LSTM: A Search Space Odysseys

Mușat Bogdan-Adrian

February 2017

Recurrent Neural Networks - RNNs

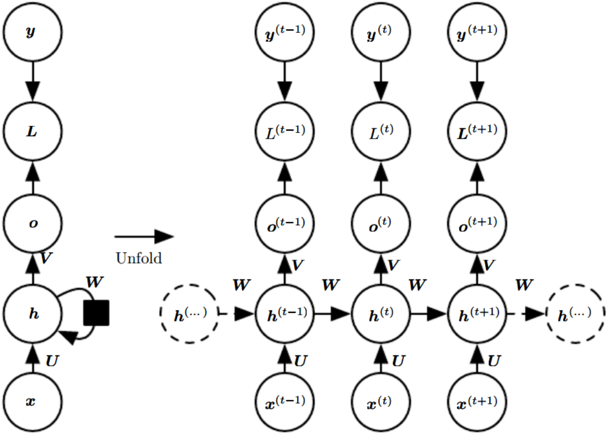
- Used to deal with sequential data, where there is a temporal dependence from a time instance to another
- Mathematically, they model a conditional distribution of the form $P(x_t|x_{t-1}, ..., x_2, x_1)$, where x_t is the current input at time t
- The output of a vanilla RNN cell at each time step is computed using the current input x_t and also the previous cell state h_{t-1}[1] as:

$$y_t = tanh(Ux_t + Wh_{t-1} + b),$$

where $U \in R^{M \times N}$, $W \in R^{N \times N}$ are the shared parameter matrices for the RNN cells and $b \in R^N$ is the bias

[1] Meant to encompass a summary of the past information

Recurrent Neural Networks Unrolled



(a) RNN unrolled[2]

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

Vanishing/Exploding Gradient

- The main algorithm for learning the weights of an RNN is called Backpropagation Through Time (BPTT)
- When computing the gradients with respect to the weights of the network, W depends on a recurrent connection from a previous timestep and so the partial derivative w.r.t. W becomes:

$$\frac{\partial L^{(t)}}{\partial W} = \sum_{k=0}^{t} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \left(\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} \right) \frac{\partial h^{(k)}}{\partial W},$$

  where $L^{(t)}$ is the error and $o^{(t)}$ is the predicted value at the t^{th} output step
- If the number of timesteps is big, the product from the formula above will either diminish to 0 or explode to ∞, thus prohibiting the learning process further on

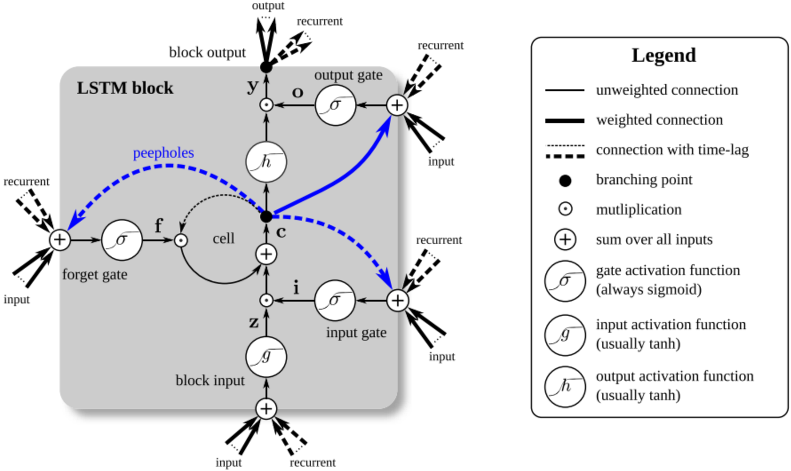
Long Short-Term Memory - LSTM

- LSTM[3] solves the problem of vanishing/exploding gradients
- The LSTM architecture is a memory cell, which can maintain its state over time, and nonlinear gating units, which regulate the information flow into and out of the cell
- Further experiments have proven that no other variants that differ from the vanilla LSTM by adding, removing or modifying exactly one aspect can obtain a much better performance[4]

[3] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: 9 (Dec. 1997), pp. 1735–80.

[4] Klaus Greff et al. "LSTM: A Search Space Odyssey". In: *CoRR* abs/1503.04069 (2015). arXiv: 1503.04069. URL: http://arxiv.org/abs/1503.04069.

LSTM Cell



(b) LSTM cell

LSTM Cell Computation

- The formulas for a vanilla LSTM layer forward pass can be written as:

$$
\begin{aligned}
\bar{z}^t &= W_z x^t + R_z y^{t-1} + b_z && \\
z^t &= g(\bar{z}^t) && \text{block input} \\
\bar{i}^t &= W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i && \\
i^t &= \sigma(\bar{i}^t) && \text{input gate} \\
\bar{f}^t &= W_f x^t + R_f y^{t-1} + p_f \odot c^{t-1} + b_f && \\
f^t &= \sigma(\bar{f}^t) && \text{forget gate} \\
c^t &= z^t \odot i^t + c^{t-1} \odot f^t && \text{cell} \\
\bar{o}^t &= W_o x^t + R_o y^{t-1} + p_o \odot c^{t-1} + b_o && \\
o^t &= \sigma(\bar{o}^t) && \text{output gate} \\
y^t &= h(c^t) \odot o^t && \text{block output}
\end{aligned}
$$

LSTM Gates

- Each gate is computed in terms of the current input, the previous output, a peephole connection to the previous cell state and a bias
- i^t, f^t and o^t are called the input, forget and output gates and their role is to determine how much information flows from previous and current timesteps. This effect is achieved by squashing the gates through a sigmoid function. Then an element wise multiplication is performed such that the input gate determines how much of the current information is used, the forget gate deals with the information flow from the previous cell state and finally, the output gate specifies how much information to send to the output y
- The peephole connections were added such that the network can learn precise timings easier