Chương 2: LÝ THUYẾT NỀN TẢNG

1. Lý thuyết nền tảng của dịch máy:

1.1. Định nghĩa:

Phần này trình bày một số khái niệm cốt lõi về dịch máy, các mô hình ngôn ngữ và phương pháp học theo đặc trưng trong xử lý ngôn ngữ tự nhiên (Neural Language Proc).

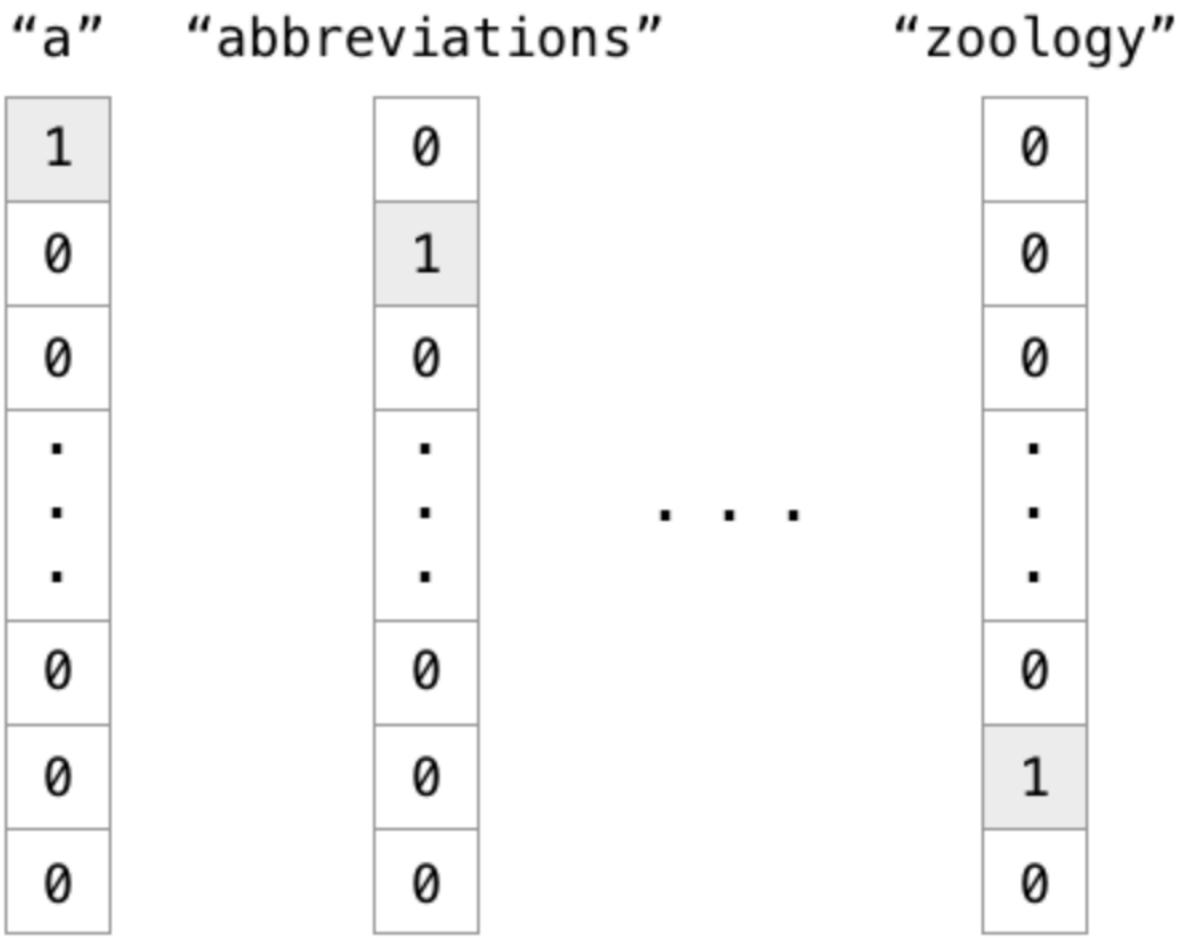
1.1.1. Định nghĩa dịch máy:

Dịch máy (machine translation) là một quá trình thay đổi văn bản từ ngôn ngữ này sang ngôn ngữ khác (gọi là ngôn ngữ đích) một cách tự động, không có sự can thiệp của con người trong quá trình dịch.

1.1.2. Word embeddings:

- Xử lý đầu vào cho bài toán dịch máy là một bước rất quan trọng, các thuật toán, vì các kiến trúc Machine learning, Deep learning chúng chỉ có thể hiểu được đầu vào ở dạng là số nên cần chuyển đầu vào ở dạng text sang dạng số để chúng có thể hiểu được.
- Nhưng nếu chỉ đơn giản biểu diễn từ bằng một con số có thể dẫn đến sai lệch mối quan hệ ngữ nghĩa giữa các từ.Ví dụ như nếu đánh dấu “mèo” là số 1 và “chó” là số 2, như vậy “mèo” + “mèo” = “chó”.
- Một kỹ thuật đơn giản được sử dụng để khác phục là One-hot vector, chúng chuyển các từ thành vector có số chiều bằng số từ của bộ từ vựng đầu vào, trong đó chỉ có duy nhất một phần tử bằng 1 (các phần tử khác bằng 0) tương ứng với vị trí từ đó

trong bộ từ vựng. Tuy nhiên cách biểu diễn này là số chiều của vector lại rất lớn, ảnh hưởng đến quá trình xử lý và lưu trữ.



Hình 1: Hình mô tả cách mã hóa one-hot-vector

- Một cách khác là xử dụng vector ngẫu nhiên, mỗi từ được biểu thị bằng một vector có giá trị các chiều là ngẫu nhiên, mỗi từ là một điểm trong không gian 3D, do đó làm giảm số chiều vector, tuy nhiên nó lại không biểu diễn quan hệ tương đồng giữa các từ.

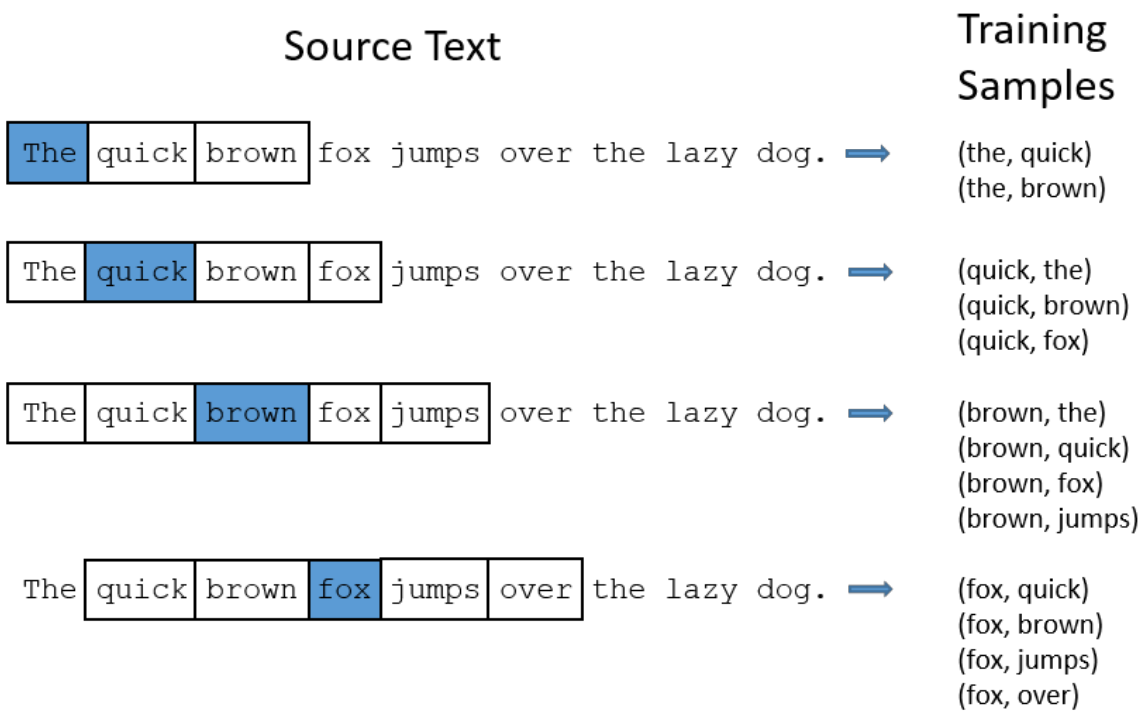
- Sử dụng **Word embeddings** được coi là cách tốt nhất để thể hiện các từ trong văn bản nó cũng gán mỗi từ với một vector nhưng các vector được tính toán để biểu diễn quan hệ tương đồng giữa các từ.
- Word embeddings có 2 model nổi tiếng là Word2vec và Glove.

