PyTorch

链式法则

主讲人：龙良曲

Derivative Rules

Rules	Function	Derivative
Multiplication by constant	cf	cf'
Power Rule	x^n	nx^{n-1}
Sum Rule	$f + g$	$f' + g'$
Difference Rule	$f - g$	$f' - g'$
Product Rule	fg	$f g' + f' g$
Quotient Rule	f/g	$(f' g - g' f)/g^2$
Reciprocal Rule	$1/f$	$-f'/f^2$
Chain Rule (as "Composition of Functions")	$f \circ g$	$(f' \circ g) \times g'$
Chain Rule (using $'$)	$f(g(x))$	$f'(g(x))g'(x)$
Chain Rule (using $\frac{d}{dx}$)	$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$	

Basic Rule

- $f + g$

- $f - g$

Product rule

- $(fg)' = f'g + fg'$

- ${x^4}' = (x^2 * x^2)' = 2x * x^2 + x^2 * 2x = 4x^3$

Quotient Rule

- $\frac{f}{g} = \frac{f'g+fg\prime}{g^2}$
- e.g. Softmax

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}$$

$$\frac{\partial p_i}{\partial a_j} = \frac{\partial \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}}{\partial a_j}$$

$$\frac{\partial \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}}{\partial a_j} = \frac{e^{a_i} \sum_{k=1}^{N} e^{a_k} - e^{a_j} e^{a_i}}{\left(\sum_{k=1}^{N} e^{a_k}\right)^2}$$

$$= \frac{e^{a_i} \left(\sum_{k=1}^{N} e^{a_k} - e^{a_j}\right)}{\left(\sum_{k=1}^{N} e^{a_k}\right)^2}$$

$$= \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \times \frac{\left(\sum_{k=1}^{N} e^{a_k} - e^{a_j}\right)}{\sum_{k=1}^{N} e^{a_k}}$$

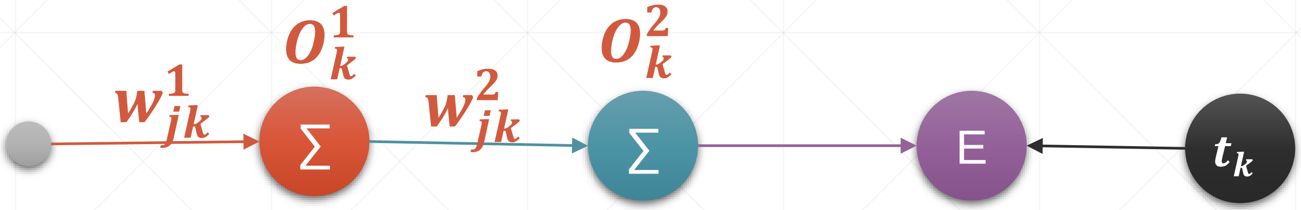
$$= p_i(1 - p_j)$$

Chain rule

- $\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial x}$

- $y_2 = y_1 w_2 + b_2$
- $y_1 = x w_1 + b_1$
- $\frac{\partial y_2}{\partial w_1} = \frac{\partial f(y_1)}{\partial w_1} = \frac{\partial f(y_1)}{\partial y_1}\frac{\partial y_1}{\partial w_1} = w_2 * x$
  - $y_2 = (x w_1 + b_1) * \mathrm{w}_2 + \mathrm{b}_2$

Chain rule



- $\frac{\partial E}{\partial w_{jk}^1} = \frac{\partial E}{\partial O_k^1} \frac{\partial O_k^1}{\partial x} = \frac{\partial E}{\partial O_k^2} \frac{\partial O_k^2}{\partial O_k^1} \frac{\partial O_k^1}{\partial x}$

```
In [63]: x=torch.tensor(1.)
In [65]: w1=torch.tensor(2.,requires_grad=True)
In [66]: b1=torch.tensor(1.)
In [67]: w2=torch.tensor(2.,requires_grad=True)
In [68]: b2=torch.tensor(1.)

In [69]: y1=x*w1+b1
In [70]: y2=y1*w2+b2

In [72]: dy2_dy1=autograd.grad(y2,[y1],retain_graph=True)[0]

In [73]: dy1_dw1=autograd.grad(y1,[w1],retain_graph=True)[0]

In [74]: dy2_dw1=autograd.grad(y2,[w1],retain_graph=True)[0]

In [75]: dy2_dy1*dy1_dw1
Out[75]: tensor(2.)

In [76]: dy2_dw1
Out[76]: tensor(2.)
```

