

Sequence to Sequence Learning

In Natural Language Processing

Winston Carlile

wjc140030@utdallas.edu

May 13, 2017

Overview

1 TensorFlow

2 Word Embeddings

3 BPTT

4 LSTM

Installation

`https://www.tensorflow.org/install/`

Why?

Why TensorFlow?

- Expressiveness of Python
- Efficiency of optimized CUDA
- Scalability

Why?

Why TensorFlow?

- Expressiveness of Python
- Efficiency of optimized CUDA
- Scalability
- Free 😊

Why?

Why TensorFlow?

- Expressiveness of Python
- Efficiency of optimized CUDA
- Scalability
- Free 😊

<https://www.tensorflow.org/install/>

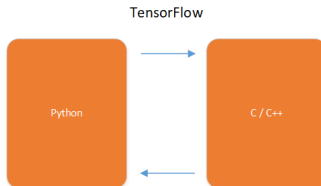
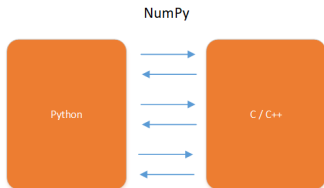
How it Works

NumPy

```
a = np.random.rand(100,100)
b = np.random.rand(100,100)
np.matmul(a,b)
```

TensorFlow

```
sess = tf.Session()
a = tf.Variable(tf.zeros([100,100]))
b = tf.Variable(tf.zeros([100,100]))
y = tf.matmul(a,b)
sess.run(tf.global_variables_initializer())
sess.run(y)
```



Neural Net Crash Course

- Optimize parameters of function to minimize error
- Search parameter space with Gradient Descent Algorithm

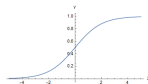
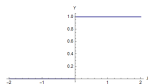
$$\theta_{t+1} := \theta_t - \gamma \frac{dl_n(\theta)}{d\theta} \Big|_{\theta_t}$$

Activation Function Petting Zoo

Element-wise nonlinearities

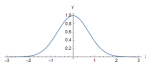
■ Sigmoid

$$\sigma(\phi) = \frac{1}{1 + e^{-\phi}}$$



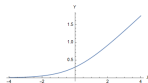
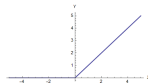
■ Gaussian Radial Basis

$$\psi(\phi) = \exp(-\phi \odot \phi)$$



■ Softplus

$$q(\phi) = \log(1 + e^{\phi})$$



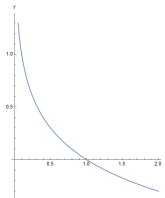
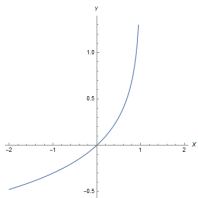
Loss Function Petting Zoo

■ Continuous Output (regression)

$$l_n(\theta) = \frac{1}{n} \sum_{i=1}^n |y_i - \ddot{y}(\theta, s_i)|^2$$

■ Binary Output (classification)

$$l_n(\theta) = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\ddot{y}(\theta, s_i)) + (1 - y_i) \log(1 - \ddot{y}(\theta, s_i))]$$



MNIST

Clone the Demo

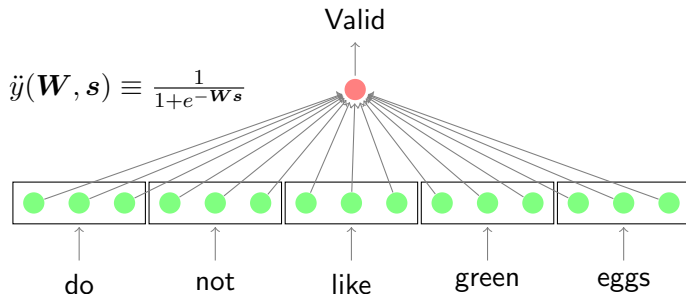
```
$ git clone https://github.com/WChurchill/seq2seq_workshop.git  
$ cd seq2seq_workshop/code  
$ python mnist.py
```

Or,

```
<path-to-tensorflow>/tensorflow/examples/tutorials/mnist/mnist.py
```

Word Embeddings

- Pretrain using unsupervised learning
- Train a small network to recognize valid n-grams



