RECURRENT NEURAL NETWORKS

{Grady Kurpasi}

{SSIE 616}

{Prof H. Lewis}

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

Multilayer Perceptrons

x, y

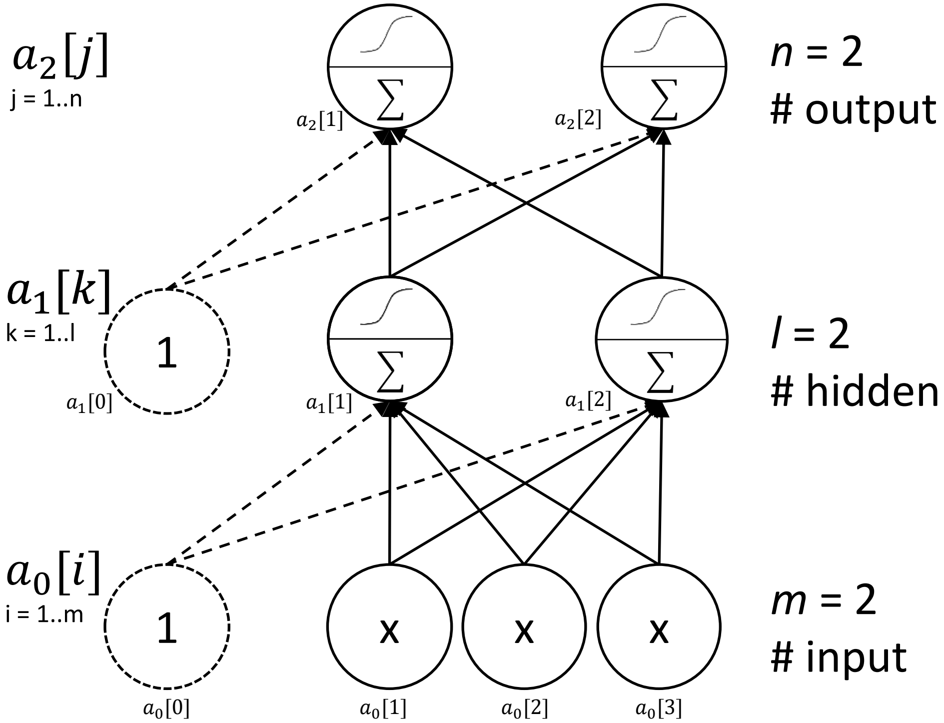
x_1	x_2	x_3	y_1	y_2
1.0000	4.0000	5.0000	0.1000	0.0500
0.1000	-5.0000	3.0000	0.1221	0.0964
6.0000	-5.5420	4.8970	0.1061	0.0702
4.0000	8.0000	9.0000	0.0996	0.0641
12.0000	-2.0000	0.0063	0.1110	0.0732
6.0000	-5.5000	4.8970	0.1060	0.0701

$w_1[i, k]$

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

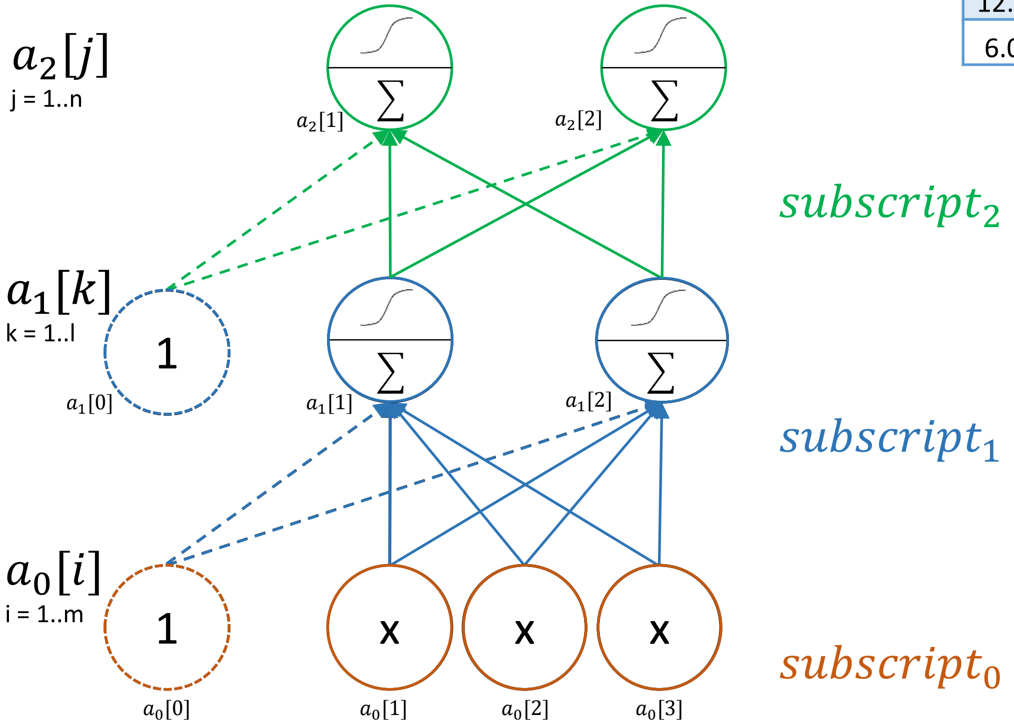
x_1	x_2	x_3	y_1	y_2
1.0000	4.0000	5.0000	0.1000	0.0500
0.1000	-5.0000	3.0000	0.1221	0.0964
6.0000	-5.5420	4.8970	0.1061	0.0702
4.0000	8.0000	9.0000	0.0996	0.0641
12.0000	-2.0000	0.0063	0.1110	0.0732
6.0000	-5.5000	4.8970	0.1060	0.0701

$w_1[i, k]$

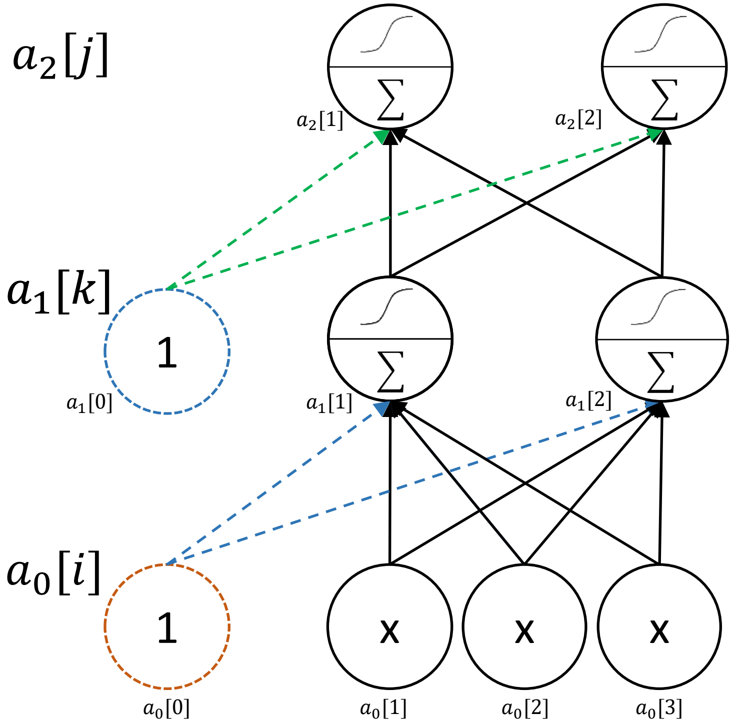
i	w[i,1]	w[i,2]
0	0.5	0.5
1	0.1	0.2
2	0.3	0.4
3	0.5	0.6

$w_2[k, j]$

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

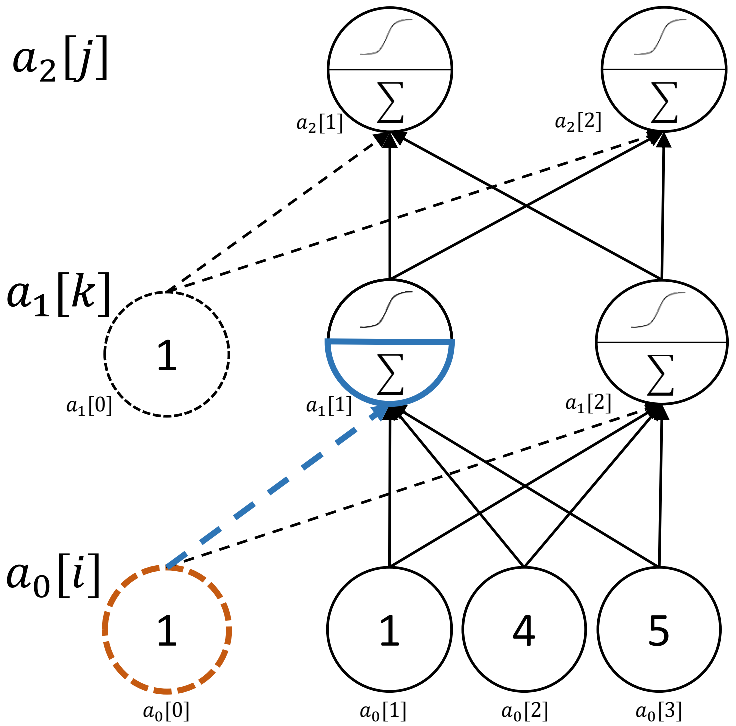
x_1	x_2	x_3	y_1	y_2
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i, k]$

i	w[i,1]	w[i,2]
0	**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



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

$$s_1[1] = w_1[0,1] * a_0[0] = .05 * 1$$

Forward Propagation

x, y

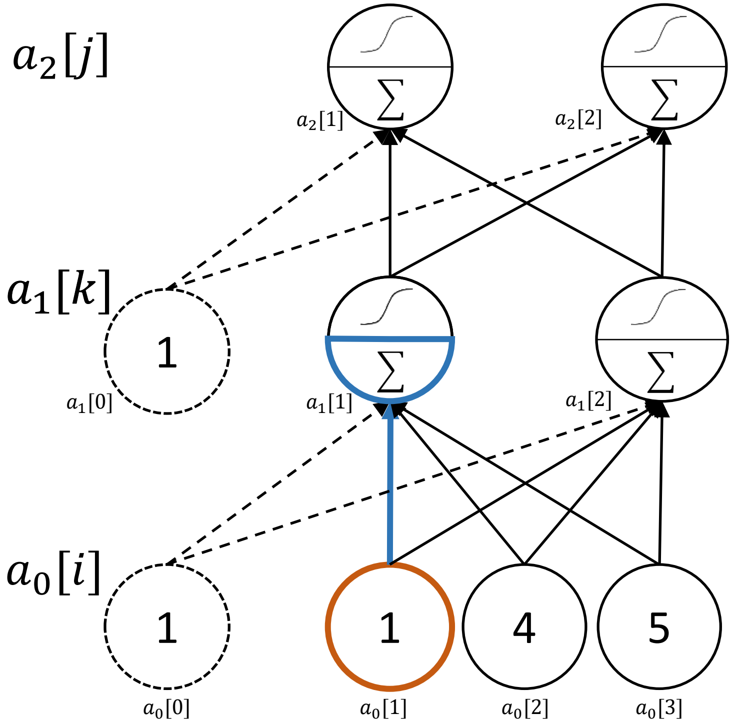
x_1	x_2	x_3	y_1	y_2
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i,k]$

i	w[i,1]	w[i,2]
0	0.5	0.5
1	**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



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

$$s_1[1] = \quad w_1[0,1] * a_0[0] = \quad .05 \; * \; 1$$

$$w_1[1,1] * a_0[1] = \quad .1 * 1$$

Forward Propagation

x, y

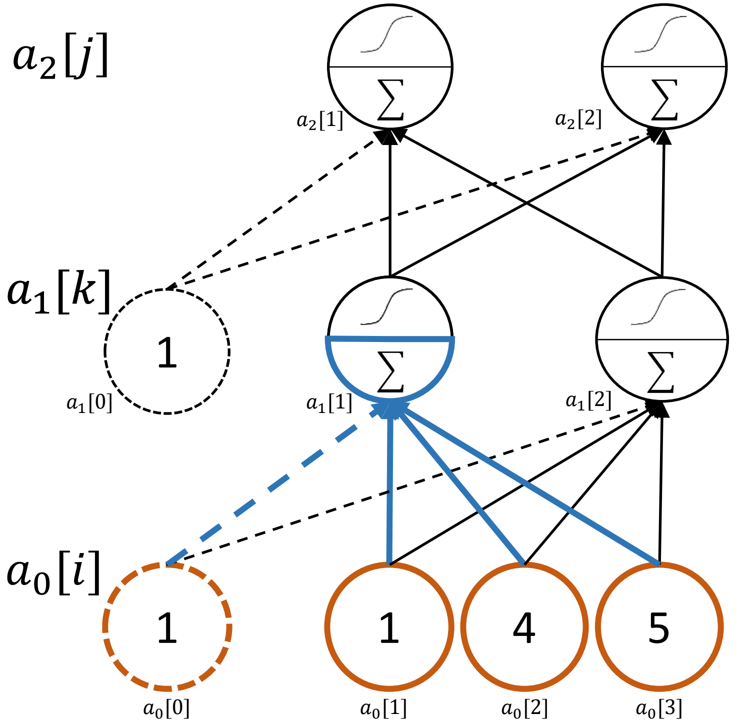
x_1	x_2	x_3	y_1	y_2
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i, k]$

i	w[i,1]	w[i,2]
0	**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



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

$$\begin{aligned} s_1[1] = \quad & w_1[0,1] * a_0[0] = \ .05 * 1 + \\ & w_1[1,1] * a_0[1] = \ .1 * 1 + \\ & w_1[2,1] * a_0[2] = \ .3 * 4 + \\ & w_1[3,1] * a_0[3] = \ .5 * 5 \\ & \qquad\qquad\qquad\ = \ 4.3 \end{aligned}$$

Forward Propagation

x, y

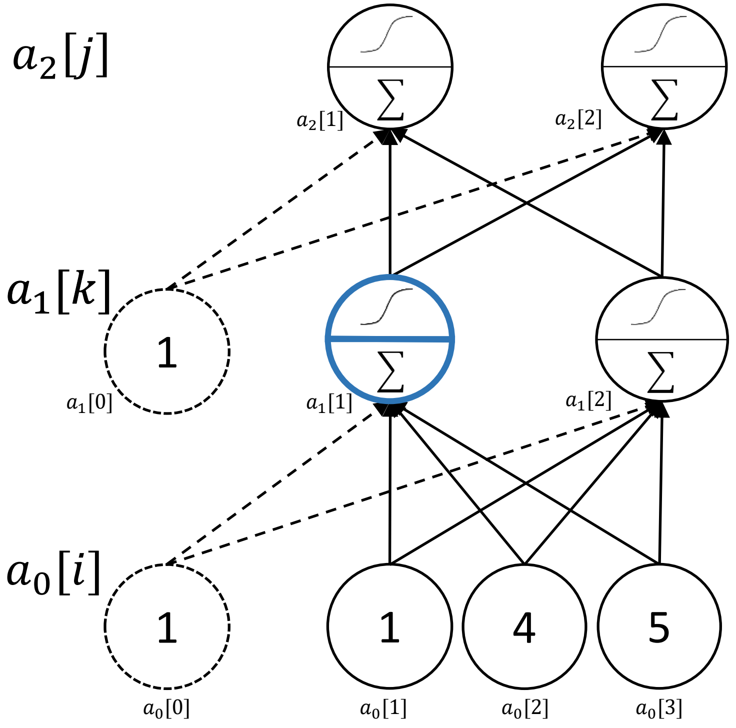
x_1	x_2	x_3	y_1	y_2
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i,k]$

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

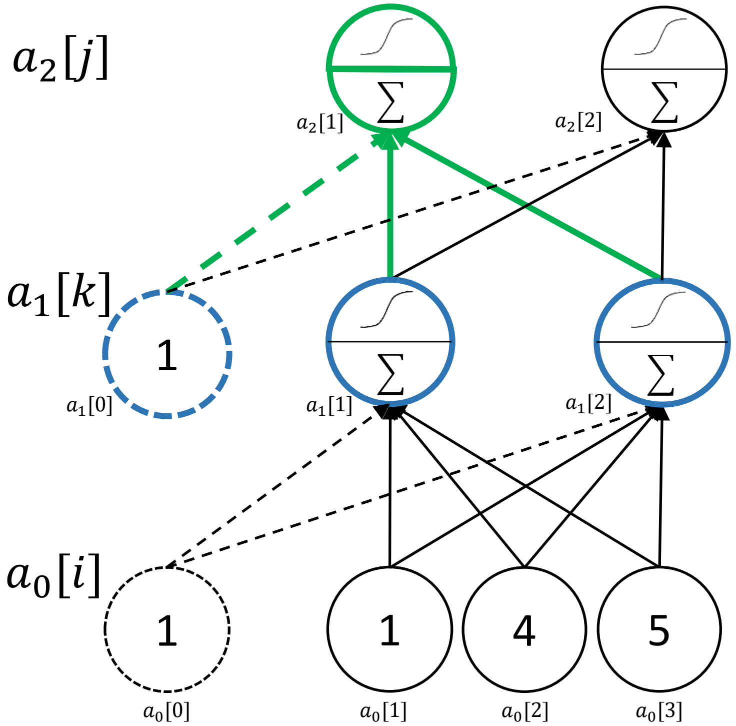
x_1	x_2	x_3	y_1	y_2
1.0000	4.0000	5.0000	0.1000	0.0500

$w_1[i, k]$

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



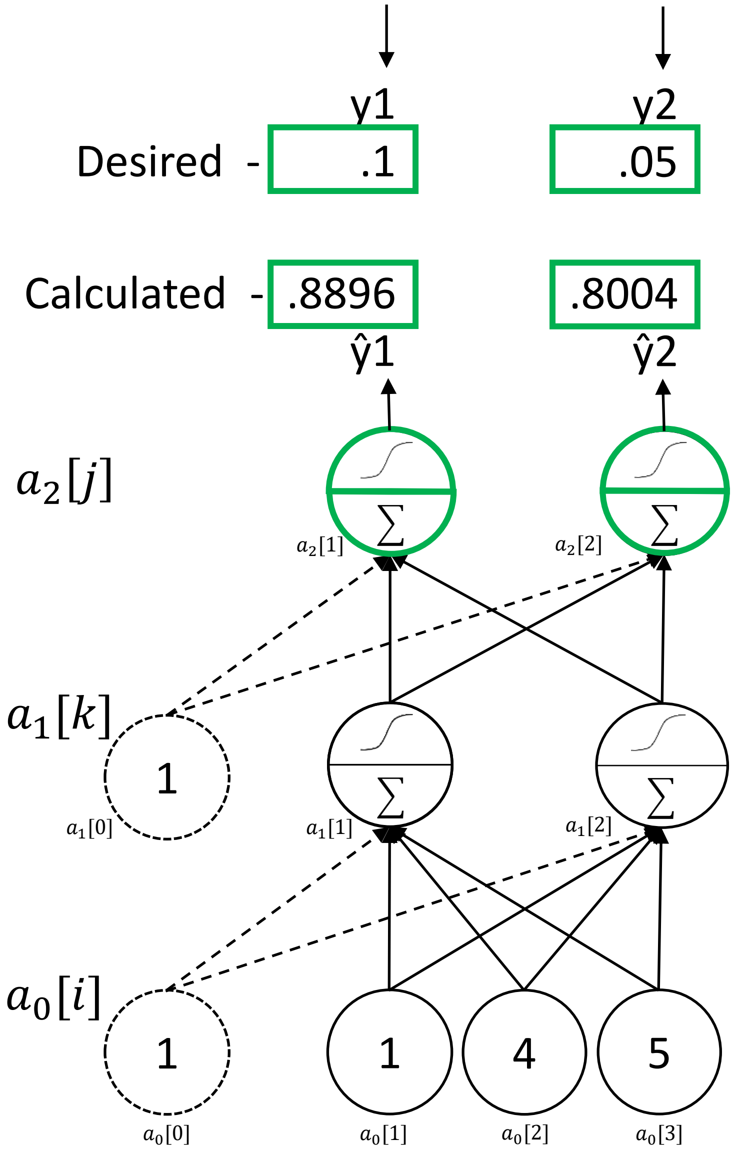
$$s_2[j] = \sum_{k=0}^{l} w_2[k, j] * a_1 i \quad \text{for j} = 1..\text{m}$$

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

$$\begin{aligned} s_2[1] = \quad & w_2[0,2] * a_1[0] = .5 * 1 \\ & w_2[1,2] * a_1[1] = .7 * .9866 \\ & w_2[2,2] * a_1[2] = .9 * .9950 \\ & \qquad\qquad\qquad = 2.0862 \end{aligned}$$

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

Error



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

Loss:

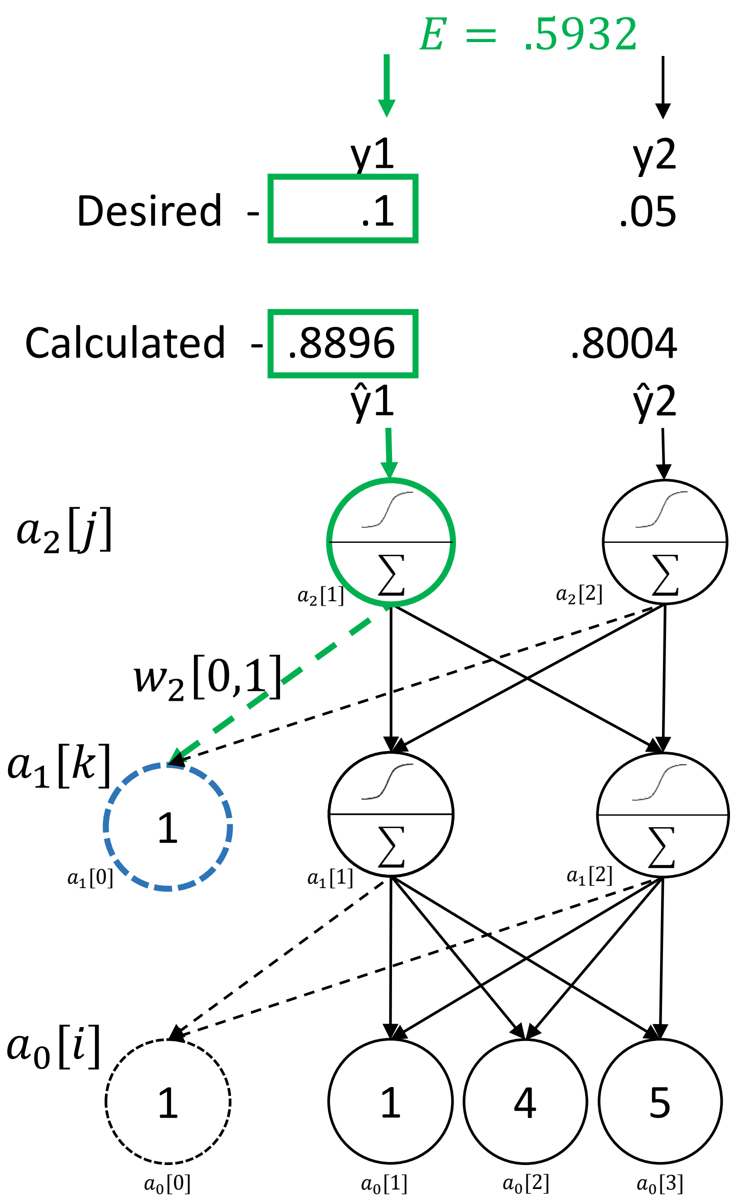
Mean Squared Error

$$E = \frac{1}{2}\sum_{j=1}^{n}(a_2[j] - y_r[j])^2$$

$$= \frac{1}{2}\left((.8896 - .1)^2 + (.8004 - .5)^2\right)$$

$$= .5932$$

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

$$E = \frac{1}{2}(\hat{y} - y1)^2 \longrightarrow \mathrm{E}(\hat{y})$$

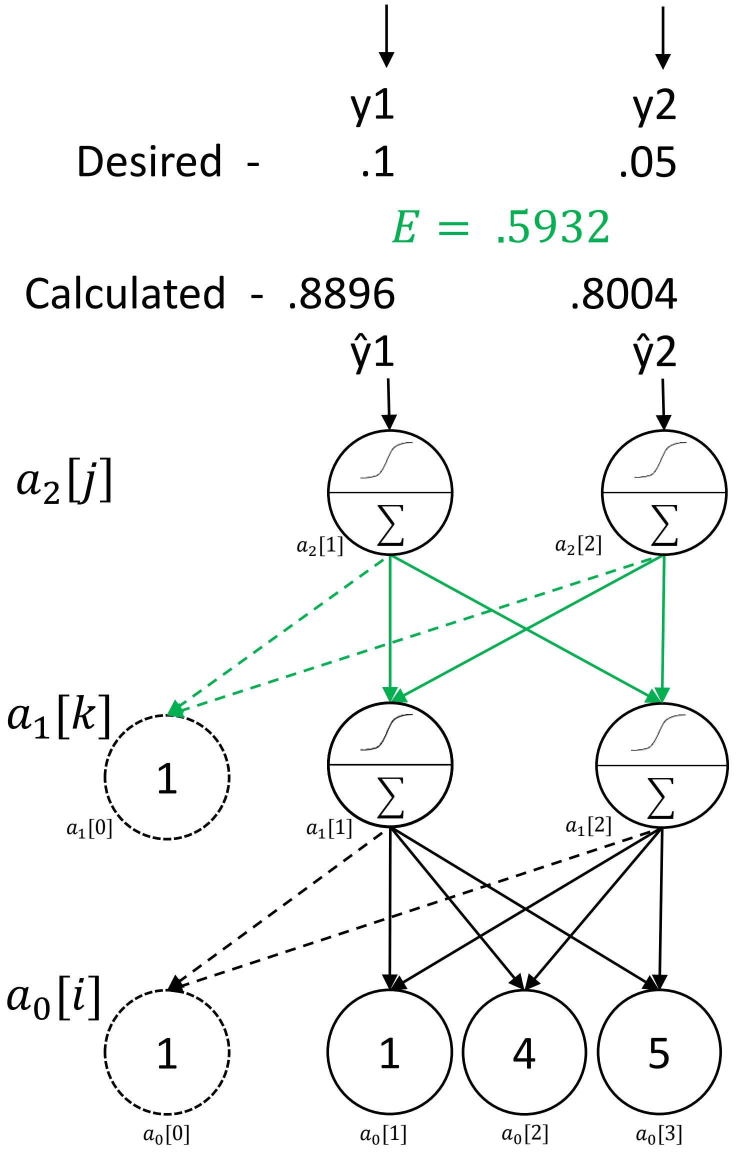
$$\hat{y} = \sigma(s_2) \longrightarrow \mathrm{f}(s_2)$$

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

Chain Rule

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

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

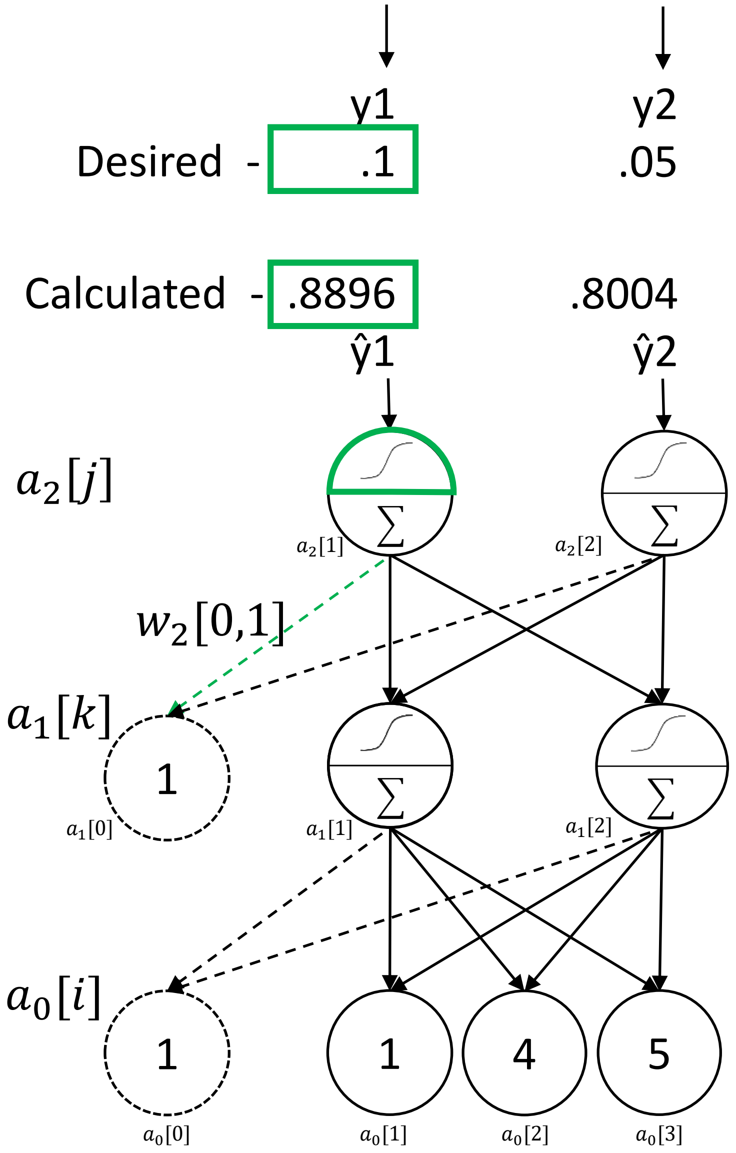
$w_2[k,j]$

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

$$\frac{\partial E}{\partial w_2[k,j]} = \frac{\partial E}{\partial a_2[j]} * \frac{\partial a_2[j]}{\partial s_2[j]} * \frac{\partial s_2[j]}{\partial w_2[k,j]}$$

$$= \{\hat{y}_j - y_k\} * \{a_2[j](1 - a_2[j])\} * \{a_1[k]\}$$

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

$w_2[k,j]$

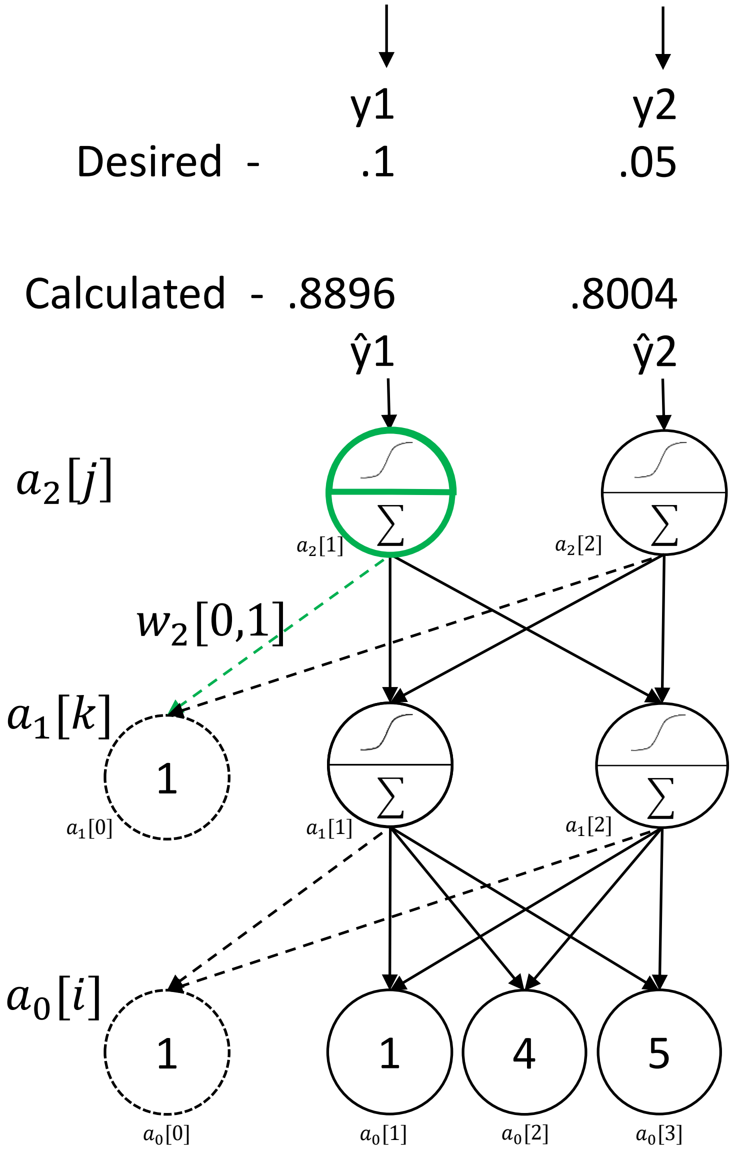
k	w[k,1]	w[k,2]
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

$$\frac{\partial E}{\partial w_2[0,1]} = \frac{\partial E}{\partial a_2[1]} * \frac{\partial a_2[1]}{\partial s_2[1]} * \frac{\partial s_2[1]}{\partial w_2[0,1]}$$

$$= \{\hat{y}_j - y_k\} * \{a_2[1](1 - a_2[1])\} * \{a_1[0]\}$$

$$= .8896 - .1$$

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

$w_2[k,j]$

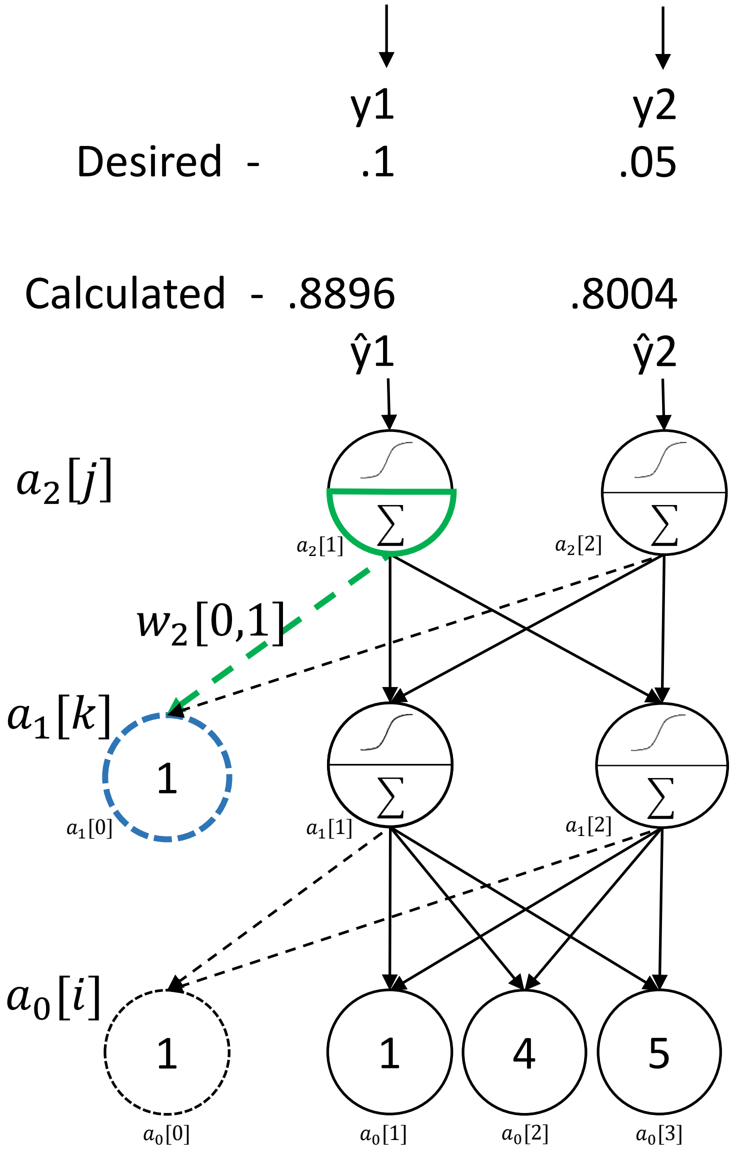
k	w[k,1]	w[k,2]
0	0.5	0.5
1	0.7	0.8
2	0.9	0.1

$$\frac{\partial E}{\partial w_2[0,1]} = \frac{\partial E}{\partial a_2[1]} * \frac{\partial a_2[1]}{\partial s_2[1]} * \frac{\partial s_2[1]}{\partial w_2[0,1]}$$

$$= \{\hat{y}_j - y_k\} * \{a_2[1](1 - a_2[1])\} * \{a_1[0]\}$$

$$= .8896 - .1 * .8896(1 - .8896)$$

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

$$\frac{\partial E}{\partial w_2[0,1]} = \frac{\partial E}{\partial a_2[1]} * \frac{\partial a_2[1]}{\partial s_2[1]} * \frac{\partial s_2[1]}{\partial w_2[0,1]}$$

$$= \{\hat{y}_j - y_k\} * \{a_2[1](1 - a_2[1])\} * \{a_1[0]\}$$

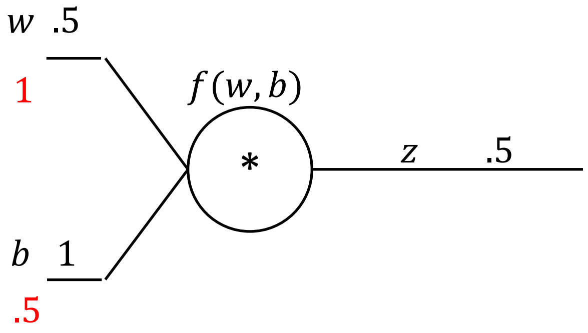
$$= .8896 - .1 * .8896(1 - .8896) * 1$$

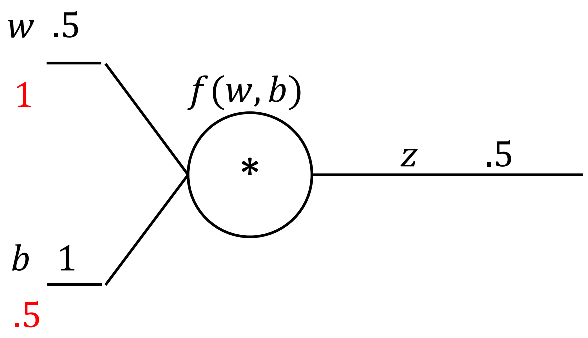
$$= .0775$$

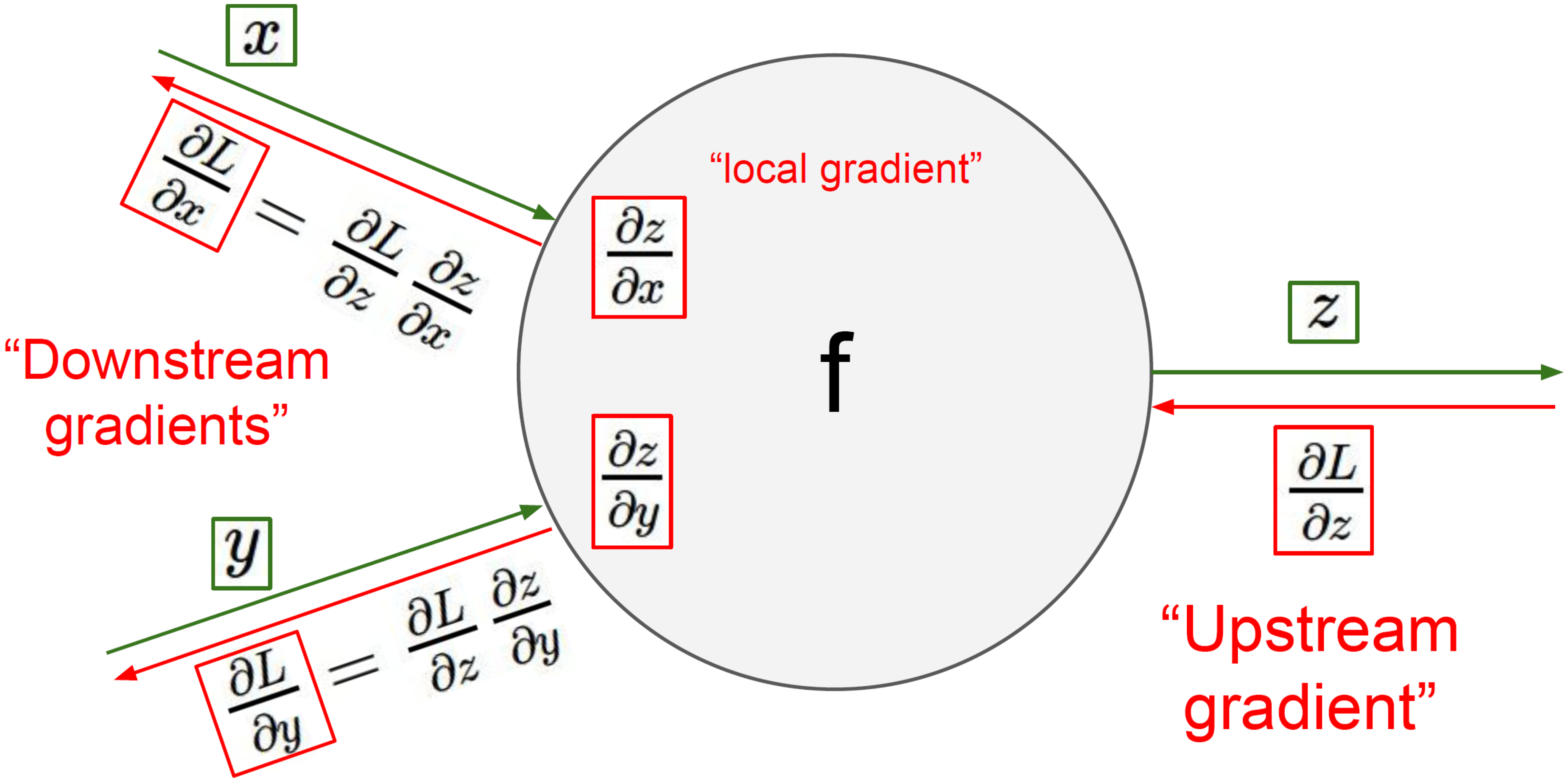
$c_2[k,j]$

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

Computational Graphs

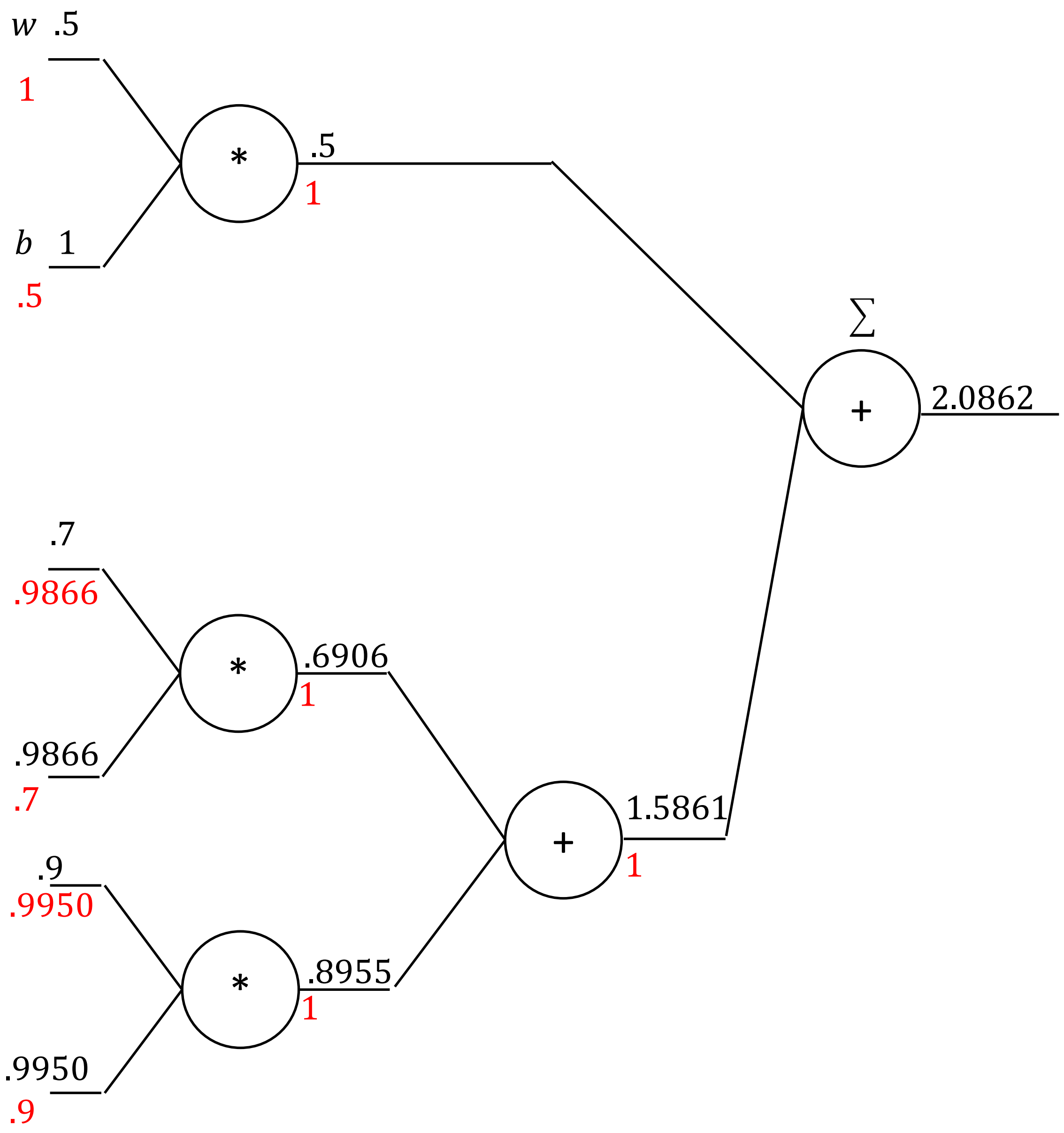






cs231n_2019_lecture04.pdf (stanford.edu)

