Recurrent Neural Networks (RNNs), and the Exploding and Vanishing Gradient Problems

Nguyen Vo Ngoc Bao 23520131
Chu Duong Huy Phuoc 23521229

University of Information Technology

Ngày 13 tháng 1 năm 2025

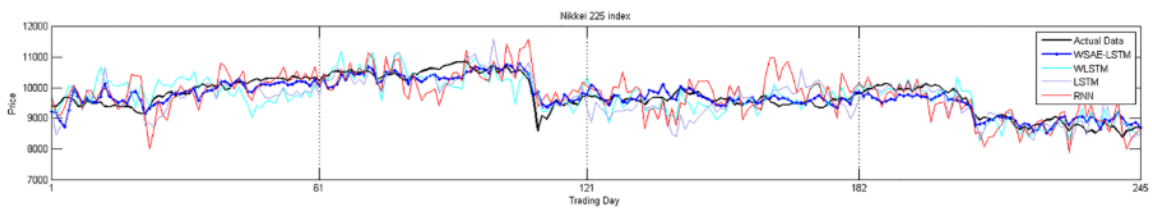
UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Table of Contents

Table of Contents

Sequential Data

- **Dữ liệu tuần tự (Sequential data)**, hay thường được gọi là *sequence data* hoặc *sequences*.
- Điểm đặc biệt của nó nằm ở việc: các phần tử trong một dãy xuất hiện theo **thứ tự nhất định** và không độc lập với nhau.
- **Ví dụ: Dự đoán giá cổ phiếu**
  Trong ví dụ này, dữ liệu đầu vào là các giá cổ phiếu trong quá khứ. i_1 là giá cổ phiếu vào ngày 1/1, i_2 là giá cổ phiếu vào ngày 2/1, v.v... Chuỗi $(i_1, i_2, \ldots, i_{30})$ được gọi là *sequences*. Mô hình RNN sẽ học từ dữ liệu đầu vào và dự đoán giá cổ phiếu trong tương lai.



Hình 1.1: *Ví dụ về dự đoán giá cổ phiếu*

Sequential data vs. time series data

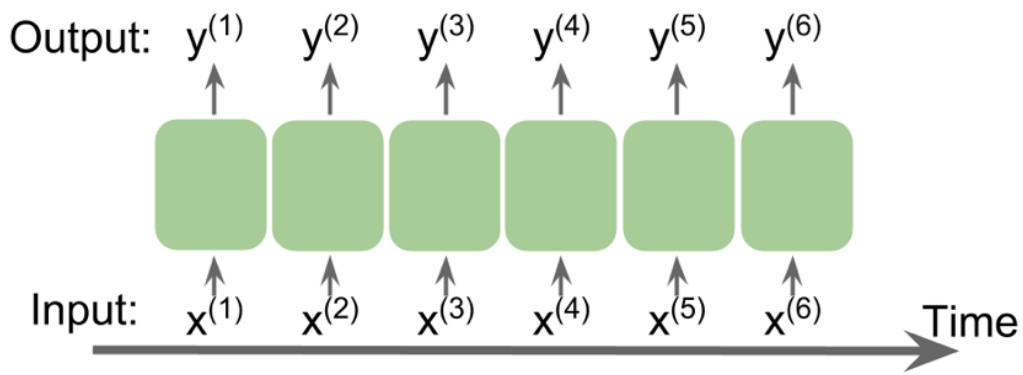
- **Dữ liệu chuỗi thời gian (time series data)** là một loại đặc biệt của dữ liệu tuần tự, trong đó mỗi ví dụ được gắn với một chiều liên quan đến thời gian.
- Trong dữ liệu chuỗi thời gian, các mẫu được ghi nhận ở những mốc thời gian kế tiếp nhau; do đó, chiều thời gian quyết định thứ tự giữa các điểm dữ liệu. Ví dụ, giá cổ phiếu hay các bản ghi giọng nói đều là dữ liệu chuỗi thời gian.
- **Ngược lại**, không phải tất cả dữ liệu tuần tự đều có cùng cấu trúc thời gian này. Chẳng hạn, dữ liệu văn bản hoặc chuỗi DNA cũng có tính tuần tự, nhưng lại thiếu một chiều thời gian rõ ràng.

Sequential data vs. time series data

- **Ví dụ về dịch máy (machine translation)**: Đầu vào là một câu, chẳng hạn: "Tôi yêu UIT". Thứ tự từ trong câu có ảnh hưởng đáng kể đến ý nghĩa. Dữ liệu đầu vào gồm chuỗi các từ ['Tôi', 'yêu', 'UIT'], được gọi là *sequences*.
- Trong **xử lý ngôn ngữ tự nhiên (NLP)**, việc xử lý cả câu đầy đủ làm đầu vào thường không khả thi. Tương tự như cách xử lý video bằng cách chia nhỏ thành từng khung hình, trong NLP, câu được tách thành các từ riêng lẻ để phục vụ xử lý.

Representing sequences

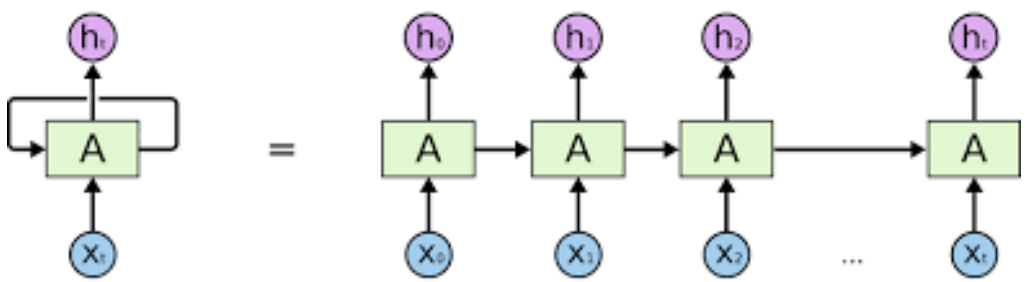
- Bộ phim mà bạn tôi **chưa** xem thì hay.
- Bộ phim mà bạn tôi đã xem thì **không** hay.



Hình 1.2: *Ví dụ về dữ liệu chuỗi thời gian*

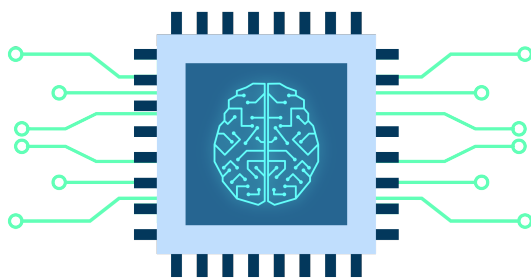
What is Recurrent Neural Networks (RNNs)?

- **Mạng nơ-ron hồi quy (RNNs)** là một loại kiến trúc mạng nơ-ron được dùng để phát hiện các mẫu (pattern) trong *các dãy dữ liệu*.
- Chúng cũng có thể được mở rộng để áp dụng cho ảnh, bằng cách chia ảnh thành từng mảng (patch) nối tiếp nhau và xem đó như một chuỗi (sequence).
- **Điểm khác biệt** giữa RNN và mạng nơ-ron truyền thẳng (feedforward neural networks), còn được gọi là **Multi-Layer Perceptrons (MLPs)**, nằm ở *cách thông tin được truyền qua mạng*.



What is Recurrent Neural Networks (RNNs)?

- **Multilayer Perceptrons (MLPs)** và **CNNs** dành cho dữ liệu ảnh thường giả định rằng các mẫu huấn luyện **độc lập** với nhau, do đó không đưa vào yếu tố **thứ tự** (ordering information).
- Chúng ta có thể nói rằng các mô hình này **không có bộ nhớ** để lưu giữ các ví dụ huấn luyện đã thấy trước đó.
- **Ngược lại,** RNNs được thiết kế riêng cho việc mô phỏng (modeling) các chuỗi (sequences) và có khả năng **ghi nhớ** thông tin trong quá khứ, đồng thời xử lý các sự kiện mới dựa trên thông tin đã được lưu. Đây là một lợi thế rõ ràng khi làm việc với dữ liệu có tính tuần tự.