1.1.3. Word2vec:

Word2vec là một model unsupervised learning nó dùng để thể hiện mỗi quan hệ giữa các từ, nó được kết hợp từ hai thuật toán Skip-gram và Continuous bag of words (CBOW).

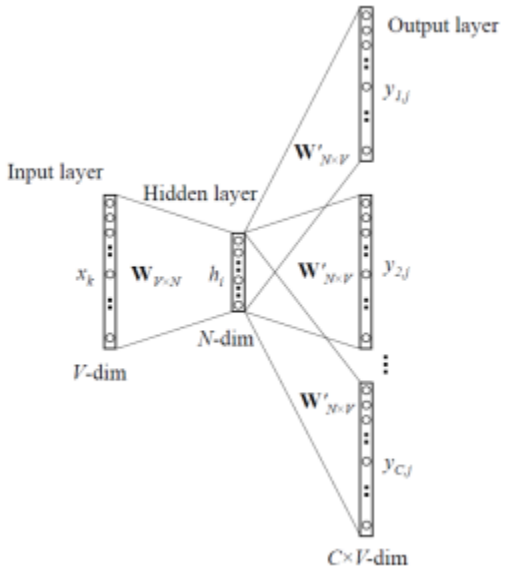
1.1.3.1. Skip-gram:

Ý tưởng chính của mô hình này là xác định các từ xung quanh từ mục tiêu trong một khoảng nhất định gọi là 'window'.



Hình 2: Hình Mô Tả Trainning Với Window Bằng 2. (Nguồn: Leonardo Barazza)

Đối với Skip-gram, đầu vào là từ đích, trong khi đầu ra là các từ xung quanh từ đích. Tất cả dữ liệu đầu vào và đầu ra có cùng kích thước được mã hóa bằng one-hot. Mạng chứa một lớp ẩn có kích thước bằng kích thước nhúng, nhỏ hơn vector đầu vào và đầu ra. Ở cuối lớp đầu ra, một hàm kích hoạt softmax được áp dụng sao cho mỗi phần tử của vector đầu ra mô tả khả năng một từ cụ thể sẽ xuất hiện trong ngữ cảnh.



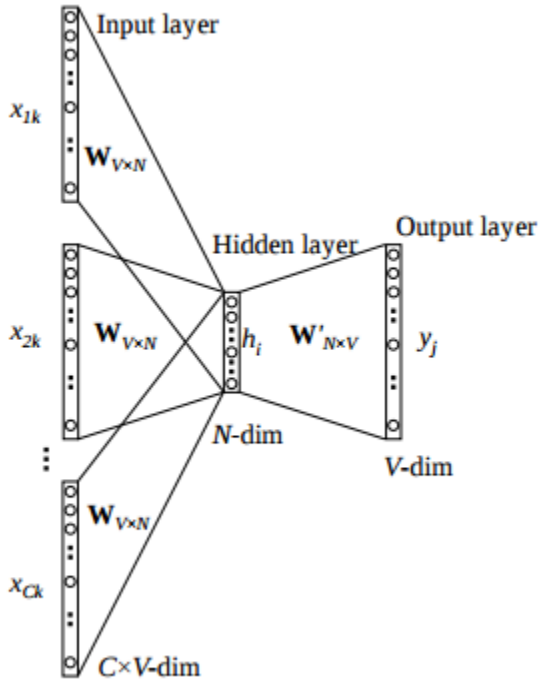
Hình 3: Hình mô tả cấu trúc mạng của Skip-gram

(Nguồn https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/)

Với skip-gram, kích thước biểu diễn từ giảm từ kích thước bằng số từ trong bộ từ vựng xuống bằng chiều dài lớp ẩn. Hơn nữa các vector có ý nghĩa nhiều hơn về mặt mô tả mối quan hệ giữa các từ.

1.1.3.2. Continuous bag of words (CBOW).

Ngược lại với Skip-gram nó hoán đổi đầu vào và đầu ra, ý tưởng của thuật toán CBOW là đưa ra một bối cảnh và cho biết từ nào có khả năng xuất hiện nhiều nhất trong đó.



Hình 4: Mô Tả Mạng CBOW

(Nguồn https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/)

Đối với CBOW, tất cả các ví dụ với từ đích là mục tiêu được vào mạng, sau đó lấy trung bình trong lớp ẩn. Giả sử chúng ta chỉ có hai câu, “Anh ấy là một chàng trai tốt” và “Cô ấy là một cô gái tốt”. Để tính toán đại diện cho từ “một”, chúng ta cần cung cấp hai ví dụ này cho mạng nơ-ral và lấy giá trị trung bình giá trị trong lớp ẩn.

1.1.4. Greedy Search và Beam Search (Thuật toán tìm kiếm tham lam và thuật toán tìm kiếm chùm tia)

Thuật toán Greedy Search chọn một ứng cử viên tốt nhất làm chuỗi đầu vào cho mỗi bước thời gian(ứng cử viên có xác suất cao nhất). Chọn chỉ một ứng cử viên tốt nhất có thể phù hợp với bước thời gian hiện tại, nhưng khi xây dựng câu đầy đủ, nó có thể là một lựa chọn tối ưu. Tuy nhiên thì xác suất cao nhất ở bước hiện tại chưa chăc sẽ cho ra xác xuất cao nhất ở bước tiếp theo, vậy nên thay vì chỉ giữ 1 kết quả có xác xuất cao nhất chúng ta sẽ giữ lại k kết quả có xác suất cao nhất và đó chính là Beam Search.

Thuật toán Beam Search chọn nhiều lựa chọn thay thế cho chuỗi đầu vào tại mỗi dấu thời gian dựa trên xác suất có điều kiện. Số lượng nhiều lựa chọn thay thế phụ thuộc vào một tham số gọi là Beam Width B. Ở mỗi bước thời gian, tìm kiếm chùm tia chọn B số lựa chọn thay thế tốt nhất với xác suất cao nhất là lựa chọn khả dĩ nhất cho bước thời gian(ví dụ với Beam Width B = 3 thì tại mỗi bước thời gian sẽ chọn ra 3 ứng cử viên có xác suất cao nhất làm đầu vào cho bước thời gian tiếp theo(t). Sau đó lại tiếp tục chọn 3 ứng cử viên ở bước thời gian tiếp theo(t+1) làm đầu vào ở bước thời gian t+2. Cứ như vậy cho đến cuối cùng ta sẽ thu được 3 kết quả và chọn ra kết quả từ đó). Thuật toán Beam Search nếu có Beam Width B = 1 thì nó trở thành thuật toán Greedy Search. Beam Width B = 10 thường được sử dụng và mang lại hiệu quả đủ tốt.

1.1.5. Bleu Score

BLEU là một thuật toán để đánh giá chất lượng văn bản đã được dịch bằng máy từ ngôn ngữ tự nhiên này sang ngôn ngữ tự nhiên khác. Chất lượng được coi là sự tương ứng giữa đầu ra của máy và của con người: "bản dịch máy càng gần với bản dịch chuyên nghiệp của con người thì càng tốt" - đây là ý tưởng trung tâm của BLEU.

