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

Dang Huu Phat 22521065
Phan Nguyen Huu Phong 22521090
Chau The Vi 22521653

University of Information Technology

December 31, 2023

References

The contents of this document are taken mainly from the follow sources:

- Robin M.Schmidt, Recurrent Neural Networks (RNNs): a gentle introductionand overview[1];
- Razvan Pascanu, Tomas Mikolov, Yoshua Bengio, On the difficulty of training RNNs[2]
- Yoshua Bengio, Patrice Simard, Paolo Frasconi, Learning long-term dependencies with gradients descent is difficult[3]

[1] https://arxiv.org/pdf/1912.05911.pdf

[2] https://arxiv.org/pdf/1211.5063.pdf

[3] https://www.researchgate.net/profile/Y-Bengio/publication/5583935_Learning_long-term_dependencies_with_gradient_descent_is_difficult/links/546b702d0cf2397f7831c03d/Learning-long-term-dependencies-with-gradient-descent-is-difficult.pdf

Table of Contents

Table of Contents

Sequential Data

- Sequential data, which more commonly known as sequence data or **sequences**
- What makes it unique? It is that elements in a sequence appear in a **certain order** and are **not independent** of each other.
- **Example: Stock Price Prediction**
  In this example, the input data is the historical stock prices. i_1 is the stock price on January 1st, i_2 is the stock price on January 2nd, and so on. $(i_1, i_2, ...i_{30})$ is called sequences. RNN will learn from the input and predict the stock price in the future.

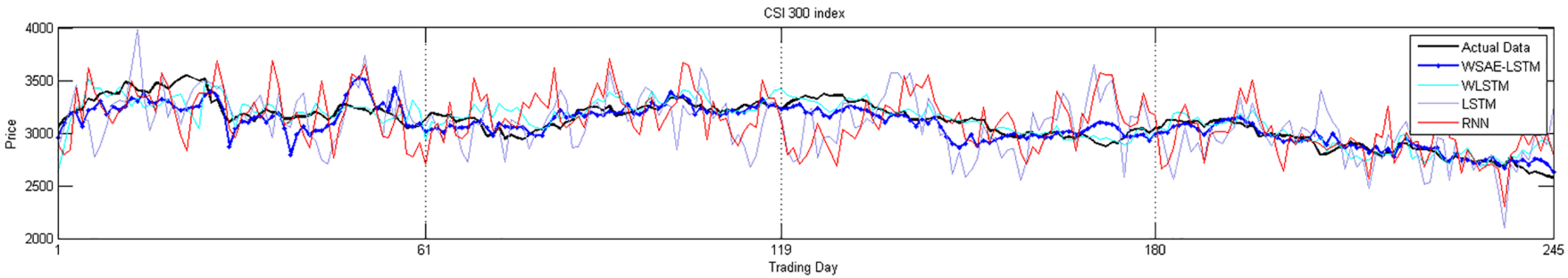


Figure: Example of Stock Price Prediction

Sequential data versus time series data

- Time series data is a special type of sequential data where each example is associated with a dimension for time.
- In time series data, samples are taken at successive timestamps, and therefore, the time dimension determines the order among the data points. For example, stock prices and voice or speech records are time series data.
- In contrast, not all sequential data shares this temporal structure. Text data or DNA sequences, for instance, exhibit ordered patterns, but lack a distinct time dimension.

Sequential data versus time series data

- A example of machine translation, input is sentence: "I love CS115", the order of words significantly impact on the meaning of the sentence. Input data, which consists of the sequence of words ['I', 'love', 'CS115'] is refered to as **sequences**.
- In natural language processing (NLP), it is not feasible to process entire sentences as inputs. Similar to how videos are processed by breaking them down into individual frames, in NLP, sentences are broken down into individual words for input processing.

Representing sequences

- The movie my friend has **not** seen is good
- The movie my friend has seen is **not** good

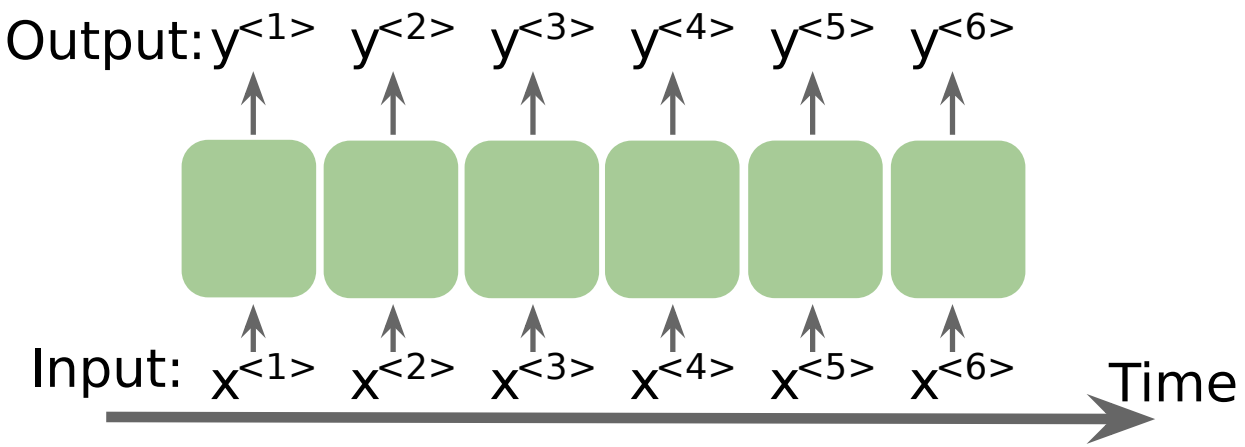
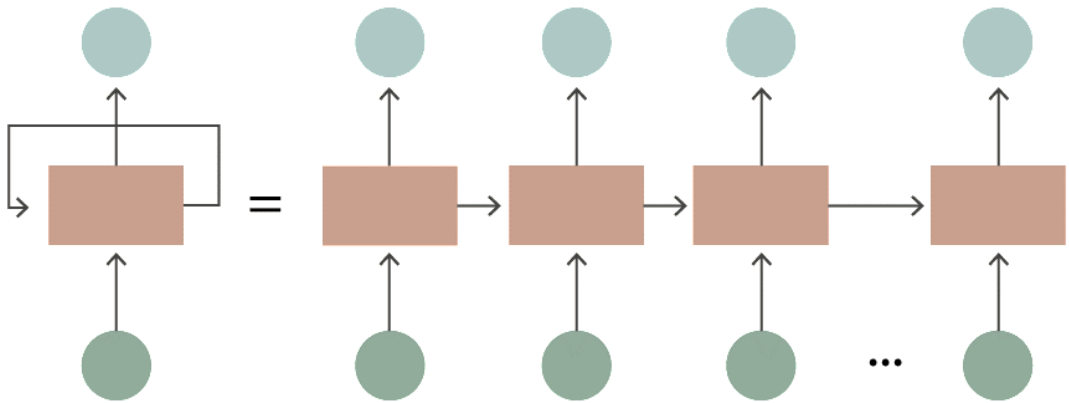


Figure: An example of time series data

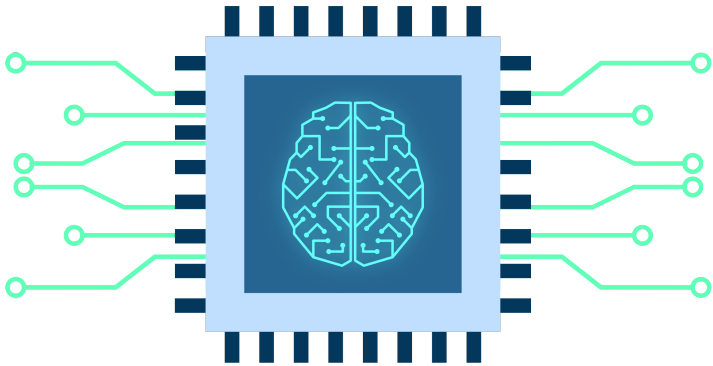
What is Recurrent Neural Networks (RNNs)?

- Recurrent Neural Networks (RNNs) is a type of neural network architecture which is used to detect patterns in a sequences.
- They are also can be extended to images by breaking them down into a series of patches and treating them as a sequence.
- What differentiates Recurrent Neural Networks from Feedforward Neural Networks also known as Multi-Layer Perceptrons (MLPs) is how information gets passed through the network.



What is Recurrent Neural Networks (RNNs)?

