CS 291A: Deep Learning for NLP

Neural Networks: Recurrent Neural Networks

William Wang
UCSB Computer Science
william@cs.ucsb.edu

Slides adapted from Y. V. Chen.

Project Proposals

1. I should have given you either oral feedback or written feedback.

2. If not, please come to my office hour today after the class.

Questions about Google Cloud Credits

1. If you are requesting Google Cloud Credits, please email our reader Ke Ni ke00@ucsb.edu.
2. Not a hard requirement. You don't have to use Google Cloud. For example, AWS also offers student credits, and you can register yourself.
3. But you are strongly encouraged to use GPU instead of CPU for your projects.

Homework 1

Due date: a week from now.

Homework assignments must be done independently, not in a team.

Note that CodaLab uses UTC, so please refer to the handout for PT.

Two models to implement:

- Word2Vec (SkipGram)
- Glove

Use your UCSB umail address to register CodaLab:

Or we could not locate your submission.

The scoreboard is anonymous, and your can use your teamname as your nickname.

Don’t be afraid of tensors

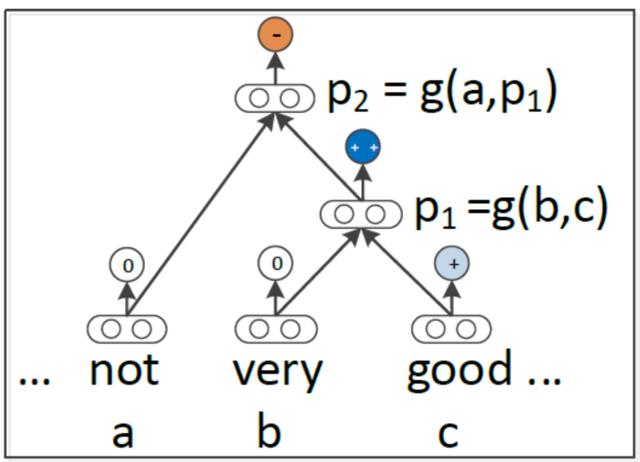
- It’s just a generalization of vector, and matrix.
- A rank-3 tensor is basically a set of matrices.
- So, let’s look at neural tensor network again.

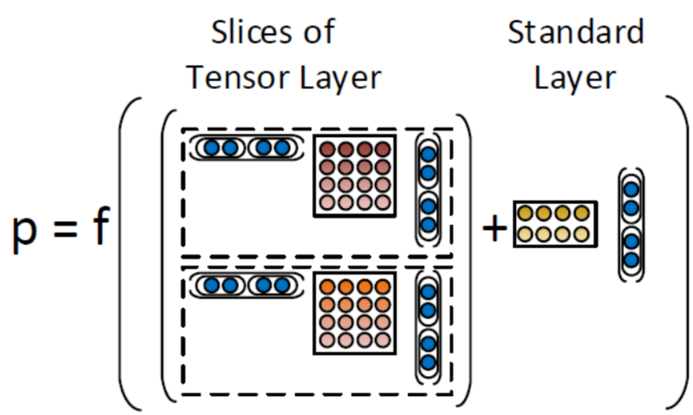
Review: Recursive Neural Tensor Network

$$v_p = \sigma(W \begin{bmatrix} v_{c_1} \\ v_{c_2} \end{bmatrix} + b)$$

Idea: allow more interactions of vectors

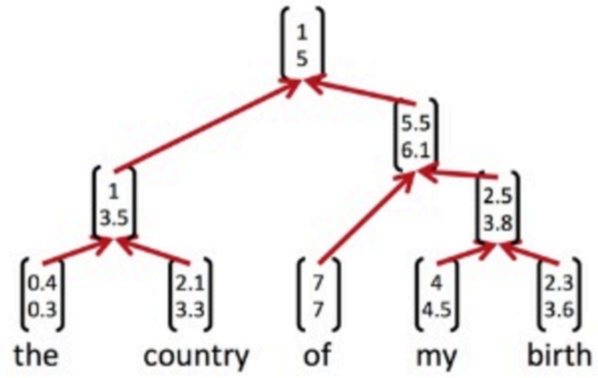
$$v_p = \sigma(\begin{bmatrix} v_{c_1} \\ v_{c_2} \end{bmatrix}^T V_{c_1,c_2} \begin{bmatrix} v_{c_1} \\ v_{c_2} \end{bmatrix} + W \begin{bmatrix} v_{c_1} \\ v_{c_2} \end{bmatrix} + b)$$

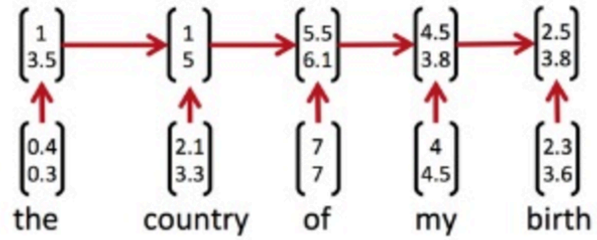




$$p = f\left(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:2]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix} \right)$$

Recursive vs. Recurrent





Review

Word Vector

Word2Vec Variants

Skip-gram: predicting surrounding words given the target word (Mikolov+, 2013)

$$p(w_{t-m}, \cdots w_{t-1}, w_{t+1}, \cdots , w_{t+m} \mid w_t)$$

CBOW (continuous bag-of-words): predicting the target word given the surrounding words (Mikolov+, 2013)

$$p(w_t \mid w_{t-m}, \cdots w_{t-1}, w_{t+1}, \cdots , w_{t+m})$$

LM (Language modeling): predicting the next words given the proceeding contexts (Mikolov+, 2013)

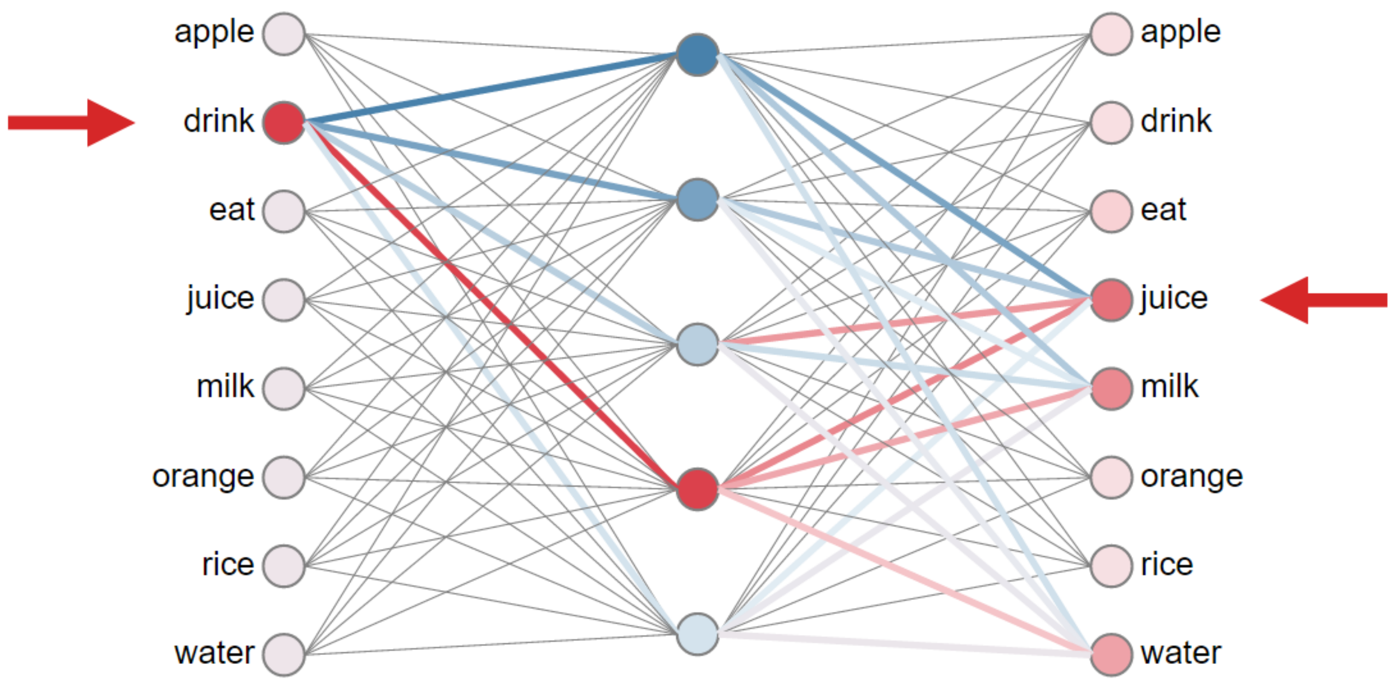
$$p(w_{t+1} \mid w_t)$$

Mikolov et al., "Efficient estimation of word representations in vector space," in *ICLR Workshop*, 2013.
Mikolov et al., "Linguistic regularities in continuous space word representations," in *NAACL HLT*, 2013.

Word2Vec LM

Goal: predicting the next words given the proceeding contexts

$$p(w_{t+1} \mid w_t)$$



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)

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

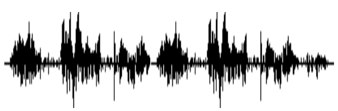
- Sequential Input
- Sequential Output
  - Aligned Sequential Pairs (Tagging)
  - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

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

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 $(n\text{-}1)$ 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

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

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(jumped | dog) = ~~0~~ 0.0001

P(ran | cat) = ~~0~~ 0.0001

give some small probability
→ smoothing

- ➢ The probability is not accurate.
- ➢ The phenomenon happens because we cannot collect all the possible text in the world as training data.