Bilingual Evaluation Understudy Score hay ngắn gọn là BLEU score là một thang điểm được dùng phổ biến trong đánh giá Machine Translation. BLEU được Kishore Papineni và cộng sự đề xuất lần đầu vào năm 2002 qua bài nghiên cứu "a Method for Automatic Evaluation of Machine Translation".

BLEU được tính dựa trên số lượng n-grams[1] giống nhau giữa câu dịch của mô hình (output) với các câu tham chiếu tương ứng (reference) có xét tới yếu tố độ dài của câu.

Số n-grams tối đa của BLEU là không giới hạn, nhưng vì xét về ý nghĩa, cụm từ quá dài thường không có nhiều ý nghĩa, và nghiên cứu cũng đã cho thấy là với 4-gram, điểm số BLEU trung bình cho khả năng dịch thuật của con người cũng đã giảm khá nhiều nên n-grams tối đa thường được sử dụng là 4-gram.

1.2. Lý thuyết nền tảng mạng nơ-ron (Neural Networds)

1.2.1. Mô Tả mạng nơ-ron:

Mạng nơ-ron là một tập hợp các mô hình toán học được được xây dựng dựa trên tập hợp các nút, được kết nối với các hàm kích hoạt phi tuyến tính cùng các tham số có khả năng học. Mạng nơ-ron hiện là mô hình phổ biến nhất được sử dụng cho các ứng dụng máy học trong một loạt các lĩnh vực như thị giác máy tính, nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên,…

Một tế nào (nút) của mạng nơ-ron là một hàm của tập các trọng số tương ứng với các giá trị đầu vào (inputs) $\{x_0, \dots, x_N\}$.

$$y = a\left(\sum_i^N w_i x_i + b\right)$$

Trong đó:

- w_i : trọng số của đầu vào x_i
- a: hàm kích hoạt (activation function)
- b: độ sai lệch (bias)

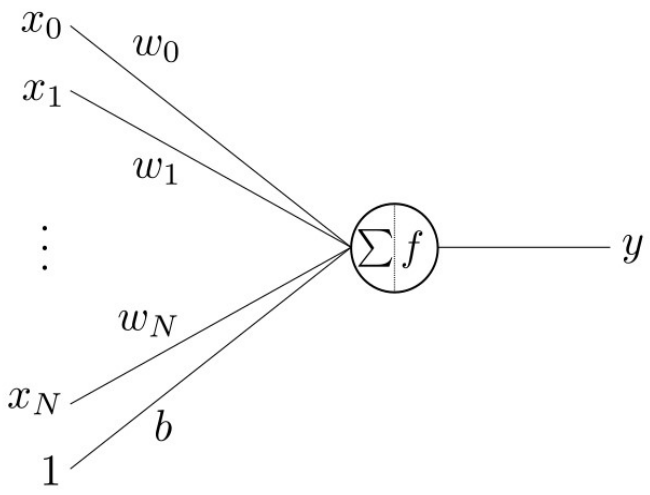
Ta sẽ sử dụng kí hiệu ma trận để làm đơn giản cách thể hiện, trong đó mỗi tế bào nơ-ron bao gồm một vector đầu vào $x = \{x_0, \dots, x_N\}$, một vector trọng số $w = \{w_0, \dots, w_N\}$ và một vector sai lệch b, khi đó đầu ra là:

$$y = a(w^T x + b)$$

Nếu hàm kích hoạt a là một biến thể của hàm Heaviside,

$$a(x) = \begin{cases} 1, & x \geq 0 \\ 0 \text{ hoặc } -1, & x < 0 \end{cases}$$

thì tế bào nơ-ron này được gọi là một perceptron, một bộ phân loại nhị phân đơn giản, là một trong những phương pháp học kết nối sớm nhất được phát minh bởi Rosenblatt.



Hình 5: Minh họa kiến trúc điển hình của một tế bào mạng nơ-ron

1.2.2. Hàm kích hoạt (Activation function)

Hàm kích hoạt là phần rất quan trọng trong mạng nơ-ron, đặc biệt là mạng nơ-ron nhiều lớp ẩn. Nếu không có hàm kích hoạt phi tuyến tính, cho dù mạng nơ-ron có nhiều lớp ẩn đến cỡ nào thì cũng chỉ có sức mạnh đại diện cho phân loại tuyến tính, điều này tương đương với một mạng mà không có lớp ẩn nào. Vì bản chất tổng hợp các hàm tuyến tính là một hàm tuyến tính. Do đó, hàm kích hoạt a là một hàm phi tuyến tính được áp dụng cho đầu

ra tại mỗi nút và input data cho tầng tiếp theo, cho phép mạng nơ-ron nhiều lớp ẩn học các hàm phi tuyến phức tạp.

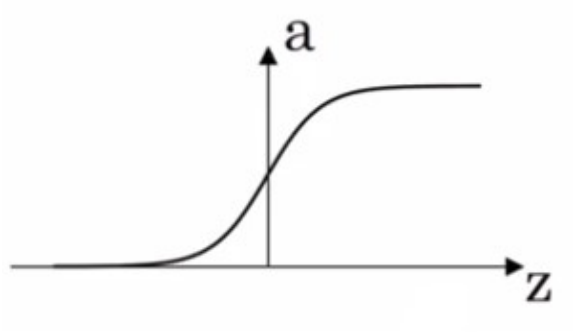
Hàm kích hoạt phổ biến là hàm sigmon, nó là một hàm phi tuyến với đầu vào là các số thực cho kết quả nằm trong khoảng từ 0 đến 1, phù hợp cho các mạng phân loại nhị phân (nổi tiếng như là thuật toán Logistic Regression), nó có công thức theo phương trình:

$$z^{[i]} = W^{[i]}x + b^{[i]}$$

$$a = \frac{1}{1 + e^{-z}}$$

Trong đó:

- $W^{[i]}$: trọng số tại lớp thứ i
- $b^{[i]}$: tham số sai lệch tại lớp thứ i
- a: hàm kích hoạt

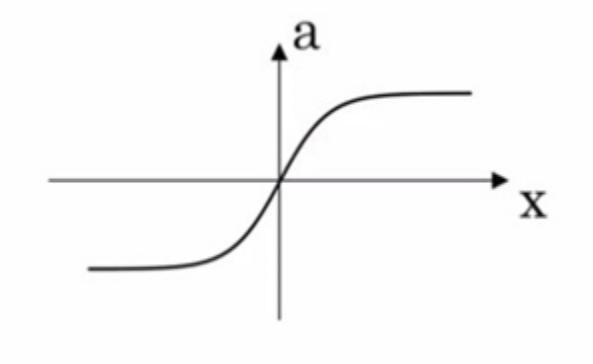


Hình 6: Minh họa hàm kích hoạt sigmoid. (Nguồn: Coursera Sequence Models)

Ngoài ra, có một hàm kích hoạt luôn hoạt động tốt hơn hàm sigmoid là hàm tiếp tuyến hyperbolic (hyperbolic tangent

function), ánh xạ các giá trị đầu vào vào khoảng biên từ -1 đến 1. Có công thức là:

$$a = \tanh(x) = \frac{e^{z} - e^{-z}}{e^{z} + e^{z}}$$

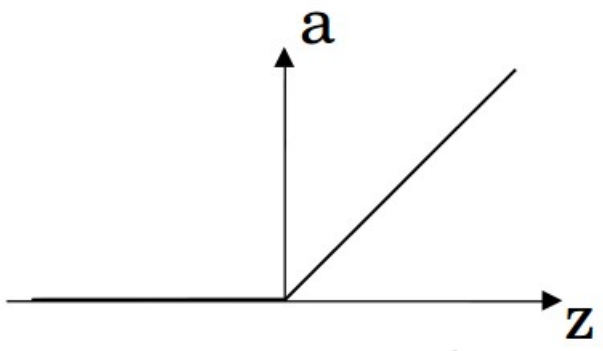


Hình 7 Minh họa hàm kích hoạt tanh. (Nguồn: Coursera Sequence Models)

