Recurrent Neural Network

Jun-ichi Sato

May 16, 2016

Outline

- Review of last seminar
- Neural Networks
  - Activation function
  - Feed foword calculation
  - Backpropagation
- Recurrent Neural Networks
  - Structure
  - Calculation
- Example of RNN application

Review of last seminar

- We proposed the method of **neural decoding** of c-VEP
- **Spatio-temporal inverse filter** for neural decoding
  - Linear model (LMSE, lasso)
  - Nonlinear model (Neural network)
- We adopt the feed **forward neural network** for the nonlinear model
- Feature works : more appropriate network for neural network
  - e.g. **Recurrent Neural Network**

Neural Network

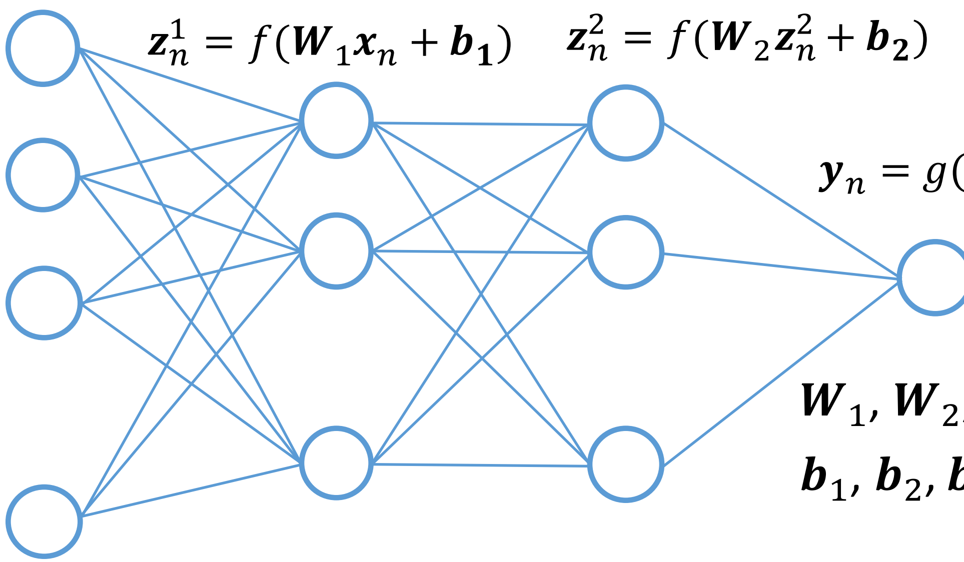
- **Feed-forward** structure
  - Input
  - Hidden
  - Output
- Model for **Classification** or **Regression**

$\boldsymbol{x}_n = x_i[n + \tau + \tau_0]$

$\boldsymbol{z}_n^1 = f(\boldsymbol{W}_1 \boldsymbol{x}_n + \boldsymbol{b}_1)$

$\boldsymbol{z}_n^2 = f(\boldsymbol{W}_2 \boldsymbol{z}_n^2 + \boldsymbol{b}_2)$

$\boldsymbol{y}_n = g(\boldsymbol{w}_3^\top \boldsymbol{z}_n^2 + \boldsymbol{b}_3)$



$\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{w}_3$: weights

$\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3$: bias

History of Neural Network

year	Events
1940s	Research stated
1980s	**Backpropagation [1]**
1980s (late)	Convolutional neural network (CNN)
1990s	**Recurrent neural networks (RNN) [2]**
2000s	Deep belief networks (DBN) [3]

[1] D. E. Rumelhart and J. Mcclelland. "Parallel Distributed Processing: Explorations in the Microstructure of Cognition.", MIT Press, 1986

[2] C. Goller, "Learning task-dependent distributed representations by backpropagation through structure." Neural Networks, 1996., IEEE International Conference on. Vol. 1. IEEE, 1996.

[3] G. E. Hinton, "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.

Neural Network in Computer Vision

ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)

- Classification problem
- 1000 class
- 1.2 million training

Year	Method	Accuracy
2010	Fast descriptor coding, large-scale SVM	72 %
2011	Fisher vector	75 %
2012	**Deep Learning (AlextNet)**	**85 %**
2013	**Deep Learning**	**89 %**
2014	**Deep Learning (GoogleNet)**	**93 %**
2015	**Deep Learning**	**95.18 %**

- Human (hard studied) : 95 %

Beyond the human

Feed forward Neural Network

- Each unit calculates the multi-inputs and the output
- Total input is calculated by

$$u = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

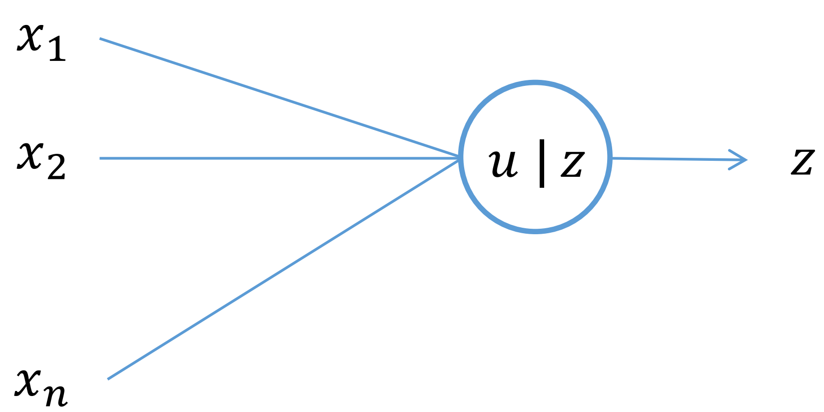
w_i : weight of input

b : bias

- Output is given by

$$z = f(u)$$

f : Activation function



Feed forward calculation of neural network

- Unit of first layer : $i = 1, \ldots, I$
- Unit of second layer : $j = 1, \ldots, J$
- The calculation is generalized by

