Deep Learning Seminar
Chapter 10. Sequence Modeling: Recurrent and Recursive Net
- Part 2 -

Hyun-Lim YANG

Department of Information and Communication Engineering

DGIST

2017.11.22

Contents

Chapter 10. Sequence Modeling

- **Part 1**
  - **10.1** Unfolding Computational Graphs
  - **10.2** Recurrent Neural Networks
  - **10.3** Bidirectional RNNs
  - **10.4** Encoder-Decoder Sequence-to-Sequence Architecture
  - **10.5** Deep Recurrent Networks
  - **10.6** Recursive Neural Networks
- **Part 2**
  - **10.7** The challenge of Long-term dependencies
  - **10.8** Echo State Networks
  - **10.9** Leaky Units and Other strategies for Multiple Time Scales
  - **10.10** The Long Short-Term Memory and Other Gated RNNs
  - **10.11** Optimization for Long-Term Dependencies
  - **10.12** Explicit Memory

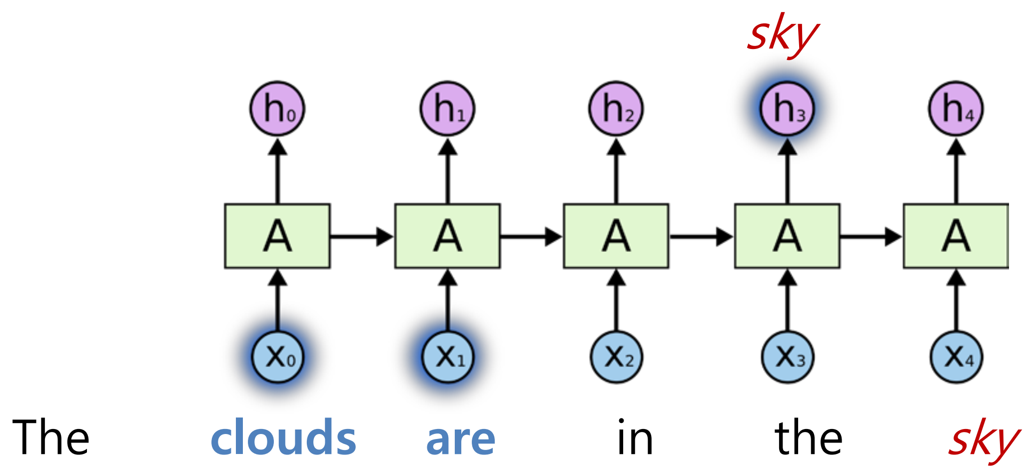
Chapter 10. Sequence Modeling

- Part 1
  - 10.1 Unfolding Computational Graphs
  - 10.2 Recurrent Neural Networks
  - 10.3 Bidirectional RNNs
  - 10.4 Encoder-Decoder Sequence-to-Sequence Architecture
  - 10.5 Deep Recurrent Networks
  - 10.6 Recursive Neural Networks
- **Part 2**
  - **10.7** The challenge of Long-term dependencies
  - **10.8** Echo State Networks
  - **10.9** Leaky Units and Other strategies for Multiple Time Scales
  - **10.10** The Long Short-Term Memory and Other Gated RNNs
  - **10.11** Optimization for Long-Term Dependencies
  - **10.12** Explicit Memory

The Challenge of Long-term dependencies

Long-Term Dependencies

- **RNN might be able to connect previous information to present task**



- **But, can they?**

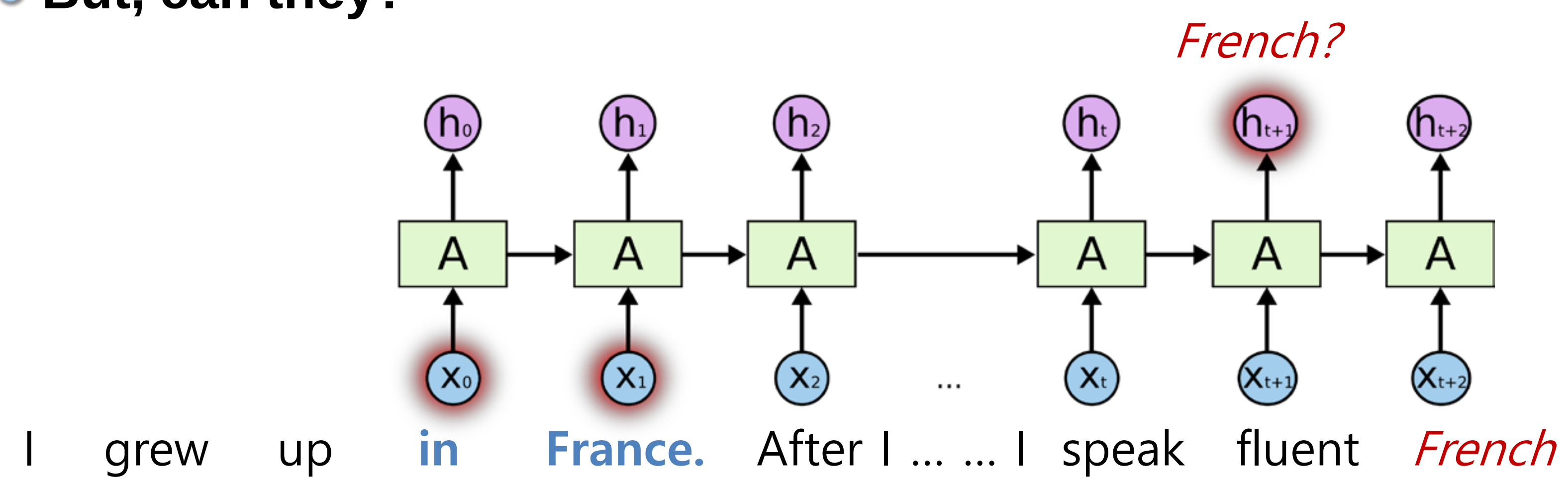


Image from colah's blog on github (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

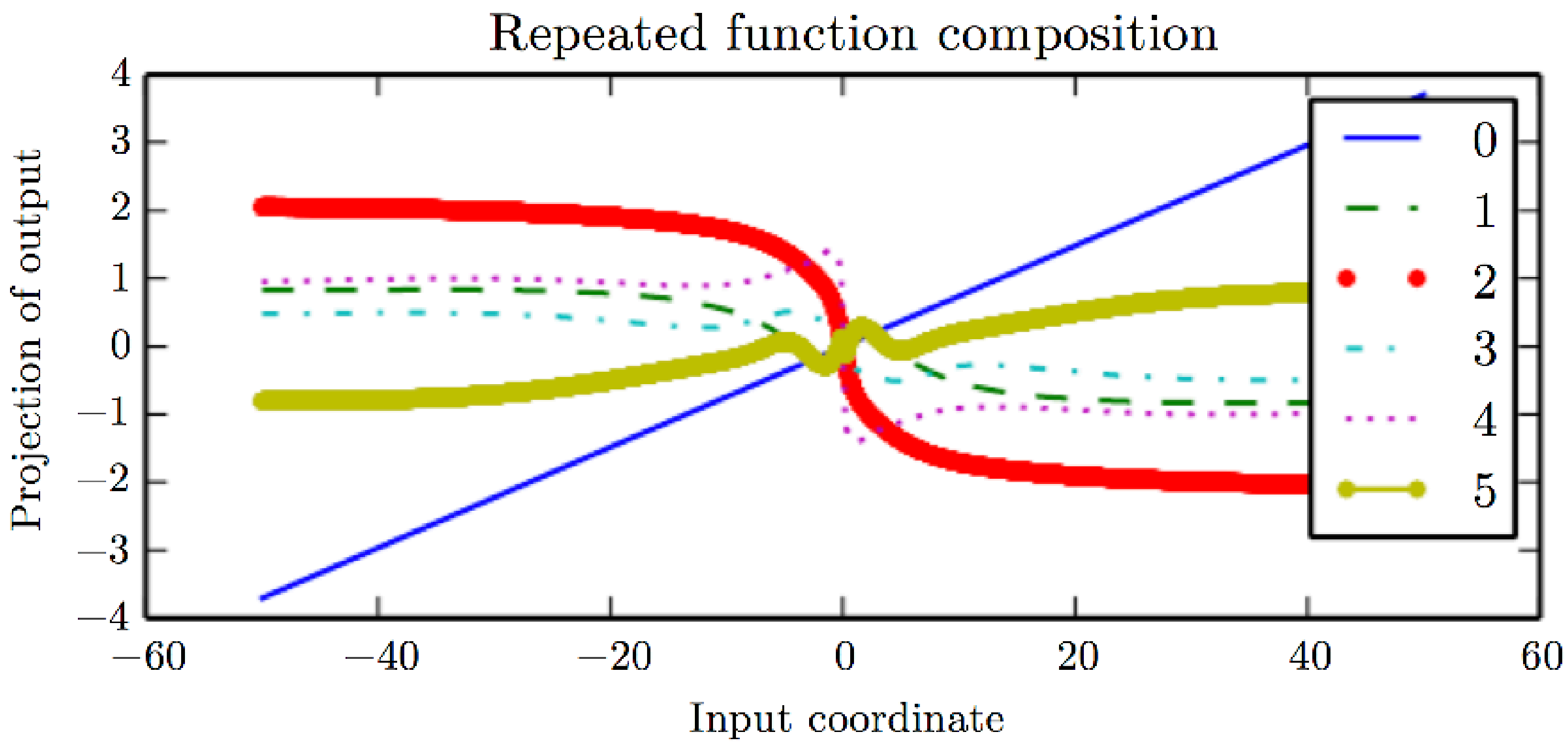
Repeatedly multiplying weight

- **The simple case**
  - Let model's parameter matrix W
  - In recurrent case, k step is equivalent to W^k
  - Suppose that W has an eigen-decomposition

$$W = Vdiag(\lambda)V^{-1}$$

  - $W^k = \left(Vdiag(\lambda)V^{-1}\right)^k = Vdiag(\lambda)^k V^{-1}$
  - Any eigenvalue λ_i that are not near an absolute value of 1 will either explode or vanishing
- **In general models**
  - $h^t = W^T h^{(t-1)} = (W^t)^T h^{(0)} = Q^T \Lambda^t Q h^{(0)}$
  - Long-term dependencies arises from the exponentially smaller weights given to long-term interactions compared to short-term ones.

