Gradient backpropagation through a long short-term memory (LSTM) cell

Mostafa Sadeghi

Abstract

In this brief document, I am going to derive the equations for gradient backpropagation through an LSTM cell. Although with the presence of many deep learning softwares there is no longer any need to compute gradients by hand, this would be a nice exercise on the backpropagation method. Note that this document is by no means a tutorial on LSTM; instead, it is only a guide on how to do backpropagation in LSTMs. So, it assumes the reader has a prior knowledge on RNN and LSTM.

Index Terms

LSTM, RNN, backpropagation, gradient computation, computational graphs

I. Introduction

Long short-term memory (LSTM) networks have been proven to be much more effective than vanilla recurrent neural network (RNN) networks in training over long sequences. This is due to the fact that in contrast to RNNs, in LSTMs, there is a direct flow of the so-called state variables all across the time axis that prevents gradient vanishing or explosion. However, gradient computations for an LSTM cell are more involved than vanilla RNNs.

A. *LSTM: forward pass*

The overall structure of an LSTM cell is shown in Fig. 1. "[1]Similar to the vanilla RNN, at each timestep we receive an input $\mathbf{x}_t \in \mathbb{R}^D$ and the previous hidden state $\mathbf{h}_{t-1} \in \mathbb{R}^H$; the LSTM also maintains an H-dimensional *cell state*, so we also receive the previous cell state $\mathbf{c}_{t-1} \in \mathbb{R}^H$. The learnable parameters

Electrical Engineering Department, Sharif University of Technology, Tehran, Iran (e-mail: m.saadeghii@gmail.com.)

[1]http://cs231n.github.io/

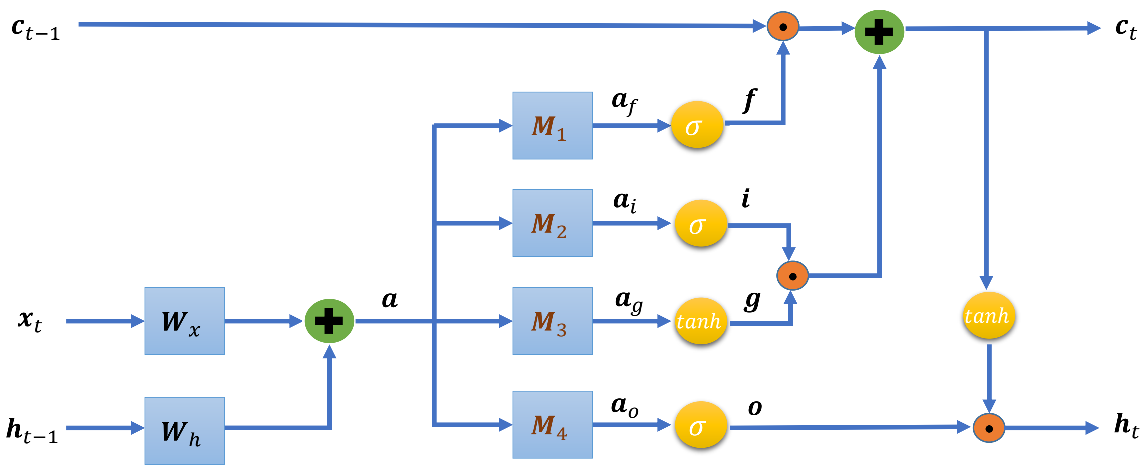

Fig. 1: An LSTM cell. Square blocks denote matrix-vector multiplication, yellow and brown blocks denote elementwise operations, and the green blocks denote summation. Note that the matrices $\mathbf{M}_1, \ldots, \mathbf{M}_4$ are fixed and so, they are not trained. The bias vector has not been shown in this figure, because its gradient computation is very easy. For the definition of each variable, see the text.

of the LSTM are an *input-to-hidden* matrix $\mathbf{W}_x \in \mathbb{R}^{4H \times D}$, a *hidden-to-hidden* matrix $\mathbf{W}_h \in \mathbb{R}^{4H \times H}$ and a *bias vector* $\mathbf{b} \in \mathbb{R}^{4H}$.

At each timestep we first compute an *activation vector* $\mathbf{a} \in \mathbb{R}^{4H}$ as $\mathbf{a} = \mathbf{W}_x \mathbf{x}_t + \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b}$. We then divide this into four vectors $\mathbf{a}_i, \mathbf{a}_f, \mathbf{a}_o, \mathbf{a}_g \in \mathbb{R}^H$ where $\mathbf{a}_i$ consists of the first H elements of $\mathbf{a}$, $\mathbf{a}_f$ is the next H elements of $\mathbf{a}$, etc. We then compute the *input gate* $\mathbf{i} \in \mathbb{R}^H$, *forget gate* $\mathbf{f} \in \mathbb{R}^H$, *output gate* $\mathbf{o} \in \mathbb{R}^H$ and *block input* $\mathbf{g} \in \mathbb{R}^H$ as