$$u_j = \sum_{i}^{I} w_{ji} x_i + b_j$$

$$z_j = f(u_j)$$

- We can rewrite the equation by vector and matrix representation

$$\boldsymbol{u} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

$$\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{u})$$

Multi-layer neural network

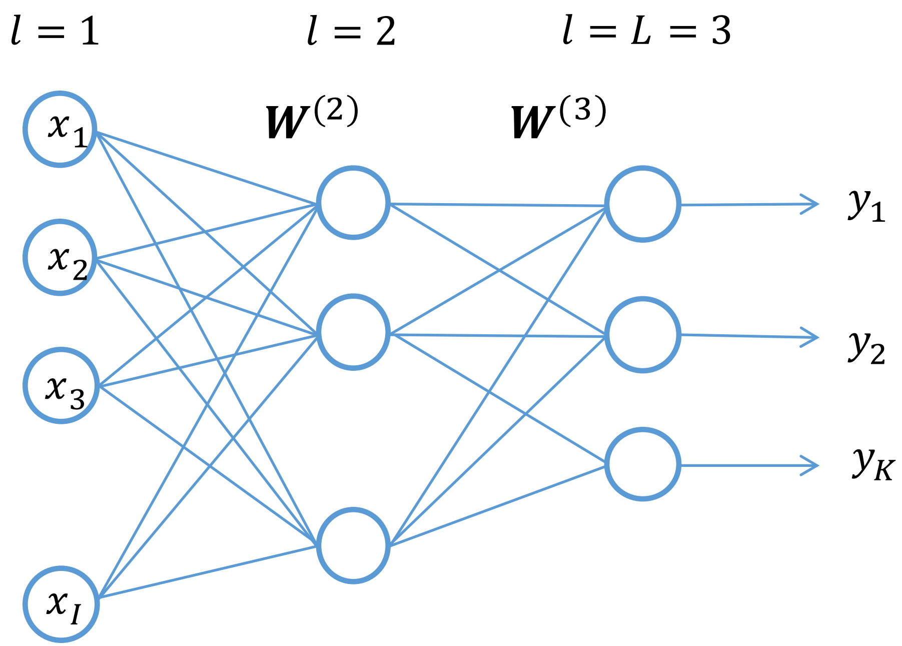
- $L = 3$ layer network

$$\boldsymbol{u}^{(l+1)} = \boldsymbol{W}^{(l+1)}\boldsymbol{z}^{(l)} + \boldsymbol{b}^{(l+1)}$$

$$\boldsymbol{z}^{(l+1)} = \boldsymbol{f}\left(\boldsymbol{u}^{(l+1)}\right)$$

- Output of the network is

$$\boldsymbol{y} \equiv \boldsymbol{z}^{(L)}$$



Activation functions

- Logistic sigmoid function
  - Most used
  - domain : $(-\infty, \infty)$, range : $(0,\ \ 1)$

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

- Hyperbolic tangent
  - Similar to logistic sigmoid function
  - domain : $(-\infty, \infty)$, range : $(-1,\ \ 1)$

