Applied Deep Learning

Sequence Modeling
如何考慮序列資訊

http://adl.miulab.tw

Instructor: Yun-Nung (Vivian) Chen 陳縕儂

National Taiwan University
國立臺灣大學

Outline

- Language Modeling
  - N-gram Language Model
  - Feed-Forward Neural Language Model
  - Recurrent Neural Network Language Model (RNNLM)
- Recurrent Neural Network
  - Definition
  - Training via Backpropagation through Time (BPTT)
  - Training Issue
  - Extension
- RNN Applications
  - Sequential Input
  - Sequential Output
    - Aligned Sequential Pairs (Tagging)
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

3

Language Modeling

語言模型

Outline

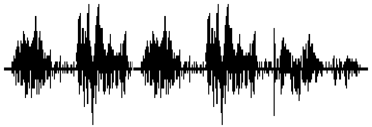
- **Language Modeling**
  - N-gram Language Model
  - Feed-Forward Neural Language Model
  - Recurrent Neural Network Language Model (RNNLM)
- Recurrent Neural Network
  - Definition
  - Training via Backpropagation through Time (BPTT)
  - Training Issue
  - Extension
- RNN Applications
  - Sequential Input
  - Sequential Output
    - Aligned Sequential Pairs (Tagging)
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

Language Modeling

- Goal: estimate the probability of a word sequence

$$P(w_1, \cdots, w_m)$$

- Example task: determinate whether a sequence is grammatical or makes more sense



recognize speech
or
wreck a nice beach

If P(recognize speech)
> P(wreck a nice beach)

Output = "recognize speech"

Outline

- Language Modeling
  - **N-gram Language Model**
  - Feed-Forward Neural Language Model
  - Recurrent Neural Network Language Model (RNNLM)
- Recurrent Neural Network
  - Definition
  - Training via Backpropagation through Time (BPTT)
  - Training Issue
  - Extension
- RNN Applications
  - Sequential Input
  - Sequential Output
    - Aligned Sequential Pairs (Tagging)
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

N-Gram Language Modeling

- Goal: estimate the probability of a word sequence

$$P(w_1, \cdots, w_m)$$

- N-gram language model
  - Probability is conditioned on a window of (*n*-1) previous words

$$P(w_1, \cdots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \cdots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \cdots, w_{i-1})$$

  - Estimate the probability based on the training data

$$P(\text{beach}|\text{nice}) = \frac{C(\text{nice each})}{C(\text{nice})}$$

Count of “nice beach” in the training data

Count of “nice” in the training data

Issue: some sequences may not appear in the training data

N-Gram Language Modeling

- Training data:
  - The dog ran ……
  - The cat jumped ……

$$P(\text{ jumped} \mid \text{dog }) = \cancel{0}\ \ 0.0001$$

$$P(\text{ ran} \mid \text{cat }) = \cancel{0}\ \ 0.0001$$

give some small probability
→ smoothing

- The probability is not accurate
- Reason: impossible to collect all possible texts as training data

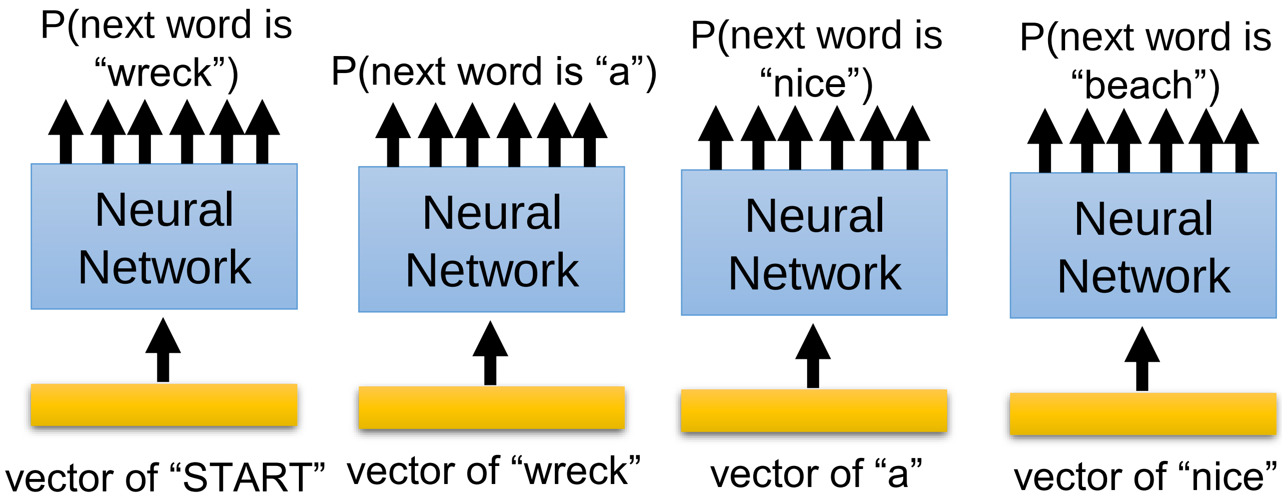
Outline

- Language Modeling
  - N-gram Language Model
  - **Feed-Forward Neural Language Model**
  - Recurrent Neural Network Language Model (RNNLM)
- Recurrent Neural Network
  - Definition
  - Training via Backpropagation through Time (BPTT)
  - Training Issue
  - Extension
- RNN Applications
  - Sequential Input
  - Sequential Output
    - Aligned Sequential Pairs (Tagging)
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

Neural Language Modeling

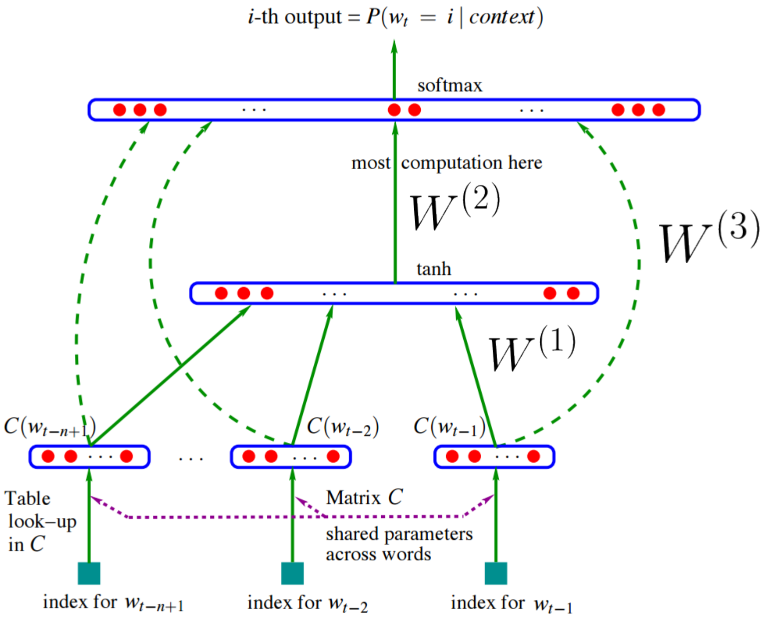
- Idea: estimate $P(w_i \mid w_{i-(n-1)}, \cdots, w_{i-1})$ not from count, but from NN prediction

P("wreck a nice beach") = P(wreck | START) P(a | wreck) P(nice | a) P(beach | nice)



Neural Language Modeling

$$\hat{y} = \mathrm{softmax}(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + W^{(3)}x + b^{(3)})$$



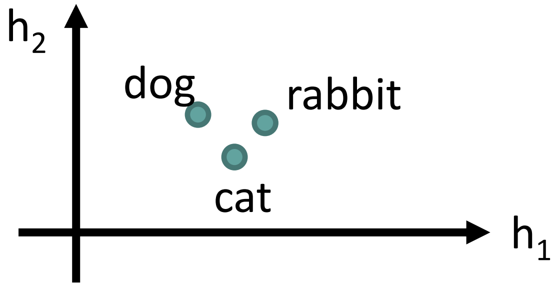
Probability distribution of the next word

context vector

Bengio et al., "A Neural Probabilistic Language Model," in *JMLR*, 2003.

Neural Language Modeling

- The input layer (or hidden layer) of the related words are close



  - If P(jump | cat) is large, P(jump | dog) increases accordingly (even there is not "… dog jumps …" in the data)

Smoothing is automatically done

Issue: fixed context window for conditioning

Outline

- Language Modeling
  - N-gram Language Model
  - Feed-Forward Neural Language Model
  - **Recurrent Neural Network Language Model (RNNLM)**
- Recurrent Neural Network
  - Definition
  - Training via Backpropagation through Time (BPTT)
  - Training Issue
  - Extension
- RNN Applications
  - Sequential Input
  - Sequential Output
    - Aligned Sequential Pairs (Tagging)
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

Recurrent Neural Network

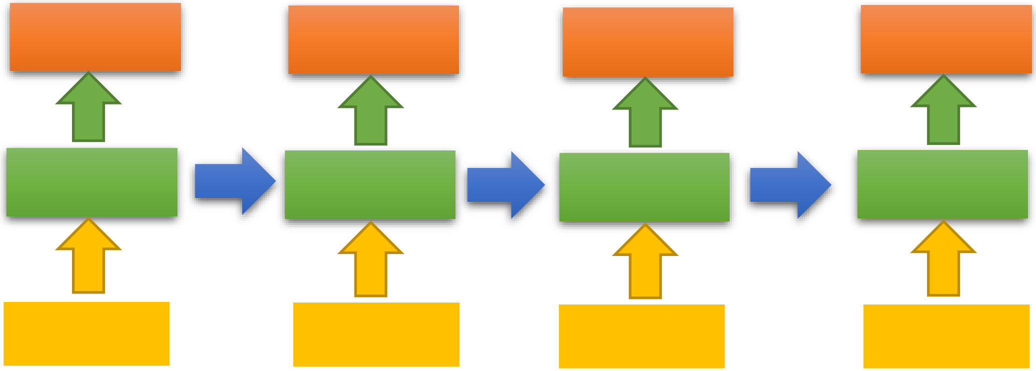
- Idea: condition the neural network on all previous words and tie the weights at each time step
- Assumption: temporal information matters

