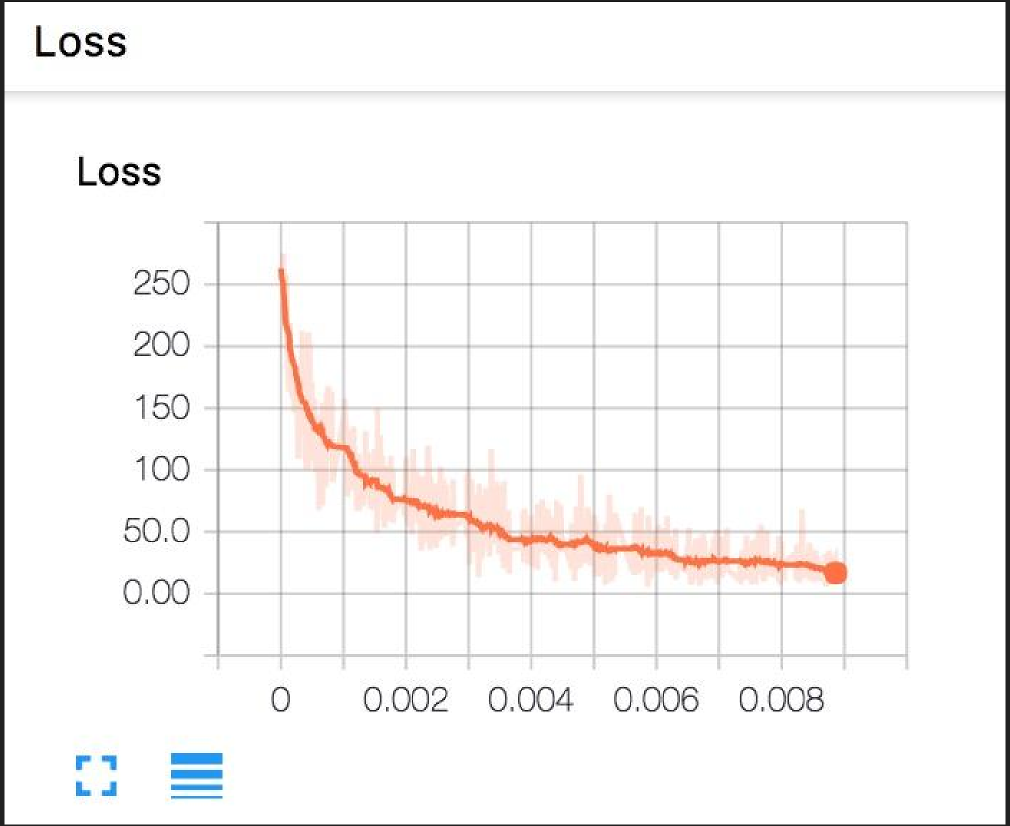
Like everything else in TF, summaries are ops

Step 3: write summaries to file

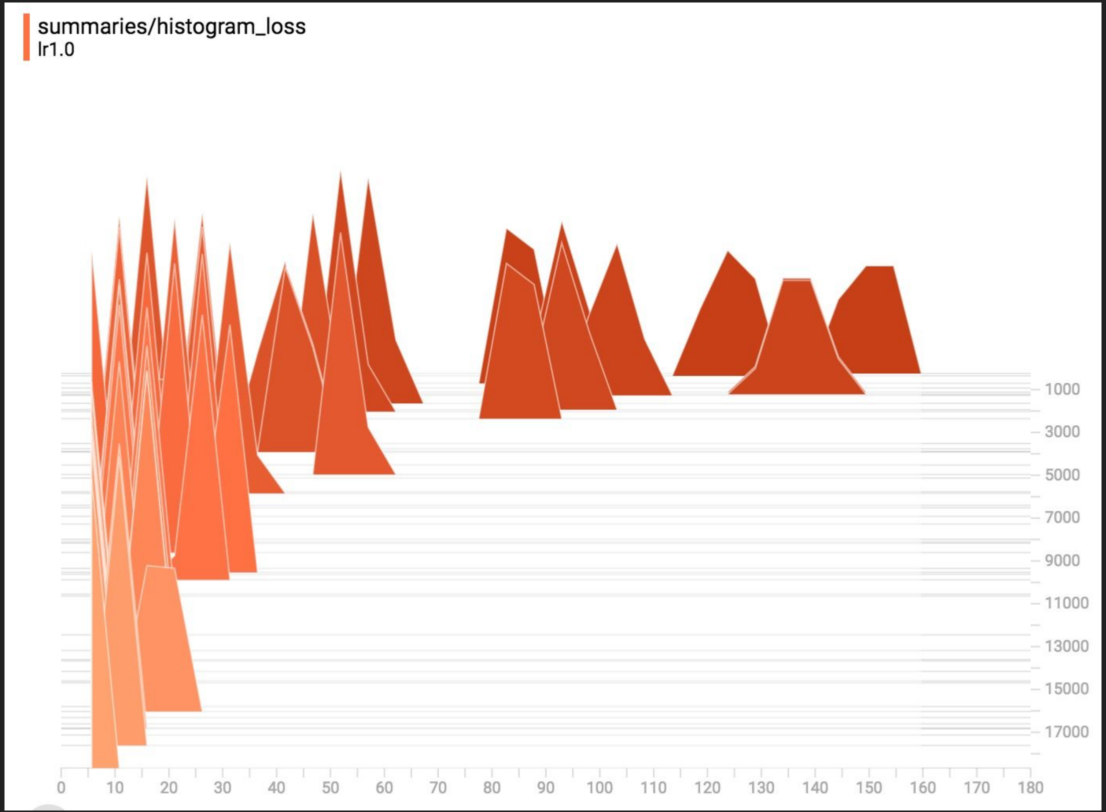
```
writer.add_summary(summary, global_step=step)
```

