InfoLab

Deep Learning Seminar
CH 10. Sequence modeling: Recurrent and Recursive Nets – Part 1

2017-10-25

Eunjeong Yi

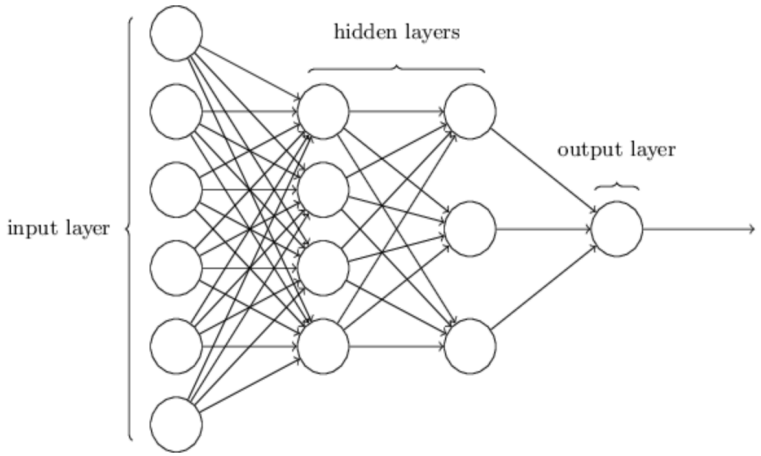
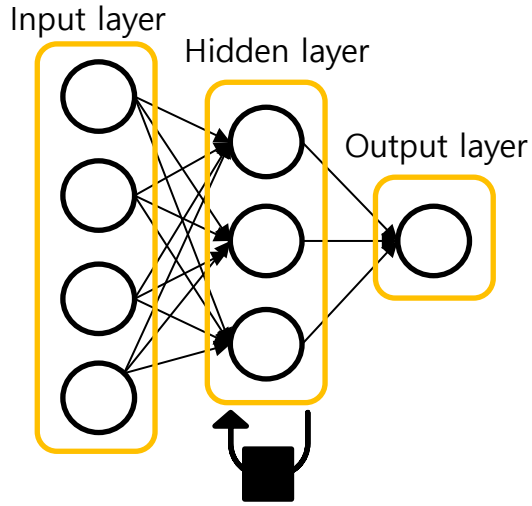
DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Chapter 10. Sequence modeling: Recurrent and Recursive Nets

- **Part 1**
  - **10.1 Unfolding Computational Graphs**
  - **10.2 Recurrent Neural Networks**
  - **10.3 Bidirectional RNNs**
  - **10.4 Encoder-Decoder Sequence-to-Sequence Architectures**
  - **10.5 Deep Recurrent Networks**
  - **10.6 Recursive Neural Networks**

RNN(Recurrent Neural Network)

- **Specialized for processing a sequence data**
- **Sharing parameters across different parts of a model**
- **Time step index refer to the position in sequence**

Neural network	Recurrent neural network
	

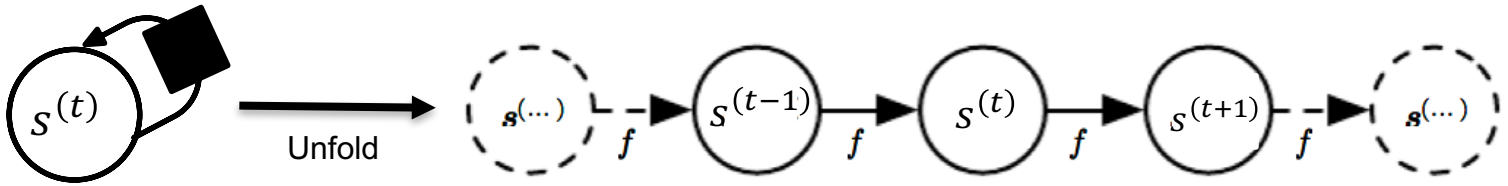
Michael Nielsen. "CHAPTER 1 Using neural nets to recognize handwritten digits"
http://neuralnetworksanddeeplearning.com/chap1.html

Unfolding Computational Graphs

- **Computational graph**
  - a way to formalize the structure of a set of computations
  - Ex) $\boldsymbol{s^{(t)} = f\left(s^{(t-1)}; \theta\right)}$

