Speech and Language Processing

Lecture 3
Artificial neural network

Information and Communications Engineering Course

Takahiro Shinozaki

Manabu Okumura

2024/10/1

Artificial Neural Network

Neuron

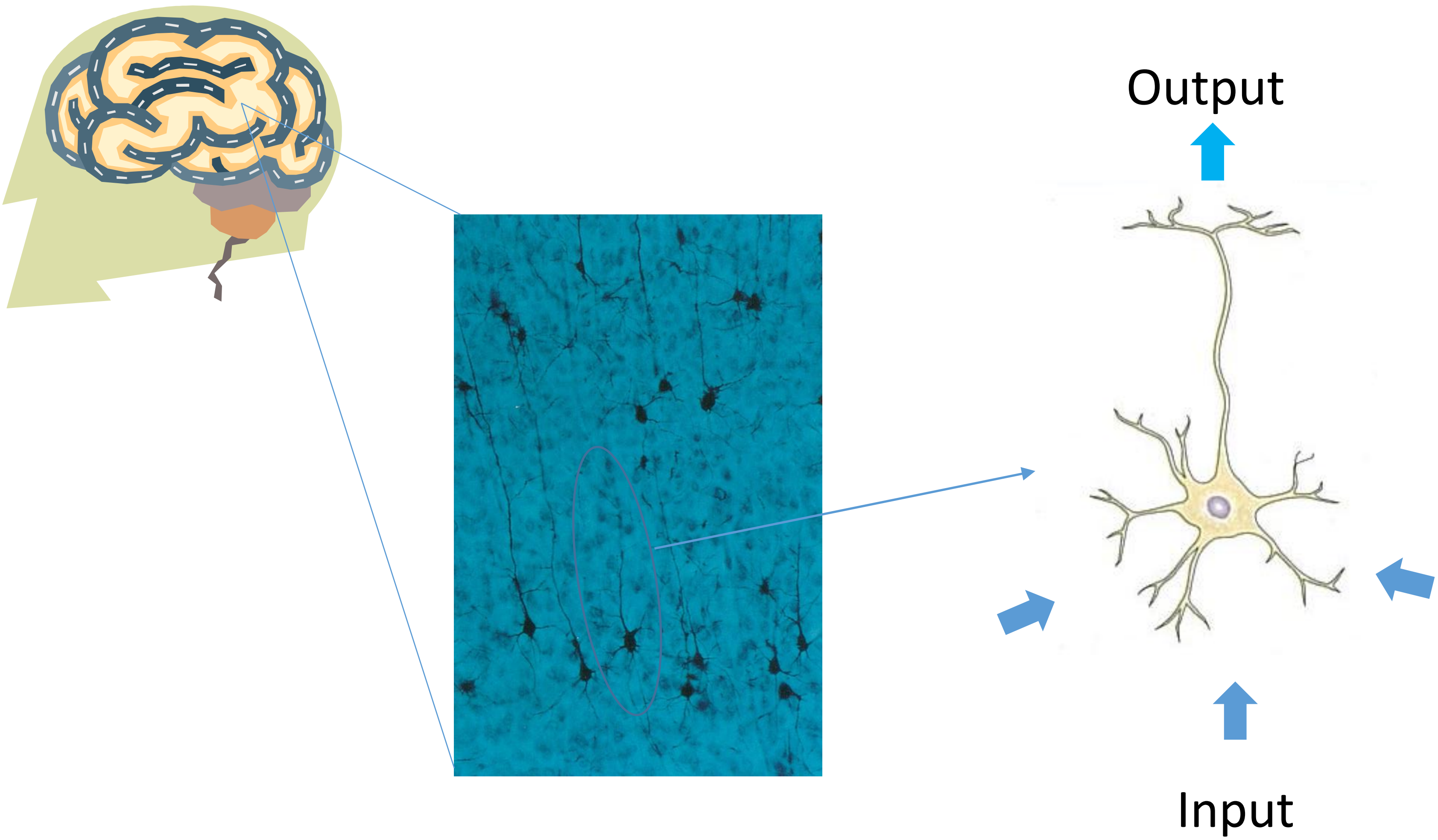
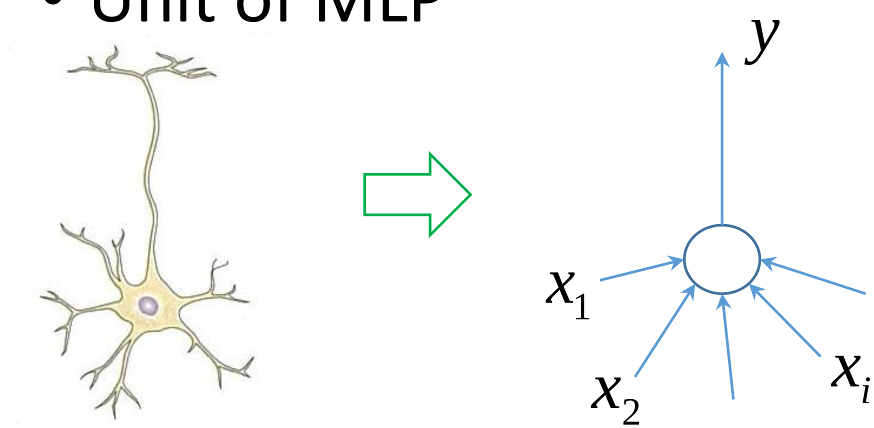


*Figure adapted from「ニューロンの生物学」

Multi Layer Perceptron (MLP)

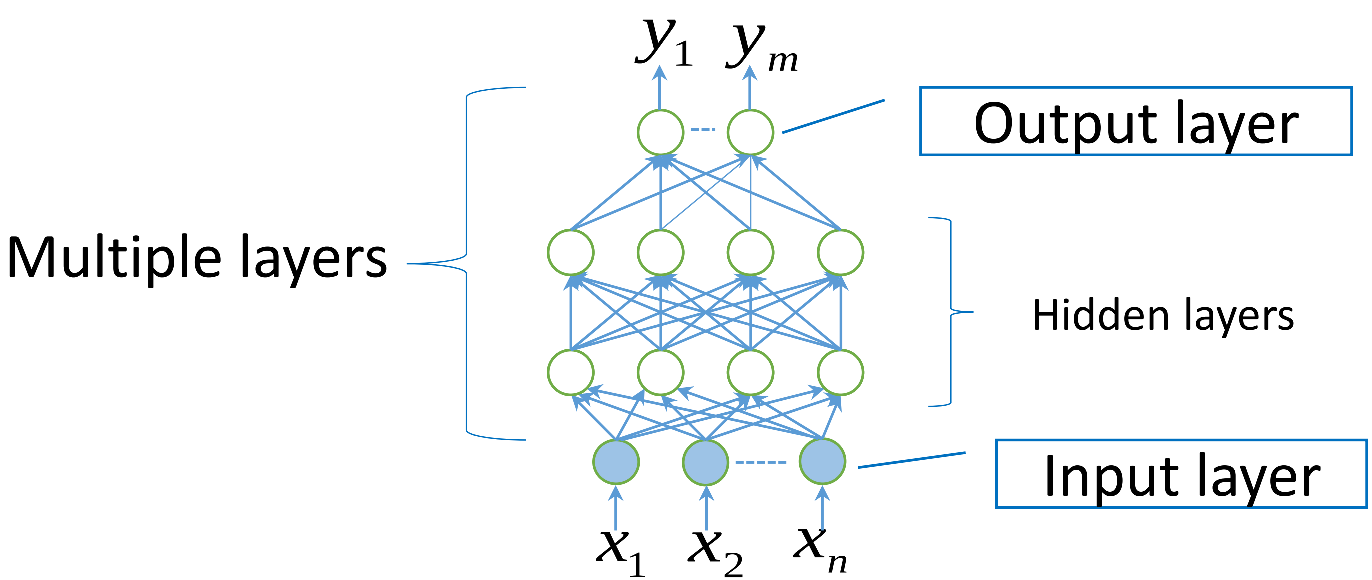
- Unit of MLP



$$y = h\left(\sum_i w_i x_i + b\right)$$

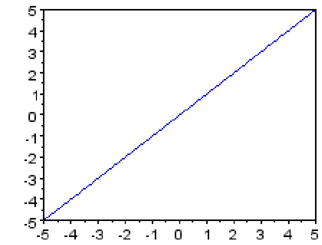
h: activation function
w: weight
b : bias

- MLP consists of multiple layers of the units

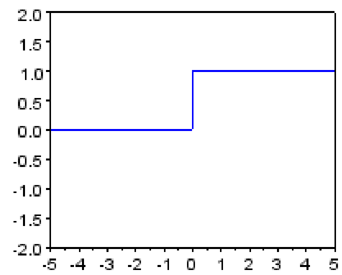


Activation Functions

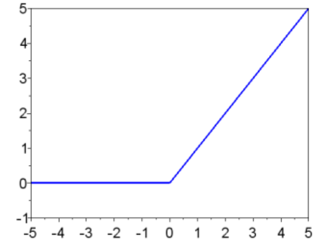
- Linear function $h(x) = x$



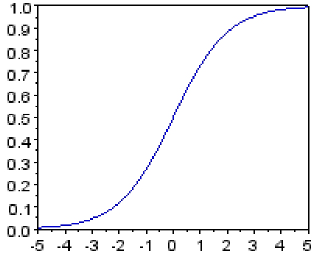
- Unit step function $h(x) = \begin{cases} 1 & \text{if } 0 \leq x \\ 0 & \text{otherwise} \end{cases}$