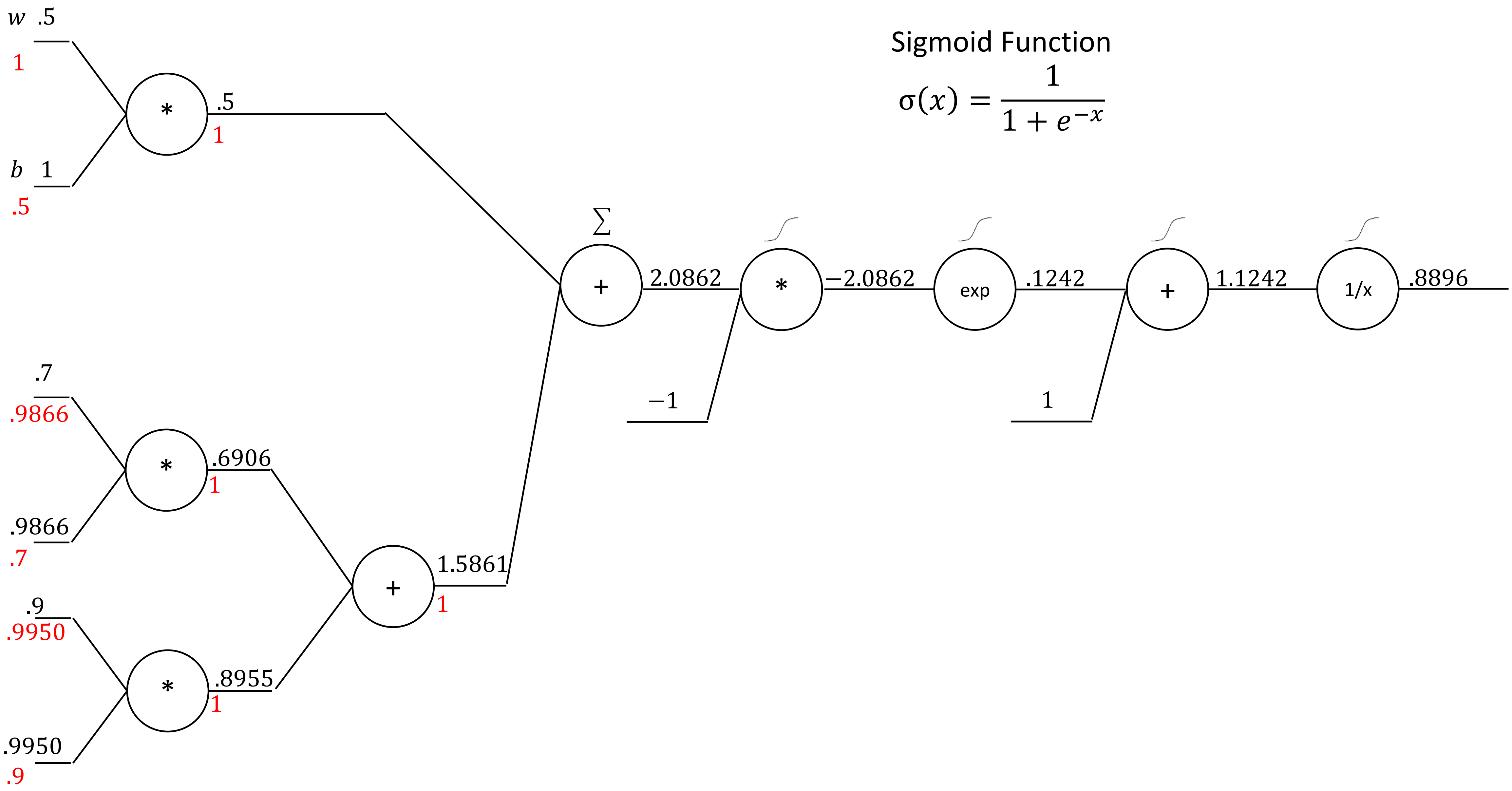
Computational Graphs



Computational Graphs

Sigmoid Function

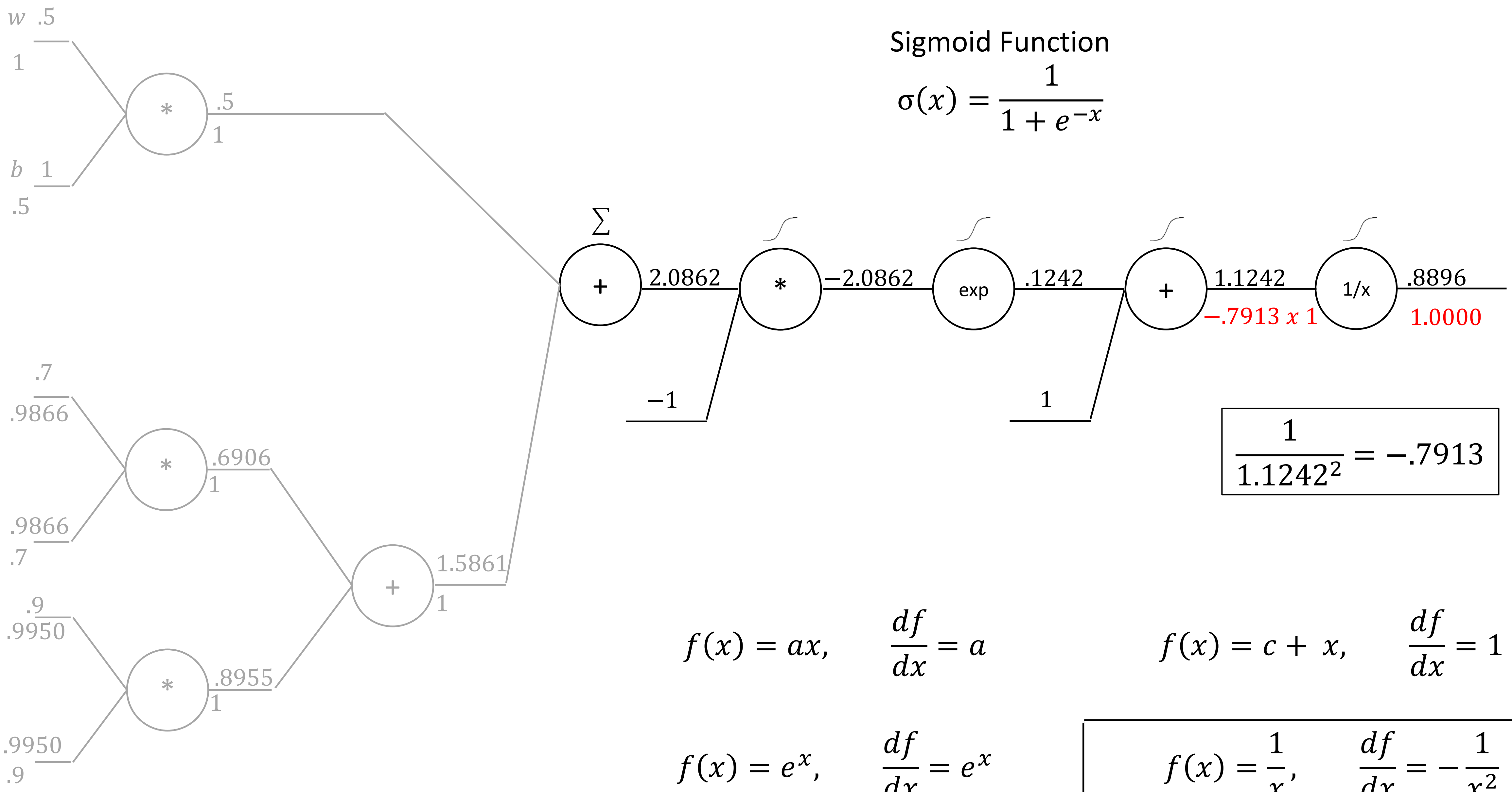
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Computational Graphs

Sigmoid Function

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



$$\boxed{\frac{1}{1.1242^2} = -.7913}$$

$$f(x) = ax, \quad \frac{df}{dx} = a$$

$$f(x) = c + x, \quad \frac{df}{dx} = 1$$

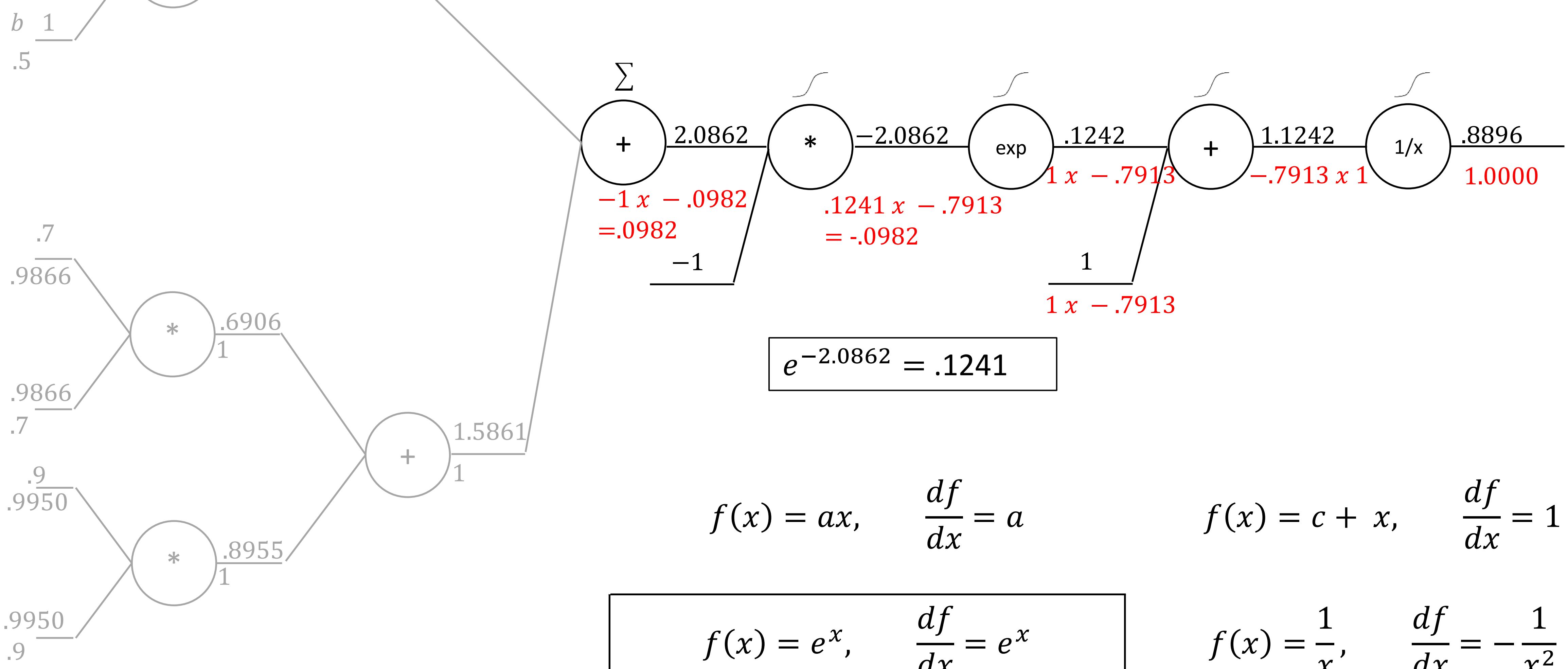
$$f(x) = e^x, \quad \frac{df}{dx} = e^x$$

$$\boxed{f(x) = \frac{1}{x}, \quad \frac{df}{dx} = -\frac{1}{x^2}}$$

Computational Graphs

Sigmoid Function

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



$$f(x) = ax, \qquad \frac{df}{dx} = a$$

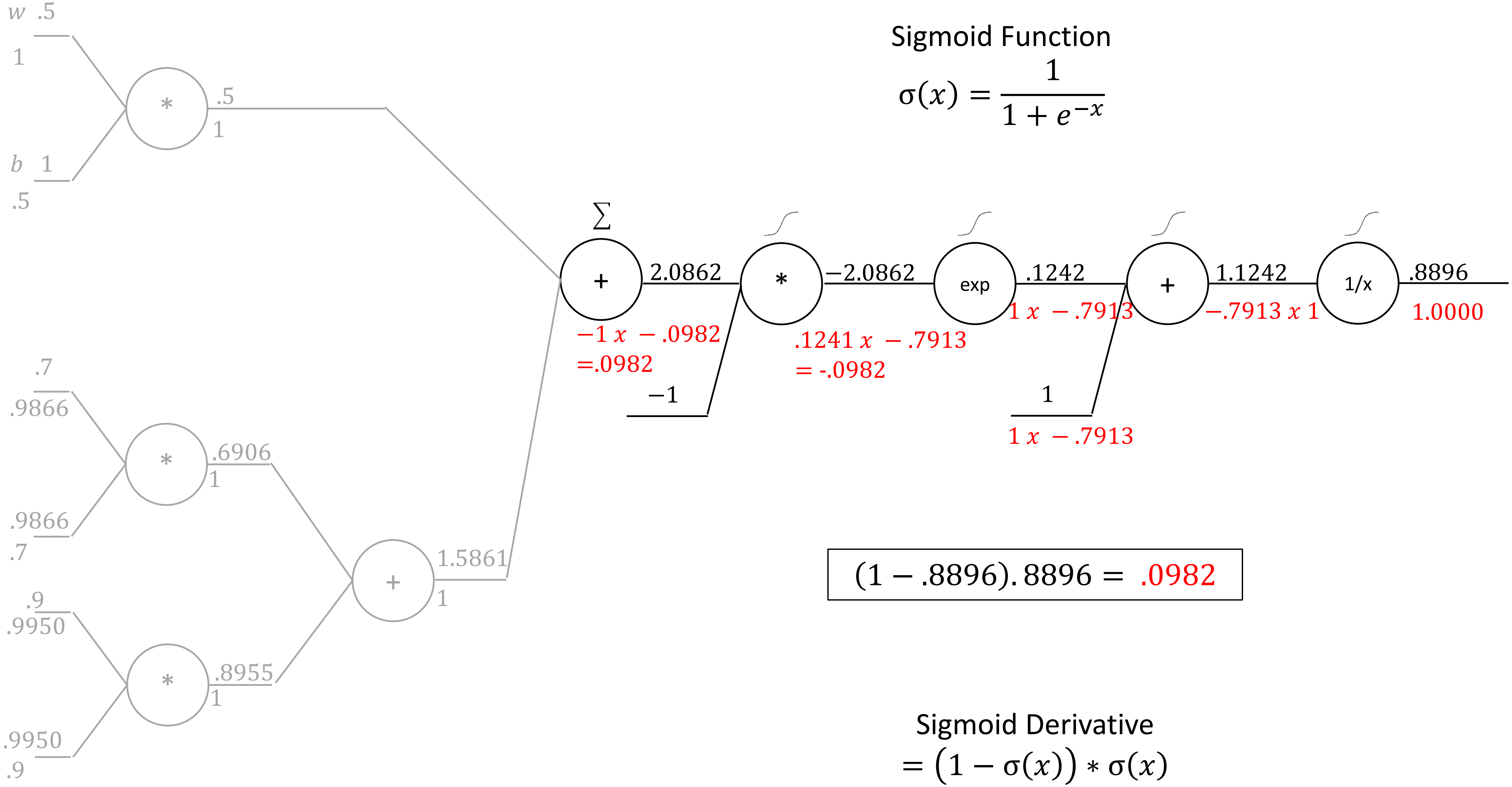
$$f(x) = c + x, \qquad \frac{df}{dx} = 1$$

$$\boxed{f(x) = e^x, \qquad \frac{df}{dx} = e^x}$$

$$f(x) = \frac{1}{x}, \qquad \frac{df}{dx} = -\frac{1}{x^2}$$

Sigmoid Function

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

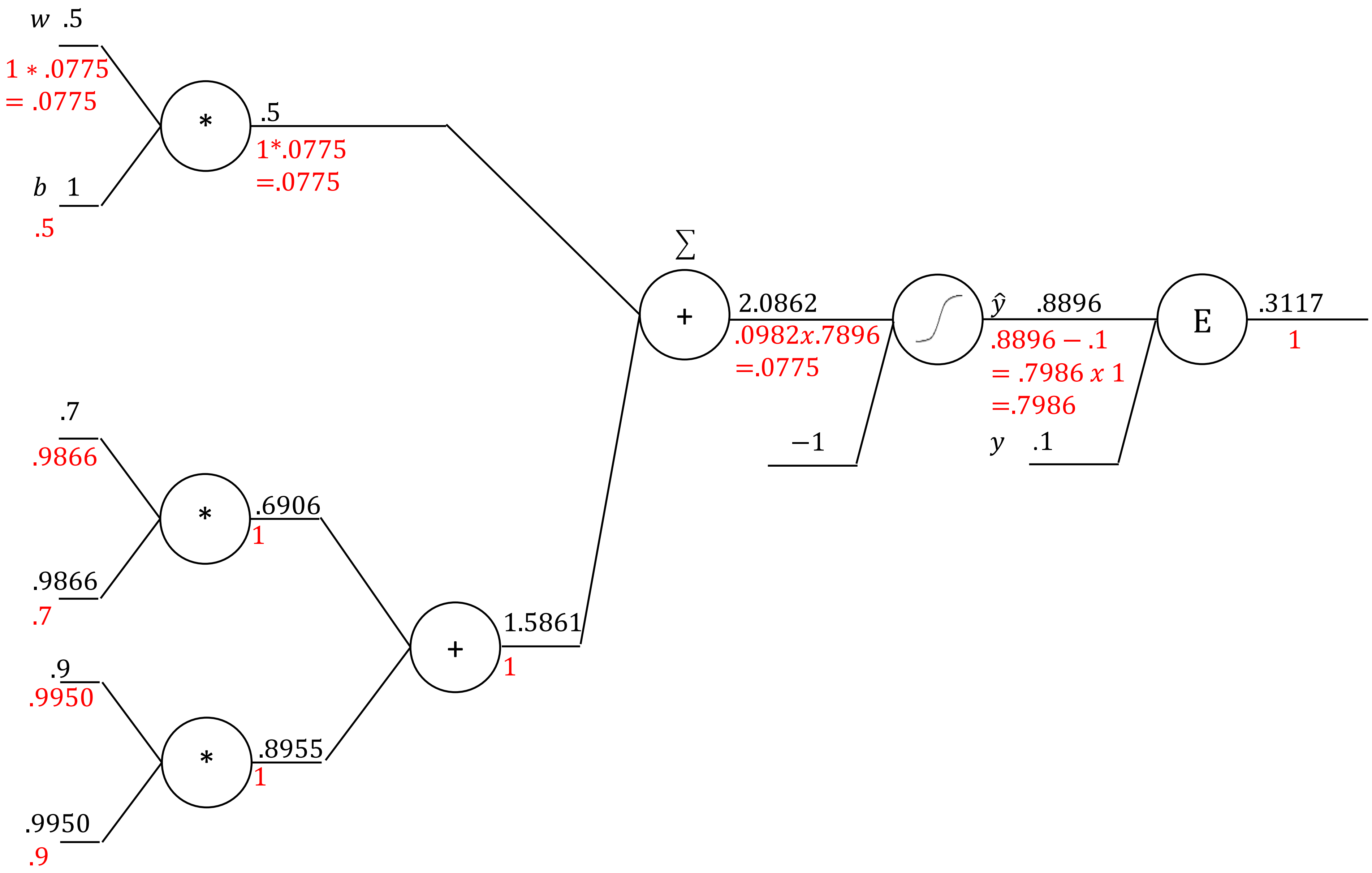


$$\boxed{(1 - .8896).8896 = .0982}$$

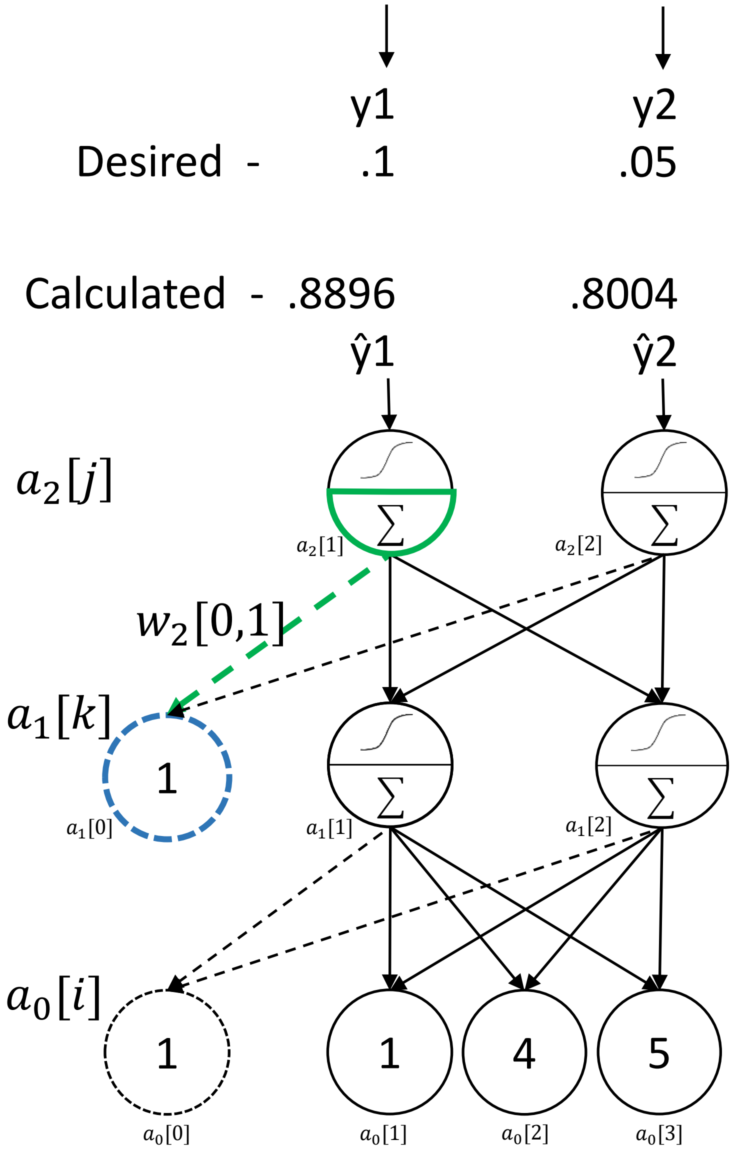
Sigmoid Derivative

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

Computational Graphs



Computational Graphs



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

$$\frac{\partial E}{\partial w_2[0,1]} = \frac{\partial E}{\partial a_2[1]} * \frac{\partial a_2[1]}{\partial s_2[1]} * \frac{\partial s_2[1]}{\partial w_2[0,1]}$$