Pros & Cons

Ưu điểm (+)

- Khả năng xử lý đầu vào với **độ dài bất kỳ**;
- Kích thước mô hình **không** tăng theo kích thước đầu vào;
- Quá trình tính toán **xét đến** thông tin trong quá khứ;
- Các trọng số (weights) được **chia sẻ theo thời gian**.

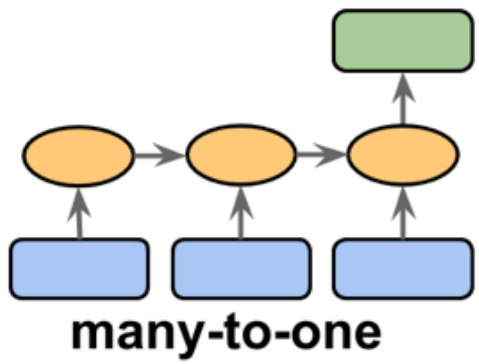
Nhược điểm (-)

- Tốc độ tính toán **chậm**;
- Khó truy cập thông tin từ **thời điểm rất lâu** trước đó;
- **Không** thể xét đến dữ liệu tương lai cho trạng thái hiện tại.

The different categories of sequence modeling

Many-to-One RNN

- **Many-to-One** được dùng khi ta cần một *đầu ra duy nhất* từ nhiều đơn vị đầu vào hoặc từ một chuỗi các đầu vào. Mô hình sẽ nhận một chuỗi dữ liệu đầu vào để tạo ra *một đầu ra cố định*.
- **Phân tích cảm xúc (Sentiment Analysis)** là một ví dụ phổ biến cho loại RNN này.

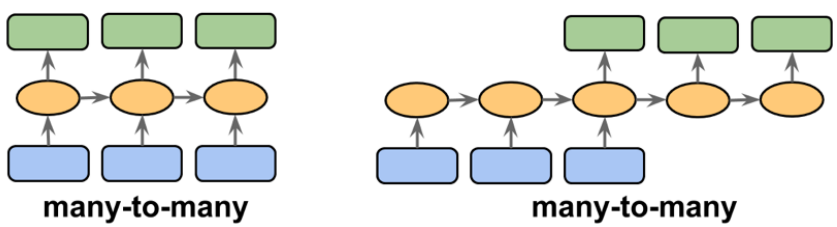


Hình 1.3: *Chế độ Many-to-One*

The different categories of sequence modeling

Many-to-Many RNN

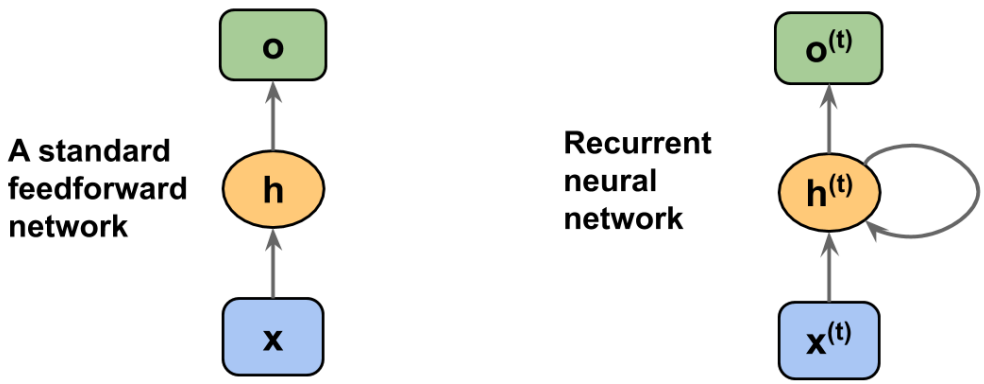
- **Many-to-Many** được dùng để tạo ra một chuỗi dữ liệu đầu ra từ một chuỗi các đơn vị đầu vào.
- Loại RNN này tiếp tục được chia thành hai phân nhóm chính:
  1. **Kích thước đơn vị bằng nhau (Equal Unit Size)**: Ở đây, số lượng đơn vị đầu vào và đầu ra **bằng nhau**. Một ứng dụng phổ biến là *Nhận dạng Thực thể Tên* (Name-Entity Recognition).
  2. **Kích thước đơn vị không bằng nhau (Unequal Unit Size)**: Ở đây, số lượng đơn vị đầu vào và đầu ra **khác nhau**. Ứng dụng có thể thấy trong *Dịch máy* (Machine Translation).



Hình 1.4: *Chế độ Many-to-Many*

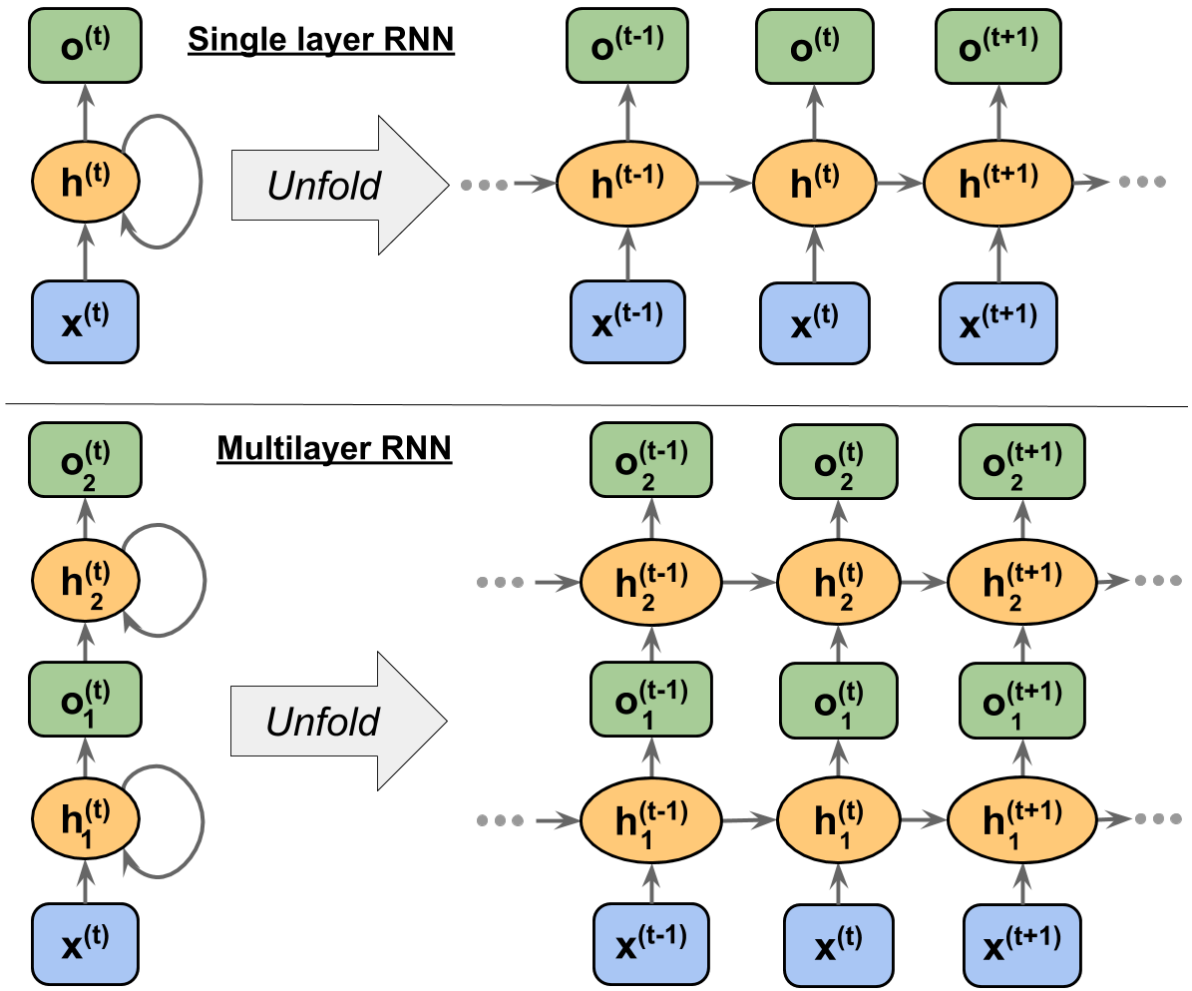
Understanding the dataflow in RNNs

- Trong một mạng nơ-ron truyền thẳng (feedforward network) tiêu chuẩn, thông tin được chuyển từ lớp đầu vào (input) sang lớp ẩn (hidden), rồi từ lớp ẩn sang lớp đầu ra (output).
- Ở chiều ngược lại, trong 1 RNN, lớp ẩn nhận đầu vào vừa từ lớp đầu vào ở thời điểm hiện tại (current time step), vừa từ chính lớp ẩn của thời điểm trước đó.



