CSE: **Faculty of Computer Science and Engineering**
Thuyloi University

Mạng nơ ron hồi tiếp

Recurrent Neural Network - RNN

TS. Nguyễn Thị Kim Ngân

Nội dung

- Mạng Nơ-ron Hồi tiếp
- Mạng Nơ-ron Hồi tiếp hiện đại
  - Nút Hồi tiếp có Cổng (GRU)
  - Bộ nhớ Ngắn hạn Dài (LSTM)
  - Mạng Nơ-ron Hồi tiếp 2 chiều

Nội dung

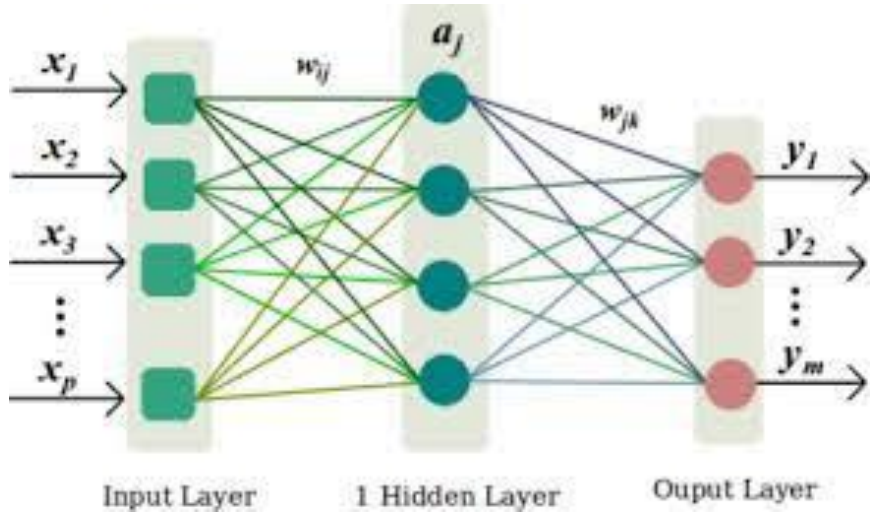
- **Mạng Nơ-ron Hồi tiếp**
- Mạng Nơ-ron Hồi tiếp hiện đại
  - Nút Hồi tiếp có Cổng (GRU)
  - Bộ nhớ Ngắn hạn Dài (LSTM)
  - Mạng Nơ-ron Hồi tiếp 2 chiều

Đặt vấn đề

- Với dữ liệu hình ảnh, nếu ta hoán vị các điểm ảnh trong một ảnh, ta sẽ thu được một bức ảnh trông giống như các khuôn mẫu kiểm tra
- Nhưng trong một đoạn văn, nếu các từ bị hoán vị đi một cách ngẫu nhiên thì sẽ rất khó để giải mã ý nghĩa của chúng
- Các khung hình trong video, tín hiệu âm thanh trong một cuộc hội thoại hoặc hành vi duyệt web, tất cả đều có cấu trúc tuần tự
- Các mạng nơ-ron hồi tiếp được thiết kế để xử lý thông tin tuần tự tốt hơn. Các mạng này sử dụng các biến trạng thái để lưu trữ thông tin trong quá khứ, sau đó dựa vào chúng và các đầu vào hiện tại để xác định các đầu ra hiện tại

Mạng Hồi tiếp không có trạng thái ẩn

- Xét một perception đa tầng với một tầng ẩn duy nhất
  - Giả sử ta có một minibatch $X \in R^{n \times d}$ với n mẫu và d đầu vào
  - Tầng ẩn có h nút ẩn, hàm kích hoạt của tầng ẩn là ϕ
- Đầu ra của tầng ẩn $H \in R^{n \times h}$ được tính như sau:



$$\mathbf{H} = \phi(\mathbf{X}\mathbf{W}_{xh} + \mathbf{b}_h)$$

$W_{xh} \in R^{d \times h}$: tham số trọng số, $b_h \in R^{1 \times h}$: hệ số điều chỉnh

- Tầng đầu ra được tính toán bởi:

$$\mathbf{O} = \mathbf{H}\mathbf{W}_{hq} + \mathbf{b}_q$$

$O \in R^{n \times q}$: biến đầu ra, $W_{hq} \in R^{h \times q}$:tham số trọng số , $b_q \in R^{1 \times q}$:hệ số điều chỉnh

Mạng hồi tiếp có trạng thái ẩn

Thời điểm t tương ứng với bước t trong một vòng lặp. Giả sử, tại bước thời gian t ta có:

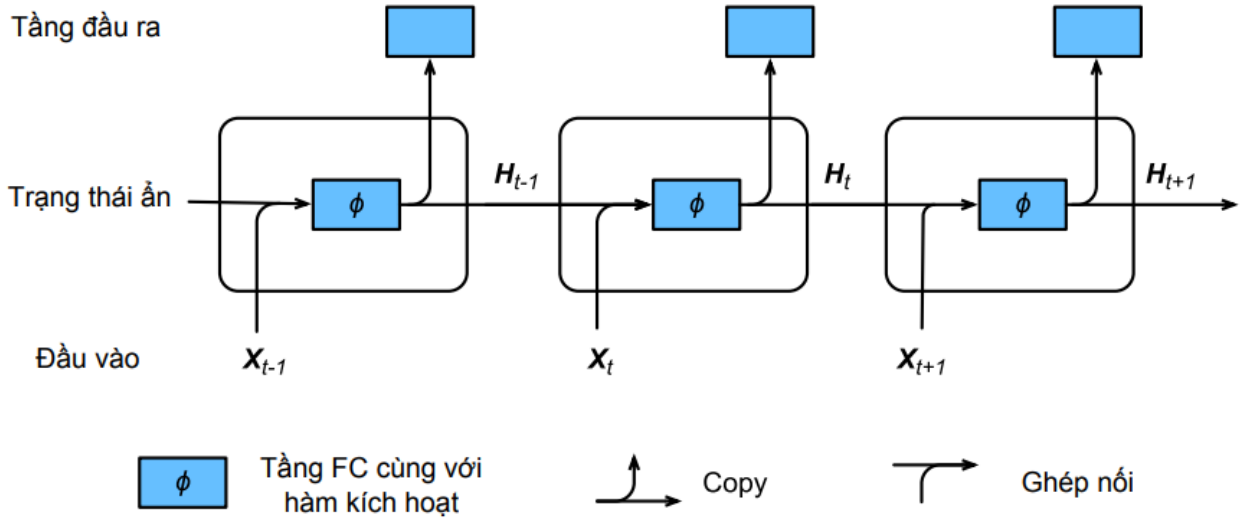
- Đầu vào $X_t \in R^{n \times d}$, $t = 1, \ldots, T$
- $H_t \in R^{n \times h}$ là biến ẩn tại bước thời gian t của chuỗi

