**Phần 4 - Về xử lý tín hiệu chuỗi thời gian**

**Notes:**

* *Về format cấu trúc soạn như sau:*
  + *Lý thuyết…*
  + *Bộ code mấu/ ví dụ …*
  + *Ứng dụng (nếu có)...*
* *Mems làm nhớ note tên để mn dễ contact*

**Outline**

[**I. Nội dung 1**](#_jbp87p2p857b)

[**II. Phần soạn chi tiết nội dung 1**](#_vhgil1udi0w)

[**[Author 1] VINH - Phần lý thuyết 1**](#_yixc8zxot818)

[**I. 1+2 .Nguyên tắc làm việc của một cell LSTM (Long short-term memory). Trách nhiệm của các cổng (gate) trong LSTM. 1**](#_sdharbg2n6yw)

[1. Sơ lược về LSTM 1](#_hmuujqlpyqct)

[2. Tại sao lại là LSTM ? 2](#_q6m40o59qsbr)

[3. Mô hình và nguyên tắc làm việc của LSTM. 3](#_gre1tz6zzms2)

[4. Trách nhiệm của các gate trong LSTM 3](#_t81394ct14m2)

[**II. Số lượng tham số cần học 7**](#_l8wqxjiwy2vy)

[**[Author 2] HÙNG - Phần hiện thực 8**](#_vqlsmw6eulft)

### **I. Nội dung**

Về xử lý tín hiệu chuỗi thời gian: **LSTM**

* (1) Nguyên tắc làm việc của một cell LSTM.
* (2) Cách nhiệm các cổng bên trong LSTM, như: forget, input, etc.
* (3) Số lượng tham số cần học.

### **II. Phần soạn chi tiết nội dung**

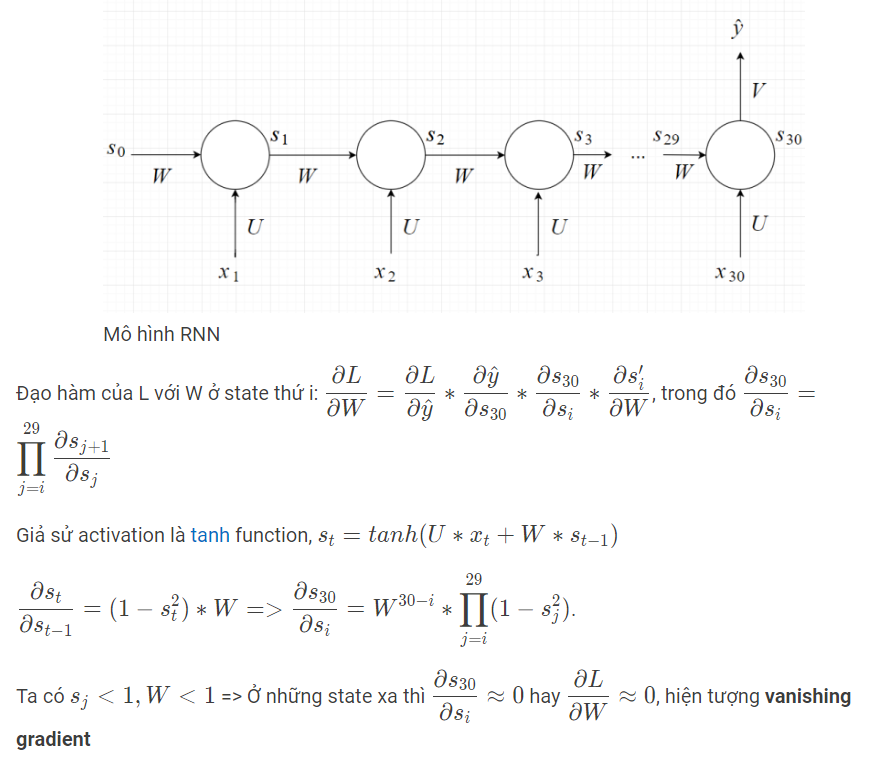
#### [Author 1] VINH - Phần lý thuyết

#### **I. 1+2 .Nguyên tắc làm việc của một cell LSTM (Long short-term memory). Trách nhiệm của các cổng (gate) trong LSTM.**