Hình 1.5: *FFNN và RNN*

Understanding the dataflow in RNNs



Hình 1.6: *Ví dụ về RNN một lớp ẩn (single layer) và nhiều lớp ẩn (multilayer)*

Table of Contents

Weight matrices in a single-hidden layer RNN

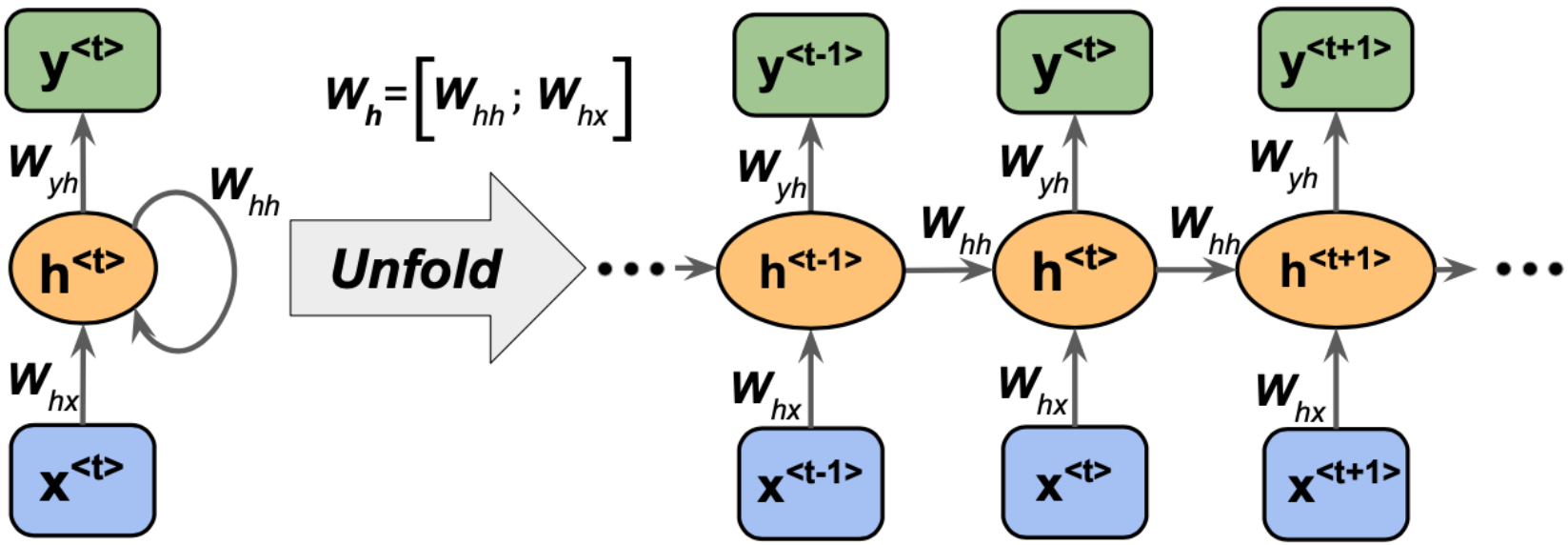


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition*. Birmingham, UK: Packt Publishing, 2019

Weight matrices in a single-hidden layer RNN

- Hình bên cho thấy cách mở (unfold) RNN qua các bước thời gian.
- Trạng thái ẩn $h^{(t)}$ phụ thuộc vào đầu vào hiện tại $x^{(t)}$ và trạng thái ẩn trước đó $h^{(t-1)}$.

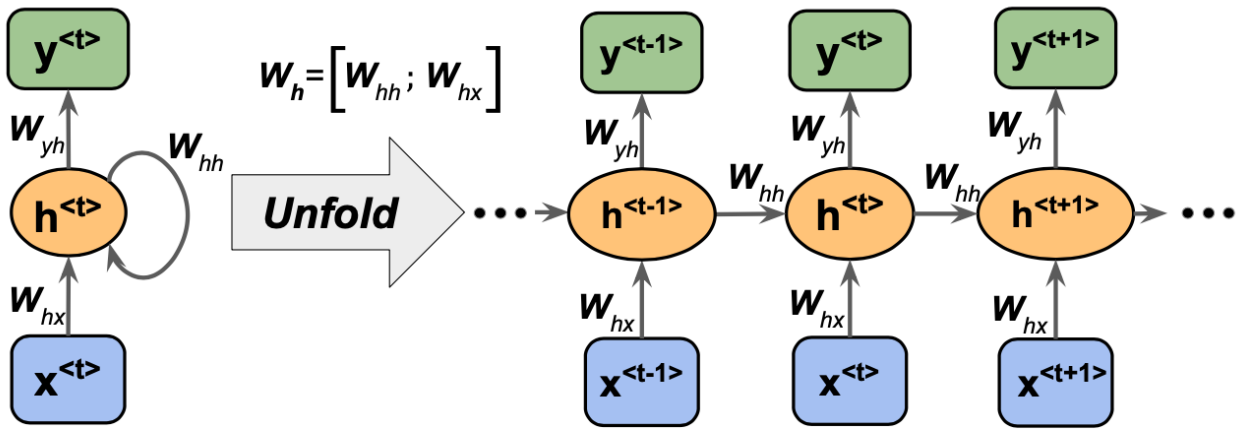


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

$$\text{Net input: } z_h^{(t)} = W_{hx}\, x^{(t)} + W_{hh}\, h^{(t-1)} + b_h$$

$$\text{Activation: } \mathrm{h}^{(t)} = \phi_h(z_h^{(t)}).$$

Weight matrices in a single-hidden layer RNN

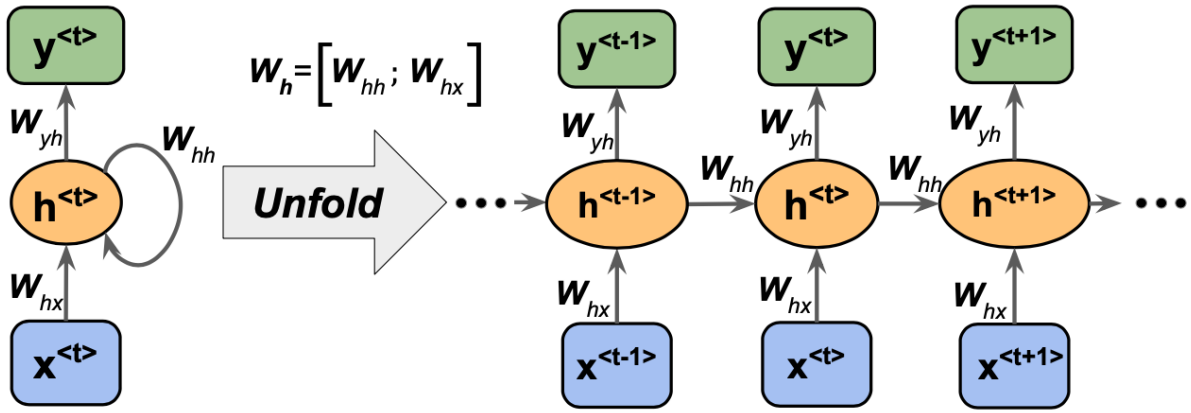


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning, 3rd Edition*. Birmingham, UK: Packt Publishing, 2019

Net input: $z_h^{(t)} = W_{hx}\, x^{(t)} \; + \; W_{hh}\, h^{(t-1)} \; + \; b_h$

Activation: $\mathrm{h}^{(t)} = \phi_h(z_h^{(t)})$.