RNN Language Modeling

word prob dist

context vector

P(next word is "wreck") P(next word is "a") P(next word is "nice") P(next word is "beach")



vector of "START" vector of "wreck" vector of "a" vector of "nice"

Idea: pass the information from the previous hidden layer to leverage all contexts

16

Recurrent Neural Network

詳細解析鼎鼎大名的RNN

Outline

- Language Modeling
  - N-gram Language Model
  - Feed-Forward Neural Language Model
  - Recurrent Neural Network Language Model (RNNLM)
- **Recurrent Neural Network**
  - Definition
  - Training via Backpropagation through Time (BPTT)
  - Training Issue
  - Extension
- RNN Applications
  - Sequential Input
  - Sequential Output
    - Aligned Sequential Pairs (Tagging)
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

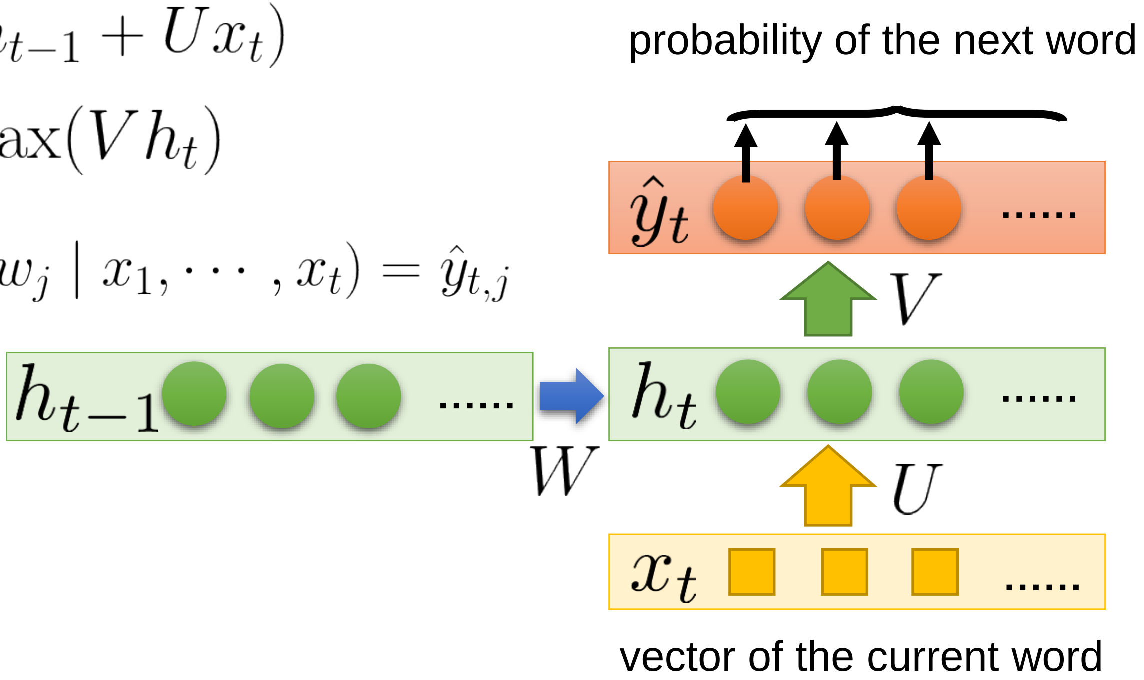
RNNLM Formulation

- At each time step,

$$h_t = \sigma(W h_{t-1} + U x_t)$$

$$\hat{y}_t = \mathrm{softmax}(V h_t)$$

$$P(x_{t+1} = w_j \mid x_1, \cdots, x_t) = \hat{y}_{t,j}$$



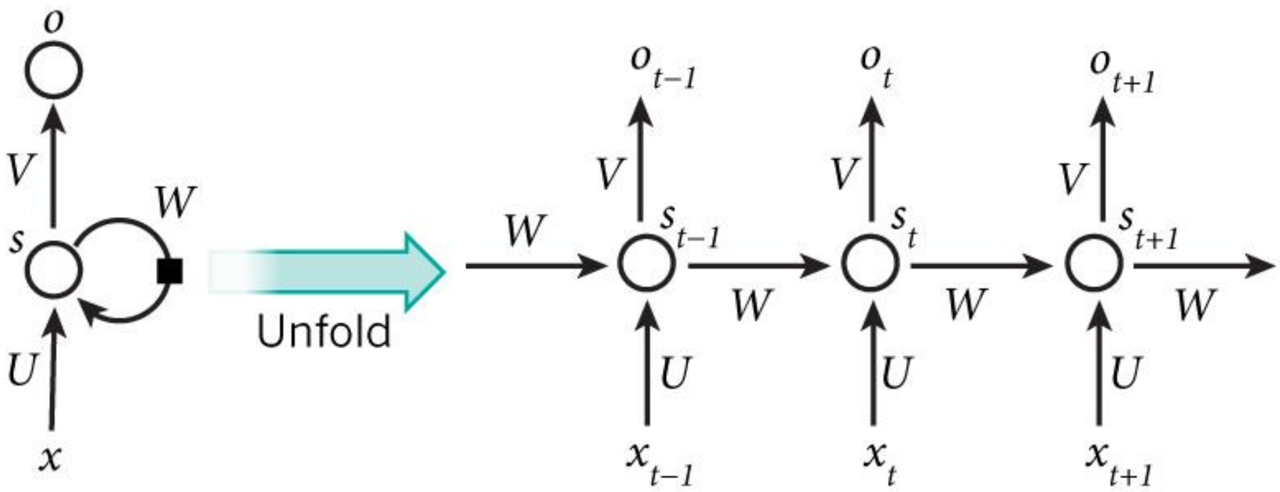
Outline

- Language Modeling
  - N-gram Language Model
  - Feed-Forward Neural Language Model
  - Recurrent Neural Network Language Model (RNNLM)
- Recurrent Neural Network
  - **Definition**
  - Training via Backpropagation through Time (BPTT)
  - Training Issue
  - Extension
- RNN Applications
  - Sequential Input
  - Sequential Output
    - Aligned Sequential Pairs (Tagging)
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

Recurrent Neural Network Definition

$$s_t = \sigma(W s_{t-1} + U x_t)$$
$$o_t = \mathrm{softmax}(V s_t)$$

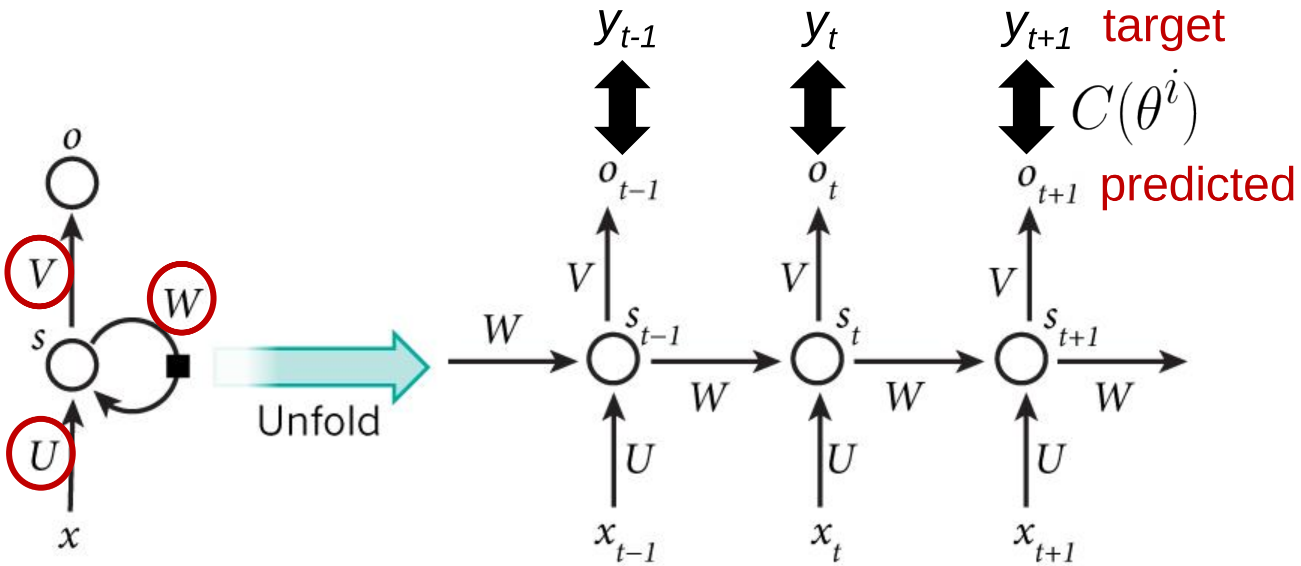
$\sigma(\cdot)$: tanh, ReLU



Model Training

- All model parameters $\theta = \{U, V, W\}$ can be updated by

$$\theta^{i+1} \leftarrow \theta^i - \eta \nabla_\theta C(\theta^i)$$

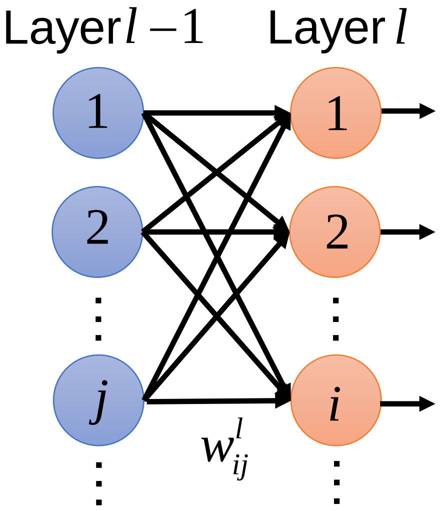


Outline

- Language Modeling
  - N-gram Language Model
  - Feed-Forward Neural Language Model
  - Recurrent Neural Network Language Model (RNNLM)
- Recurrent Neural Network
  - Definition
  - **Training via Backpropagation through Time (BPTT)**
  - Training Issue
  - Extension
- RNN Applications
  - Sequential Input
  - Sequential Output
    - Aligned Sequential Pairs (Tagging)
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

