The Introduction To Artificial Intelligence

Yuni Zeng yunizeng@zstu.edu.cn

2022-2023-1

The Introduction to Artificial Intelligence

- Part I Brief Introduction to AI & Different AI tribes
- Part II Knowledge Representation & Reasoning
- Part III AI GAMES and Searching
- Part IV Model Evaluation and Selection
- Part V Machine Learning
- Part VI Neural Networks

Homework

题目：AI相关的任务及其神经网络方法

作业	
小组形式	1-5人一组
提交格式	PPT，10页及以上
讲解时长	5-7分钟
截止时间	2023.12.05 13:30

- 逻辑清晰，讲解清楚
- 仅仅是一次作业，期末考试在最后一周！

Neural Networks

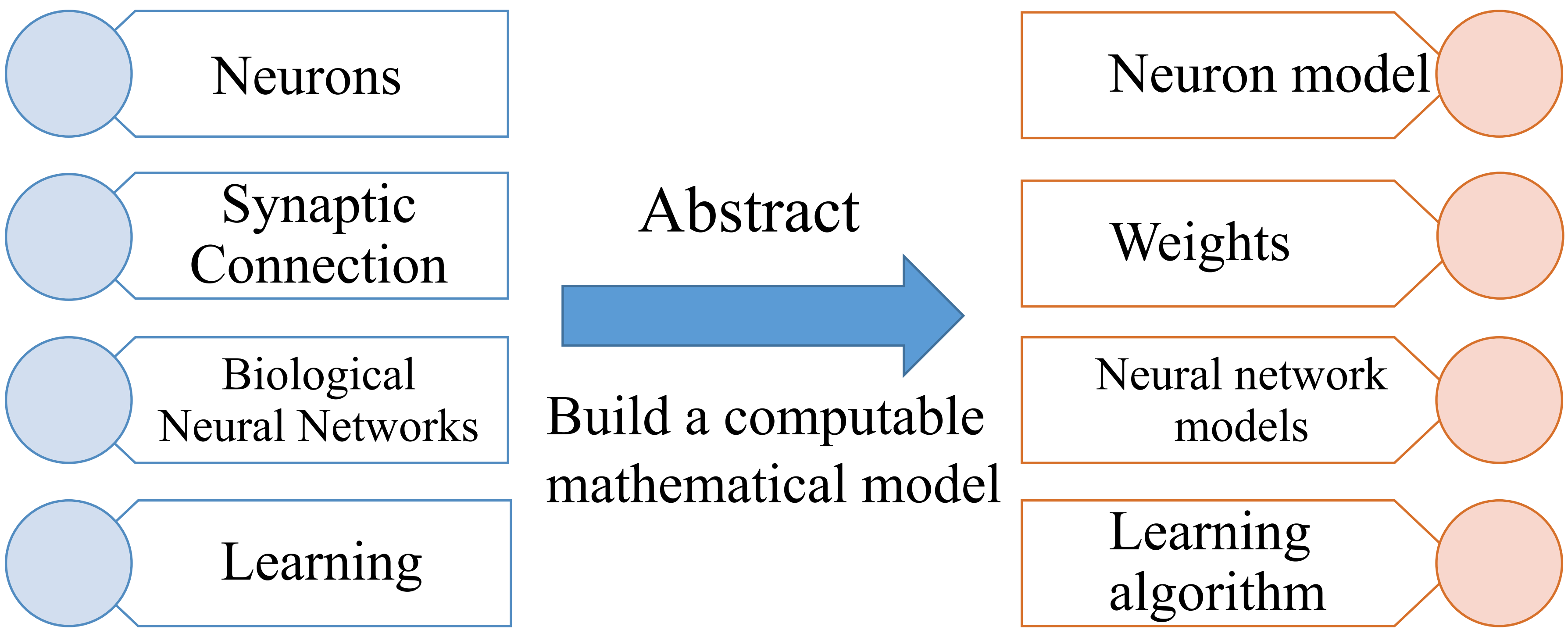
- *Brief review*
- Feedforward Neural Networks
- Recurrent Neural Networks
- The Learning of Neural Networks
- Model Performance: Cost Function
- Steepest Descent Method
- Backpropagation