- hinge function $h(x) = \max\{0, x\}$



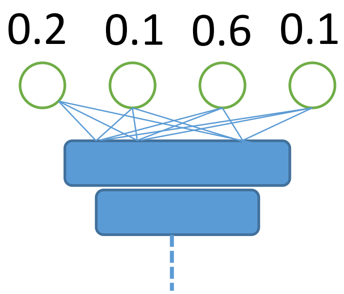
- Sigmoid function $h(x) = \dfrac{1}{1 + \exp(-x)}$



Softmax Function

- For *N* variables z_i, softmax function is:

$$h(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$



- Properties of softmax
  - Positive $\quad 0 < h(z_i)$
  - Sum is one $\quad \sum_{i=1}^{N} h(z_i) = 1.0$

$\Rightarrow$ Expresses a probability distribution

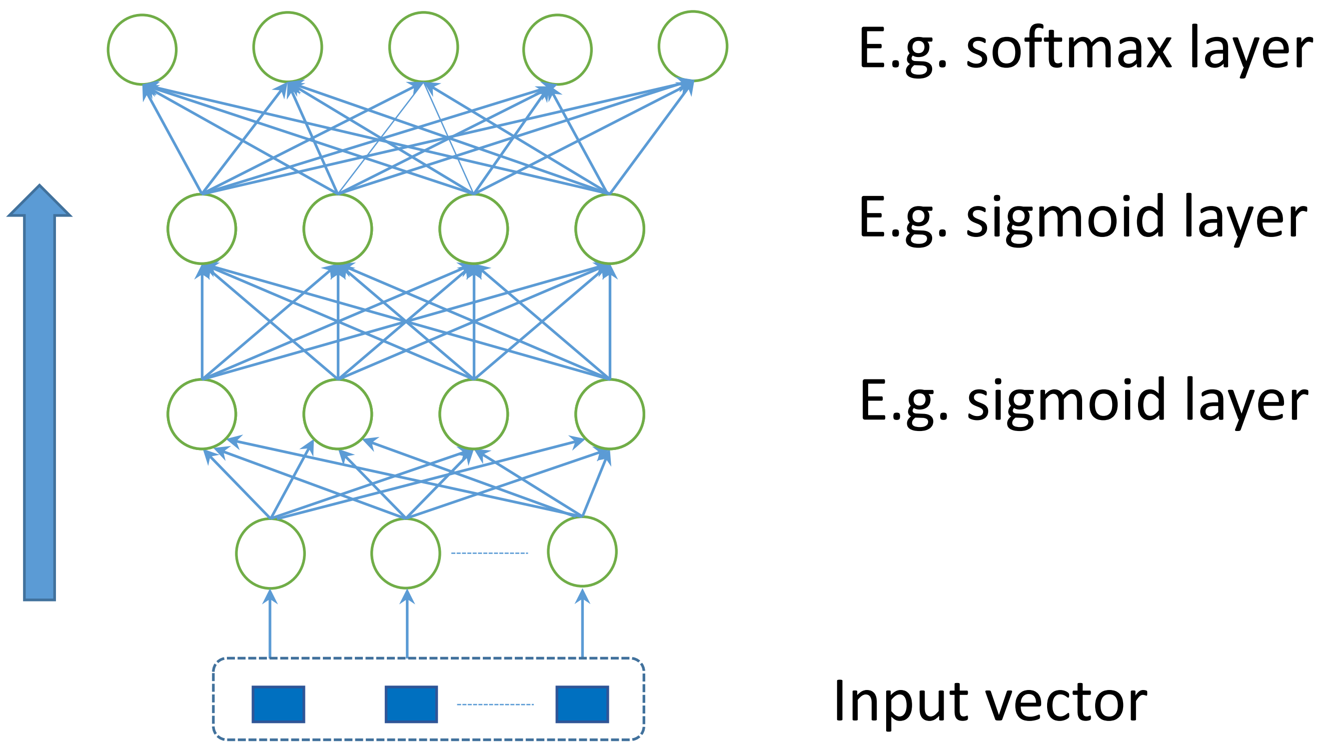
- Example

$$Z = \langle z_1, z_2, z_3 \rangle = \langle -1,2,1 \rangle \Rightarrow h(Z) = \langle h(z_1), h(z_2), h(z_3) \rangle = \langle 0.0351, \quad 0.7054, \quad 0.2595 \rangle$$

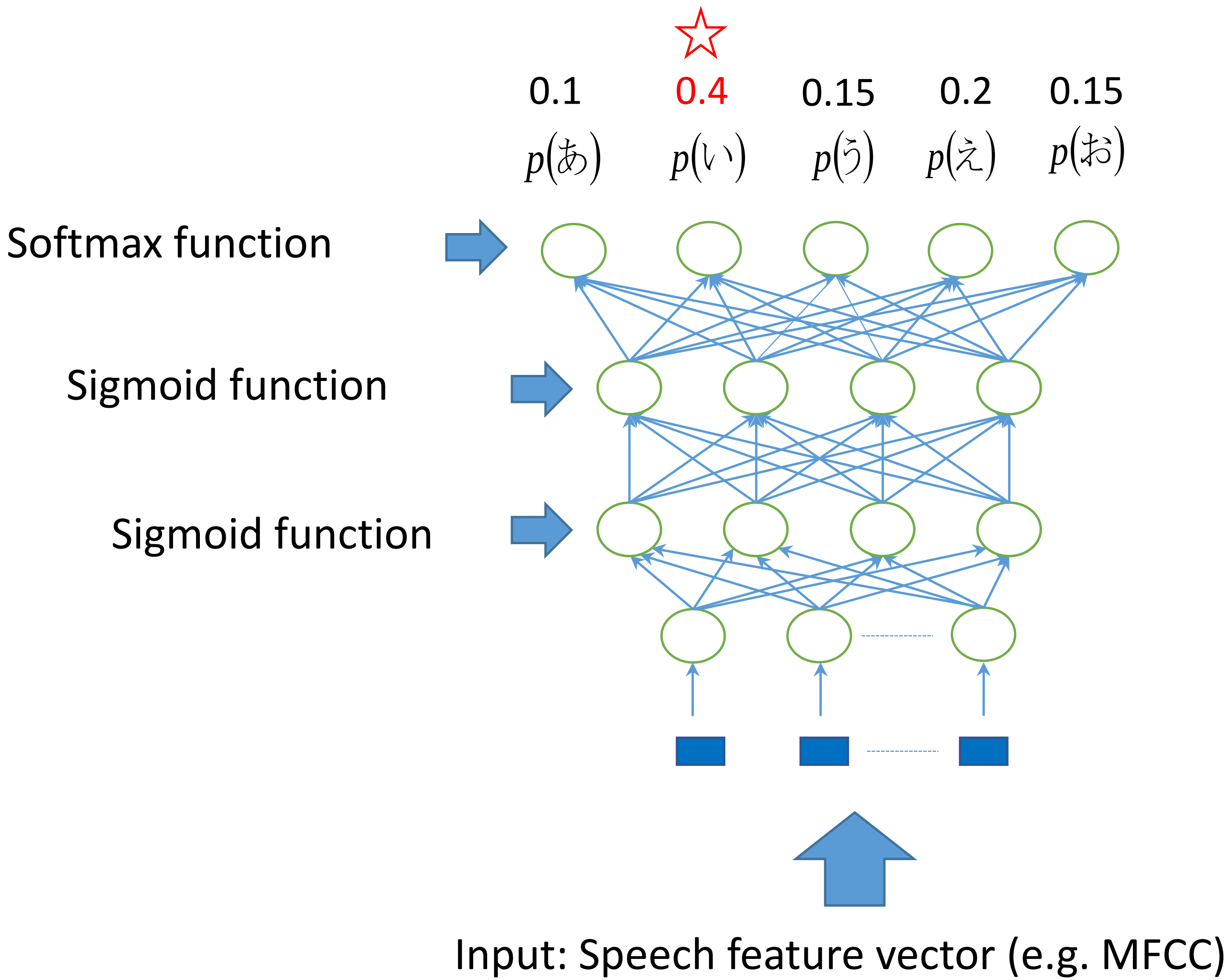
$$Z = \langle z_1, z_2, z_3 \rangle = \langle 16,8,12 \rangle \Rightarrow h(Z) = \langle h(z_1), h(z_2), h(z_3) \rangle = \langle 0.9817, \quad 0.0003, \quad 0.0180 \rangle$$

