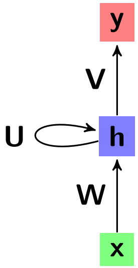
Fancy RNNs

CS114 Lab 9

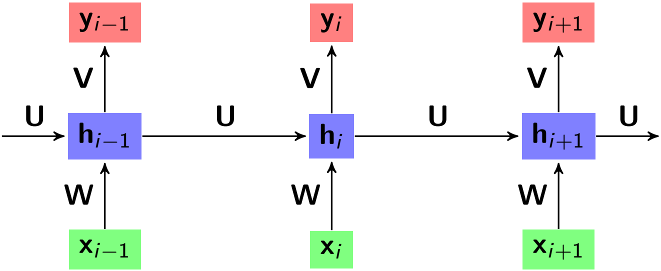
Kenneth Lai

March 20, 2020

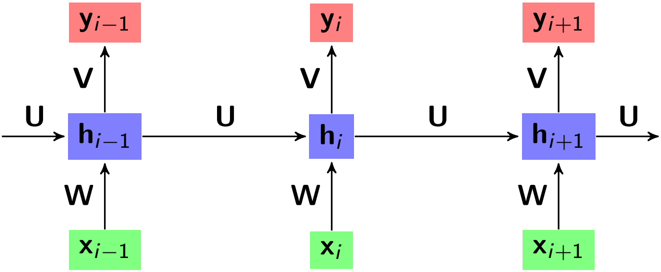
Recurrent Neural Networks



Recurrent Neural Networks

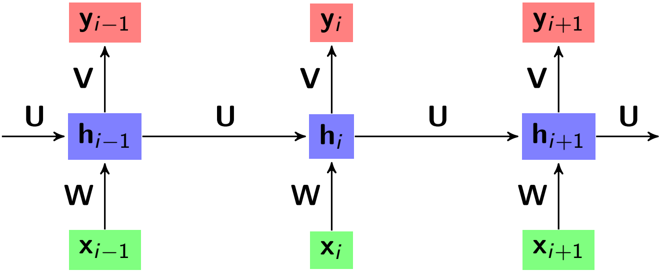


Recurrent Neural Networks



- Output $\mathbf{y}_i$ depends on hidden state $\mathbf{h}_i$ (i.e. current word $\mathbf{x}_i$ and history/(past) context $\mathbf{h}_{i-1}$)

Recurrent Neural Networks



- Output $\mathbf{y}_i$ depends on hidden state $\mathbf{h}_i$ (i.e. current word $\mathbf{x}_i$ and history/(past) context $\mathbf{h}_{i-1}$)
- What about future context?

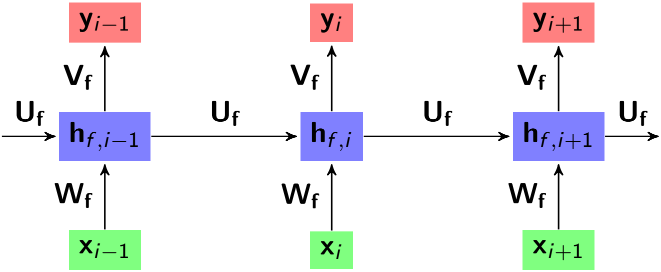
Bidirectional RNNs

- Idea: Train two RNNs: passing the input into one forward and one backward

Bidirectional RNNs

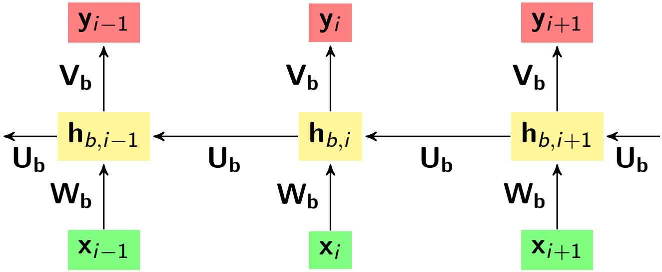
- Idea: Train two RNNs: passing the input into one forward and one backward
- Output $\mathbf{y}_i$ depends on forward hidden state $\mathbf{h}_{f,i}$ and backward hidden state $\mathbf{h}_{b,i}$

