UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack:Divya K,
Teaching Assistant

Overview of lecture

- Introduction
- Whiteboard Analogy
- LSTM Architecture
- How LSTMs avoid the problem of vanishing gradients

Introduction

- In the previous lecture, we learnt about how RNNs suffer from the vanishing gradient problem and cannot handle long-term dependencies. We encountered long short-term memory (LSTM) as a solution to this problem.

- LSTMs solve this problem with the help of special structures called gates. A typical LSTM cell consists of three special gates called the input gate, output gate, and forget gate to control the flow of information.

- These three gates are responsible for deciding what information to add, output, and forget from the memory. With these gates, an LSTM cell effectively keeps information in the memory only as long as required.

Introduction

- Within an LSTM cell, we have two primary states: the cell state and the hidden state.
- The cell state serves as the "memory" of the LSTM cell, allowing it to store and access information over time. The cell state is passed between LSTM cells, enabling them to retain relevant information throughout the sequence.
- The hidden state represents the current state of the LSTM cell and is used for computing outputs at each time step. It encapsulates learned information from previous states and influences predictions made by the network.
- Let us understand the LSTM operations with the help of a whiteboard analogy.

Whiteboard Analogy

- We can think of the cell state as a fixed size memory, compare this to a fixed size white board that you use to record information. At each time step (periodic intervals) we keep writing something to the board.

- To maximize the important information we can store in that limited space we follow these steps:
  - ➢Selectively write on the board
  - ➢Selectively read the already written content
  - ➢Selectively forget (erase) some content

- Say we want to compute the given equation on the board:

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

Say "board" can have only 3 statements at a time.

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

Whiteboard Analogy

➢ Selective Write:

- We have written 3 statements ac, bd, bd+a on the board.
- We could have written individual values of a, b, c, d but we only have finite space/memory and cannot include unnecessary information.
- We are writing only specific information instead of writing everything on to the board, this is called **Selective Write.**

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

Say "board" can have only 3 statements at a time.

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

$$ac = 5$$
$$bd = 33$$
$$bd + a = 34$$

Whiteboard Analogy

➢ Selective Read:

- While writing one step we typically read some of the previous steps we have already written and then decide what to write next, but we do not read everything.
- For example at Step 3, information from Step 2 is important.
- This is called **Selective Read.**

$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$

Compute $ac(bd + a) + ad$

Say "board" can have only 3 statements at a time.

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

$$ac = 5$$
$$bd = 33$$
$$bd + a = 34$$

Whiteboard Analogy

➢ Selective Forget:

- Once the board is full, we need to delete some obsolete information.
- We will typically delete the least useful information.
- This is called **Selective Forget**
- In this case, we forget the 3 statements as we are done with them, and use the values calculated there to further complete the computation.
- We repeatedly perform selective write, selective read and selective forget until we have solved the equation.

$$a = 1 \quad b = 3 \quad c = 5 \quad d = 11$$

Compute $ac(bd + a) + ad$

Say "board" can have only 3 statements at a time.

