

# RECURRENT NEURAL NETWORKS

{Grady Kurpasi}

{SSIE 616}

{Prof H. Lewis}

{<https://github.com/GradyKurpasi/RNN-Tutorial>}



# Multilayer Perceptrons

$$x, y$$

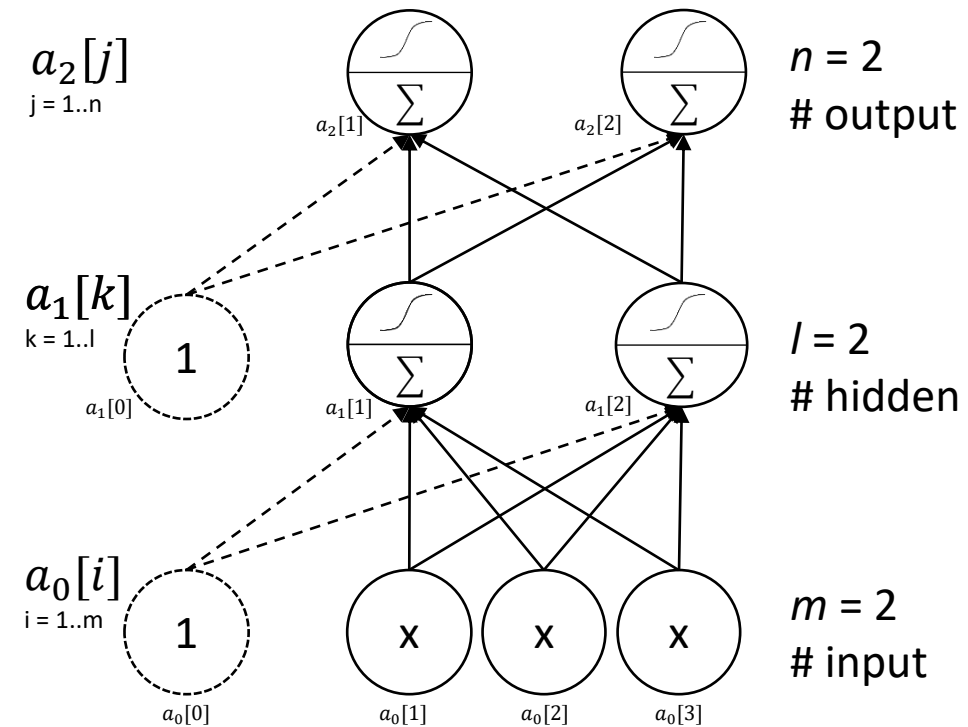
$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500
0.1000	-5.0000	3.0000	0.1221	0.0964
6.0000	-5.5420	4.8970	0.1061	0.0702
4.0000	8.0000	9.0000	0.0996	0.0641
12.0000	-2.0000	0.0063	0.1110	0.0732
6.0000	-5.5000	4.8970	0.1060	0.0701

$$w_1[i, k]$$

i	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$$w_2[k, j]$$

k	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1



# Multilayer Perceptrons

$$x, y$$

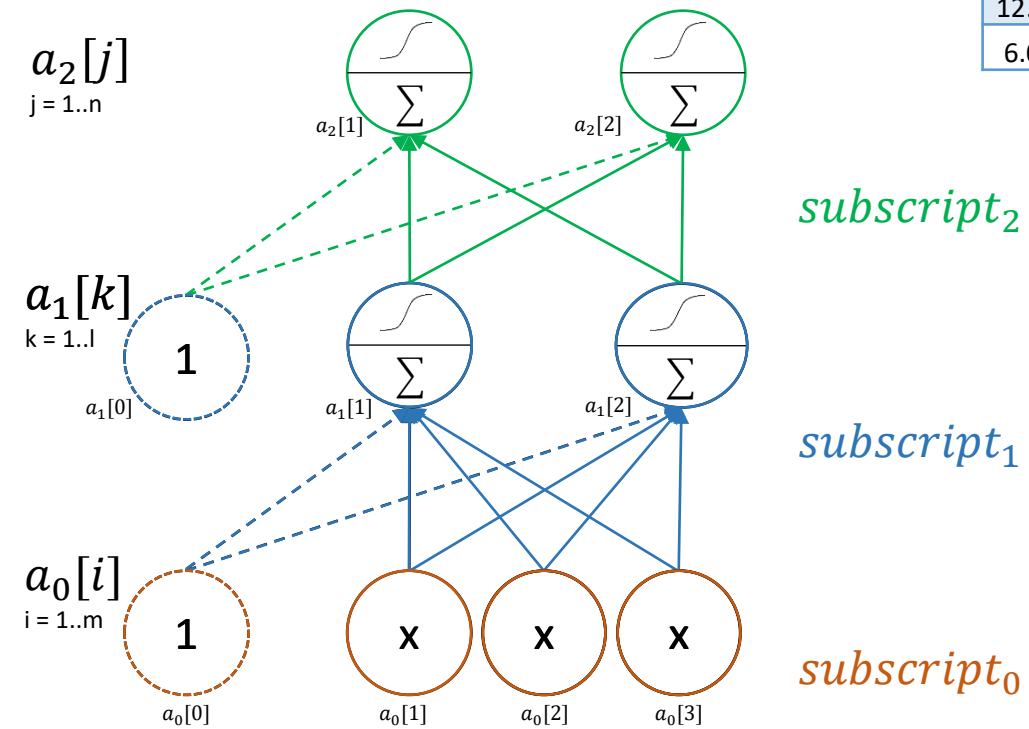
$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500
0.1000	-5.0000	3.0000	0.1221	0.0964
6.0000	-5.5420	4.8970	0.1061	0.0702
4.0000	8.0000	9.0000	0.0996	0.0641
12.0000	-2.0000	0.0063	0.1110	0.0732
6.0000	-5.5000	4.8970	0.1060	0.0701

$$w_1[i, k]$$

i	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$$w_2[k, j]$$

k	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

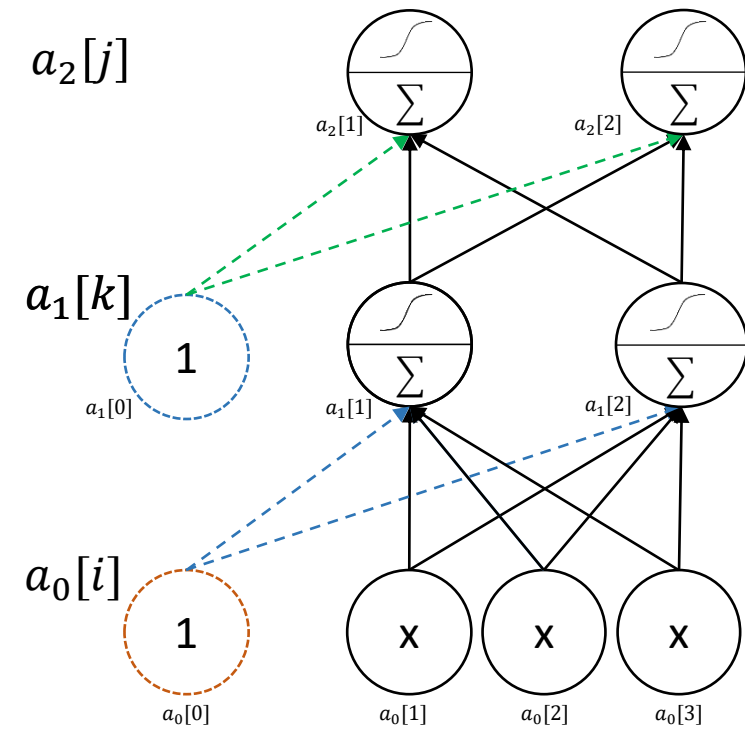


# Multilayer Perceptrons

$x, y$				
$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500
0.1000	-5.0000	3.0000	0.1221	0.0964
6.0000	-5.5420	4.8970	0.1061	0.0702
4.0000	8.0000	9.0000	0.0996	0.0641
12.0000	-2.0000	0.0063	0.1110	0.0732
6.0000	-5.5000	4.8970	0.1060	0.0701

$w_1[i, k]$		
$i$	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$w_2[k, j]$		
$k$	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

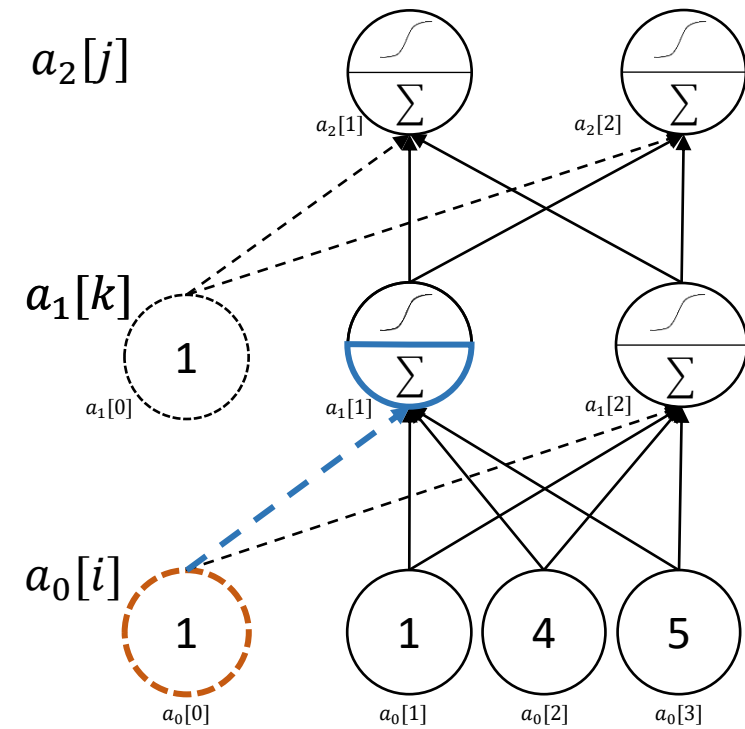


# Forward Propagation

$x, y$				
$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i, k]$		
$i$	$w[i, 1]$	$w[i, 2]$
0	<b>0.5</b>	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$w_2[k, j]$		
$k$	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1



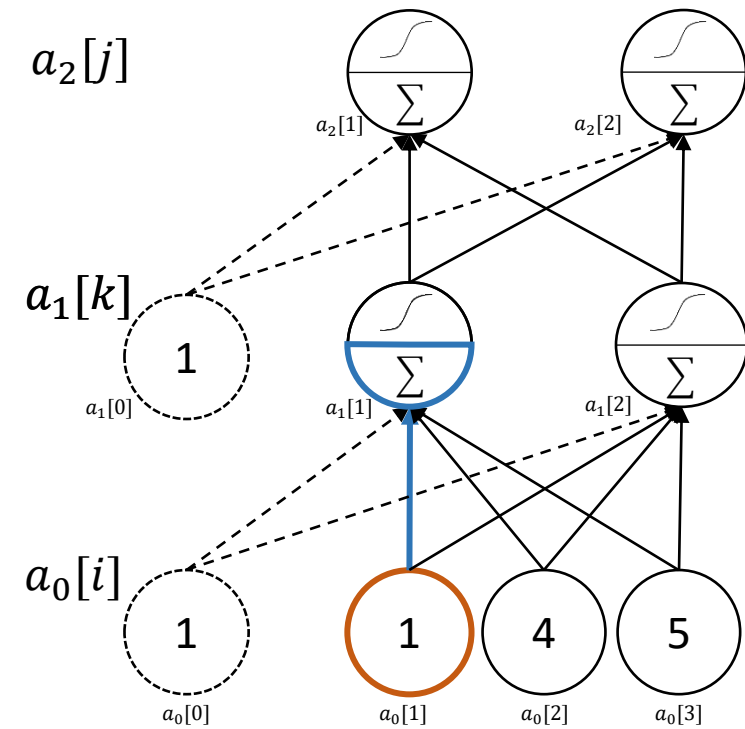
$$s_1[k] = \sum_{i=0}^m w_1[i, k] * a_0[i] \quad \text{for } k = 1..l$$
$$s_1[1] = w_1[0, 1] * a_0[0] = .05 * 1$$

# Forward Propagation

$x, y$				
$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i, k]$		
$i$	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	<b>0.1</b>	0.2
2	0.3	0.4
3	0.5	0.6

$w_2[k, j]$		
$k$	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1



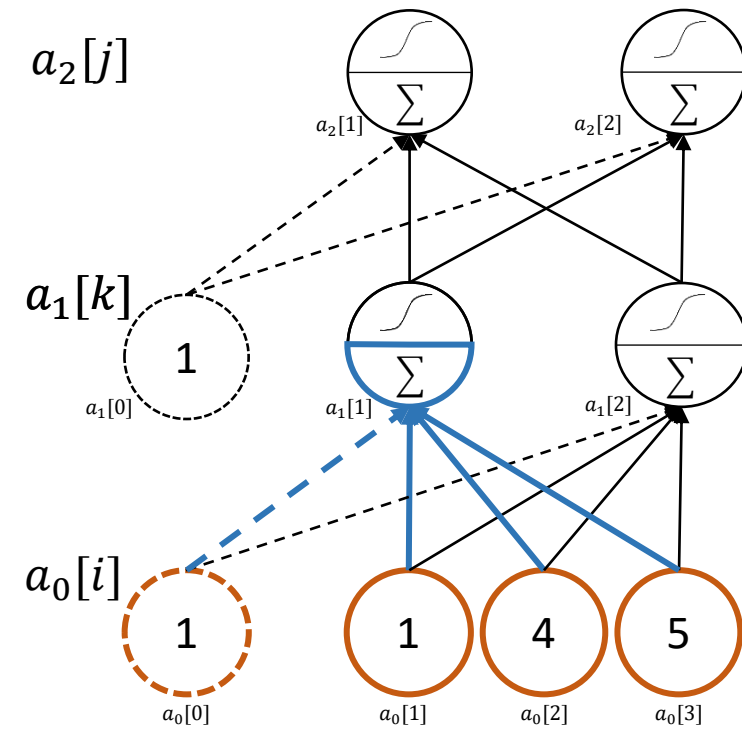
$$s_1[k] = \sum_{i=0}^3 w_1[i, k] * a_0[i] \quad \text{for } k = 1, 2$$
$$s_1[1] = w_1[0, 1] * a_0[0] = .05 * 1$$
$$w_1[1, 1] * a_0[1] = .1 * 1$$

# Forward Propagation

$x, y$				
$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i, k]$		
$i$	$w[i, 1]$	$w[i, 2]$
0	<b>0.5</b>	0.5
1	<b>0.1</b>	0.2
2	<b>0.3</b>	0.4
3	<b>0.5</b>	0.6

$w_2[k, j]$		
$k$	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1



$$s_1[k] = \sum_{i=0}^3 w_1[i, k] * a_0[i] \quad \text{for } k = 1, 2$$
$$s_1[1] = w_1[0,1] * a_0[0] = .05 * 1 +$$
$$w_1[1,1] * a_0[1] = .1 * 1 +$$
$$w_1[2,1] * a_0[2] = .3 * 4 +$$
$$w_1[3,1] * a_0[3] = .5 * 5$$
$$= 4.3$$

# Forward Propagation

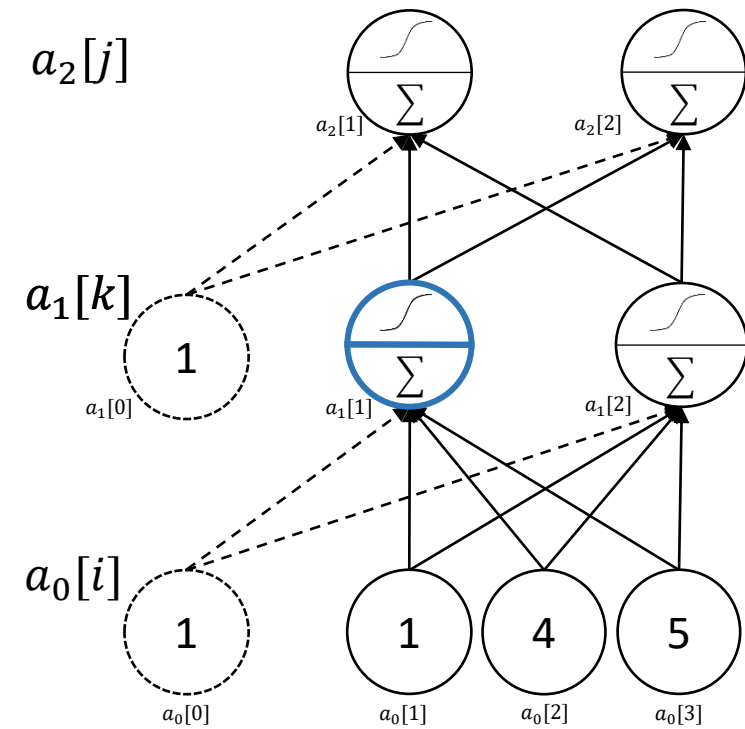
$x, y$				
$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i, k]$		
$i$	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$w_2[k, j]$		
$k$	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

$$a_1[k] = \frac{1}{1 + e^{-s_1[k]}} \quad \text{for } k = 1, 2$$

$$a_1[1] = \frac{1}{1 + e^{-4.3}} = .9866$$



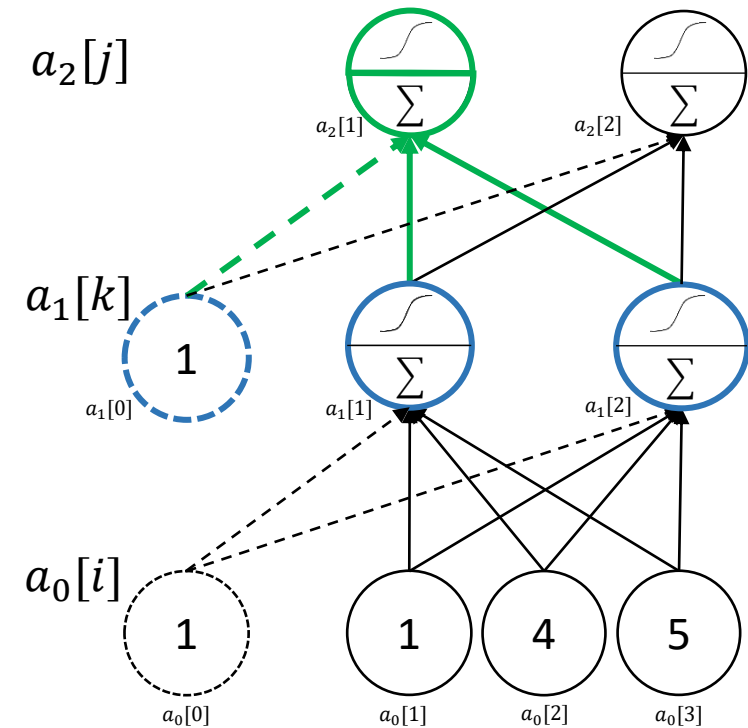


# Forward Propagation

$x, y$				
$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i, k]$		
$i$	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$w_2[k, j]$		
$k$	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1



$$s_2[j] = \sum_{k=0}^l w_2[k, j] * a_1[k] \quad \text{for } j = 1..m$$

$$a_2[j] = \frac{1}{1 + e^{-s_2[j]}} \quad \text{for } j = 1, 2$$

$$s_2[1] = w_2[0, 2] * a_1[0] = .5 * 1$$

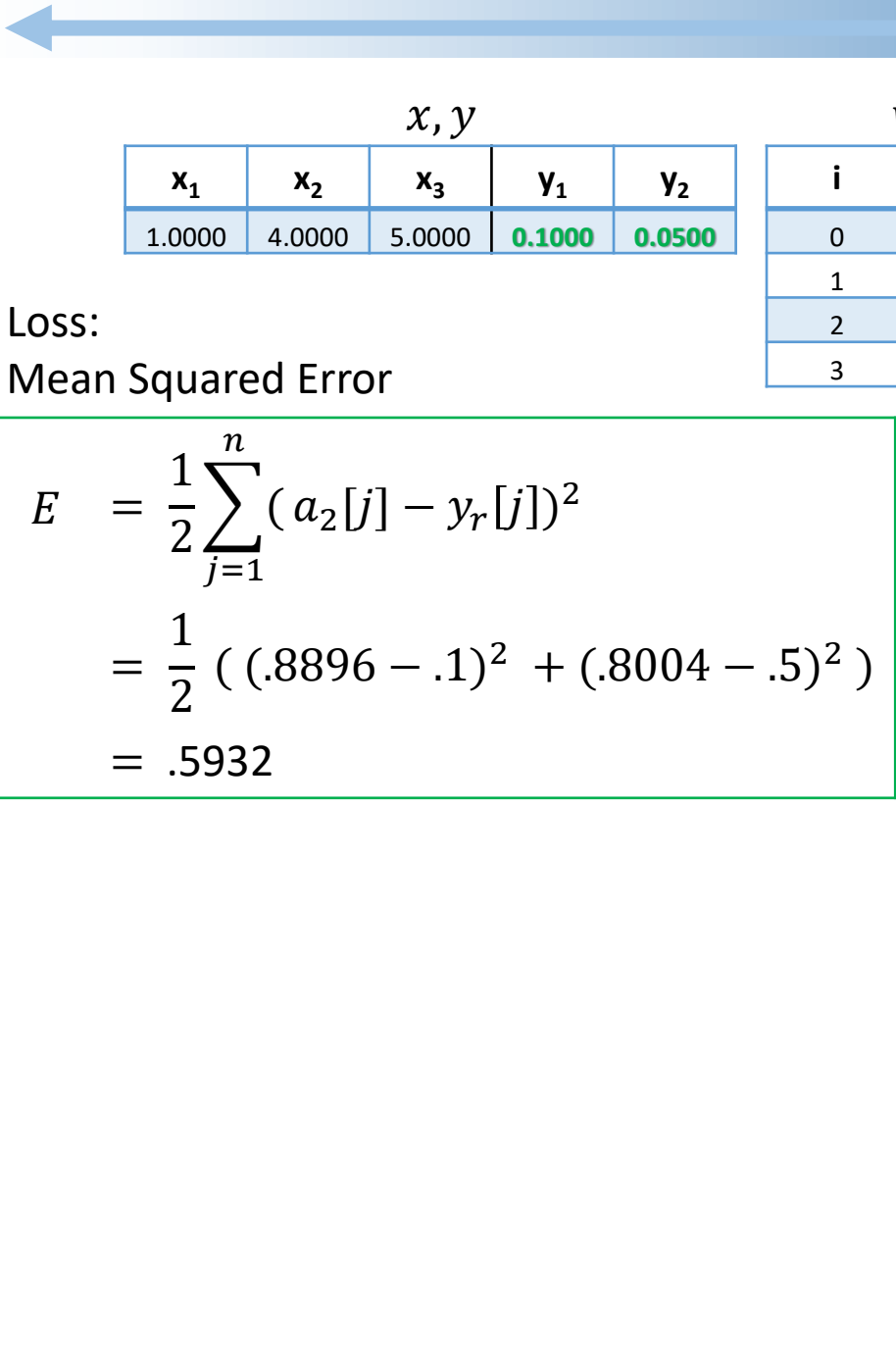
$$w_2[1, 2] * a_1[1] = .7 * .9866$$

$$w_2[2, 2] * a_1[2] = .9 * .9950$$

$$= 2.0862$$

$$a_2[1] = \frac{1}{1 + e^{-2.0862}} = .8896 = \hat{y}$$

←



$w_1[i, k]$		
i	w[i,1]	w[i,2]
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