Forward RNN

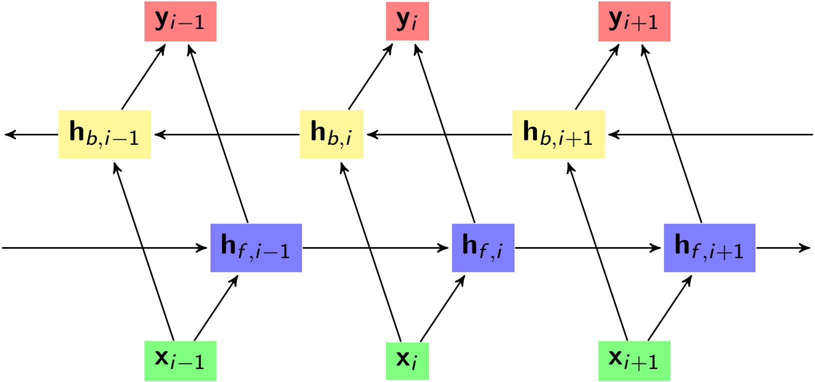


$\mathbf{y}_{i-1}$
$\mathbf{y}_i$
$\mathbf{y}_{i+1}$
$\mathbf{V_f}$
$\mathbf{V_f}$
$\mathbf{V_f}$
$\mathbf{U_f}$
$\mathbf{U_f}$
$\mathbf{U_f}$
$\mathbf{U_f}$
$\mathbf{h}_{f,i-1}$
$\mathbf{h}_{f,i}$
$\mathbf{h}_{f,i+1}$
$\mathbf{W_f}$
$\mathbf{W_f}$
$\mathbf{W_f}$
$\mathbf{x}_{i-1}$
$\mathbf{x}_i$
$\mathbf{x}_{i+1}$

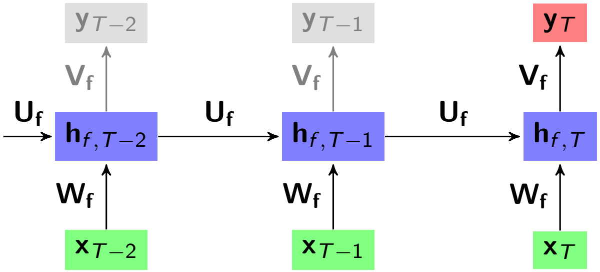
Backward RNN



Bidirectional RNN

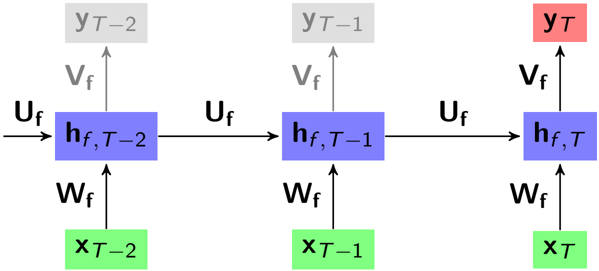


Bidirectional RNNs for Text Classification



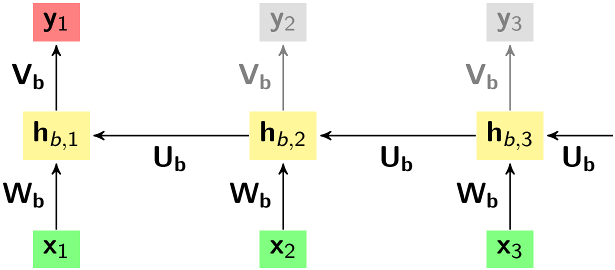
- $\mathbf{h}_{f,T}$ encodes the whole text

Bidirectional RNNs for Text Classification



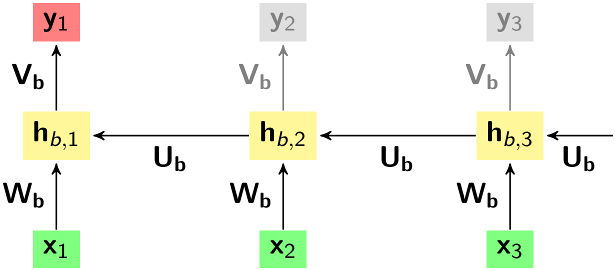
- $\mathbf{h}_{f,T}$ encodes the whole text
  - Use $\mathbf{h}_{f,T}$ to predict class $\mathbf{y}_T$ of entire document

Bidirectional RNNs for Text Classification



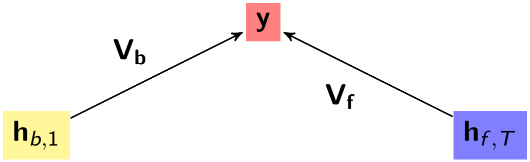
- $\mathbf{h}_{f,T}$ encodes the whole text
  - Use $\mathbf{h}_{f,T}$ to predict class $\mathbf{y}_T$ of entire document
- $\mathbf{h}_{b,1}$ also encodes the whole text