Tuy nhiên, một vấn đề với hàm kích hoạt sigmoid và tanh nếu z quá lớn hoặc quá nhỏ thì độ dốc của hàm sẽ rất nhỏ, điều này làm chậm quá trình tìm điểm cực tiểu của hàm chi phí, dẫn đến làm chậm quá trình học. Vì lý do này, dựa vào các kết quả thực nghiệm được cải thiện, mạng nơ-ron hiện đại có xu hướng sử dụng hàm kích hoạt đơn vị tuyến tính chỉnh lưu (ReLu - Rectified Linear Unit). Có công thức như sau:

$$z^{[i]} = W^{[i]}x + b^{[i]}$$

$$a = max\ (0,z)$$



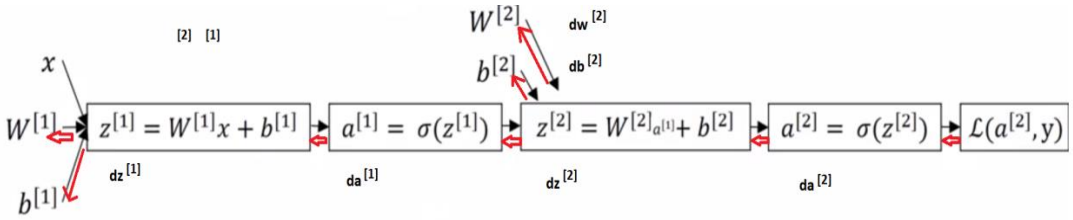
Hình 8: Minh họa hàm kích hoạt ReLu. (Nguồn: Coursera Sequence Models)

Vì vậy, đạo hàm luôn bằng 1 nếu z dương, và bằng 0 nếu z âm. Dựa trên thực nghiệm, sử dụng hàm kích hoạt ReLu, mạng nơ-ron sẽ học nhanh hơn so với khi dùng với hàm sigmoid hoặc hàm tanh. Lý do chính là có ít hơn sự ảnh hưởng của độ dốc hàm bằng 0 làm chậm việc học. Vì mặc dù, có một nửa phạm vi của z làm độ dốc hàm ReLu bằng 0, nhưng trong thực tế, đủ các đơn vị ẩn thì ta sẽ có z lớn hơn 0, vì vậy việc học vẫn khá nhanh với hầu hết các ví dụ đào tạo.

1.2.3. Lan truyền ngược (Back propagation)

Các thuật toán học sâu tương phản với các thuật toán học nông bởi số biến đổi được tham số hóa một tín hiệu gặp phải khi nó lan truyền từ các lớp đầu vào đến các lớp đầu ra. Mỗi chuỗi các biến đổi từ đầu vào đến đầu ra gọi là một đường gán kế thừa (CAP - Credit Assignment Path). Vấn đề gán kế thừa (Credit Assignment Problem) được giải quyết với khám phá lan truyền ngược (backpropagation), cho phép học với mạng nơ-ron nhiều lớp. Sau đây, nhóm sinh viên sẽ trình bày ý tưởng của quá trình lan truyền ngược.

Hình 9 minh họa một mạng nơ-ron 2 lớp, gồm một lớp đầu vào, một lớp ẩn và một lớp đầu ra.



Hình 9: Minh họa một mạng nơ-ron 2 lớp (Nguồn: Coursera Sequence Model)

Lưu ý ở lan truyền tiến, các bước tính toán là như sau: đầu tiên ta tính toán $z^{[1]}$, sau đó tính toán $a^{[1]}$, rồi tính toán $z^{[2]}$ ghi chú là $z^{[2]}$ cũng phụ thuộc vào các tham số $W^{[2]}$ và $b^{[2]}$, sau đó dựa vào $z^{[2]}$ tính toán $a^{[2]}$ và cuối cùng là tính toán chi phí. Đối với lan truyền ngược, ta sẽ đi tính toán theo chiều ngược lại, cụ thể là tính toán $da^{[2]}$, sau đó tính $dz^{[2]}$, quay ngược lên tính $dw^{[2]}$ và $db^{[2]}$, tương tự ta tính toán tiếp cho các biến $da^{[1]}$, $dz^{[1]}$, $dW^{[1]}$, $db^{[1]}$. Thông thường, ta sẽ bỏ qua tính đạo hàm của $da^{[2]}$, thay vào đó nhập lại thành một bước là tính trên $dz^{[2]}$. Sau cùng ta rút ra các công thức sau:

$$da^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]}a^{[1]^T}$$

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]^T}dz^{[2]} * g^{[1]'}(z^{[1]})\ dW^{[1]} = dz^{[1]}x^T$$

$$db^{[1]} = dz^{[1]}$$

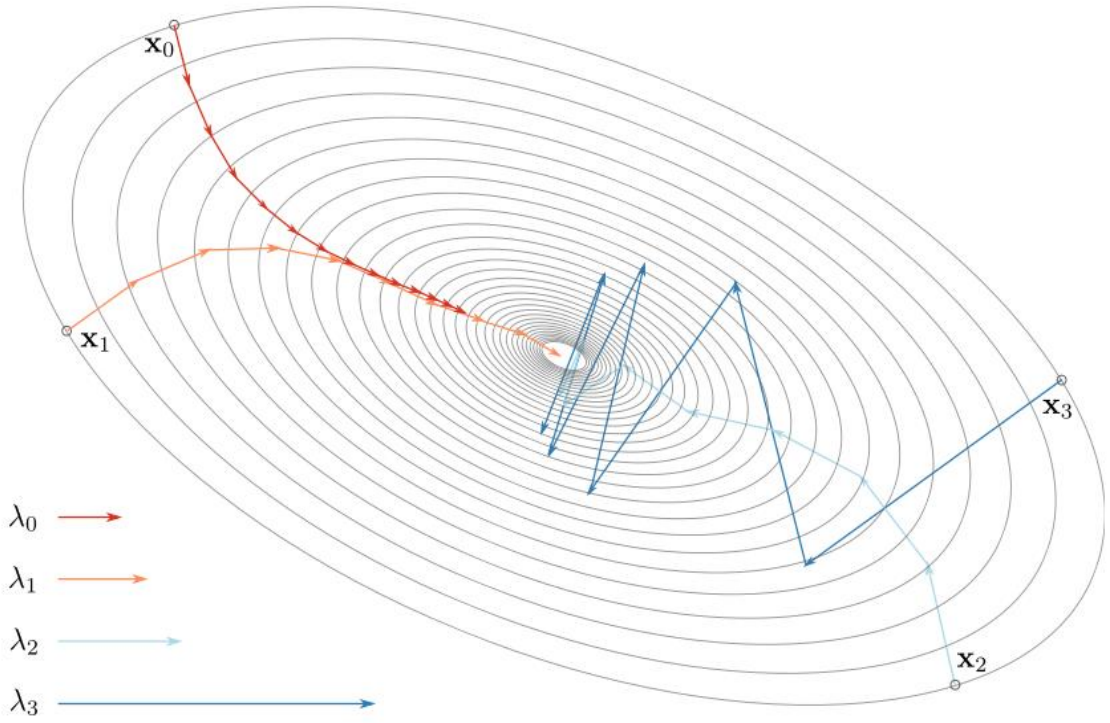
1.2.4. Học với lan truyền ngược

Học với lan truyền ngược giống như quy tắc delta, các độ nhạy được dùng để điều chỉnh trọng số tỷ lệ với một hằng số tỉ lệ học (learning rate) α. Việc cập nhật trọng số thực hiện theo công thức sau:

$$\Delta w_{ij}^n = -\alpha \frac{\partial E^n}{\partial w_{ij}}$$
$$= -\alpha \delta_j^n y_i$$

Trong đó:

- δ_j^n: đã được định nghĩa ở công thức
- yi: đầu ra ở nơ-ron i



Hình 10 Minh họa sự ảnh hưởng của tỉ lệ học và chính sách học lên độ hội tụ với lan truyền ngược. (Nguồn: [2])

Lan truyền ngược là một phương pháp gốc dốc nhất (steepest descent). Hình 10 minh họa quy tắc học với lan truyền ngược, đặc trưng cho kích thước mỗi bước là tham số tỉ lệ học. Tham số tỉ lệ học thay đổi kích thước bước hay độ lớn của vector thay đổi trọng số. Hình 10 cũng minh hoạ độ ảnh hưởng của tỉ lệ học trên độ giảm dốc. Tỉ lệ học quá nhỏ làm cho kết quả học rất chậm như a_0, tuy nhiên tỉ lệ học quá lớn lại làm cho bước nhảy xung quanh khu vực điểm cực tiểu và làm mất thời gian để tiếp cận điểm cực tiểu này như a_2 và a_3.