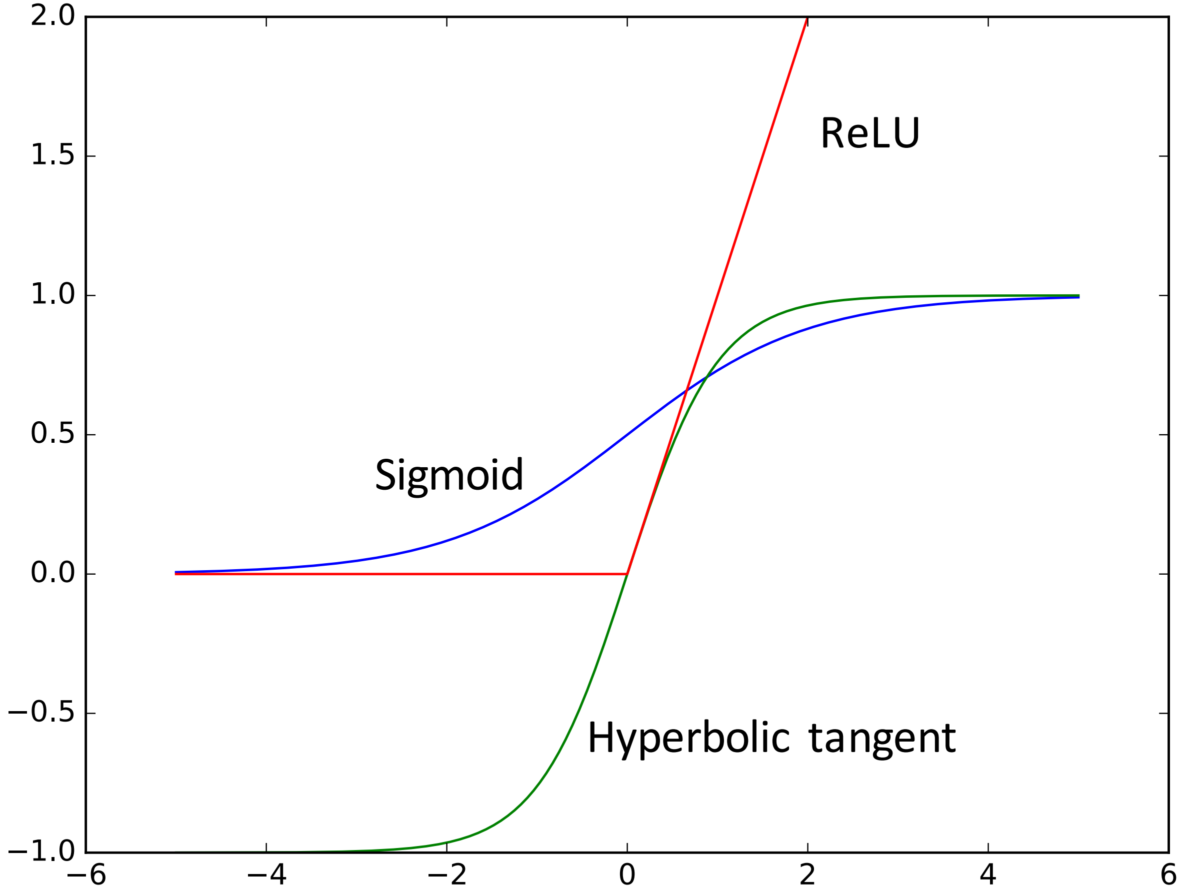
$$f(u) = \frac{1}{1 + e^{-u}}$$

- Rectified linear function (ReLU)
  - Used recently

$$f(u) = \max(u, 0)$$

  - **Faster training** and **better results** than sigmoid and hyperbolic tangent

Activation functions



Design of output layer

- Linear activation function

$$f(u) = u$$

Problem	Activation function	Error function
Regression	Linear activation function	Squared error
Multi-class classification	Softmax function	Cross entropy

  - Softmax function

$$y_k = \frac{\exp(u_k^L)}{\sum_{j=1}^{K} \exp(u_j^L)}$$

  - Cross entropy

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} d_{nk} \log y_k(\boldsymbol{x}_n; \boldsymbol{w})$$

- $\boldsymbol{d}_n$: one-hot encoded vector (like $[0, 0, 0, 1, 0]$)

Backpropagation

- Output layer error :

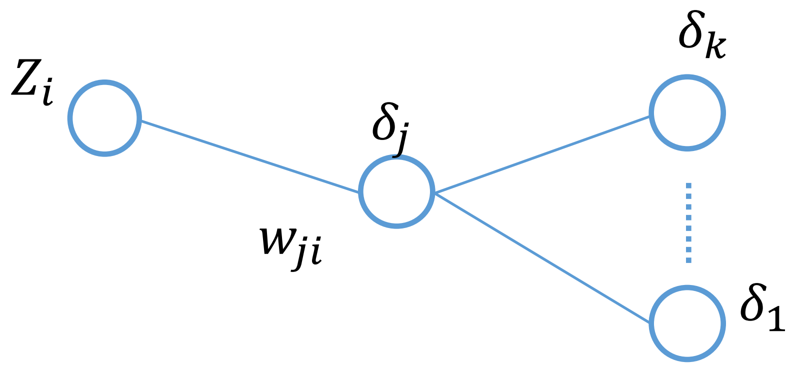
$$\delta_j \equiv \frac{\partial E}{\partial u_j}$$

- Backpropagation formula is given by

$$\delta_j = \sum_k w_{kj} \delta_k f'(u_j)$$

- Differential of E_n is given by

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$$



Classification of sequential data

Sequential data :

- $x^1, x^2, \dots, x^T$
- T is variable
- Ordered data
- E.g. voice, video, text