Brief review

Artificial Neuron

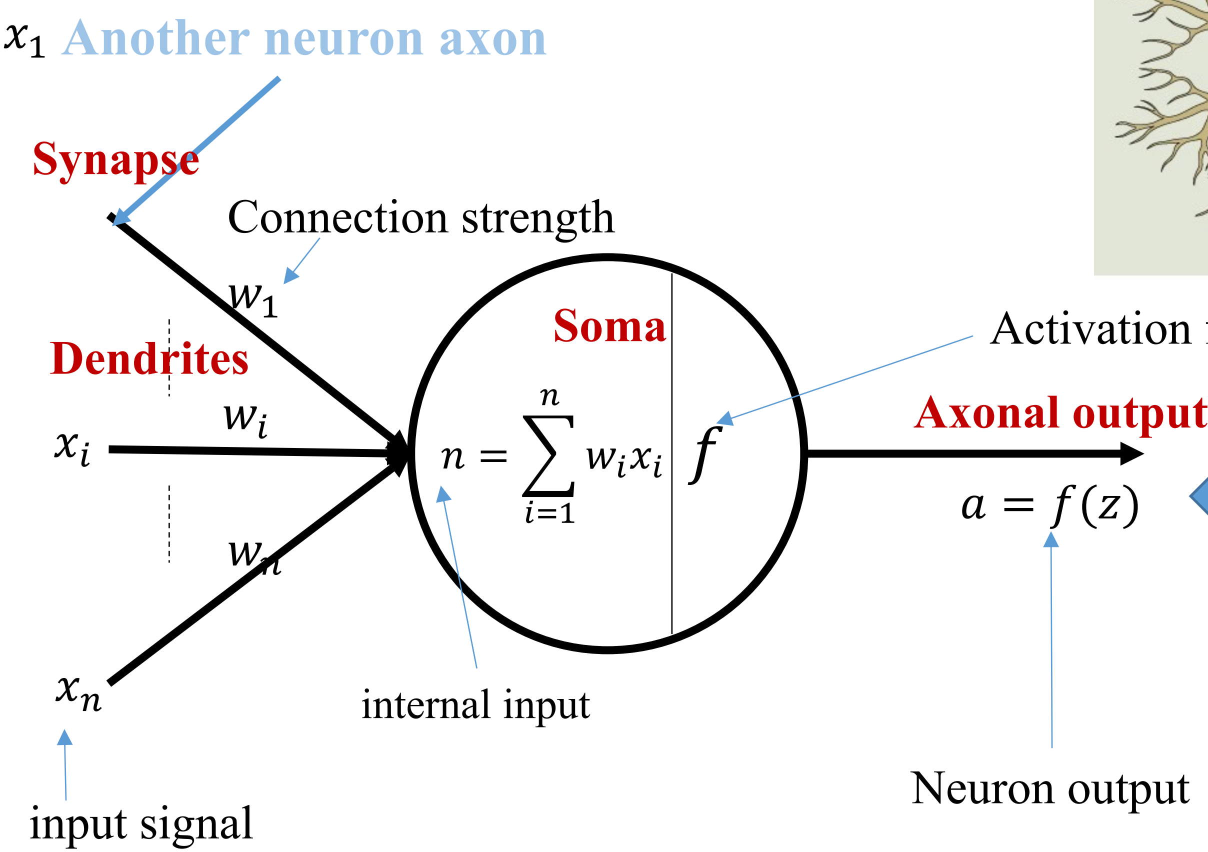
Biological neural network

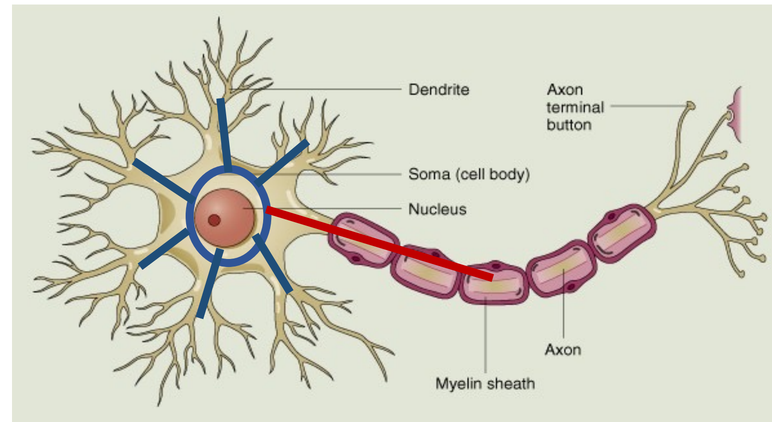
Artificial neural networks



Brief review

Artificial Neuron





$$a = f\left(\sum_{i=1}^{R} w_i x_i\right)$$

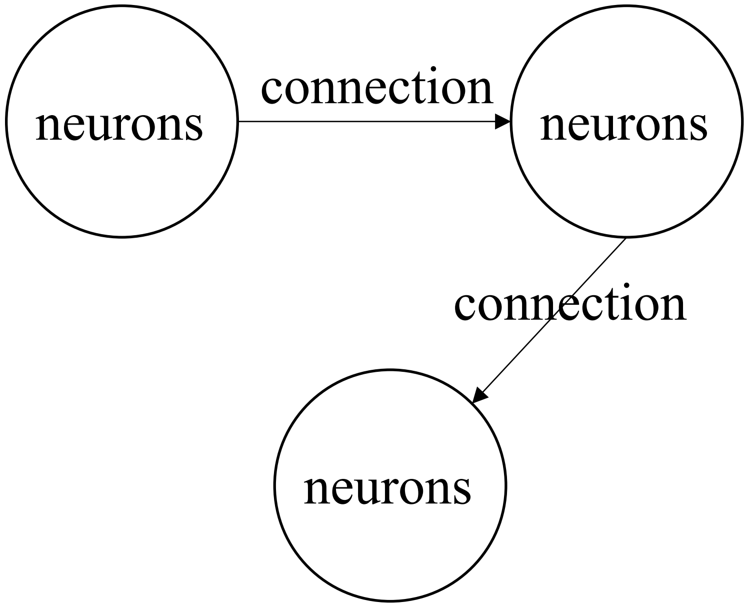
Computational Model of Neural Network

Neural Networks

Feedforward neural network

=

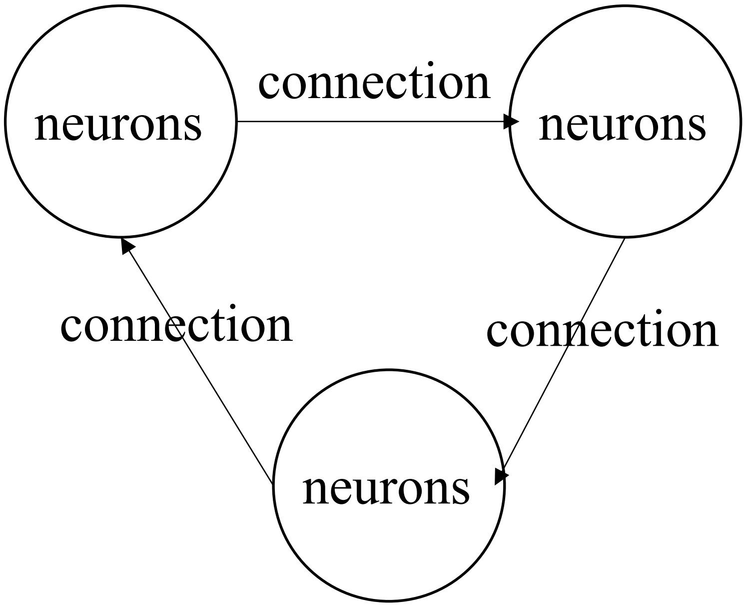
neurons + feedforward connections



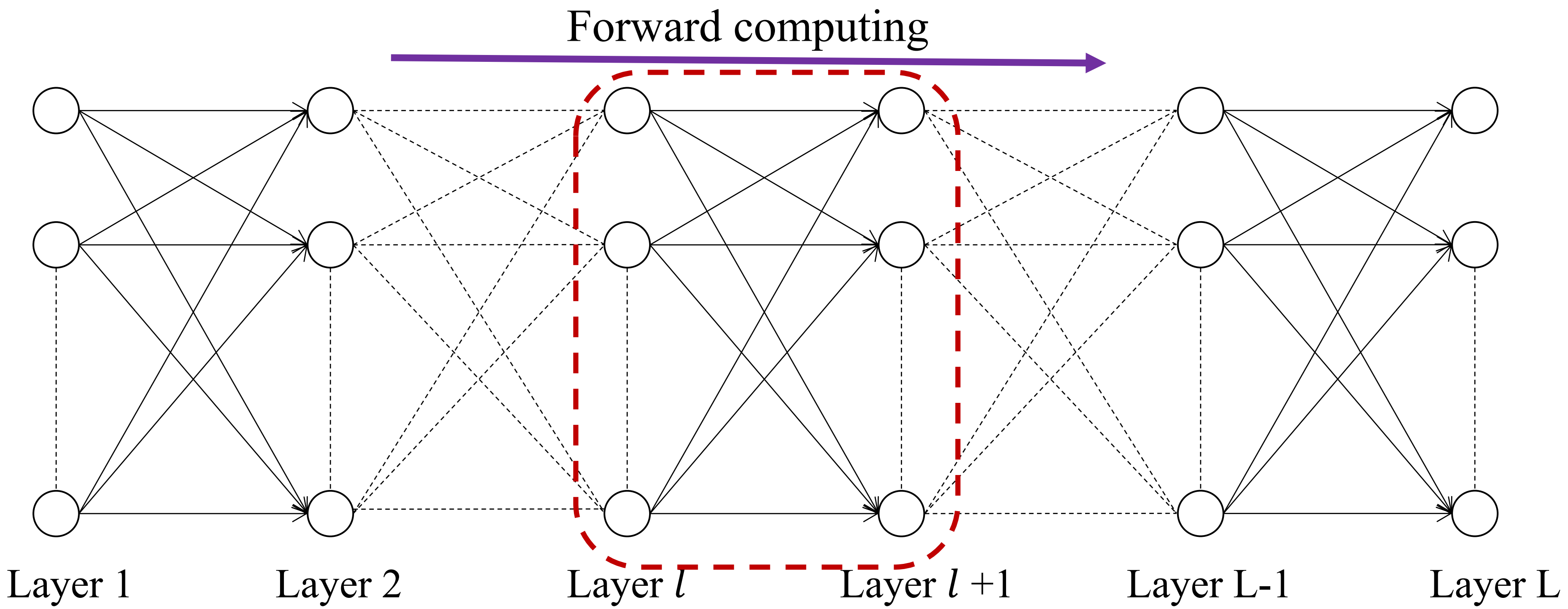
Recurrent neural network

=

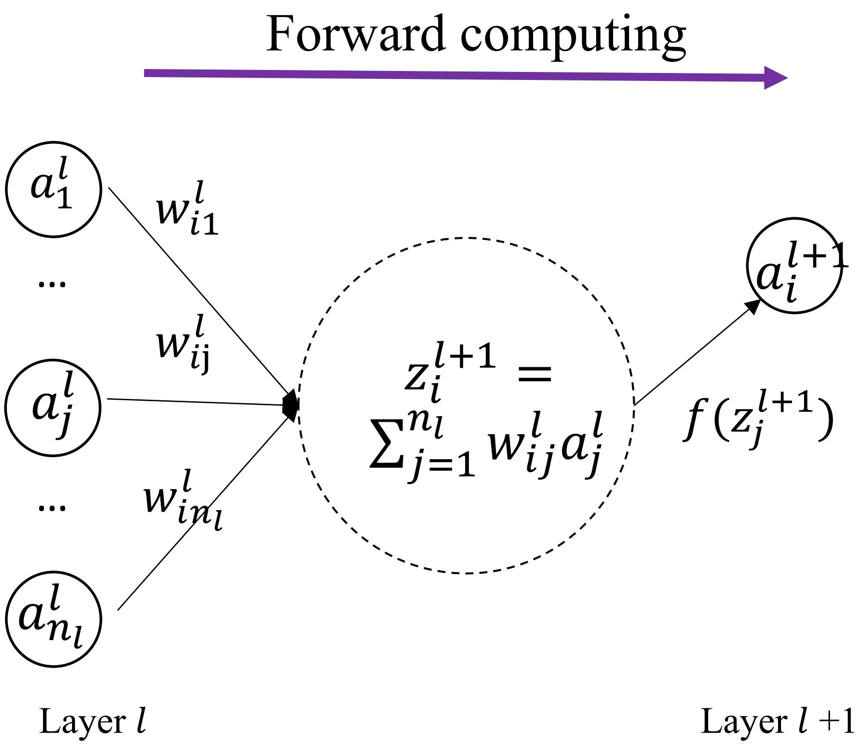
neurons + recurrent connections

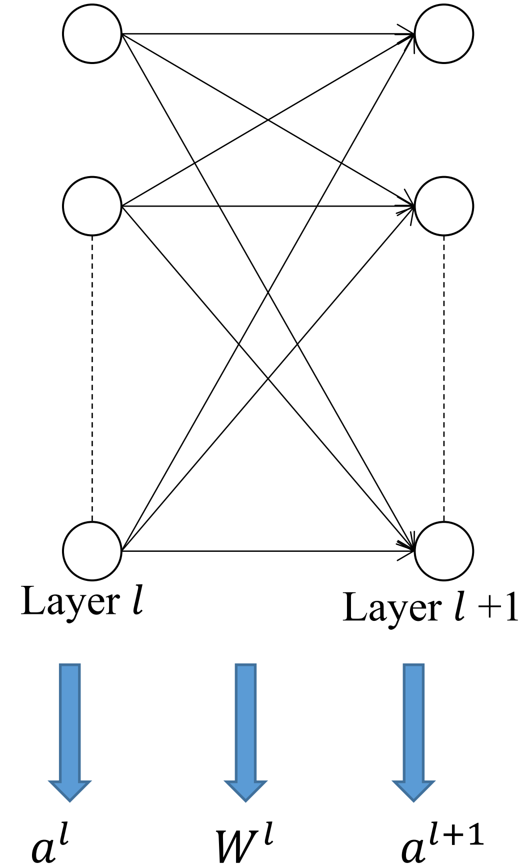


Feedforward Neural Network



Feedforward Neural Network





Component form

$$a_i^{l+1} = f(z_i^{l+1})$$

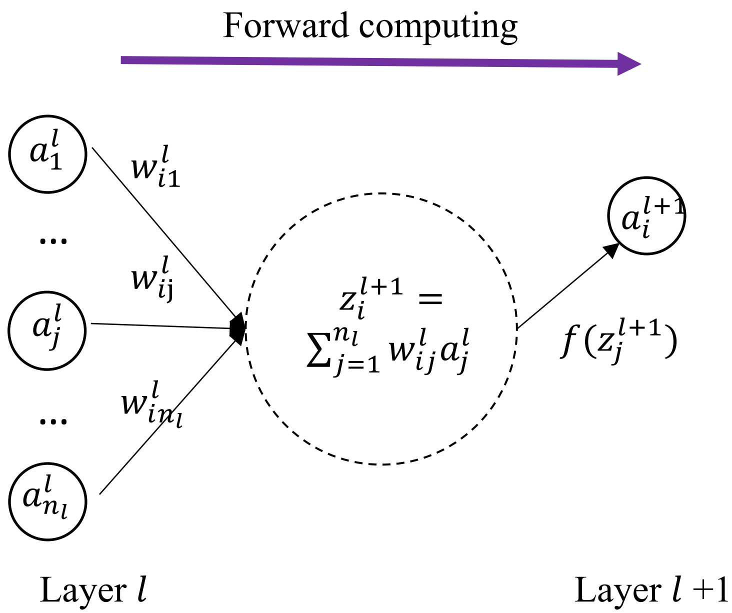
$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l$$

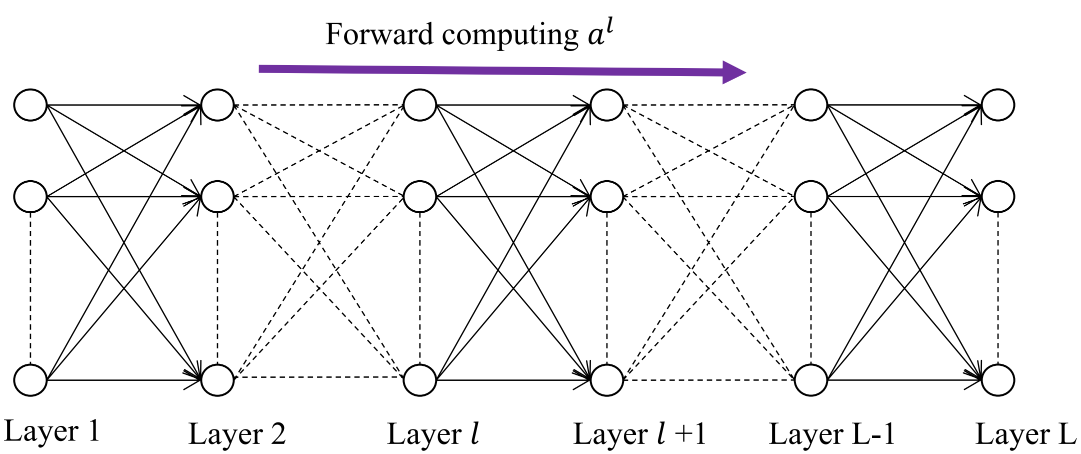
Vector form

$$a^{l+1} = f(z^{l+1})$$

$$z^{l+1} = W^l a^l$$

Feedforward Neural Network





Algorithm:

Input W^l, a^l

for l = 1:L, run function:

$$a^{l+1} = fc(W^l, a^l)$$

return

Function $fc(W^l a^l)$

For i = 1: n_{l+1}

$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l$$

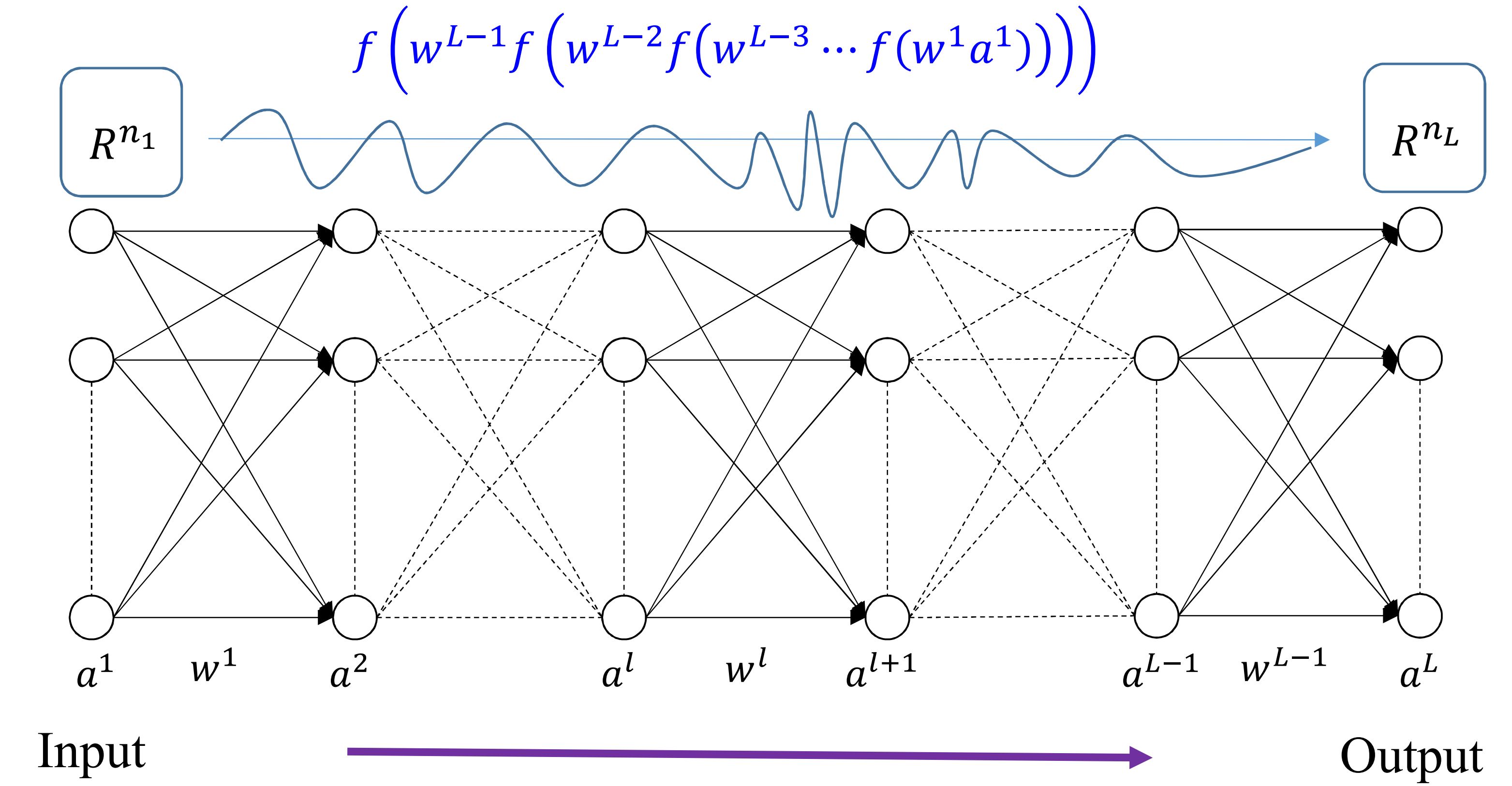
$$a_i^{l+1} = f(z_i^{l+1})$$

end

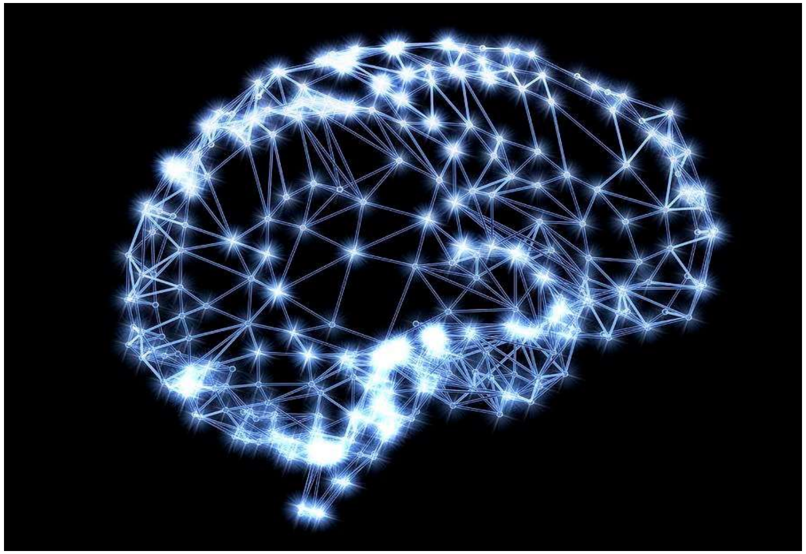
Feedforward Neural Network

In fact, FNN is a nonlinear mapping from R^{n_1} space to R^{n_L} space.

$$a^L = f(w^{L-1}a^{L-1}) = f\left(w^{L-1}f\left(w^{L-2}f\left(w^{L-3}\cdots f(w^1a^1)\right)\right)\right)$$



