Section 2: Learning

Backpropagation

Recurrent Neural Nets

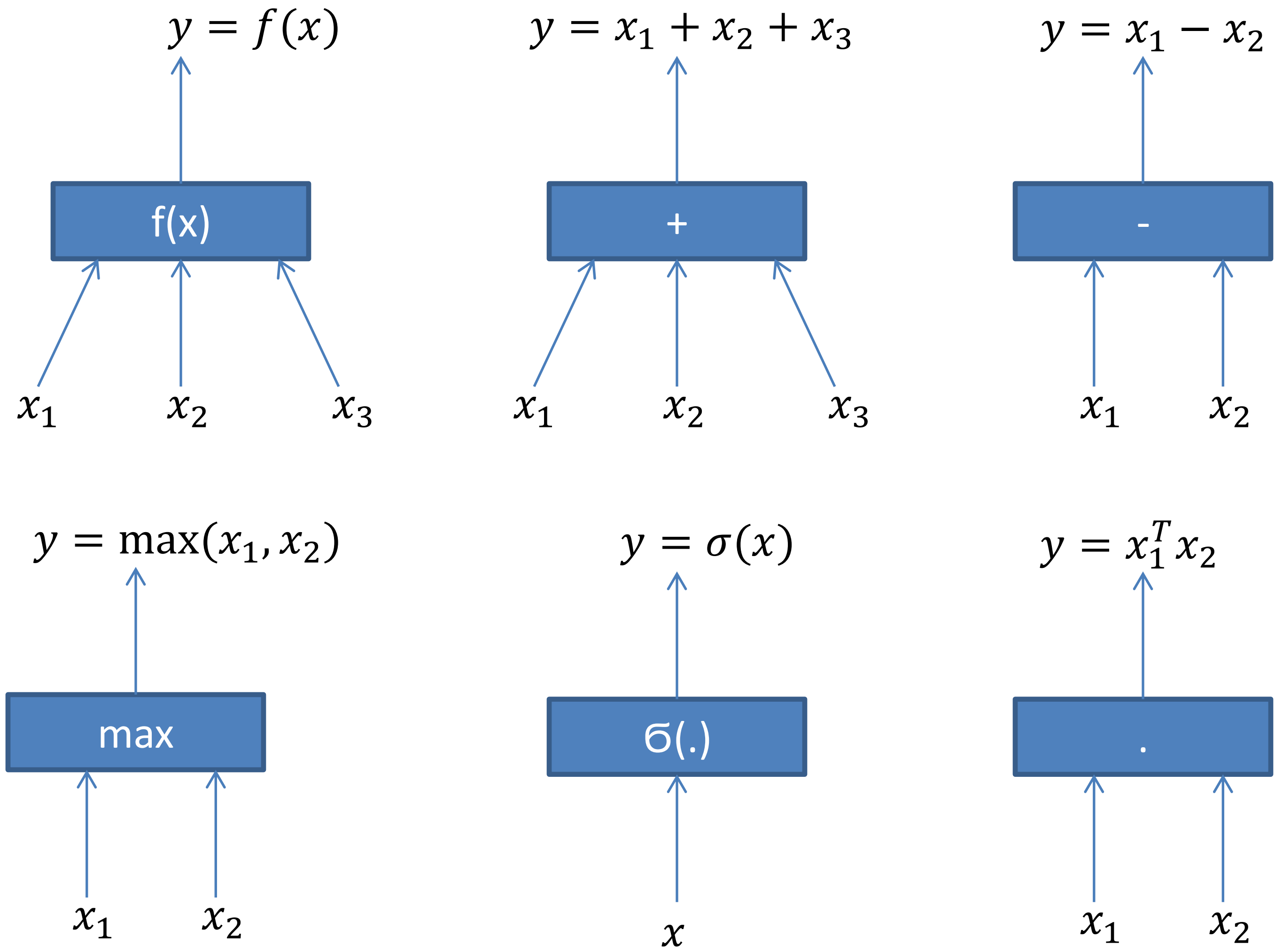
Topics

- Review of backprop
  - Basic operations
  - Class example
- Additional complexity
  - Shared parameters
  - Dealing with vectors (optional)
- Recurrent Neural Net (RNN)
  - Motivation
  - Simple backprop (vector one optional)
  - Demo

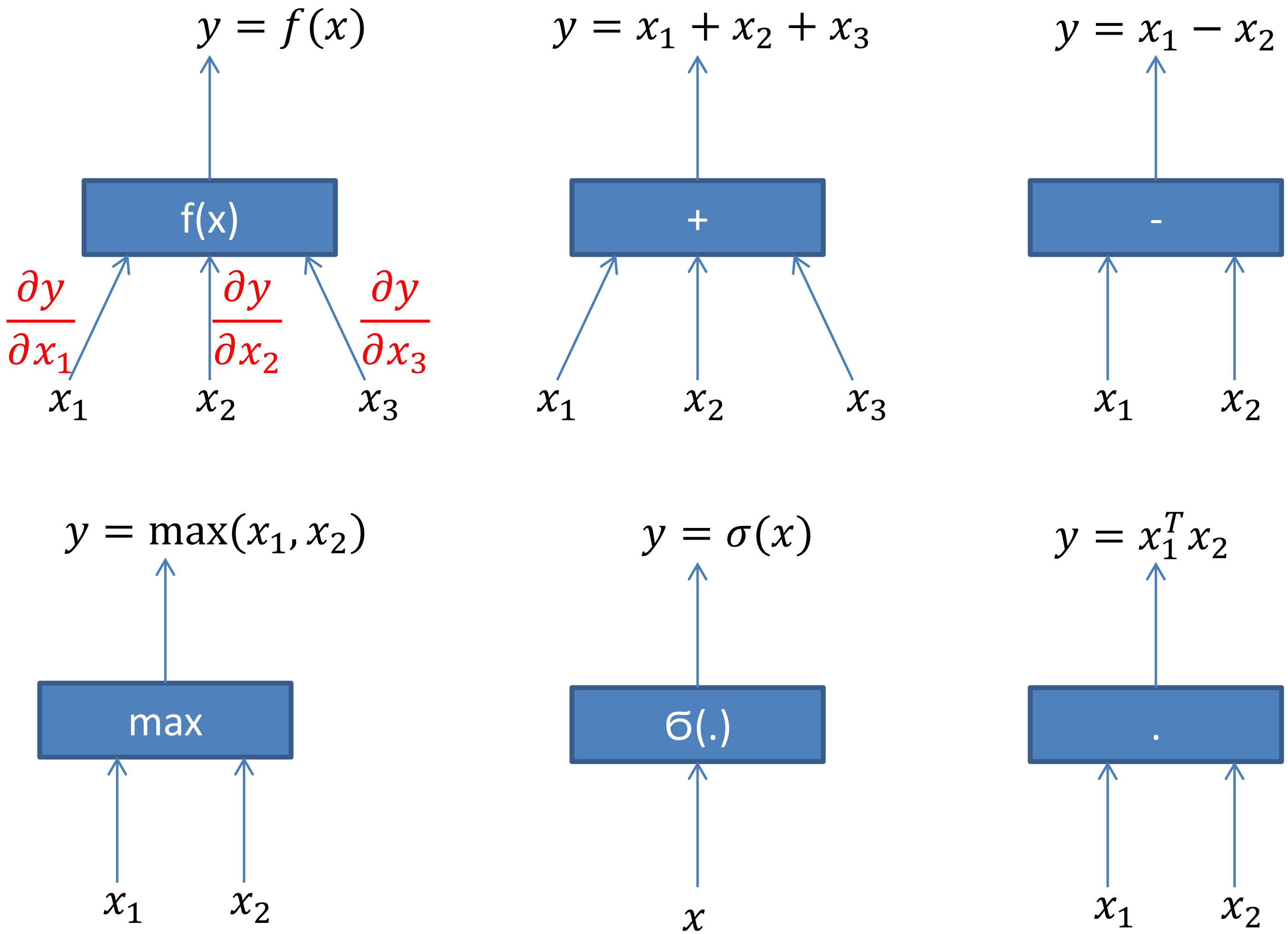
Topics

- Review of backprop
  - Basic operations
  - Class example
- Additional complexity
  - Shared parameters
  - Dealing with vectors (optional)
- Recurrent Neural Net (RNN)
  - Motivation
  - Simple backprop (vector one optional)
  - Demo