$$\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h)$$

- Đầu ra tại bước thời gian t

$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq} + \mathbf{b}_q$$

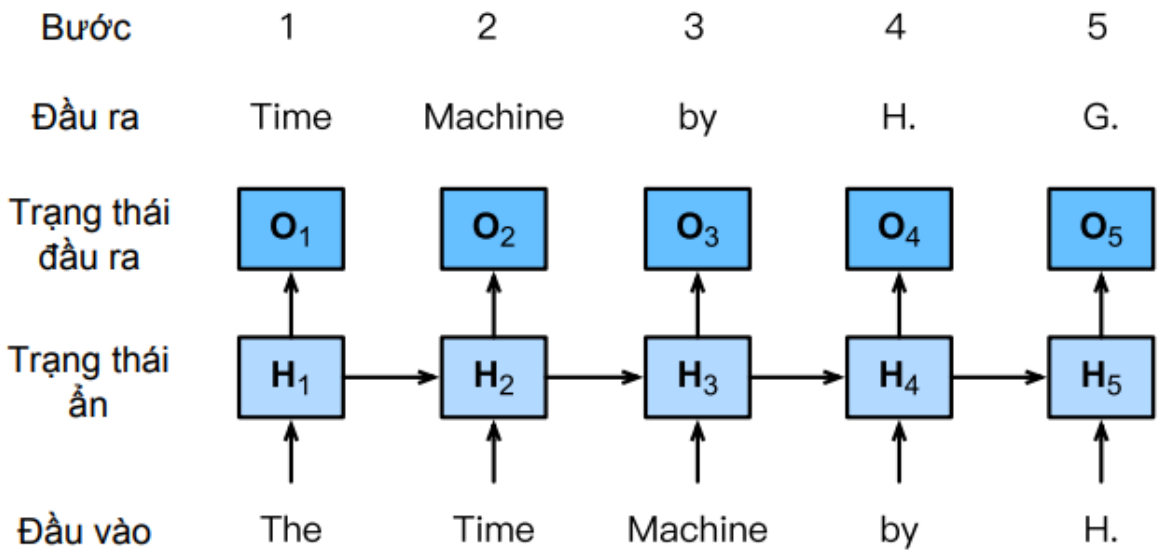
H_{t-1}: biến ẩn tại thời gian trước
W_{xh}, W_{hh}, b_h, W_{hq}, b_q: tham số trọng số



Một RNN với một trạng thái ẩn.

Ví dụ

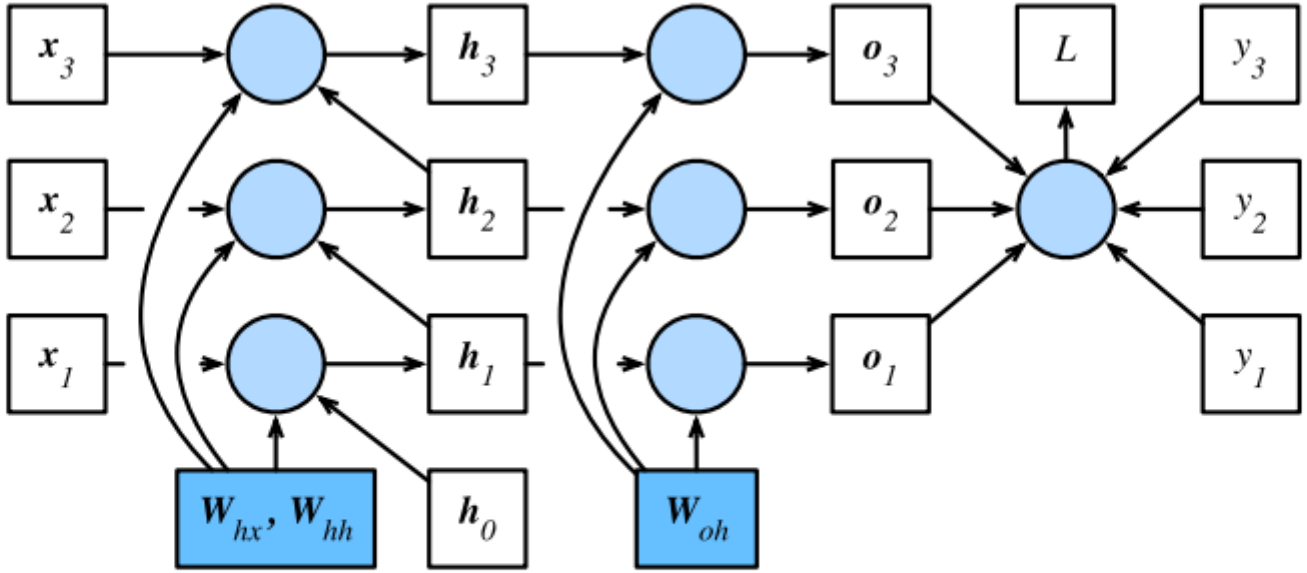
Minh họa cách ước lượng từ tiếp theo dựa trên từ hiện tại và các từ trước đó của chuỗi "the time machine by H. G. Wells"



Mô hình ngôn ngữ ở mức từ ngữ RNN. Đầu vào và chuỗi nhãn lần lượt là `the time machine by H.` và `time machine by H. G.`

Mạng hồi tiếp có trạng thái ẩn

Ví dụ, tại bước thời gian 3, h_3, phụ thuộc vào các tham số W_{hx} và W_{hh} của mô hình, trạng thái ẩn ở bước thời gian trước đó h_2, và đầu vào x_3



Sự phụ thuộc về mặt tính toán của mạng nơ-ron hồi tiếp với ba bước thời gian. Ô vuông tượng trưng cho các biến (không tô đậm) hoặc các tham số (tô đậm), hình tròn tượng trưng cho các phép toán.

Định nghĩa mô hình RNN

Thời điểm t tương ứng với bước t trong một vòng lặp. Giả sử, tại bước thời gian t ta có:

- Đầu vào $X_t \in R^{n \times d}$, t = 1, . . . , T
- $H_t \in R^{n \times h}$ là biến ẩn tại bước thời gian t của chuỗi

$$\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h)$$

- Đầu ra tại bước thời gian t

$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq} + \mathbf{b}_q$$

H_{t-1}: biến ẩn tại thời gian trước
W_{xh}, W_{hh}, b_h, W_{hq}, b_q: tham số trọng số

```python
def init_rnn_state(batch_size, num_hiddens, ctx):
    return (np.zeros(shape=(batch_size, num_hiddens), ctx=ctx), )
```

```python
def rnn(inputs, state, params):
    # Inputs shape: (num_steps, batch_size, vocab_size)
    W_xh, W_hh, b_h, W_hq, b_q = params
    H, = state
    outputs = []
    for X in inputs:
        H = np.tanh(np.dot(X, W_xh) + np.dot(H, W_hh) + b_h)
        Y = np.dot(H, W_hq) + b_q
        outputs.append(Y)
    return np.concatenate(outputs, axis=0), (H,)
```

Nội dung

- Mạng Nơ-ron Hồi tiếp
- **Mạng Nơ-ron Hồi tiếp hiện đại**
  - Nút Hồi tiếp có Cổng (GRU)
  - Bộ nhớ Ngắn hạn Dài (LSTM)
  - Mạng Nơ-ron Hồi tiếp 2 chiều

Đặt vấn đề

Tích của một chuỗi dài các ma trận có thể dẫn đến việc gradient tiêu biến hoặc bùng nổ. Thực tế:

- Những quan sát xuất hiện sớm có thể ảnh hưởng lớn đến việc dự đoán toàn bộ những quan sát trong tương lai. Trường hợp này, ảnh hưởng của token đầu tiên là quan trọng. Do đó, cần lưu trữ những thông tin quan trọng ban đầu trong ô nhớ, hoặc gán một giá trị gradient cực lớn cho quan sát ban đầu

Đặt vấn đề

Tích của một chuỗi dài các ma trận có thể dẫn đến việc gradient tiêu biến hoặc bùng nổ. Thực tế:

- Khi một vài ký hiệu không chứa thông tin phù hợp. Ví dụ, khi phân tích một trang web, các mã HTML không giúp ích gì cho việc xác định thông tin được truyền tải. Do đó, cần có cơ chế để bỏ qua những ký hiệu như vậy trong các biểu diễn trạng thái tiềm ẩn

- Có thể gặp những khoảng ngắt giữa các phần trong một chuỗi. Ví dụ như những phần chuyển tiếp giữa các chương của một quyển sách. Vậy, cần có cách để xóa hay đặt lại các biểu diễn trạng thái ẩn về giá trị ban đầu

Nội dung

- Mạng Nơ-ron Hồi tiếp
- Mạng Nơ-ron Hồi tiếp hiện đại
  - **Nút Hồi tiếp có Cổng (GRU)**
  - Bộ nhớ Ngắn hạn Dài (LSTM)
  - Mạng Nơ-ron Hồi tiếp 2 chiều

So sánh RNN và GRU

Sự khác biệt chính giữa RNN thông thường và GRU là GRU hỗ trợ việc kiểm soát trạng thái ẩn. GRU có các cơ chế được học để quyết định khi nào nên cập nhật và khi nào nên xóa trạng thái ẩn

- Nếu ký tự đầu tiên có mức độ quan trọng cao, mô hình sẽ học để không cập nhật trạng thái ẩn sau lần quan sát đầu tiên
- Học cách bỏ qua những quan sát tạm thời không liên quan, cũng như cách xóa trạng thái ẩn khi cần thiết
- Một ô nhớ GRU có 2 cổng: Cổng xóa, cổng cập nhật

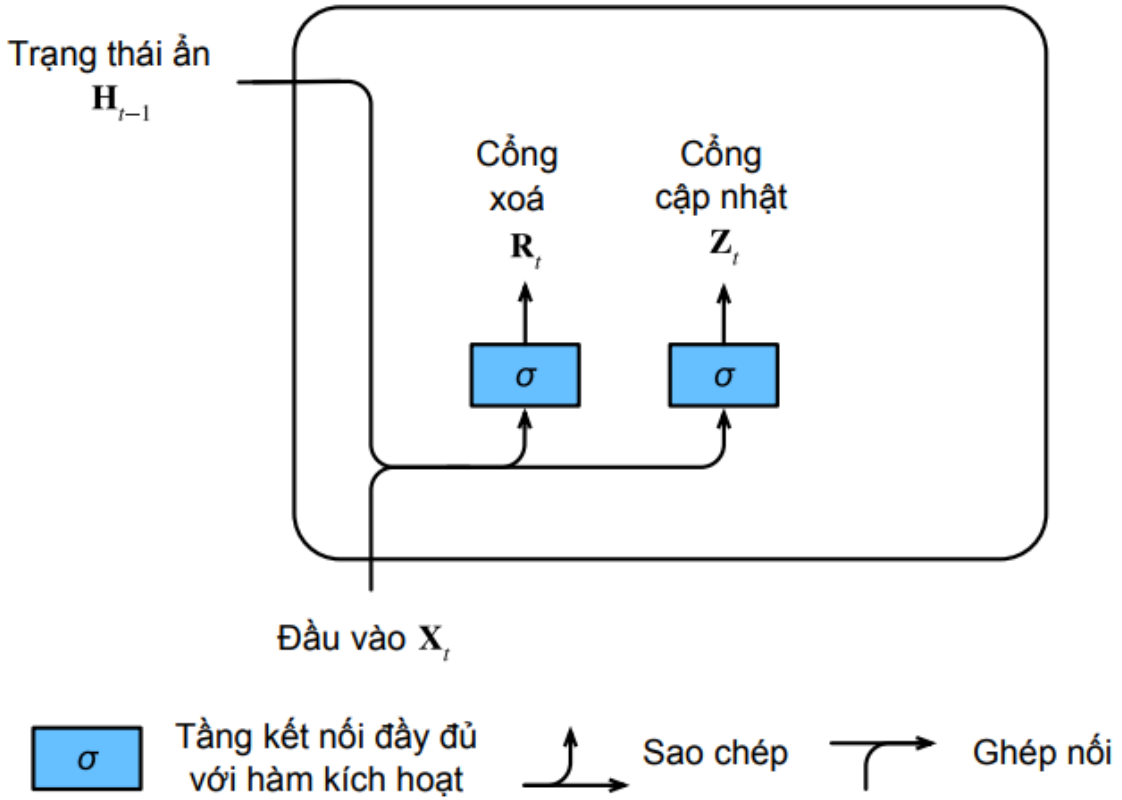
Cổng Xóa và Cổng Cập nhật

- Cổng xóa cho phép kiểm soát bao nhiêu phần của trạng thái trước đây được giữ lại

$$\mathbf{R}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xr} + \mathbf{H}_{t-1}\mathbf{W}_{hr} + \mathbf{b}_r).$$

- Cổng cập nhật cho phép kiểm soát bao nhiêu phần của trạng thái mới sẽ giống trạng thái cũ

$$\mathbf{Z}_t = \sigma(\mathbf{X}_t\mathbf{W}_{xz} + \mathbf{H}_{t-1}\mathbf{W}_{hz} + \mathbf{b}_z)$$



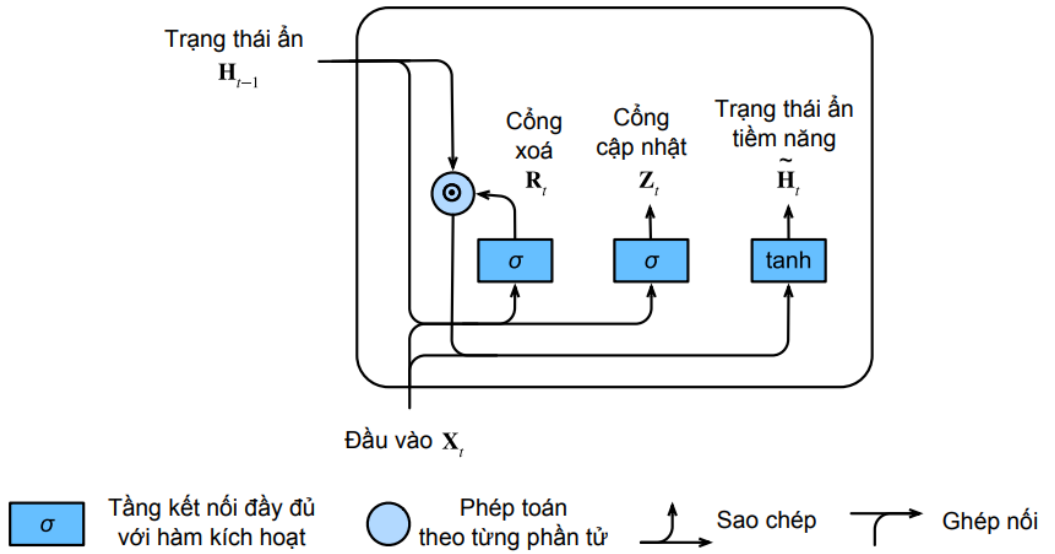
Cổng xóa và cổng cập nhật trong GRU.

