Notes on Sequence to Sequence Learning with Neural Networks

David Meyer

dmm@brocade.com

October 23, 2015

Abstract

1 Introduction

The purpose of this document is to (i). make a few notes on [2] and (ii). fill in some of the gaps in the math that underlies the paper.

2 Logistic Regression

Since Sequence to Sequence Learning [2] uses two LSTMs, one for the input sequence and one for the output sequence which follow the formulation given in Graves [1]. Figure 1 shows the structure of the prediction architecture used there. More detail on the architecture of LSTM cells is shown in Figure 2 (note that this design uses the so-called "peephole connections").

Graves [1] models sequences as a multinomial distribution which can be naturally parameterized by a softmax function at the output layer. That is, if there are K text classes in total and class k is read at time t, then x_t is a K length vector which is one-hot encoded (all of its entries are zero except for the k^{th} entry which is one). Hence $Pr(x_{t+1} = k \mid y_t)$ is a multinomial distribution:

$$Pr(x_{t+1} = k \mid y_t) = y_t^k = \frac{e^{\hat{y}_t^k}}{\sum\limits_{k'=1}^{K} e^{\hat{y}_{k'}^k}} \tag{1}$$

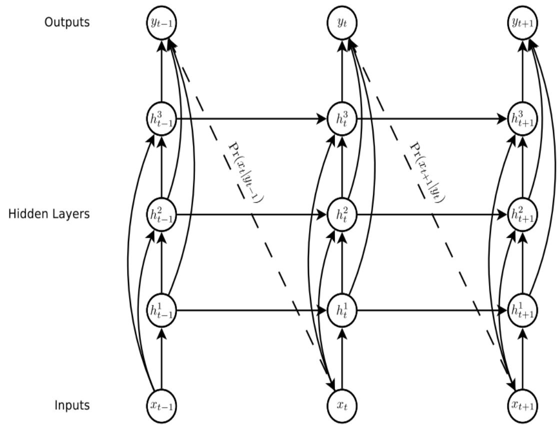

Figure 1: Deep RNN Prediction Architecture from Graves [1]

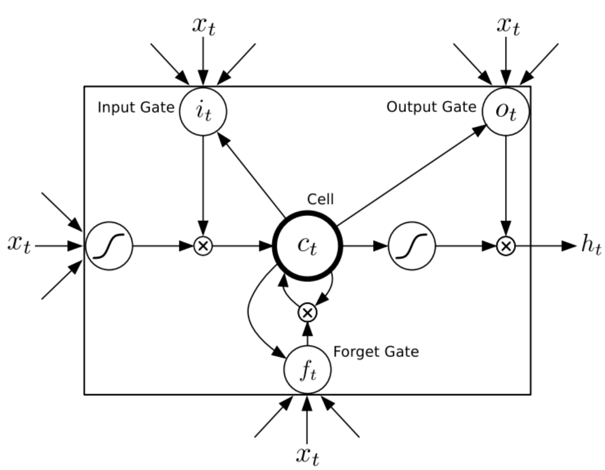

Figure 2: Long Short-Term Memory Cell

2.1 LSTM Activations

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) && (2)\\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) && (3)\\
c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) && (4)\\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) && (5)\\
h_t &= o_t \tanh(c_t) && (6)
\end{aligned}
$$

One way to understand the behavior of the multinomial distribution and its loss function $\mathcal{L}(x)$ is to see how we compute the required gradients for logistic regression. Recall the that *logistic function* $\sigma(x)$ is defined as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

We are actually interested in the derivative of $\sigma(x)$ so that we can compute gradients for back propagation. Looking forward, note that the *softmax* classifier, which is used in [2], is defined as follows:

$$p_j = \frac{e^{o_j}}{\sum\limits_{k=1}^{K} e^{o_k}} \tag{8}$$

When $i = j$ the derivative of softmax is similar to the derivative of the logistic function, which why its useful to look at the logistic function.

