Tensorflow

1. TensorFlow works by first defining and describing our model in abstract, and then, when we are ready, we make it a reality in the session.

After normal coding, no actual calculations have been run, only operations created. Each operation, or "op," in our computation graph is a "node" in the graph.

To actually see the result, we need to run the session *sess.run()*.

When you are finished with a session, you need to close it in order to free up the resources that were used *sess.close()*.

TensorFlow operations are performed as the flow in the computational graph. Each operation is treated as a node. When running it, the chain rule is carried through the computation. Start from the current operations and track all its necessary information until the last linked operation with its graph.

1. *tf.placeholder* variable declaration declare the basic structure of the data need to be used, if we did not know this value of this variable. It tells the tensorflow what type, shape, will be “injected” into the corresponding variables.

x = tf.placeholder('float', [None, 784])

# len(x) = 784, input vec

y = tf.placeholder('float')

# y is the label, scalar

.

.

.

sess.run([optimizer, cost], feed\_dict={x: epoch\_x, y:epoch\_y})

# the declared data is passed by feed\_dict using a dictionary type, *i.e.*, x,y is the key. The input of *feed\_dict* to be supplied is a Python dictionary, with each key being the name of the *placeholder* that we are filling.

Each row vector of the input data in TensorFlow NN represents a point.

1. In the LSTM, the input is chopped into a sequence of batches {X1, X2, …, Xt, …, Xn}. During the multiple training batches {Xt, t>1} executed in each epoch, we load the final state variables from the previous training batch Xt into LSTM cells for the current training batch X\_{t+1}. The related gates (forget, input, candidate and output) will be applied to the state St passed from Xt.