1. ac
2. bd
3. $bd + a$
4. $ac(bd + a)$
5. ad
6. $ac(bd + a) + ad$

$$ad + ac(bd + a) = 181$$
$$ac(bd + a) = 170$$
$$ad = 11$$

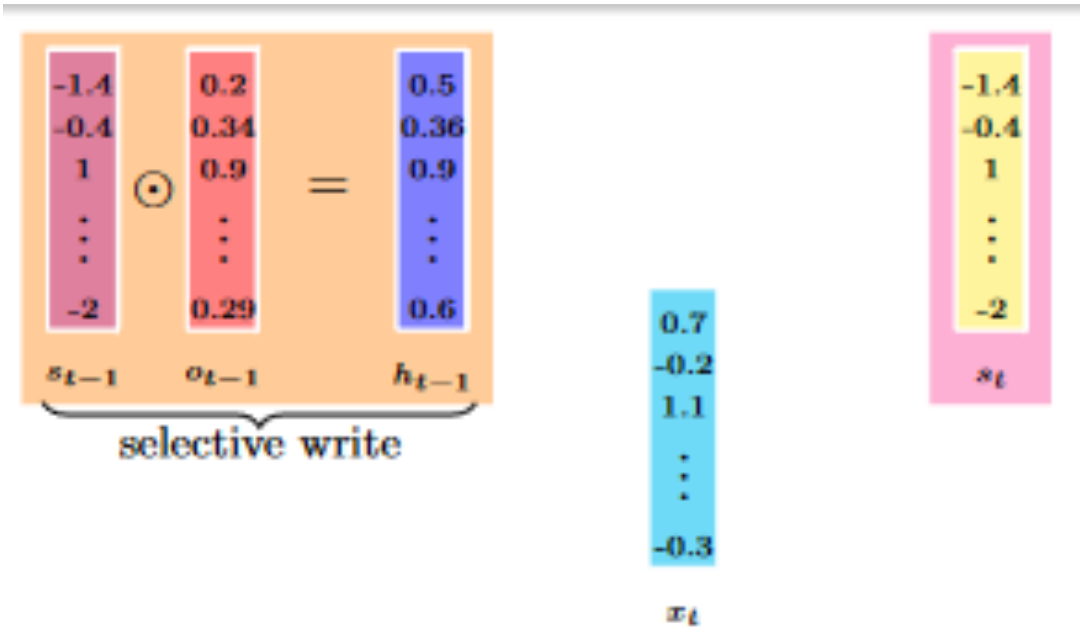
LSTM Architecture

➢ The architecture of LSTM networks is shaped by the addition of selective read, selective write, and selective forget capabilities to the traditional RNN architecture. Let us understand how these capabilities are incorporated with the help of gates.

- Recall that in RNNs we use s_{t-1} to compute s_t : $s_t = \sigma(W*s_{t-1} + U*x_t)$ (ignoring bias)
- But now instead of passing s_{t-1} as it is to s_t we want to pass (write) only some portions of it to the next state => Selective Write
- To do this we introduce another gate called the output gate

  $$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$

- a vector o_{t-1} which decides what fraction of each element of s_{t-1} should be passed to the next state. Each element of o_{t-1} gets multiplied with the corresponding element of s_{t-1}