Để đạt được điểm cực tiểu cục bộ, tỉ lệ học cũng nên giảm dần trong quá trình huấn luyện. Tuy nhiên nếu giảm quá nhanh, nó có thể không đạt được lưu vực gần với điểm cực tiêu như a_0, ngược lại nếu giảm quá chậm, nó có thể mất một khoảng thời gian dài để tiến vào lưu vực này như a_2 và a_3.

Cân bằng việc cố gắng tìm một tỉ lệ học và chính sách học phù hợp không may là một phần "ma thuật đen" đằng sau việc huấn luyện DNN đến từ kinh nghiệm. Tuy nhiên Bottou (2012) và I. Goodfellow, Y.Bengio, và Courville (2016) là tài liệu tham khảo tuyệt vời về một số các tiếp cận phổ biến để làm công việc này dễ dàng hơn.

1.2.5. Phương pháp giảm độ dốc với Gradient desent và các biến thể

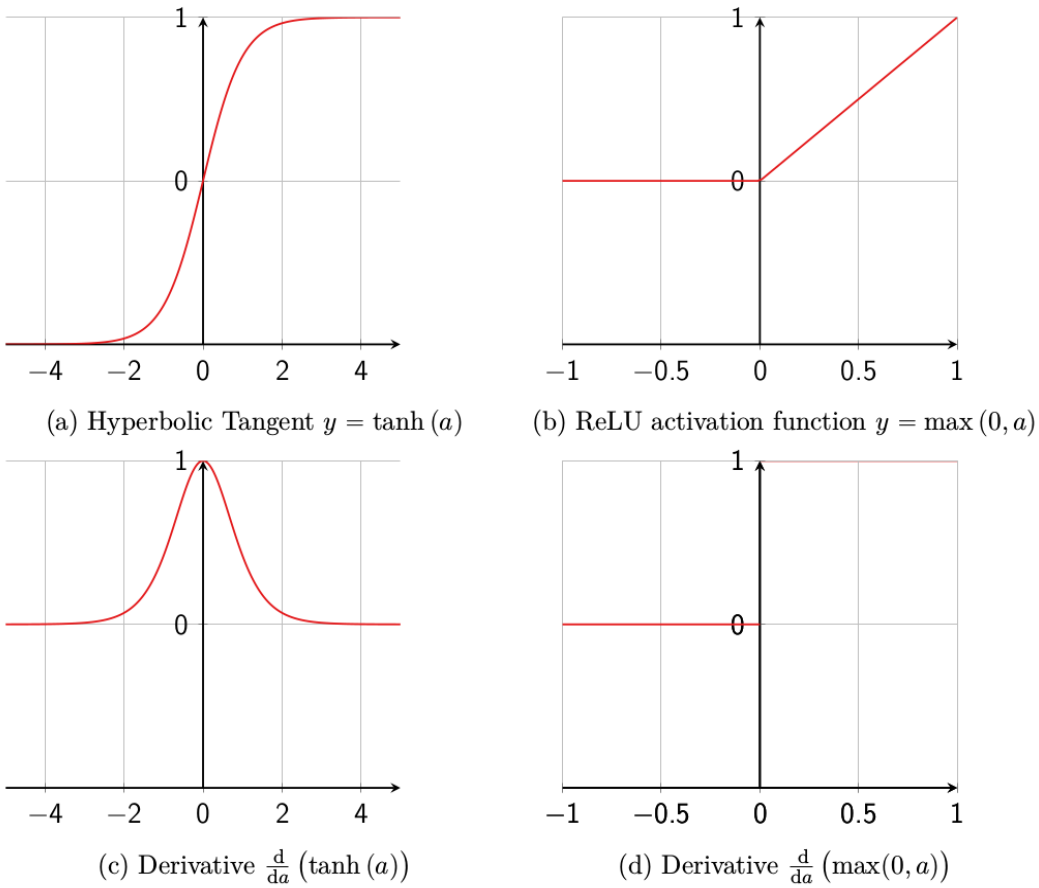
1.3. Các phương pháp huấn luyện mạng nơ-ron hiện đại

1.3.1. Hàm kích hoạt đơn vị tuyến tính chỉnh lưu (Rectified Linear Unit)

Một phần không thể thiếu của bất kỳ mạng nơ-ron nào là hàm kích hoạt phi tuyến tính. Trong lịch sử, mạng nơ-ron đã từng sử dụng chức năng kích hoạt dạng sigmoid. Tuy nhiên một vấn đề lớn với các hàm kích hoạt dạng sigmoid là độ dốc bên ngoài một vùng tương đối hẹp trên miền hàm là một số rất nhỏ. Khi huấn luyện với lan truyền ngược (backpropagation) điều này có nghĩa là hầu hết độ dốc có độ lớn rất nhỏ và việc huấn luyện có thể sẽ tốn một thời gian rất dài, hoặc thậm chí bị đình trệ hoàn toàn - tình huống này được gọi là độ dốc biến mất (vanishing gradient).
ReLu được đề xuất như một giải pháp, đầu tiên cho các máy Boltzmann bị hạn chế (Nair và Geoffrey E. Hinton, 2010), sau đó là cho các mạng nơ-ron ((Glorot và Y. Bengio, 2010), trong đó về mặt thực nghiệm, nó được chứng minh là cho phép đào tạo dễ dàng hơn với lan truyền ngược.

$$z^{[i]} = W^{[i]}x + b^{[i]}$$

$$a = max\ (0,z)$$



(a) Hyperbolic Tangent $y = \tanh(a)$ (b) ReLU activation function $y = \max(0, a)$

(c) Derivative $\frac{\mathrm{d}}{\mathrm{d}a}(\tanh(a))$ (d) Derivative $\frac{\mathrm{d}}{\mathrm{d}a}(\max(0, a))$

Hình 2.10: Minh họa các hàm kích hoạt thường dùng trong mạng nơ-ron và đạo hàm tương ứng của nó. (Nguồn: [2])

ReLu không thể hiện độ bảo hòa như các hàm dạng sigmoid, nó luôn cho độ dốc có giá trị 0 hoặc 1, được minh họa ở hình 2.10. Trong thực tế, điều này có thể làm tăng tốc độ đào tạo thậm chí cho phép các mạng mà không thể đào tạo thực tế với hàm kích hoạt dạng sigmoid chẳng hạn như mạng học sâu Krizhevsky, Sutskever và Geoffrey E. Hinton (2012).

1.3.2. Phương pháp chuẩn hóa theo lô (Batch Nomalization)

1.3.3. Phương pháp cắt giảm (Dropout)

Geoffrey E. Hinton, Srivastava, et al. (2012a) và Srivastava et al. (2014) đã giới thiệu phương pháp cắt giảm (dropout), một phương pháp ngăn chặn vượt mức (overfit) trong các mạng lớn khi đào tạo. Ý tưởng chính là: Trong quá trình huấn luyện ta loại bỏ một tập các nút nơ-ron, được lấy mẫu ngẫu nhiên từ mỗi lớp với xác suất cố định p.

Cơ chế ảnh hưởng của phương pháp cắt giảm được giải thích theo nhiều cách khác nhau, nhưng đáng chú ý nhất là các cách giải thích được đưa ra bởi Geoffrey E. Hinton, Srivastava, et al. (2012a) và Srivastava et al. (2014).