Forward Propagation

- Compute the output of MLP step by step from the input layer to output layer

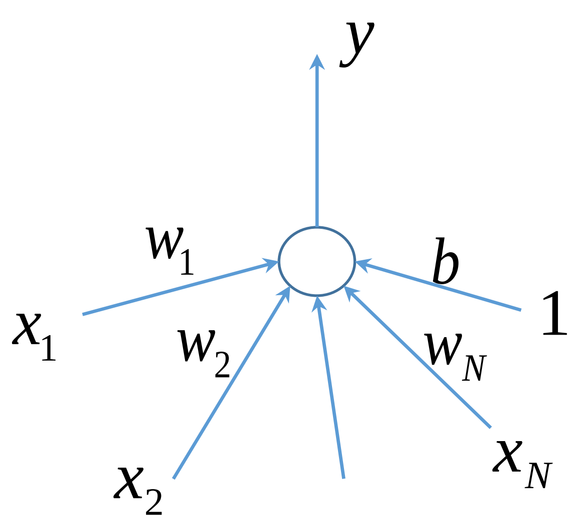


Example: Frame Level Vowel Recognition



Parameters of Neural Network

- The weights and a bias of each unit need training before the network is used



$$y = h\left(\sum_i w_i x_i + b\right)$$
$$= h(\mathbf{w} \cdot \mathbf{x})$$

h: activation function

w: weight vector

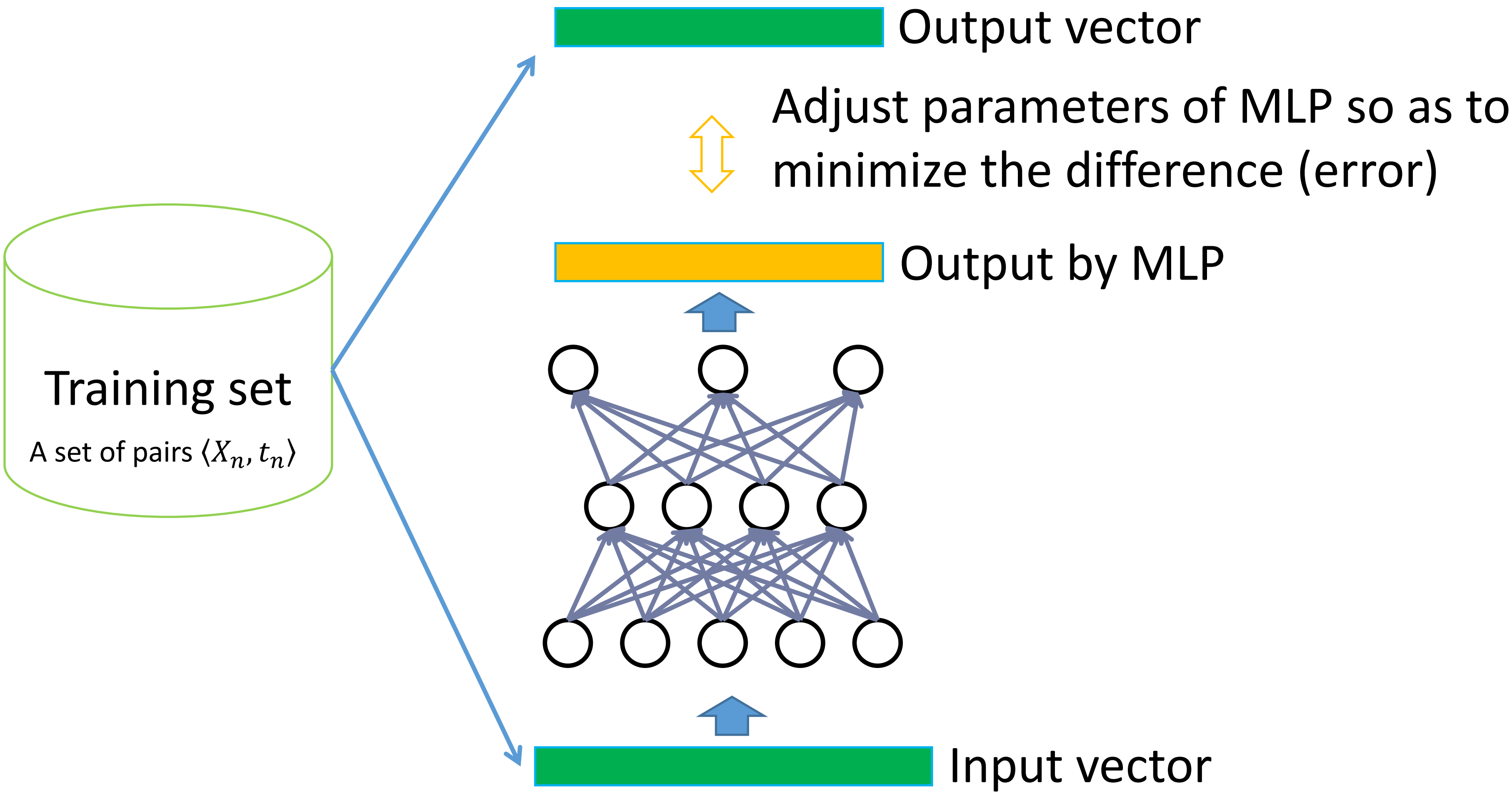
$w=(w_1,w_2,\ldots,w_N,b)$

X: input vector

$x=(x_1,x_2,\ldots,x_N,1)$

The bias *b* can be regarded as one of the weights whose input takes a constant value 1.0

Principle of Supervised NN Training



Definitions of Errors

- Sum of square error
  - Used for continuous outputs

$$E(W) = \frac{1}{2}\sum_{n} \|y(X_n, W) - t_n\|^2$$

W : Set of weights in MLP

X_n: Vector of a training sample (input)

t_n : Vector of a training sample (output)

n : Index of training samples

- Cross-entropy
  - Used categorical outputs (e.g. softmax)

$$E(W) = -\sum_{n}\sum_{k} t_{n,k} \ln y_k\,(X_n, W)$$

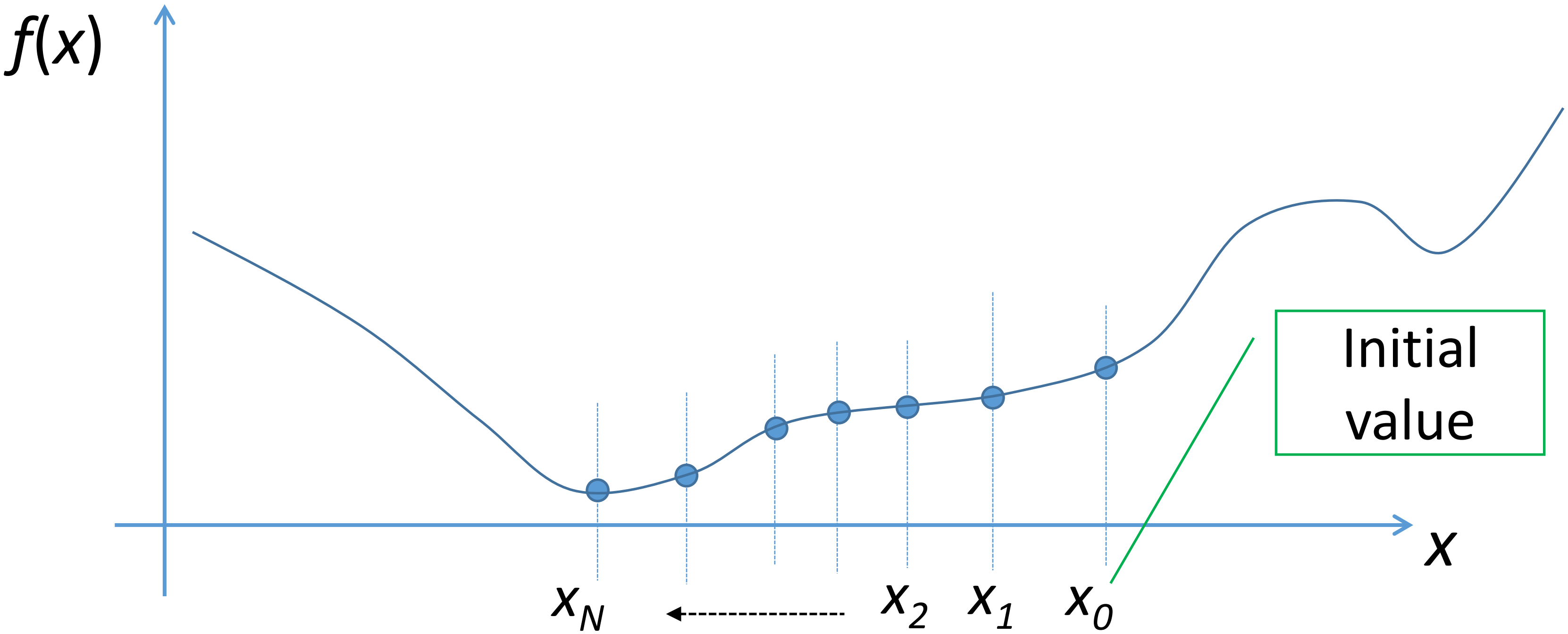
k : Index of output unit

$t_{n,k}$: Reference output (Takes 1 if unit k corresponds to correct output, 0 otherwise)

y_k : Value of k-th output unit

Gradient Descent

- An iterative optimization method



$$x_{t+1} = x_t - \varepsilon \frac{\partial f(x)}{\partial x_t}$$

ε : Learning rate (small positive value)

