

1 LSTMs

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 $x_t \in \mathbf{R}^p$ denote the input at time t ; let $\text{ltm}_t \in \mathbf{R}^d$ and $\text{wm}_t \in \mathbf{R}^d$ 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 $t - 1$.

$$\text{rem}_t = \sigma(W_r \cdot x_t + U_r \cdot \text{wm}_{t-1} + b_r). \quad (1)$$

This is accomplished using a one-layer neural network with a sigmoid activation function that estimates the weight matrices $W_r \in \mathbf{R}^{p \times d}$, $U_r \in \mathbf{R}^{d \times d}$ and the bias vector $b_r \in \mathbf{R}^d$. 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 .

$$\text{ltm}'_t = \sigma(W_l \cdot x_t + U_l \cdot \text{wm}_{t-1} + b_l). \quad (2)$$

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

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.

$$\text{save}_t = \sigma(W_s \cdot x_t + U_s \cdot \text{wm}_{t-1} + b_s). \quad (3)$$

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

$$\text{ltm}_t = \text{rem}_t \odot \text{ltm}_{t-1} + \text{save}_t \odot \text{ltm}'_t, \quad (4)$$

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 $\text{focus}_t \in \mathbf{R}^d$.

$$\text{focus}_t = \sigma(W_f \cdot x_t + U_f \cdot \text{wm}_{t-1} + b_f). \quad (5)$$

It then updates its working memory using:

$$\text{wm}_t = \text{focus}_t \odot \tanh(\text{ltm}_t). \quad (6)$$