Hoạt động của Cổng Xóa

Để giảm ảnh hưởng của các trạng thái trước đó, ta nhân H_{t-1} với R_t theo từng phần tử

- Nếu các phần tử của R_t có giá trị gần với 1, kết quả sẽ giống RNN thông thường
- Nếu tất cả các phần tử của R_t gần với 0, trạng thái ẩn sẽ là đầu ra của một perceptron đa tầng với đầu vào là X_t. Bất kỳ trạng thái ẩn nào tồn tại trước đó đều được đặt lại về giá trị mặc định
- Tại đây, nó được gọi là trạng thái ẩn tiềm năng

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + (\mathbf{R}_t \odot \mathbf{H}_{t-1}) \mathbf{W}_{hh} + \mathbf{b}_h)$$



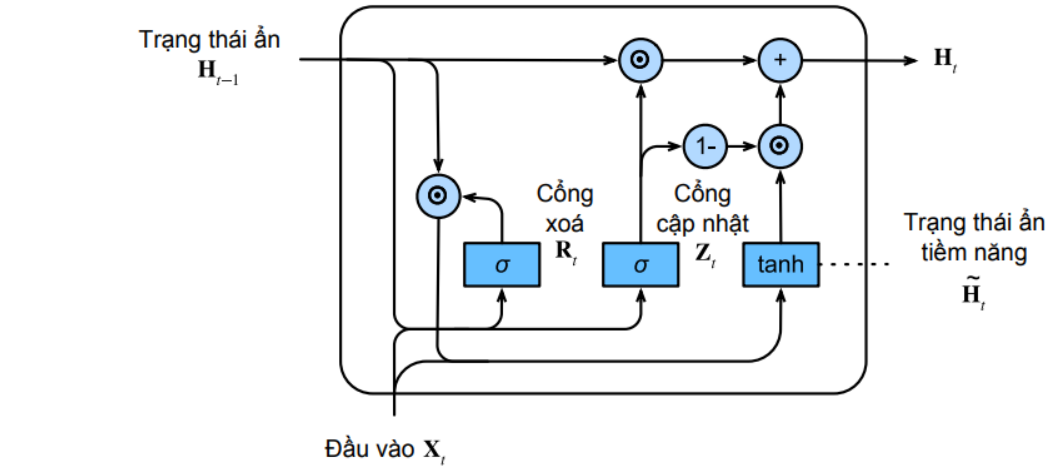
Tính toán của trạng thái ẩn tiềm năng trong một GRU. Phép nhân được thực hiện theo phần tử.

Hoạt động của Cổng Cập nhật

Cổng Cập nhật xác định mức độ giống nhau giữa trạng thái mới H_t và trạng thái cũ H_{t-1}, cũng như mức độ trạng thái ẩn tiềm năng được sử dụng

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t$$

- Nếu $Z_t = 1$, chúng ta giữ lại trạng thái cũ. Thông tin từ Xt được bỏ qua, tương đương với việc bỏ qua bước thời gian t trong chuỗi phụ thuộc
- Nếu Zt gần 0, trạng thái ẩn H_t sẽ gần với trạng thái ẩn tiềm năng



Tính toán trạng thái ẩn trong GRU. Như trước đây, phép nhân được thực hiện theo từng phần tử.

Tính chất của GRU

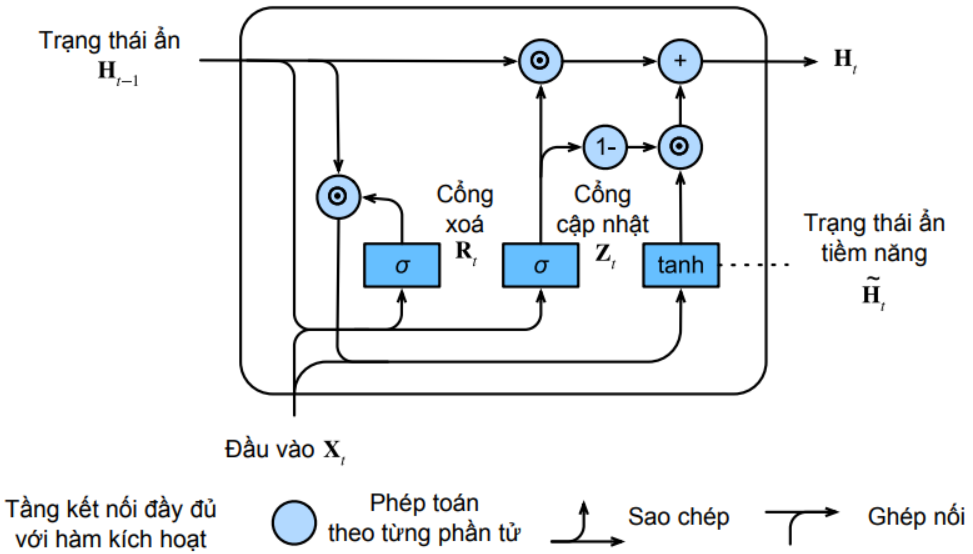
Các mạng GRU có hai tính chất nổi bật sau:

- Cổng xóa giúp nắm bắt các phụ thuộc ngắn hạn trong chuỗi thời gian.
- Cổng cập nhật giúp nắm bắt các phụ thuộc dài hạn trong chuỗi thời gian

Định nghĩa mô hình GRU

```python
def init_gru_state(batch_size, num_hiddens, ctx):
    return (np.zeros(shape=(batch_size, num_hiddens), ctx=ctx), )
```

```python
def gru(inputs, state, params):
    W_xz, W_hz, b_z, W_xr, W_hr, b_r, W_xh, W_hh, b_h, W_hq, b_q = params
    H, = state
    outputs = []
    for X in inputs:
        Z = npx.sigmoid(np.dot(X, W_xz) + np.dot(H, W_hz) + b_z)
        R = npx.sigmoid(np.dot(X, W_xr) + np.dot(H, W_hr) + b_r)
        H_tilda = np.tanh(np.dot(X, W_xh) + np.dot(R * H, W_hh) + b_h)
        H = Z * H + (1 - Z) * H_tilda
        Y = np.dot(H, W_hq) + b_q
        outputs.append(Y)
    return np.concatenate(outputs, axis=0), (H,)
```



Tính toán trạng thái ẩn trong GRU. Như trước đây, phép nhân được thực hiện theo từng phần tử.