$$= \{\hat{y}_j - y_k\} * \{a_2[1](1 - a_2[1])\} * \{a_1[0]\}$$

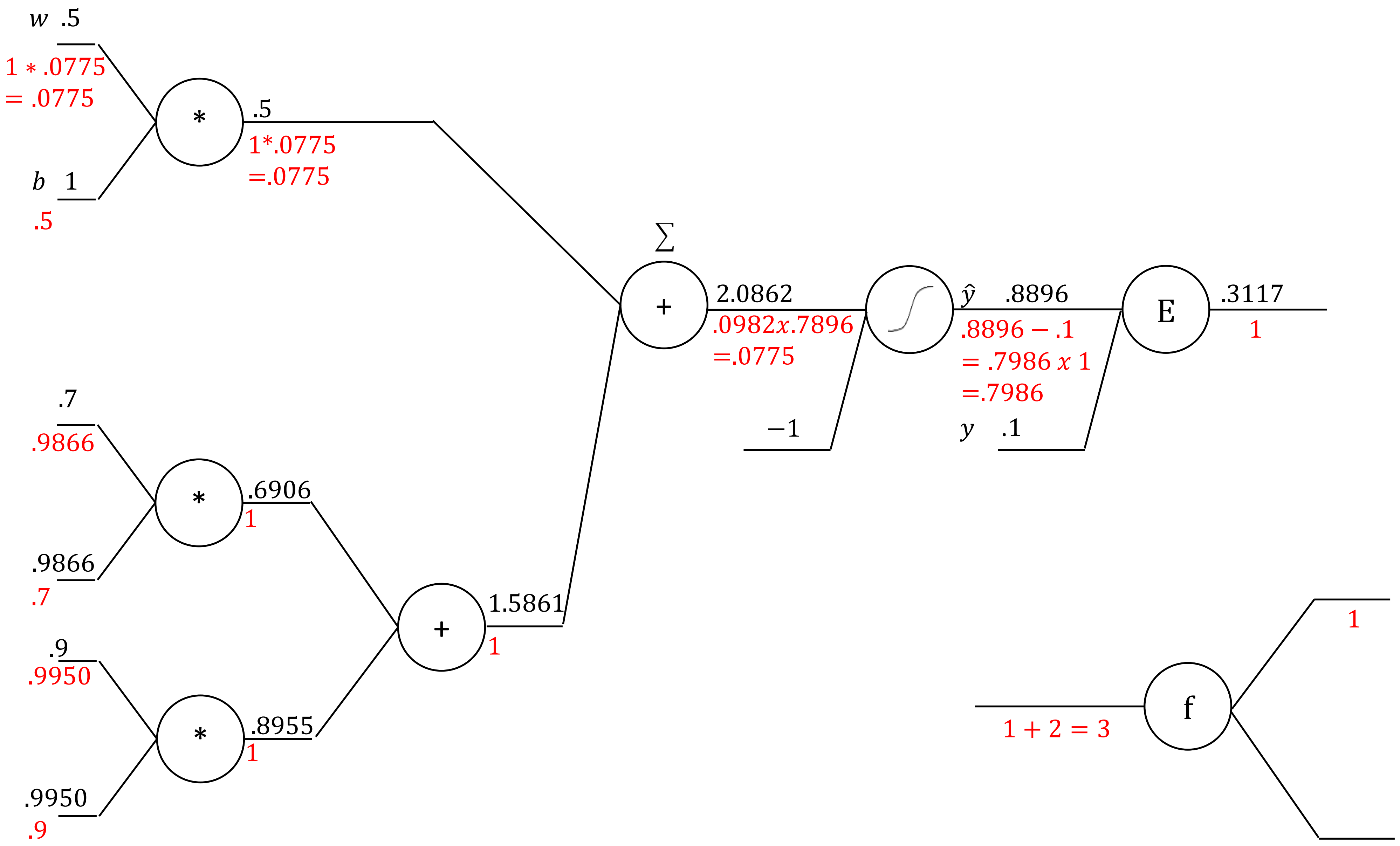
$$= .8896 - .1 * .8896(1 - .8896) * 1$$

$$= .0775$$

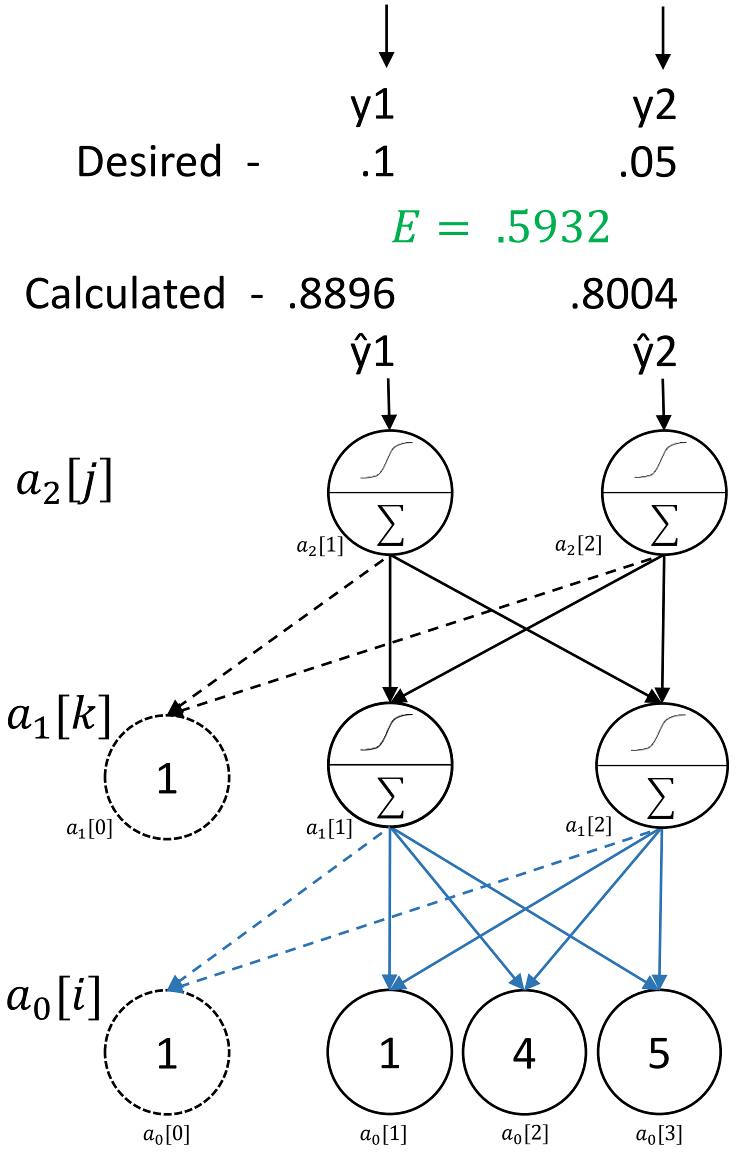
$c_2[k,j]$

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

Computational Graphs



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

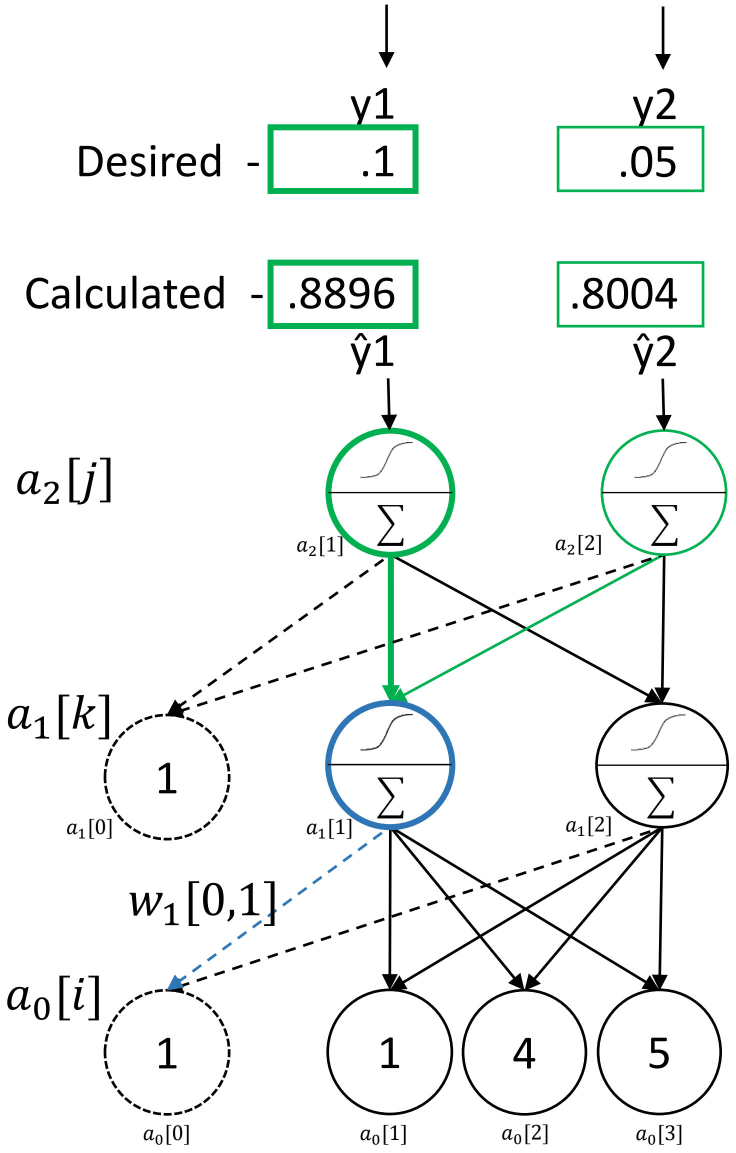
$w_2[k,j]$

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

$$\frac{\partial E}{\partial w_1[i,k]} = \frac{\partial E}{\partial a_1[k]} * \frac{\partial a_1[k]}{\partial s_1[k]} * \frac{\partial s_1[k]}{\partial w_1[i,k]}$$

$$= \frac{\partial E}{\partial a_1[k]} * \{a_1[j](1 - a_1[j])\} * \{a_0[i]\}$$

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

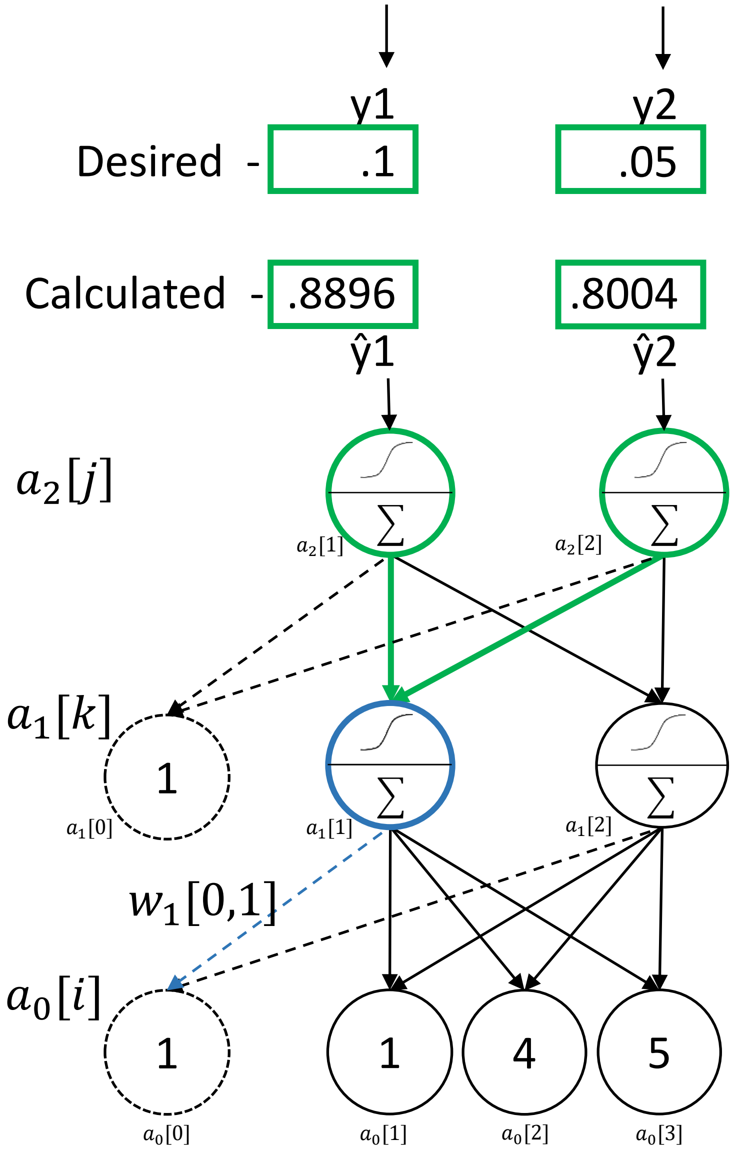
$$\frac{\partial E}{\partial w_1[i,k]} = \frac{\partial E}{\partial a_1[k]} * \frac{\partial a_1[k]}{\partial s_1[k]} * \frac{\partial s_1[k]}{\partial w_1[i,k]}$$

$$= \frac{\partial E}{\partial a_1[k]} * \{a_1[k](1 - a_1[k])\} * \{a_0[i]\}$$

$$\frac{\partial E}{\partial a_1[k]} = \sum_{j=1}^{n} \frac{\partial E}{\partial a_2[j]} * \frac{\partial a_2[j]}{\partial s_2[j]} * \frac{\partial s_2[j]}{\partial a_1[k,j]}$$

$$= \sum_{j=1}^{n} \{\hat{y}_j - y_k\} * \{a_2[j](1 - a_2[j])\} * \{w_2[k,j]\}$$

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

$$\frac{\partial E}{\partial w_1[0,1]} = \frac{\partial E}{\partial a_1[1]} * \frac{\partial a_1[1]}{\partial s_1[1]} * \frac{\partial s_1[1]}{\partial w_1[0,1]}$$

$$= \frac{\partial E}{\partial a_1[1]} * \{a_1[1](1 - a_1[1])\} * \{a_0[0]\}$$

$$\frac{\partial E}{\partial a_1[k]} = \sum_{j=1}^{n} \frac{\partial E}{\partial a_2[j]} * \frac{\partial a_2[j]}{\partial s_2[j]} * \frac{\partial s_2[j]}{\partial a_1[k,j]}$$

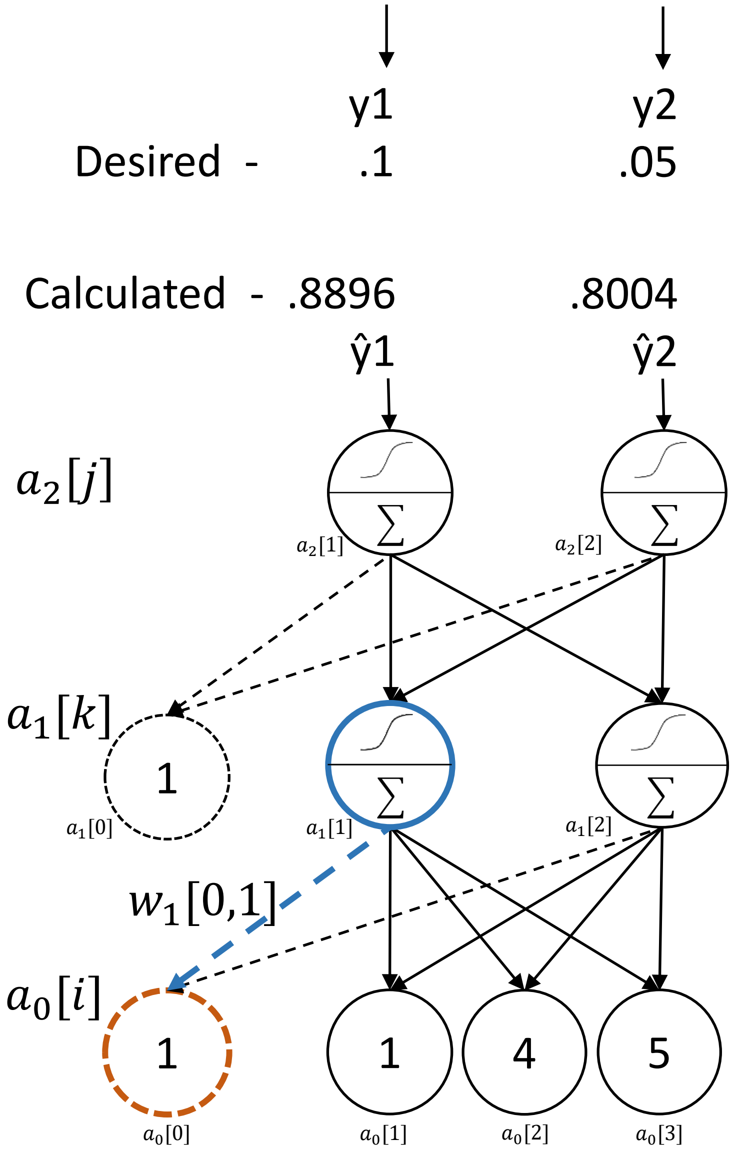
$$= \sum_{j=1}^{n} \{\hat{y}_j - y_k\} * \{a_2[j](1 - a_2[j])\} * \{w_2[k,j]\}$$

$$= .8896 - .1 * .8896(1 - .8896) * .7$$

$$+ .8004 - .05 * .8004(1 - .8004) * .8$$

$$= .1502$$

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

$$\frac{\partial E}{\partial w_1[0,1]} = \frac{\partial E}{\partial a_1[1]} * \frac{\partial a_1[1]}{\partial s_1[1]} * \frac{\partial s_1[1]}{\partial w_1[0,1]}$$

$$= \frac{\partial E}{\partial a_1[1]} * \{a_1[1](1 - a_1[1])\} * \{a_0[0]\}$$

$$= .1502 * .9866(1 - .9866) * 1$$

$$= .0020$$

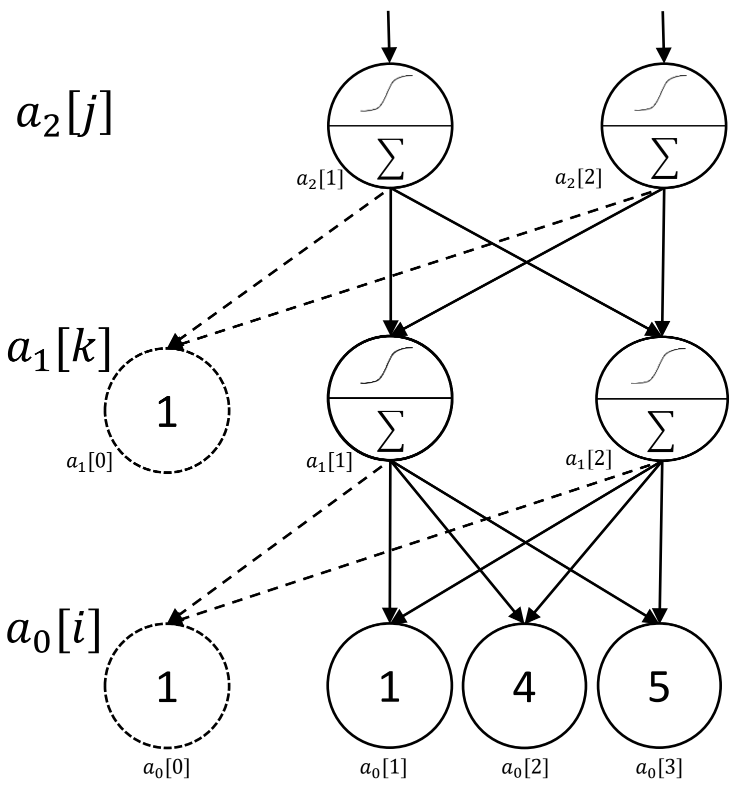
$c_1[i,k]$

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

Backpropagation

Desired - y1 y2

Calculated - ŷ1 ŷ2



x, y

	x_1	x_2	x_3	y_1	y_2
r =1	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
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