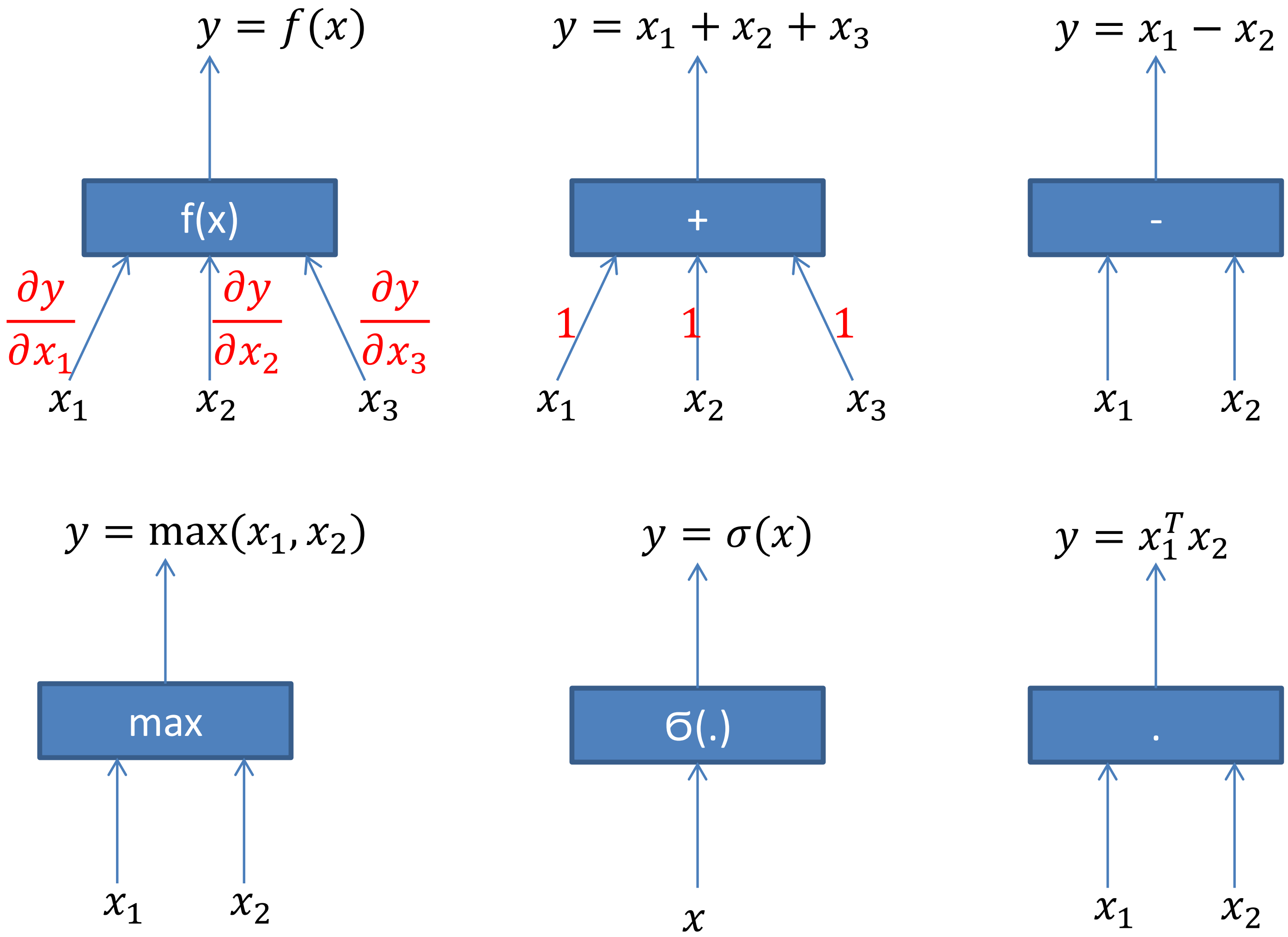
Review of backprop: Basic Operations



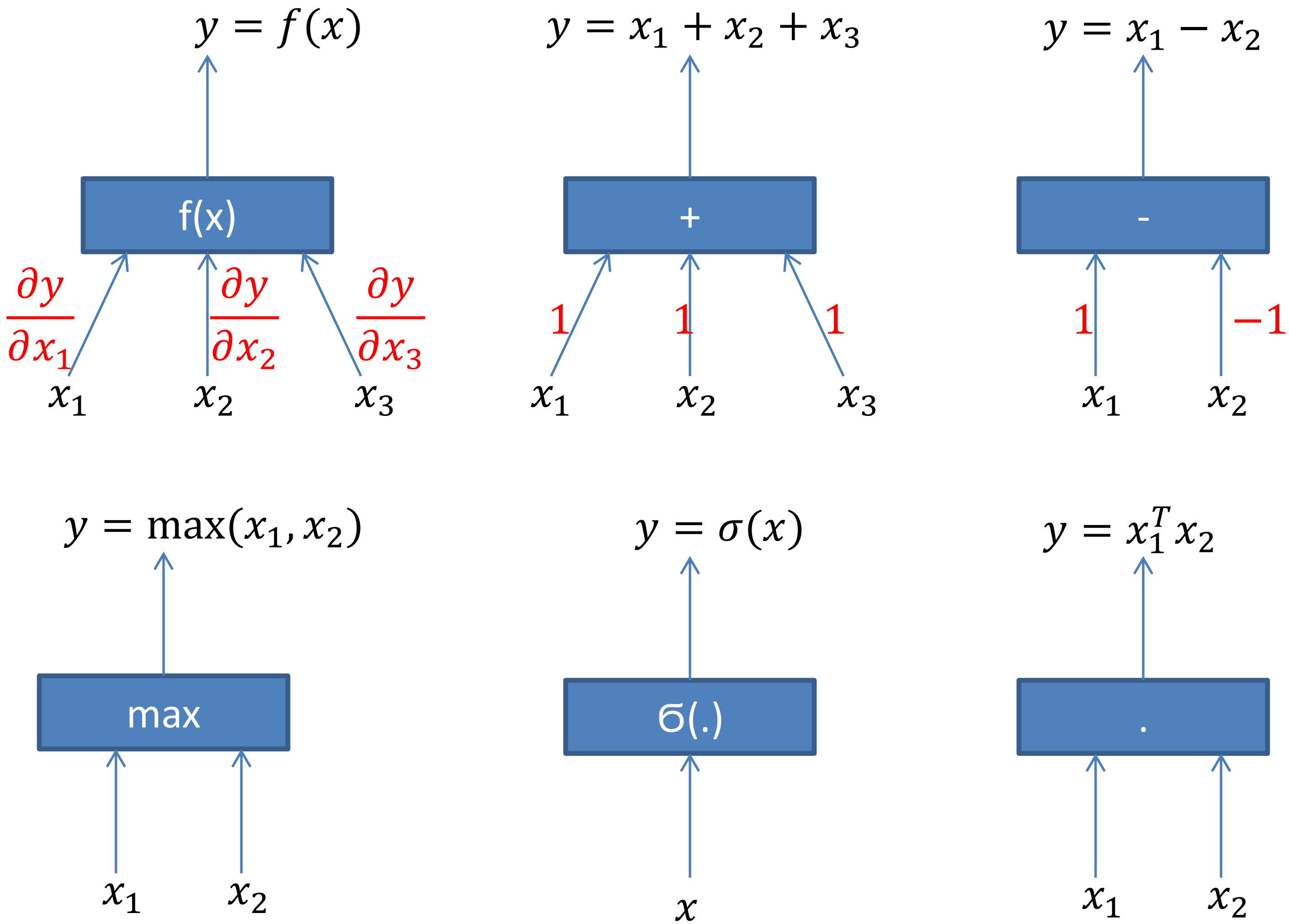
Review of backprop: Basic Operations



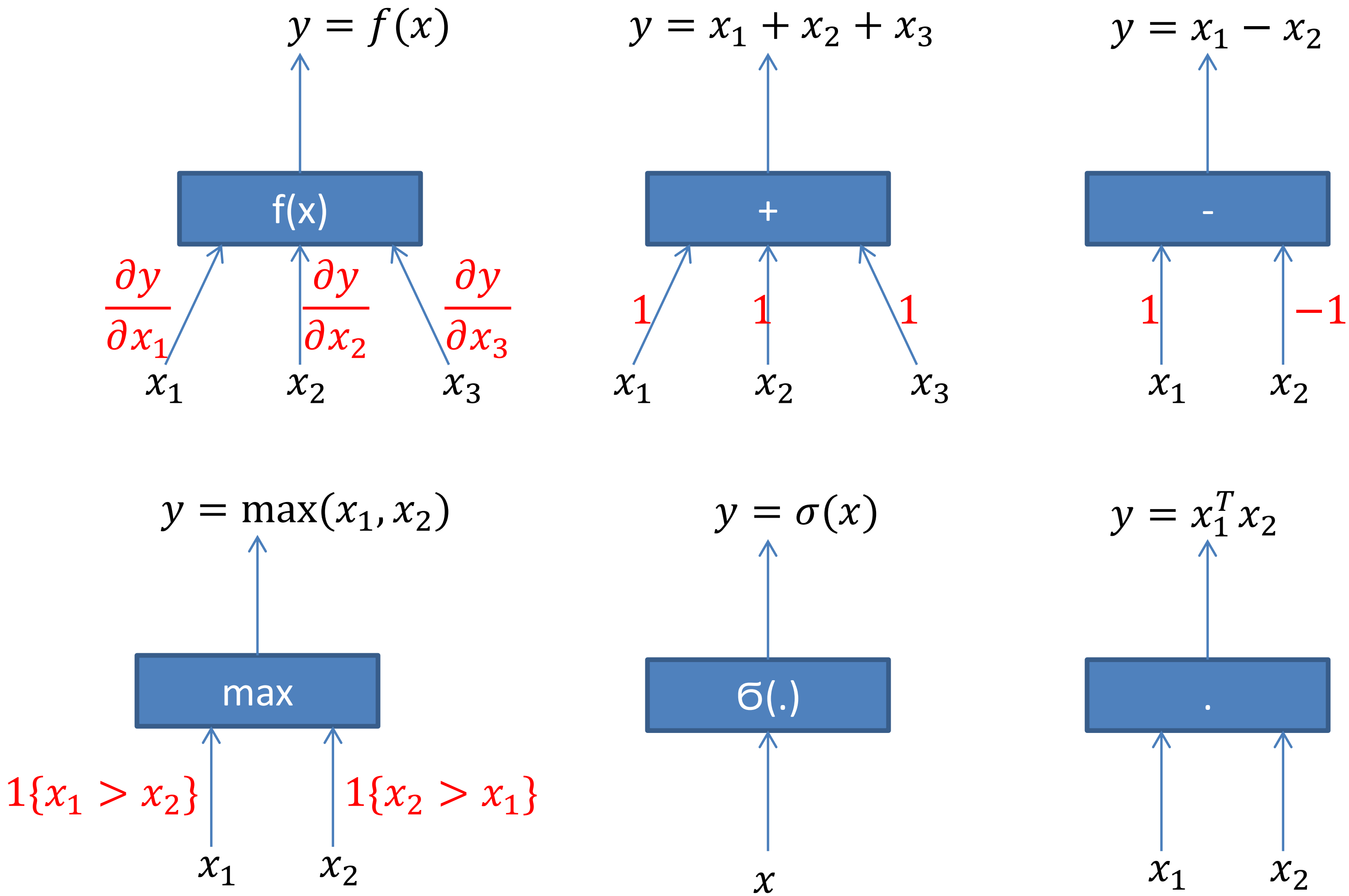
Review of backprop: Basic Operations



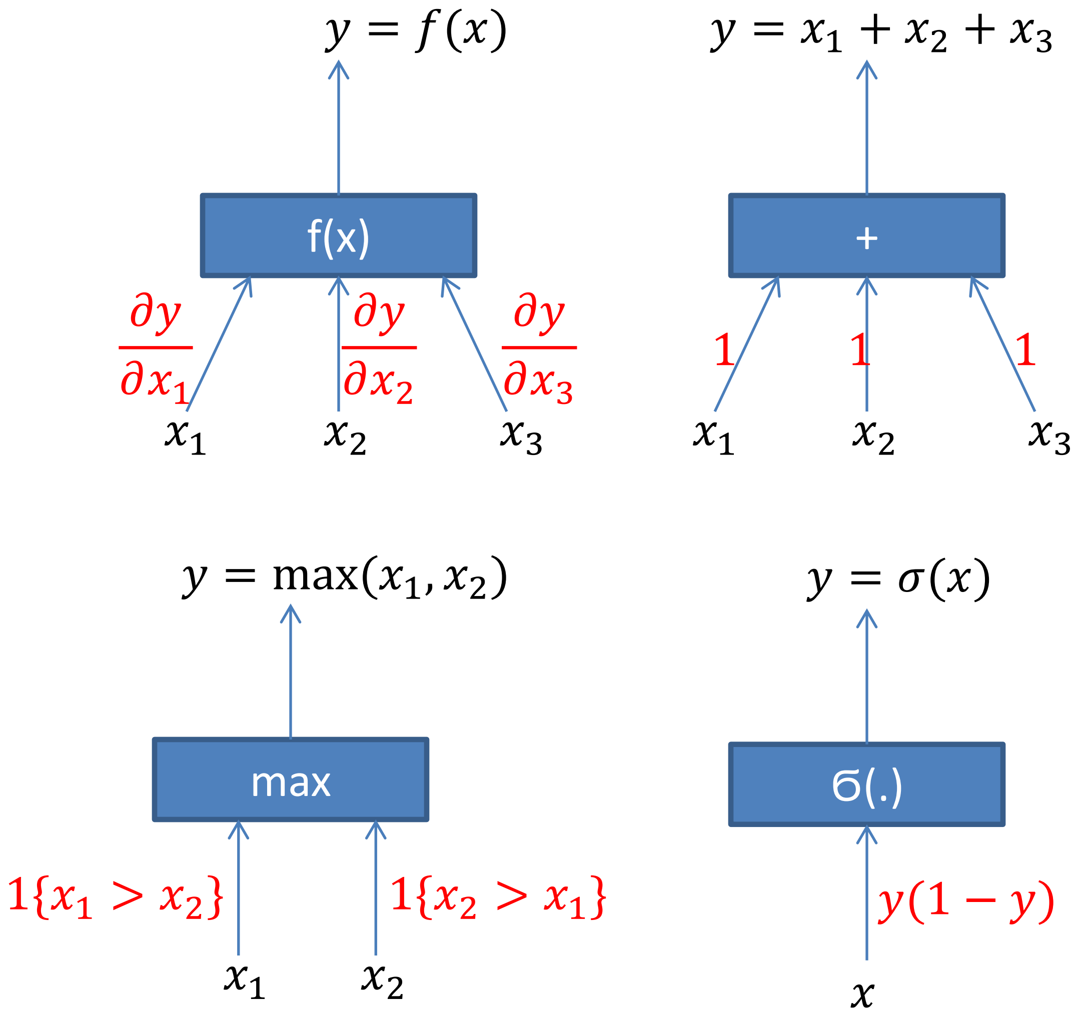
Review of backprop: Basic Operations



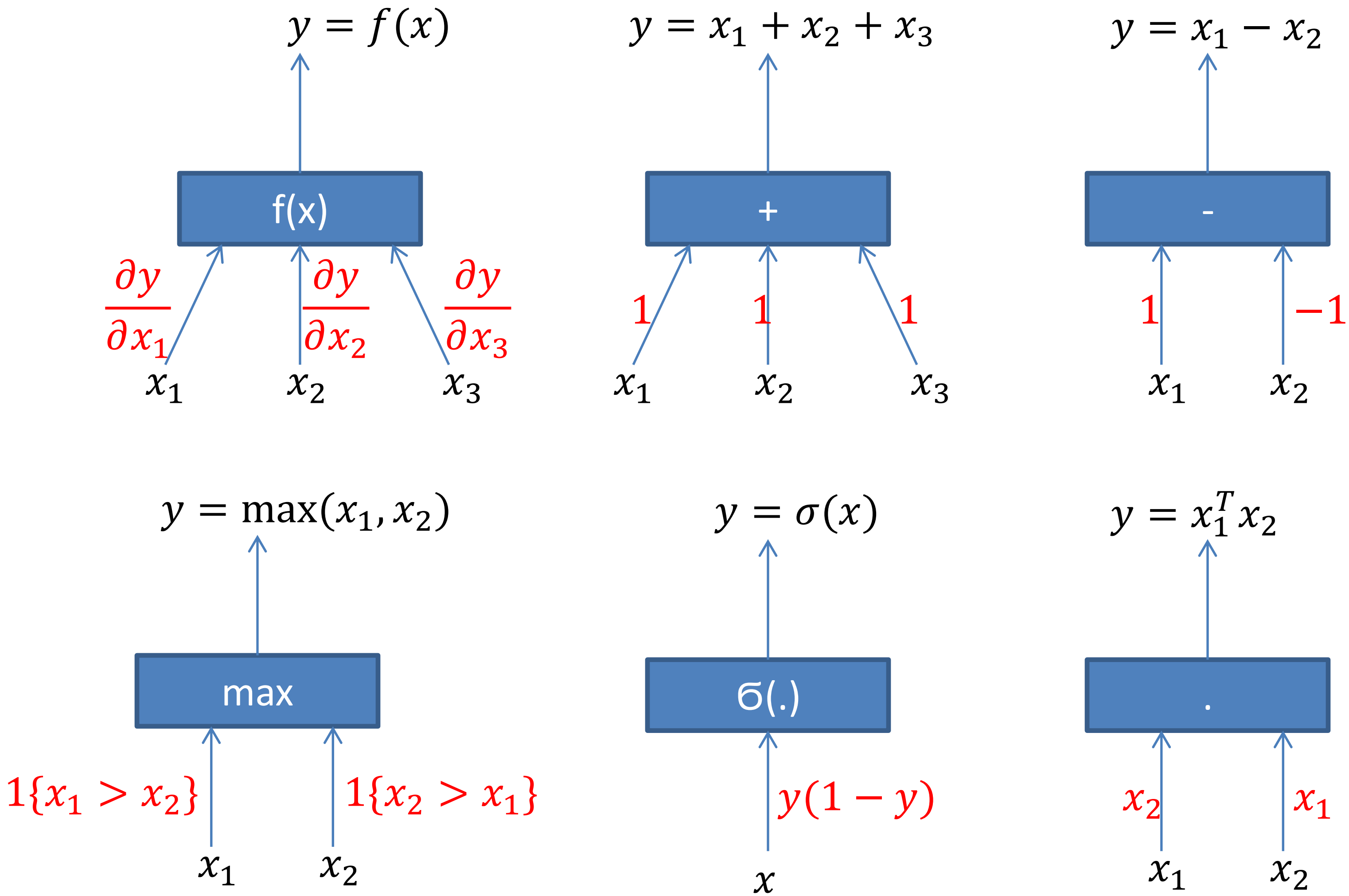
Review of backprop: Basic Operations



Review of backprop: Basic Operations



Review of backprop: Basic Operations



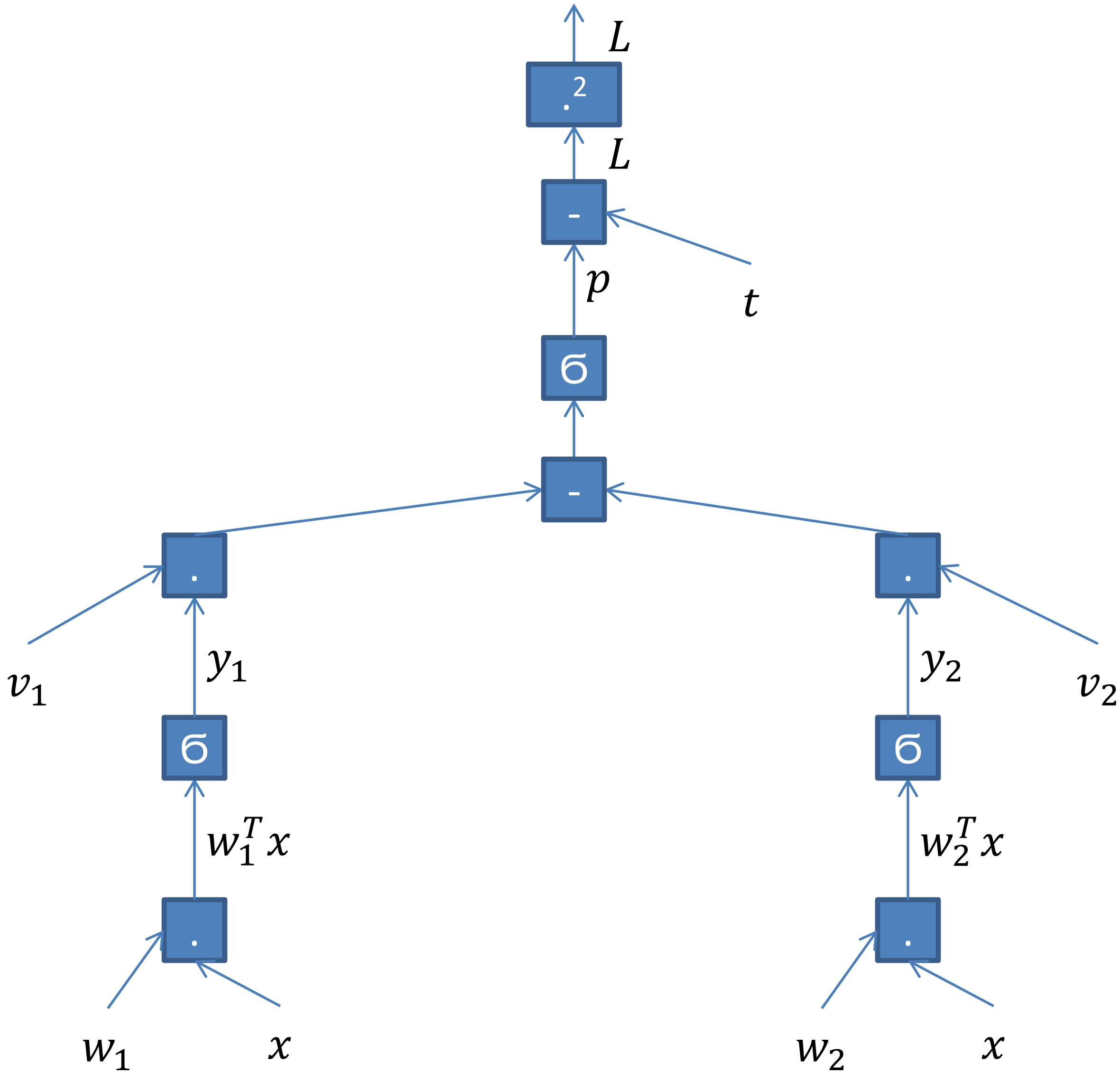
