RNN and Vanishing/Exploding gradient problem (Recurrent Neural Network)

*

Tóm tắt nội dung

Bài báo cáo này nhằm giới thiệu cho bạn đọc một cái nhìn tổng quan nhất về mặt lí thuyết dành cho phương pháp Recurrent Neural Network(RNN) và vấn đề về biến mất và bùng nổ đạo hàm. Bài báo cáo sẽ giới thiệu từ ý tưởng sơ khai đến những phân tích về cơ sở toán học của bài toán, về cách hoạt động, và phương pháp LSTM trong việc giải quyết khó khăn liên quan đến biến mất và bùng nổ đạo hàm.

Từ khóa liên quan: recurrent neural network, vanishing gradient, exploding gradient, LSTM, GRU.

Giới thiệu

Recurrent neural networks (RNNs) là tập hợp các phương pháp có mối quan hệ mật thiết với mạng nơ-ron trong sử lý dữ liệu có tuần tự. Về cơ bản, nếu muốn áp dụng deep learning mà thấy dữ liệu có dạng tuần tự hay dãy số liệu theo thời gian thì hãy nghĩ ngay đến RNN như chuyển giọng nói sang văn bản, máy dịch tự động, nhận diện trong video,...

*

Mục lục

1 Đặt vấn đề

1.1 Định nghĩa bài toán

Một mô hình ngôn ngữ gồm tập hữu hạn từ vựng $\boldsymbol{V}$, và một hàm xác suất $p(x_1, x_2, ..., x_n)$ sao cho:

$$\forall \in \boldsymbol{V}^+, p(x_1, x_2, ...x_n) \geq 0$$

và

$$\sum_{\in \boldsymbol{V}^+} p(x_1, x_2, ...x_n) = 1$$

Ta có thể định nghĩa xác suất trên bằng công thức đơn giản như sau:

$$p(x_1, x_2, ..., x_n) = \frac{\mathrm{c(x_1, x_2, ..., x_n)}}{\mathrm{N}}$$

1.2 Thang đo đánh giá

Một thách thức lớn của bài toán trên là cần một thang đo phù hợp để đánh giá và so sánh các phương pháp. Một trong những phương pháp phổ biến đó là perplexity(độ hỗn độn).

Giả sử ta có các câu để test (held-out: không nằm trong tập training)$x^{(1)}, x^{(2)}, ..., x^{(m)}$. Mỗi câu $x^{(i)}$ có $\boldsymbol{i} \in \{1, 2, ...m\}$ là chuỗi các từ $x_1^{(i)}, x_2^{(i)}, ...x_{n_i}^{(i)}$ trong đó n_i là độ dài thứ i.

Ta tính xác suất $p(x^{(i)})$ cho từng $x^{(i)}$ thông qua mô hình ngôn ngữ vừa training xong. Khi đó, chất lượng của mô hình ngôn ngữ này sẽ được tính như sau:

$$\prod_{i=1}^{m} p(x^{(i)})$$

Giá trị thu được từ phép tính trên càng cao thì chất lượng của mô hình càng tốt đối với dữ liệu chưa hề thấy trong tập training.

Perplexity được định nghĩa như sau:

$$2^{-l}$$

Trong đó:

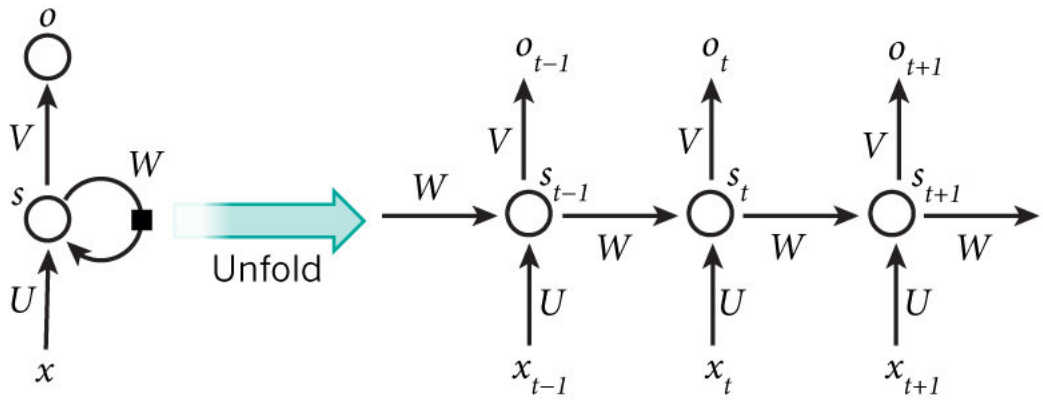
$$\begin{cases} M = \sum_{i=1}^{m} n_i \\ l = \frac{1}{\mathrm{M}} \sum_{i=1}^{m} \log_2 p(x^{(i)}) \end{cases}$$