Backpropagation

$$\frac{\partial C(\theta)}{\partial w_{ij}^l} = \boxed{\frac{\partial C(\theta)}{\partial z_i^l}}\,\boxed{\frac{\partial z_i^l}{\partial w_{ij}^l}}$$



$\boxed{\delta_i^l}$ Error signal

$$\boxed{\begin{cases} a_j^{l-1} & l > 1 \\ x_j & l = 1 \end{cases}}$$

Backward Pass

$$\delta^L = \sigma'(z^L) \odot \nabla C(y)$$
$$\delta^{L-1} = \sigma'(z^{L-1}) \odot (W^L)^T \delta^L$$
$$\vdots$$
$$\delta^l = \sigma'(z^l) \odot (W^{l+1})^T \delta^{l+1}$$
$$\vdots$$

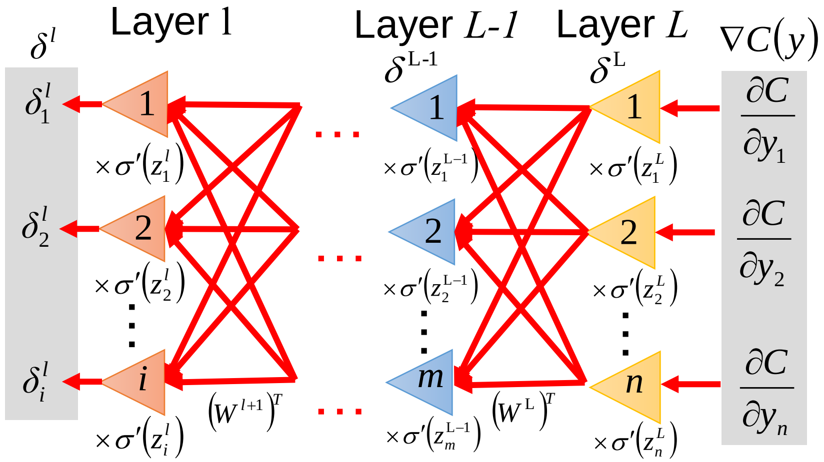
Forward Pass

$$z^1 = W^1 x + b^1$$
$$a^1 = \sigma(z^1)$$
$$\vdots$$
$$z^l = W^l a^{l-1} + b^l$$
$$a^l = \sigma(z^l)$$
$$\vdots$$

Backpropagation

$$\frac{\partial C(\theta)}{\partial w_{ij}^l} = \boxed{\frac{\partial C(\theta)}{\partial z_i^l}} \frac{\partial z_i^l}{\partial w_{ij}^l}$$

$\boxed{\delta_i^l}$ Error signal

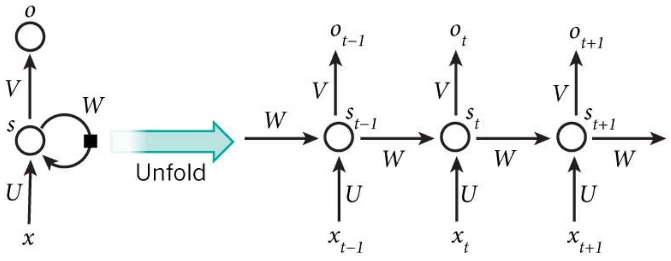


Backward Pass

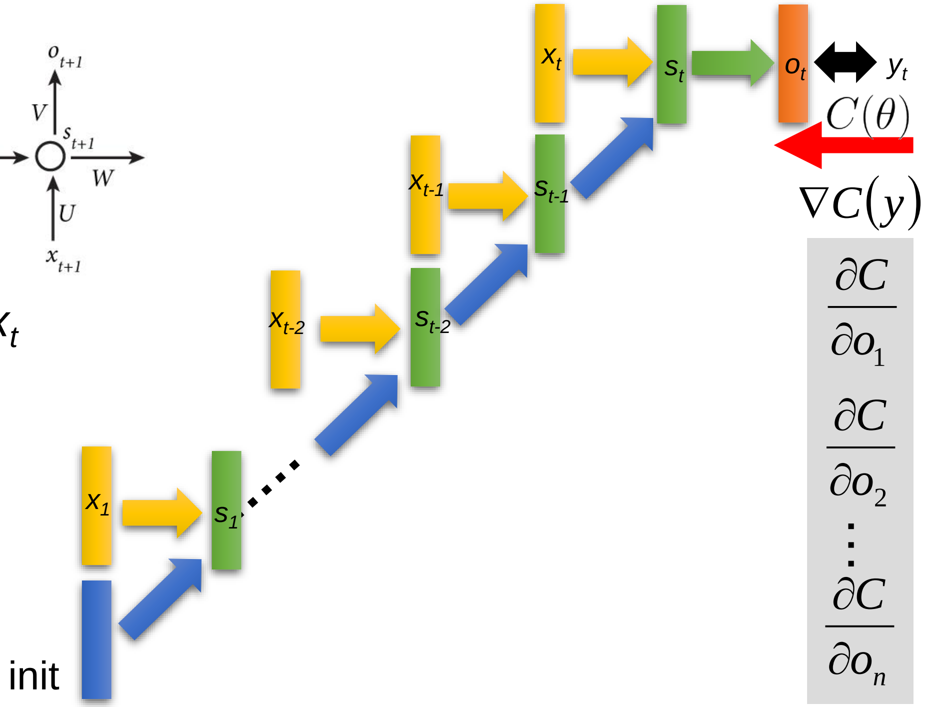
$$\delta^L = \sigma'(z^L) \odot \nabla C(y)$$
$$\delta^{L-1} = \sigma'(z^{L-1}) \odot (W^L)^T \delta^L$$
$$\vdots$$
$$\delta^l = \sigma'(z^l) \odot (W^{l+1})^T \delta^{l+1}$$
$$\vdots$$

Backpropagation through Time (BPTT)

- Unfold

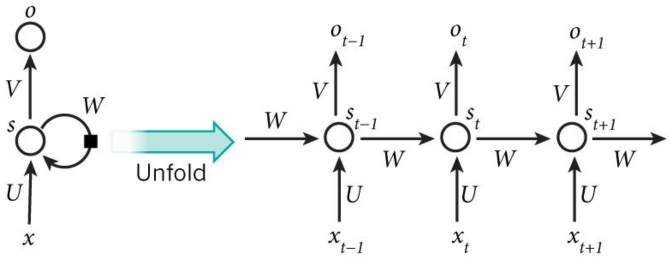


  - Input: init, x_1, x_2, …, x_t
  - Output: o_t
  - Target: y_t

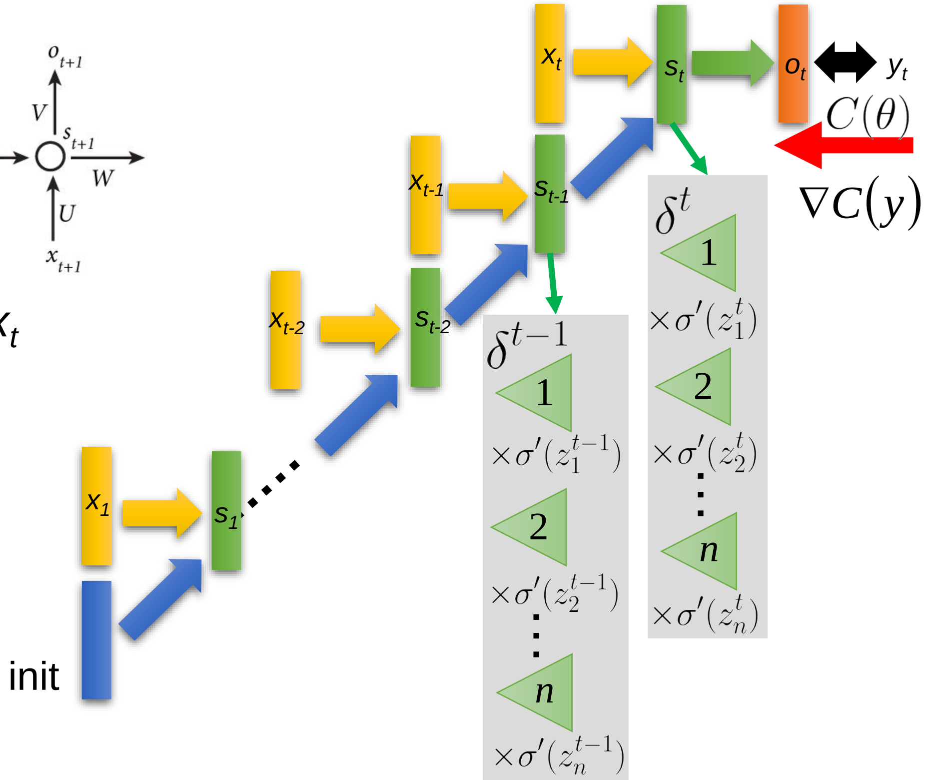


Backpropagation through Time (BPTT)