- Multilayer perceptrons (MLPs) and CNNs for image data, assume that the training examples are **independent** of each other and thus do not incorporate *ordering information*.
- We can say that such models do not have a *memory* of previously seen training examples.
- RNNs, by contrast, are designed for modeling sequences and are capable of remembering past information and processing new events accordingly, which is a clear advantage when working with sequence data.



Pros and Cons

Benefits

- Possibility of processing input of any length;
- Model size not increasing with size of input;
- Computation takes into account historical information;
- Weights are shared across time

Costs

- Computation being slow;
- Difficulty of accessing information from a long time ago;
- Cannot consider any future input for the current state;

The different categories of sequence modeling

- One to One
- One to Many
- Many to One
- Many to Many

One to One RNN

- This type of neural network is known as the Vanilla Neural Network. It's used for general machine learning problems, which has a single input and a single output.

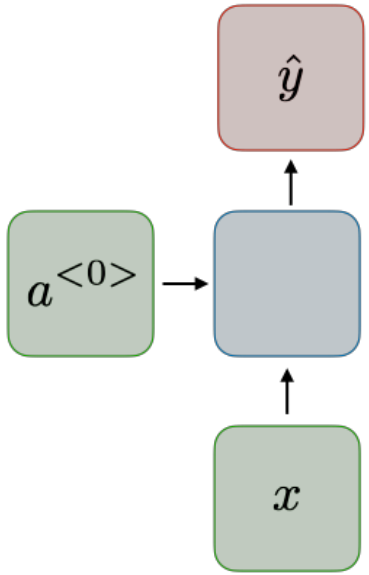


Figure: Vanilla Neural Network

The different categories of sequence modeling

One to Many RNN

- A type of RNN that gives multiple outputs when given a single input. It takes a fixed input size and gives a sequence of data outputs.
- Its applications can be found in Music Generation and Image Captioning.

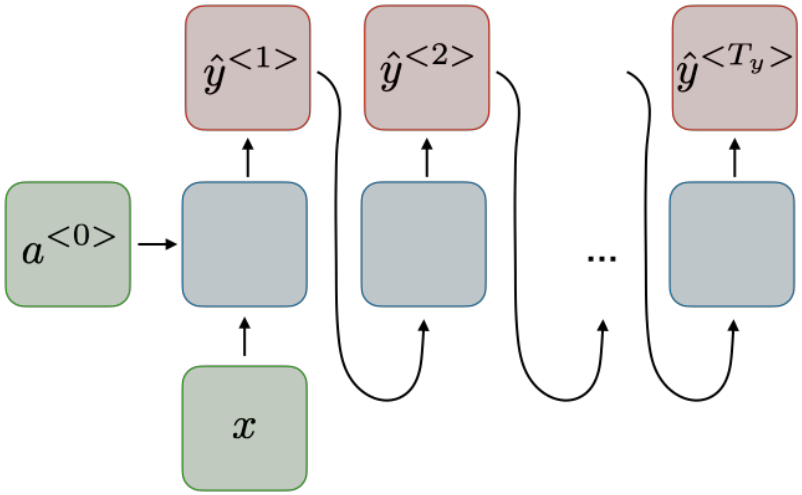


Figure: One-to-Many mode

Many to One RNN

- Many-to-One is used when a single output is required from multiple input units or a sequence of them. It takes a sequence of inputs to display a fixed output.
- Sentiment Analysis is a common example of this type of Recurrent Neural Network.

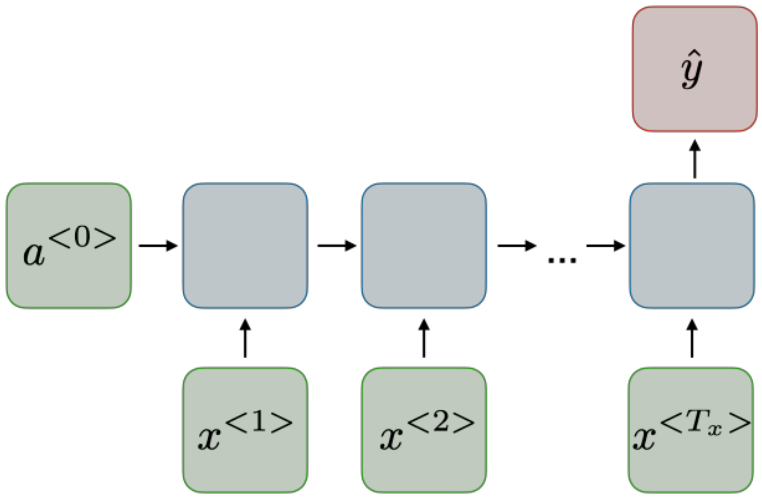
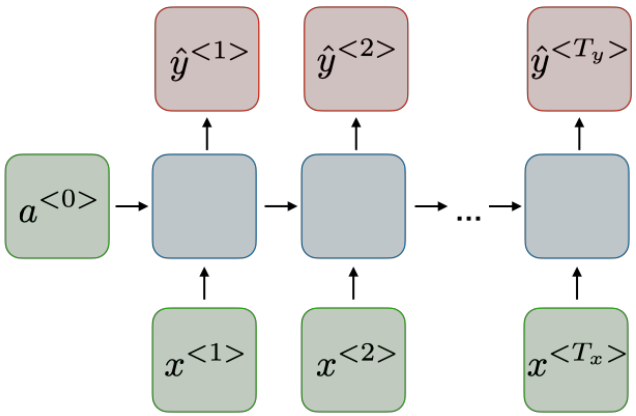


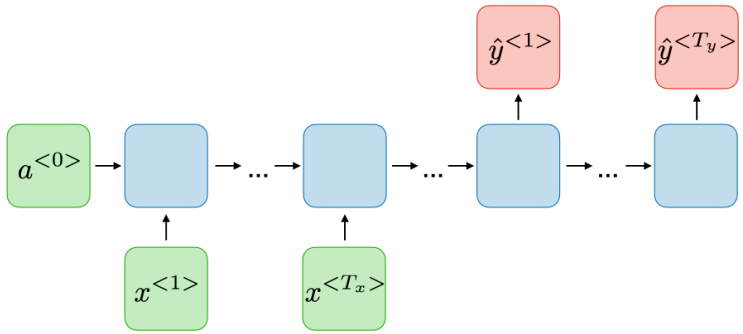
Figure: Many to One mode

The different categories of sequence modeling

Many to Many RNN

- Many-to-Many is used to generate a sequence of output data from a sequence of input units.
- This type of RNN is further divided into the following two subcategories:
  1. Equal Unit Size: In this case, the number of both the input and output units is the same. A common application can be found in Name-Entity Recognition.
  2. Unequal Unit Size In this case, inputs and outputs have different numbers of units. Its application can be found in Machine Translation.





Understanding the dataflow in RNNs

- In a standard feedforward network, information flows from the input to the hidden layer, and then from the hidden layer to the output layer.
- On the other hand, in an RNN, the hidden layer receives its input from both the input layer of the current time step and the hidden layer from the previous time step.