Bidirectional RNNs for Text Classification

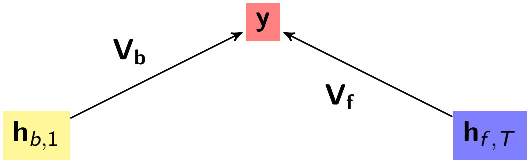


- $\mathbf{h}_{f,T}$ encodes the whole text
  - Use $\mathbf{h}_{f,T}$ to predict class $\mathbf{y}_T$ of entire document
- $\mathbf{h}_{b,1}$ also encodes the whole text
  - Use $\mathbf{h}_{b,1}$ to predict class $\mathbf{y}_1$ of entire document

Bidirectional RNNs for Text Classification



Bidirectional RNNs for Text Classification



- Use $\mathbf{h}_{f,T}$ and $\mathbf{h}_{b,1}$ to predict class $\mathbf{y}$ of entire document

Context and Long-Distance Dependencies

- $\mathbf{h}_{i-1}$ encodes the (past, in a forward RNN) context $\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}$

Context and Long-Distance Dependencies

- $\mathbf{h}_{i-1}$ encodes the (past, in a forward RNN) context $\mathbf{x}_1, ..., \mathbf{x}_{i-1}$
  - But mostly $\mathbf{x}_{i-1}$, less $\mathbf{x}_{i-2}$, even less $\mathbf{x}_{i-3}$, ..., very little $\mathbf{x}_1$

Context and Long-Distance Dependencies

- $\mathbf{h}_{i-1}$ encodes the (past, in a forward RNN) context $\mathbf{x}_1, \dots, \mathbf{x}_{i-1}$
  - But mostly $\mathbf{x}_{i-1}$, less $\mathbf{x}_{i-2}$, even less $\mathbf{x}_{i-3}$, ..., very little $\mathbf{x}_1$
- Context is local

Context and Long-Distance Dependencies

- Example: subject-verb agreement

Context and Long-Distance Dependencies

- Example: subject-verb agreement
- The flights the airline was cancelling were full.

Context and Long-Distance Dependencies

- Example: subject-verb agreement
- The flights the airline was cancelling were full.

Context and Long-Distance Dependencies

- Example: subject-verb agreement
- The flights the airline was cancelling were full.
  - The context for “was” is mostly “airline”

Context and Long-Distance Dependencies

- Example: subject-verb agreement
- The flights the airline was cancelling were full.
  - The context for was is mostly airline

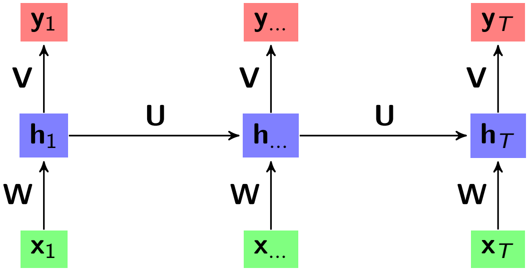
Context and Long-Distance Dependencies

- Example: subject-verb agreement
- The flights the airline was cancelling were full.
  - The context for was is mostly airline
  - The context for were is mostly cancelling, was, airline

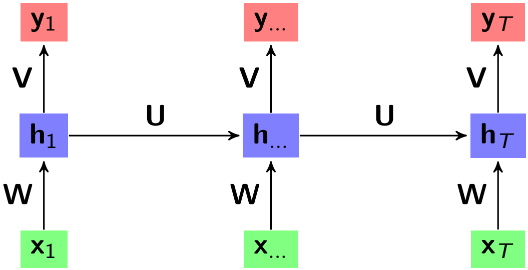
Context and Long-Distance Dependencies

- Example: subject-verb agreement
- The flights the airline was cancelling were full.
  - The context for was is mostly airline
  - The context for were is mostly cancelling, was, airline
    - Very little flights

Vanishing and Exploding Gradients

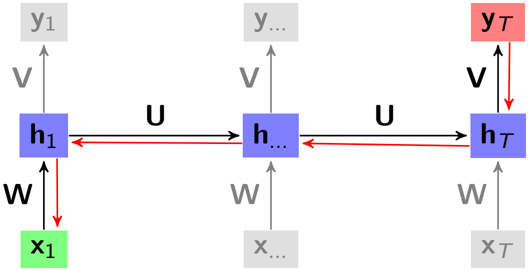


Vanishing and Exploding Gradients



- What is $\nabla_{\mathbf{w},1,T} L$?