Net input: $z_y^{(t)} = W_{yh} h^{(t)} \; + \; b_y$

Output: $\mathrm{y}^{(t)} = \phi_y(z_y^{(t)})$.

Gradient-based Learning Methods

- **Gradient Descent (GD)** điều chỉnh các trọng số của mô hình bằng cách tìm *đạo hàm của hàm lỗi* (error function derivatives) theo từng phần tử trong ma trận trọng số.

Batch Gradient Descent

Tính gradient cho toàn bộ tập dữ liệu trong mỗi vòng lặp tối ưu, sau đó thực hiện *một* bước cập nhật duy nhất:

$$\theta_{t+1} = \theta_t - \frac{\lambda}{U} \sum_{k=1}^{U} \frac{\partial \ell_k}{\partial \theta}$$

- Tuy nhiên, việc tính đạo hàm qua thời gian (error-derivatives through time) là rất **khó**. Lý do chủ yếu nằm ở mối quan hệ giữa các tham số và động lực (dynamics) của RNN, dẫn đến tình trạng bất ổn cao và làm cho GD kém hiệu quả.

Back-propagation Through Time (BPTT)

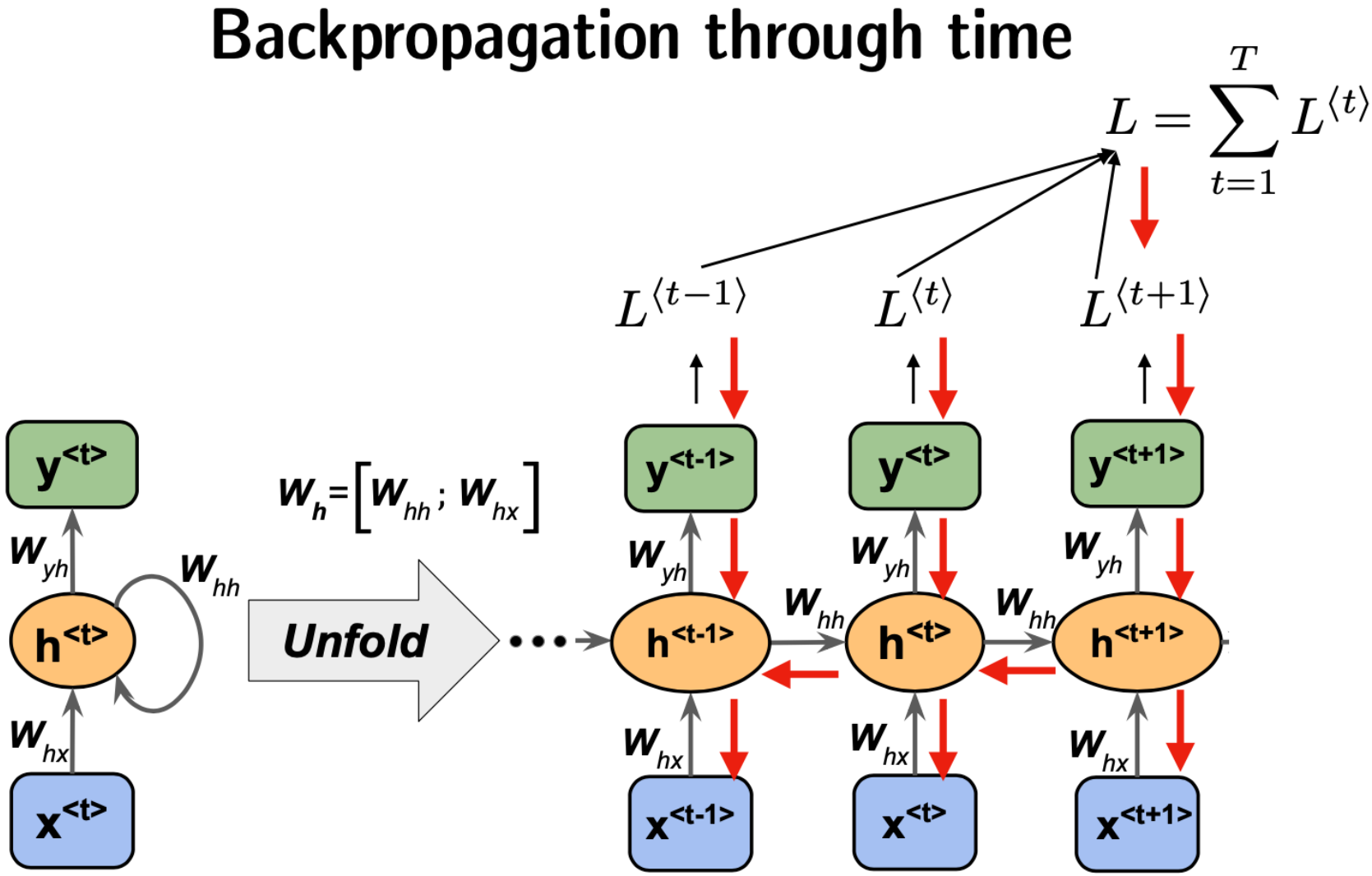


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition*. Birmingham, UK: Packt Publishing, 2019

Back-propagation Through Time (BPTT)

Biến ẩn (Hidden variable)

$$h^{(t)} = \phi_h(x_t \cdot W_{xh} + h^{(t-1)} \cdot W_{hh} + b_h)$$

Biến đầu ra (Output variable)

$$y^{(t)} = \phi_y(h^{(t)} \cdot W_{yh} + b_y)$$

- Ta định nghĩa một **hàm mất mát (loss function)** $L(y, o)$ để mô tả sự khác biệt giữa tất cả các đầu ra y_t và giá trị mục tiêu o_t, như trong công thức dưới đây.

Hàm mất mát (Loss Function)

$$L(y, o) = \sum_{t=1}^{T} \ell^{(t)}(y^{(t)}, o^{(t)})$$

Back-propagation Through Time (BPTT)

- Giả sử ta có **ba** ma trận trọng số: W_{xh}, W_{hh}, W_{yh}. Để huấn luyện RNN, ta cần lấy **đạo hàm riêng phần** (partial derivative) theo từng ma trận, sử dụng quy tắc dây chuyền (chain rule).

Đạo hàm riêng phần theo W_{yh}

$$\frac{\partial L}{\partial W_{yh}} = \sum_{t=1}^{T} \frac{\partial \ell^{(t)}}{\partial y^{(t)}} \frac{\partial y^{(t)}}{\partial z_y^{(t)}} \frac{\partial z_y^{(t)}}{\partial W_{yh}} = \sum_{t=1}^{T} \frac{\partial \ell^{(t)}}{\partial y^{(t)}} \frac{\partial y^{(t)}}{\partial z_y^{(t)}} h^{(t)}$$

