Giảng viên hướng dẫn: Thầy Lê Anh Cường

Sinh viên thực hiện: Hoàng Kiến Thiết - 51702187

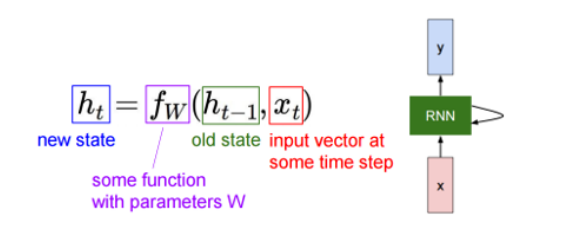
Nguyễn Trọng Nhật - 51702149

Recurrent Neural Networks

# I. Tổng quan về RNN

## 1.1 Định nghĩa RNN

Recurrent neural network (RNN) là một lớp mạng neural nhân tạo có các kết nối giữa các units tạo thành một chu trình có hướng (directed cycle). So với mạng neural truyền thẳng, RNN có thể sử dụng bộ nhớ trong của nó để xử lý các chuỗi đầu vào tùy ý. Chính nhờ đặc trưng này làm cho RNN có thể áp dụng cho các tác vụ như nhận dạng chữ viết tay hoặc nhận dạng giọng nói.

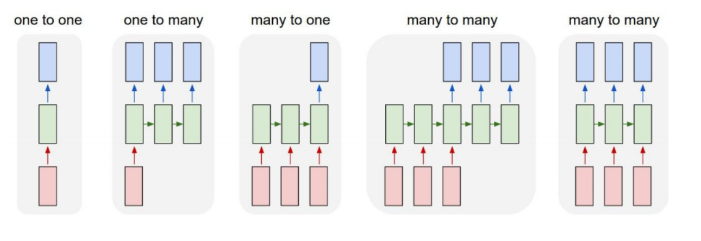


Hình 1. RNN

Thông thường ta sẽ có RNN có độ dài tối đa là 25 vì ta không thể có RNN có độ dài lớn vì ta phải lưu trữ trong bộ nhớ trạng thái ẩn (hidden state) sau mỗi time steps để có thể lan truyền ngược. Để xử lý các chuỗi input (sequences) có độ dài lớn hơn, chúng ta sẽ chia chuỗi input thành 25 phần và trạng thái ẩn cuối cùng của một đoạn là trạng thái ẩn ban đầu của đoạn tiếp theo.

## 1.2 Phân loại bài toán với RNN

Các dạng bài toán có thể áp dụng RNN



Hình 2. Các bài toán với RNN

* One to one: mẫu bài toán cho Neural Network (NN) và Convolutional Neural Network (CNN), 1 input và 1 output, ví dụ với CNN input là ảnh và output là ảnh được segment.
* One to many: bài toán có 1 input nhưng nhiều output, ví dụ: bài toán caption cho ảnh, input là 1 ảnh nhưng output là nhiều chữ mô tả cho ảnh đấy, dưới dạng một câu.
* Many to one: bài toán có nhiều input nhưng chỉ có 1 output, ví dụ bài toán phân loại hành động trong video, input là nhiều ảnh (frame) tách ra từ video, ouptut là hành động trong video
* Many to many: bài toán có nhiều input và nhiều output, ví dụ bài toán dịch từ tiếng anh sang tiếng việt, input là 1 câu gồm nhiều chữ: “I love Vietnam” và output cũng là 1 câu gồm nhiều chữ “Tôi yêu Việt Nam”.

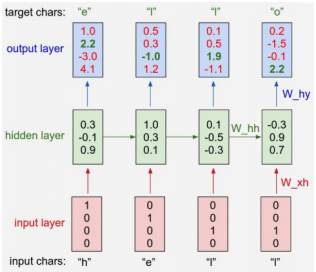
## 1.3 Ví dụ minh họa

Em lấy ví dụ với bài toán thuộc loại Many to many trong RNN. Bài toán yêu cầu tạo một RNN đưa vào một chuỗi các ký tự và dự đoán ký tự tiếp theo. Đầu ra sẽ là kết quả của mỗi chữ cái trong từ vựng là ký tự tiếp theo:

• Chuỗi input: [h, l, e, o]

• Kết quả dự đoán: 'hello’

Hình dưới đây mô phỏng cách RNN giải quyết vấn đề này:

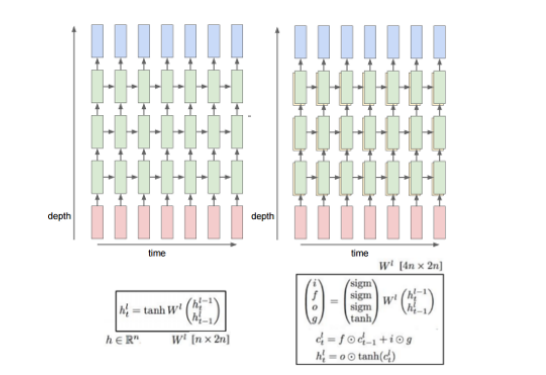


Hình 3. Ví dụ minh họa RNN

# II. Tổng quan về LSTM

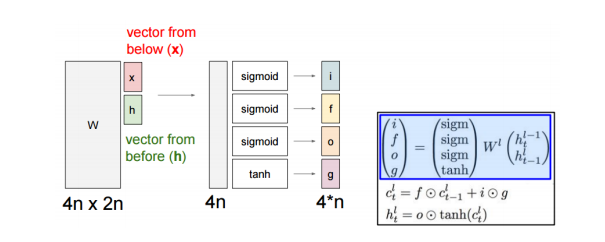
## 1.1 Định nghĩa

Long Short Term Memory (LSTM) là một mạng neural cải tiến từ RNN. Thông thường chúng ta sẽ không sử dụng RNN mà thay vào đó sử dụng LSTM. Hai mạng neural này rất giống nhau, nó vẫn tính đến đầu vào và trạng thái cuối cùng nhưng sự kết hợp của cả hai phức tạp hơn và hoạt động tốt hơn. Với RNN, chúng ta có một vectơ h tại mỗi time steps nhưng với LSTM, chúng ta có hai vectơ tại mỗi time steps là và



## 1.2 Cách thức hoạt động

Sơ đồ dưới đây mô tả cách thức hoạt động của LSTM:



Trong công thức trên có các ký hiệu:

• *c* : cells

• *h* : hidden states

• : vectơ từ tầng dưới có kích thước n × 1 ( là *x* trong hình trên)

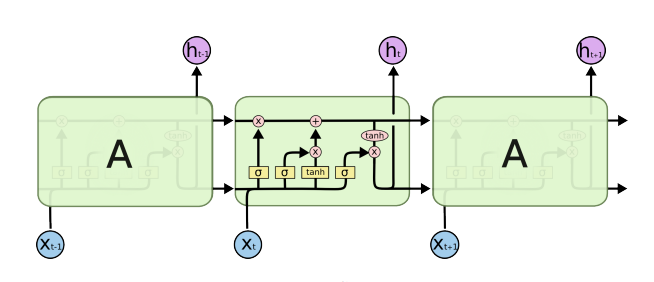
• : vectơ từ trước có kích thước n × 1 (là *h* trong hình trên)