NN Training by Gradient Descent

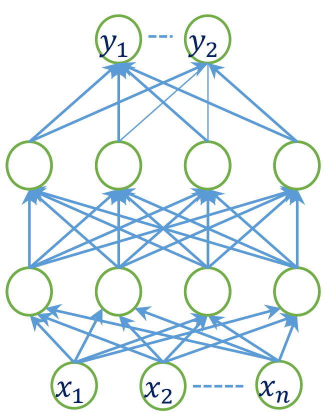
- Define an error measure $E(W)$ using a training set

e.g. $$E(W) = \frac{1}{2}\sum_{n} \|Y(X_n, W) - R_n\|^2$$

- Initialize parameters $W_0 = \left[w_{1,0}, w_{2,0}, \cdots, w_{M,0}\right]^T$
- Repeatedly update the parameter set using gradient descent

$$W(t+1) = W(t) - \varepsilon \left.\frac{\partial E(W)}{\partial W}\right|_{W=W(t)}$$

R_n



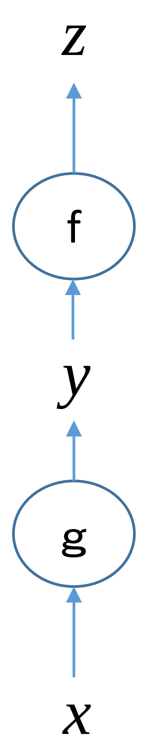
Y: output of NN

W: all parameters of NN

X: input to NN

Chain Rule of Differentiation

$$z = f(y)$$
$$y = g(x)$$



- When x,y,z are scalars:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

- When x,y,z are vectors:

$$x = \langle x_1, x_2, x_3 \rangle,\ \ y = \langle y_1, y_2 \rangle,\ \ z = \langle z_1, z_2 \rangle$$

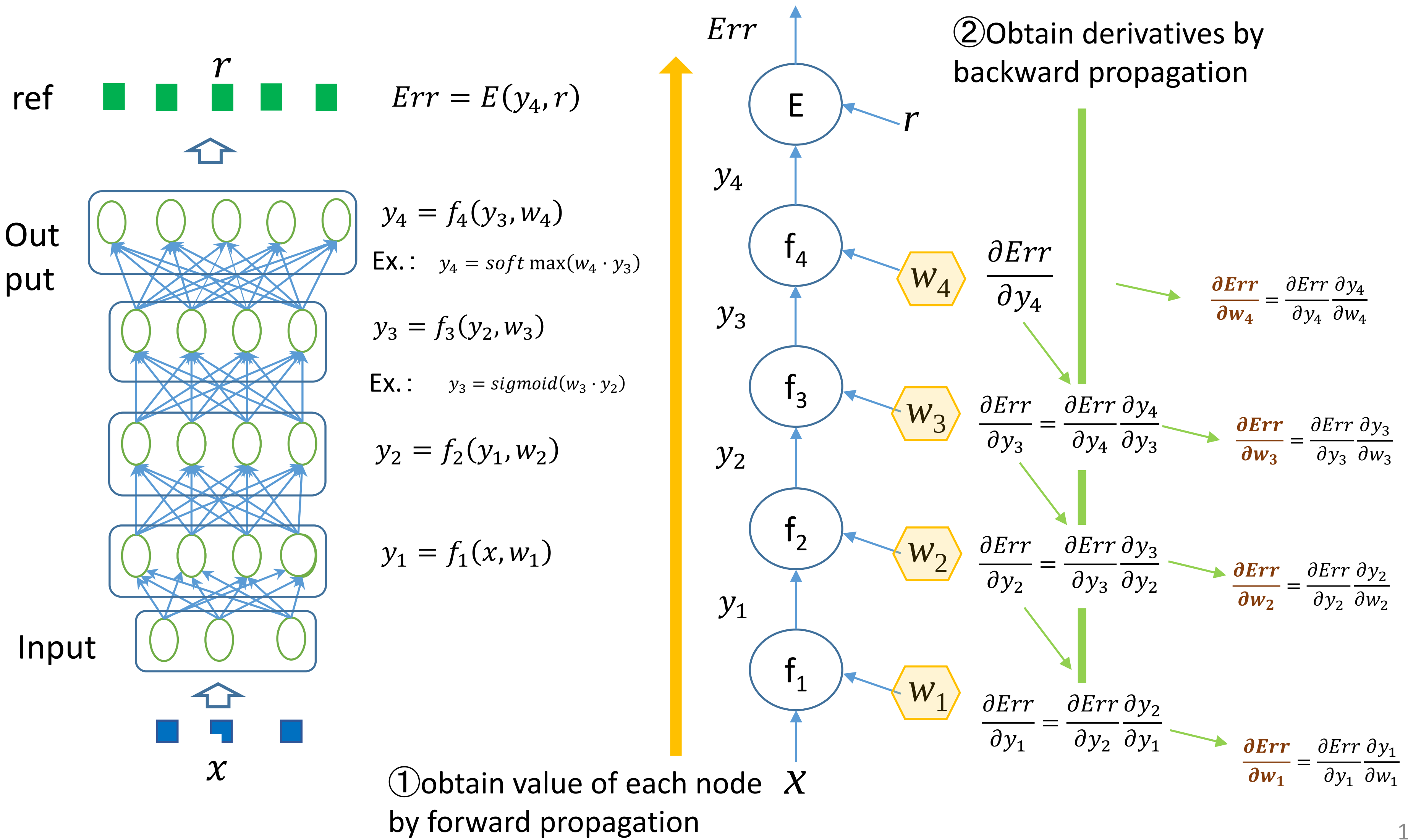
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

The same rule holds for Jacobian matrix

Jacobian matrix

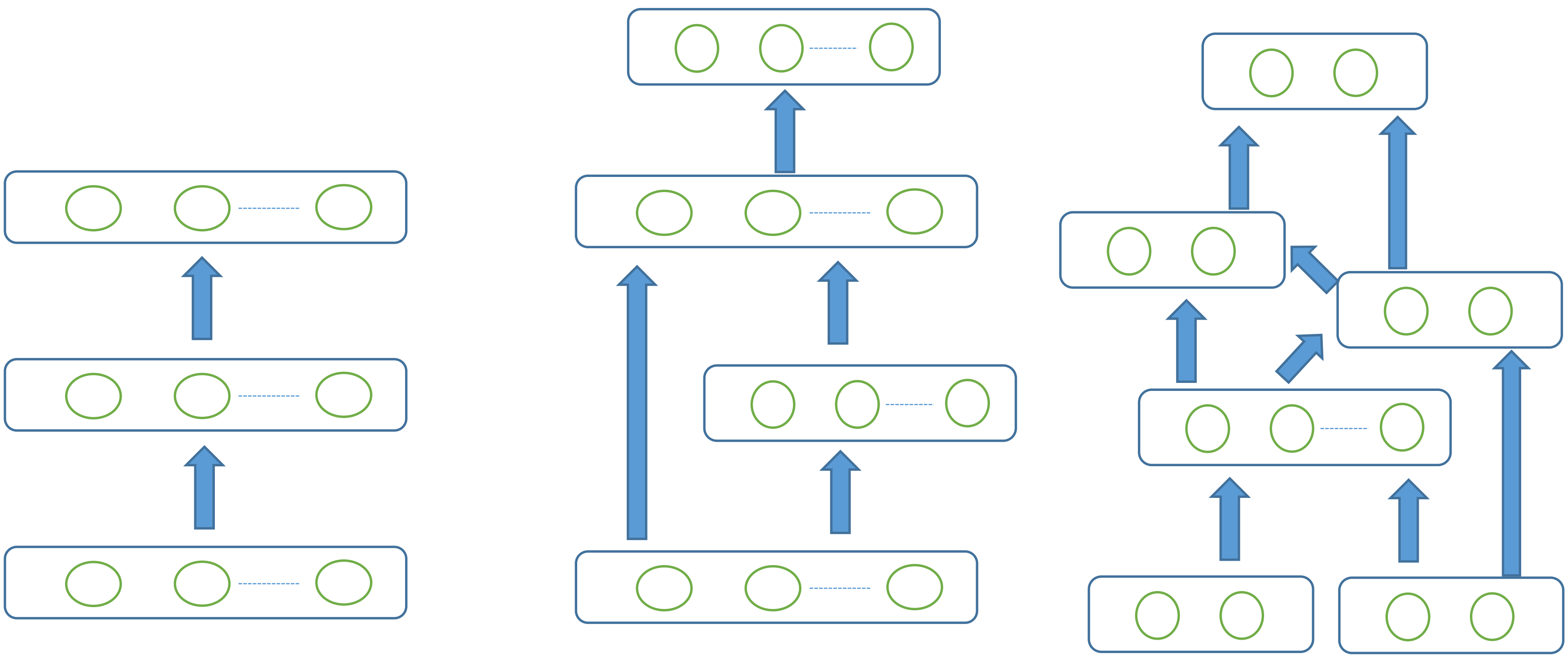
$$\begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} & \frac{\partial z_1}{\partial x_3} \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} & \frac{\partial z_2}{\partial x_3} \end{pmatrix} = \begin{pmatrix} \frac{\partial z_1}{\partial y_1} & \frac{\partial z_1}{\partial y_2} \\ \frac{\partial z_2}{\partial y_1} & \frac{\partial z_2}{\partial y_2} \end{pmatrix} \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \end{pmatrix}$$

Back Propagation(BP)



Neural Network Structures

So far, we considered layer-structured network. In general, we can consider more complex ones.