- Unfold

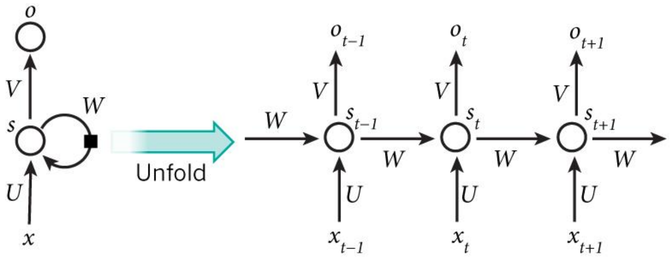


  - Input: init, x_1, x_2, …, x_t
  - Output: o_t
  - Target: y_t

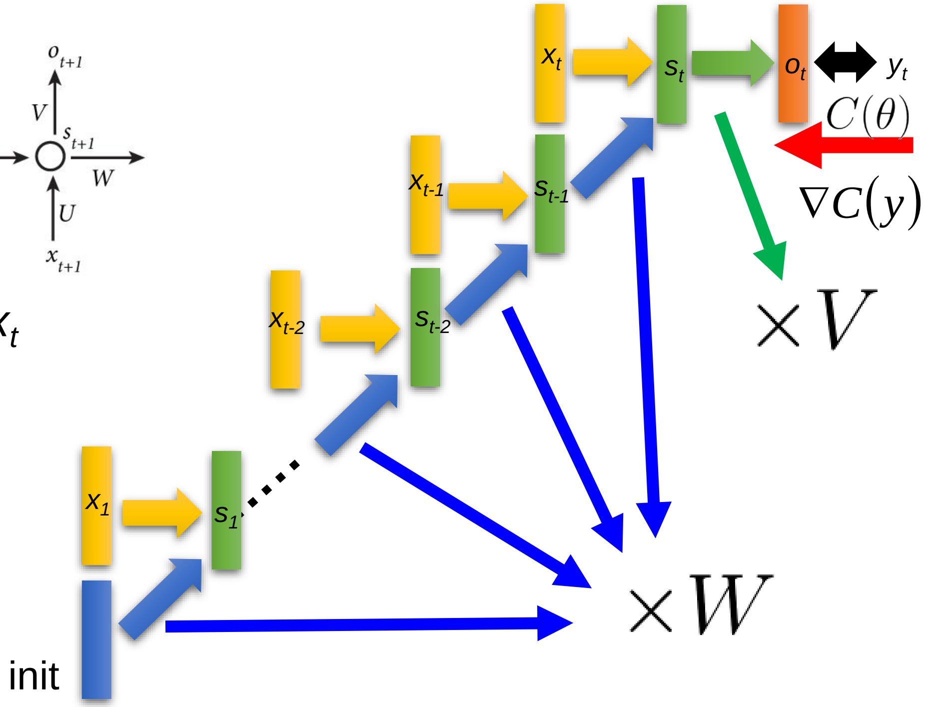


Backpropagation through Time (BPTT)

- Unfold

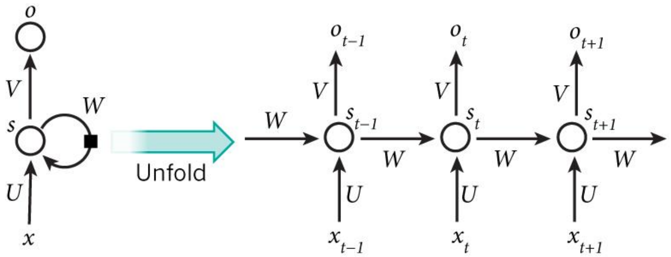


  - Input: init, x_1, x_2, …, x_t
  - Output: o_t
  - Target: y_t

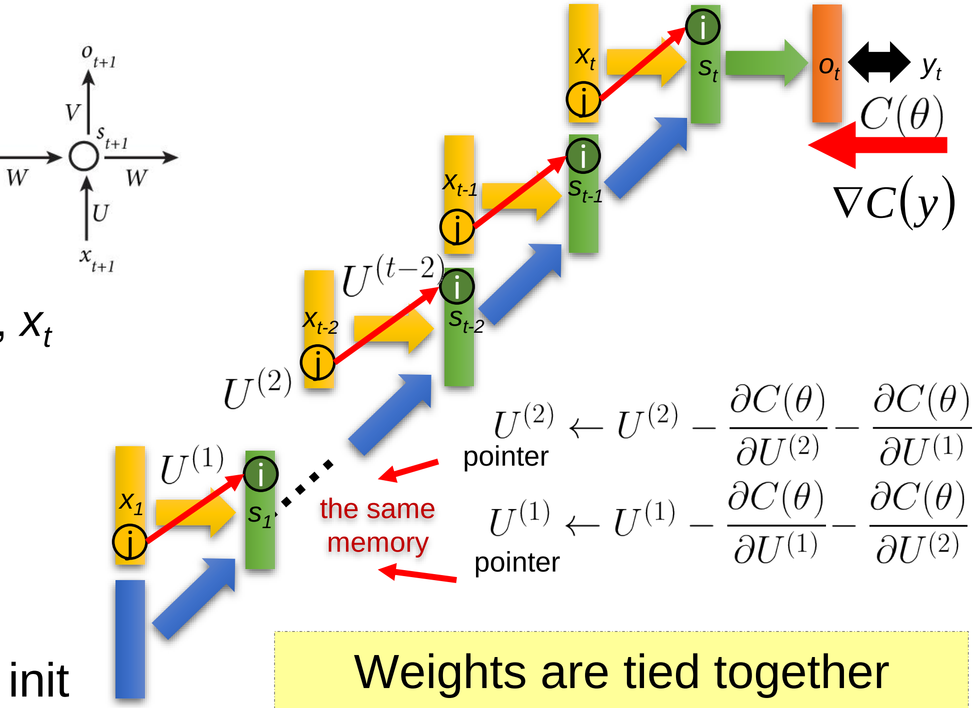


Backpropagation through Time (BPTT)

- Unfold



  - Input: init, x_1, x_2, …, x_t
  - Output: o_t
  - Target: y_t



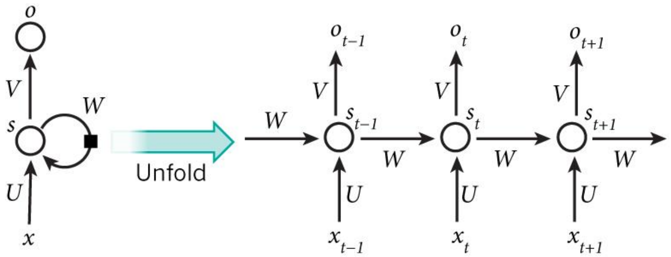
$$U^{(2)} \leftarrow U^{(2)} - \frac{\partial C(\theta)}{\partial U^{(2)}} - \frac{\partial C(\theta)}{\partial U^{(1)}}$$

$$U^{(1)} \leftarrow U^{(1)} - \frac{\partial C(\theta)}{\partial U^{(1)}} - \frac{\partial C(\theta)}{\partial U^{(2)}}$$

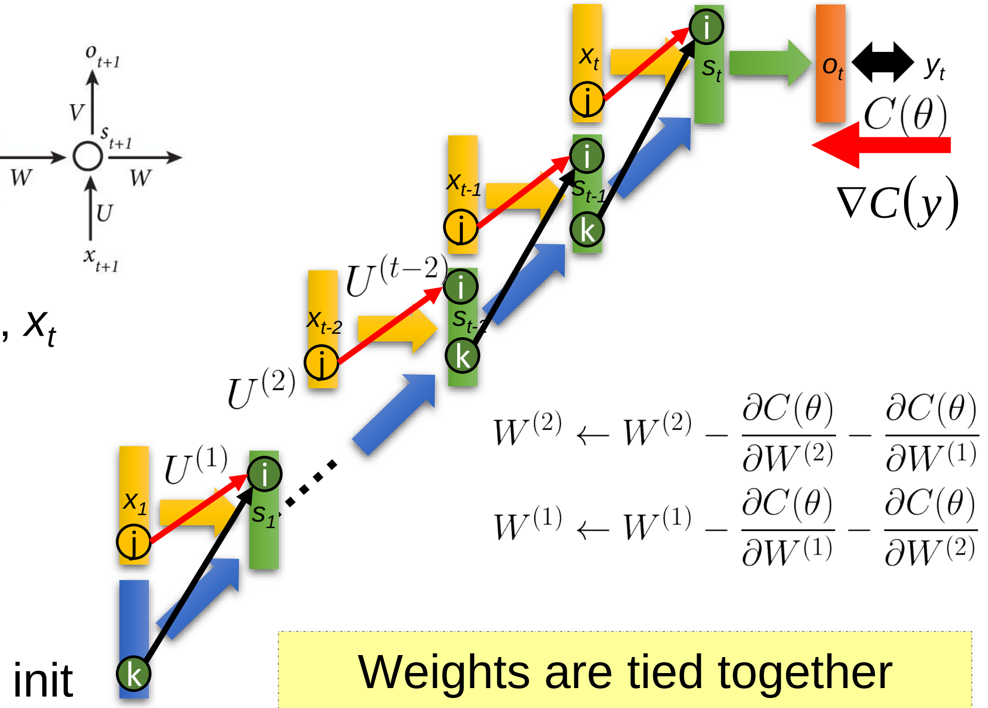
Weights are tied together

Backpropagation through Time (BPTT)

- Unfold



  - Input: init, x_1, x_2, …, x_t
  - Output: o_t
  - Target: y_t



$$W^{(2)} \leftarrow W^{(2)} - \frac{\partial C(\theta)}{\partial W^{(2)}} - \frac{\partial C(\theta)}{\partial W^{(1)}}$$