下一课时

MLP反向传播

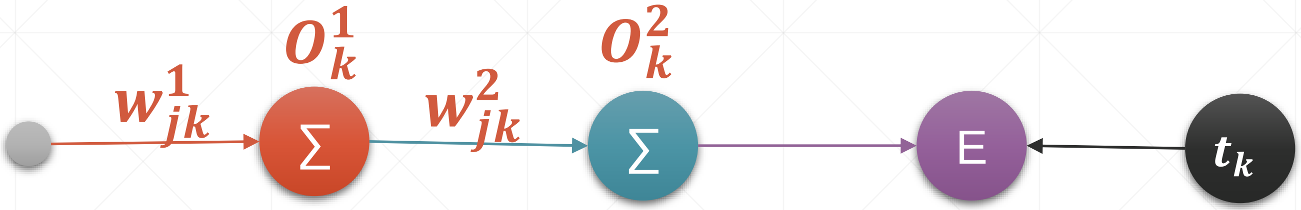
Thank You.

PyTorch

MLP反向传播

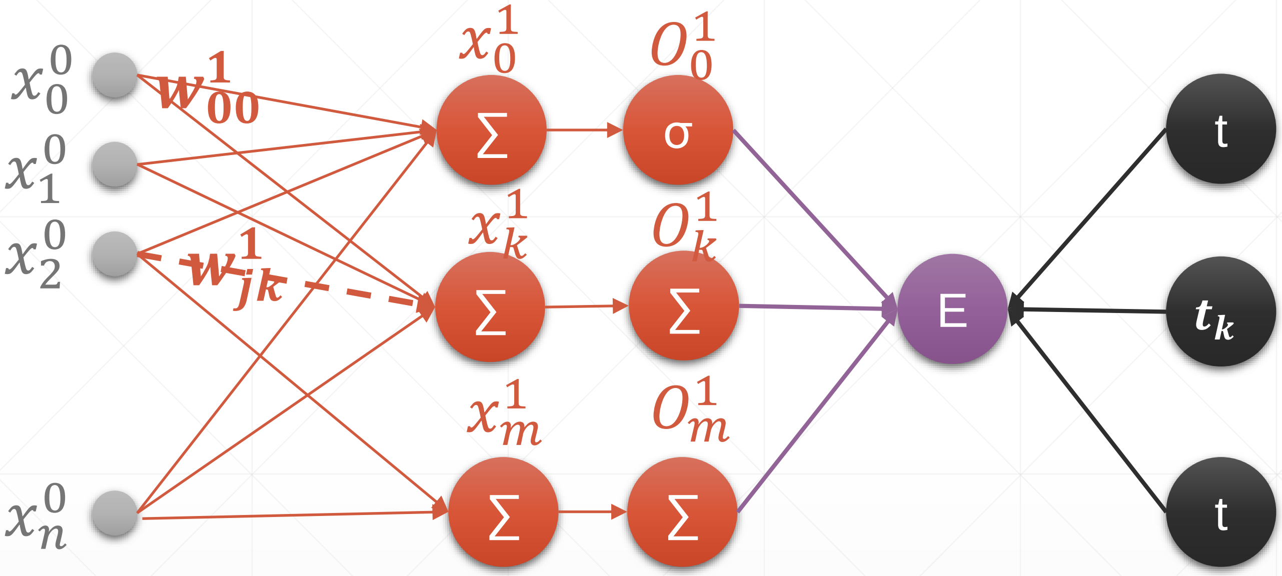
主讲人：龙良曲

Chain rule



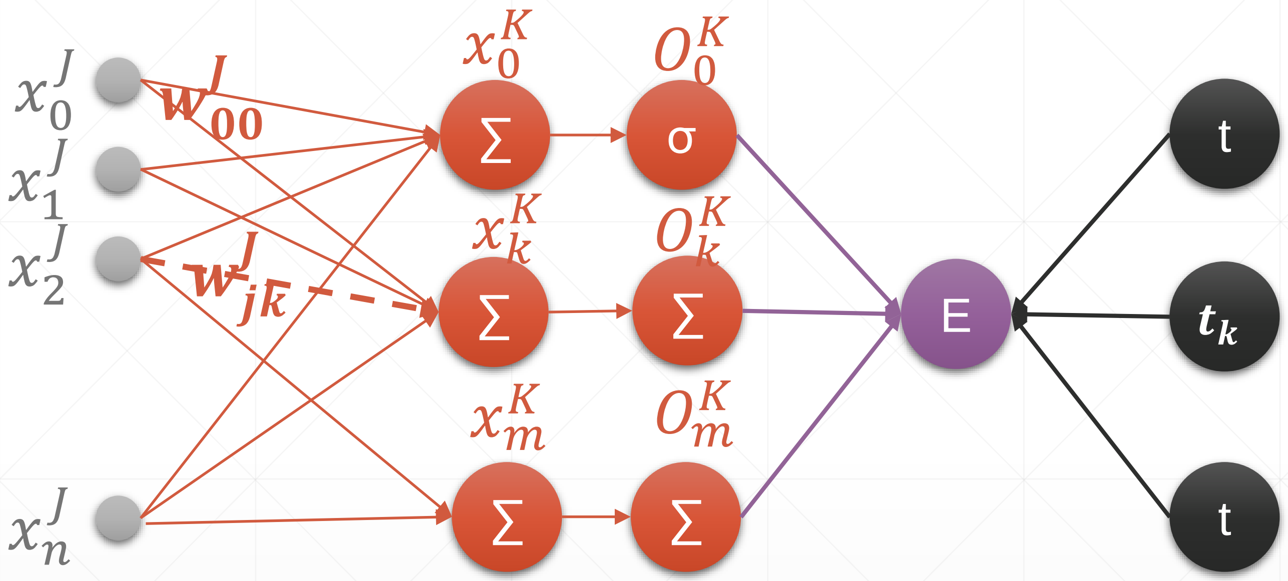
- $\frac{\partial E}{\partial w_{jk}^1} = \frac{\partial E}{\partial o_k^1} \frac{\partial o_k^1}{\partial x} = \frac{\partial E}{\partial o_k^2} \frac{\partial o_k^2}{\partial o_k^1} \frac{\partial o_k^1}{\partial x}$