Example : Predict the next word in the text

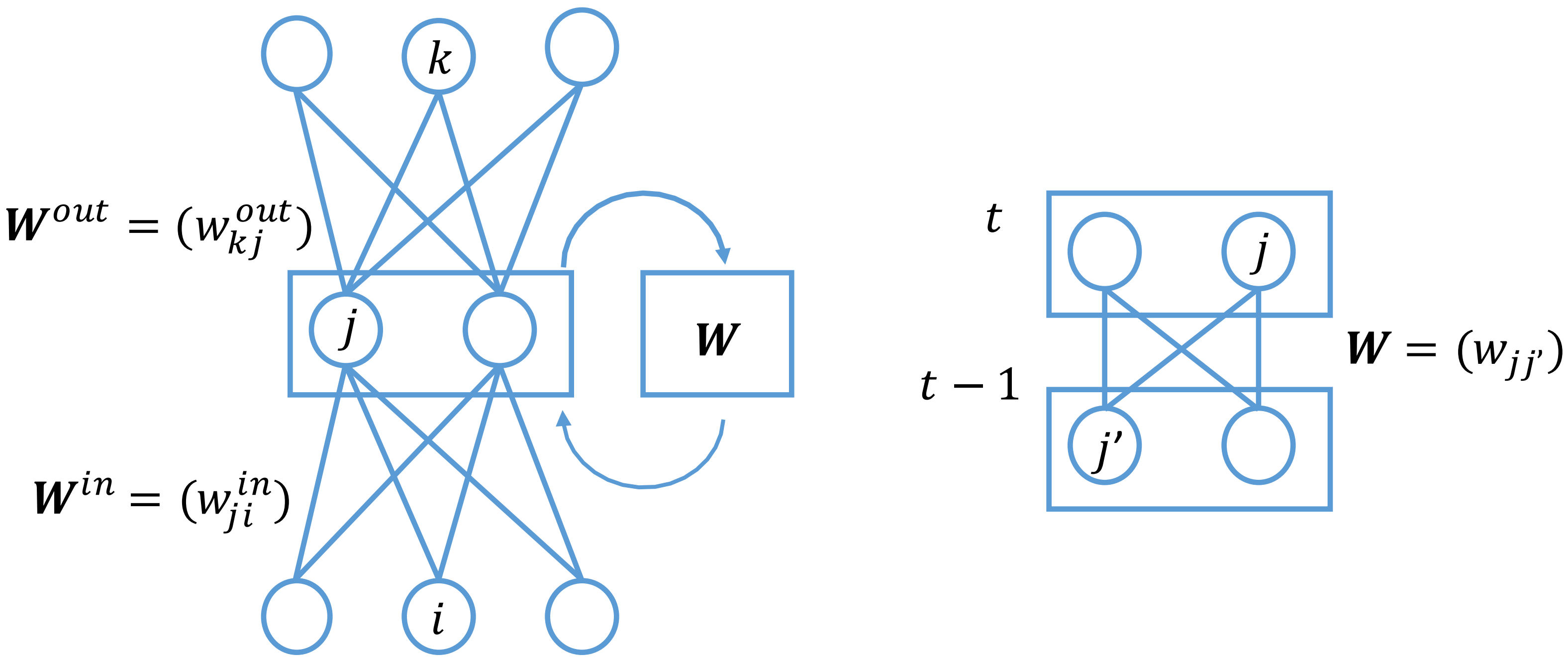
We can get an idea of the quality of the learned **feature** vectors

word	We	can	get	...	the	learned	?
input	x^1	x^2	x^3		x^{t-1}	x^t	x^{t+1}
output		y^1	y^2		y^{t-2}	y^{t-1}	y^t

Predict word y^t is influenced by previous words $x^1, \dots, x^t$

Recurrent Neural Network (RNN)

- RNN is a neural networks which has **closed circuits**
- RNN catches the **context** of a sequential data
- RNN receives $\boldsymbol{x}^t$ and outputs $\boldsymbol{y}^t$ for each time
- Theoretically, RNN outputs $\boldsymbol{y}^t$ from all inputs $(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^t)$



Design of output layer for RNN

- Same as feed forward network
- In classification problem,
  softmax function is used for activation function
- Error function is given by,

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K} d_{nk}^{t} \log y_{k}^{t}(\boldsymbol{x}_n; \boldsymbol{w})$$

  - $\boldsymbol{y}^1, \dots, \boldsymbol{y}^T$: output sequence
  - $\boldsymbol{d}^1, \dots, \boldsymbol{d}^T$: target of the output
  - $\boldsymbol{x}_n$: input of n sample
  - d_{nk}^{t} : target of n sample at time t

RNN : Feed forward calculation

- Each unit is different state at each time $t = 1, 2, \ldots$

$\boldsymbol{x}^t = (x_i^t)$	input
$\boldsymbol{u}^t = (u_j^t)$	input of hidden layer
$\boldsymbol{z}^t = (z_j^t)$	output of hidden layer
$\boldsymbol{v}^t = (v_k^t)$	input of output layer
$\boldsymbol{y}^t = (y_k^t)$	output of output layer
$\boldsymbol{d}^t = (d_k^t)$	target of the output
$\boldsymbol{W}^{in} = (w_{ji}^{in})$	weight of the input to hidden layer
$\boldsymbol{W} = (w_{jj'})$	weight of feedback path
$\boldsymbol{W}^{out} = (w_{kj}^{out})$	weight of hidden to output

- $\boldsymbol{W}$ is constant value at any time

RNN : Feed forward calculation

- Input of the hidden layer at $t = t$

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

  - Sum of **inputs at $\boldsymbol{t}$** and output of **hidden layer at $\boldsymbol{t-1}$**
  - Bias is $w_{j0}^{(in)}$

- Output of hidden layer

$$z_j^t = f(u_j^t)$$

$$\boldsymbol{z}^t = \boldsymbol{f}(\boldsymbol{W}^{(in)}\boldsymbol{x}^t + \boldsymbol{W}\boldsymbol{z}^{t-1})$$