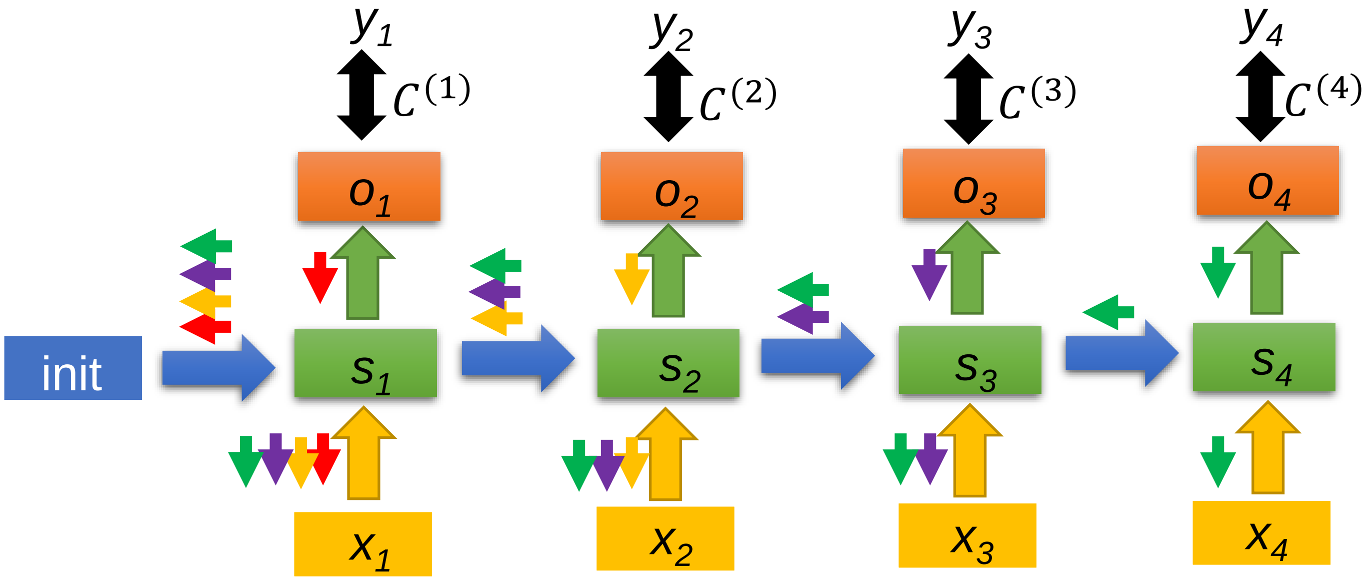
$$W^{(1)} \leftarrow W^{(1)} - \frac{\partial C(\theta)}{\partial W^{(1)}} - \frac{\partial C(\theta)}{\partial W^{(2)}}$$

Weights are tied together

BPTT

Forward Pass: Compute s_1, s_2, s_3, s_4 ……

Backward Pass: For $C^{(4)}$ For $C^{(3)}$ For $C^{(2)}$ For $C^{(1)}$



Outline

- Language Modeling
  - N-gram Language Model
  - Feed-Forward Neural Language Model
  - Recurrent Neural Network Language Model (RNNLM)
- Recurrent Neural Network
  - Definition
  - Training via Backpropagation through Time (BPTT)
  - **Training Issue**
  - Extension
- RNN Applications
  - Sequential Input
  - Sequential Output
    - Aligned Sequential Pairs (Tagging)
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

RNN Training Issue

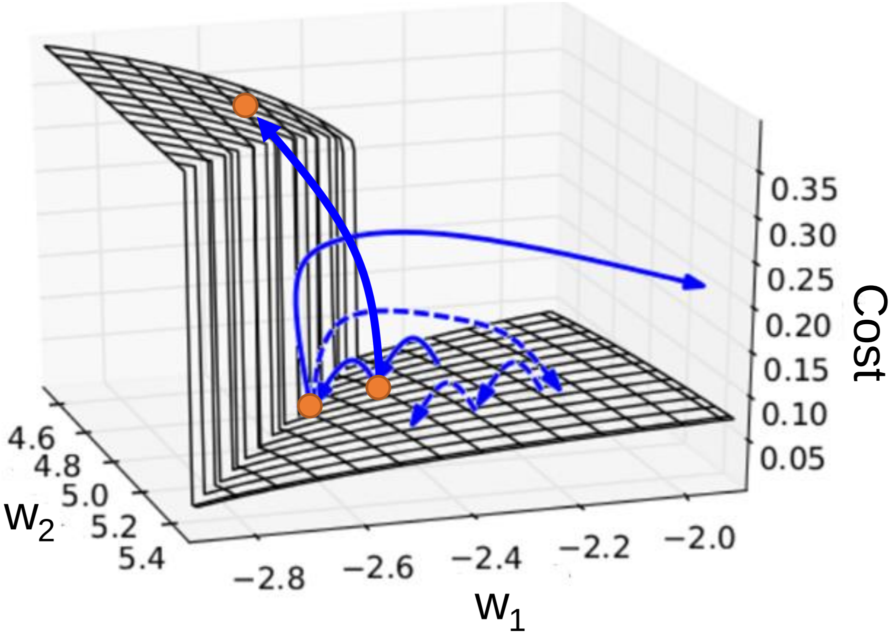
- The gradient is a product of Jacobian matrices, each associated with a step in the forward computation
- Multiply the <u>same</u> matrix at each time step during backprop

$$\delta^l = \sigma'(z^l) \odot \boxed{(W^{l+1})^T} \delta^{l+1}$$

The gradient becomes very small or very large quickly
→ **vanishing or exploding gradient**

Bengio et al., “Learning long-term dependencies with gradient descent is difficult,” *IEEE Trans. of Neural Networks*, 1994. [link]
Pascanu et al., “On the difficulty of training recurrent neural networks,” in *ICML*, 2013. [link]

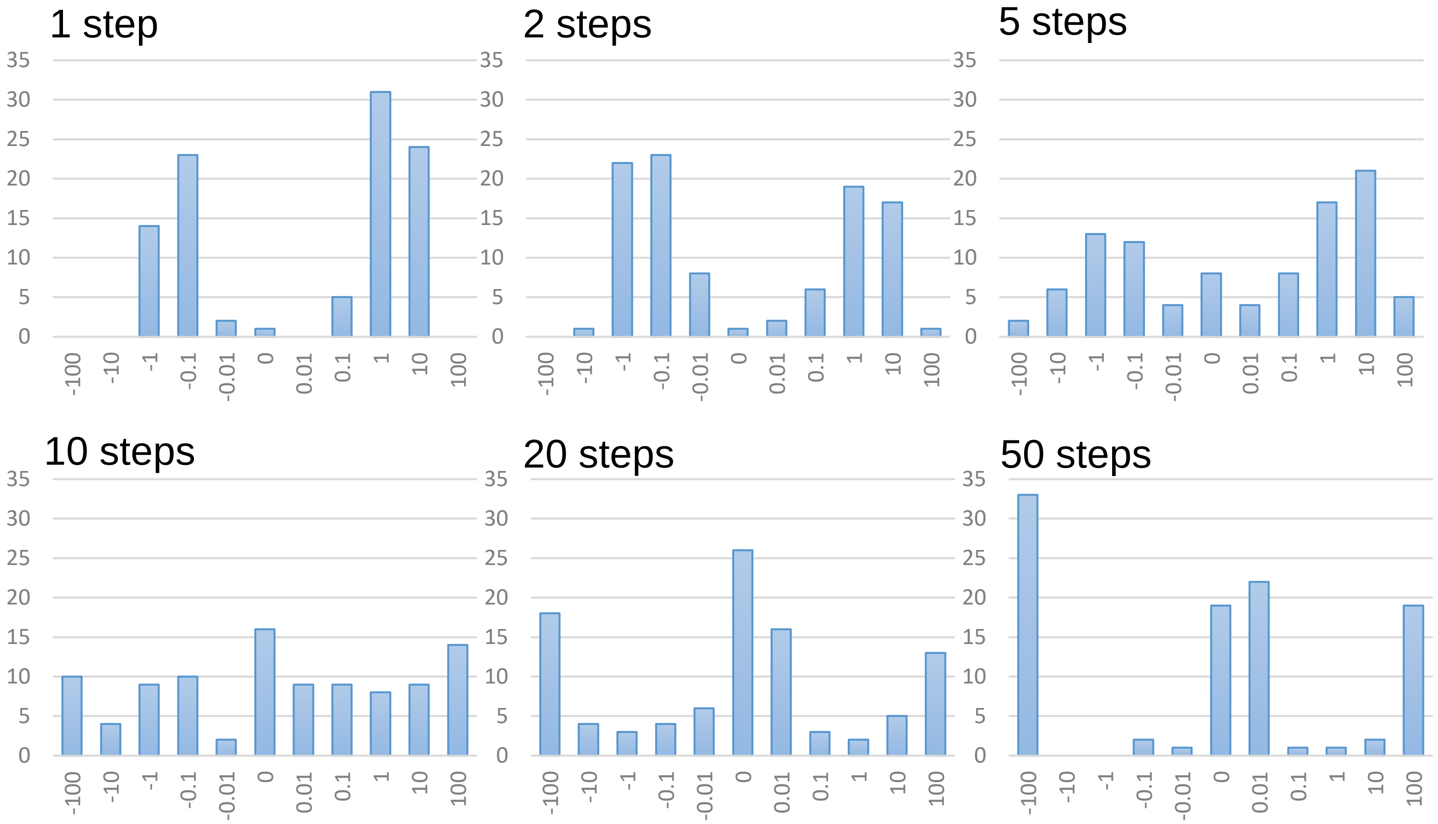
Rough Error Surface



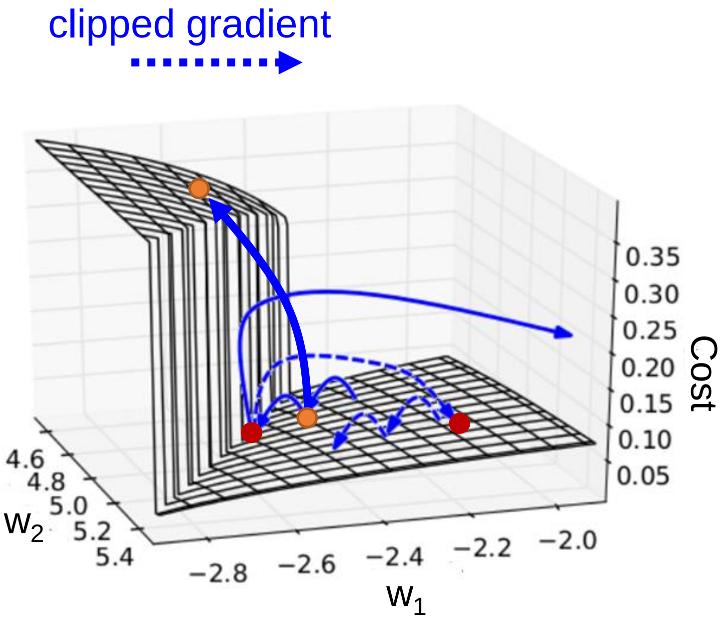
The error surface is either very flat or very steep