Vanishing and Exploding Gradients

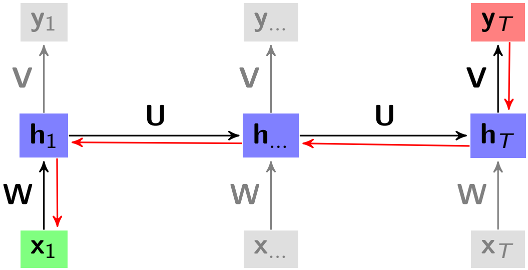


- What is $\nabla_{\mathbf{W},1,T} L$?

Vanishing and Exploding Gradients

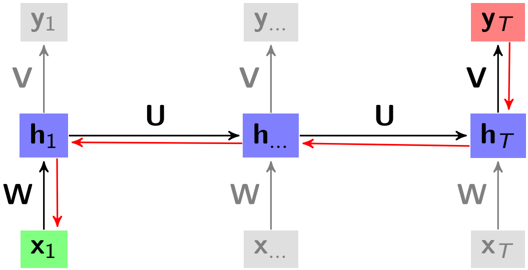
- For all layers l:
  - $\nabla_l L = \delta_l \odot \mathbf{x}_l$
- For an output layer $\mathcal{L}$:
  - $\delta_{\mathcal{L}} = \hat{\mathbf{y}} - \mathbf{y}$
- For a non-output layer J (with next layer K):
  - $\delta_J = (\mathbf{W}_K^T \cdot \delta_K) \odot \sigma'(\mathbf{z}_J)$

Vanishing and Exploding Gradients



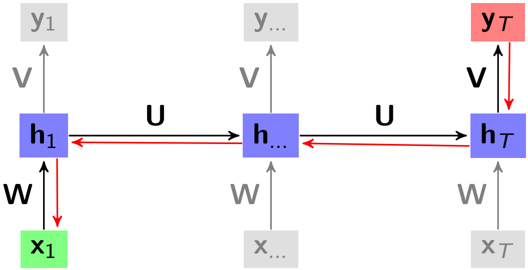
$$\nabla_{\mathbf{W},1,T} L = \delta_{\mathbf{h}_1} \odot \mathbf{x}_1$$

Vanishing and Exploding Gradients



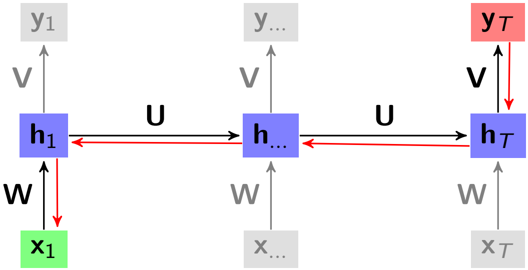
$$
\begin{aligned}
\nabla_{\mathbf{W},1,T} L &= \delta_{\mathbf{h}_1} \odot \mathbf{x}_1 \\
&= (\mathbf{U}^T \cdot \delta_{\mathbf{h}_2}) \odot \sigma'(\mathbf{z}_{\mathbf{h}_1}) \odot \mathbf{x}_1
\end{aligned}
$$

Vanishing and Exploding Gradients



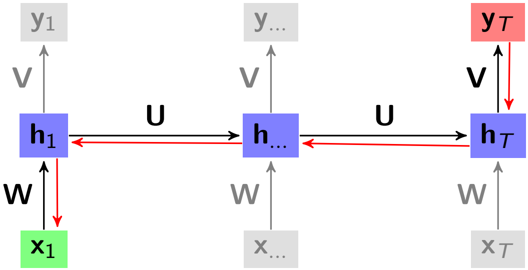
$$
\begin{aligned}
\nabla_{\mathbf{W},1,T} L &= \delta_{\mathbf{h}_1} \odot \mathbf{x}_1 \\
&= (\mathbf{U}^T \cdot \delta_{\mathbf{h}_2}) \odot \sigma'(\mathbf{z}_{\mathbf{h}_1}) \odot \mathbf{x}_1 \\
&= (\mathbf{U}^T \cdot ((\mathbf{U}^T \cdot \delta_{\mathbf{h}_3}) \odot \sigma'(\mathbf{z}_{\mathbf{h}_2}))) \odot \sigma'(\mathbf{z}_{\mathbf{h}_1}) \odot \mathbf{x}_1
\end{aligned}
$$