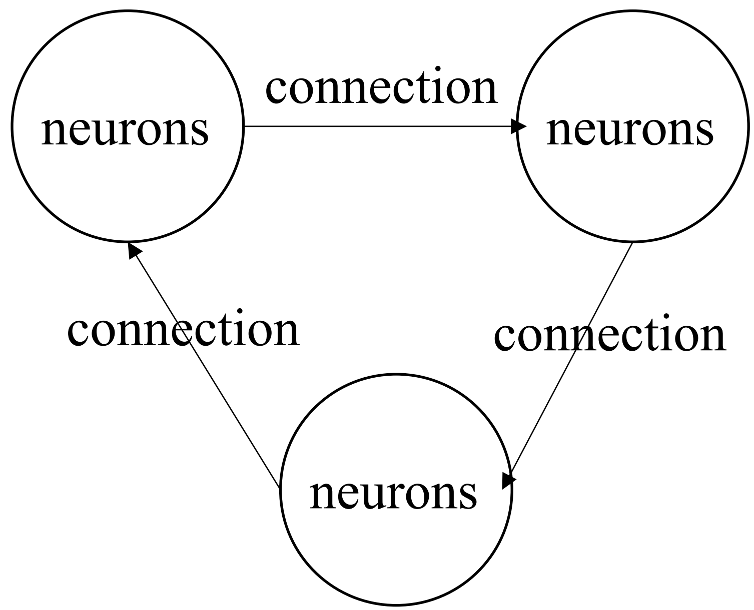
Recurrent Neural Networks



Recurrent neural network

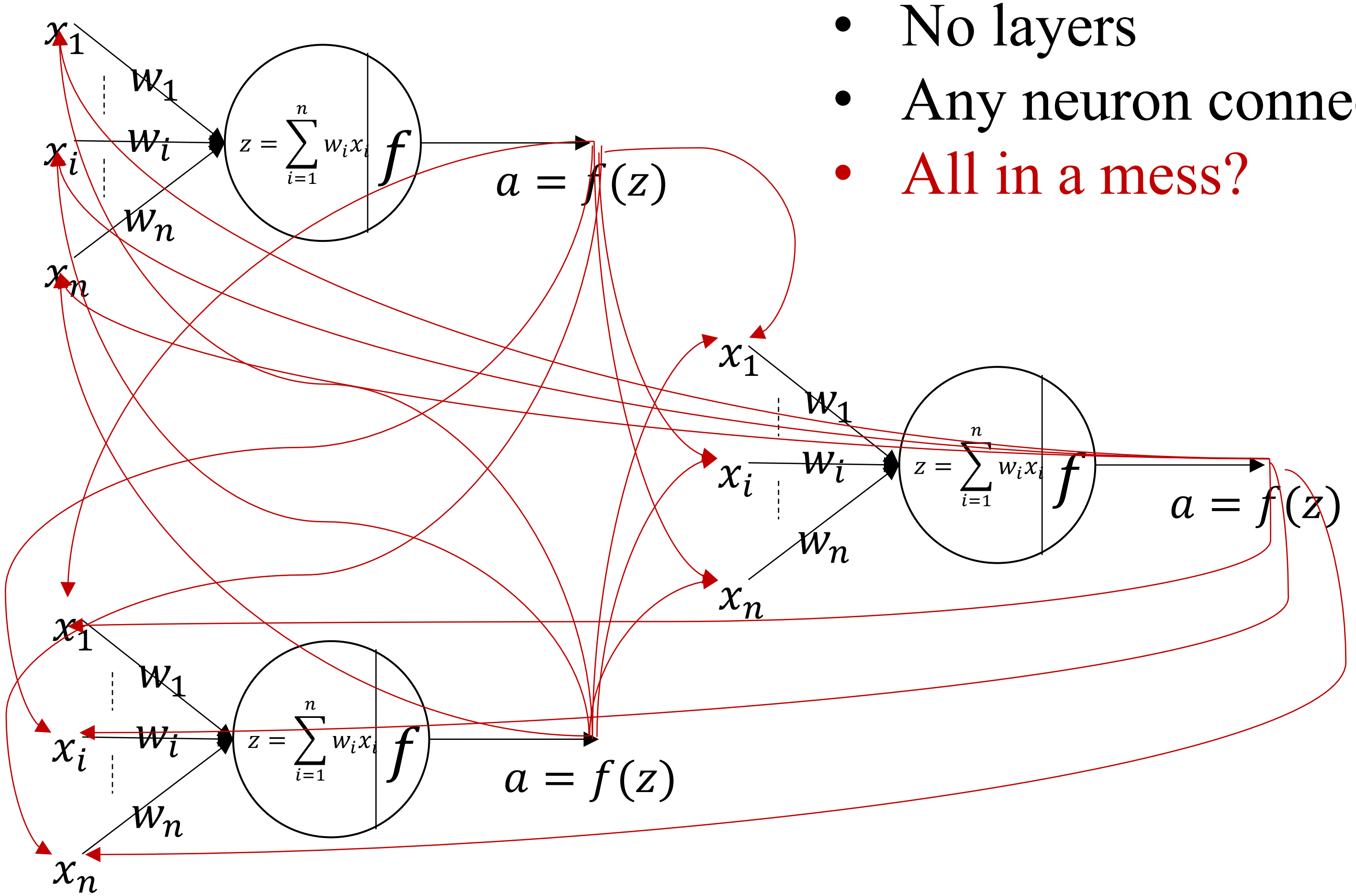
=

neurons + recurrent connections



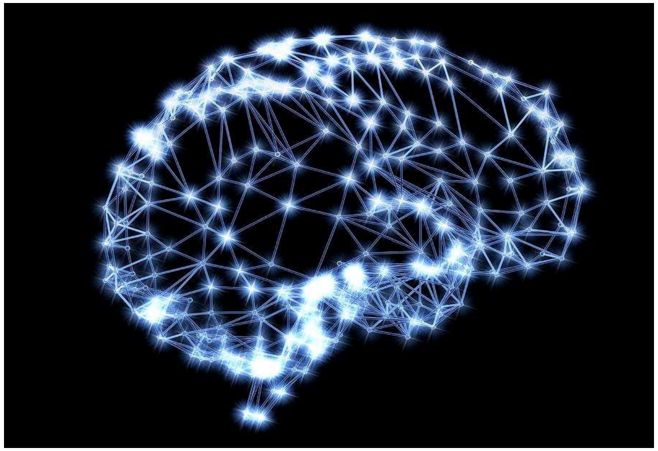
RNNs ---- with feedback connections

Recurrent Neural Networks

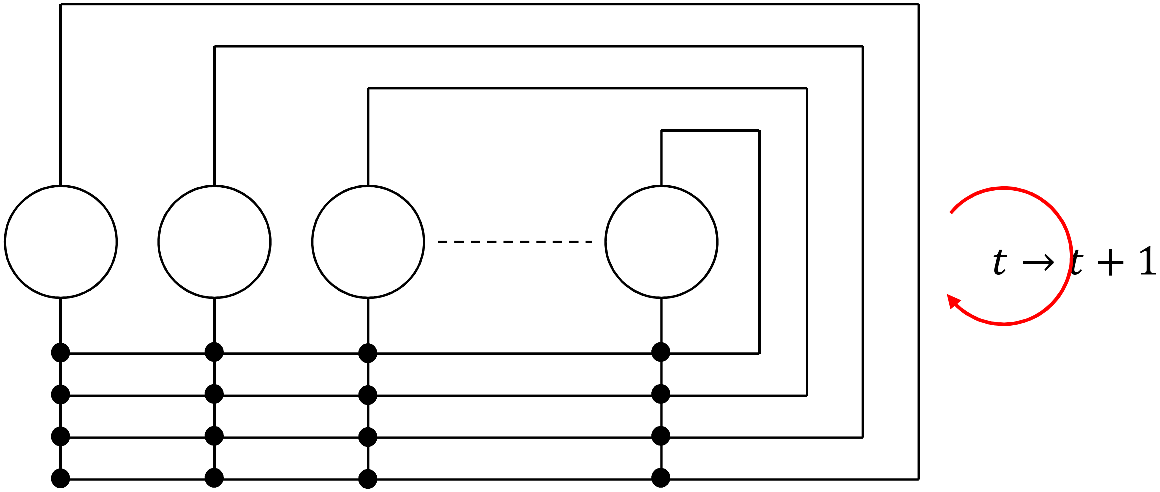


- No layers
- Any neuron connects to any others
- All in a mess?

Recurrent Neural Networks

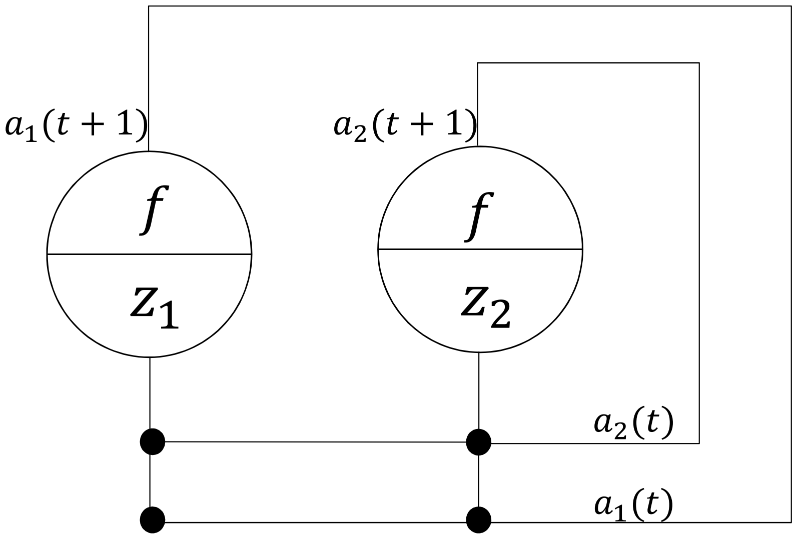


Topology Structure



Problem: how to develop computational model of the RNNs ?

Recurrent Neural Networks



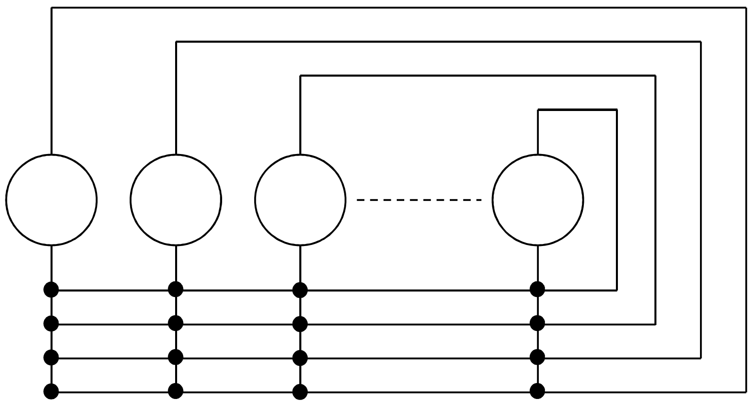
RNNs – Computational Neural Networks Model:

$$\begin{cases} a_1(t+1) = f\left(w_{11}a_1(t) + w_{12}a_2(t)\right) \\ a_2(t+1) = f(w_{21}a_1(t) + w_{22}a_2(t)) \end{cases}$$

Recurrent Neural Networks

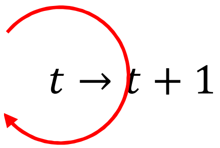
Computational Model of RNNs:

$$a_i(t+1) = f\left(\sum_{j=1}^{n} w_{ij} a_j(t)\right)$$



Vector form:

$$a(t+1) = f(Wa(t))$$

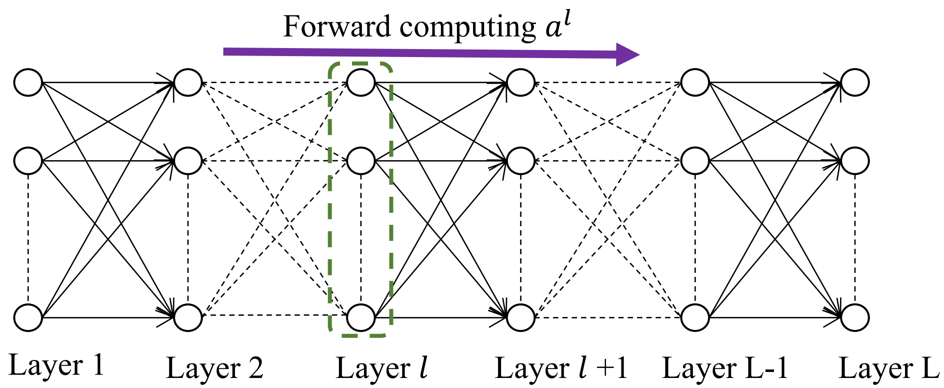


$$W = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nn} \end{bmatrix}, a(t) = \begin{bmatrix} a_1(t) \\ \vdots \\ a_n(t) \end{bmatrix}$$

The time changes in discrete manner.

This model is a discrete time dynamic system.

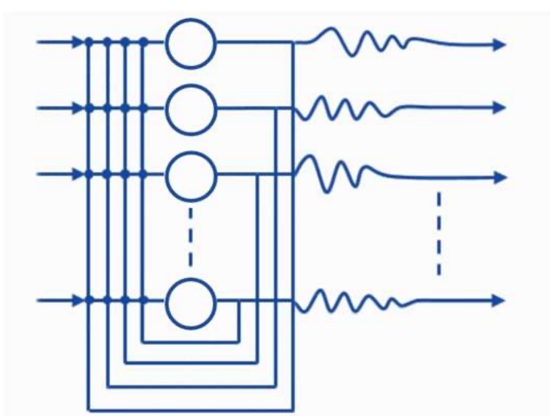
FNNs VS. RNNs



no recurrent connection

FNNs

- Extract the spatial features of static data
- Describe spatial correlation



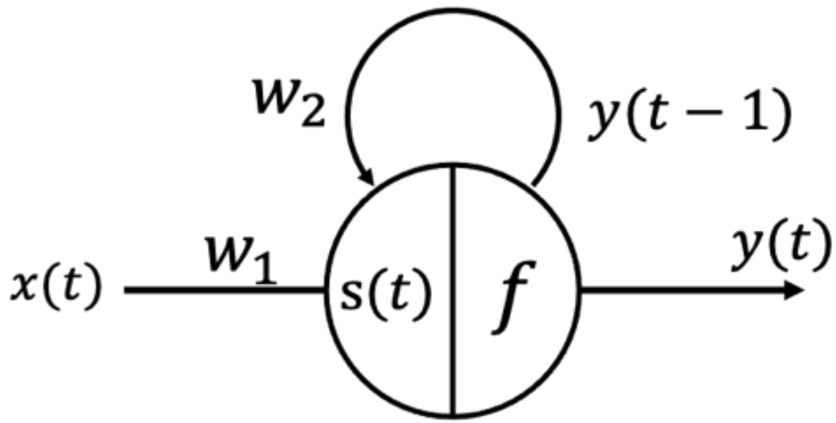
with recurrent connection

RNNs

- Memory mechanism
- Extract spatiotemporal features of time sequence data
- Describe time correlation

Exercise

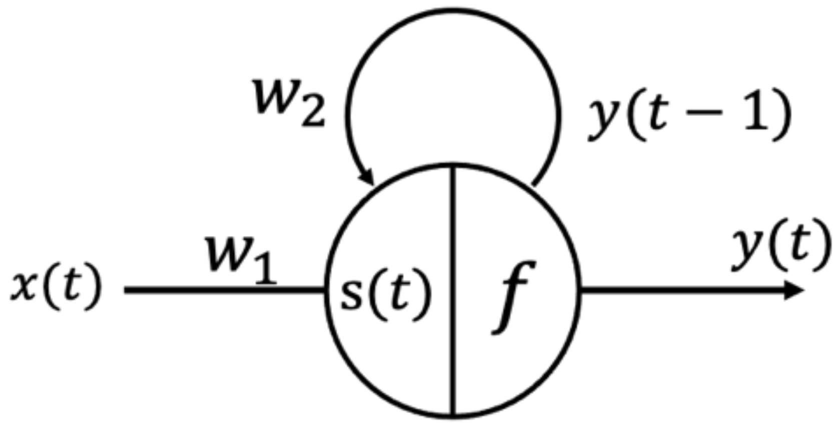
1. 给定以下回复式神经网络结构，其中包含两个权重 w_1和w_2，和激活函数$f(s) = s$：