Note: Gradients may be vectors!

Review of backprop: Example

x: input
p: predicted value
t: true value
L’: (squared) loss
w_1, w_2, v_1, v_2: parameters to learn

$$y_1 = \sigma(w_1^T x)$$
$$y_2 = \sigma(w_2^T x)$$
$$p = \sigma(v_1 y_1 - v_2 y_2)$$
$$L' = (p - t)^2$$

Review of backprop: Example



$$y_1 = \sigma(w_1^T x)$$

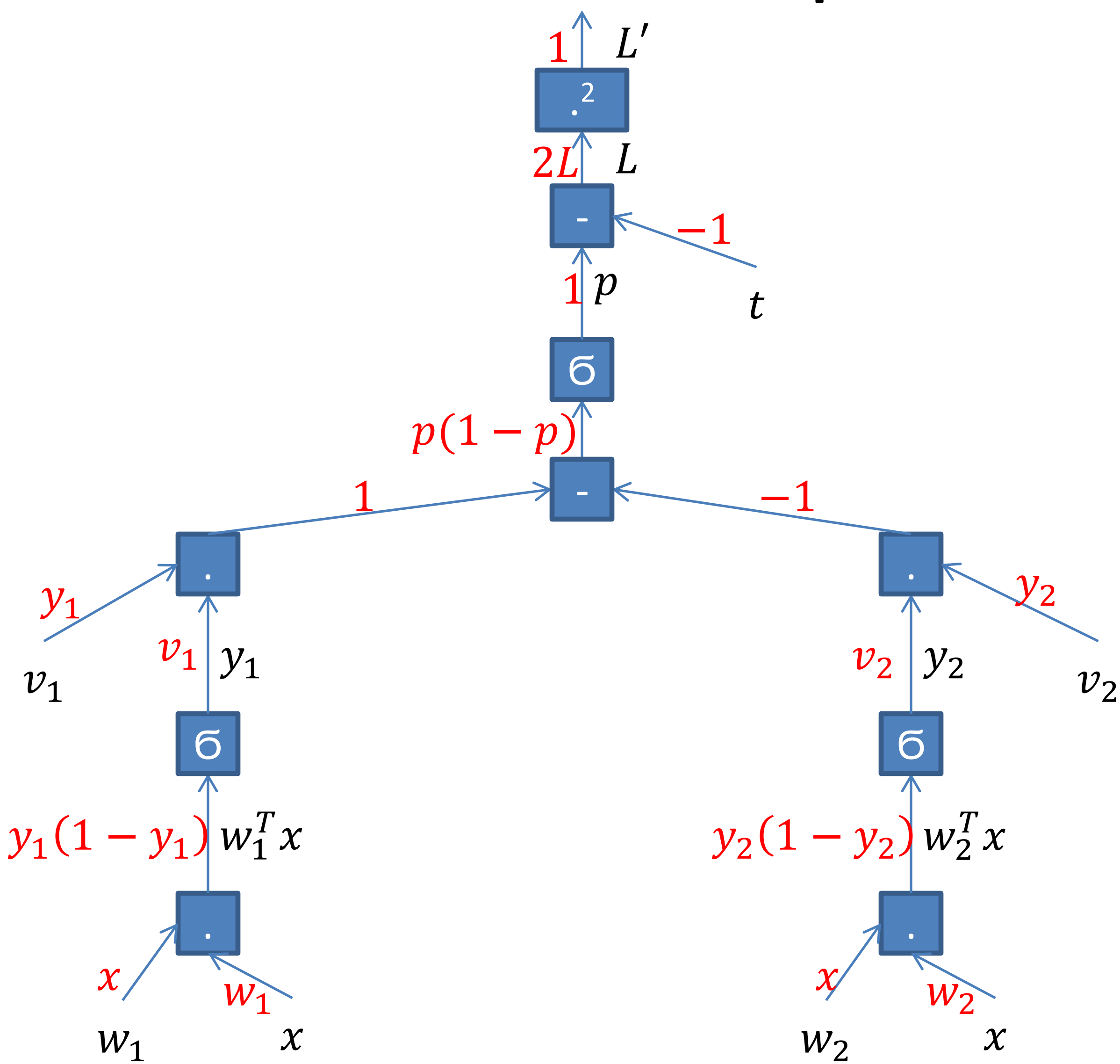
$$y_2 = \sigma(w_2^T x)$$

$$p = \sigma(v_1 y_1 - v_2 y_2)$$

$$L' = (p - t)^2$$

x: input
p: predicted value
t: true value
L': (squared) loss
w_1, w_2, v_1, v_2: parameters