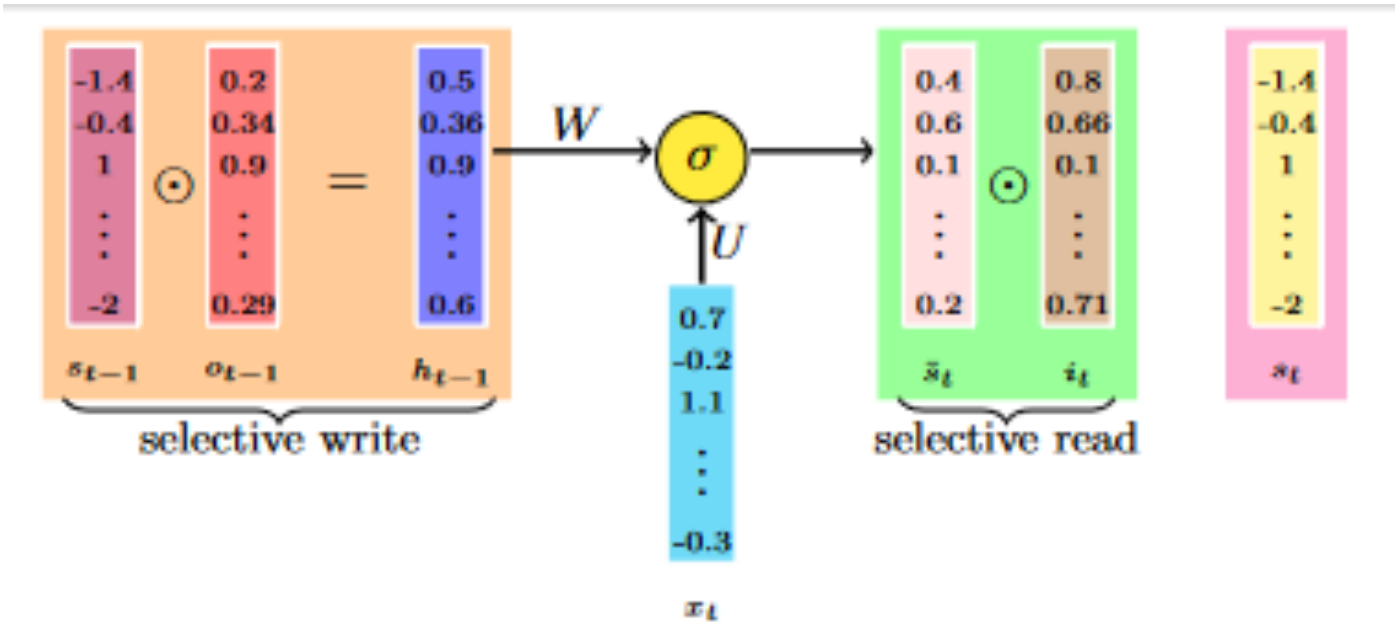


LSTM Architecture

- $\tilde{s}_t$ captures all the information from the previous state (h_{t-1}) and the current input x_t.
- However, we may not want to use all this new information from it before constructing the new cell state s_t => Selective Read
- To do this we introduce another gate called the input gate

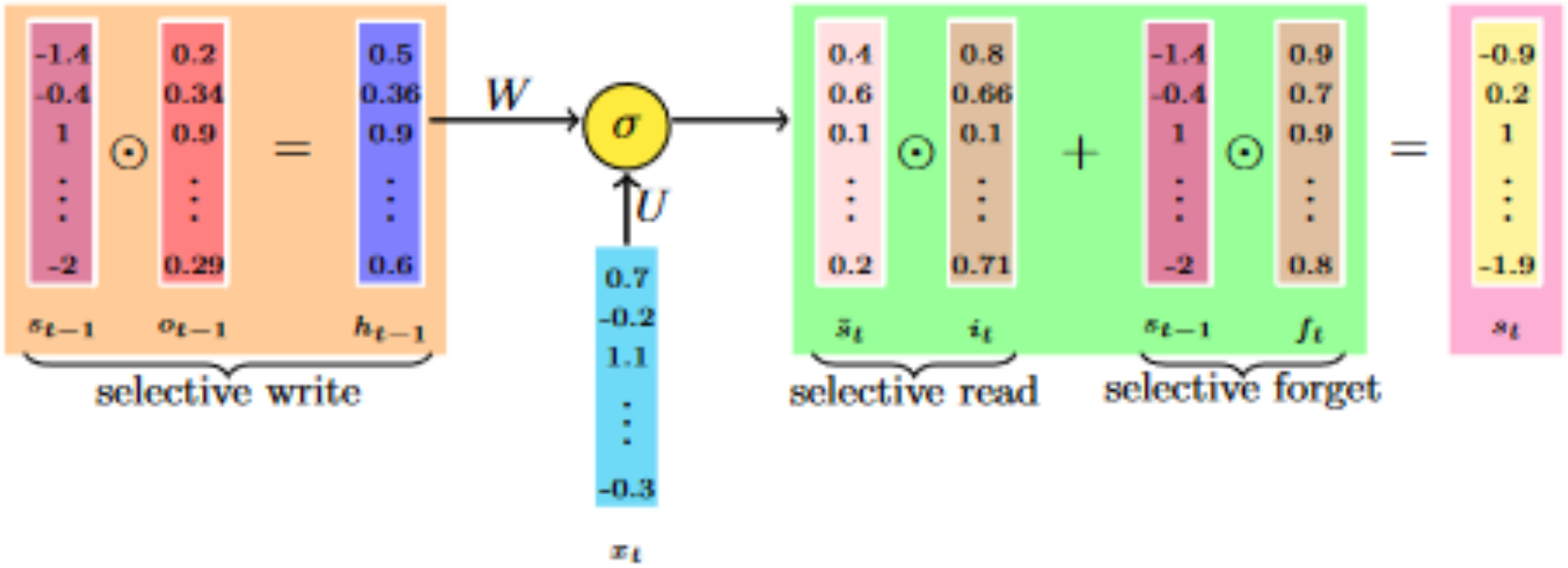
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$