2.2 Derivative of the Logistic Function

Recall the *chain rule* for derivatives is

$$\frac{d}{dx}\Big[f(g(x)\Big] = f'(g(x))g'(x))] \tag{9}$$

Now, let the logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$. Then for purposes of the chain rule we define

$$g(x) = 1 + e^{-x}$$
$$f(x) = \frac{1}{x}$$

so that $\sigma(x) = f(g(x)) = \frac{1}{1+e^{-x}}$. Taking the derivatives of g and f we get

$$g'(x) = e^{-x} \tag{10}$$
$$f'(g(x)) = (1 + e^{-x})^{-2} \tag{11}$$

Given these definitions, we have:

$$\frac{d\sigma(x)}{dx} = \frac{d}{dx}\Big[f(g(x))\Big] = f'(g(x))g'(x)$$
$$= \frac{e^{-x}}{(1+e^{-x})^2}$$
$$= \frac{1+e^{-z}-1}{(1+e^{-z})^2}$$
$$= \frac{1+e^{-z}}{(1+e^{-z})^2} - \left(\frac{1}{1+e^{-z}}\right)^2$$
$$= \frac{1}{(1+e^{-z})} - \left(\frac{1}{1+e^{-z}}\right)^2$$
$$= \frac{1}{(1+e^{-z})}\left(1 - \frac{1}{(1+e^{-z})}\right)$$
$$= \sigma(x)(1-\sigma(x))$$

This result is sometimes written as $\frac{dy_i}{dz_i} = y_i(1 - y_i)$.

3 Softmax

The following describes the derivative of the softmax function. Recall that the softmax function is defined as follows:

$$p_j = \frac{e^{o_j}}{\sum\limits_{k=1}^{K} e^{o_k}}$$

When $i = j$, the softmax function derivative is similar to the derivative of the logistic function, namely:

$$\frac{\partial y_i}{\partial z_i} = y_i(1 - y_i) \tag{12}$$

Since [2] follows the notation and approach outlined in [1], it is useful to understand the derivation of the derivative of the loss function $\mathcal{L}(x)$ defined there.

$$\mathcal{L}(x) = -\sum_{t=1}^{T} \log y_t^{x_t+1} \tag{13}$$

For the purposes of the derivation, we'll use

$$\mathcal{L}(x) = -\sum_{j} t_j \log y_i \tag{14}$$

Applying the chain rule gives us

$$\frac{\partial \mathcal{L}(x)}{\partial z_i} = -\frac{\partial \mathcal{L}(x)}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_i} \tag{15}$$

Then the gradient of the loss function $\frac{\partial \mathcal{L}(x)}{\partial z_i}$ can be derived as follows:

$$\begin{aligned}
\frac{\partial \mathcal{L}(x)}{\partial z_i} &= -\frac{\partial \mathcal{L}}{\partial y_i} \left[\sum_j t_j \cdot \log y_i \right] (y_i \cdot (1 - y_i)) \\
&= -t_j \frac{\partial \mathcal{L}}{\partial y_i} \log y_i \cdot (y_i \cdot (1 - y_i)) \\
&= -t_j \frac{1}{y_i} \cdot (y_i \cdot (1 - y_i)) \\
&= -(t_j \cdot (1 - y_i)) \\
&= y_i - t_j
\end{aligned}$$

Note that $\frac{\partial y_i}{\partial z_i}$ is replaced with $y_i(1 - y_i)$ in the above (Section 2.2). Graves [1] uses an analogous derivation to get the following result:

$$\begin{aligned} \mathcal{L}(x) &= -\sum_{t=1}^{T} \log y_t^{x_t+1} \\ \implies \frac{\partial \mathcal{L}(x)}{\partial \hat{y}_t^k} &= y_t^k - \delta_{k,x_{t+1}} \end{aligned} \tag{16}$$

4 Acknowledgements

References

[1] Alex Graves. Generating sequences with recurrent neural networks. 08 2013.

[2] Ilya Sutskever, Oriol Vinyals, and Quoc V. V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.