Loss:	2
Mean Squared Error	3

---


$$\begin{aligned}
 E &= \frac{1}{2} \sum_{j=1}^n (a_2[j] - y_r[j])^2 \\
 &= \frac{1}{2} ( (.8896 - .1)^2 + (.8004 - .5)^2 ) \\
 &= .5932
 \end{aligned}$$

$$\begin{aligned} E &= \frac{1}{2} \sum_{j=1}^n (a_2[j] - y_r[j])^2 \\ &= \frac{1}{2} ( (.8896 - .1)^2 + (.8004 - .5)^2 ) \\ &= .5932 \end{aligned}$$

# Backpropagation



$x, y$				
$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i, k]$		
$i$	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$w_2[k, j]$		
$k$	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

Partial Derivative of Error with respect to  $Weight_2[0,1]$

Functional Dependencies

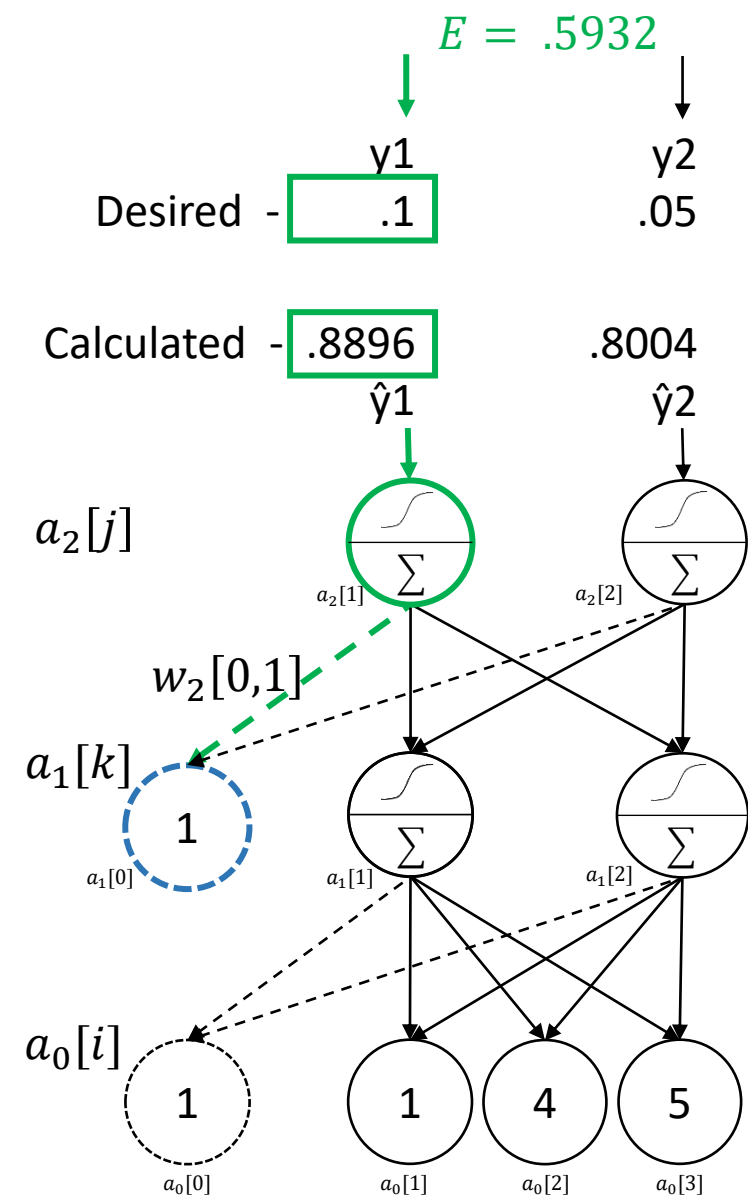
$$E = \frac{1}{2} (\hat{y} - y_1)^2 \rightarrow E(\hat{y})$$

$$\hat{y} = \sigma(s_2) \rightarrow f(s_2)$$

$$s_2 = \sum a_1[k] * w_2[0,1] + .6906 + .8955 \rightarrow g(w_2)$$

Chain Rule

$$E(f(g(w_2)))$$

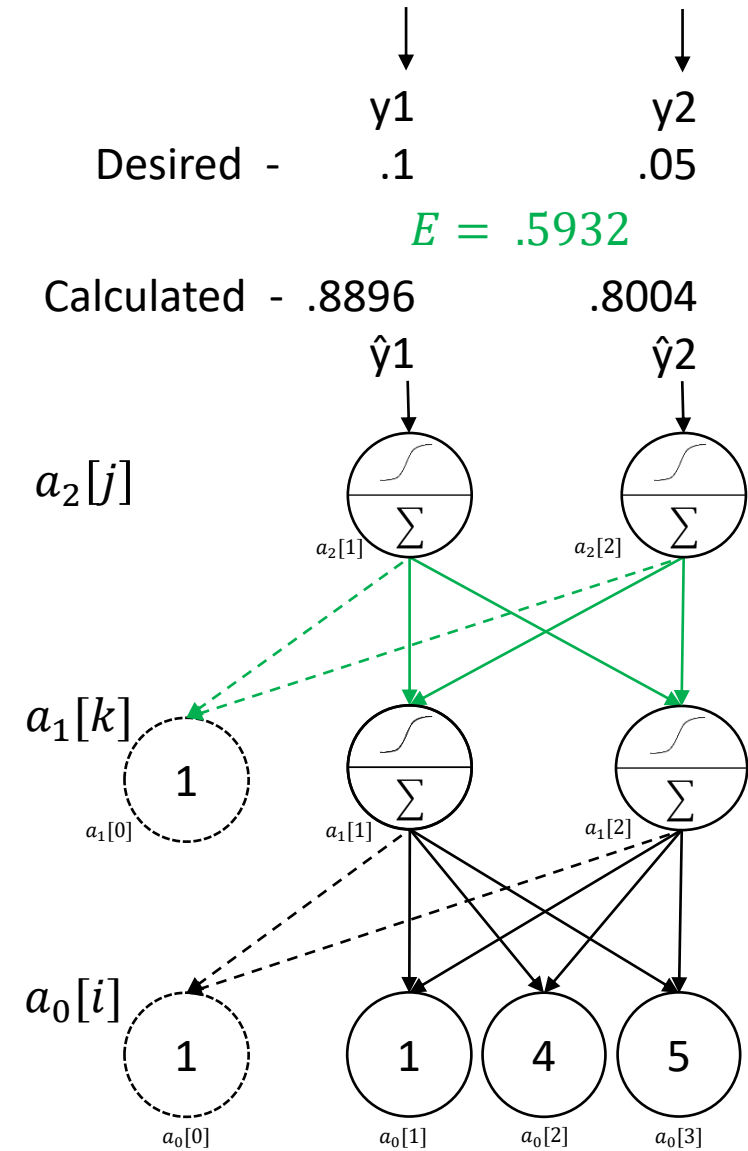


# Backpropagation

Desired -  $y_1 = .1$   $y_2 = .05$

$E = .5932$

Calculated -  $\hat{y}_1 = .8896$   $\hat{y}_2 = .8004$



$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$i$	$w[i,1]$	$w[i,2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$k$	$w[k,1]$	$w[k,2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

$$\begin{aligned} \frac{\partial E}{\partial w_2[k, j]} &= \frac{\partial E}{\partial a_2[j]} * \frac{\partial a_2[j]}{\partial s_2[j]} * \frac{\partial s_2[j]}{\partial w_2[k, j]} \\ &= \{\hat{y}_j - y_k\} * \{a_2[j](1 - a_2[j])\} * \{a_1[k]\} \end{aligned}$$

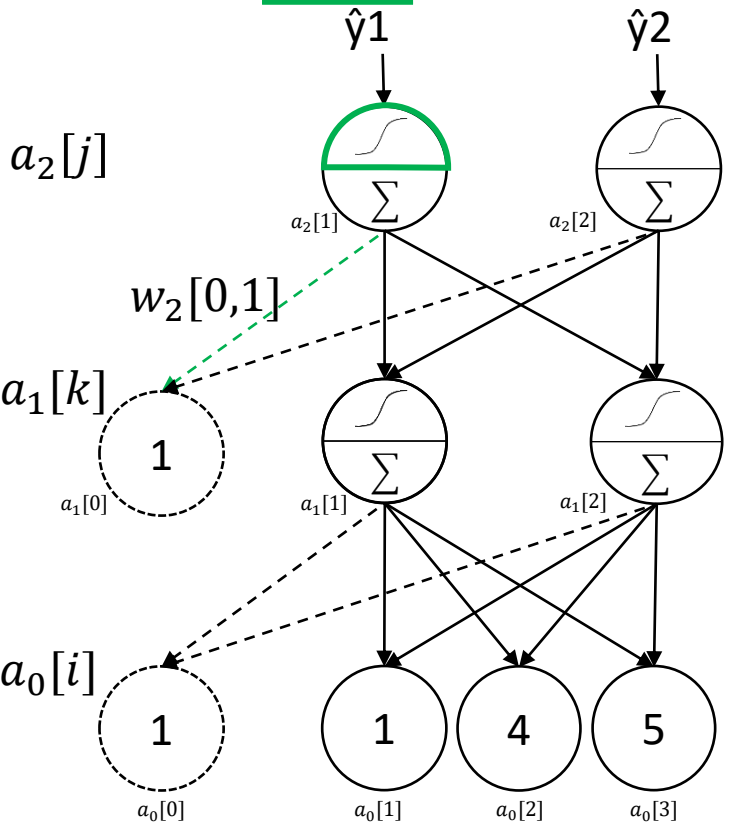
# Backpropagation

Desired -  $y_1$  .1

$y_2$   
.05

Calculated - .8896

.8004



$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$i$	$w[i,1]$	$w[i,2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

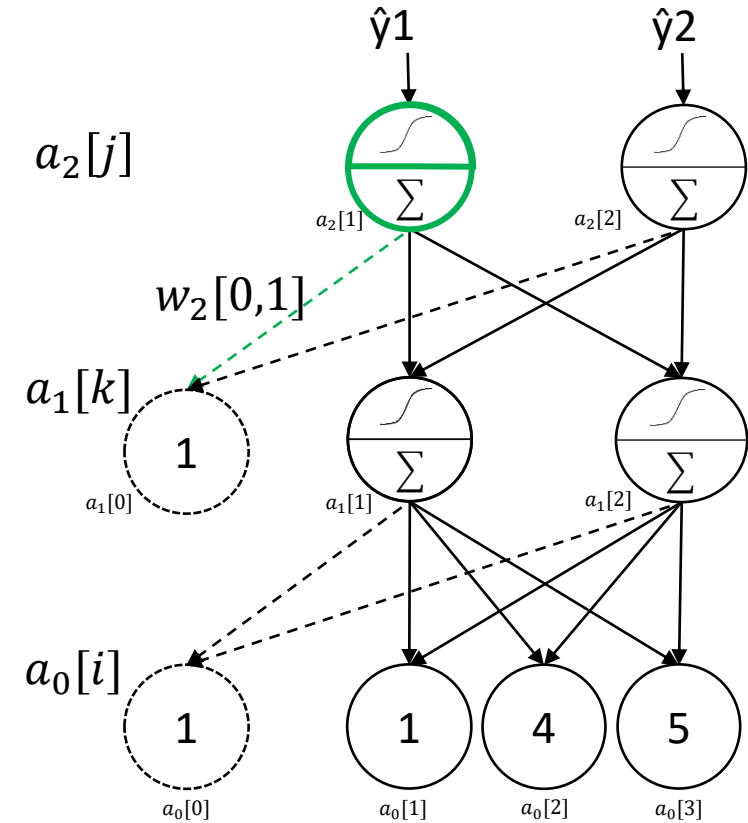
$k$	$w[k,1]$	$w[k,2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

$$\begin{aligned}
 \frac{\partial E}{\partial w_2[0,1]} &= \frac{\partial E}{\partial a_2[1]} * \frac{\partial a_2[1]}{\partial s_2[1]} * \frac{\partial s_2[1]}{\partial w_2[0,1]} \\
 &= \{\hat{y}_j - y_k\} * \{a_2[1](1 - a_2[1])\} * \{a_1[0]\} \\
 &= .8896 - .1
 \end{aligned}$$

# Backpropagation

Desired -  $y_1 = .1$   $y_2 = .05$

Calculated -  $\hat{y}_1 = .8896$   $\hat{y}_2 = .8004$



$$x, y$$

$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$$w_1[i, k]$$

i	$w[i,1]$	$w[i,2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$$w_2[k, j]$$

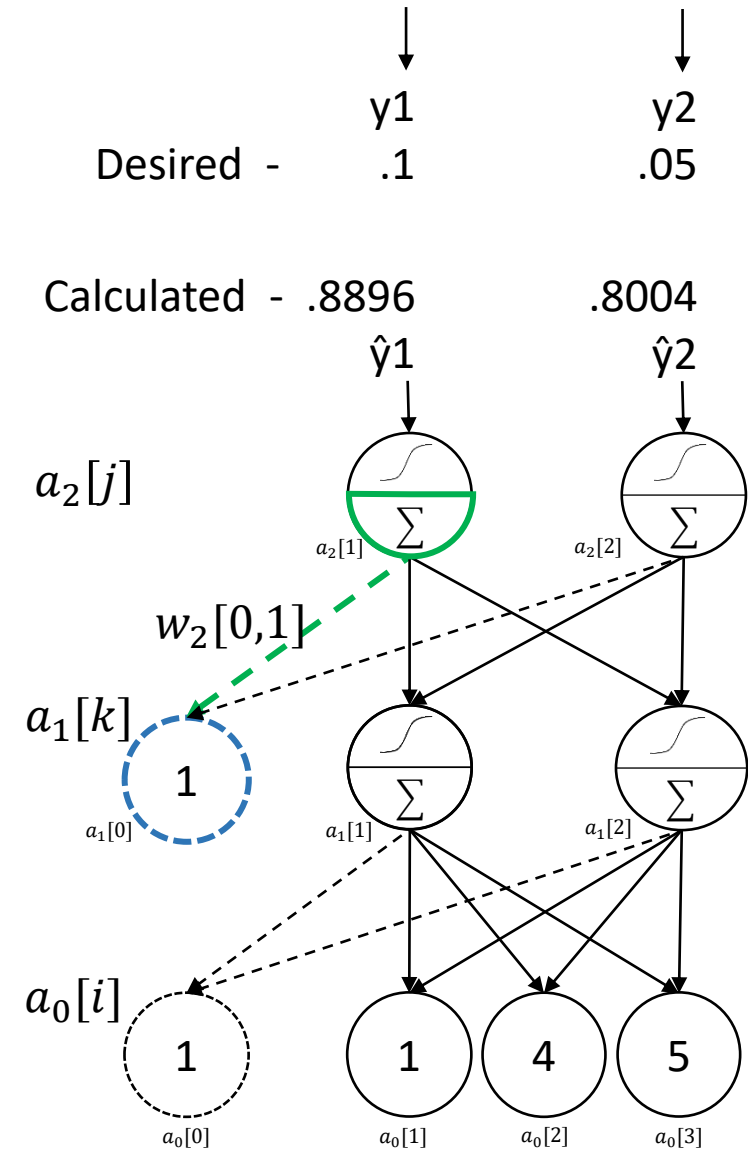
k	$w[k,1]$	$w[k,2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

$$\begin{aligned}
 \frac{\partial E}{\partial w_2[0,1]} &= \frac{\partial E}{\partial a_2[1]} * \frac{\partial a_2[1]}{\partial s_2[1]} * \frac{\partial s_2[1]}{\partial w_2[0,1]} \\
 &= \{\hat{y}_j - y_k\} * \{a_2[1](1 - a_2[1])\} * \{a_1[0]\} \\
 &= .8896 - .1 * .8896(1 - .8896)
 \end{aligned}$$

# Backpropagation

Desired -  $y_1 = .1$   $y_2 = .05$

Calculated -  $\hat{y}_1 = .8896$   $\hat{y}_2 = .8004$



$$x, y$$

$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$$w_1[i, k]$$

i	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

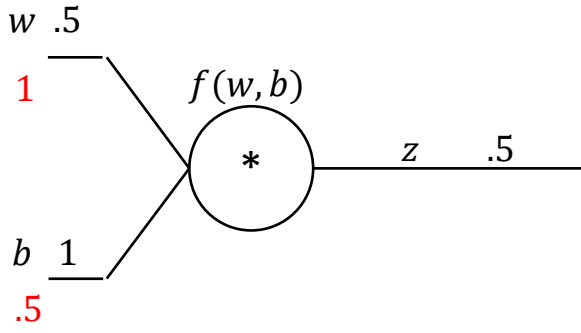
$$w_2[k, j]$$

k	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

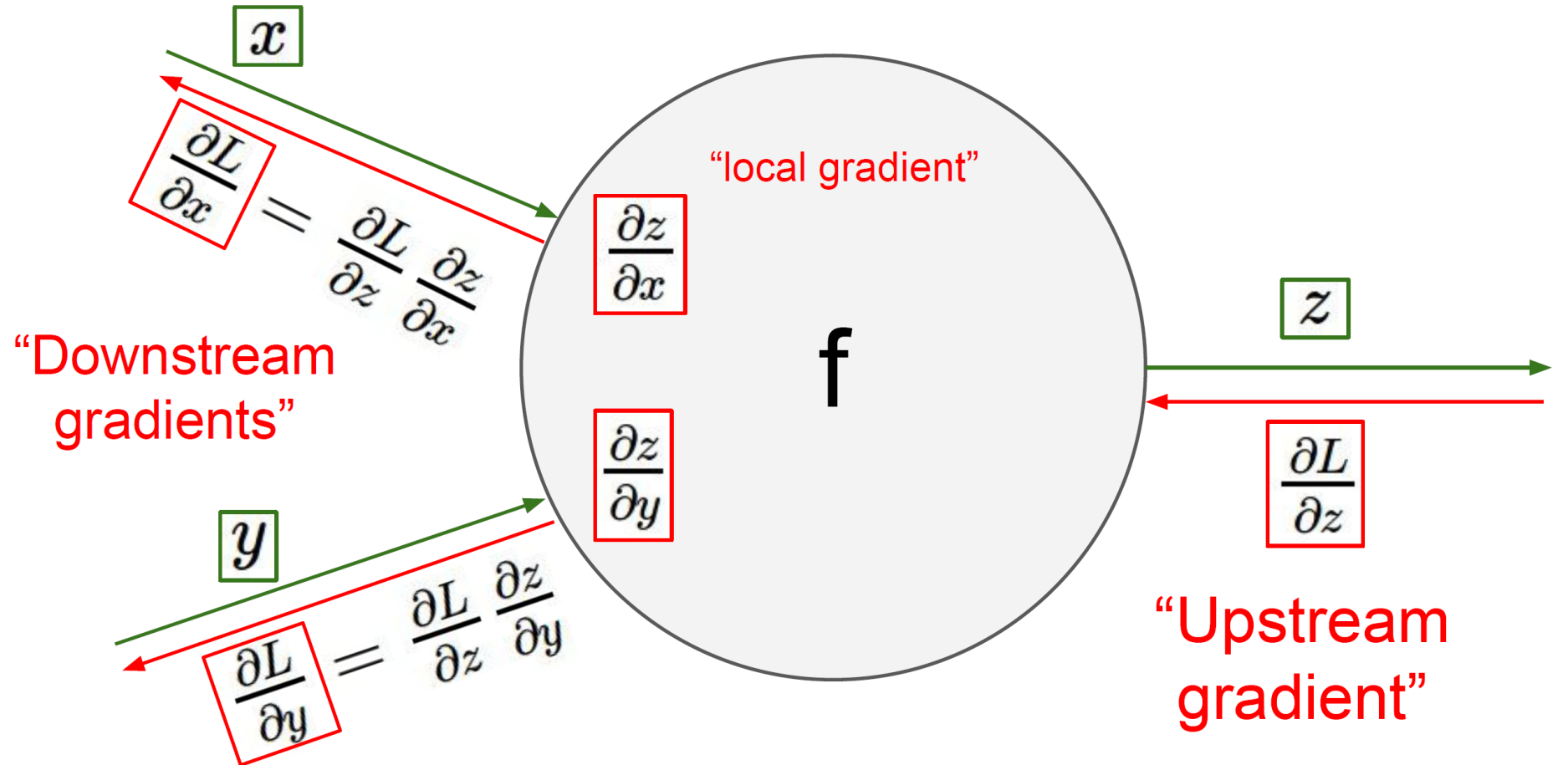
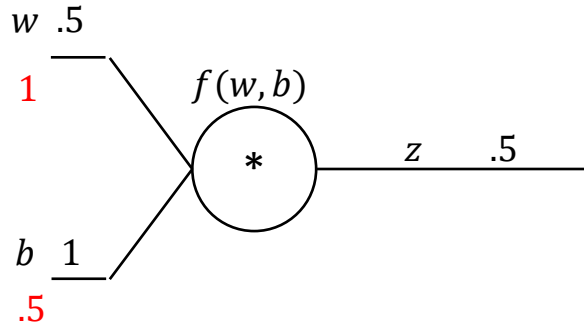
$$\begin{aligned}
 \frac{\partial E}{\partial w_2[0,1]} &= \frac{\partial E}{\partial a_2[1]} * \frac{\partial a_2[1]}{\partial s_2[1]} * \frac{\partial s_2[1]}{\partial w_2[0,1]} \\
 &= \{\hat{y}_j - y_k\} * \{a_2[1](1 - a_2[1])\} * \{a_1[0]\} \\
 &= .8896 - .1 * .8896(1 - .8896) * 1 \\
 &= .0775
 \end{aligned}$$

