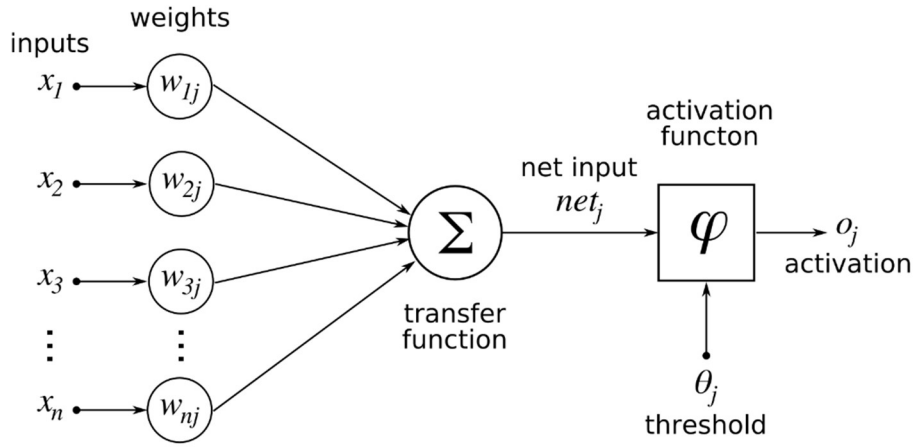
Mô Hình Mạng Hồi Quy (RNN) trong chuỗi thời gian

Nguyễn Minh Nhựt - 220104018

Yêu cầu 1: Thuật toán tìm phương trình dự báo theo chuỗi thời gian và đánh giá độ phức tạp thuật toán này.

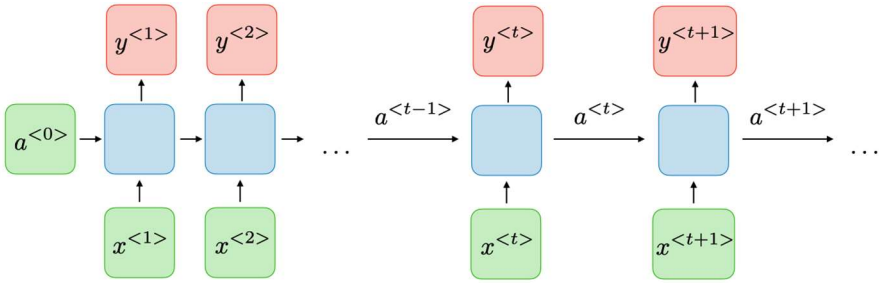
- Mạng hồi truy hồi (Recurent Neural Network) là một họ của các mạng neural dùng để chuyên xử lý dữ liệu ***dạng chuỗi***.



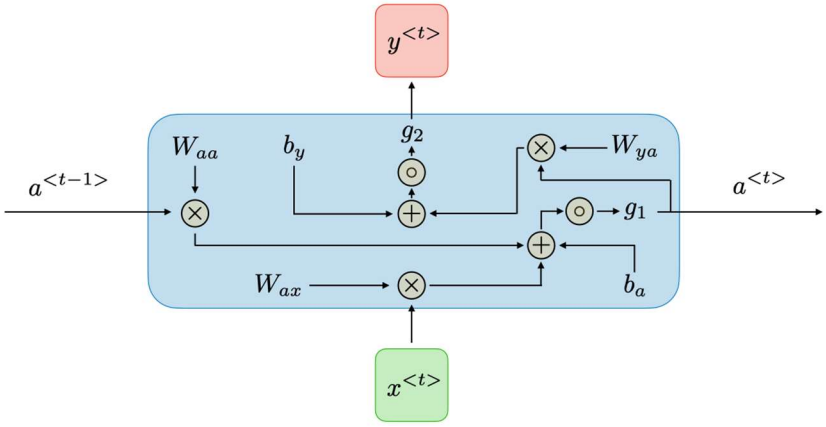
Hình 1. 1 Mạng Neural bình thường

Khác với mạng Neural ***không học dữ liệu*** tại thời điểm (timestep) quá khứ.