Review of backprop: Example



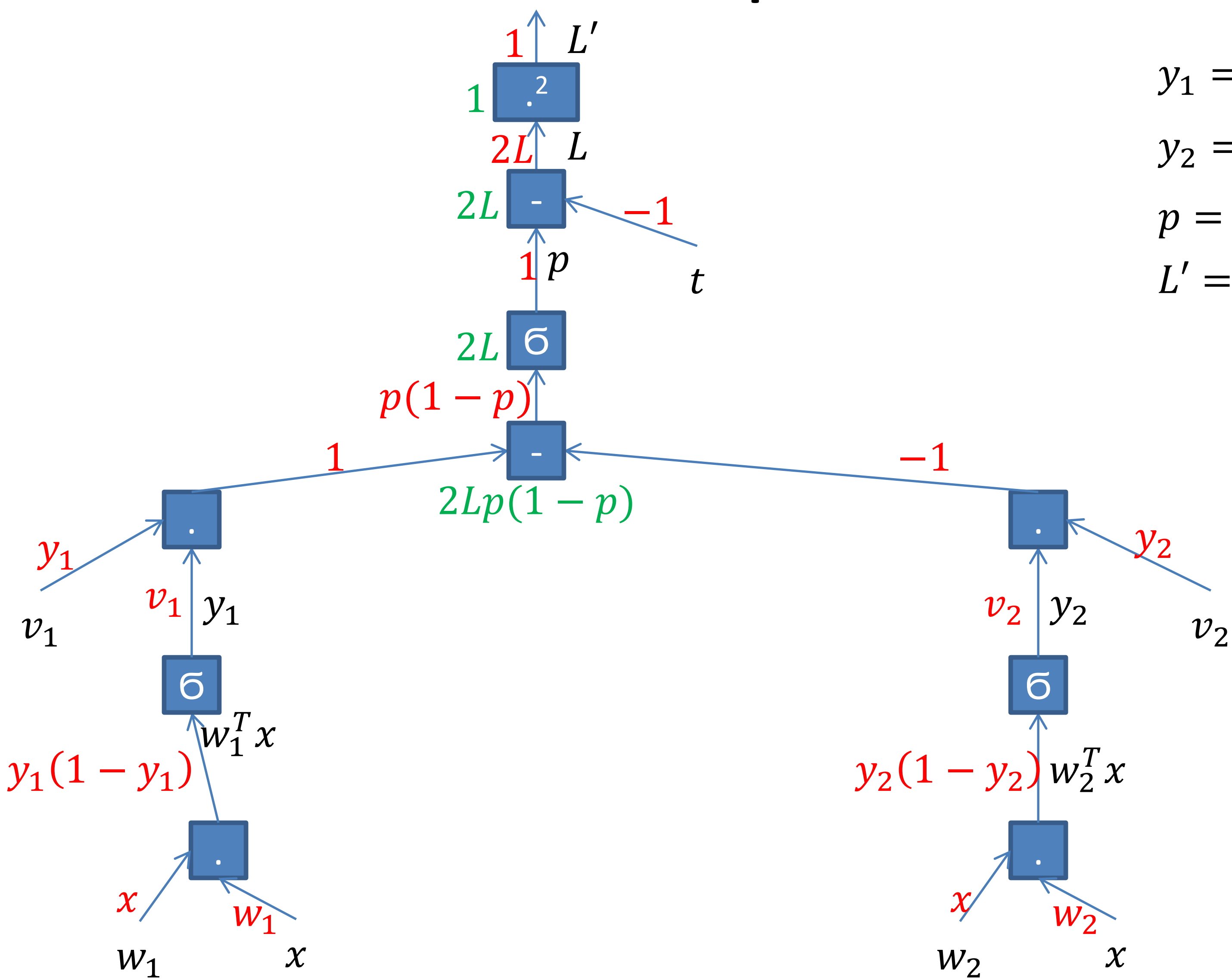
$$y_1 = \sigma(w_1^T x)$$

$$y_2 = \sigma(w_2^T x)$$

$$p = \sigma(v_1 y_1 - v_2 y_2)$$

$$L' = (p - t)^2$$

Review of backprop: Example



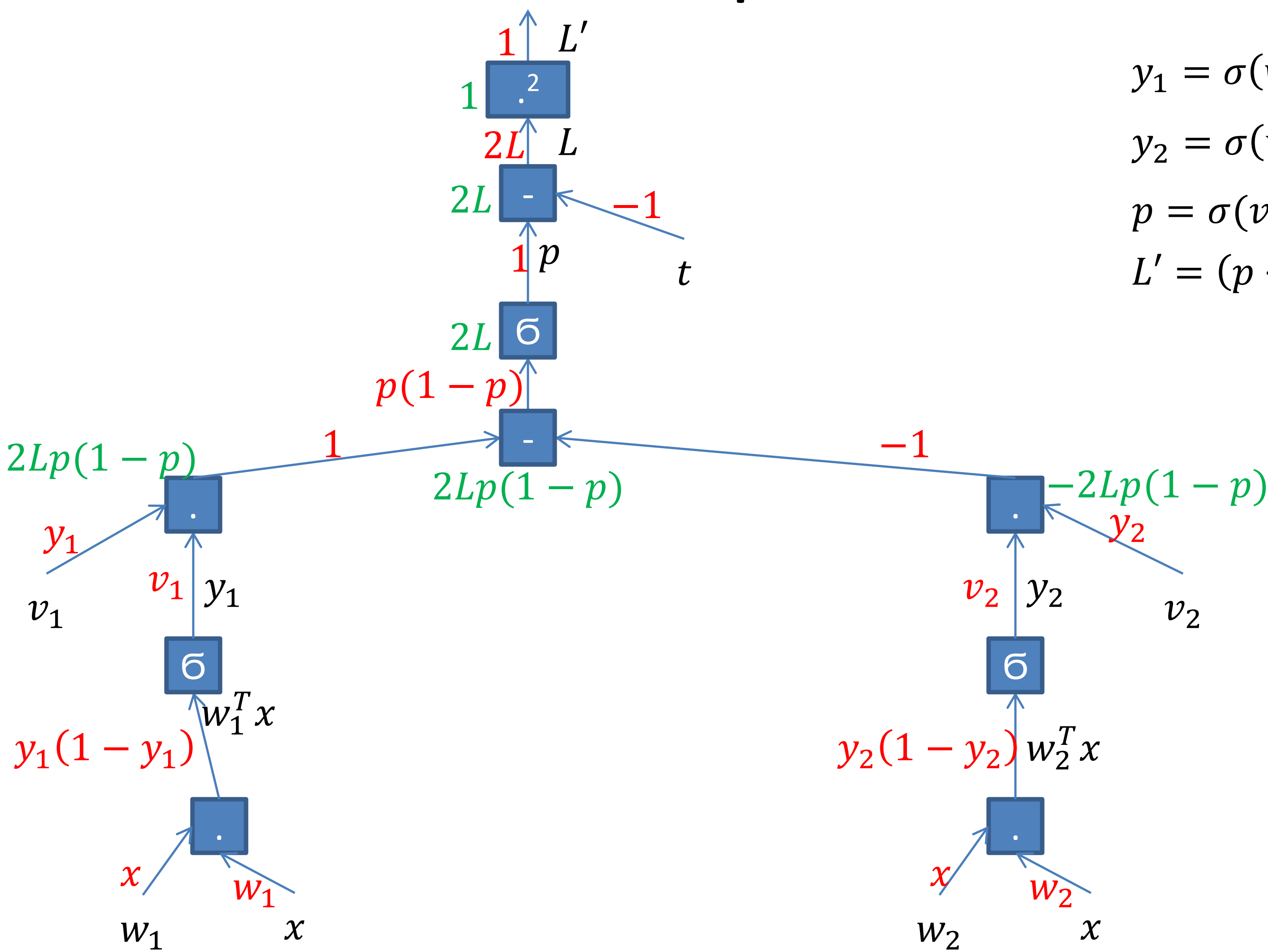
$y_1 = \sigma(w_1^T x)$

$y_2 = \sigma(w_2^T x)$

$p = \sigma(v_1 y_1 - v_2 y_2)$

$L' = (p - t)^2$

Review of backprop: Example



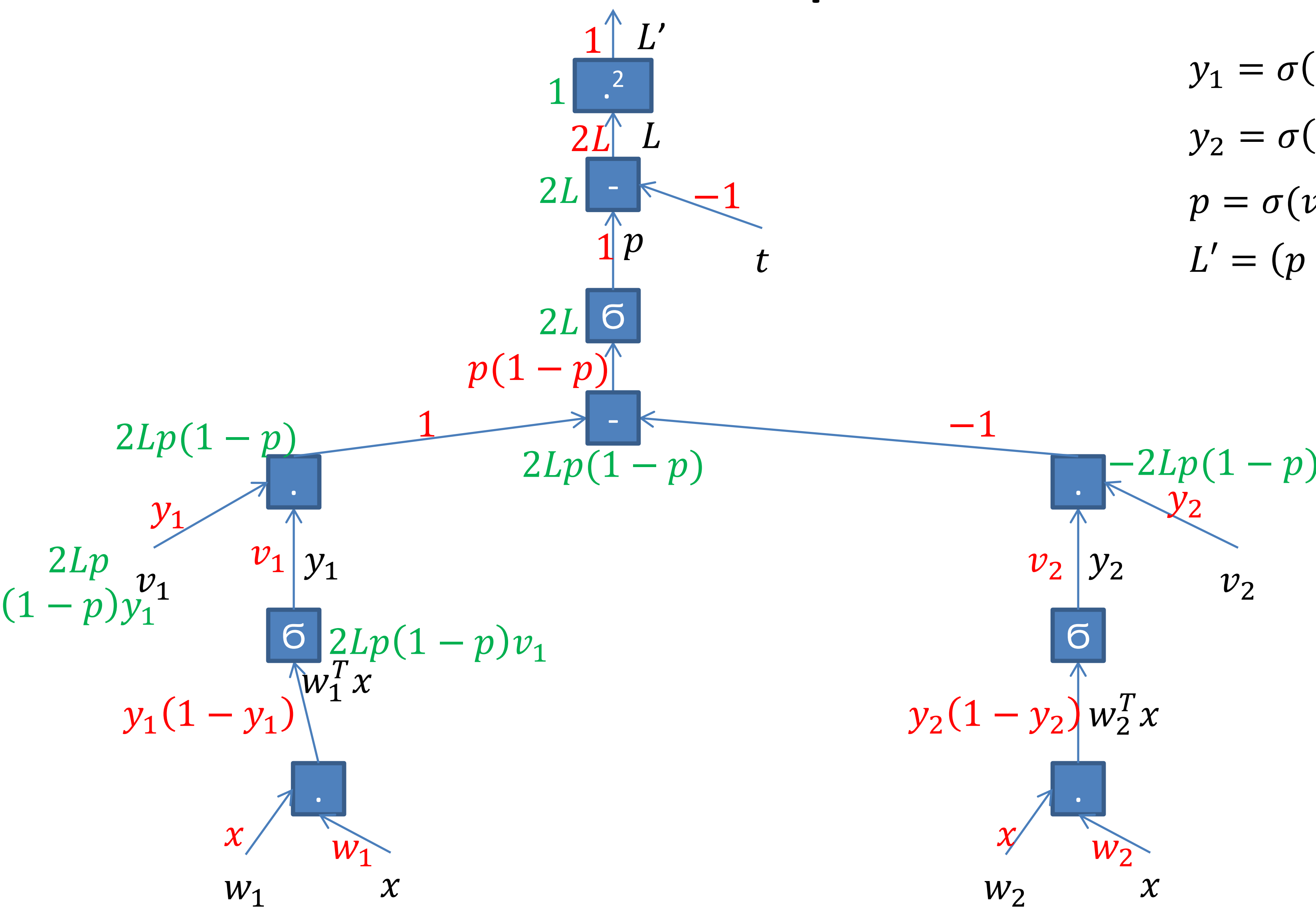
$$y_1 = \sigma(w_1^T x)$$

$$y_2 = \sigma(w_2^T x)$$

$$p = \sigma(v_1 y_1 - v_2 y_2)$$

$$L' = (p - t)^2$$