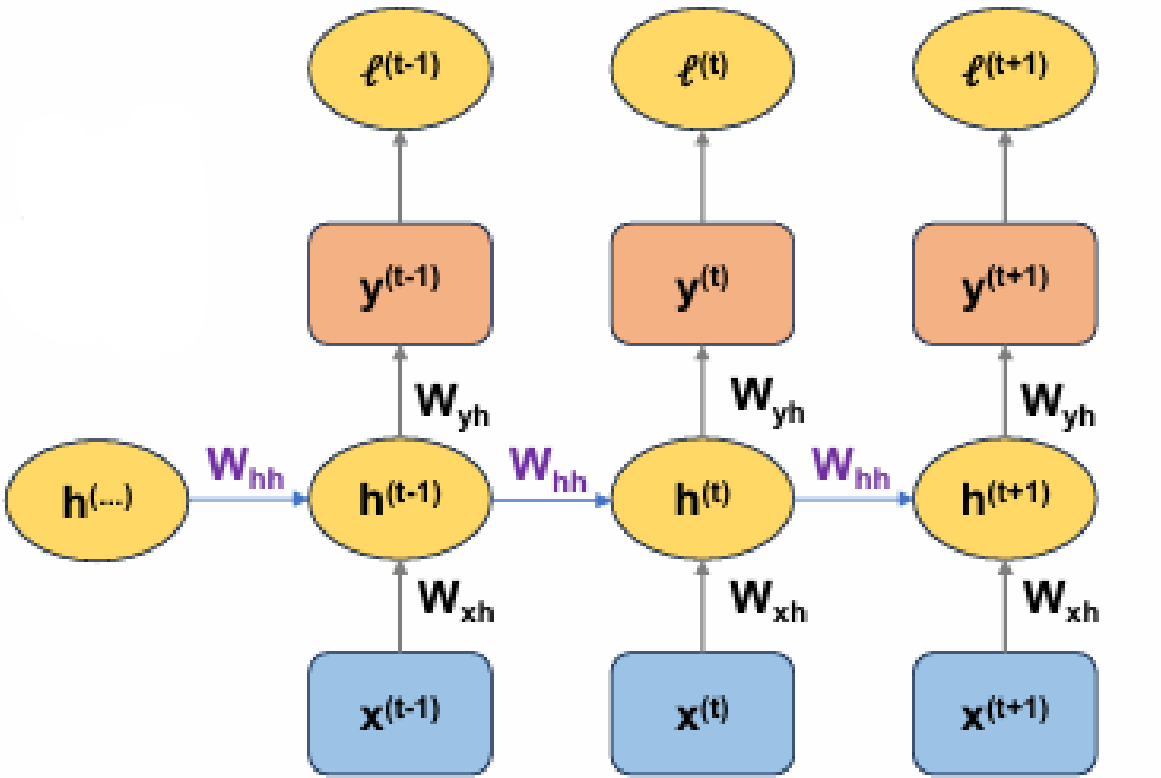
Đạo hàm riêng phần theo W_{hh}

$$\frac{\partial L}{\partial W_{hh}} = \sum_{t=1}^{T} \frac{\partial \ell^{(t)}}{\partial y^{(t)}} \frac{\partial y^{(t)}}{\partial z_y^{(t)}} \frac{\partial z_y^{(t)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial z_h^{(t)}} \frac{\partial z_h^{(t)}}{\partial W_{hh}}$$

$$= \sum_{t=1}^{T} \frac{\partial \ell^{(t)}}{\partial y^{(t)}} \frac{\partial y^{(t)}}{\partial z_y^{(t)}} W_{yh} \frac{\partial h^{(t)}}{\partial z_h^{(t)}} \frac{\partial z_h^{(t)}}{\partial W_{hh}}$$

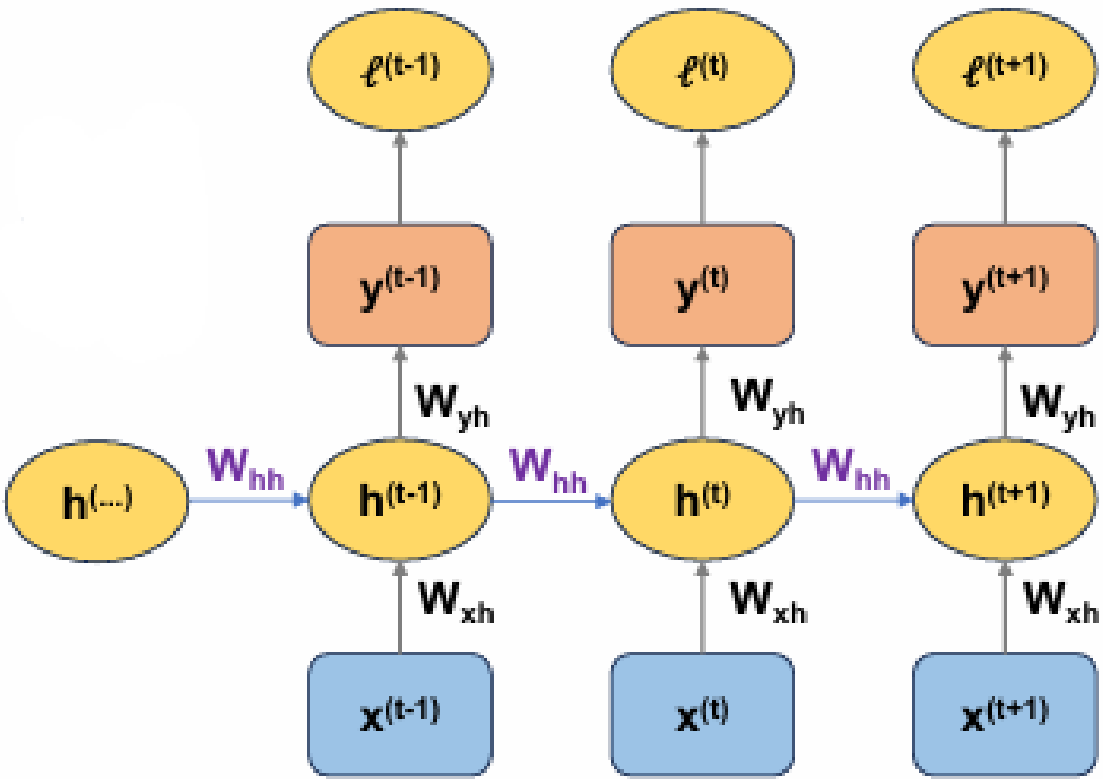
Back-propagation Through Time (BPTT)

$$\frac{\partial \ell^{(t)}}{\partial W_{hh}} = \frac{\partial \ell^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial z_y^{(t)}} \cdot \frac{\partial z_y^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial z_h^{(t)}} \cdot \frac{\partial z_h^{(t)}}{\partial W_{hh}}$$



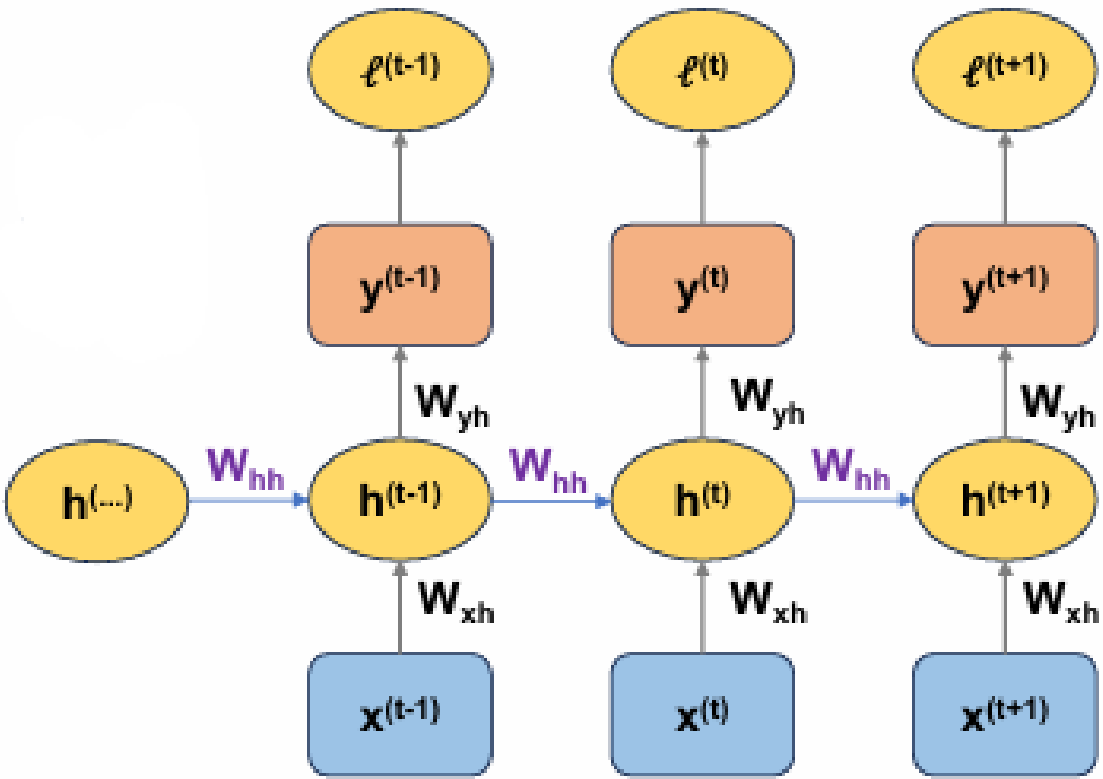
Back-propagation Through Time (BPTT)

$$\frac{\partial \ell^{(t)}}{\partial W_{hh}} = \frac{\partial \ell^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial z_y^{(t)}} \cdot \frac{\partial z_y^{(t)}}{\partial h^{(t)}} \cdot \boxed{\frac{\partial h^{(t)}}{\partial z_h^{(t)}} \cdot \frac{\partial z_h^{(t)}}{\partial W_{hh}}}$$