Multi-output Perceptron

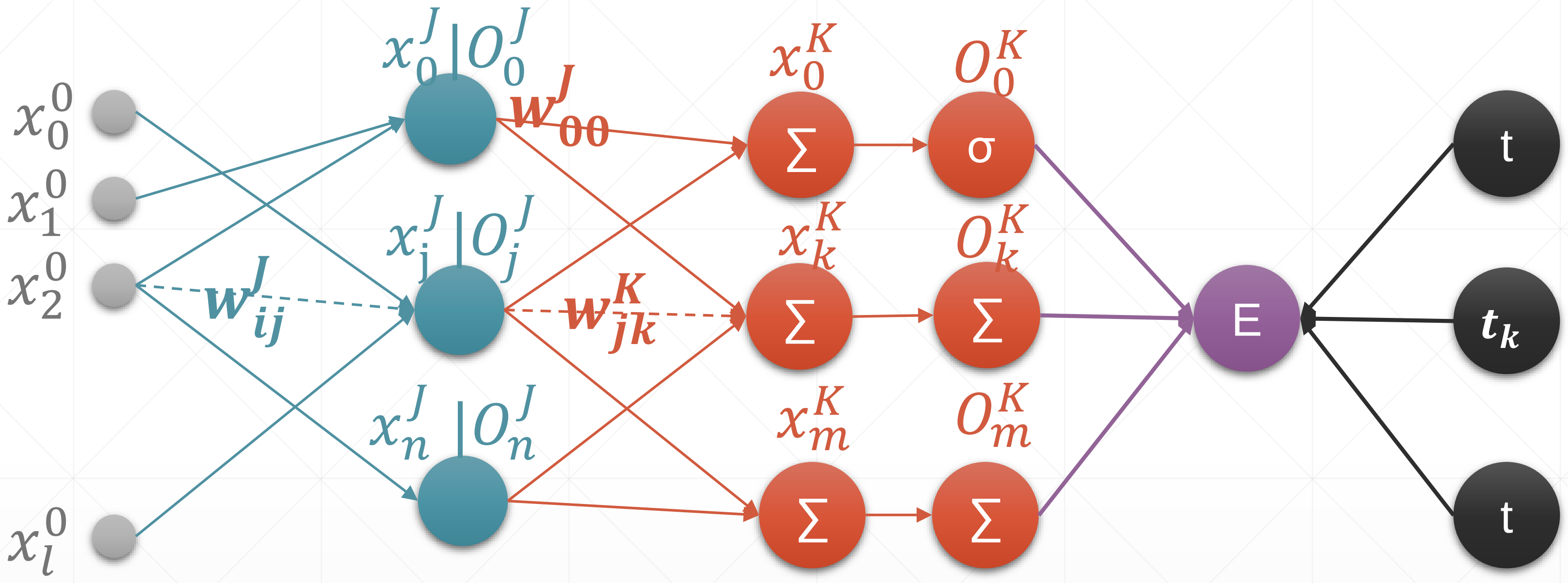


$$\frac{\partial E}{\partial w_{jk}} = (O_k - t_k) O_k (1 - O_k)\, x_j^0$$

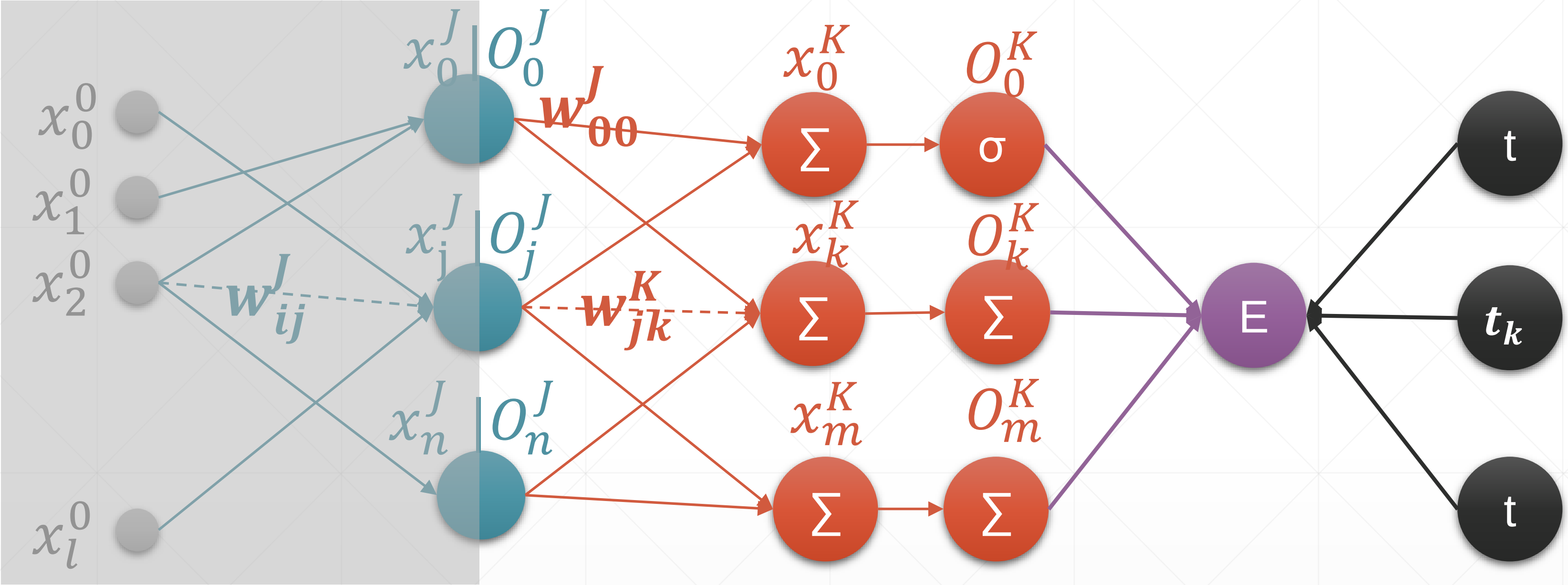
Multi-Layer Perceptron



Multi-Layer Perceptron



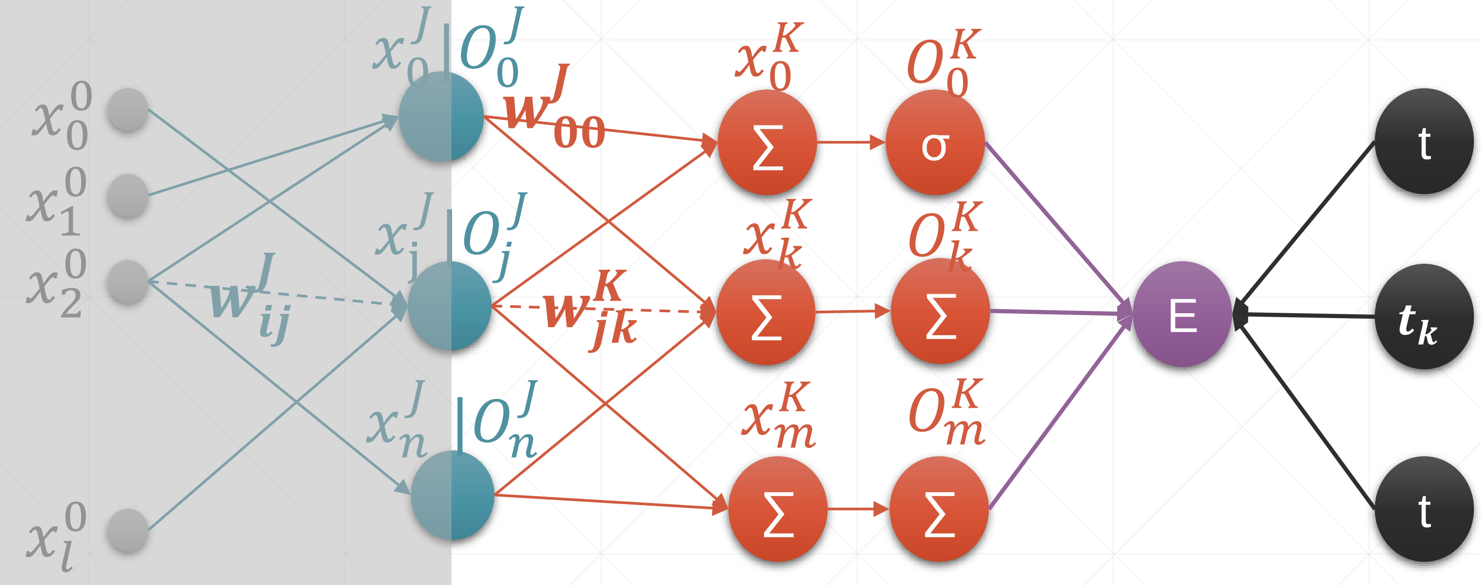
Multi-Layer Perceptron



$$\frac{\partial E}{\partial w_{jk}} = (O_k - t_k) O_k (1 - O_k)\, x_j^0$$

$$\frac{\partial E}{\partial w_{jk}} = (O_k - t_k) O_k (1 - O_k)\, O_j^J$$