(1) 确定该网络的前向计算。

Exercise

1. 给定以下回复式神经网络结构，其中包含两个权重 w_1和w_2，和激活函数$f(s) = s$：

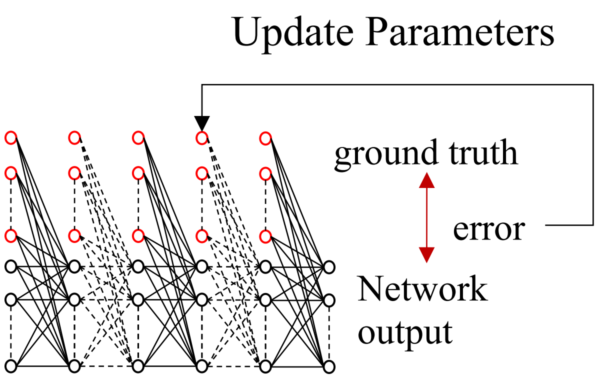


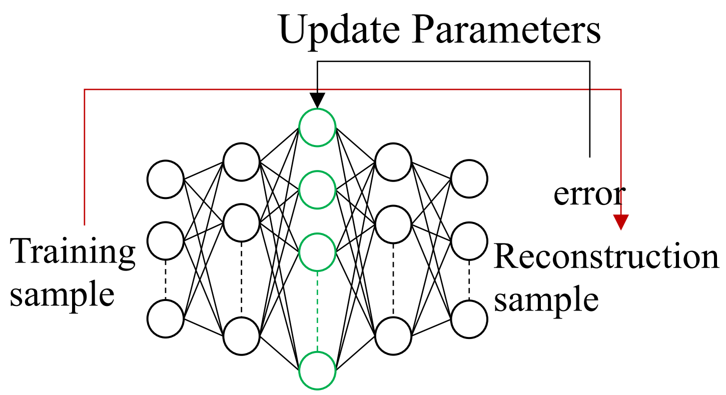
(1) 确定该网络的前向计算。

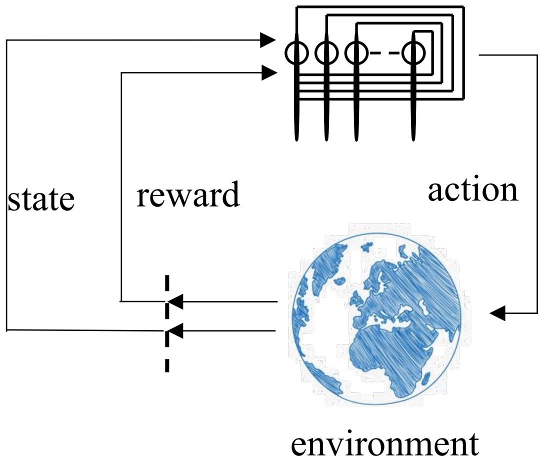
$$y(t) = f(s(t)),$$
$$s(t) = x(t)w_1 + y(t-1)w_2$$

The Learning of Neural Networks

- Learning is to change the connections by some rules.
- Similar with the three learning model of human:







Supervised Learning: Update the network parameters according to the error between the target output and the actual network output of the training sample

Unsupervised learning: For non-label samples, the network parameters are updated by reconstructing these samples.

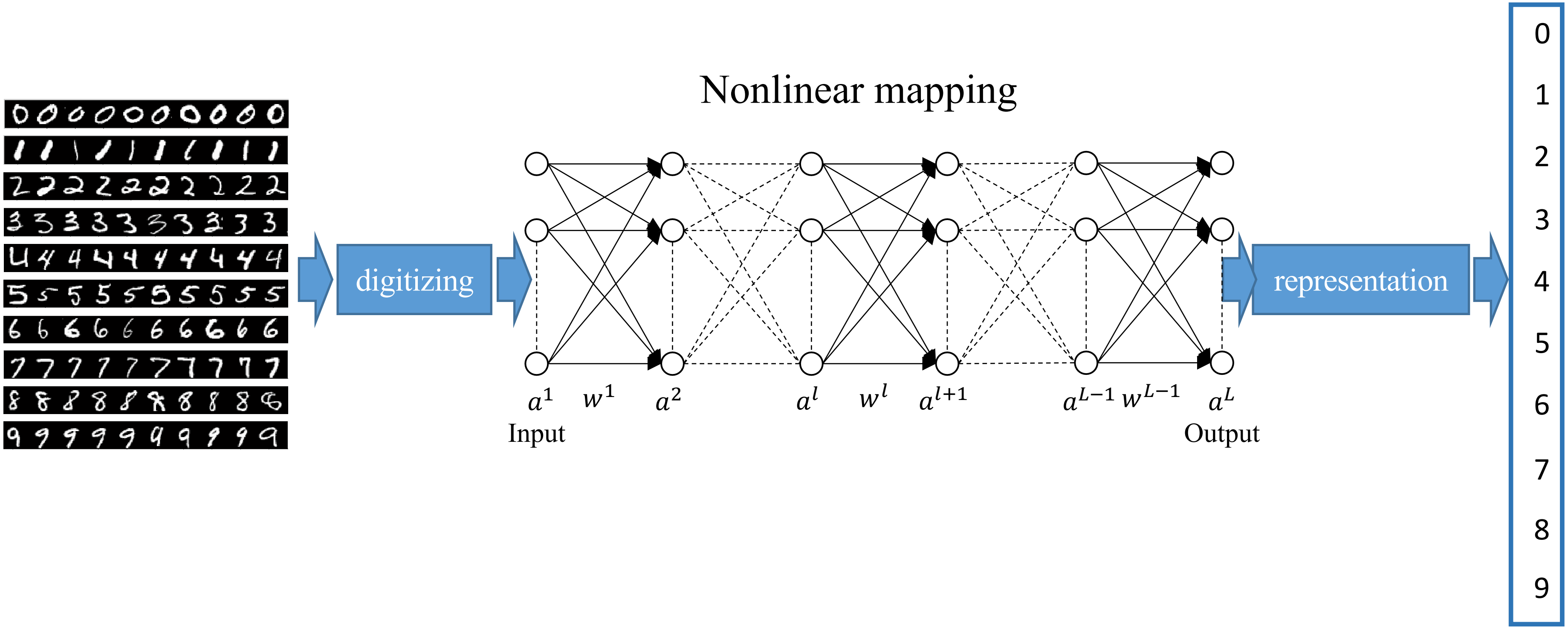
Reinforcement learning: Update network parameters with the goal of maximizing rewards during interactions with the environment

Neural Networks

- Brief review
- Feedforward Neural Networks
- Recurrent Neural Networks
- The Learning of Neural Networks
- *Model Performance: Cost Function*
- Steepest Descent Method
- Backpropagation

Model Performance: Cost Function

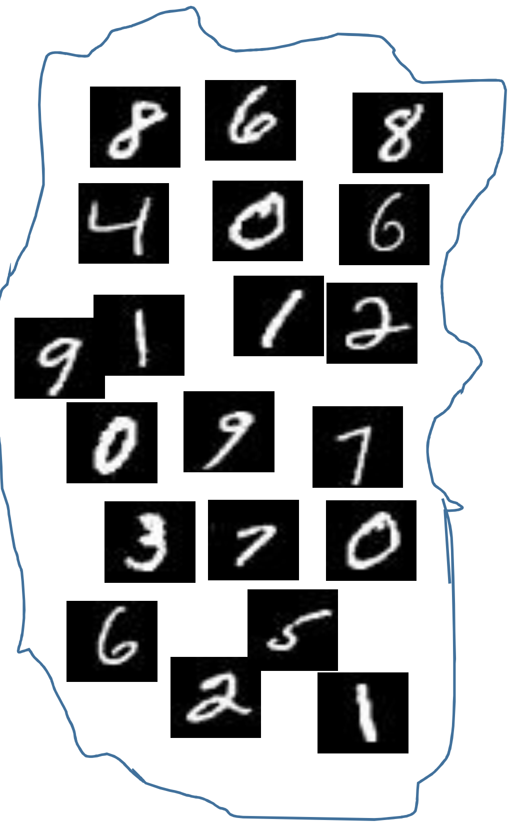
Nonlinear Mapping



Problem: How to design the NN? Are there any methods to find "good" connection weights?

Model Performance: Cost Function

Cost Function



Good performance!
The mother knows correct answer.

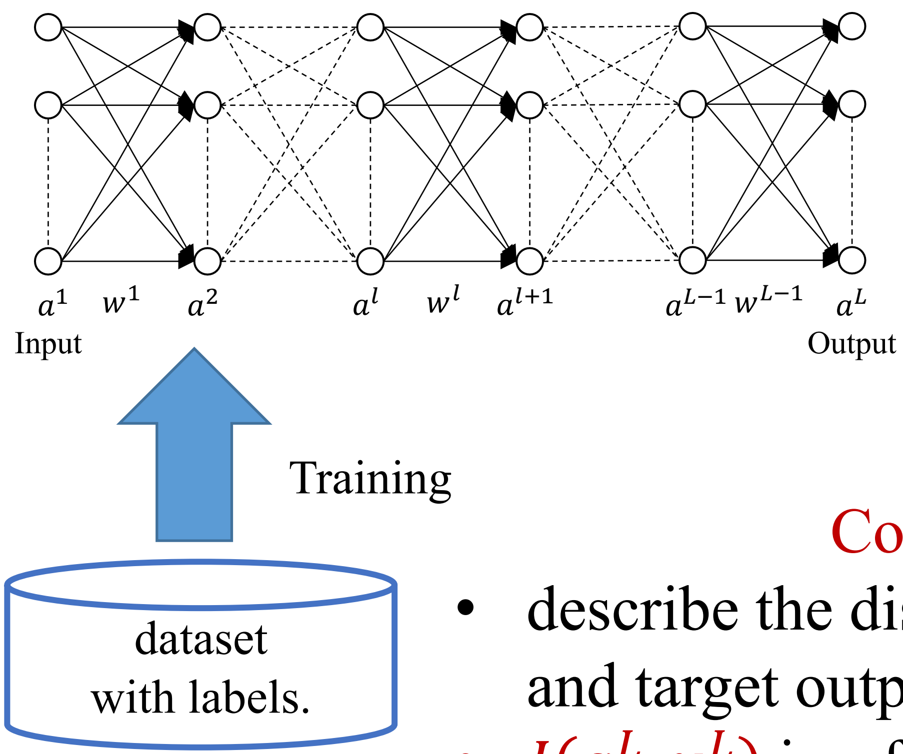
5

Two important factors:

1. There must be a measure to measure the correctness between correct answer and the girl's real output. ----- Performance function (性能函数) .
2. There must be a mechanism to change the knowledge system of the girl. ---- Learning algorithm (学习算法) .

Model Performance: Cost Function

Cost Function



The goal of Learning:
Network output ≈ Target output

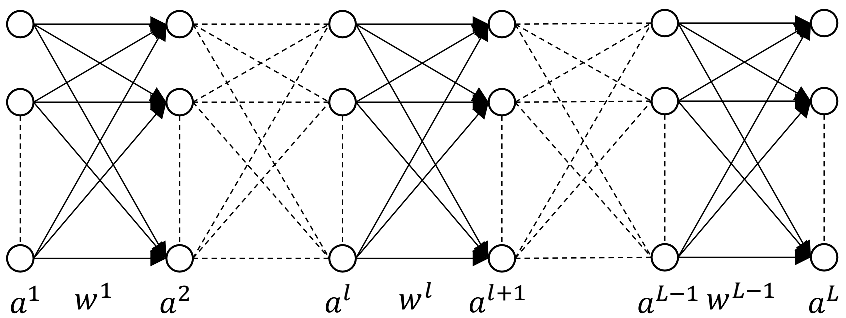
Cost Function $J(a^L, y^L)$:

- describe the distance between network output a^L and target output y^L
- $J(a^L, y^L)$ is a function related to $(w^1, \cdots, w^{L-1})$

$$J = J(w^1, \cdots, w^{L-1})$$

Model Performance: Cost Function

Cost Function



Target Output	Network Output
$y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix}$	$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$

- The cost function describes the performance of the network. The smaller J is, the closer the network output is to the target output, and the better the network performance is.
- $J(a^L, y^L)$ is a function related to $(w^1, \cdots, w^{L-1})$, to get a good performance is to find a good $(w^1, \cdots, w^{L-1})$.
- To find the good $(w^1, \cdots, w^{L-1})$ is the learning of neural network.