Theo công thức trên nếu giá trị của Perplexity càng nhỏ, mô hình ngôn ngữ xây dựng được càng tốt.

2 Recurrent Neural Network

2.1 Ý tưởng

Ý tưởng chính của RNN (Recurrent Neural Network) là sử dụng chuỗi các thông tin. Khác so với các mạng nơ-ron truyền thống khi tất cả các đầu vào và đầu ra độc lập với nhau. Tức là chúng không liên kết với nhau thành chuỗi. RNN được đề xuất cho vấn đề này. RNN có khả năng thực hiện cùng một tác vụ cho tất cả các phần tử của một chuỗi với đầu ra phụ thuộc vào tất cả các phép tính trước đó.



Hình 1: A recurrent neural network and the unfolding in time of the computation involved in its forward computation.

Lúc đó việc tính toán của RNN sẽ như sau:

- x_t là đầu vào tại bước t.
- s_t là trạng thái ẩn tại bước t. Nó chính là ***bnh*** của mạng. s_t được tính toán dựa trên cả các trạng thái ẩn phía trước và đầu vào tại bước đó: $s_t = f(Ux_t + Ws_{t-1})$. Hàm f sẽ là một hàm phi tuyến như ***tanh*** hay ***ReLU***. Khởi tạo s_{-1} thường gán bằng 0.
- o_t là đầu ra tại bước t. o_t có thể là một vectơ xác suất. $o_t = softmax(Vs_t)$

2.2 Lượt truyền xuôi (Forward Pass)

Xét chuỗi đầu vào $\boldsymbol{x}$ có chiều dài $\boldsymbol{T}$ với RNN có:

- $\boldsymbol{I}$ là nút đầu vào.
- $\boldsymbol{H}$ là nút ẩn.
- $\boldsymbol{K}$ là nút đầu ra.

Giả sử, x_i^t là giá trị của đầu vào thứ i tại thời gian t. a_j^t và b_j^t lần lượt là mạng đầu vào nút thứ j tại thời điểm t và nút kích hoạt j tại thời điểm t.

Các nút ẩn được biểu diễn như sau:

$$a_h^t = \sum_{i=1}^{I} w_{ih} x_i^t + \sum_{h`=1}^{H} w_{h`h} b_{h`}^{t-1}$$

Hàm kích hoạt được biểu diễn như sau:

$$b_h^t = \theta_h(a_t^h)$$

Mạng đầu vào(network input) đến nút đầu ra(output unit) được tính toán đồng thời hàm kích hoạt ẩn:

$$a_k^t = \sum_{h=1}^{H} w_{hk} b_h^t$$

2.3 Backpropagation through time (BPTT)

Như những giải thuật lan truyền ngược chuẩn khác, BPTT cũng là áp dụng lặp đi lặp lại quy tắc dây chuyền. Tuy nhiên, đối với mạng nơ-ron hồi tiếp, hàm mất mát phụ thuộc vào việc kích hoạt lớp ẩn không chỉ thông qua ảnh hưởng của nó đối với lớp đầu ra, mà còn thông qua ảnh hưởng của nó đối với lớp ẩn kế tiếp. Vậy nên:

$$\delta_h^t = \theta`(a_h^t)(\sum_{k=1}^{K} \delta_h^t w_{hk} + \sum \delta_{h`}^{t+1} w_{hh`})$$

với

$$\delta_h^t \triangleq \frac{\partial \mathcal{L}}{\partial a_j^t}$$

Lưu ý: các trọng số w được sử dụng như nhau tại mỗi timestep, nên ta có:

