

# VIP Cheatsheet: Mạng nơ-ron hồi quy

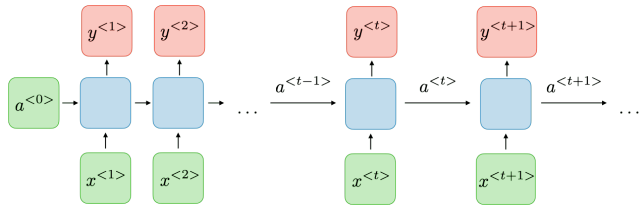
Afshine AMIDI và Shervine AMIDI

Ngày 17 tháng 5 năm 2020

Dịch bởi Trần Tuấn Anh, Đàm Minh Tiến, Hung Nguyễn và Nguyễn Trí Minh

## Tổng quan

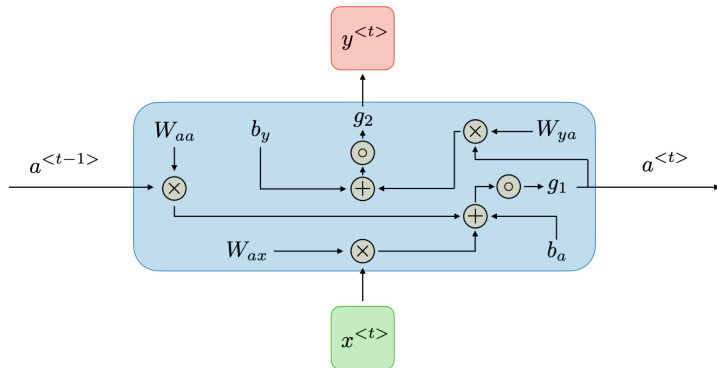
□ **Kiến trúc của một mạng RNN truyền thống** – Các mạng neural hồi quy, còn được biến đến như là RNNs, là một lớp của mạng neural cho phép đầu ra được sử dụng như đầu vào trong khi có các trạng thái ẩn. Thông thường là như sau:



Tại mỗi bước  $t$ , giá trị kích hoạt  $a^{<t>}$  và đầu ra  $y^{<t>}$  được biểu diễn như sau:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad \text{và} \quad y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

với  $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$  là các hệ số được chia sẻ tạm thời và  $g_1, g_2$  là các hàm kích hoạt.



Ưu và nhược điểm của một kiến trúc RNN thông thường được tổng kết ở bảng dưới đây:

Ưu điểm	Hạn chế
<ul style="list-style-type: none"> <li>- Khả năng xử lý đầu vào với bất kì độ dài nào</li> <li>- Kích cỡ mô hình không tăng theo kích cỡ đầu vào</li> <li>- Quá trình tính toán sử dụng các thông tin cũ</li> <li>- Trọng số được chia sẻ trong suốt thời gian</li> </ul>	<ul style="list-style-type: none"> <li>- Tính toán chậm</li> <li>- Khó để truy cập các thông tin từ một khoảng thời gian dài trước đây</li> <li>- Không thể xem xét bất kì đầu vào sau này nào cho trạng thái hiện tại</li> </ul>

□ **Ứng dụng của RNNs** – Các mô hình RNN hầu như được sử dụng trong lĩnh vực xử lý ngôn ngữ tự nhiên và ghi nhận tiếng nói. Các ứng dụng khác được tổng kết trong bảng dưới đây:

Các loại RNN	Hình minh họa	Ví dụ
Một-Một $T_x = T_y = 1$		Mạng neural truyền thống
Một-nhiều $T_x = 1, T_y > 1$		Sinh nhạc
Nhiều-một $T_x > 1, T_y = 1$		Phân loại ý kiến
Nhiều-nhiều $T_x = T_y$		Ghi nhận thực thể tên
Nhiều-nhiều $T_x \neq T_y$		Dịch máy

□ **Hàm mất mát** – Trong trường hợp của mạng neural hồi quy, hàm mất mát  $\mathcal{L}$  của tất cả các bước thời gian được định nghĩa dựa theo mất mát ở mọi thời điểm như sau:

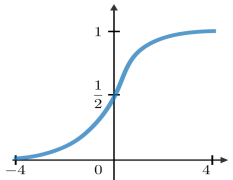
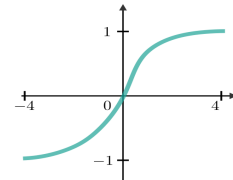
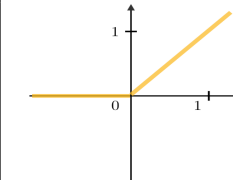
$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{y}^{<t>}, y^{<t>})$$