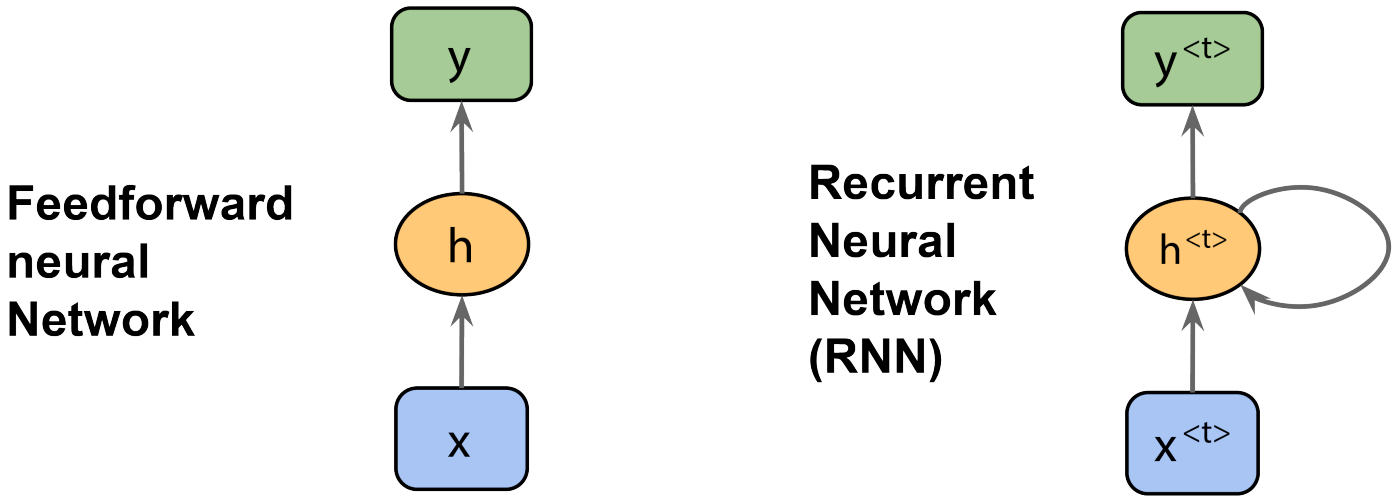


Figure: The dataflow of a standard feedforward NN and an RNN

Understanding the dataflow in RNNs

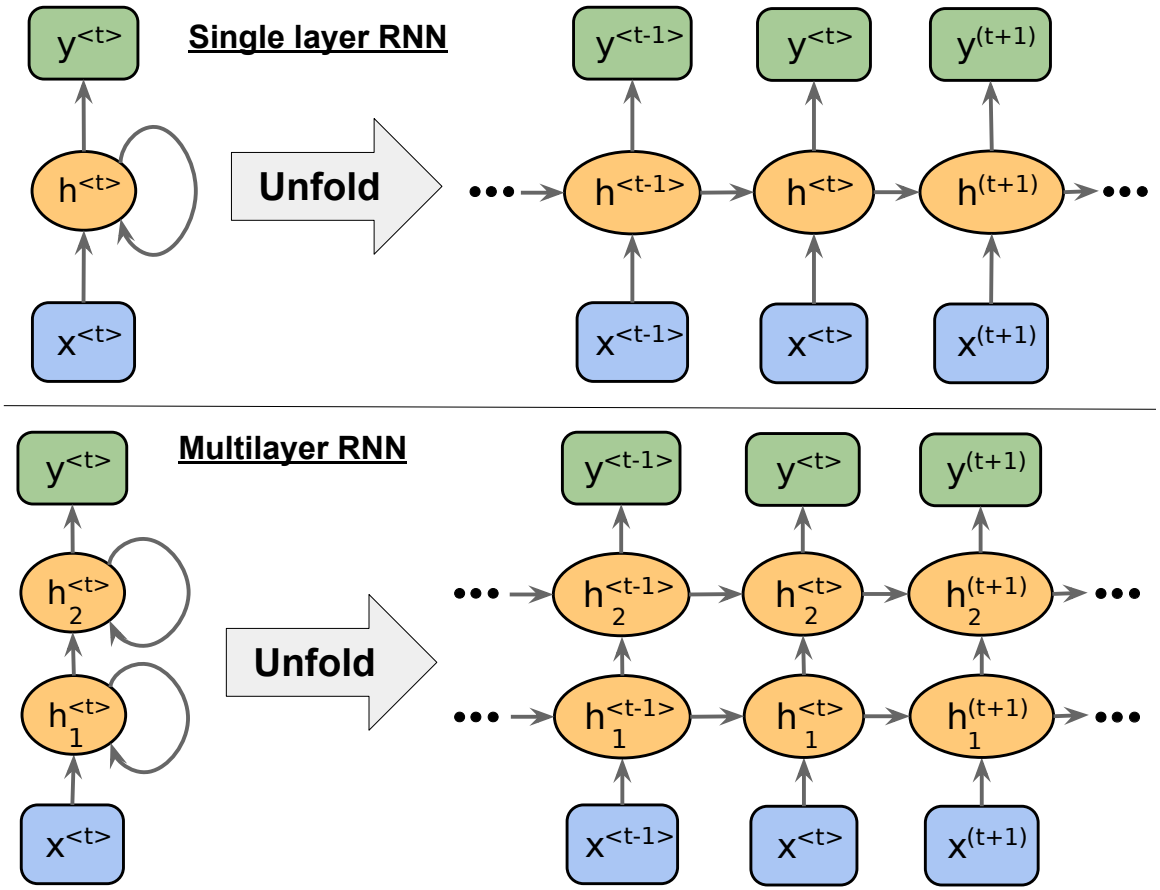
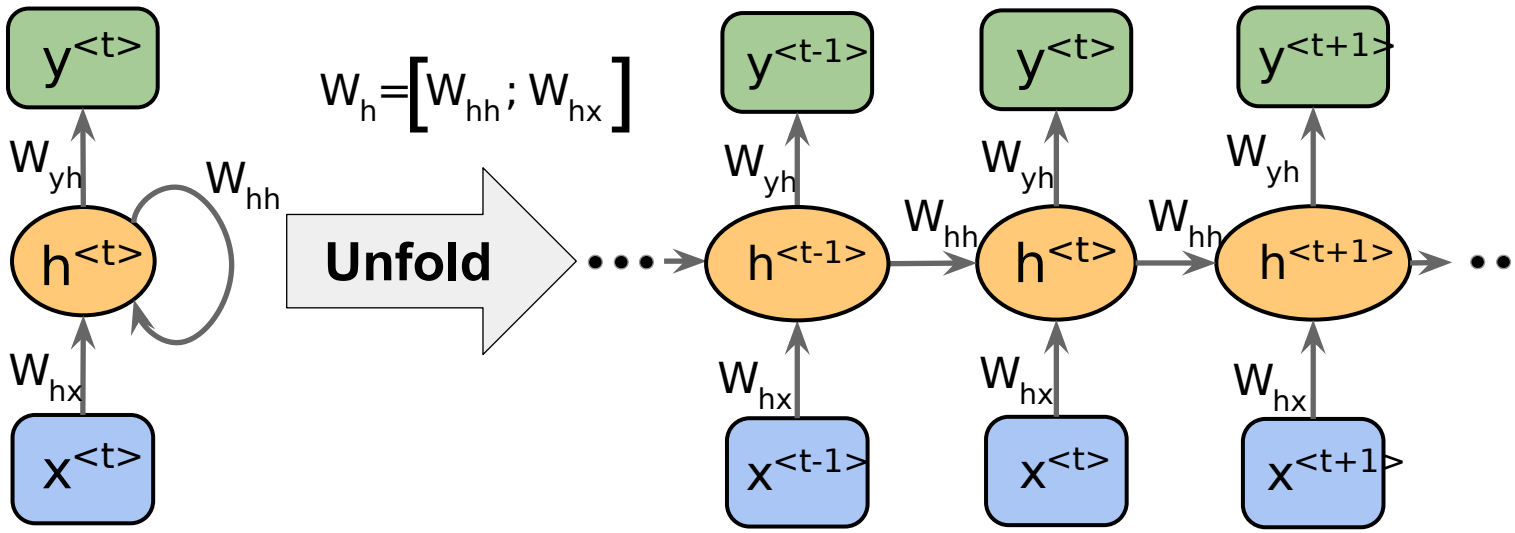