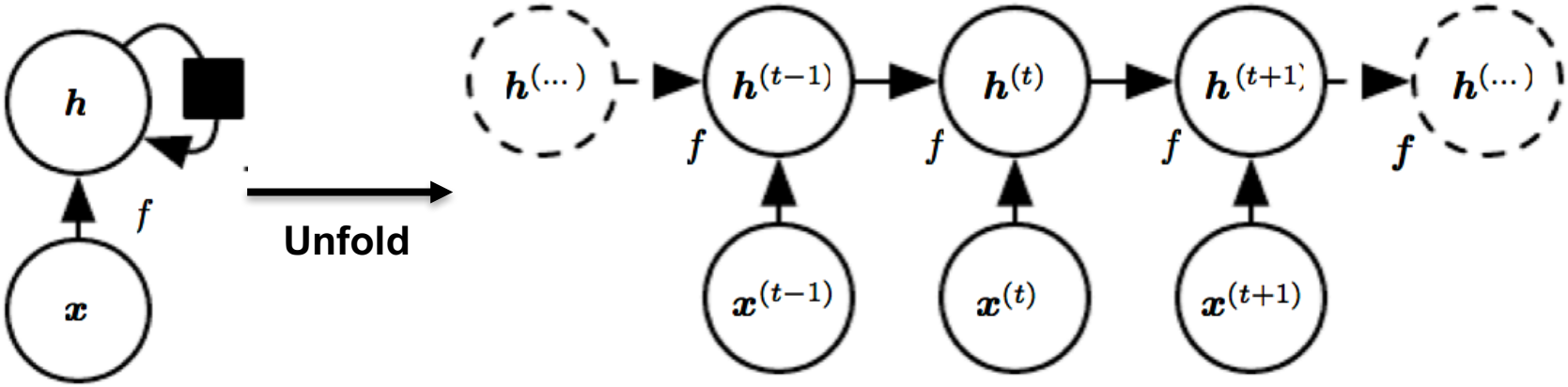
- **Unfolding equation**

$$\begin{aligned} \boldsymbol{s^{(t)}} &= \boldsymbol{f\left(s^{(t-1)}; \theta\right)} \\ &= \boldsymbol{f(f\left(s^{(t-2)}; \theta\right)} \\ &= \boldsymbol{f(f(f\left(s^{(t-3)}; \theta\right)} \\ &\vdots \\ &= \boldsymbol{f(f(f(\cdots(f\left(s^{(1)}; \theta\right))} \end{aligned}$$



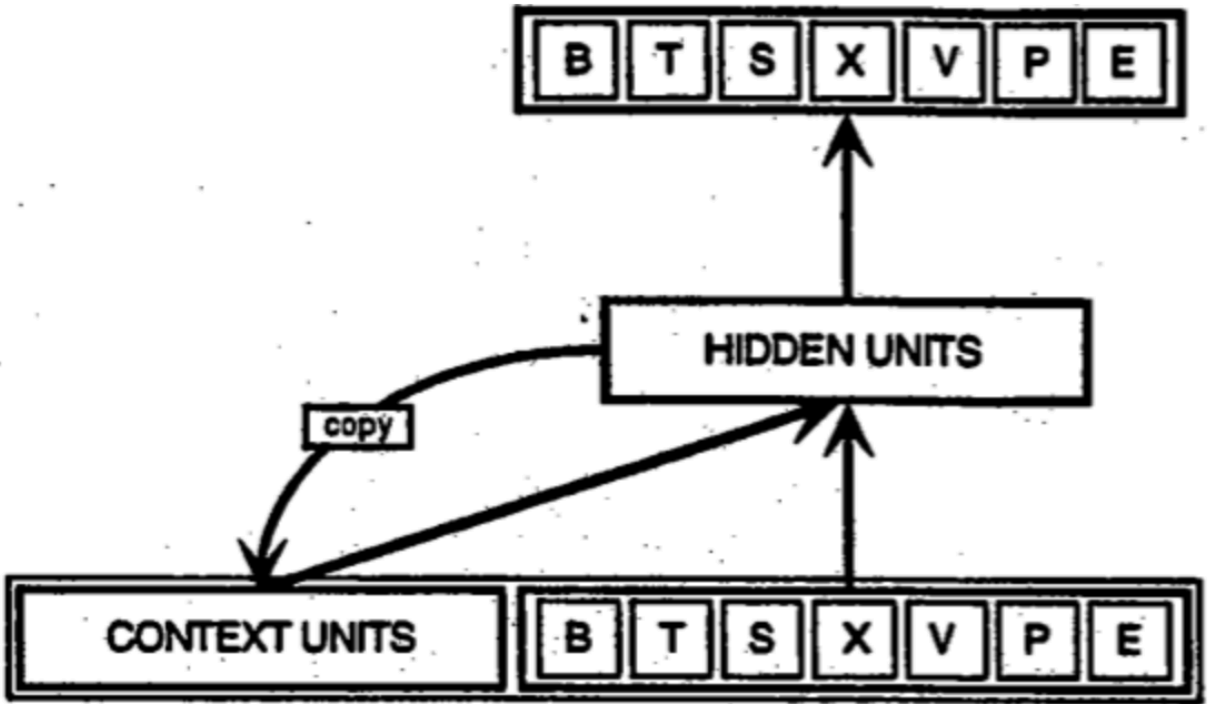
Unfolding recurrent network

- $\boldsymbol{h^{(t)} = g^{(t)}(x^{(t)}, x^{(t-1)}, \cdots, x^{(1)}) = f(h^{(t-1)}, x^{(t)}; \theta)}$
- **Advantage**
  - To use the transition function f with the same parameters at every time step.