□ **Lan truyền ngược theo thời gian** – Lan truyền ngược được hoàn thành ở mỗi một thời điểm cụ thể. Ở bước  $T$ , đạo hàm của hàm mất mát  $\mathcal{L}$  với ma trận trọng số  $W$  được biểu diễn như sau:

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^T \frac{\partial \mathcal{L}^{(T)}}{\partial W} \Big|_{(t)}$$

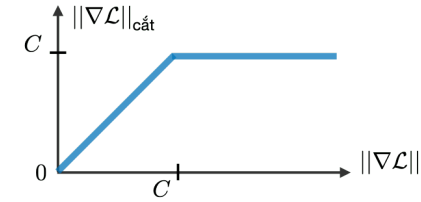
## Xử lý phụ thuộc dài hạn

□ **Các hàm kích hoạt thường dùng** – Các hàm kích hoạt thường dùng trong các modules RNN được miêu tả như sau:

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

□ **Vanishing/exploding gradient** – Hiện tượng vanishing và exploding gradient thường gặp trong ngữ cảnh của RNNs. Lí do tại sao chúng thường xảy ra đó là khó để có được sự phụ thuộc dài hạn vì multiplicative gradient có thể tăng/giảm theo hàm mũ tương ứng với số lượng các tầng.

□ **Gradient clipping** – Là một kĩ thuật được sử dụng để giải quyết vấn đề exploding gradient xảy ra khi thực hiện lan truyền ngược. Bằng việc giới hạn giá trị lớn nhất cho gradient, hiện tượng này sẽ được kiểm soát trong thực tế.



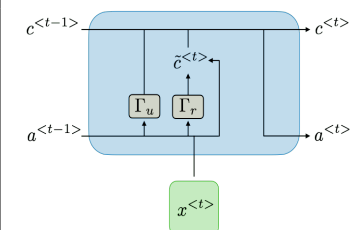
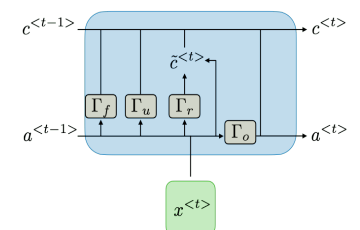
□ **Các loại cổng** – Để giải quyết vấn đề vanishing gradient, các cổng cụ thể được sử dụng trong một vài loại RNNs và thường có mục đích rõ ràng. Chúng thường được kí hiệu là  $\Gamma$  và bằng với:

$$\Gamma = \sigma(Wx^{<t>} + Ua^{<t-1>} + b)$$

Với  $W, U, b$  là các hệ số của một cổng và  $\sigma$  là hàm sigmoid. Các loại chính được tổng kết ở bảng dưới đây:

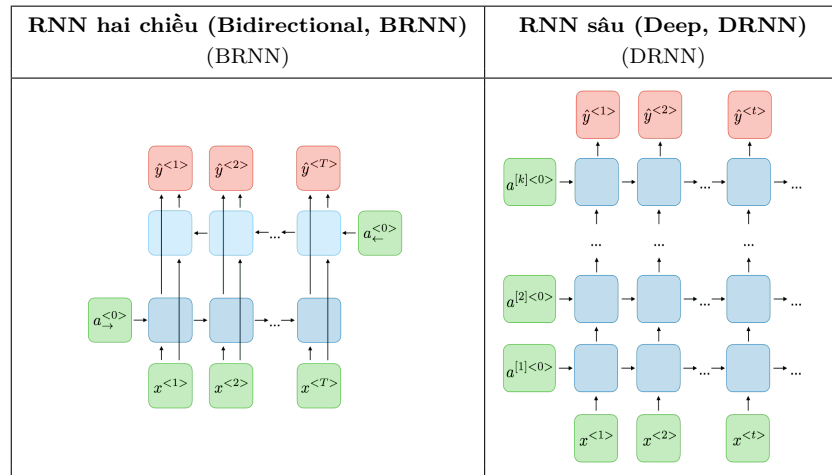
Loại cổng	Vai trò	Được sử dụng trong
Cổng cập nhật $\Gamma_u$	Dữ liệu cũ nên có tầm quan trọng như thế nào ở hiện tại?	GRU, LSTM
Cổng relevance $\Gamma_r$	Bỏ qua thông tin phía trước?	GRU, LSTM
Cổng quên $\Gamma_f$	Xoá ô hay không xoá?	LSTM
Cổng ra $\Gamma_o$	Biểu thị một ô ở mức độ bao nhiêu?	LSTM