• *i* ∈ [ 0, 1] để chọn xem chúng ta có muốn thêm vào một cell hay không

• *f* ∈ [0, 1] : cổng forget dùng để reset các cells về 0

• *o* ∈ [0, 1]: dùng để chọn cell nào được sử dụng làm output

• *g* ∈ [−1, 1] để thêm −1 hoặc 1 vào các cell



Hình 4. Sơ đồ LSTM

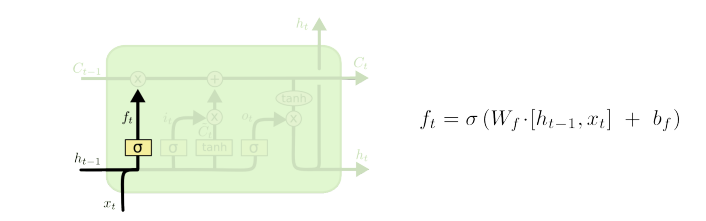
Trong LSTM có các cell state, nó chạy thẳng xuống toàn bộ chuỗi, chỉ với một số tương tác tuyến tính nhỏ và thông tin sẽ trôi theo nó mà không bị thay đổi. LSTM có khả năng loại bỏ hoặc thêm thông tin vào cell state và được điều chỉnh cẩn thận bởi các cấu trúc được gọi là gates. Ở công thức trên, ta coi *i, f, o* là các biến boolean. Các biến này là kết quả của một hàm sigmoid dùng để phân biệt *i, f, o* với nhau. Mục đích sử dụng các biến này là để thiết lập lại và thêm vào bộ đếm cũng như chọn cell nào nên được sử dụng để cập nhật . Chúng ta có thể thực hiện hai thao tác với các cell như sau:

* Reset cells bằng *f* ʘ
* Thêm -1 hoặc 1 bằng  *i* ʘ *g*
* Chọn cells nào là tốt nhất bằng cách cập nhật theo công thức: *o* ʘ *tanh()*

## 1.3 Các bước cụ thể

### 1.3.1 Bước 1 - Quyết định thông tin nào sẽ được loại bỏ khỏi cells state

Quyết định thông tin nào sẽ được loại bỏ khỏi cells state. Quyết định này được thực hiện bởi một sigmoid layer được gọi là forget gate layer. Nó xem xét giữa và sau đó cho ra một output là một số từ 0 đến 1 cho mỗi số ở cell state . Số 1 đại diện cho việc quyết định giữ lại thông tin này trong khi số 0 đại diện cho việc loại bỏ thông tin đó đi.

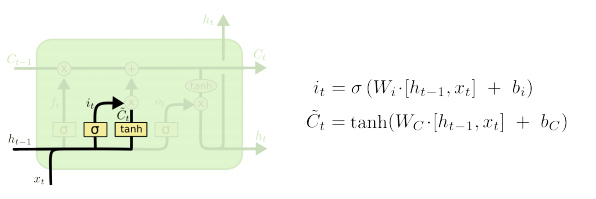


Ví dụ trong một mô hình về ngôn ngữ tập trun dự đoán từ tiếp theo dựa trên tất cả các từ trước đó, cells state khi đó có thể bao gồm giới tính của chủ thể hiện tại, để có thể sử dụng các đại từ chính xác. Khi chúng ta nhìn thấy một chủ đề mới, chúng ta sẽ không cần đến các thông tin như giới tính của chủ thể cũ nữa và có thể quên nó đi

.

### 1.3.2 Bước 2 - Quyết định thông tin nào sẽ được lưu trữ tại cells state

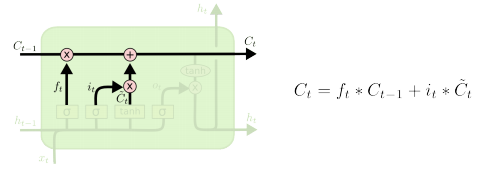
Bước 2 là LSTM là bước quyết định thông tin mới sẽ được lưu trữ trong cells state hay không và bước này được chi làm hai phần.



Đầu tiên, một sigmoid layer được gọi là input gate layers sẽ quyết định giá trị nào được cập nhật tốt. Tiếp theo, một tanh layer tạo ra một vectơ chứa các giá trị ứng viên mới và có thể được thêm vào state. Bước tiếp theo kết hợp hai điều từ phần trước để tạo bản cập nhật cho state.

Ví dụ: trong ví dụ về mô hình dự đoán ngôn ngữ như trên, mô hình LSTM sẽ thêm giới tính của chủ thể mới vào cells state để thay thế giới tính cũ đang bị quên.

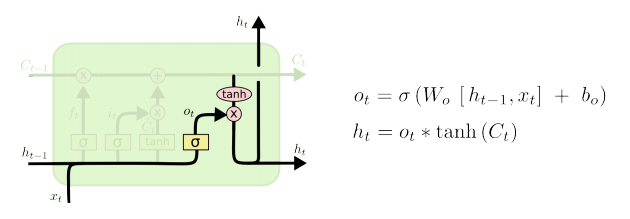
### 1.3.3 Bước 3 – Cập nhật lại cell state



Trong bước 3 này ta sẽ cập nhật cell state cũ () thành cell state mới (. Các bước trước đã quyết định những gì cần làm, chúng ta chỉ cần triển khai các bước đó sau đó nhân trạng thái cũ với vecto , quên đi những thứ mà chúng ta đã quyết định quên trước đó. Sau bước này, ta thêm , đây là các giá trị ứng cử viên mới, được chia tỷ lệ theo mức độ mà mô hình LSTM quyết định cập nhật từng giá trị state.