Outline

Language Modeling
- N-gram Language Model
- **Feed-Forward Neural Language Model**
- Recurrent Neural Network Language Model (RNNLM)

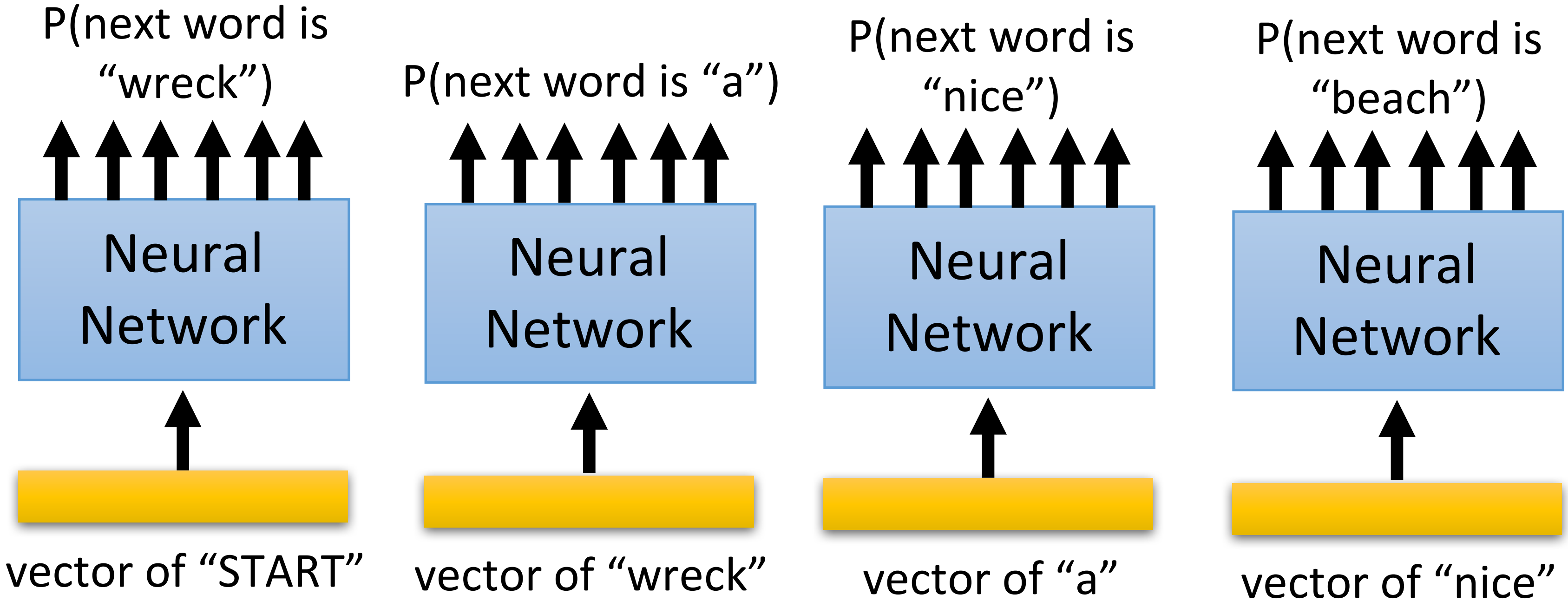
Recurrent Neural Network
- Definition
- Training via Backpropagation through Time (BPTT)
- Training Issue

Applications
- Sequential Input
- Sequential Output
  - Aligned Sequential Pairs (Tagging)
  - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

Neural Language Modeling

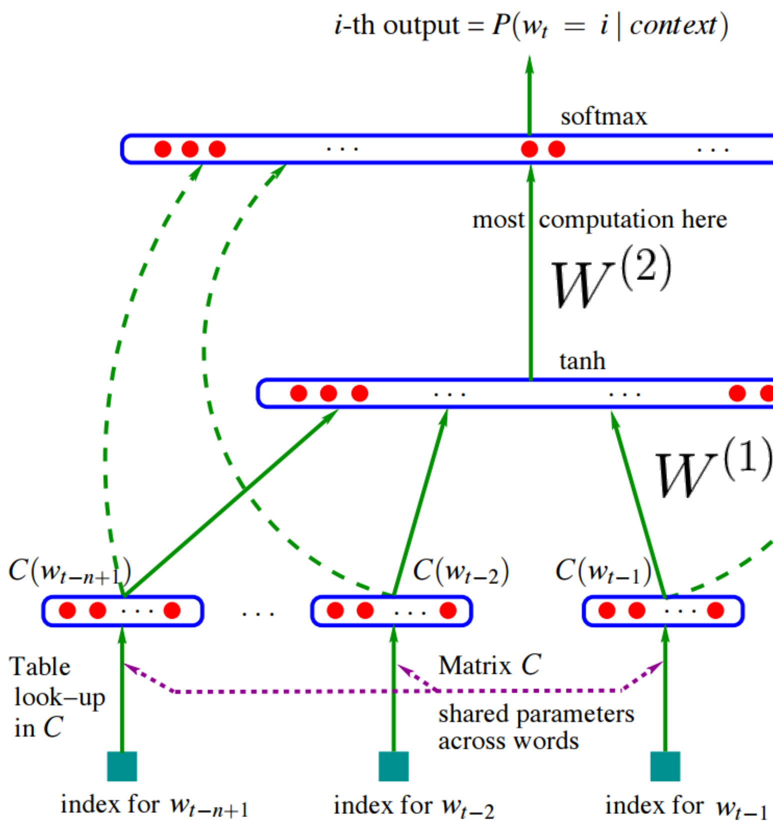
Idea: estimate $P(w_i \mid w_{i-(n-1)}, \cdots, w_{i-1})$ not from count, but from the NN prediction

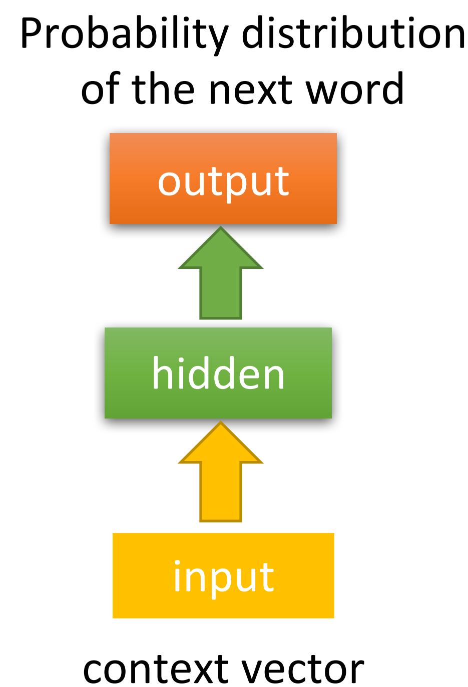
P("wreck a nice beach") = P(wreck|START)P(a|wreck)P(nice|a)P(beach|nice)



Neural Language Modeling

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



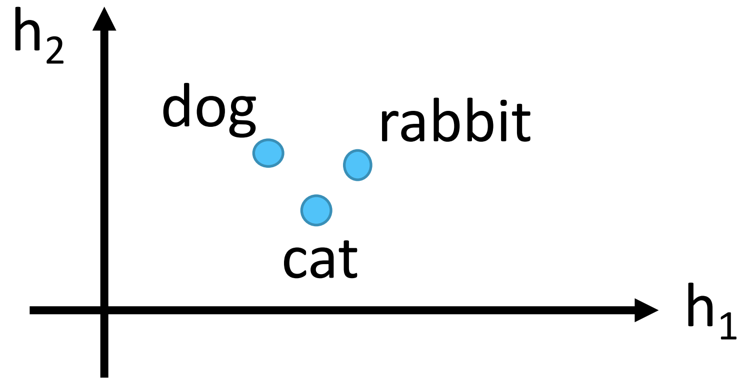


Issue: fixed context window for conditioning

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|dog) is large, P(jump|cat) increase accordingly (even there is not "… cat jump …" in the data)

Smoothing is automatically done

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

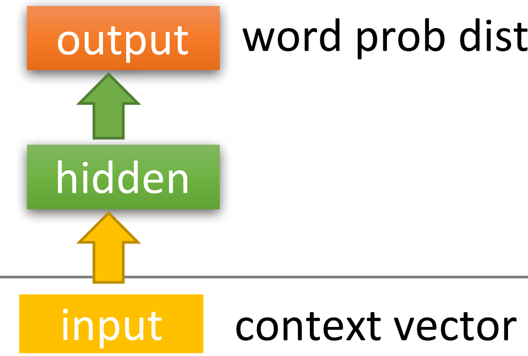
Applications
- Sequential Input
- Sequential Output
  - Aligned Sequential Pairs (Tagging)
  - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