Repeatedly multiplying weight

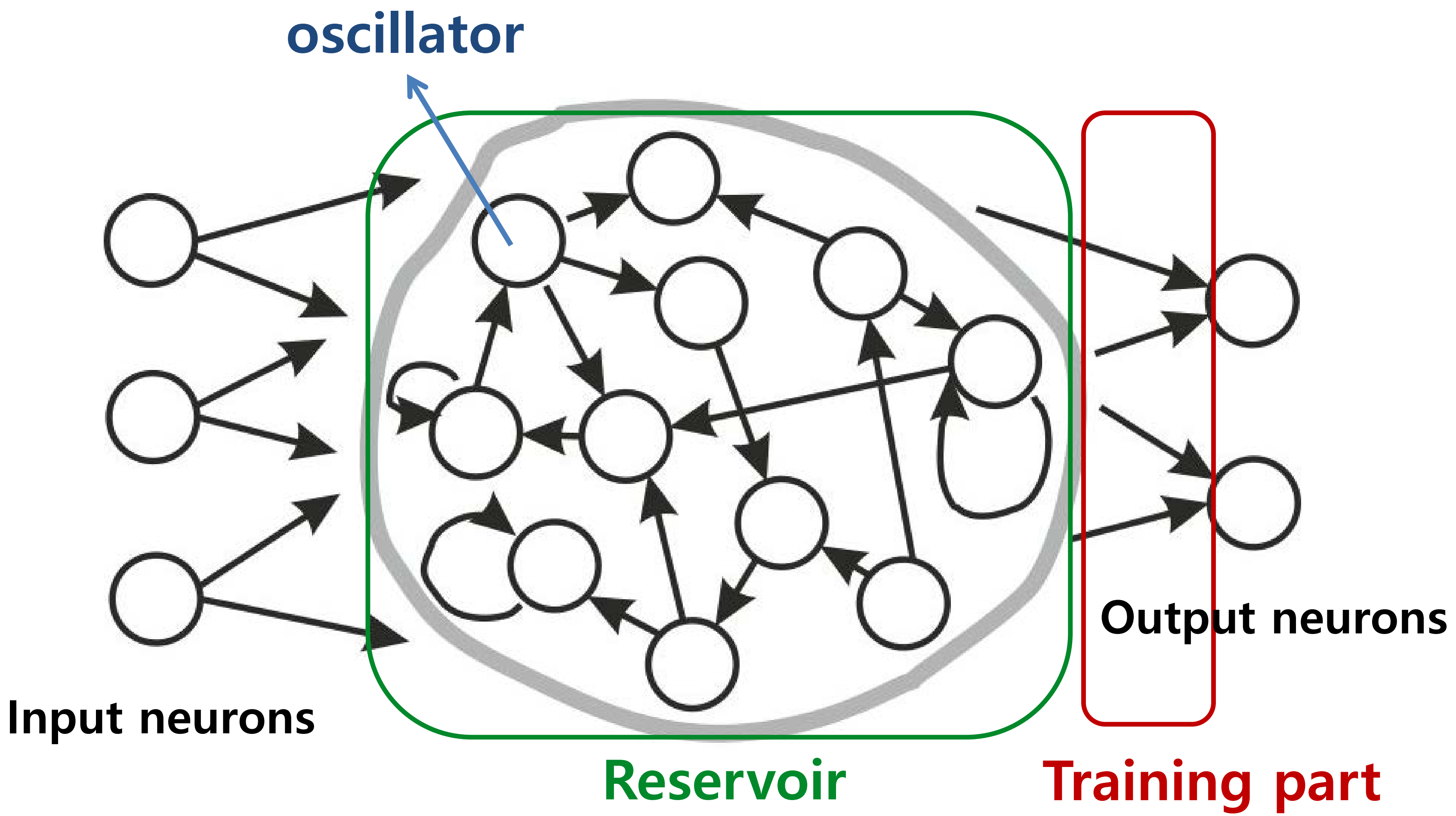


Echo State Network

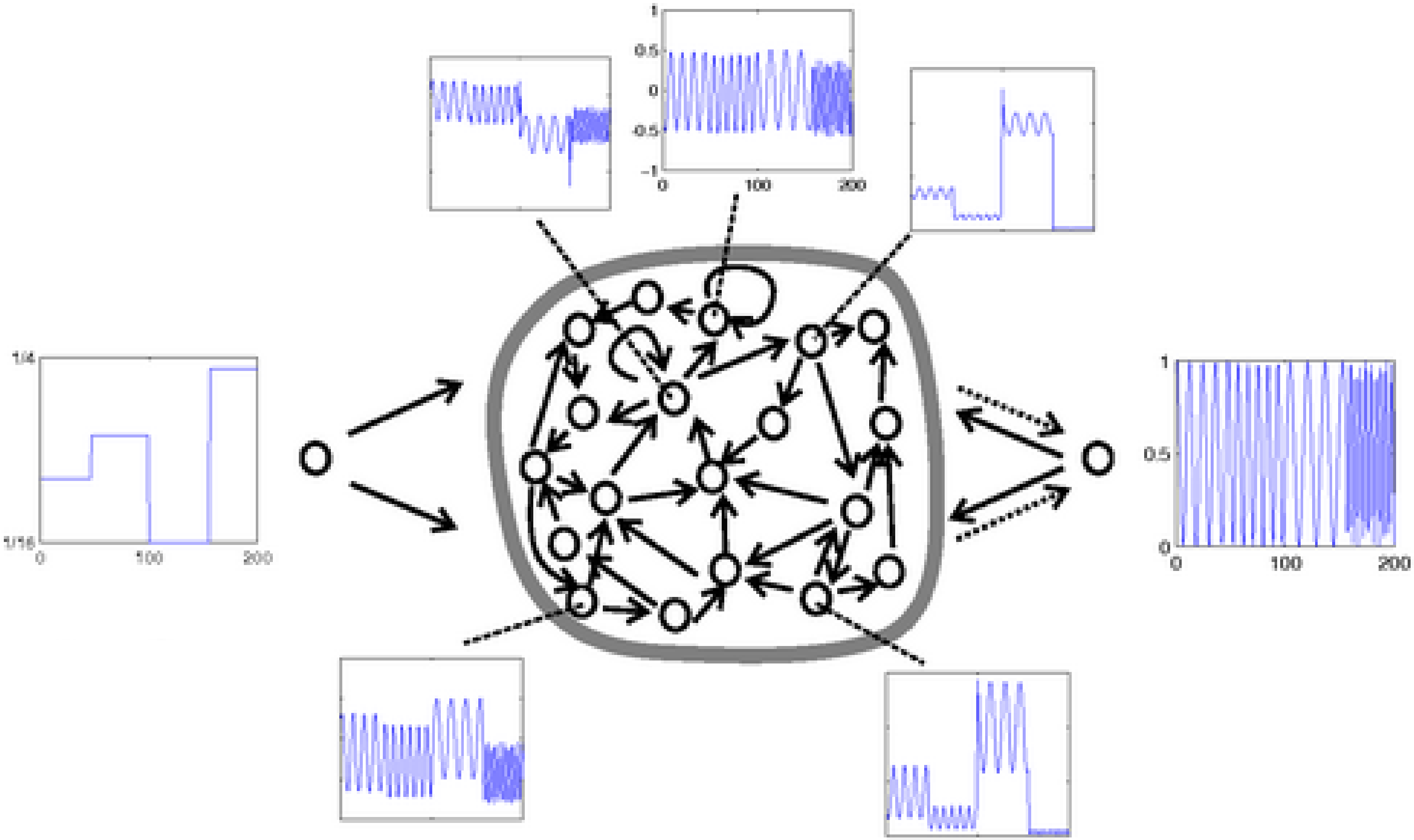
Echo State Network

- Simple trick to make RNN much easier to learn
- Initialize the connections in the RNN randomly in such a way that it has big reservoir of coupled oscillators
- Do not consider input to hidden or hidden to hidden learn
- The only thing have learn is how to couple the output to the oscillators (state to output)
- In the end, it just fit in a linear model
- Similar with SVM
  - Reservoir can be considered as kernel
  - State to output connection is classifier

Echo State Network



Transforming sine wave example



Pros and Cons

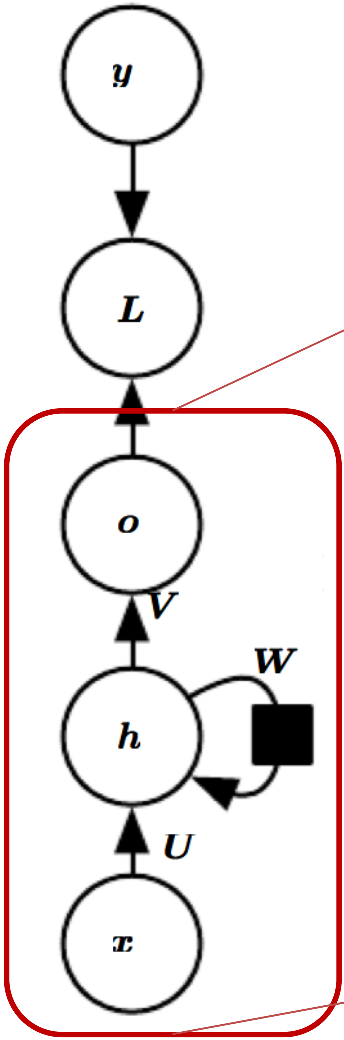
- **Pros**
  - ESN can be **trained very fast** because they just fit linear model
  - We can do many experiments
  - ESN can do impressive modeling of 1-dimensional time-series

