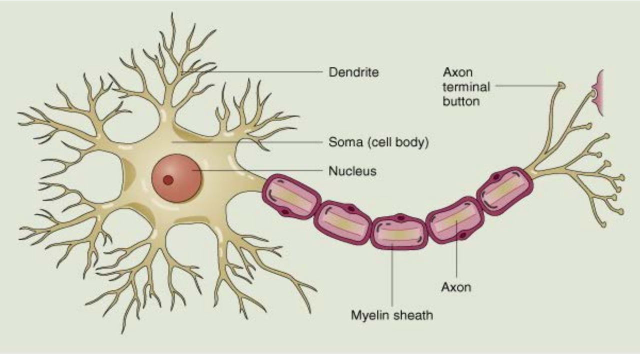
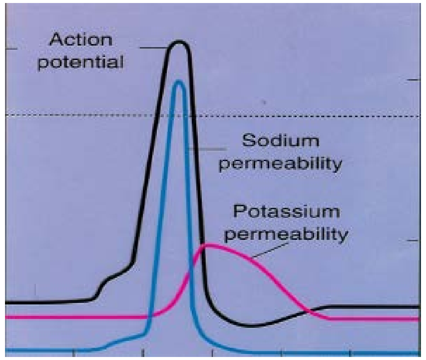
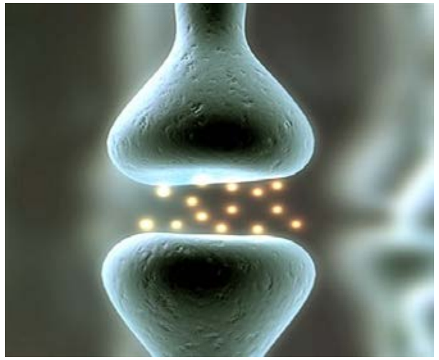
Outline

- Brief Review of Computational Model of Neural Networks
- Network Performance: Cost Function
- Steepest Gradient Method
- Backpropagation
- Three Pages to Understand BP
- Only One Page to Understand BP
- The BP Algorithm
- Assignment

Computational Model of Neurons







Neuron output

a

----

z

Neuron input

x_1 Connection weights

w_1

x_i w_i

w_n

x_n

$z = \sum_{i=1}^{n} w_i x_i$ | f

Activation function

$a = f(z)$

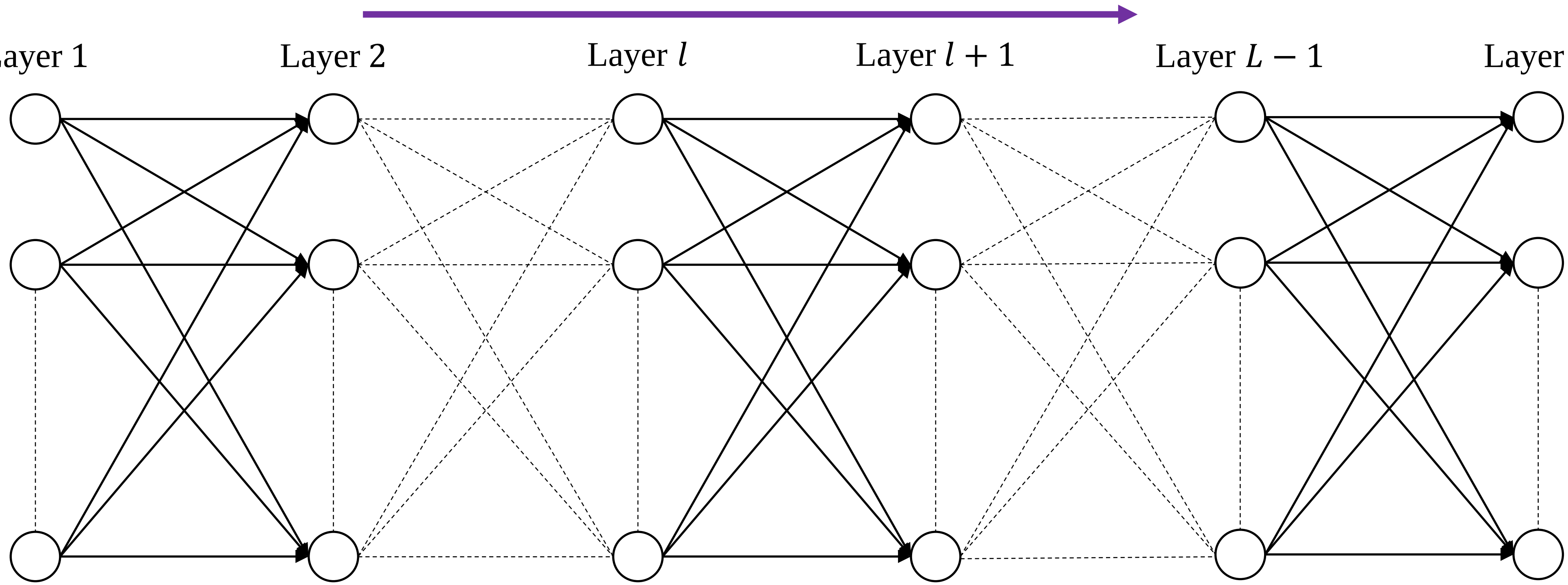
Total input

Neuron output

inputs

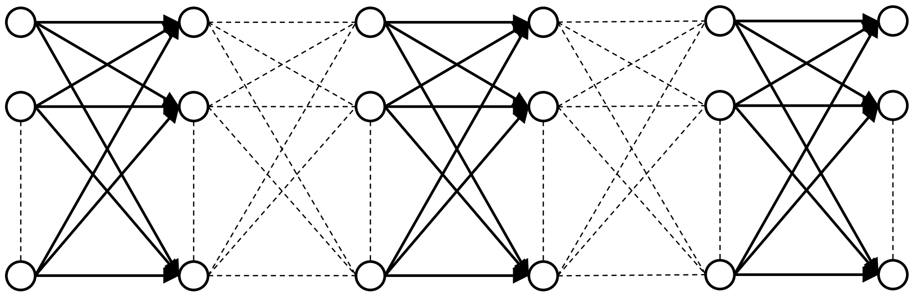
$$y = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

Computational Model of Neural Networks

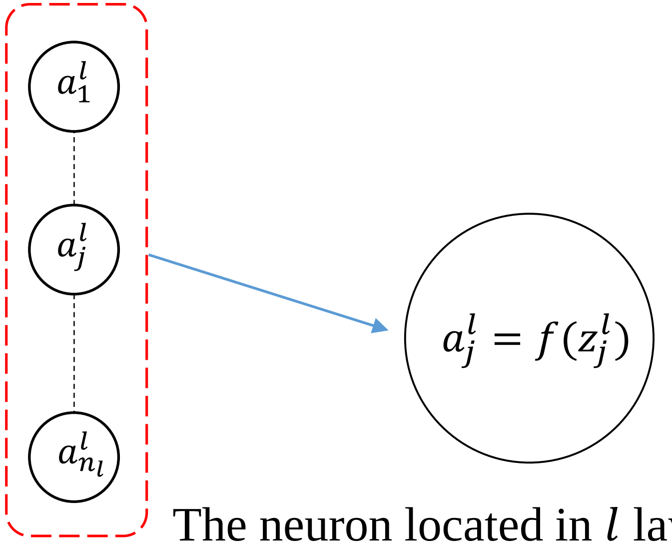


neurons	n_1		n_2		n_l		n_{l+1}		n_{L-1}		n_L
weights		w^1				w^l				w^{L-1}	
values	a^1		a^2		a^l		a^{l+1}		a^{L-1}		a^L

Computational Model of Neural Networks



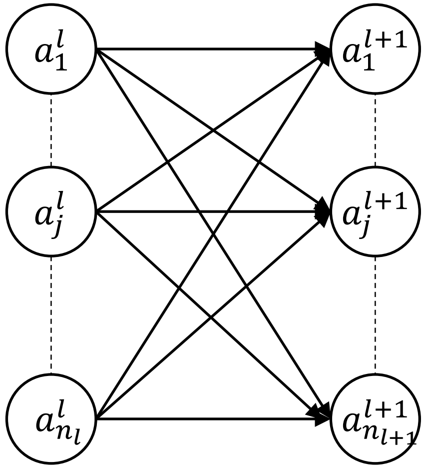
layer l contains n_l neurons.



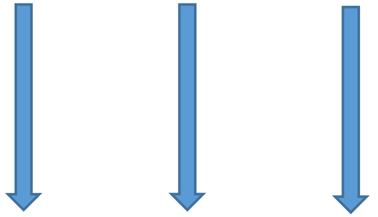
Vector form

$$a^l = \begin{bmatrix} a_1^l \\ \vdots \\ a_j^l \\ \vdots \\ a_{n_l}^l \end{bmatrix}$$

The neuron located in l layer j^{th} place,
a_j^l denotes the output value of the neuron.

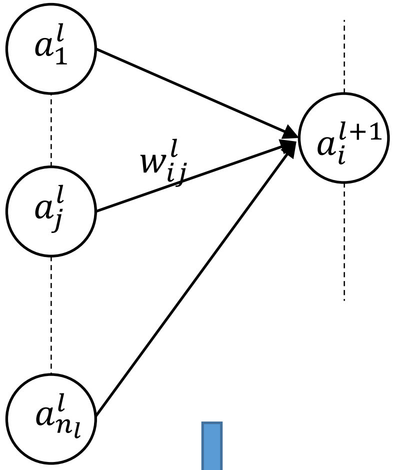


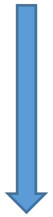
Layer l Layer $l+1$



a^l w^l a^{l+1}

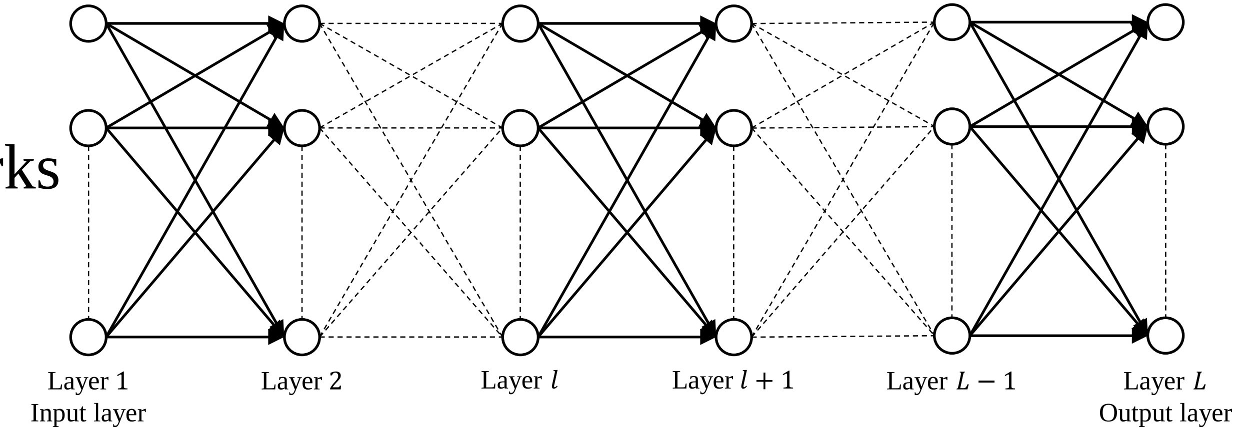
a^l: input of $l+1$ layer
a^{l+1}: representation of a^l

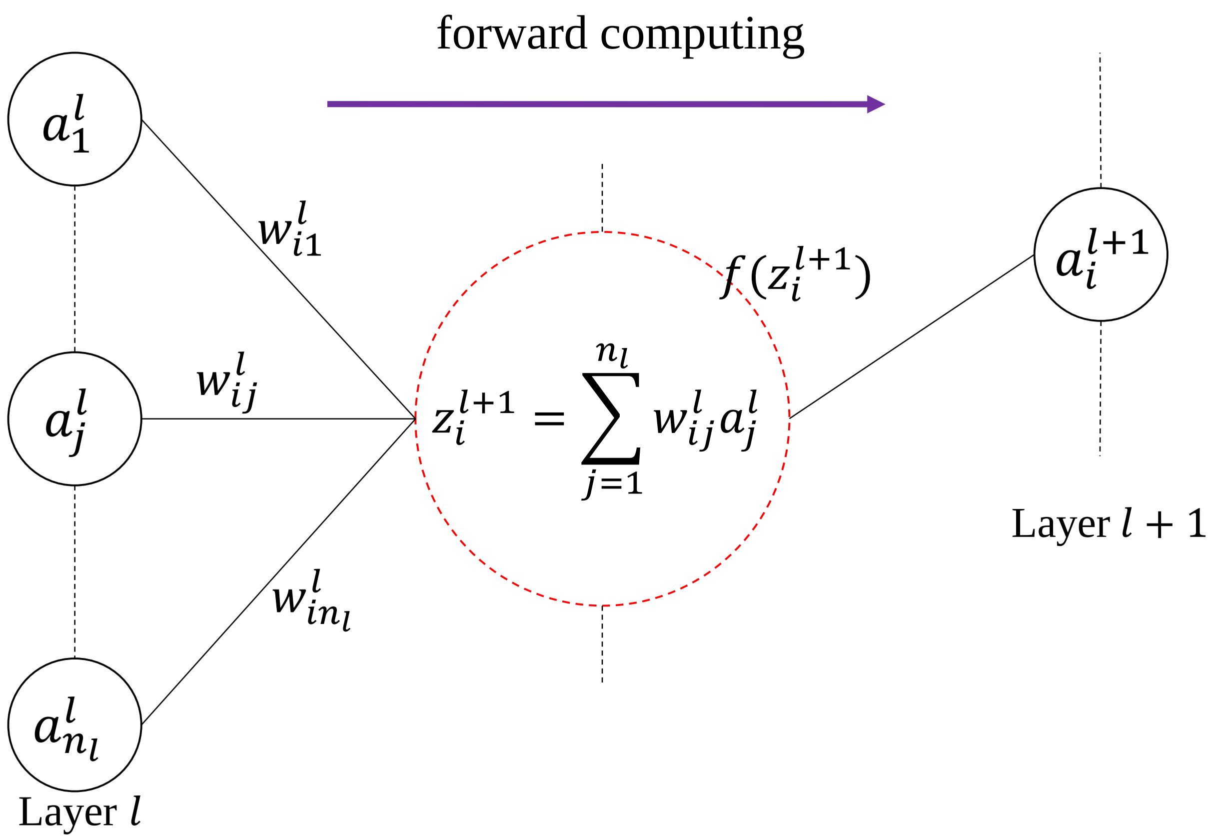




$$W^l = \begin{bmatrix} w_{11}^l & \cdots & w_{1n_l}^l \\ \vdots & w_{ij}^l & \vdots \\ w_{n_{l+1}1}^l & \cdots & w_{n_{l+1}n_l}^l \end{bmatrix}_{n_{l+1} \times n_l}$$