- and use i_t* $\tilde{s}_t$ as the selectively read state information



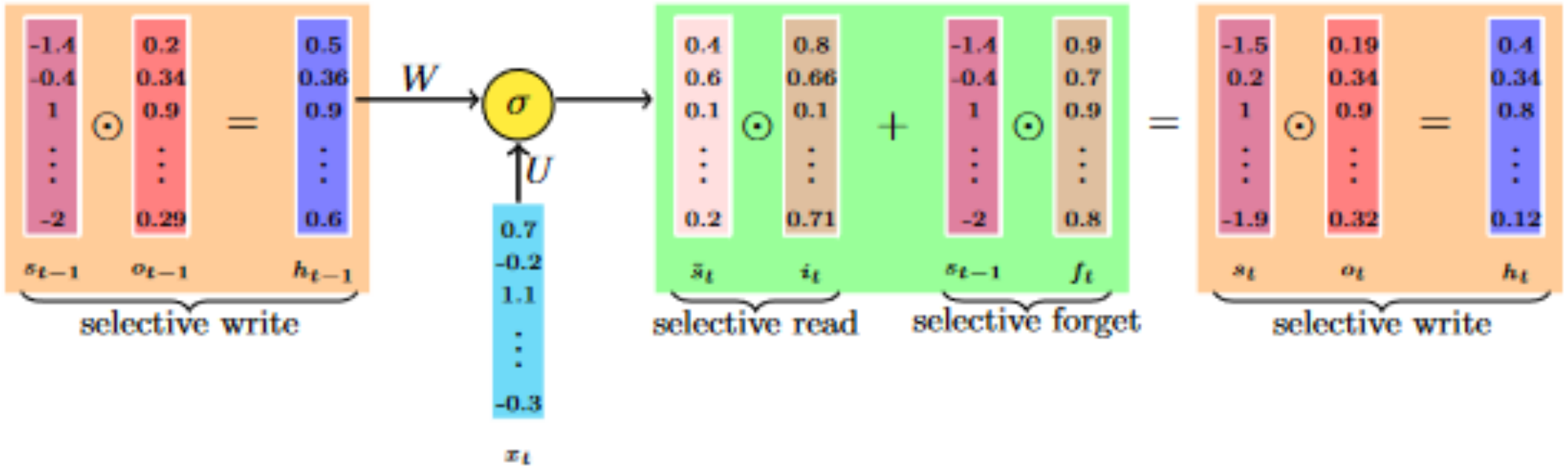
LSTM Architecture

- But now we may not want to use whole of s_{t-1} but forget some parts of it => Selective forget
- To do this we introduce another gate called the forget gate, $f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$
- it is multiplied elementwise with the previous state and the result is added to the outcome of selective read to get the final state (s_t) in that time step



LSTM Architecture

➢ We not have the final architecture of LSTMs:



Gates:

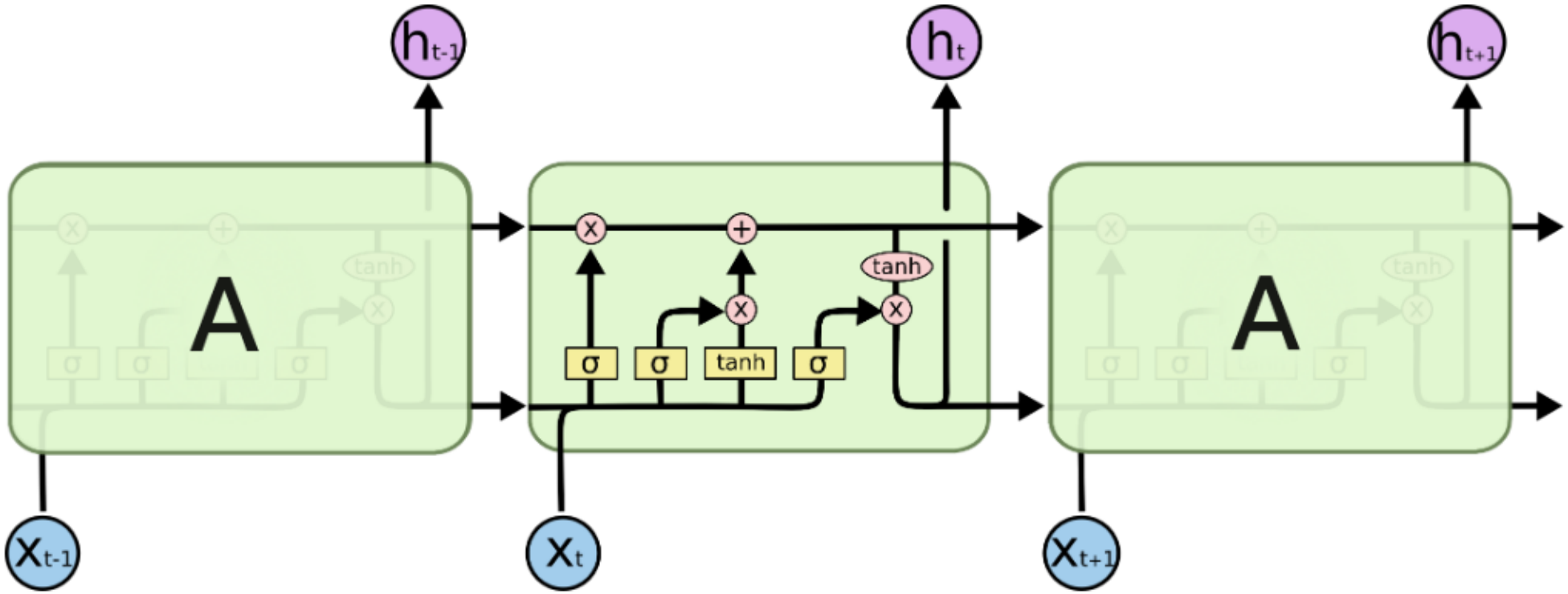
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

States:

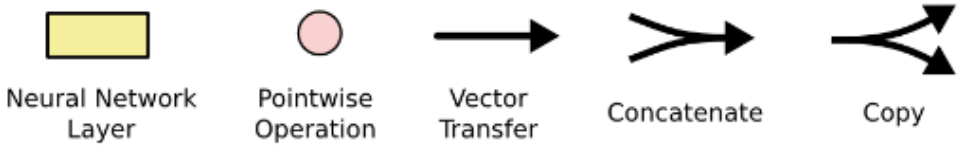
$$\tilde{s_t} = \sigma(W h_{t-1} + U x_t + b)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s_t}$$
$$h_t = o_t \odot \sigma(s_t) \text{ and } rnn_{out} = h_t$$

LSTM Architecture

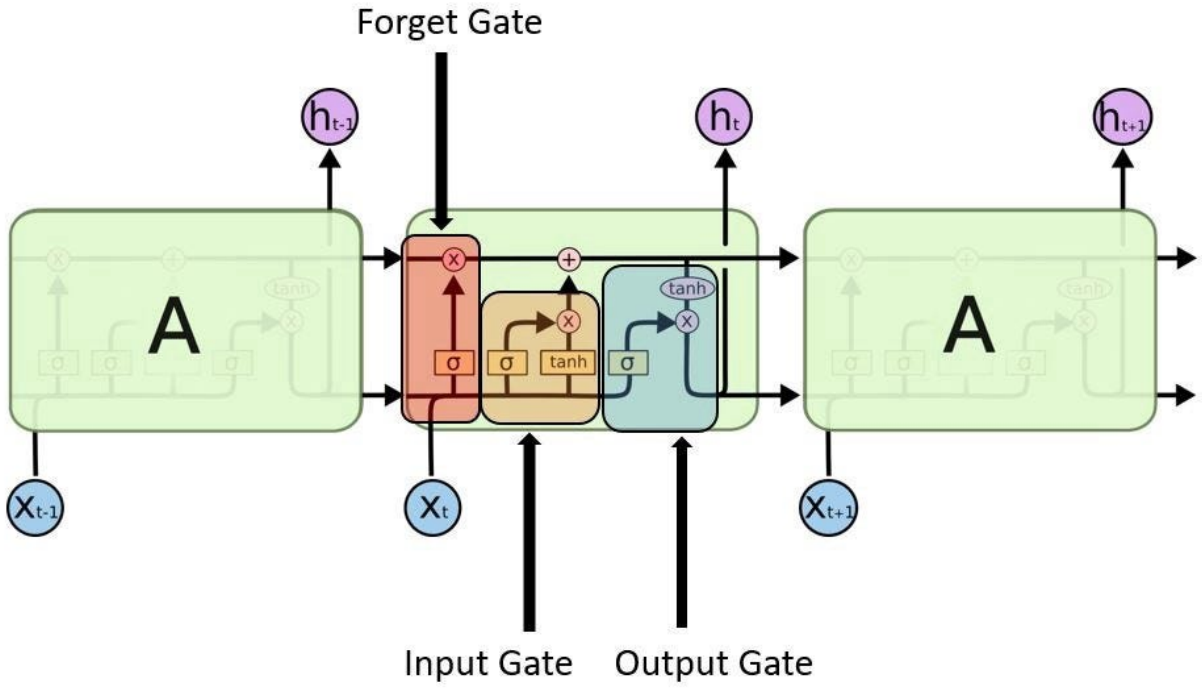
➢ Let's look at another representation:

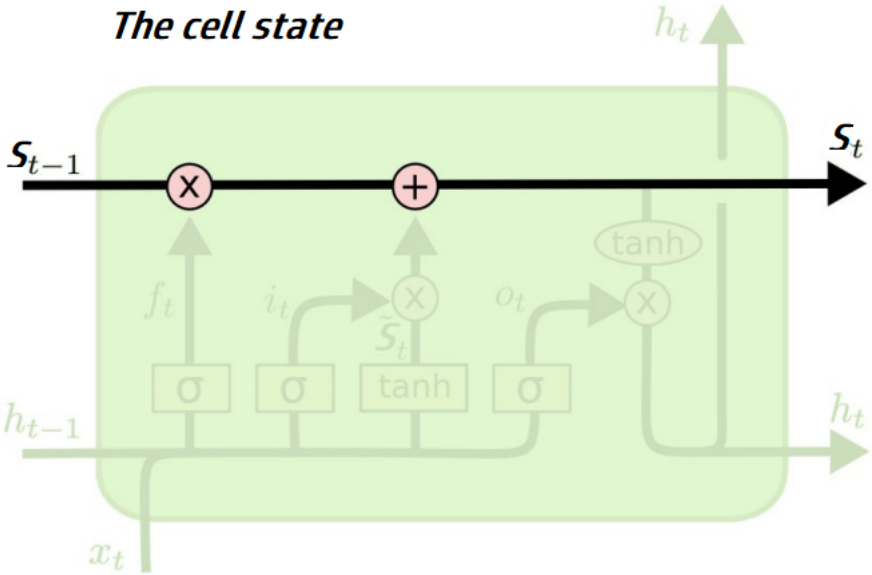


The repeating module in an LSTM contains four interacting layers.



LSTM Architecture





Gates:

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

States:

$$\tilde{S}_t = \tanh(W \cdot [h_{t-1}, x_t] \; + \; b)$$
$$S_t = f_t * S_{t-1} + i_t * \tilde{S}_t$$
$$h_t = o_t * \tanh\left(S_t\right)$$

How LSTMs avoid the problem of vanishing gradients

- During forward propagation the gates control the flow of information
- They prevent any irrelevant information from being written to the state
- Similarly during backward propagation they control the flow of gradients
- During backward pass the gradients will get multiplied by the gate
- The key difference of LSTMs from vanilla RNNs is that the flow of information and gradients is controlled by the gates which ensure that the gradients vanish only when they should (i.e., when s_{t-1} didn't contribute much to s_t)

- LSTMs implement the concept of Selective Read, Selective Write and Selective Forget using gates and solve the problem of vanishing gradients.

- However multiple gates lead to increase in number learnable parameters, and this is solved by one more modified architecture called GRUs.

Acknowledgements

https://nptel.ac.in/courses/106106184

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

https://youtube.com/playlist?list=PLKnIA16_RmvYuZauWaPlRTC54KxSNLtNn&si=a2L-j8rAkG15EWKY

https://www.deeplearning.ai/courses/deep-learning-specialization/

UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack:Divya K,
Teaching Assistant