- **Cons**
  - ESN **need many more units in reservoir** for more complex problems
  - Initialization is very important to performance

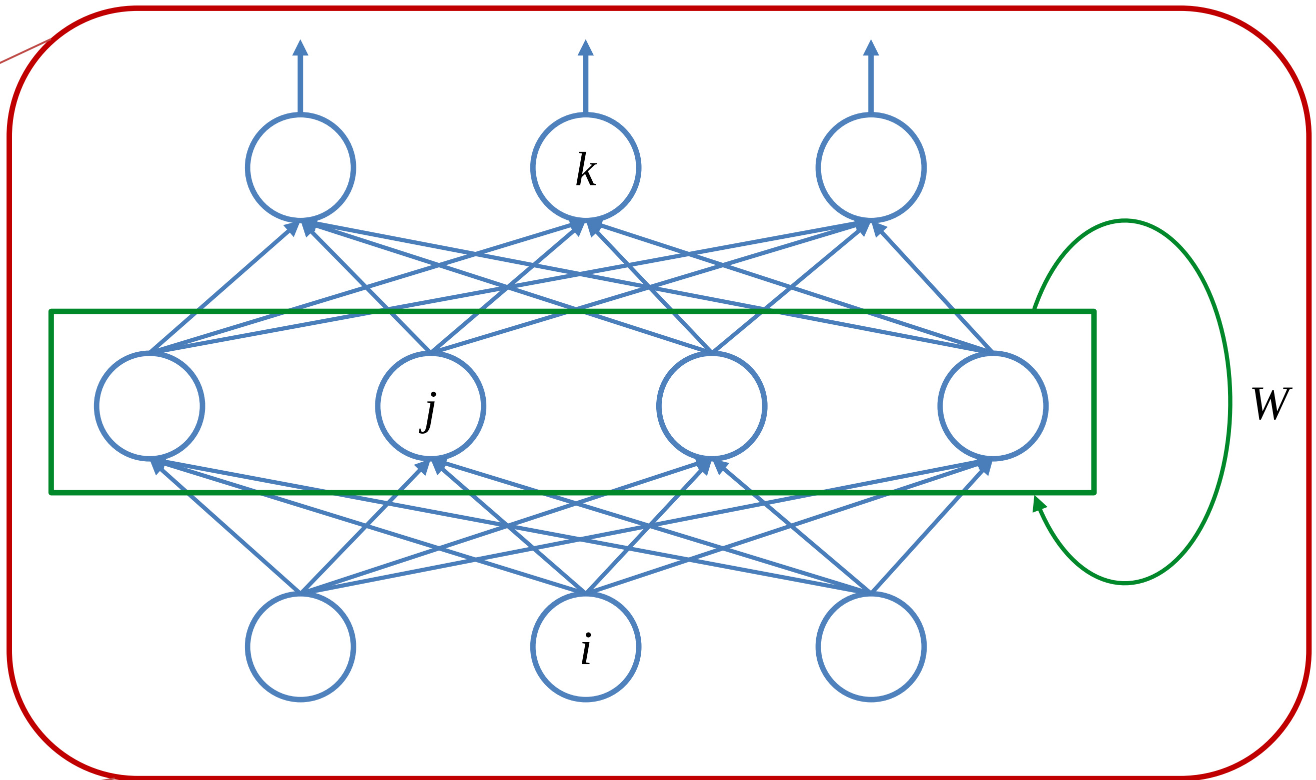
- **Applications**
  - ESN can be used for initialization of other networks

Long-Short Term Memory

RNN Review



$$Y^t = [\,---vector\ representation\ ---]$$



$$X^t = [\,---vector\ representation\ ---]$$

RNN Review

$$Y^t = [\,---vector\ representation\ ---]$$

output layer ($\boldsymbol{k}$)

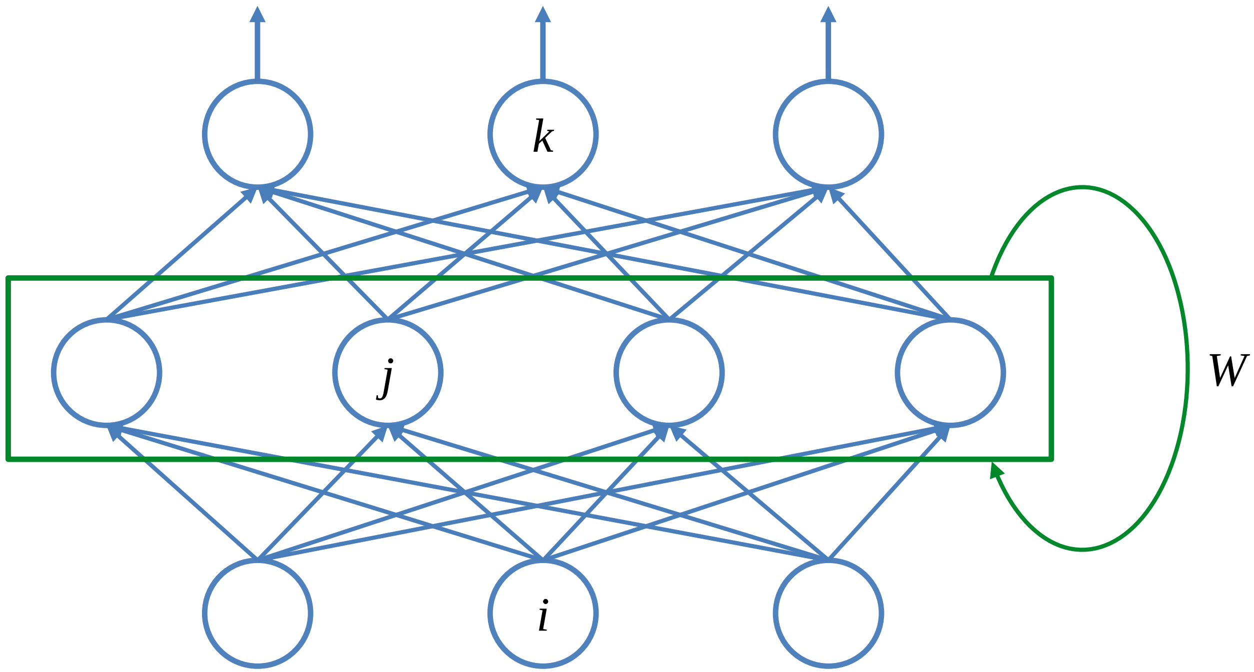
$W^{out} = (w_{kj}^{out})$

hidden layer ($\boldsymbol{j}$)

$W = (w_{jj'})$

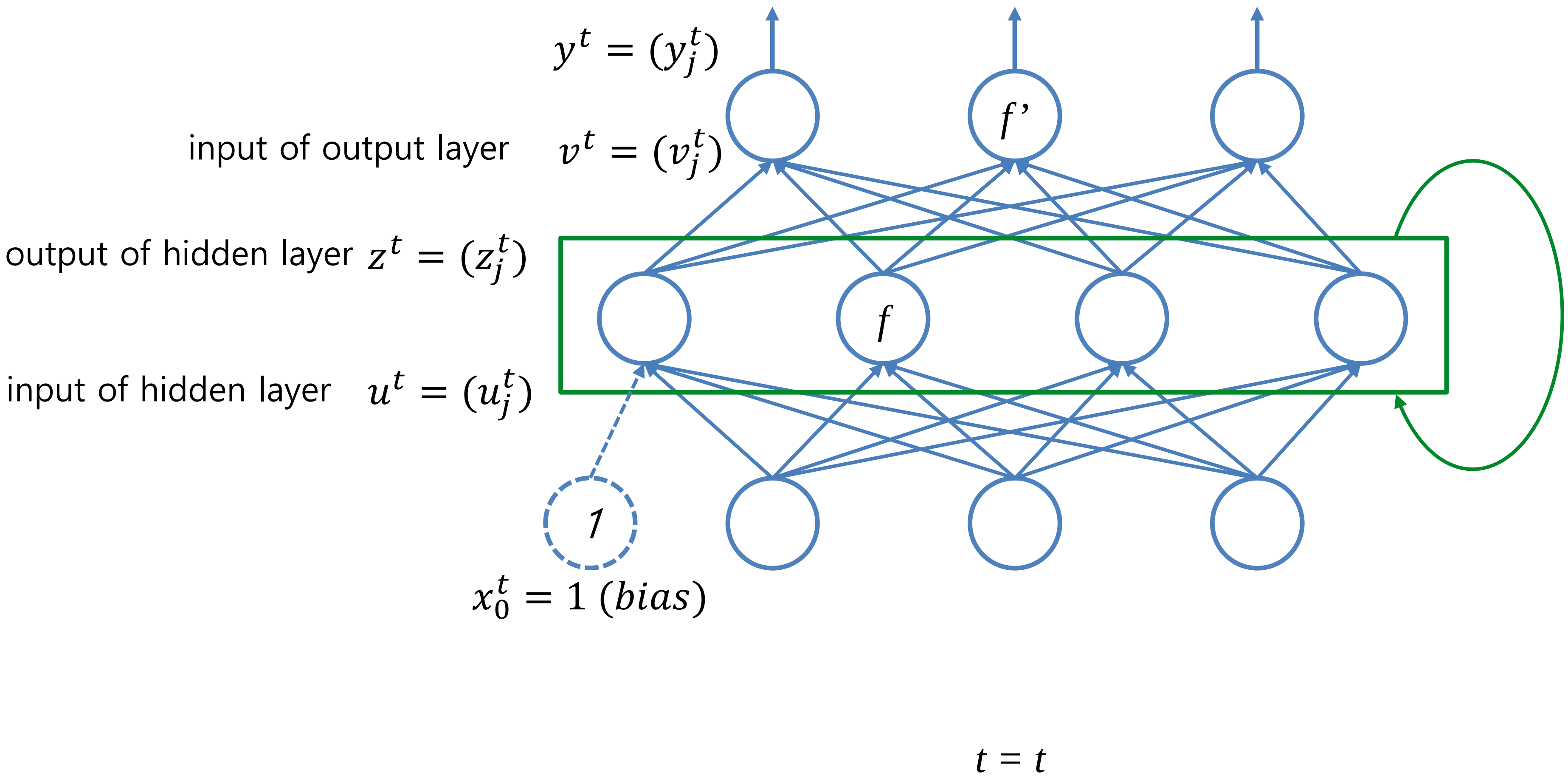
$W^{in} = (w_{ji}^{in})$

input layer ($\boldsymbol{i}$)