Recurrent Neural Networks

- Receiving input data and context unit in hidden layer
- Feedback structure that seems like memory

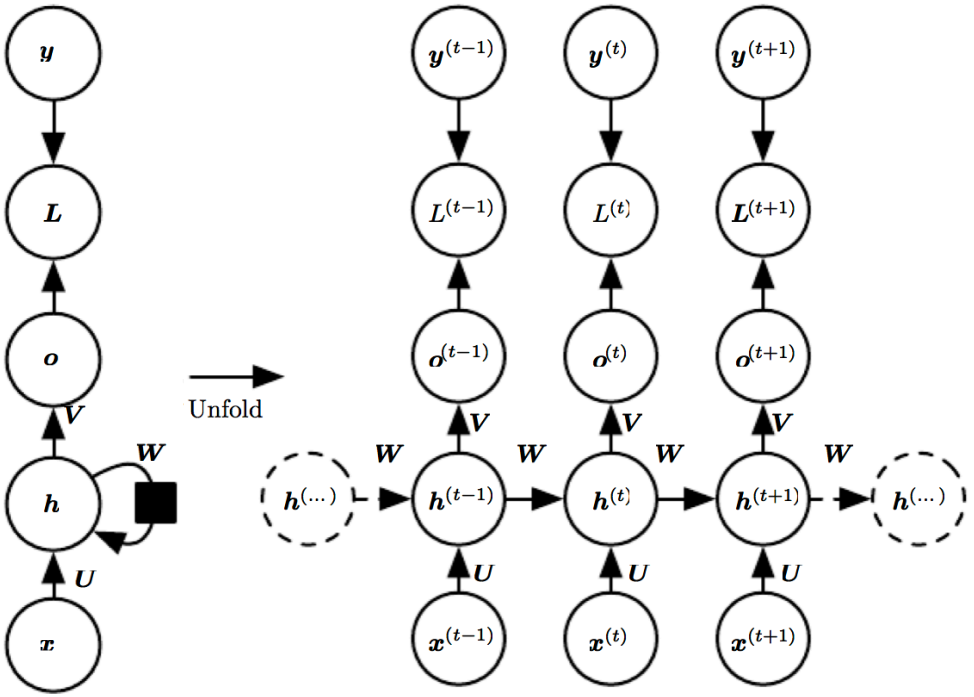


Elman, J. L. Finding structure in time. In *Cognitive Science,* 1990

Recurrent Neural Networks

- $\boldsymbol{h^{(t)} = f(Wh^{(t-1)} + Ux^{(t)})}$
- $\boldsymbol{o^{(t)} = Vh^{(t)}}$
- $\boldsymbol{\hat{y}^{(t)} = softmax(o^{(t)})}$

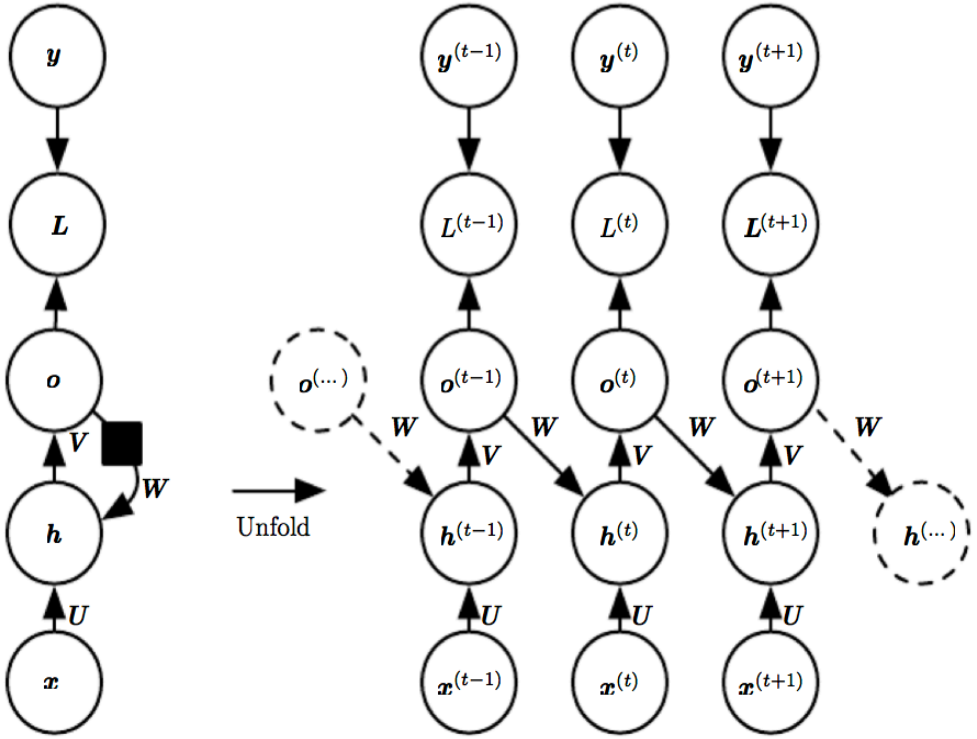
} **Forward propagation**



(o: output, L: loss, y: target, W: weight)