Computational Model of Neural Networks





Component form

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

Vector form

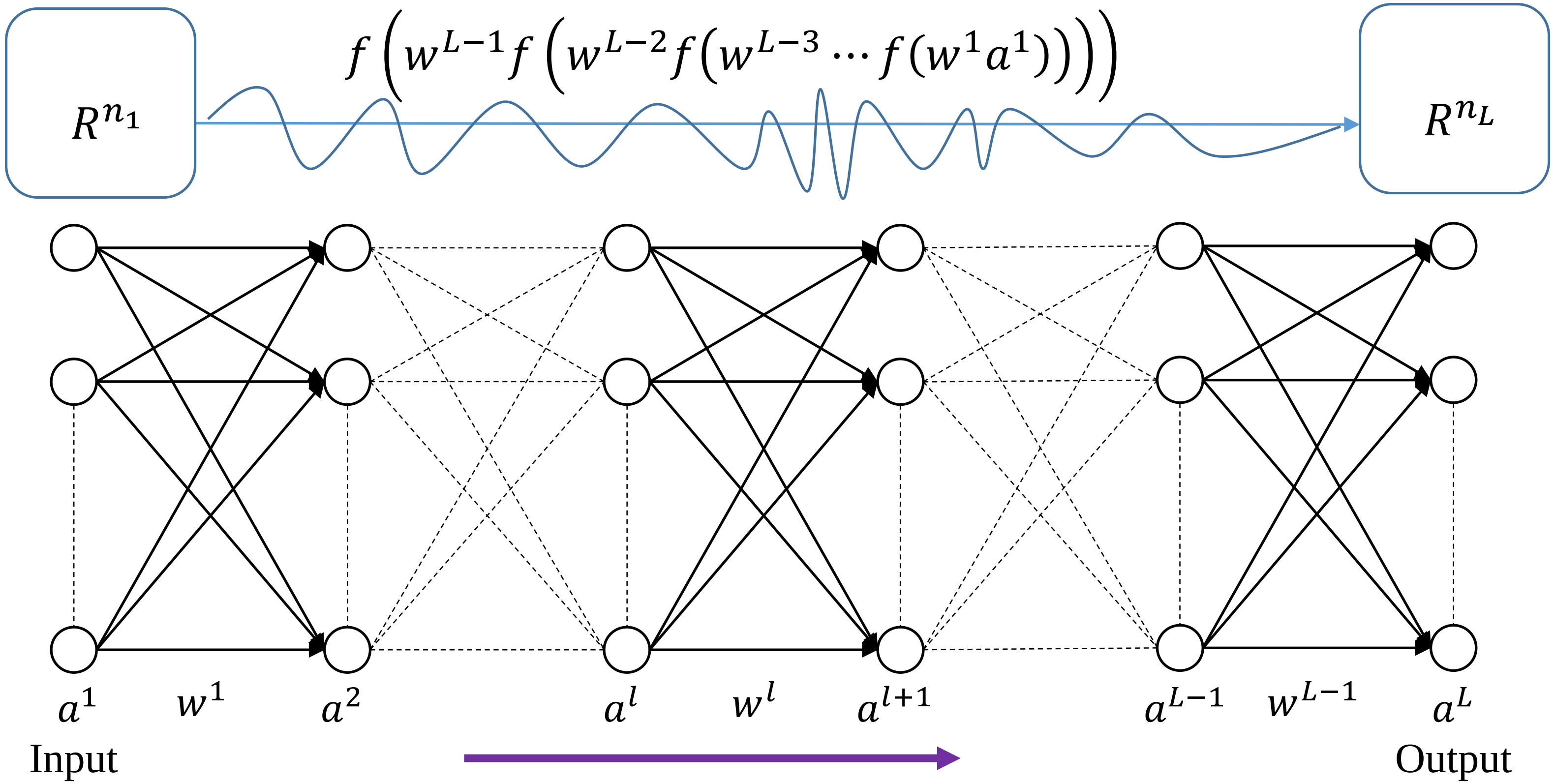
$$\begin{cases} a^{l+1} = f\left(z^{l+1}\right) \\ z^{l+1} = w^l a^l \end{cases}$$

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

Computational Model of Neural Networks

$A\ feedforward\ neural\ network\ is\ in\ fact\ a\ nonlinear\ mapping\ from\ R^{n_1}\ space\ to\ R^{n_L}\ space.$

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

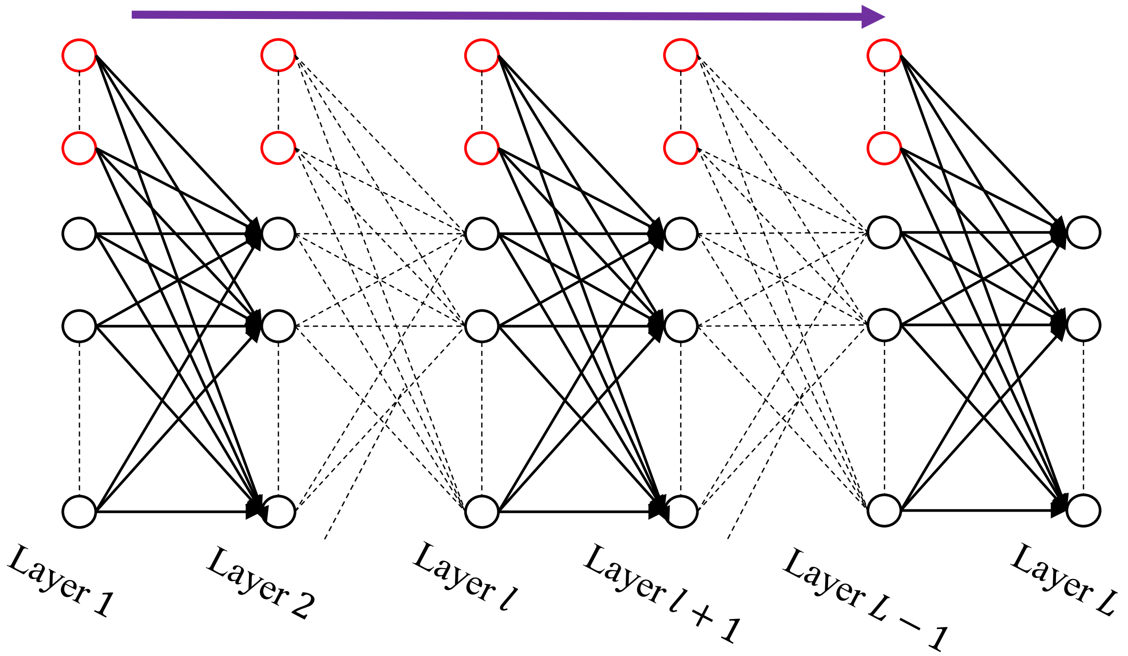


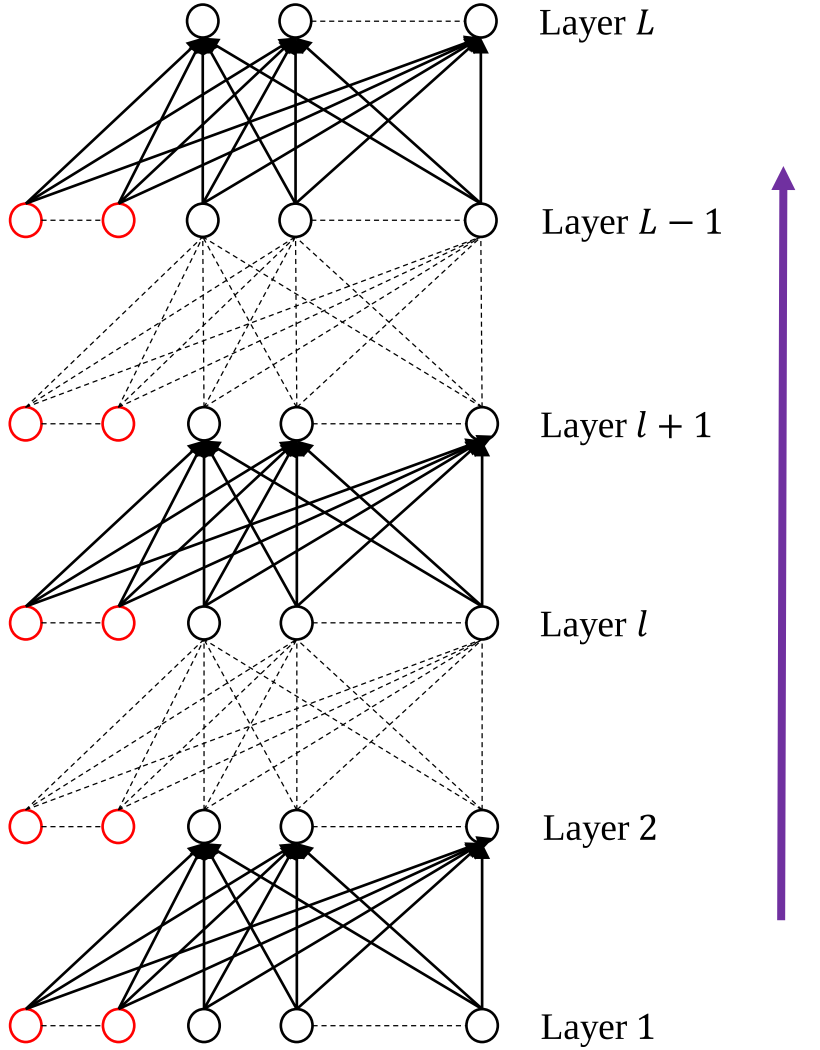
Computational Model of Neural Networks

External inputs:

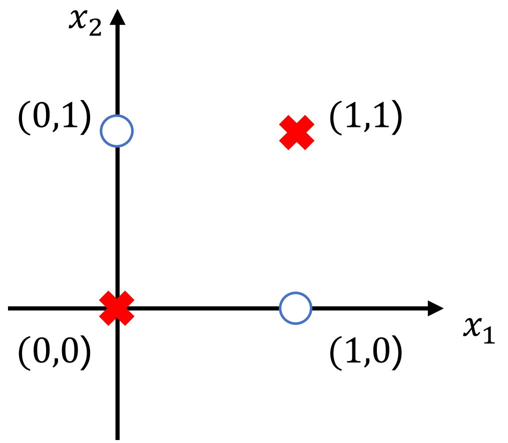
If neurons in l layer are not connected to any neurons in the $l-1$ layer, these neurons are called external inputs of $l+1$ layer.
External inputs can exist in any layer except the last one.

External inputs



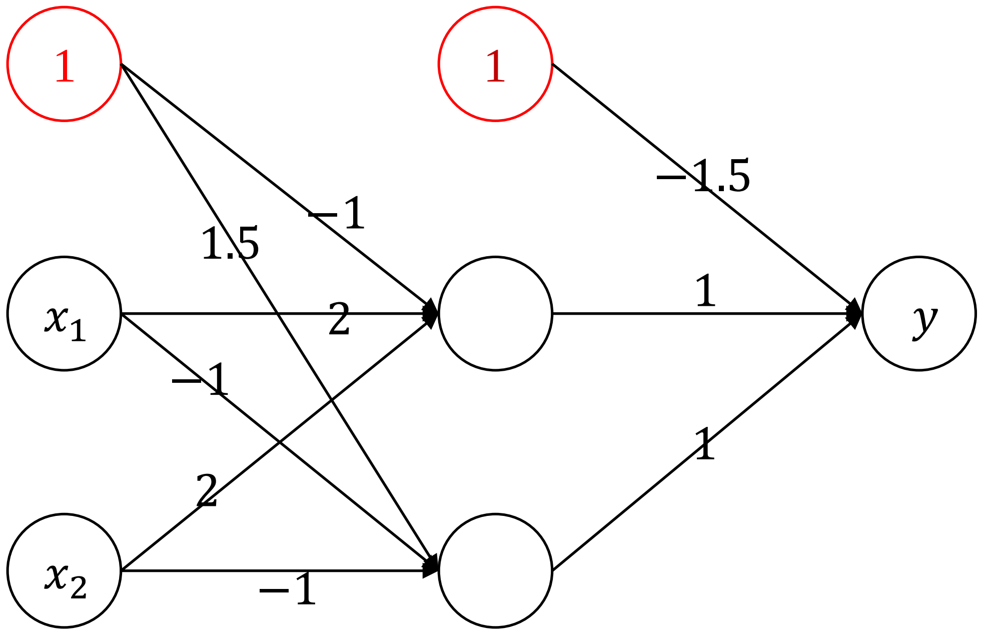


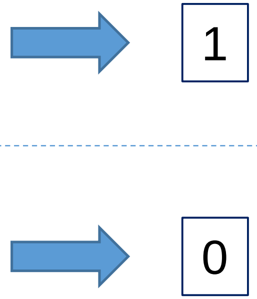
Example: XOR problem



$\begin{bmatrix}0\\1\end{bmatrix}$ $\begin{bmatrix}1\\0\end{bmatrix}$

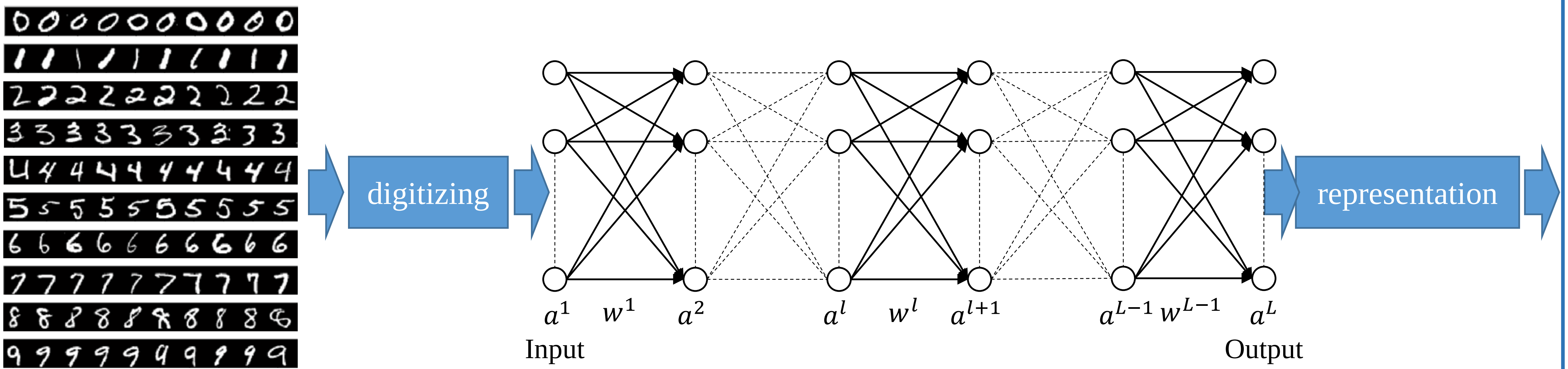
$\begin{bmatrix}0\\0\end{bmatrix}$ $\begin{bmatrix}1\\1\end{bmatrix}$