$$X^t = [\,---vector\ representation\ ---]$$

RNN Review

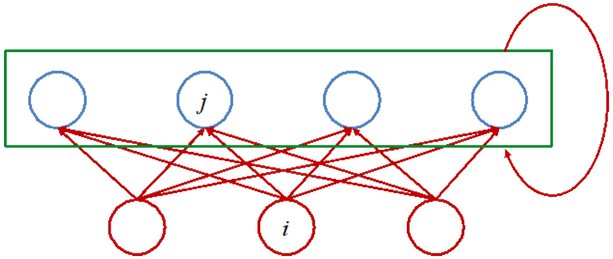


$y^t = (y_j^t)$
input of output layer $v^t = (v_j^t)$
f'
output of hidden layer $z^t = (z_j^t)$
f
input of hidden layer $u^t = (u_j^t)$
1
$x_0^t = 1\ (bias)$
$t = t$

RNN Review - Forward propagation

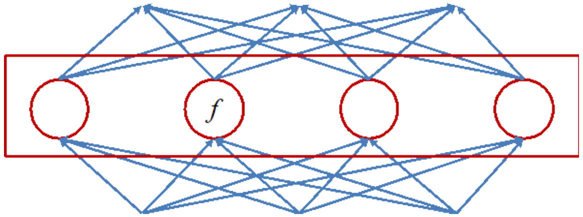
- **input of hidden layer**

$$u_j^t = \sum_i w_{ji}^{(in)} x_i^t + \sum_{j} w_{jj'} z_{j'}^{t-1}$$



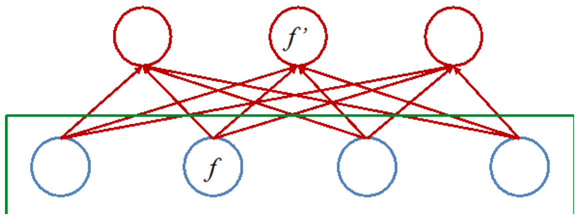
- **output of hidden layer**

$$z_j^t = f(u_j^t) \qquad z^t = f\left(W^{in} X^t + W z^{t-1}\right)$$



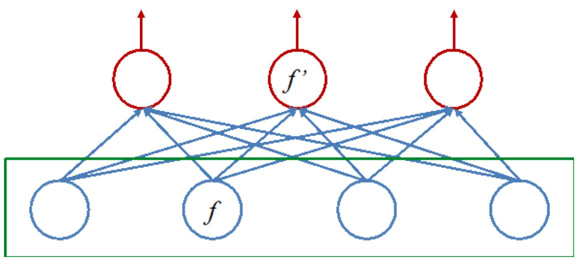
- **input of output layer**

$$v_k^t = \sum_j w_{kj}^{(out)} z_j^t$$



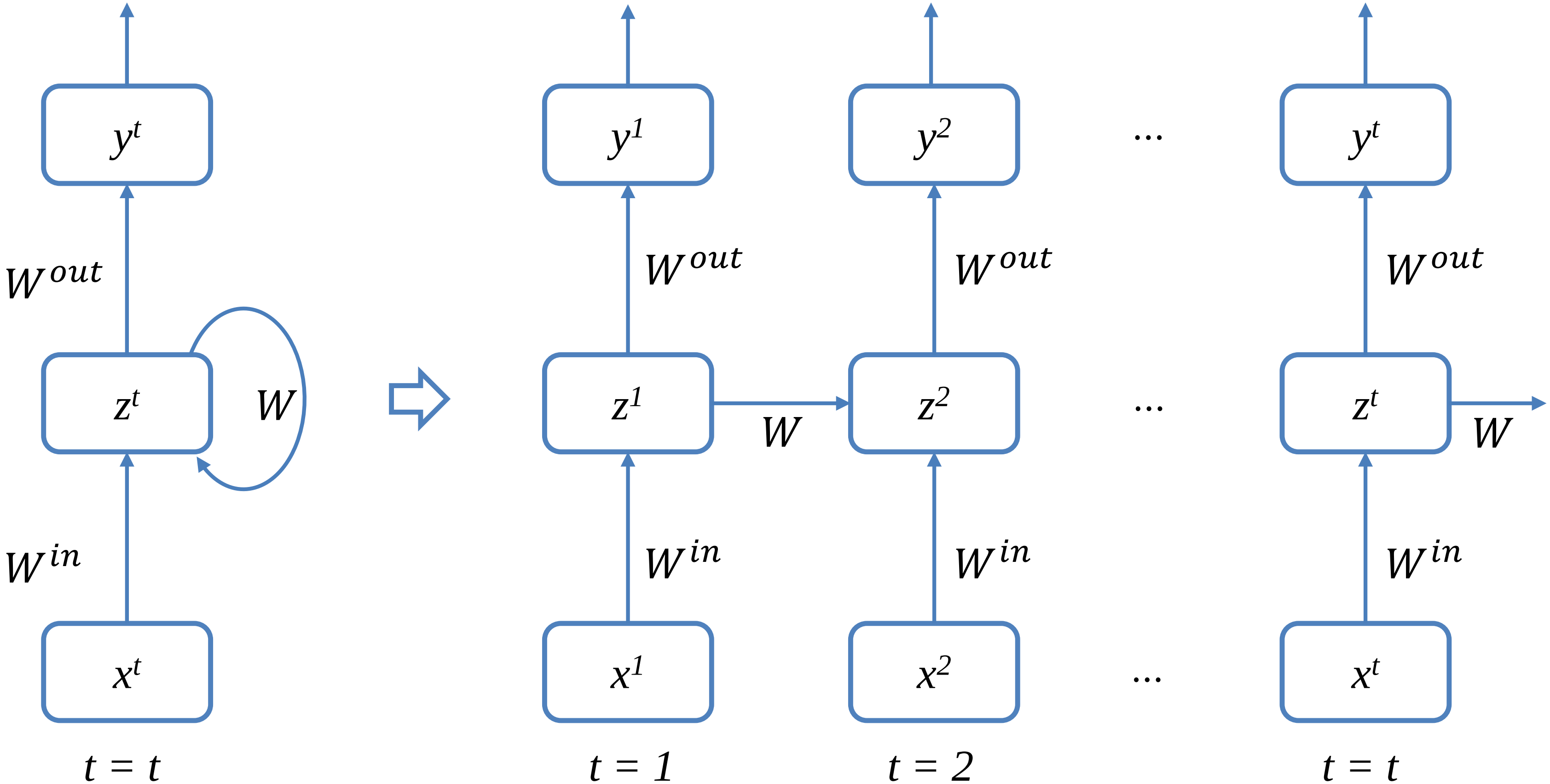
- **output of output layer**

$$y^t = f^{out}(v^t) = f^{out}\left(W^{out} z^t\right)$$



Back Propagation Through Time(BPTT) (1)

- **Faster and more simple than RTRL(RealTime Recurrent Learning)**
- **Deploy the RNN with time direction (like Feedforward NN)**
  - assumes that each layer is separated by time

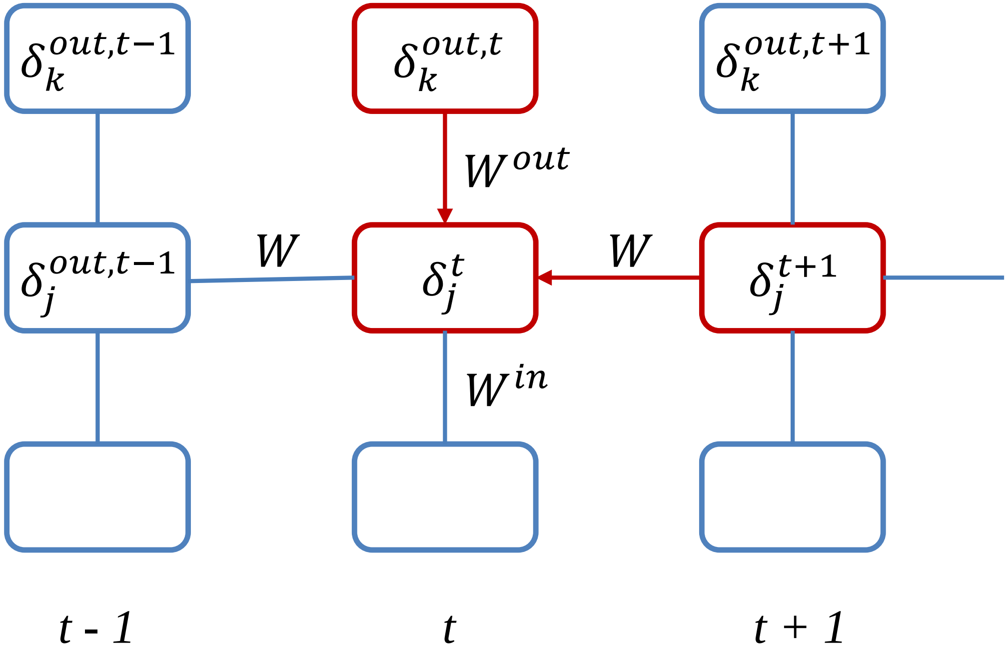


Back Propagation Through Time(BPTT) (2)