- Output of output layer

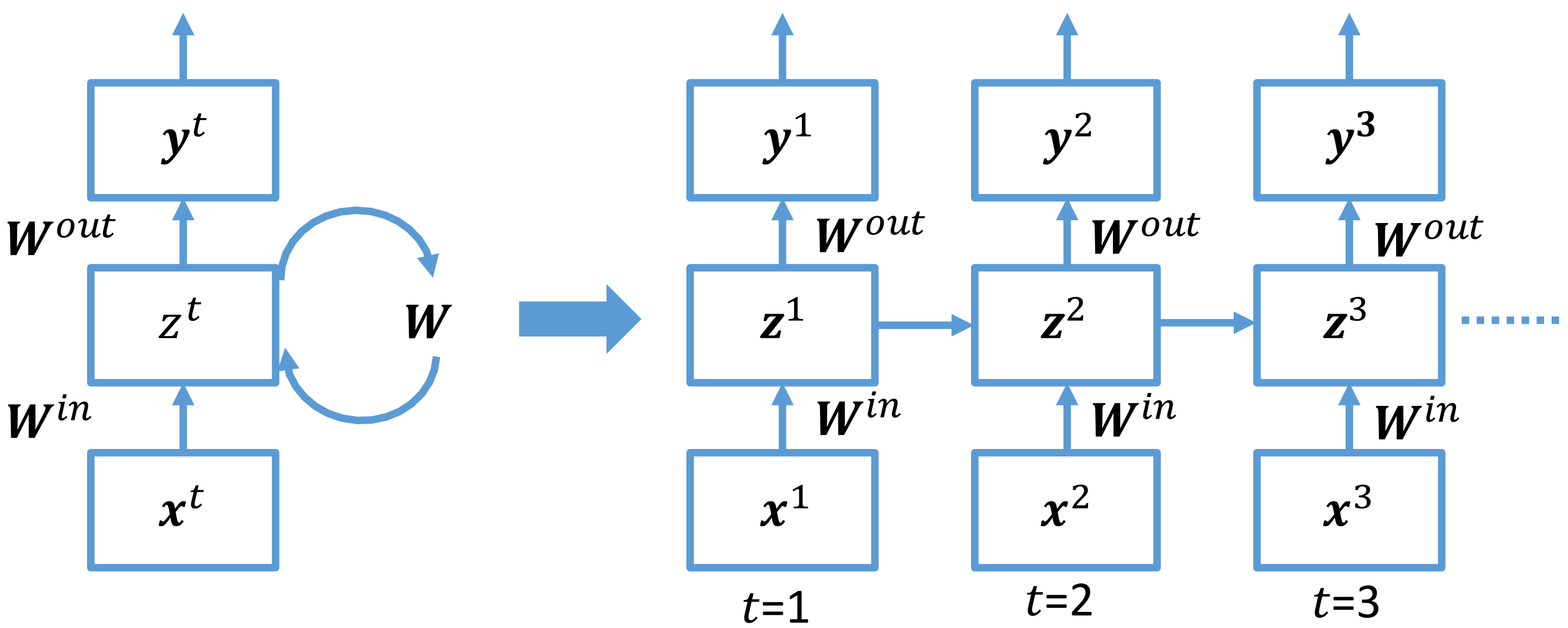
$$v_k^t = \sum_j w_{kj}^{(out)} z_j^t$$

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

RNN : Backpropagation (1)

BPTT : backpropagation through time

- Replace RNN with feed forward NN
- Unfolding RNN through time



RNN : Backpropagation (2)

- Backpropagation of feed forward networks

$$\delta_j^{(l)} = \sum_k w_{kj}^{(l+1)} \delta_k^{(l+1)} f'\left(u_j^{(l)}\right)$$

- Output delta of unit k at t

$$\delta_k^{out,t} \equiv \frac{\partial E}{\partial v_k^t}$$

- Hidden delta of unit j at t

$$\delta_j^t \equiv \frac{\partial E}{\partial u_j^t}$$

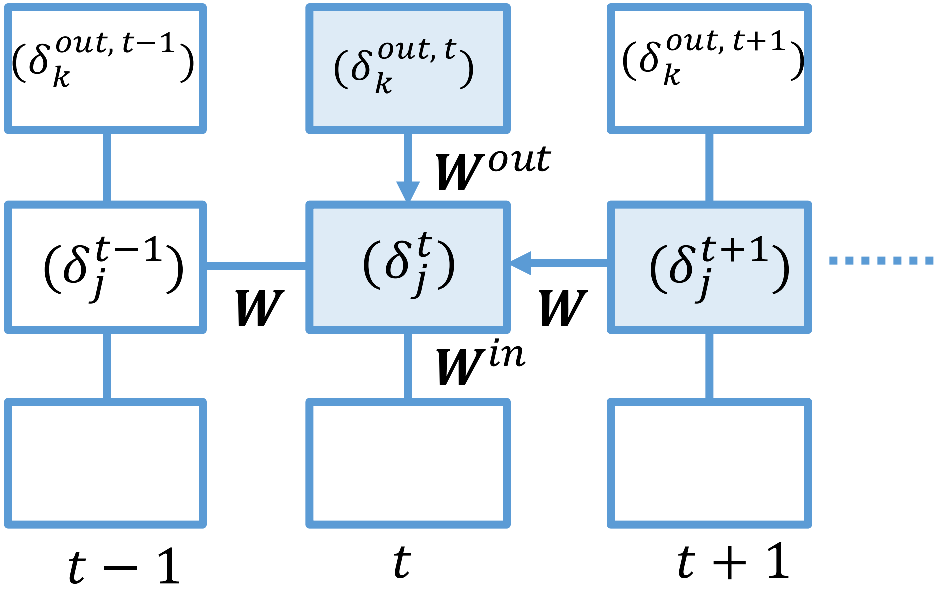
RNN : Backpropagation (3)