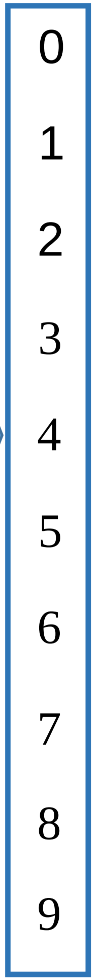




Problem: How to design the NN? Are there any methods to design the connection weights?

Example: Handwritten Digits Recognition



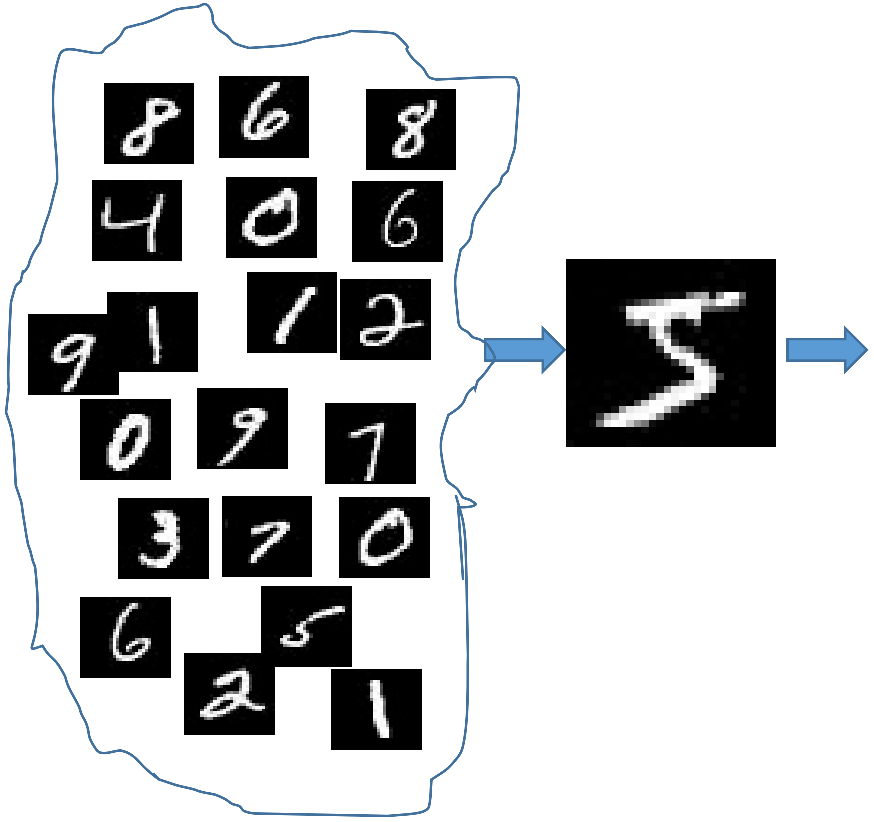


Problem: How to design the NN? Are there any methods to design the connection weights?

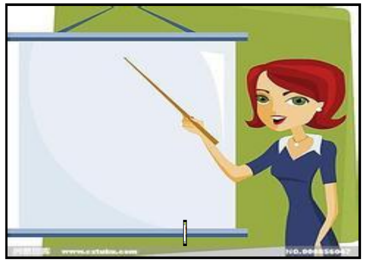
Outline

- Brief Review of Computational Model of Neural Networks
- Network Performance: Cost Function
- Steepest Gradient Method
- Backpropagation
- Three Pages to Understand BP
- Only One Page to Understand BP
- The BP Algorithm
- Assignment

Network Performance: Cost Function



Training



Good Performance!

The father knows the correct answer.

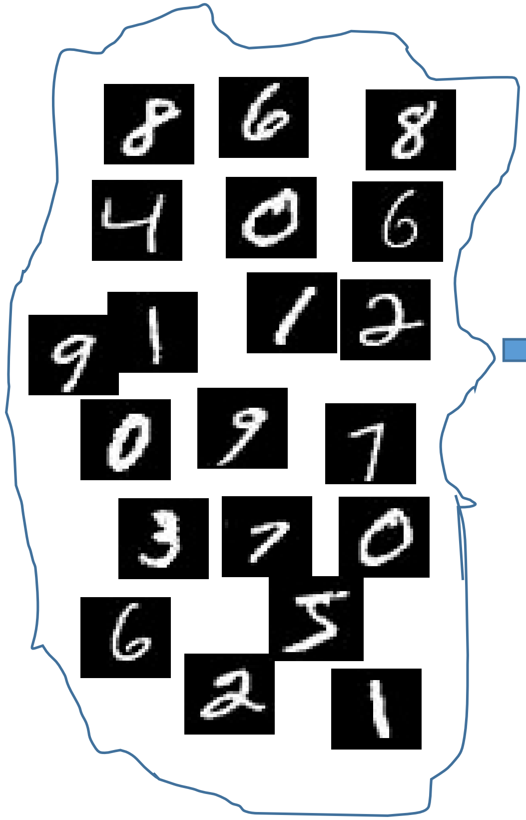
5

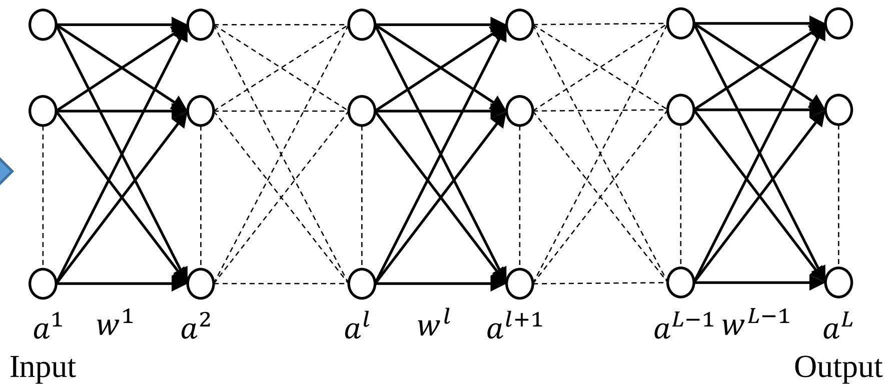
Two important factors:

1. There must be a measure to measure the correctness between correct answer and the boy's real output. ----- Performance function.

2. There must be a mechanism to change the knowledge system of the boy. ---- Learning algorithm.

Network Performance: Cost Function





Changing the weights: Learning algorithm

Network Output

$$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$$

Target Output

$$y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix}$$

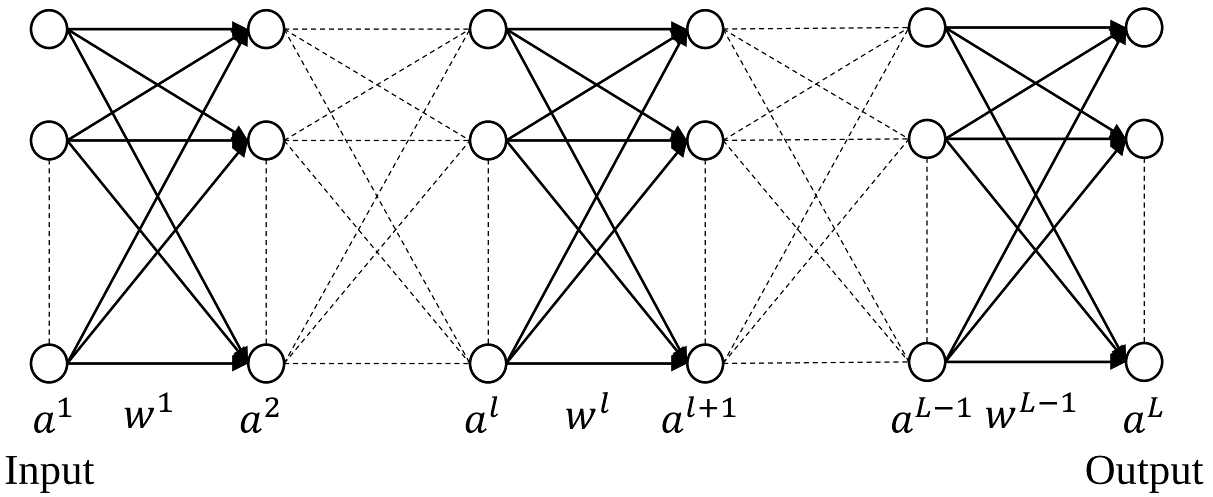
$J(a^L, y^L)$

Performance function $J(a^L, y^L)$, or cost function, is used to describe the distance between a^L and y^L, $J(a^L, y^L)$ is indeed a function of $(w^1, \cdots, w^L)$, i.e.,

$$J = J(w^1, \cdots, w^L).$$

Problem: How to construct a cost function?

Network Performance: Cost Function



Target Output

$$y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix}$$

Network Output

$$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$$

A cost function J describes the performance of the network. If the J is small, it implies that the network output a^L close to the target output y^L, the network is called in good performance. Since J is a function with variables $(w^1, \cdots, w^L)$, good performance means to find suitable $(w^1, \cdots, w^L)$ such that J is small. The process of looking for suitable $(w^1, \cdots, w^L)$ is called network learning.

Problem: How to learn?

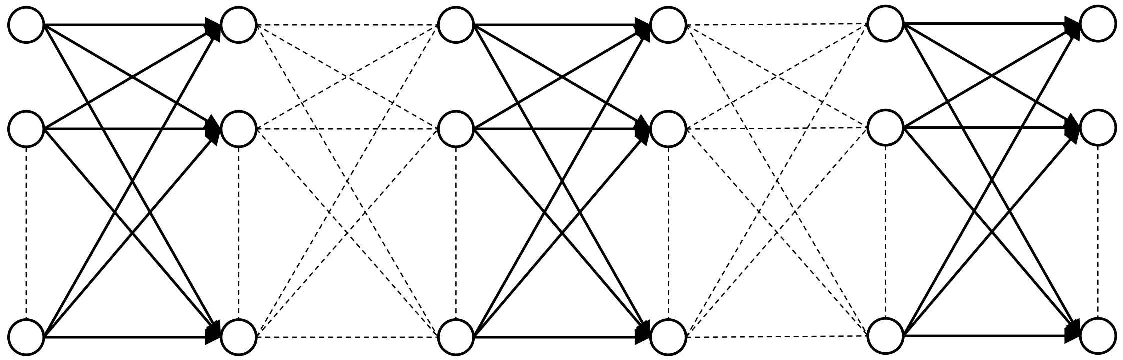
There are many ways to construct a cost function. A frequently used cost is as follows:

$$e_j = a_j^L - y_j^L$$

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

Clearly, J is a function of $w^1, \cdots, w^L$.

Network Performance: Cost Function



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 $w^l (l = 1, \cdots, L)$, thus the network learning is to looking for some $w^l (l = 1, \cdots, L)$ such that $w^l (l = 1, \cdots, L)$ is a minimum point of J.

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

Target Output

$$y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix}$$

Network Output

$$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$$

A frequently used cost function:

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

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

Outline

- Brief Review of Neural Networks Structure
- Network Performance: Cost Function
- Steepest Gradient Method
- Backpropagation
- Three Pages to Understand BP
- Only One Page to Understand BP
- The BP Algorithm
- Assignment

Minimum Points

General Nonlinear function

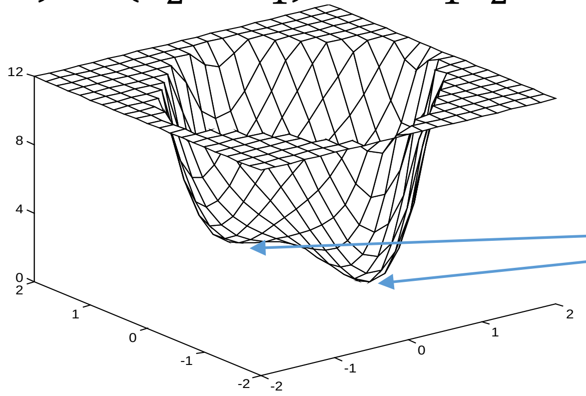
$$F(x),\, x \in R^n$$

x^* is a minimum point if

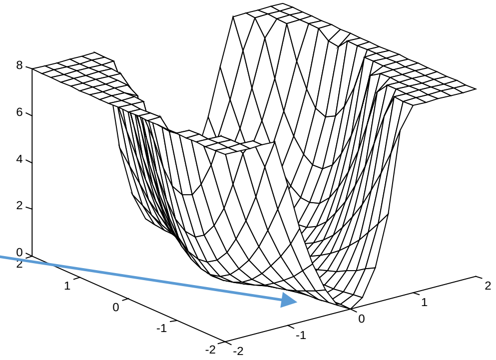
$$F(x^*) \le F(x)$$

for any x that very close to x^*.