- In Feedforward NN, we can find *delta* at layer *l* as :

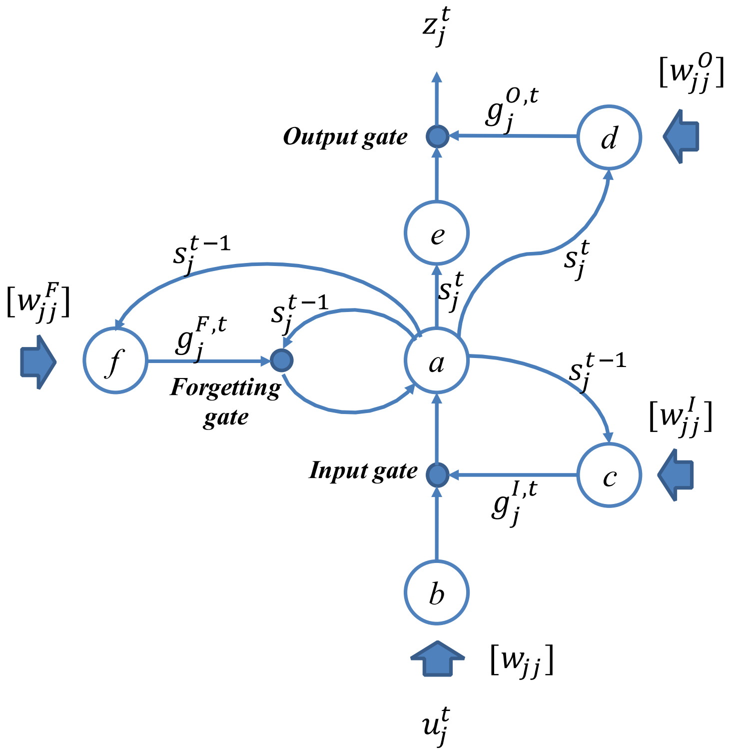
$$\delta_j^l \equiv \frac{\partial E}{\partial u_j^l} \qquad \delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} f'(u_j^l)$$

- In RNN, we *delta* is affected by :



LSTM (Long Short-Term Memory)

- **Designed to overcome RNN's limitation**
  - in practice, pure RNN dose not contain the information of whole inputs
  - pure RNN memorize the data up to (*t-10*)



a. memory cell

b. a RNN's node

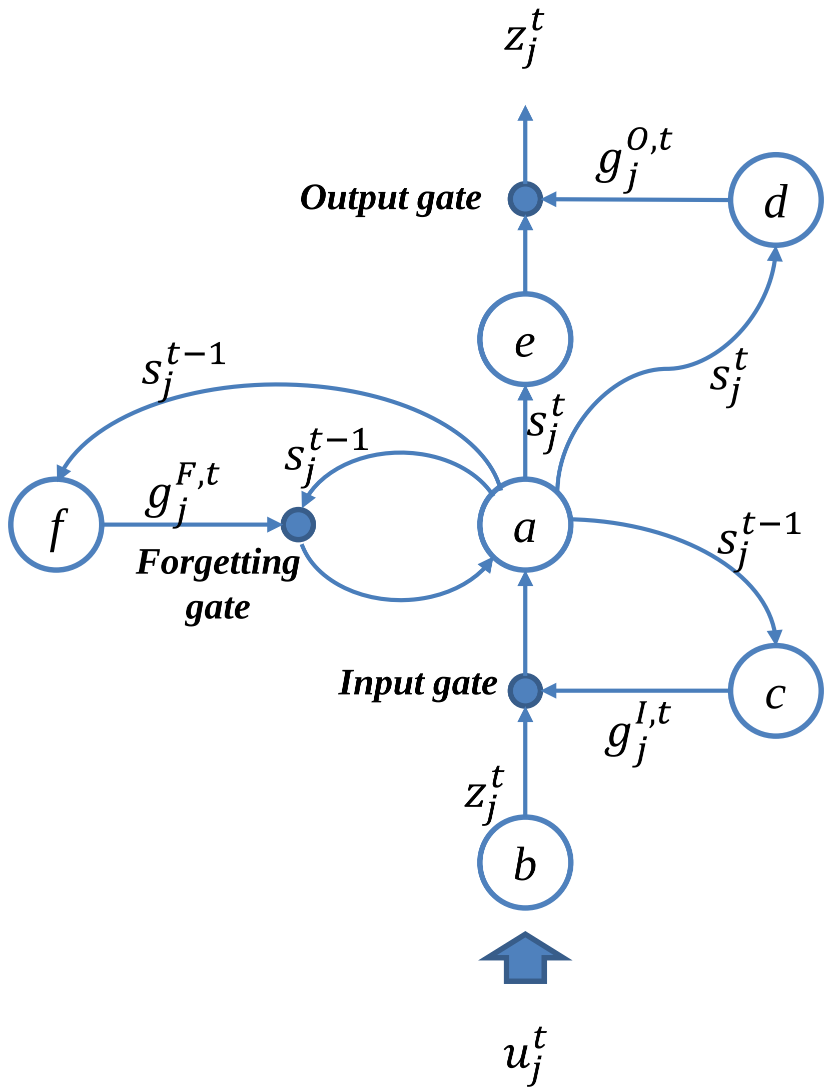
c. node for input gate

d. node for output gate

e. node for activation function of memory cell output

f. node for forgetting gate

LSTM gate operation



output gate :

$g_j^{O,t} \in [0,1]$

$g_j^{O,t} \cdot s_j^t$

forgetting gate :

$g_j^{F,t} \in [0,1]$

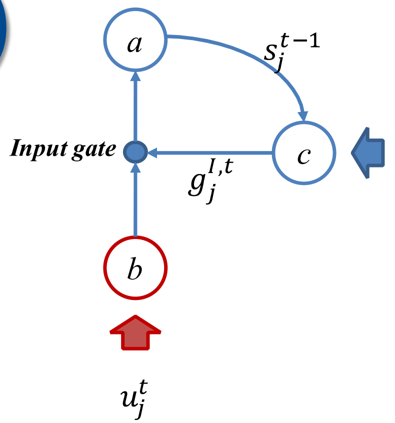
$g_j^{F,t} \cdot s_j^{t-1}$

input gate :

$g_j^{I,t} \in [0,1]$

$g_j^{I,t} \cdot z_j^t$

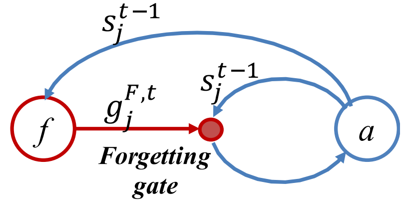
LSTM forward propagation (1)



- **before node *b*,**

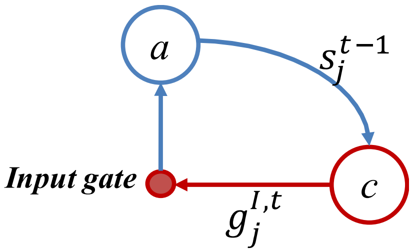
$$u_j^t = \sum_i w_{ji}^{(in)} x_i^t + \sum_j w_{jj} z_{j}^{t\text{-}1}$$

- **from node f,**



$$g_j^{F,t} = f_f(u_j^{F,t}) = f_f(\sum_i w_{ji}^{F,in} x_i^t + \sum_{j'} w_{jj'}^F z_{j'}^{t-1} + w_j^F s_j^{t-1})$$

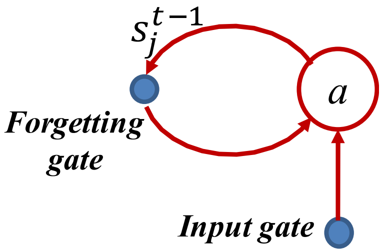
- **from node c,**



$$g_j^{I,t} = f_c(u_j^{I,t}) = f_c(\sum_i w_{ji}^{I,in} x_i^t + \sum_{j'} w_{jj'}^I z_{j'}^{t-1} + w_j^I s_j^{t-1})$$

- **so, memory cell a is:**

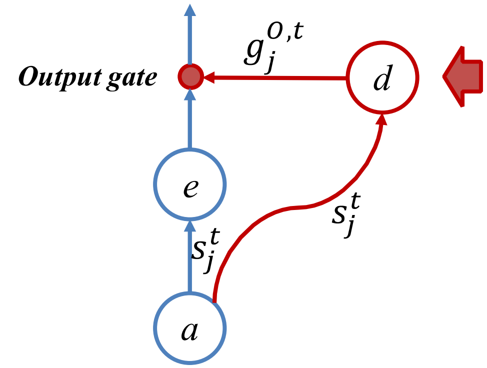
$$s_j^t = g_j^{F,t} s_j^{t-1} + g_j^{I,t} f_b(u_j^t)$$



LSTM forward propagation (2)