Recurrent Neural Networks

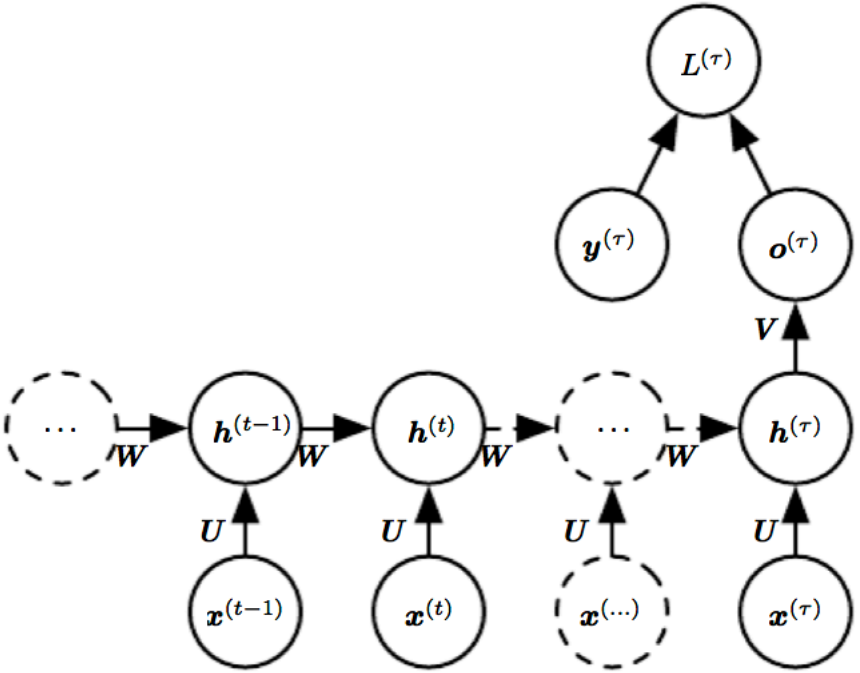
- Produce an output at each time step and have recurrent connections only from the output at one time step to the hidden units at the next time step
- Lack important information from the past
- Difficult to parallelization



(o: output, L: loss, y: target, W: weight)

Recurrent Neural Networks

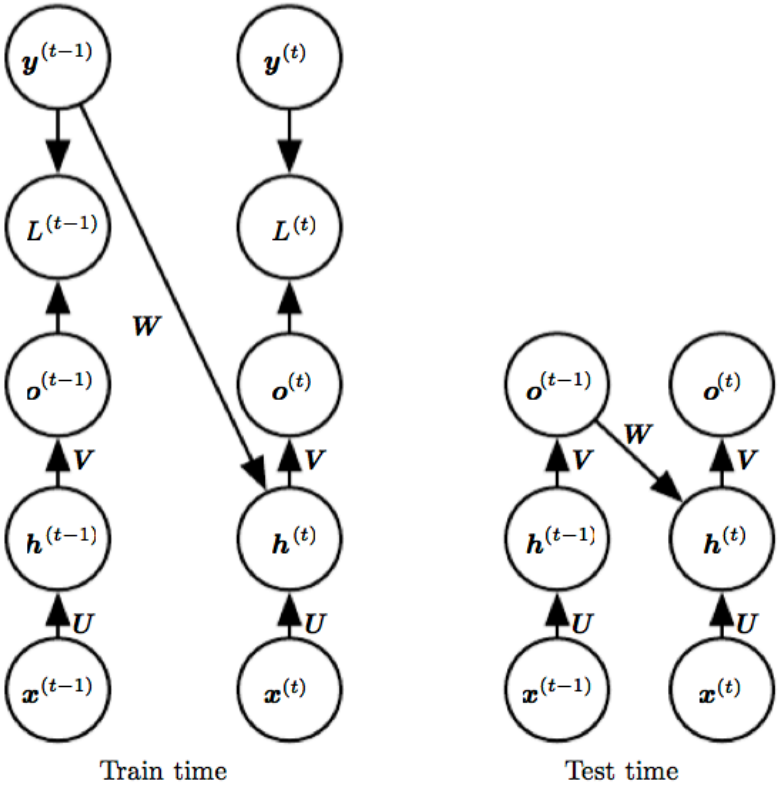
- **Single output at the end of the sequence**
- **Used to summarize a sequence and produce a fixed-size representation**
  - To use as input for further processing



(o: output, L: loss, y: target, W: weight)

Teacher forcing and Networks with output recurrence

- **During training the model receives the correct output $y^{(t)}$ as input at time $t + 1$**



Teacher forcing and Networks with output recurrence

- **Advantage**
  - Training can be parallelized with the gradient for each step computed in isolation

