CHƯƠNG 5
MẠNG NEURAL HỒI QUY (P2)

Khoa Khoa học và Kỹ thuật thông tin
Bộ môn Khoa học dữ liệu

NỘI DUNG

1. Các loại language model.
2. Hạn chế của RNN: Vanishing gradients
3. Các mô RNN.
4. Deep RNN network.

Các loại language model

Language model

– Có 2 dạng language thường gặp:

+ Character-level language model (tạm dịch: mô hình hướng ký tự).

+ Word-level language model (tạm dịch: mô hình hướng từ).

Ví dụ

Character-level model

- Vocab = {cat, sleep, a, day, ...}.
- Biểu diễn câu:

cats sleep 15 hour per day

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$ $x^{<6>}$

Word-level model

- Vocab = {a, b, c, d, ...}.
- Biểu diễn câu:

c a t s s l e e p 1 5 h o u r p e r d a y

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$ $x^{<6>}$ .....

Một số nhận xét

- Trong các bài toán về dependencies, sử dụng character-level model sẽ không hiệu quả.
  + Nếu dữ liệu quá lớn thì việc biểu diễn tốn rất nhiều chi phí tính toán.
- Trong xử lý ngôn ngữ tự nhiên, mô hình word-level thường được dùng.
  + Tuy nhiên, cần lưu ý khái niệm word (từ) đối với các ngôn ngữ khác nhau.
  + Nhận diện word (từ) trong câu: bài toán tokenization (tách từ).

Ví dụ về từ

Tiếng anh

- Sentence: Cats sleep 15 hours per day.

Vocab = {

cats, sleeps, 15,

hours, per, days

}

Tiếng Việt

- Câu: những con mèo ngủ 15 giờ đồng hồ một ngày.

Vocab = {

những, con_mèo,

ngủ, 15, giờ_đồng_hồ,

một_ngày

}

Sự khác biệt giữa tiếng Việt và tiếng Anh

Tiếng Việt

- Ngôn ngữ đơn thể (isolate).
  - + Hình thái giữ nguyên.
  - + Ý nghĩa ngữ pháp nằm ở ngoài từ.
- Phương thức ngữ pháp chủ yếu: trật tự từ và hư từ.
- Ranh giới từ khó nhận diện, không thể dựa vào khoảng trắng hoặc dấu.

VD: học sinh học sinh học.

Tiếng Anh

- Ngôn ngữ biến cách (flexion).
  - + Hình thái thay đổi.
  - + Ý nghĩa ngữ pháp nằm ở trong từ.
- Phương thức ngữ pháp chủ yếu: phụ tố.
- Ranh giới từ được nhận diện bằng khoảng trắng hoặc dấu câu.

VD: he sees me and I saw him.

Đinh Điền, Xử lý ngôn ngữ tự nhiên, NXB ĐHQG.TPHCM (2006)

Hạn chế của mô hình RNN: Vanising gradient

Long-term dependencies

Sentence 1: The cat, which already ate fish, was full.

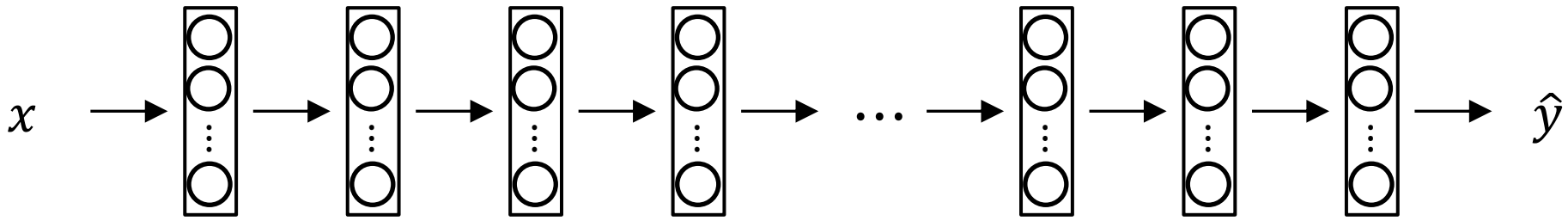
Sentence 2: The cats, which already ate fist, were full.

Trong 2 ví dụ trên, sự xuất hiện của "was" hoặc "were" *phụ thuộc vào sự xuất hiện* của chữ "cat" hoặc "cats" trước đó.

➔ *Long-term dependencies*

Long-term dependencies

The cat which already ..... was full.



- Mô hình RNN truyền thống làm việc không tốt với dữ liệu long-term dependencies như ví dụ trên.
  - Phải xây dựng nhiều layer ➔ **vanishing gradient descent.**

Nhận xét

- Thông thường, mạng RNN sẽ có khoảng từ 1000 layers cho tới 10,000 layer để xử lý cho một sequence model.
  + Trên mạng RNN, rất dễ xảy ra hiện tượng vanishing/exploding gradient descent.
- So với vanishing gradient descent, exploding gradient descent dễ giải quyết hơn nhiều vì có thể dùng kỹ thuật gradient clipping: *sử dụng một ngưỡng threshold để re-scale giá trị trong 1 vector*.
- Tuy nhiên, vanishing gradient descent khó giải quyết hơn.

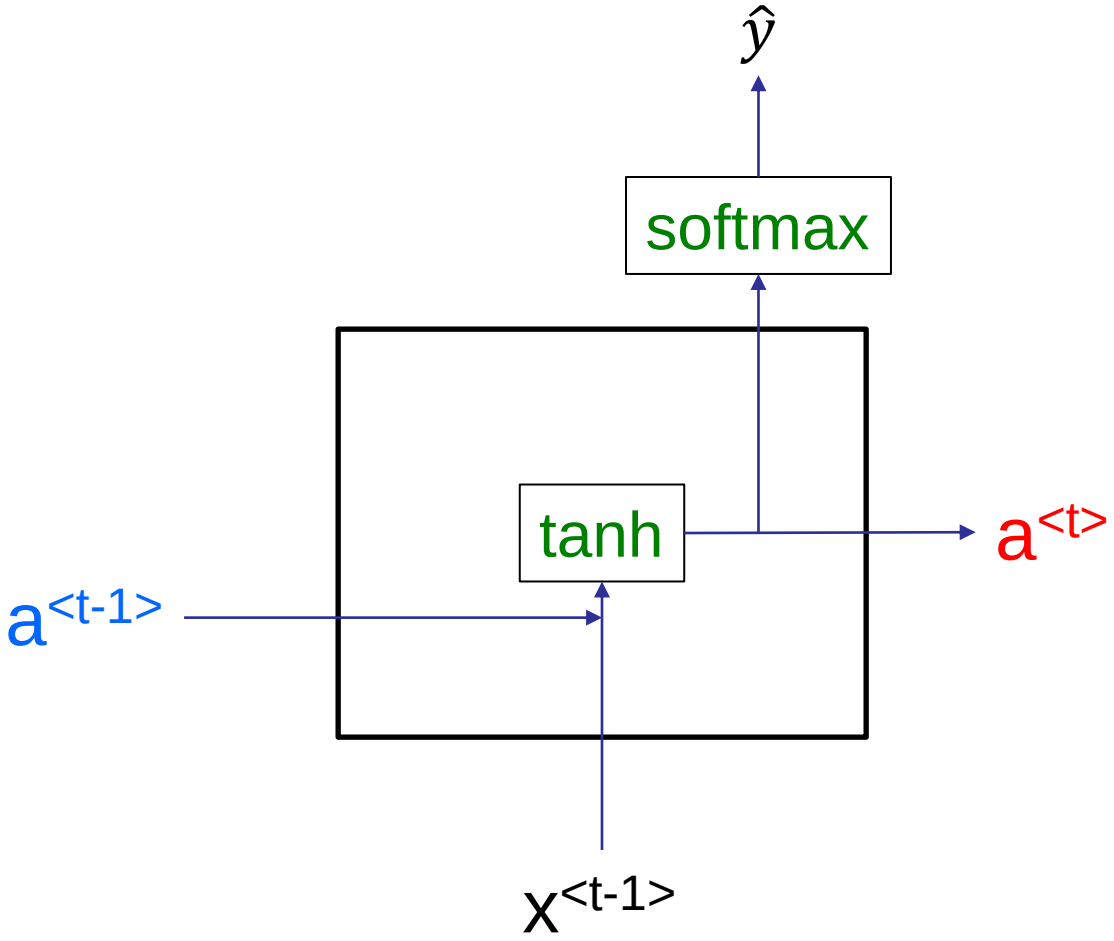
➔ *Cần cải tiến kiến trúc RNN truyền thống.*