- δ_j^t is given by

$$\delta_j^t = \left(\sum_k w_{kj}^{out} \delta_k^{out,t} + \sum_{j'} w_{jj'} \delta_{j'}^{t+1} \right) f'(u_j^t)$$

- Delta of $T + 1$ is not able to calculate

$$\delta_j^{T+1} = 0$$



RNN : Backpropagation (4)

- Differential of E are given by

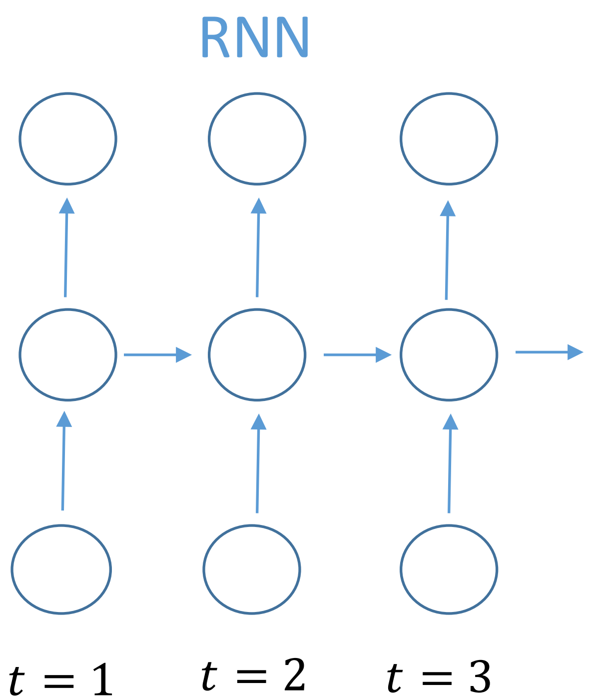
$$\frac{\partial E}{\partial w_{ji}^{in}} = \sum_{t=1}^{T} \frac{\partial E}{\partial u_j^t} \frac{\partial u_j^t}{\partial w_{ji}^{in}} = \sum_{t=1}^{T} \delta_j^t x_i^t$$

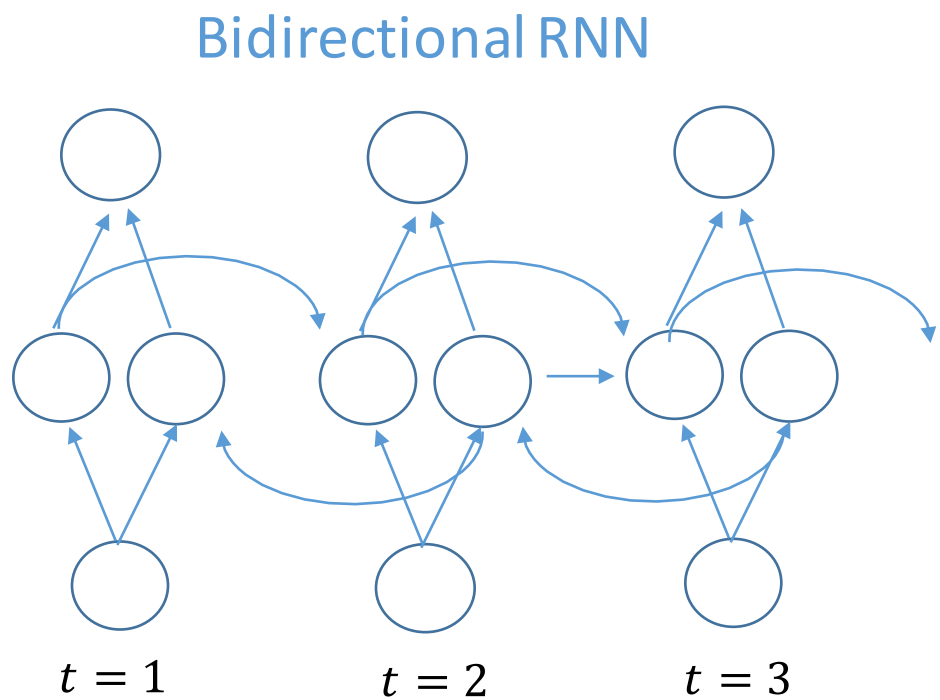
$$\frac{\partial E}{\partial w_{j\,j'}} = \sum_{t=1}^{T} \frac{\partial E}{\partial u_j^t} \frac{\partial u_j^t}{\partial w_{j\,j'}} = \sum_{t=1}^{T} \delta_j^t x z_j^{t-1}$$

$$\frac{\partial E}{\partial w_{kj}^{out}} = \sum_{t=1}^{T} \frac{\partial E}{\partial v_k^t} \frac{\partial v_k^t}{\partial w_{kj}^{out}} = \sum_{t=1}^{T} \delta_j^t z_j^t$$