##### **1. Sơ lược về LSTM**

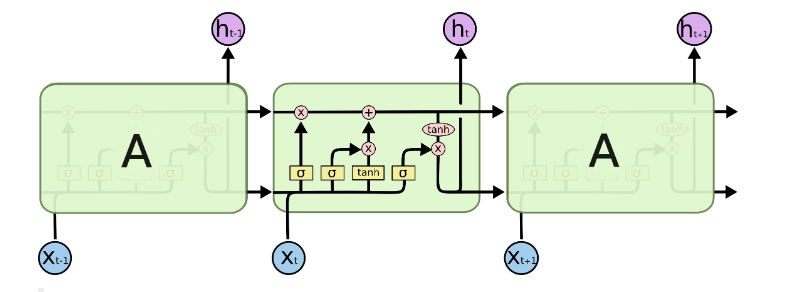
* Long short-term memory (LSTM) là một kiến ​​trúc artificial recurrent neural network (RNN) được sử dụng trong lĩnh vực Deep learning.
* Không giống như các feed-forward neural networks, LSTM có các kết nối phản hồi. Nó có thể xử lý không chỉ các điểm dữ liệu đơn lẻ (chẳng hạn như hình ảnh) mà còn toàn bộ chuỗi dữ liệu (chẳng hạn như speech hoặc video).

##### ***2. Tại sao lại là LSTM ?***

* Nhắc lại RNN, RNN có thể xử lý thông tin dạng chuỗi (sequence/ time-series). , RNN có thể mang thông tin của frame (ảnh) từ state trước tới các state sau, rồi ở state cuối là sự kết hợp của tất cả các ảnh để dự đoán kết quả.
* 
* Có thể thấy ở state càng xa → vanishing gradient và các hệ số không update được ở frame xa
* Như vậy RNN chỉ mang được thông tin qua một số lượng state nhất định, sau đó sẽ bị vanishing gradient -> **short term memory**
* Ví dụ ta cần một model giải quyết bài toán dự đoán từ cho một đoạn văn, RNN sẽ làm tốt với những câu, đoạn ngắn (thông tin sẽ qua ít layers), tuy nhiên sẽ gặp khó khăn nếu input là một đoạn văn dài, cần thông tin từ đầu đến cuối để dự đoán → cần một mô hình có thể giải quyết “long term”

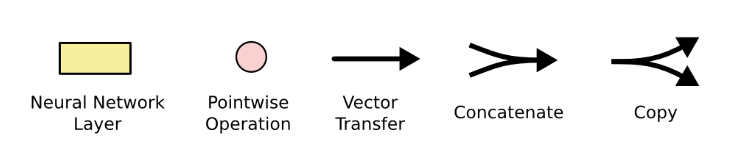
**→ Long Short Term Memory (LSTM)** ra đời để giải quyết hạn chế này.

##### ***3. Mô hình và nguyên tắc làm việc của LSTM.***



Mô hình LSTM rất gần với kiến trúc của một mạng RNN tiêu chuẩn, tuy nhiên phần kiến trúc lặp lại khá khác biệt

Thay vì một tầng đơn, chuỗi lặp lại có 4 tầng ẩn gồm 3 sigmoid và 1 tanh (như hình) tương tác với nhau theo một cấu trúc đặc biệt.



Trong sở đồ tính toán trên, mỗi một phép tính sẽ triển khai trên một véc tơ.

Trong đó hình tròn màu hồng biểu diễn một toán tử đối với véc tơ như phép cộng véc tơ, phép nhân vô hướng các véc tơ.

Màu vàng thể hiện hàm activation mà mạng nơ ron sử dụng để học trong tầng ẩn, thông thường là các hàm phi tuyến sigmoid và tanh.

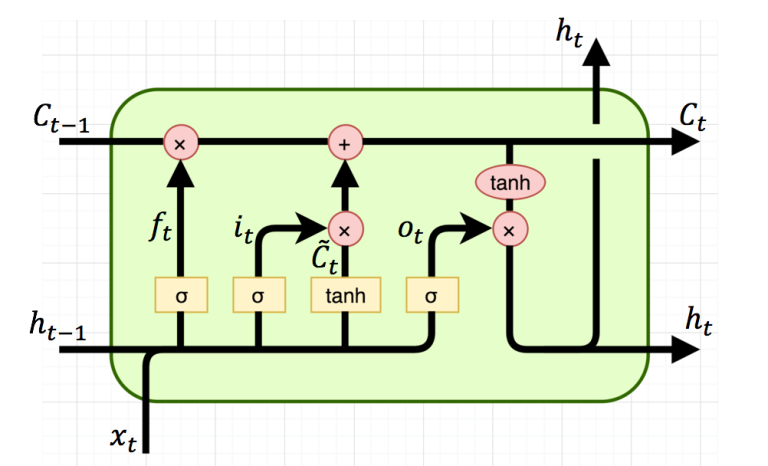
Kí hiệu 2 đường thẳng nhập vào thể hiện phép chập kết quả trong khi kí hiệu 2 đường thẳng rẽ nhánh thể hiện cho nội dung véc tơ trước đó được sao chép để đi tới một phần khác của mạng nơ ron.