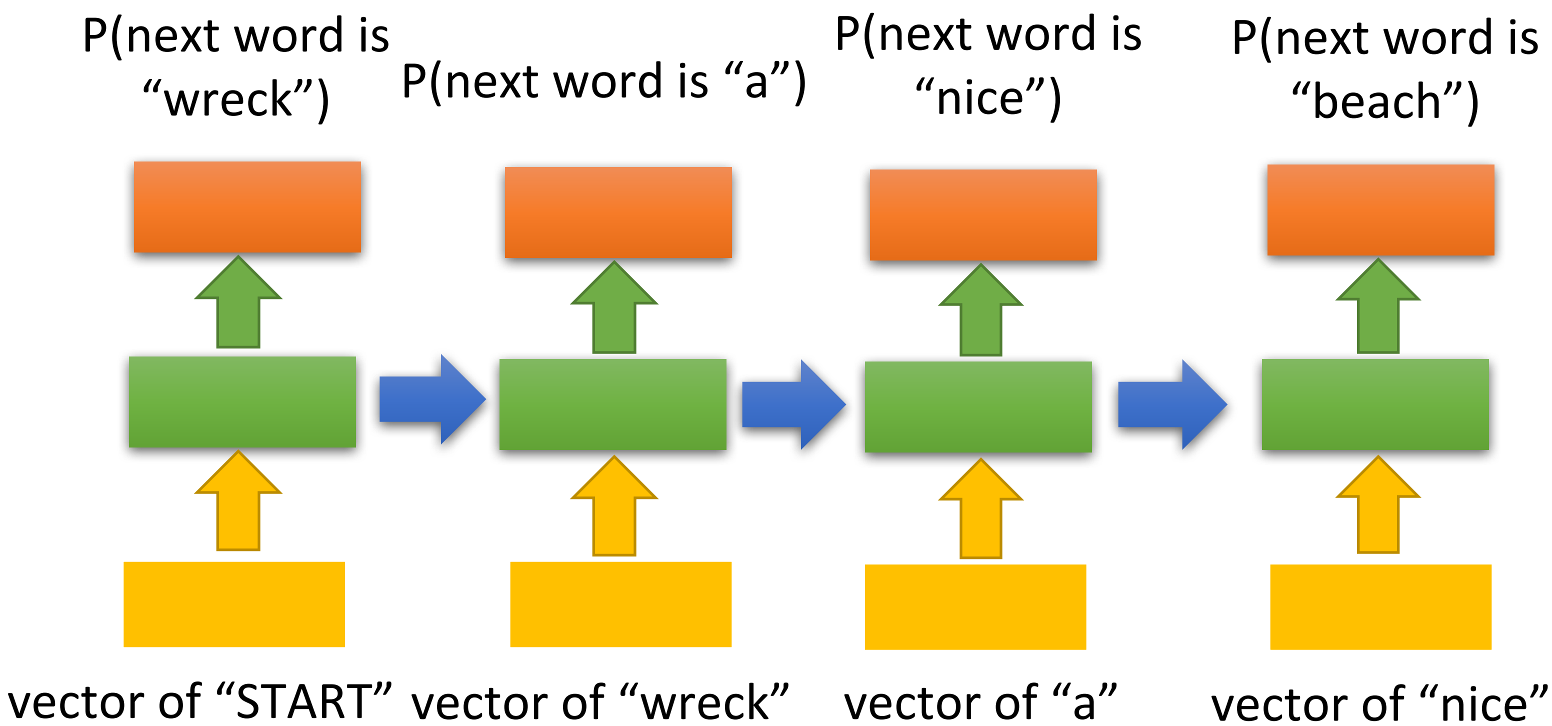
Recurrent Neural Network

Idea: condition the neural network on all previous words and tie the weights at each time step

Assumption: temporal information matters

RNN Language Modeling





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

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

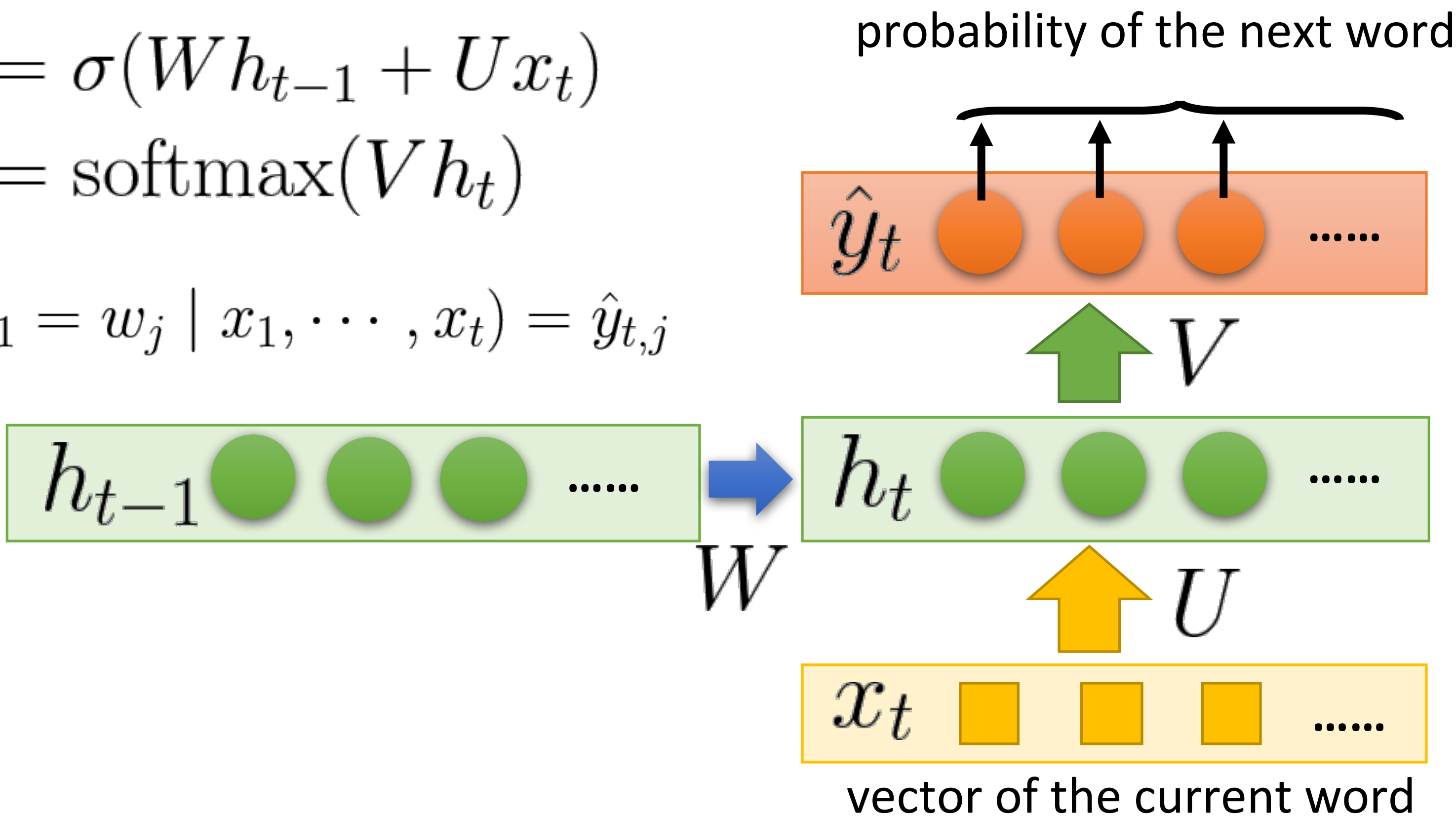
Applications
- Sequential Input
- Sequential Output
  - Aligned Sequential Pairs (Tagging)
  - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

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

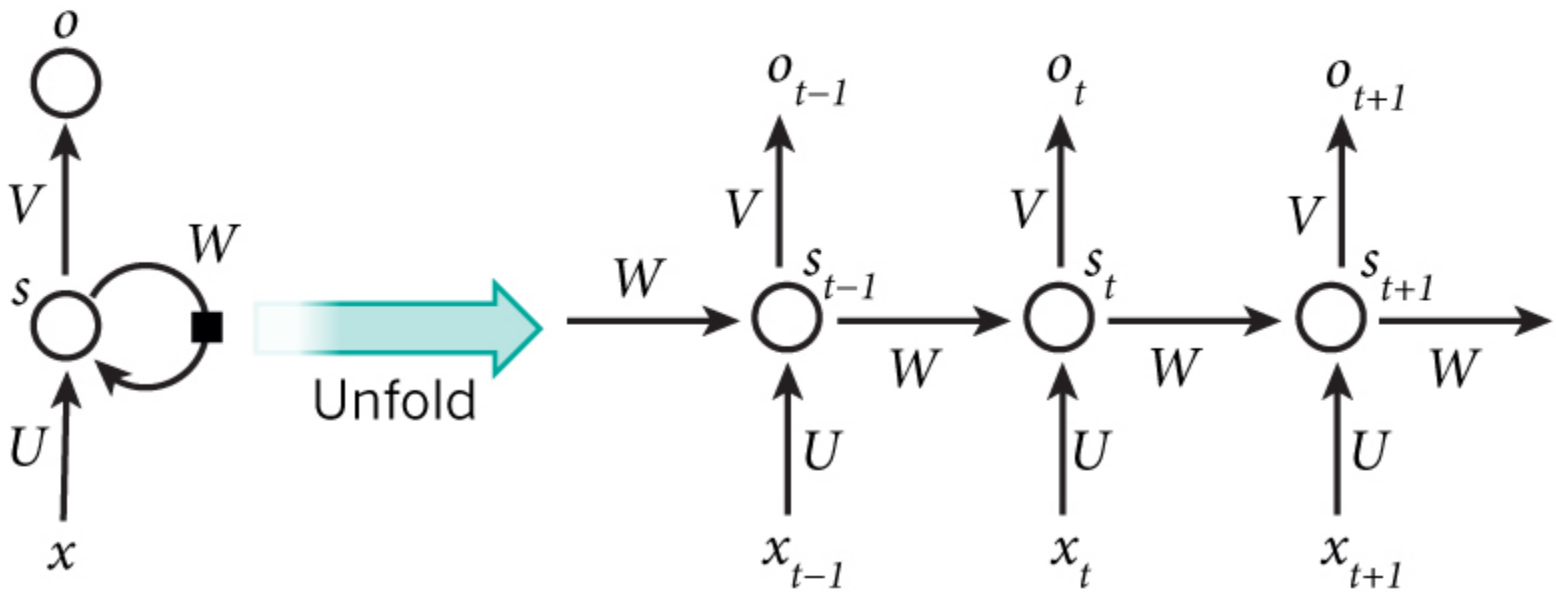
Applications
- Sequential Input
- Sequential Output
  - Aligned Sequential Pairs (Tagging)
  - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