Multi-Layer Perceptron



$$\frac{\partial E}{\partial w_{jk}} = (O_k - t_k) O_k (1 - O_k)\ O_j^J$$

$$\frac{\partial E}{\partial w_{jk}} = \delta_k^K \qquad O_j^J$$

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \frac{1}{2} \sum_{k \in K} (\mathcal{O}_k - t_k)^2$$

$$\frac{\partial E}{\partial W_{ij}} = \sum_{k \in K} (\mathcal{O}_k - t_k) \frac{\partial}{\partial W_{ij}} \mathcal{O}_k$$

$$\frac{\partial E}{\partial W_{ij}} = \sum_{k \in K} (\mathcal{O}_k - t_k) \frac{\partial}{\partial W_{ij}} \sigma(x_k)$$

$$\frac{\partial E}{\partial W_{ij}} = \sum_{k \in K} (\mathcal{O}_k - t_k) \sigma(x_k)(1 - \sigma(x_k)) \frac{\partial x_k}{\partial W_{ij}}$$

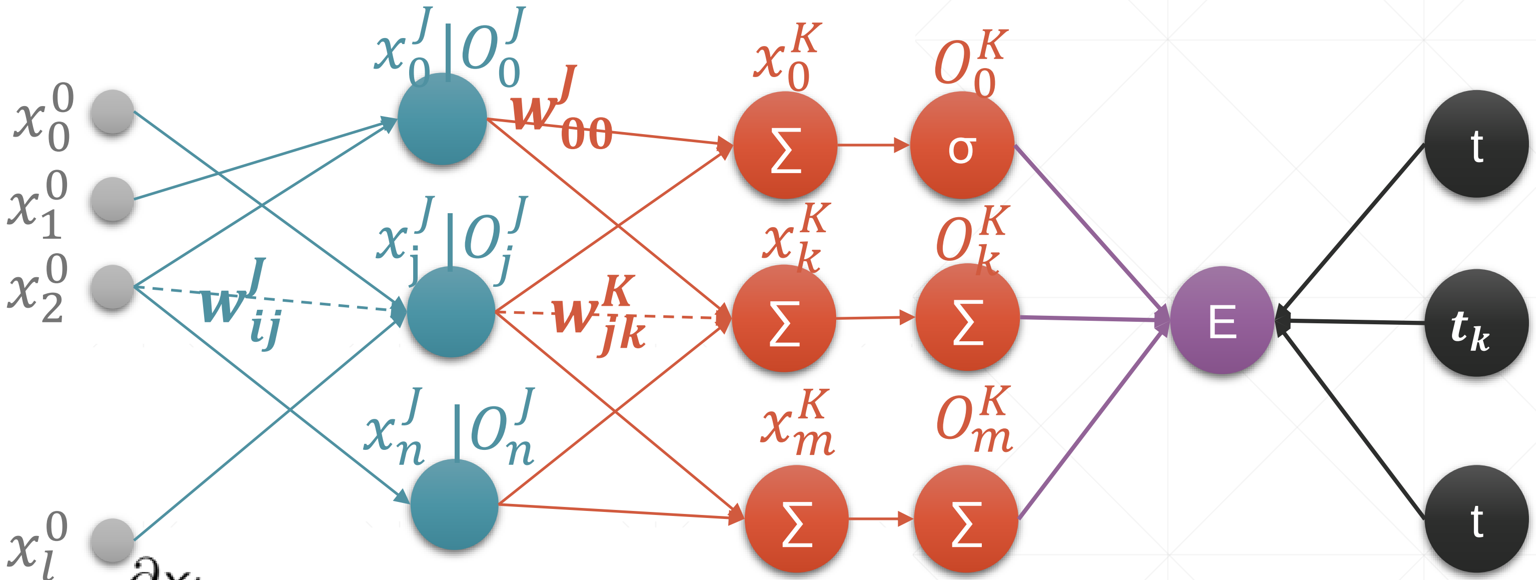
$$\frac{\partial E}{\partial W_{ij}} = \sum_{k \in K} (\mathcal{O}_k - t_k) \mathcal{O}_k (1 - \mathcal{O}_k) \frac{\partial x_k}{\partial \mathcal{O}_j} \cdot \frac{\partial \mathcal{O}_j}{\partial W_{ij}}$$

