Speech and Language Processing

Lecture 5

Neural network based acoustic and language models

Information and Communications Engineering Course

Takahiro Shinozaki

Lecture Plan (Shinozaki's part)

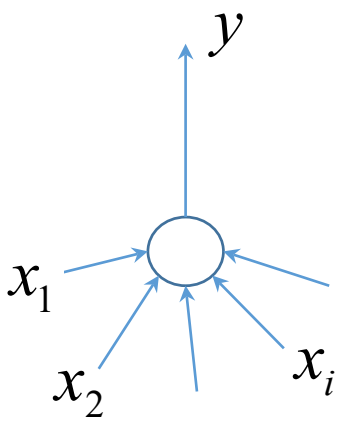
I gives the first 6 lectures about speech recognition. Through these lectures, the backbone of the latest speech recognition techniques is explained.

1. 10/19 (remote)
   Speech recognition based on GMM, HMM, and N-gram
2. 10/19 (remote)
   Maximum likelihood estimation and EM algorithm
3. 10/20 (remote)
   Bayesian network and Bayesian inference
4. 10/20 (remote)
   Variational inference and sampling
5. 10/22 (remote)
   Neural network based acoustic and language models
6. 10/22 (remote)
   Weighted finite state transducer (WFST) and speech decoding

Neural network

Multi Layer Perceptron (MLP)

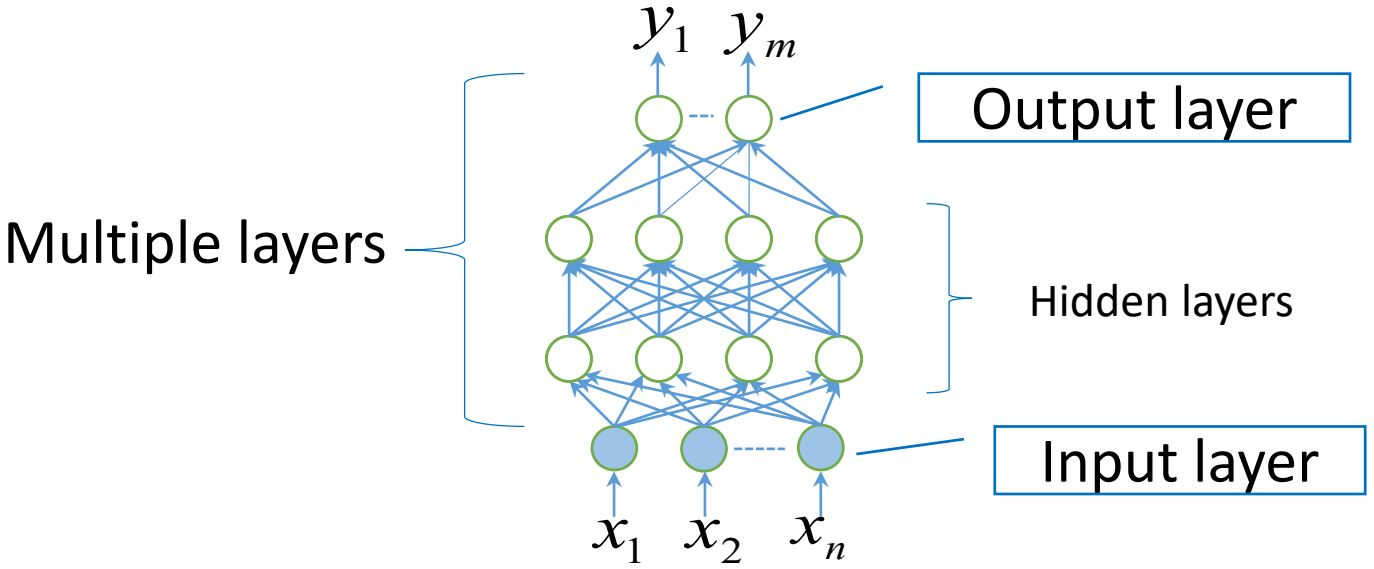
- Unit of MLP



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

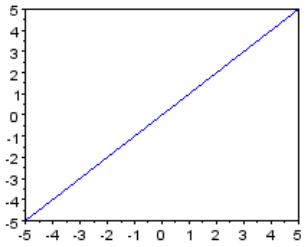
h: activation function
w: weight
b : bias

- MLP consists of multiple layers of the units

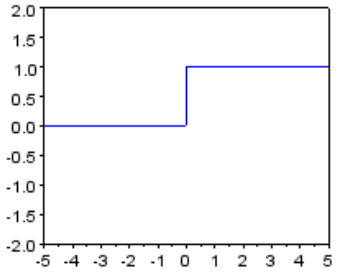


Activation Functions

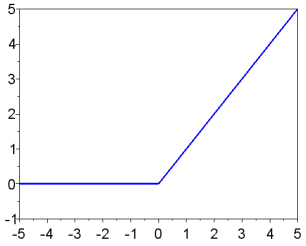
- Linear function $h(x) = x$



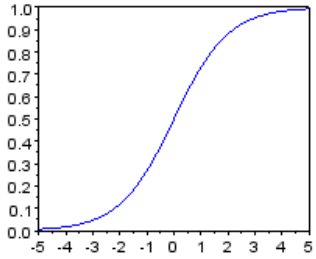
- Unit step function $h(x) = \begin{cases} 1 & \text{if } 0 \le x \\ 0 & \text{otherwise} \end{cases}$



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

Expresses a probability distribution

- Example

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

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

Exercise 5.1

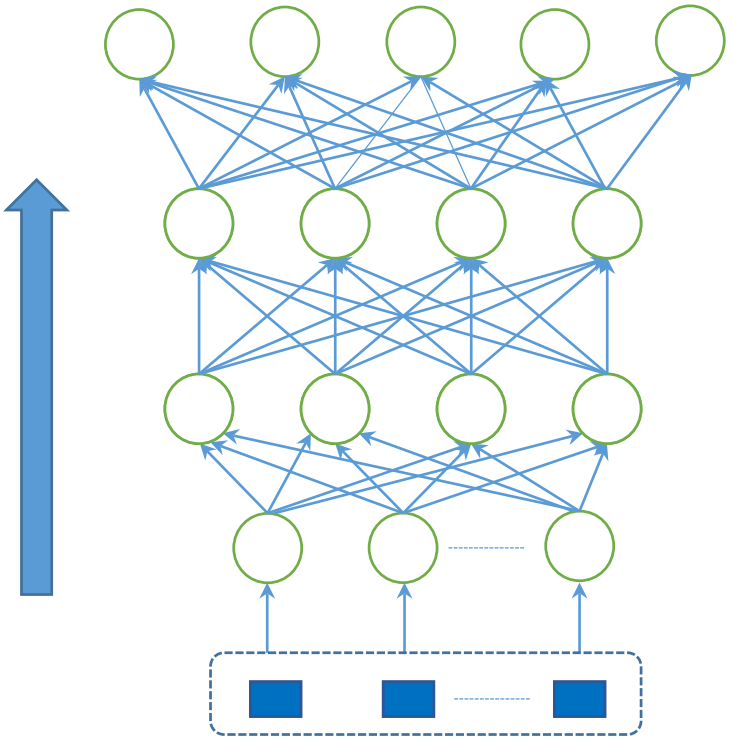
- Let h be a softmax function having inputs z_1, z_2,…,z_N.