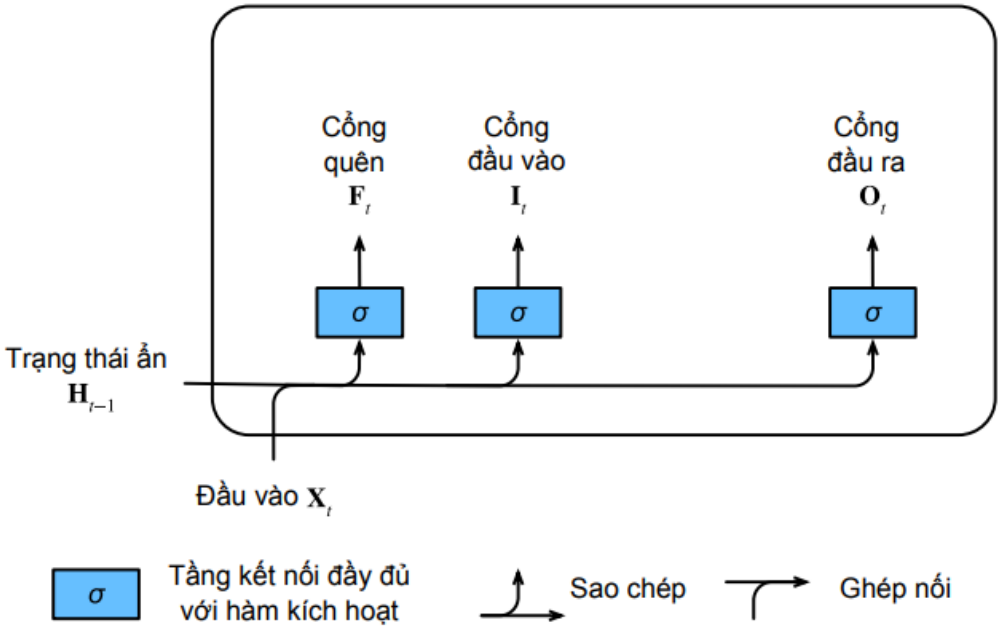
Nội dung

- Mạng Nơ-ron Hồi tiếp
- Mạng Nơ-ron Hồi tiếp hiện đại
  - Nút Hồi tiếp có Cổng (GRU)
  - **Bộ nhớ Ngắn hạn Dài (LSTM)**
  - Mạng Nơ-ron Hồi tiếp 2 chiều

Bộ nhớ Ngắn hạn Dài (LSTM)

Khi nào cần nhớ và khi nào nên bỏ qua đầu vào trong trạng thái tiềm ẩn? Mạng LSTM được thiết kế gồm 3 cổng:

- Cổng đầu ra (ouput gate): để đọc các thông tin từ ô nhớ
- Cổng đầu vào (input gate): để quyết định khi nào cần ghi dữ liệu vào ô nhớ
- Cổng quên (forget gate): để thiết lập lại nội dung chứa trong ô nhớ
- X_t, và H_{t-1} được xử lý bởi một tầng kết nối đầy đủ và một hàm kích hoạt sigmoid để tính toán các giá trị của các cổng



Các phép tính tại cổng đầu vào, cổng quên và cổng đầu ra trong một đơn vị LSTM.

Cổng đầu vào, Cổng quên, Cổng đầu ra

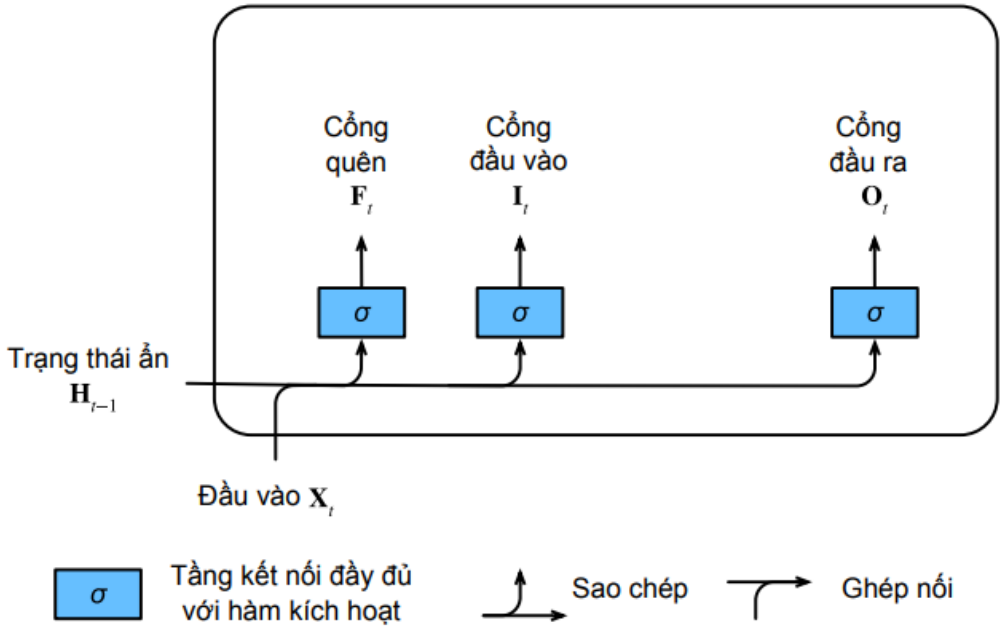
Giả sử có h nút ẩn, mỗi minibatch có kích thước n và kích thước đầu vào là d

- Đầu vào: $X_t \in R^{n \times d}$
- Trạng thái ẩn: $H_{t-1} \in R^{n \times h}$
- Cổng đầu vào: $I_t \in R^{n \times h}$
- Cổng quên: $F_t \in R^{n \times h}$
- Cổng đầu ra: $O_t \in R^{n \times h}$

$$\mathbf{I}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xi} + \mathbf{H}_{t-1} \mathbf{W}_{hi} + \mathbf{b}_i),$$
$$\mathbf{F}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xf} + \mathbf{H}_{t-1} \mathbf{W}_{hf} + \mathbf{b}_f),$$
$$\mathbf{O}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xo} + \mathbf{H}_{t-1} \mathbf{W}_{ho} + \mathbf{b}_o),$$

$W_{xi}, W_{xf}, W_{xo} \in R^{d \times h}$, $W_{hi}, W_{hf}, W_{ho} \in R^{h \times h}$