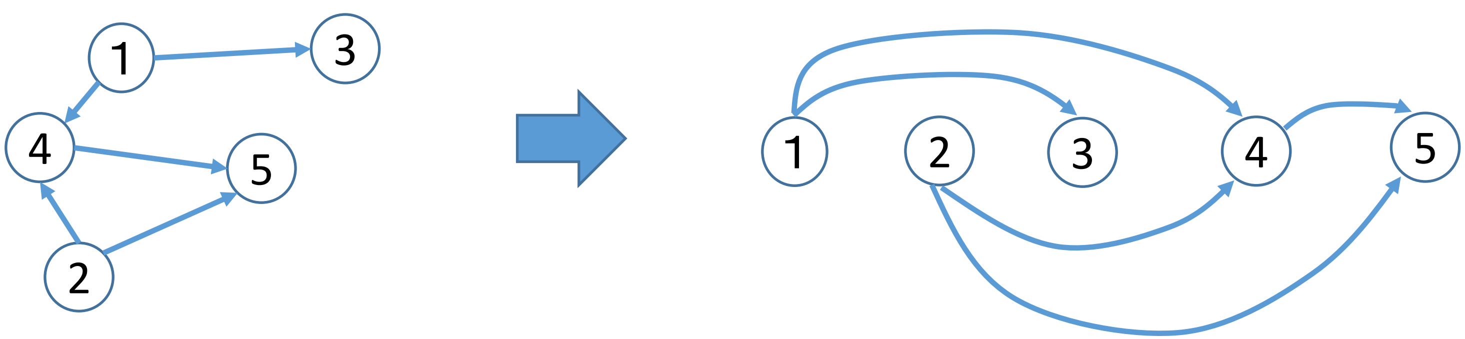
Can we still perform forward/backward propagations?

Directed Graph and Node Ordering

A directed graph is a DAG

Eq

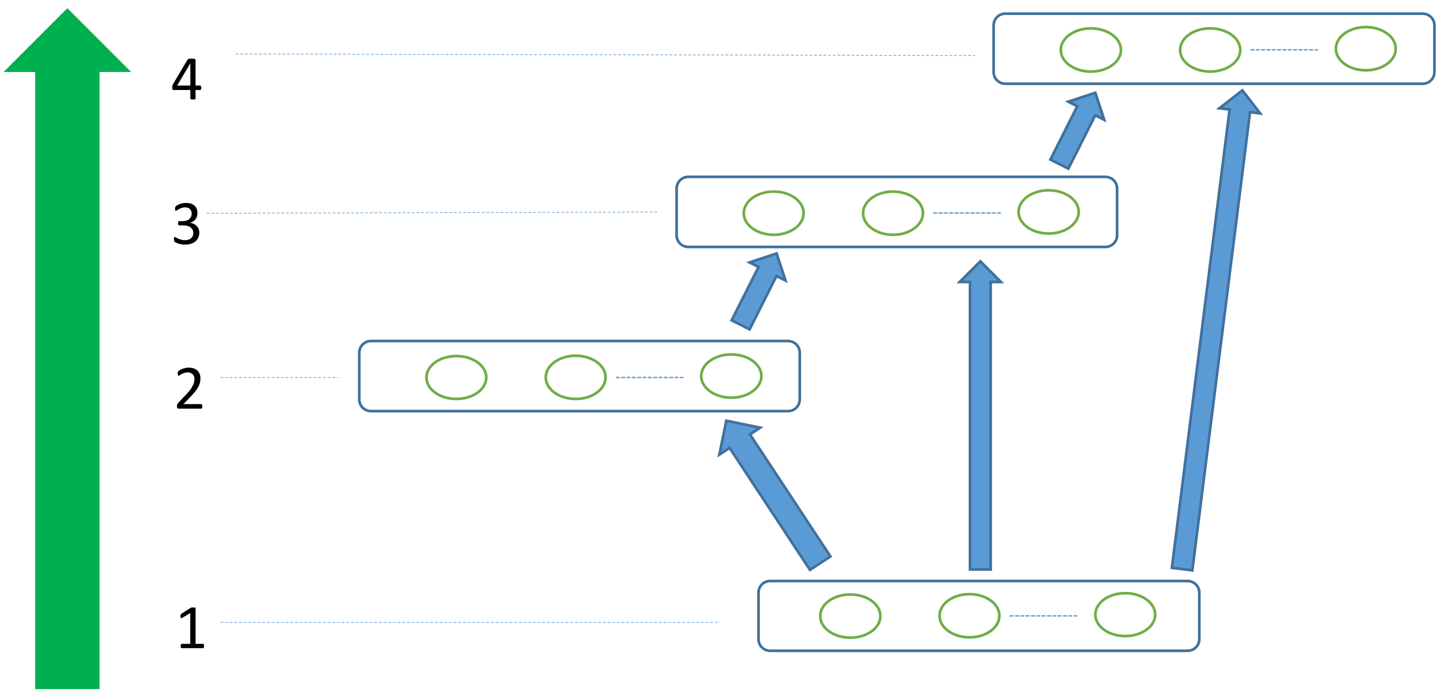
There is a ordering of nodes where all arcs face the same direction (=There is a numbering of nodes where all arcs go from a lower numbered to higher numbered nodes)



Feed-Forward (FF) Neural Networks

When the network has a DAG structure, it is called a feed-forward network

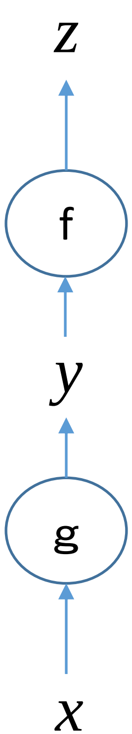
- The nodes can be ordered in a line so that all the connections have the same direction
- The forward propagation is directly applied (by calculating from the bottom to top, it is always guaranteed that the inputs of the current node are already calculated)

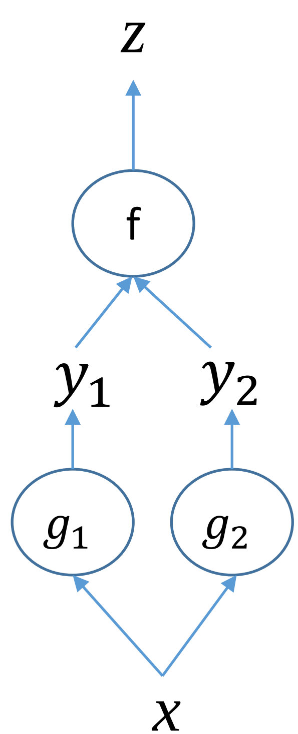


Backpropagation of FF Networks

$$z = f(y)$$
$$y = g(x)$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$





Branches correspond to considering block matrices

$$\begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} & \frac{\partial z_1}{\partial x_3} \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} & \frac{\partial z_2}{\partial x_3} \end{pmatrix} = \left(\begin{array}{c|c} \frac{\partial z_1}{\partial y_1} & \frac{\partial z_1}{\partial y_2} \\ \frac{\partial z_2}{\partial y_1} & \frac{\partial z_2}{\partial y_2} \end{array}\right) \left(\begin{array}{ccc} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \hline \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \end{array}\right)$$

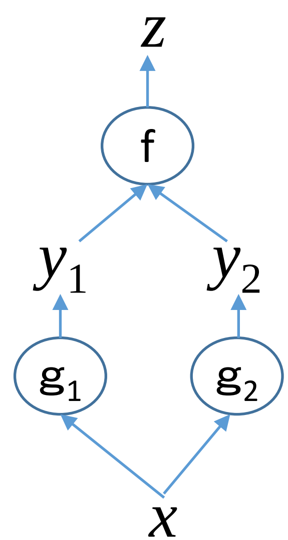
$$= \begin{pmatrix} \frac{\partial z_1}{\partial y_1} \\ \frac{\partial z_2}{\partial y_1} \end{pmatrix} \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \end{pmatrix} + \begin{pmatrix} \frac{\partial z_1}{\partial y_2} \\ \frac{\partial z_2}{\partial y_2} \end{pmatrix} \begin{pmatrix} \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \end{pmatrix}$$