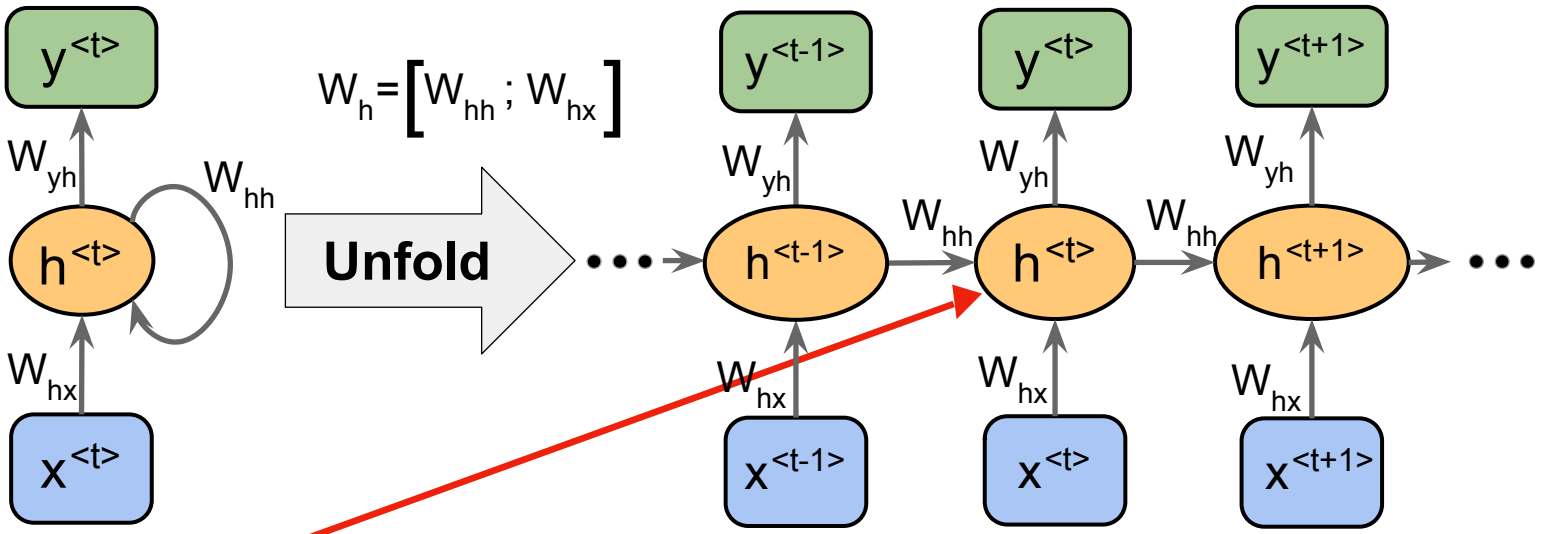
Figure: Examples of an RNN with one and two hidden layers

Table of Contents

Weight matrices in a single-hidden layer RNN



Weight matrices in a single-hidden layer RNN



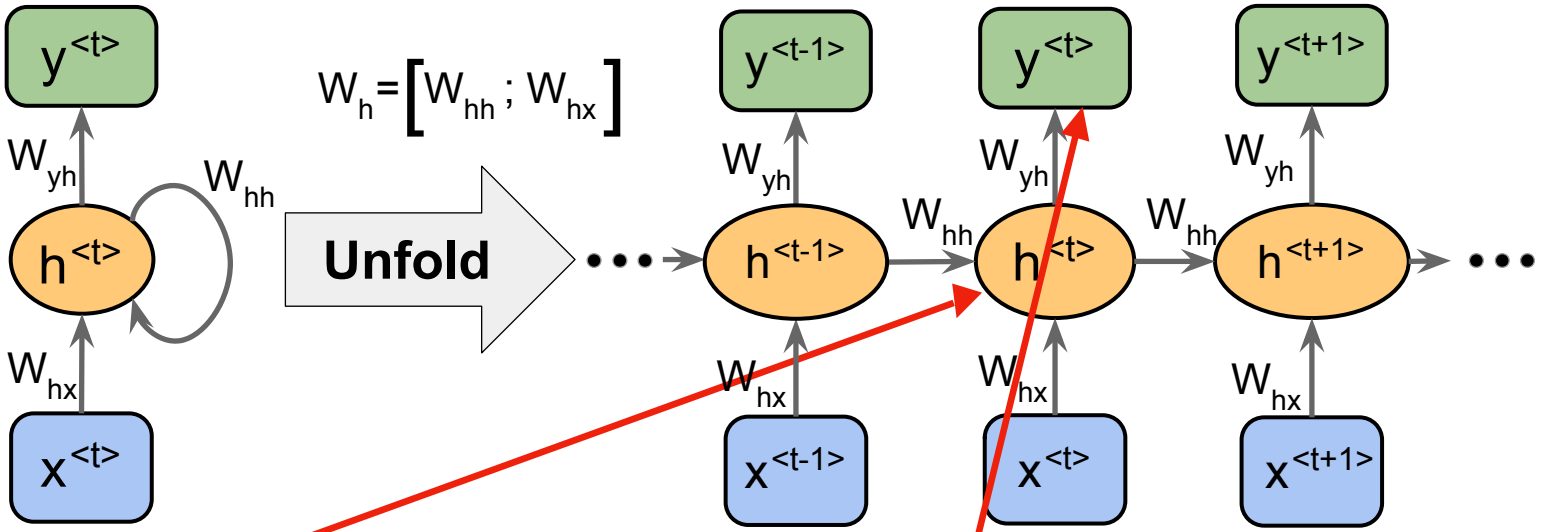
Net input:

$$z_h^{\langle t \rangle} = W_{hx} x^{\langle t \rangle} + W_{hh} h^{\langle t-1 \rangle} + b_h$$

Activation:

$$h^{\langle t \rangle} = \phi_h\left(z_h^{\langle t \rangle}\right)$$

Weight matrices in a single-hidden layer RNN



Net input:

$$z_h^{\langle t \rangle} = W_{hx} x^{\langle t \rangle} + W_{hh} h^{\langle t-1 \rangle} + b_h$$

Activation:

$$h^{\langle t \rangle} = \varnothing_h \left(z_h^{\langle t \rangle} \right)$$

Net input:

$$z_y^{\langle t \rangle} = W_{yh} h^{\langle t \rangle} + b_y$$

Output:

$$y^{\langle t \rangle} = \varnothing_y \left(z_y^{\langle t \rangle} \right)$$

Gradient-based Learning Methods

- Gradient descent (GD) adjusts the weights of the model by finding the error function derivatives with respect to each member of the weight matrices.

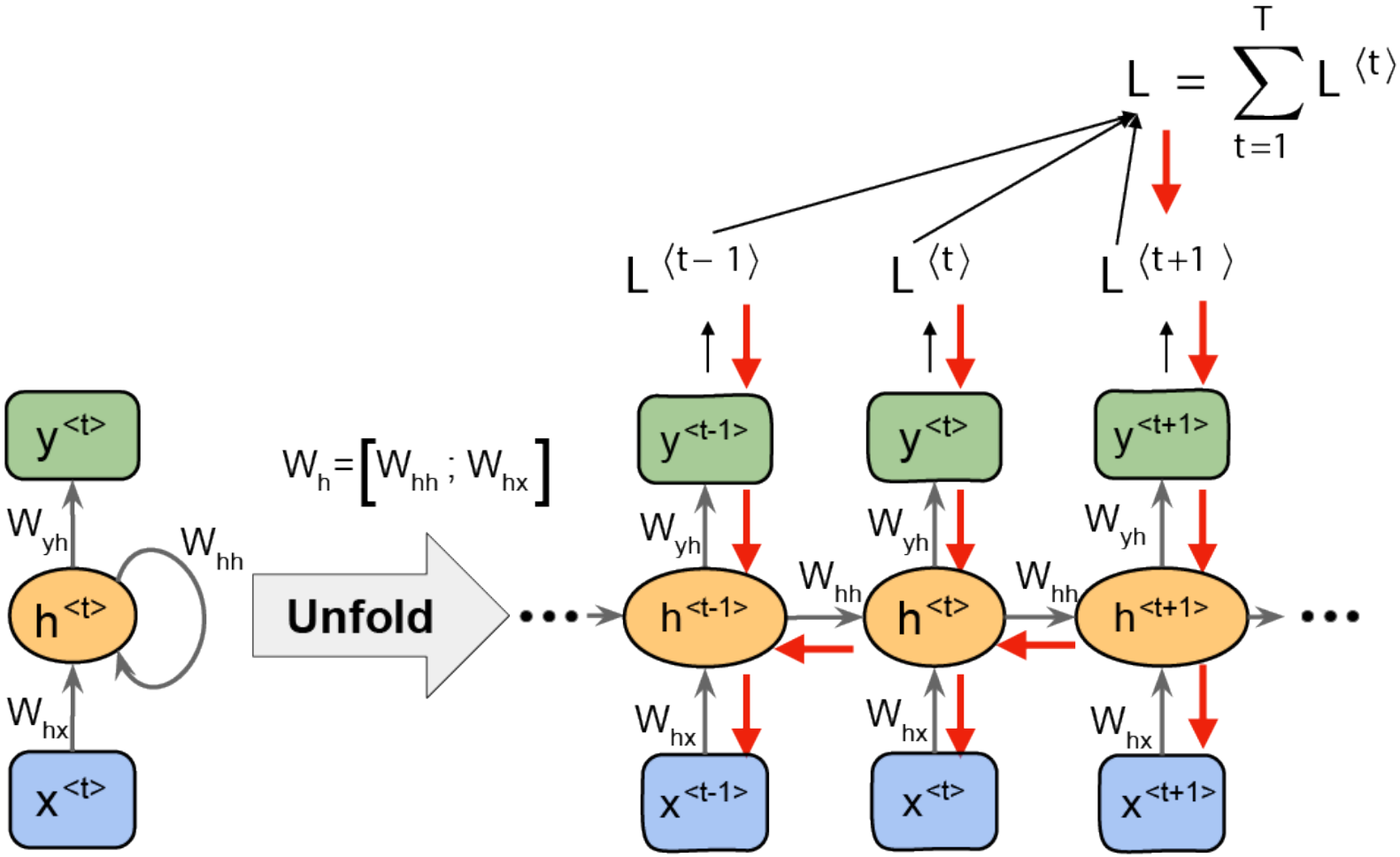
Batch Gradient Descent

Computing the gradient for the whole dataset in each optimization iteration to perform a single update as

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

- However, computing error-derivatives through time is difficult. This is mostly due to the relationship among the parameters and the dynamics of the RNN, that is highly unstable and makes GD ineffective.