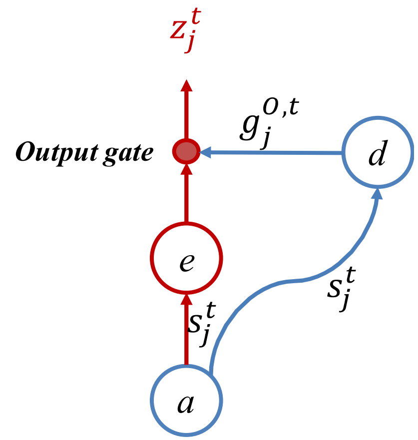
- from node d,

$$g_j^{O,t} = f_d\left(u_j^{O,t}\right) = f_d\left(\sum_i w_{ji}^{O,in} x_i^t + \sum_{j'} w_{jj'}^{O} z_{j'}^{t-1} + w_j^{O} s_j^t\right)$$



- so the output z_j^t *is :*

$$z_j^t = g_j^{O,t} f_e(s_j^t)$$



- usually, g is activation function with *Logistic Sigmoid Func.*

LSTM Back propagation (1)

- **In general feedforward NN structure,**

$$\delta_j^l = \sum_k \delta_k^{l+1} \frac{\partial u_k^{l+1}}{\partial u_j^l}$$

- **In LSTM, $\boldsymbol{z_j^t}$ is the only value which affected by external node**

$$v_k^t = \sum_j w_{kj}^{out} z_j^t$$

- **Since $\boldsymbol{z_j^t}$ is as follows,**

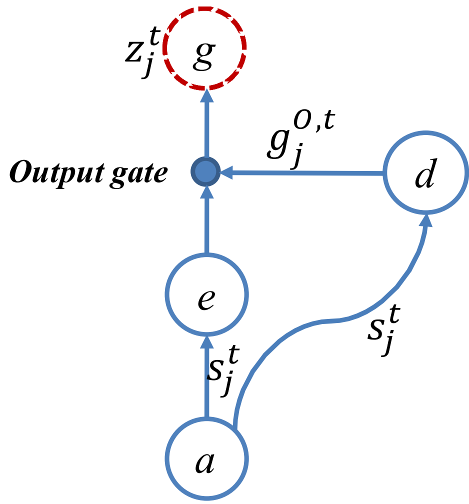
$$z_j^t = g_i^{O,t} f_e(s_j^t)$$

- **so, gradient of $\boldsymbol{v_k^t}$ is:**

$$\frac{\partial v_k^t}{\partial u_j^{O,t}} = w_{kj}^{out} f'(u_j^{O,t}) f_e(s_j^t)$$

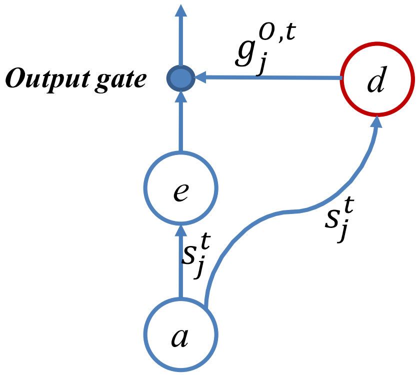
LSTM Back propagation (2)

- **It means, gradient of temporal node g is :**



$$\frac{\partial v_k^t}{\partial u_j^{O,t}} = w_{kj}^{out} f'(u_j^{O,t}) f_e(s_j^t)$$

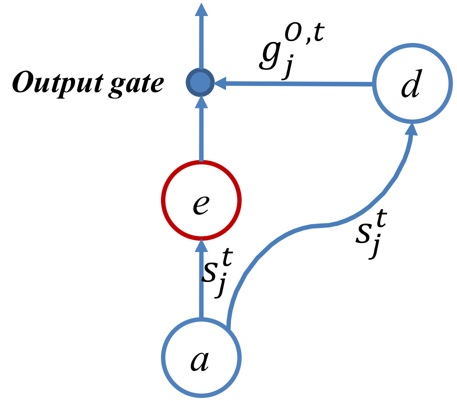
- **At node d, gradient is :**



$$\varepsilon_j^t = \sum_k w_{kj}^{out} \delta_k^{out,t} + \sum_{j'} w_{jj'} \delta_j^{t+1}$$

$$\delta_j^{O,t} = f'(u_j^{O,t}) f_d(s_j^t) \varepsilon_j^t$$

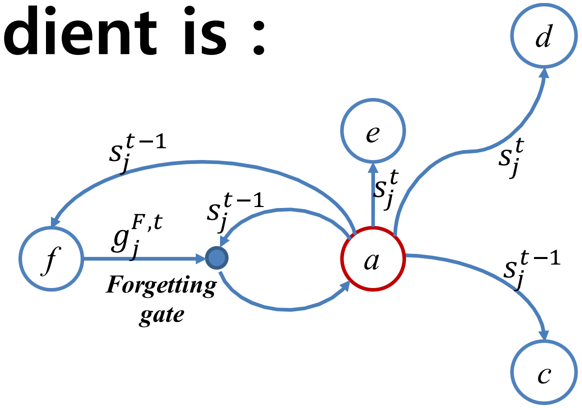
- **At node e, gradient is :**



$$\widetilde{\delta_j^t} = g_j^{O,t} f_d'(s_j^t) \varepsilon_j^t$$

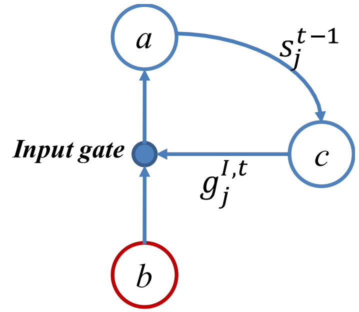
LSTM Back propagation (3)

- **At node *a*, it has 5 connection, and gradient is :**



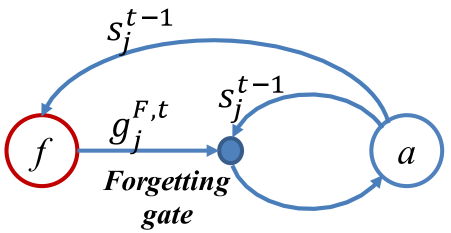
$$\delta_j^{cell,t} = \widetilde{\delta_j^t} + g_j^{F,t-1}\delta_j^{cell,t+1} + w_j^F\delta_j^{F,t+1} + w_j^I\delta_j^{I,t+1} + w_j^O\delta_j^{O,t}$$

- **At node *b*, gradient is :**



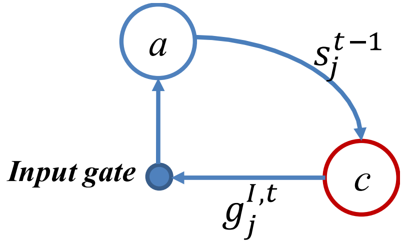
$$\delta_j^t = g_j^{I,y} f_c'(u_j^t)\delta_j^{cell,t}$$

- **At node *f*, gradient is :**



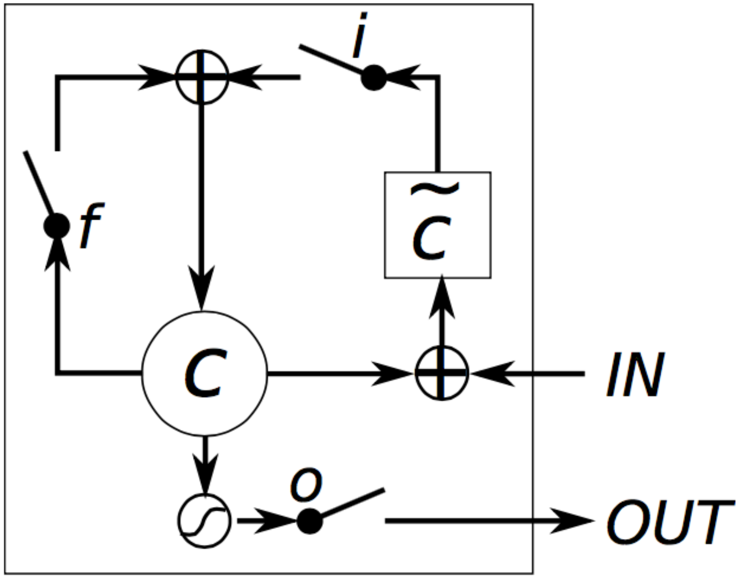
$$\delta_j^{F,t} = f'(u_j^{F,t})s_j^{t-1}\delta_j^{cell,t}$$

- **At node *c*, gradient is :**

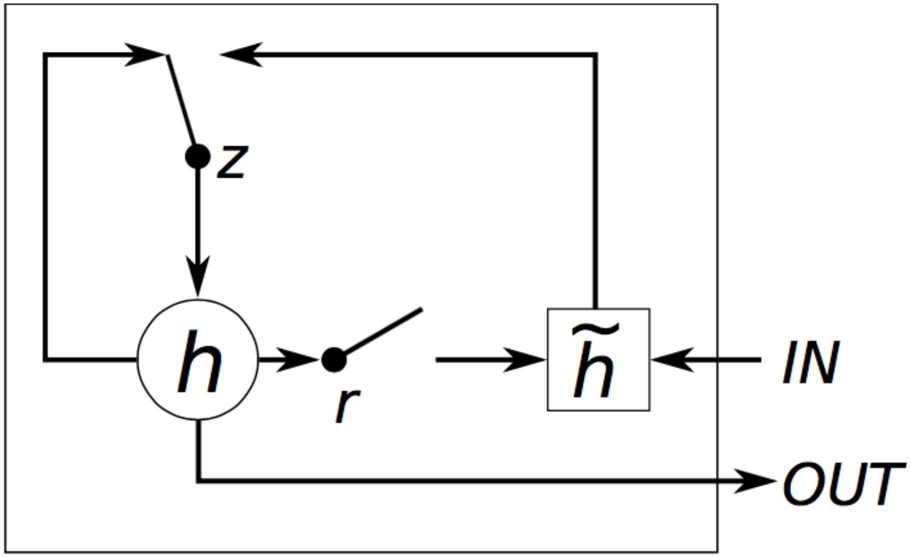


$$\delta_j^{I,t} = f'(u_j^{I,t})f(u_j^t)\delta_j^{cell,t}$$

Other Method: GRU (Gated Recurrent Unit)