Review of backprop: Example



$$y_1 = \sigma(w_1^T x)$$

$$y_2 = \sigma(w_2^T x)$$

$$p = \sigma(v_1 y_1 - v_2 y_2)$$

$$L' = (p - t)^2$$

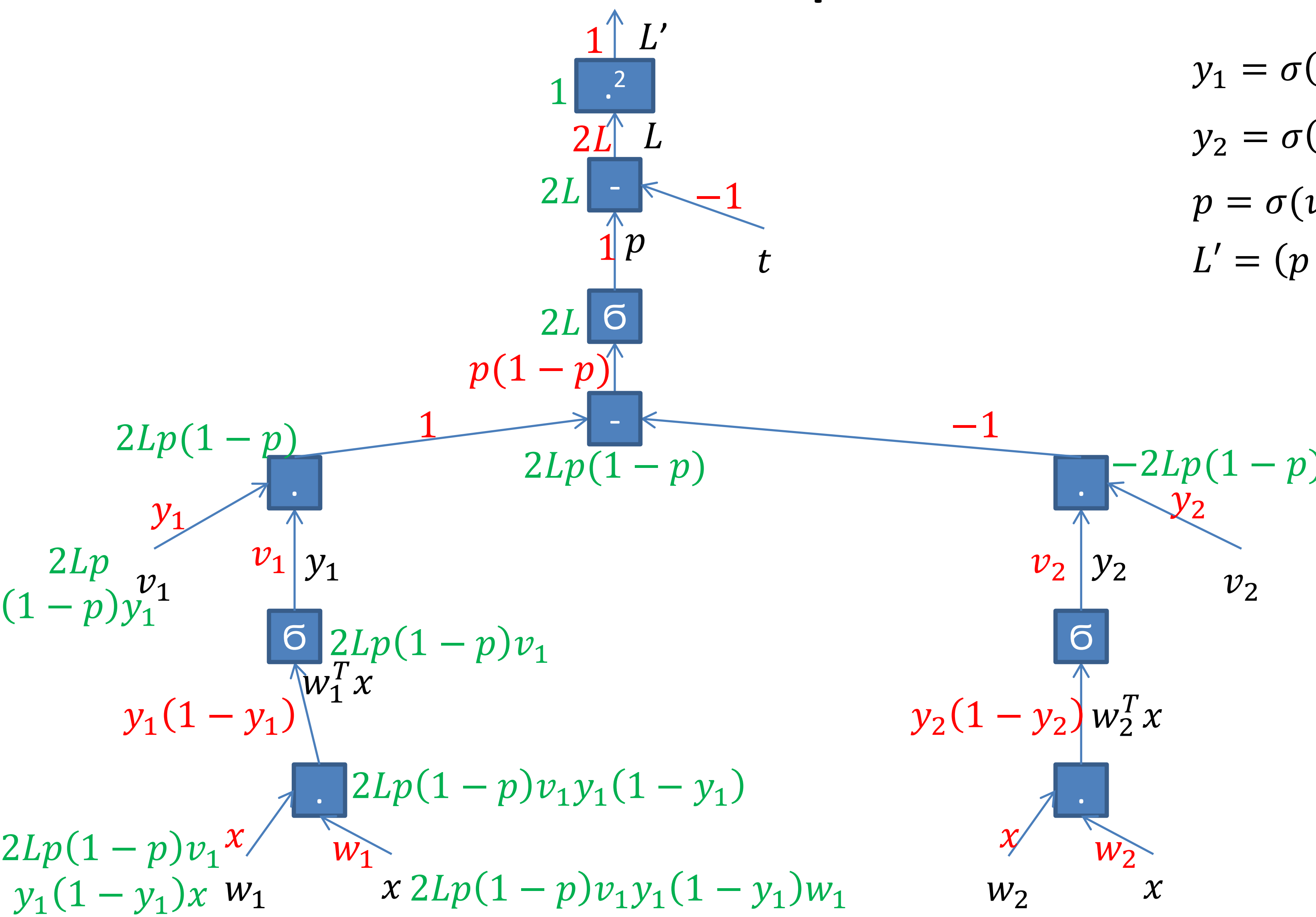
Review of backprop: Example

$$y_1 = \sigma(w_1^T x)$$

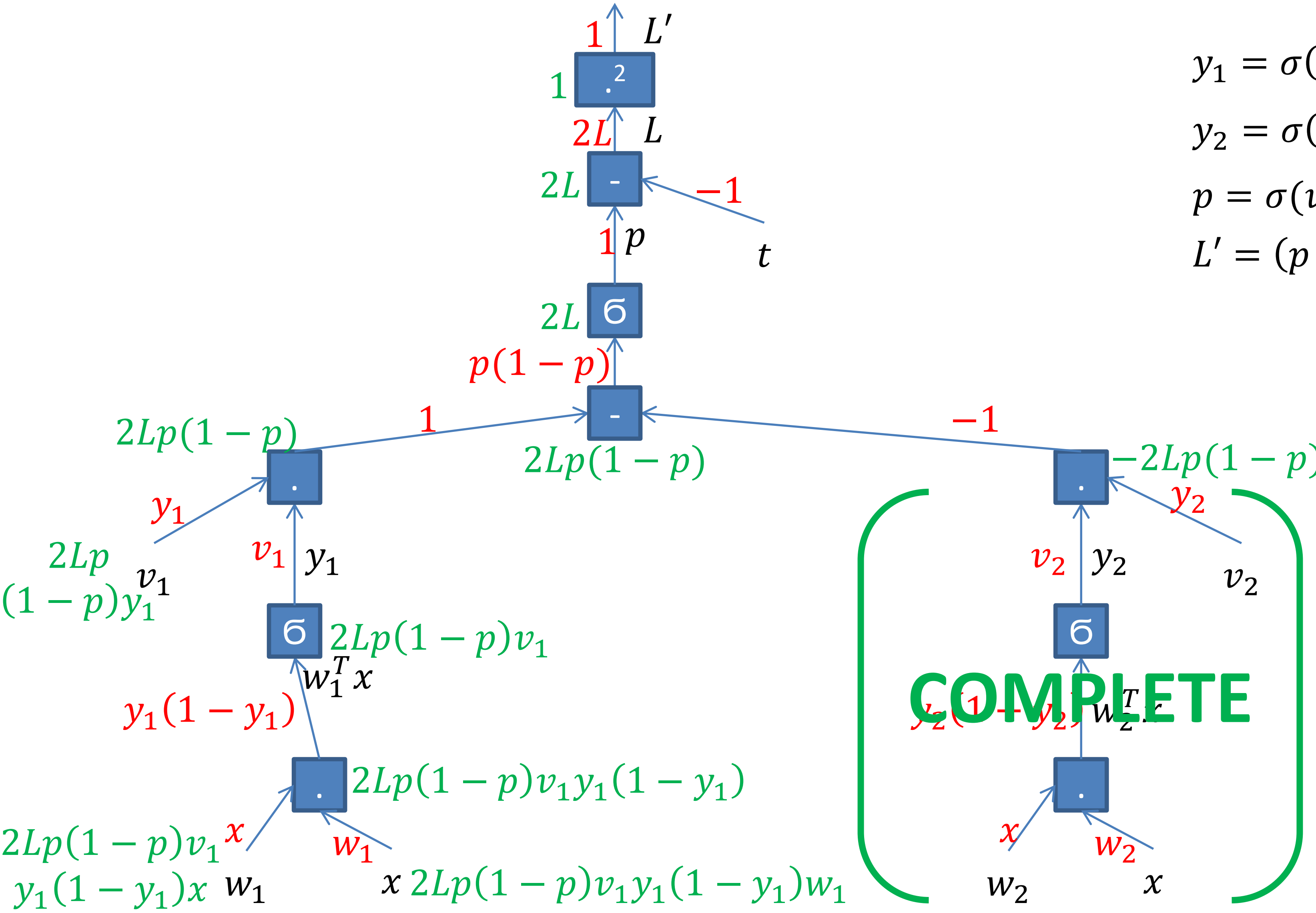
$$y_2 = \sigma(w_2^T x)$$

$$p = \sigma(v_1 y_1 - v_2 y_2)$$

$$L' = (p - t)^2$$



Review of backprop: Example



$y_1 = \sigma(w_1^T x)$

$y_2 = \sigma(w_2^T x)$

$p = \sigma(v_1 y_1 - v_2 y_2)$

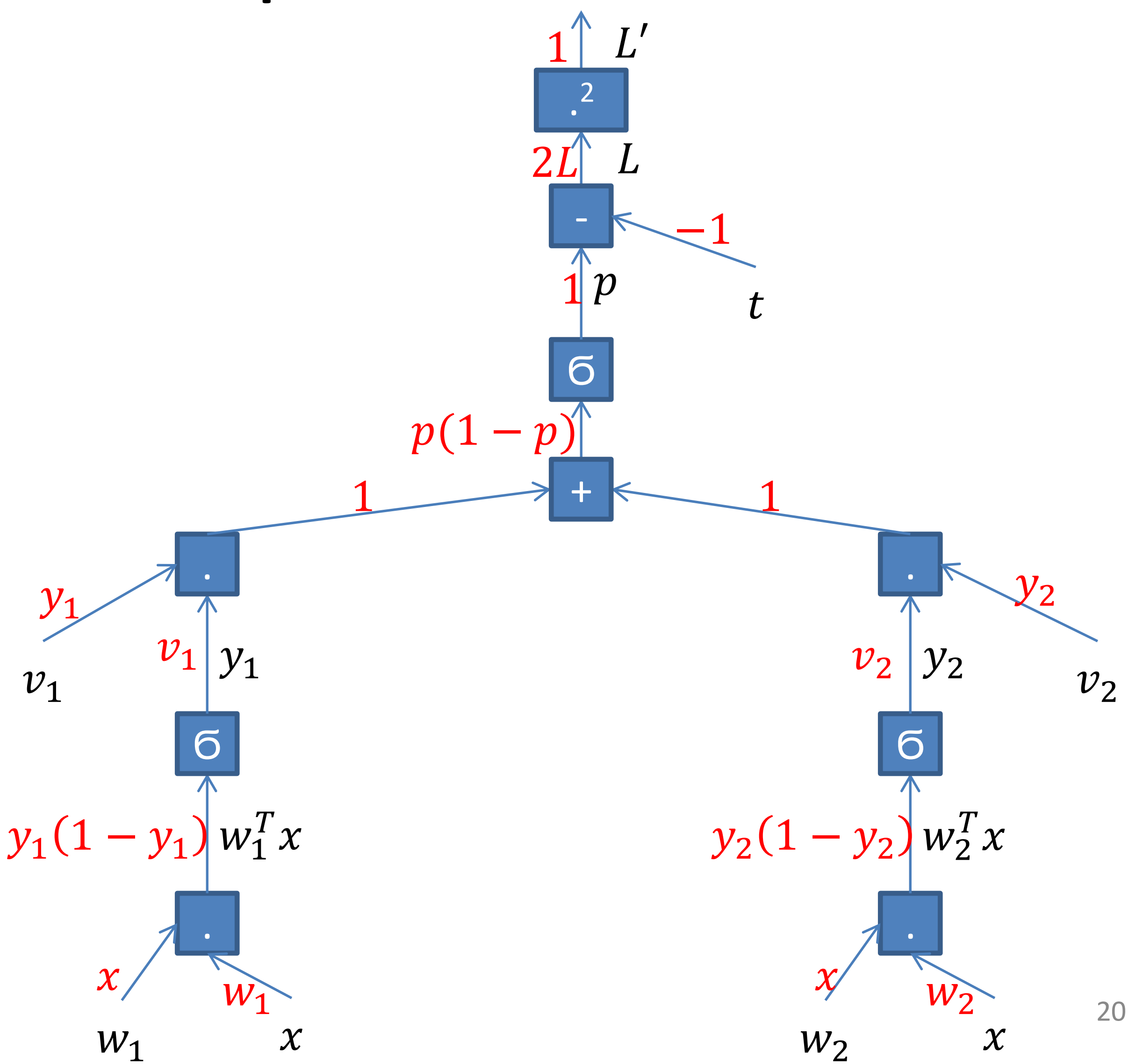
$L' = (p-t)^2$