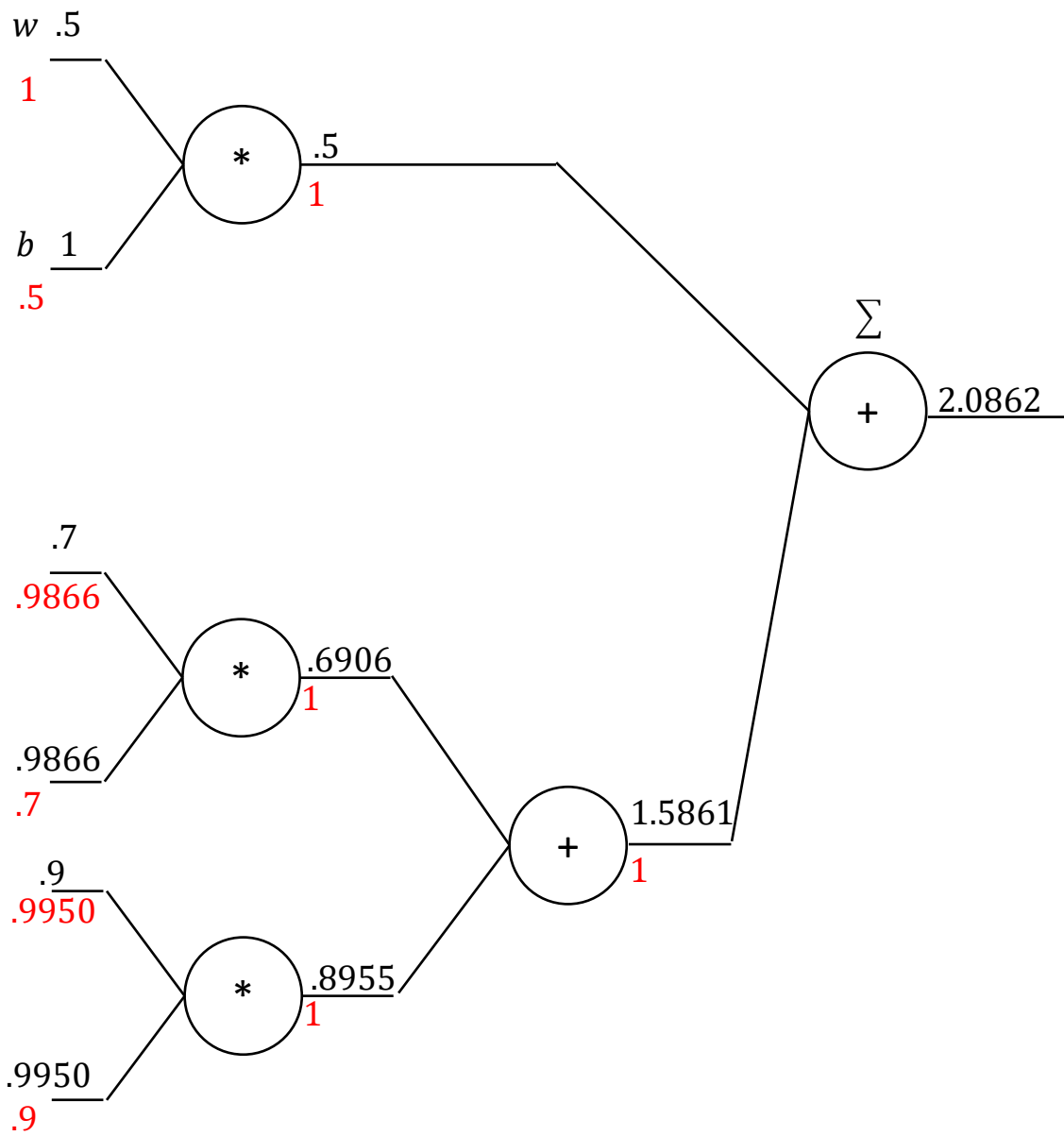
$$c_2[k, j]$$

k	$c[k, 1]$	$c[k, 2]$
0	.0775	0
1	0	0
2	0	0

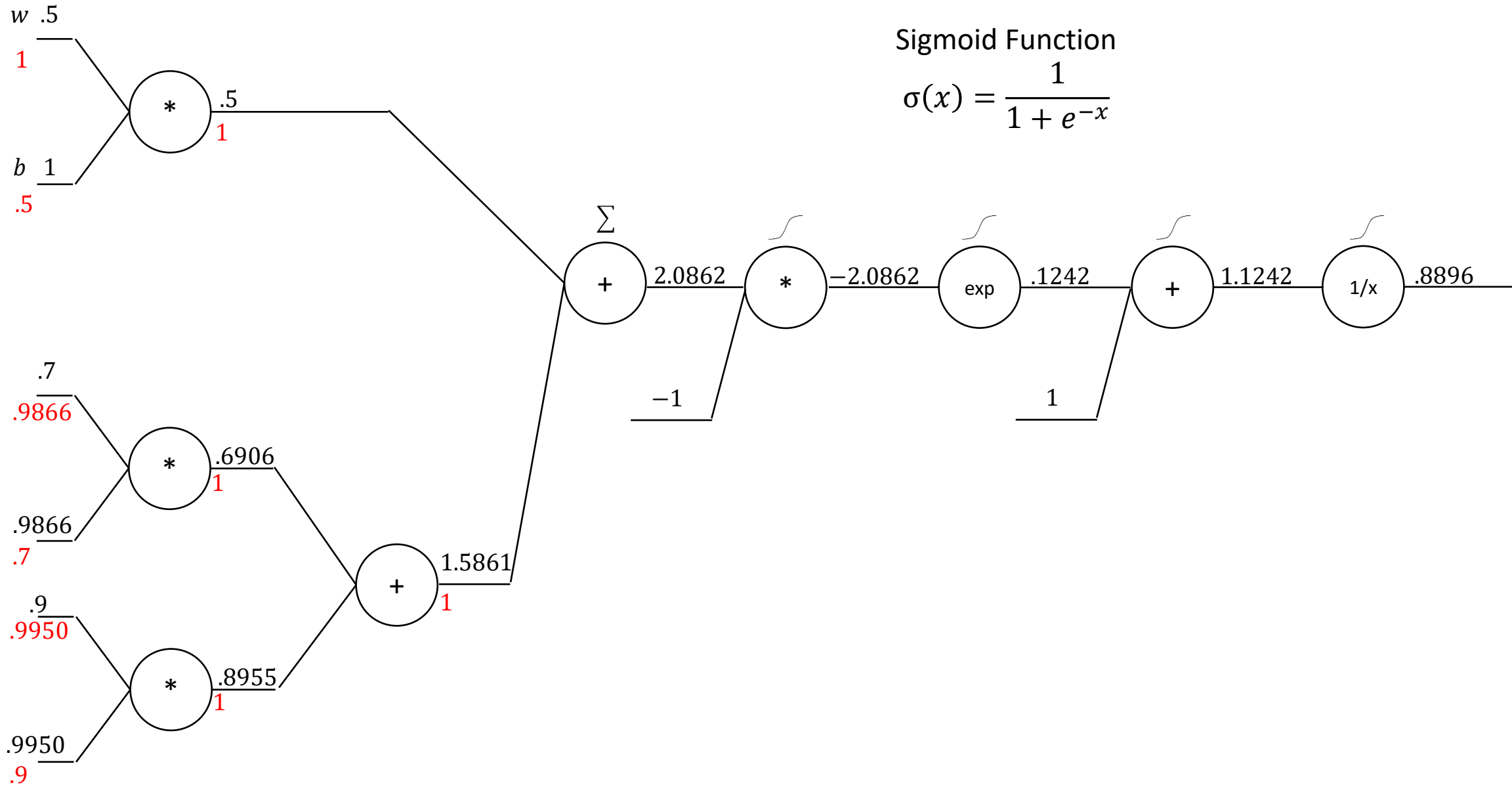




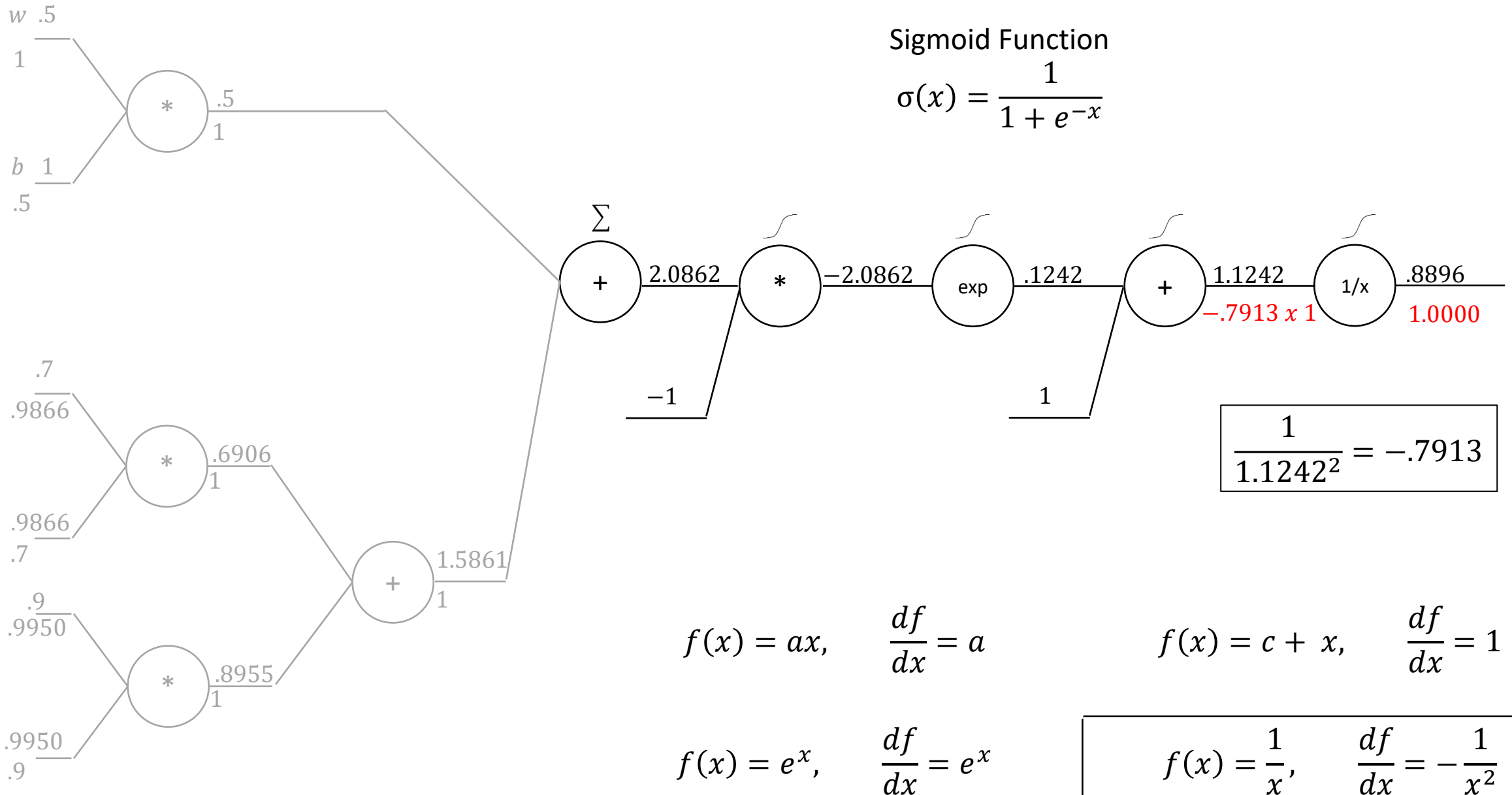




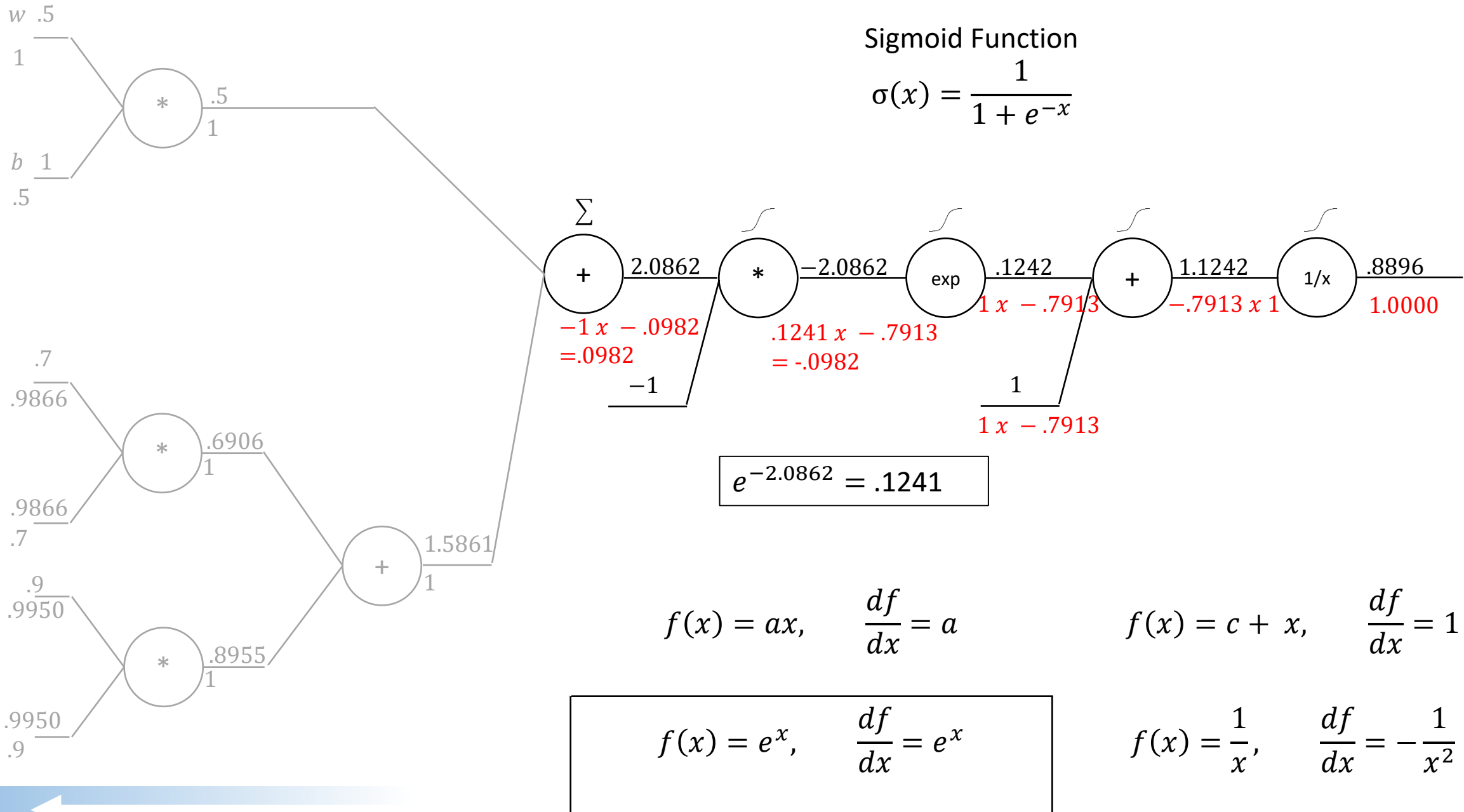
# Computational Graphs



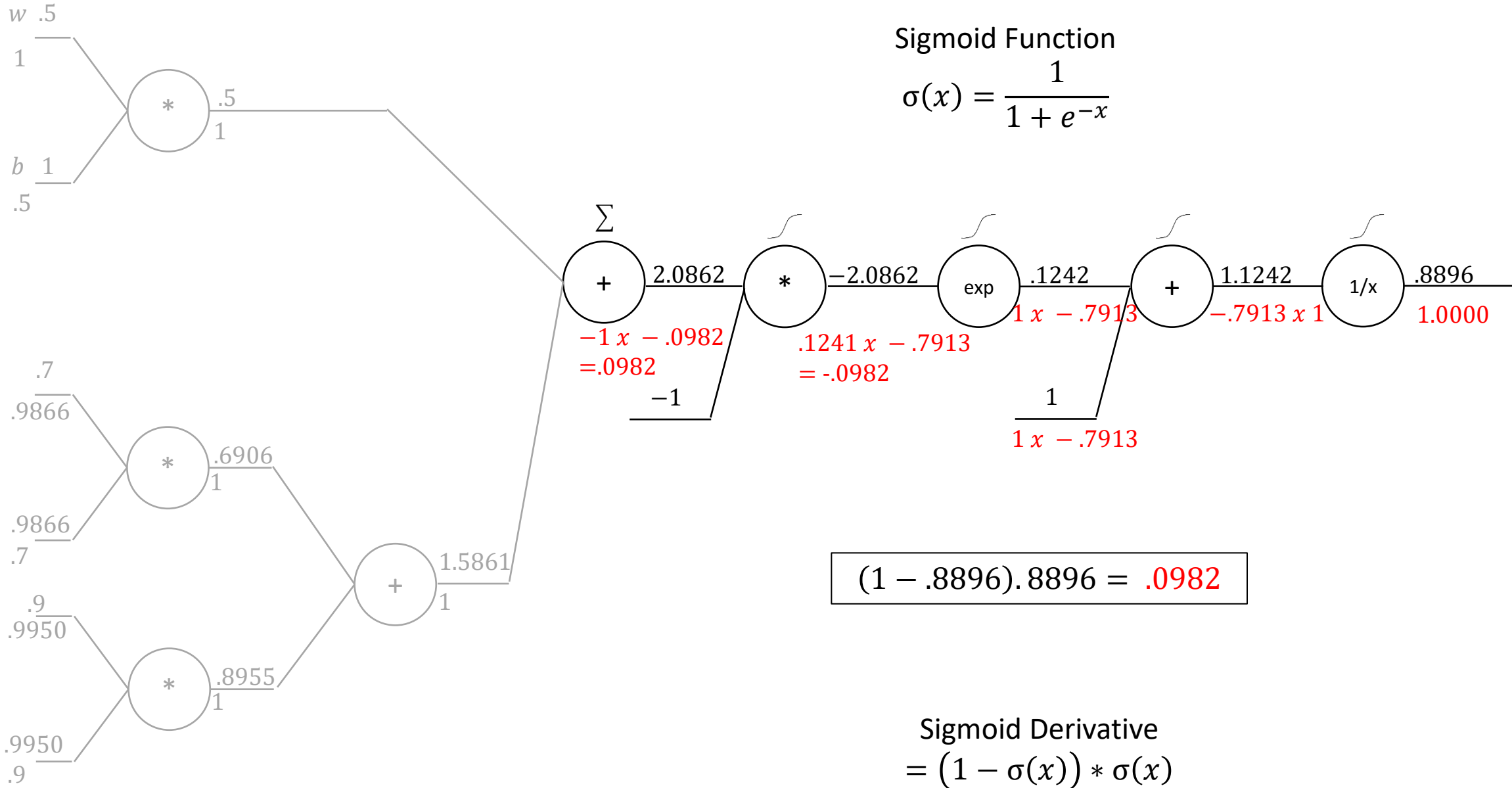
# Computational Graphs



# Computational Graphs



# Computational Graphs



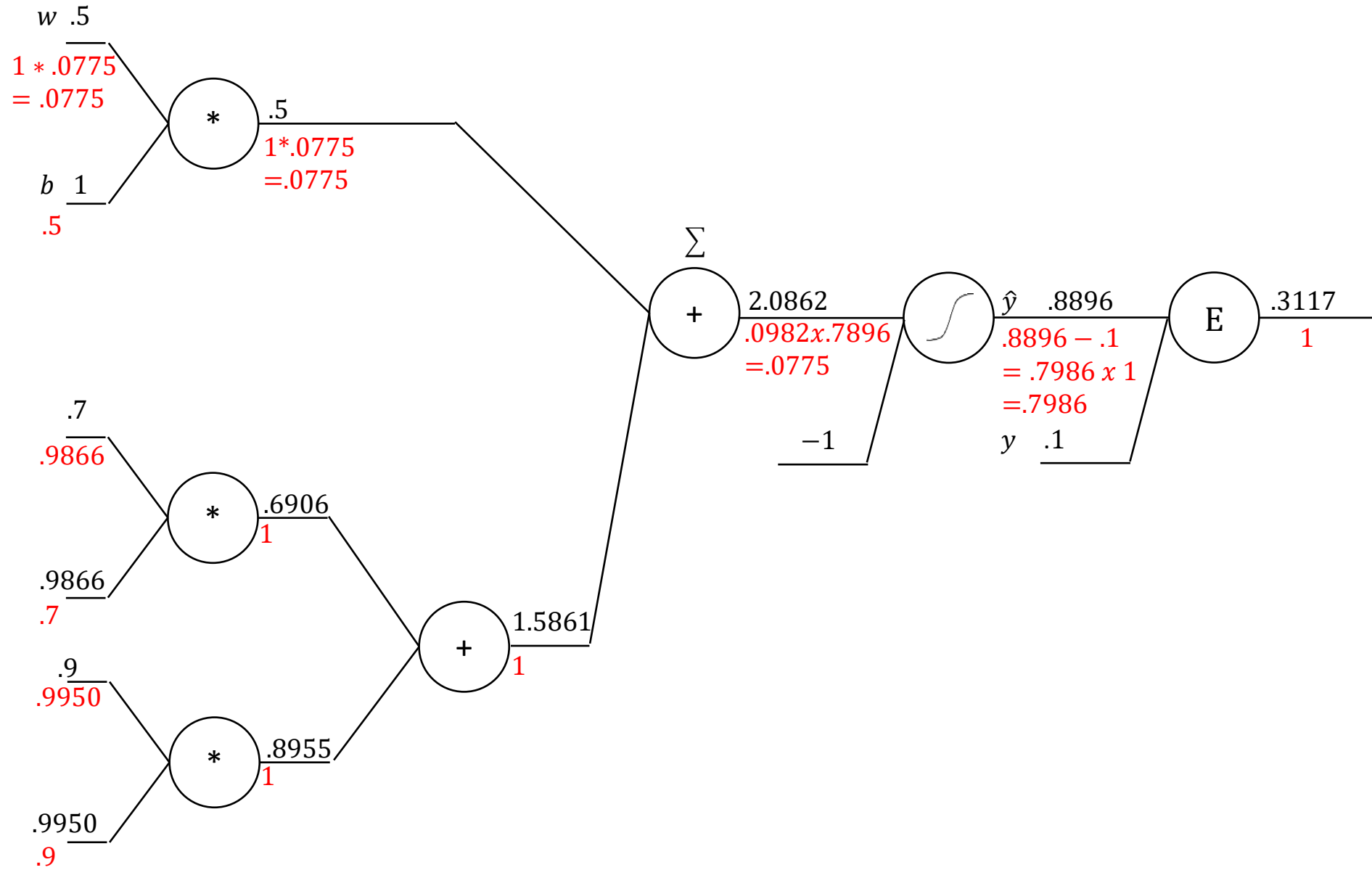
Sigmoid Function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid Derivative

$$= (1 - \sigma(x)) * \sigma(x)$$

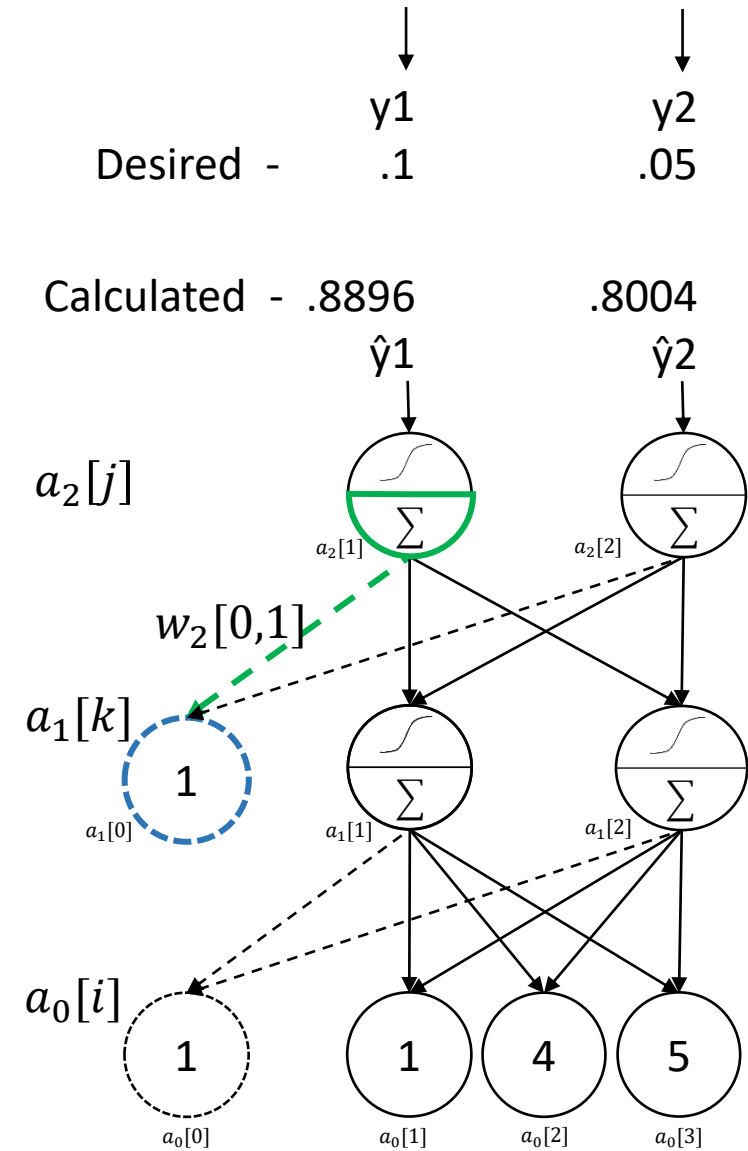
# Computational Graphs



# Computational Graphs

Desired -  $y_1 = .1$   $y_2 = .05$

Calculated -  $\hat{y}_1 = .8896$   $\hat{y}_2 = .8004$



$x, y$				
$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i, k]$		
$i$	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