$$\mathbf{i} = \sigma(\mathbf{a}_i) \qquad \mathbf{f} = \sigma(\mathbf{a}_f) \qquad \mathbf{o} = \sigma(\mathbf{a}_o) \qquad \mathbf{g} = \tanh(\mathbf{a}_g)$$

where σ is the sigmoid function and $\tanh$ is the hyperbolic tangent, both applied elementwise. Finally we compute the next cell state $\mathbf{c}_t$ and next hidden state $\mathbf{h}_t$ as

$$\mathbf{c}_t = f \odot \mathbf{c}_{t-1} + \mathbf{i} \odot \mathbf{g} \qquad\qquad \mathbf{h}_t = \mathbf{o} \odot \tanh(\mathbf{c}_t)$$

where $\odot$ is the elementwise product of vectors.”

B. LSTM: backward pass

To make the computations easier, we define the following matrices: $\mathbf{M}_1, \dots, \mathbf{M}_4$, where $\mathbf{M}_i$ is the i-th $H \times 4H$ submatrix of the $4H \times 4H$ identity matrix. For instance,

$$\mathbf{M}_1 = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 & 0 & \dots & 0 \end{bmatrix}_{H \times 4H}$$

It is then straightforward to verify the following

$$\mathbf{a}_i = \mathbf{M}_1 \cdot \mathbf{a} \qquad \mathbf{a}_f = \mathbf{M}_2 \cdot \mathbf{a} \qquad \mathbf{a}_o = \mathbf{M}_3 \cdot \mathbf{a} \qquad \mathbf{a}_g = \mathbf{M}_4 \cdot \mathbf{a}$$

Note that introducing these matrices is mainly for simplicity of the derivations, and it's not an efficient way for practical implementations. To derive the gradient expressions, we use graph computations by breaking the overall computations in an LSTM cell into simple operations. To this end, let's define the following additional intermediate variables:

$$\mathbf{i}_g = \mathbf{i} \odot \mathbf{g} \qquad \mathbf{c}'_t = \tanh(\mathbf{c}_t) \qquad \mathbf{f}_c = \mathbf{f} \odot \mathbf{c}_{t-1}$$

With these definitions, the cell outputs can be written as

$$\mathbf{c}_t = \mathbf{f}_c + \mathbf{i}_g \qquad \mathbf{h}_t = \mathbf{o} \odot \mathbf{c}'_t$$

Now, we are ready for gradient computations! Let's go ahead! Given the output gradients denoted $\mathrm{d}\mathbf{h}_t$ and $\mathrm{d}\mathbf{c}_t$ coming from the next LSTM cell, we want to compute the gradients of all the variables. In other words, we want to backpropagate the output gradients from the cell outputs all the way to the cell inputs. To do this, we can write (see also Fig. 2)

- $\mathrm{d}\mathbf{o} = \mathrm{d}\mathbf{h}_t \odot \mathbf{c}'_t$ (Because $\mathbf{h}_t = \mathbf{o} \odot \mathbf{c}'_t$)
- $\mathrm{d}\mathbf{c}'_t = \mathrm{d}\mathbf{h}_t \odot \mathbf{o}$ (Because $\mathbf{h}_t = \mathbf{o} \odot \mathbf{c}'_t$)
- $\mathrm{d}\mathbf{i}_g = \mathrm{d}\mathbf{c}_t + \dot{\tanh}(\mathbf{c}_t) \odot \mathrm{d}\mathbf{c}'_t$ (Gradient comes from two sources to the top summation node; see Fig. 2)
- $\mathrm{d}\mathbf{f}_c = \mathrm{d}\mathbf{c}_t + \dot{\tanh}(\mathbf{c}_t) \odot \mathrm{d}\mathbf{c}'_t = \mathrm{d}\mathbf{c}_t + (1 - \tanh^2(\mathbf{c}_t)) \odot \mathrm{d}\mathbf{c}'_t = \mathrm{d}\mathbf{c}_t + (1 - {\mathbf{c}'_t}^2) \odot \mathrm{d}\mathbf{c}'_t$
- $\mathrm{d}\mathbf{i} = \mathrm{d}\mathbf{i}_g \odot \mathbf{g}$ (Because $\mathbf{i}_g = \mathbf{i} \odot \mathbf{g}$)
- $\mathrm{d}\mathbf{g} = \mathrm{d}\mathbf{i}_g \odot \mathbf{i}$
- $\mathrm{d}\mathbf{f} = \mathrm{d}\mathbf{f}_c \odot \mathbf{c}_{t-1}$ (Because $\mathbf{f}_c = \mathbf{f} \odot \mathbf{c}_{t-1}$)
- $\mathrm{d}\mathbf{a}_i = \dot{\sigma}(\mathbf{a}_i) \odot \mathrm{d}\mathbf{i} = \sigma(\mathbf{a}_i) \odot (1 - \sigma(\mathbf{a}_i)) \odot \mathrm{d}\mathbf{i} = \mathbf{i} \odot (1 - \mathbf{i}) \odot \mathrm{d}\mathbf{i}$ (Because $\mathbf{i} = \sigma(\mathbf{a}_i)$ and $\dot{\sigma} = \sigma(1 - \sigma)$)
- $\mathrm{d}\mathbf{a}_f = \dot{\sigma}(\mathbf{a}_f) \odot \mathrm{d}\mathbf{f} = \mathbf{f} \odot (1 - \mathbf{f}) \odot \mathrm{d}\mathbf{f}$