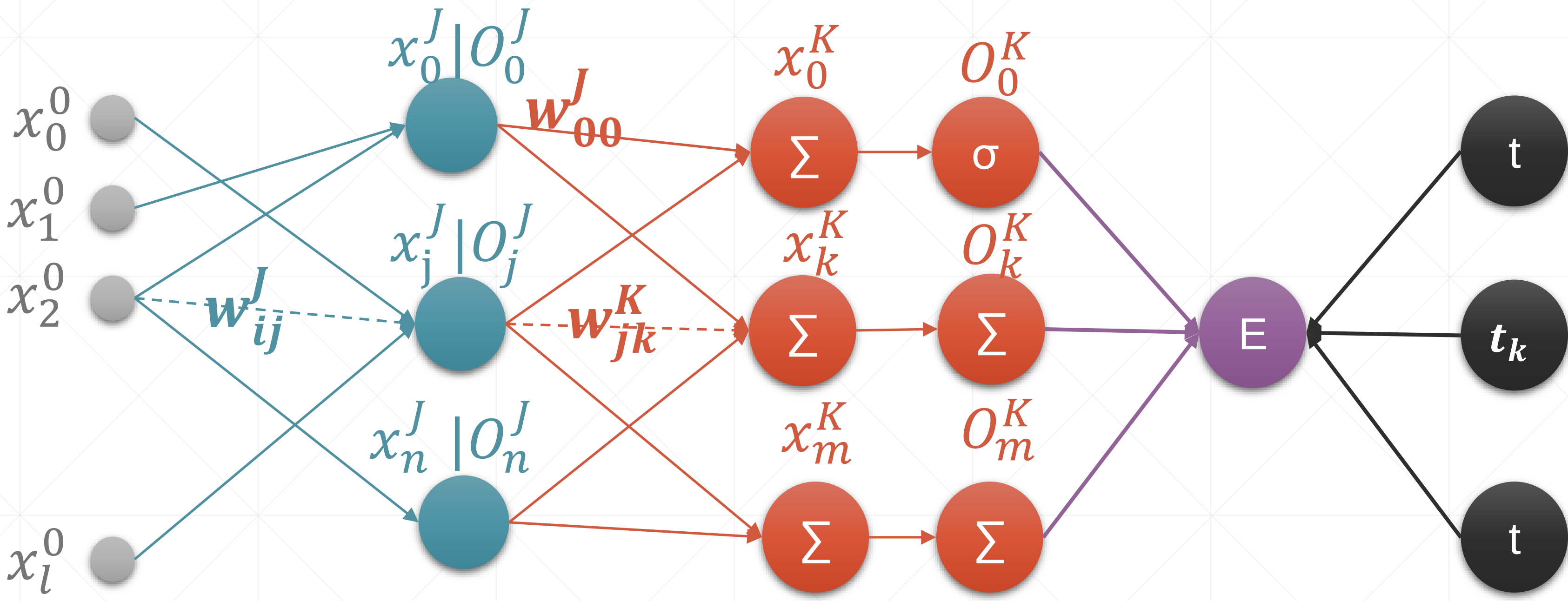
$$\frac{\partial E}{\partial W_{ij}} = \sum_{k \in K} (\mathcal{O}_k - t_k) \mathcal{O}_k (1 - \mathcal{O}_k) W_{jk} \frac{\partial \mathcal{O}_j}{\partial W_{ij}}$$

