Understanding LSTM Networks
Recurrent Neural Networks
An unrolled recurrent neural network
The Problem of Long-Term Dependencies
RNN short-term dependencies

Language model trying to predict the next word based on the previous ones

the clouds are in the sky,
RNN long-term dependencies

Language model trying to predict the next word based on the previous ones

I grew up in India… I speak fluent Hindi.
Standard RNN

A

\[ h_{t-1} \]

A

\[ h_t \]

A

\[ h_{t+1} \]

\[ X_{t-1} \]

\[ X_t \]

\[ X_{t+1} \]

tanh
Backpropagation Through Time (BPTT)
RNN forward pass

\[ s_t = \tanh(Ux_t + Ws_{t-1}) \]

\[ \hat{y}_t = \text{softmax}(Vs_t) \]

\[ E(y, \hat{y}) = -\sum_t E_t(y_t, \hat{y}_t) \]
Backpropagation Through Time

\[
\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}
\]

\[
\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial W}
\]

But \( s_3 = \tanh(Ux_t +Ws_2) \)

\( S_3 \) depends on \( s_2 \), which depends on \( W \) and \( s_1 \), and so on.

\[
\frac{\partial E_3}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}
\]
The Vanishing Gradient Problem

\[
\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}
\]

\[
\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \left( \prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}
\]

- Derivative of a vector w.r.t a vector is a matrix called jacobian
- 2-norm of the above Jacobian matrix has an upper bound of 1
- \textbf{tanh} maps all values into a range between -1 and 1, and the derivative is bounded by 1
- With multiple matrix multiplications, gradient values \textbf{shrink exponentially}
- Gradient contributions from “far away” steps become zero
- Depending on activation functions and network parameters, gradients could \textbf{explode} instead of vanishing
Activation function

Some Common Activation Functions

Activation Function Derivatives
Basic LSTM
Unrolling the LSTM through time
Constant error carousel

\[ s_t = \tanh(Ux_t + Ws_{t-1}) \]

\[ C_t = \tilde{C}_t \cdot i_c^{(t)} + C_{t-1} \]
Input gate

- Use contextual information to decide
- Store input into memory
- Protect memory from overwritten by other irrelevant inputs

\[
C_t = \tilde{C}_t \cdot i_c^{(t)} + C_{t-1}
\]
Output gate

- Use contextual information to decide
- Access information in memory
- Block irrelevant information

\[
C_t = \tilde{C}_t \cdot i_c^{(t)} + C_{t-1}
\]

\[
C_t \cdot O_t = O_t
\]
Forget or reset gate

\[ C_t = \tilde{C}_t \cdot i_c^{(t)} + C_{t-1} \cdot f_t \]

\[ C_t \cdot o_t \]

\[ \sigma \]

\[ f_t \]

\[ \Pi \]

\[ o_t \]

\[ \Pi \]

\[ \Pi \]

\[ \Pi \]

\[ \tilde{C}_t \]

\[ i_t \]

Edge to next time step

Edge from previous time step (and current input)

Weight fixed at 1
LSTM with four interacting layers
The cell state
Gates

sigmoid layer
Step-by-Step LSTM Walk Through
Forget gate layer

\[ f_t = \sigma \left( W_f \cdot \left[ h_{t-1}, x_t \right] + b_f \right) \]
Input gate layer

\[ i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \]

\[ \tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \]
The current state

\[ C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \]
Output layer

\[ o_t = \sigma (W_o [ h_{t-1}, x_t ] + b_o) \]
\[ h_t = o_t \times \text{tanh} (C_t) \]
Reference

- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- http://www.wildml.com/
- http://deeplearning.net/tutorial/lstm.html
- https://theclevermachine.files.wordpress.com/2014/09/act-funs.png
- A Critical Review of Recurrent Neural Networks for Sequence Learning, Zachary C. Lipton, John Berkowitz
- Long Short-Term Memory, Hochreiter, Sepp and Schmidhuber, Jurgen, 1997