Bengio et al., "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. of Neural Networks*, 1994. [link]
Pascanu et al., "On the difficulty of training recurrent neural networks," in *ICML*, 2013. [link]

Vanishing/Exploding Gradient Example



Solution for Exploding Gradient: Clipping



Idea: control the gradient value to avoid exploding

Algorithm 1 Pseudo-code for norm clipping

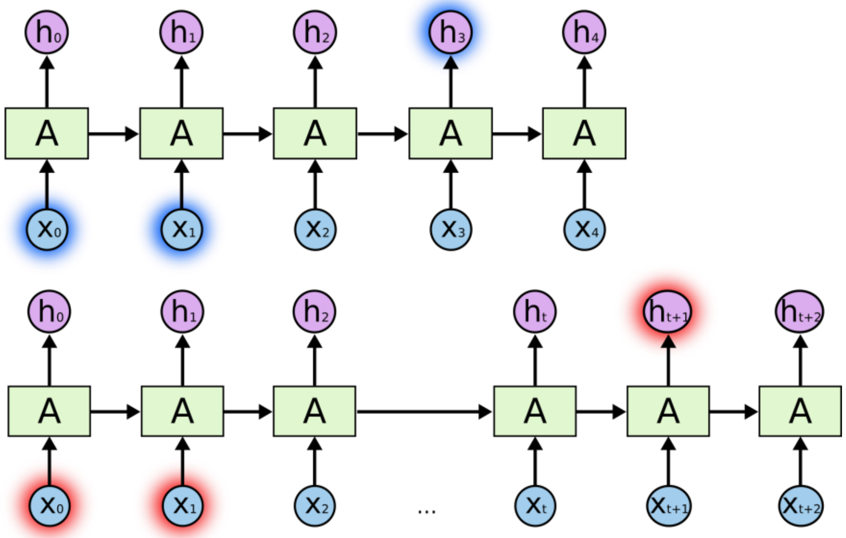
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

Parameter setting: values from half to ten times the average can still yield convergence

Pascanu et al., "On the difficulty of training recurrent neural networks," in *ICML*, 2013. [link]

Solution for Vanishing Gradient: Gating

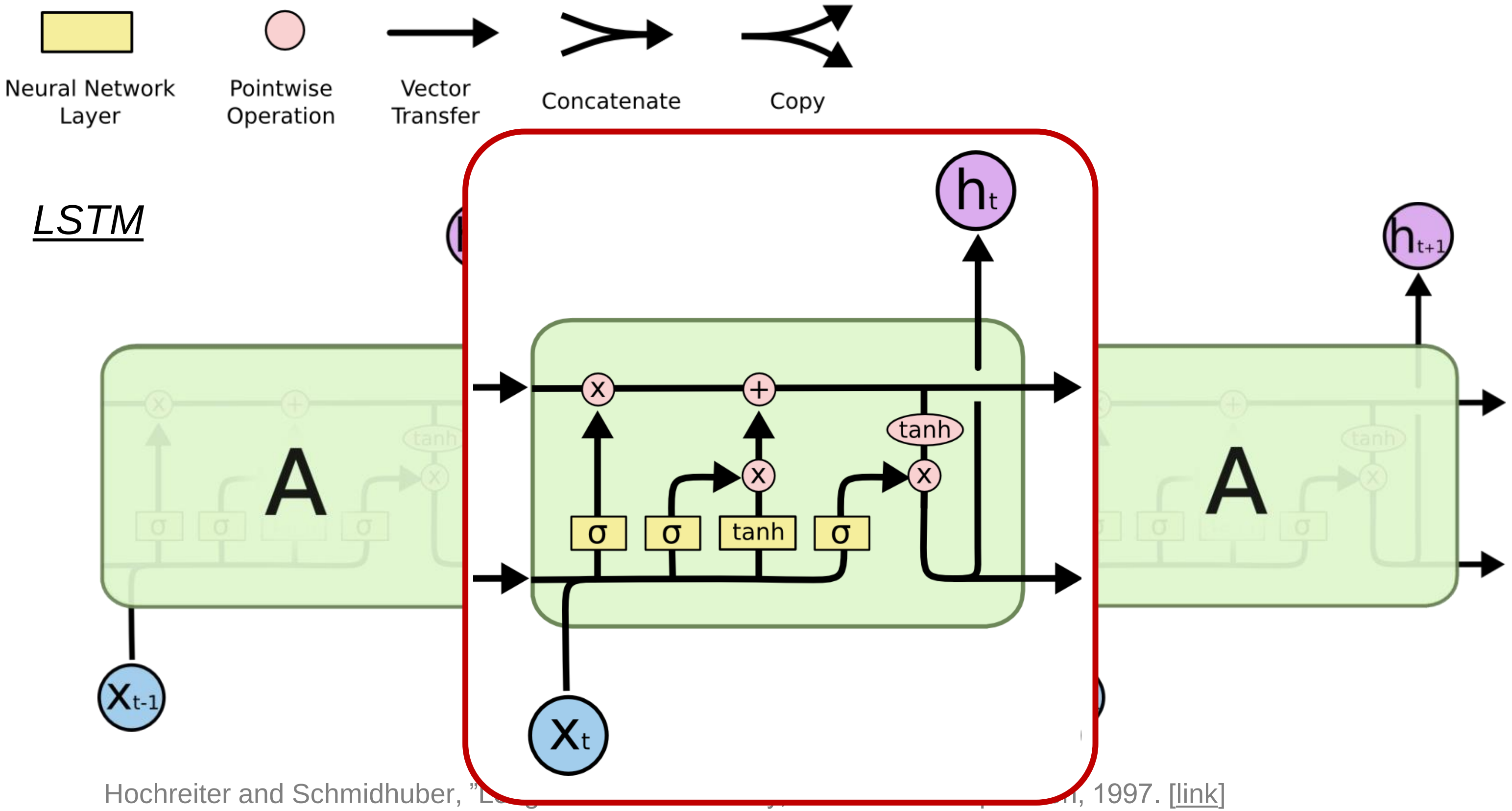
- RNN models temporal sequence information
  - can handle "long-term dependencies" *in theory*



"I grew up in France…
I speak fluent *French*."

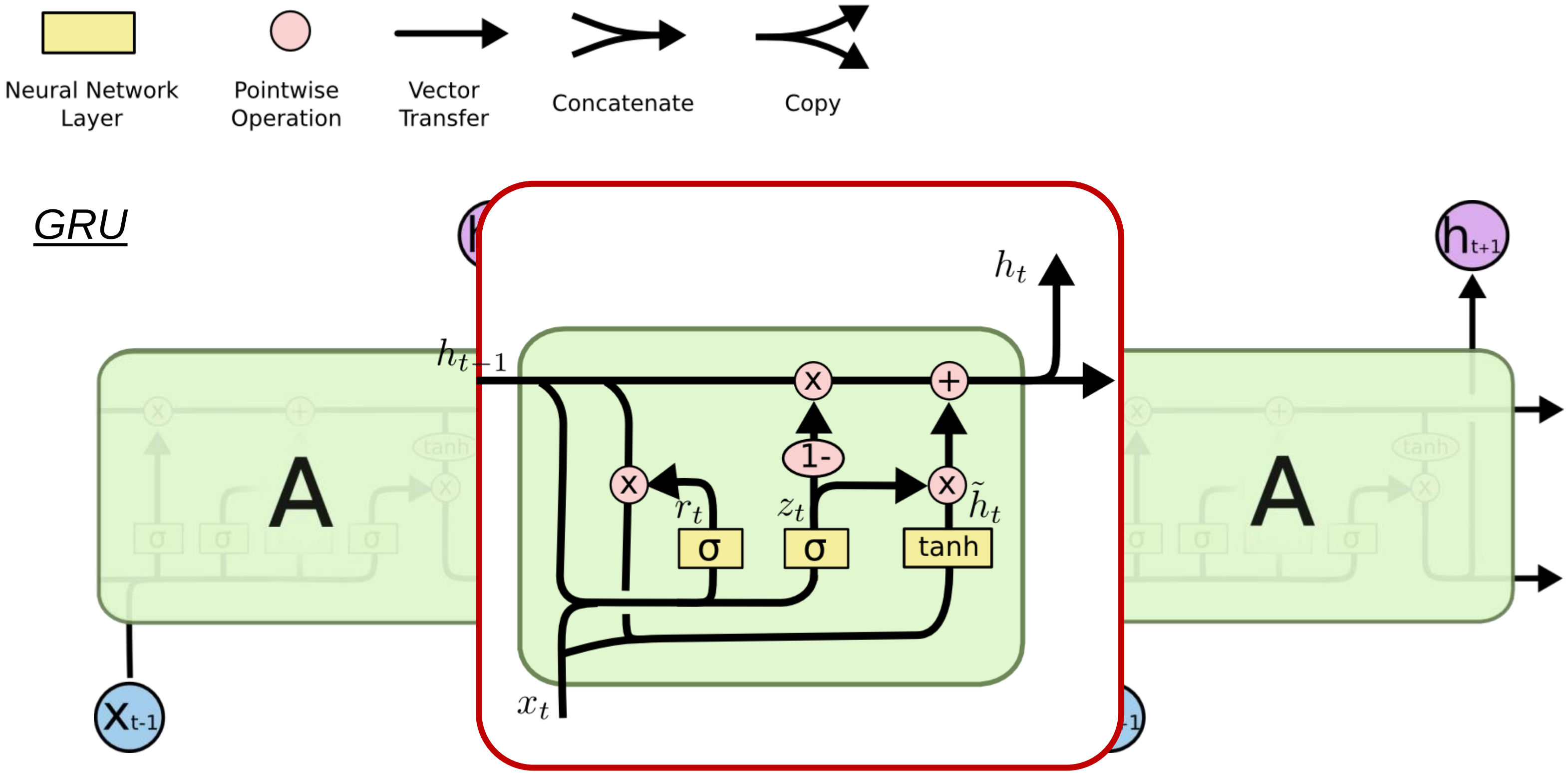
Issue: RNN cannot handle "long-term dependencies" due to vanishing gradient
→ gating directly encodes long-distance information