Optimizer:
Stochastic Gradient Descent

Learning rate α = .01
$w^* = w - \alpha * c$

$c_1[i, k]$

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

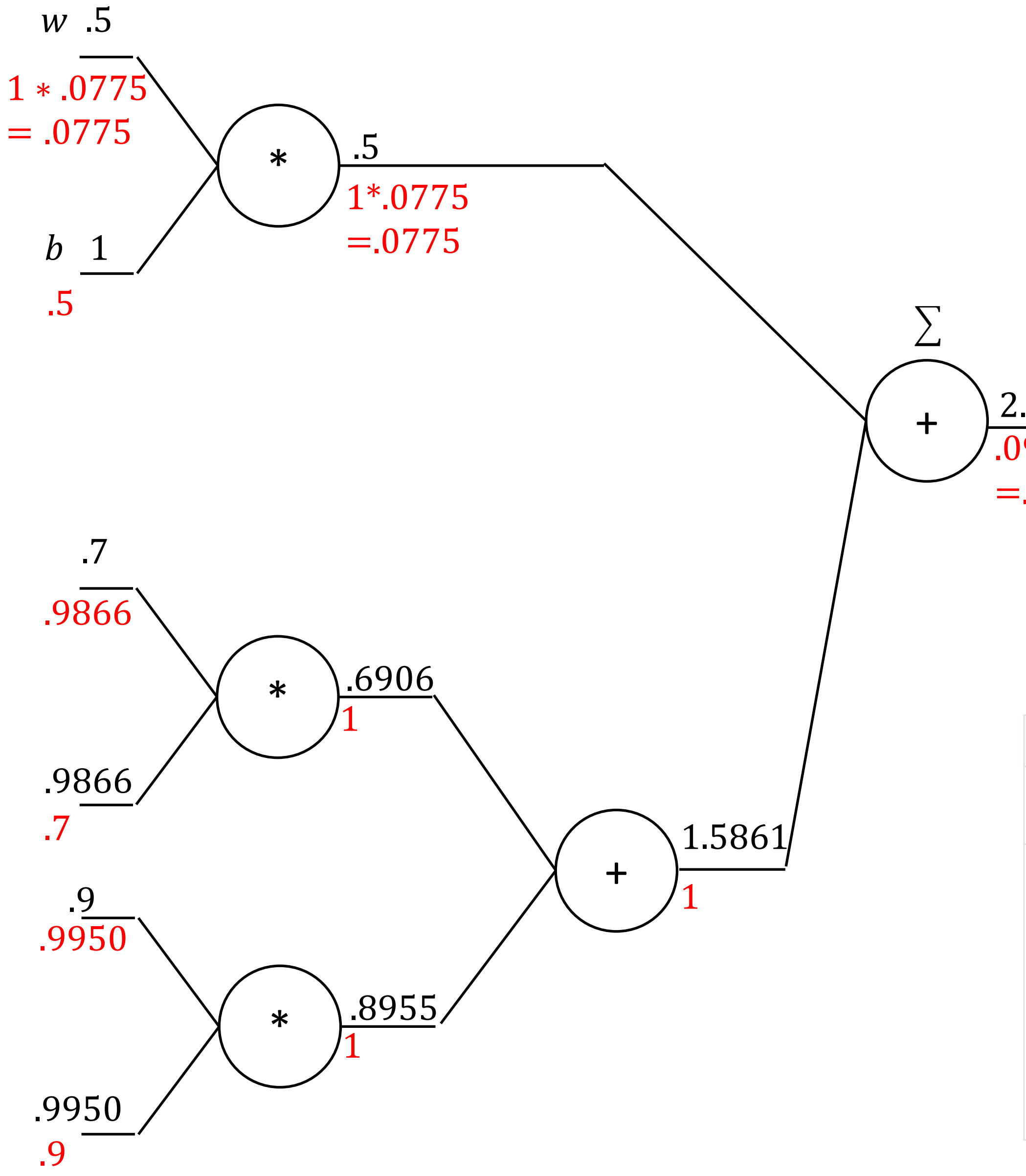
$w_1^*[i, k]$

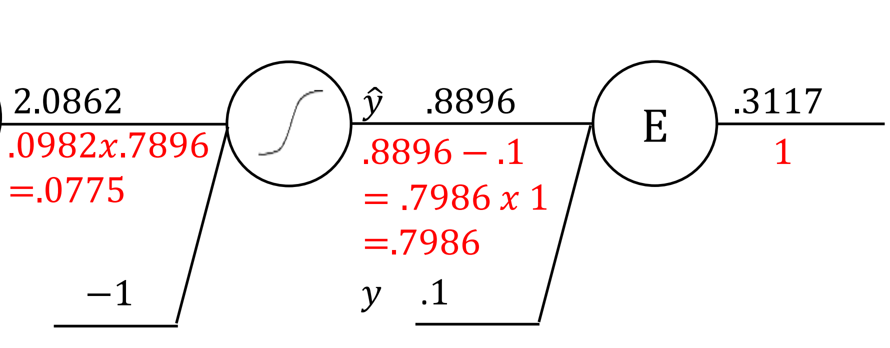
i	w[i,1]	w[i,2]
0	.4994	.4997
1	.0989	.1996
2	.3029	.4017
3	.4978	.5989

$w_2^*[k, j]$

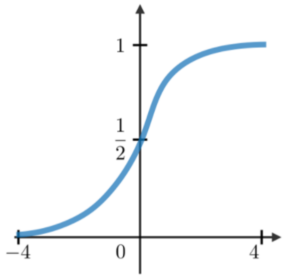
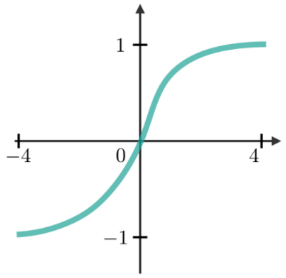
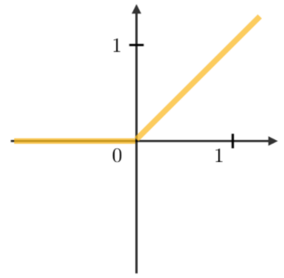
k	w[k,1]	w[k,2]
0	.4966	.0039
1	.4971	.0038
2	.0021	.0004

Backpropagation



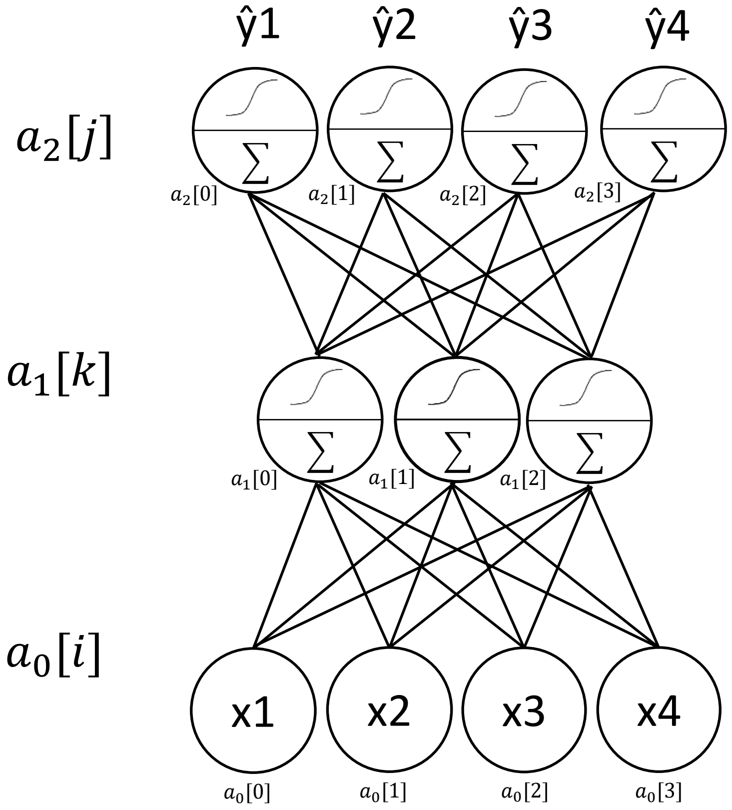


Rectified Linear Unit

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

[CS 230 - Recurrent Neural Networks Cheatsheet (stanford.edu)](https://stanford.edu)

	y1	y2	y3	y4
"e"→	0	1	0	0



"h"→	1	0	0	0

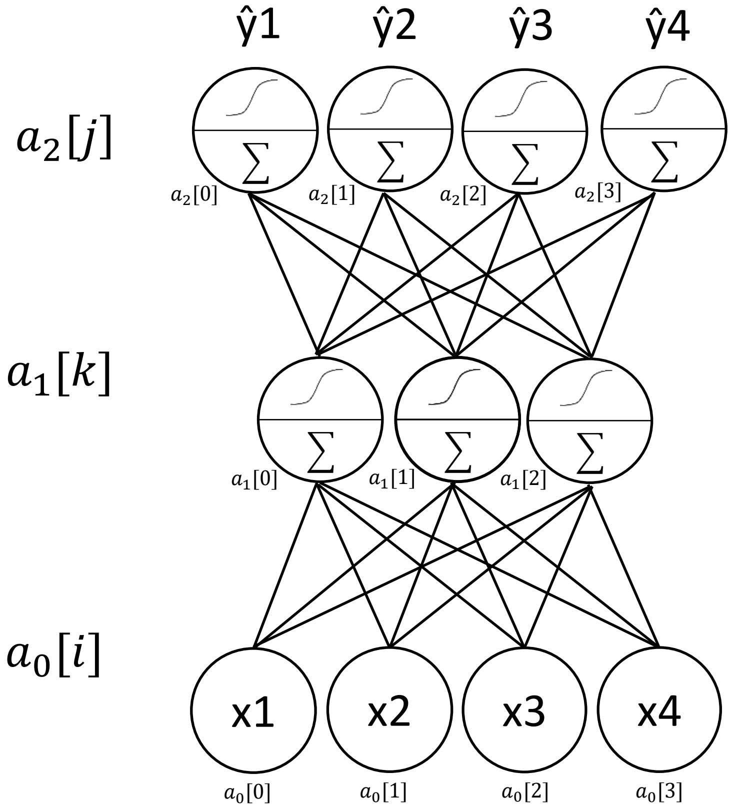
"hello"

Vocabulary

"h"	[1, 0, 0, 0]
"e"	[0, 1, 0, 0]
"l"	[0, 0, 1, 0]
"o"	[0, 0, 0, 1]

Character Prediction

	y1	y2	y3	y4
"e"→	0	1	0	0



"h"→	1	0	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

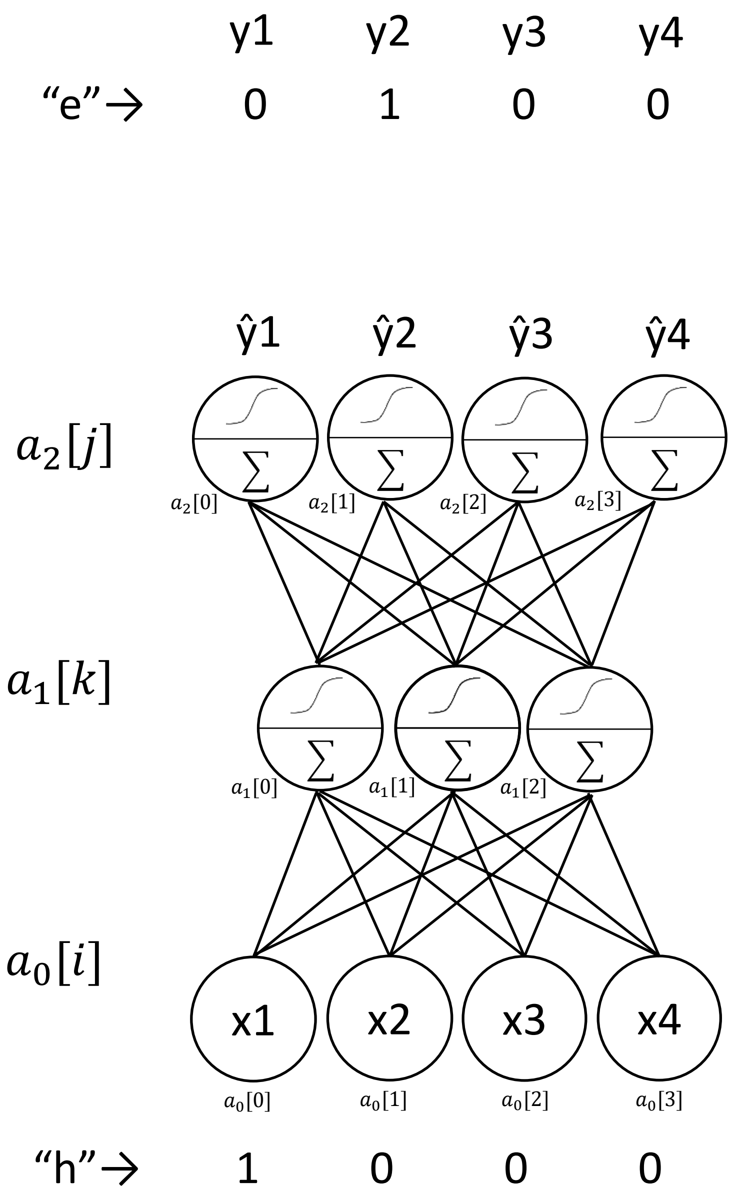
"hel"→ "p"

"hell"→ "o"

"heliu" → "m"

Fixed Width Input/Output

Character Prediction



“hello”

Vocabulary

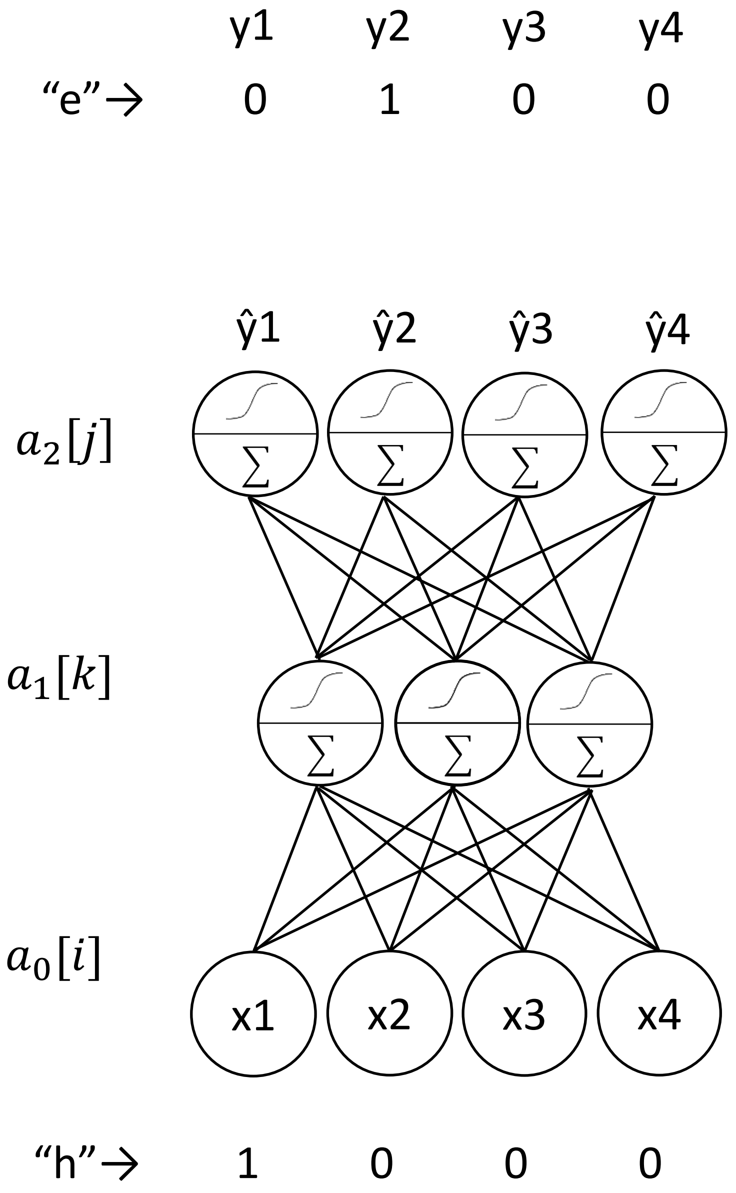
“h”	[1, 0, 0, 0]
“e”	[0, 1, 0, 0]
“l”	[0, 0, 1, 0]
“o”	[0, 0, 0, 1]

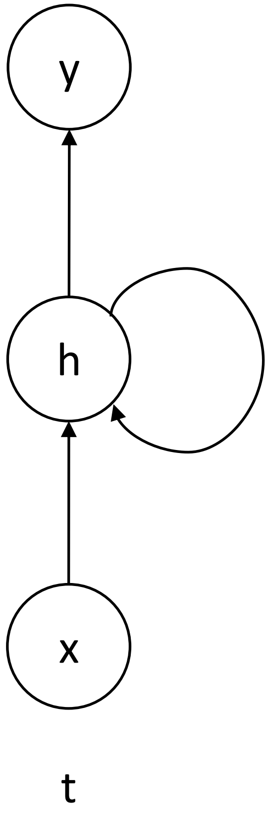
“h”→ “e”

“e”→ “l”

“l” → “l”
“l” → “o” ?

Ambiguous Targets





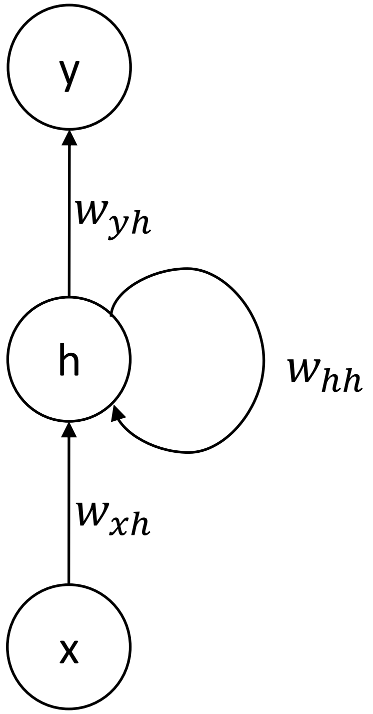
$$\hat{y} = \mathrm{softmax}(s_2)$$
$$s_2 = g(h, w_2)$$

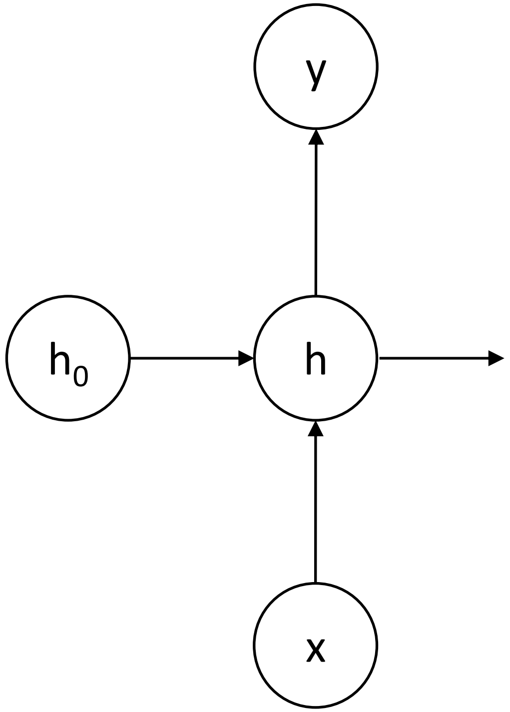
$$h = \tanh(s_1)$$
$$s_1 = f(x, h_{t-1}, w_1)$$

$$
\begin{array}{cc}
W_{yh} & H \\
Y & \begin{bmatrix} y_1h_1, y_2h_2 \ldots y_1h_k \\ y_2h_1, y_2h_2 \ldots y_2h_k \\ \vdots \\ y_nh_1, y_nh_2 \ldots y_nh_k \end{bmatrix}
\end{array}
$$

$$
\begin{array}{cc}
W_{hh} & H(t-1) \\
H & \begin{bmatrix} h_1h_1, h_1h_2 \ldots h_1h_k \\ h_2h_1, h_2h_2 \ldots h_2h_k \\ \vdots \\ h_kh_1, h_kh_2 \ldots h_kh_k \end{bmatrix}
\end{array}
$$

$$
\begin{array}{cc}
W_{xh} & X \\
H & \begin{bmatrix} h_1x_1, h_1x_2 \ldots h_1x_m \\ h_2x_1, h_2x_2 \ldots h_2x_m \\ \vdots \\ h_kx_1, h_kx_2 \ldots h_kx_m \end{bmatrix}
\end{array}
$$