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

- Prove that $\sum_{i=1}^{N} h(z_i) = 1.0$

Forward Propagation

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



E.g. softmax layer

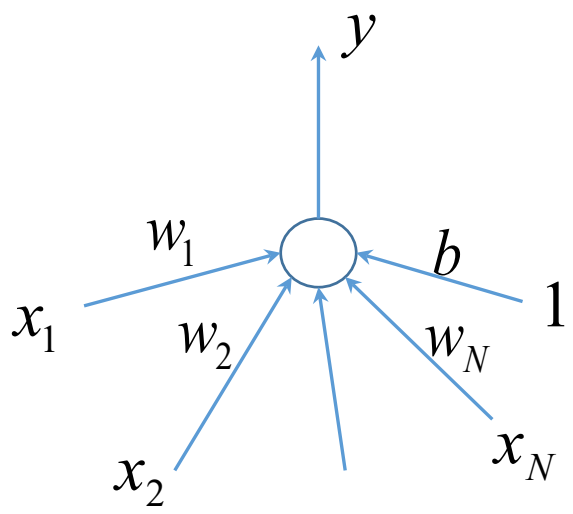
E.g. sigmoid layer

E.g. sigmoid layer

Input vector

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

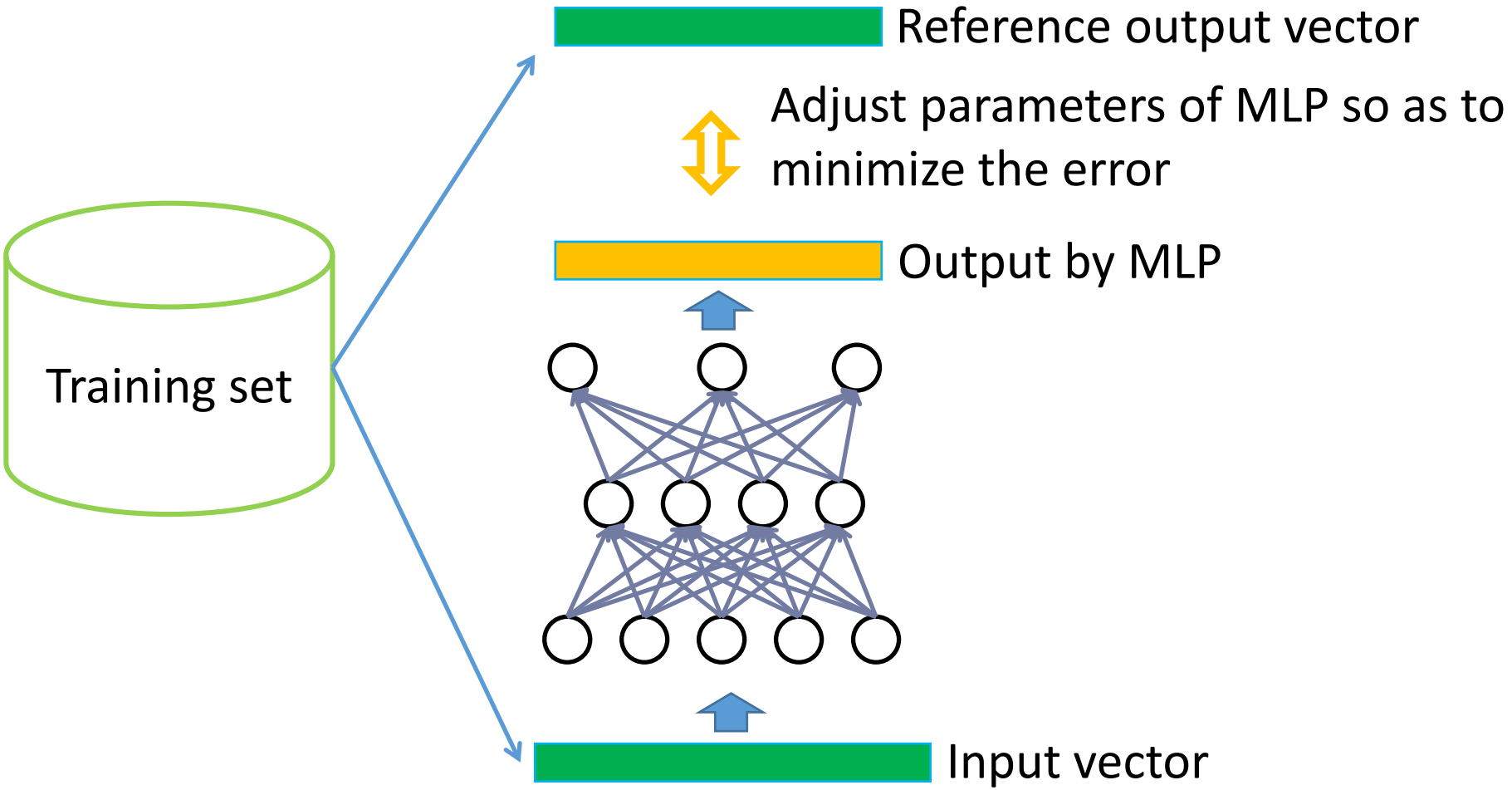
$w=(w_1,w_2,\ldots,w_N,b)$

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 NN Training



Definitions of Errors

- Sum of square error
  - Used when output layer uses linear functions

$$E(W) = \frac{1}{2}\sum_n \left\| y(X_n, W) - t_n \right\|^2$$

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 when the output layer is a softmax

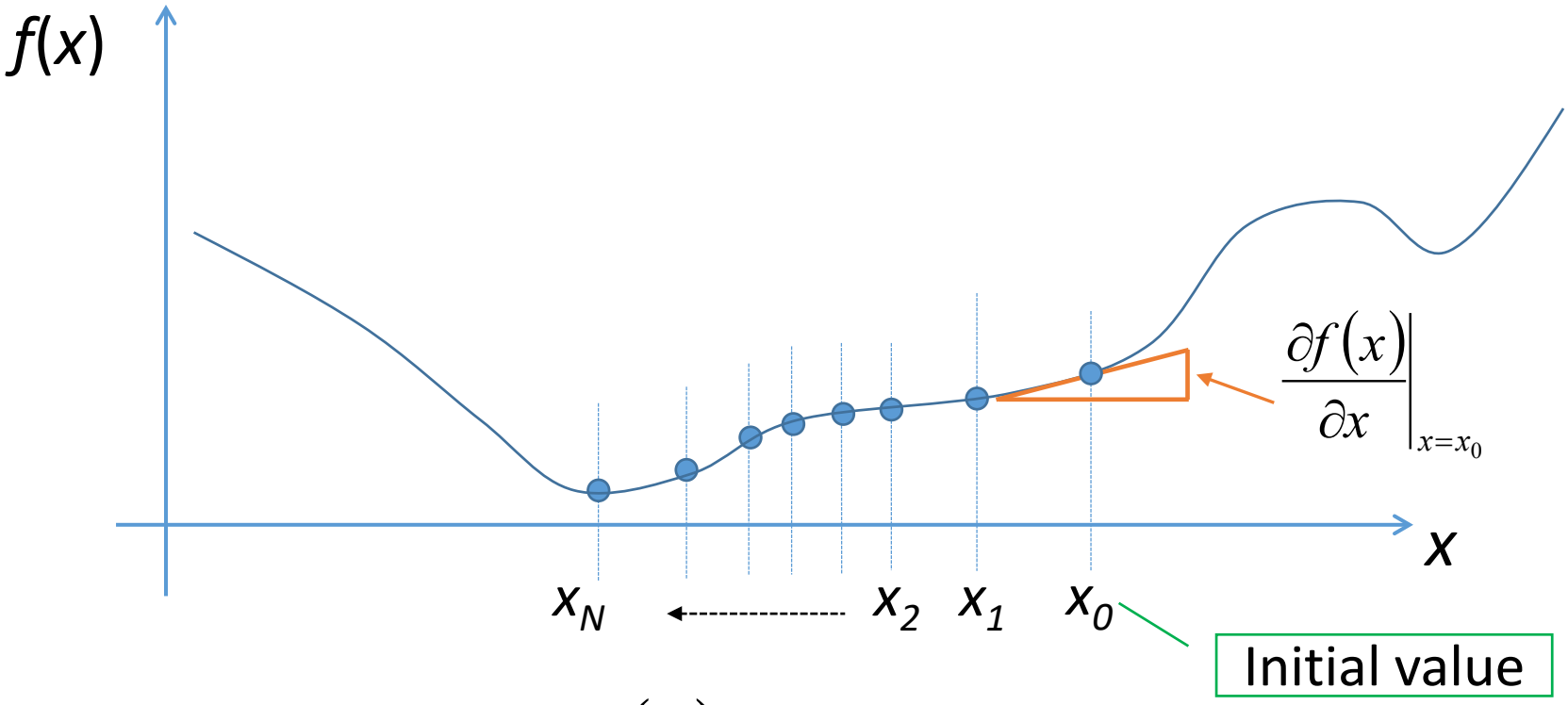
$$E(W) = -\sum_n \sum_k \left\{ t_{nk} \ln y_{nk}(X_n, W) \right\}$$

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

k : Index of 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)

MLP Training by Gradient Descent

- Define an error measure $E(W)$ for training samples

e.g. $$E(W)=\frac{1}{2}\sum_n \left\| y(X_n, W)-t_n \right\|^2$$

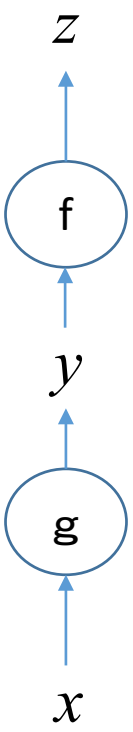
- Initialize parameters $W=\{w_1, w_2,\ldots, w_M\}$
- Repeatedly update the parameter set using gradient descent

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

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:

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

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

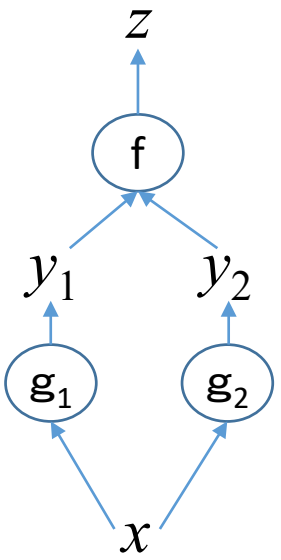
The same rule holds using Jacobian matrix

Jacobian matrix

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

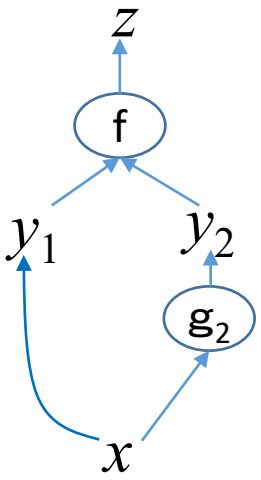
When There Are Branches

$$z = f(y_1, y_2)$$
$$y_1 = g_1(x)$$
$$y_2 = g_2(x)$$



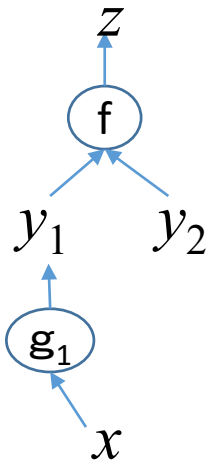
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$