Topics

- Review of backprop
  - Basic operations
  - Class example
- Additional complexity
  - Shared parameters
  - Dealing with vectors (optional)
- Recurrent Neural Net (RNN)
  - Motivation
  - Simple backprop (vector one optional)
  - Demo

Additional complexity: Shared parameters

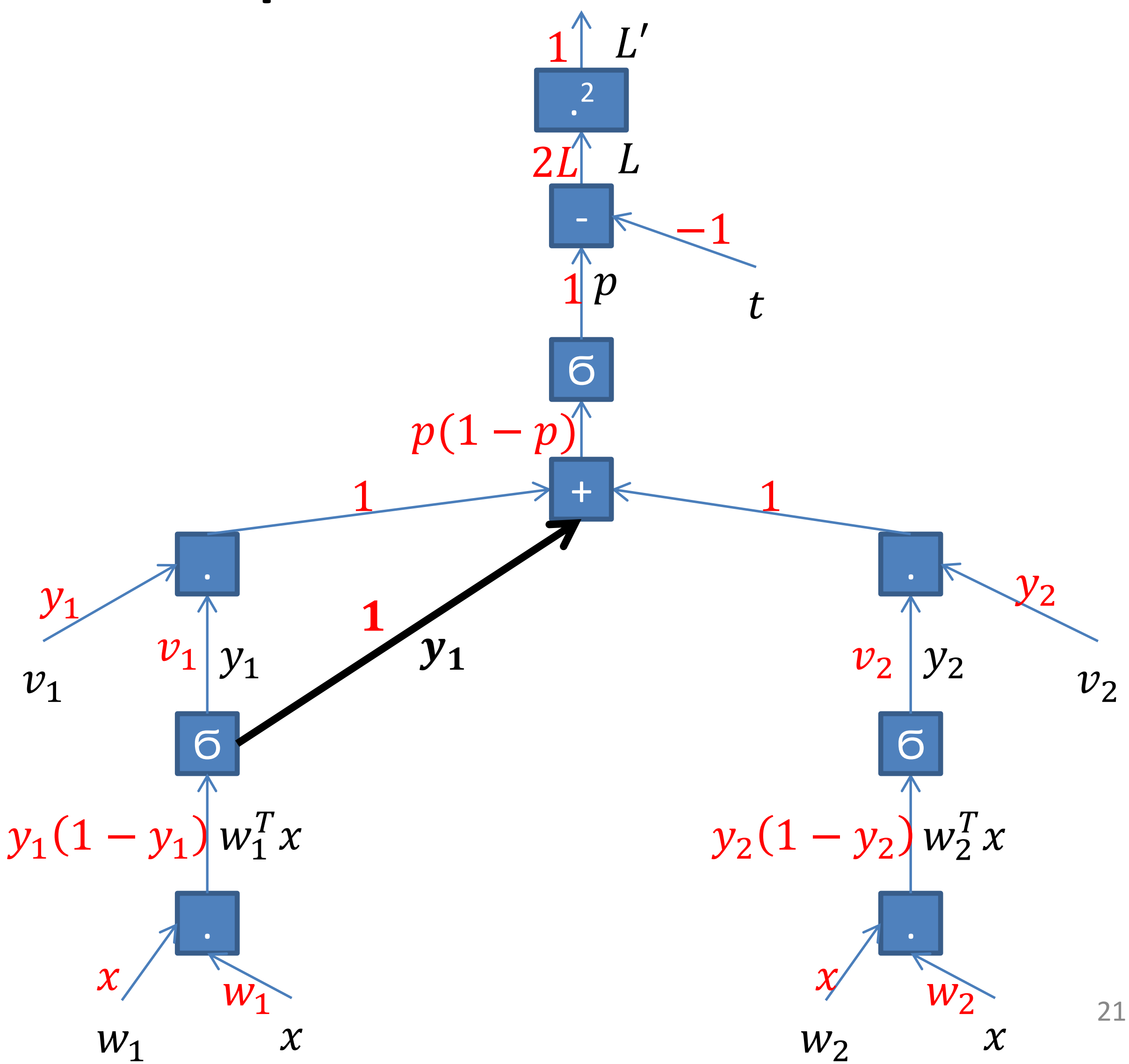
So far, each computed output goes to 1 unit only



Additional complexity: Shared parameters

So far, each computed output goes to 1 unit only

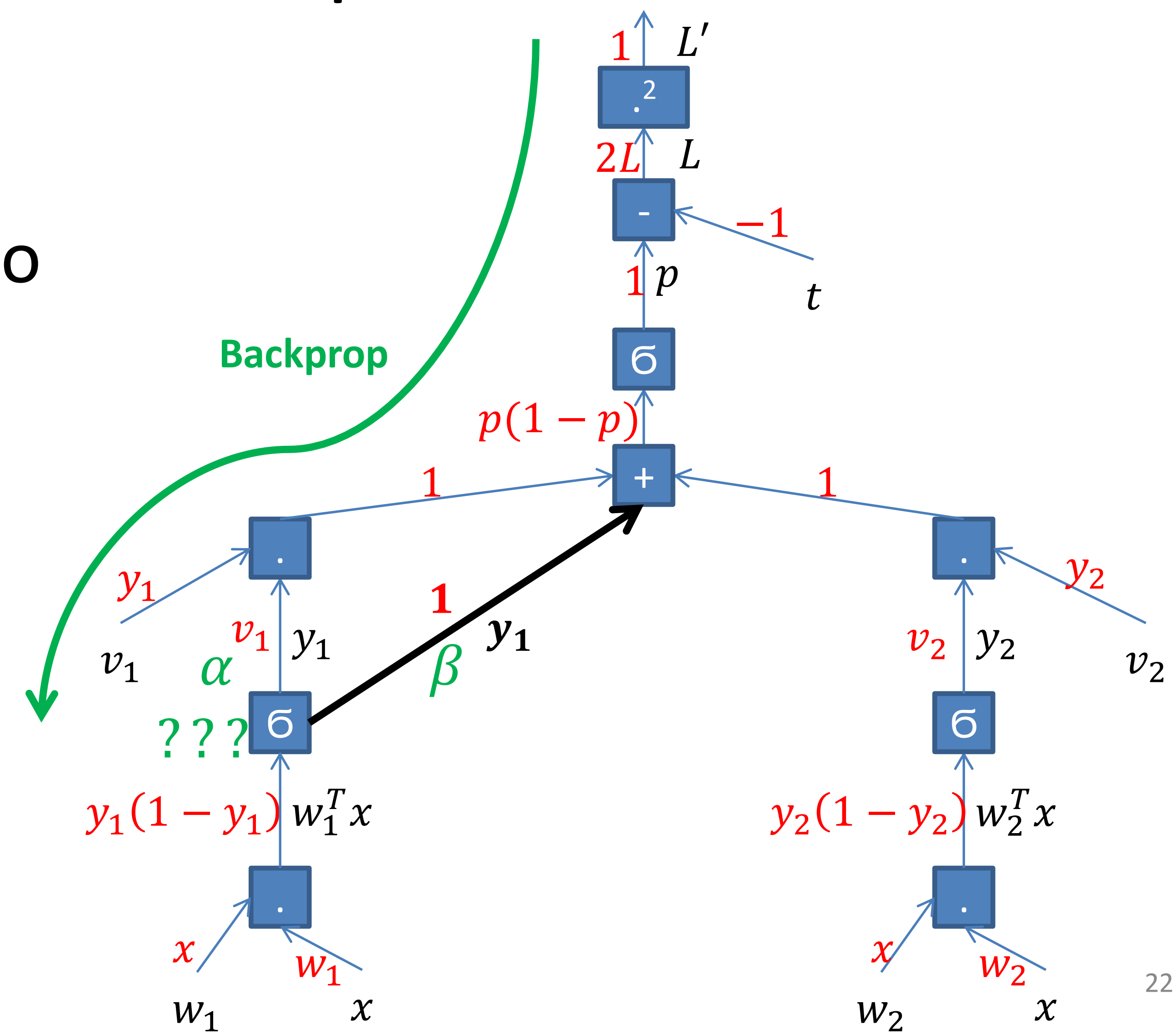
What if?