$B_i, b_f, b_o \in R^{1 \times h}$



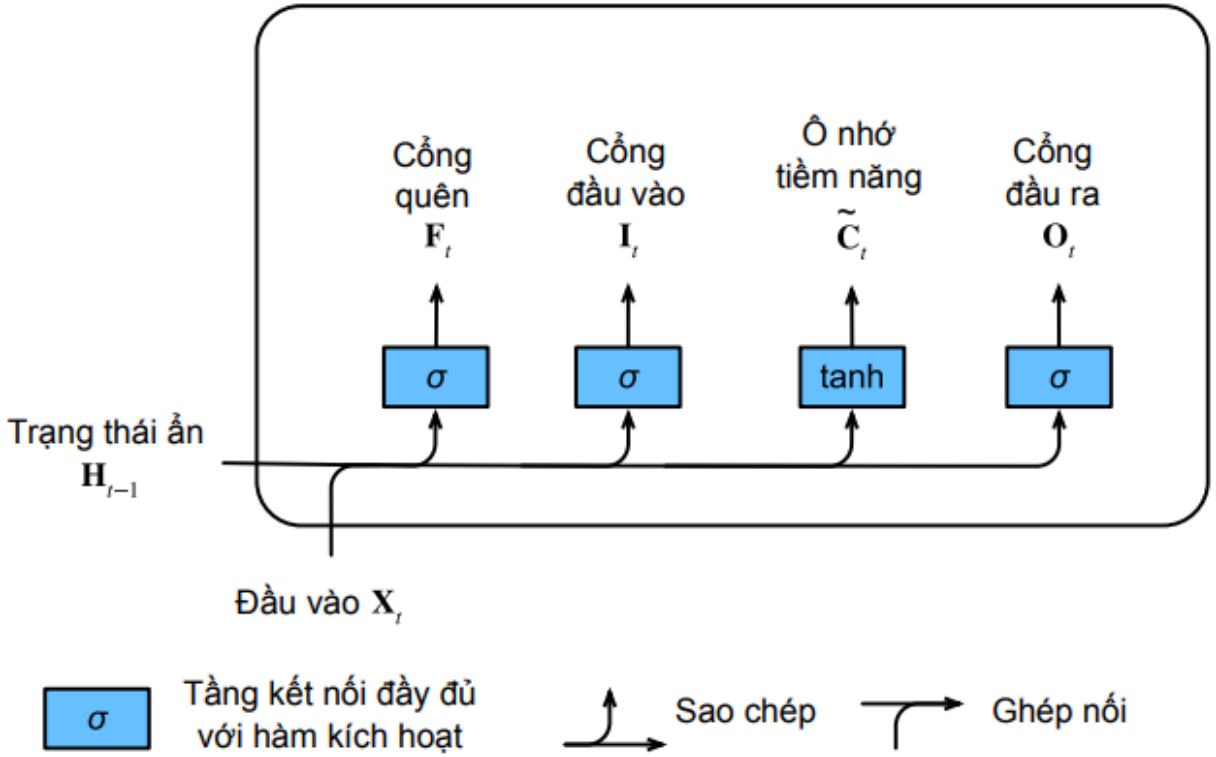
Các phép tính tại cổng đầu vào, cổng quên và cổng đầu ra trong một đơn vị LSTM.

Ô nhớ Tiềm năng

- Ô nhớ tiềm năng $\tilde{C}_t \in R^{n \times h}$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xc} + \mathbf{H}_{t-1} \mathbf{W}_{hc} + \mathbf{b}_c).$$

$W_{xc} \in R^{d \times h}$, $W_{hc} \in R^{h \times h}$ là các tham số trọng số
$b_c \in R^{1 \times h}$ là hệ số điều chỉnh



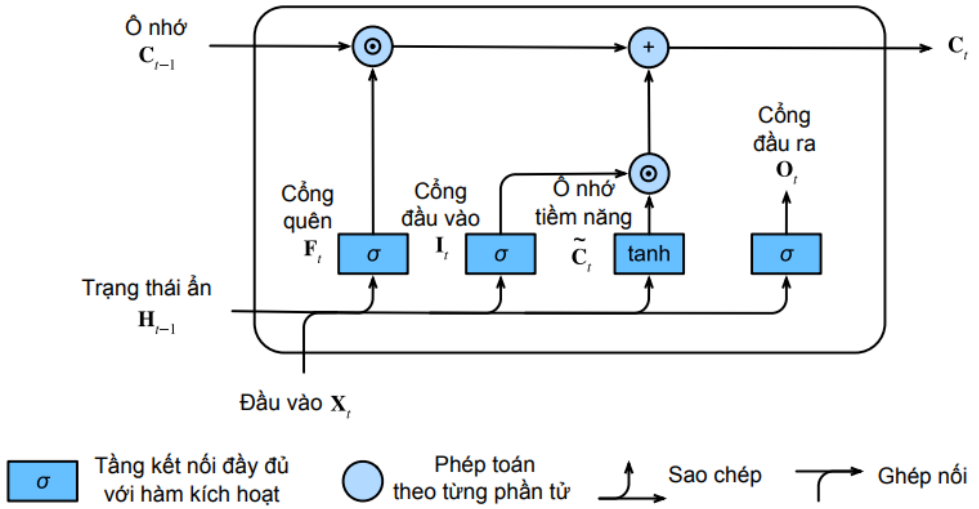
Các phép tính toán trong ô nhớ tiềm năng của LSTM.

Ô nhớ

- I_t điều chỉnh lượng dữ liệu mới được lấy vào thông qua $\tilde{C}_t$
- F_t chỉ định lượng thông tin cũ cần giữ lại trong ô nhớ $C_{t-1} \in R^{n \times h}$

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t$$

- Nếu F_t xấp xỉ bằng 1 và I_t xấp xỉ bằng 0, thì giá trị ô nhớ trong quá khứ C_{t-1} sẽ được lưu lại qua thời gian và truyền tới bước thời gian hiện tại
- Thiết kế này nhằm giảm bớt sự tiêu biến gradient, nắm bắt các phụ thuộc dài hạn trong chuỗi thời gian tốt hơn



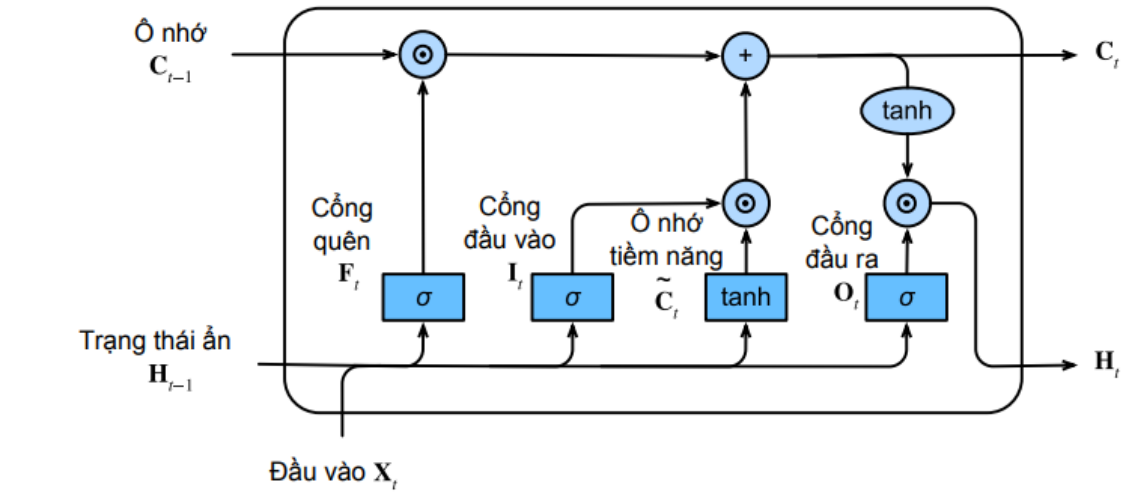
Các phép tính toán trong ô nhớ của LSTM. Ở đây, ta sử dụng phép nhân theo từng phần tử.