See summaries on TensorBoard

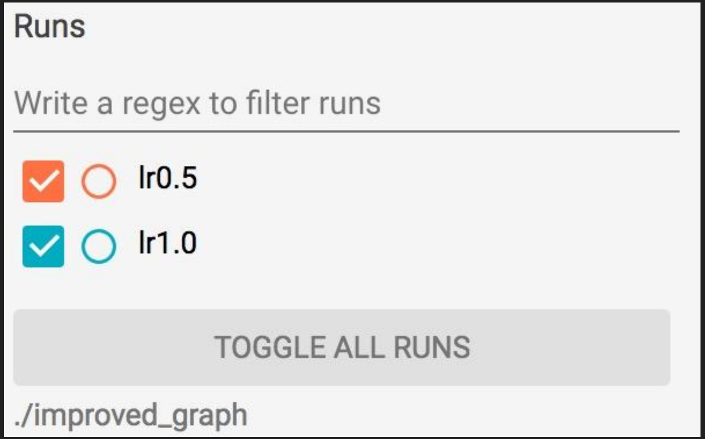
Scalar loss

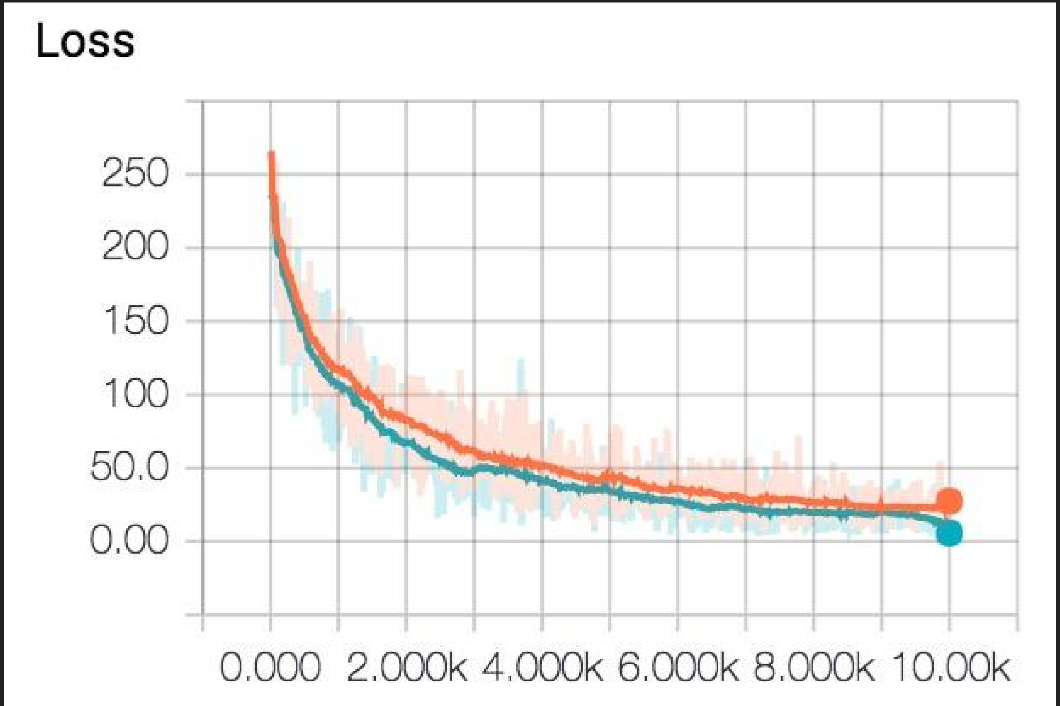


Histogram loss



Toggle run to compare experiments





Control Randomization

Op level random seed

e.g.

```
my_var = tf.Variable(tf.truncated_normal((-1.0,1.0), stddev=0.1, seed=0))
```