Variations:



$$g_1(x) = x$$

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

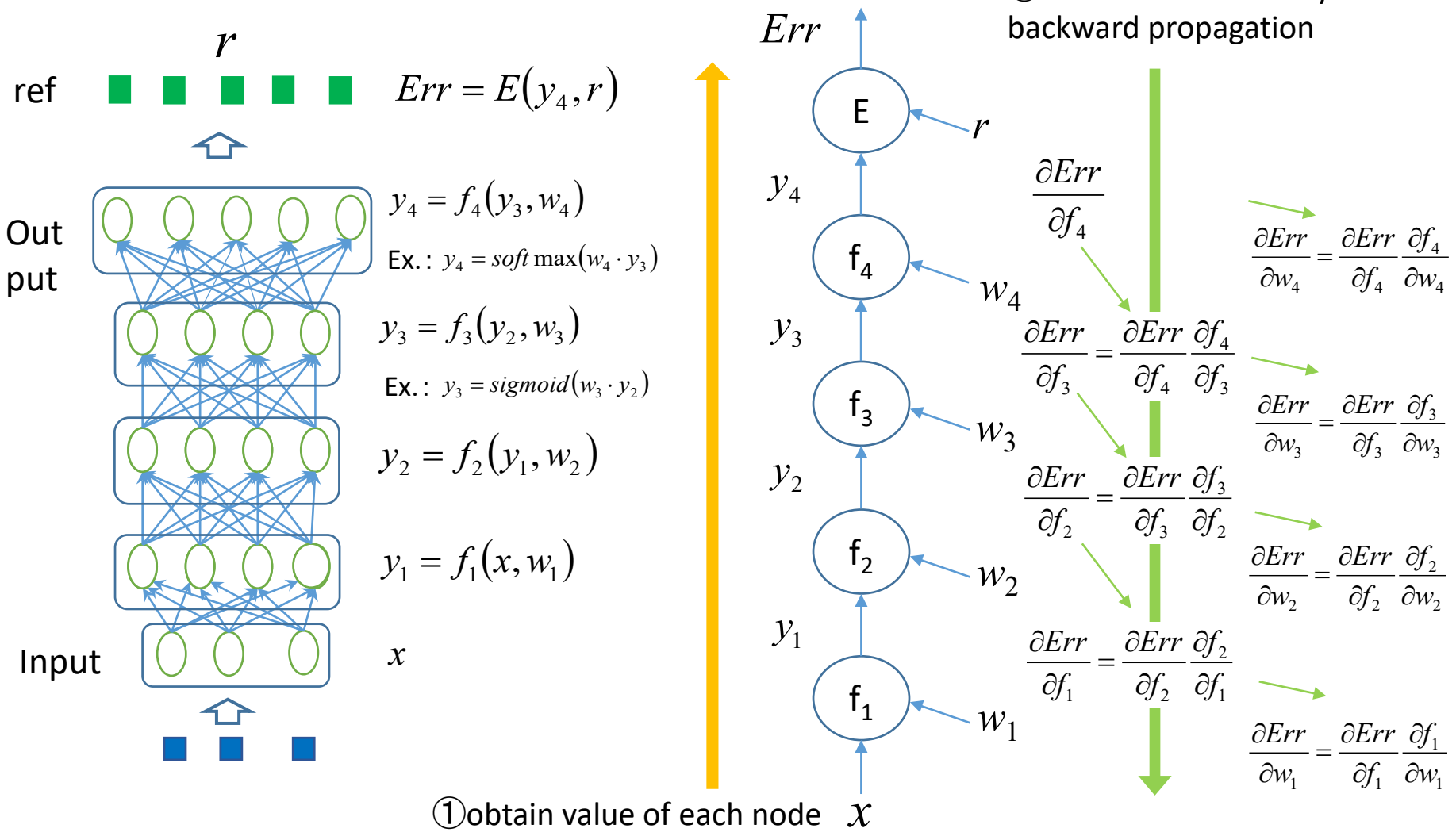


$$g_2(x) = C$$

(independent of x)

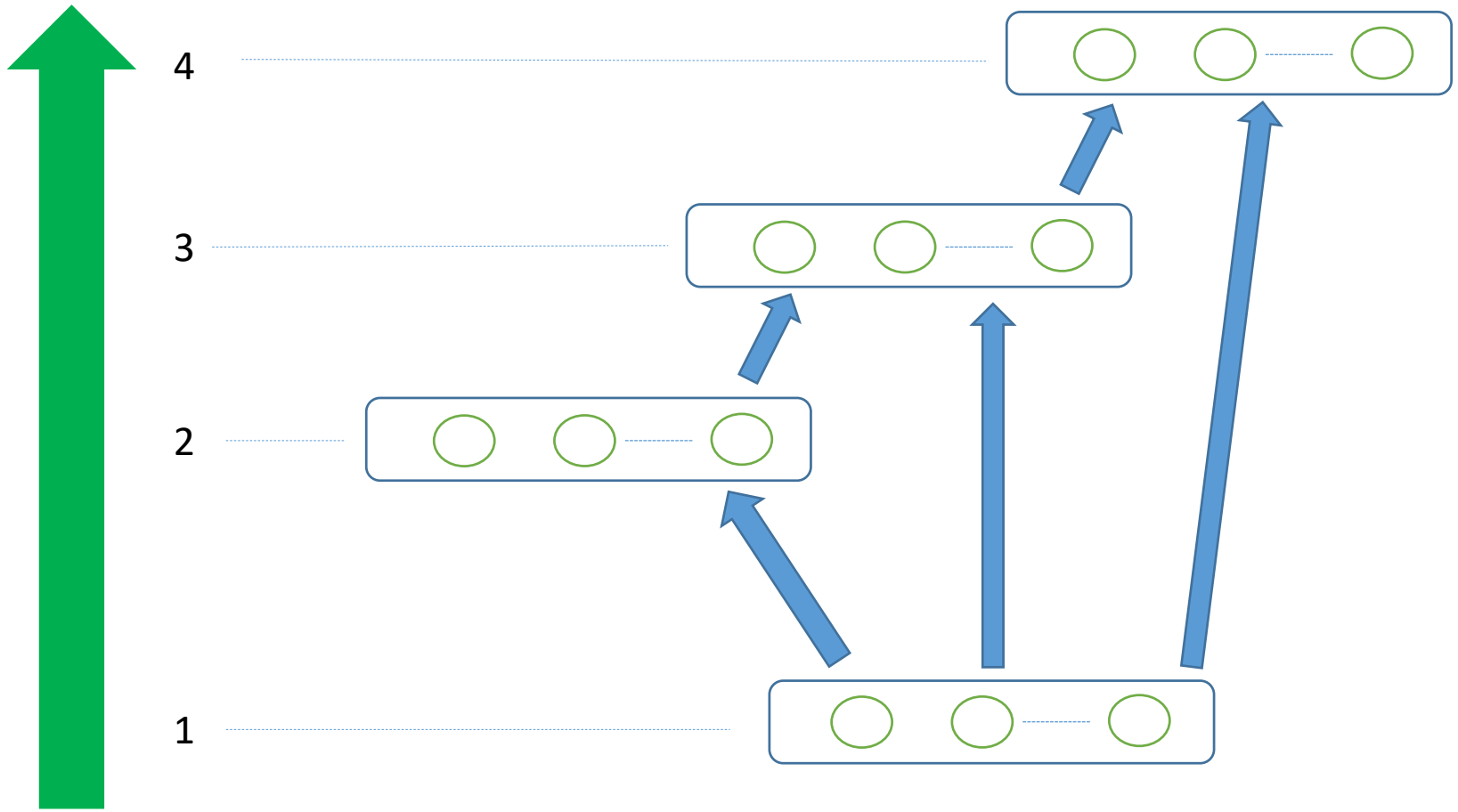
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x}$$

Back Propagation(BP)



Feed-Forward Neural Network

- When the network structure is a DAG, it is called feed-forward network
- The nodes are ordered in a line so that all connections have the same direction
- The forward/backward propagation can be efficiently applied