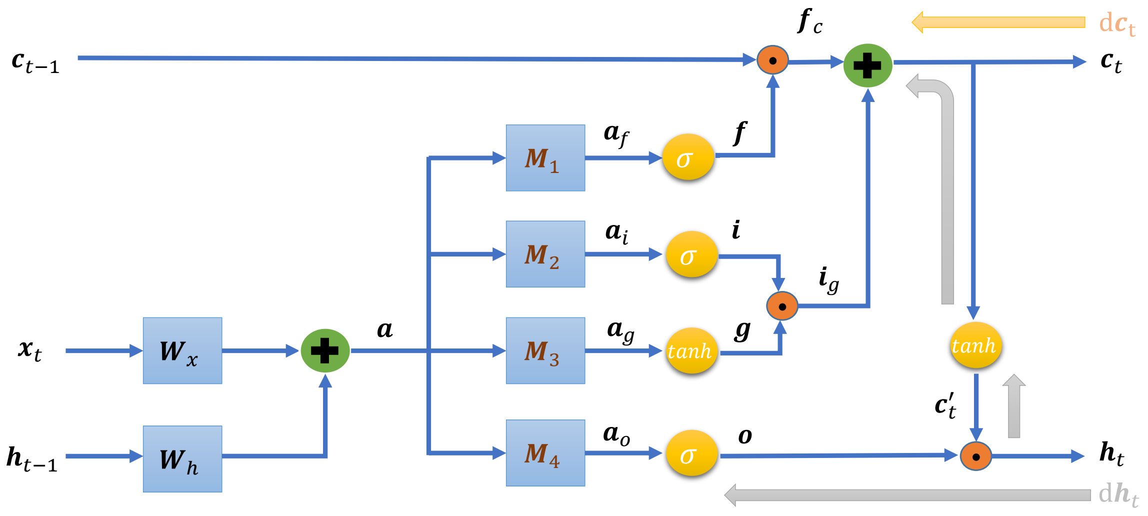

Fig. 2: The same as Fig. 1 but with illustration of gradient flows directions as well as indicating some intermediate variables.

- $\mathrm{d}\mathbf{a}_o = \dot{\sigma}(\mathbf{a}_o) \odot \mathrm{d}\mathbf{o} = \mathbf{o} \odot (1 - \mathbf{o}) \odot \mathrm{d}\mathbf{o}$
- $\mathrm{d}\mathbf{a}_g = \dot{\tanh}(\mathbf{a}_g) \odot \mathrm{d}\mathbf{g} = (1 - \mathbf{g}^2) \odot \mathrm{d}\mathbf{g}$
- $\mathrm{d}\mathbf{a} = \mathbf{M}_1^T \cdot \mathrm{d}\mathbf{a}_i + \mathbf{M}_2^T \cdot \mathrm{d}\mathbf{a}_f + \mathbf{M}_3^T \cdot \mathrm{d}\mathbf{a}_o + \mathbf{M}_4^T \cdot \mathrm{d}\mathbf{a}_g$ (Because the gradient backpropagates through four paths to reach $\mathbf{a}$; see Fig. 2)
- $\mathrm{d}\mathbf{x}_t = \mathbf{W}_x^T \cdot \mathrm{d}\mathbf{a}$
- $\mathrm{d}\mathbf{h}_{t-1} = \mathbf{W}_h^T \cdot \mathrm{d}\mathbf{a}$
- $\mathrm{d}\mathbf{W}_x = \mathrm{d}\mathbf{a} \cdot \mathbf{x}_t^T$
- $\mathrm{d}\mathbf{W}_h = \mathrm{d}\mathbf{a} \cdot \mathbf{h}_{t-1}^T$
- $\mathrm{d}\mathbf{b} = \mathrm{d}\mathbf{a}$ (Bias gradient)

In the above, the power of 2 is applied elementwise. Note that the effect of $\mathbf{M}_1, \ldots, \mathbf{M}_4$ is to put $\mathrm{d}\mathbf{a}_i, \ldots, \mathrm{d}\mathbf{a}_g$ in their corresponding positions in a $4H$-long vector to create $\mathrm{d}\mathbf{a}$. That is, they just concatenate $\mathrm{d}\mathbf{a}_i, \ldots, \mathrm{d}\mathbf{a}_g$ on top of each other.