DSCI 572: Supervised Learning II

Muhammad Abdul-Mageed

muhammad.mageed@ubc.ca

Deep Learning & NLP Lab

The University of British Columbia

Table of Contents

Vanishing and Exploding Gradients

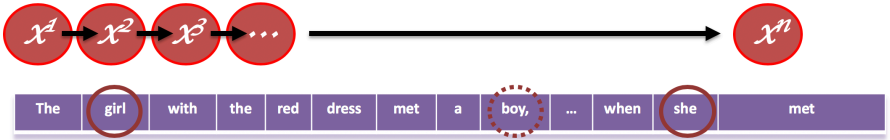

Figure: A very long sequence, modelled with an RNN. Gradients can vanish and we will not know which gender a pronoun should be (male or female, e.g., in an MT task), or to which entity the pronoun refers (boy or girl)

Gradient Problems

- Gradients can **explode**, in which case we can clip them.
- Gadients can also **vanish**, which is a more serious problem.

Solving Long-Term Dependencies

Solutions For Gradient Problems

- Long-Short Term Memory (**LSTM**) networks introduced to solve the problem of long-term dependencies
- Some notation modified from Andrew Ng, for pedagogical simplicity

Introducing a Memory Cell

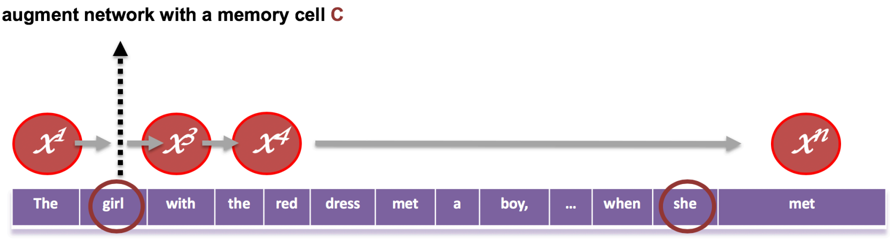

Figure: We will augment the network with a memory cell

Gradient Problems

- The **memory cell** will help us retain information over long sequences
- For example, we can still know **we need pronoun she**, maintaining the **female gender** (and retaining the correct **reference** to "the girl")

LSTM Gates

Notation

- Γ_u: Update state (sometimes called ***input gate*** f_i)
- Γ_r: Relevance state *(reset gate)*
- $\tilde{C}_t$: New candidate memory cell state
- C_t: Final LSTM memory cell
- a_t: LSTM hidden state (final) (you may see it elsewhere as h_t)

LSTM: Update and Forget Gates

Notes

- We will use **two gates** to control cell content: Γ_u **(update gate)**, sometimes called ***input gate*** f_i, and Γ_f **(forget gate)**
- **Forget gate** will give the new memory cell C_t the option to keep or forget the **old cell** (C_{t-1}), but just add to it via **update gate** (Γ_u)

1: LSTM

$$\Gamma_u = \sigma(W_u.[a_{t-1}, x_t])$$

$$\Gamma_f = \sigma(W_f.[a_{t-1}, x_t])$$

$$C_t = \Gamma_u * \tilde{C}_t + \Gamma_f * C_{t-1}$$

Output Gate

- We will use an **output gate (Γ_o)**
- **Output gate** will enable us to update our a_t via element-wise multiplication by Γ_o

2: Output Gate

$$\Gamma_o = \sigma(W_o.[a_{t-1}, x_t])$$

$$a_t = \Gamma_o * tanh(C_t)$$

3: LSTM

$$\Gamma_u = \sigma(W_u.[a_{t-1}, x_t])$$

$$\Gamma_f = \sigma(W_f.[a_{t-1}, x_t])$$

$$\Gamma_o = \sigma(W_o.[a_{t-1}, x_t])$$

$$\tilde{C}_t = tanh(W_c.[a_{t-1}, x_t])$$

$$C_t = \Gamma_u * \tilde{C}_t + \Gamma_f * C_{t-1}$$

$$a_t = \Gamma_o * tanh(C_t)$$

LSTM: Putting it All Together, With Bias

4: LSTM

$$\Gamma_u = \sigma(W_u.[a_{t-1}, x_t] + b_u)$$

$$\Gamma_f = \sigma(W_f.[a_{t-1}, x_t] + b_f)$$

$$\Gamma_o = \sigma(W_o.[a_{t-1}, x_t] + b_o)$$

$$\tilde{C}_t = tanh(W_c.[a_{t-1}, x_t] + b_c)$$

$$C_t = \Gamma_u * \tilde{C}_t + \Gamma_f * C_{t-1}$$

$$a_t = \Gamma_o * tanh(C_t)$$

LSTM Schematic Illustrated

$$\tilde{C}_t = tanh(W_c.[a_{t-1}, x_t])$$

$$\Gamma_u = \sigma(W_u.[a_{t-1}, x_t])$$

$$\Gamma_f = \sigma(W_f.[a_{t-1}, x_t])$$

$$\Gamma_o = \sigma(W_o.[a_{t-1}, x_t])$$

$$C_t = \Gamma_u * \tilde{C}_t + \Gamma_f * C_{t-1}$$

$$a_t = \Gamma_o * tanh(C_t)$$

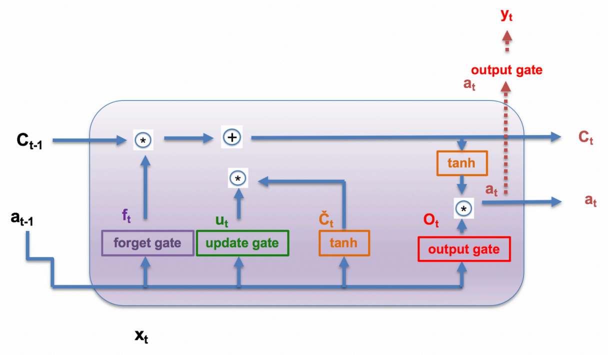

Figure: LSTM cell. [Inspired by Chris Olah]

Stacking LSTM Cells

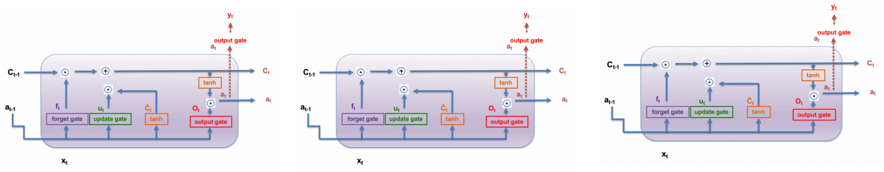

Figure: LSTM cell.s stacked. Note: Each cell will need new-indexing (not shown in the Figure)