Deep Learning: Theory and Practice

Tensorflow Practical Session

Overview

Material

`https://github.com/dalab/deep-learning-workshop-public.git`

Program

1. Session 1: Introduction & simple logistic regression
2. Session 2: Convolutional neural networks for image classification
3. Session 3: Recurrent and convolutional neural networks for text sentiment

Requirements for practical exercises

- Have tensorflow installed
- Have the datasets downloaded

Quick introduction

Now

1. Tensorflow basics
2. Constructing a simple model and training it
3. Monitoring, Printing, Debugging

Variables and Tensors in Tensorflow

Variables

- Maintain their state
- Different methods to create...

```
W1 = tf.Variable([[0.1,2.2], [-2.7,0.3]], name = "weights")
W2 = tf.Variable(tf.zeros([200, 200]), name = "weights")
W3 = tf.Variable(tf.random_uniform([200, 200], -10, 10), name = "weights")
```

Tensors

At every node of the graph, the result of a computation is a tensor.
Tensors have...

- ... a shape, e.g.
  - {} (scalar)
  - [20] (20-d vector)
  - [50,10] (50x10 matrix)
  - [32,3,100,100] (higher order tensors)
- ...a type e.g. `tf.int32`, `tf.float32`, `tf.bool`, `tf.string`

The graph concept in TF

- Variables
- (Input) Placeholders

  ```
  x = tf.placeholder(tf.float32, [200,10], "input")
  ```

- Operations (e.g. for matrices `A,B,C`)

  ```
  C = A - B
  C = tf.matmul(A,B)
  C = tf.nn.relu(A)
  ```

  Common operations include

  - Pointwise: `tf.mul`, `tf.add`, `tf.sigmoid`, `tf.tanh`, `tf.nn.relu`
  - Summing and averaging: `tf.reduce_sum`, `tf.reduce_mean`

Initialization and Sessions

Starting the session fixes the graph and places the ops on devices.

```
x = tf.placeholder(tf.float32, [200,10], "input") # Input placeholder
W_1 = tf.Variable(tf.zeros([200, 200]), name = "weights")
hidden_1 = ... # Some deep network here
hidden_2 = ...

...
y = tf....     # Network output

# Run graph to get output given an input
with tf.Session() as session:
  datapoint = ...  # some np array of size [200, 10]
  feed_dict = {x : datapoint} # Map TF placeholders to numpy arrays
  y_output = session.run(y, feed_dict = feed_dict)   #run the graph

  print("Output is " + y_output)
```

Training

Example:

```
loss = ...   # some scalar tensor
opt = tf.train.GradientDescentOptimizer(0.01) # here fixed learning rate
update_step = opt.minimize(loss, global_step)
```

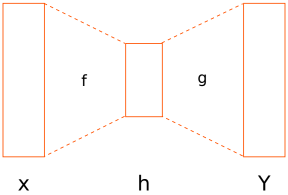
See also
https://www.tensorflow.org/versions/r0.9/api_docs/python/train.html

Example: Autoencoder

The Problem

Given data	$\mathcal{X} = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$
and $\tilde{d} < d$, find	$f : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ and $g : \mathbb{R}^{\tilde{d}} \to \mathbb{R}^d$
so that for	$h_i = f(x_i) \in R^{\tilde{d}}$
the reconstruction error	$\sum_i \lVert x_i - g(h_i) \rVert_2^2$ is small

Neural network approach with single hidden layer

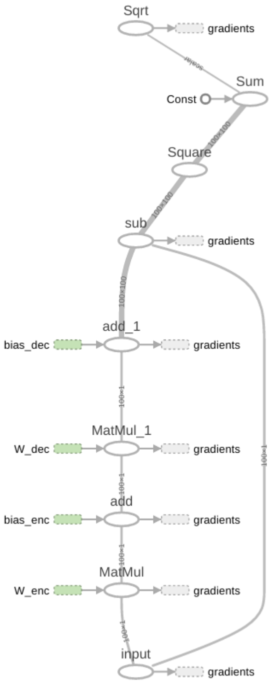


$$h = f(x) = \sigma\left(\mathbf{W}_{\text{enc}} x + b_{\text{enc}}\right) \qquad \text{with } \mathbf{W}_{\text{enc}} \in \mathbb{R}^{\tilde{d} \times d},\ b_{\text{enc}} \in \mathbb{R}^{\tilde{d}}$$

$$y = g(h) = \sigma\left(\mathbf{W}_{\text{dec}} h + b_{\text{dec}}\right) \qquad \text{with } \mathbf{W}_{\text{dec}} \in \mathbb{R}^{d \times \tilde{d}},\ b_{\text{dec}} \in \mathbb{R}^{d}$$