Exercise 5.2

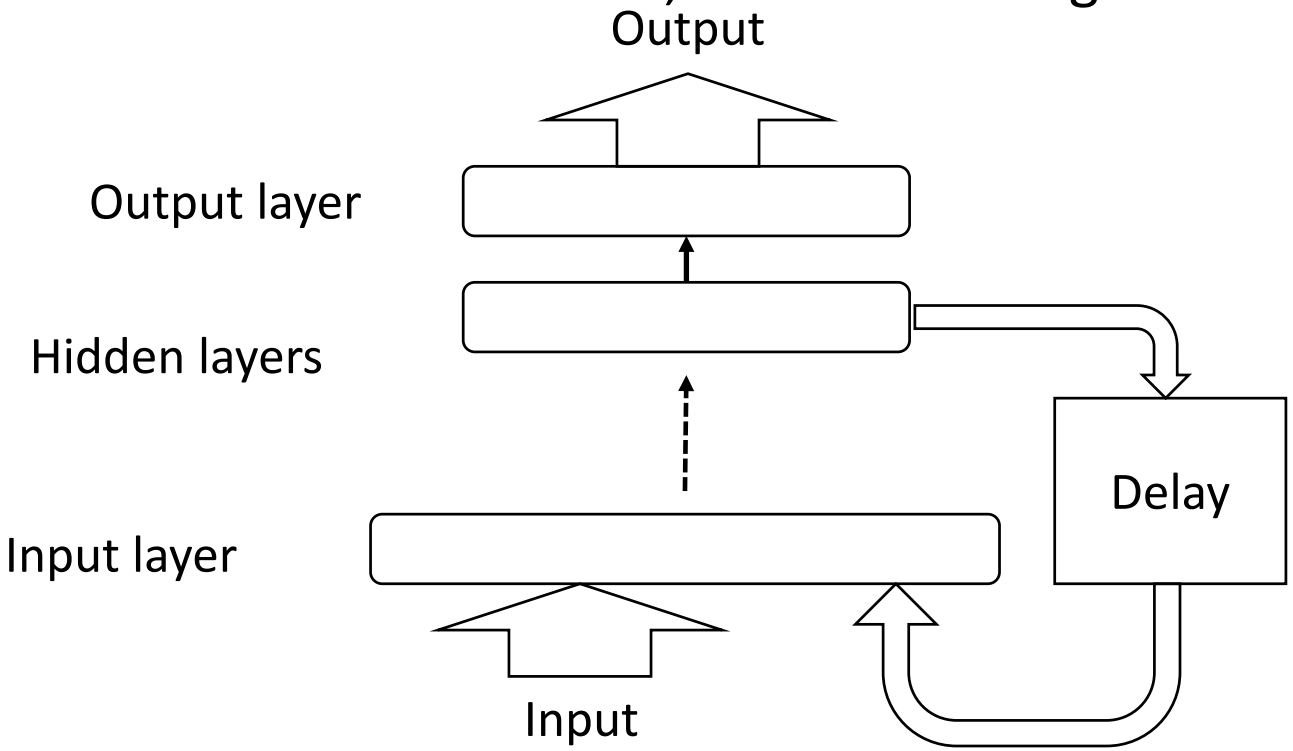
When h(y) and y(x) are given as follows, obtain $\frac{\partial h}{\partial x}$

$$h(y) = \frac{1}{1+\exp(-y)}$$

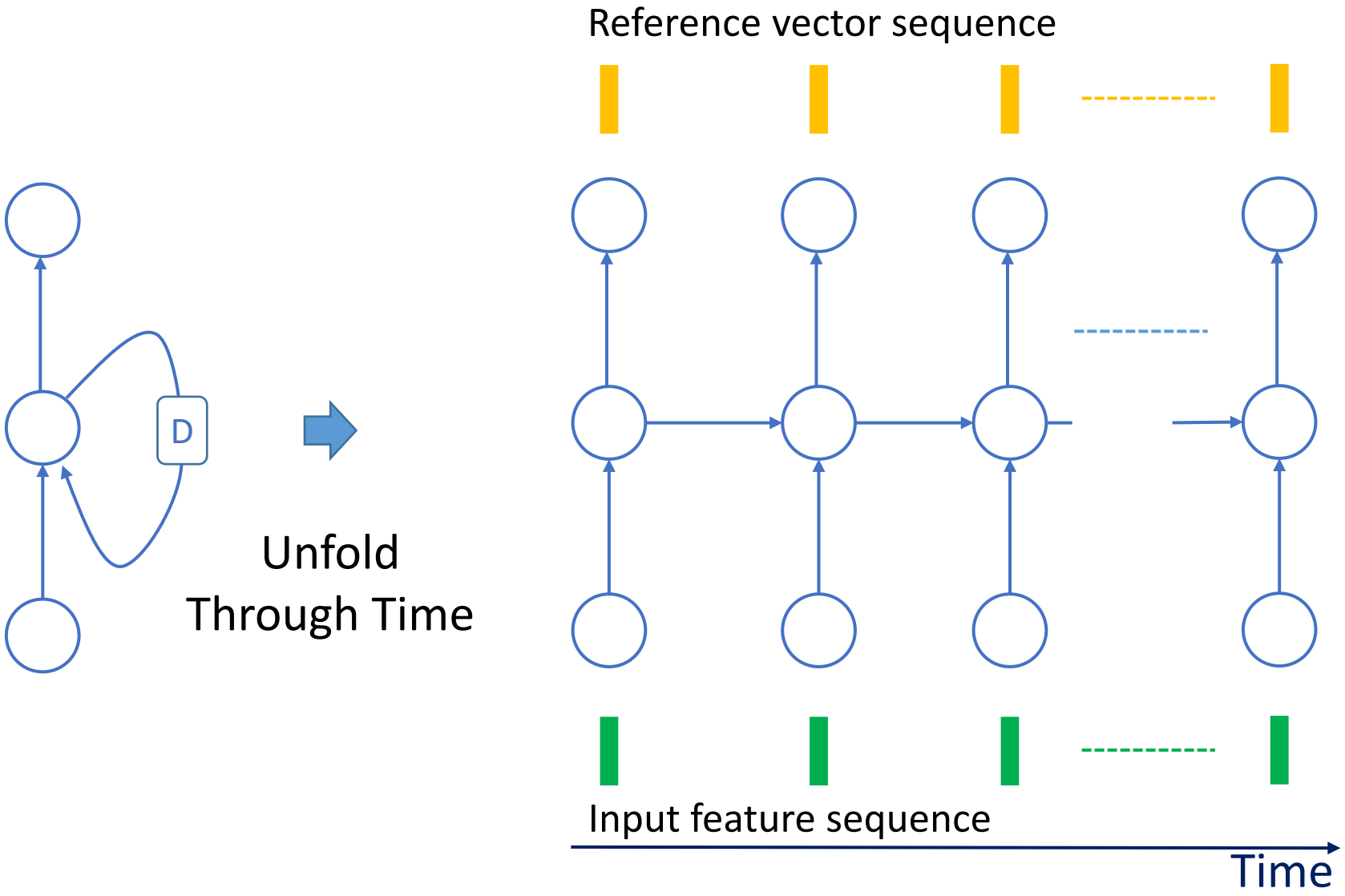
$$y = ax + b$$

Recurrent Neural Network (RNN)

- Neural network having a feedback
  - Expected to be more powerful modeling performance than feed-forward MLP, but the training is more difficult