Các mô hình RNN

Các mô hình deep RNN

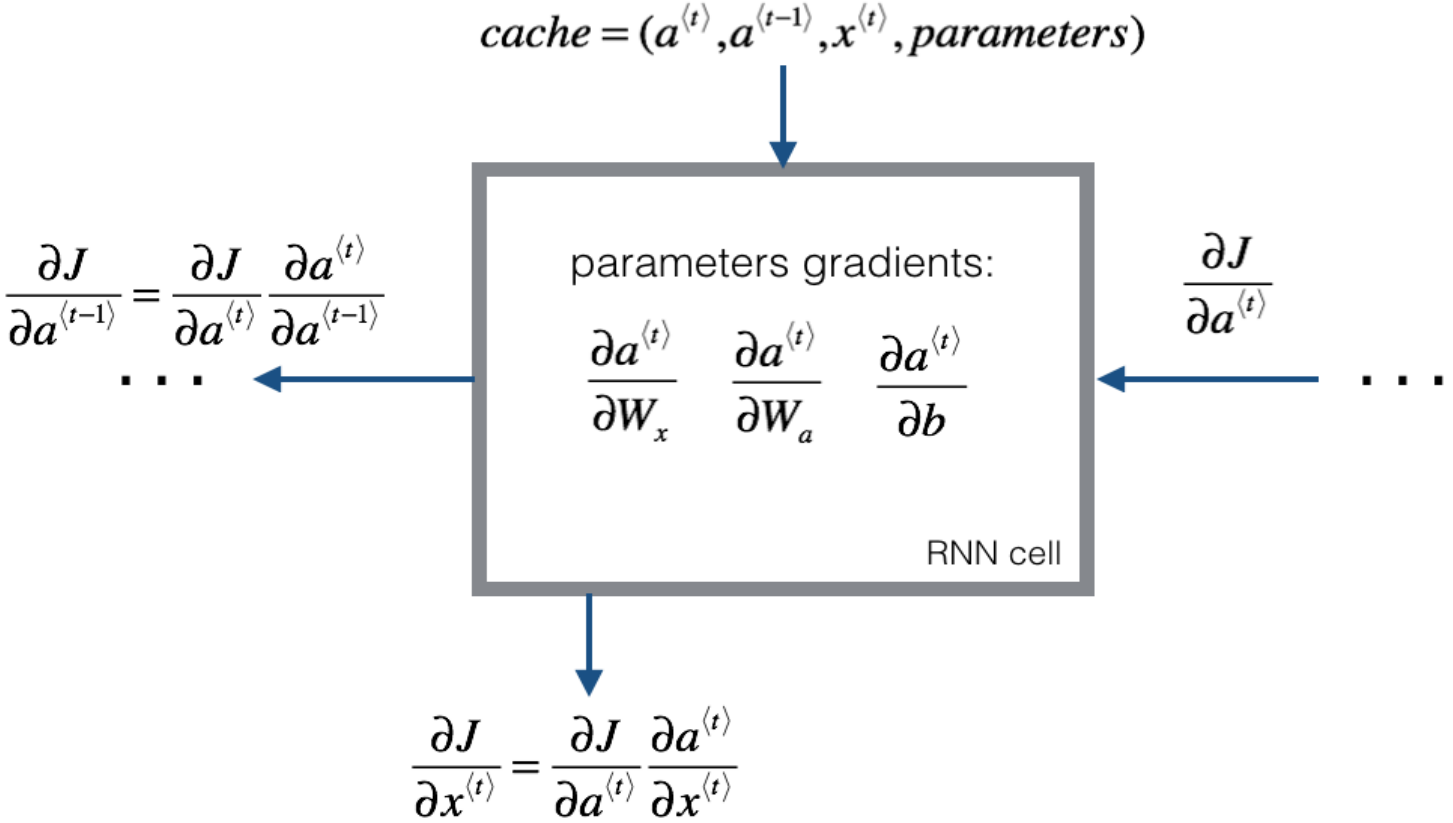
- Gate Recurrent Units (GRU).
- Long-short term memory (LSTM).
- Bi-directional Long-short term memory (Bi-LSTM).

RNN Unit



$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

RNN Unit backward derivation



$$a^{\langle t \rangle} = \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)$$

$$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh(x)^2$$

$$\frac{\partial a^{\langle t \rangle}}{\partial W_{ax}} = (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2) x^{\langle t \rangle T}$$

$$\frac{\partial a^{\langle t \rangle}}{\partial W_{aa}} = (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2) a^{\langle t-1 \rangle T}$$

$$\frac{\partial a^{\langle t \rangle}}{\partial b} = \sum_{batch} (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2)$$

$$\frac{\partial a^{\langle t \rangle}}{\partial x^{\langle t \rangle}} = W_{ax}{}^T . (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2)$$

$$\frac{\partial a^{\langle t \rangle}}{\partial a^{\langle t-1 \rangle}} = W_{aa}{}^T . (1 - \tanh(W_{ax} x^{\langle t-1 \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2)$$

Gate Recurrent Unit – GRU

Dẫn nhập

The cat, which already ate …, was full.

Xét chữ cat: nếu là số ít (singular) thì đặt là 1, nếu là số nhiều (plural) thì đặt là 0. Giá trị 0, 1 được lưu trữ trong biến **c** (tạm đặt).

Chữ was: sẽ giữ nguyên nếu c = 1, ngược lại sẽ là were nếu c = 0.

Gate recurrent unit

– Mô hình GRU đặt ra 2 khái niệm:

+ Memory cell. Ký hiệu là **c**.

+ Gate. Ký hiệu là Γ (đôi khi người ta dùng là **G**).

- Nếu gate = 1 ➔ cập nhật giá trị của c.
- Nếu gate = 0 ➔ bỏ qua.