Recurrent Neural Network Definition

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

$$o_t = \mathrm{softmax}(V s_t)$$

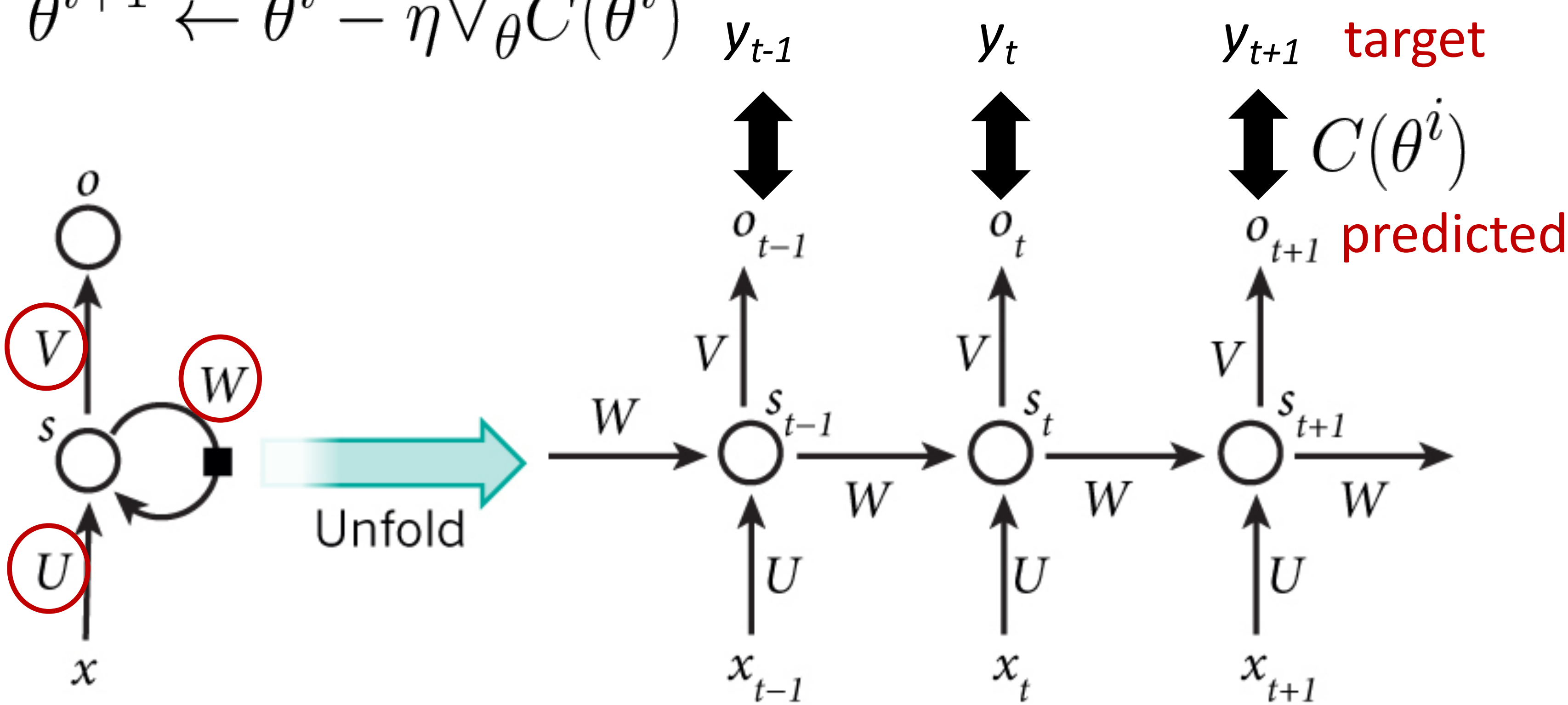
$\sigma(\cdot)$: tanh, ReLU



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

Applications
- Sequential Input
- Sequential Output
  - Aligned Sequential Pairs (Tagging)
  - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

In-class exercise:

Derive the delta rule for backpropagation

$y_j = \sigma(z_j), z_j = \sum x_i w_{ji}, t_j$: ground truth.

The quadratic error is defined as:

$$E = \sum_j \frac{1}{2}(t_j - y_j)^2$$

What is $\frac{\partial E}{\partial w_{ji}}$?

(Hint: use chain rule twice)

Derive the delta rule for backpropagation.

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial \left(\frac{1}{2}\left(t_j - y_j\right)^2\right)}{\partial w_{ji}}$$

$$= \frac{\partial \left(\frac{1}{2}\left(t_j - y_j\right)^2\right)}{\partial y_j} \frac{\partial y_j}{\partial w_{ji}}$$

$$= -\left(t_j - y_j\right) \frac{\partial y_j}{\partial w_{ji}}$$

$$= -\left(t_j - y_j\right) \frac{\partial y_j}{\partial z_j} \frac{\partial z_j}{\partial w_{ji}}$$

$$= -\left(t_j - y_j\right) \sigma'(z_j) \frac{\partial z_j}{\partial w_{ji}}$$

$$= -\left(t_j - y_j\right) \sigma'(z_j) \frac{\partial \left(\sum_k x_k w_{jk}\right)}{\partial w_{ji}}$$

$$\frac{\partial x_i w_{ji}}{\partial w_{ji}} = x_i$$

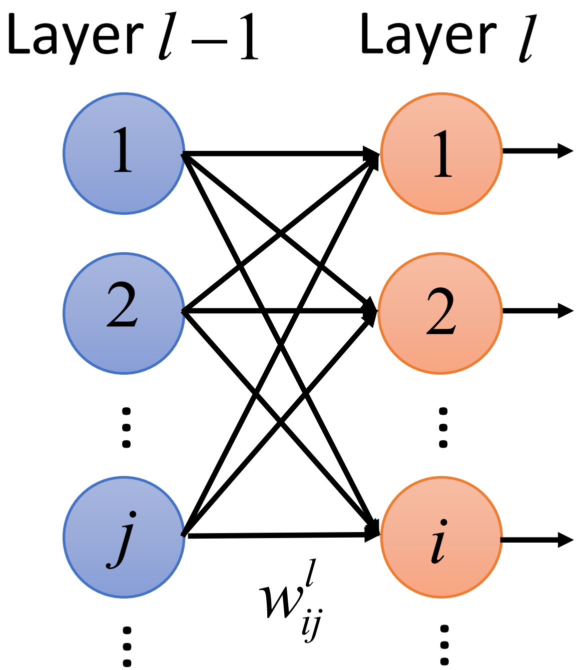
$$\frac{\partial E}{\partial w_{ji}} = -\left(t_j - y_j\right) \sigma'(z_j) x_i$$

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

$$\frac{\partial z_i^l}{\partial w_{ij}^l} = \begin{cases} a_j^{l-1} & l > 1 \\ x_j & l = 1 \end{cases}$$



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

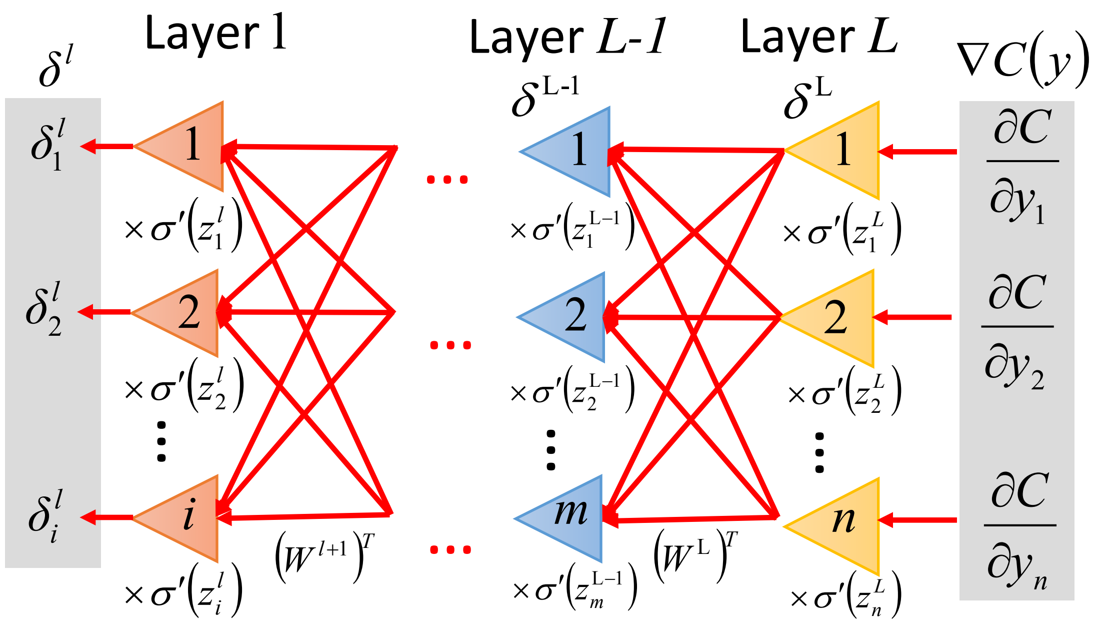
$$z^1 = W^1 x + b^1$$
$$a^1 = \sigma(z^1)$$
$$\vdots$$
$$z^l = W^l a^{l-1} + b^l$$
$$a^l = \sigma(z^l)$$
$$\vdots$$