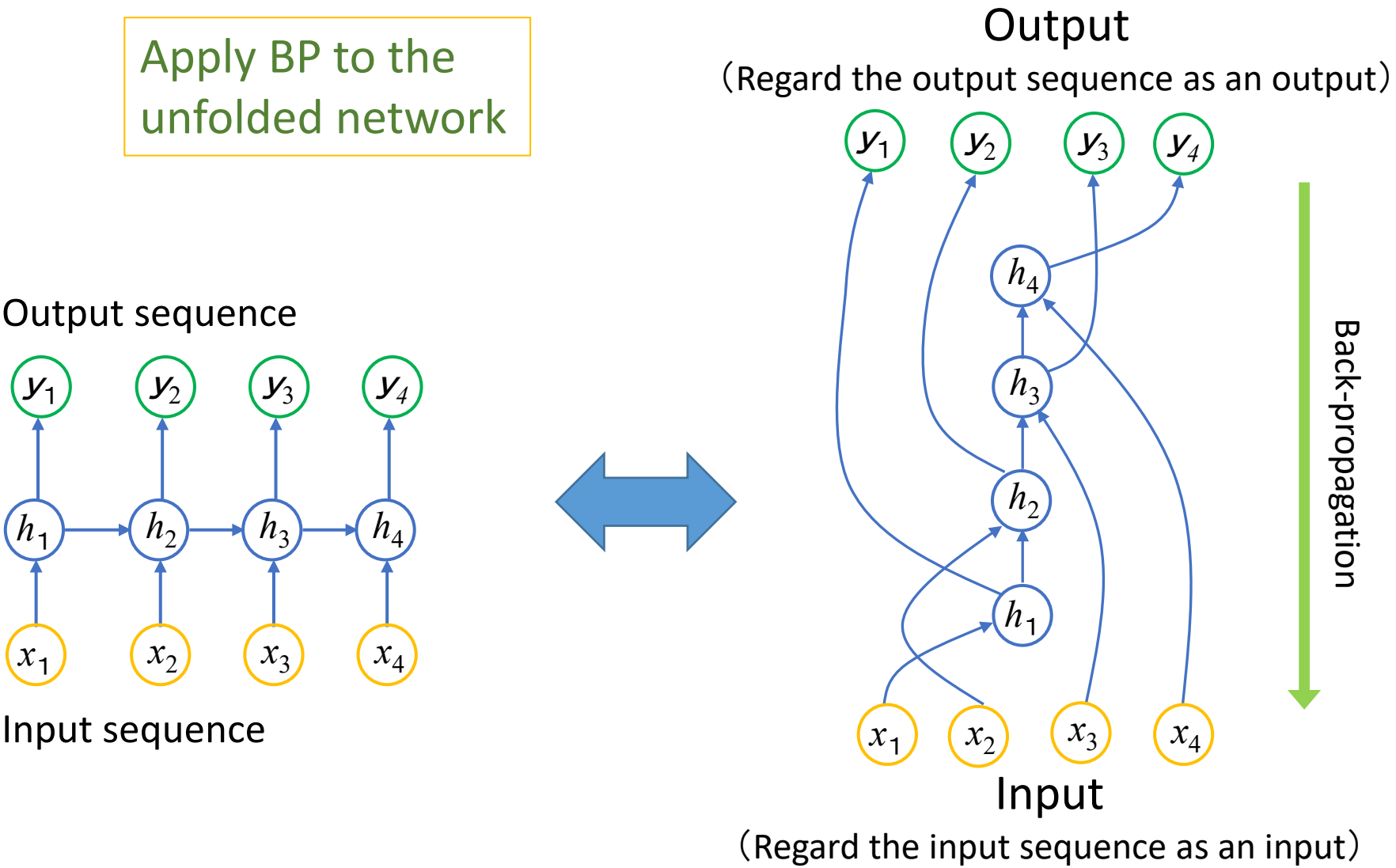


Unfolding of RNN to Time Axis



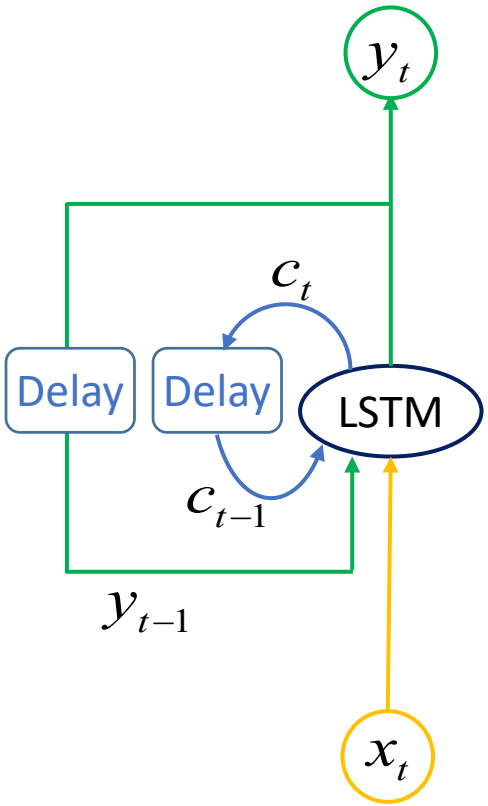
Training of RNN by BP Through Time (BPTT)

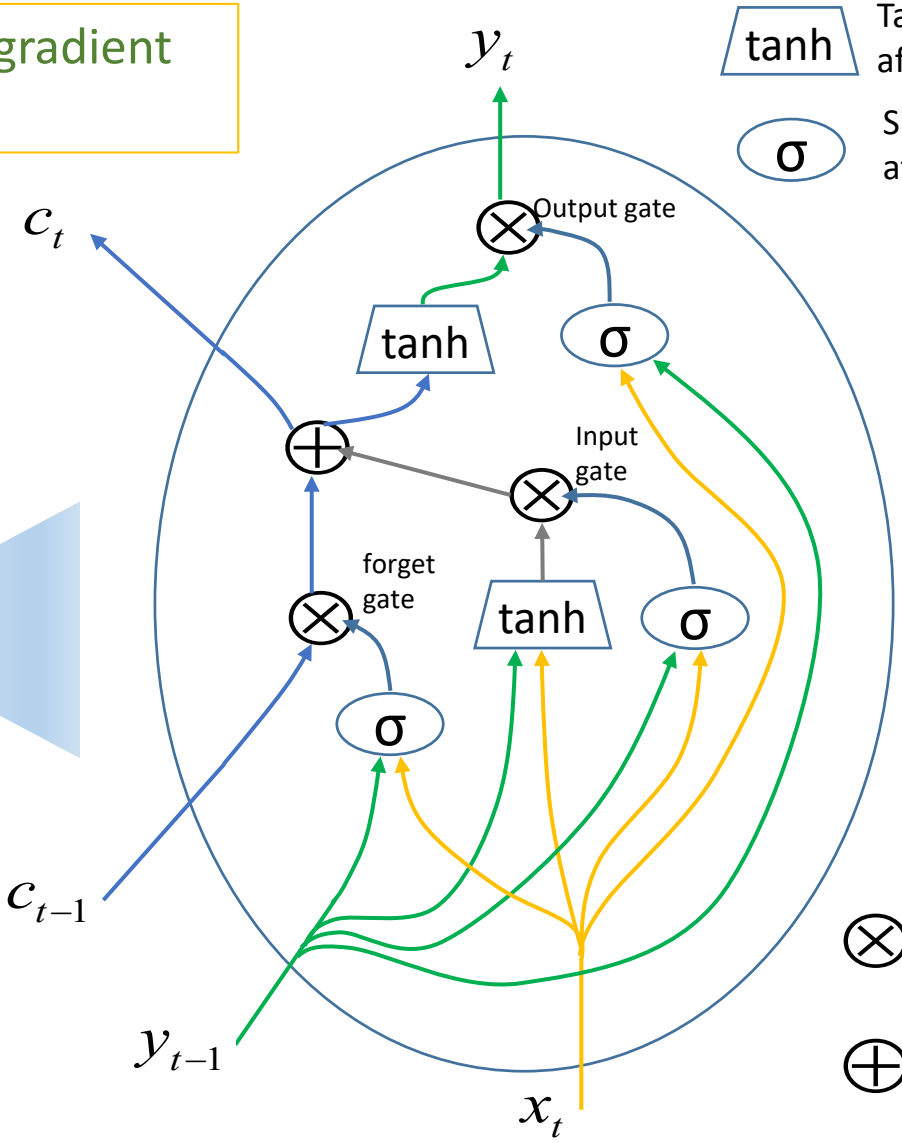
Apply BP to the unfolded network

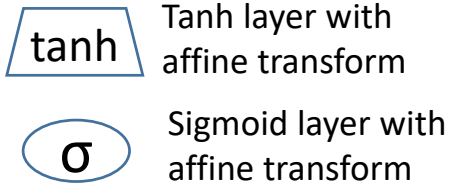


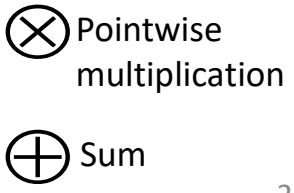
Long Short-Term Memory (LSTM)