Ví dụ

$\Gamma_u = 0$ $\Gamma_u = 0$ $\Gamma_u = 0$

The cat, which already ate …, was full.

không cập nhật $c^{<t>}$

$\Gamma_u = 1$ → $\Gamma_u = 1$

$c^{<t>} = 1$ → $c^{<t>} = 1$

$\tilde{c}^{<t>} = tanh(W_c [c^{<t-1>}, x^{<t>}] + b_c)$

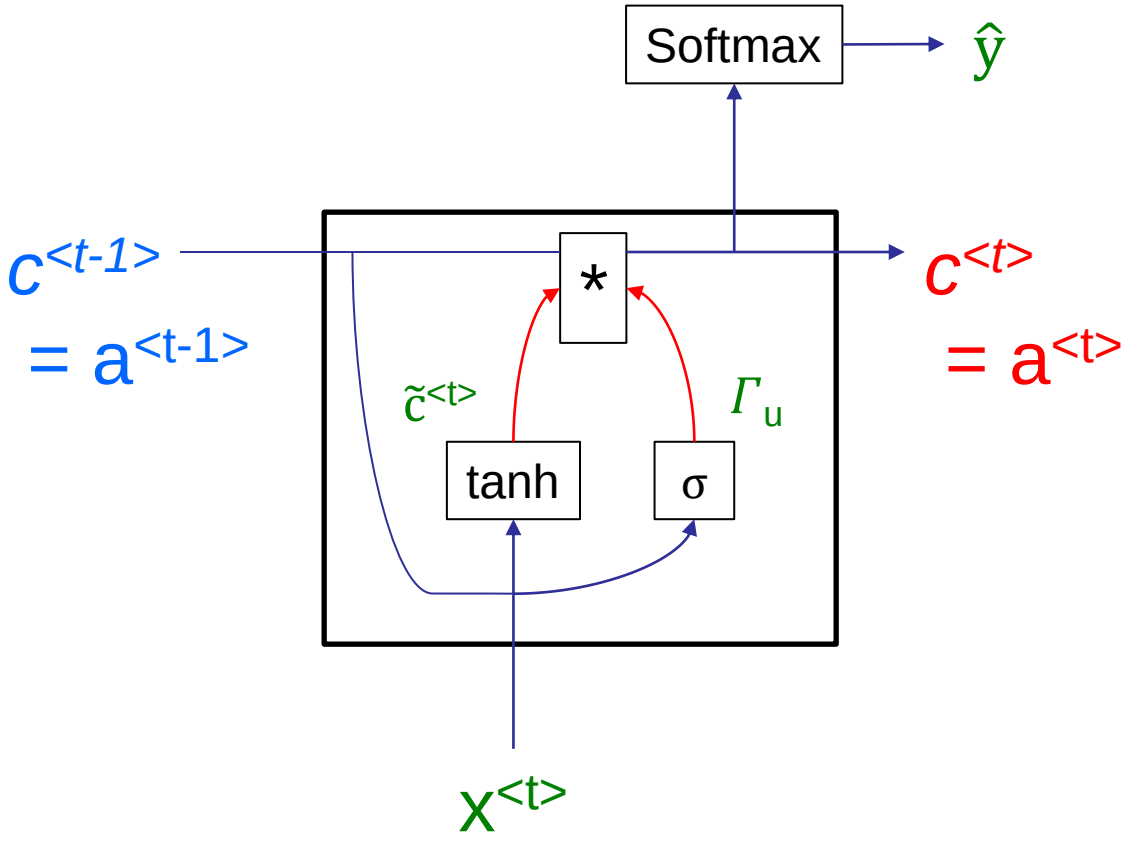
$\Gamma_u = \sigma(W_u [c^{<t-1>}, x^{<t>}] + b_u)$.

$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t - 1>}$ ***(*)***

$c^{<t>} = a^{<t>}$

$\Gamma_u = 0$ ➔ $c^{<t>}$ sẽ bằng $c^{<t-1>}$ *(*)*
không cập nhật $c^{<t>}$ so với giá trị trước đó

Simplified GRU unit

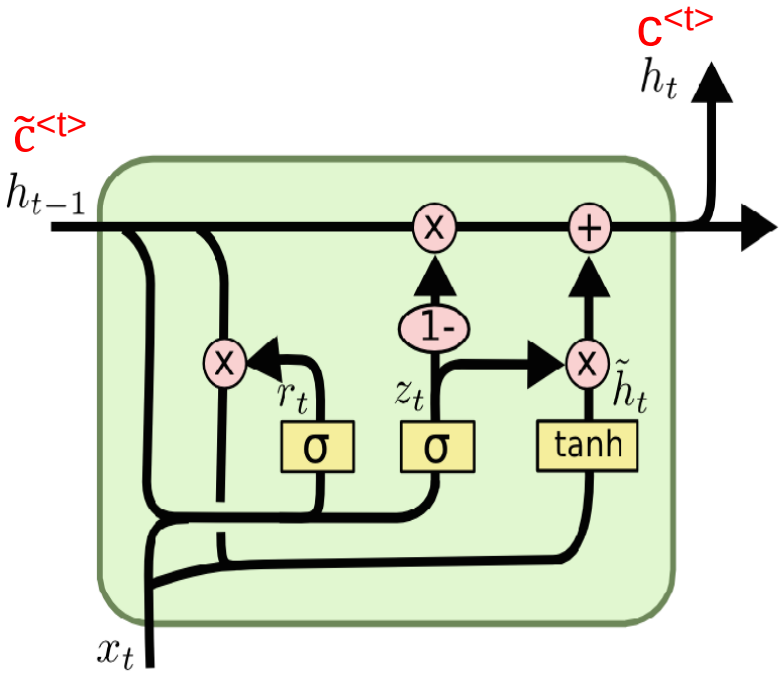


$$\tilde{c}^{<t>} = tanh(W_c [c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u [c^{<t-1>}, x^{<t>}] + b_u).$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>} \textbf{(*)}$$

Full GRU units



Ngoài cổng Γ_u, mô hình GRU còn bổ sung thêm một cổng Ký hiệu là Γ_c, dùng để điều khiển thông tin đầu vào của $c^{<t>}$ là $c^{<t-1>}$

$\tilde{c}^{<t>} = tanh(W_c [\Gamma_c * c^{<t-1>}, x^{<t>}] + b_c)$

$\Gamma_c = \sigma(W_r [c^{<t-1>}, x^{<t>}] + b_r).$

$\Gamma_u = \sigma(W_u [c^{<t-1>}, x^{<t>}] + b_u).$

$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t - 1>}$

Long short term memory

Long-short term memory

- Được phát minh bởi Hochreiter & Schmidhuber vào năm 1996.
- Là một trong các kiến trúc mạng neural hồi quy (RNN) có ảnh hưởng lớn đến lĩnh vực AI nói chung và deep learning nói riêng.