$F(x) = (x_2 - x_1)^4 + 8x_1x_2 - x_1 + x_2 + 3$

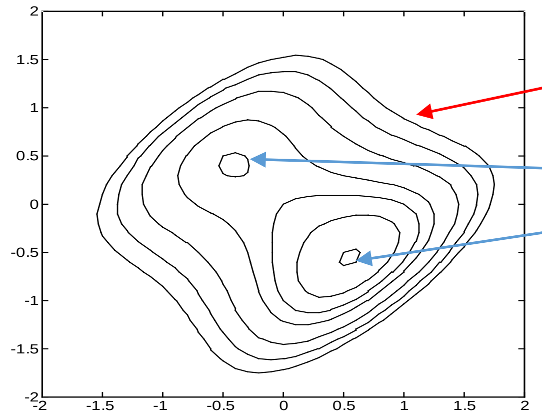


$F(x) = (x_1^2 - 1.5x_1x_2 + 2x_2^2)x_1^2$

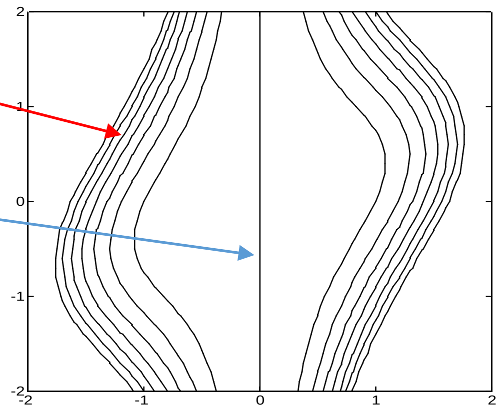


Minima

Contour

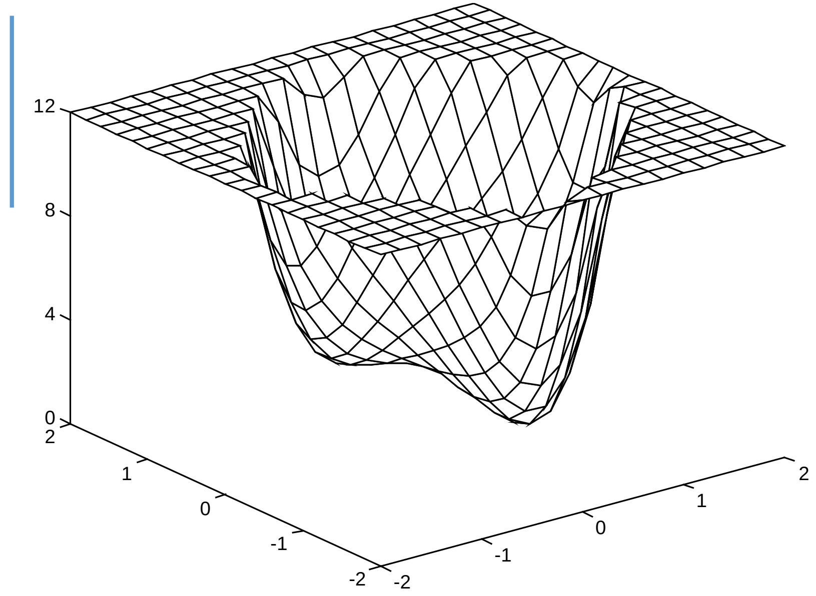


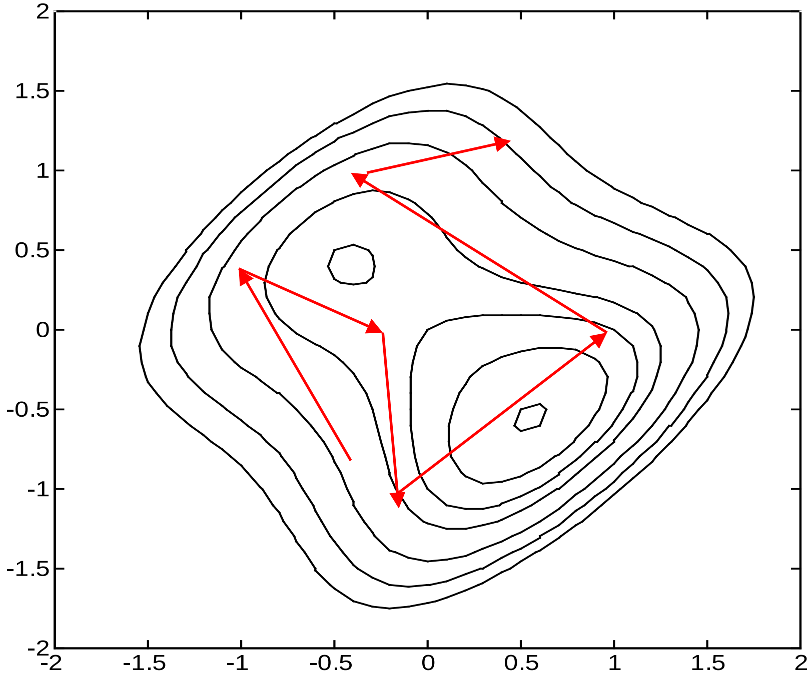
Minimum points



Problem:
How to find the minimum points?

Iteration Method

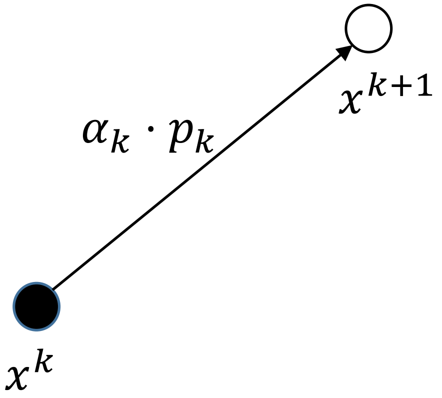




Finding a minimum point step by step

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

To begin the iteration, you must need a given starting point x_0.

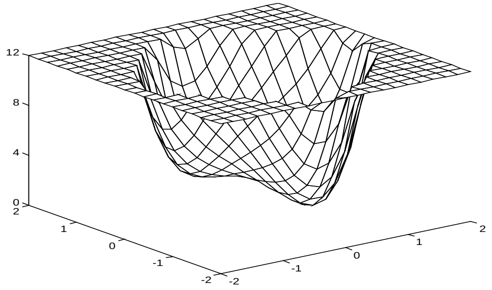


p_k is called searching direction

α_k is learning rate at step k.

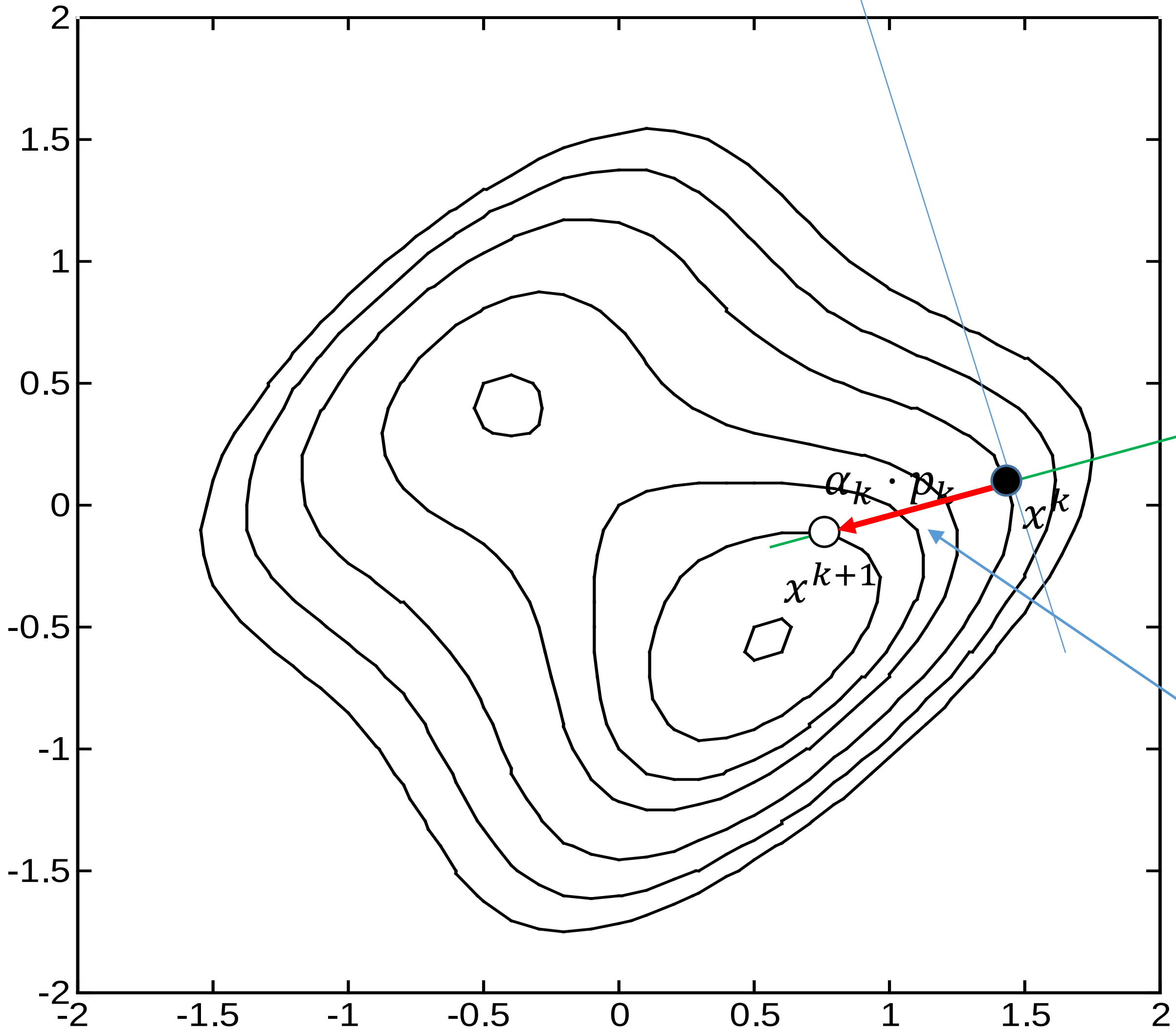
Problem: How to get the searching direction p_k?

Steepest Descent Method



Slowest changing direction

Fastest increasing direction



Steepest descent direction

Gradient:

$$g_k = \nabla F(x)\Big|_{x^k} = \frac{\partial F}{\partial x}\Big|_{x^k} = \begin{pmatrix} \frac{\partial F}{\partial x_1} \\ \vdots \\ \frac{\partial F}{\partial x_n} \end{pmatrix}\Bigg|_{x^k}$$

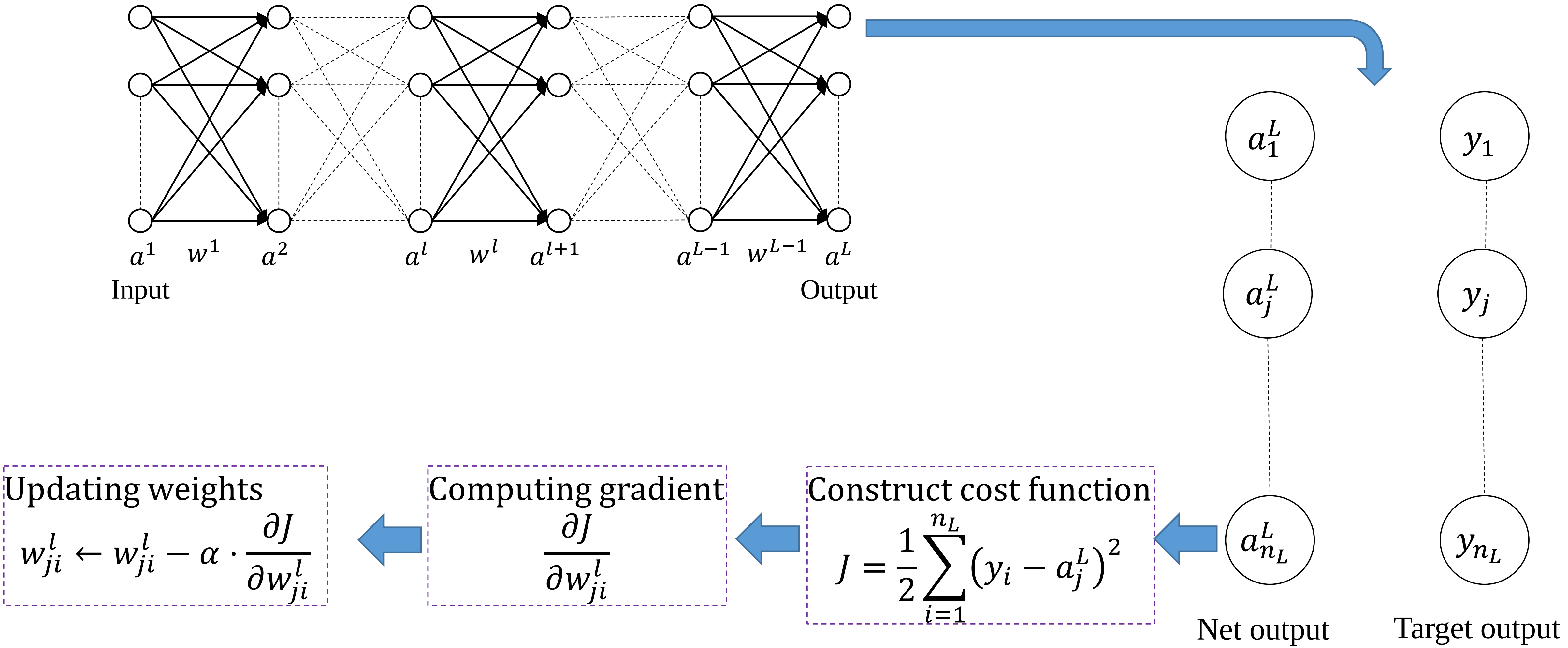
Steepest Descent Algorithm:

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

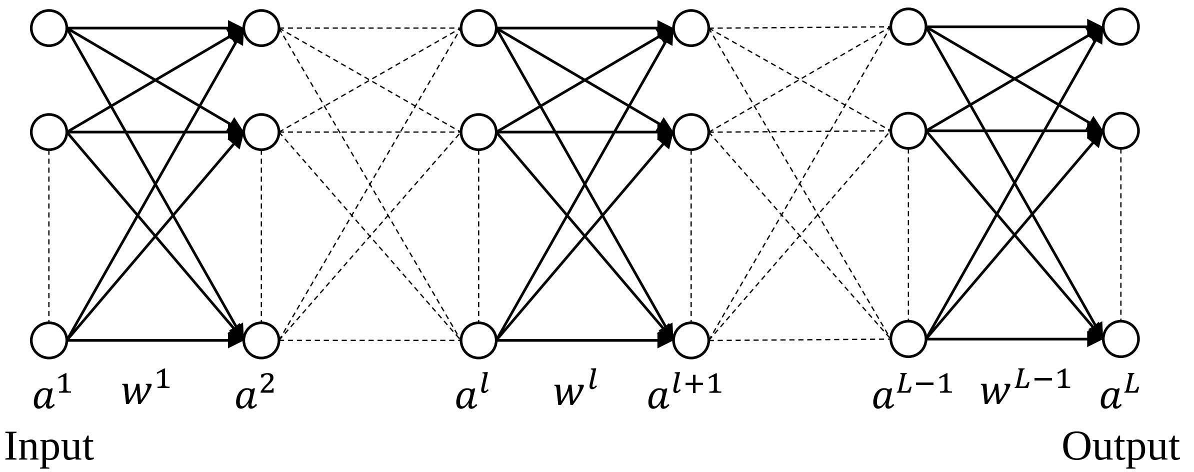
or

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

Steepest Descent Method



Steepest Descent Method



Target Output

$$y^L = \begin{bmatrix} y_1^L \\ \vdots \\ y_{n_L}^L \end{bmatrix}$$

Network Output

$$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_{n_L}^L \end{bmatrix}$$

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

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

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

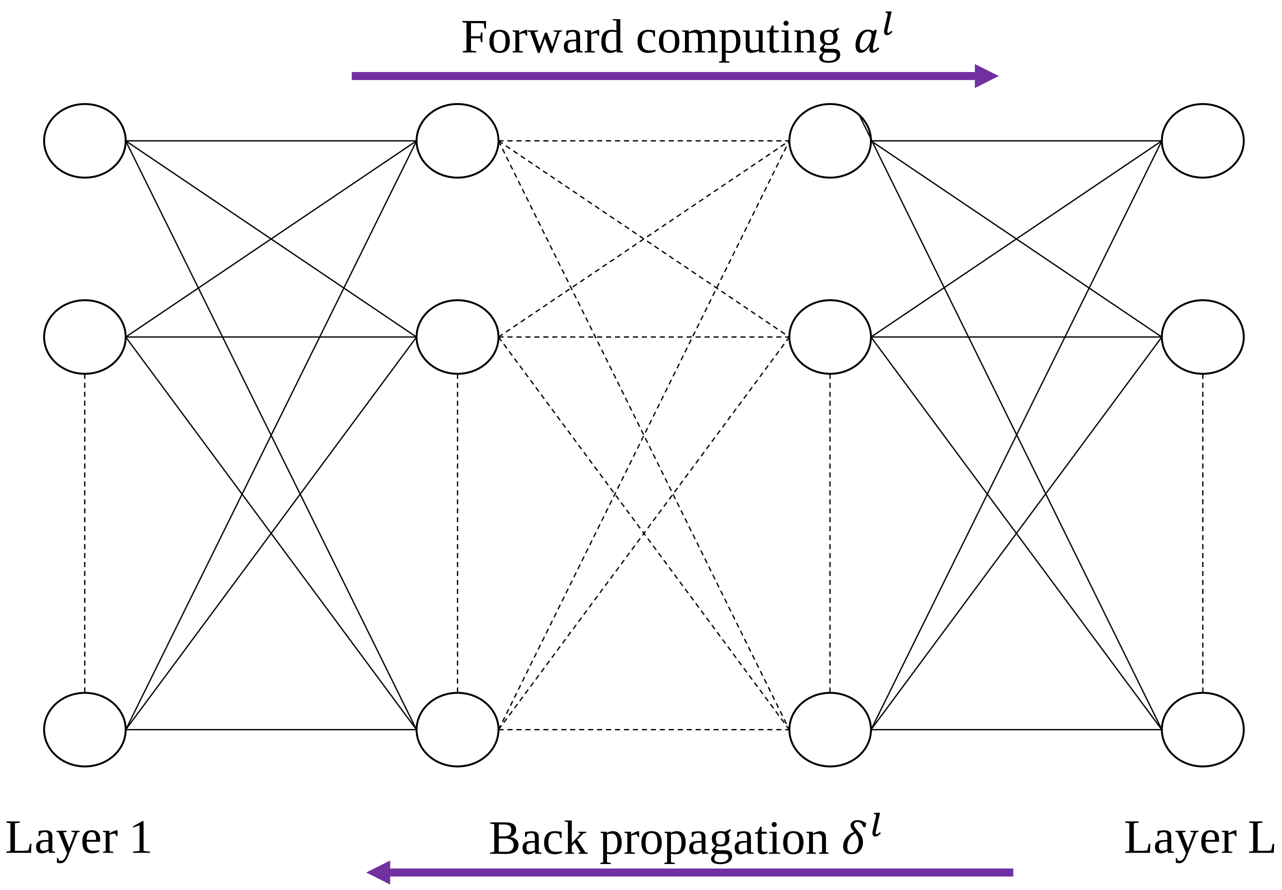
Problem: How to compute $\frac{\partial J}{\partial w_{ji}^l}$?

Answer:
Using the well-known BP method.

Outline

- Brief Review of Neural Networks Structure
- Network Performance: Cost Function
- Steepest Gradient Method
- Backpropagation
- Three Pages to Understand BP
- Only One Page to Understand BP
- The BP Algorithm
- Assignment

Backpropagation



Backpropagation is a efficient way to calculate

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

Cost function:

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

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

l layer

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

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

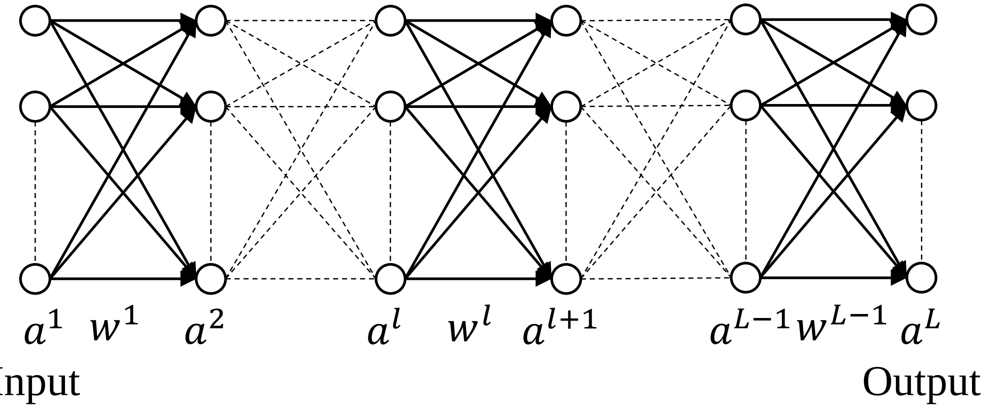
w_{ji}^l

$l+1$ layer

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

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

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



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

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

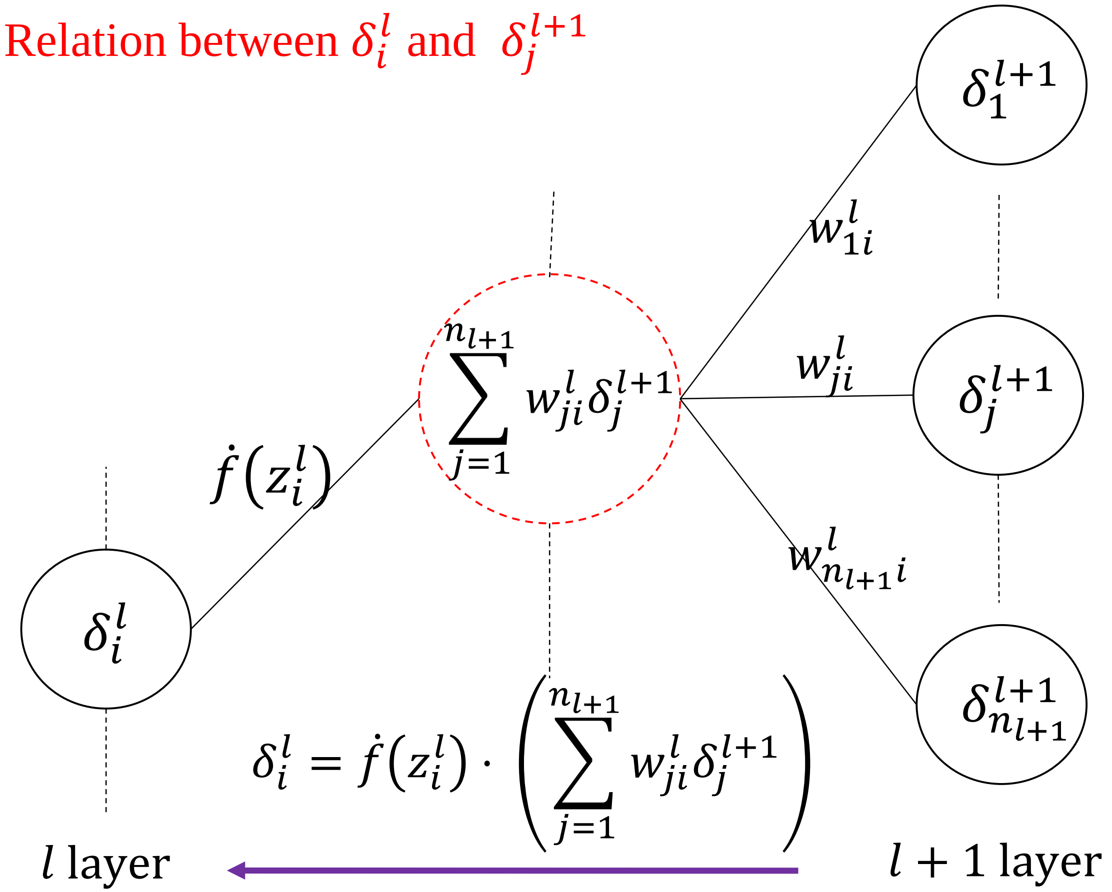
Why?

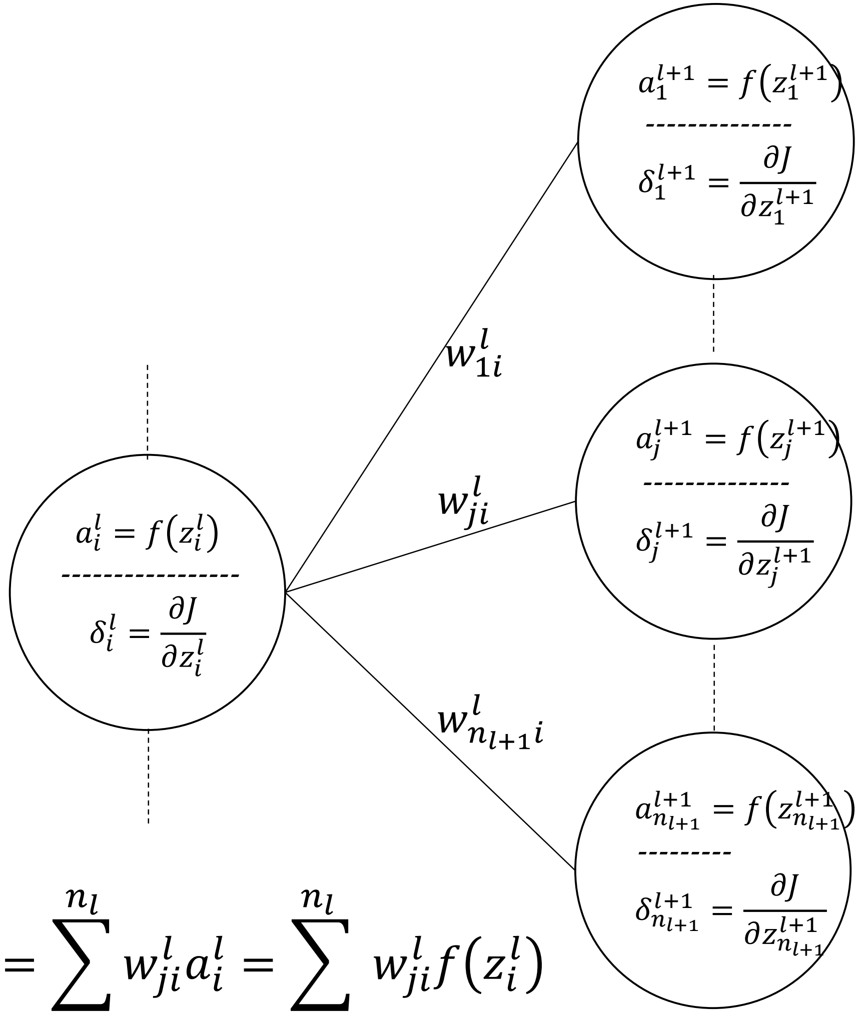
$$\frac{\partial J}{\partial w_{ji}^l} = \frac{\partial J}{\partial z_j^{l+1}} \cdot \frac{\partial z_j^{l+1}}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$$