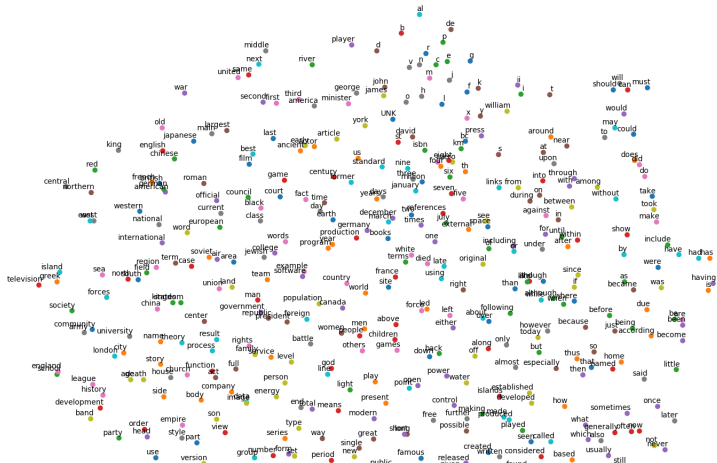
Code

Try it!

Run word2vec.py

```
$ python word2vec_basic.py  
$ ls  
tsne.png
```

tsne.png



But What About Sequences?

- Matrix dimensions are fixed
- Architecture is static
- How do we make networks with variable length inputs?
- How do we make networks with variable length outputs?

But What About Sequences?

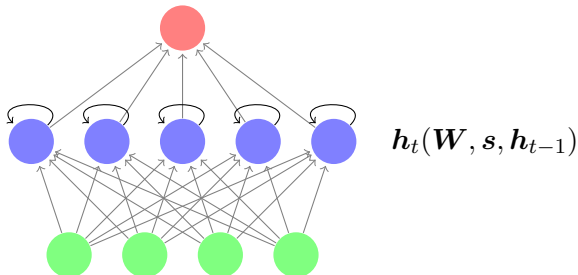
- Matrix dimensions are fixed
- Architecture is static
- How do we make networks with variable length inputs?
- How do we make networks with variable length outputs?

Bad Idea

Find maximum length of input & output vectors during preprocessing

Recurrent Neural Networks

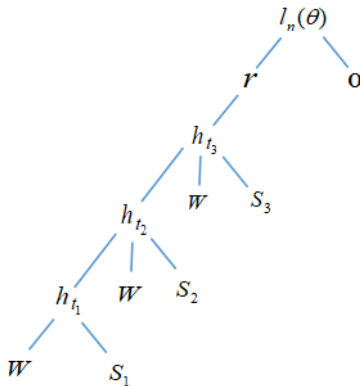
- Networks that feed into themselves
- Hidden units remember by self-activation