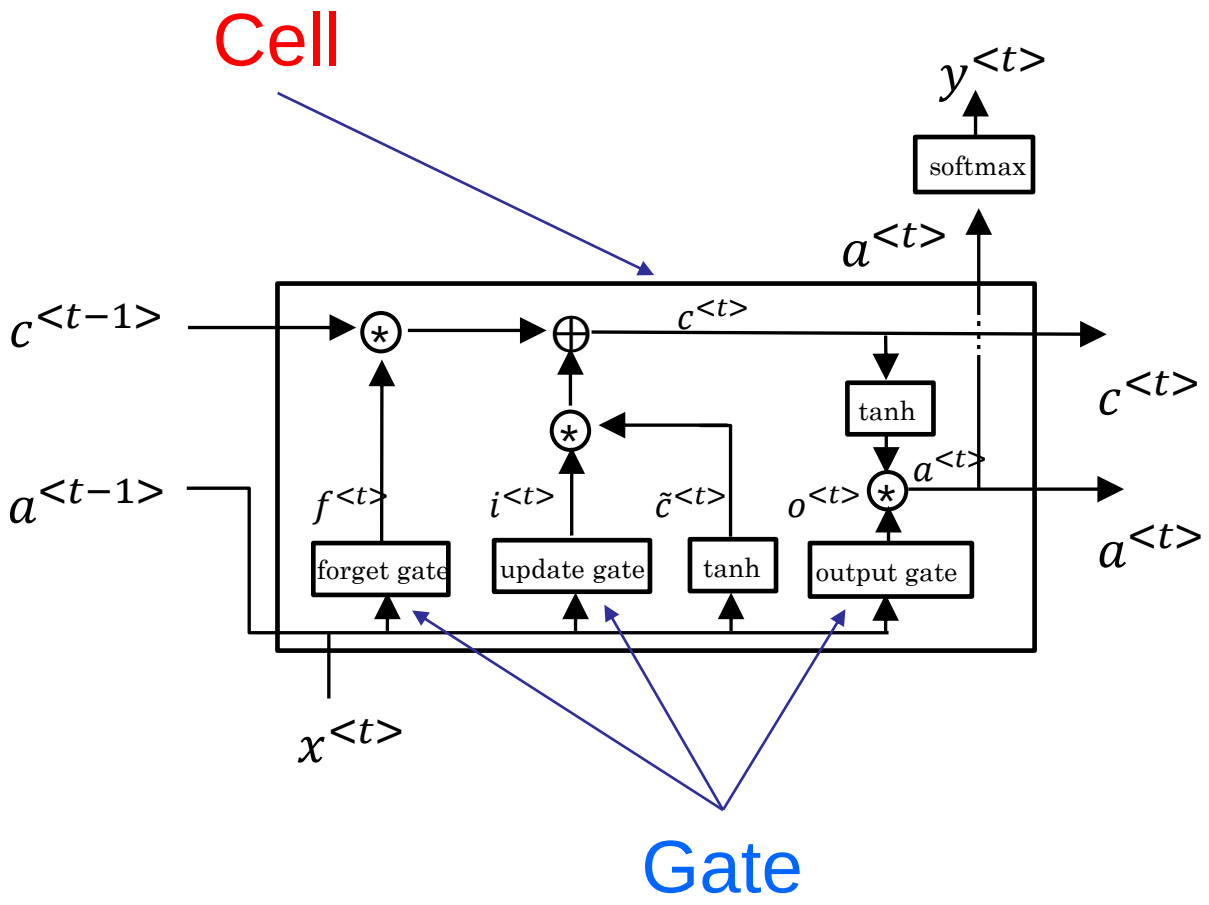
Sepp Hochreiter

Jürgen Schmidhuber

Schmidhuber, Jürgen, and Sepp Hochreiter. "Long short-term memory." *Neural Comput* 9.8 (1997)

Kiến trúc LSTM

- Gồm các thành phần sau:
  - + Các tế bào nhớ dùng để chia sẻ thông tin.
  - + Các cổng (gate) dùng để điều khiển luồng thông tin trong mạng.
- Có 3 dạng cổng:
  - + Update gate.
  - + Output gate.
  - + Forget gate.



Kiến trúc LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

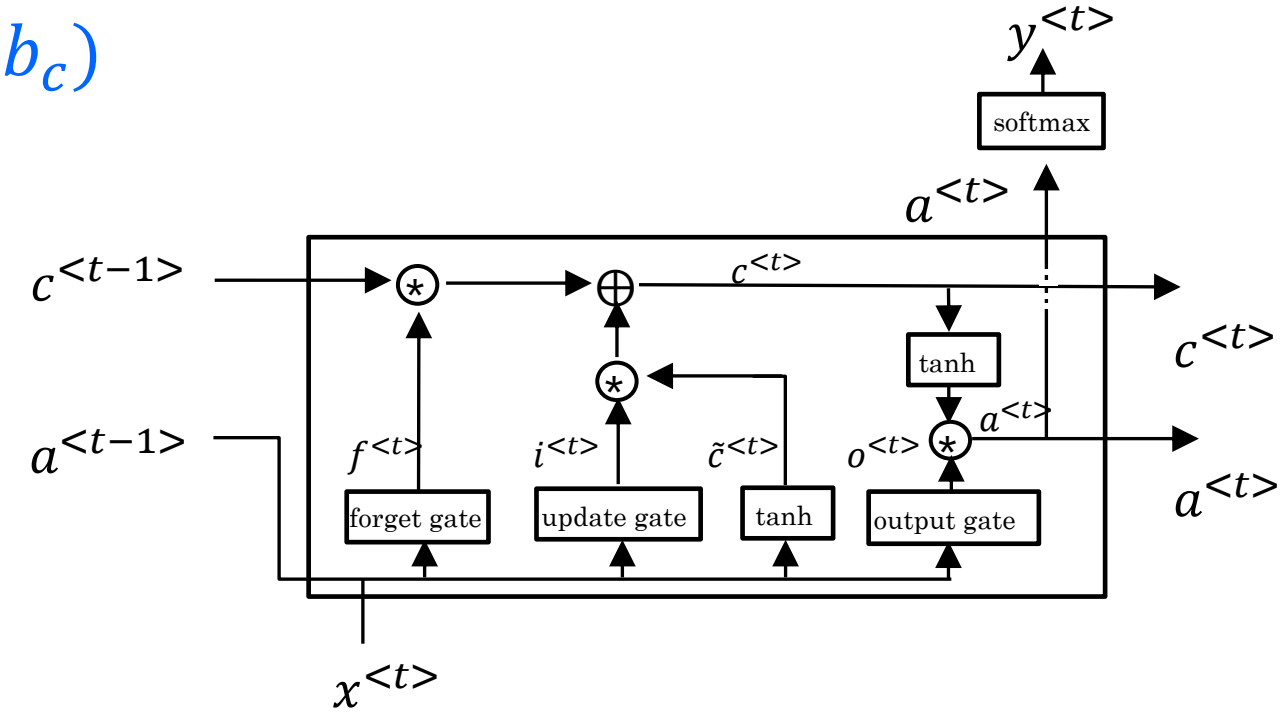
$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * c^{<t>}$$