Next problem:

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

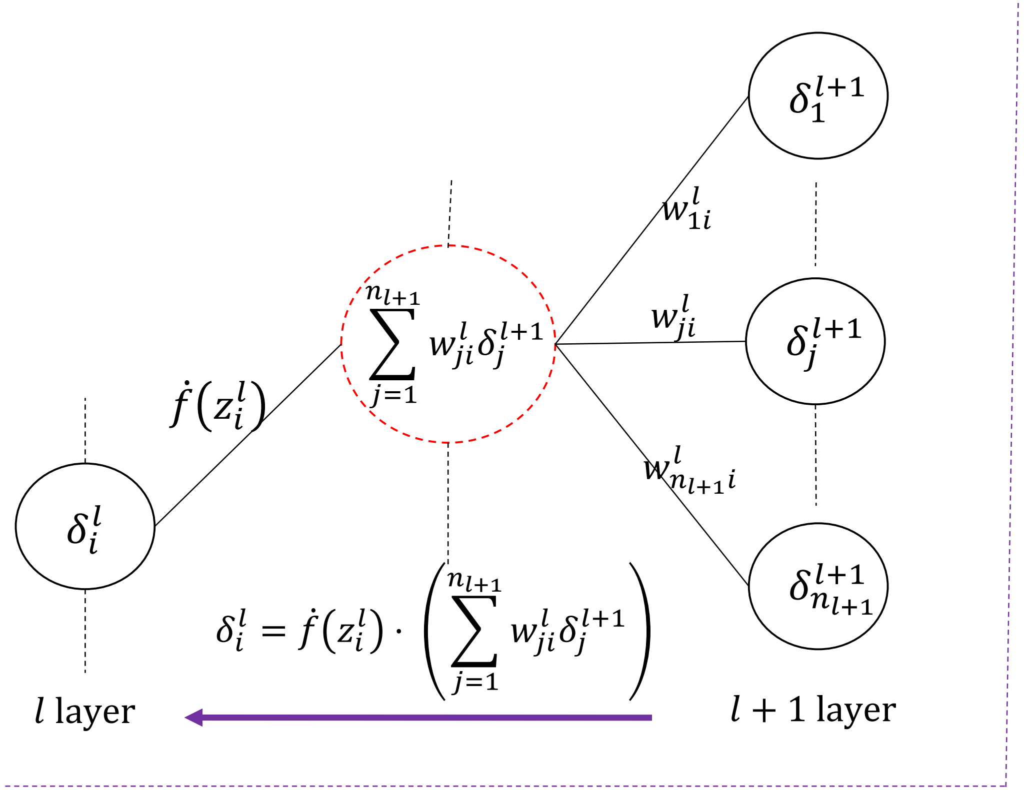
Relation between δ_i^l and δ_j^{l+1}





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

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



Relation between δ_i^l and δ_j^{l+1}

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

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

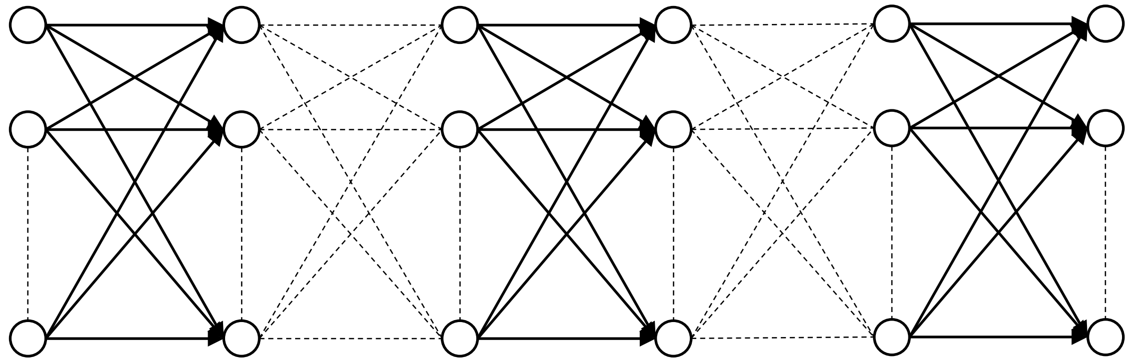
If

$$J = \frac{1}{2} \sum_{j=1}^{n_L} (a_j^L - y_j^L)^2 \quad \text{公式3}$$

then,

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

公式4



Backpropagation δ

Outline

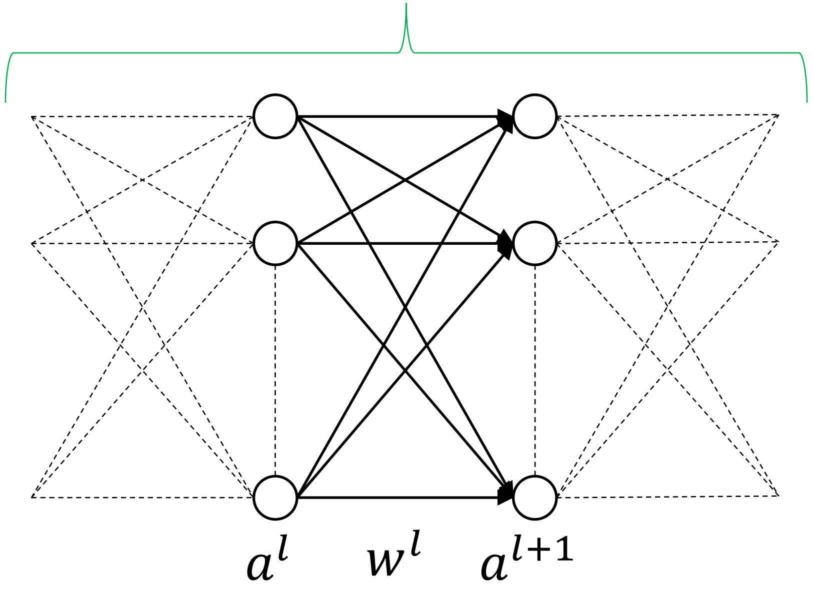
- Brief Review of Neural Networks Structure
- Network Performance: Cost Function
- Steepest Gradient Method
- Backpropagation
- Three Pages to Understand BP
- Only One Page to Understand BP
- The BP Algorithm
- Assignment

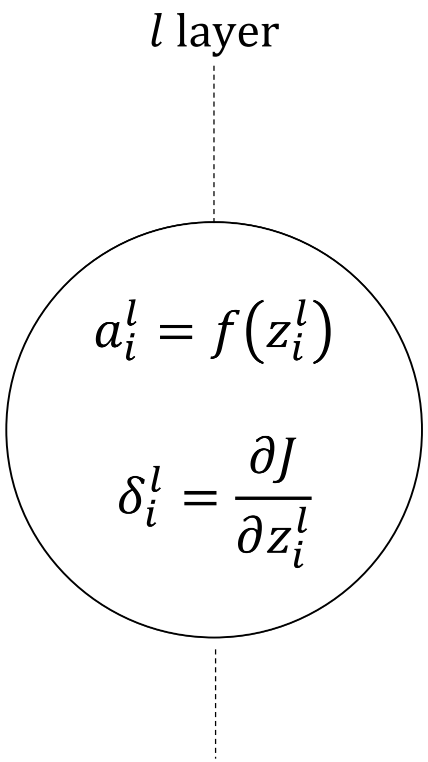
Three Pages to Understand BP: *The first page*

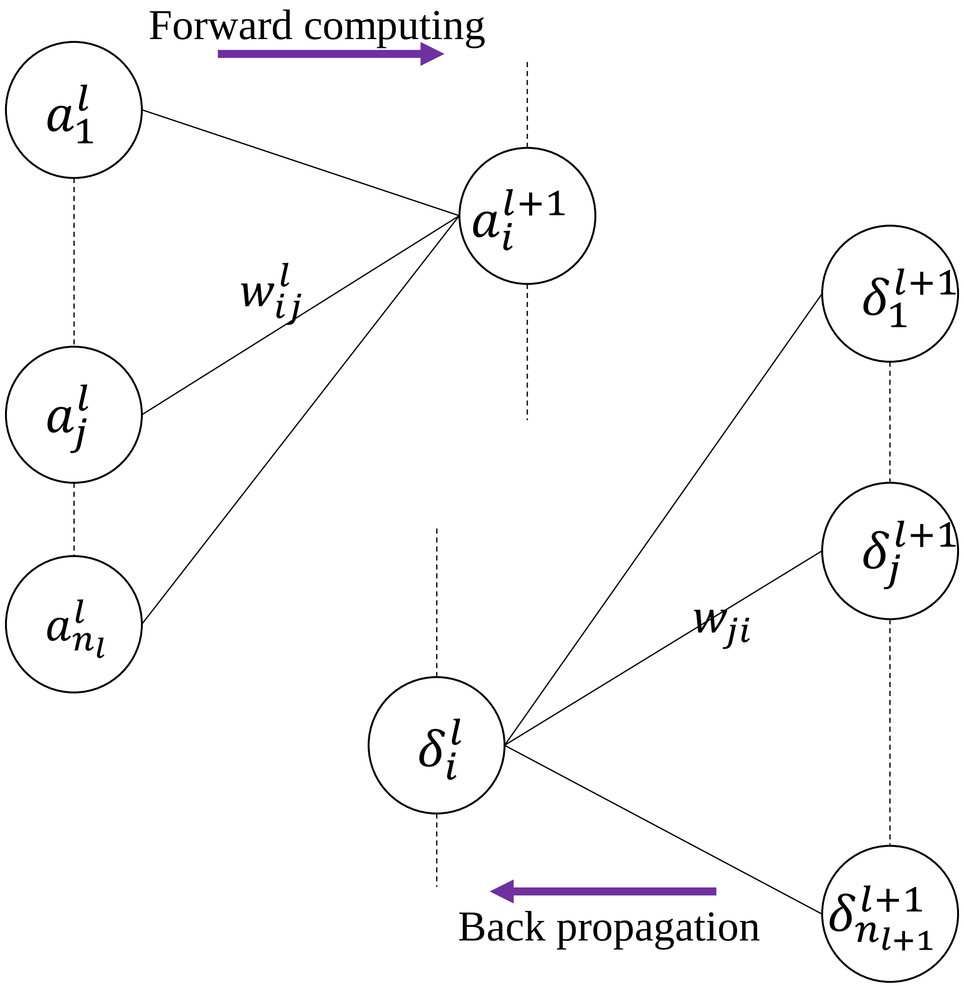
Cost function: $J(w^1, \cdots, w^L)$

Updating rule: $w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$ 最终更新方法，公式7

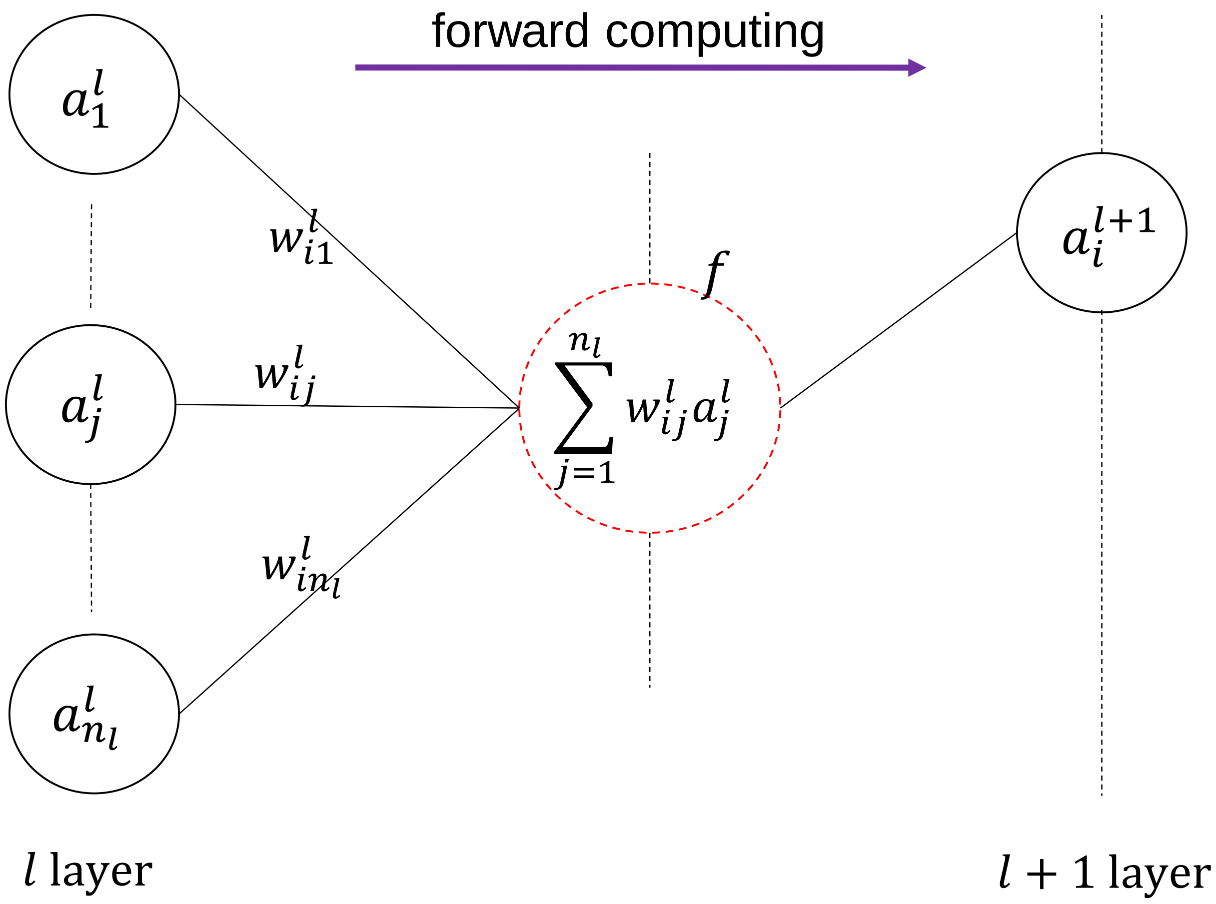
Relationship: $\frac{\partial J}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$ 公式6