Vanishing and Exploding Gradients



$$
\begin{aligned}
\nabla_{\mathbf{W},1,T} L &= \delta_{\mathbf{h}_1} \odot \mathbf{x}_1 \\
&= (\mathbf{U}^T \cdot \delta_{\mathbf{h}_2}) \odot \sigma'(\mathbf{z}_{\mathbf{h}_1}) \odot \mathbf{x}_1 \\
&= (\mathbf{U}^T \cdot ((\mathbf{U}^T \cdot \delta_{\mathbf{h}_3}) \odot \sigma'(\mathbf{z}_{\mathbf{h}_2}))) \odot \sigma'(\mathbf{z}_{\mathbf{h}_1}) \odot \mathbf{x}_1 \\
&= (\mathbf{U}^T \cdot ((\mathbf{U}^T \cdot ((\mathbf{U}^T \cdot \dots \odot \sigma'(\mathbf{z}_{\mathbf{h}_2}))) \odot \sigma'(\mathbf{z}_{\mathbf{h}_1}) \odot \mathbf{x}_1
\end{aligned}
$$

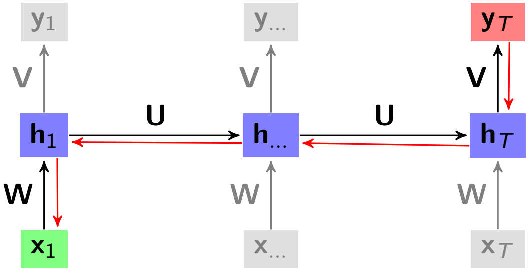
Vanishing and Exploding Gradients



$$
\begin{aligned}
\nabla_{\mathbf{W},1,T} L &= \delta_{\mathbf{h}_1} \odot \mathbf{x}_1 \\
&= (\mathbf{U}^T \cdot \delta_{\mathbf{h}_2}) \odot \sigma'(\mathbf{z}_{\mathbf{h}_1}) \odot \mathbf{x}_1 \\
&= (\mathbf{U}^T \cdot ((\mathbf{U}^T \cdot \delta_{\mathbf{h}_3}) \odot \sigma'(\mathbf{z}_{\mathbf{h}_2}))) \odot \sigma'(\mathbf{z}_{\mathbf{h}_1}) \odot \mathbf{x}_1 \\
&= (\mathbf{U}^T \cdot ((\mathbf{U}^T \cdot ((\mathbf{U}^T \cdot \dots \odot \sigma'(\mathbf{z}_{\mathbf{h}_2}))) \odot \sigma'(\mathbf{z}_{\mathbf{h}_1}) \odot \mathbf{x}_1
\end{aligned}
$$

- If weights/derivatives are small, vanishing gradient

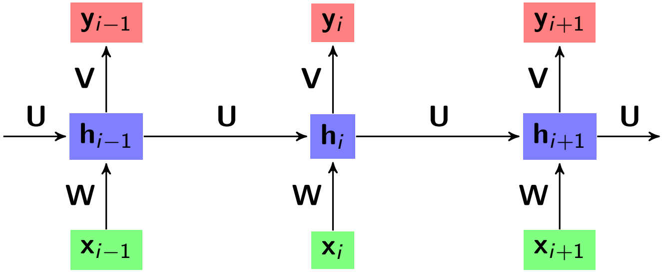
Vanishing and Exploding Gradients



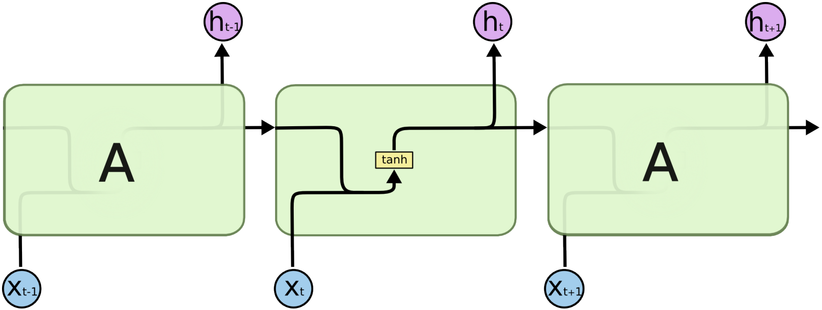
$$
\begin{aligned}
\nabla_{\mathbf{W},1,T} L &= \delta_{\mathbf{h}_1} \odot \mathbf{x}_1 \\
&= (\mathbf{U}^T \cdot \delta_{\mathbf{h}_2}) \odot \sigma'(\mathbf{z}_{\mathbf{h}_1}) \odot \mathbf{x}_1 \\
&= (\mathbf{U}^T \cdot ((\mathbf{U}^T \cdot \delta_{\mathbf{h}_3}) \odot \sigma'(\mathbf{z}_{\mathbf{h}_2}))) \odot \sigma'(\mathbf{z}_{\mathbf{h}_1}) \odot \mathbf{x}_1 \\
&= (\mathbf{U}^T \cdot ((\mathbf{U}^T \cdot ((\mathbf{U}^T \cdot \dots \odot \sigma'(\mathbf{z}_{\mathbf{h}_2}))) \odot \sigma'(\mathbf{z}_{\mathbf{h}_1}) \odot \mathbf{x}_1
\end{aligned}
$$