Back-propagation Through Time (BPTT)



Back-propagation Through Time (BPTT)

Hidden variable

$$\mathbf{h}^{(t)} = \phi_h(\mathbf{x}_t \cdot W_{xh} + \mathbf{h}^{(t-1)} \cdot W_{hh} + \mathbf{b}_h)$$

Output variable

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

We can define a loss function $L(y, o)$ to describe the difference between all outputs y_t and the target values o_t as shown in the equation below.

Loss Function

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

Back-propagation Through Time (BPTT)

Since we have three weight matrices $\mathbf{W}_{xh}, \mathbf{W}_{hh}$ and $\mathbf{W}_{yh}$ we need to compute the partial derivative to each of these weight matrices using the chain rule.

Partial derivative with respect to $\mathbf{W}_{yh}$

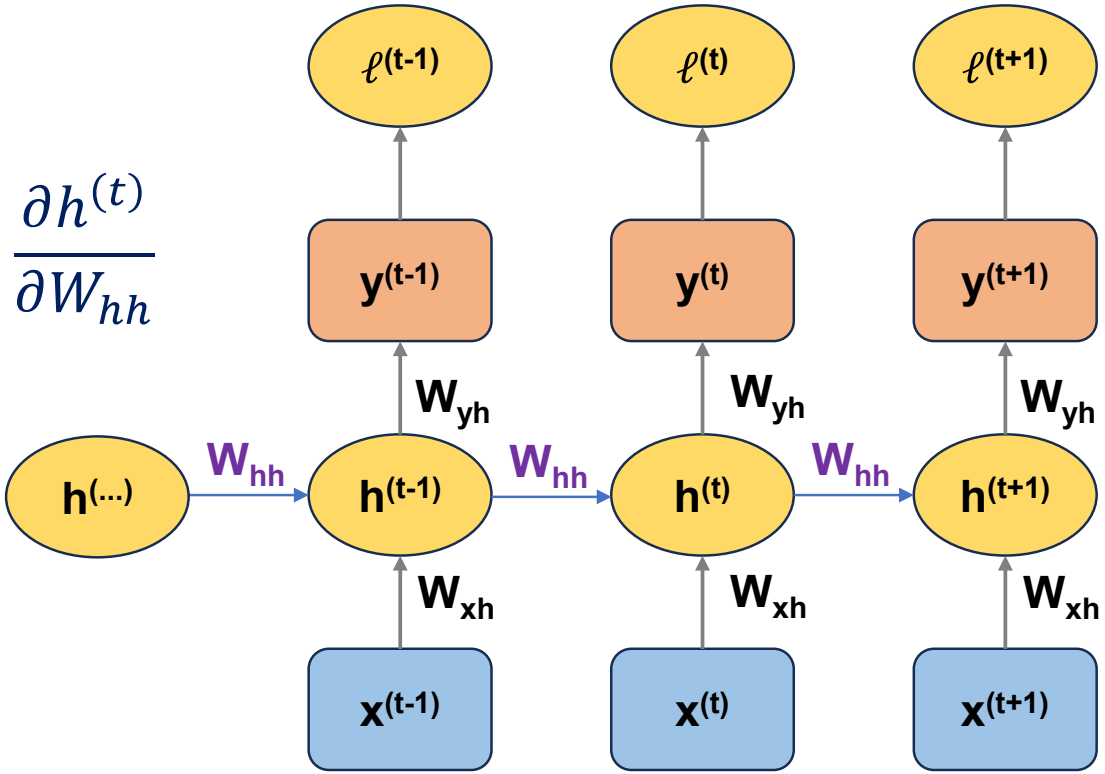
$$\frac{\partial L}{\partial \mathbf{W}_{yh}} = \sum_{t=1}^{T} \frac{\partial l^{(t)}}{\partial \mathbf{y}^{(t)}} \cdot \frac{\partial \mathbf{y}^{(t)}}{\partial z_y^{(t)}} \cdot \frac{\partial z_y^{(t)}}{\partial \mathbf{W}_{yh}} = \sum_{t=1}^{T} \frac{\partial l^{(t)}}{\partial \mathbf{y}^{(t)}} \cdot \frac{\partial \mathbf{y}^{(t)}}{\partial z_y^{(t)}} \cdot \mathbf{h}^{(t)}$$

Partial derivative with respect to $\mathbf{W}_{hh}$

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{W}_{hh}} &= \sum_{t=1}^{T} \frac{\partial \ell^{(t)}}{\partial \mathbf{y}^{(t)}} \cdot \frac{\partial \mathbf{y}^{(t)}}{\partial z_y^{(t)}} \cdot \frac{\partial z_y^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{W}_{hh}} \\ &= \sum_{t=1}^{T} \frac{\partial \ell^{(t)}}{\partial \mathbf{y}^{(t)}} \cdot \frac{\partial \mathbf{y}^{(t)}}{\partial z_y^{(t)}} \cdot \mathbf{W}_{yh} \cdot \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{W}_{hh}}\end{aligned}$$

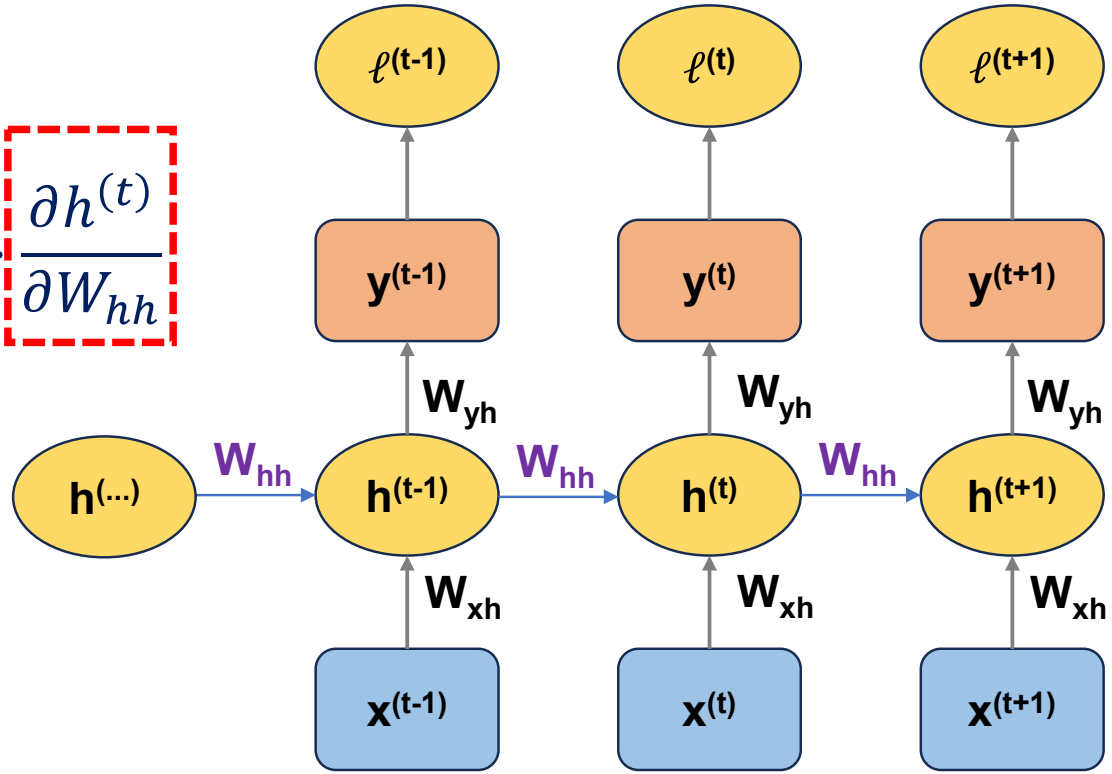
Back-propagation through time (BPTT)