- Gọi ***t*** là thời điểm xét, ***t – 1*** là thời điểm quá khứ 1 đơn vị của t



- $a^{<t>}$ là trạng thái *thời điểm t*
- $y^{<t>}$ là giá trị output *thời điểm t*
- $x^{<t>}$ là giá trị input *thời điểm t*



- Tại thời điểm t tính lan truyền xuôi forward:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

$$\tilde{y}_t = y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

  - Với W_{aa}, W_{ax}, W_{ya} là các trọng số thường là random
  - g_1, g_2 là các hàm kích hoạt thường làm ***tanh***
  - b_a và b_y là các bias
  - $y^{<t>} = g_2(W_{ya} * g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) + b_y)$
- Loss Function = $\frac{1}{N}\sum_{t=1}^{N}(y_t - \tilde{y}_t)^2$
- Đặt W = [W_{aa}, W_{ax}, W_{ya}], r = $(y_t - \tilde{y}_t)^2$, B = [b_a, b_y]
- Đạo hàm Loss function theo W: $\frac{\partial \mathcal{L}}{\partial W} = \frac{1}{N}\sum_{t=1}^{N}\frac{\partial r}{\partial W} = \frac{1}{N}\sum_{t=1}^{N}\frac{\partial r}{\partial \tilde{y}_t}\frac{\partial \tilde{y}_t}{\partial a^{<t>}}\frac{\partial a^{<t>}}{\partial W}$
- Ví dụ: $\frac{\partial L_3}{\partial W} = \frac{\partial L_3}{\partial a^{<3>}}\frac{\partial a^{<3>}}{\partial W} + \frac{\partial L_3}{\partial a^{<3>}}\frac{\partial a^{<3>}}{\partial a^{<2>}}\frac{\partial a^{<2>}}{\partial W} + \frac{\partial L_3}{\partial a^{<3>}}\frac{\partial a^{<3>}}{\partial a^{<2>}}\frac{\partial a^{<2>}}{\partial a^{<1>}}\frac{\partial a^{<1>}}{\partial W}$
- Từ đây có công thức tổng quát của

$$\frac{\partial L}{\partial W} = -\frac{1}{N}\sum_{t=1}^{N}\sum_{k=1}^{t}\frac{\partial L_t}{\partial a^{<t>}}\left(\prod_{j=k+1}^{t}\frac{\partial a^{<j>}}{\partial a^{<j-1>}}\right)\frac{\partial a^{<k>}}{\partial W}$$

  - $\frac{\partial \mathcal{L}}{\partial W_{ya}} = \sum_{t=1}^{T}\frac{\partial \mathcal{L}}{\partial \widehat{y_t}}\frac{\partial \widehat{y_t}}{\partial W_{ya}}$
  - $\frac{\partial \mathcal{L}}{\partial W_{aa}} = \sum_{t=1}^{T}\frac{\partial \mathcal{L}}{\partial \widehat{y_t}}\frac{\partial \widehat{y_t}}{\partial a^{<t>}}\frac{\partial a^{<t>}}{\partial W_{aa}}$
  - $\frac{\partial \mathcal{L}}{\partial W_{xa}} = \sum_{t=1}^{T}\frac{\partial \mathcal{L}}{\partial \widehat{y_t}}\frac{\partial \widehat{y_t}}{\partial a^{<t>}}\frac{\partial a^{<t>}}{\partial W_{xa}}$
- Tương tự đạo hàm Loss function theo B:
  - $\frac{\partial \mathcal{L}}{\partial b_y} = \sum_{t=1}^{T}\frac{\partial \mathcal{L}}{\partial \widehat{y_t}}\frac{\partial \widehat{y_t}}{\partial b_y}$

- $\frac{\partial \mathcal{L}}{\partial b_h} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \widehat{y_t}} \frac{\partial \widehat{y_t}}{\partial h_t} \frac{\partial h_t}{\partial b_h}$

- Cập nhật trọng số với tỉ lệ học η
  - $W_{ay}^{(t+1)} = W_{ay}^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial W_{ay}}$
  - $W_{aa}^{(t+1)} = W_{aa}^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial W_{aa}}$
  - $W_{xa}^{(t+1)} = W_{xa}^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial W_{xa}}$
  - $b_y^{(t+1)} = b_y^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial b_y}$
  - $b_h^{(t+1)} = b_h^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial b_h}$

- Thuật toán RNN trong chuỗi thời gian

Algorithm 1: Thuật toán Backpropagation trong mạng nơ-ron hồi quy (RNN)