- If weights/derivatives are small, vanishing gradient
- If weights/derivatives are large, exploding gradient

Simple RNN

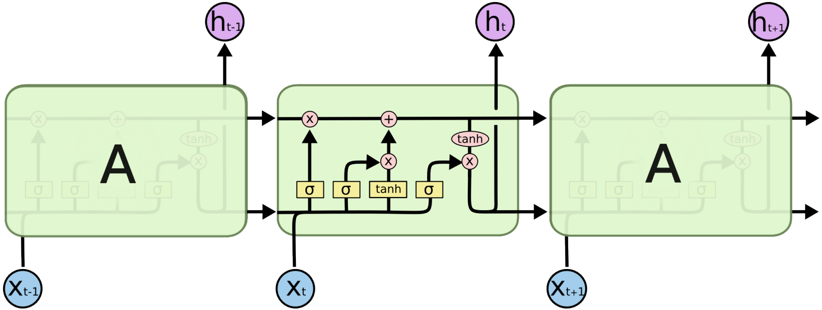

$\mathbf{y}_{i-1}$
$\mathbf{y}_i$
$\mathbf{y}_{i+1}$
V
V
V
U
$\mathbf{h}_{i-1}$
U
$\mathbf{h}_i$
U
$\mathbf{h}_{i+1}$
U
W
W
W
$\mathbf{x}_{i-1}$
$\mathbf{x}_i$
$\mathbf{x}_{i+1}$

Simple RNN



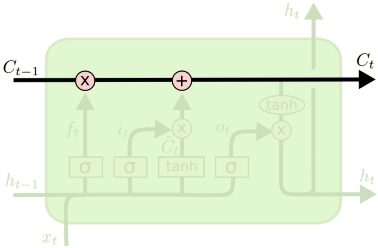
Source

Long Short-Term Memory



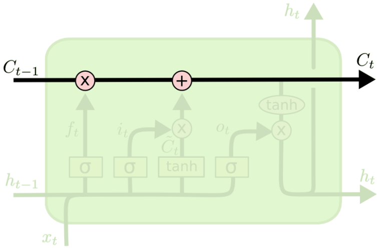
Source

Long Short-Term Memory



Source

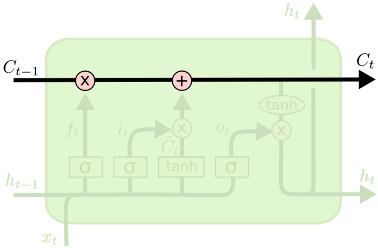
Long Short-Term Memory



Source

- Separate memory (cell) state

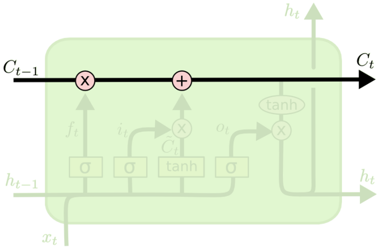
Long Short-Term Memory



Source

- Separate memory (cell) state
  - Reading from and writing to memory controlled by gates

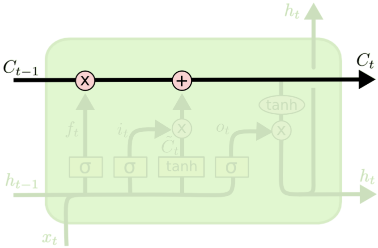
Long Short-Term Memory



Source

- Separate memory (cell) state
  - Reading from and writing to memory controlled by gates
    - Each gate contains one or two neural network layers

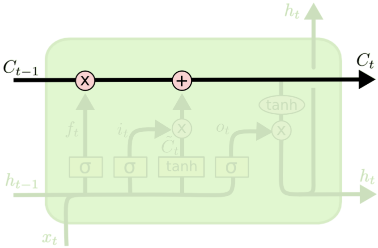
Long Short-Term Memory



Source

- Separate memory (cell) state
  - Reading from and writing to memory controlled by gates
    - Each gate contains one or two neural network layers
  - State persists across time