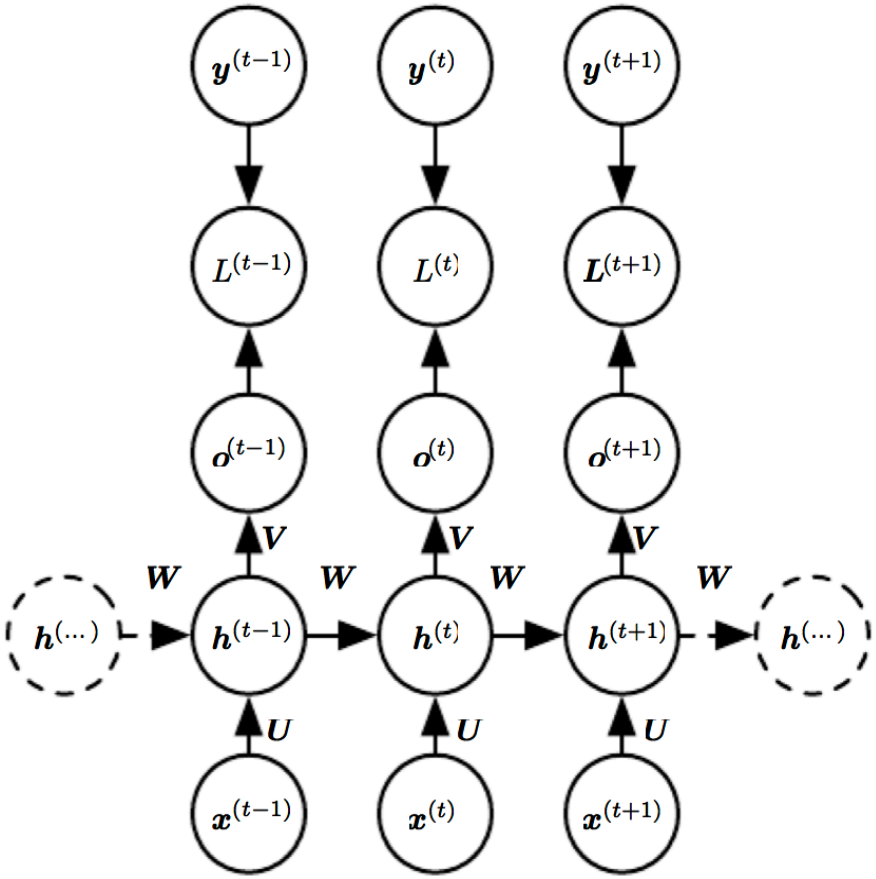
- **Disadvantage**
  - The kind of inputs that the network sees during training could be quite different from the kind of inputs that it will see at test time

Back propagation through time (BPTT)

- $h^{(t)} = tanh(Wh^{(t-1)} + Ux^{(t)})$
- $o^{(t)} = Vh^{(t)}$
- $L\left(\{x^{(1)}, \cdots, x^{(\tau)}\}, \{y^{(1)}, \cdots, y^{(\tau)}\}\right)$

U: weight from input to hidden layer
W: hidden-to-hidden connection weight
V: weight from hidden to output layer
t: time index $(1, 2, \cdots, \tau)$

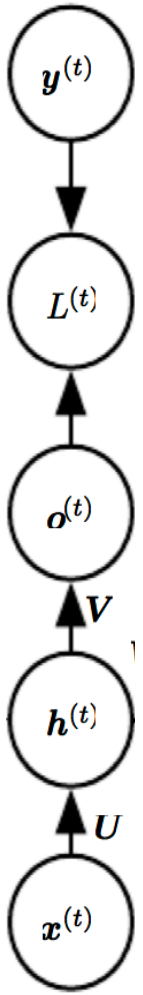
$x^{(t)}$: input layer
$h^{(t)}$: hidden layer
$o^{(t)}$: output layer
$L^{(t)}$: loss function



Back propagation through time (BPTT)

- **In feedforward neural network**
  - $h^{(t)} = tanh(Ux^{(t)})$
  - $o^{(t)} = Vh^{(t)}$
  - $L(\{x^{(1)}, \cdots, x^{(\tau)}\}, \{y^{(1)}, \cdots, y^{(\tau)}\})$