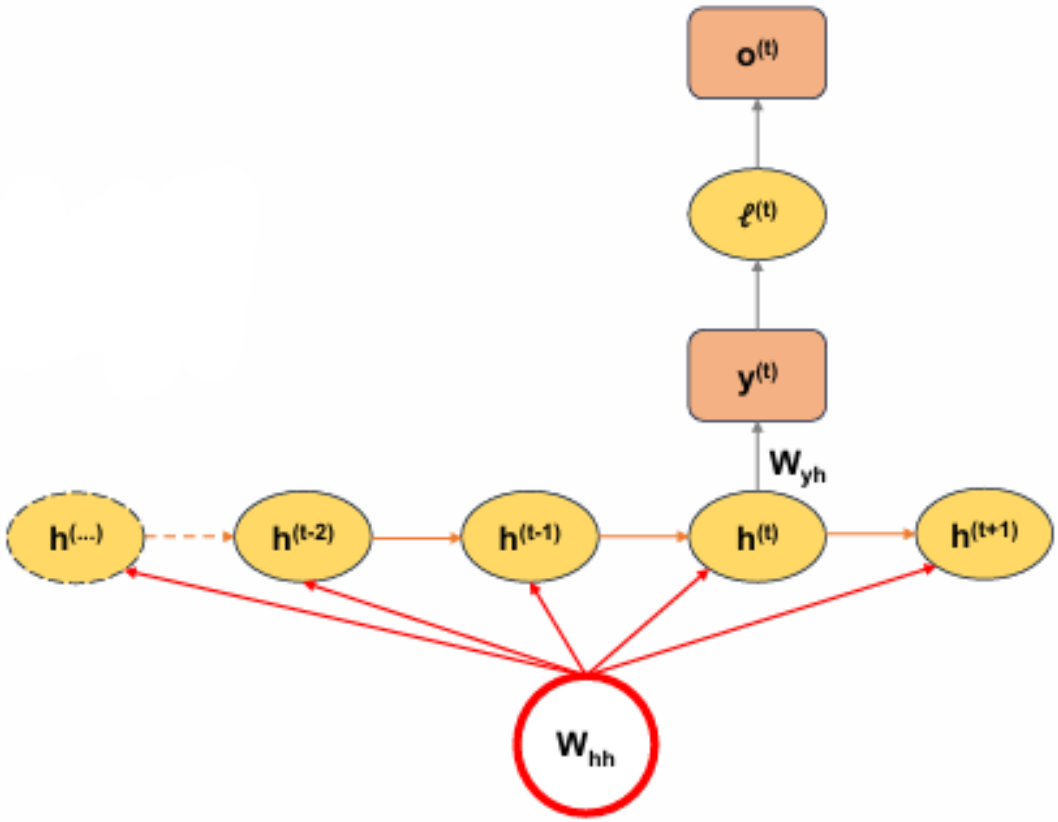


Back-propagation Through Time (BPTT)

$$\frac{\partial \ell^{(t)}}{\partial W_{hh}} = \frac{\partial \ell^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial z_y^{(t)}} \cdot \frac{\partial z_y^{(t)}}{\partial h^{(t)}} \cdot \boxed{\frac{\partial h^{(t)}}{\partial W_{hh}}}$$

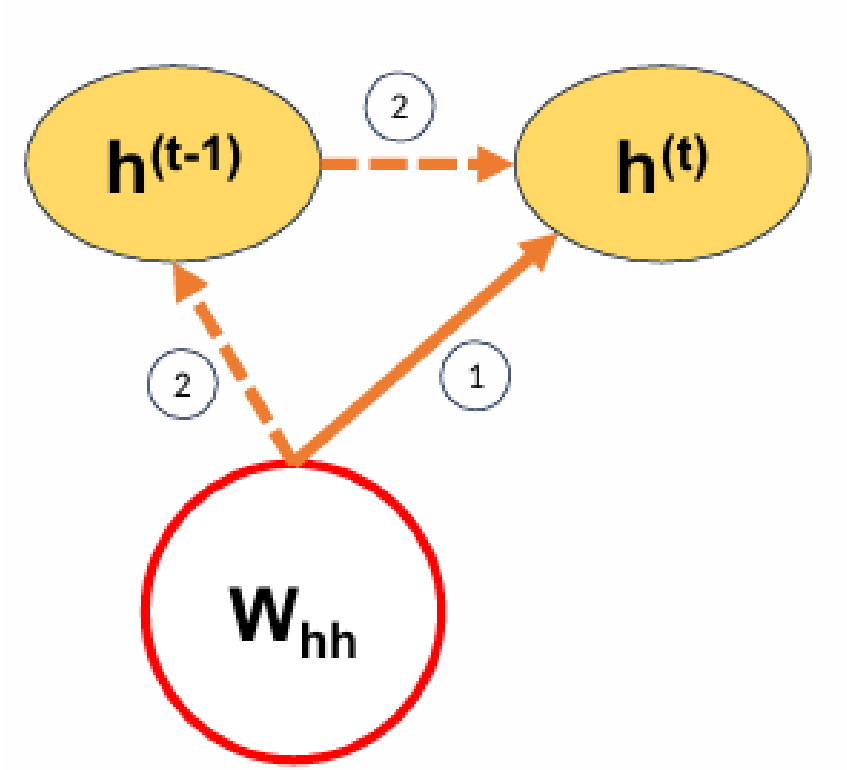


Back-propagation Through Time (BPTT)



- Trạng thái ẩn $h^{(t)}$ phụ thuộc vào $h^{(t-1)}$, và $h^{(t-1)}$ lại phụ thuộc vào $h^{(t-2)}$, v.v...
- Để tìm $\frac{\partial h^{(t)}}{\partial W_{hh}}$, ta cần "quay lui" qua tất cả các bước thời gian, nên gradient "lan truyền" (propagate) ngược lại như hình.