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial \mathcal{O}_j}{\partial W_{ij}} \sum_{k \in K} (\mathcal{O}_k - t_k) \mathcal{O}_k (1 - \mathcal{O}_k) W_{jk}$$

$$\boxed{\frac{\partial E}{\partial W_{ij}} = \mathcal{O}_j (1 - \mathcal{O}_j) \mathcal{O}_i \sum_{k \in K} (\mathcal{O}_k - t_k) \mathcal{O}_k (1 - \mathcal{O}_k) W_{jk}}$$

$$\frac{\partial E}{\partial W_{ij}} = \mathcal{O}_j (1 - \mathcal{O}_j) \frac{\partial x_j}{\partial W_{ij}} \sum_{k \in K} (\mathcal{O}_k - t_k) \mathcal{O}_k (1 - \mathcal{O}_k) W_{jk}$$



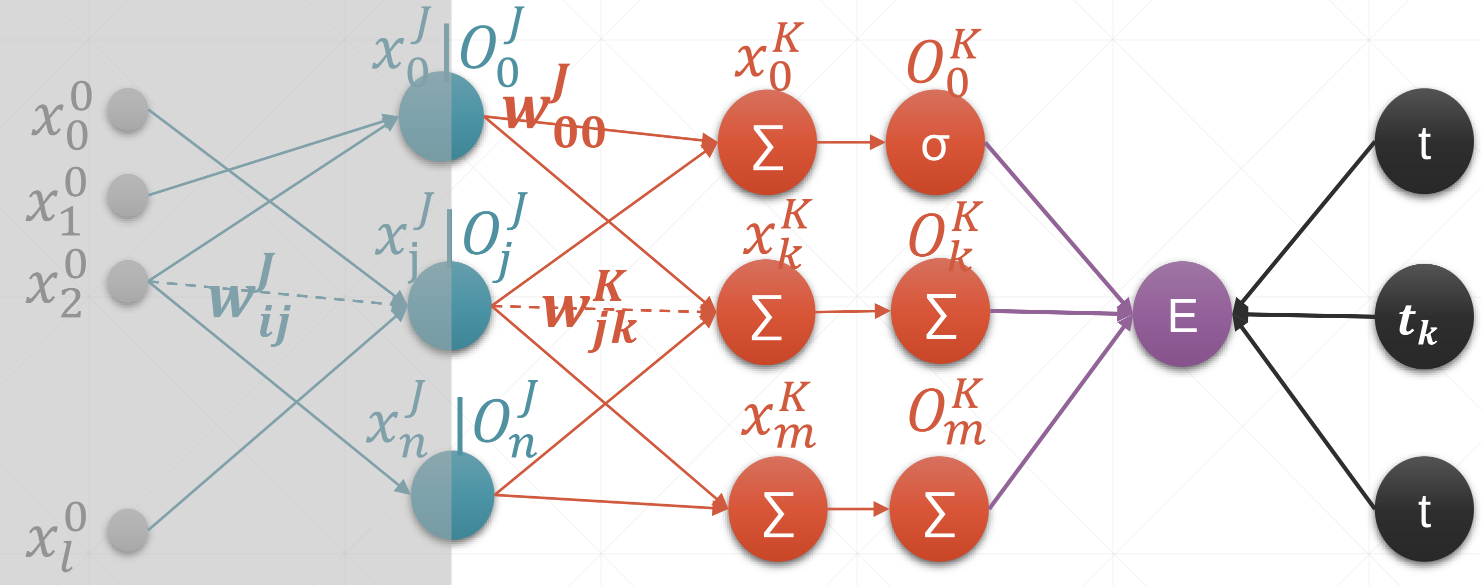


$$\frac{\partial E}{\partial W_{ij}} = \mathcal{O}_j(1-\mathcal{O}_j)\mathcal{O}_i \sum_{k\in K}(\mathcal{O}_k - t_k)\mathcal{O}_k(1-\mathcal{O}_k)W_{jk}$$

$$\frac{\partial E}{\partial W_{ij}} = \mathcal{O}_i\mathcal{O}_j(1-\mathcal{O}_j) \sum_{k\in K}\delta_k W_{jk}$$

$$\frac{\partial E}{\partial w_{jk}} = (O_k - t_k) O_k (1 - O_k)\ O_j^J$$

$$\frac{\partial E}{\partial w_{jk}} = \delta_k^K\ O_j^J$$



$$\frac{\partial E}{\partial W_{ij}} = \mathcal{O}_j(1 - \mathcal{O}_j)\mathcal{O}_i \sum_{k \in K} (\mathcal{O}_k - t_k)\mathcal{O}_k(1 - \mathcal{O}_k) W_{jk}$$

$$\frac{\partial E}{\partial W_{ij}} = \mathcal{O}_i \mathcal{O}_j(1 - \mathcal{O}_j) \sum_{k \in K} \delta_k W_{jk}$$