Giải thích của Geoffrey E. Hinton, Srivastava, et al. (2012a) là phương pháp cắt giảm là một hình thức chính quy hóa bằng độ ồn, ngăn chặn sự thích nghi của các tế bào nơ-ron. Giải thích chính được đưa ra bởi Srivastava et al. (2014) rằng phương pháp cắt giảm là một hình thức tích hợp mô hình, tính trung bình trên một số lượng lớn các kiến trúc mô hình "mỏng hơn" ngẫu nhiên tại thời điểm đào tạo để cải thiện khái quát hóa. Tuy nhiên, việc tính trung bình trên tất cả các mô hình được xem xét trong suốt quá trình đào tạo là cực kỳ tốn kém, bởi vì số lượng mô hình có thể có tăng theo cấp số mũ.

1.4. Các kiến trúc mạng nơ-ron hồi quy:

Một danh sách nghiên cứu đầy đủ của mọi kiến trúc học sâu DNN là điều không khả thi và nằm ngoài phạm vi của luận án này, tuy nhiên ở đây, nhóm sinh viên đã nỗ lực khái quát về những kiến trúc nơ-ron hồi quy nổi bật trong những năm gần đây đồng thời truyền cảm hứng cho việc thiết lập kiến trúc hệ thống dịch trong các chương sau của nhóm.

1.4.1. Mạng nơ-ron hồi quy (RNN – Recurrent Neural Network)

Mạng nơ-ron hồi quy (RNN - Recurrent Neural Network) là mạng thần kinh lan truyền tới (Feedforward Neural Networks) [11] được tăng cường bằng cách bao gồm các cạnh mà kéo dài các bước thời gian liền kề, đưa ra khái niệm về thời gian cho mô hình.

Giống như mạng lan truyền tới, mạng RNN không có chu trình giữa các cạnh thông thường. Tuy nhiên, các cạnh mà kết nối các bước thời gian liền kề, được gọi là cạnh hồi quy, có thể hình thành chu trình có độ dài bằng một, tự kết nối từ nột nút(node) tới chính nó theo thời gian. Tại thời điểm t, nút với cạnh hồi quy nhận đầu vào từ điểm dữ liệu hiện tại $x^{(t)}$ và từ các giá trị nút ẩn $h^{(t-1)}$ trong trạng thái trước đó của mạng. Đầu ra $\hat{y}^{(t)}$ cho mỗi thời điểm t được tính bằng các giá trị nút ẩn $h^{(t)}$ tại thời điểm t. Đầu vào $x^{(t-1)}$ tại thời điểm t-1 có thể ảnh hưởng đến giá trị đầu ra $\hat{y}^{(t)}$ tại thời điểm t và sau đó, bằng các kết nối hồi quy.

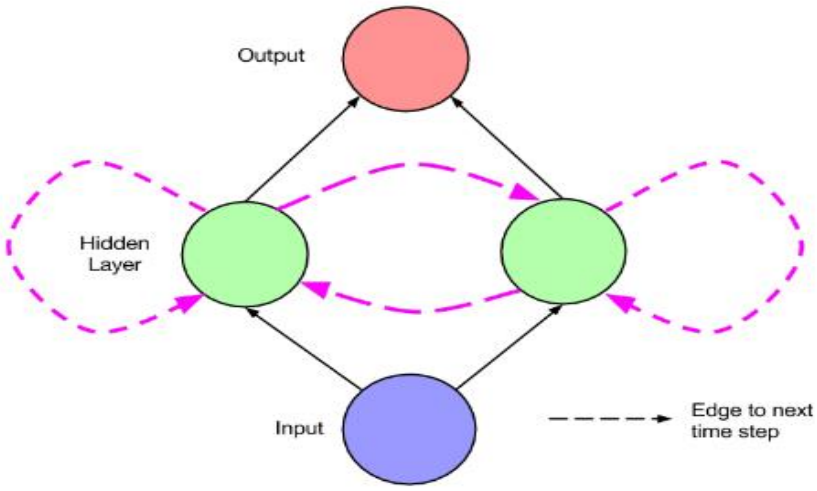
Các tính toán cần thiết tại mỗi bước thời gian trên đường là truyền tới của mạng RNN đơn giản (hình 2.11), được thể hiện như sau:

$$\mathrm{h}(t) = \sigma(W^{\mathrm{h}x} + W^{\mathrm{hh}}\,\mathrm{h}(t-1) + b_{\mathrm{h}})$$

$$\hat{y}(t) = softmax(W^{yh}\, h(t) + b_y)$$

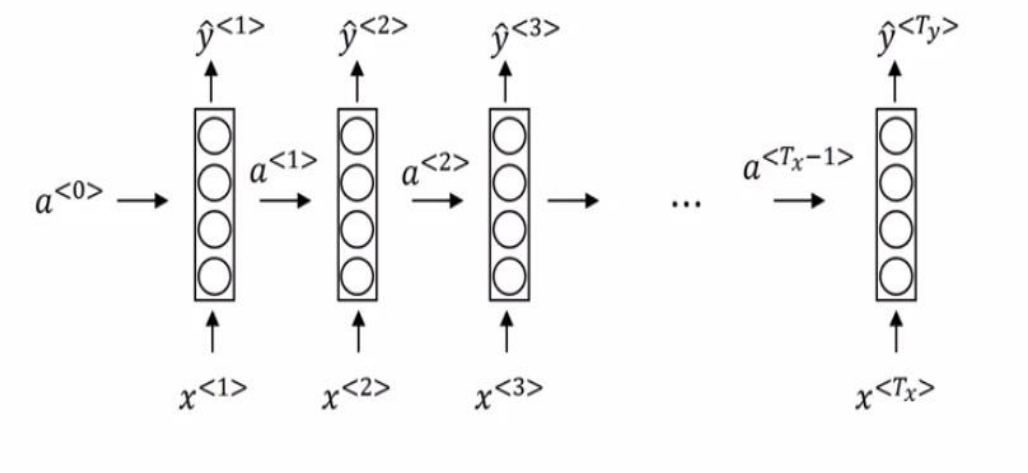
Trong đó:

- W^{hx} là ma trận trọng số thông thường giữa đầu vào và lớp ẩn
- W^{hh} là ma trận trọng số hồi quy giữa lớp ảnh và chính nó ở các bước thời gian liền kề
- Các vector b_h và b_y là các tham số sai lệ



Hình 2.11 Mạng RNN đơn giản (Nguồn: [3])

Một biểu diễn dễ hiểu cho hình 2.11 được thể hiện ở hình 2.12, trong đó các bước thời gian được mở ra. Với hình ảnh này, mạng được hiểu không phải là chu trình, mà là một mạng học sâu với mỗi lớp tương ứng với mỗi bước thời gian, được chia sẻ trọng số qua các lớp.



Hình 2.12 Minh họa mạng RNN được mở ra từ hình 2.11 (Nguồn: Coursera Sequence Models)

Bây giờ lấy ví dụ, xét bài toán: Xác định chữ nào là một phần của tên người trong một câu.

Đọc một câu từ trái sang phải, đánh số mỗi từ trong câu là x_i ($1 \leq i \leq N$). Tiến trình nạp từ thứ nhất x_1 vào một lớp ẩn mạng nơ-ron, lớp này sẽ dự đoán kết quả $\hat{y}^{<1>}$ xem x_1 có phải là một phần của tên người hay không. Xét từ thứ hai x_2 thay vì chỉ dự đoán $\hat{y}^{<2>}$ từ x_2, thì lớp này nhận thêm giá trị kích hoạt $a^{<1>}$ từ bước 1. Các bước sau được thực hiện tương tự, cho đến khi kết thúc câu. Thông thường ở bước đầu tiên cũng được truyền thêm vào giá trị kích hoạt với $a^{<0>}$ (là một vector 0). Hình 2.12 minh hoạ mạng RNN lan truyền tới với giá trị $a^{<t>}$, $\hat{y}^{<t>}$ được tính theo công thức như sau:

$$a^{<t>} = g(W_{aa}\, a^{<t-1>} + W_{ax}\, x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g\,(W_{ya}\, a^{<t>} + b_y)$$

RNN duyệt dữ liệu từ trái sang phải và tham số dùng cho mỗi bước là được chia sẻ. Vì vậy, khi dự đoán kết quả $\hat{y}^{<3>}$, RNN không chỉ sử dụng đầu vào x_3 mà còn sử dụng thông tin từ x_1 và x_2.