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



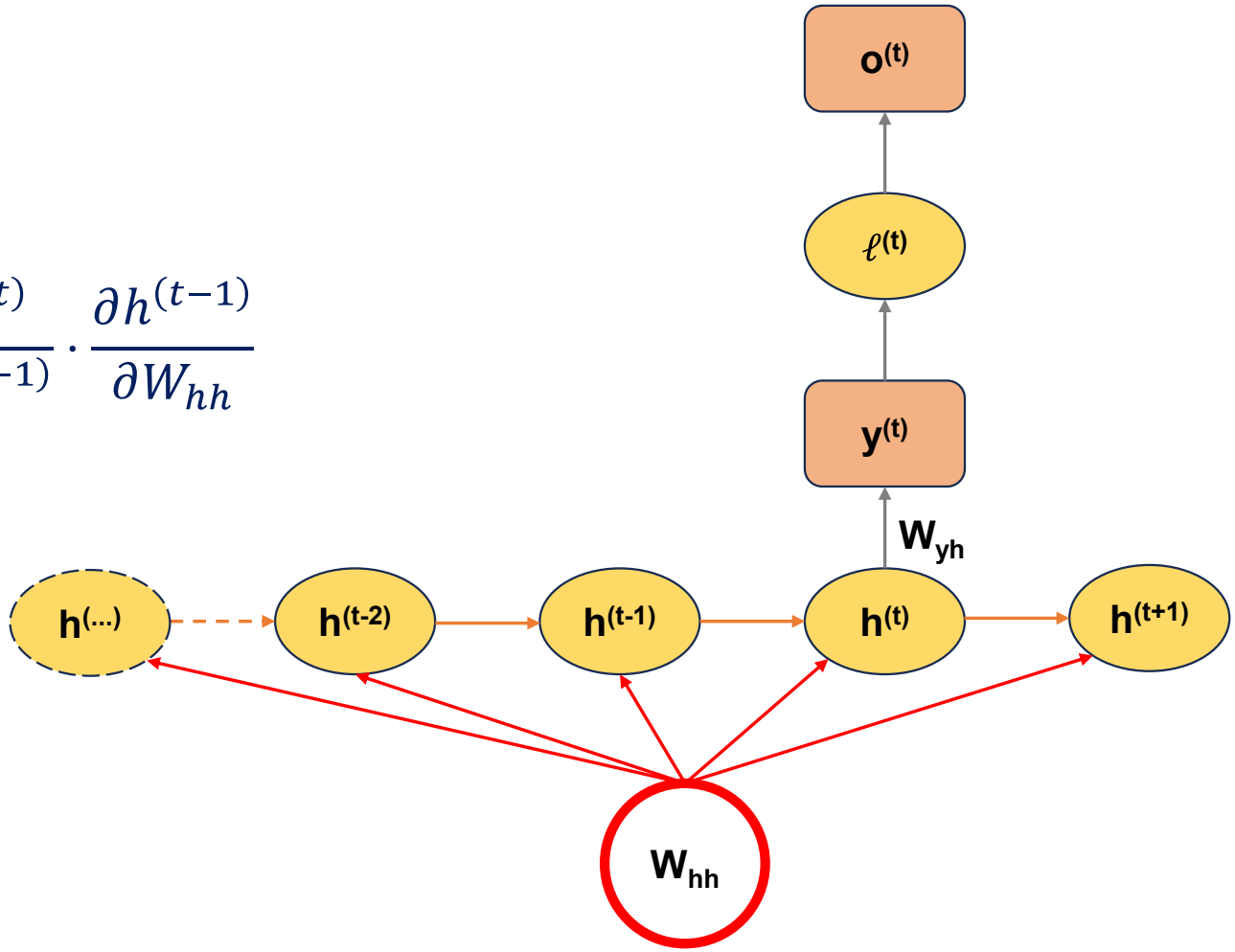
Back-propagation through time (BPTT)

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



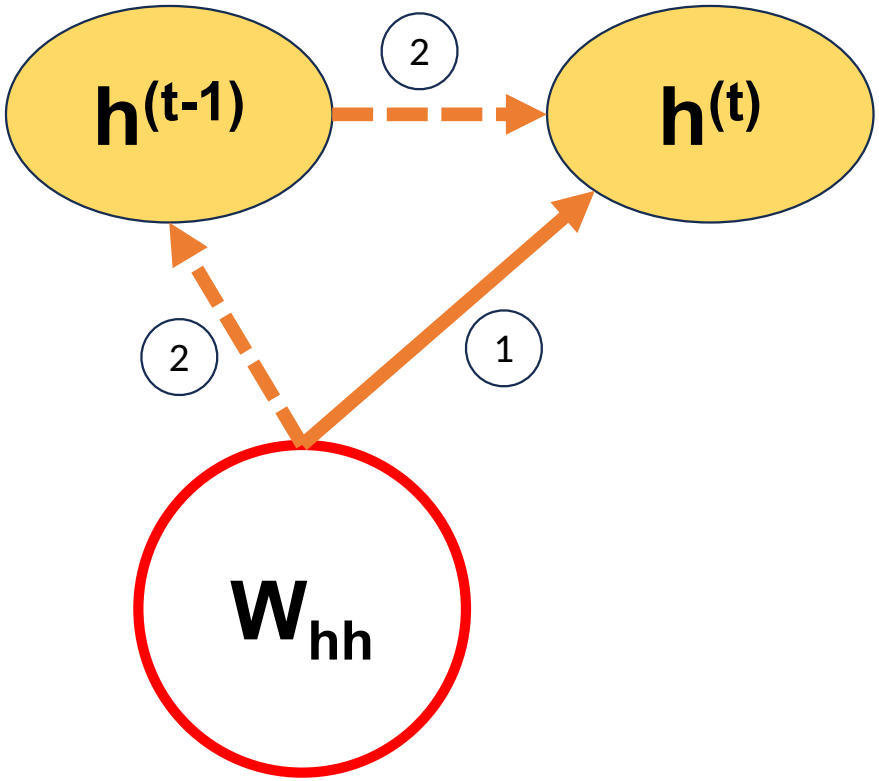
Back-propagation through time (BPTT)

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



Back-propagation through time (BPTT)