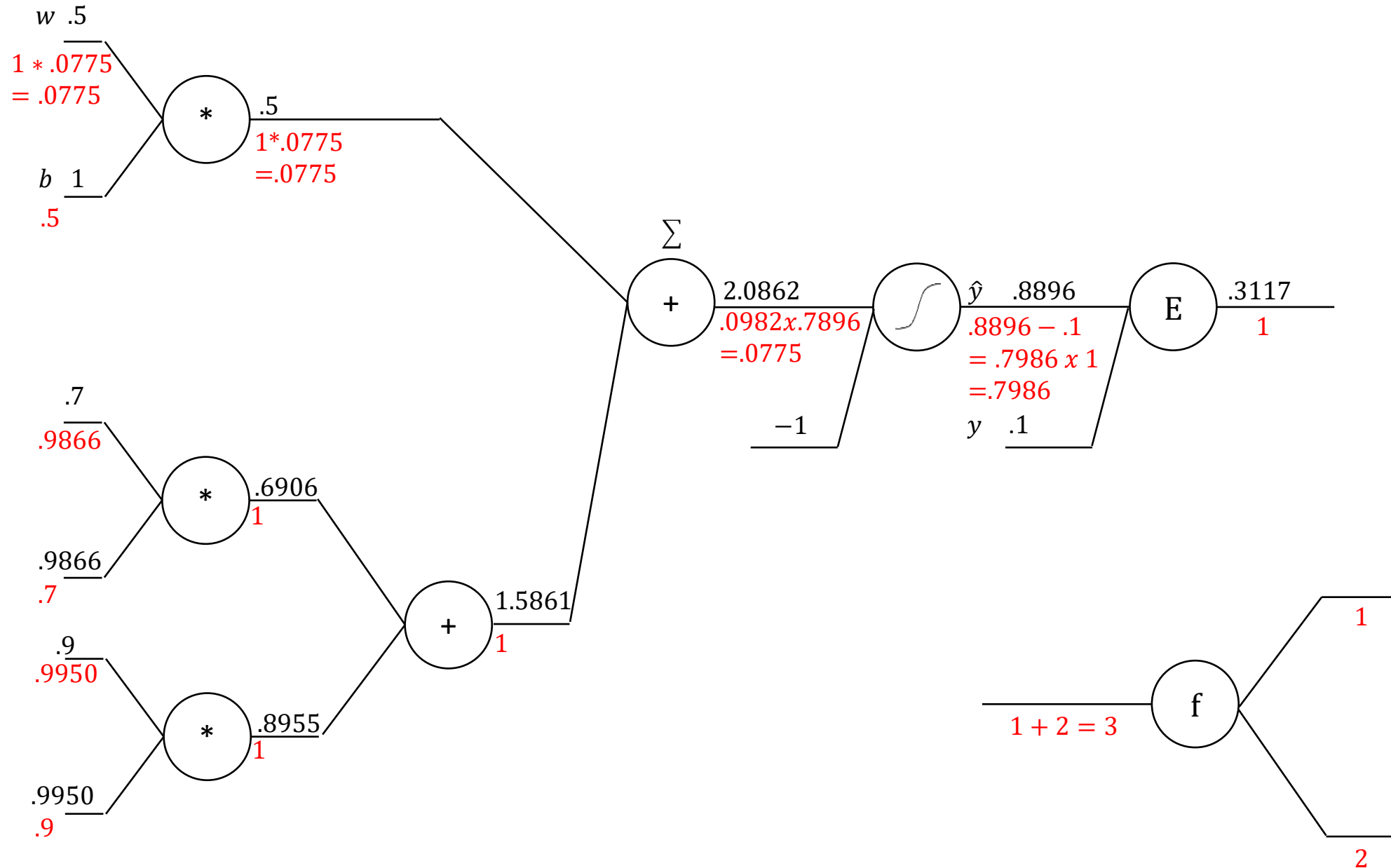
$w_2[k, j]$		
$k$	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

$$\begin{aligned}
 \frac{\partial E}{\partial w_2[0,1]} &= \frac{\partial E}{\partial a_2[1]} * \frac{\partial a_2[1]}{\partial s_2[1]} * \frac{\partial s_2[1]}{\partial w_2[0,1]} \\
 &= \{\hat{y}_j - y_k\} * \{a_2[1](1 - a_2[1])\} * \{a_1[0]\} \\
 &= .8896 - .1 * .8896(1 - .8896) * 1 \\
 &= .0775
 \end{aligned}$$

$c_2[k, j]$		
$k$	$c[k, 1]$	$c[k, 2]$
0	.0775	0
1	0	0
2	0	0



# Computational Graphs

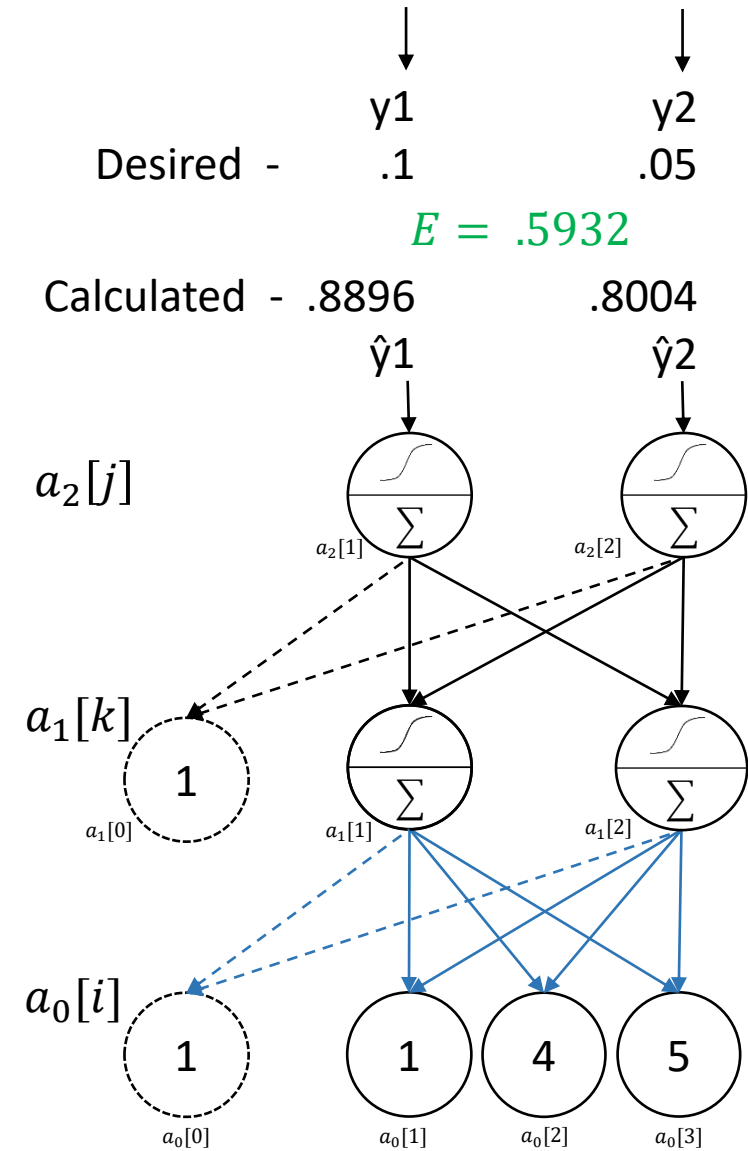


# Backpropagation

Desired -  $y_1 = .1$   $y_2 = .05$

$E = .5932$

Calculated -  $\hat{y}_1 = .8896$   $\hat{y}_2 = .8004$



$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i, k]$		
$i$	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$w_2[k, j]$		
$k$	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

$$\begin{aligned} \frac{\partial E}{\partial w_1[i, k]} &= \frac{\partial E}{\partial a_1[k]} * \frac{\partial a_1[k]}{\partial s_1[k]} * \frac{\partial s_1[k]}{\partial w_1[i, k]} \\ &= \frac{\partial E}{\partial a_1[k]} * \{a_1[j](1 - a_1[j])\} * \{a_0[i]\} \end{aligned}$$

# Backpropagation



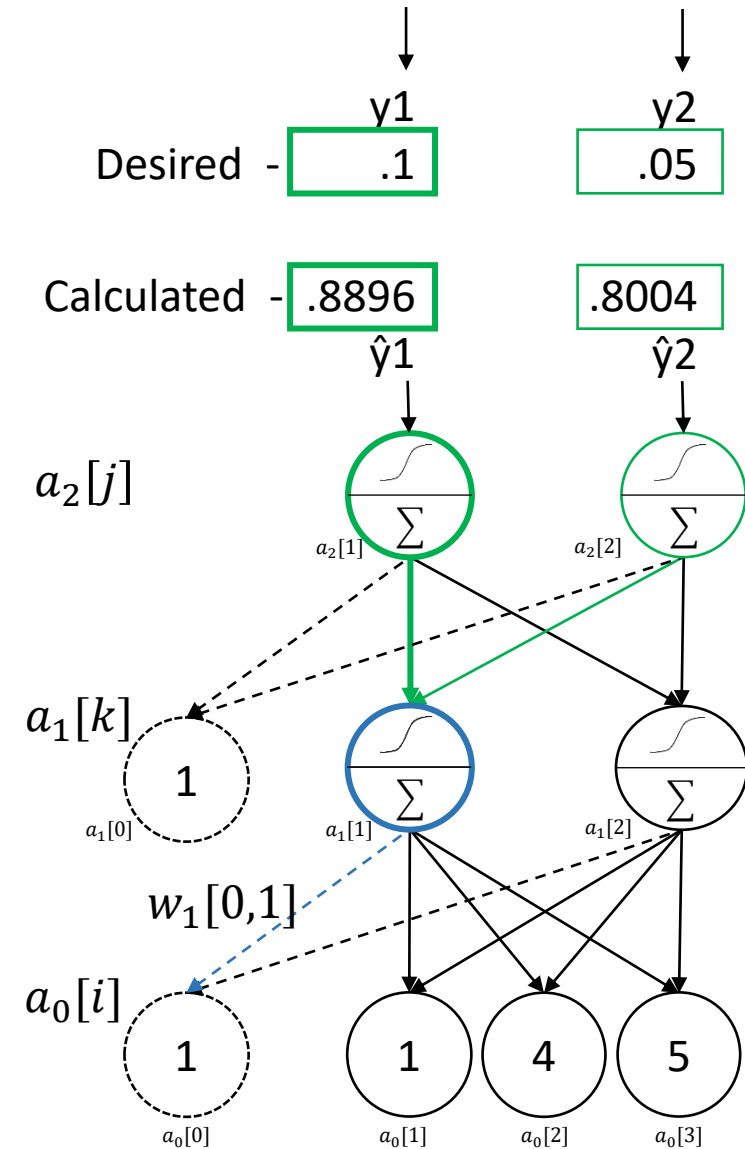
Desired -  $y_1$  .1       $y_2$  .05

Calculated -  $\hat{y}_1$  .8896       $\hat{y}_2$  .8004

$x, y$				
$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i, k]$		
i	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$w_2[k, j]$		
k	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1



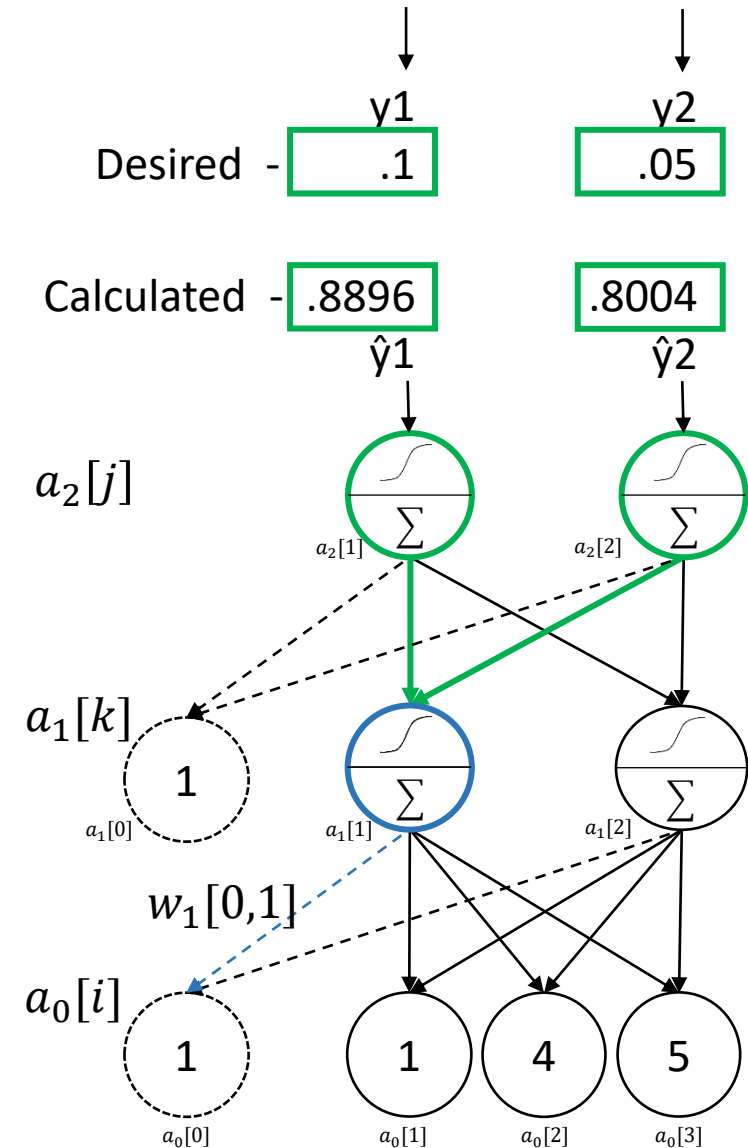
$$\begin{aligned} \frac{\partial E}{\partial w_1[i, k]} &= \frac{\partial E}{\partial a_1[k]} * \frac{\partial a_1[k]}{\partial s_1[k]} * \frac{\partial s_1[k]}{\partial w_1[i, k]} \\ &= \frac{\partial E}{\partial a_1[k]} * \{a_1[k](1 - a_1[k])\} * \{a_0[i]\} \\ \frac{\partial E}{\partial a_1[k]} &= \sum_{j=1}^n \frac{\partial E}{\partial a_2[j]} * \frac{\partial a_2[j]}{\partial s_2[j]} * \frac{\partial s_2[j]}{\partial a_1[k, j]} \\ &= \sum_{j=1}^n \{\hat{y}_j - y_k\} * \{a_2[j](1 - a_2[j])\} * \{w_2[k, j]\} \end{aligned}$$



# Backpropagation

Desired -  $y_1$  .1       $y_2$  .05

Calculated -  $\hat{y}_1$  .8896       $\hat{y}_2$  .8004



$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$i$	$w[i,1]$	$w[i,2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

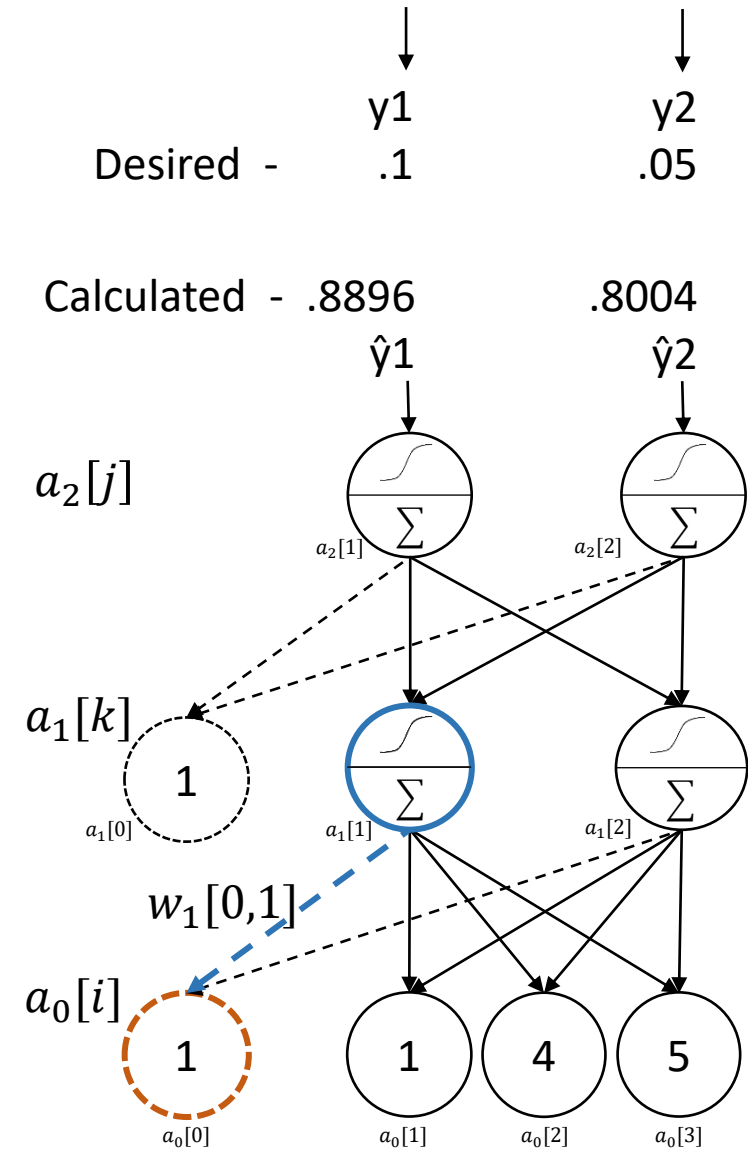
$k$	$w[k,1]$	$w[k,2]$
0	0.5	0.5
1	<span style="color: green;">0.7</span>	<span style="color: green;">0.8</span>
2	0.9	0.1

$$\begin{aligned}
 \frac{\partial E}{\partial w_1[0,1]} &= \frac{\partial E}{\partial a_1[1]} * \frac{\partial a_1[1]}{\partial s_1[1]} * \frac{\partial s_1[1]}{\partial w_1[0,1]} \\
 &= \frac{\partial E}{\partial a_1[1]} * \{a_1[1](1 - a_1[1])\} * \{a_0[0]\} \\
 \frac{\partial E}{\partial a_1[k]} &= \sum_{j=1}^n \frac{\partial E}{\partial a_2[j]} * \frac{\partial a_2[j]}{\partial s_2[j]} * \frac{\partial s_2[j]}{\partial a_1[k,j]} \\
 &= \sum_{j=1}^n \{\hat{y}_j - y_k\} * \{a_2[j](1 - a_2[j])\} * \{w_2[k,j]\} \\
 &= .8896 - .1 * .8896(1 - .8896) * .7 \\
 &\quad + .8004 - .05 * .8004(1 - .8004) * .8 \\
 &= .1502
 \end{aligned}$$

# Backpropagation

Desired -  $y_1 = .1$   $y_2 = .05$

Calculated -  $\hat{y}_1 = .8896$   $\hat{y}_2 = .8004$



$$x, y$$

$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
1.0000	4.0000	5.0000	0.1000	0.0500

$$w_1[i, k]$$

i	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$$w_2[k, j]$$

k	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

$$\begin{aligned}
 \frac{\partial E}{\partial w_1[0,1]} &= \frac{\partial E}{\partial a_1[1]} * \frac{\partial a_1[1]}{\partial s_1[1]} * \frac{\partial s_1[1]}{\partial w_1[0,1]} \\
 &= \frac{\partial E}{\partial a_1[1]} * \{a_1[1](1 - a_1[1])\} * \{a_0[0]\} \\
 &= .1502 * .9866(1 - .9866) * 1 \\
 &= .0020
 \end{aligned}$$

$$c_1[i, k]$$

i	$w[i, 1]$	$w[i, 2]$
0	.0020	0
1	0	0
2	0	0
3	0	0

# Backpropagation



	$x, y$				
	$x_1$	$x_2$	$x_3$	$y_1$	$y_2$
$r=1$	1.0000	4.0000	5.0000	0.1000	0.0500
$\vdots$	0.1000	-5.0000	3.0000	0.1221	0.0964
	6.0000	-5.5420	4.8970	0.1061	0.0702
$r=4$	4.0000	8.0000	9.0000	0.0996	0.0641
	12.0000	-2.0000	0.0063	0.1110	0.0732
	6.0000	-5.5000	4.8970	0.1060	0.0701