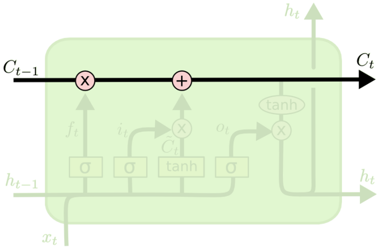
Long Short-Term Memory



Source

- Separate memory (cell) state
  - Reading from and writing to memory controlled by gates
    - Each gate contains one or two neural network layers
  - State persists across time
    - May remember information from long ago

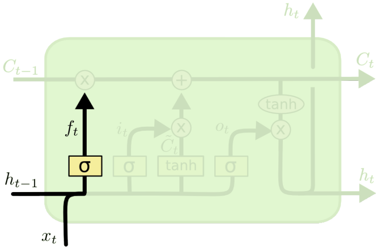
Long Short-Term Memory



Source

- Separate memory (cell) state
  - Reading from and writing to memory controlled by gates
    - Each gate contains one or two neural network layers
  - State persists across time
    - May remember information from long ago
    - Gradients for memory don't decay with time

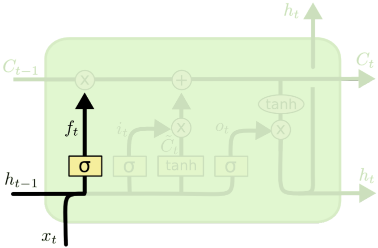
Forget Gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Source

Forget Gate

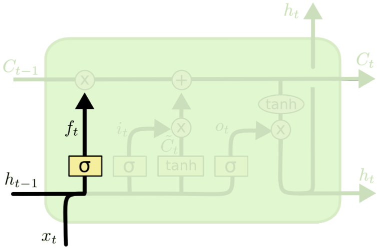


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Source

- Neural network layer with logistic activation function

Forget Gate

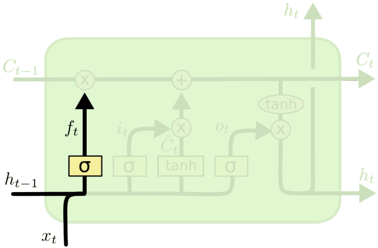


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Source

- Neural network layer with logistic activation function
- Element-wise multiplication of forget gate output with memory state

Forget Gate

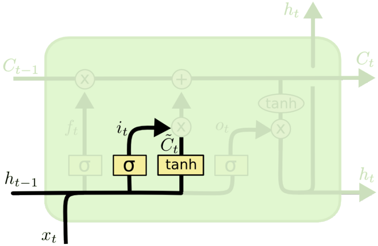


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Source

- Neural network layer with logistic activation function
- Element-wise multiplication of forget gate output with memory state
  - Mask: What parts of memory to forget/remember?

Input Gate

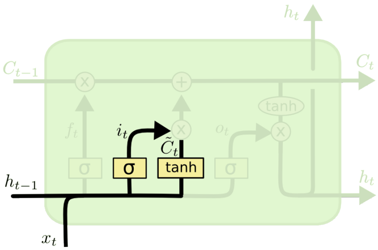


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Source

Input Gate

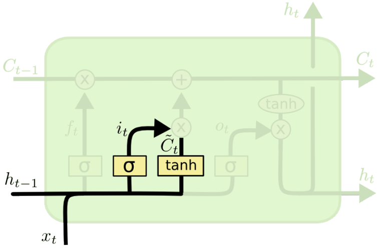


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Source

- Two parts

Input Gate

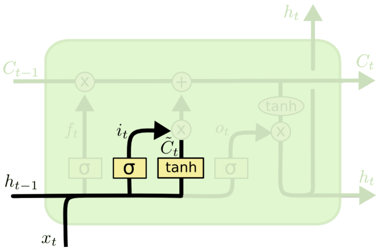


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Source

- Two parts
  1. Candidate choice

Input Gate



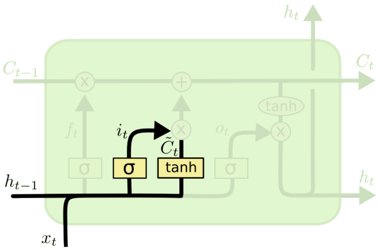
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Source

- Two parts
  1. Candidate choice
     - Logistic activation function

Input Gate



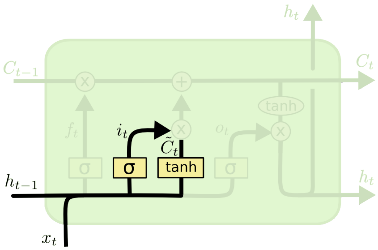
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Source

- Two parts
  1. Candidate choice
     - Logistic activation function
     - What parts of memory to update?