Backpropagation of Skip Structures

$$z = f(y_1, y_2)$$

$$y_1 = g_1(x)$$

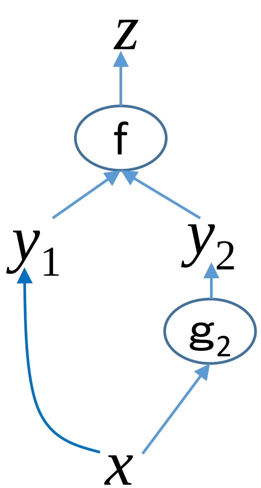
$$y_2 = g_2(x)$$



$$\frac{\partial z}{\partial x} = \begin{bmatrix} \frac{\partial z}{\partial y_1} & \frac{\partial z}{\partial y_2} \end{bmatrix} \begin{bmatrix} \frac{\partial y_1}{\partial x} \\ \frac{\partial y_2}{\partial x} \end{bmatrix}$$

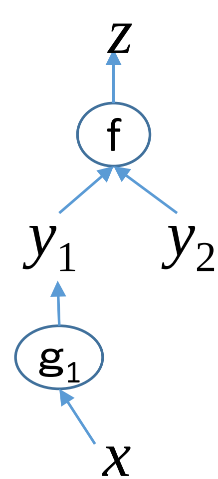
$$= \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$

Skip structures are just special cases of fully connected structures



$$y_1 = g_1(x) \quad = x$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$

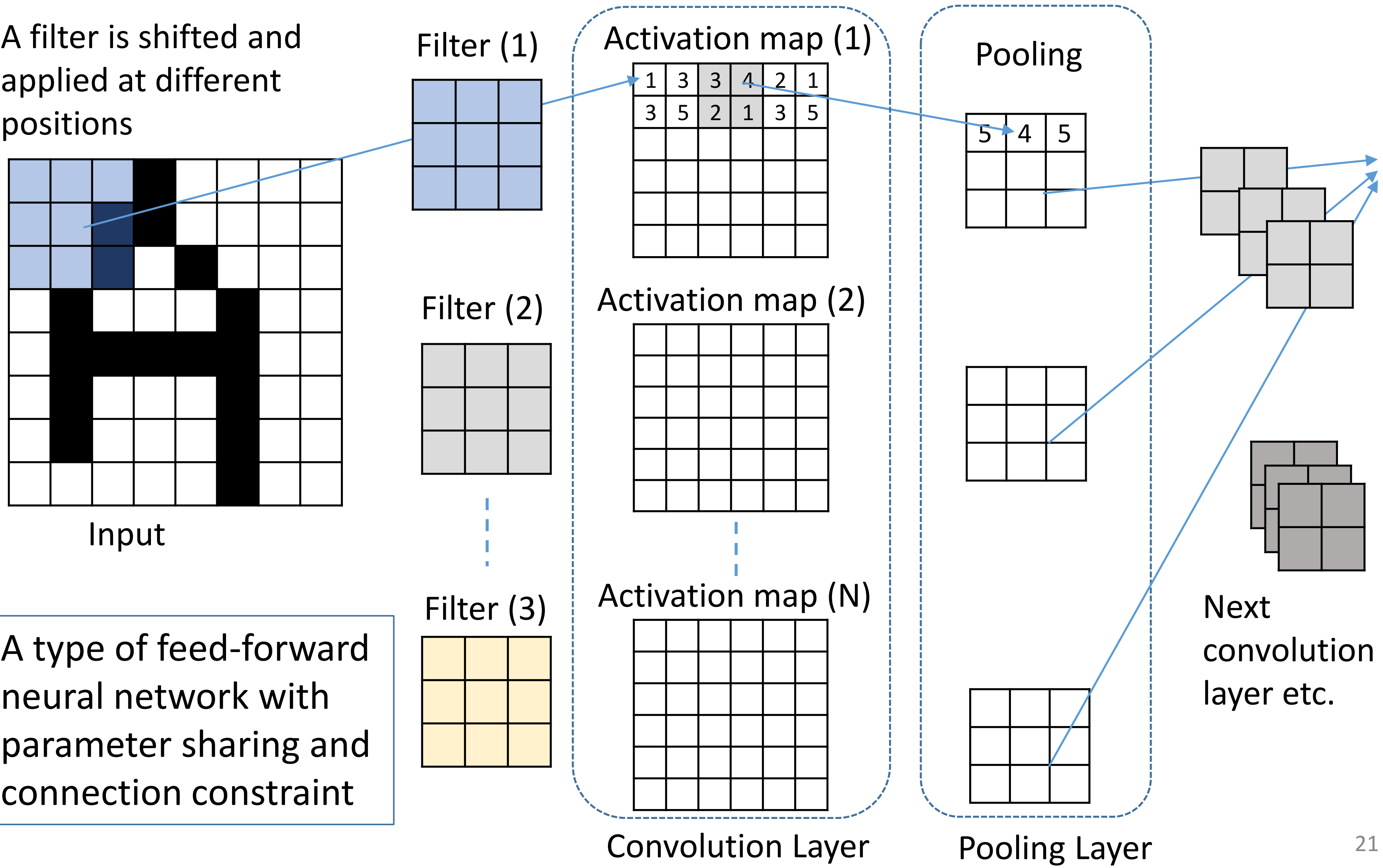


$$y_2 = g_2(x) \quad = C$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x}$$

Convolutional Neural Network (CNN)

A filter is shifted and applied at different positions

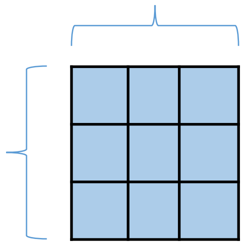


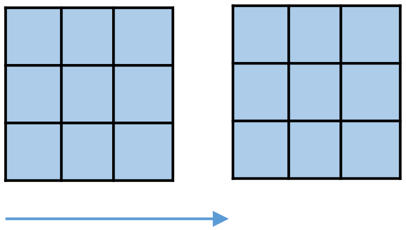
A type of feed-forward neural network with parameter sharing and connection constraint

Kernel Size, Stride, and Padding

Kernel Size (e.g. 3x3)

Size of the filter

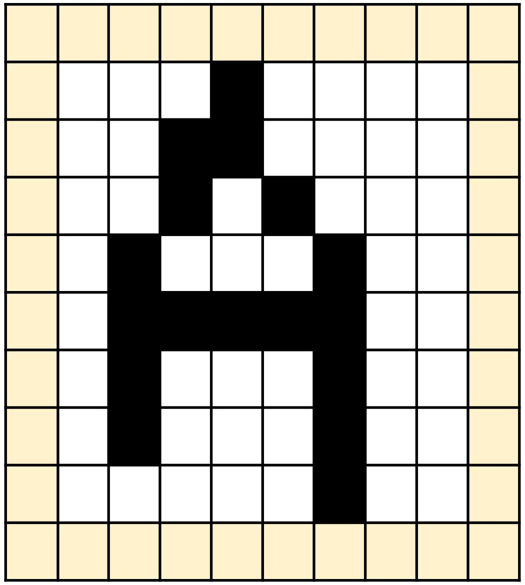




Stride (e.g. 4)

Sift size

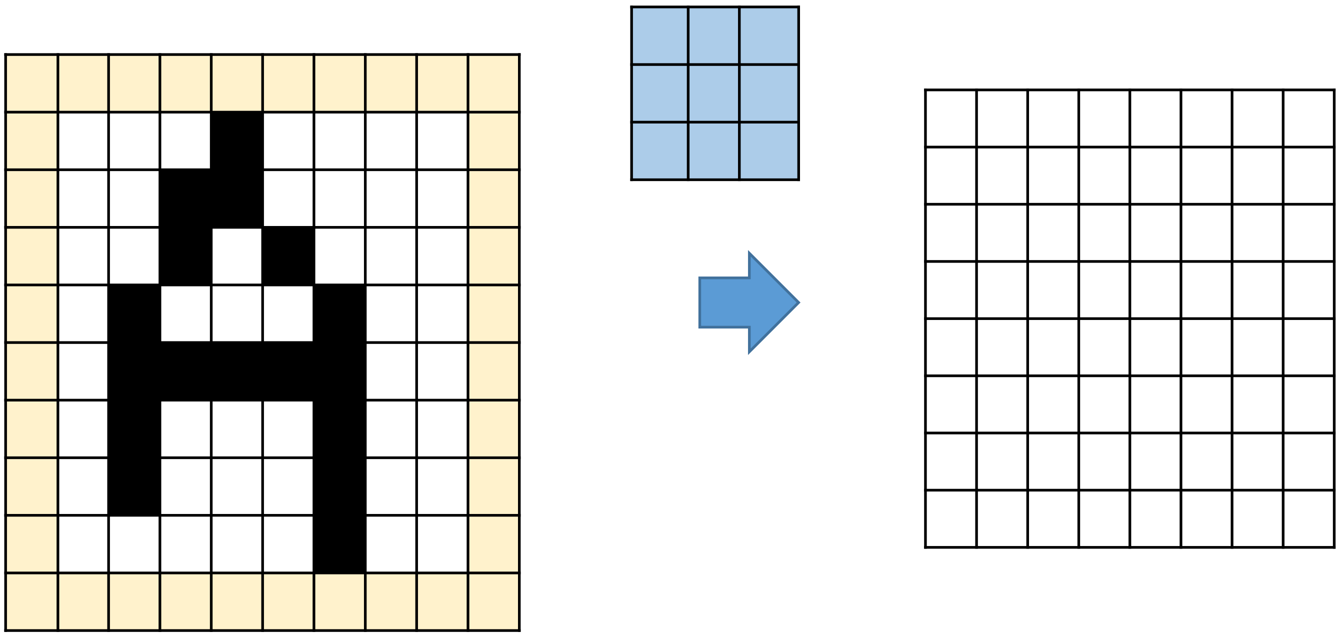
Padding (e.g. 1)



Extend the original input size and fill the extended pixels with a constant value (e.g. 0).