$w_1[i, k]$		
$i$	$w[i, 1]$	$w[i, 2]$
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$w_2[k, j]$		
$k$	$w[k, 1]$	$w[k, 2]$
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

$c_1[i, k]$		
$i$	$w[i, 1]$	$w[i, 2]$
0	.0604	.0332
1	.1134	.0427
2	-.2929	-.1658
3	.2193	.1129

$c_2[k, j]$		
$k$	$c[k, 1]$	$c[k, 2]$
0	.3447	.4816
1	.2929	.4176
2	.2930	.4191

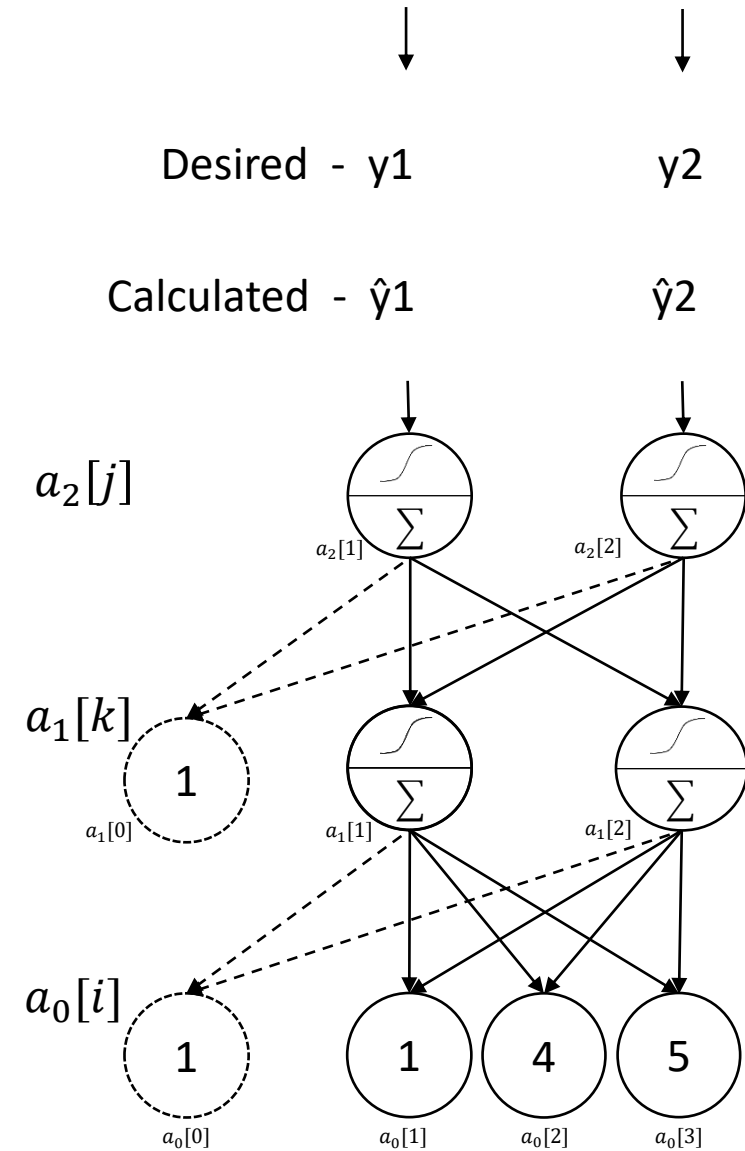
$w_1^*[i, k]$		
$i$	$w[i, 1]$	$w[i, 2]$
0	.4994	.4997
1	.0989	.1996
2	.3029	.4017
3	.4978	.5989

$w_2^*[k, j]$		
$k$	$w[k, 1]$	$w[k, 2]$
0	.4966	.0039
1	.4971	.0038
2	.0021	.0004

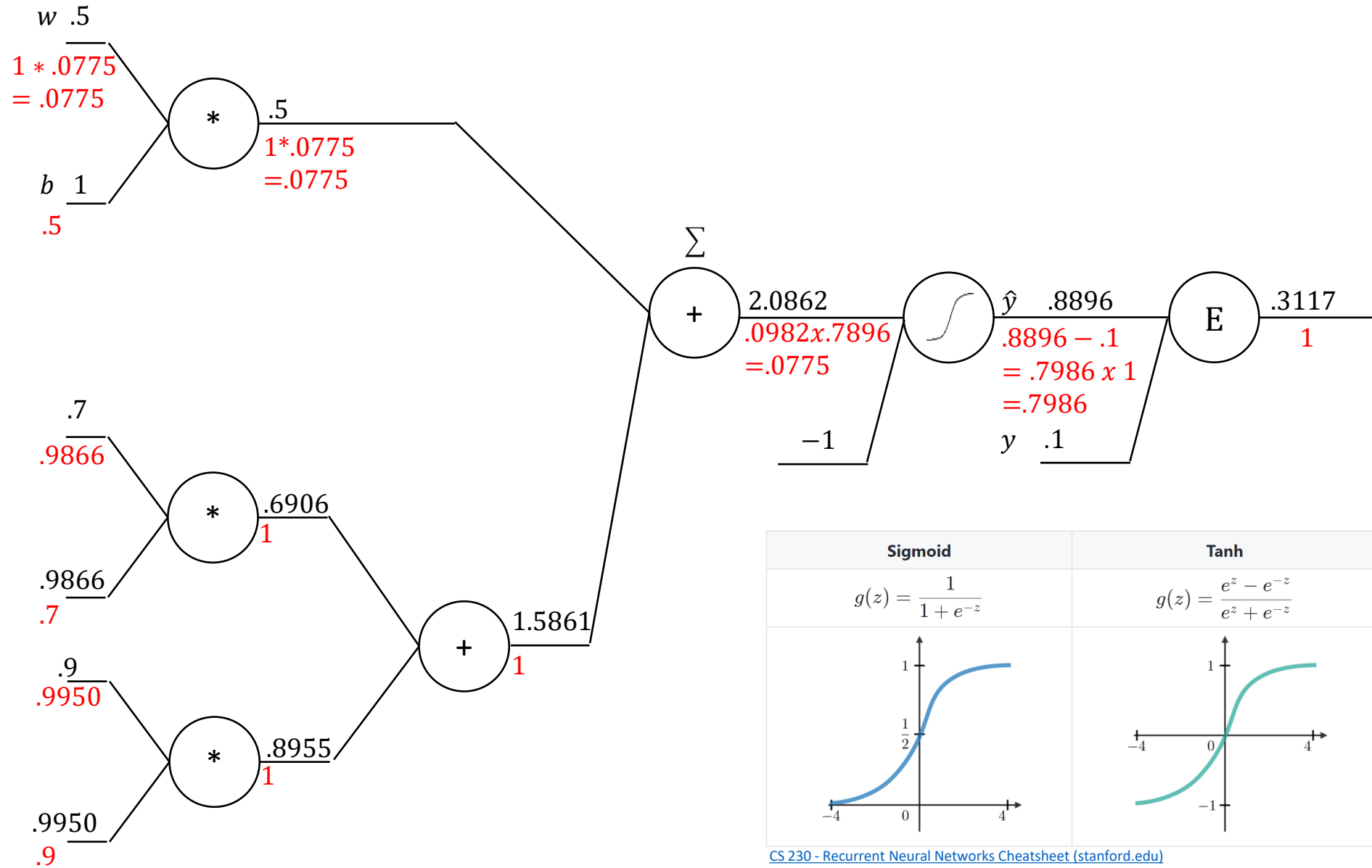
Optimizer:  
Stochastic Gradient Descent

Learning rate  $\alpha = .01$

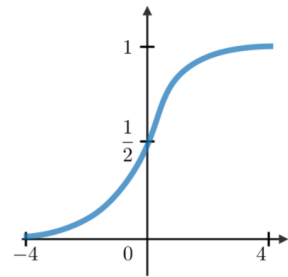
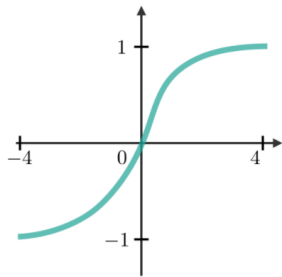
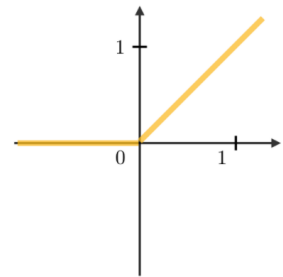
$$w^* = w - \alpha * c$$



# Backpropagation



Rectified Linear Unit

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

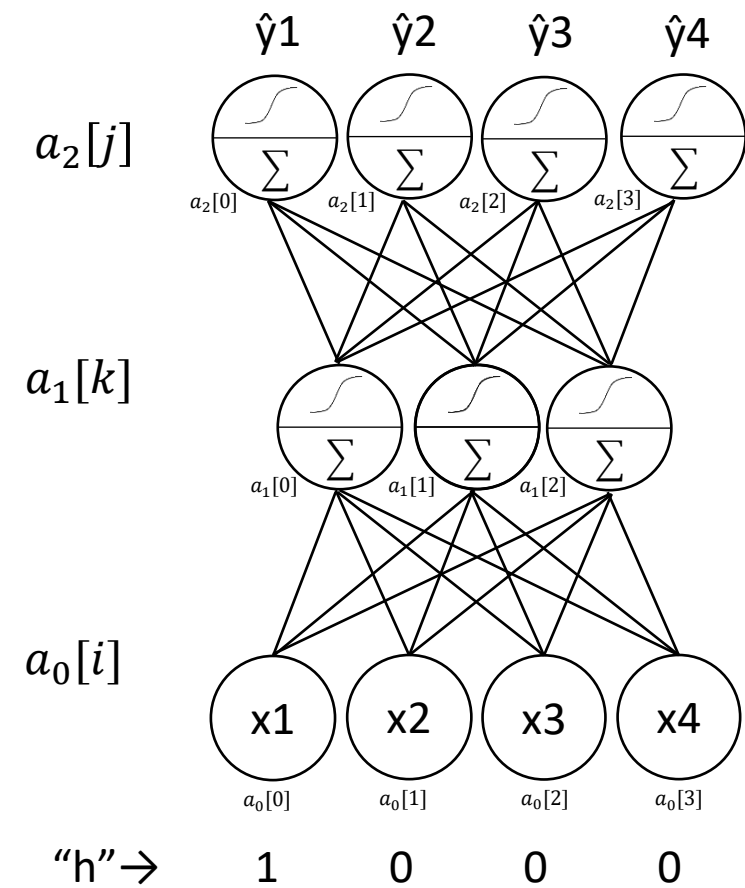
“e” →

y1	y2	y3	y4
0	1	0	0

“hello”

Vocabulary

“h”	[1, 0, 0, 0]
“e”	[0, 1, 0, 0]
“l”	[0, 0, 1, 0]
“o”	[0, 0, 0, 1]





“e” →

y1	y2	y3	y4
0	1	0	0

“hello”

Vocabulary

“h”	[1, 0, 0, 0]
“e”	[0, 1, 0, 0]
“l”	[0, 0, 1, 0]
“o”	[0, 0, 0, 1]

“h” → “e”

“e” → “l”

“l” → “l”  
“l” → “o” ?

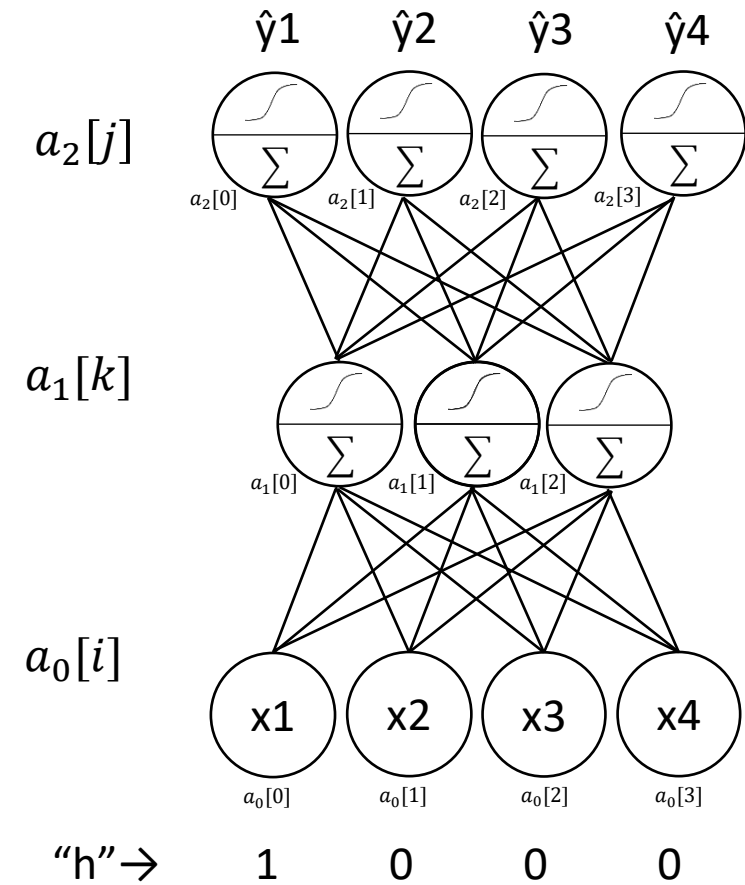
Ambiguous Targets

“hel” → “p”

“hell” → “o”

“heliu” → “m”

Fixed Width Input/Output



“e” →      y1      y2      y3      y4  
                  0      1      0      0

“hello”

Vocabulary

“h”	[1, 0, 0, 0]
“e”	[0, 1, 0, 0]
“l”	[0, 0, 1, 0]
“o”	[0, 0, 0, 1]

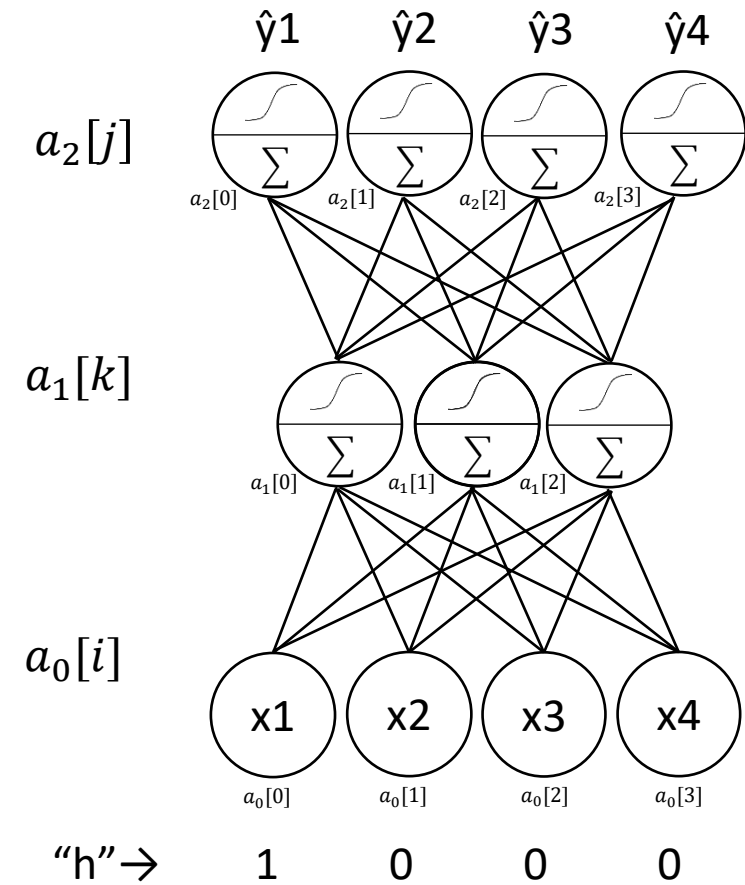
“h” → “e”

“e” → “l”

“l” → “l”

“l” → “o” ?

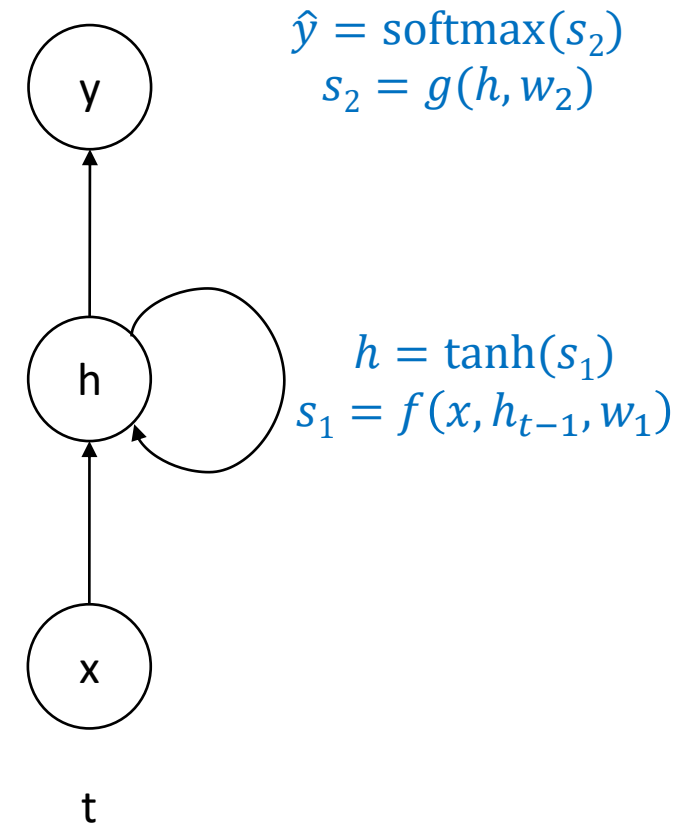
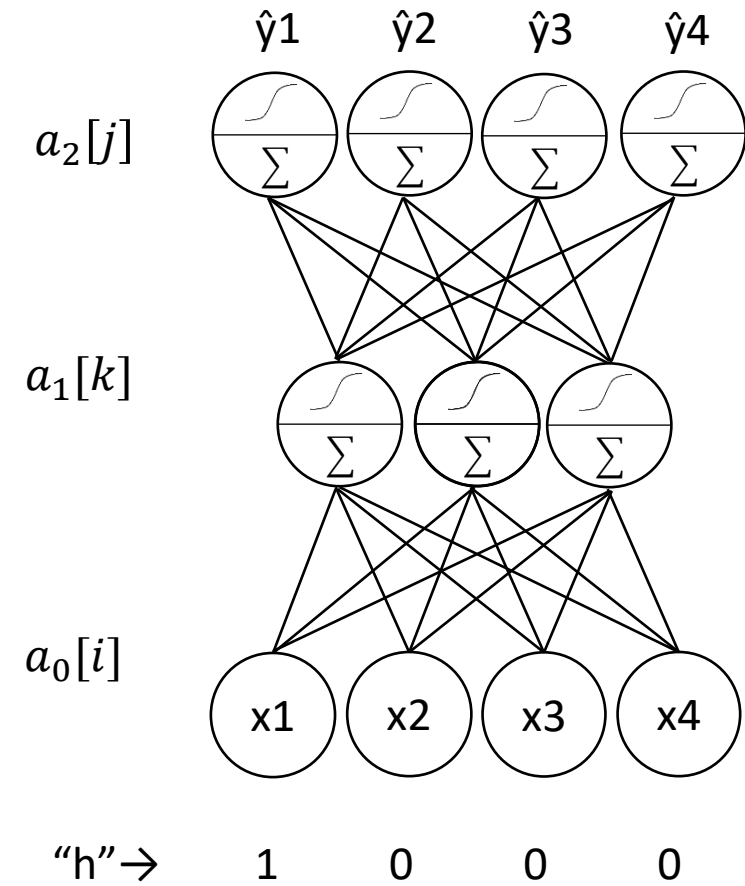
Ambiguous Targets



# Recurrent Neural Networks

"e" →

y1	y2	y3	y4
0	1	0	0



$$W_{yh} \quad H$$

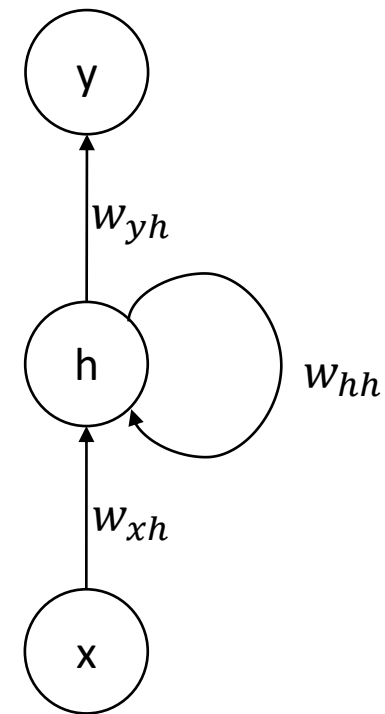
$$Y \begin{bmatrix} y_1 h_1, y_2 h_2 \dots y_1 h_k \\ y_2 h_1, y_2 h_2 \dots y_2 h_k \\ y_n h_1, y_n h_2 \dots y_n h_k \end{bmatrix}$$