□ **GRU/LSTM** – Gated Recurrent Unit (GRU) và Các đơn vị bộ nhớ dài-ngắn hạn (LSTM) đối phó với vấn đề vanishing gradient khi gặp phải bằng mạng RNNs truyền thống, với LSTM là sự tổng quát của GRU. Phía dưới là bảng tổng kết các phương trình đặc trưng của mỗi kiến trúc:

	Gated Recurrent Unit (GRU) (GRU)	Bộ nhớ dài-ngắn hạn (LSTM) (LSTM)
$\tilde{c}^{<t>}$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$
$c^{<t>}$	$\Gamma_u \star \tilde{c}^{<t>} + (1 - \Gamma_u) \star c^{<t-1>}$	$\Gamma_u \star \tilde{c}^{<t>} + \Gamma_f \star c^{<t-1>}$
$a^{<t>}$	$c^{<t>}$	$\Gamma_o \star c^{<t>}$
Các phụ thuộc		

Chú ý: kí hiệu  $\star$  chỉ phép nhân từng phần tử với nhau giữa hai vectors.

□ **Các biến thể của RNNs** – Bảng dưới đây tổng kết các kiến trúc thường được sử dụng khác của RNN:



## Học từ đại diện

Trong phần này, chúng ta kí hiệu  $V$  là tập từ vựng và  $|V|$  là kích cỡ của nó.

□ **Các kĩ thuật biểu diễn** – Có hai cách chính để biểu diễn từ được tổng kết ở bảng bên dưới:

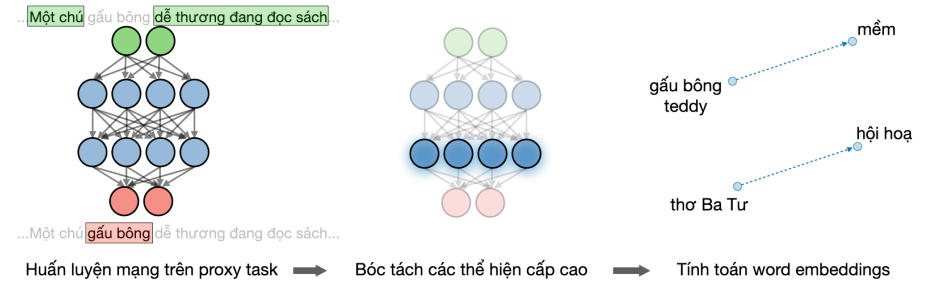
Biểu diễn 1-hot	Word embedding
<ul style="list-style-type: none"> <li>- Lưu ý <math>o_w</math></li> <li>- Tiếp cận Naive, không có thông tin chung</li> </ul>	<ul style="list-style-type: none"> <li>- Lưu ý <math>e_w</math></li> <li>- Xem xét độ tương đồng của các từ</li> </ul>

□ **Embedding matrix** – Cho một từ  $w$ , embedding matrix  $E$  là một ma trận tham chiếu thể hiện 1-hot  $o_w$  của nó với embedding  $e_w$  của nó như sau:

$$e_w = E o_w$$

Chú ý: học embedding matrix có thể hoàn thành bằng cách sử dụng các mô hình target/context likelihood.

□ **Word2vec** – Word2vec là một framework tập trung vào việc học word embeddings bằng cách ước lượng khả năng mà một từ cho trước được bao quanh bởi các từ khác. Các mô hình phổ biến bao gồm skip-gram, negative sampling và CBOW.



□ **Skip-gram** – Mô hình skip-gram word2vec là một task học có giám sát, nó học các word embeddings bằng cách đánh giá khả năng của bất kì target word  $t$  cho trước nào xảy ra với context word  $c$ . Bằng việc kí hiệu  $\theta_t$  là tham số đi kèm với  $t$ , xác suất  $P(t|c)$  được tính như sau:

$$P(t|c) = \frac{\exp(\theta_t^T e_c)}{\sum_{j=1}^{|V|} \exp(\theta_j^T e_c)}$$

Chú ý: Cộng tổng tất cả các từ vựng trong mẫu số của phần softmax khiến mô hình này tốn nhiều chi phí tính toán. CBOW là một mô hình word2vec khác sử dụng các từ xung quanh để dự đoán một từ cho trước.

