LING530F: Deep Learning for Natural Language Processing (DL-NLP)

Muhammad Abdul-Mageed
muhammad.mageed@ubc.ca
Natural Language Processing Lab

The University of British Columbia

Table of Contents

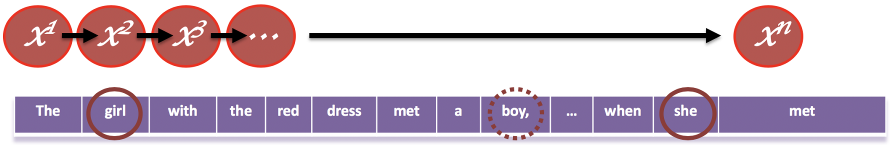

Figure: A very long sequence, modelled with an RNN. Gradients can vanish and we will not know which gender a pronoun should be (male or female), or to which entity the pronoun refers (boy or girl)

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
- Gated Recurrent Units (**GRU**) (Cho et al., 2014; Chung et al., 2014): Simplification of LSTMs.
- We will **introduce GRUs first**, as they are simpler
- **Notation modified from Andrew Ng**, rather than original papers, for pedagogical simplicity (and elegance!)

Introducing a Memory Cell

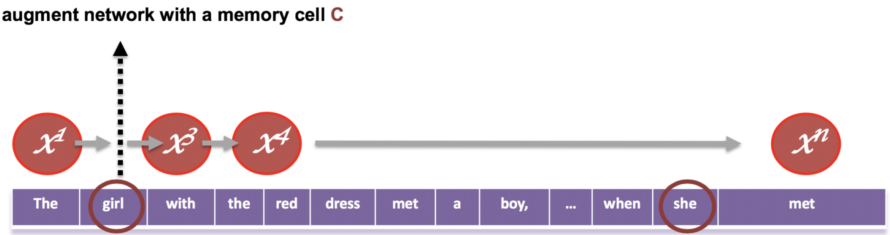

Figure: We will augment the network with a memory cell

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")

Simple GRU

1: Simple GRU

$$C_t = a_t$$

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

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

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

GRU Gates

- **Note:** Bias terms are dropped from equations
- C_t: Memory cell. $\tilde{C}_t$: New candidate memory cell for replacing C_t.
- Γ_u: Update gate.

Simple GRU Update Rule

2: Simple GRU Update

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

GRU Gates

- Γ_u: Result of a a sigmoid (between 0 and 1)
- When Γ_u is close to zero, we update very little $\rightarrow$ we almost keep value of old memory cell C_{t-1}

Full GRU

3: GRU With Relevance Gate

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

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

$$\Gamma_r = \sigma(W_r.[C_{t-1}, x_t])$$

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

$$a_t = C_t$$

GRU Gates

- C_t: Memory cell. $\tilde{C}_t$: New candidate memory cell for replacing C_t.
- Γ_u: Update gate. Γ_r: Relevance gate (*aka* **reset** gate)

Full GRU With Bias Added

4: GRU With Bias

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

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

$$\Gamma_r = \sigma(W_r.[C_{t-1}, x_t] + b_r)$$

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

GRU Gates

- C_t: Memory cell. $\tilde{C}_t$: New candidate memory cell for replacing C_t.
- Γ_u: Update gate. Γ_r: Relevance gate (*aka* **reset** gate)

Alternative Notation For GRU

5: GRU With Bias

$$z_t = \sigma(W_z.[h_{t-1}, x_t])$$

$$r_t = \sigma(W_r.[h_{t-1}, x_t])$$

$$\tilde{h}_t = tanh(W.[r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU Cells Notation Translation Table

- **z**: Update state $\rightarrow \Gamma_u$
- **r**: Relevance state *(reset gate)* $\rightarrow \Gamma_r$
- $\tilde{h}$: New candidate memory cell state $\rightarrow \tilde{C}_t$
- h_t: GRU output $\rightarrow C_t$

Alternative Notation For GRU, With Bias

6: GRU With Bias

$$z_t = \sigma(W_z.[h_{t-1}, x_t] + b_z)$$

$$r_t = \sigma(W_r.[h_{t-1}, x_t] + b_r)$$

$$\tilde{h}_t = tanh(W.[r_t * h_{t-1}, x_t] + b_h)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU Cells Notation Translation Table

- **z**: Update state $\rightarrow \Gamma_u$
- **r**: Relevance state *(reset gate)* $\rightarrow \Gamma_r$
- $\tilde{h}$: New candidate memory cell state $\rightarrow \tilde{C}_t$
- h_t: GRU output $\rightarrow C_t$

7: Recall: Full GRU

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

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

$$\Gamma_r = \sigma(W_r.[C_{t-1}, x_t])$$

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

$$a_t = C_t$$

Toward an LSTM

Changes

- Parts in red **will change!**
- a_t $!=$ C_t and so we will use a_t
- We will also need to **add new parts**, namely a **forget gate** and **output gate** . . .

8: Recall: Full GRU

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

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

$$\Gamma_r = \sigma(W_r.[C_{t-1}, x_t])$$

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

$$a_t = C_t$$

LSTM: New Candidate Cell $\tilde{C}_t$

Updating $\tilde{C}_t$

- To acquire $\tilde{C}_t$, instead of C_{t-1}, we use the new a_{t-1} (since a_{t-1} is acquired differently than C_{t-1}, as we explain later)

9: New Candidate $\tilde{C}_t$

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

LSTM: Update and Forget Gates

Changes

- We will not use an **relevance gate** (Γ_r)
- Instead of using one update gate Γ_u, we will use **two gates** to control cell content: Γ_u (update gate) and Γ_f (forget gate)
- **Forget gate** will give the new memory cell the option to keep or forget the **old cell** (C_{t-1}), but just add to it via **update gate** (Γ_u) $\rightarrow$ See C_t below

10: 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

- As mentioned, 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

11: Output Gate

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

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

LSTM: Putting it All Together

12: LSTM

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

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

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

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

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

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

LSTM: Putting it All Together, With Bias

13: LSTM

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

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

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

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

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

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

LSTM Schematic Illustrated

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

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

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

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

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

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

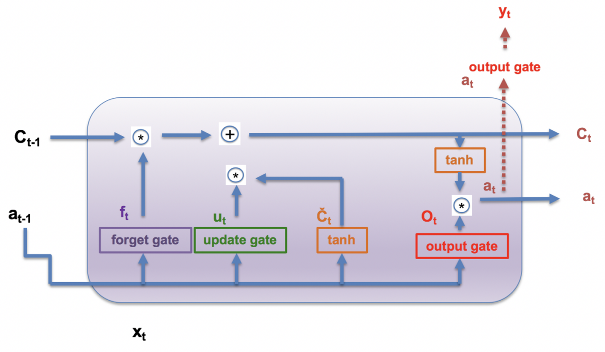

Figure: LSTM cell. [Inspired by Chris Olah and Andrew Ng]

Stacking LSTM Cells

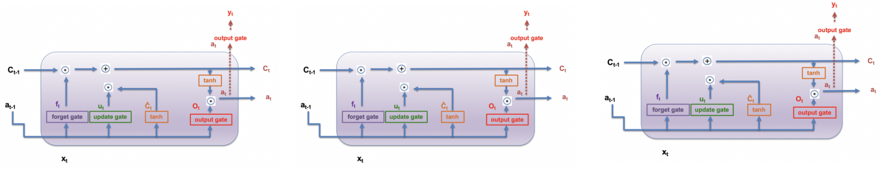

Figure: LSTM cell.s stacked. Note: Each cell will need new-indexing (not shown in the Figure)