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

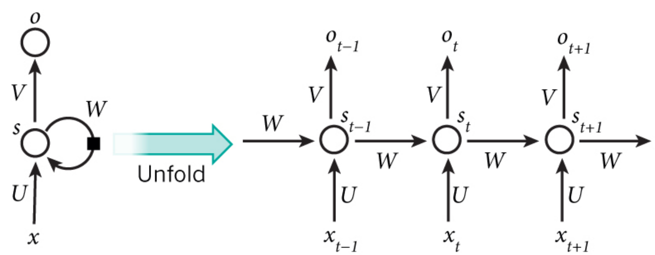


Backward Pass

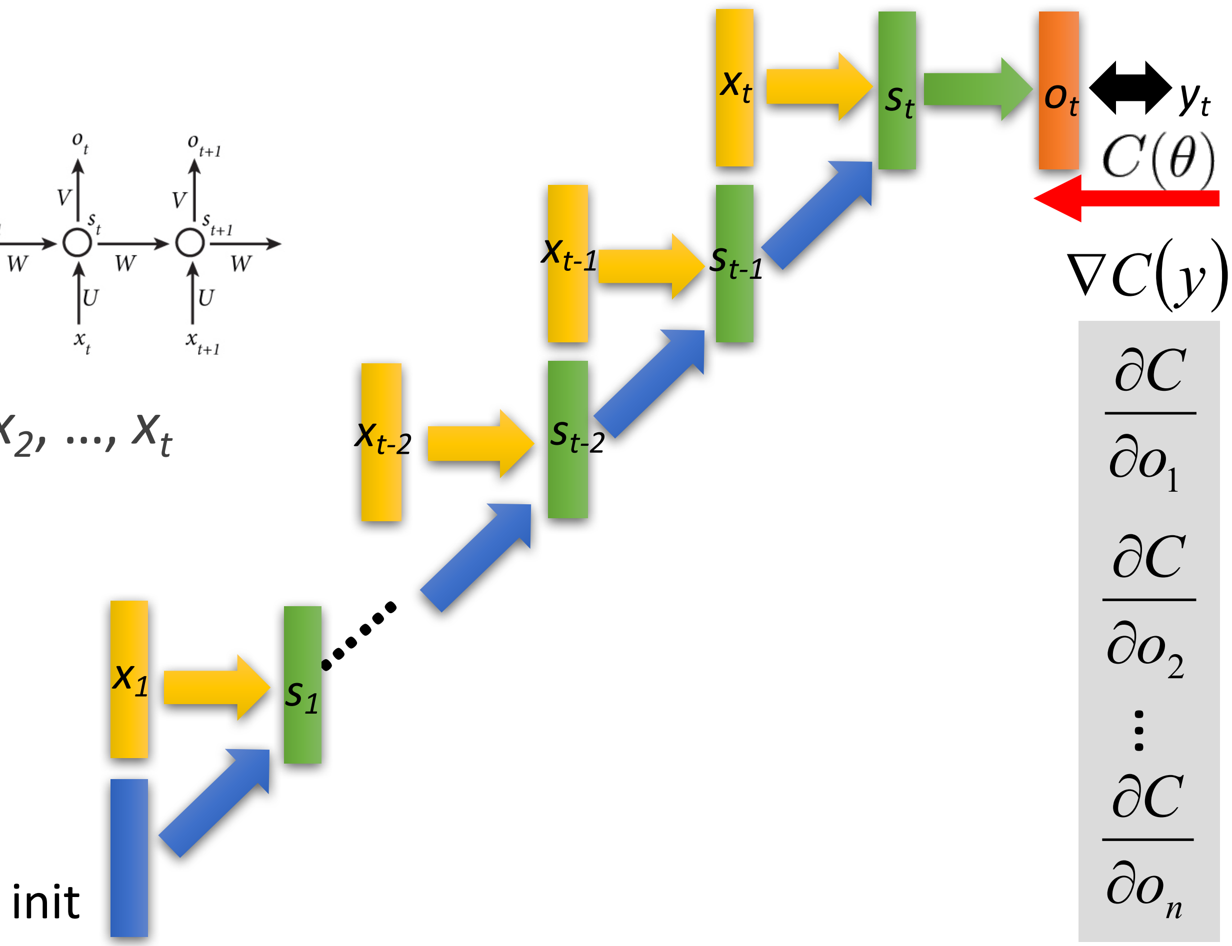
$$\delta^L = \sigma'(z^L) \odot \nabla C(y)$$
$$\delta^{L-1} = \sigma'(z^{L-1}) \odot (W^L)^T \delta^L$$
$$\vdots$$
$$\delta^l = \sigma'(z^l) \odot (W^{l+1})^T \delta^{l+1}$$
$$\vdots$$

Backpropagation through Time (BPTT)

Unfold

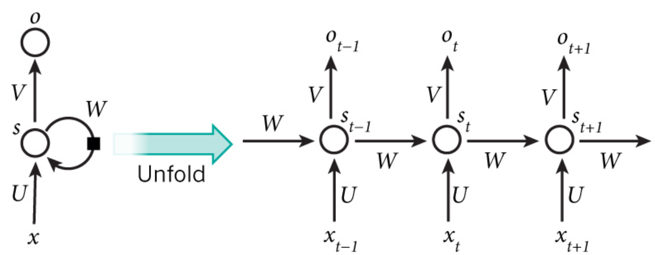


- Input: init, x_1, x_2, …, x_t
- Output: o_t
- Target: y_t

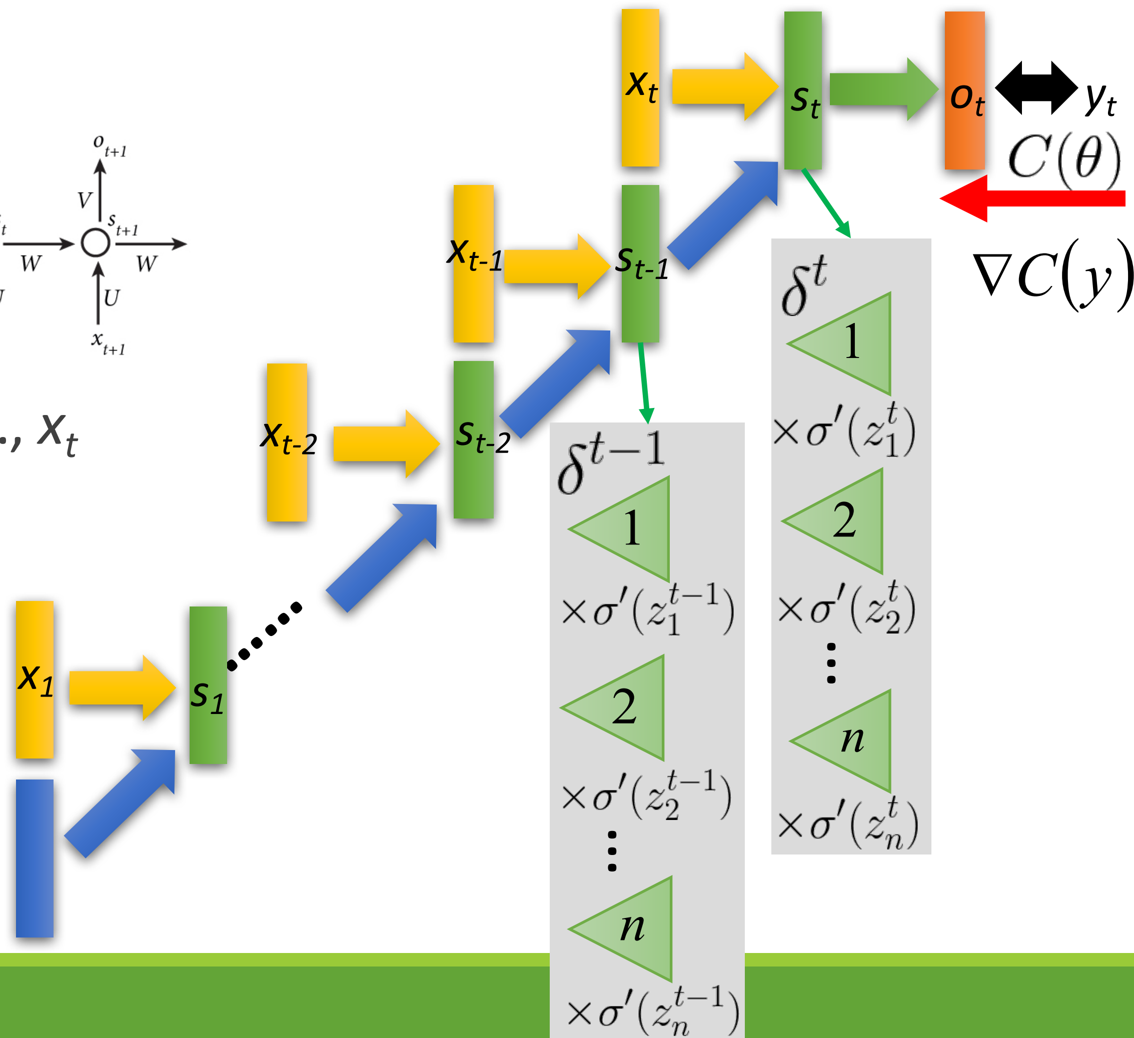


Backpropagation through Time (BPTT)

Unfold

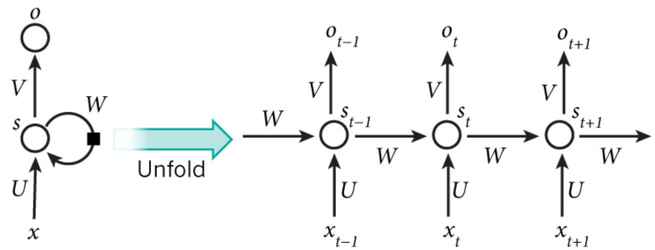


- Input: init, x_1, x_2, …, x_t
- Output: o_t
- Target: y_t

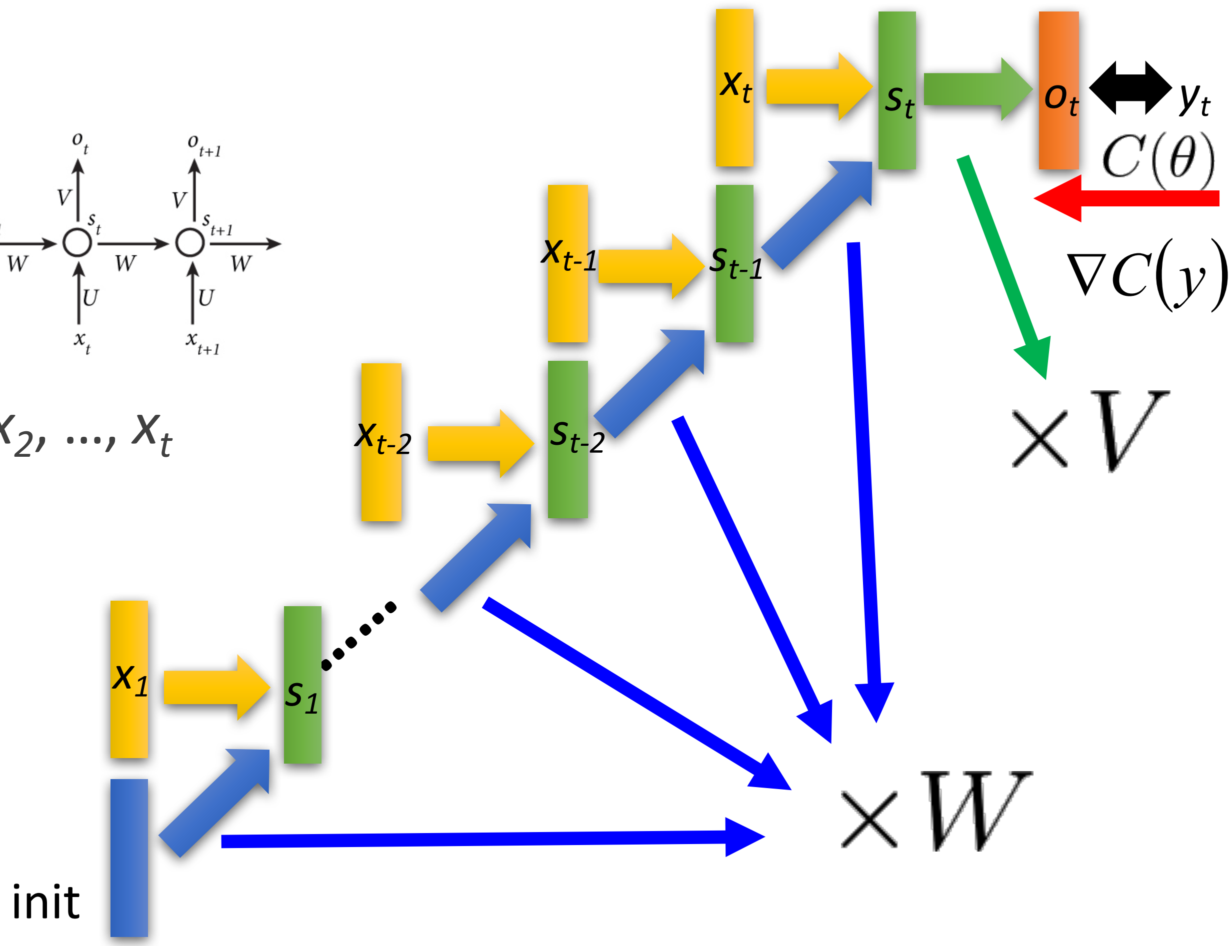


Backpropagation through Time (BPTT)