□ **Negative sampling** – Nó là một tập của các bộ phân loại nhị phân sử dụng logistic regressions với mục tiêu là đánh giá khả năng mà một ngữ cảnh cho trước và các target words cho trước có thể xuất hiện đồng thời, với các mô hình đang được huấn luyện trên các tập của  $k$  negative examples và 1 positive example. Cho trước context word  $c$  và target word  $t$ , dự đoán được thể hiện bởi:

$$P(y = 1|c, t) = \sigma(\theta_t^T e_c)$$

Chú ý: phương thức này tốn ít chi phí tính toán hơn mô hình skip-gram.

□ **GloVe** – Mô hình GloVe, viết tắt của global vectors for word representation, nó là một kĩ thuật word embedding sử dụng ma trận đồng xuất hiện  $X$  với mỗi  $X_{i,j}$  là số lần mà từ đích (target)  $i$  xuất hiện tại ngữ cảnh  $j$ . Cost function  $J$  của nó như sau:

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{|V|} f(X_{i,j})(\theta_i^T e_j + b_i + b'_j - \log(X_{i,j}))^2$$

$f$  là hàm trong số với  $X_{i,j} = 0 \implies f(X_{i,j}) = 0$ . Với tính đối xứng mà  $e$  và  $\theta$  có được trong mô hình này, word embedding cuối cùng  $e_w^{(\text{final})}$  được định nghĩa như sau:

$$e_w^{(\text{final})} = \frac{e_w + \theta_w}{2}$$

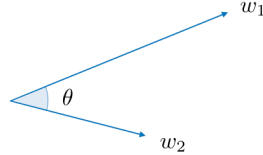
Chú ý: Các phần tử riêng của các word embedding học được không nhất thiết là phải thông dịch được.

## So sánh các từ

□ **Độ tương đồng cosine** – Độ tương đồng cosine giữa các từ  $w_1$  và  $w_2$  được trình bày như sau:

$$\text{similarity} = \frac{w_1 \cdot w_2}{\|w_1\| \|w_2\|} = \cos(\theta)$$

Chú ý:  $\theta$  là góc giữa các từ  $w_1$  và  $w_2$ .



□ **t-SNE** – t-SNE (t-distributed Stochastic Neighbor Embedding) là một kĩ thuật nhằm giảm đi số chiều của không gian embedding. Trong thực tế, nó thường được sử dụng để trực quan hoá các word vectors trong không gian 2 chiều (2D).



## Mô hình ngôn ngữ

□ **Tổng quan** – Một mô hình ngôn ngữ sẽ dự đoán xác suất của một câu  $P(y)$ .

□ **Mô hình n-gram** – Mô hình này là cách tiếp cận naive với mục đích định lượng xác suất mà một biểu hiện xuất hiện trong văn bản bằng cách đếm số lần xuất hiện của nó trong tập dữ liệu huấn luyện.

□ **Độ hỗn tạp** – Các mô hình ngôn ngữ thường được đánh giá dựa theo độ đo hỗn tạp, cũng được biết đến là PP, có thể được hiểu như là nghịch đảo xác suất của tập dữ liệu được chuẩn hoá bởi số lượng các từ  $T$ . Độ hỗn tạp càng thấp thì càng tốt và được định nghĩa như sau:

$$PP = \prod_{t=1}^T \left( \frac{1}{\sum_{j=1}^{|V|} y_j^{(t)} \cdot \hat{y}_j^{(t)}} \right)^{\frac{1}{T}}$$

Chú ý: PP thường được sử dụng trong t-SNE.

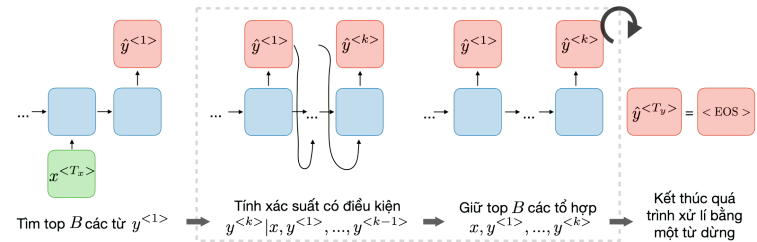
## Dịch máy

□ **Tổng quan** – Một mô hình dịch máy tương tự với mô hình ngôn ngữ ngoại trừ nó có một mạng encoder được đặt phía trước. Vì lí do này, đôi khi nó còn được biết đến là mô hình ngôn ngữ có điều kiện. Mục tiêu là tìm một câu văn  $y$  như sau:

$$y = \arg \max_{y^{<1>}, \dots, y^{<T_y>}} P(y^{<1>}, \dots, y^{<T_y>} | x)$$

□ **Tìm kiếm Beam** – Nó là một giải thuật tìm kiếm heuristic được sử dụng trong dịch máy và ghi nhận tiếng nói để tìm câu văn  $y$  đúng nhất tương ứng với đầu vào  $x$ .