LSTM Gating



GRU Gating

- More simpler version than LSTM
  - Use only 2 gate
- Both method have pros and cons

Image form, Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." (2014)

Optimization for Long-Term Dependencies

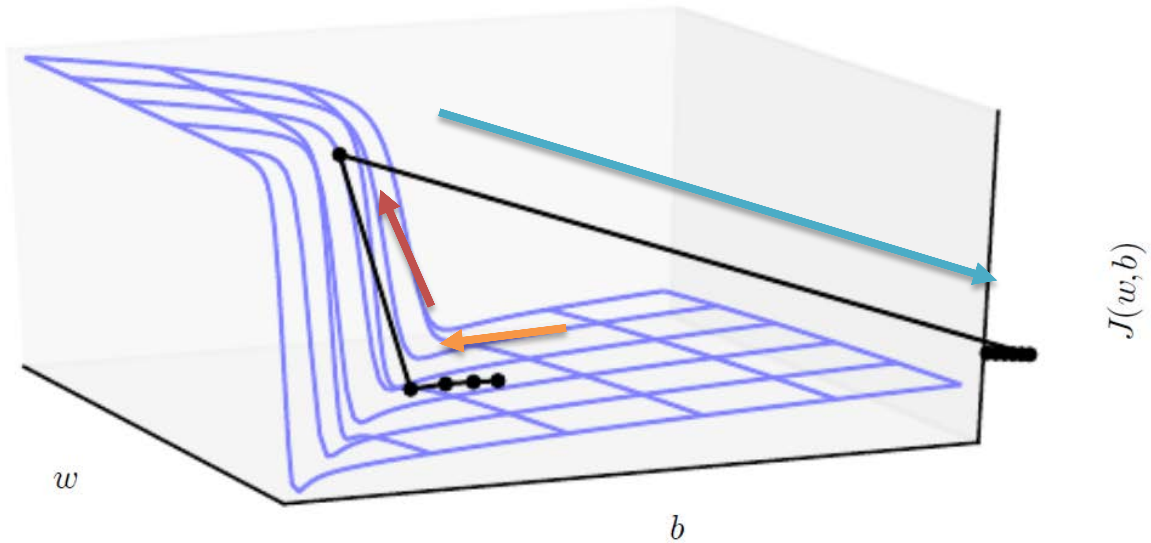
Two major issues in LTD optimizations

- **Gradient Exploding**
  - **Deal with Gradient clipping**

- **Gradient Vanishing**
  - **LSTM**

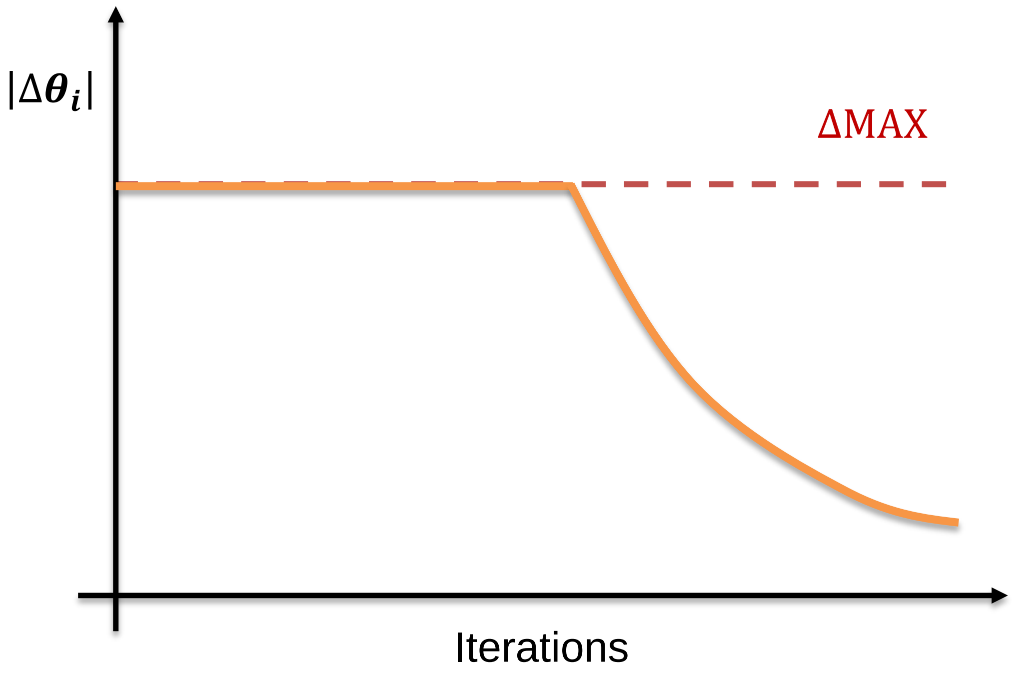
Cliffs and Exploding Gradient

- **Cliff is also a common issue in such as those computed by a recurrent net over many time steps**
  - resulting from the multiplication of several parameters
- **The gradient update step can move the parameter extremely far (exploding gradient)**
  - losing most of optimization works that has been done

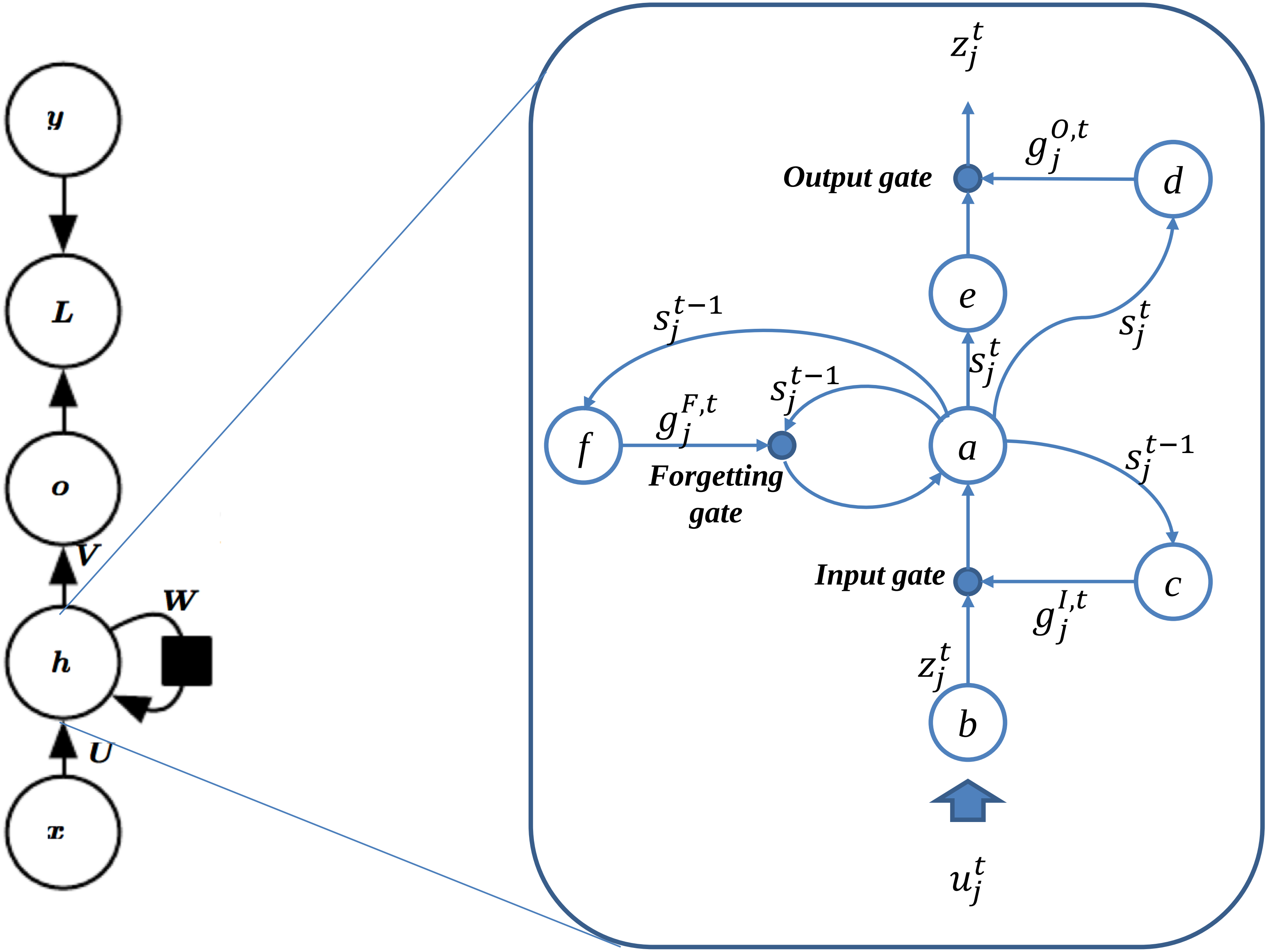


Gradient Clipping for Handling Cliffs

$$\Delta\boldsymbol{\theta}_i \leftarrow \Delta\boldsymbol{\theta}_i \frac{\mathbf{\Delta MAX}}{\boldsymbol{max}(|\Delta\boldsymbol{\theta}_i|,\ \mathbf{\Delta MAX})}$$