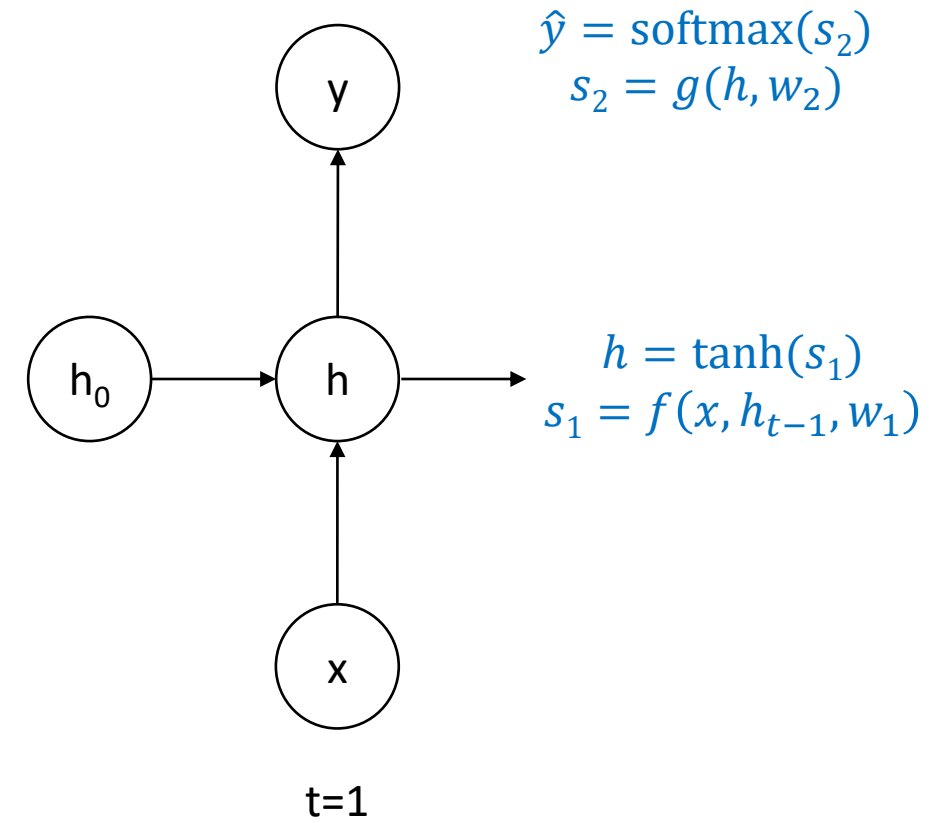
$$W_{hh} \quad H(t-1)$$

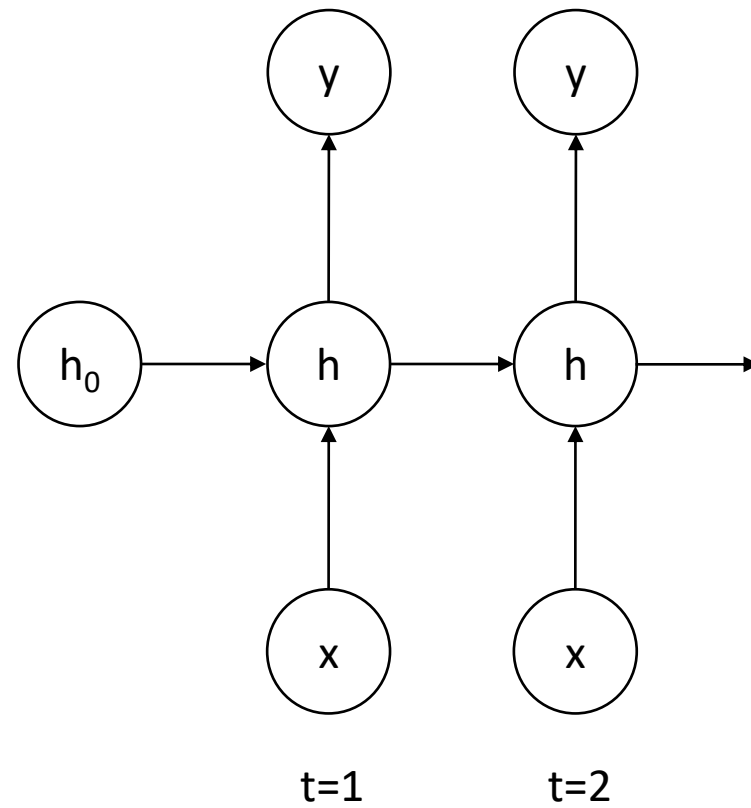
$$H \begin{bmatrix} h_1 h_1, h_1 h_2 \dots h_1 h_k \\ h_2 h_1, h_2 h_2 \dots h_2 h_k \\ h_k h_1, h_k h_2 \dots h_k h_k \end{bmatrix}$$

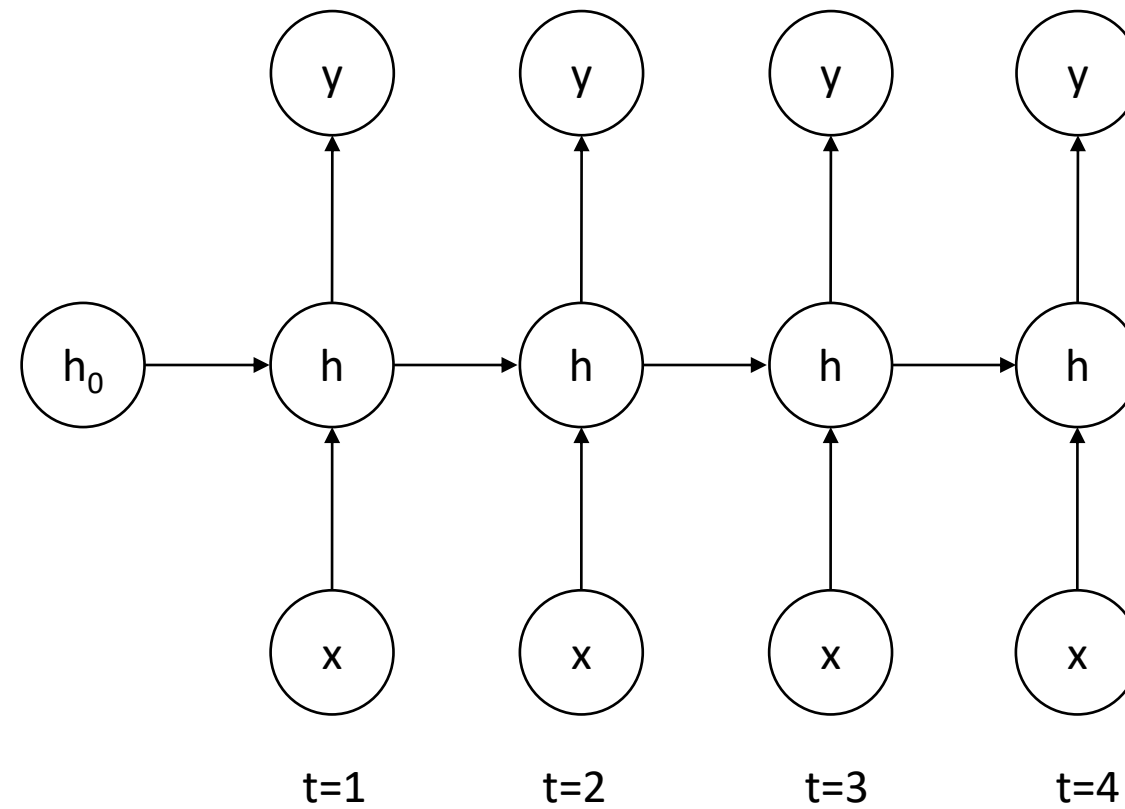
$$W_{xh} \quad X$$

$$H \begin{bmatrix} h_1 x_1, h_1 x_2 \dots h_1 x_m \\ h_2 x_1, h_2 x_2 \dots h_2 x_m \\ h_k x_1, h_k x_2 \dots h_k x_m \end{bmatrix}$$

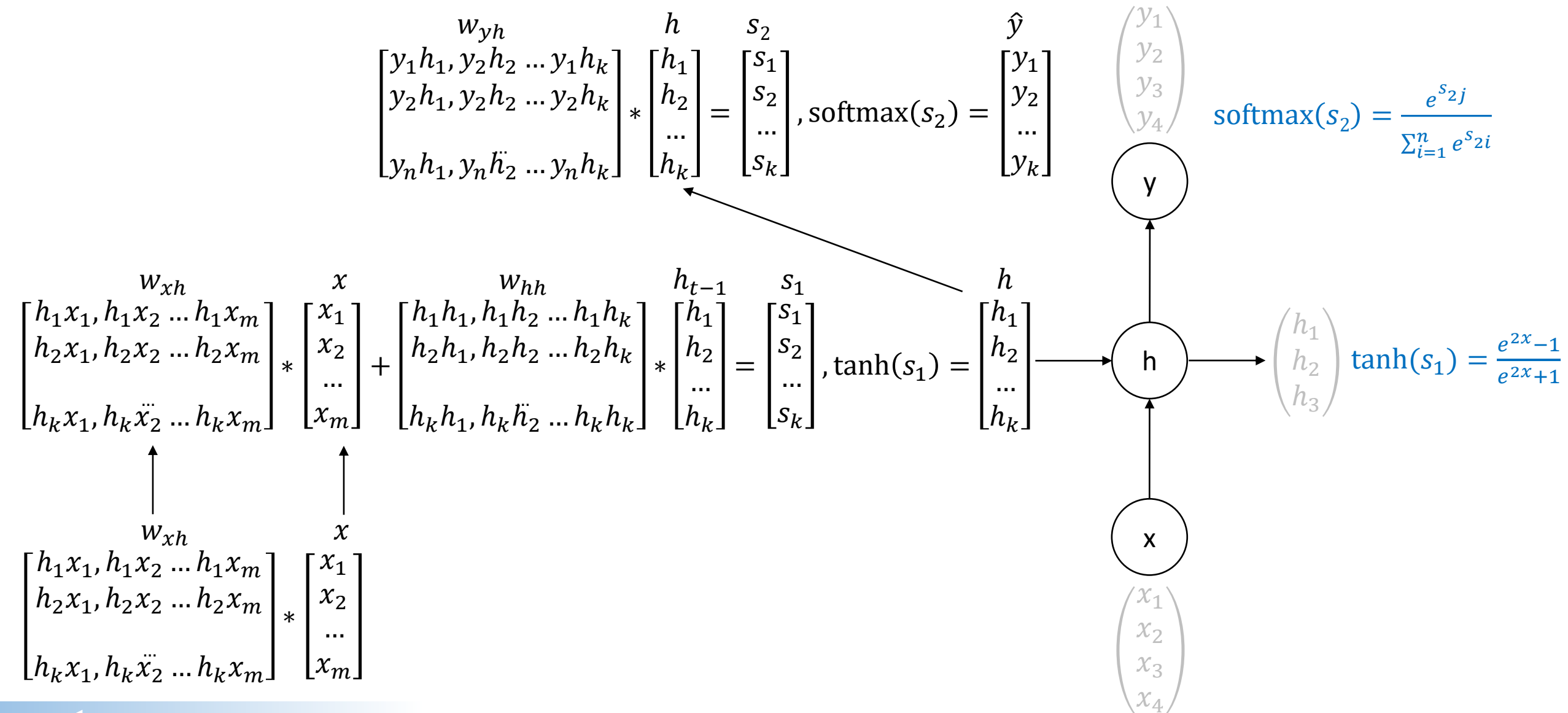








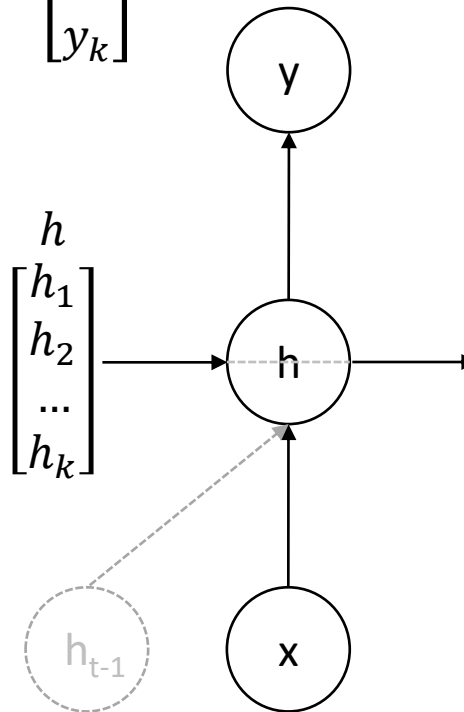
# Recurrent Neural Networks





$$\begin{matrix} w_{yh} \\ \begin{bmatrix} y_1 h_1, y_2 h_2 & \dots & y_1 h_k \\ y_2 h_1, y_2 h_2 & \dots & y_2 h_k \\ \vdots & & \vdots \\ y_n h_1, y_n h_2 & \dots & y_n h_k \end{bmatrix} \end{matrix} * \begin{matrix} h \\ \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \end{bmatrix} \end{matrix} = \begin{matrix} s_2 \\ \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_k \end{bmatrix} \end{matrix}, \text{softmax}(s_2) = \begin{matrix} \hat{y} \\ \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} \end{matrix}$$

$$\begin{matrix} w_{xh} \\ \begin{bmatrix} h_1 x_1, h_1 x_2 & \dots & h_1 x_m \\ h_2 x_1, h_2 x_2 & \dots & h_2 x_m \\ \vdots & & \vdots \\ h_k x_1, h_k x_2 & \dots & h_k x_m \end{bmatrix} \end{matrix} * \begin{matrix} x \\ \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \end{matrix} + \begin{matrix} w_{hh} \\ \begin{bmatrix} h_1 h_1, h_1 h_2 & \dots & h_1 h_k \\ h_2 h_1, h_2 h_2 & \dots & h_2 h_k \\ \vdots & & \vdots \\ h_k h_1, h_k h_2 & \dots & h_k h_k \end{bmatrix} \end{matrix} * \begin{matrix} h_{t-1} \\ \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \end{bmatrix} \end{matrix} = \begin{matrix} s_1 \\ \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_k \end{bmatrix} \end{matrix}, \text{tanh}(s_1) = \begin{matrix} h \\ \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \end{bmatrix} \end{matrix}$$

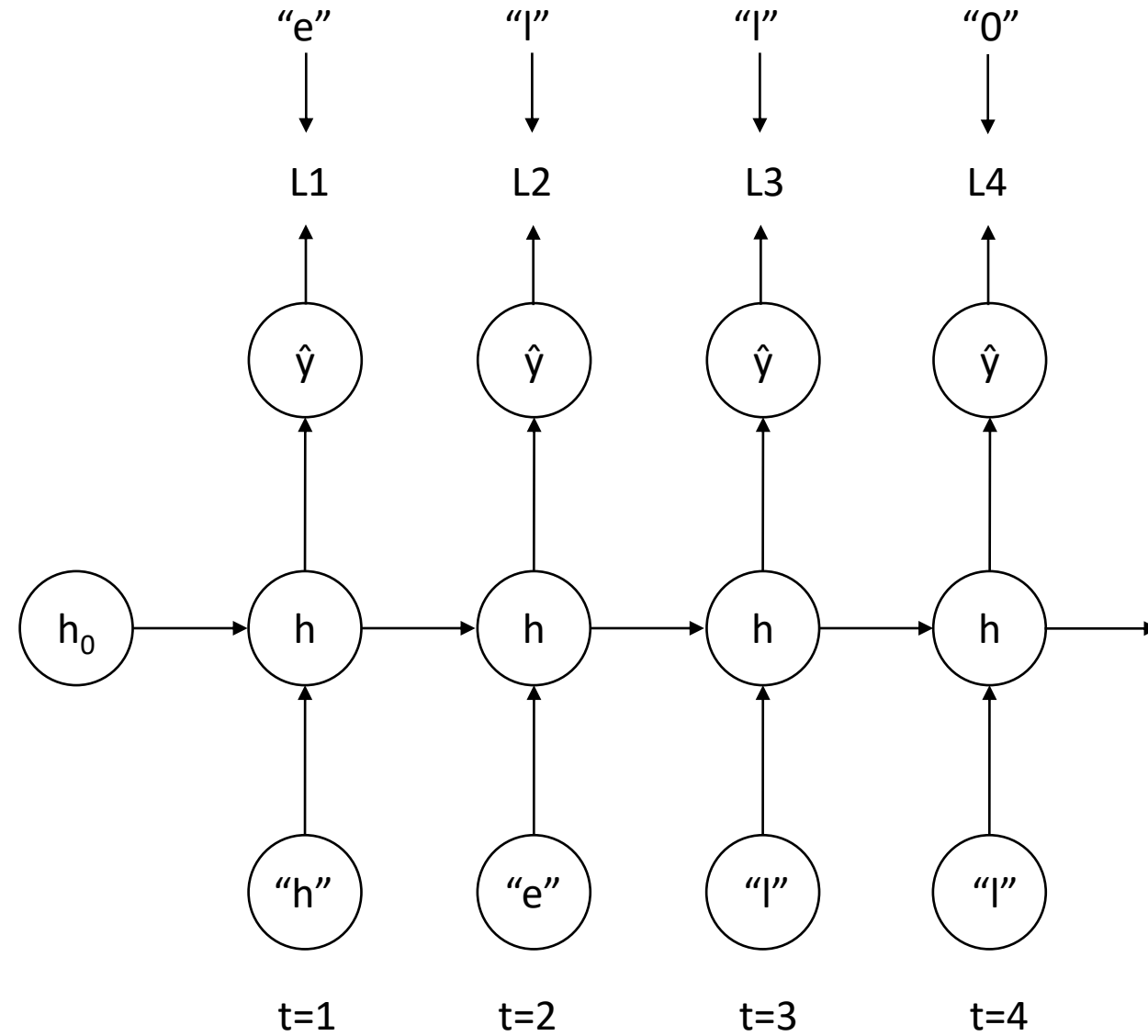


# Recurrent Neural Networks

$$L_{total} = - \sum_{t=1}^n y_t \log(\hat{y})$$

$$L_t = -y_t \log(\hat{y})$$

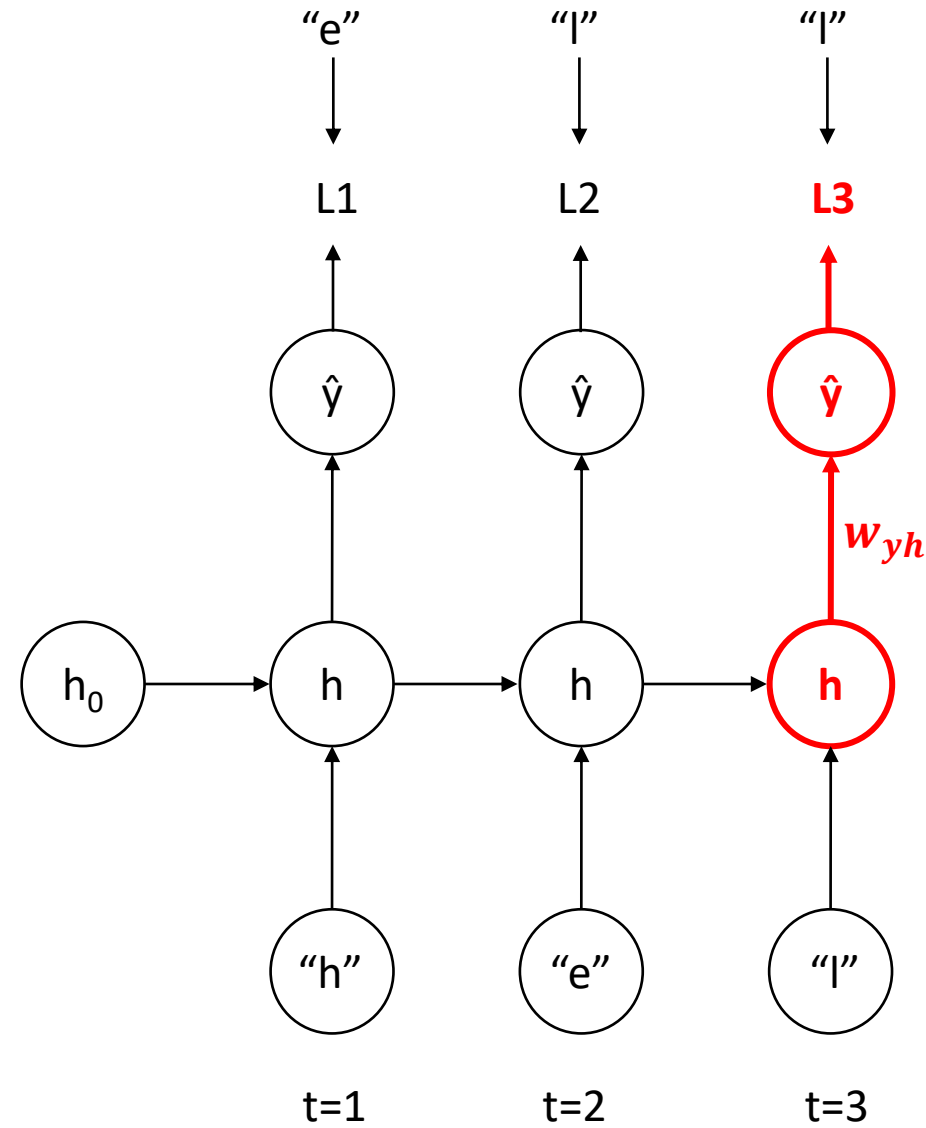
Multi-class Cross Entropy Loss



# Recurrent Neural Networks

$$L_{total} = - \sum_{t=1}^n y_t \log(\hat{y})$$

$$L_t = -y_t \log(\hat{y})$$



$$\frac{\partial L3}{\partial W_{yh}} = \frac{\partial L3}{\partial \hat{Y}3} * \frac{\partial \hat{Y}3}{\partial W_{yh}}$$

$$\frac{\partial LT}{\partial W_{yh}} = \frac{\partial LT}{\partial \hat{Y}T} * \frac{\partial \hat{Y}T}{\partial W_{yh}}$$

For derivatives of Softmax and Cross Entropy Loss

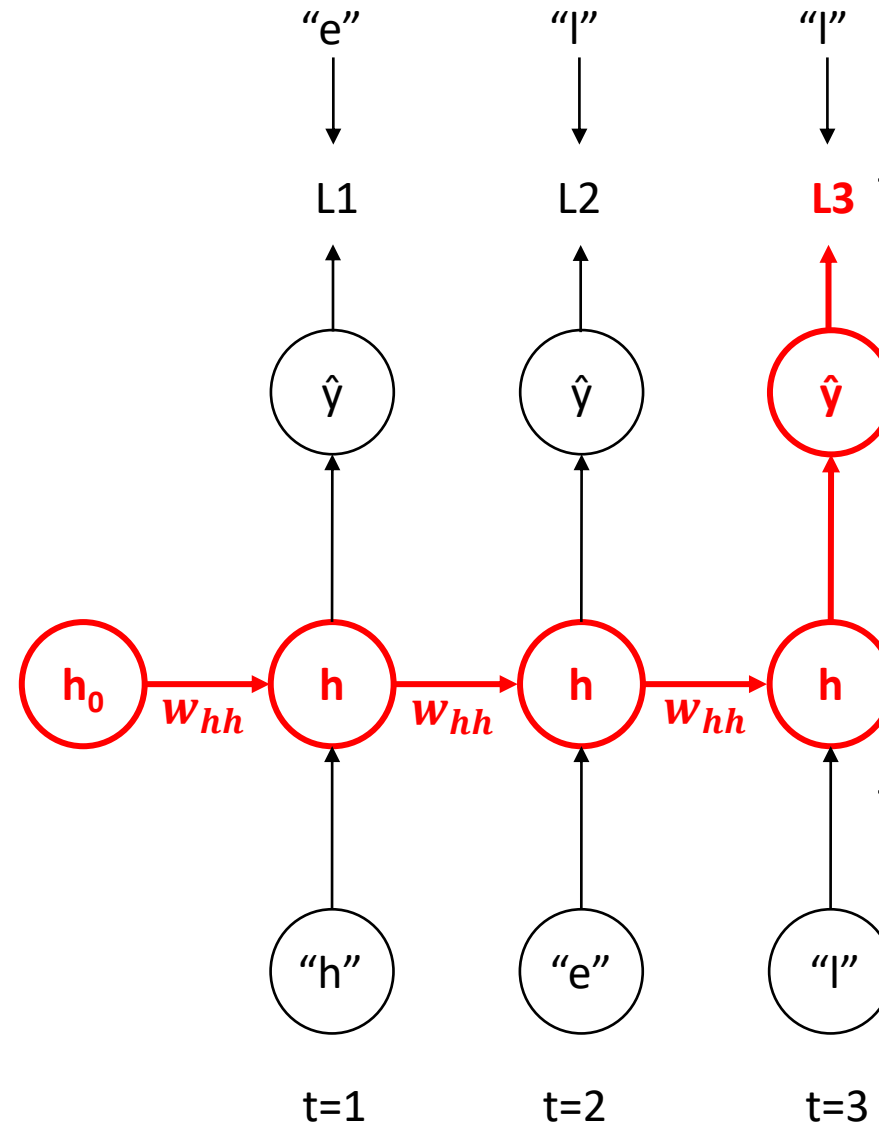
[Part 2: Softmax Regression \(saitcelebi.com\)](http://saitcelebi.com)