##### **4. Trách nhiệm của các gate trong LSTM**

Một đơn vị LSTM chung bao gồm một **cell**, một **input gate**, một **output gate** và một **forget gate**.

* Cell ghi nhớ các giá trị trong khoảng thời gian tùy ý và ba gate điều chỉnh luồng thông tin input và output.
* LSTM rất phù hợp để classify, process, và predict có khoảng thời gian không xác định.

\*\*Xét ví dụ một state thứ t của mô hình LSTM:



Output: - c là cell state, h là hidden state

Input: - với là input ở state thứ t của model.

là output của layer trước. h đóng vai trò khá giống s ở mô hình RNN, c là điểm mới của LSTM.

* Ở biểu đồ trên: σ, tanh tương ứng với hàm sigmoid, tanh activation function. phép nhân là [element-wise multiplication](http://nttuan8.com/bai-1:-linear-regression-va-gradient-descent/#Element-wise_multiplication_matrix), phép cộng là cộng ma trận.
* tương ứng với forget gate, input gate, output gate.
* **Forget gate:** Nó khám phá các chi tiết cần loại bỏ khỏi khối. Một hàm sigmoid quyết định nó. Nó xem xét trạng thái trước đó và đầu vào nội dung và xuất ra một số giữa 0 (bỏ qua điều này) và 1 (giữ nguyên điều này) cho mỗi số trong trạng thái ô .

* **Input gate:** Nó phát hiện ra giá trị nào từ đầu vào sẽ được sử dụng để sửa đổi bộ nhớ. Hàm Sigmoid quyết định giá trị nào sẽ cho qua 0 hoặc 1. Và hàm tanh đưa ra trọng số cho các giá trị được truyền, quyết định mức độ quan trọng của chúng trong khoảng từ -1 đến 1
* **Output gate:** Đầu vào và bộ nhớ của khối được sử dụng để quyết định đầu ra. Hàm Sigmoid quyết định giá trị nào cho qua 0 hoặc 1. Và hàm tanh quyết định giá trị nào cho qua 0, 1. Và hàm tanh đưa ra trọng số cho các giá trị được truyền, quyết định mức độ quan trọng của chúng trong khoảng từ -1 đến 1 và nhân lên với đầu ra là sigmoid.
* Nhận xét thấy là các hệ số bias, hệ số W,U giống như RNN

)

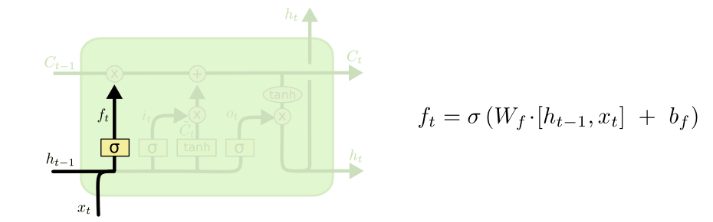
→ **forget gate** quyết định xem cần lấy bao nhiêu từ cell state trước và **input gate** sẽ quyết định lấy bao nhiêu từ input của state và hidden layer của layer trước

→ **output gate** sẽ quyết định xem cần lấy bao nhiêu từ cell state để trở thành output của hidden state. Ngoài ra cũng được dùng để tính ra output cho state t.

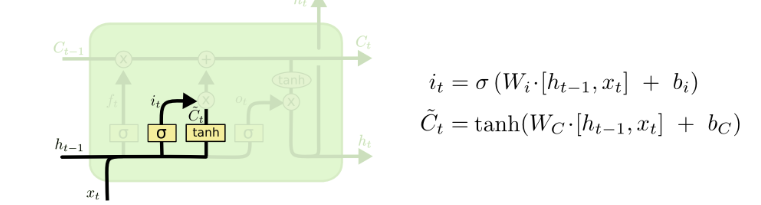
**Chi tiết thứ tự các bước LSTM**

* Bước đầu tiên trong LSTM sẽ quyết định xem thông tin nào chúng ta sẽ cho phép đi qua ô trạng thái (cell state)
* Nó được kiểm soát bởi hàm sigmoid trong một tầng gọi là tầng quên (*forget gate layer*). Đầu tiên nó nhận đầu vào là 2 giá trị và và trả về một giá trị nằm trong khoảng 0 và 1 cho mỗi giá trị của ô trạng thái
* Nếu giá trị bằng 1 thể hiện ‘giữ toàn bộ thông tin’ và bằng 0 thể hiện ‘bỏ qua toàn bộ chúng’.