Chức năng các cổng

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$$

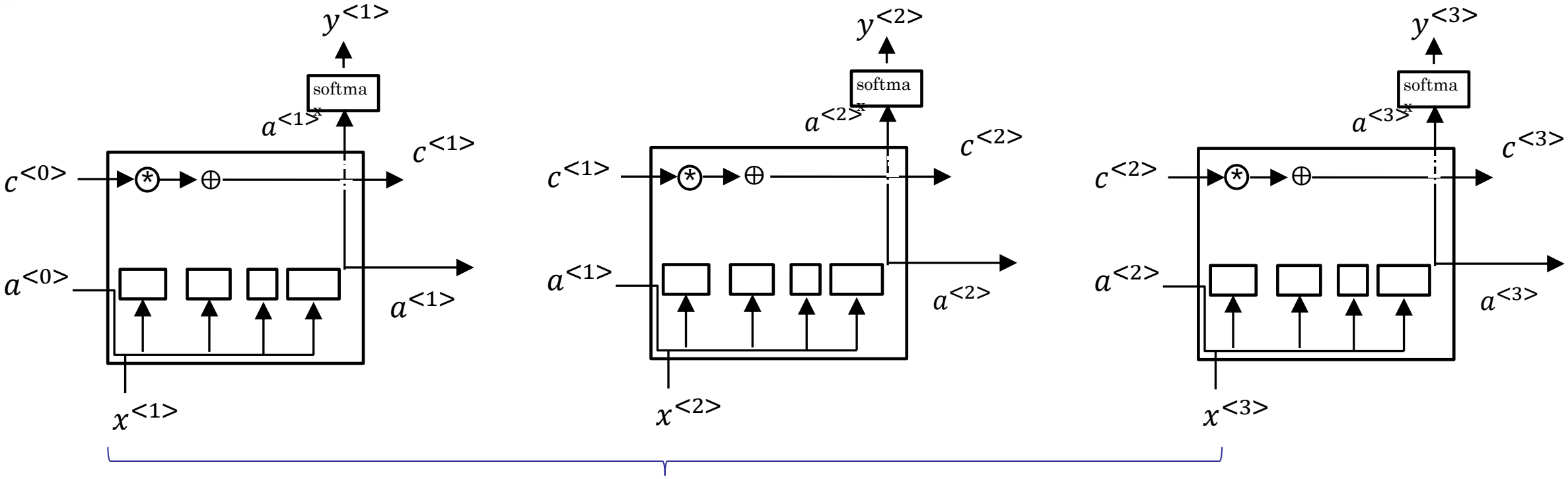
$$\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \boldsymbol{\Gamma_o} * \tanh(\boldsymbol{c}^{<t>})$$

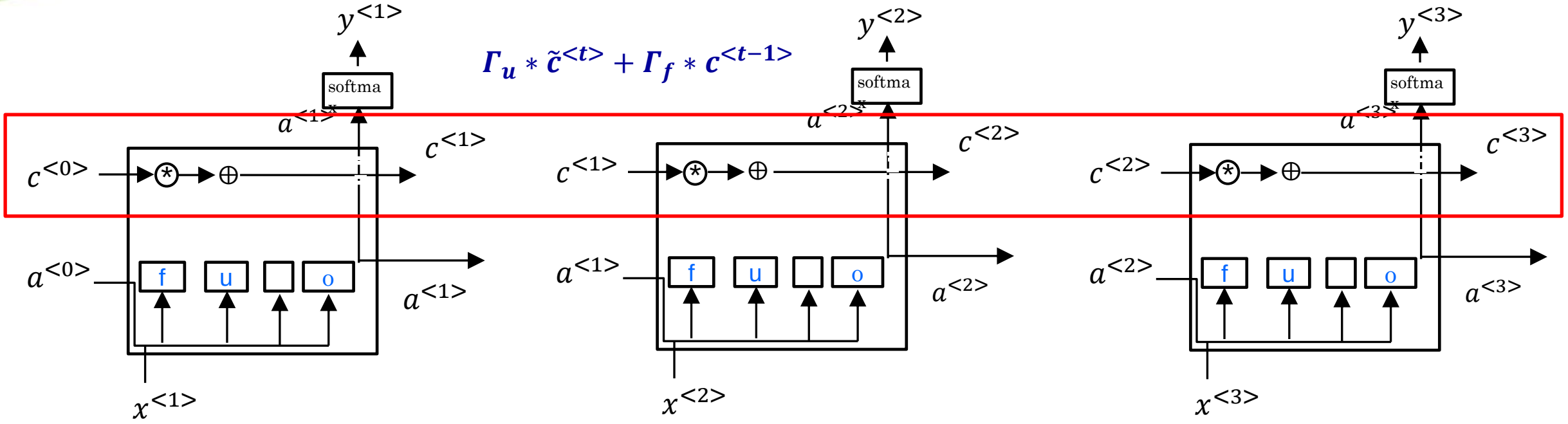
- Update gate: điều khiển việc lấy thông tin của layer (hay unit) trước đó.
- Output gate: quyết định xem lấy bao nhiêu thông tin để xuất ra output.
- Forget gate: quyết định xem thông tin nào cần lấy, và thông tin nào bỏ qua.

Cách hoạt động



Sequence X

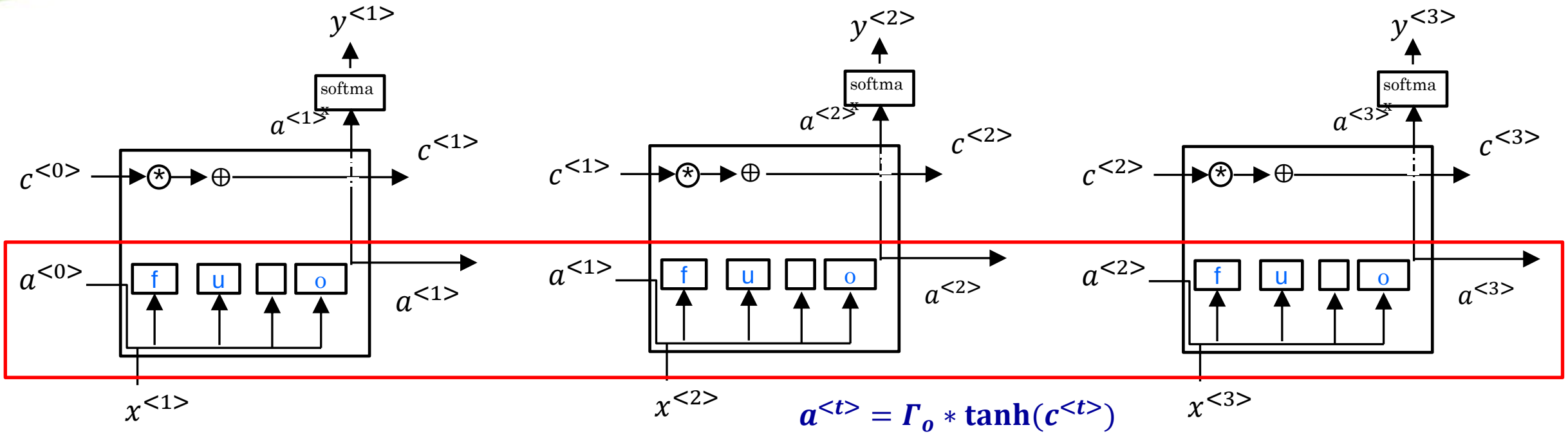
Cách hoạt động



Giá trị của $c^{<0>}$ luôn được giữ nguyên sau khi truyền qua 3 lớp nhờ vào cổng forget và update gate.