[How to compute the derivative of softmax and cross-entropy – Charlee Li](#)

# Recurrent Neural Networks

$$L_{total} = - \sum_{t=1}^n y_t \log(\hat{y})$$

$$L_t = -y_t \log(\hat{y})$$



$$\frac{\partial L_3}{\partial W_{hh}} = \frac{\partial L_3}{\partial \hat{Y}_3} * \frac{\partial \hat{Y}_3}{\partial H_3} * \frac{\partial H_3}{\partial W_{hh}} + \frac{\partial L_3}{\partial \hat{Y}} * \frac{\partial \hat{Y}_3}{\partial H_3} * \frac{\partial H_3}{\partial H_2} * \frac{\partial H_2}{\partial W_{hh}} + \frac{\partial L_3}{\partial \hat{Y}} * \frac{\partial \hat{Y}_3}{\partial H_3} * \frac{\partial H_3}{\partial H_2} * \frac{\partial H_2}{\partial H_1} * \frac{\partial H_1}{\partial W_{hh}}$$

$$\frac{\partial LT}{\partial W_{hh}} = \sum_{t=1}^T \frac{\partial LT}{\partial \hat{Y}_T} * \frac{\partial \hat{Y}_T}{\partial H_t} * \frac{\partial H_t}{\partial W_{hh}}$$

For derivatives of Softmax and Cross Entropy Loss

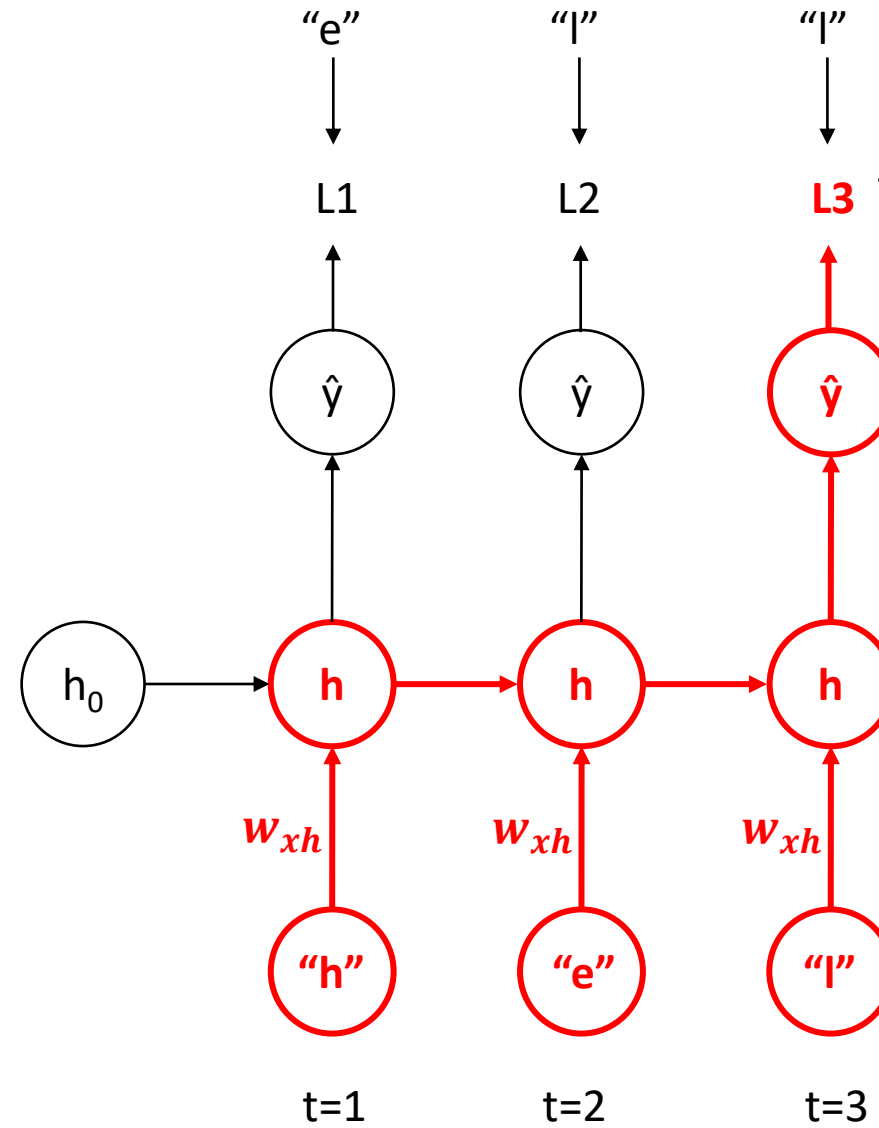
[Part 2: Softmax Regression \(saitcelebi.com\)](http://saitcelebi.com)

[How to compute the derivative of softmax and cross-entropy – Charlee Li](#)

# Recurrent Neural Networks

$$L_{total} = - \sum_{t=1}^n y_t \log(\hat{y})$$

$$L_t = -y_t \log(\hat{y})$$



$$\frac{\partial L3}{\partial W_{hh}} = \frac{\partial L3}{\partial \hat{Y}3} * \frac{\partial \hat{Y}3}{\partial H3} * \frac{\partial H3}{\partial W_{hx}} + \frac{\partial L3}{\partial \hat{Y}} * \frac{\partial \hat{Y}3}{\partial H3} * \frac{\partial H3}{\partial H2} * \frac{\partial H2}{\partial W_{hx}} + \frac{\partial L3}{\partial \hat{Y}} * \frac{\partial \hat{Y}3}{\partial H3} * \frac{\partial H3}{\partial H2} * \frac{\partial H2}{\partial H1} * \frac{\partial H1}{\partial W_{hx}}$$

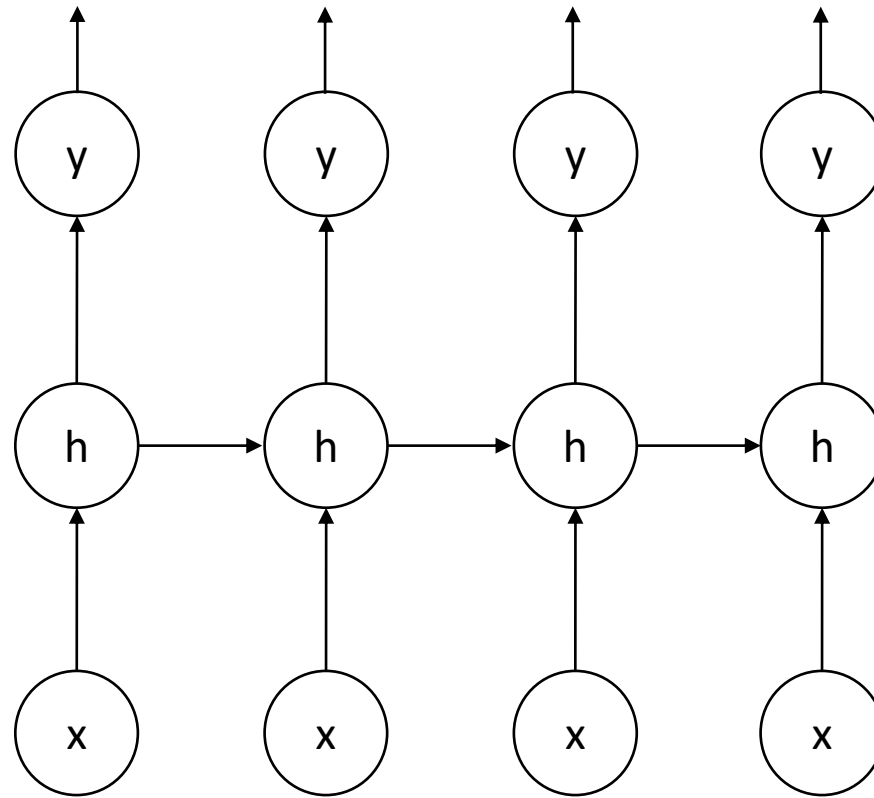
$$\frac{\partial LT}{\partial W_{xh}} = \sum_{t=1}^T \frac{\partial LT}{\partial \hat{Y}T} * \frac{\partial \hat{Y}T}{\partial H_t} * \frac{\partial H_t}{\partial W_{xh}}$$

For derivatives of Softmax and Cross Entropy Loss

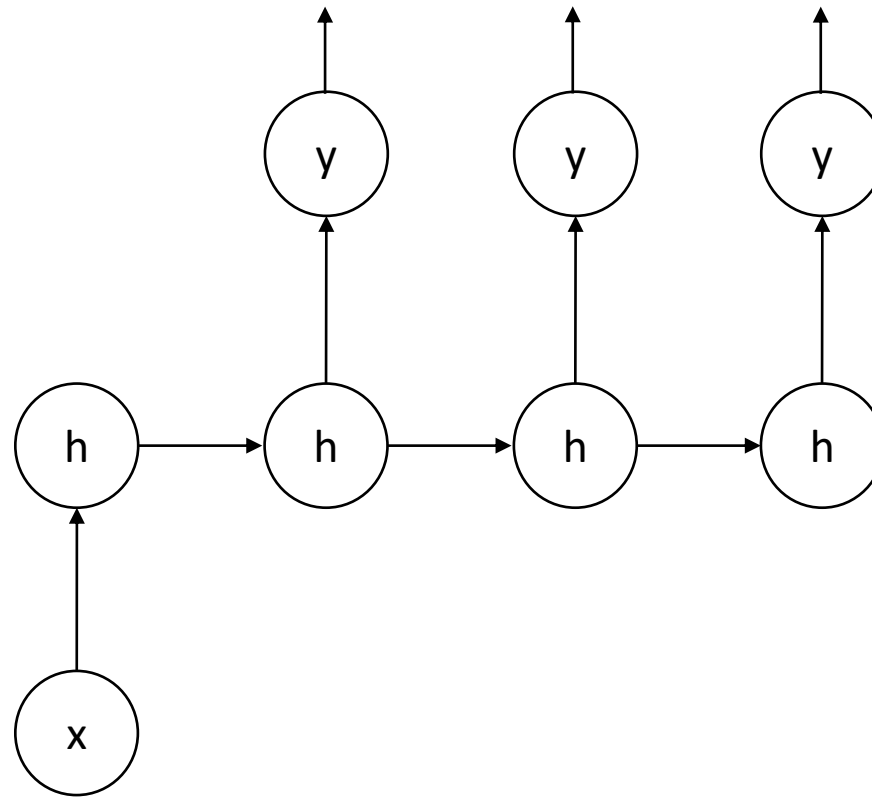
[Part 2: Softmax Regression \(saitcelebi.com\)](#)

[How to compute the derivative of softmax and cross-entropy – Charlee Li](#)