For an output layer node $k \in K$

$$\frac{\partial E}{\partial W_{jk}} = \mathcal{O}_j \delta_k$$

where

$$\delta_k = \mathcal{O}_k(1 - \mathcal{O}_k)(\mathcal{O}_k - t_k)$$

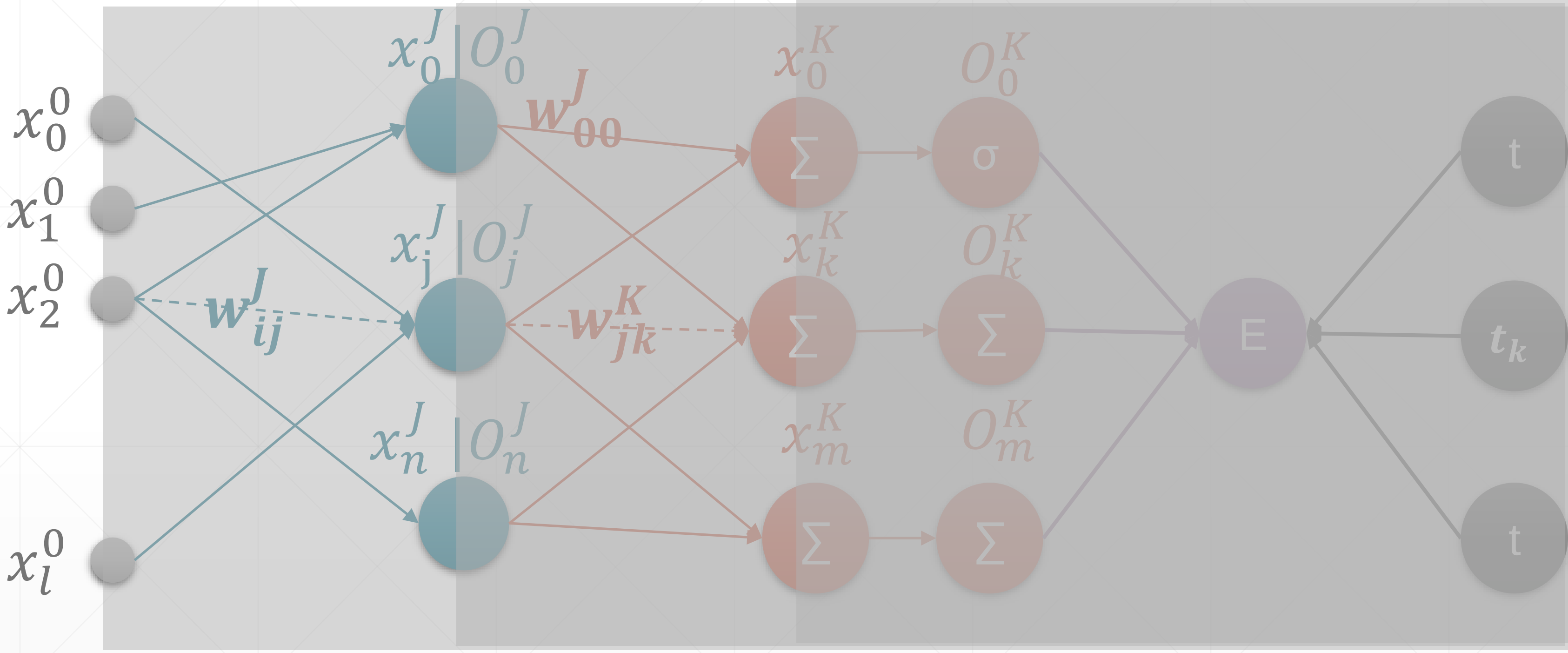
For a hidden layer node $j \in J$

$$\frac{\partial E}{\partial W_{ij}} = \mathcal{O}_i \delta_j$$

where

$$\delta_j = \mathcal{O}_j(1 - \mathcal{O}_j) \sum_{k \in K} \delta_k W_{jk}$$

- δ_k^K
- $\frac{\partial E}{\partial w_{jk}}$
- δ_j^J
- $\frac{\partial E}{\partial w_{ij}}$
- δ_i^I
- $\frac{\partial E}{\partial w_{ni}}$



GAME OVER

下一课时

2D函数优化

Thank You.