Three Pages to Understand BP: *The second page*



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

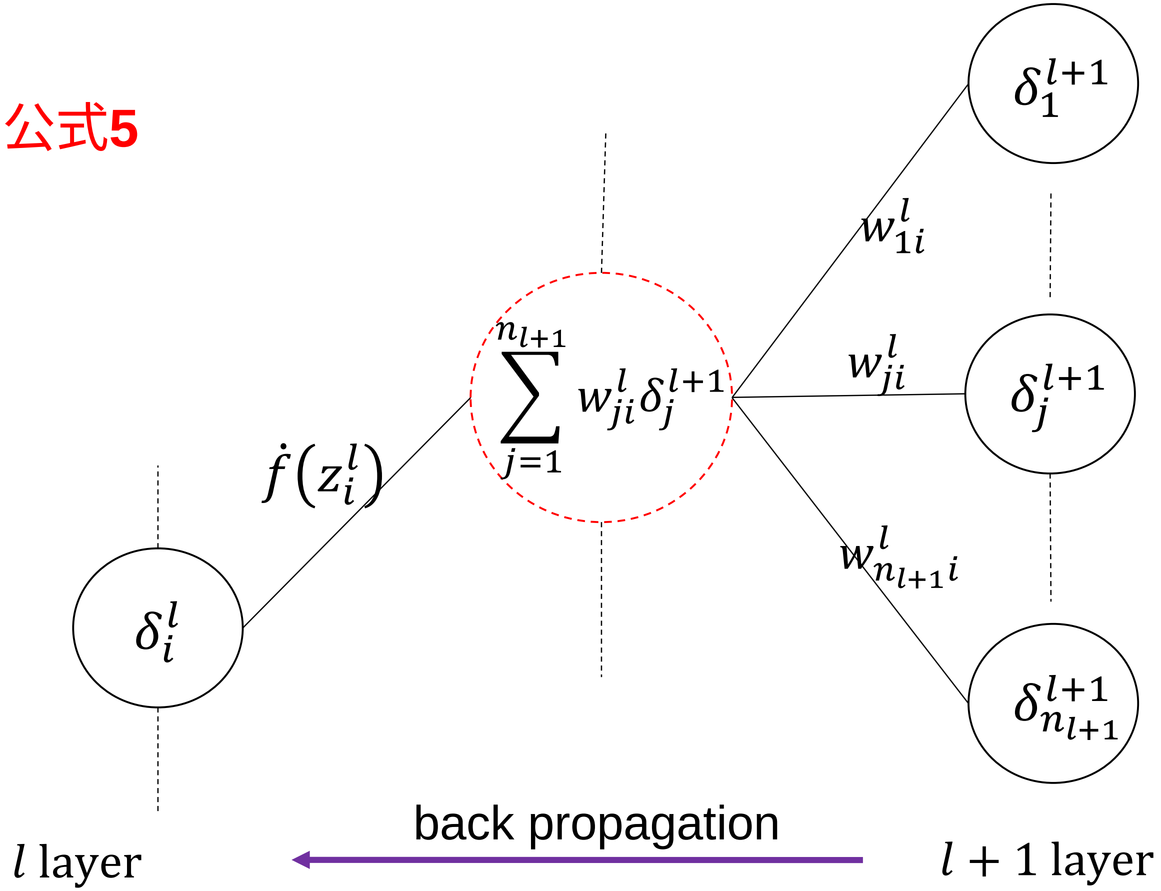
or --------------------

$$a_i^{l+1} = f\left(z_i^{l+1}\right) \quad \text{公式1}$$

$$z_i^{l+1} = \sum_{j=1}^{n_l} w_{ij}^l a_j^l \quad \text{公式2}$$

Three Pages to Understand BP: *The third page*

$$\delta_i^l = \dot{f}(z_i^l) \cdot \left(\sum_{j=1}^{n_{l+1}} w_{ji}^l \delta_j^{l+1} \right)$$ 公式5



Outline

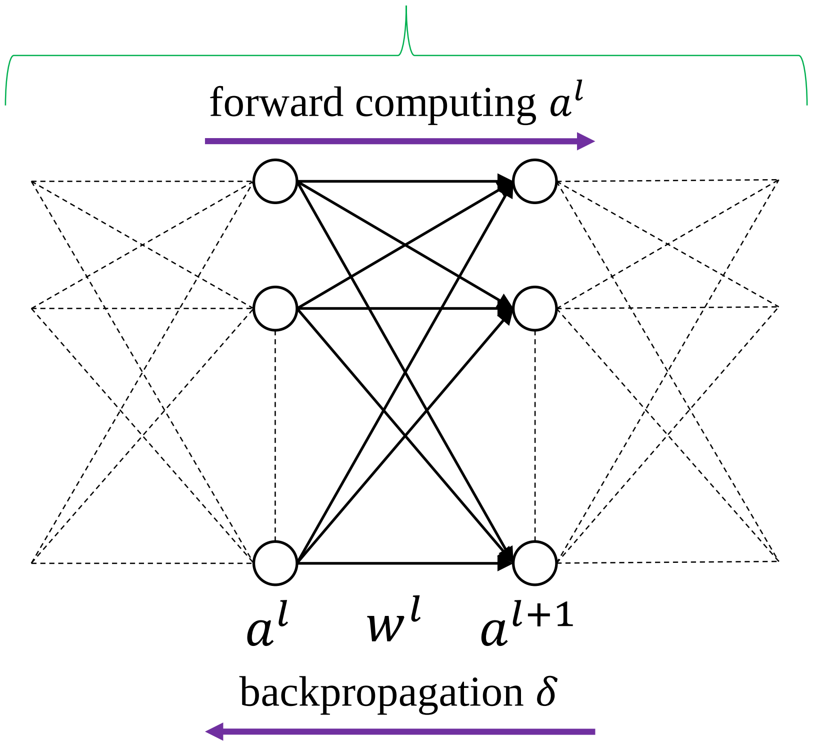
- Brief Review of Neural Networks Structure
- Network Performance: Cost Function
- Steepest Gradient Method
- Backpropagation
- Three Pages to Understand BP
- Only One Page to Understand BP
- The BP Algorithm
- Assignment

Only One Pages to Understand BP

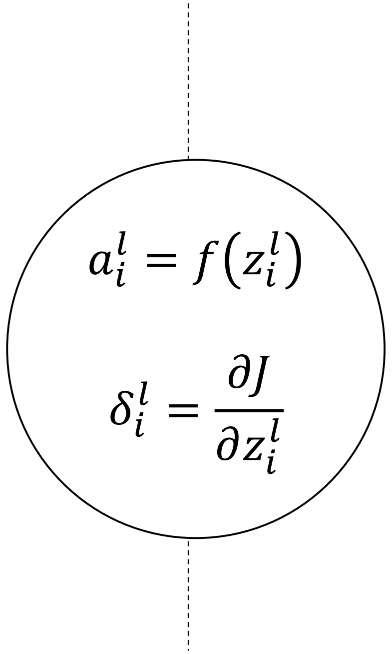
Cost function: $J(w^1, \cdots, w^L)$

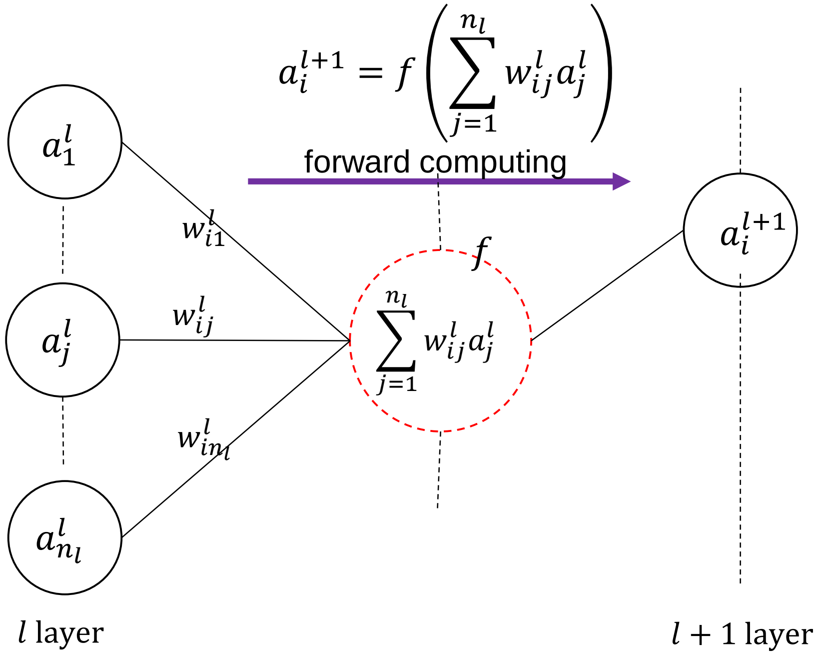
Updating rule: $w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \dfrac{\partial J}{\partial w_{ji}^l}$

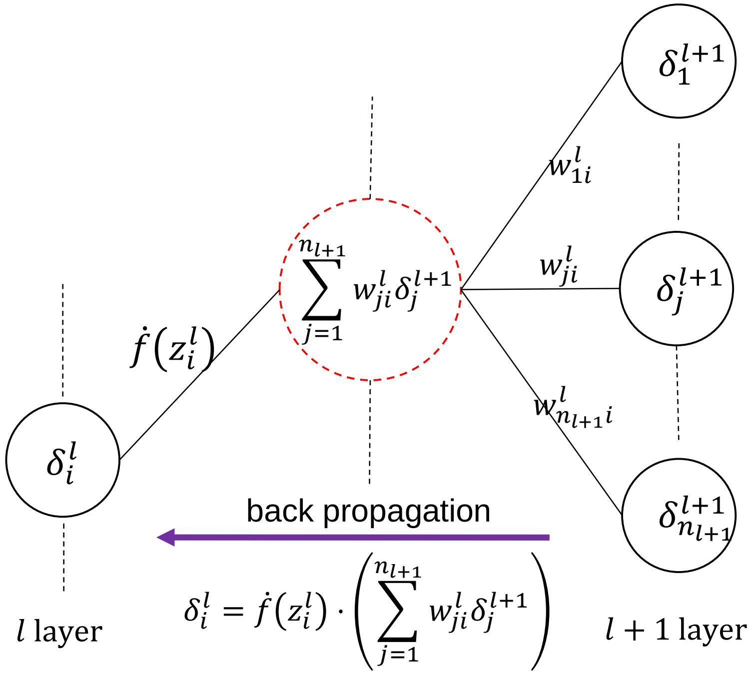
Relationship: $\dfrac{\partial J}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$