Long Short-Term Memory (LSTM)



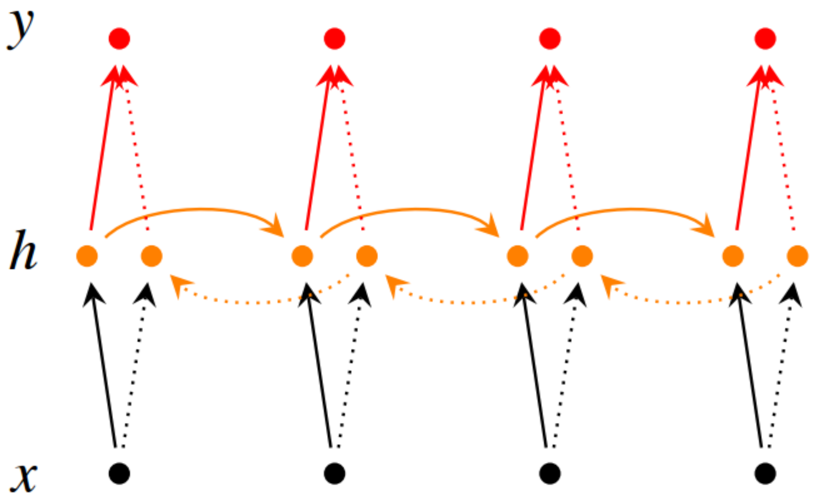
Hochreiter and Schmidhuber, ”Lo… …, 1997. [link]

Gated Recurrent Unit (GRU)



Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014. [link]

Extension: Bidirectional RNN



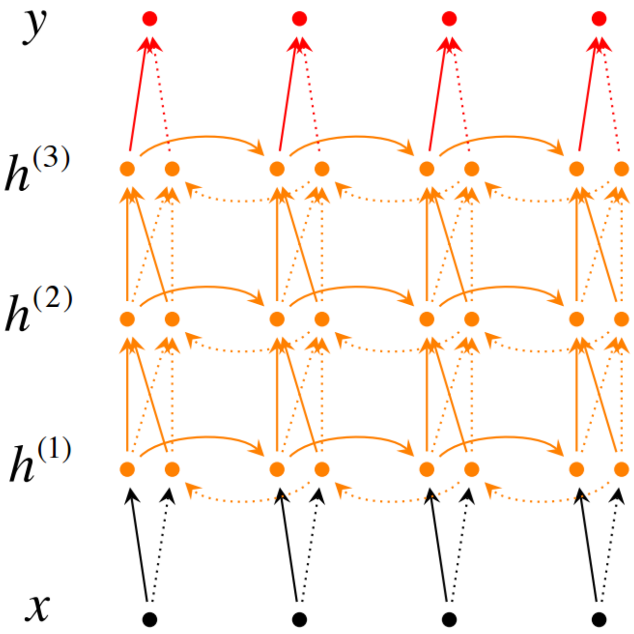
$$\overrightarrow{h}_t = f(\overrightarrow{W}x_t + \overrightarrow{V}\overrightarrow{h}_{t-1} + \overrightarrow{b})$$

$$\overleftarrow{h}_t = f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

$$y_t = g(U[\overrightarrow{h}_t;\overleftarrow{h}_t] + c)$$

$h = [\overrightarrow{h}; \overleftarrow{h}]$ represents (summarizes) the past and future around a single token

Extension: Deep Bidirectional RNN



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

Each memory layer passes an intermediate representation to the next

41

RNN Applications
RNN各式應用情境

Outline

- Language Modeling
  - N-gram Language Model
  - Feed-Forward Neural Language Model
  - Recurrent Neural Network Language Model (RNNLM)
- Recurrent Neural Network
  - Definition
  - Training via Backpropagation through Time (BPTT)
  - Training Issue
  - Extension
- **RNN Applications**
  - Sequential Input
  - Sequential Output
    - Aligned Sequential Pairs (Tagging)
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

How to Frame the Learning Problem?

- The learning algorithm f is to map the input domain X into the output domain Y

$$f : X \to Y$$

- Input domain: word, word sequence, audio signal, click logs
- Output domain: single label, sequence tags, tree structure, probability distribution

Network design should leverage input and output domain properties

Outline

- Language Modeling
  - N-gram Language Model
  - Feed-Forward Neural Language Model
  - Recurrent Neural Network Language Model (RNNLM)
- Recurrent Neural Network
  - Definition
  - Training via Backpropagation through Time (BPTT)
  - Training Issue
- Applications
  - **Sequential Input**
  - Sequential Output
    - Aligned Sequential Pairs (Tagging)
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

Input Domain – Sequence Modeling

- Idea: aggregate the meaning from all words into a vector
- Method:
  - Basic combination: average, sum
  - Neural combination:
    - ✓ Recursive neural network (RvNN)
    - ✓ Recurrent neural network (RNN)
    - ✓ Convolutional neural network (CNN)
    - ✓ Transformer

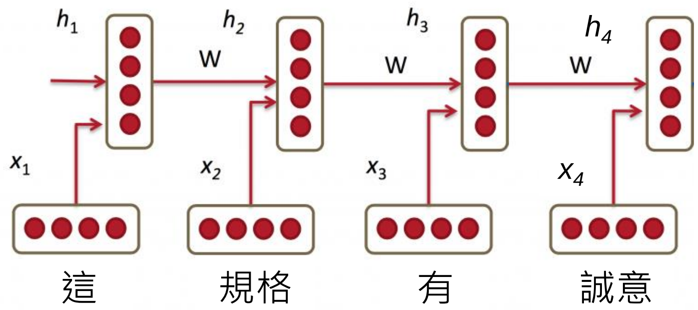
$$\begin{array}{cc}
 & \overbrace{\hphantom{0.2\ \ 0.6\ \ 0.3\ \ \cdots\ \ 0.4}}^{N\text{-dim}} \\
\text{這 (this)} & \begin{bmatrix} 0.2 & 0.6 & 0.3 & \cdots & 0.4 \end{bmatrix} \\
\text{規格 (specification)} & \begin{bmatrix} 0.9 & 0.8 & 0.1 & \cdots & 0.1 \end{bmatrix} \\
\text{有 (have)} & \begin{bmatrix} 0.1 & 0.3 & 0.1 & \cdots & 0.7 \end{bmatrix} \\
\text{誠意 (sincerity)} & \begin{bmatrix} 0.5 & 0.0 & 0.6 & \cdots & 0.4 \end{bmatrix}
\end{array}$$

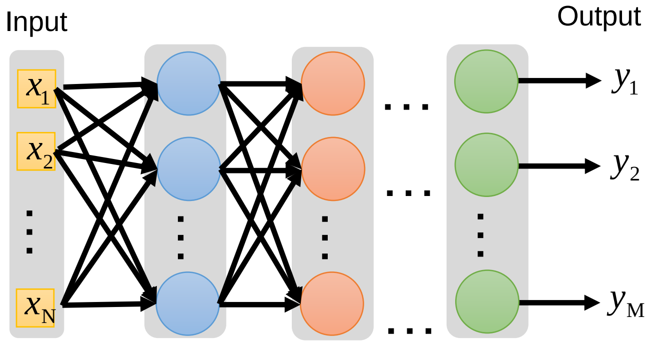
How to compute $\vec{x} = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_N \end{bmatrix}$

Sentiment Analysis

- Encode the sequential input into a vector using RNN

$$\vec{x} = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_N \end{bmatrix}$$





RNN considers temporal information to learn sentence vectors as classifier’s input

Outline

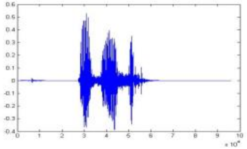
- Language Modeling
  - N-gram Language Model
  - Feed-Forward Neural Language Model
  - Recurrent Neural Network Language Model (RNNLM)
- Recurrent Neural Network
  - Definition
  - Training via Backpropagation through Time (BPTT)
  - Training Issue
  - Extension
- RNN Applications
  - Sequential Input
  - **Sequential Output**
    - Aligned Sequential Pairs (Tagging)
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

Output Domain – Sequence Prediction

- POS Tagging

“推薦我台大後門的餐廳” ⟶ 推薦/VV 我/PN 台大/NR 後門/NN 的/DEG 餐廳/NN

- Speech Recognition



⟶ “大家好”

- Machine Translation

“How are you doing today?” ⟶ “你好嗎?”