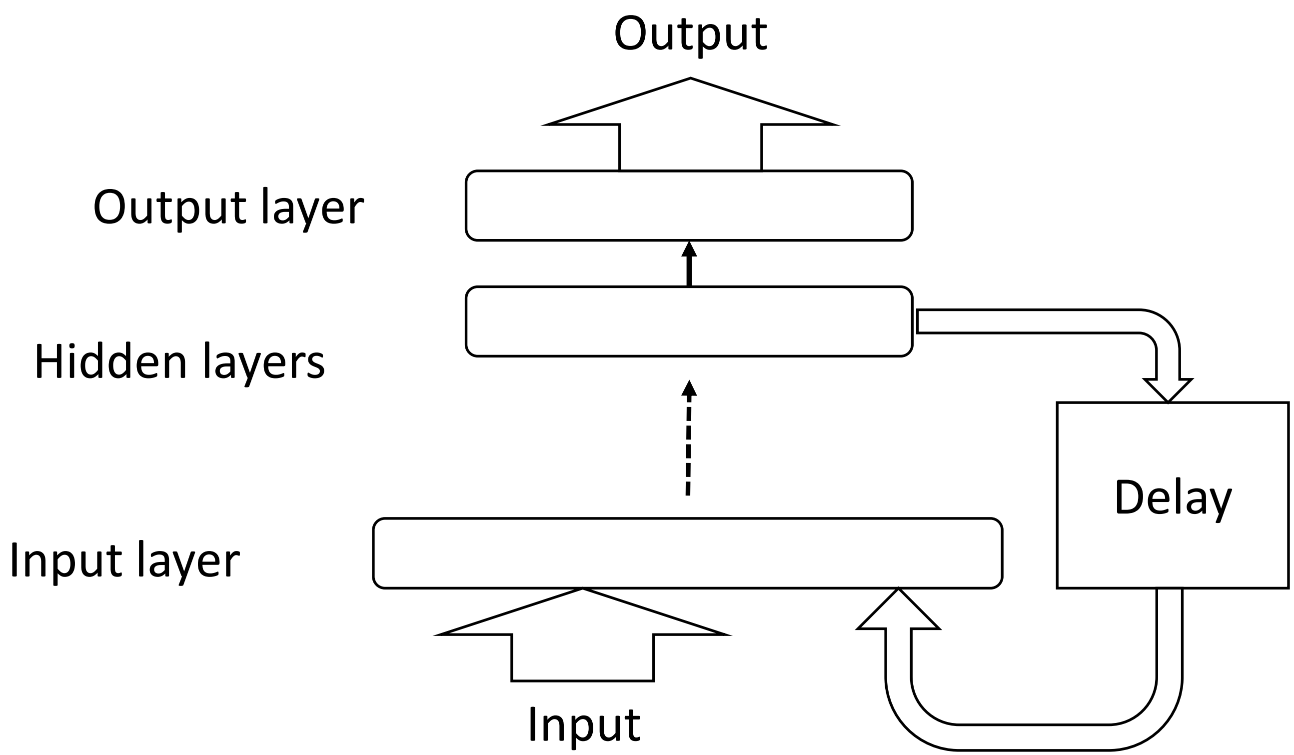
Example

- Kernel size = 3x3, Stride = 1, Padding = 1

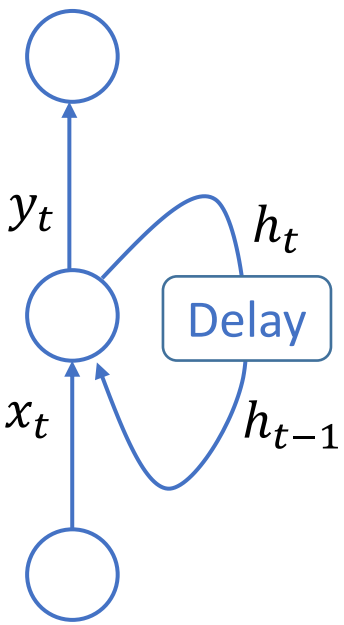


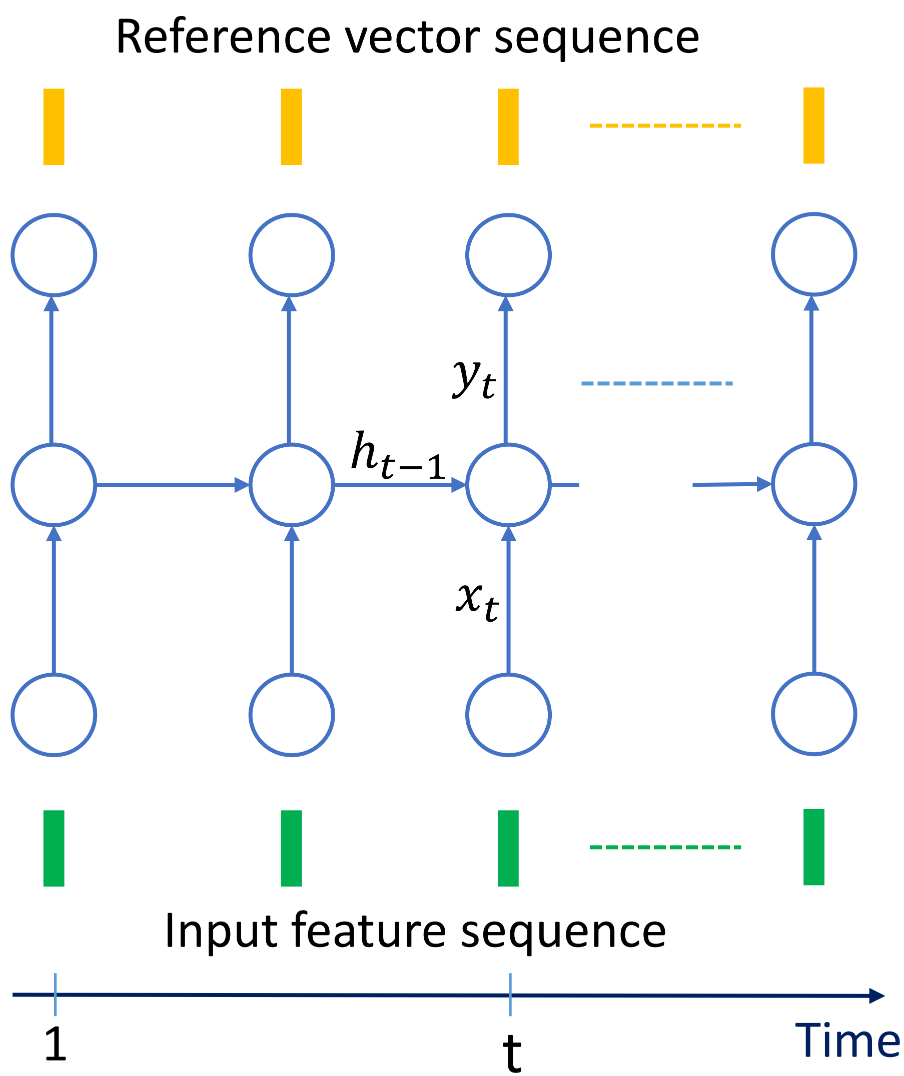
Recurrent Neural Network (RNN)

- Neural network having a feedback
  - More flexible in model design than feed-forward NN, but the training is more difficult