Xét hai trường hợp đầu vào sau:

Trường hợp một:

(1) Anh ấy nói "*Teddy Roosevelt là một tổng thống tuyệt vời*".

(2) Anh ấy nói "*Teddy là loại gấu bông được mua nhiều nhất ở cửa hàng này*".

Trường hợp hai:

Thời thơ ấu, tôi thường nghe bố tôi nhắc tới một người anh hùng, ... bố tôi đang nhắc tới Phan Đình Giót.

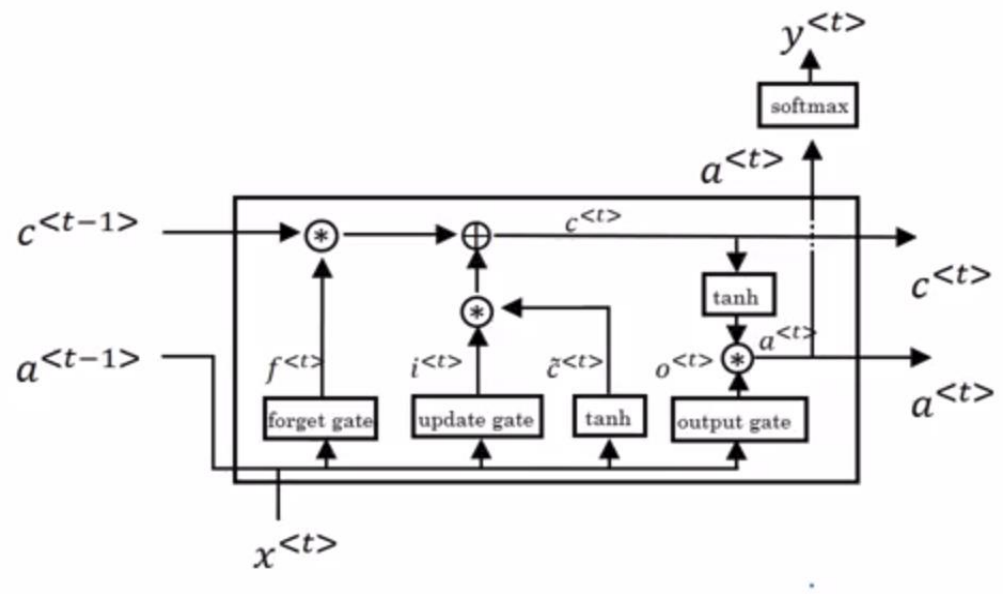
Ở trường hợp một, "*Teddy*" là một phần tên người trong câu (1), còn câu (2) thì không phải. Như vậy, một điểm yếu của mạng RNN là chỉ dùng thông tin từ các bước phía trước trong chuỗi để thực hiện dự đoán kết quả, thông tin dùng để dự đoán này là không đủ. Để khắc phục nhược điểm này, mạng nơ-ron hồi quy hai chiều (BiRNN - Bidirectional recurrent neural network) ra đời.

Còn ở trường hợp hai, cụm từ "*Phan Đình Giót*" có được dự đoán là tên của người hay không phụ thuộc thông tin được mang đến từ cụm "*người anh hùng*". Nhưng mạng RNN bị hạn chế trong duy trì phụ thuộc tầm xa, nghĩa là nếu chuỗi đầu vào đủ dài, mạng đủ sâu, thì RNN khó khăn trong việc mang thông tin từ các bước trước tới các bước sau, thậm chí RNN có thể bỏ qua thông tin quan trọng đến từ những bước đầu tiên. Đây là vấn đề biến mất độ dốc (vanishing gradient).

1.4.2. Mạng bộ nhớ dài ngắn (LSTM - Long Short Term Memory)

Như đã trình bày ở mục 1.4.1, mạng RNN bị vấn đề độ dốc biến mất (vanishing gradient). Bản chất của mô hình mô hình RNN có nhiều ảnh hưởng cục bộ, nghĩa là đầu ra $\hat{y}^{<3>}$ sẽ bị ảnh hương bởi các giá trị gần $\hat{y}^{<3>}$ như x<1>, x<2>, x<3>, đầu ra vị trí cuối chuỗi $\hat{y}^{<T>}$ sẽ bị ảnh hưởng bởi các giá trị gần T. Điều này gây nên khó khăn nếu muốn đầu ra $\hat{y}^{<T>}$ bị ảnh hưởng mạnh mẽ bỏi những giá trị ở vị trí đầu chuỗi (hình 2.12).

Mạng bộ nhớ dài-ngắn (Long Short-Term Memory - LSTM) được tạo ra để giải quyết vấn đề duy trì phụ thuộc tầm xa. Mạng LSTM giống với mạng RNN tiêu chuẩn, nhưng với mỗi nút gốc trong lớp ẩn sẽ được thay thế bằng một ô nhớ (memory cell) và sử dụng thêm ba cổng riêng biệt là cổng quên (forget gate), cổng vào (input gate), cổng ra (output gate) để điều chỉnh luồng thông tin trong một ô LSTM.



Hình 2.13 Mình họa một đơn vị LSTM (Nguồn: Coursera Sequence Models)

Cơ chế điều chỉnh luồng thông tin và cập nhật trạng thái ô nhớ được thể hiện như sau:

❖ Cổng quên (forget gate)

Cổng này quyết định thông tin nào là quan trọng để giữ lại từ bước phía trước. Thông tin từ lớp ẩn trước kết hợp với thông tin đầu vào hiện tại, sau đó được truyền qua hàm sigmoid, giá trị được chuyển đổi vào đường biên từ 0 đến 1. Giá trị càng gần 0 thì bị bỏ qua, càng gần 1 thì được giữ lại. Công thức tính toán giá trị cổng quên:

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

❖ Cổng vào (input gate)

Cổng này quyết định thông tin nào là quan trọng để thêm vào từ bước hiện tại. Kết hợp thông tin từ lớp ẩn trước và thông tin đầu vào hiện tại cho truyền qua hàm sigmoid. Hàm này sẽ quyết định giá trị nào sẽ được cập nhật, bằng cách chuyển giá trị về khoảng biên từ 0 đến 1, gần 0 là không quan trọng, gần 1 là quan trọng. Đồng thời cũng truyền thông tin từ lớp ẩn trước và thông tin đầu vào hiện tại qua hàm tanh, giá trị được chuyển về khoảng biên từ -1 đến 1. Sau đó nhân đầu ra sigmoid với

đầu ra tanh, để quyết định thông tin nào quan trọng để giữ lại từ đầu ra hàm tanh. Công thức tính giá trị cổng vào:

$$\Gamma_{\mathrm{i}} = \sigma(W_{\mathrm{i}}[a^{<t-1>}, x^{<t>}] + b_{\mathrm{i}})$$

❖ Trạng thái ô (cell status)

Bây giờ, ta đã đủ thông tin để tính toán trạng thái mới của ô. Trạng thái mới được tính toán bằng trạng thái ô đang xét nhân với vector đầu ra của cổng quên để loại bỏ một số giá trị của ô trạng thái nếu nhân với giá trị gần 0, sau đó cộng giá trị này với đầu ra của cổng vào. Công thức để tính trạng thái ô hiện tại:

$$\tilde{c}^{<t>} = tan\mathrm{h}(W_{\mathrm{c}}[a^{<t-1>}, x^{<t>}] + b_{\mathrm{c}})$$

$$c^{<t>} = \Gamma * \tilde{c}^{<t>} + \Gamma_{\mathrm{f}} * c^{<t-1>}$$

❖ Cổng ra (output gate)

Cổng này quyết định trạng thái ẩn truyền cho bước tiếp theo là gì. Truyền trạng thái lớp ẩn bước trước và giá trị đầu vào hiện tại vào hàm sigmoid. Sau đó đưa trạng thái mới của ô đã cập nhật qua hàm tanh, nhân giá trị đầu ra của hàm sigmoid và giá trị đầu ra của hàm tanh để quyết định những thông tin trạng thái ẩn nào nên mang theo, kết quả này và trạng thái mới của ô được truyền tới bước tiếp theo. Công thức tính giá trị cổng ra:

$$\Gamma_{\mathrm{o}} = \sigma(W_{\mathrm{o}}[a^{<t-1>}, x^{<t>}] + b_{\mathrm{o}})$$

1.4.3. Mạng nơ-ron hồi quy hai chiều (BIRNN - Bidirectional Recurrent Neural Network)

Cùng với mạng LSTM, một trong những kiến trúc RNN được sử dụng nhiều nhất là mạng nơ-ron hồi quy hai chiều (BiRNN - Bidirectional recurrent neural network), khắc phục nhược điểm chỉ nhận thông tin từ các bước thời gian phía trước để dự đoán kết quả bước hiện tại.