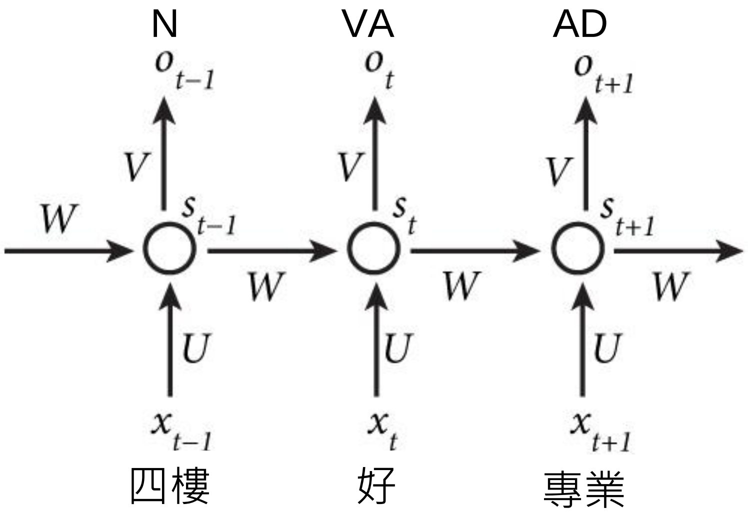
The output can be viewed as a sequence of classification

Outline

- Language Modeling
  - N-gram Language Model
  - Feed-Forward Neural Language Model
  - Recurrent Neural Network Language Model (RNNLM)
- Recurrent Neural Network
  - Definition
  - Training via Backpropagation through Time (BPTT)
  - Training Issue
  - Extension
- RNN Applications
  - Sequential Input
  - Sequential Output
    - **Aligned Sequential Pairs (Tagging)**
    - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

POS Tagging

- Tag a word at each timestamp
  - Input: word sequence
  - Output: corresponding POS tag sequence



Natural Language Understanding (NLU)

- Tag a word at each timestamp
  - Input: word sequence
  - Output: IOB-format slot tag and intent tag

<START>	just	sent	email	to	bob	about	fishing	this	weekend	<END>
O	O	O	O	O	B-contact_name	O	B-subject	I-subject	I-subject	send_email

→ send_email(contact_name="bob", subject="fishing this weekend")

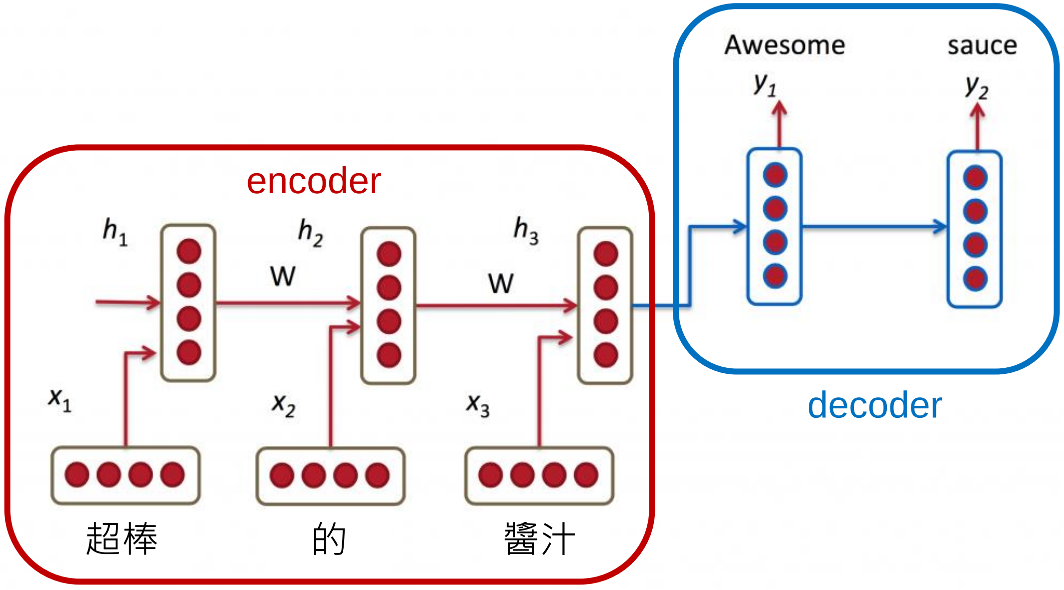
Temporal orders for input and output are the same

Outline

- Language Modeling
  - N-gram Language Model
  - Feed-Forward Neural Language Model
  - Recurrent Neural Network Language Model (RNNLM)
- Recurrent Neural Network
  - Definition
  - Training via Backpropagation through Time (BPTT)
  - Training Issue
  - Extension
- RNN Applications
  - Sequential Input
  - Sequential Output
    - Aligned Sequential Pairs (Tagging)
    - **Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)**

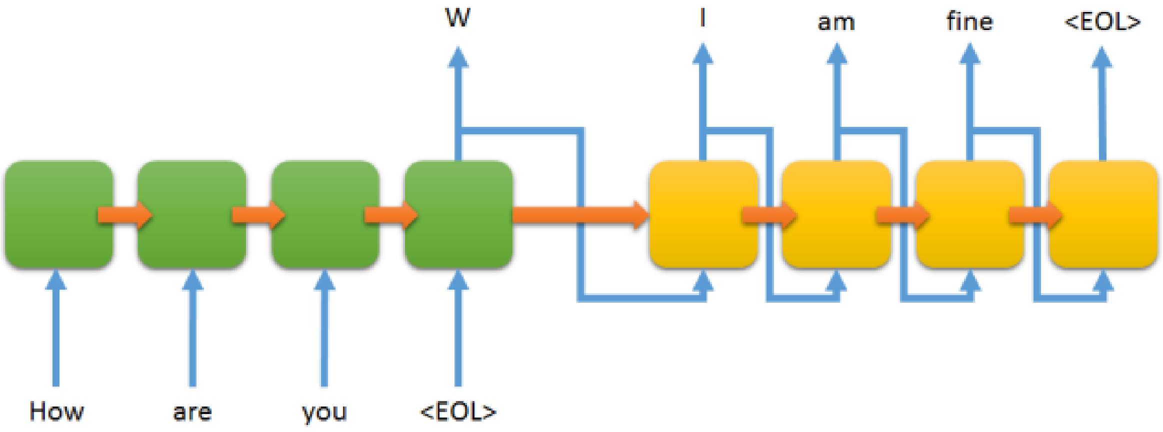
Machine Translation

- Cascade two RNNs, one for encoding and one for decoding
  - Input: word sequences in the source language
  - Output: word sequences in the target language



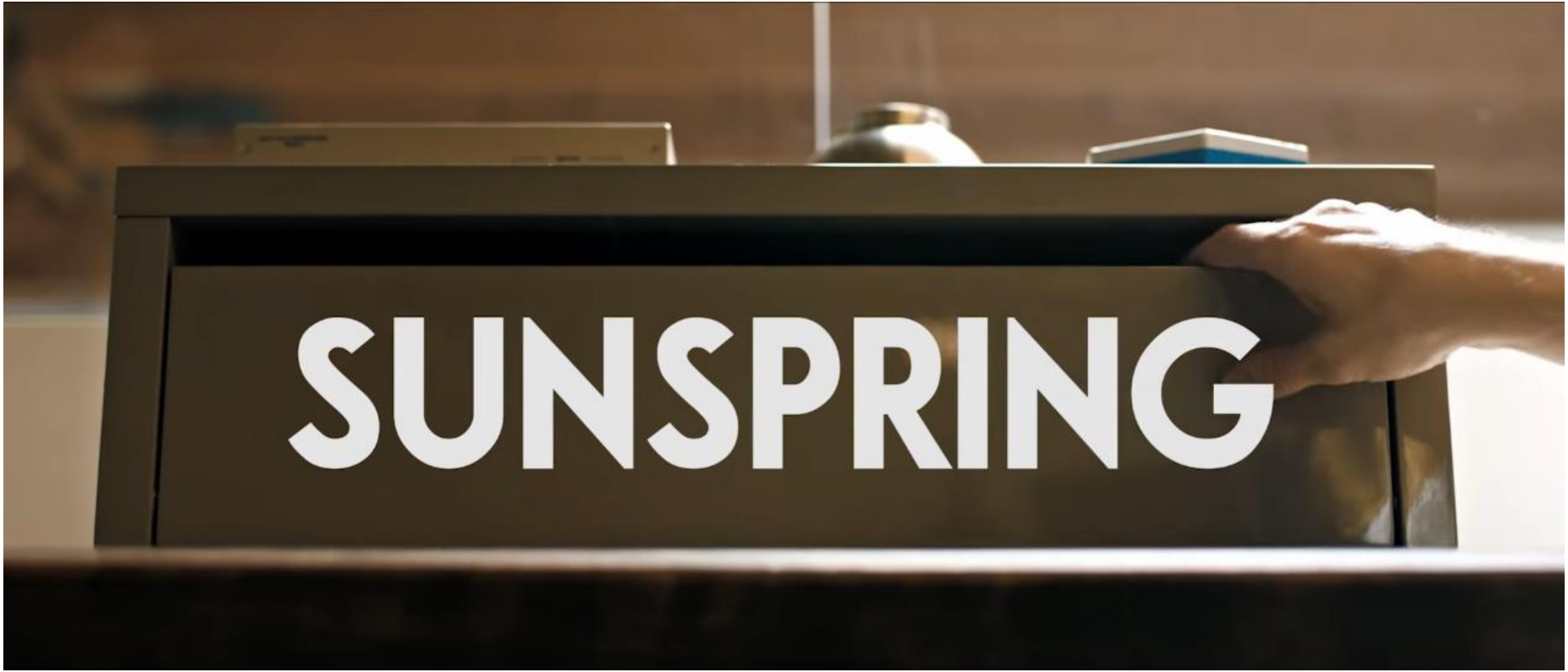
Chit-Chat Dialogue Modeling

- Cascade two RNNs, one for encoding and one for decoding
  - Input: word sequences in the question
  - Output: word sequences in the response



Temporal ordering for input and output may be different

Sci-Fi Short Film - SUNSPRING



Concluding Remarks

- Language Modeling
  - RNNLM
- Recurrent Neural Networks
  - Definition

$$s_t = \sigma(W s_{t-1} + U x_t)$$
$$o_t = \mathrm{softmax}(V s_t)$$

  - Backpropagation through Time (BPTT)
  - Vanishing/Exploding Gradient
- RNN Applications
  - Sequential Input: Sequence-Level Embedding
  - Sequential Output: Tagging / Seq2Seq (Encoder-Decoder)