Bước 1: Cập nhật thông tin mới hoặc lưu giữ giá trị thông tin cũ theo thời gian



Bước 2: Quyết định loại thông tin sẽ được lưu trữ trong ô trạng thái.

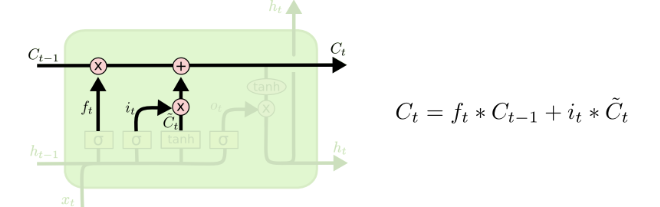


Bước này gồm 2 phần nhỏ.

Phần đầu tiên là tầng ẩn sigmoid được gọi là tầng cổng vào quyết định giá trị bao nhiêu sẽ được cập nhật

Phần thứ hai là tầng ẩn tanh sẽ tạo ra một vectơ trạng thái mới có thể được thêm vào trạng thái

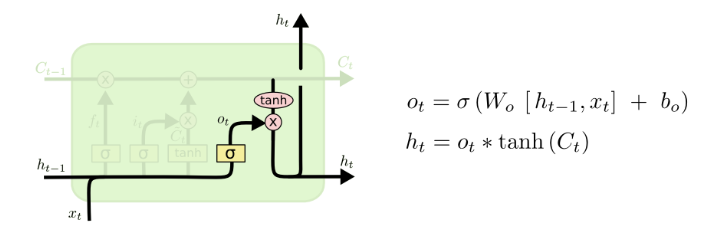
Bước 3: Cập nhật lại ô trạng thái cũ sang trạng thái mới ( )



Ở 2 bước trước đó đã xác định được thông tin cần update và loại thông tin đó, ở bước này sẽ thực hiện.

tương ứng với những thứ được phép quên ở trạng thái cũ và là giá trị được tính toán tương ứng với bao nhiêu được cập nhật vào mỗi giá trị trạng thái

Bước 4: Quyết định xem đầu ra trả lại bao nhiêu.



Kết quả đầu ra sẽ dựa vào ô trạng thái, nhưng là kết quả sau khi lọc.

Đầu tiên thông tin sẽ chạy ra một tầng sigmoid nơi quyết định phần nào của ô trạng thái sẽ ở đầu ra.

Sau đó, ô trạng thái được đưa qua hàm tanh ( trả về giá trị -1 và 1) và nhân nó với đầu ra của cổng sigmoid lúc nãy, do đó chỉ trả ra phần mà chúng ta quyết định.

**Cách LSTM chống vanishing gradient**

* Ta cũng áp dụng thuật toán [back propagation through time](https://nttuan8.com/bai-13-recurrent-neural-network/#Backpropagation_Through_Time_BPTT) cho LSTM tương tự như RNN.
* Thành phần chính gây là vanishing gradient trong RNN là với s,W < 1
* Tương tự như vậy, trong LSTM ta quan tâm đến . Mà do nên về cơ bản LSTM vẫn bị vanishing gradient, nhưng ít hơn RNN.
* Hơn nữa, khi mang thông tin trên cell state, LSTM ít khi cần quên giá trị cũ →

→ tránh được vanishing gradient.

→ LSTM được sử dụng phổ biến hơn RNN cho các bài toán thông tin dạng chuỗi.

#### **II. Số lượng tham số cần học**

<https://www.kaggle.com/code/kmkarakaya/lstm-understanding-the-number-of-parameters>

Dựa theo nguồn này, nếu mô hình LSTM có

* 2 weight matrix: W và U cho và
* 1 bias vectơ b

Chúng ta có thể tính toán dựa trên forget gate



Ta giả sử size của W,U và b là :

* Input size ( và )
* Output size ()

Vì output có size bằng với hidden state (hx1)

* W :
* U :
* Bias param : h
* Tổng tham số = W param + U param + Bias param

=

=

Do có 4 hàm được định nghĩa giống nhau nên

Tổng tham số của LSTM =

#### [Author 2] HÙNG - Phần hiện thực

[Phần 4 - LSTM.ipynb - Colaboratory.pdf](https://drive.google.com/file/d/12x2wP0YZQFik4cvZz8MBmVY2Mi4_CTA4/view?usp=drive_link)