Đầu vào: Dữ liệu đầu vào chuỗi thời gian $x_1, x_2, ..., x_T$, nhãn thực tế tương ứng $y_1, y_2, ..., y_T$, số lượng mẫu trong chuỗi T, tỷ lệ học η

Đầu ra : Cập nhật các tham số W_{ay}, b_y, W_{aa}, W_{xa}, b_h

Khởi tạo tham số: Khởi tạo W_{ay}, b_y, W_{aa}, W_{xa}, b_h;

for $t \leftarrow 1$ **to** T **do**

  Lần truyền tiến (Forward pass):;

  $a_t = \sigma(W_{aa} a_{t-1} + W_{xa} x_t + b_h)$;

  $y_t = \text{softmax}(W_{ay} a_t + b_y)$;

Tính toán hàm mất mát:;

$\mathcal{L} = -\sum_{t=1}^{T} \sum_i y_{t,i} \log(\hat{y}_{t,i})$;

Lần truyền ngược (Backward pass):;

for $t \leftarrow T$ **to** 1 **do**

  Tính gradient của loss theo đầu ra y_t:;

  $dL\_dy = \text{gradient\_cross\_entropy\_loss}(y[t], y\_true[t])$;

  Tính gradient của loss theo các tham số:;

  $dW_{ay} += dL\_dy \times a_t$;

  $db_y += dL\_dy$;

  $dh = (dL\_dy \times W_{ay}) + dh\_next$;

  $dh\_raw = dh \times \text{activation\_derivative}(h[t])$;

  $dW_{aa} += dh\_raw \times a_{t-1}$;

  $dW_{xa} += dh\_raw \times x_t$;

  $db_h += dh\_raw$;

  $dh\_next = dh\_raw \times W_{aa}$;

Cập nhật tham số:;

$W_{ay} -= \eta \times dW_{ay}$;

$b_y -= \eta \times db_y$;

$W_{aa} -= \eta \times dW_{aa}$;

$W_{xa} -= \eta \times dW_{xa}$;

$b_h -= \eta \times db_h$;

Lặp lại quá trình trên cho một số lần hoặc cho đến khi hàm mất mát giảm đủ;

- **Độ phức tạp thuật toán [7]**
    - Một mạng nơ-ron hồi quy (RNN) có độ phức tạp thời gian là xấp xỉ O(N), trong đó n là độ dài của chuỗi, nhưng vì phương pháp này là tự hồi quy, nó phụ thuộc vào đầu ra của bước trước đó, điều này làm cho thuật toán không thể được phân tán song song.
    - **Lan truyền xuôi**: Trong quá trình lan truyền tiến, mỗi đơn vị thời gian T trong chuỗi đầu vào được xử lý một cách tuần tự. Đối với mỗi bước thời gian t, chúng ta thực hiện các phép tính tuyến tính và phi tuyến. Số lượng phép tính này không phụ thuộc vào độ dài của chuỗi đầu vào, do đó, độ phức tạp của lan truyền tiến là O(1) cho mỗi bước thời gian.
    - **Lan truyền ngược:** Trong quá trình lan truyền ngược, chúng ta tính toán gradient của hàm mất mát theo các tham số của mạng. Tính toán này cũng diễn ra tuần tự theo các bước thời gian và không phụ thuộc vào độ dài của chuỗi đầu vào. Do đó, độ phức tạp của lan truyền ngược cũng là O(1) cho mỗi bước thời gian.
    - Tổng cộng, vì cả lan truyền tiến và lan truyền ngược đều có độ phức tạp là O(1) cho mỗi bước thời gian, và vì mỗi bước thời gian đều phụ thuộc vào độ dài của N chuỗi đầu vào, nên độ phức tạp toàn bộ của mạng nơ-ron hồi quy là O(N)

TÀI LIỆU THAM KHẢO

[1] Géron, A. "Hands-On machine learning with scikit-learn, keras & tensorflow farnham." Canada: O'Reilly (2019).

[2] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[3] https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

[4] https://mmuratarat.github.io/2019-02-07/bptt-of-rnn

[5] https://courses.cs.washington.edu/courses/csep517/20wi/slides/csep517wi20-RNNs.pdf

[6] Lioutas, Vasileios. Sequence Modeling with Linear Complexity. Diss. Carleton University, 2020.