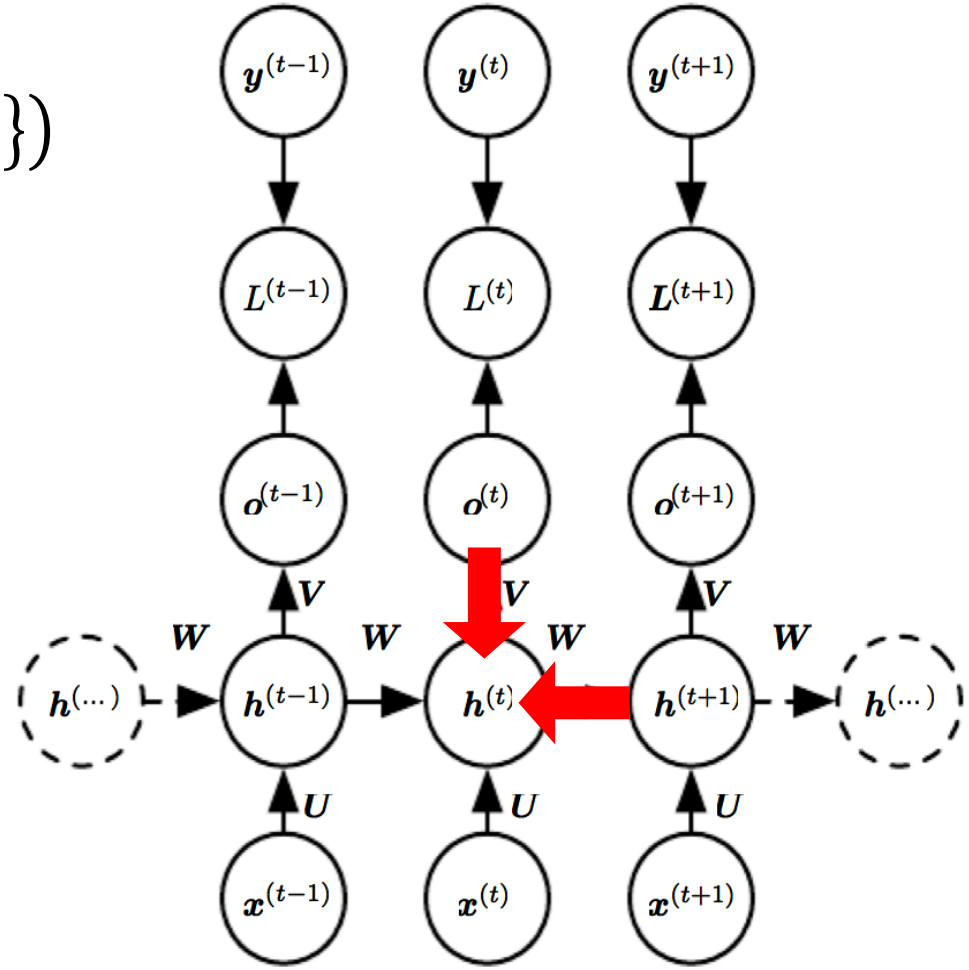
- $\nabla_{h^{(t)}} L = \frac{\partial L}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}}$
- $\nabla_V L = \frac{\partial L}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial V}$
- $\nabla_U L = \frac{\partial L}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial U}$



Back propagation through time (BPTT)

- $h^{(t)} = tanh(Wh^{(t-1)} + Ux^{(t)})$
- $o^{(t)} = Vh^{(t)}$
- $L\left(\{x^{(1)}, \cdots, x^{(\tau)}\}, \{y^{(1)}, \cdots, y^{(\tau)}\}\right)$

- $\nabla_{h^{(t)}} L$



Back propagation through time (BPTT)

- $\nabla_{h^{(\tau)}} L = \frac{\partial L}{\partial h^{(\tau)}} = \frac{\partial L}{\partial o^{(\tau)}} \frac{\partial o^{(\tau)}}{\partial h^{(\tau)}} = V^T \frac{\partial L}{\partial o^{(\tau)}} = V^T \nabla_{o^{(\tau)}} L$
- $\nabla_{h^{(t)}} L = \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right)^T \nabla_{h^{(t)}} L + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right)^T \nabla_{o^{(t)}} L$

$$\frac{\partial h^{(t+1)}}{\partial h^{(t)}} = \frac{\partial tanh\left(a^{(t+1)}\right)}{\partial h^{(t)}} = diag\left(1 - tanh^2\left(a^{(t+1)}\right)\right) \frac{\partial a^{(t+1)}}{\partial h^{(t)}} = W diag\left(1 - \left(h^{(t+1)}\right)^2\right)$$

$$= \left(W diag\left(1 - \left(h^{(t+1)}\right)^2\right)\right)^T \nabla_{h^{(t)}} L + V^T \nabla_{o^{(t)}} L$$

$$= W^T diag\left(1 - \left(h^{(t+1)}\right)^2\right) \nabla_{h^{(t)}} L + V^T \nabla_{o^{(t)}} L$$

Back propagation through time (BPTT)

- $\nabla_V L = \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}} \right) \nabla_V o_i^{(t)} = \sum_t \left(\nabla_{o^{(t)}} L \right) h^{(t)^T}$
- $\nabla_W L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{W^{(t)}} h_i^{(t)}$