A type of RNN addressing the gradient vanishing problem

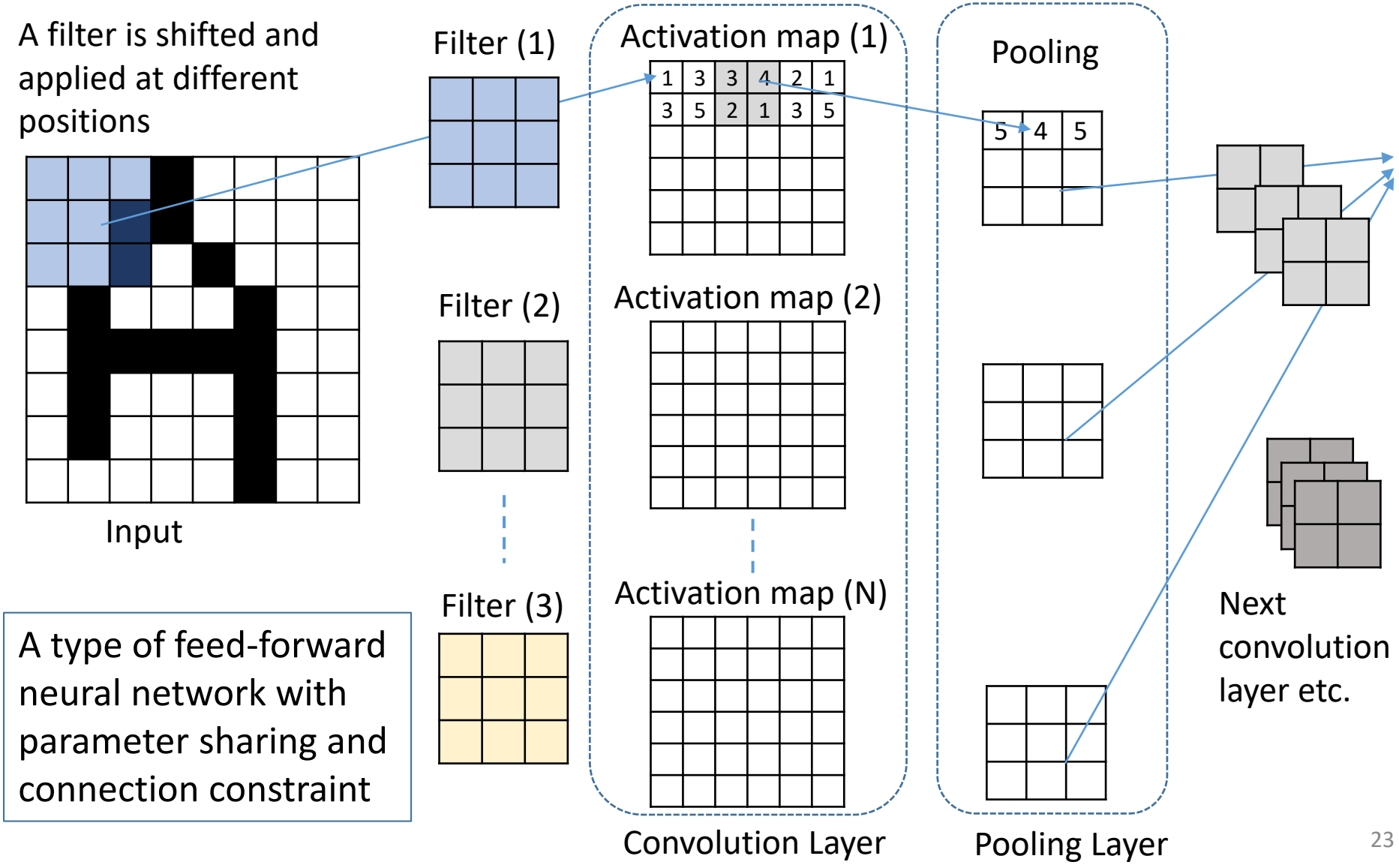






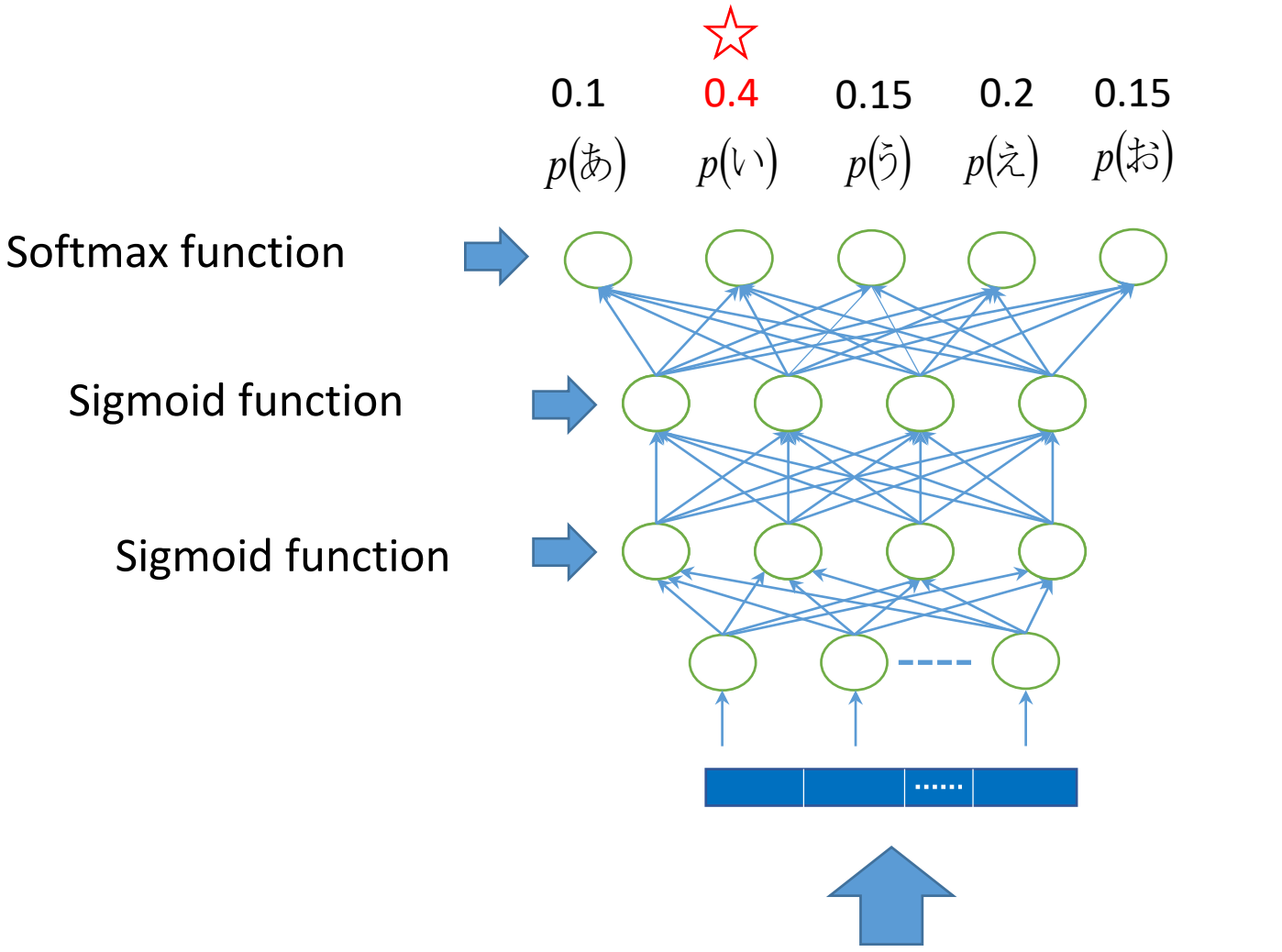


Convolutional Neural Network (CNN)



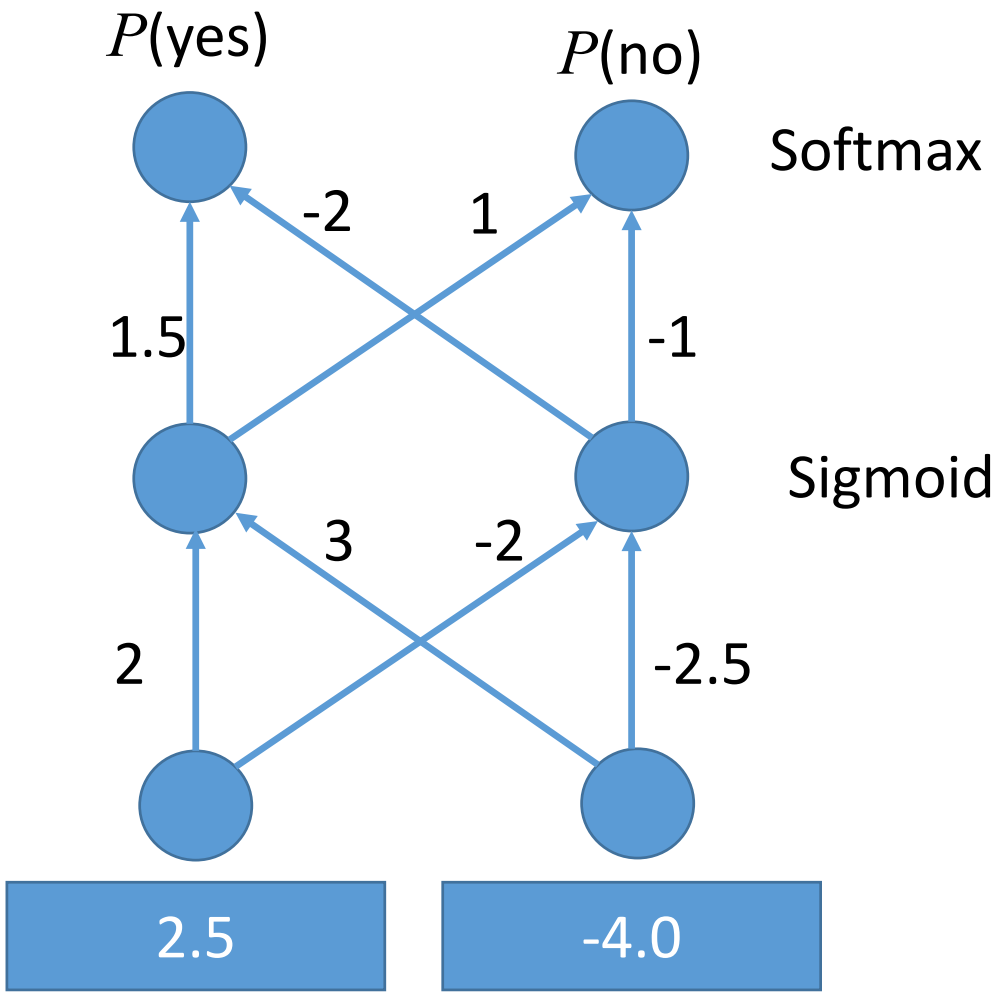
Neural network based acoustic model

Frame Level Vowel Recognition Using MLP



Exercise 5.3

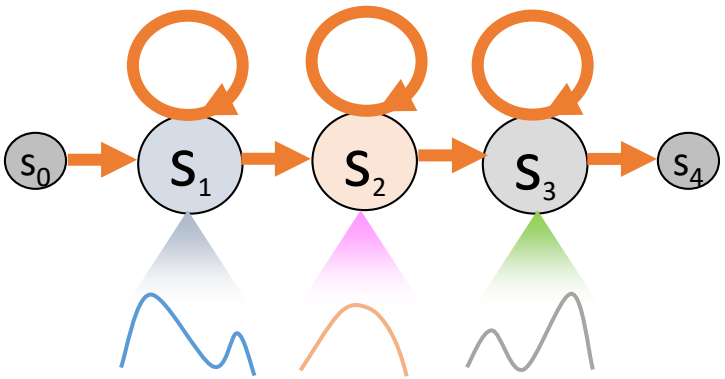
Obtain recognition result (yes or no), and its probability. You may use a calculator.