Input Gate

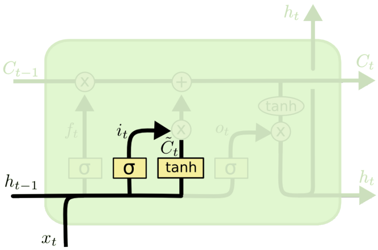


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Source

- Two parts
  1. Candidate choice
     - Logistic activation function
     - What parts of memory to update?
  2. Candidate values

Input Gate



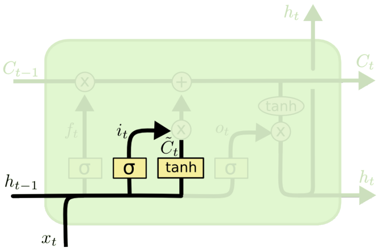
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Source

- Two parts
  1. Candidate choice
     - Logistic activation function
     - What parts of memory to update?
  2. Candidate values
     - Tanh activation function

Input Gate



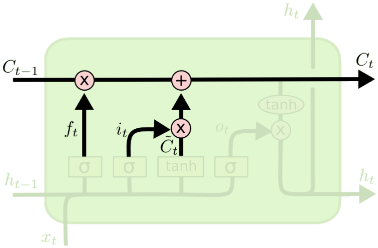
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Source

- Two parts
  1. Candidate choice
     - Logistic activation function
     - What parts of memory to update?
  2. Candidate values
     - Tanh activation function
     - How much to update them by?

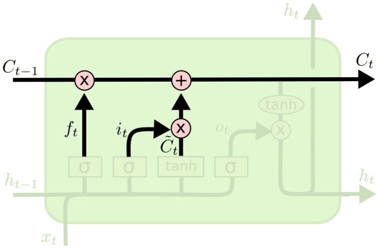
Input Gate



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Source

Input Gate

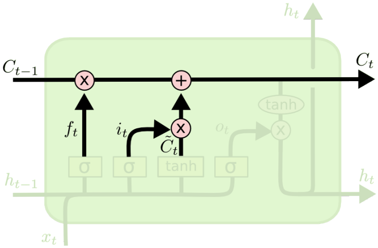


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Source

- Element-wise multiplication of two outputs

Input Gate

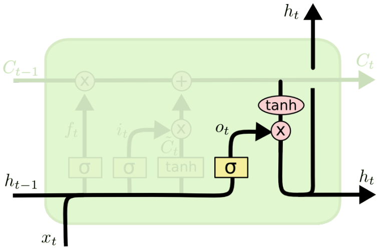


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Source

- Element-wise multiplication of two outputs
- Then element-wise addition with memory state

Output Gate

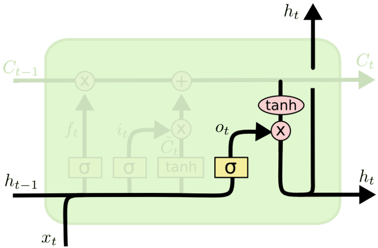


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Source

Output Gate



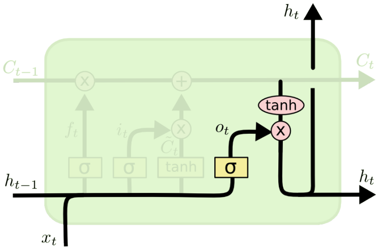
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Source

- Logistic activation function

Output Gate



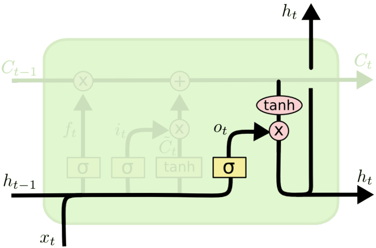
$$o_t = \sigma\left(W_o \, [h_{t-1}, x_t] \, + \, b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Source

- Logistic activation function
  - What parts of memory to output?

Output Gate



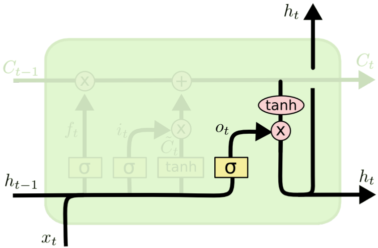
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Source

- Logistic activation function
  - What parts of memory to output?
- Element-wise multiplication with tanh of memory state

Output Gate



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Source

- Logistic activation function
  - What parts of memory to output?
- Element-wise multiplication with tanh of memory state
  - This is the "hidden layer output" that gets passed on to the output layer/next time step