Bidirectional RNN

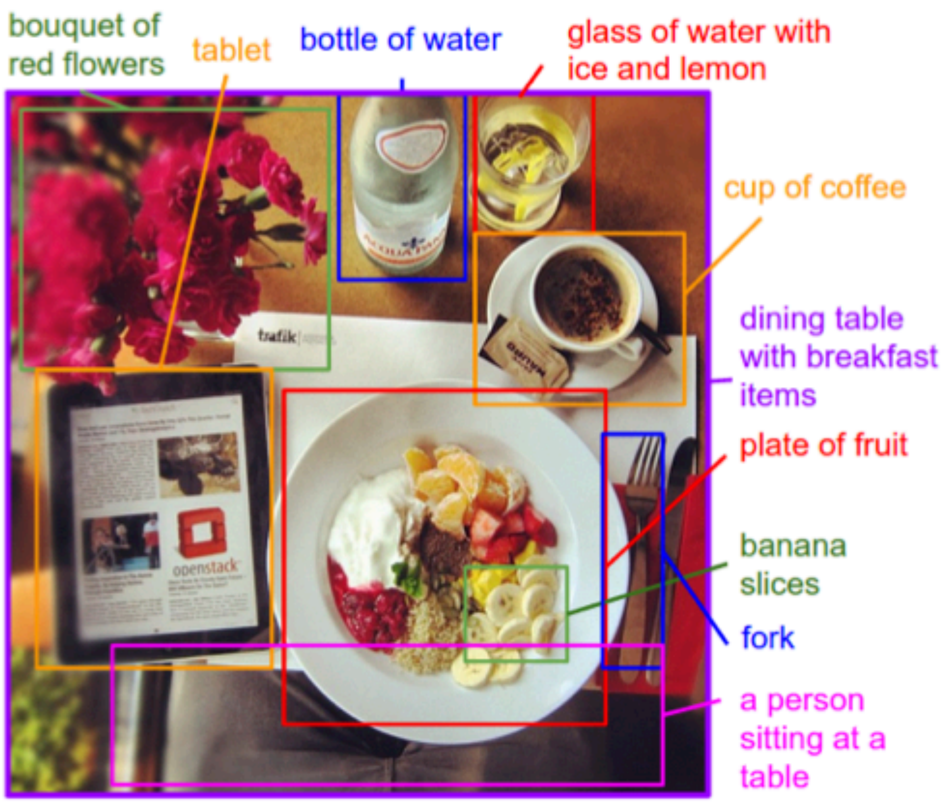
- If all data is given at the same time,
  We can input the reversal data ($t = T, T - 1 \ldots, 1$) to the RNN
- B-RNN has better performance than normal RNN





Example of RNN

- "Deep Visual-Semantic Alignments for Generating Image Descriptions", K. Andrej and F. Li, CVPR, 2015
- Generate the description from the **image region**



Proposed method

Image representation : CNN

- CNN which is trained by **ImageNet**
- I_b: pixel of image region
- Image representation is given by

$$v = W_m\left[CNN_{\theta_c}(I_b)\right] + b_m$$

Sentence representation : BRNN

- $\boldsymbol{I}_t$: input vector, word of the number t. 1 of K representation
- s_t : output vector
- **BRNN** is used to get output vector
- W_w : weight of **word2vec**
- $f: x \mapsto \max(0, x)$, ReLU

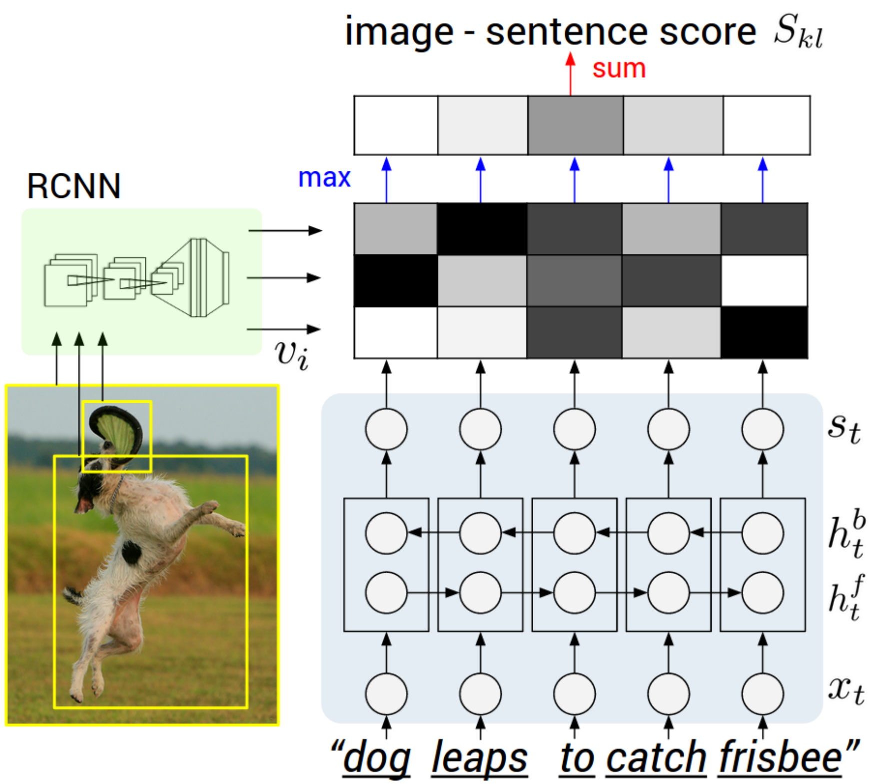
$$\begin{aligned}
x_t &= W_w \boldsymbol{I_t} \\
e_t &= f(W_e x_t + b_e) \\
h_t^f &= f(e_t + W_f h_{t-1}^f + b_f) \\
h_t^b &= f(e_t + W_b h_{t+1}^b + b_b) \\
s_t &= f\left(W_d\left(h_t^f + h_t^b\right) + b_d\right)
\end{aligned}$$