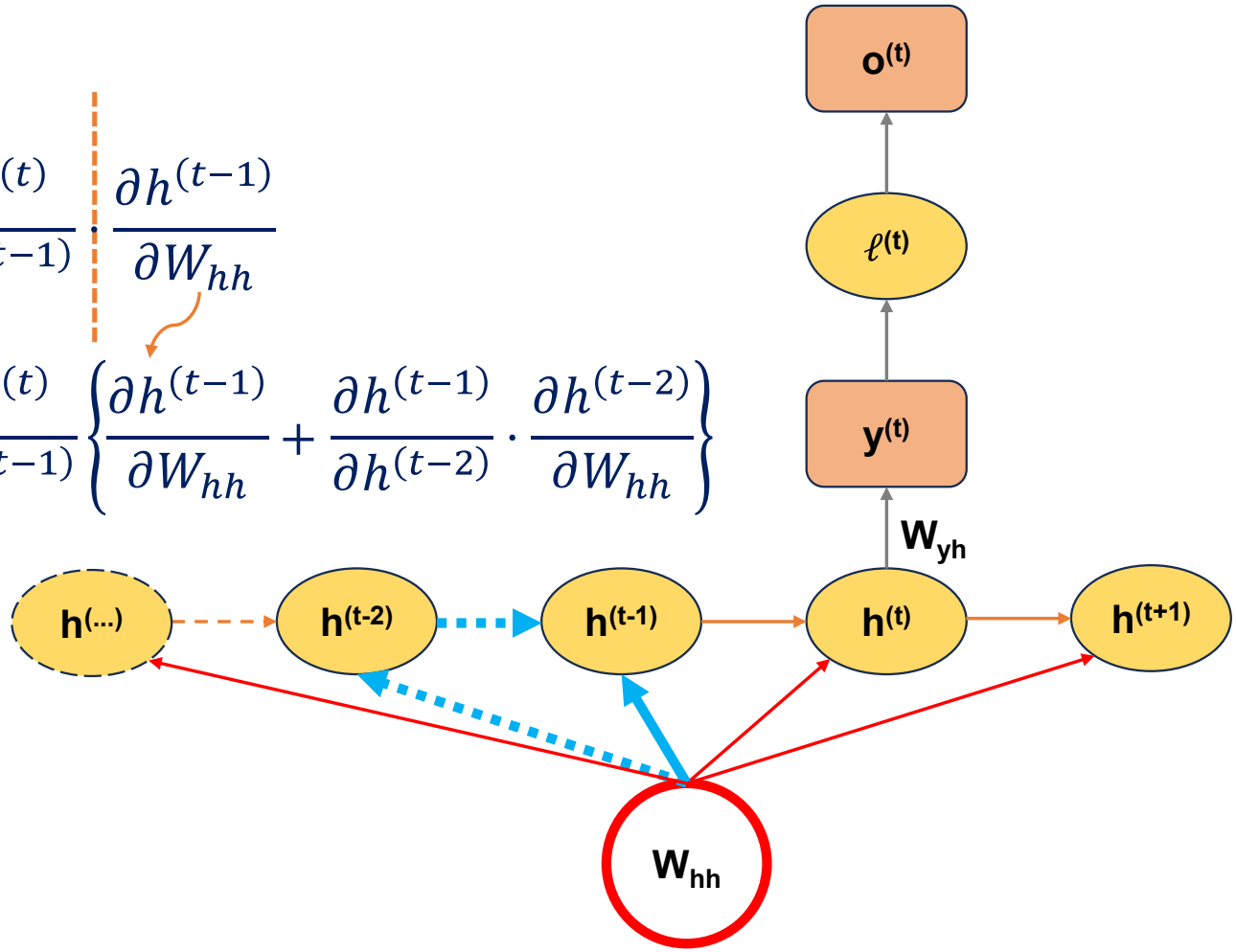
$$\frac{\partial h^{(t)}}{\partial W_{hh}} = \underbrace{\frac{\partial h^{(t)}}{\partial W_{hh}}}_{1} + \underbrace{\frac{\partial h^{(t)}}{\partial h^{(t-1)}} \cdot \frac{\partial h^{(t-1)}}{\partial W_{hh}}}_{2}$$



Back-propagation through time (BPTT)

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

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

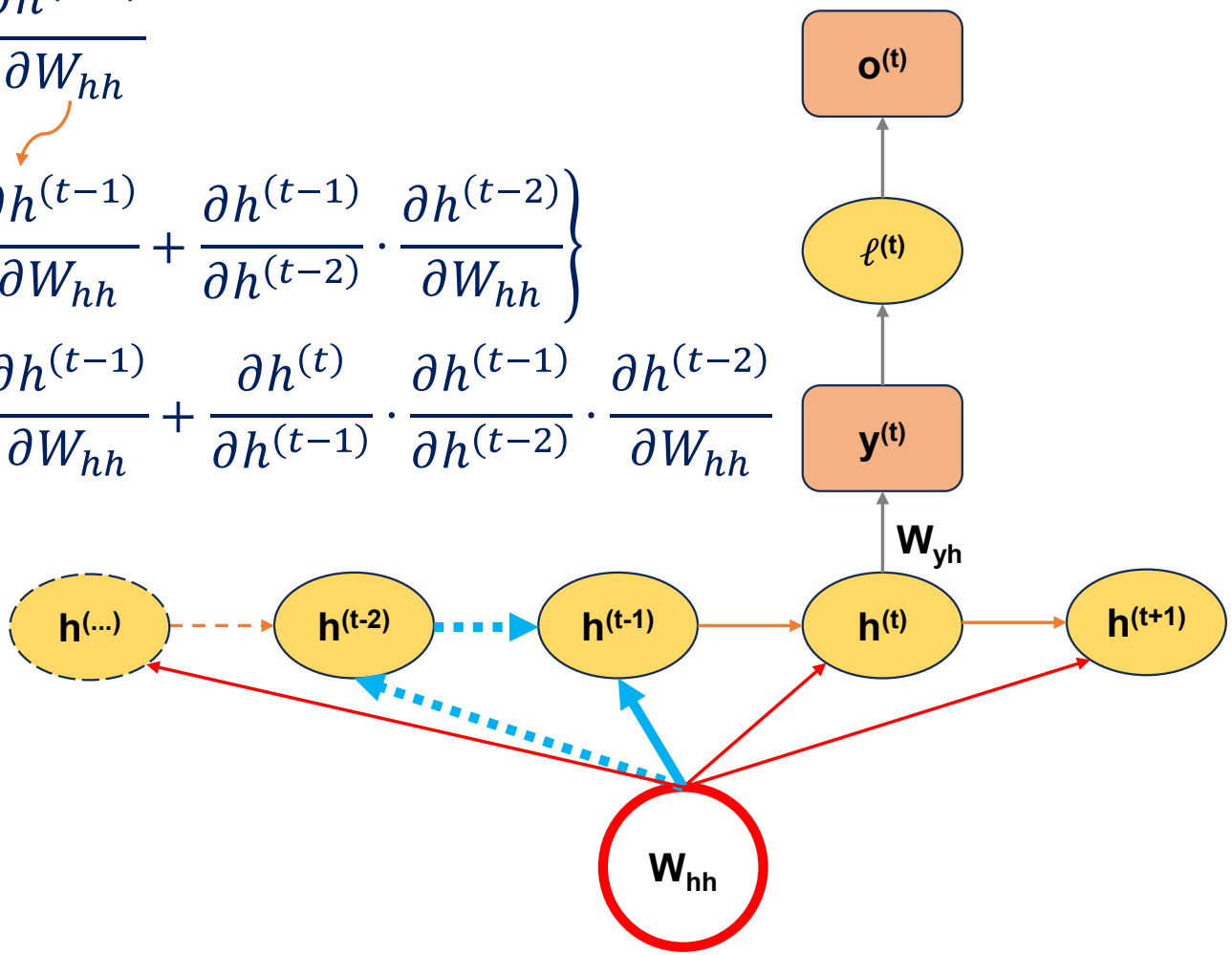


Back-propagation through time (BPTT)

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

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

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



Back-propagation through time (BPTT)

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

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

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

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

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

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

Back-propagation through time (BPTT)

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

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

Back-propagation through time (BPTT)

- Since each $\mathbf{h}_t$ depends on the previous time step we can substitute the last part from above equation

Partial derivative with respect to $\mathbf{W}_{hh}$

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

- Similarly, we have the partial derivative with respect to $\mathbf{W}_{xh}$ as below

Partial derivative with respect to $\mathbf{W}_{xh}$

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

Back-propagation through time (BPTT)

- In order to transport the error through time from timestep t back to timestep k we can have

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

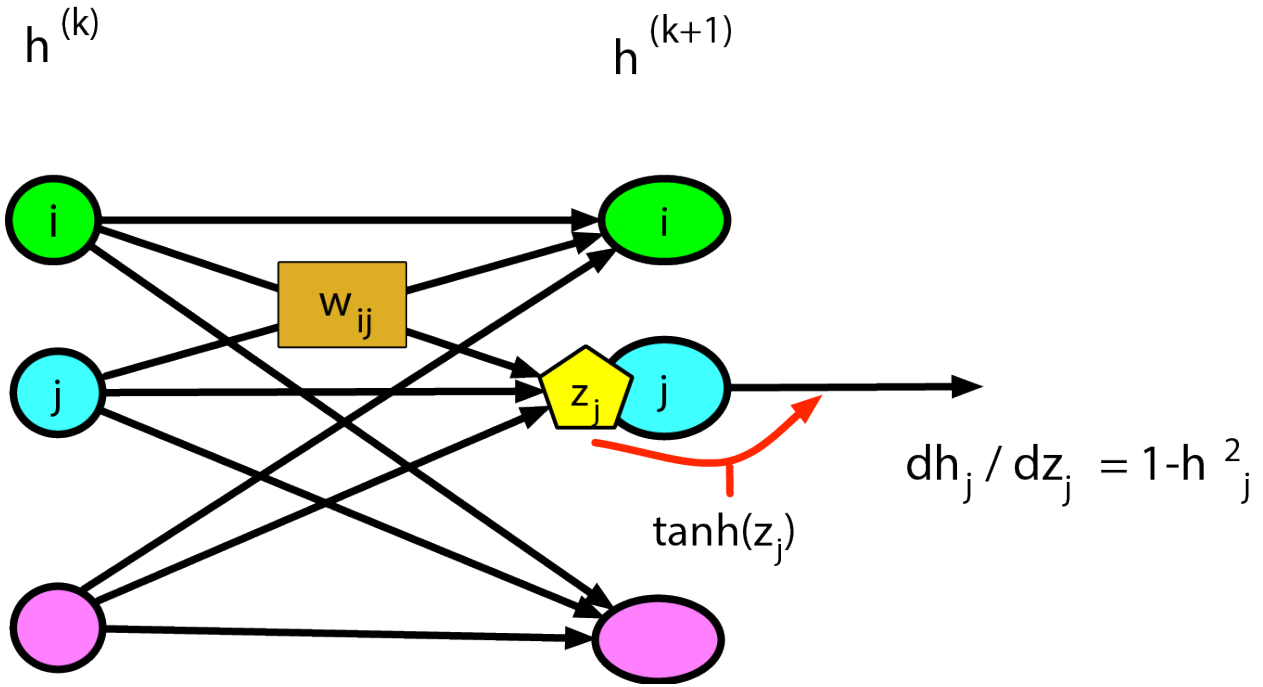
- We can consider previous equation as a Jacobian matrix for the hidden state parameter as