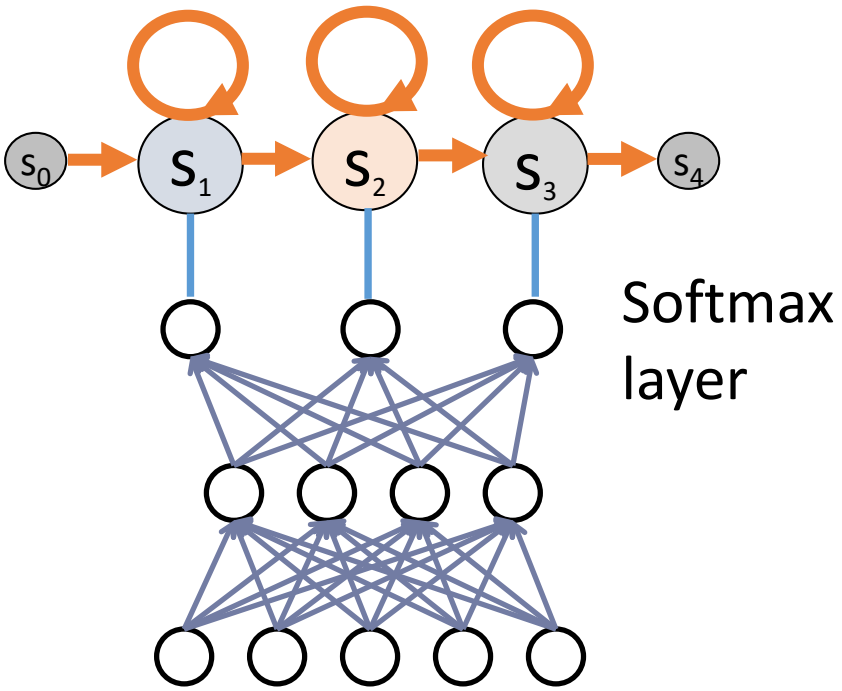
Combination of HMM and MLP

$$p(X \mid s) = GMM_s(X)$$

$$p(X \mid s) \propto \frac{p(s \mid X)}{p(s)} = \frac{MLP(s \mid X)}{p(s)}$$

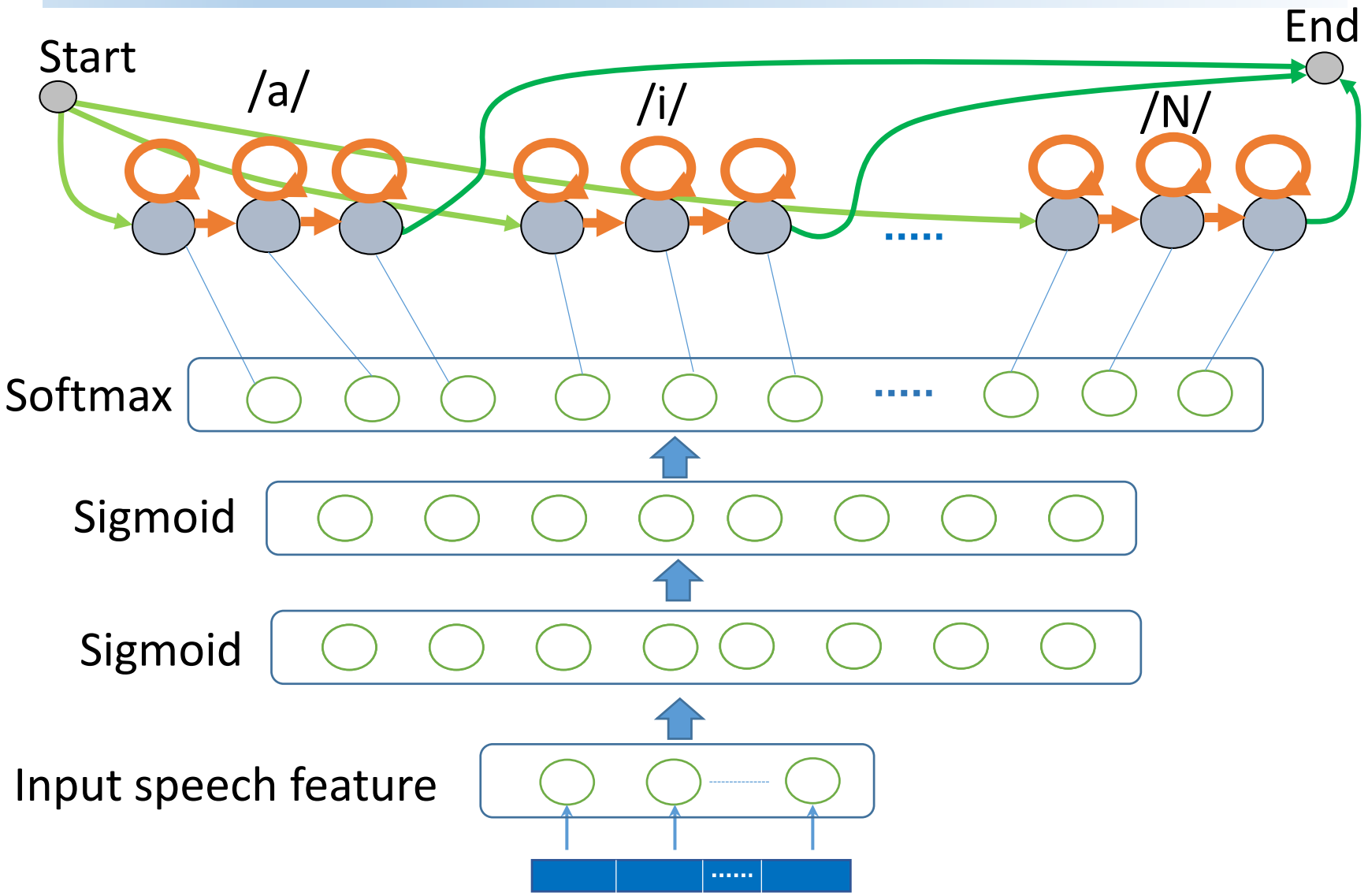


GMM-HMM



MLP-HMM

MLP-HMM based Phone Recognizer



Neural network based language model

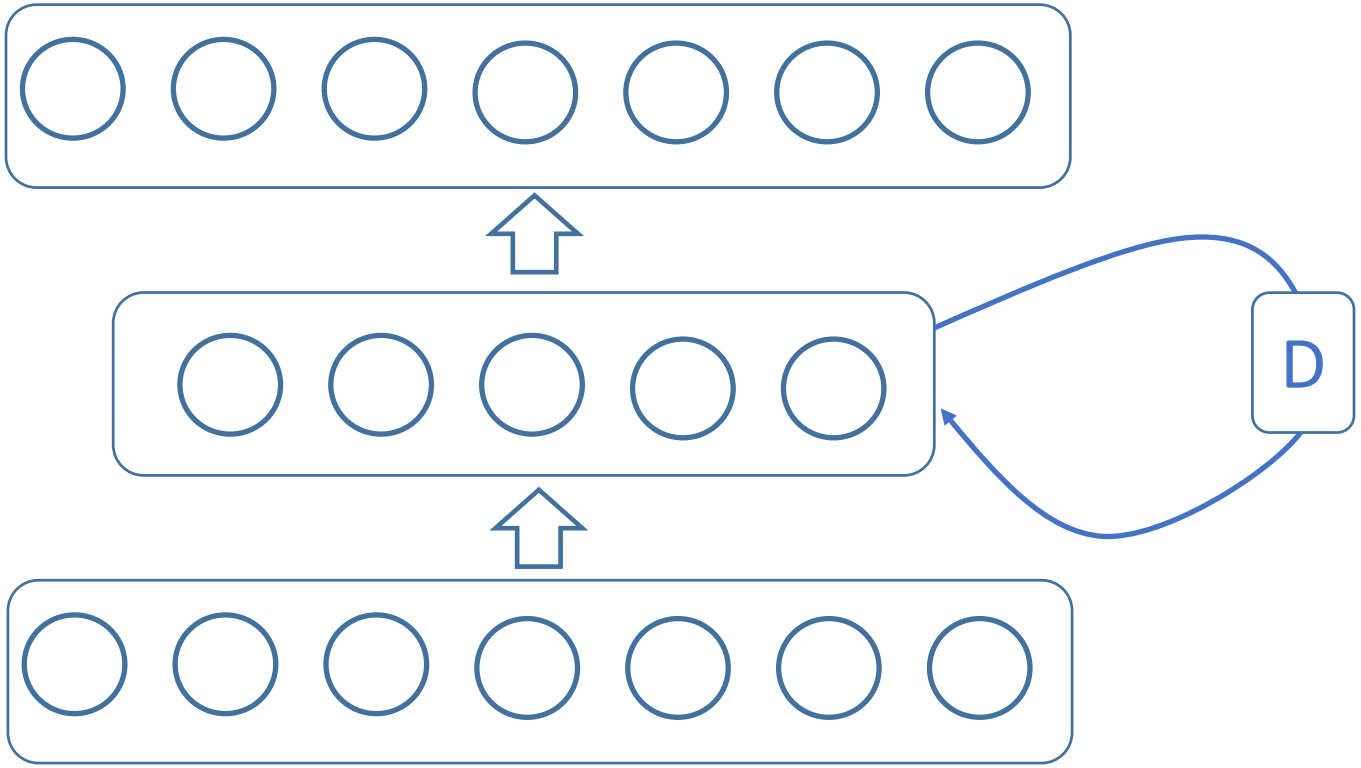
Word Vector

- One-of-K representation of a word for a fixed vocabulary

word	ID	1-of-K
Apple	1	<1,0,0,0,0,0,0>
Banana	2	<0,1,0,0,0,0,0>
Cherry	3	<0,0,1,0,0,0,0>
Durian	4	<0,0,0,1,0,0,0>
Orange	5	<0,0,0,0,1,0,0>
Pineapple	6	<0,0,0,0,0,1,0>
Strawberry	7	<0,0,0,0,0,0,1>