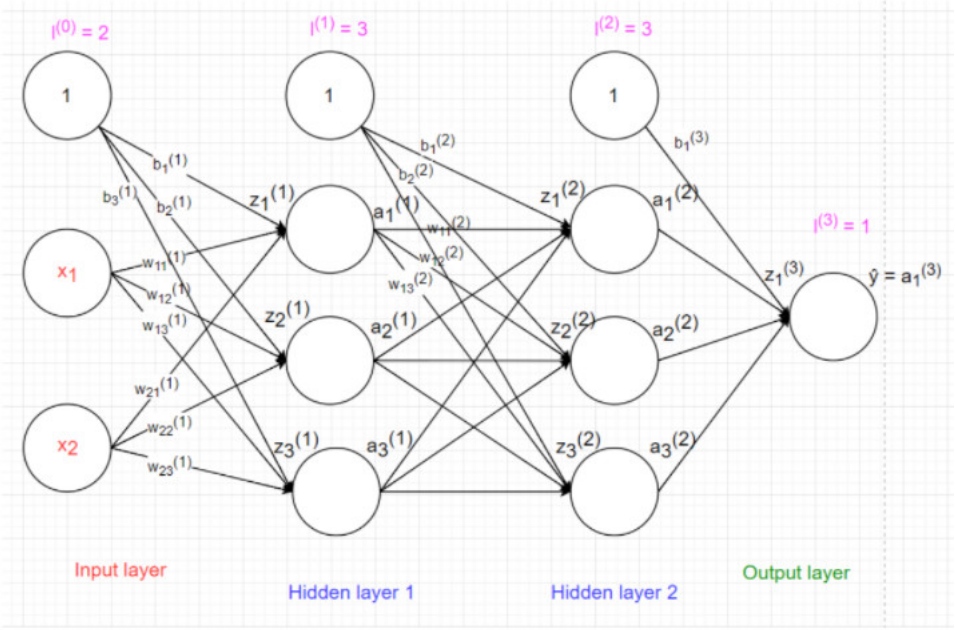
$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial a_j^t} \frac{\partial a_j^t}{\partial w_{ij}} = \sum_{t=1}^{T} \delta_j^t b_i^t$$

3 Thảo luận

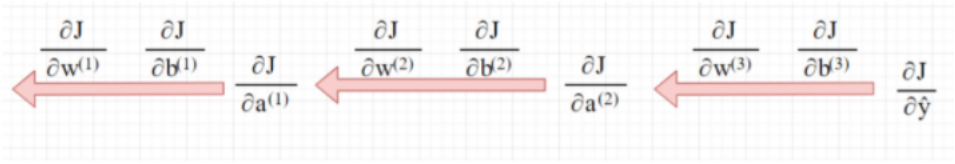
3.1 Vanishing/Exploding Gradient Problem

Backpropagation là thuật toán được dùng để tính đạo hàm các hệ số trong neural network với loss function đề rồi áp dụng gradient descent để tìm các hệ số.

Nhận xét:



Hình 2: Figure 1: A picture of the same gull looking the other way!



Hình 3: Quá trình backpropagation

- Nếu các hệ số W và D đều lớn hơn 1 thì khi tính gradient ở các layer đầu ta sẽ phải nhân tích của rất nhiều số lớn hơn 1 nên giá trị sẽ tiến dần về vô cùng và bước cập nhật hệ số trong gradient descent trở nên không chính xác và các hệ số neural network sẽ không học được nữa.–> Exploding gradient
- Nếu các hệ số W và D đều nhỏ hơn 1 thì khi tính gradient ở các layer đầu ta sẽ phải nhân tích của rất nhiều số nhỏ hơn 1 nên giá trị sẽ tiến dần về 0 và bước cập nhật hệ số trong gradient descent trở nên vô nghĩa và các hệ số neural network sẽ không học được nữa.–> Vanishing gradient

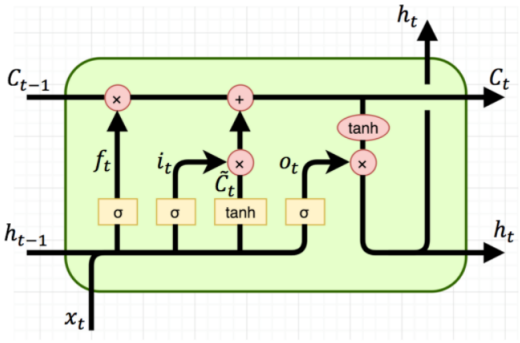
3.2 LSTM

3.2.1 Mô hình LSTM:

Ở state thứ t của mô hình LSTM:

- Output: c_t, h_t ta gọi c là cell state, h là hidden state.