$$\prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}} = \prod_{i=k+1}^{t} \mathbf{W}_{hh}^{T} diag(\phi_h'(\mathbf{z}_h^{(i)}))$$

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)



- Assume activation function for $\mathbf{h}$ is $tanh$.
- $\frac{\partial tanh(x)}{\partial x} = 1 - tanh(x)^2$
- Thus:

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

Back-propagation through time (BPTT)

$$\frac{\partial h^{(k+1)}}{\partial h^{(k)}} = \begin{bmatrix} \left(1-(h_1^{(k+1)})^2\right)W_{11} & \left(1-(h_1^{(k+1)})^2\right)W_{21} & \cdots & \left(1-(h_1^{(k+1)})^2\right)W_{m1} \\ \left(1-(h_2^{(k+1)})^2\right)W_{12} & \left(1-(h_2^{(k+1)})^2\right)W_{22} & \cdots & \left(1-(h_2^{(k+1)})^2\right)W_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \left(1-(h_m^{(k+1)})^2\right)W_{1m} & \left(1-(h_m^{(k+1)})^2\right)W_{2m} & \cdots & \left(1-(h_m^{(k+1)})^2\right)W_{mm} \end{bmatrix}$$

This is just W^T with the jth row multiplied by $(1-(h_j^{(k+1)})^2$, or:

$$\frac{\partial h^{(k+1)}}{\partial h^{(k)}} = \text{diag}\left(1-(h_1^{(k+1)})^2\right) \bullet W^T$$

$$= \begin{bmatrix} \left(1-h_1^{(k+1)}\right)^2 & 0 & \cdots & 0 \\ 0 & \left(1-h_2^{(k+1)}\right)^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \bullet \begin{bmatrix} W_{11} & W_{21} & w_{31} & \cdots \\ W_{12} & W_{22} & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Back-propagation through time (BPTT)

This is very problematic: Vanishing/Exploding gradient problem!

Table of Contents

Difficulty of training RNNs

$$\| \frac{\partial h_i}{\partial h_{i-1}} \| = \| \operatorname{diag}\left(\sigma^{'}(z_i)\right) W \|$$

$$\leq \|\operatorname{diag}\left(\sigma^{'}(z_i)\right)\| \, \| W \|$$

$$\| \frac{\partial h_i}{\partial h_{i-1}} \| \leq \gamma \, \| W \|$$

$$\| \frac{\partial h_i}{\partial h_{i-1}} \| \leq \gamma\lambda$$

$$\sigma^{'}(z_i) \leq \frac{1}{4} = \gamma \quad [\text{ if } \sigma \text{ is sigmoid }]$$

$$\sigma^{'}(z_i) \leq 1 = \gamma \quad [\text{ if } \sigma \text{ is tanh }]$$

Prove the derivative of the sigmoid function

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

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

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

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

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

Gradient of Sigmoid Function

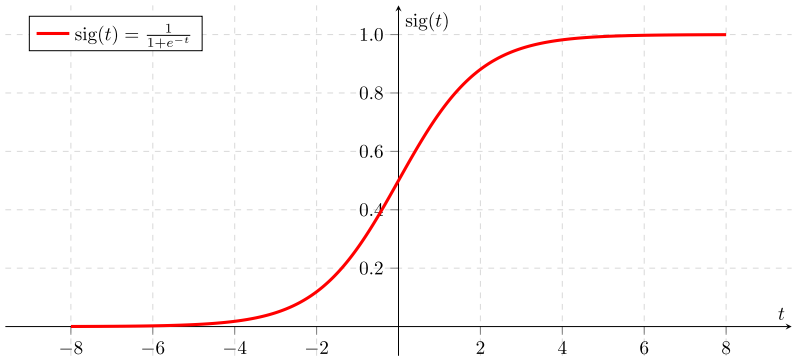


Figure: Sigmoid function

$$\sigma'(x) = \frac{d}{dx}\sigma(x) = slope\ of\ \sigma(x)\ \ at\ x$$

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

Derivative of $\sigma(x)$ with respect to x

Gradient of Sigmoid Function

$x = 10 \to g(x) \approx 1$

$$\frac{d}{dx}\sigma(x) \approx 1(1-1) = 0$$

$x = -10 \to g(x) \approx 0$

$$\frac{d}{dx}\sigma(x) \approx 1(1-1) = 0$$

$x = 0 \to g(x) = \frac{1}{2}$

$$\frac{d}{dx}\sigma(x) = 1\left(1 - \frac{1}{2}\right) = \frac{1}{4}$$

Difficulty of training RNNs

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

1. If $(\gamma\lambda) > 1$ then gradient will **explode**
2. If $(\gamma\lambda) < 1$ then gradient will **vanish**

Exploding Gradient

- Gradients in training RNNs on long sequences may explode as the weights become larger and the norm of the gradient during training largely increases.

- As it is stated before, the necessary condition for this situation to happen is $\lambda > \frac{1}{\gamma}$.

Vanishing Gradient

- This problem refers to the exponential shrinking of gradient magnitudes as they are propagated back through time.
- This phenomena causes memory of the network to ignore long term dependencies and hardly learn the correlation between temporally distant events
- There are two reasons for that:
  1. Standard nonlinear functions have a gradient which is almost everywhere close to zero;
  2. The magnitude of gradient is multiplied over and over by the recurrent matrix as it is back-propagated through time.

Conclusion for the 2 problems

Vanishing Gradient

$$\left\| \frac{\partial \mathbf{h}^{(i+i)}}{\partial \mathbf{h}^{(i)}} \right\| < 1$$

Exploding Gradient

$$\left\| \frac{\partial \mathbf{h}^{(i+1)}}{\partial \mathbf{h}^{(i)}} \right\| > 1$$

Dealing with Exploding Gradient

1. Gradient Clipping:
   To deal with the exploding gradients problem, we propose a solution that involves clipping the norm of the exploded gradients when it is too large.

$$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathscr{E}}{\partial \theta}$$
$$\text{if } \| \hat{\mathbf{g}} \| \geq \text{threshold then}$$
$$\hat{\mathbf{g}} \leftarrow \frac{\text{threshold}}{\| \hat{\mathbf{g}} \|} \hat{\mathbf{g}}$$
$$\text{end if}$$

2. Penalize or artificially reduce gradient;
3. Stop backpropagating after a certain point, which is usually not optimal because not all of the weights get updated

Dealing with Vanishing Gradient

1. Initializing weights so that the potential for vanishing gradient is minimized;
2. Having Echo State Networks that are designed to solve the vanishing gradient problem;
3. Having Long Short-Term Memory Networks (LSTMs).

Other ways for Vanishing/Exploding Gradient problems: Truncated BPTT,...

RNN's memory

- Repeated multiplication by hidden-to-hidden weights, along with the integration of new inputs, causes memories (i.e. hidden-node states) to fade over time.
- For some tasks (e.g. reinforcement learning), this is desirable: older information should have a diminishing effect upon current decision making.
- In other tasks, such as natural-language processing, some key pieces of old information need to remain influential:
  I found my bicycle in the same town and on the same street, but not on the same front porch from which I had reported to the police that it had been stolen.
- What was stolen?
  A clear memory of bicycle must be maintained throughout processing of the entire sentence (or paragraph, or story).

Long-Short Term Memory (LSTM)