➔ *Mô hình LSTM "nhớ" được giá trị của $c^{<0>}$* (long-term memory)

Cách hoạt động



Ngoài ra, mô hình LSTM còn quyết định được thông tin đầu ra ở mỗi unit ra sao nhờ vào cổng output gate.

➔ *Short term memory.*

LSTM vs GRU

LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \boldsymbol{\Gamma_o} * \tanh(\boldsymbol{c}^{<t>})$$

GRU

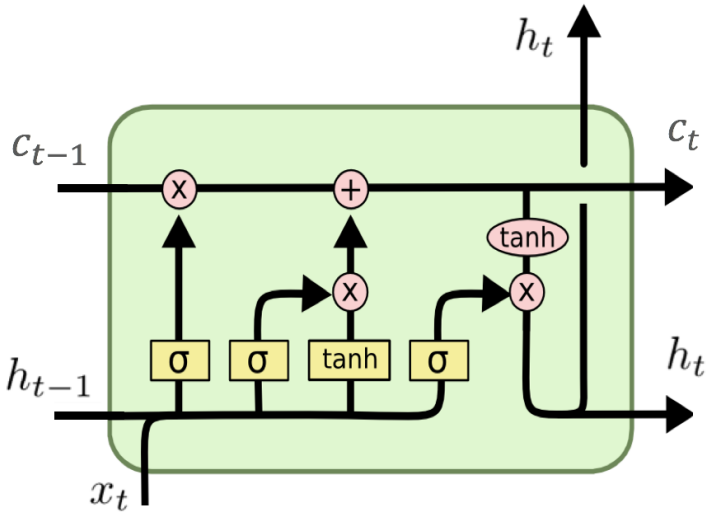
$$\tilde{c}^{<t>} = tanh(W_c\,[\Gamma_c * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_c = \sigma(W_r\,[\, c^{<t-1>}, x^{<t>}] + b_r).$$

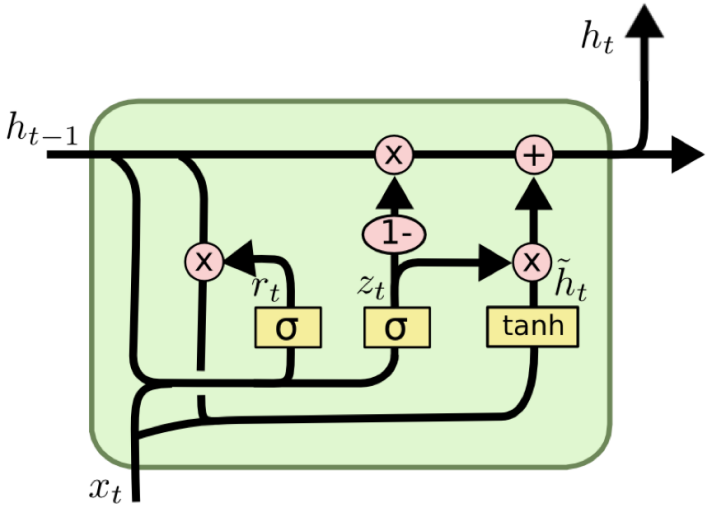
$$\Gamma_u = \sigma(W_u\,[\, c^{<t-1>}, x^{<t>}] + b_u).$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

LSTM vs GRU



LSTM
(Long-Short Term Memory)



GRU
(Gated Recurrent Unit)

Bidirectional LSTM

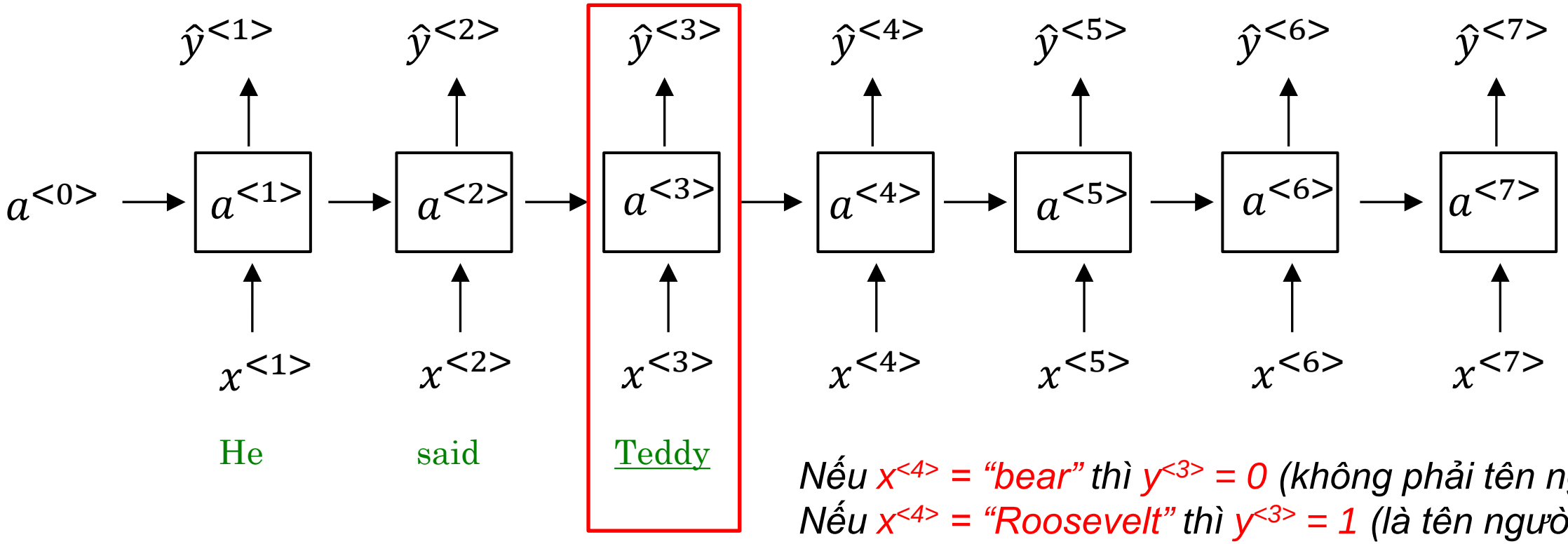
Dẫn nhập

- Sentence 1: He said, "Teddy bears are on sale!"
- Sentence 2: He said, "Teddy Roosevelt was a great President!"

Bài toán đặt ra: Dự đoán xem "Teddy" có phải là tên riêng (person name) hay không ?

