NLP
Deep Learning

*Libraries for Deep Learning*
Libraries for Deep Learning

• Theano (Python): http://deeplearning.net/software/theano/
  – Lasagne: Built on top of Theano, has pre–designed networks: https://github.com/Lasagne/Lasagne

• Torch (Lua): http://torch.ch/

• TensorFlow (Python and C++): https://www.tensorflow.org/
import theano
import theano.tensor as T
Import numpy as np

# “symbolic” variables
x = T.matrix('x')
y = T.matrix('y')
dot = T.dot(x, y)
import theano
import theano.tensor as T
import numpy as np

# “symbolic” variables
x = T.matrix('x')
y = T.matrix('y')
dot = T.dot(x, y)

# this is the slow part
f = theano.function([x, y], [dot])

# now we can use this function
a = np.random.random((2, 3))
b = np.random.random((3, 4))
c = f(a, b) # now a 2 x 4 array
in = T.vector('in')
sigmoid = 1 / (1 + T.exp(-in))
#same as T.nnet.sigmoid

#shared means that it is not symbolic
w = theano.shared(np.random.randn(n))
b = theano.shared(0.)
Theano

\[ p_1 = \text{sigmoid}(T.\text{dot}(x, w) + b) \]

\[ \text{xent} = -y \ast T.\text{log}(p_1) - (1-y) \ast T.\text{log}(1-p_1) \] \# Cross-entropy

\[ \text{cost} = \text{xent}.\text{mean}() \] \# The cost to minimize

\[ gw, gb = T.\text{grad}(\text{cost}, [w, b]) \]
train = theano.function(
    inputs=[x,y],
    outputs=[prediction, xent],
    updates=((w, w - 0.1 * gw), (b, b - 0.1 * gb)))
LSTM Sentiment Analysis Demo

• If you’re new to deep learning and want to work with Theano, do yourself a favor and work through http://deeplearning.net/tutorial/

• Today’s LSTM demo is described here: http://deeplearning.net/tutorial/lstm.html

• Sentiment analysis model trained on IMDB movie reviews

[Slides from Catherine Finegan-Dollak]
How to Make It Work

• Download lstm.py and imdb.py to one folder.
• Create a “data” folder one directory above where you put those (or comment out line 66 of imdb.py and create data/in the same folder with lstm.py and imdb.py)
• Run the following command:
  ```python
  THEANO_FLAGS="floatX=float32" python lstm.py
  ```
• The code will download a dataset to the data folder and train a model on it.
Output Example

...  
Epoch  79 Update  9970 Cost  1.16459978017e-05  
Epoch  79 Update  9980 Cost  1.54052886501e-05  
Epoch  79 Update  9990 Cost  2.14830633922e-05  
Saving...  
Done  
('Train ', 0.0, 'Valid ', 0.16190476190476188, 'Test ', 0.20999999999999996)  
Early Stop!  
Seen 1840 samples  
Train  0.0015015015015 Valid  0.142857142857 Test  0.194  
The code run for 80 epochs, with 63.503504 sec/epochs  
Training took 5080.3s
LSTMs: One Time Step

\[ c_0 \]

\[ h_0 \]

\[ \sigma \]

\[ c_1 \]

\[ f_1 \]

\[ i_1 \]

\[ o_1 \]

\[ x_1 \]

\[ \tanh \]

\[ \sigma \]

\[ h_1 \]
LSTMs: Building a Sequence
Theano Implementation of an LSTM Step

(lstm.py, L. 174)

```python
def _step(m_, x_, h_, c_):
    preact = tensor.dot(h_, tparams[_p(prefix, 'U'))
    preact += x_

    i = tensor.nnet.sigmoid(_slice(preact, 0, options['dim_proj']))
    f = tensor.nnet.sigmoid(_slice(preact, 1, options['dim_proj']))
    o = tensor.nnet.sigmoid(_slice(preact, 2, options['dim_proj']))
    c = tensor.tanh(_slice(preact, 3, options['dim_proj']))

    c = f * c_ + i * c
    c = m_[..., None] * c + (1. - m_)[... , None] * c_

    h = o * tensor.tanh(c)
    h = m_[..., None] * h + (1. - m_)[... , None] * h_

    return h, c
```

“preact” is the sum of $Wx$ with the dot product of the previous step’s $h$ with the weight matrix $U$; $U$ concatenates $U_i$, $U_f$, $U_o$, and $U_c$, for computational efficiency; $W$ does the same with all the $W$ matrices. Then the _slice function splits the dot product back out again to generate the three gates, $i$, $f$, and $o$, and the candidate $C$.

$m_-$ is a mask, used for dealing with variable-length input.
theano.scan iterates through a series of steps

```python
rval, updates = theano.scan(_step,
    sequences=[mask, state_below],
    outputs_info=[tensor.alloc(numpy_floatX(0.), n_samples, dim_proj),
                  tensor.alloc(numpy_floatX(0.), n_samples, dim_proj)],
    name=_p(prefix, '_layers'),
    n_steps=nsteps)
```

(lstm.py, L. 195)
What if I want to ... ?

• Load different data: Look at how load_data in imdb.py works; you’ll probably want to model your code on that.
• Change my hyperparameters: Look at line 448 of lstm.py for an overwhelming selection of tunable hyperparameters.
• Learn how to implement other neural networks in Theano: Work through deeplearning.net/tutorial.
Links About Deep Learning

• Richard Socher’s Stanford class: http://cs224d.stanford.edu/
• Learn Theano + deep learning in one tutorial http://deeplearning.net/tutorial/
• http://web.eecs.umich.edu/~radev/dlnlp/list.txt
Is everything else we’ve learned this semester moot?

• Given how magical deep learning is, why do anything else?
• Deep learning requires a lot of data… seriously, a lot. Sometimes we don’t have enough data.
• Deep learning requires a lot of computing power.
• Benefits of combining power of deep learning with linguistic knowledge
NLP