Unfold

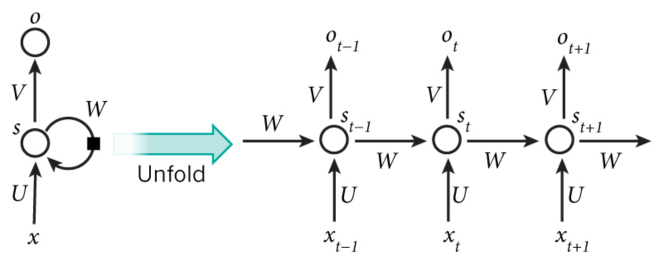


- Input: init, x_1, x_2, …, x_t
- Output: o_t
- Target: y_t

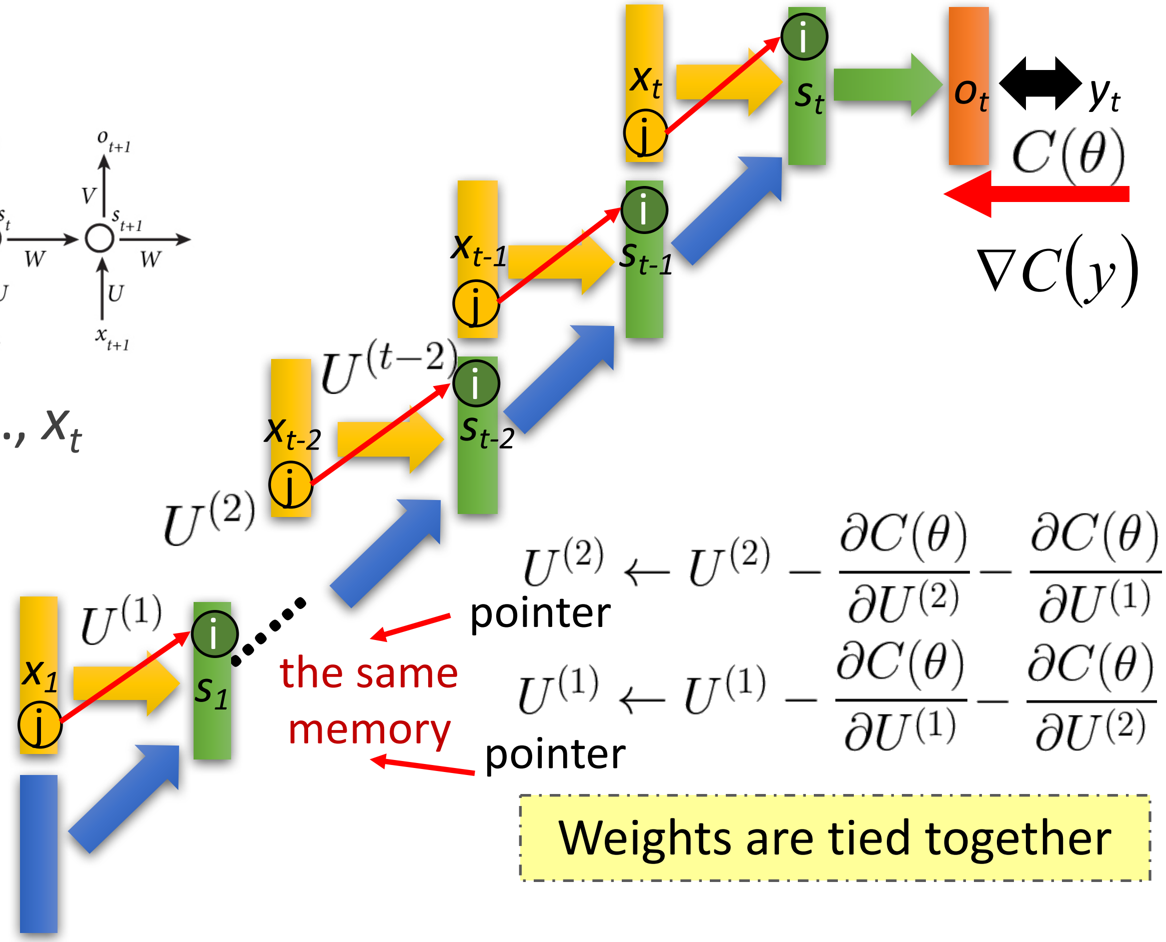


Backpropagation through Time (BPTT)

Unfold



- Input: init, $x_1, x_2, \ldots, x_t$
- Output: o_t
- Target: y_t



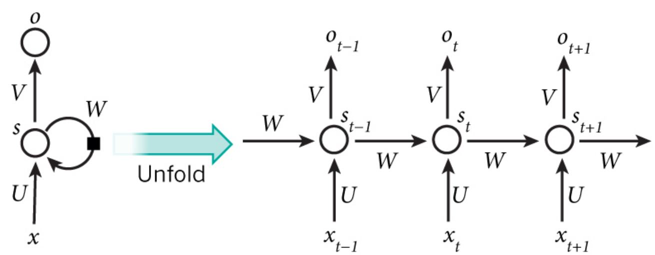
$$U^{(2)} \leftarrow U^{(2)} - \frac{\partial C(\theta)}{\partial U^{(2)}} - \frac{\partial C(\theta)}{\partial U^{(1)}}$$

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

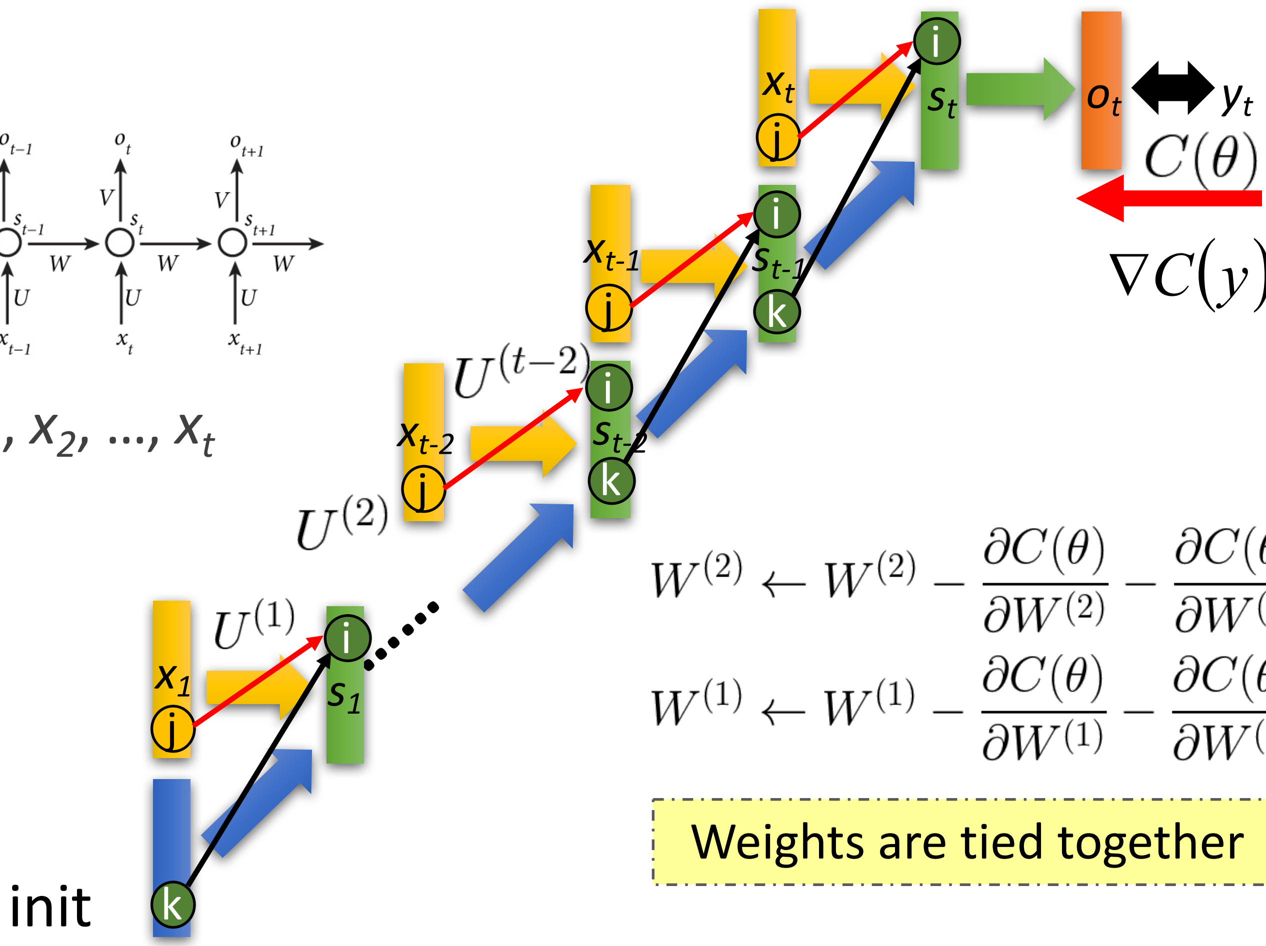
Weights are tied together

Backpropagation through Time (BPTT)