Model Performance: Cost Function

Cost Function

Target Output	Network Output
$y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix}$	$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$

Learning is a process such that a^L is close to y^L, i.e., the cost function J reaches minimum.

A cost function $J = J(w^1, \cdots, w^{L-1})$ is a function with variables $w^l (l = 1, \cdots, L-1)$, thus the network learning is to looking for some $w^l (l = 1, \cdots, L-1)$ such that $w^l (l = 1, \cdots, L-1)$ is a minimum point of J.

A frequently used cost function:

$$J = \frac{1}{2} \sum_{j=1}^{n_L} e_j^2 = J(w^1, \cdots, w^{L-1})$$

J is a function of $w^1, \cdots, w^{L-1}$.

Problem: How to find out the minimum points of J ?

Learning = Looking for minimum points of J

Neural Networks

- Brief review
- Feedforward Neural Networks
- Recurrent Neural Networks
- The Learning of Neural Networks
- Model Performance: Cost Function
- *Steepest Descent Method*
- Backpropagation

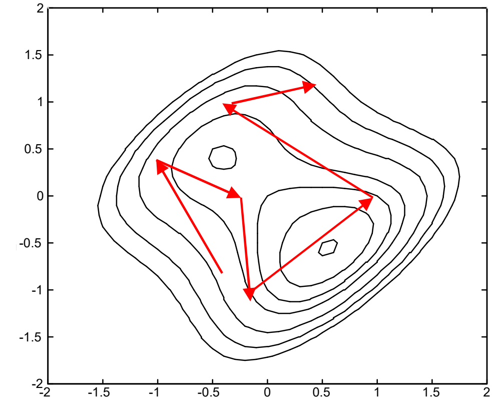
Steepest Descent Method

Minimum Points

General Nonlinear function

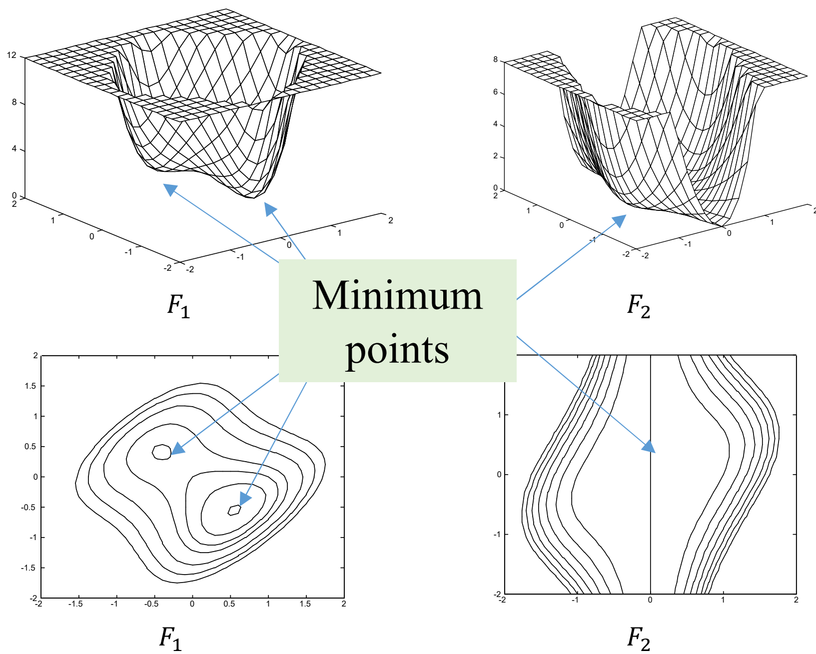
$$F(x), x \in R^n$$

x^* is a minimum point if $F(x^*) \le F(x)$ for any x that very close to x^*.



$$F_1(w) = (w_2 - w_1)^4 + 8w_1w_2 - w_1 + w_2 + 3$$

$$F_2(w) = (w_1^2 - 1.5w_1w_2 + 2w_2^2)w_1^2$$



Iteration Methods:

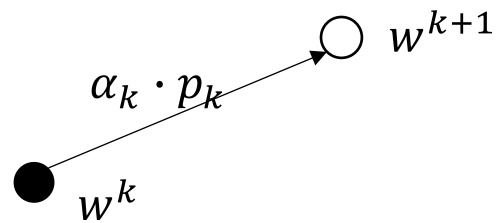
1. Setting a starting point x_0

2.Finding a minimum point step by step:

$$w^{k+1} = w^k + \alpha_k \cdot p_k,$$

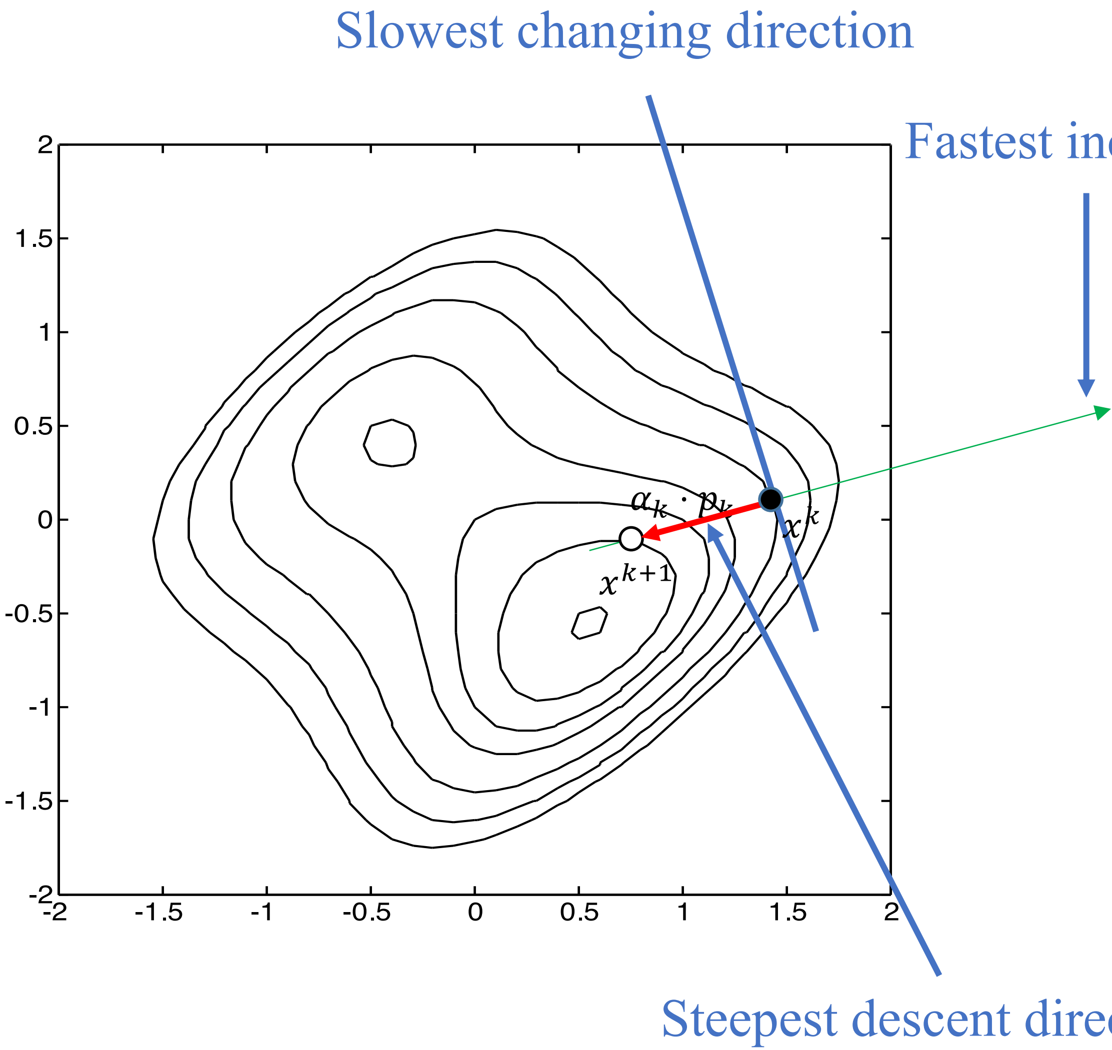
p_k: is called searching direction

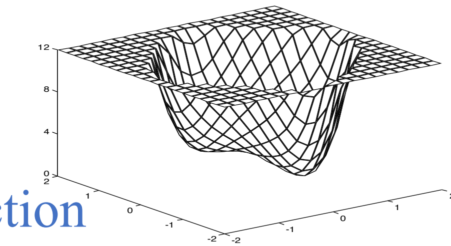
α_k: is leaning rate at step k



Problem: how to get the searching direction p_k.

Steepest Descent Method





Gradient:

$$g_k = \nabla F(w)\Big|_{w^k} = \frac{\partial F}{\partial w}\Big|_{w^k} = \begin{pmatrix} \frac{\partial F}{\partial w_1} \\ \vdots \\ \frac{\partial F}{\partial w_n} \end{pmatrix}\Bigg|_{w^k}$$

Steepest Descent Algorithm:

$$p_k = -g_k$$
$$w^{k+1} = w^k - \alpha_k \cdot g_k$$

or

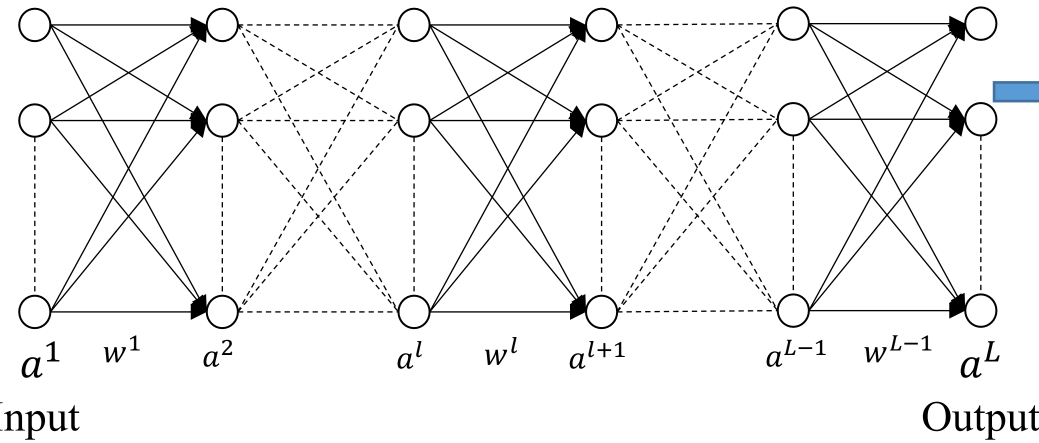
$$w^{k+1} = w^k - \alpha_k \cdot \frac{\partial F}{\partial w}\Big|_{w^k}$$