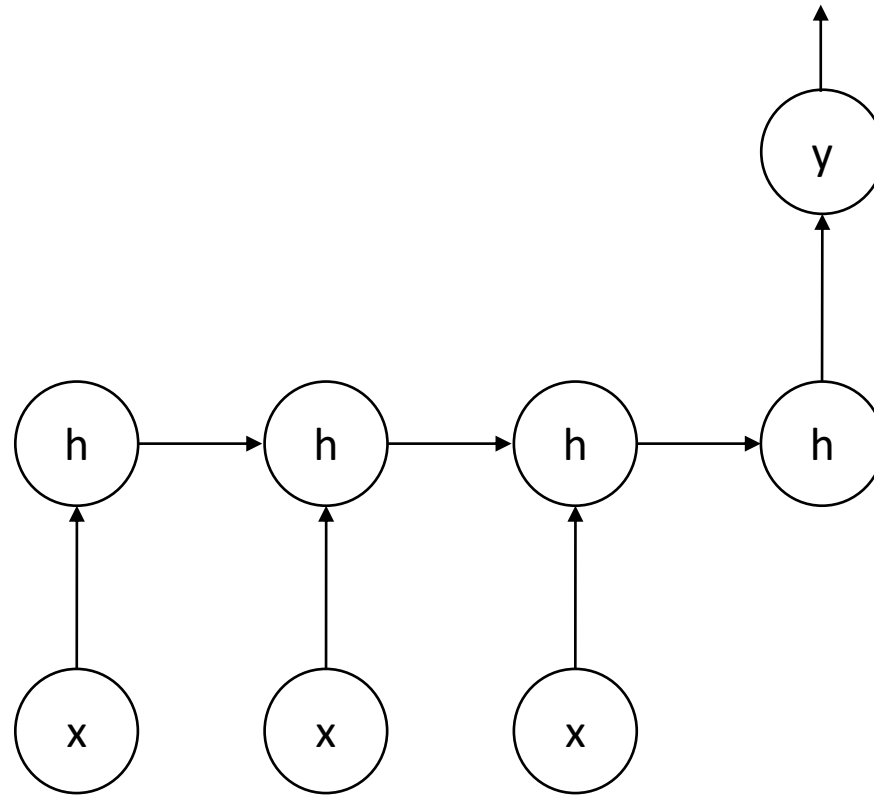
One-to-One



One-to-Many

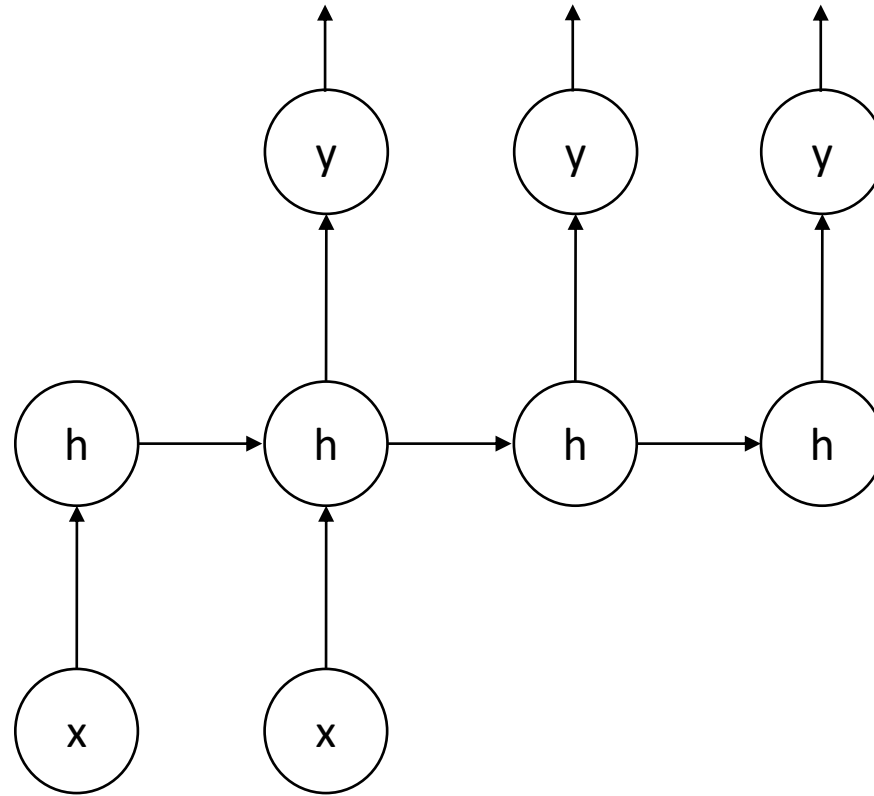


Many-to-One

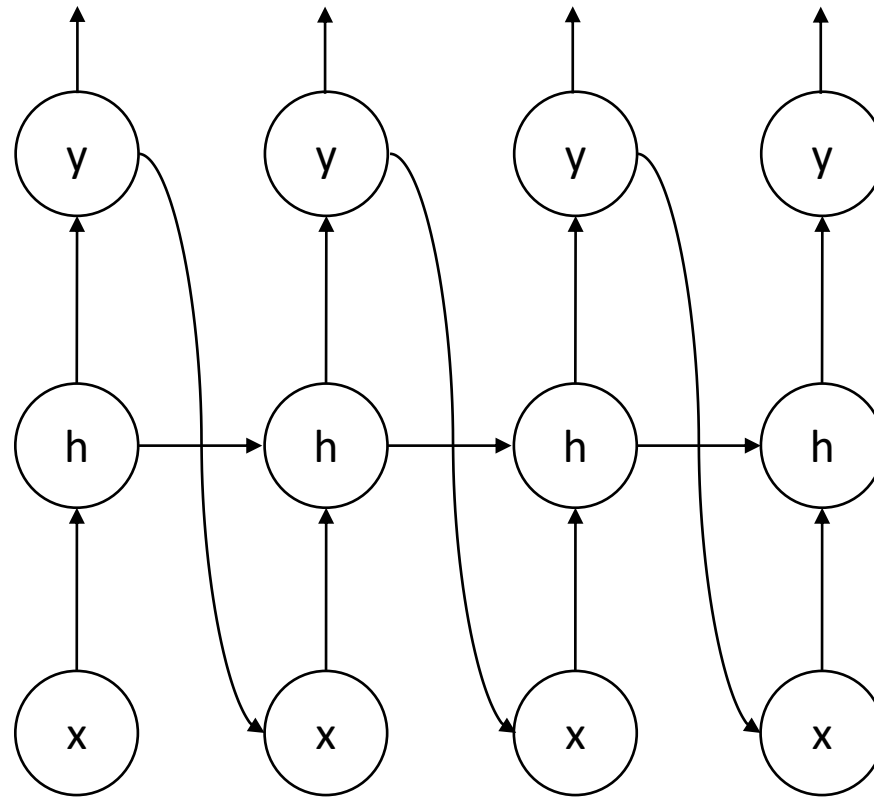




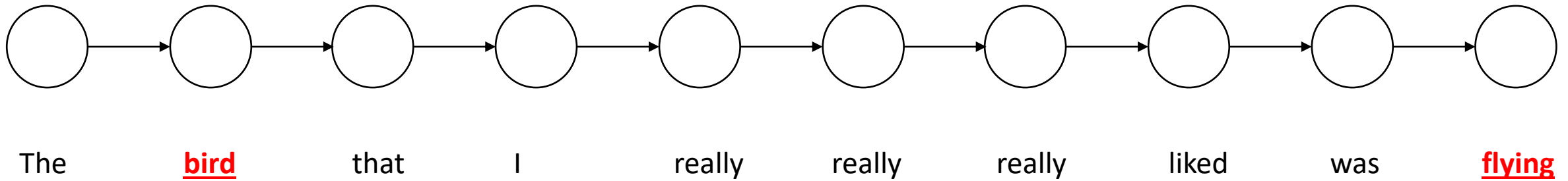
Many-to-Many



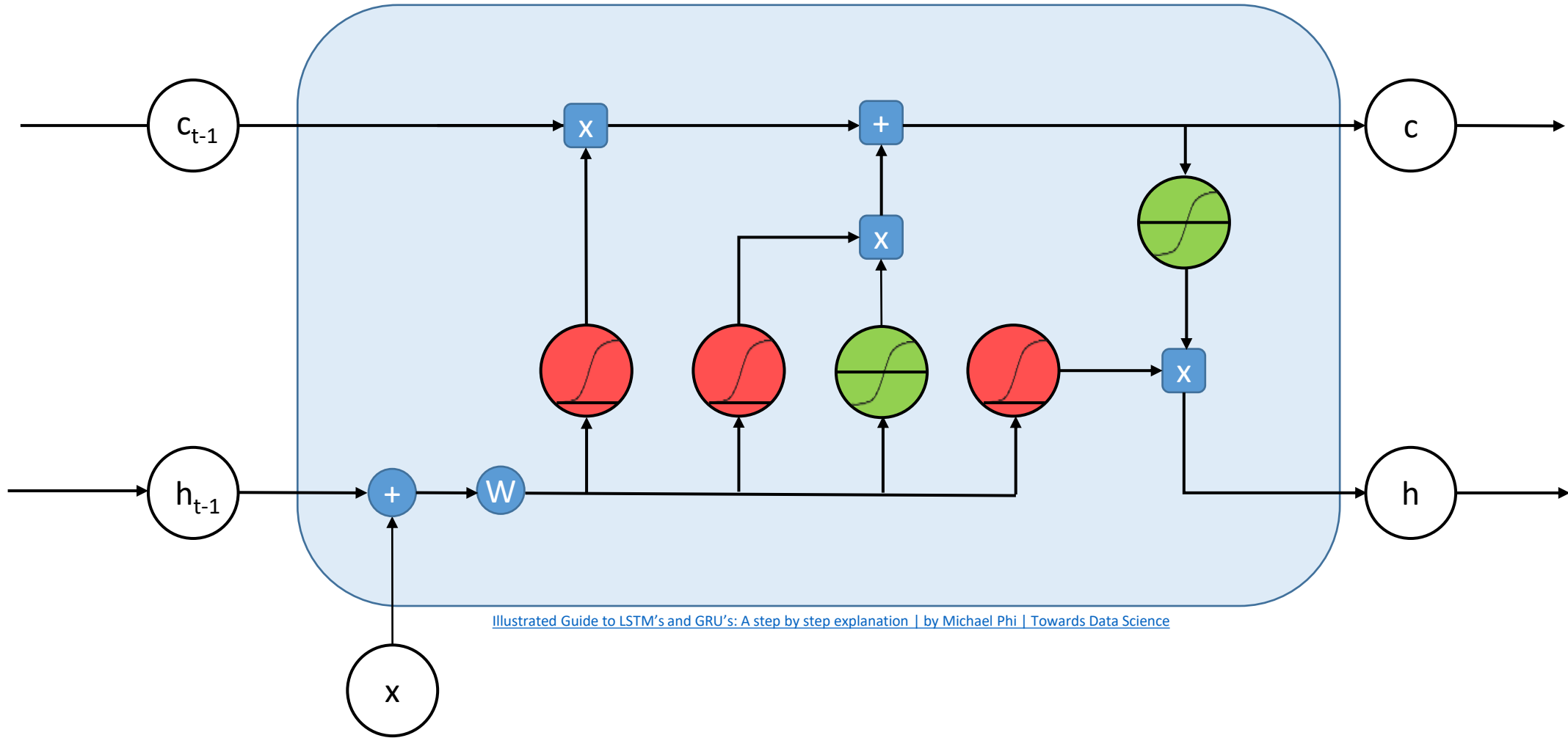
Generation



## Vanishing Gradients

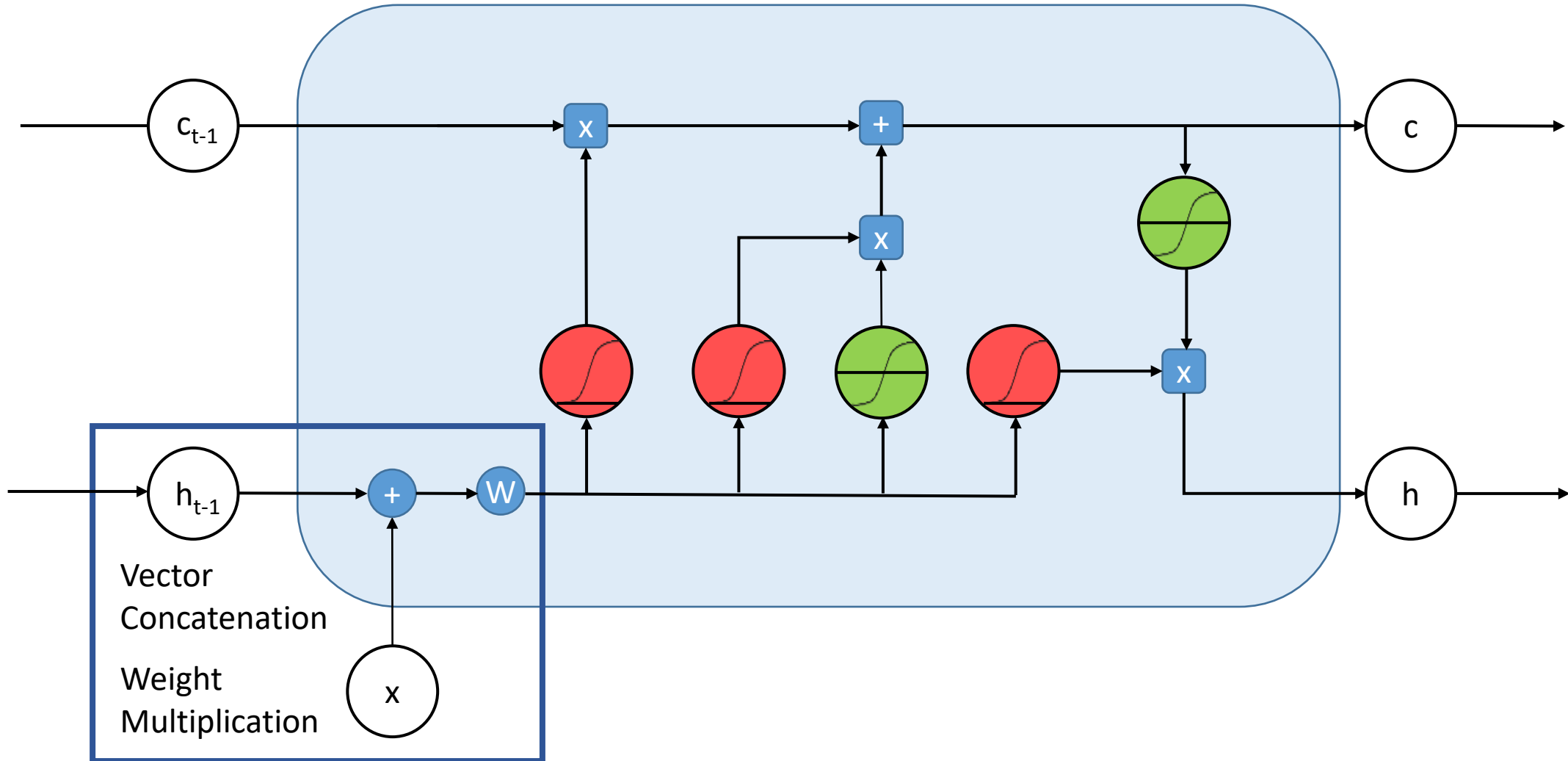


# Long Short Term Memory

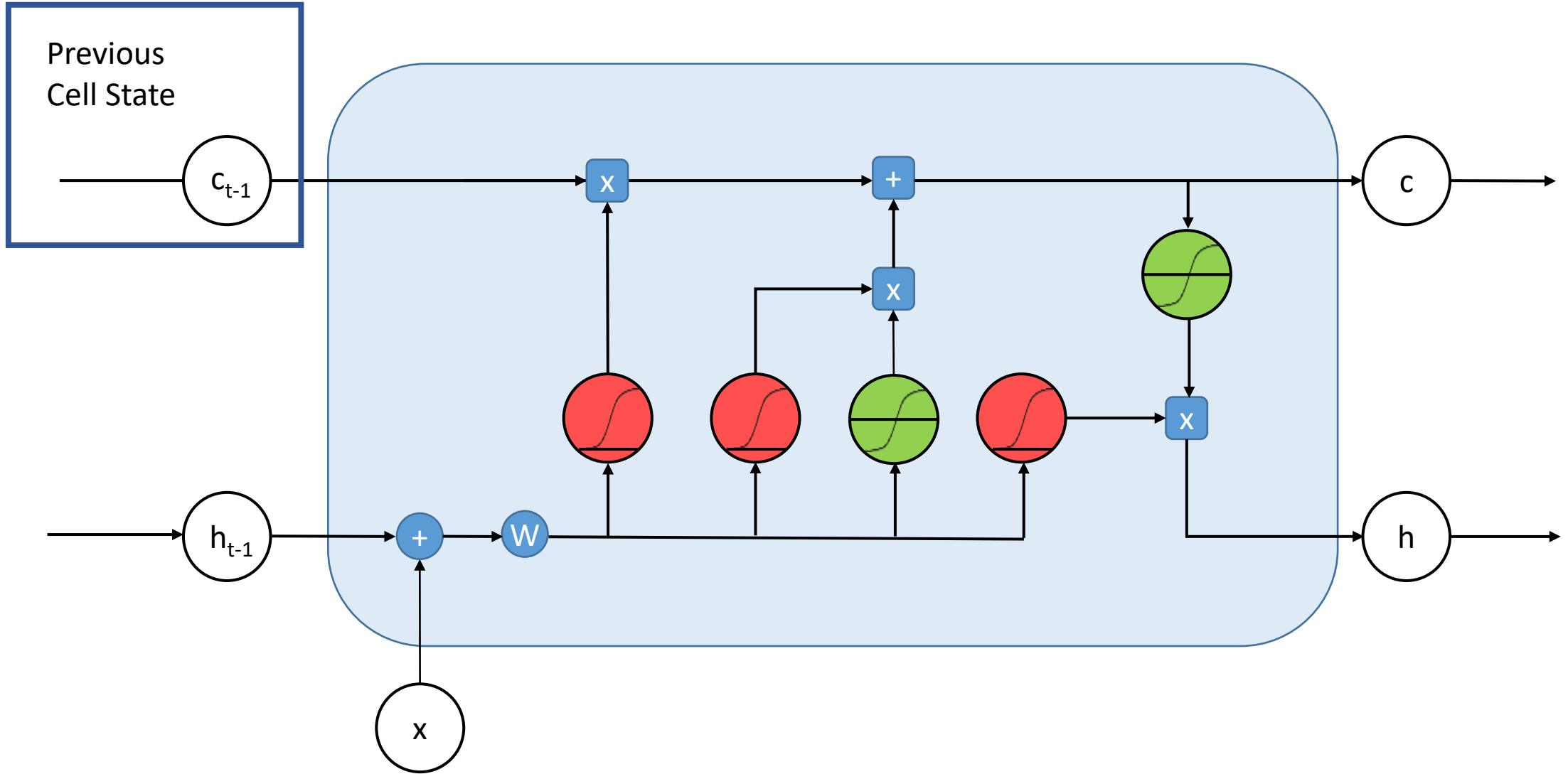


[Illustrated Guide to LSTM's and GRU's: A step by step explanation](#) | by Michael Phi | Towards Data Science

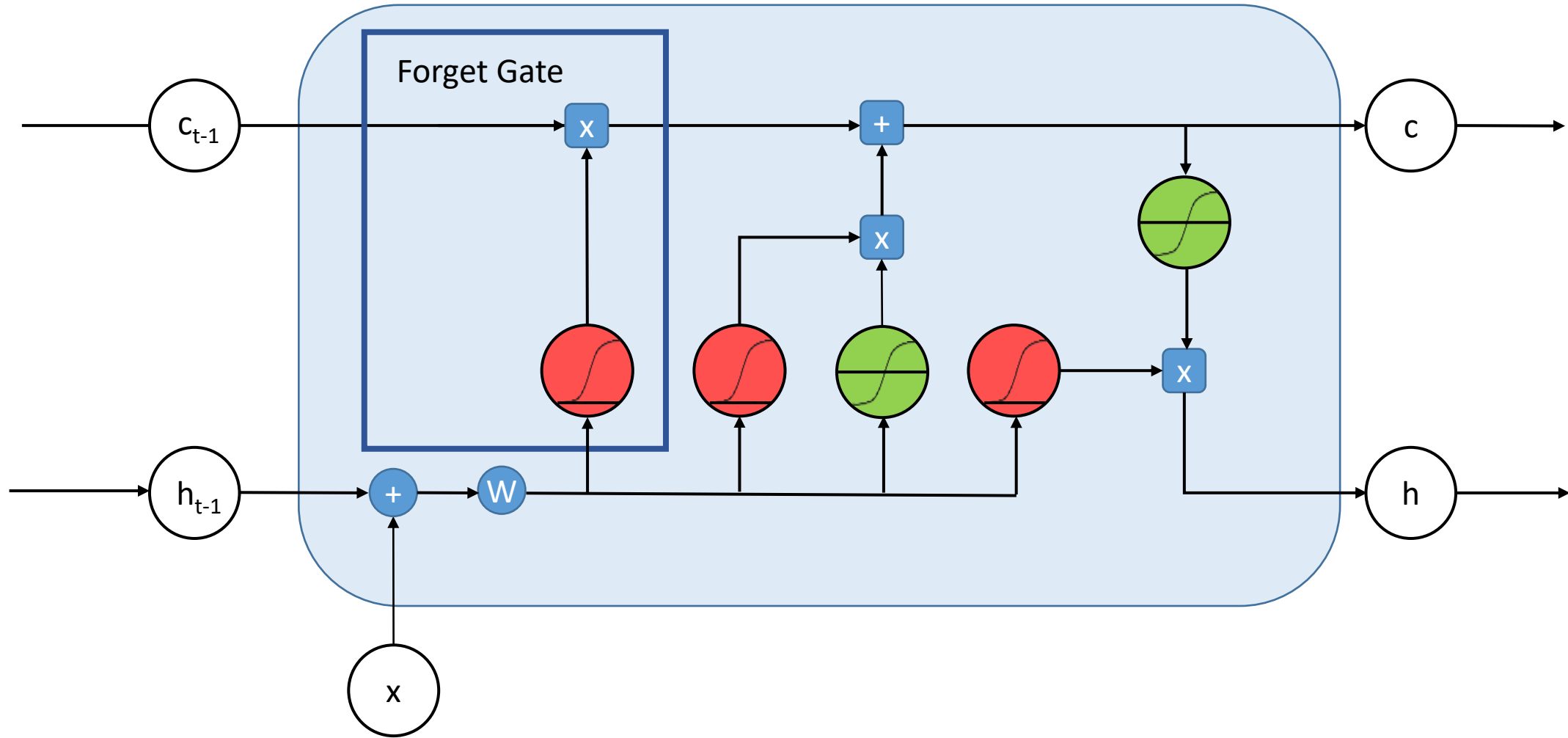
# Long Short Term Memory



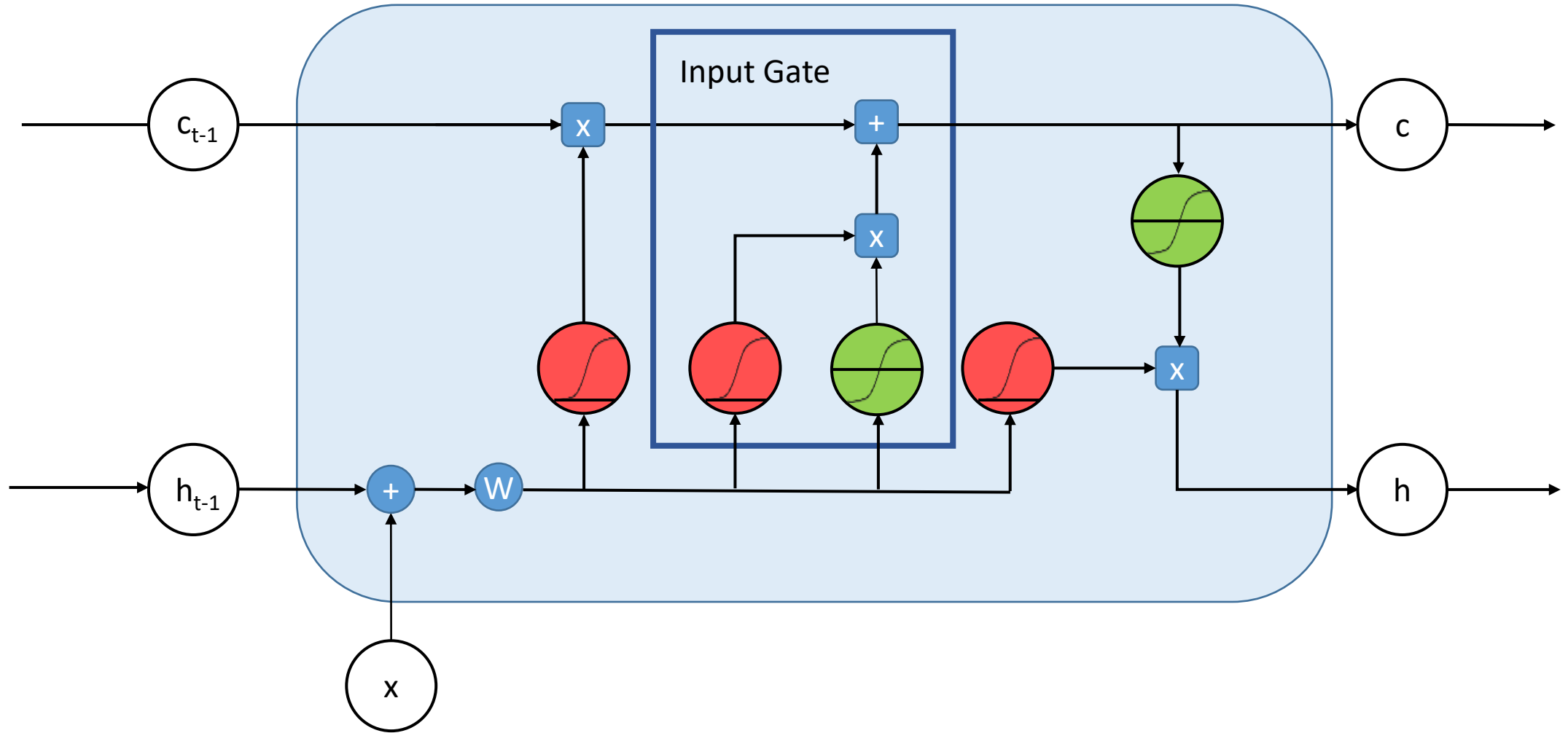
# Long Short Term Memory



# Long Short Term Memory

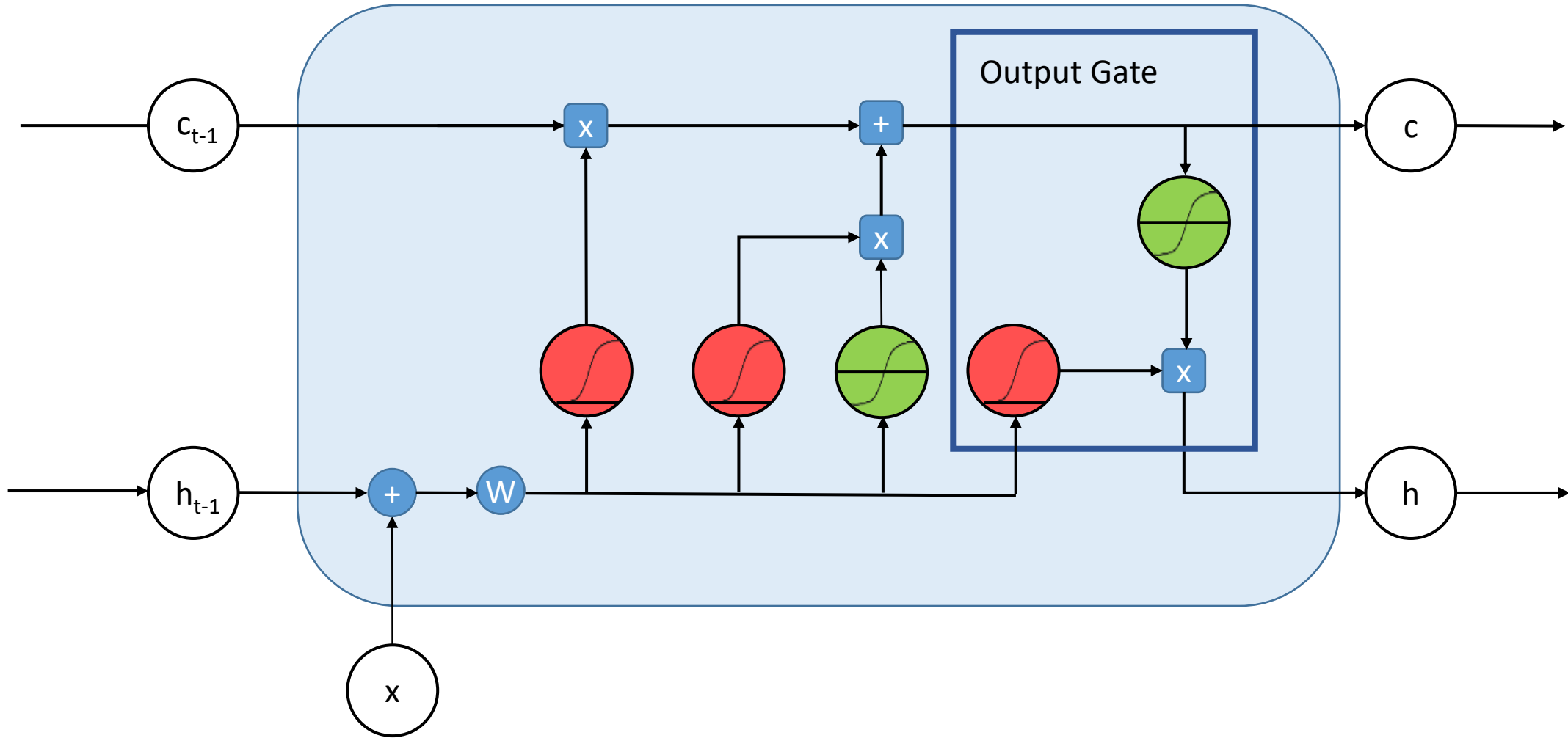


# Long Short Term Memory

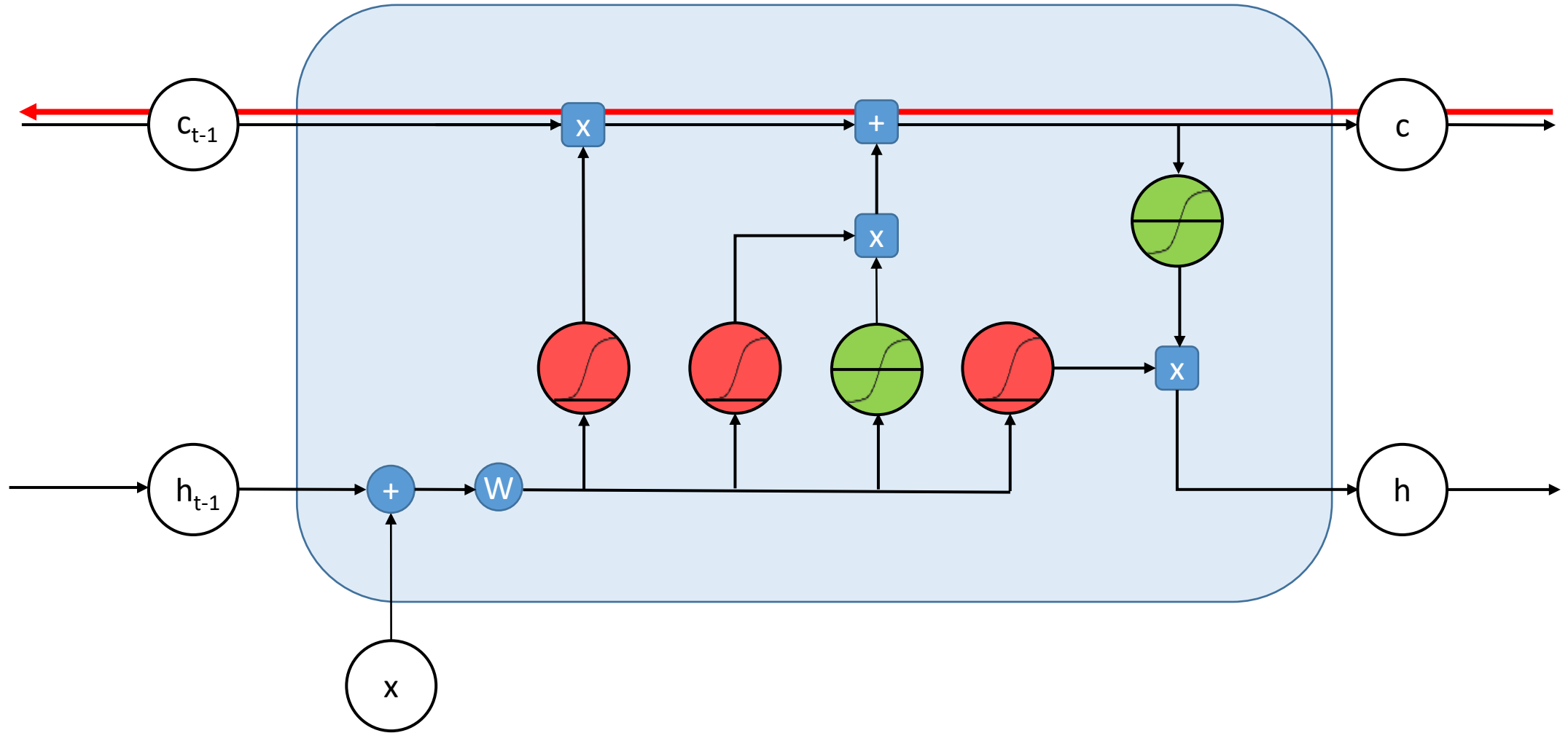




# Long Short Term Memory



# Long Short Term Memory



"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

quote detection cell

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

line length tracking cell