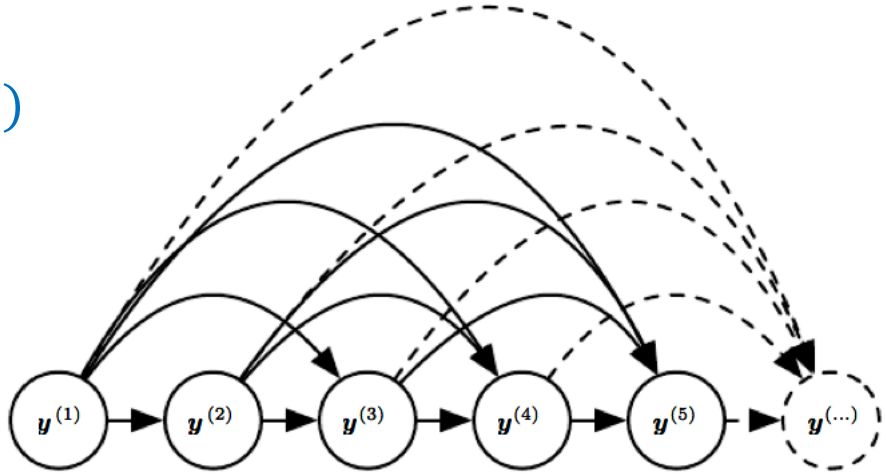
$$= \sum_t diag\left(1 - \left(h^{(t)}\right)^2\right) \left(\nabla_{h^{(t)}} L\right) h^{(t-1)^T}$$

- $\nabla_U L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{U^{(t)}} h_i^{(t)}$

$$= \sum_t diag\left(1 - \left(h^{(t)}\right)^2\right) \left(\nabla_{h^{(t)}} L\right) x^{(t)^T}$$

Recurrent Network as Directed Graphical Models

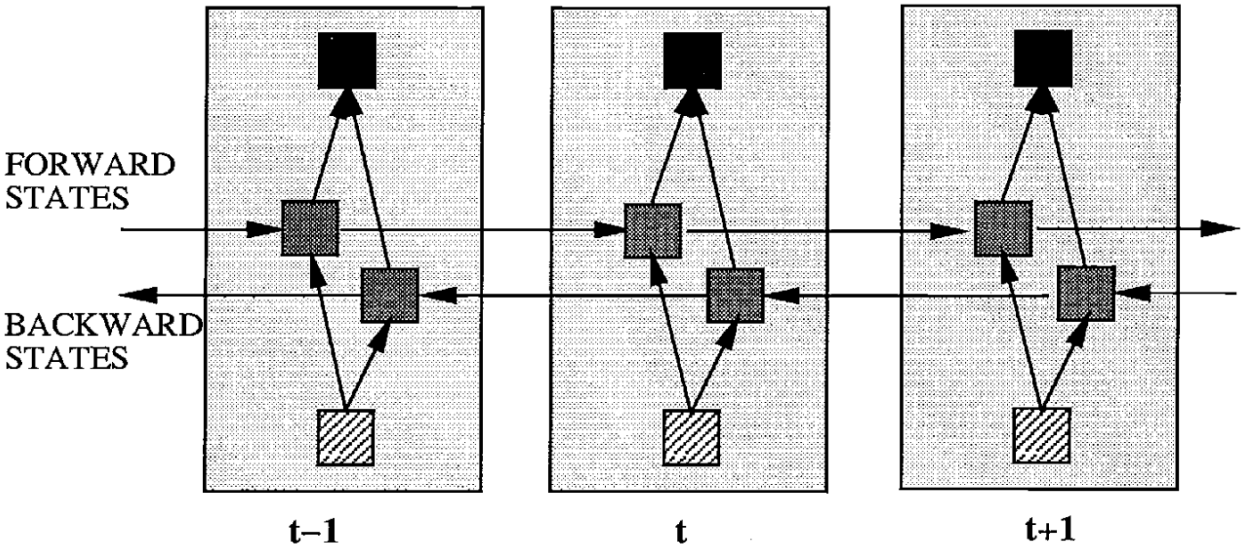
- **Fully connected graphical model**
  - Inefficient parametrization
  - $ex)\ \boldsymbol{h}^{(t)} = \boldsymbol{g}^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \cdots, \boldsymbol{x}^{(1)})$
- **The edge indicate which variables depend directly on other variables**



- **Efficient parametrization of the joint distribution**
  ex) $\boldsymbol{h}^{(t)} = \boldsymbol{g}^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \cdots, \boldsymbol{x}^{(1)}) = \boldsymbol{f}\left(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}\right)$