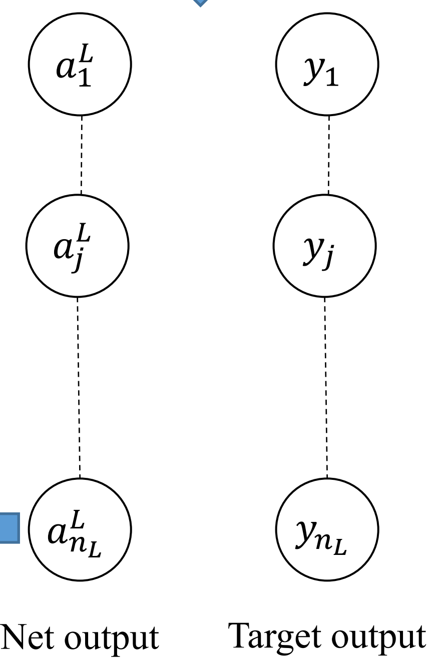
Steepest Descent Method

Deep learning

Steepest Descent Algorithm:

$$w^{k+1} = w^k - \alpha_k \cdot \left.\frac{\partial F}{\partial w}\right|_{w^k}$$





Construct cost function

$$J = \frac{1}{2}\sum_{j=1}^{n_L}\left(y_j - a_j^L\right)^2$$

Computing gradient

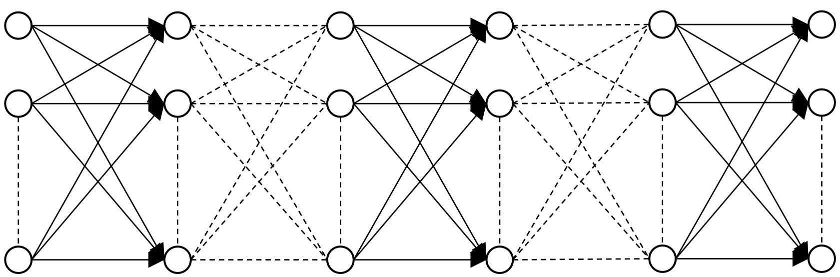
$$\frac{\partial J}{\partial w_{ji}^l}$$

Updating weights

$$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$$

Steepest Descent Method

Deep learning



Target Output	Network Output
$y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix}$	$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$

$$a^L = f(W^{L-1}a^{L-1}) = f\left(W^{L-1}f\left(W^{L-2}f\left(W^{L-3}\cdots f(W^1a^1)\right)\right)\right)$$

Steepest Descent Method

$$J = \frac{1}{2}\sum_{j=1}^{n_L} e_j^2 = \frac{1}{2}\sum_{j=1}^{n_L}\left(a_j^L - y_j^L\right)^2$$

1. Computing

$$\frac{\partial J}{\partial w_{ji}^l}$$

2. Iterating

$$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$$

Problem: How to compute $\frac{\partial J}{\partial w_{ji}^l}$?

Answer:
Using the well-known BP method.

Neural Networks

- Brief review
- Feedforward Neural Networks
- Recurrent Neural Networks
- The Learning of Neural Networks
- Model Performance: Cost Function
- Steepest Descent Method
- *Backpropagation*

Backpropagation

- Updating weights: compute $\frac{\partial J}{\partial w_{ji}^l}$

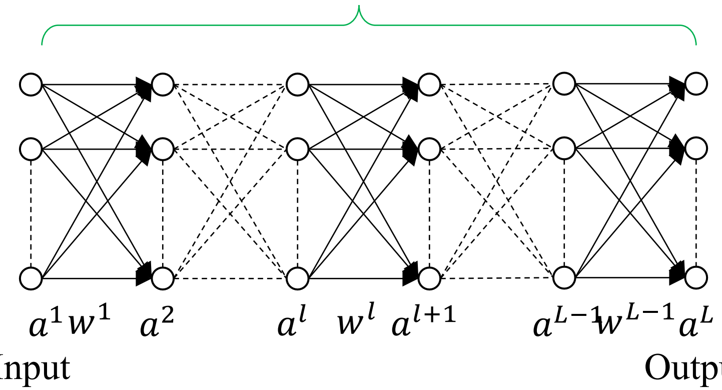
Problem: What's the relation between δ_i^l and $\frac{\partial J}{\partial w_{ji}^l}$?

l layer

$$a_i^l = f(z_i^l)$$

define $\delta_i^l = \frac{\partial J}{\partial z_i^l}$

$J(W^1, \cdots, W^{L-1})$



l layer

$$a_i^l = f(z_i^l)$$

$$\delta_i^l = \frac{\partial J}{\partial z_i^l}$$

w_{ji}^l

$$z_j^{l+1} = \sum_{i=1}^{n_l} w_{ji}^l a_i^l$$

$l+1$ layer

$$a_j^{l+1} = f(z_j^{l+1})$$

$$\delta_j^{l+1} = \frac{\partial J}{\partial z_j^{l+1}}$$

Relation between δ_i^l and $\frac{\partial J}{\partial w_{ji}^l}$

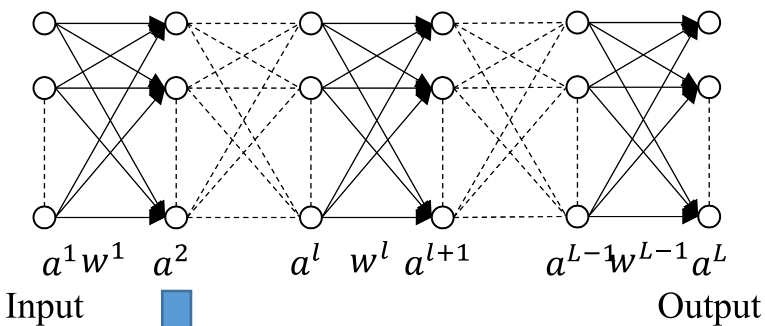
$$\frac{\partial J}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$$

Why?

$$\frac{\partial J}{\partial w_{ji}^l} = \frac{\partial J}{\partial z_j^{l+1}} \cdot \frac{\partial z_j^{l+1}}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$$

Backpropagation

Updating weights



Construct cost function

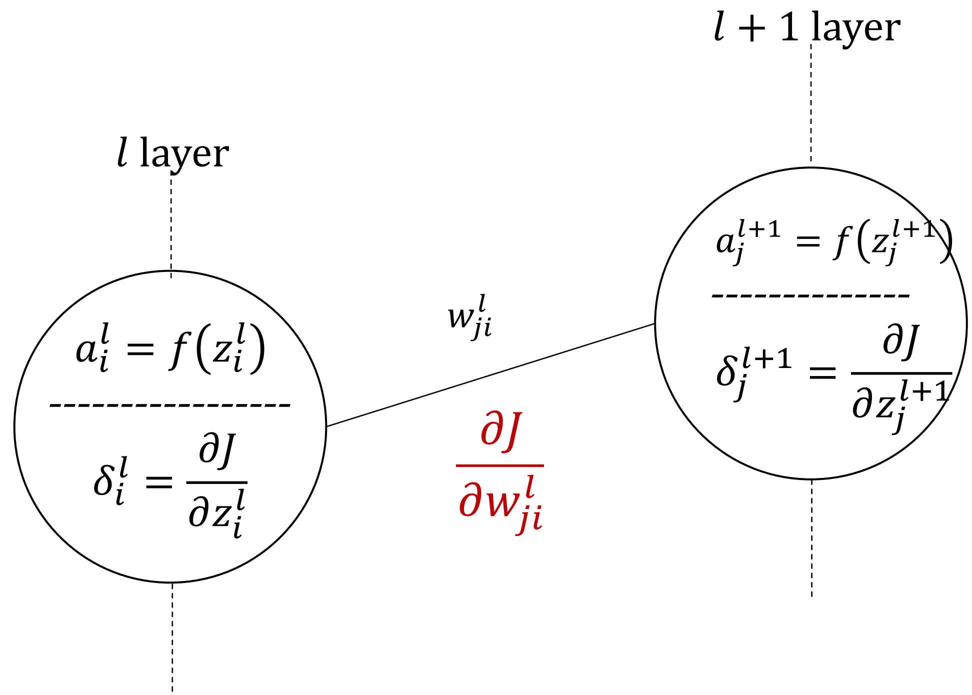
$$J = \frac{1}{2}\sum_{j=1}^{n_L}\left(y_j - a_j^L\right)^2$$

Updating weights

$$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$$

$$\frac{\partial J}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$$

$$\delta_i^{l+1} = \frac{\partial J}{\partial z_i^{l+1}}$$



Problem:

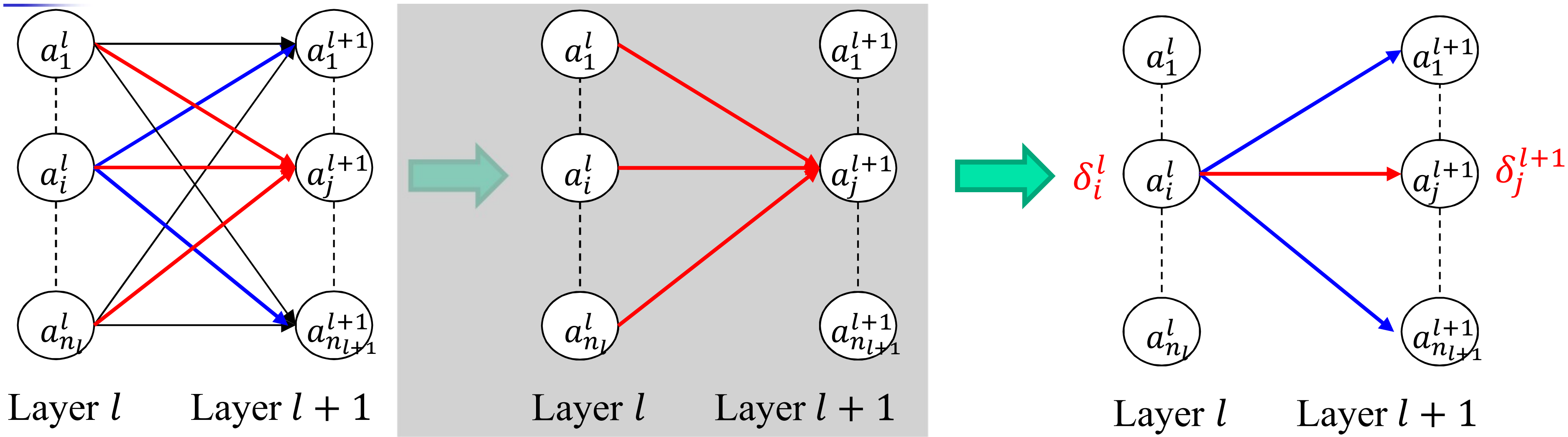
How to compute δ_i^l?

Because, it's easy to compute $\delta_i^L = \frac{\partial J}{\partial z_i^L}$,

What's the relation between δ_i^l and δ_j^{l+1} ?