Unfold



- Input: init, $x_1, x_2, \ldots, x_t$
- Output: o_t
- Target: y_t



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

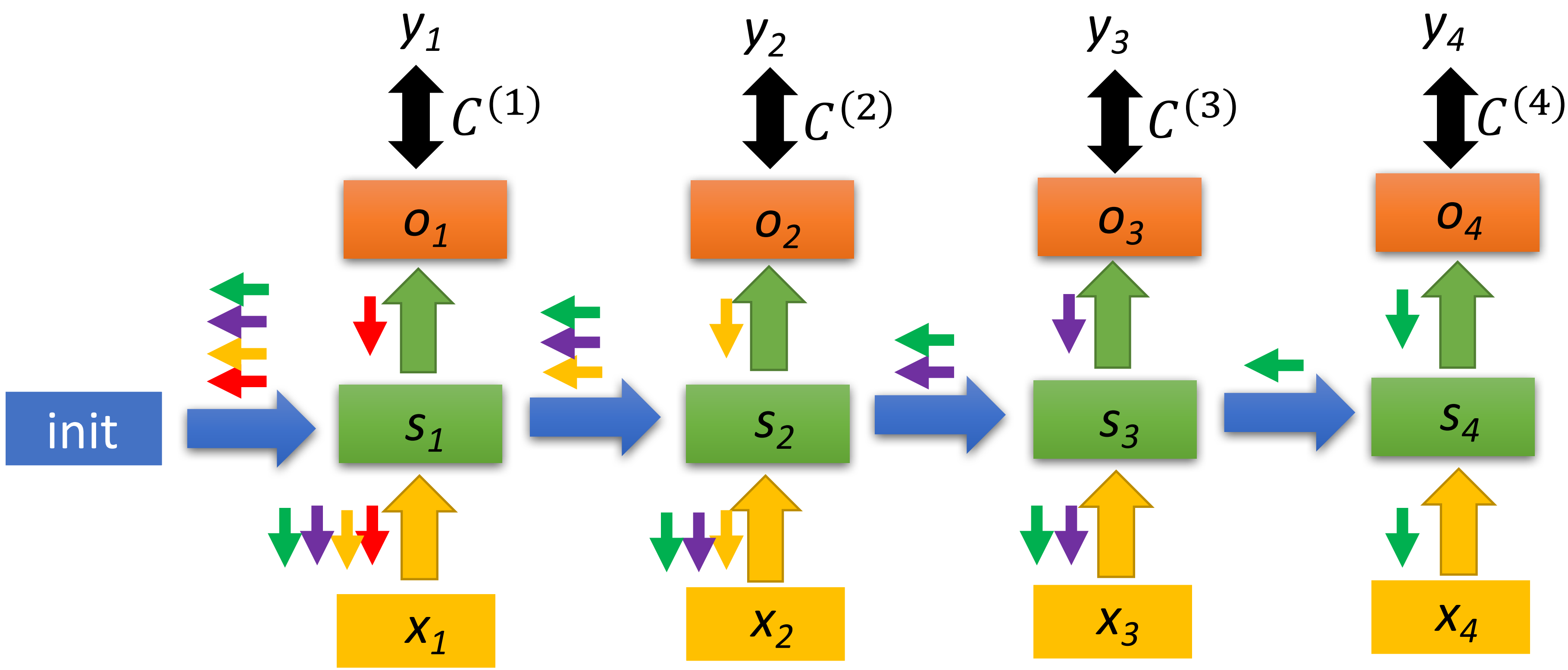
$$W^{(1)} \leftarrow W^{(1)} - \frac{\partial C(\theta)}{\partial W^{(1)}} - \frac{\partial C(\theta)}{\partial W^{(2)}}$$

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**

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

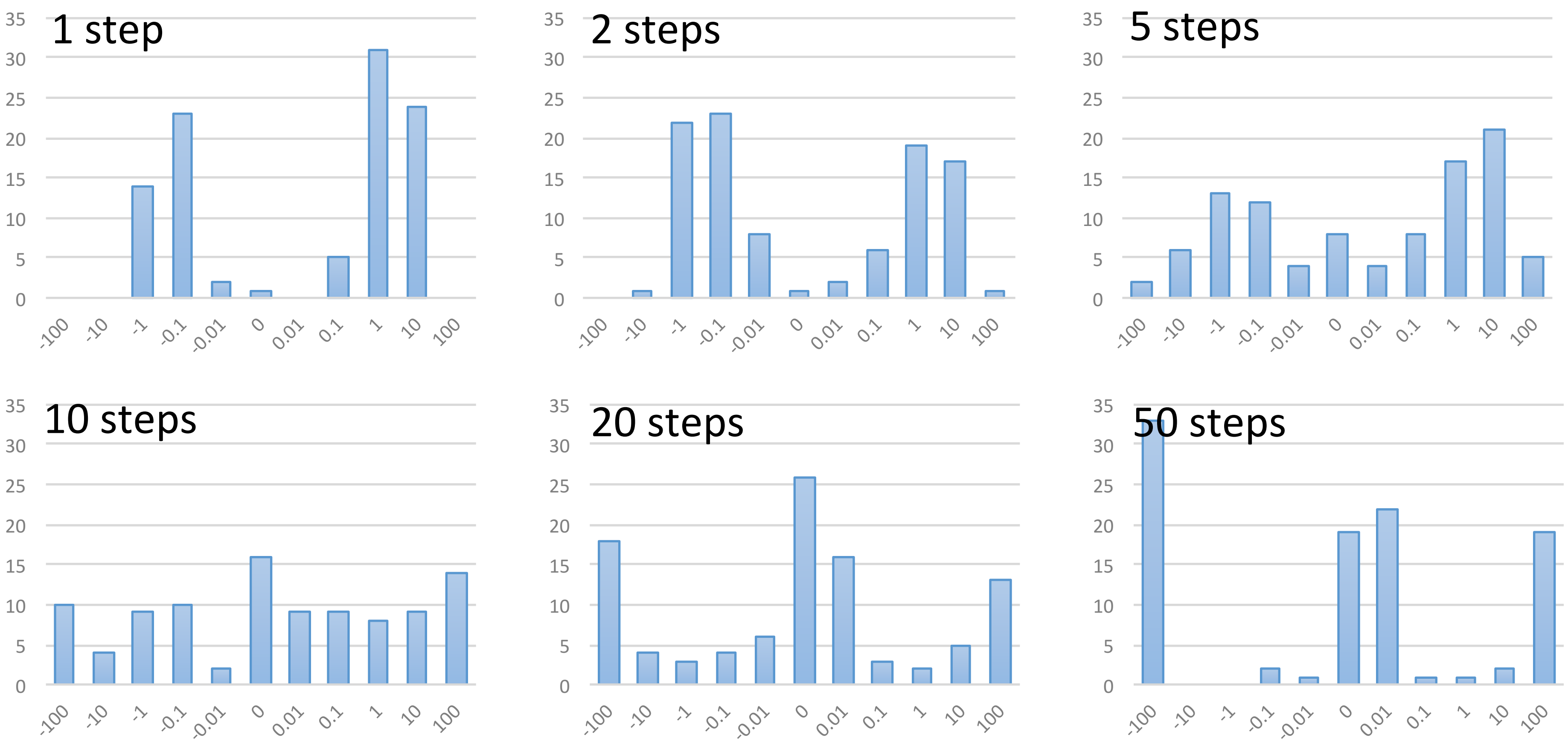
Multiply the same matrix at each time step during backprop

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

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]

Vanishing/Exploding Gradient Example



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)

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)

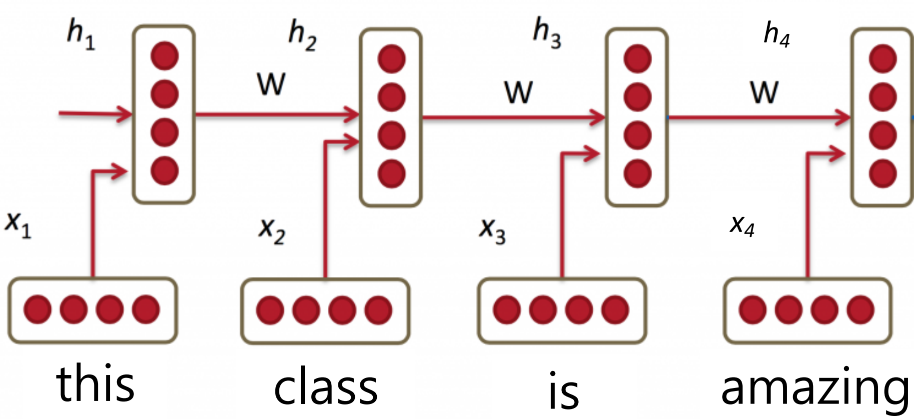
$$
\begin{array}{ll}
 & \overbrace{\hphantom{\begin{bmatrix}0.2 & 0.6 & 0.3 & \cdots & 0.4\end{bmatrix}}}^{N\text{-dim}} \\
\text{this} & \begin{bmatrix}0.2 & 0.6 & 0.3 & \cdots & 0.4\end{bmatrix} \\
\text{class} & \begin{bmatrix}0.9 & 0.8 & 0.1 & \cdots & 0.1\end{bmatrix} \\
\text{is} & \begin{bmatrix}0.1 & 0.3 & 0.1 & \cdots & 0.7\end{bmatrix} \\
\text{amazing} & \begin{bmatrix}0.5 & 0.0 & 0.6 & \cdots & 0.4\end{bmatrix}
\end{array}
$$

