1 LSTMs

These notes are based on [1]. LSTMs were developed in order to circumvent the vanishing gradient problem that plagues multi-layered RNNs. LSTMs are equipped with a long-term memory and a short-term working memory.

Let $x_t \in \boldsymbol{R}^p$ denote the input at time t; let $\mathrm{ltm}_t \in \boldsymbol{R}^d$ and $\mathrm{wm}_t \in \boldsymbol{R}^d$ denote, respectively, the long-term memory and the working memory available to the LSTM cell at time t.

Updating the long-term memory. In order to update the long-term memory at step t, the LSTM first figures out what to remember from the long-term memory of the last step $t-1$.

$$\mathrm{rem}_t = \sigma\left(W_r \cdot x_t + U_r \cdot \mathrm{wm}_{t-1} + b_r\right). \tag{1}$$

This is accomplished using a one-layer neural network with a sigmoid activation function that estimates the weight matrices $W_r \in \boldsymbol{R}^{p\times d}$, $U_r \in \boldsymbol{R}^{d\times d}$ and the bias vector $b_r \in \boldsymbol{R}^d$. Since the activation function is sigmoid, the components of rem_t are between 0 and 1. If a component is closer to 1, we would want to remember it; if it is close to 0, then we want to forget it.

It next calculates a "candidate" vector to add to its long-term memory. This is done using a single-layer neural network with a tanh activation function. Denote this candidate by ltm'_t.

$$\mathrm{ltm}'_t = \tanh\left(W_l \cdot x_t + U_l \cdot \mathrm{wm}_{t-1} + b_l\right). \tag{2}$$

As usual, $W_l \in \boldsymbol{R}^{p\times d}$, $U_l \in \boldsymbol{R}^{d\times d}$ and $b_l \in \boldsymbol{R}^d$.

Not all parts of this candidate vector may be worth remembering. As such, a save_t vector is created using another single-layer neural network with a sigmoid activation function.

$$\mathrm{save}_t = \sigma\left(W_s \cdot x_t + U_s \cdot \mathrm{wm}_{t-1} + b_s\right). \tag{3}$$

The dimensions of the weight matrices W_s, U_s and the bias vector b_s are such that $\mathrm{save}_t \in \boldsymbol{R}^d$. Now the long-term component of the cell is computed using:

$$\mathrm{ltm}_t = \mathrm{rem}_t \odot \mathrm{ltm}_{t-1} + \mathrm{save}_t \odot \mathrm{ltm}'_t, \tag{4}$$

where $\odot$ represents component-wise multiplication of the d-dimensional vectors.

Updating the working memory. To do this, the LSTM first calculates what parts of the long-term memory it currently wants to focus on. It uses another single-layer neural network with a sigmoid activation to calculate $\mathrm{focus}_t \in \boldsymbol{R}^d$.

$$\mathrm{focus}_t = \sigma\left(W_f \cdot x_t + U_f \cdot \mathrm{wm}_{t-1} + b_f\right). \tag{5}$$

It then updates its working memory using:

$$\mathrm{wm}_t = \mathrm{focus}_t \odot \tanh(\mathrm{ltm}_t). \tag{6}$$

References

[1] Edwin Chen. Blog post at `http://blog.echen.me/2017/05/30/exploring-lstms/`.

$$
\begin{aligned}
\mathrm{rem}_t &= \sigma\left(W_r \cdot x_t + U_r \cdot \mathrm{wm}_{t-1} + b_r\right) \\
\mathrm{save}_t &= \sigma\left(W_s \cdot x_t + U_s \cdot \mathrm{wm}_{t-1} + b_s\right) \\
\mathrm{focus}_t &= \sigma\left(W_f \cdot x_t + U_f \cdot \mathrm{wm}_{t-1} + b_f\right) \\
\mathrm{ltm}'_t &= \tanh\left(W_l \cdot x_t + U_l \cdot \mathrm{wm}_{t-1} + b_l\right) \\
\mathrm{ltm}_t &= \mathrm{rem}_t \odot \mathrm{ltm}_{t-1} + \mathrm{save}_t \odot \mathrm{ltm}'_t \\
\mathrm{wm}_t &= \mathrm{focus}_t \odot \tanh(\mathrm{ltm}_t).
\end{aligned}
$$

Figure 1: All the LSTM equations at once.