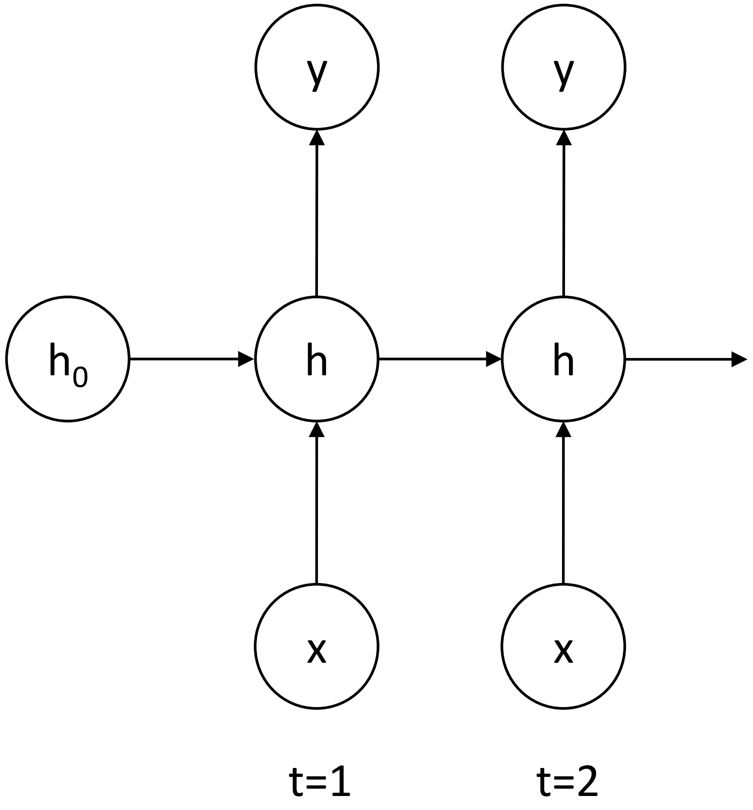


$$\hat{y} = \mathrm{softmax}(s_2)$$
$$s_2 = g(h, w_2)$$

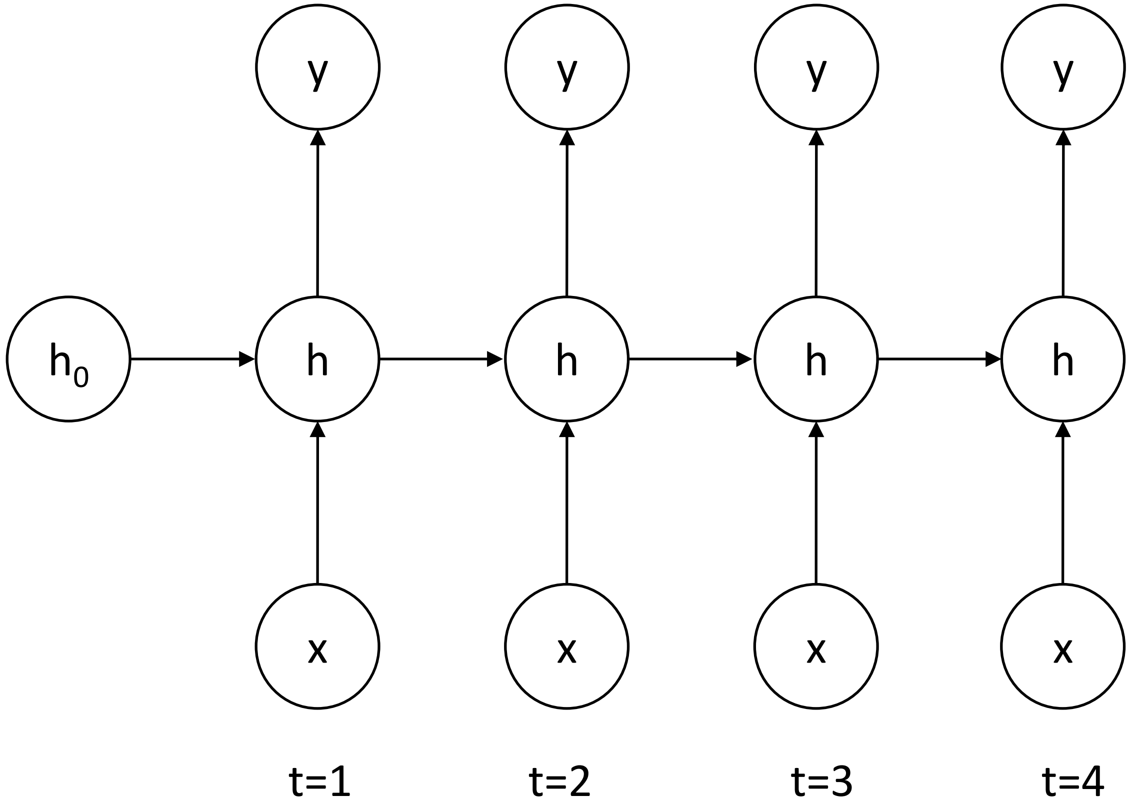
$$h = \tanh(s_1)$$
$$s_1 = f(x, h_{t-1}, w_1)$$

t=1

Recurrent Neural Networks



Recurrent Neural Networks



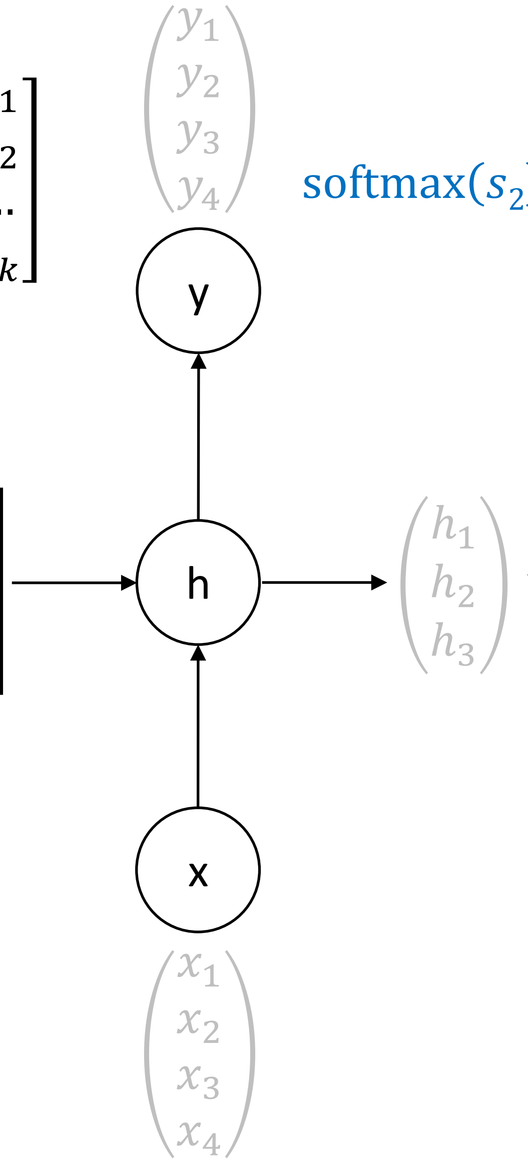
$$\overset{w_{yh}}{\begin{bmatrix} y_1h_1, y_2h_2 \dots y_1h_k \\ y_2h_1, y_2h_2 \dots y_2h_k \\ \vdots \\ y_nh_1, y_nh_2 \dots y_nh_k \end{bmatrix}} * \overset{h}{\begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_k \end{bmatrix}} = \overset{s_2}{\begin{bmatrix} s_1 \\ s_2 \\ \dots \\ s_k \end{bmatrix}}, \mathrm{softmax}(s_2) = \overset{\hat{y}}{\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_k \end{bmatrix}}$$

$$\mathrm{softmax}(s_2) = \frac{e^{s_{2j}}}{\sum_{i=1}^{n} e^{s_{2i}}}$$

$$\overset{w_{xh}}{\begin{bmatrix} h_1x_1, h_1x_2 \dots h_1x_m \\ h_2x_1, h_2x_2 \dots h_2x_m \\ \vdots \\ h_kx_1, h_kx_2 \dots h_kx_m \end{bmatrix}} * \overset{x}{\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{bmatrix}} + \overset{w_{hh}}{\begin{bmatrix} h_1h_1, h_1h_2 \dots h_1h_k \\ h_2h_1, h_2h_2 \dots h_2h_k \\ \vdots \\ h_kh_1, h_kh_2 \dots h_kh_k \end{bmatrix}} * \overset{h_{t-1}}{\begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_k \end{bmatrix}} = \overset{s_1}{\begin{bmatrix} s_1 \\ s_2 \\ \dots \\ s_k \end{bmatrix}}, \tanh(s_1) = \overset{h}{\begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_k \end{bmatrix}}$$

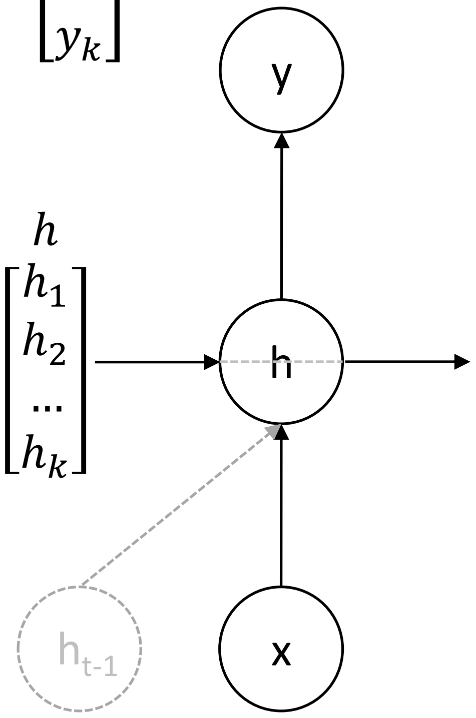
$$\tanh(s_1) = \frac{e^{2x}-1}{e^{2x}+1}$$

$$\overset{w_{xh}}{\begin{bmatrix} h_1x_1, h_1x_2 \dots h_1x_m \\ h_2x_1, h_2x_2 \dots h_2x_m \\ \vdots \\ h_kx_1, h_kx_2 \dots h_kx_m \end{bmatrix}} * \overset{x}{\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{bmatrix}}$$



$$\overset{w_{yh}}{\begin{bmatrix} y_1h_1, y_2h_2 \dots y_1h_k \\ y_2h_1, y_2h_2 \dots y_2h_k \\ \vdots \\ y_nh_1, y_nh_2 \dots y_nh_k \end{bmatrix}} * \overset{h}{\begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_k \end{bmatrix}} = \overset{s_2}{\begin{bmatrix} s_1 \\ s_2 \\ \dots \\ s_k \end{bmatrix}}, \mathrm{softmax}(s_2) = \overset{\hat{y}}{\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_k \end{bmatrix}}$$

$$\overset{w_{xh}}{\begin{bmatrix} h_1x_1, h_1x_2 \dots h_1x_m \\ h_2x_1, h_2x_2 \dots h_2x_m \\ \vdots \\ h_kx_1, h_kx_2 \dots h_kx_m \end{bmatrix}} * \overset{x}{\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{bmatrix}} + \overset{w_{hh}}{\begin{bmatrix} h_1h_1, h_1h_2 \dots h_1h_k \\ h_2h_1, h_2h_2 \dots h_2h_k \\ \vdots \\ h_kh_1, h_kh_2 \dots h_kh_k \end{bmatrix}} * \overset{h_{t-1}}{\begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_k \end{bmatrix}} = \overset{s_1}{\begin{bmatrix} s_1 \\ s_2 \\ \dots \\ s_k \end{bmatrix}}, \tanh(s_1) = \overset{h}{\begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_k \end{bmatrix}}$$

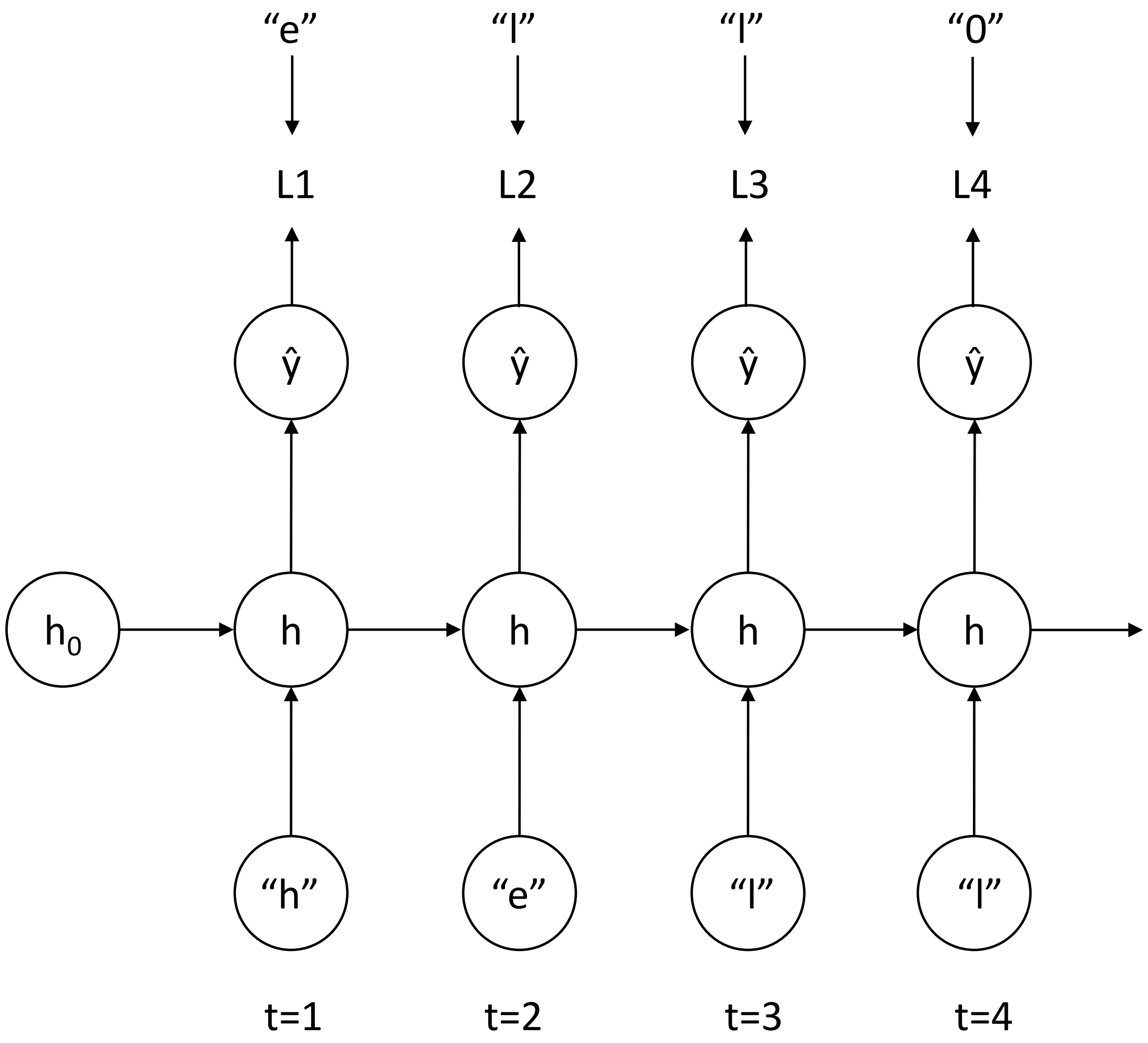


$$\overset{w_{xh}}{\begin{bmatrix} h_1x_1, h_1x_2 \dots h_1x_m \\ h_2x_1, h_2x_2 \dots h_2x_m \\ \vdots \\ h_kx_1, h_kx_2 \dots h_kx_m \end{bmatrix}} * \overset{x}{\begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{bmatrix}}$$

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

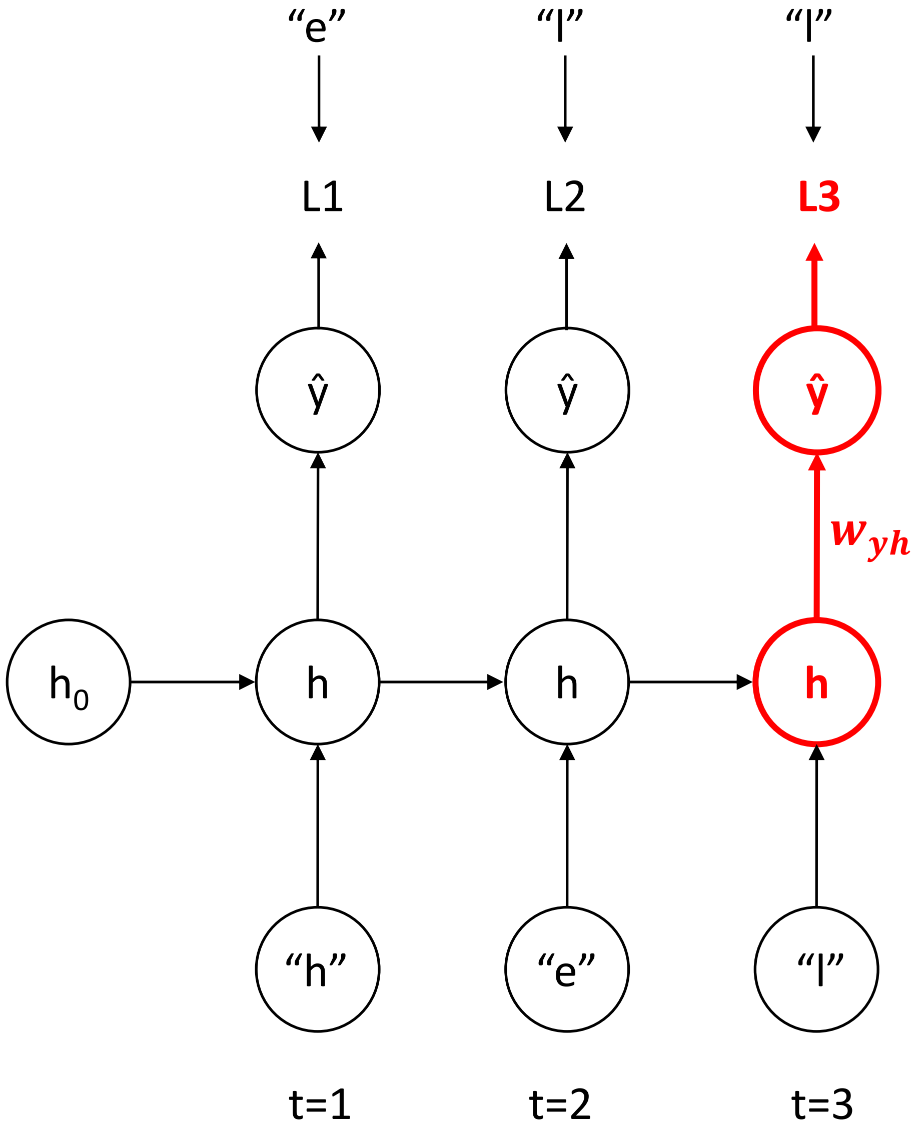
$$L_t = -y_t \log(\hat{y})$$

Multi-class Cross Entropy Loss



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

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



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

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

For derivatives of Softmax and Cross Entropy Loss

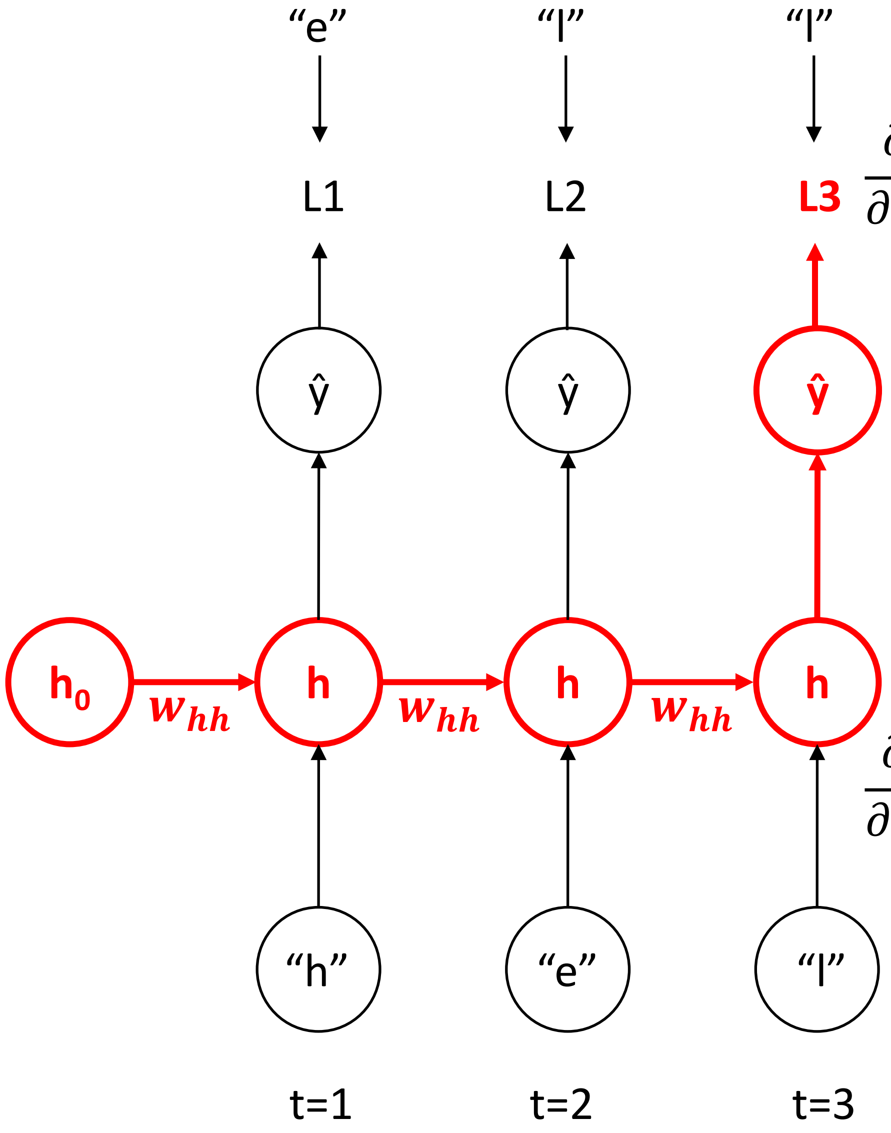
Part 2: Softmax Regression (saitcelebi.com)

How to compute the derivative of softmax and cross-entropy – Charlee Li

Recurrent Neural Networks

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

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



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

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

For derivatives of Softmax and Cross Entropy Loss

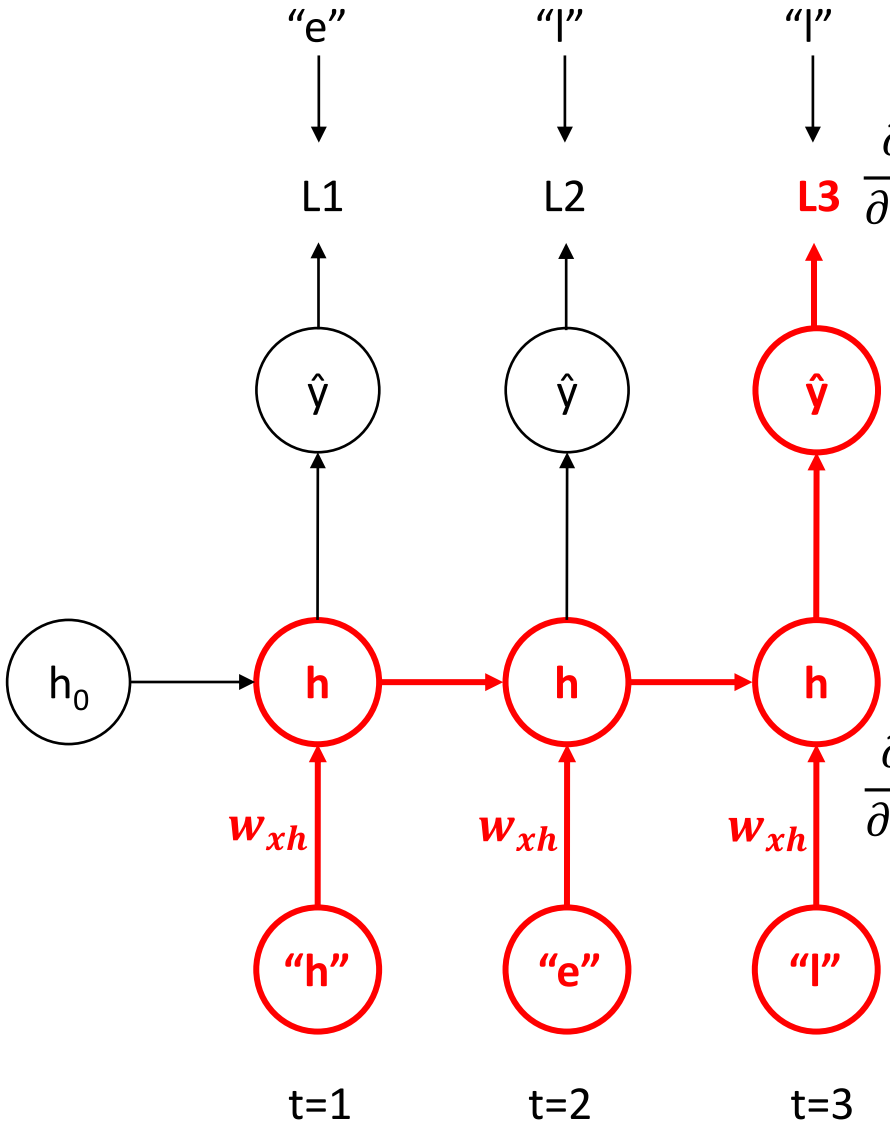
[Part 2: Softmax Regression (saitcelebi.com)]