Các Trạng thái ẩn

- Trạng thái ẩn $H_t \in R^{n \times h}$

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t)$$

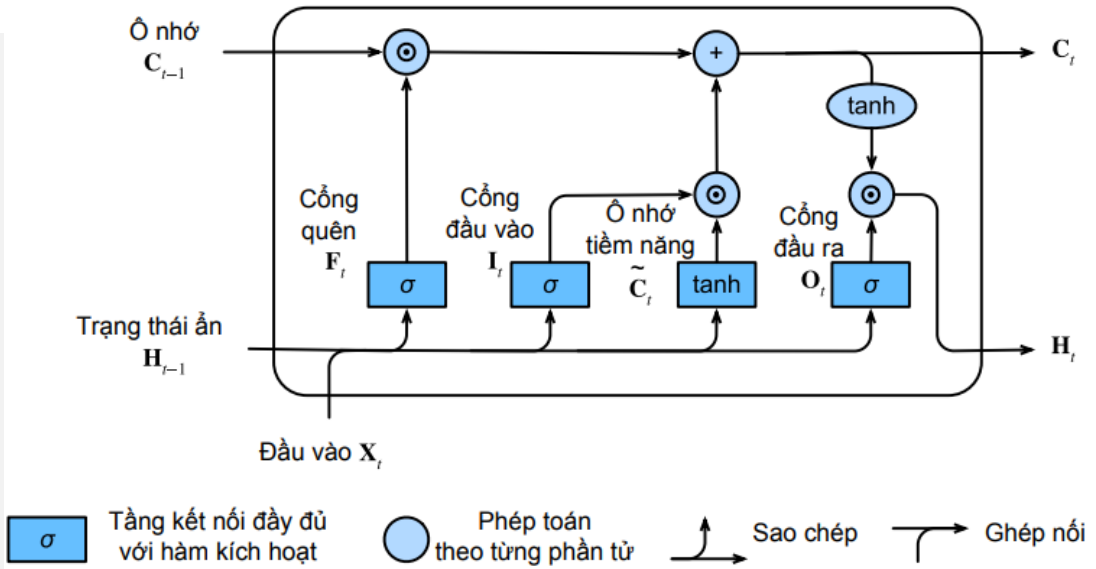
- Giá trị của H_t luôn nằm trong khoảng $(-1, 1)$
- Nếu giá trị của cổng đầu ra là 1, toàn bộ thông tin trong ô nhớ được đưa tới bộ dự đoán
- Nếu giá trị của cổng đầu ra là 0, tất cả các thông tin trong ô nhớ được giữ lại và không xử lý gì thêm



Các phép tính của trạng thái ẩn. Phép tính nhân được thực hiện trên từng phần tử.

Định nghĩa mô hình LSTM

```
def lstm(inputs, state, params):
    [W_xi, W_hi, b_i, W_xf, W_hf, b_f, W_xo, W_ho, b_o, W_xc, W_hc, b_c,
     W_hq, b_q] = params
    (H, C) = state
    outputs = []
    for X in inputs:
        I = npx.sigmoid(np.dot(X, W_xi) + np.dot(H, W_hi) + b_i)
        F = npx.sigmoid(np.dot(X, W_xf) + np.dot(H, W_hf) + b_f)
        O = npx.sigmoid(np.dot(X, W_xo) + np.dot(H, W_ho) + b_o)
        C_tilda = np.tanh(np.dot(X, W_xc) + np.dot(H, W_hc) + b_c)
        C = F * C + I * C_tilda
        H = O * np.tanh(C)
        Y = np.dot(H, W_hq) + b_q
        outputs.append(Y)
    return np.concatenate(outputs, axis=0), (H, C)
```

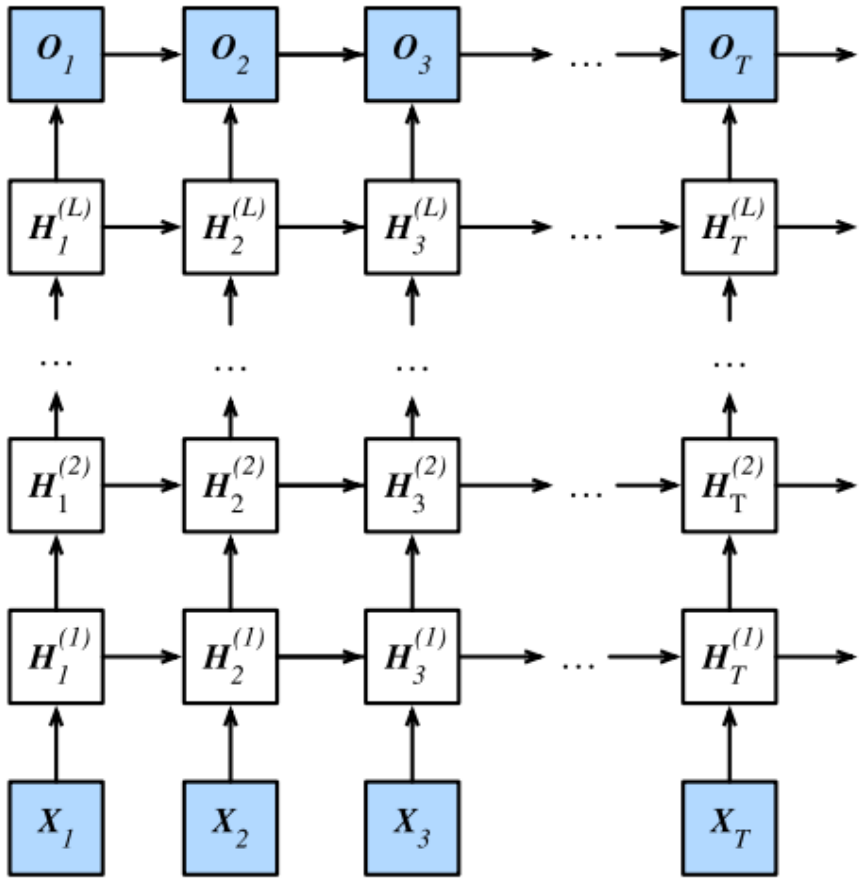


Các phép tính của trạng thái ẩn. Phép tính nhân được thực hiện trên từng phần tử.

Mạng Nơ-ron Hồi tiếp Sâu

- Chúng ta có thể chồng nhiều tầng Nơ-ron hồi tiếp lên nhau. Ví dụ, có thể chồng nhiều tầng LSMT lên nhau, để mạng nơ ron được tinh xảo hơn

- Mạng nơ ron hồi tiếp sâu với L tầng ẩn. Mỗi trạng thái ẩn liên tục được truyền tới bước thời gian kế tiếp ở tầng hiện tại và tới bước thời gian hiện tại ở tầng kế tiếp.