Dẫn nhập

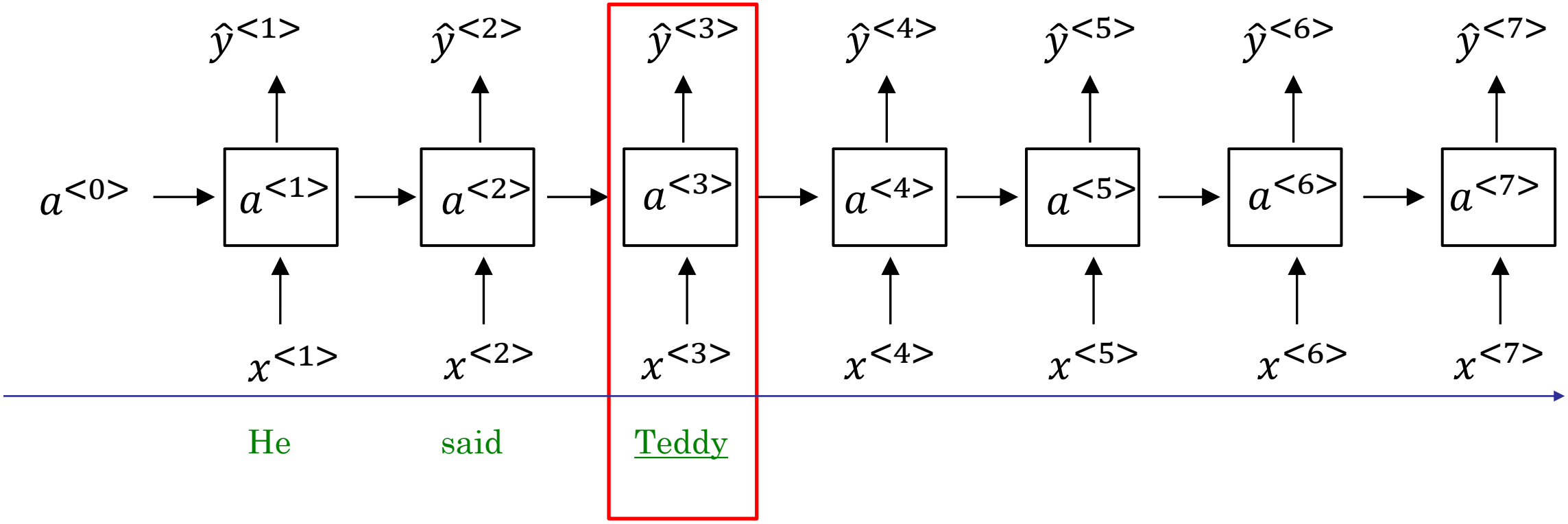
- Sentence 1: He said, "Teddy bears are on sale!"
- Sentence 2: He said, "Teddy Roosevelt was a great President!"



Nếu $x^{<4>}$ = "bear" thì $y^{<3>} = 0$ (không phải tên người)
Nếu $x^{<4>}$ = "Roosevelt" thì $y^{<3>} = 1$ (là tên người)

Điểm yếu của RNN truyền thống

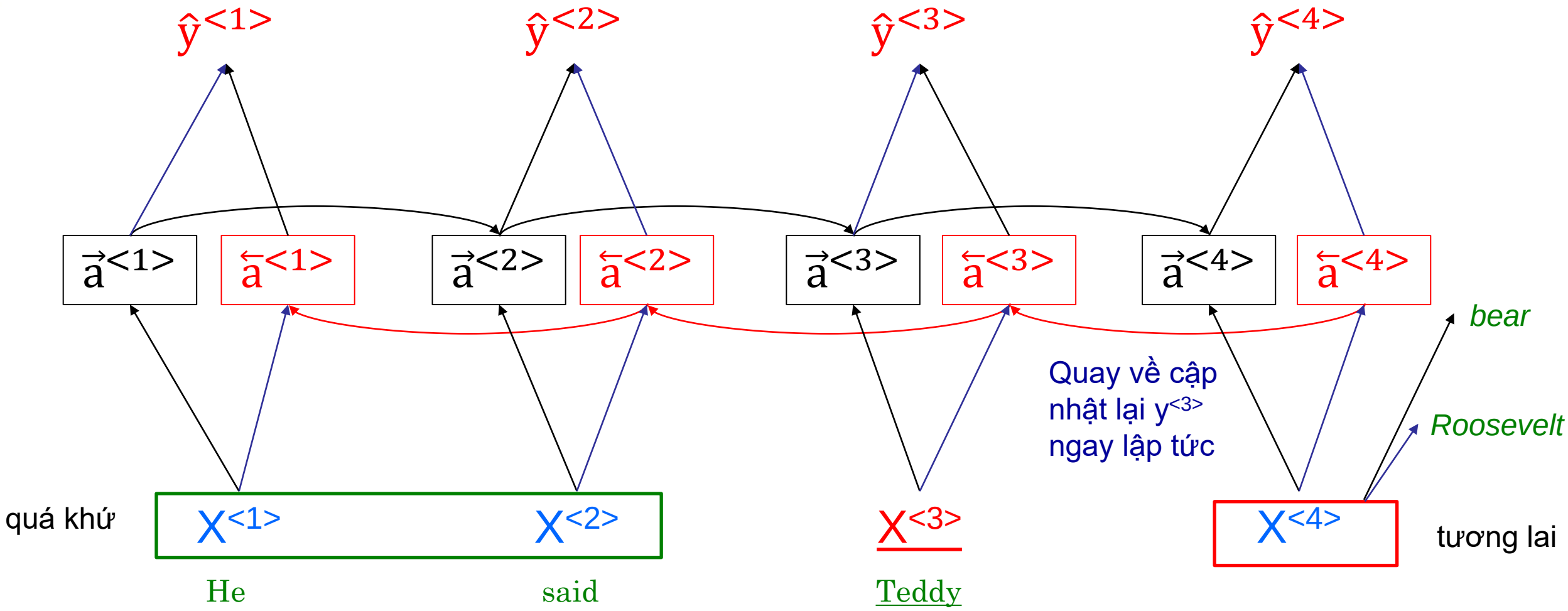
– Muốn cập nhật lại giá trị của $y^{<3>}$ thì phải chờ đến quá trình back-propagation (quay lui) về cập nhật trọng số.