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

Recurrent Neural Networks

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

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



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

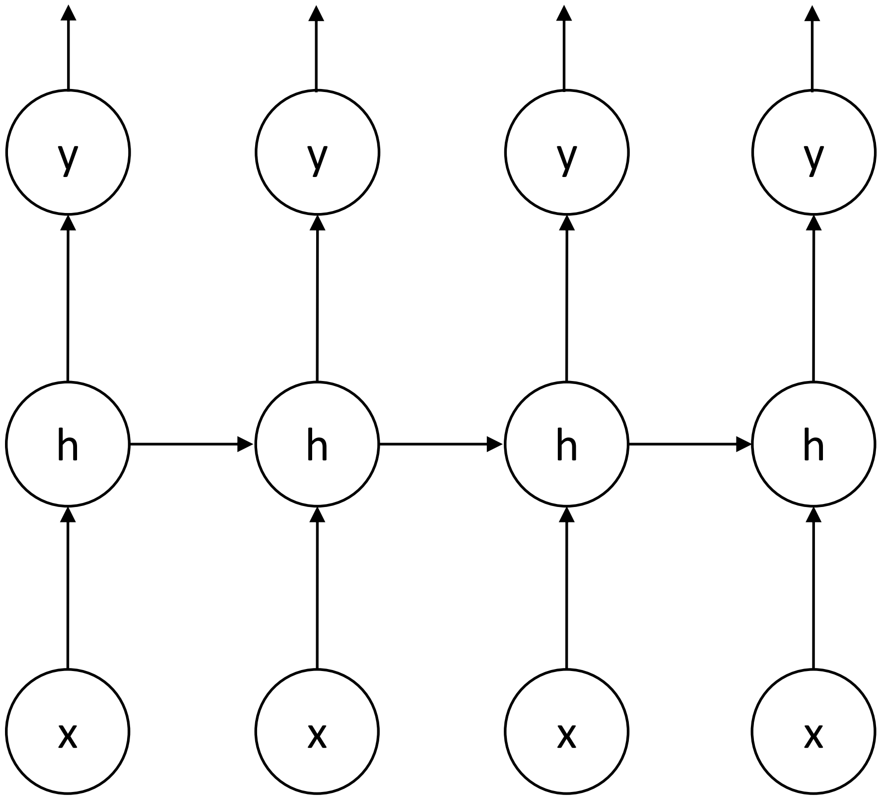
$$\frac{\partial LT}{\partial W_{xh}} = \sum_{t=1}^{T} \frac{\partial LT}{\partial \hat{Y}T} * \frac{\partial \hat{Y}T}{\partial H_t} * \frac{\partial H_t}{\partial W_{xh}}$$

For derivatives of Softmax and Cross Entropy Loss

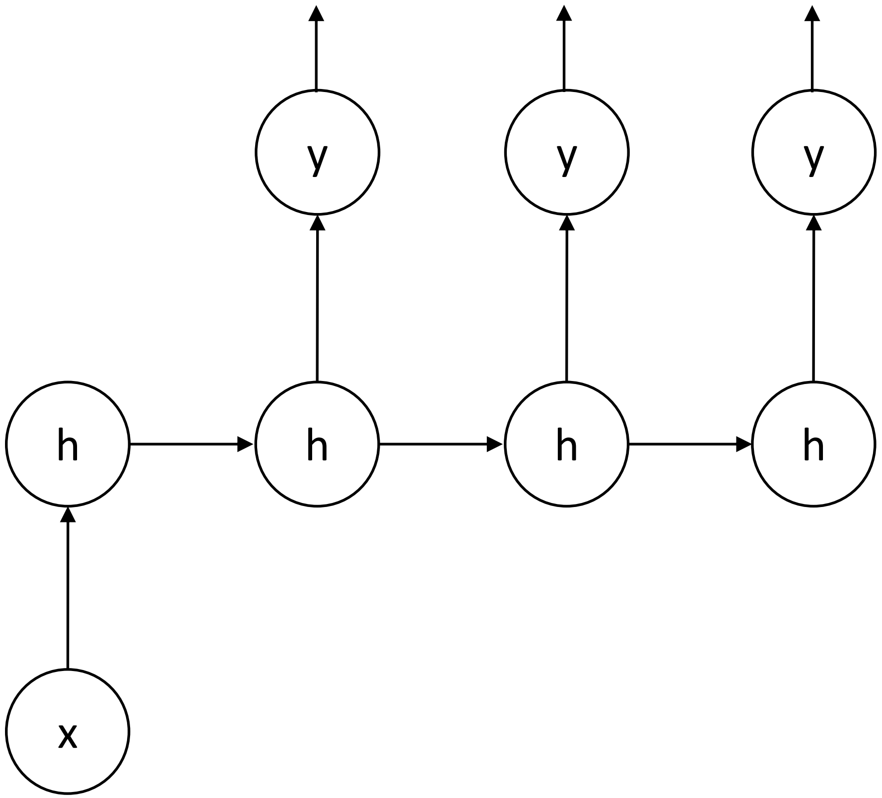
[Part 2: Softmax Regression (saitcelebi.com)]