- Bước 1: Tìm top B các từ  $y^{<1>}$
- Bước 2: Tính xác suất có điều kiện  $y^{<k>} | x, y^{<1>}, \dots, y^{<k-1>}$
- Bước 3: Giữ top B các tổ hợp  $x, y^{<1>}, \dots, y^{<k>}$



Chú ý: nếu độ rộng của beam được thiết lập là 1, thì nó tương đương với tìm kiếm tham lam naive.

□ **Độ rộng Beam** – Độ rộng beam  $B$  là một tham số của giải thuật tìm kiếm beam. Các giá trị lớn của  $B$  tạo ra kết quả tốt hơn nhưng với hiệu năng thấp hơn và lượng bộ nhớ sử dụng sẽ tăng.

□ **Chuẩn hoá độ dài** – Để cải thiện tính ổn định, beam search thường được áp dụng mục tiêu chuẩn hoá sau, thường được gọi là mục tiêu chuẩn hoá log-likelihood, được định nghĩa như sau:

$$\text{Objective} = \frac{1}{T_y^\alpha} \sum_{t=1}^{T_y} \log \left[ p(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>}) \right]$$

Chú ý: tham số  $\alpha$  có thể được xem như là softener, và giá trị của nó thường nằm trong đoạn 0.5 và 1.

□ **Phân tích lỗi** – Khi có được một bản dịch tồi  $\hat{y}$ , chúng ta có thể tự hỏi rằng tại sao chúng ta không có được một kết quả dịch tốt  $y^*$  bằng việc thực hiện việc phân tích lỗi như sau:

Trường hợp	$P(y^*   x) > P(\hat{y}   x)$	$P(y^*   x) \leq P(\hat{y}   x)$
Nguyên nhân sâu xa	Lỗi Beam search	Lỗi RNN
Biện pháp khắc phục	Tăng beam width	- Thử kiến trúc khác - Chính quy - Lấy nhiều dữ liệu hơn

□ **Điểm Bleu** – Bilingual evaluation understudy (bleu) score định lượng mức độ tốt của dịch máy bằng cách tính một độ tương đồng dựa trên dự đoán n-gram. Nó được định nghĩa như sau:

$$\text{bleu score} = \exp \left( \frac{1}{n} \sum_{k=1}^n p_k \right)$$

với  $p_n$  là bleu score chỉ trên  $n$ -gram được định nghĩa như sau:

$$p_n = \frac{\sum_{\text{n-gram} \in \hat{y}} \text{count}_{\text{clip}}(\text{n-gram})}{\sum_{\text{n-gram} \in \hat{y}} \text{count}(\text{n-gram})}$$

*Chú ý: một mức phạt ngắn có thể được áp dụng với các dự đoán dịch ngắn để tránh việc làm thổi phồng giá trị bleu score.*

### Chú ý

□ **Attention model** – Mô hình này cho phép một RNN tập trung vào các phần cụ thể của đầu vào được xem xét là quan trọng, nó giúp cải thiện hiệu năng của mô hình kết quả trong thực tế. Bằng việc kí hiệu  $\alpha^{<t,t'>}$  là mức độ chú ý mà đầu ra  $y^{<t>}$  nên có đối với hàm kích hoạt  $a^{<t'>}$  và  $c^{<t'>}$  là ngữ cảnh ở thời điểm  $t$ , chúng ta có:

$$c^{<t>} = \sum_{t'} \alpha^{<t,t'>} a^{<t'>} \quad \text{với} \quad \sum_{t'} \alpha^{<t,t'>} = 1$$

*Chú ý: Các attention scores thường được sử dụng trong chú thích ảnh và dịch máy.*



*Một chú gấu bông dễ thương đang đọc bài văn Ba Tư*



*Một chú gấu bông dễ thương đang đọc bài văn Ba Tư*

□ **Attention weight** – Sự chú ý mà đầu ra  $y^{<t>}$  nên có với hàm kích hoạt  $a^{<t'>}$  với  $\alpha^{<t,t'>}$  được tính như sau:

$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t''=1}^{T_x} \exp(e^{<t,t''>})}$$

*Chú ý: độ phức tạp tính toán là một phương trình bậc hai đối với  $T_x$ .*

★ ★ ★