Back-propagation Through Time (BPTT)



$$\frac{\partial h^{(t)}}{\partial W_{hh}} = \boxed{\frac{\partial h^{(t)}}{\partial W_{hh}}}(1) + \boxed{\frac{\partial h^{(t)}}{\partial h^{(t-1)}} \cdot \frac{\partial h^{(t-1)}}{\partial W_{hh}}}(2)$$

Back-propagation Through Time (BPTT)

$$\frac{\partial h^{(t)}}{\partial W_{hh}} = \frac{\partial h^{(t)}}{\partial W_{hh}} + \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \cdot \frac{\partial h^{(t-1)}}{\partial W_{hh}}$$

$$= \frac{\partial h^{(t)}}{\partial W_{hh}} + \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \left(\frac{\partial h^{(t-1)}}{\partial W_{hh}} + \frac{\partial h^{(t-1)}}{\partial h^{(t-2)}} \cdot \frac{\partial h^{(t-2)}}{\partial W_{hh}} \right)$$

Back-propagation Through Time (BPTT)

$$
\begin{aligned}
\frac{\partial h^{(t)}}{\partial W_{hh}} &= \frac{\partial h^{(t)}}{\partial W_{hh}} + \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \cdot \frac{\partial h^{(t-1)}}{\partial W_{hh}} \\
&= \frac{\partial h^{(t)}}{\partial W_{hh}} + \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \left(\frac{\partial h^{(t-1)}}{\partial W_{hh}} + \frac{\partial h^{(t-1)}}{\partial h^{(t-2)}} \cdot \frac{\partial h^{(t-2)}}{\partial W_{hh}} \right) \\
&= \frac{\partial h^{(t)}}{\partial W_{hh}} + \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \cdot \frac{\partial h^{(t-1)}}{\partial W_{hh}} + \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \cdot \frac{\partial h^{(t-1)}}{\partial h^{(t-2)}} \cdot \frac{\partial h^{(t-2)}}{\partial W_{hh}} \\
&= \frac{\partial h^{(t)}}{\partial W_{hh}} + \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \cdot \frac{\partial h^{(t-1)}}{\partial W_{hh}} + \frac{\partial h^{(t)}}{\partial h^{(t-2)}} \cdot \frac{\partial h^{(t-2)}}{\partial W_{hh}} \\
&= \frac{\partial h^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial W_{hh}} + \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \cdot \frac{\partial h^{(t-1)}}{\partial W_{hh}} + \frac{\partial h^{(t)}}{\partial h^{(t-2)}} \cdot \frac{\partial h^{(t-2)}}{\partial W_{hh}} \\
&= \sum_{k=1}^{t} \frac{\partial h^{(t)}}{\partial h^{(k)}} \cdot \frac{\partial h^{(k)}}{\partial W_{hh}}
\end{aligned}
$$

Back-propagation Through Time (BPTT)

Thay vào công thức, ta được:

$$\frac{\partial \ell^{(t)}}{\partial W_{hh}} = \frac{\partial \ell^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial z_y^{(t)}} \cdot \frac{\partial z_y^{(t)}}{\partial h^{(t)}} \cdot \boxed{\frac{\partial h^{(t)}}{\partial W_{hh}}}$$

$$= \frac{\partial \ell^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial z_y^{(t)}} \cdot \frac{\partial z_y^{(t)}}{\partial h^{(t)}} \cdot \sum_{k=1}^{t} \frac{\partial h^{(t)}}{\partial h^{(k)}} \cdot \frac{\partial h^{(k)}}{\partial W_{hh}}$$

Back-propagation Through Time (BPTT)

- Vì mỗi h_t phụ thuộc vào bước thời gian trước đó, ta có thể **thay thế** phần cuối trong phương trình ở trên (để rút gọn hoặc khai triển lại).

Đạo hàm riêng phần theo W_{hh}

$$\frac{\partial L}{\partial W_{hh}} = \sum_{t=1}^{T} \frac{\partial \ell^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial z_y^{(t)}} \cdot W_{yh} \sum_{k=1}^{t} \frac{\partial h^{(t)}}{\partial h^{(k)}} \cdot \frac{\partial h^{(k)}}{\partial W_{hh}}$$

- Tương tự, ta có đạo hàm riêng phần theo W_{xh} như sau:

Đạo hàm riêng phần theo W_{xh}

$$\frac{\partial L}{\partial W_{xh}} = \sum_{t=1}^{T} \frac{\partial \ell^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial z_y^{(t)}} \cdot W_{yh} \sum_{k=1}^{t} \frac{\partial h^{(t)}}{\partial h^{(k)}} \cdot \frac{\partial h^{(k)}}{\partial W_{xh}}$$

Back-propagation Through Time (BPTT)

- Để chuyển lỗi qua các bước thời gian, từ thời điểm t quay trở về bước k, ta có thể viết:

$$\frac{\partial h^{(t)}}{\partial h^{(k)}} = \prod_{i=k+1}^{t} \frac{\partial h^{(i)}}{\partial h^{(i-1)}}.$$

- Ta có thể xem phương trình trước đó như **một ma trận Jacobian** cho tham số trạng thái ẩn (hidden state), cụ thể:

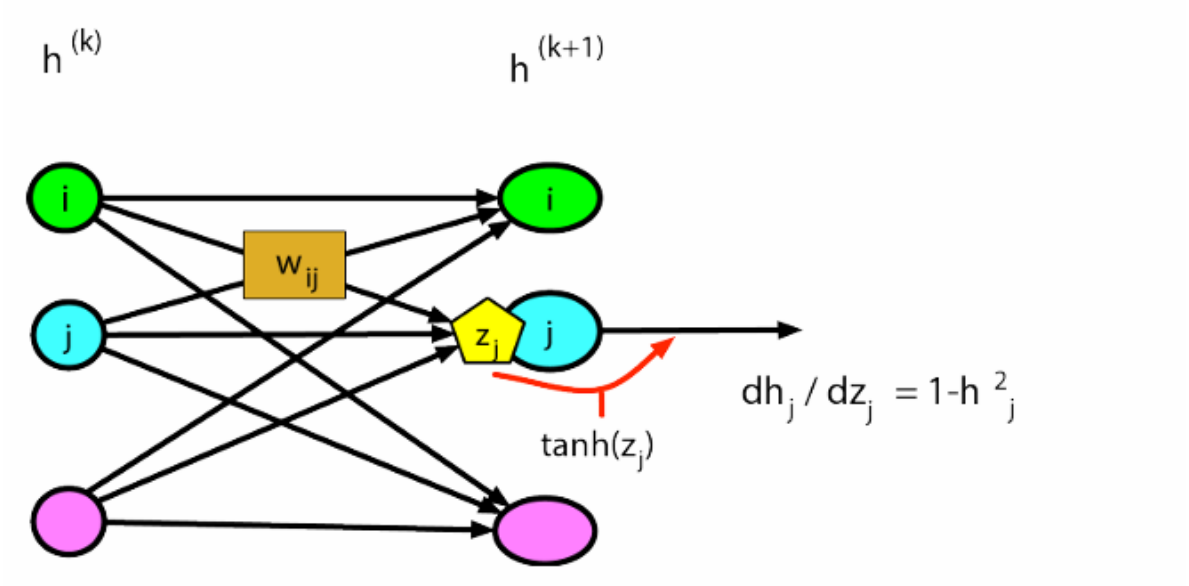
Jacobian cho $\frac{\partial h^{(t)}}{\partial h^{(k)}}$

$$\prod_{i=k+1}^{t} \frac{\partial h^{(i)}}{\partial h^{(i-1)}} = \prod_{i=k+1}^{t} W_{hh}^{\top} \operatorname{diag}(\phi_h'(z_h^{(i)})).$$

Back-propagation Through Time (BPTT)

$$\frac{\partial h^{(k+1)}}{\partial h^{(k)}} = \begin{bmatrix} \frac{\partial h_1^{(k+1)}}{\partial h_1^{(k)}} & \frac{\partial h_1^{(k+1)}}{\partial h_2^{(k)}} & \cdots & \frac{\partial h_1^{(k+1)}}{\partial h_m^{(k)}} \\ \frac{\partial h_2^{(k+1)}}{\partial h_1^{(k)}} & \frac{\partial h_2^{(k+1)}}{\partial h_2^{(k)}} & \cdots & \frac{\partial h_2^{(k+1)}}{\partial h_m^{(k)}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial h_m^{(k+1)}}{\partial h_1^{(k)}} & \frac{\partial h_m^{(k+1)}}{\partial h_2^{(k)}} & \cdots & \frac{\partial h_m^{(k+1)}}{\partial h_m^{(k)}} \end{bmatrix}$$

Back-propagation Through Time (BPTT)



- Giả sử **hàm kích hoạt** cho trạng thái ẩn h là tanh.
- $$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh^2(x).$$

‘

- Khi đó, ta có:

$$\frac{\partial h_j^{(k+1)}}{\partial h_i^{(k)}} = w_{ij}\left[1 - (h_j^{(k+1)})^2\right].$$