Sessions keep track of random state

```
c = tf.random_uniform([], -10, 10, seed=2)

with tf.Session() as sess:
     print sess.run(c) # >> 3.57493
     print sess.run(c) # >> -5.97319
----
c = tf.random_uniform([], -10, 10, seed=2)

with tf.Session() as sess:
     print sess.run(c) # >> 3.57493

with tf.Session() as sess:
     print sess.run(c) # >> 3.57493
```

Each new session restarts the random state

Op level seed: each op keeps its own seed

```
c = tf.random_uniform([], -10, 10, seed=2)
d = tf.random_uniform([], -10, 10, seed=2)

with tf.Session() as sess:
     print sess.run(c) # >> 3.57493
     print sess.run(d) # >> 3.57493
```

Graph level seed

```
tf.set_random_seed(seed)
(example: live coding)
```

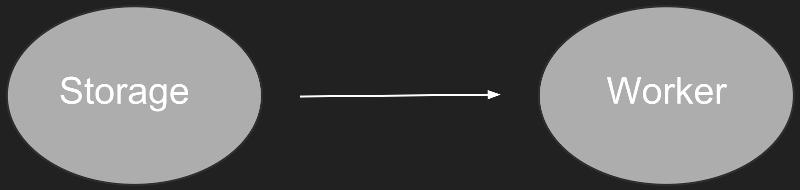
Data Readers

Problem with feed_dict?

Problem with feed_dict?

Slow when client and workers are on different machines

Data Readers



Readers allow us to load data directly into the worker process.

Data Readers

Ops that return different values every time you call them
(Think Python's generator)

Different Readers for different file types

```
tf.TextLineReader
Outputs the lines of a file delimited by newlines
E.g. text files, CSV files
```

Different Readers for different file types

```
tf.TextLineReader
Outputs the lines of a file delimited by newlines
E.g. text files, CSV files

tf.FixedLengthRecordReader
Outputs the entire file when all files have same fixed lengths
E.g. each MNIST file has 28 x 28 pixels, CIFAR-10 32 x 32 x 3
```

Different Readers for different file types

```
tf.TextLineReader
Outputs the lines of a file delimited by newlines
E.g. text files, CSV files

tf.FixedLengthRecordReader
Outputs the entire file when all files have same fixed lengths
E.g. each MNIST file has 28 x 28 pixels, CIFAR-10 32 x 32 x 3

tf.WholeFileReader
Outputs the entire file content
```

Different Readers for different file types

tf.TextLineReader
Outputs the lines of a file delimited by newlines
E.g. text files, CSV files

tf.FixedLengthRecordReader
Outputs the entire file when all files have same fixed lengths
E.g. each MNIST file has 28 x 28 pixels, CIFAR-10 32 x 32 x 3

tf.WholeFileReader
Outputs the entire file content

tf.TFRecordReader
Reads samples from TensorFlow’s own binary format (TFRecord)

Different Readers for different file types

```
tf.TextLineReader
Outputs the lines of a file delimited by newlines
E.g. text files, CSV files

tf.FixedLengthRecordReader
Outputs the entire file when all files have same fixed lengths
E.g. each MNIST file has 28 x 28 pixels, CIFAR-10 32 x 32 x 3

tf.WholeFileReader
Outputs the entire file content

tf.TFRecordReader
Reads samples from TensorFlow’s own binary format (TFRecord)

tf.ReaderBase
To allow you to create your own readers
```

Read in files from queues

```
filename_queue = tf.train.string_input_producer(["file0.csv", "file1.csv"])

reader = tf.TextLineReader()
key, value = reader.read(filename_queue)
```

tf.FIFOQueue


```
q = tf.FIFOQueue(3, "float")
init = q.enqueue_many(([0.,0.,0.],))

x = q.dequeue()
y = x+1
q_inc = q.enqueue([y])

init.run()
q_inc.run()
q_inc.run()
q_inc.run()
q_inc.run()
```

Threads & Queues

You can use tf.Coordinator and tf.QueueRunner to manage your queues

Threads & Queues

```
with tf.Session() as sess:
     # start populating the filename queue.
     coord = tf.train.Coordinator()
     threads = tf.train.start_queue_runners(coord=coord)
```

More on this in week 8

Next class

Guest lecture by Justin Johnson

Convnet

Style Transfer

Feedback: 49261200@qq.com

Thanks!