- Input: c_{t-1}, h_{t-1}, x_t, trong đó x_t là input ở state thứ t của model. c_{t-1}, h_{t-1} là output của layer trước. h đóng vai trò như s ở RNN, trong khi c là điểm mới của LSTM.



Hình 4: Mô hình LSTM

Cách đọc biểu đồ trên: bạn nhìn thấy kí hiệu σ, tanh ý là bước đấy dùng sigma, tanh activation function. Phép nhân ở đây là element-wise multiplication, phép cộng là cộng ma trận.

f_t, i_t, o_t tương ứng với forget gate, input gate và output gate.

- Forget gate: $f_t = \sigma(U_f * x_t + W_f * h_{t-1} + b_f)$
- Input gate: $i_t = \sigma(U_i * x_t + W_i * h_{t-1} + b_i)$
- Output gate: $o_t = \sigma(U_o * x_t + W_o * h_{t-1} + b_o)$

Nhận xét: $0 < f_t, i_t, o_t < 1$; b_f, b_i, b_o là các hệ số bias; hệ số W, U giống như trong bài RNN.

$\tilde{c_t} = \tanh(U_c * x_t + W_c * h_{t-1} + b_c)$, bước này giống hệt như tính s_t trong RNN.

$c_t = f_t * c_{t-1} + i_t * \tilde{c_t}$, ***forget gate*** sẽ quyết định xem cần lấy bao nhiêu từ cell state trước và ***input gate*** sẽ quyết định lấy bao nhiêu từ input của state và hidden layer của layer trước.

$h_t = o_t * \tanh c_t$, ***output gate*** sẽ quyết định xem cần lấy bao nhiêu từ cell state để trở thành output của hidden state. Ngoài ra, h_t cũng được dùng để tính ra output y_t cho state t.

Nhận xét: $h_t, \tilde{c}$ khá giống với khá giống với RNN, nên model có short term memory. Trong khi đó ct giống như một băng chuyền ở trên mô hình RNN vậy, thông tin nào cần quan trọng và dùng ở sau sẽ được gửi vào và dùng khi cần –> có thể mang thông tin từ đi xa –> long term memory. Do đó mô hình LSTM có cả short term memory và long term memory.

3.2.2 LSTM giảm vanishing gradient:

Ta cũng áp dụng thuật toán back propagation through time cho LSTM tương tự như RNN.

Thành phần chính gây nên vanishing gradient trong RNN là $\frac{\partial s_{t+1}}{\partial s_t} = (1 - s_t^2)^2 * W$, trong đó, s_t, $W < 1$.

Tương tự, trong LSTM ta quan tâm đến $\frac{\partial c_t}{\partial c_{t-1}} = f_t$. Do $0 < f_t < 1$ nên về cơ bản thì LSTM vẫn bị vanishing gradient nhưng bị ít hơn so với RNN. Hơn thế nữa, khi mang thông tin trên cell state thì ít khi cần phải quên giá trị cell cũ, nên $f_t \approx 1 \Rightarrow$ giảm được vanishing gradient.

4 Lời kết

Mặc dù LSTM tỏ ra hiệu quả trong việc giải quyết vấn đề vanishing/exploding gradient, song vẫn có một vài hàm số với tính chất đặc biệt đã tạo khó khăn trong quá trình tìm nghiệm. Thông qua những phân tích bên trên, chúng ta có thể rút ra được rằng hai yếu tố ảnh hưởng lớn nhất đến chất lượng lời giải cho các bài toán liên quan đến RNN là một hàm mục tiêu hợp lí và một kiến trúc mạng phù hợp với tác vụ cần giải quyết.

5 Tài liệu tham khảo

- Perplexity—a measure of the difficulty of speech recognition tasks, F. Jelinek, R. L. Mercer, L. R. Bahl, et al.
- Deep learning: RNNs and LSTM, Robert DiPietro, Gregory D. Hager
- Handbook of Medical Image Computing and Computer Assisted Intervention(2020), Robert DiPietro, Gregory D. Hager
- Learning representations by back-propagating errors, David E. Rumelhart, Geoffrey, E.Hintont, Ronald J. Williams
- Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network, Alex Sherstinsky