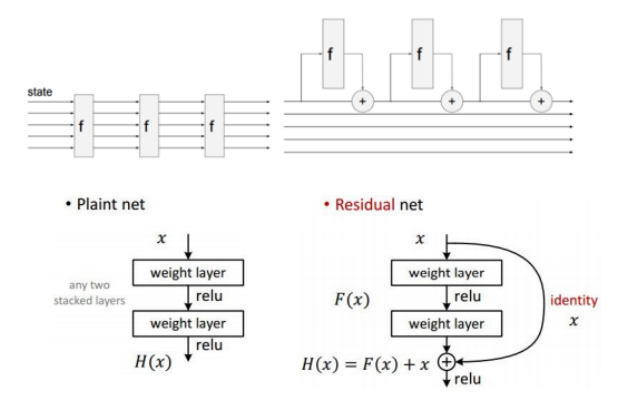
Trong trường hợp ví dụ trên, đây là lúc mô hình dự đoán thực sự loại bỏ thông tin về giới tính của đối tượng cũ và thêm thông tin mới dựa trên các quyết định ở bước trước.

### 1.3.4 Bước 4 – Quyết định output



Bước cuối cùng là trích xuất output cho một cell state. Đầu tiên LSTM model chạy một sigmoid leyer quyết định phần nào của cell state sẽ là output. Sau đó, chúng tôi đặt cell state thông qua tanh layer để đưa các giá trị từ 0 đến 1 và nhân nó với đầu ra của sigmoid gate, mục đích là để ta chỉ xuất các phần mà mô hình đã quyết định tước đó.

# III. So sánh sự khác nhau giữa RNN và LSTM



• RNN transformative các interaction của state trong khi đó LSTM additive chúng.

• Trong RNN, ta đang vận hành và biến đổi vector state, điều đó có nghĩa là ta đang thay đổi hidden state của mình theo từng time steps. Thay vào đó, LSTM dùng các cell state như một tập hợp con của các cell (hoặc tất cả chúng) để tính hidden state. Sau đó, dựa trên hidden state đó để quyết định cách hoạt động trên các cells.

• RNN giống nhưplain nets, LSTMs giống ResNets.

• RNNs có các biến gradient, vì vậy ta không thể tìm hiểu sự tương quan, phụ thuộc giữa các time step cách xa nhau.

# V. Tổng kết

• RNN cho phép linh hoạt nhiều trong thiết kế kiến ​​trúc

• Vanilla RNN rất đơn giản nhưng không hoạt động hiệu quả

• Thường được sử dụng LSTM hoặc GRU: các tương tác phụ của chúng cải thiện gradient flow.

• Backward flow của gradients trong RNN có thể bùng nổ hoặc biến mất. Sự bùng nổ được kiểm soát bằng cách cắt gradients. Sự biến mất được kiểm soát bằng cách thêm các interations (LSTM).

• Các kiến ​​trúc tốt hơn hoặc đơn giản hơn là một chủ đề nóng của nghiên cứu hiện nay.

• Cần hiểu rõ hơn (cả lý thuyết và thực nghiệm)

# Tham khảo

1. Stanford, *CS231n,* [online] Available at: <https://github.com/albertpumarola/deep-learning-notes/blob/master/individual_chapters/recurrent_neural_networks.pdf?fbclid=IwAR0rqM5kcdznirWp6knAa4eD-jYxxNxJEccxrARI5bvUK9shfshE4k9Buc0>
2. colah’s blog*, Understanding LSTM Networks*, *,* [online] Available at: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/?fbclid=IwAR2fXD9Pr_lrsQHAvqjChwxk-PKcafZisxM4oB4HQuRIEeFhJO2y4khU89w>