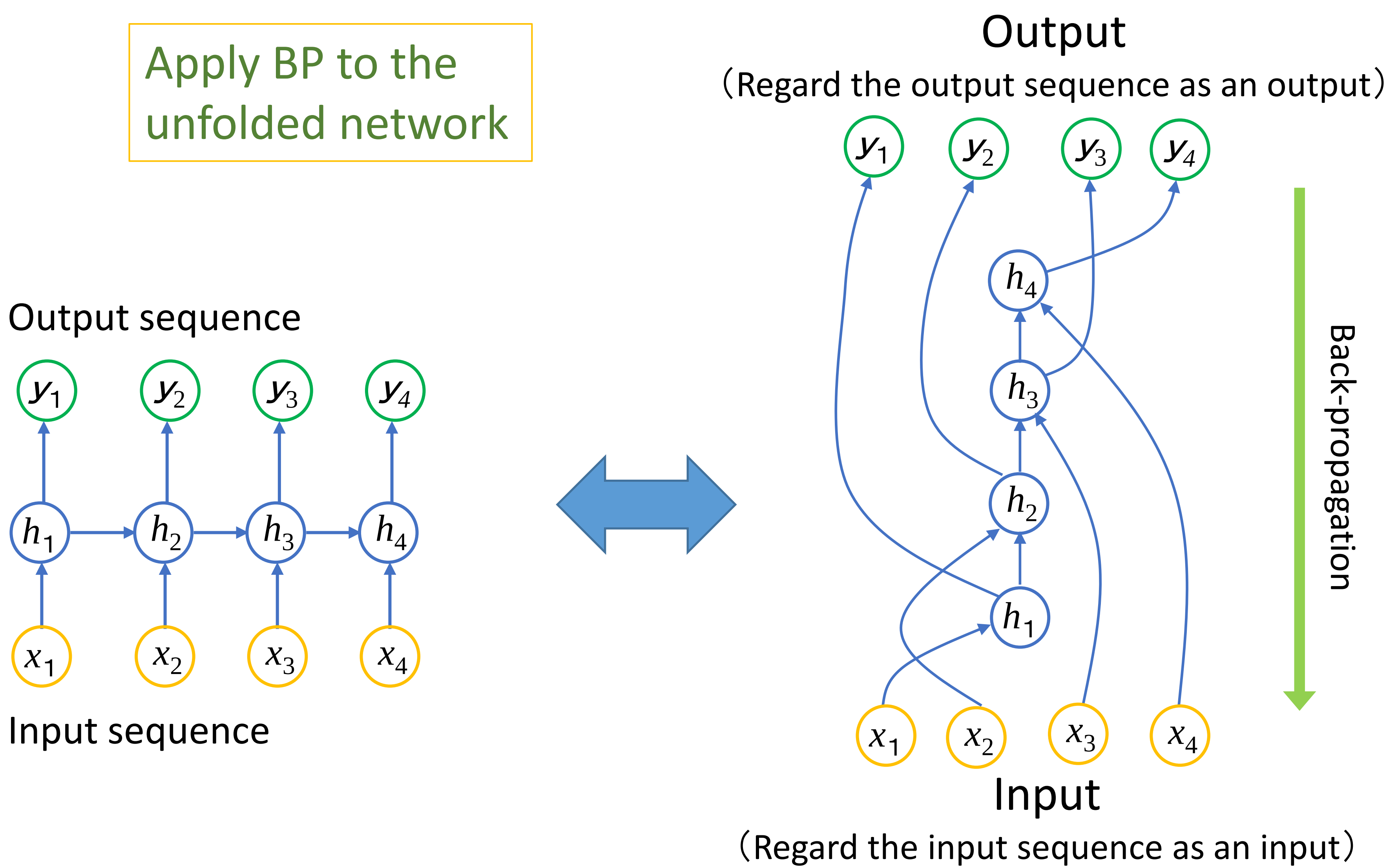
Unfolding of RNN to Time Axis





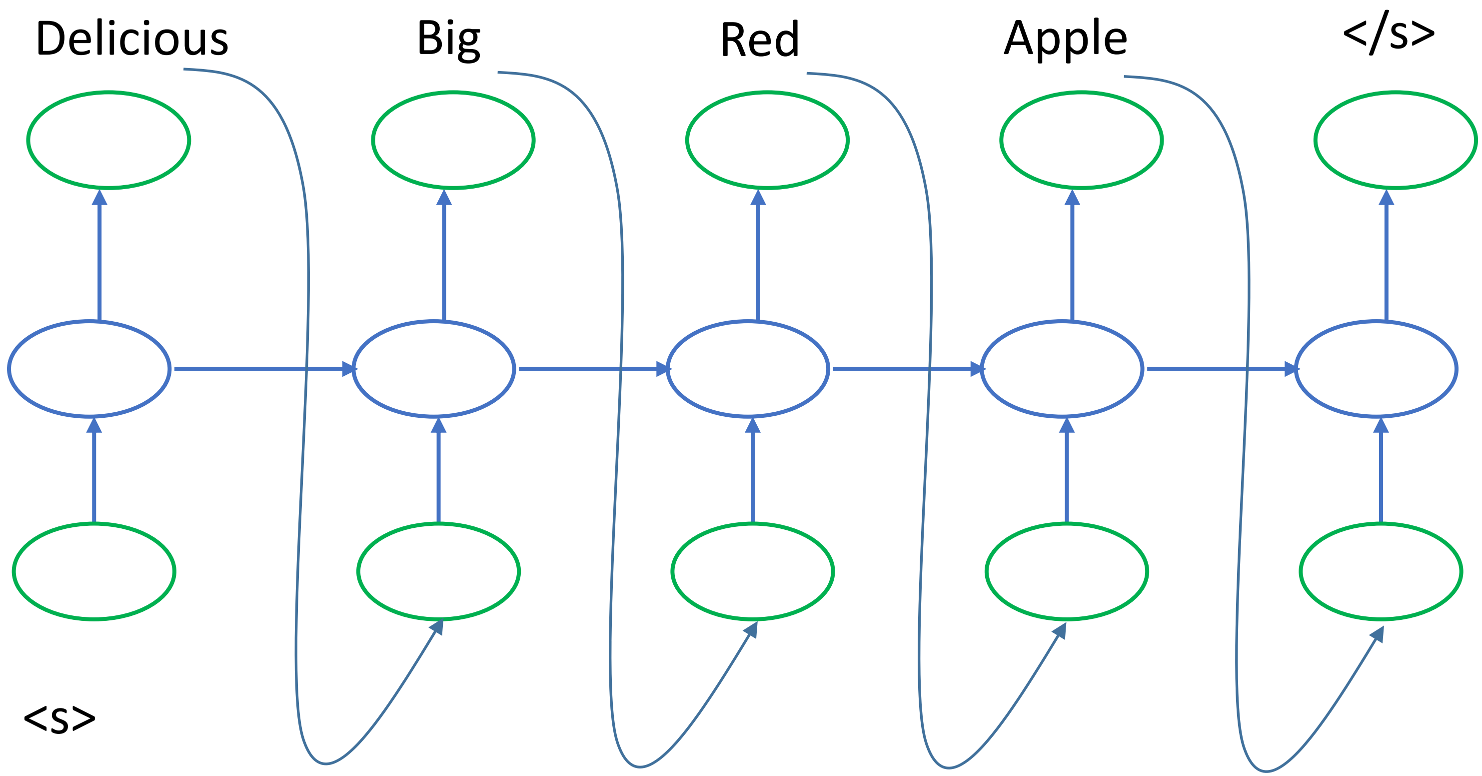
Training of RNN by BP Through Time (BPTT)

Apply BP to the unfolded network



RNN Language Model

P(<s>, Delicious, Big, Red, Apple, </s>)



Exercise (Q3.1, Q3.2 Q3.3)

Q3.1)

Obtain $\frac{\partial Err}{\partial a}$ for the following neural network when the input $x = 2.0$ and $< a, b, c, d > = < 1,1,1,1 >$

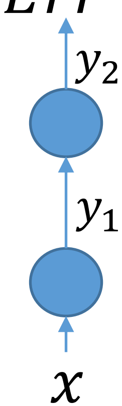
Q3.2)

Obtain $\frac{\partial Err}{\partial d}$ when the input $x = 2.0$ and $< a, b, c, d > = < 1,1,1,1 >$

Q3.3)

Obtain $\frac{\partial Err}{\partial d}$ when the input $x = 0.5$ and $< a, b, c, d > = < 2,1,3,0 >$

$$Err = (y_2 - 15)^2$$

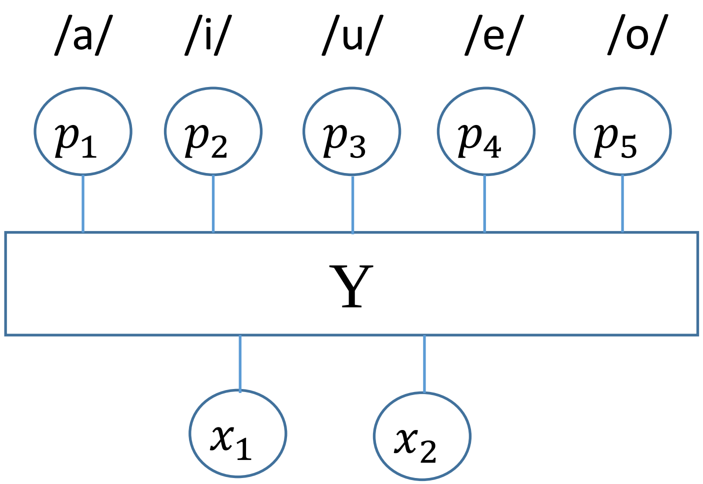


$$y_2 = (ay_1 + b)^2$$

$$y_1 = cx + d$$

Exercise (Q3.4, Q3.5)

Consider the following neural network with a softmax output layer to recognize a Japanese vowel.



$$p_i = \frac{\exp(y_i)}{\sum_j \exp(y_j)}$$

$$Y^T = [y_1, y_2, y_3, y_4, y_5]$$
$$= \begin{bmatrix} 1 & 1.5 & -1 & 1 & 0.5 \\ 2 & 0.1 & 2.5 & -1 & 1 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Q3.4)

What is the recognition result when $X^T = [x_1\ x_2] = [1,1]$?

Q3.5)

What is the recognition result when $X^T = [2, -1]$?