Backpropagation

Updating weights



$$z_j^{l+1} = \sum_{i=1}^{n_l} w_{ji}^l a_i^l = \sum_{i=1}^{n_l} w_{ji}^l f\left(z_i^l\right)$$

The relation between δ_i^l and δ_j^{l+1}

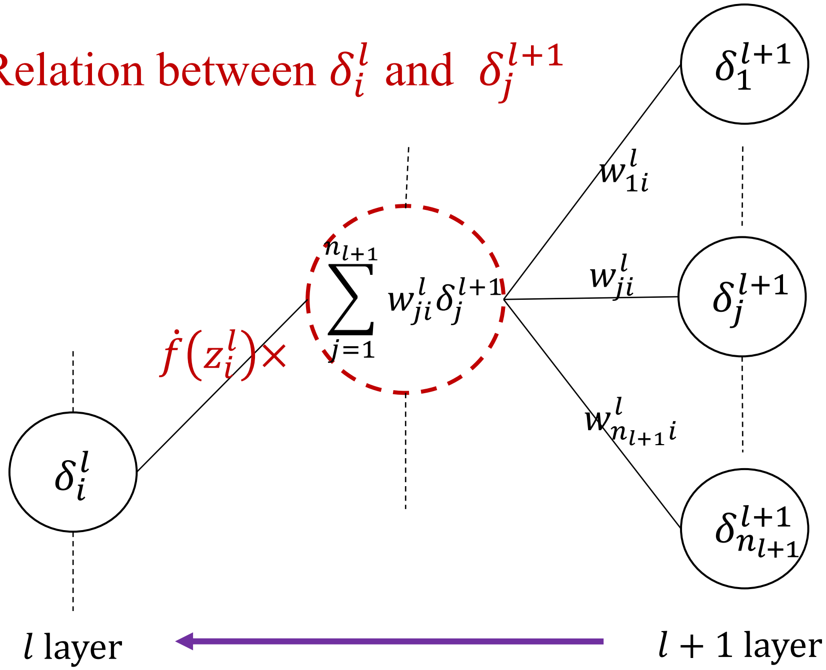
$$\delta_i^l = \frac{\partial J}{\partial z_i^l} = \sum_{j=1}^{n_{l+1}} \frac{\partial J}{\partial z_j^{l+1}} \cdot \frac{\partial z_j^{l+1}}{\partial z_i^l} = \sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \boxed{\frac{\partial z_j^{l+1}}{\partial z_i^l}} = \sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot w_{ji}^l \, \dot{f}\left(z_i^l\right) = \dot{f}\left(z_i^l\right) \cdot \left(\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot w_{ji}^l \right)$$

Backpropagation

Updating weights

$$\delta_i^l = \frac{\partial J}{\partial z_i^l} = \sum_{j=1}^{n_{l+1}} \frac{\partial J}{\partial z_j^{l+1}} \cdot \frac{\partial z_j^{l+1}}{\partial z_i^l} = \sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot \frac{\partial z_j^{l+1}}{\partial z_i^l} = \sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot w_{ji}^l\, \dot{f}(z_i^l) = \dot{f}(z_i^l) \cdot \left(\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot w_{ji}^l \right)$$

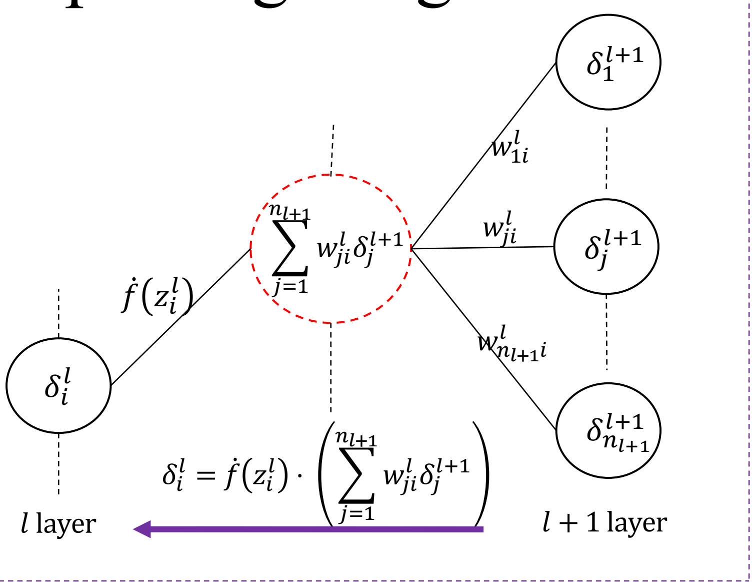
Relation between δ_i^l and δ_j^{l+1}

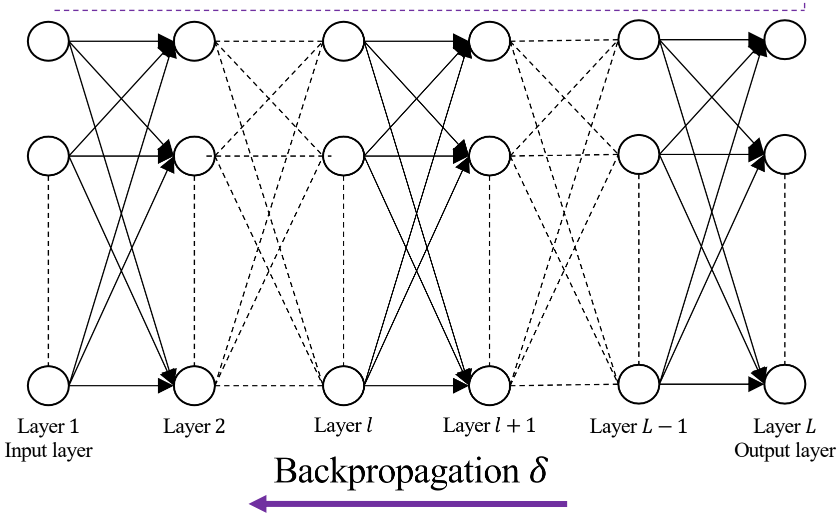


$$\delta_i^l = \dot{f}(z_i^l) \cdot \left(\sum_{j=1}^{n_{l+1}} w_{ji}^l \delta_j^{l+1} \right)$$

Backpropagation

Updating weights





Relation between δ_i^l and δ_j^{l+1}

$$\delta_i^l = \dot{f}(z_i^l) \cdot \left(\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot w_{ji}^l \right)$$

$$\delta_i^L = \frac{\partial J}{\partial z_i^L}$$

If

$$J = \frac{1}{2} \sum_{j=1}^{n_L} \left(a_j^L - y_j^L\right)^2$$

then,

$$\delta_i^L = \frac{\partial J}{\partial z_i^L} = \left(a_i^L - y_i^L\right) \cdot \frac{\partial a_j^L}{\partial z_i^L}$$
$$= \left(a_i^L - y_i^L\right) \cdot \dot{f}(z_i^L)$$

Backpropagation

Conclusion: BP for FNN

Forward computing: $y = f(\sum_{i=1}^{n} w_i x_i)$

Define cost function: $J = J(w^1, \cdots, w^{L-1})$

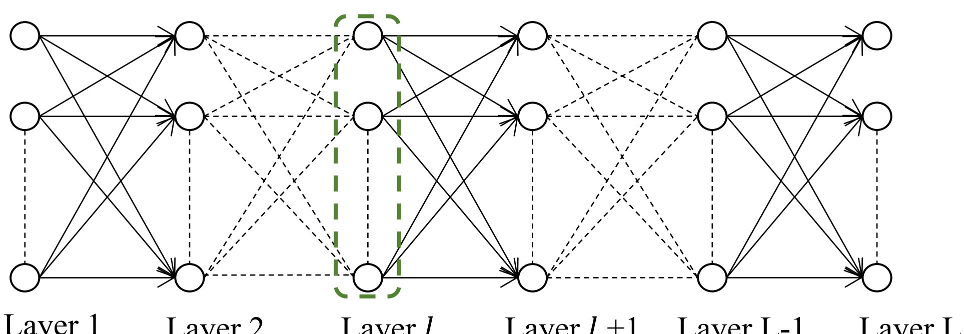
Updating rule: $w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$

Define δ: $\delta_i^l = \frac{\partial J}{\partial z_i^l}$

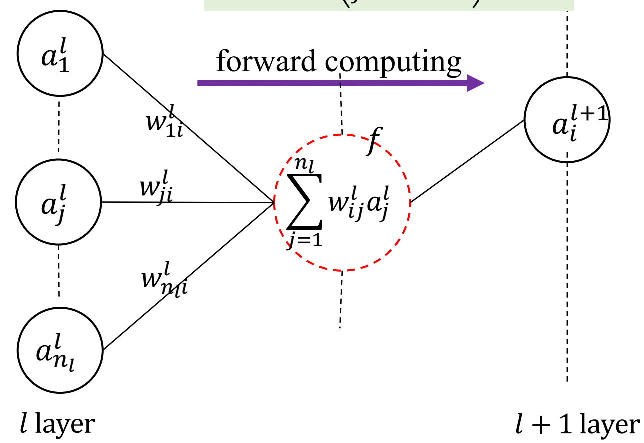
Find the relation: $\frac{\partial J}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$

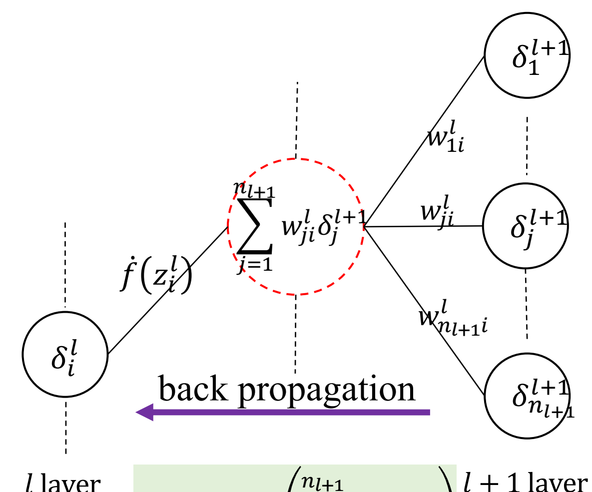
Back propagation: $\delta_i^L = \frac{\partial J}{\partial z_i^L} = (a_i^L - y_i^L) \cdot \dot{f}(z_i^L)$

$$\delta_i^l = \dot{f}(z_i^l) \cdot \left(\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot w_{ji}^l \right)$$



$$a_i^{l+1} = f\left(\sum_{j=1}^{n_l} w_{ij}^l a_j^l \right)$$



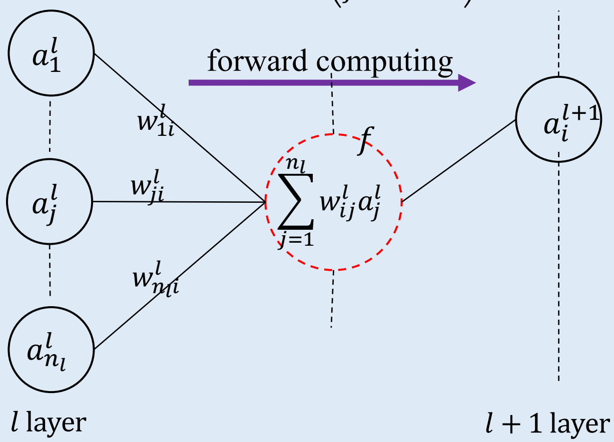


$$\delta_i^l = \dot{f}(z_i^l) \cdot \left(\sum_{j=1}^{n_{l+1}} w_{ji}^l \delta_j^{l+1} \right)$$

Backpropagation

Conclusion: BP for FNN

$$a_i^{l+1} = f\left(\sum_{j=1}^{n_l} w_{ij}^l a_j^l\right)$$



function $fc(W^l, a^l)$
$for\ i = 1{:}\,n_{l+1}$

$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l$$

$$a_i^{l+1} = f(z_i^{l+1})$$

end

function $bc(W^l, \delta^{l+1})$
$for\ i = 1{:}\,n_l$

$$\delta_i^l = \dot{f}(z_i^l) \cdot \left(\sum_{j=1}^{n_{l+1}} w_{ji}^l \delta_j^{l+1}\right)$$

end