$l\ layer\ i^{th}\ neuron$

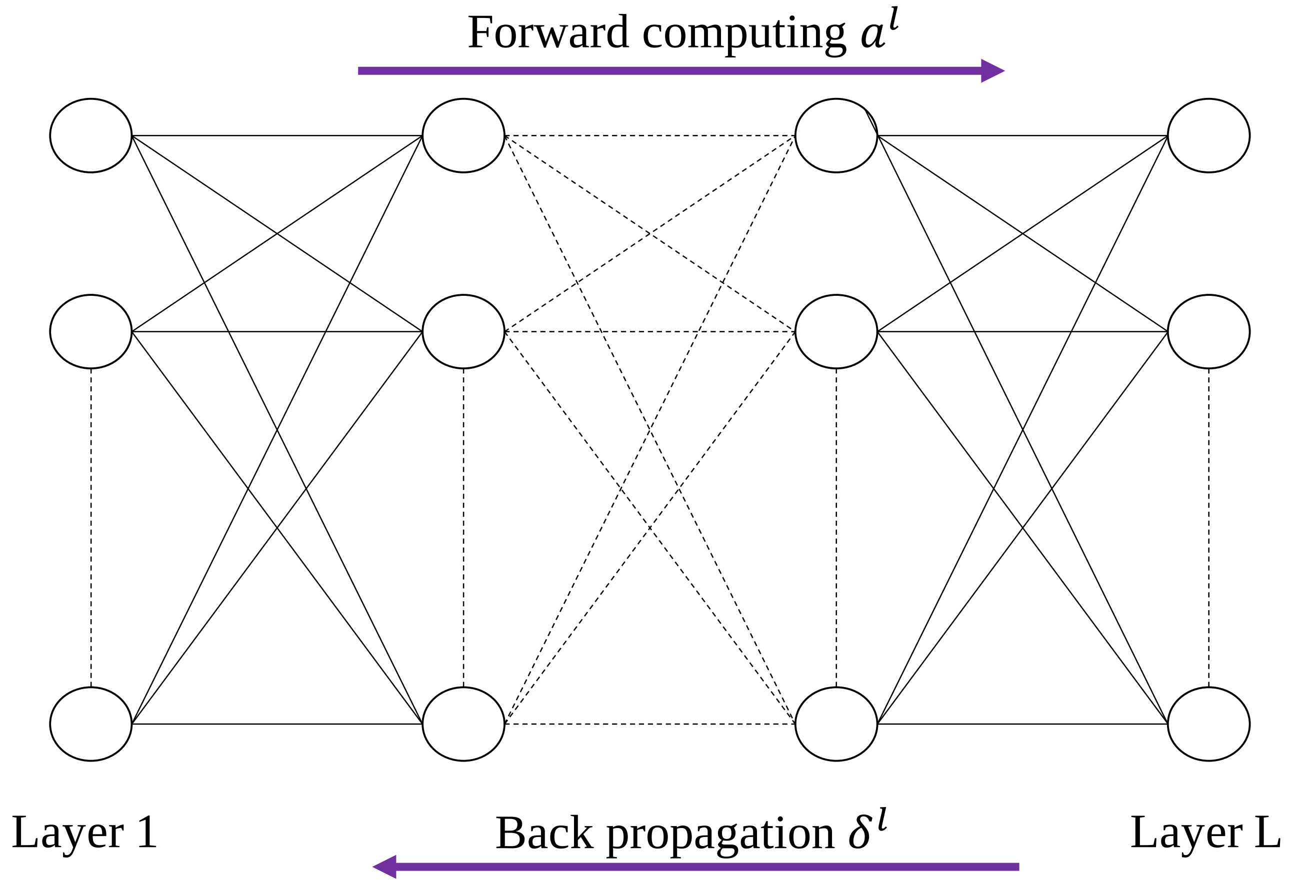


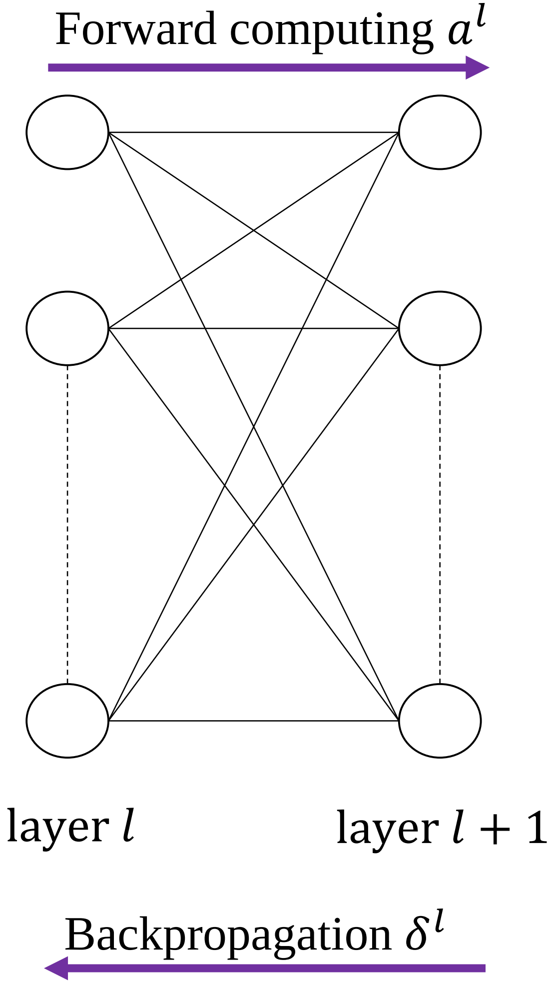




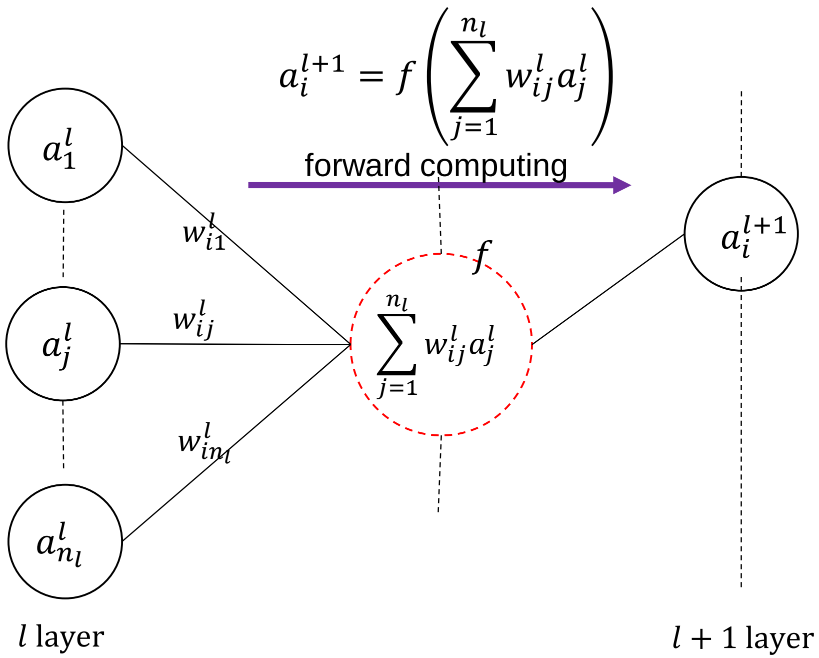
Outline

The BP Algorithm





The BP Algorithm



function $fc(W^l, a^l)$

$for\ i = 1{:}n_{l+1}$

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

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

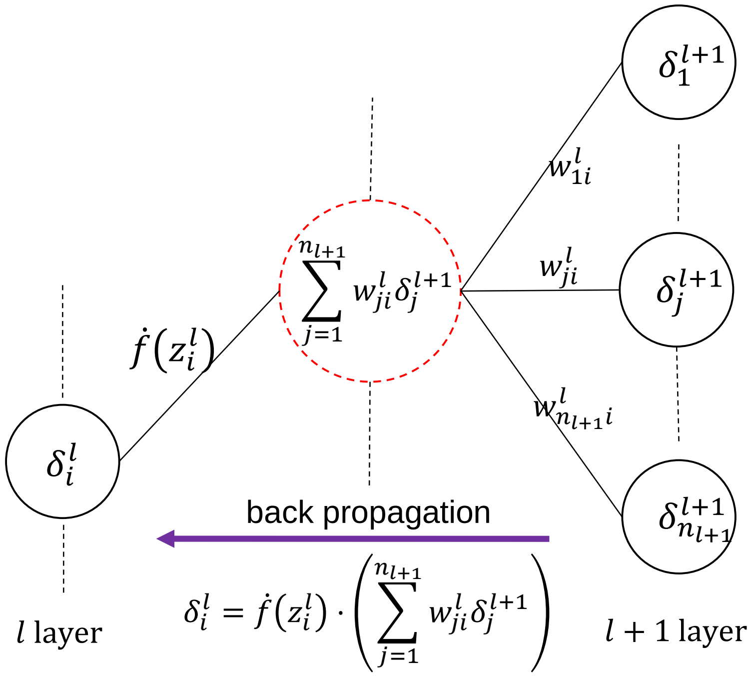
end

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



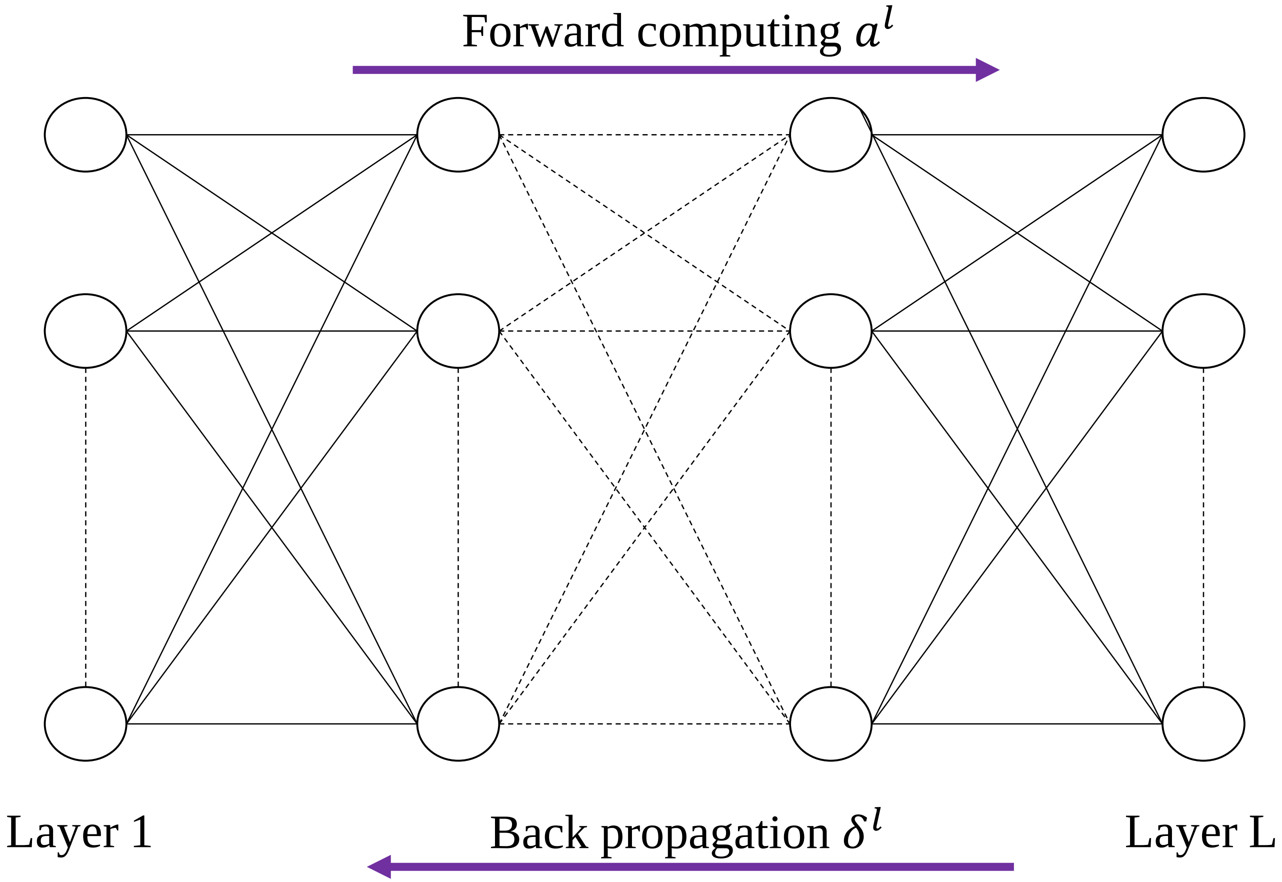
The BP Algorithm

The training data set
$D = \{(x, y)|m\ samples\}$

x: $input\ sample$
y: $target\ output$

There are two ways to train the network.

1. Online training: For each sample $(x, y) \in D$, define a cost function, for example, as
$$J(x,y) = \frac{1}{2}\sum_{j=1}^{n_L}\left(a_j^L - y_j^L\right)^2$$
2. Batch training: Define cost function as
$$J = \frac{1}{m}\sum_{(x,y)\in D} J(x,y)$$



The BP Algorithm

Online BP Algorithm:

Step 1. Input the training data set $D = \{(x, y)\}$

Step 2. Initial each w_{ij}^l, and choose a learning rate α.

Step 3. Choose a sample $(x, y) \in D$, define $J(x, y)$, set $a^1 = x$

for $l = 1: L$

$fc(w^l, a^l)$;

end

$$\delta^L = \frac{\partial J(x,y)}{\partial z^L};$$

for $l = L - 1: 1$

$bc(w^l, \delta^{l+1})$;

end

Step 4. Updating

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

$$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J(x,y)}{\partial w_{ji}^l};$$

Step 5. Return to Step 3 until each w^lconverge.

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

Relationship:

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

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

The BP Algorithm

Batch BP Algorithm:

Step 1. Input the training data set $D = \{(x, y)\}$

Step 2. Initial each w_{ij}^{l}, and choose a learning rate α.

Step 3. For each sample $(x, y) \in D$, set $a^{1} = x$

for $l = 1: L$

$fc(w^{l}, a^{l})$;

end

$$\delta^{L} = \frac{\partial J}{\partial z^{L}};$$

for $l = L - 1: 1$

$bc(w^{l}, \delta^{l+1})$;

end

$$\frac{\partial J}{\partial w_{ji}^{l}} \leftarrow \frac{\partial J}{\partial w_{ji}^{l}} + \delta_{j}^{l+1} \cdot a_{i}^{l};$$

Step 4. Updating

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

Step 5. Return to Step 3 until each w^{l}converge.

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

Relationship:

$$\frac{\partial J}{\partial w_{ji}^{l}} = \delta_{j}^{l+1} \cdot a_{i}^{l}$$

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

Outline

- Brief Review of Neural Networks Structure
- Network Performance: Cost Function
- Steepest Gradient Method
- Backpropagation
- Three Pages to Understand BP
- Only One Page to Understand BP
- The BP Algorithm
- Assignment

Assignment

Assignment: BP algorithms by MATLAB.

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

Batch BP Algorithm:

Step 1. Input the training data set $D = \{(x, y)\}$

Step 2. Initial each w_{ij}^l, and choose a learning rate α.

Step 3. For each sample $(x, y) \in D$, set $a^1 = x$

for $l = 1{:}\, L$
$fc(w^l, a^l)$;
end

$$\delta^L = \frac{\partial J}{\partial z^L};$$

for $l = L - 1{:}\, 1$
$bc(w^l, \delta^{l+1})$;
end

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

Step 4. Updating

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

Step 5. Return to Step 3 until each w^l converge.

Online BP Algorithm:

Step 1. Input the training data set $D = \{(x, y)\}$

Step 2. Initial each w_{ij}^l, and choose a learning rate α.

Step 3. Choose a sample $(x, y) \in D$, define $J(x, y)$, set $a^1 = x$

for $l = 1{:}\, L$
$fc(w^l, a^l)$;
end

$$\delta^L = \frac{\partial J(x, y)}{\partial z^L};$$

for $l = L - 1{:}\, 1$
$bc(w^l, \delta^{l+1})$;
end

Step 4. Updating

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

$$w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J(x, y)}{\partial w_{ji}^l};$$

Step 5. Return to Step 3 until each w^l converge.

Thanks