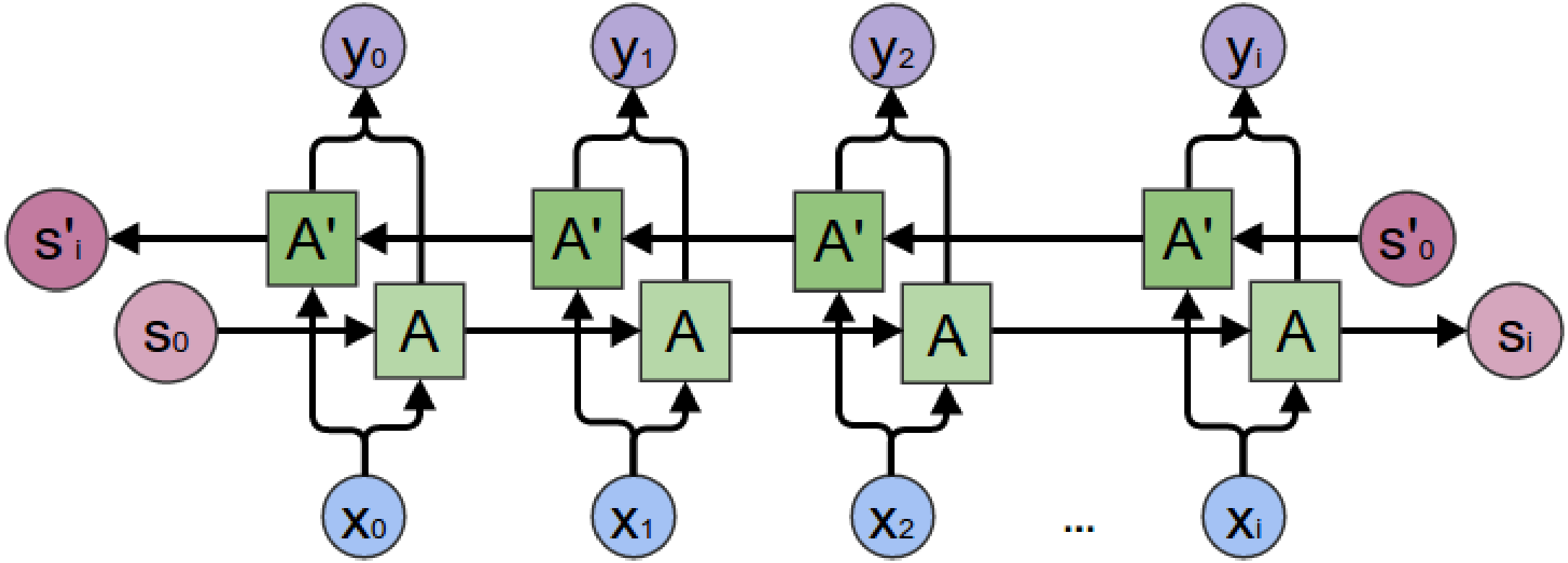
Bidirectional LSTM

- Là một biến thể (variant) của LSTM.
- Bi-LSTM thêm vào một layer đặc biệt gọi là backward recurrent layer. Ký hiệu là: $\overleftarrow{a}^{<i>}$, và kết nối layer đó cùng với forward layer để cho ra output.
- Bằng cách này, một unit của BiLSTM có thể học được các thông tin ở cả quá khứ (trước nó) và tương lai (unit kế tiếp).

Bi-LSTM


$\hat{y}^{<1>}$
$\hat{y}^{<2>}$
$\hat{y}^{<3>}$
$\hat{y}^{<4>}$
$\vec{a}^{<1>}$
$\overleftarrow{a}^{<1>}$
$\vec{a}^{<2>}$
$\overleftarrow{a}^{<2>}$
$\vec{a}^{<3>}$
$\overleftarrow{a}^{<3>}$
$\vec{a}^{<4>}$
$\overleftarrow{a}^{<4>}$
bear
Quay về cập nhật lại $y^{<3>}$ ngay lập tức
Roosevelt
quá khứ
$X^{<1>}$
$X^{<2>}$
$X^{<3>}$
$X^{<4>}$
tương lai
He
said
Teddy

Kiến trúc Bi-LSTM



Nhận xét

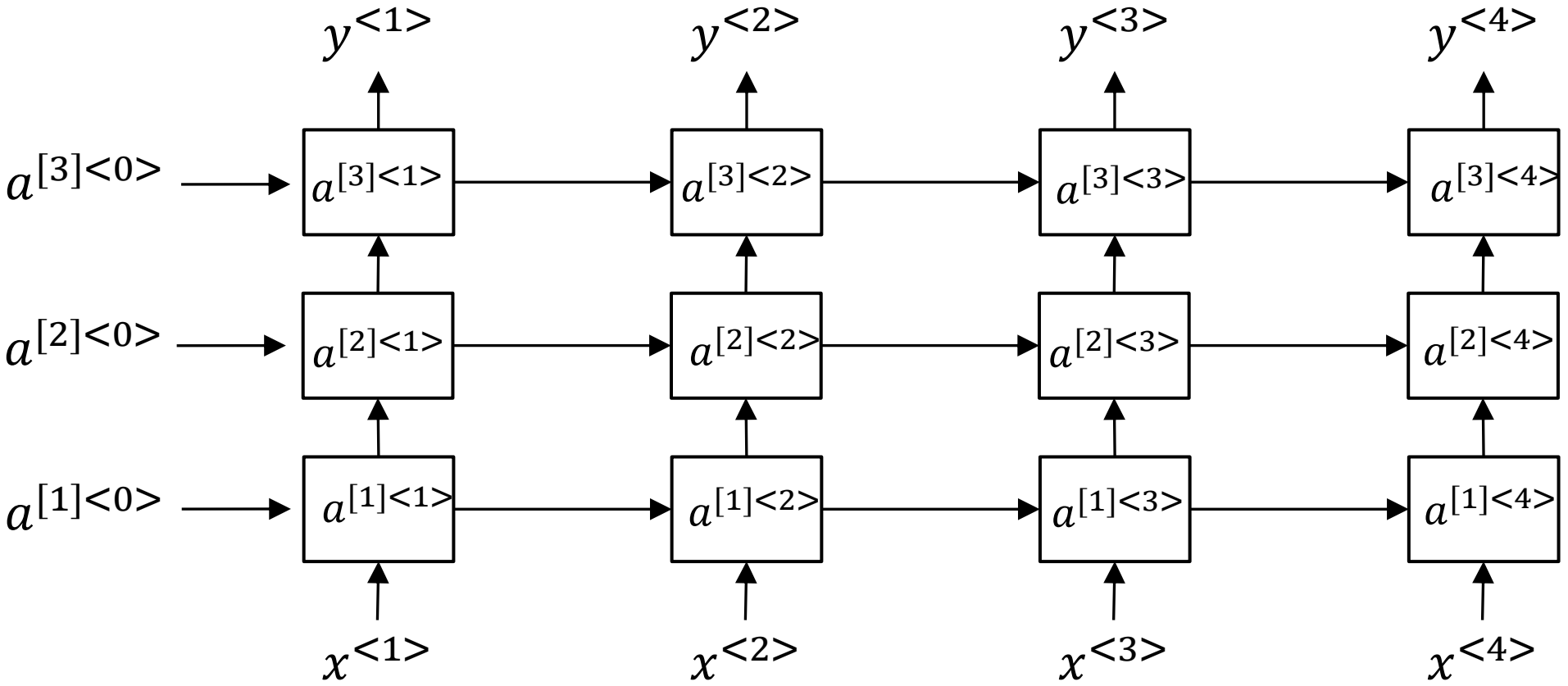
- BiLSTM và LSTM là các mô hình thường được sử dụng trong các bài toán về Xử lý ngôn ngữ tự nhiên (NLP).
- BiLSTM có thể dự đoán được giá trị tại bất cứ vị trí nào trong sequence model, kể cả khi mới tới vị trí giữa (giữa câu).
- Điểm yếu: giá trị input vào phải là một sequence hoàn chỉnh.

Deep RNN network

Kiến trúc Deep RNN

Ký hiệu: $a^{[l]<i>}$

l: lớp thứ l
i: vị trí phần tử thứ i trong sequence



TỔNG KẾT

1. Khái niệm language model, các loại language model.
2. Các dạng RNN thường gặp:
   + GRU.
   + LSTM.
   + BiLSTM.
3. Deep neural model.

TÀI LIỆU THAM KHẢO

1. Khoá học *Neural Network and Deep learning*, deeplearning.ai.
2. Ian Goodfellow, Yoshua Bengio, Aaron Courvile, *Deep learning*, MIT Press, 2016.
3. Andrew Ng., *Machine Learning Yearning*. Link: https://www.deeplearning.ai/machine-learning-yearning/
4. Vũ Hữu Tiệp, *Machine Learning cơ bản*, NXB Khoa học và Kỹ thuật, 2018.