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

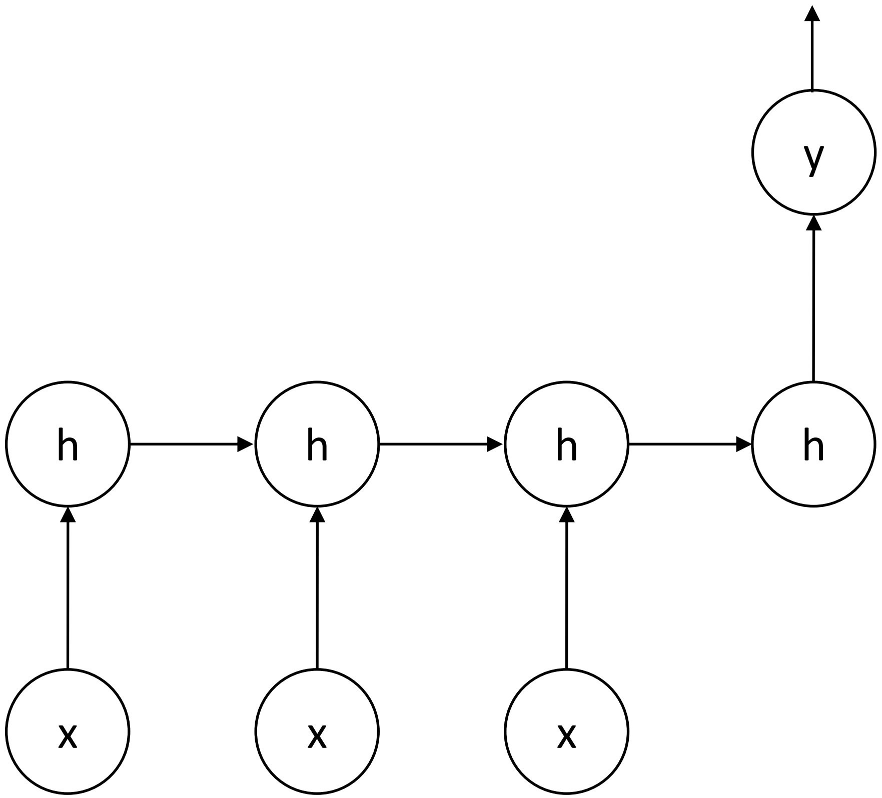
One-to-One



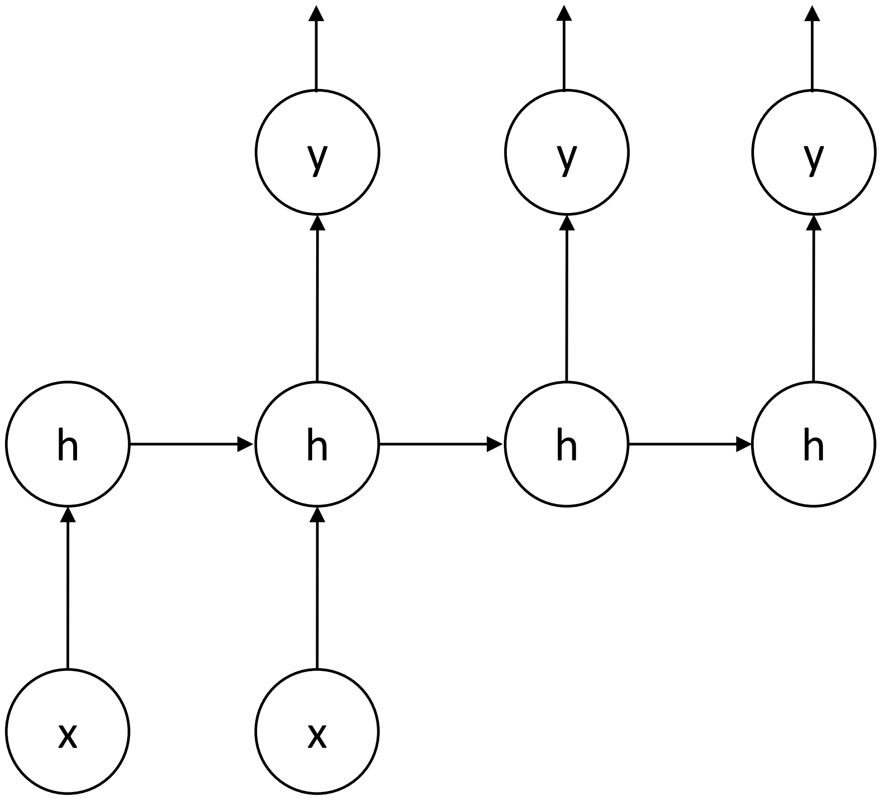
One-to-Many



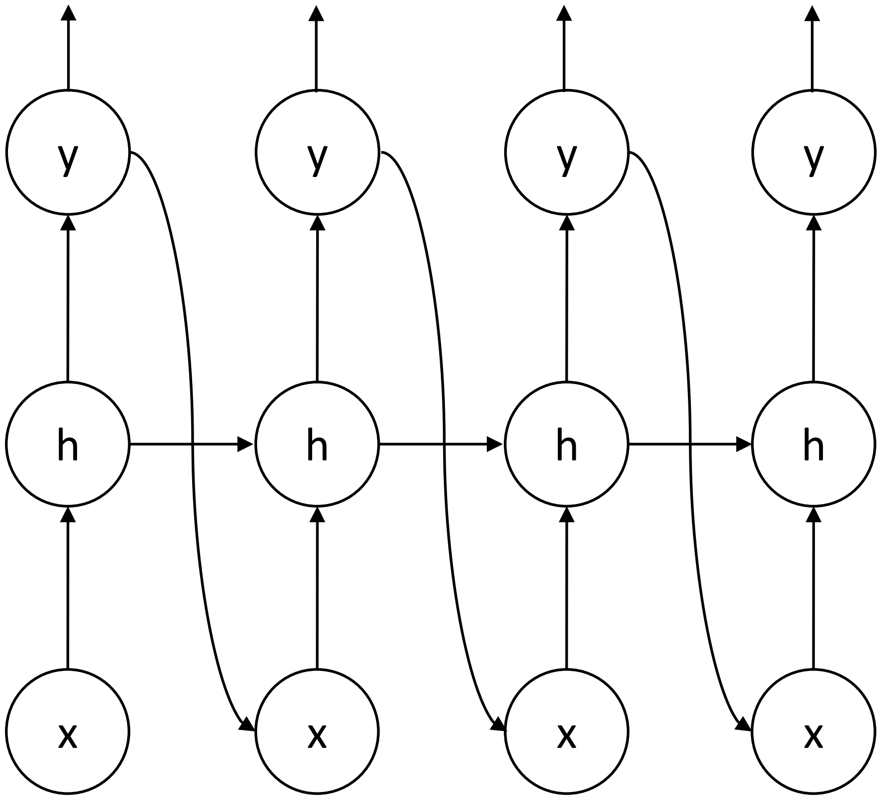
Many-to-One



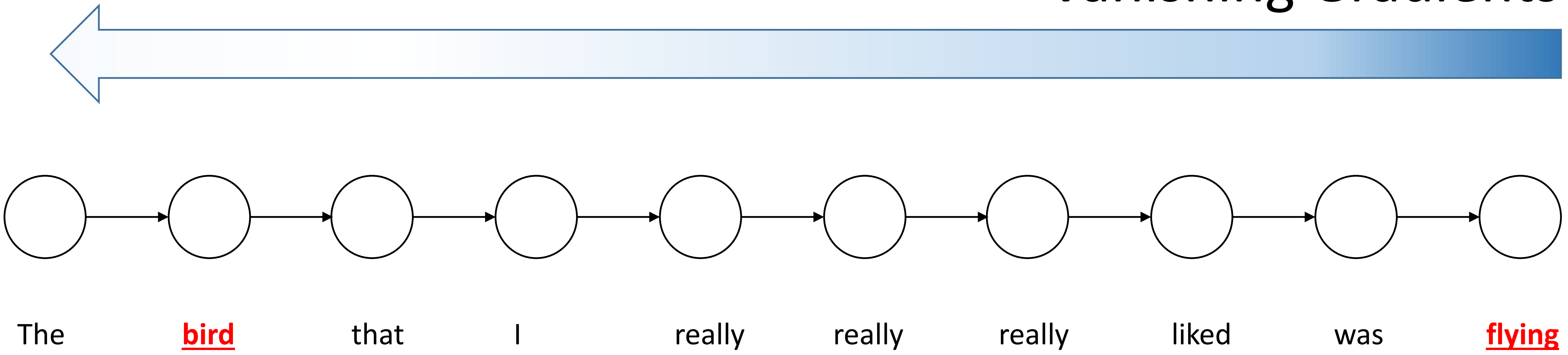
Many-to-Many



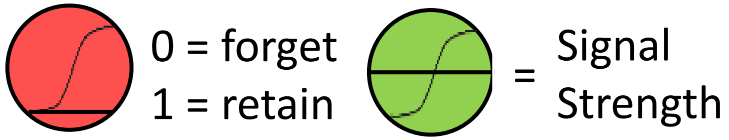
Generation

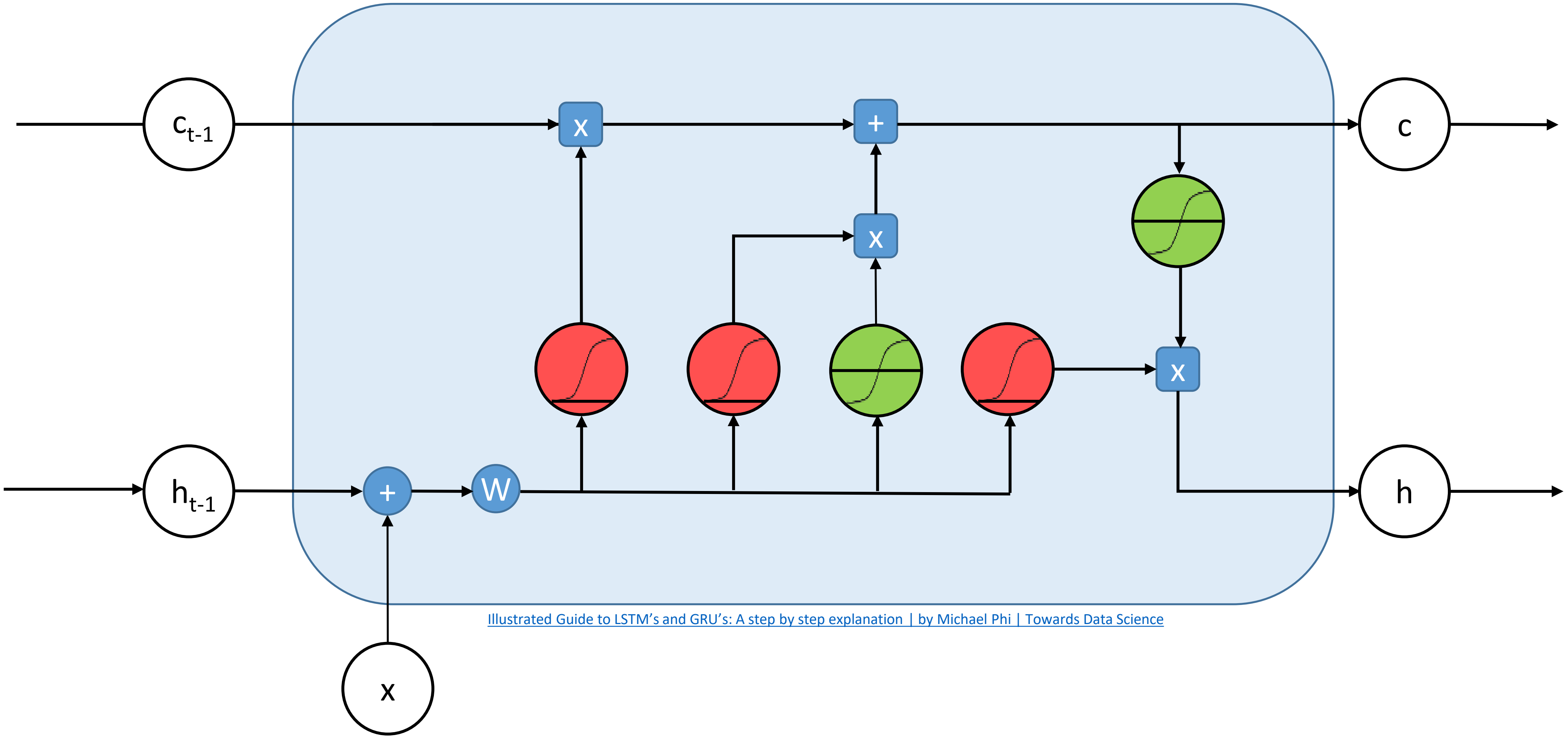


Vanishing Gradients



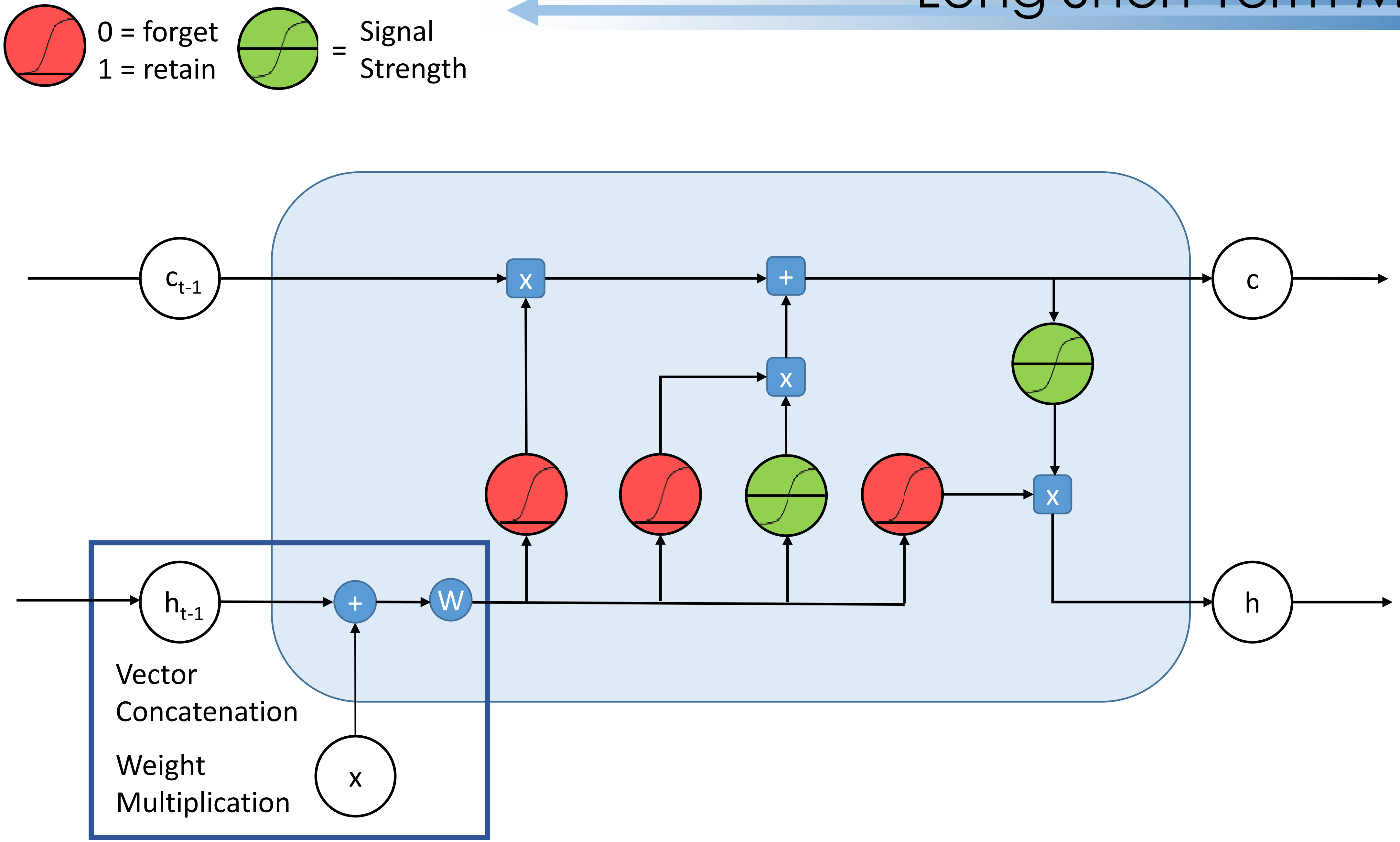
Long Short Term Memory



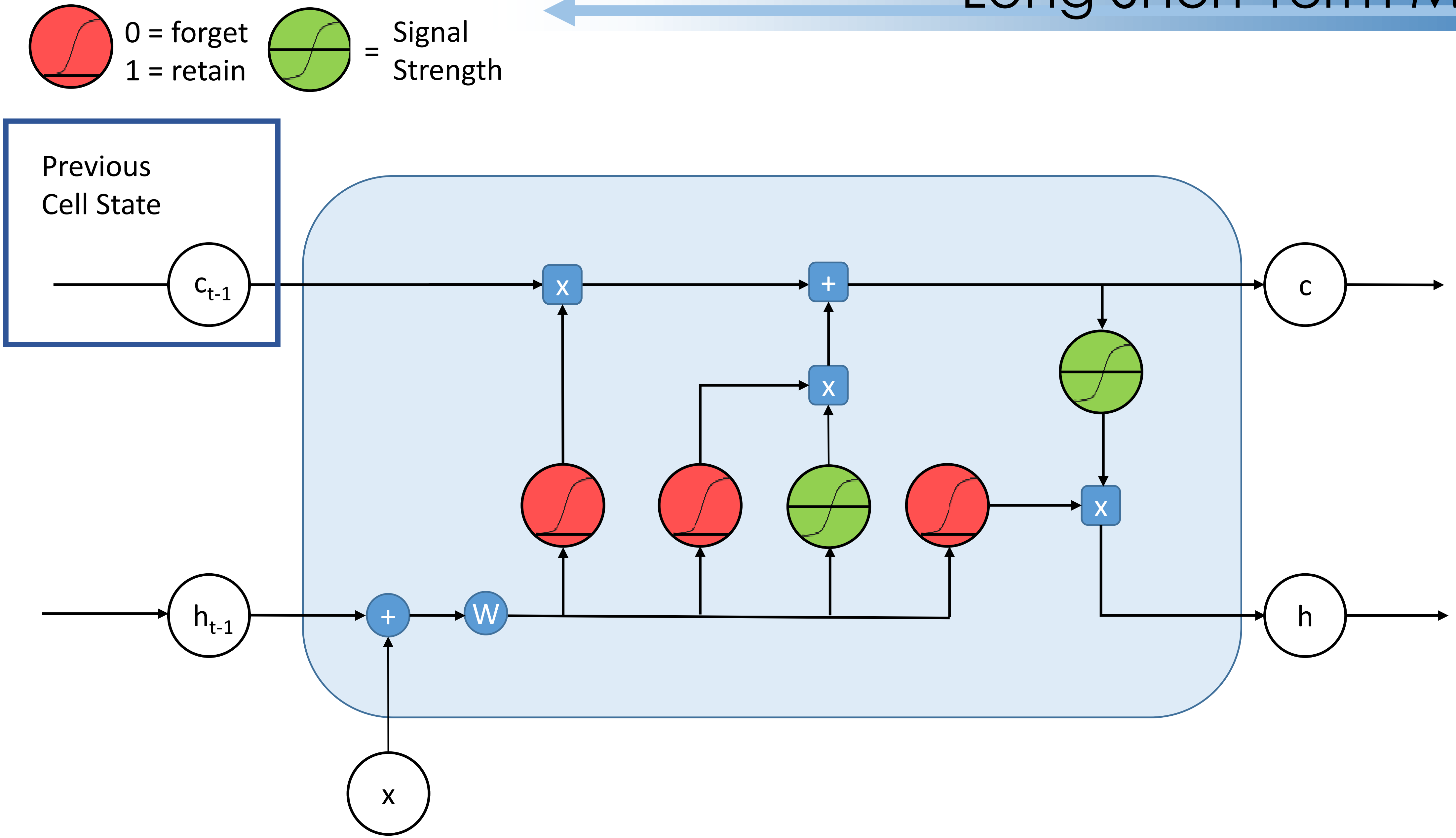


[Illustrated Guide to LSTM's and GRU's: A step by step explanation | by Michael Phi | Towards Data Science]

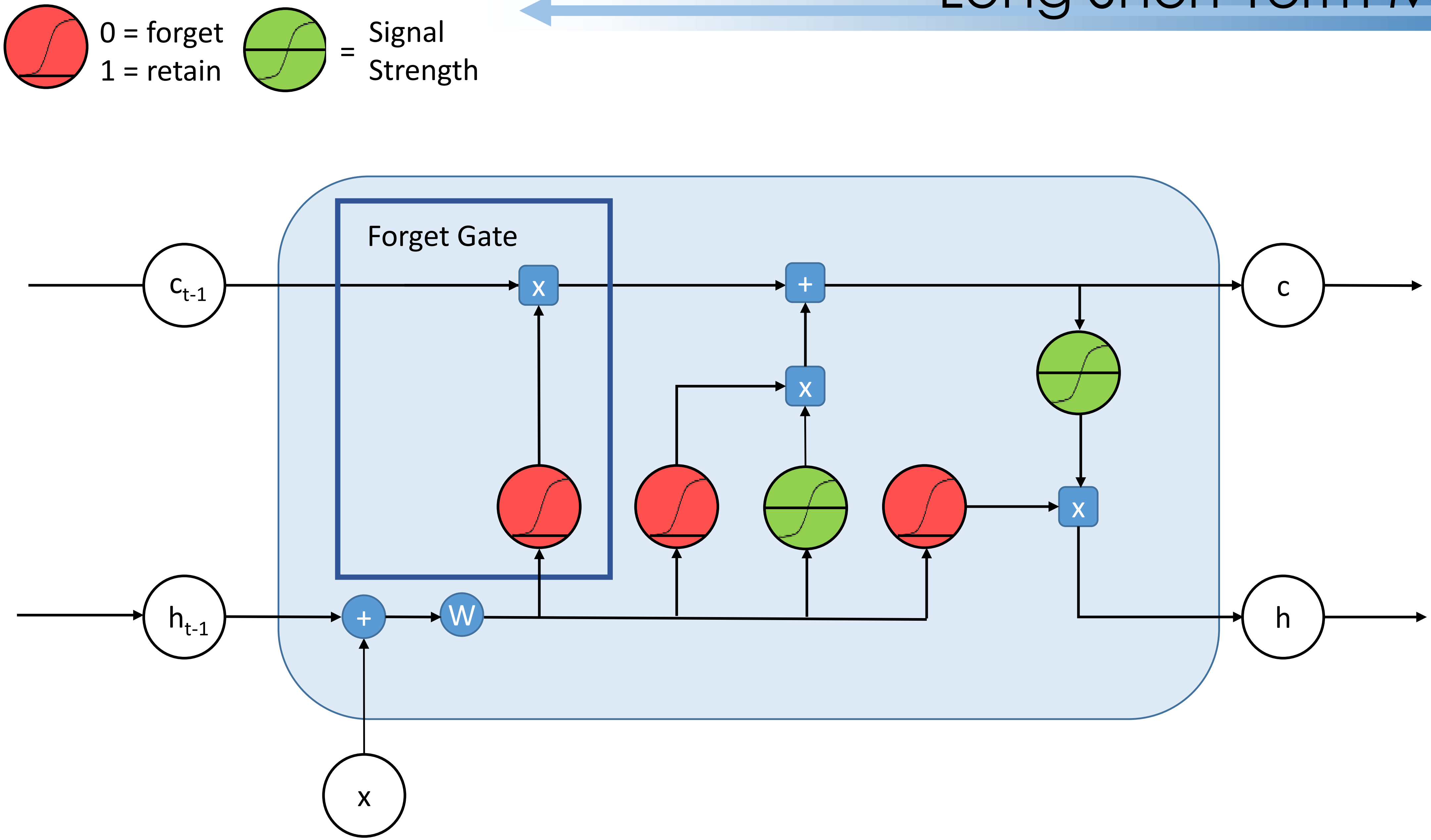
Long Short Term Memory



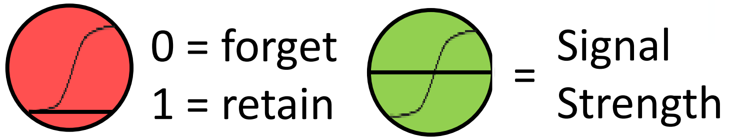
Long Short Term Memory

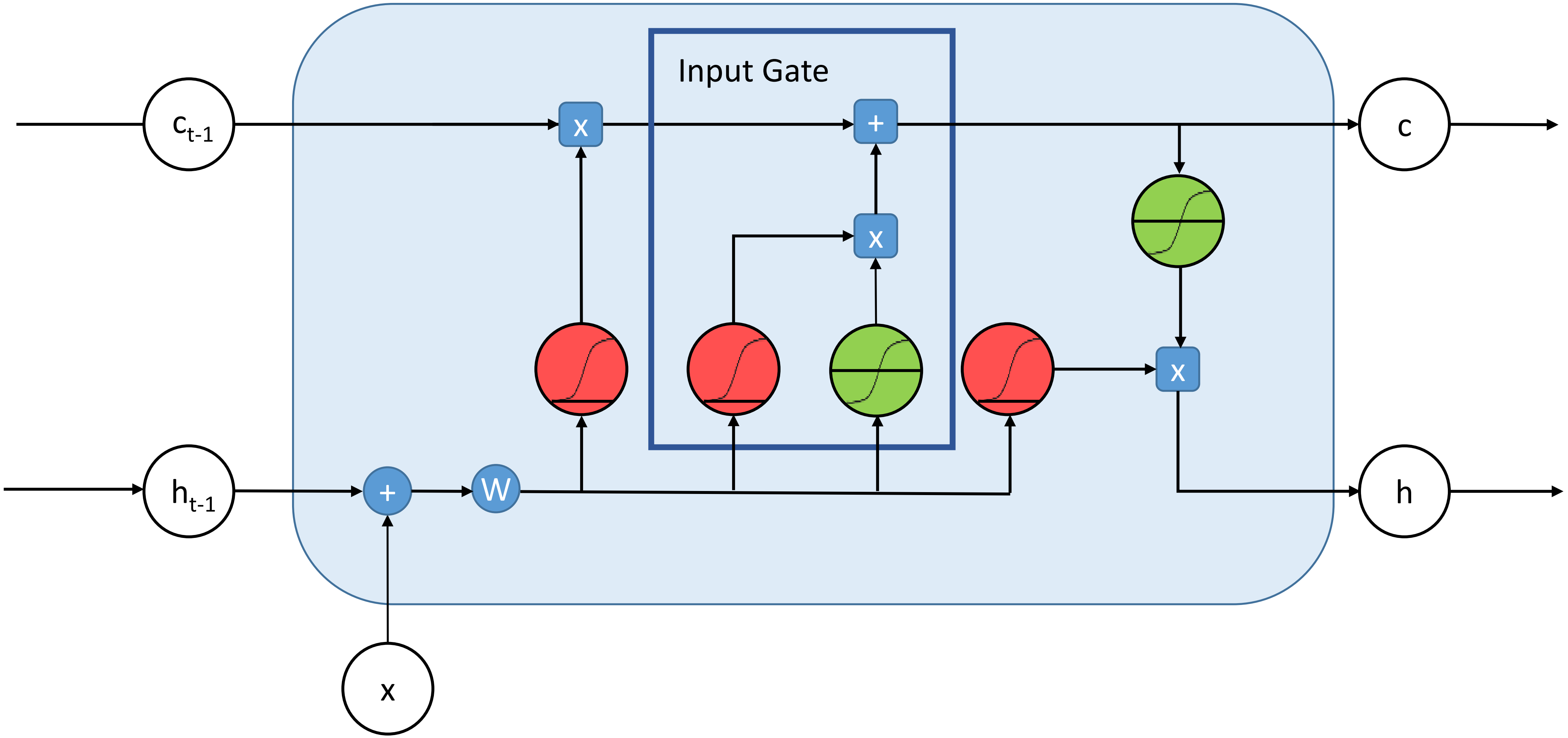


Long Short Term Memory

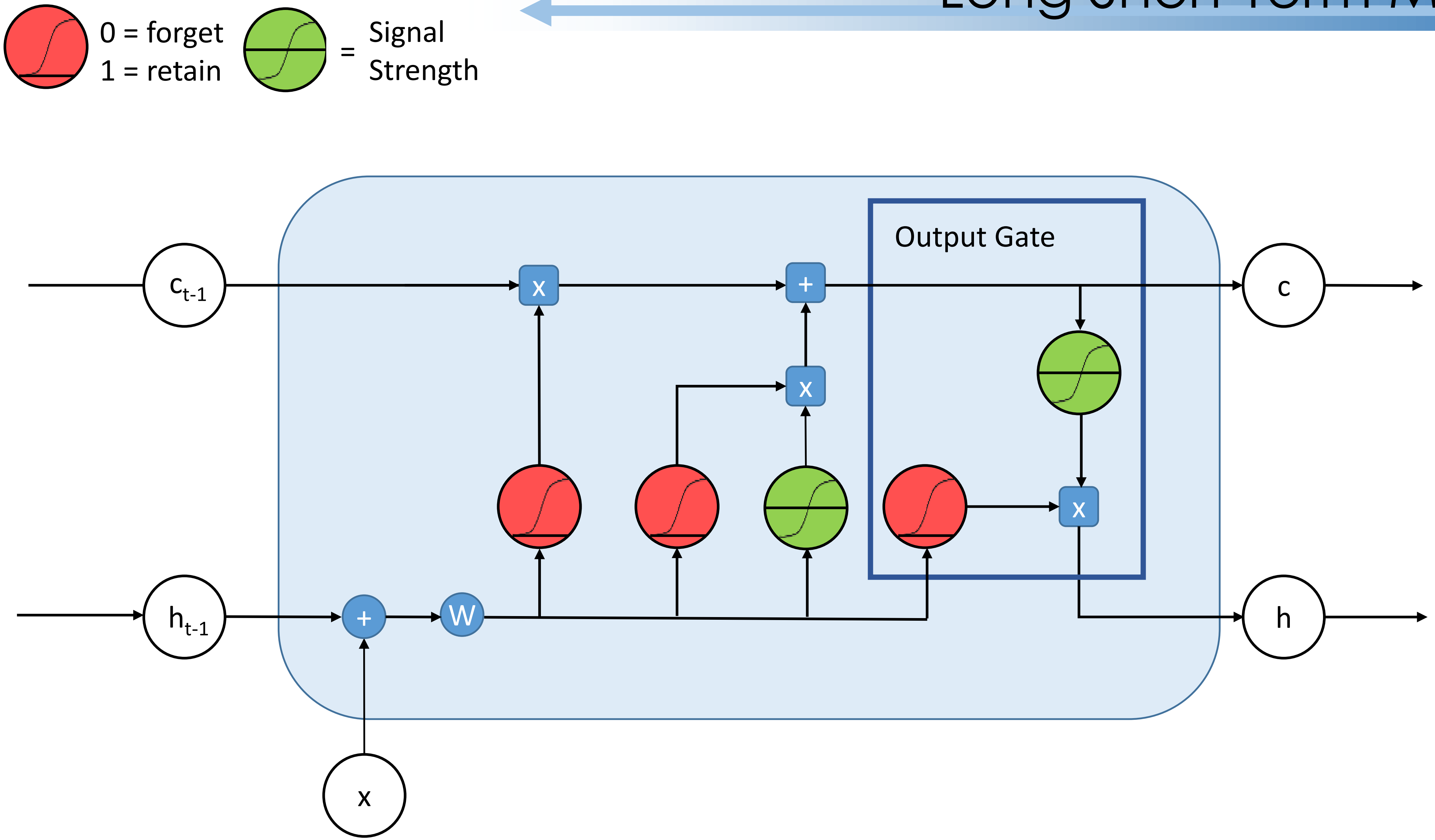


Long Short Term Memory

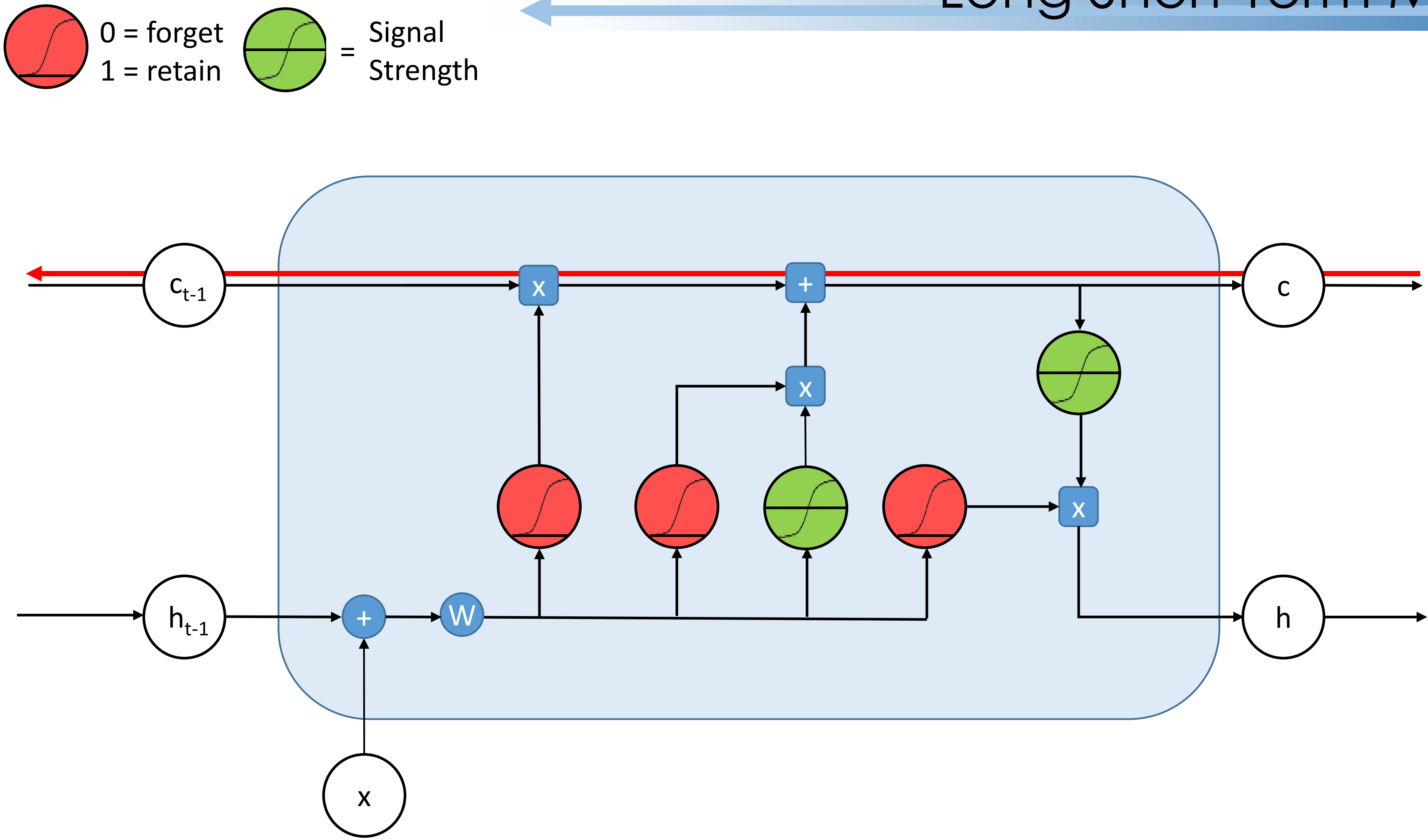




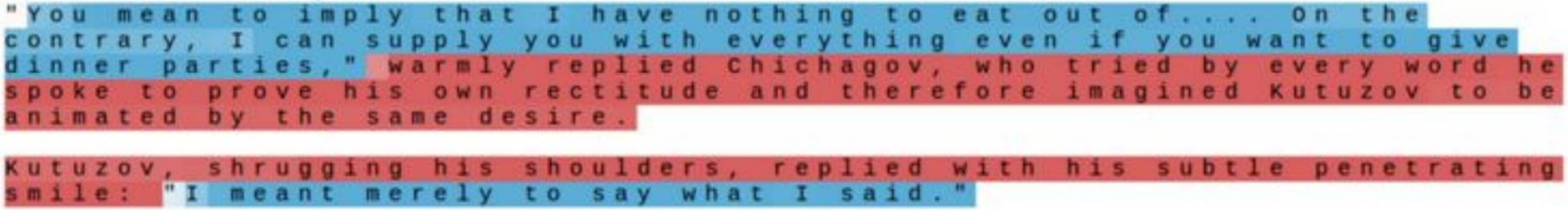
Long Short Term Memory



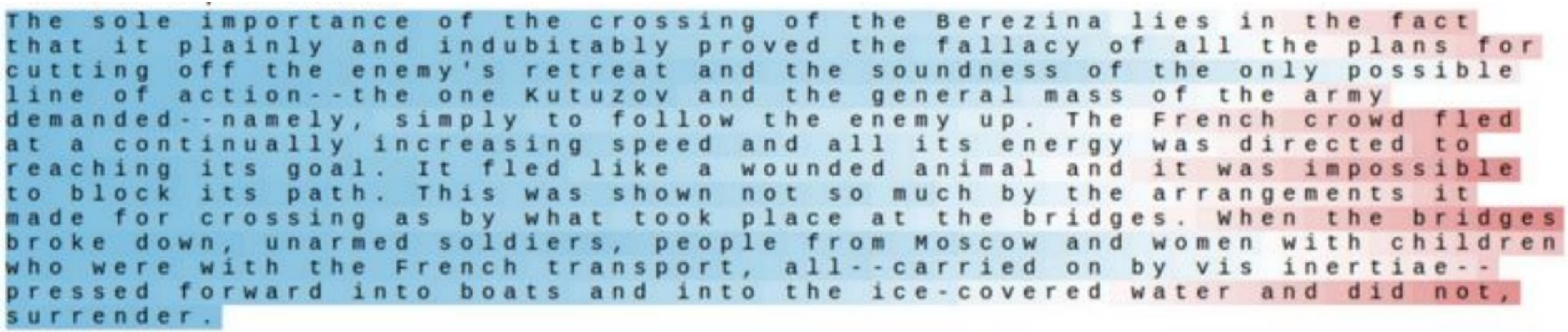
Long Short Term Memory



Long Short Term Memory



quote detection cell



line length tracking cell

CS 230 - Recurrent Neural Networks Cheatsheet (stanford.edu)