Word Prediction Using RNN

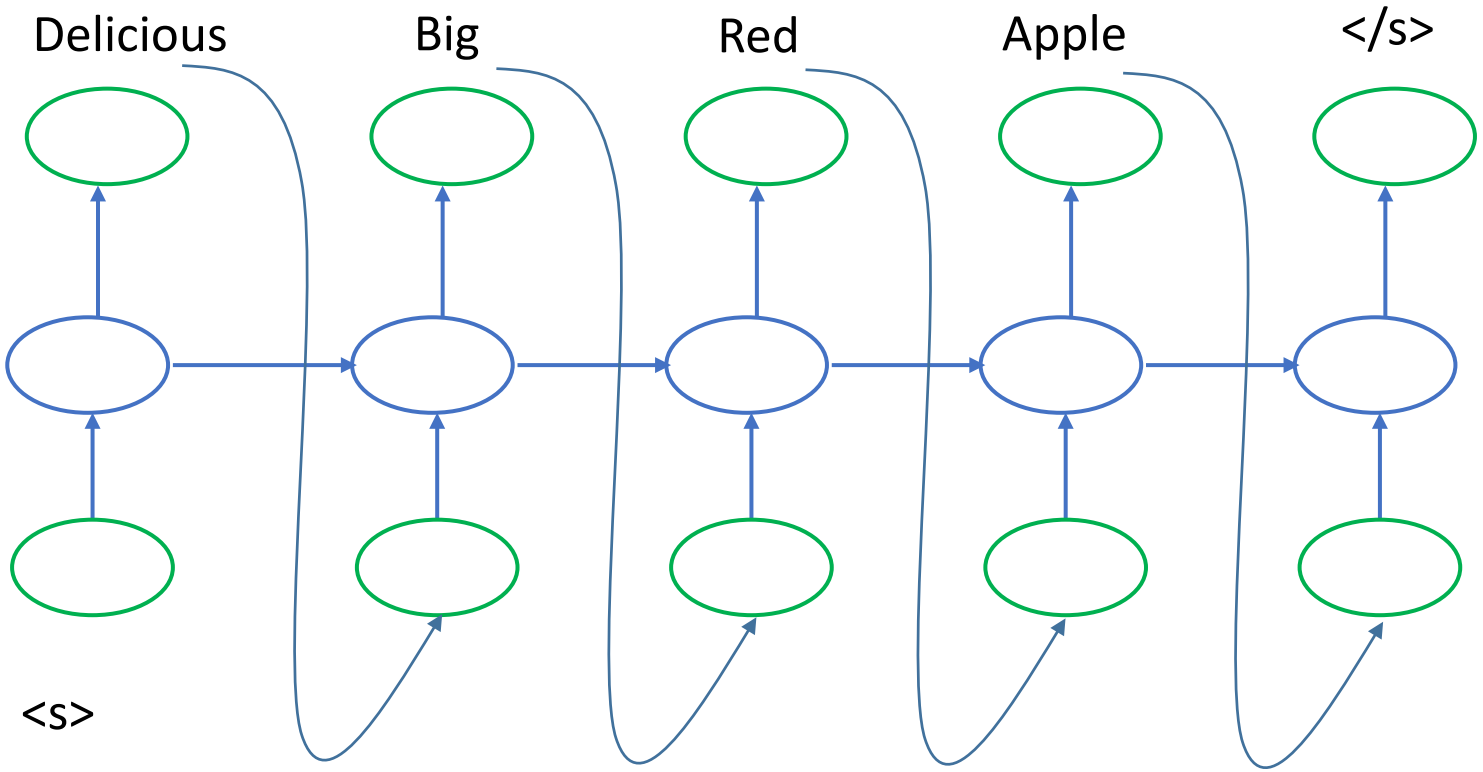
(Distribution of)
$Word_t$ → <0.02 ,0.65, 0.14 ,0.11, 0.05, 0.01 ,0.02>



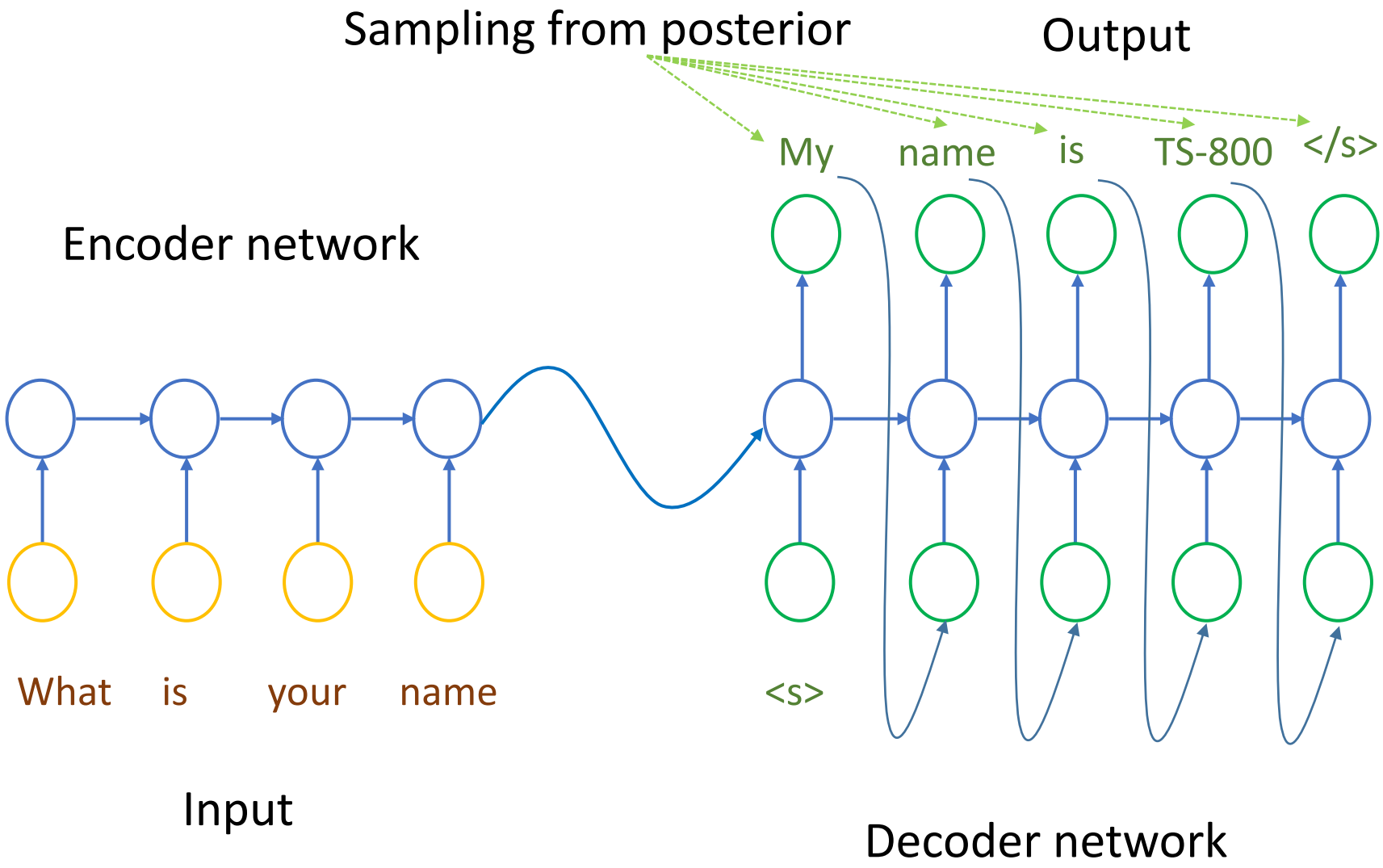
$Word_{t-1}$ → <0, 0, 0, 1, 0, 0, 0>

RNN Language Model (Unfolded)

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



Dialogue System Using Seq2Seq Network



Evolution of Compute Hardware

2002
Earth simulator
40.96TFLOPS

Picture is from wikipedia

2017
GeForce GTX 1080Ti
10.609TFLOPS
699USD

Picture is from Nvidia.com