Bidirectional RNNs

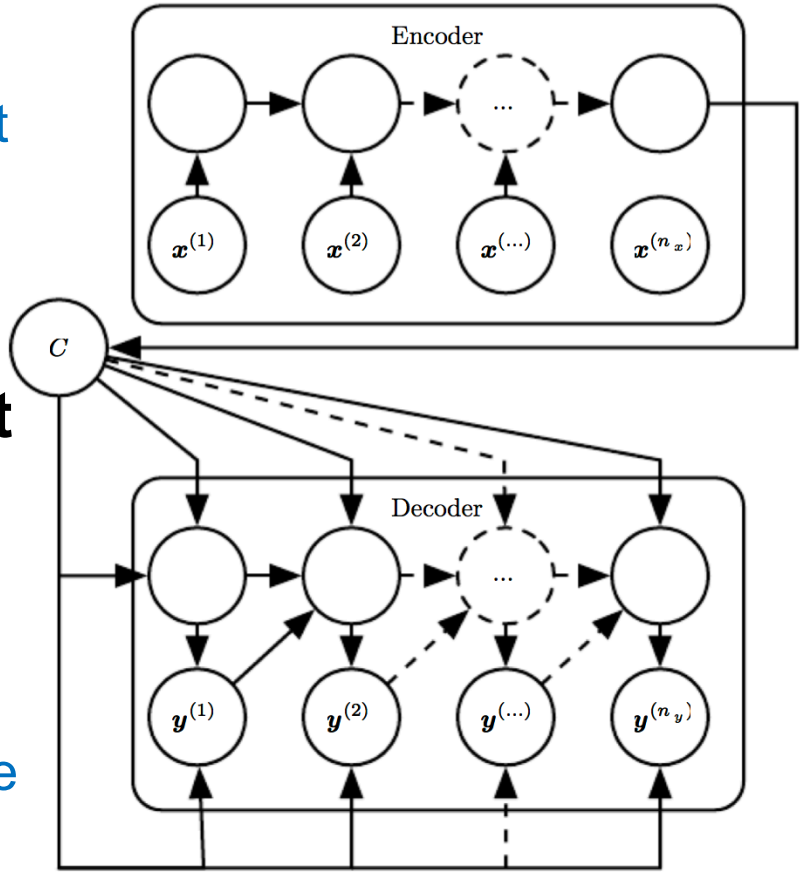
- maintains two hidden layers, one for the forward states and the other for the backward states
- consumes twice as much memory space for its weight and bias parameters
- Output units with a relevant summary of the past in the forward states and the future in the backward states



http://solarisailab.com/archives/1515

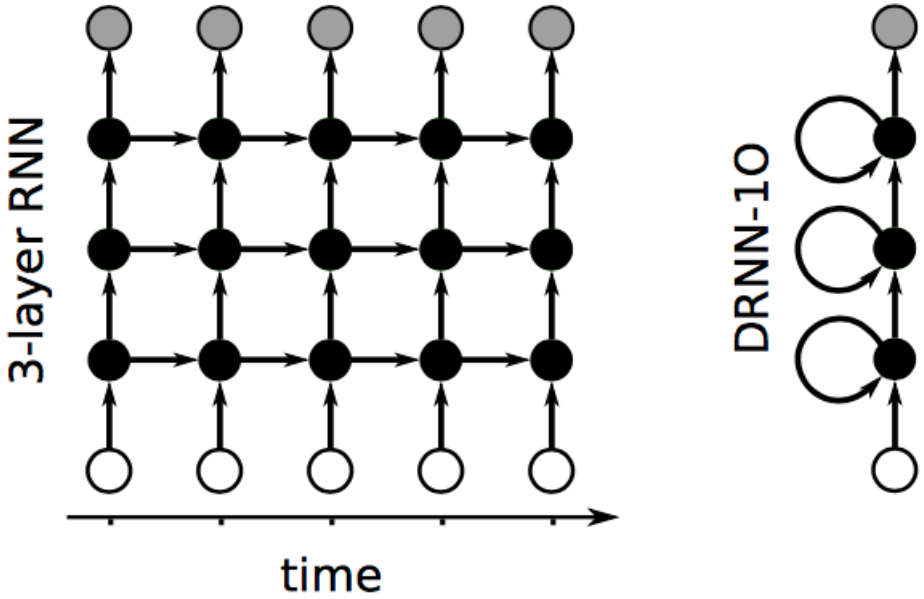
Encoder-Decoder Sequence-to-Sequence Architecture

- **To map an input sequence to an output sequence which is not necessarily of the same length**
- **Encoder (or input RNN)**
  - process input sequence
  - Emit context C that summarize input
- **Decoder (or output RNN)**
  - Conditioned on fixed-length vector to generate the output sequence
- **The length of input and output sequence can vary from each other**
- **Limitation**
  - Dimension of context C can be too small to summarize a long sequence



Deep Recurrent Networks

- **3 parts**
  - From the input to the hidden state
  - From the previous hidden state to the next hidden state
  - From the hidden state to the output
- **Increase the number of nonlinear steps the gradient at back propagation**
  - difficult to train the model to capture long-term dependency



Hermans, M., & Schrauwen, B. (2013). Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems* (pp. 190-198).

Recursive Neural Networks

- **Another generalization of recurrent neural network**
- **Applied to processing data structures as input to neural nets in natural language processing as well as computer vision**
- **Advantage**
  - For a sequence of same length τ, the depth can be reduced from τ to $O(\log \tau)$
- **How to best structure the tree**