Back-propagation Through Time (BPTT)

$$\frac{\partial h^{(k+1)}}{\partial h^{(k)}} = \begin{pmatrix} \left(1-(h_1^{(k+1)})^2\right) w_{11} & \left(1-(h_1^{(k+1)})^2\right) w_{21} & \dots & \left(1-(h_1^{(k+1)})^2\right) w_{m1} \\ \left(1-(h_2^{(k+1)})^2\right) w_{12} & \left(1-(h_2^{(k+1)})^2\right) w_{22} & \dots & \dots \\ \vdots & \vdots & \ddots & \vdots \\ \left(1-(h_m^{(k+1)})^2\right) w_{1m} & \dots & \dots & \left(1-(h_m^{(k+1)})^2\right) w_{mm} \end{pmatrix}$$

- Thực chất, ma trận trên **chính là $\mathbf{W}^\top$** nhưng với mỗi hàng thứ j **được nhân** bởi $\left[1-(h_j^{(k+1)})^2\right]$.
- Hoặc viết gọn:

$$\frac{\partial h^{(k+1)}}{\partial h^{(k)}} = \operatorname{diag}\left(1-(h^{(k+1)})^2\right) \cdot \mathbf{W}^\top.$$

Table of Contents

Difficulty of training RNNs

$$\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\| = \left\| \mathrm{diag}(\sigma'(z_i))\, W \right\| \quad \leq \quad \left\| \mathrm{diag}(\sigma'(z_i)) \right\| \, \|W\|$$

$$\sigma'(z_i) \leq \frac{1}{4} = \gamma \quad \text{(nếu } \sigma \text{ là sigmoid)}$$

$$\sigma'(z_i) \leq 1 = \gamma \quad \text{(nếu } \sigma \text{ là tanh)}$$

Từ đó suy ra:

$$\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\| \leq \gamma \, \|W\| \qquad \text{và} \qquad \left\| \frac{\partial h_i}{\partial h_{i-1}} \right\| \leq \gamma \, \lambda.$$

Prove the derivative of the sigmoid function

$$f(x) = \frac{1}{1+e^{-x}}$$

$$f'(x) = \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \left(\frac{-1}{(1+e^{-x})^2}\right)(-e^{-x})$$

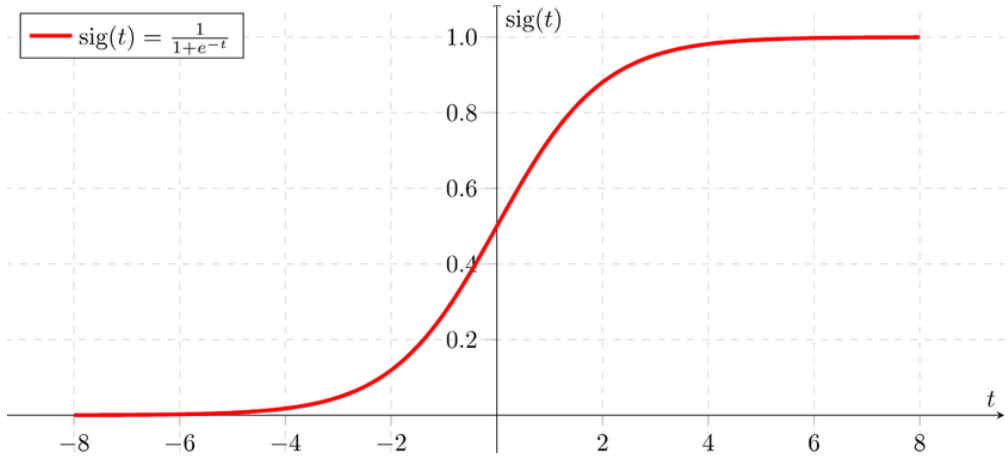
$$= \frac{e^{-x}}{(1+e^{-x})^2} = \frac{(1+e^{-x})-1}{(1+e^{-x})^2}$$

$$= \frac{1+e^{-x}}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}} - \frac{1}{(1+e^{-x})^2}$$

$$= f(x) - f(x)^2 = f(x)(1-f(x))$$

$$\boxed{f'(x) = f(x)(1-f(x)).}$$

Gradient of Sigmoid Function



Định nghĩa: Giả sử hàm sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Đạo hàm của $\sigma(x)$ tại x là

$$\sigma'(x) = \frac{d}{dx}(\sigma(x)) = \frac{1}{1 + e^{-x}}\left(1 - \frac{1}{1 + e^{-x}}\right).$$

Ý nghĩa: $\sigma'(x)$ là độ dốc (slope) của sigmoid tại điểm x.

Gradient of Sigmoid Function

$$x = 10 \quad \Rightarrow \quad \sigma(10) \approx 1 \quad \Rightarrow \quad \frac{d}{dx}\sigma(x)\Big|_{x=10} \approx 1 \cdot (1-1) = 0.$$

$$x = -10 \quad \Rightarrow \quad \sigma(-10) \approx 0 \quad \Rightarrow \quad \frac{d}{dx}\sigma(x)\Big|_{x=-10} \approx 1 \cdot (1-1) = 0.$$

$$x = 0 \quad \Rightarrow \quad \sigma(0) = \frac{1}{2} \quad \Rightarrow \quad \frac{d}{dx}\sigma(x)\Big|_{x=0} = 1 \cdot \left(1 - \frac{1}{2}\right) = \frac{1}{4}.$$

Difficulty of training RNNs

$$\left\|\frac{\partial h_t}{\partial h_k}\right\| = \left\|\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}}\right\| \leq \prod_{i=k+1}^{t} \gamma\lambda \leq (\gamma\lambda)^{(t-k)}.$$

1. $(\gamma\lambda) > 1$ thì gradient **bùng nổ** (explode).
2. $(\gamma\lambda) < 1$ thì gradient **biến mất** (vanish).

Exploding Gradient

- Khi huấn luyện RNN trên các chuỗi dài, gradient có thể **bùng nổ** vì trọng số trở nên quá lớn, và chuẩn (norm) của gradient trong quá trình huấn luyện tăng mạnh.
- Như đã nêu ở trên, điều kiện cần để xảy ra hiện tượng bùng nổ gradient là

$$\lambda > \frac{1}{\gamma}.$$

Solutions to Vanishing Gradient

- **Sử dụng hàm kích hoạt phù hợp** (ReLU, LeakyReLU, ELU...):
  - Tránh "bóp" giá trị đầu ra về quá nhỏ như sigmoid/tanh khi x lớn hoặc rất âm.
  - ReLU giữ gradient ở mức ổn định hơn cho các giá trị dương.
- **Khởi tạo trọng số tốt** (Xavier/Glorot, He initialization):
  - Bảo đảm độ lớn của giá trị đi qua mạng không bị giảm quá nhanh.
- **LSTM/GRU**:
  - Các cổng (gate) giúp "ghi nhớ"/"quên" phù hợp, giảm thiểu hiện tượng tiêu biến gradient ở các bước xa.
- **Kết nối tắt (skip connections), Residual networks**:
  - Cho phép gradient "nhảy cóc" qua một số lớp/trạng thái ẩn.
- **Batch Normalization** (tuỳ trường hợp RNN/LSTM):
  - Ổn định phân phối của các kích hoạt ẩn, giúp việc lan truyền gradient tốt hơn.

Solutions to Exploding Gradient

- **Gradient Clipping**:
  - Cắt (clip) giá trị gradient khi nó vượt quá một ngưỡng nhất định.
  - Tránh việc cập nhật trọng số quá lớn (bước nhảy quá to).
- **Giảm Learning Rate**:
  - Nếu bước nhảy (learning rate) quá cao, dễ gây bùng nổ gradient.
- **Regularization (L2, Weight decay)**:
  - Hạn chế trọng số tăng quá cao, qua đó giảm bớt độ lớn của gradient.
- **Sử dụng LSTM/GRU**:
  - Ngoài việc chống vanishing gradient, cơ chế cổng (gate) cũng giúp giảm độ "đột ngột" của gradient.
- **Cấu trúc mạng hợp lý** (một số trường hợp):
  - Tránh mô hình quá sâu hoặc quá lớn mà không có regularization, vì điều này làm gradient dễ "bùng nổ".

Questions are welcome.