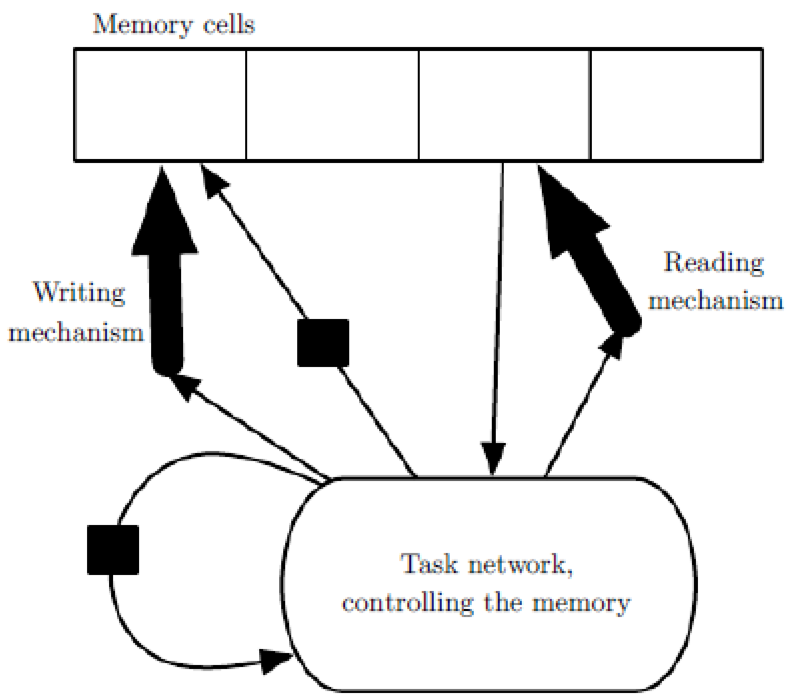
RNN with LSTM



Explicit Memory

Explicit Memory

- **Many researches using memory are ongoing to address LTD**
- **They learned mechanism of control the memory, and deciding where to read from and where to write to**



Memory Networks

- **You can find the paper at:**
  - https://arxiv.org/abs/1410.3916

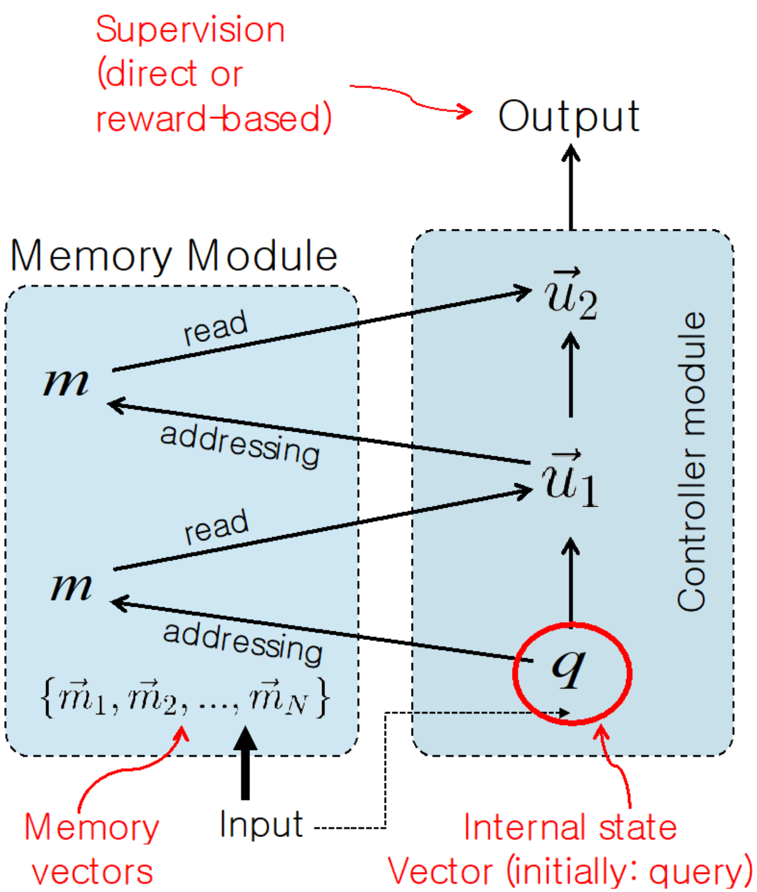
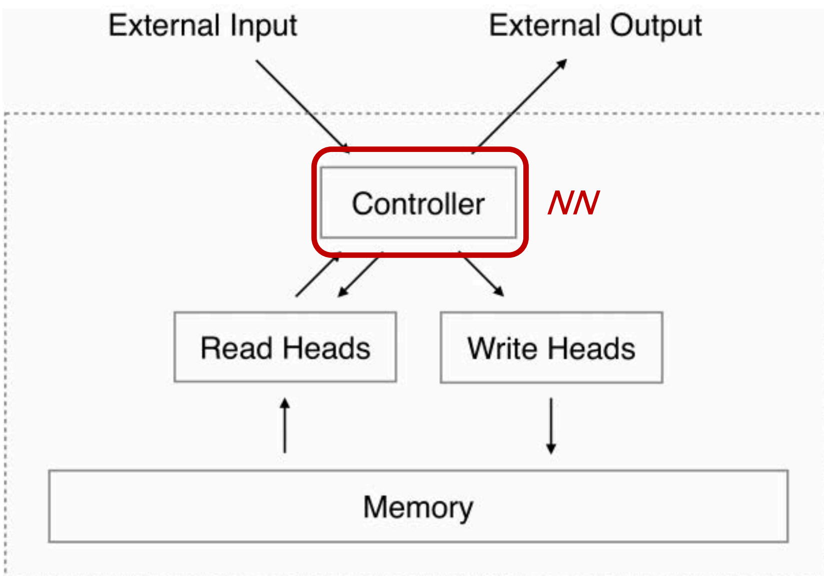


Image from Jason Weston, "Memory network tutorial", at ICML 2016

Neural Turing Machine

- **You can find the paper at:**
  - https://arxiv.org/abs/1410.5401



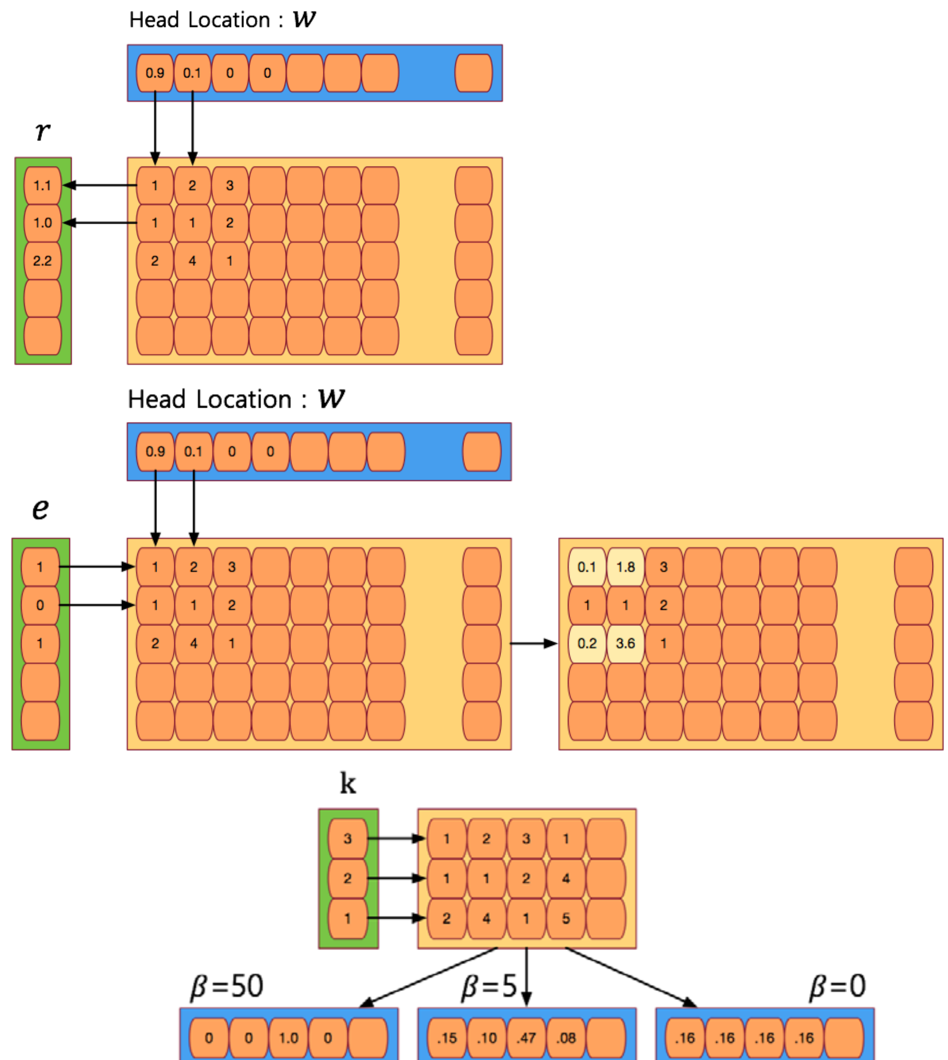


Image from https://norman3.github.io/papers/docs/neural_turing_machine.html

The next Deep Learning Seminar

Chapter 11. Practical Methodology

11.1 Performance Metrics

11.2 Default Baseline Models

11.3 Determining Whether to Gather More Data

11.4 Selecting Hyperparameters

11.5 Debugging Strategies

11.6 Examples: Multi-Digit Number Recognition

Chapter 12. Applications

12.1 Large Scale Deep Learning

12.2 Computer Vision

12.3 Speech Recognition

12.4 Natural Language Processing

12.5 Other Applications

Thank you

Any Questions?