Example: Autoencoder in TF

```
d_data = 100
d_hidden = 30

# Construct Graph
x = tf.placeholder(tf.float32, [d_data,1], name="input")

# Hidden Layer Variables
W_enc = tf.Variable(tf.random_uniform([d_hidden,d_data], -1, 1), name="W_enc")
b_enc = tf.Variable(tf.zeros([d_hidden,1]), name = "bias_enc")

W_dec = tf.Variable(tf.random_uniform([d_data,d_hidden], -1, 1), name="W_dec")
b_dec = tf.Variable(tf.zeros([d_data]), name = "bias_dec")

# Hidden layer graph
h = (tf.matmul(W_enc, x) + b_enc)

# Output and reconstruction loss
y = (tf.matmul(W_dec, h) + b_dec)
loss = tf.sqrt(tf.reduce_sum(tf.square(x - y)))

# Optimizer
opt = tf.train.GradientDescentOptimizer(0.01)
update_step = opt.minimize(loss)
```



Sqrt
gradients
Sum
Const
100×100
Square
100×100
sub
gradients
100×100
add_1
bias_dec
gradients
100×1
MatMul_1
W_dec
gradients
100×1
add
bias_enc
gradients
100×1
MatMul
W_enc
gradients
100×1
input
gradients

Printing and Debugging

Printing

You cannot access a tensor's content e.g. `W[0][1]`
Only properties such as `W.get_shape()` or `W.type` are available.

- Option 1, one-time printing: Run the tensor in a session

```
W_data = session.run(W)   #get numpy.ndarray
print(W_data)
```

- Option 2, print continuously: `Print` node

```
W = tf.Variable(tf.zeros([100,100]),...)
W = tf.Print(W, [W], message="Entries of W: ")
X = tf.matmul(W, X)... # W is still a matrix
```

Debugging the graph

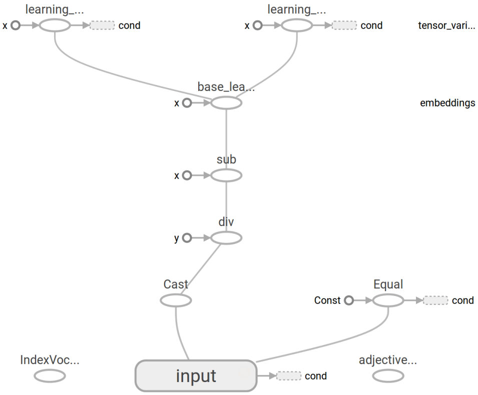
- In tensorboard manually check all dependencies in the graph

See also
`https://www.tensorflow.org/versions/r0.9/get_started/basic_usage.html`

Example: Autoencoder in TF

```
data = np.random.rand(d_data, n_datapoints + 1)

# Train
with tf.Session() as session:
# Initialize variables
init = tf.initialize_all_variables()
session.run(init)

# Do nsteps many SGD update steps
for i in range(10000):
  datapoint = data[:,np.random.randint(0, n_datapoints)]
  feed_dict = {x : np.transpose([datapoint])}
  session.run(update_step, feed_dict = feed_dict)

  # Every now and then test the lost on our hold out datapoint
  if i % 200 == 0:
    test_point = data[:, n_datapoints]
    feed_dict = {x : np.transpose([test_point])}
    test_loss = session.run(loss, feed_dict = feed_dict)
    print("test loss is %f " % test_loss)
```

Tensorboard



Tensorboard

How to add a summary

For example, track the norm of `W = tf.Variable(tf.zeros([100,100]),...)`

At graph construction time

Once globally:

- Create a summary writer
  `summary_writer = tf.train.SummaryWriter("/my/directory")`

For this particular summary:

- Get the norm: `W_norm = tf.sqrt(tf.sum_reduce(tf.square(W)))`
- Create summary: `W_summary = tf.scalar_summary("Norm of W", W_norm)`

At training time

- Merge all summaries: `summaries = tf.merge_summary([W_summary,...])`
- Fetch the data and turn into a formatted string:
  `summary_str = session.run(summaries)`
- Send the string to the writer
  `summary_writer.add_summary(summary_str, global_step)`

Resources

The *How to* and *Tutorials* section on tensorflow.org are actually good resources to recap concepts.
`https://www.tensorflow.org/versions/r0.9/get_started/index.html`
`https://www.tensorflow.org/versions/r0.9/tutorials/index.html`

For specific problems, google search often delivers quite useful threads on `http://stackoverflow.com` and `https://github.com/tensorflow/tensorflow/issues`