Score

- The score is given by

$$S_{kl} = \sum_{t \in g_l} \sum_{i \in g_k} \max(0, v_i^T s_t)$$

- g_l : index set of sentence
- g_k : index set of image region



Result : Full frame

- "man in black shirt is playing a guitar"
  does not exist in training data sets
- "man in black shirt" and "is playing guitar" exists

man in black shirt is playing guitar.

construction worker in orange safety vest is working on road.

Full frame : Failure

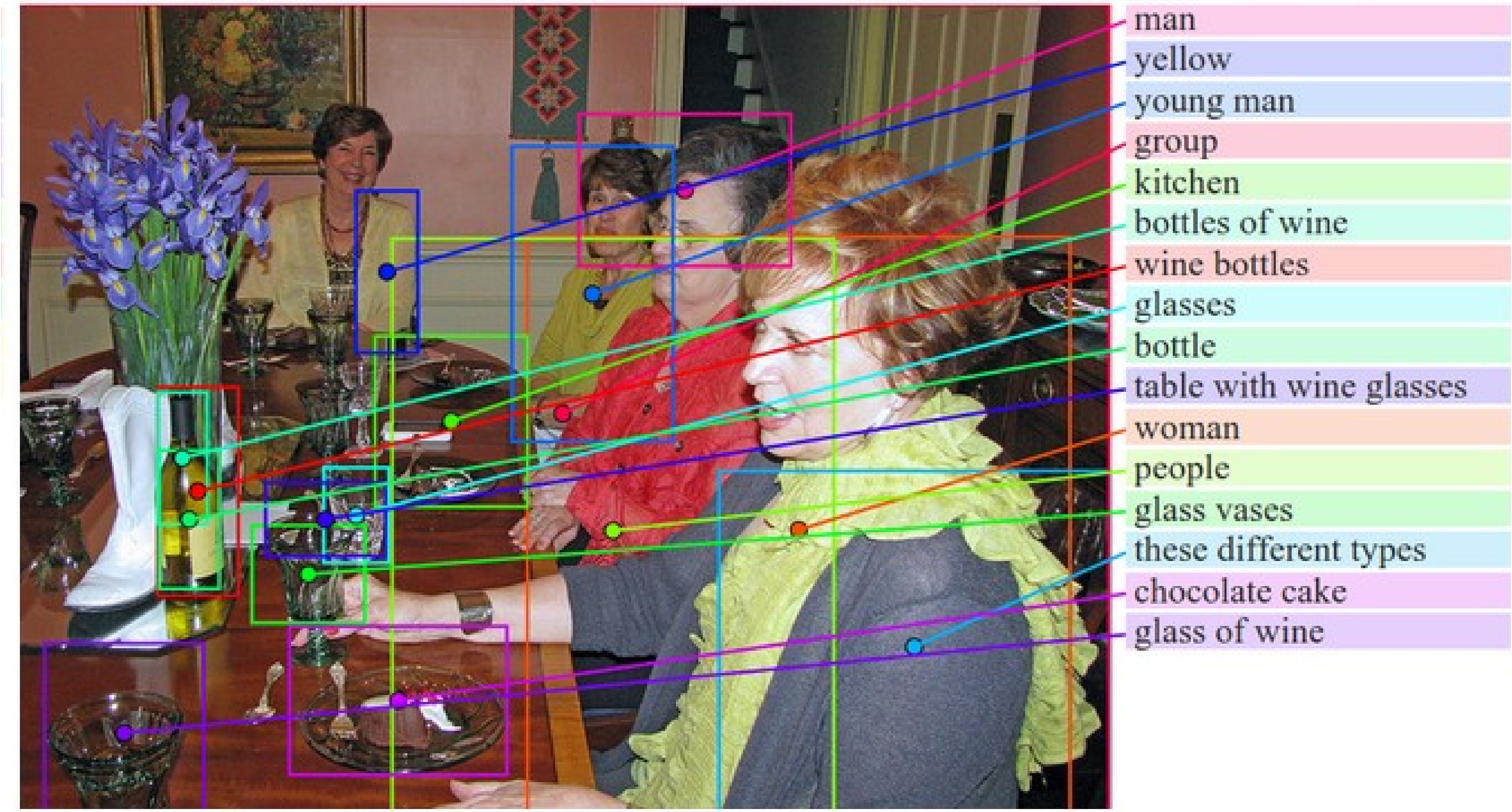
- Not young
- Not wakeboard

two young girls are playing with lego toy.

boy is doing backflip on wakeboard.

Result : region-level descriptions

- "Table with wine glasses"
  - In training, it occurred in small region only 30 times.



Future works

- To apply RNN to c-VEP BCI
- To study Long Short Term Memory (LSTM)
  - One of the RNN