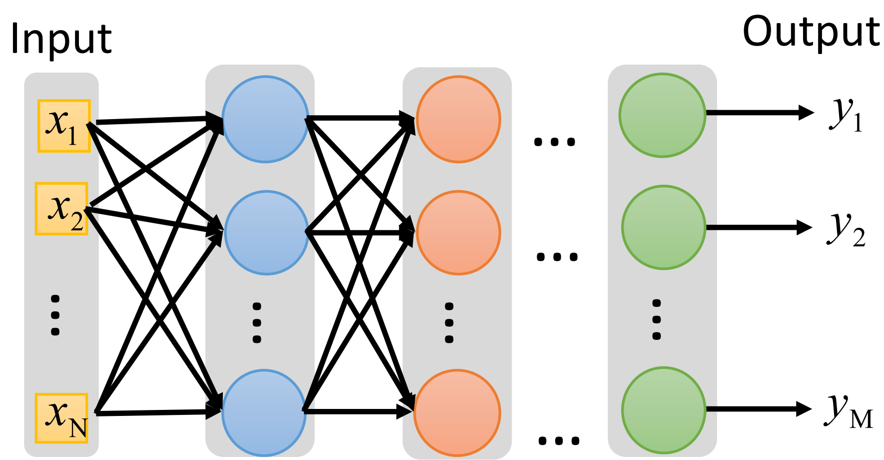
How to compute $\vec{x} = \begin{bmatrix}x_1 & x_2 & x_3 & \cdots & x_N\end{bmatrix}$

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 the input of classification tasks

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

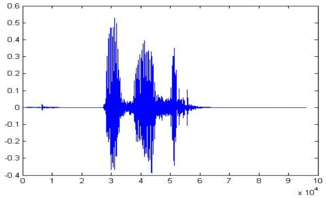
- Sequential Input
- **Sequential Output**
  - Aligned Sequential Pairs (Tagging)
  - Unaligned Sequential Pairs (Seq2Seq/Encoder-Decoder)

Output Domain – Sequence Prediction

POS Tagging

"this class is amazing" → This/NT class/NN is/VBZ amazing/JJ.

Speech Recognition



→ "how are you?"

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

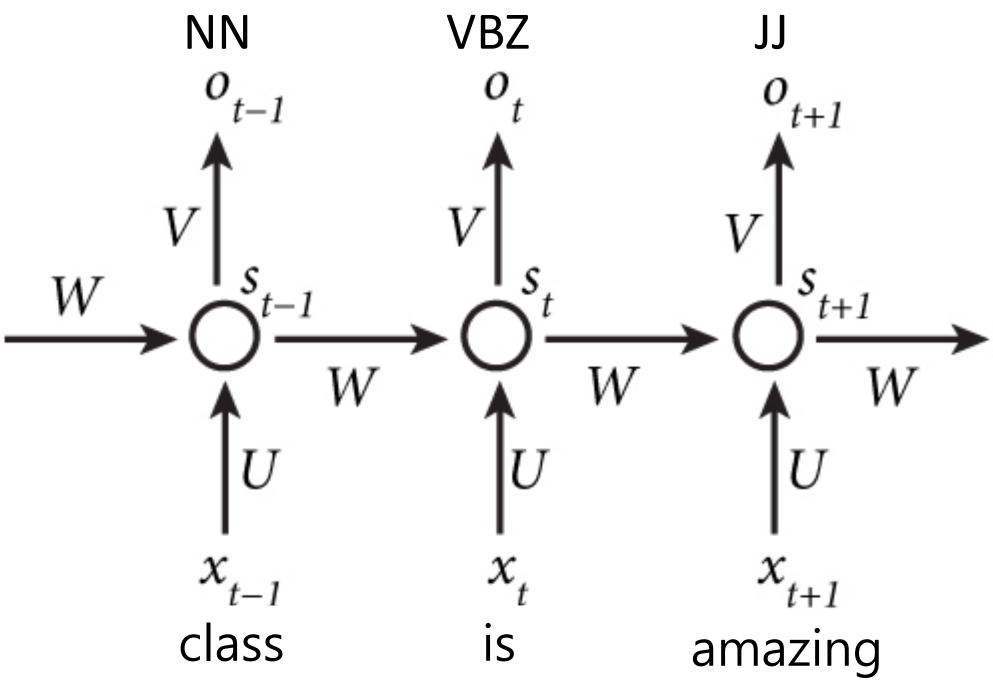
Applications

- 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)

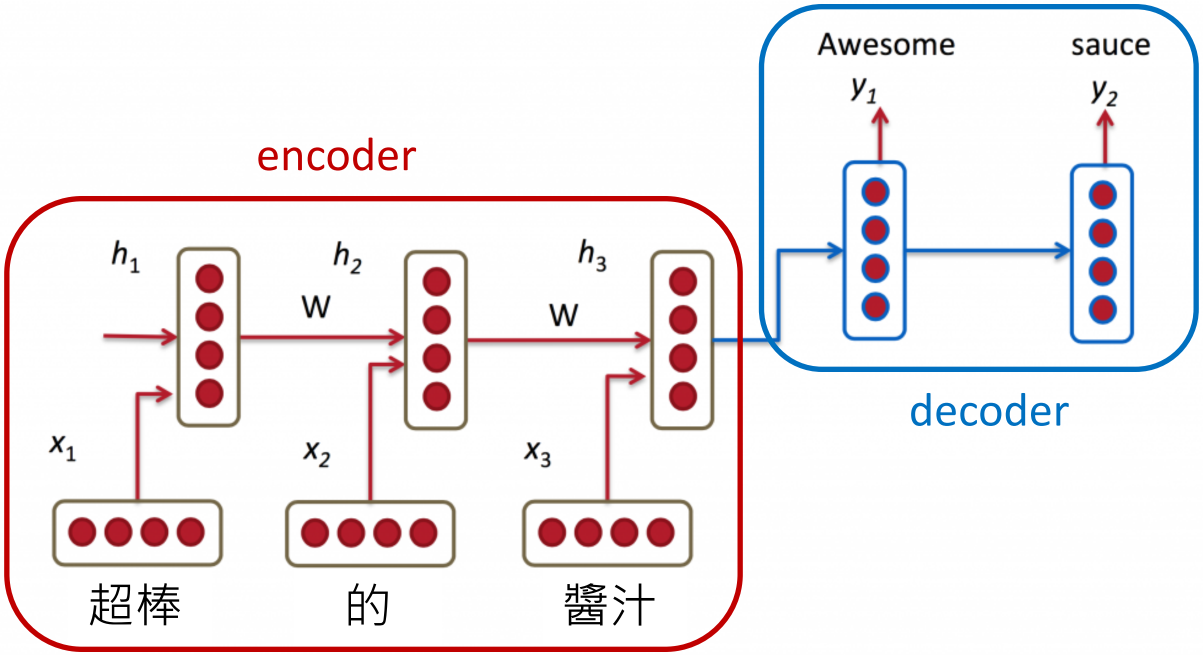
Recurrent Neural Network
- Definition
- Training via Backpropagation through Time (BPTT)
- Training Issue

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

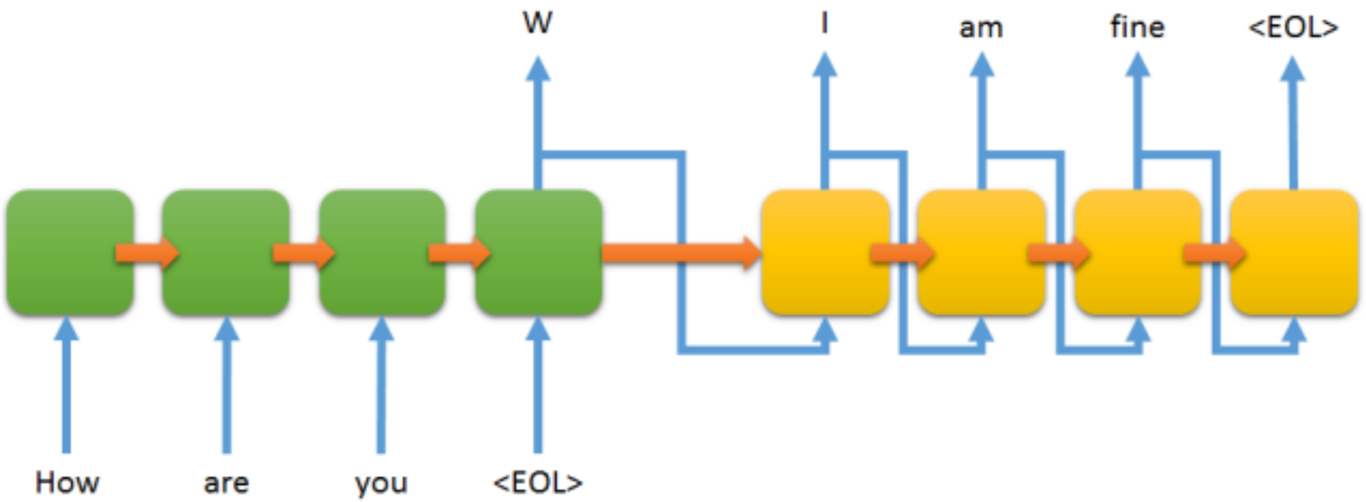
- Input: word sequences in the source language
- Output: word sequences in the target language



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

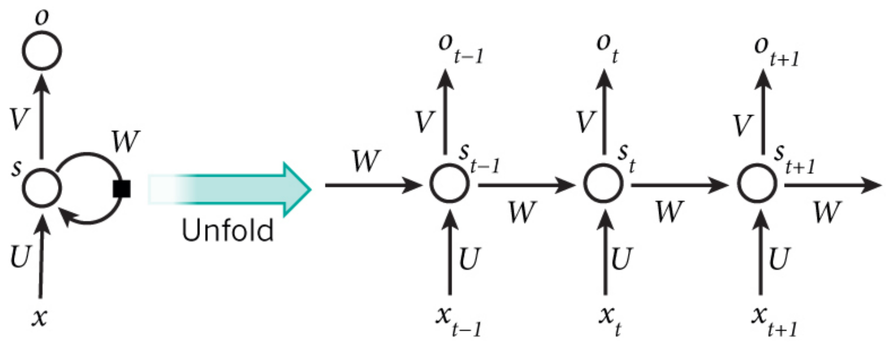
Concluding Remarks

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



Applications
- Sequential Input: Sequence-Level Embedding
- Sequential Output: Tagging / Seq2Seq (Encoder-Decoder)