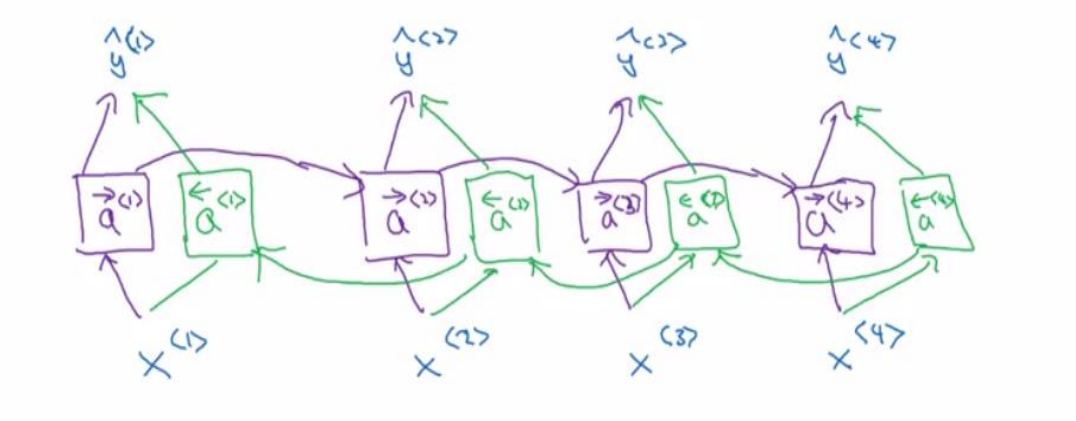
Trong kiến trúc này, có hai lớp nút ẩn. Cả hai lớp ẩn đều được kết nối với đầu vào và đầu ra. Hai lớp này được phân biệt ở chỗ, lớp đầu tiên có các kết nối hồi quy đến từ các bước thời gian trước, trong khi ở lớp thứ hai, hướng hồi quy của các kết nối bị đảo ngược, nghĩa là truyền giá trị kích hoạt ngược theo chiều chuỗi (hình 2.14). Ba phương trình sau mô tả một BiRNN.

$$\mathrm{h}^{(t)} = \sigma(W^{\mathrm{h}x}x^{(t)} + W^{\mathrm{hh}}\,\mathrm{h}^{(t-1)} + b_{\mathrm{h}})$$

$$z^{(t)} = \sigma(W^{zx}x^{(t)} + W^{zz}\,z^{(t+1)} + b_z)$$

$$\hat{y}^{(t)} = softmax(W^{y\mathrm{h}}\,\mathrm{h}^{(t)} + W^{yz}\,z^{(t)} + b_y)$$

Trong đó $\mathrm{h}^{(t)}$ và $z^{(t)}$ là các giá trị của lớp ẩn theo hai hướng lan truyền tới và lan truyền ngược tương ứng.

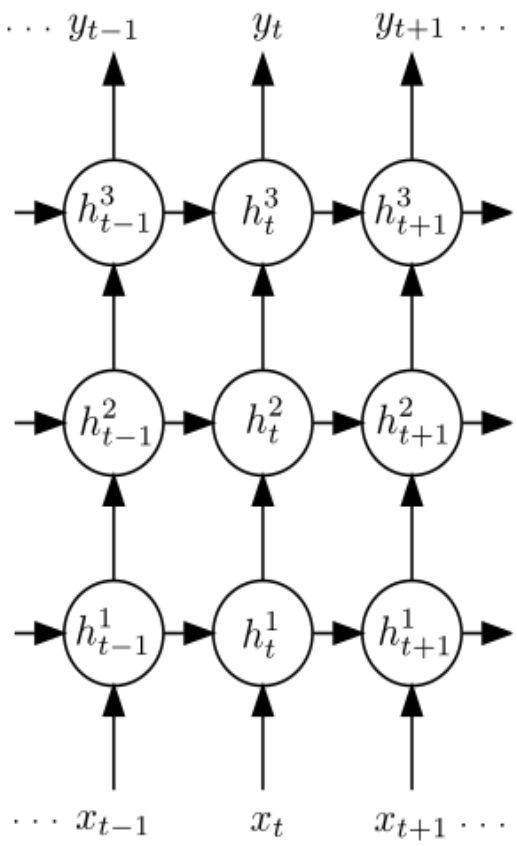


Hình 2.14: Minh họa mạng BiRNN (Nguồn: Coursera Sequence Models)

Tuy nhiên BiRNN có một hạn chế là không thể chạy liên tục, nó yêu cầu một điểm cuối cố định cho cả tương lai và quá khứ, vì BiRNN vẫn mang bản chất của một mạng RNN cơ bản. Trên thực tế LSTM và BiRNN khá tương thích với nhau, vì LSTM giới thiệu một đơn vị cơ bản mới để tạo thành một lớp ẩn, còn BiRNN liên quan đến việc kết nối giữa các lớp ẩn, bất kể chúng được cấu thành từ loại đơn vị nào. Sự kết hợp này, đưa đến khái niệm BiLSTM. Dựa vào thực nhiệm [4], cách tiếp cận này đã đạt được kết quả hiện đại về nhận dạng chữ viết tay và phân loại âm vị.

1.4.4. Mạng nơ-ron hồi quy học sâu (Deep RNN – Deep Reccurent Neural Network)

Một yếu tố quan trọng cho sự thành công gần đây của các hệ thống lai là sử dụng kiến trúc học sâu. Deep RNN có thể được tạo bằng cách xếp chồng nhiều lớp ẩn RNN lên nhau, với đầu ra của một lớp trở thành đầu vào của lớp tiếp theo, được biểu diễn như hình 2.15.



Hình 2.15: Mạng nơ-ron hồi quy học sâu (Nguồn: [4])

1.5. Các kỹ thuật trong dịch máy

1.5.1. Mô hình seq2seq (Encoder - Decoder)

1.5.2. Cơ chế Attention (Attention Mechanism)

2. Hệ thống dịch máy

TÀI LIỆU THAM KHẢO

[1] Hovy E.H.: Toward finely differentiated evaluation metrics for machine translation. Proceedings of the Eagles Workshop on Standards and Evaluation, Pisa, Italy, 1999.

[2] Yani Andrew Ioannou. "*Structural Priors in Deep Neural Networks*" September 2017.

[3] Zachary C. Lipton, John Berkowitz, Charles Elkan. "*A Critical Review of Recurrent Neural Networks for Sequence Learning*" June 5th, 2015, pp.10-11.

[4] Alex Graves, Navdeep Jaitly. "*Towards End-to-End Speech Recognition with Recurrent Neural Networks*", pp.3-4.

[4] Zachary C. Lipton, John Berkowitz, Charles Elkan. "*A Critical Review of Recurrent Neural Networks for Sequence Learning*" June 5th, 2015, pp.10-11.