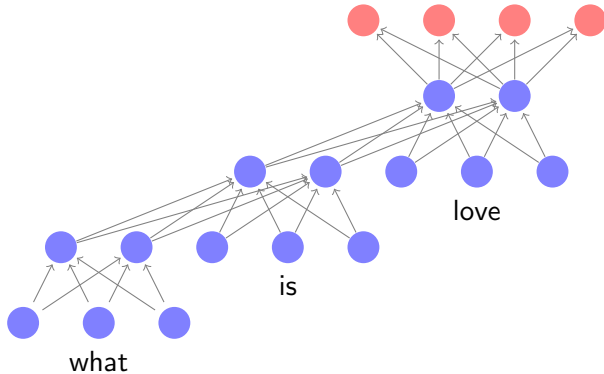
Backpropagation Through Time

$$D \equiv [S_1, S_2, \dots, S_n]$$

$$S_i \equiv [s_{i1}, s_{i2}, \dots, s_{iT_i}]$$

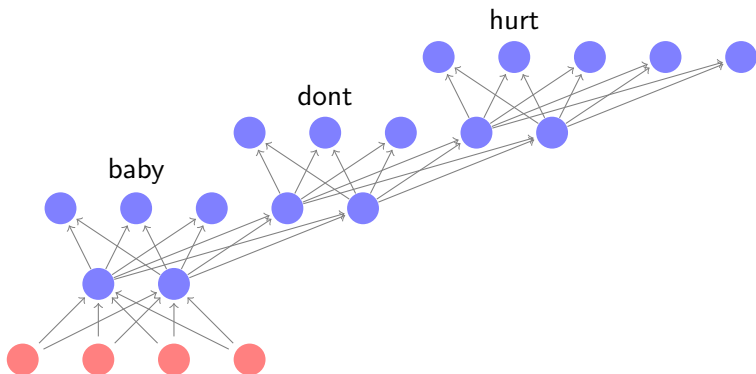


Expanded Network Diagram



Input sequence is encoded as a finite-length vector

Decoding



Output sequence is a repeated projection of the encoded vector

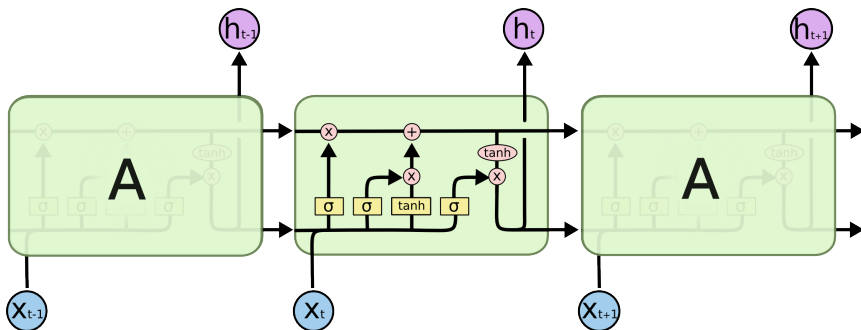
Vanishing Gradient

- Gradient is disproportionately small in bottom layers
- Output layers receive most of the blame
- Performs poorly on long sequences

Long Short-Term Memory

- Information highway through time
- Network learns explicitly what to remember

Anatomy of an LSTM Cell



<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Code

rnn_tutorial.py

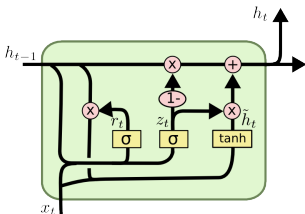
```
lstm = tf.contrib.rnn.BasicLSTMCell(<num_units>)  
state = tf.zeros([<batch size>, <lstm state size>])
```


Even More Improvements!

Some questions you might have:

- What's the difference between the cell state and hidden state?
- What's the difference between remembering new information and forgetting the old?
- Why are the gates blind to the cell state?

Gated Recurrent Units



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Code

Try It! (if you have 6 hours or a beefy GPU)

```
$ python translate.py  
downloading dataset... # very large file  
# wait 3 hours depending on internet quality  
# dataset is downloaded  
# begin preprocessing  
# 3 more hours
```

Code

Try It! (if you have 6 hours or a beefy GPU)

```
$ python translate.py  
downloading dataset... # very large file  
# wait 3 hours depending on internet quality  
# dataset is downloaded  
# begin preprocessing  
# 3 more hours
```

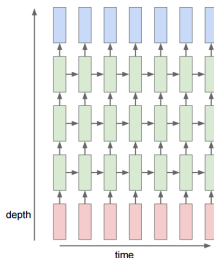
ERROR!

Your model is incorrect

Stacking LSTM Networks

rnn_tutorial.py

```
lstm = tf.contrib.rnn.BasicLSTMCell(<num_units>)  
stacked_lstm = tf.contrib.rnn.MultiRNNCell([lstm] * <num_layers>)
```



https://leonardoaraujasantos.gitbooks.io/artificial-intelligence/content/recurrent_neural_networks.html

Other Applications in NLP

- Speech recognition
- Conversational Agents
- Automated Summarization

What Now?

Amazon Web services

- Discount supercomputer
- Gigabit internet speed
- Harness the scalability of tensorflow

What Now?

Amazon Web services

- Discount supercomputer
- Gigabit internet speed
- Harness the scalability of tensorflow
- Additional setup

Questions and Discussion