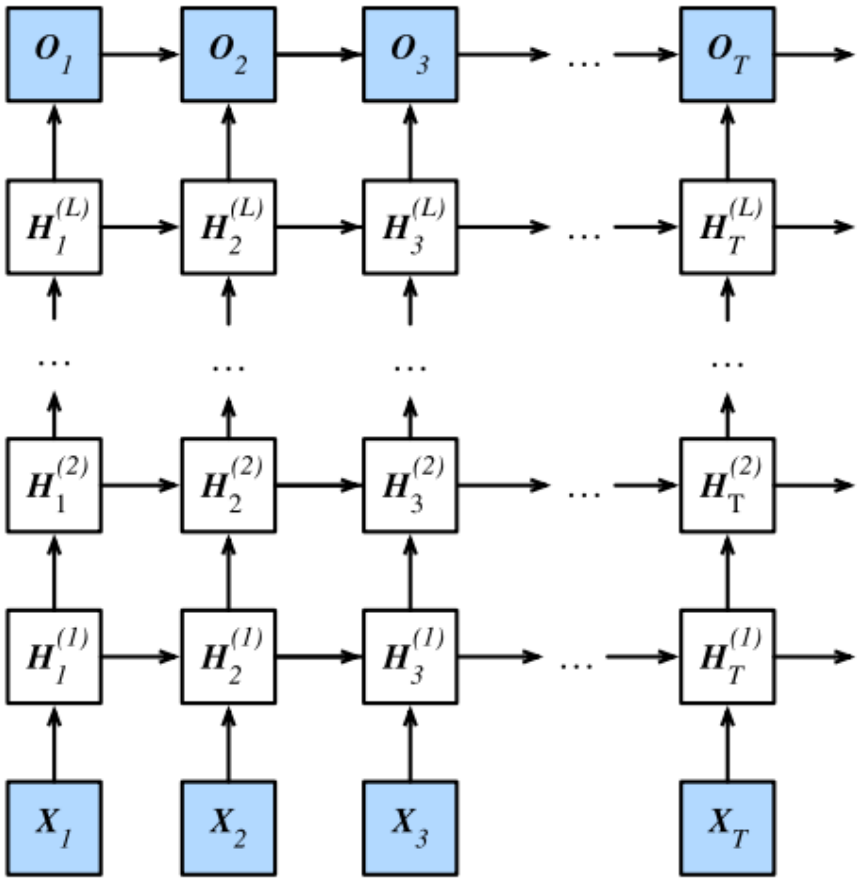


Kiến trúc của một mạng nơ-ron hồi tiếp sâu.

Mạng Nơ-ron Hồi tiếp Sâu

$$\mathbf{H}_t^{(1)} = f_1\left(\mathbf{X}_t, \mathbf{H}_{t-1}^{(1)}\right),$$
$$\mathbf{H}_t^{(l)} = f_l\left(\mathbf{H}_t^{(l-1)}, \mathbf{H}_{t-1}^{(l)}\right).$$

$$\mathbf{O}_t = g\left(\mathbf{H}_t^{(L)}\right).$$



Kiến trúc của một mạng nơ-ron hồi tiếp sâu.

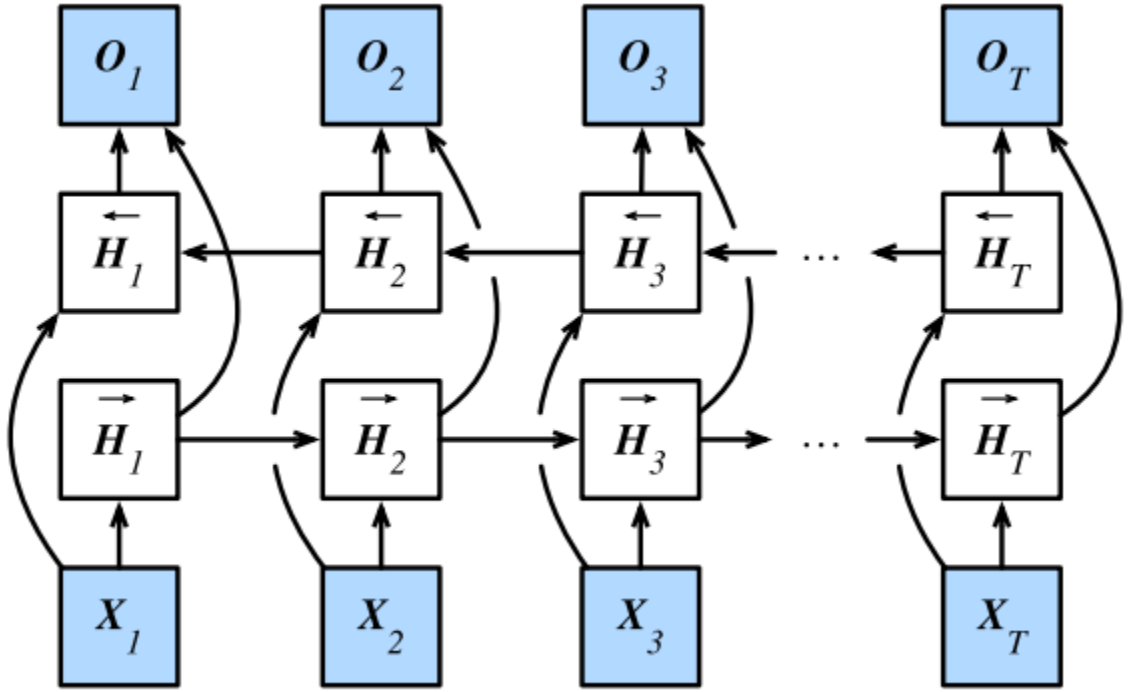
Nội dung

- Mạng Nơ-ron Hồi tiếp
- Mạng Nơ-ron Hồi tiếp hiện đại
  - Nút Hồi tiếp có Cổng (GRU)
  - Bộ nhớ Ngắn hạn Dài (LSTM)
  - **Mạng Nơ-ron Hồi tiếp 2 chiều**

Mạng Nơ-ron Hồi tiếp hai chiều

- Mạng Nơ-ron Hồi tiếp hai chiều (Bidirectional recurrent neural network) thêm một tầng ẩn cho phép xử lý dữ liệu theo chiều ngược lại một cách linh hoạt hơn so với RNN truyền thống

- Ví dụ mạng BiRNN với 1 tầng ẩn



Mạng Nơ-ron Hồi tiếp hai chiều

$\overrightarrow{\mathbf{H}}_t \in \mathbb{R}^{n \times h}$ và $\overleftarrow{\mathbf{H}}_t \in \mathbb{R}^{n \times h}$

$$\overrightarrow{\mathbf{H}}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh}^{(f)} + \overrightarrow{\mathbf{H}}_{t-1} \mathbf{W}_{hh}^{(f)} + \mathbf{b}_h^{(f)}),$$
$$\overleftarrow{\mathbf{H}}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh}^{(b)} + \overleftarrow{\mathbf{H}}_{t+1} \mathbf{W}_{hh}^{(b)} + \mathbf{b}_h^{(b)}).$$

Nối các trạng thái ẩn xuôi và ngược để thu được trạng thái ẩn $\mathbf{H}_t \in \mathbb{R}^{n \times 2h}$

$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq} + \mathbf{b}_q.$$