Additional complexity: Shared parameters

So far, each computed output goes to 1 unit only

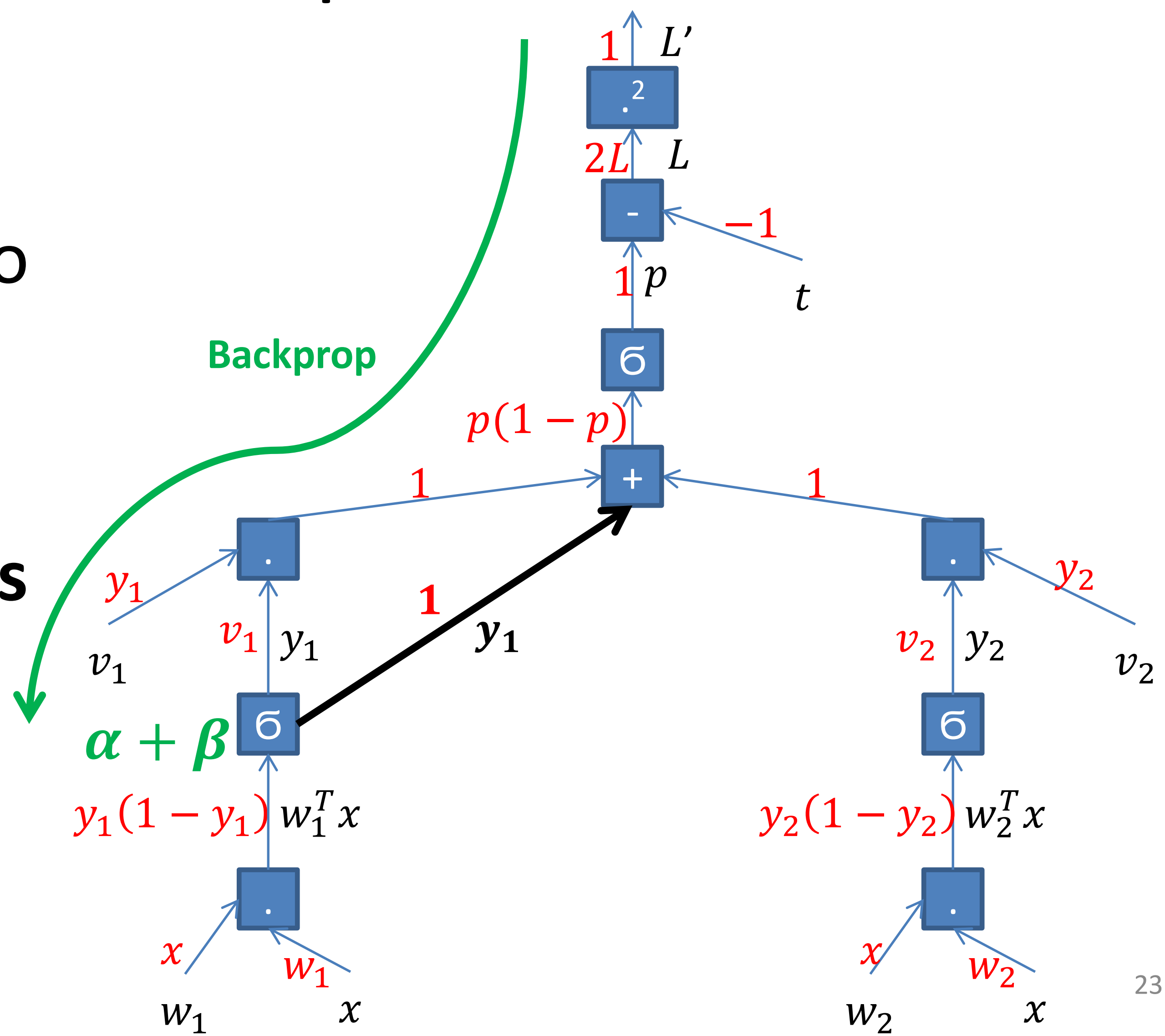
What if?



Additional complexity: Shared parameters

So far, each computed output goes to 1 unit only

ADD gradients



Additional complexity: Shared parameters

But what if same output goes to multiple units?

Example: $\boldsymbol{z = w^2(w - 3)}$, compute $\frac{dz}{dw}$

Additional complexity: Shared parameters

But what if same output goes to multiple units?

Example: $\boldsymbol{z} = \boldsymbol{w^2}(\boldsymbol{w} - \mathbf{3})$, compute $\frac{dz}{dw}$

By hand, use product rule:

$$\frac{dz}{dw} = \frac{dw^2}{dw}(w-3) + w^2 \frac{d(w-3)}{dw}$$
$$= 2w(w-3) \quad + w^2$$
$$= 3w^2 - 6w$$

Additional complexity: Shared parameters

But what if same output goes to multiple units?

Example: $\boldsymbol{z = w^2(w - 3)}$, compute $\frac{dz}{dw}$

By hand, use product rule:

$$\frac{dz}{dw} = \frac{dw^2}{dw}(w-3) + w^2\frac{d(w-3)}{dw}$$
$$= 2w(w-3) \quad + w^2$$
$$= 3w^2 - 6w$$

Computational steps:

$x = w^2$

$y = w - 3$

$z = x.y$

Additional complexity: Shared parameters

But what if same output goes to multiple units?

Example: $\boldsymbol{z = w^2(w - 3)}$, compute $\frac{dz}{dw}$

By hand, use product rule:

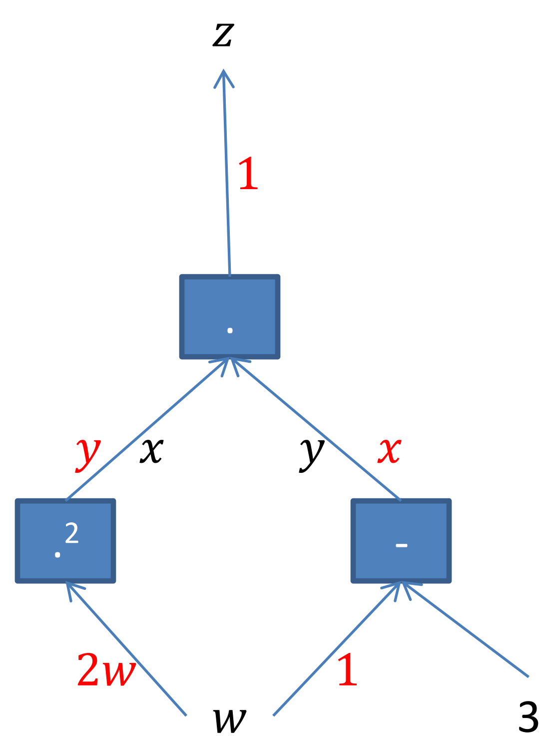
$$\frac{dz}{dw} = \frac{dw^2}{dw}(w-3) + w^2\frac{d(w-3)}{dw}$$
$$= 2w(w-3) \quad + w^2$$
$$= 3w^2 - 6w$$

Computational steps:

$x = w^2$

$y = w - 3$

$z = x.y$



Additional complexity: Shared parameters

But what if same output goes to multiple units?

Example: $z = w^2(w - 3)$, compute $\frac{dz}{dw}$

By hand, use product rule:

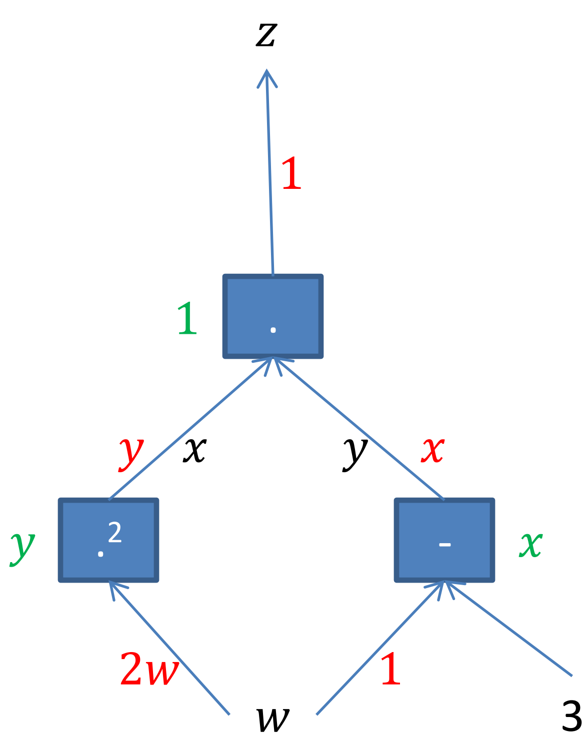
$$\frac{dz}{dw} = \frac{dw^2}{dw}(w-3) + w^2\frac{d(w-3)}{dw}$$
$$= 2w(w-3) \quad + w^2$$
$$= 3w^2 - 6w$$

Computational steps:

$x = w^2$

$y = w - 3$

$z = x.y$



Additional complexity: Shared parameters

But what if same output goes to multiple units?

Example: $\boldsymbol{z = w^2(w-3)}$, compute $\frac{dz}{dw}$

By hand, use product rule:

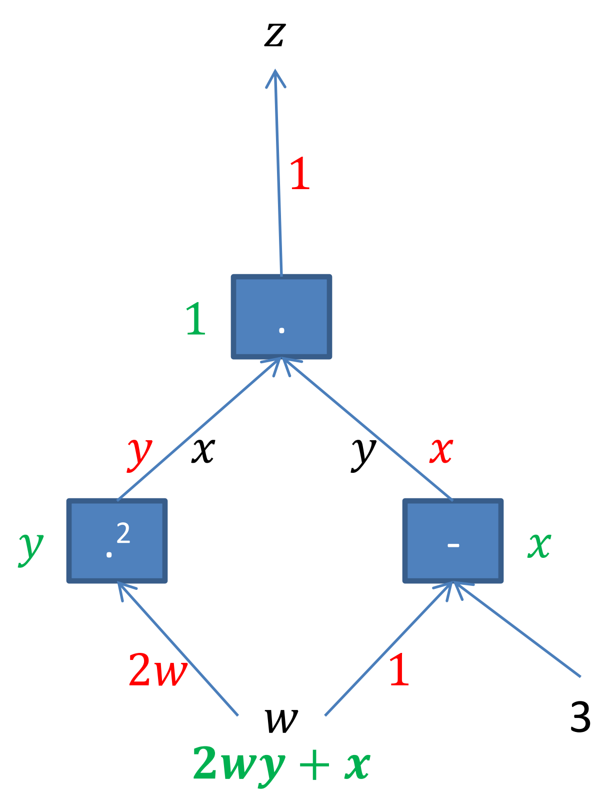
$$\frac{dz}{dw} = \frac{dw^2}{dw}(w-3) + w^2\frac{d(w-3)}{dw}$$
$$= 2w(w-3) \quad + w^2$$
$$= 3w^2 - 6w$$

Computational steps:

$x = w^2$

$y = w - 3$

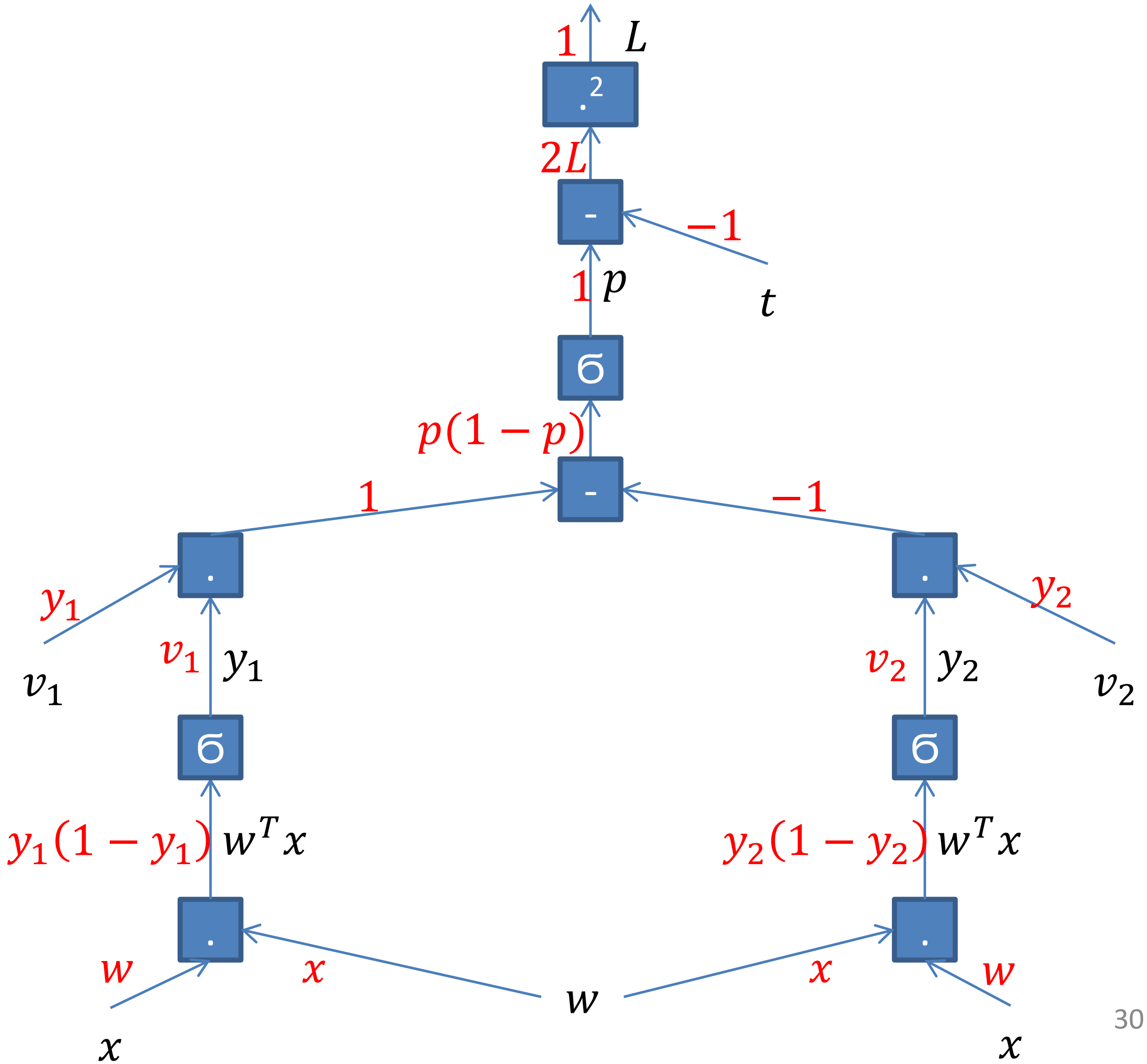
$z = x.y$



Additional complexity: Shared parameters

So far, each computed output goes to 1 unit only

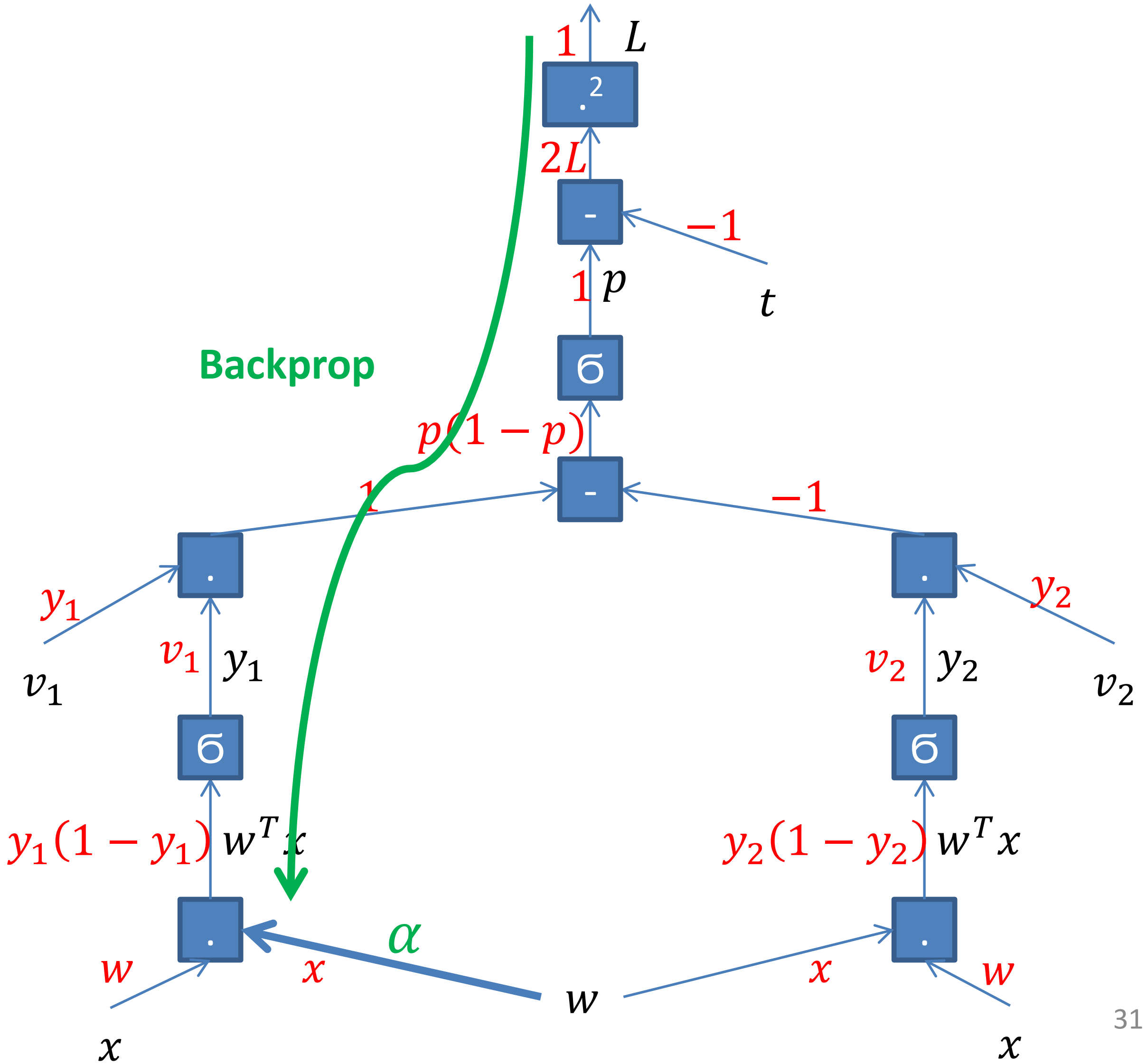
Similarly…



Additional complexity: Shared parameters

So far, each computed output goes to 1 unit only

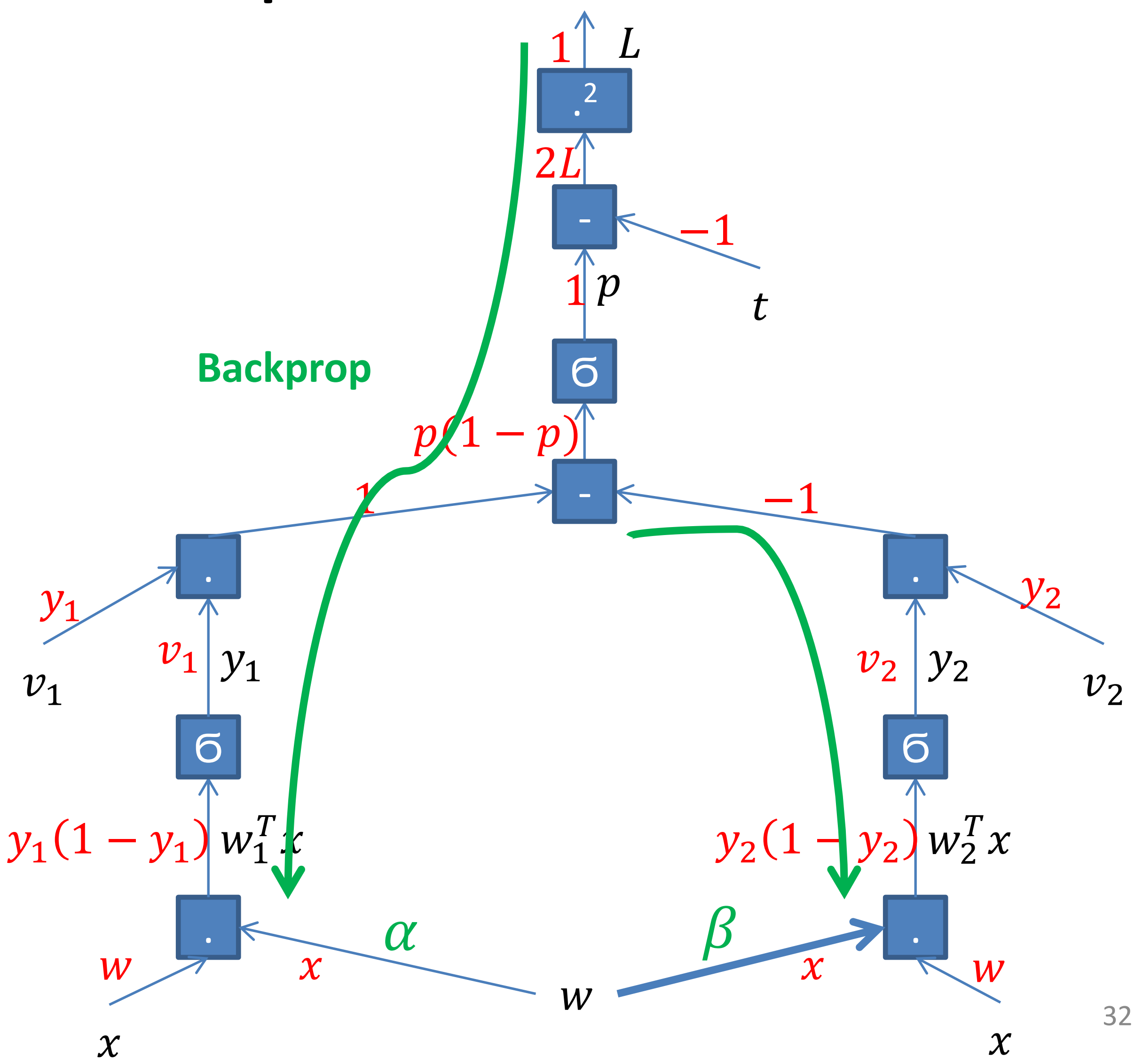
Similarly…



Additional complexity: Shared parameters

So far, each computed output goes to 1 unit only

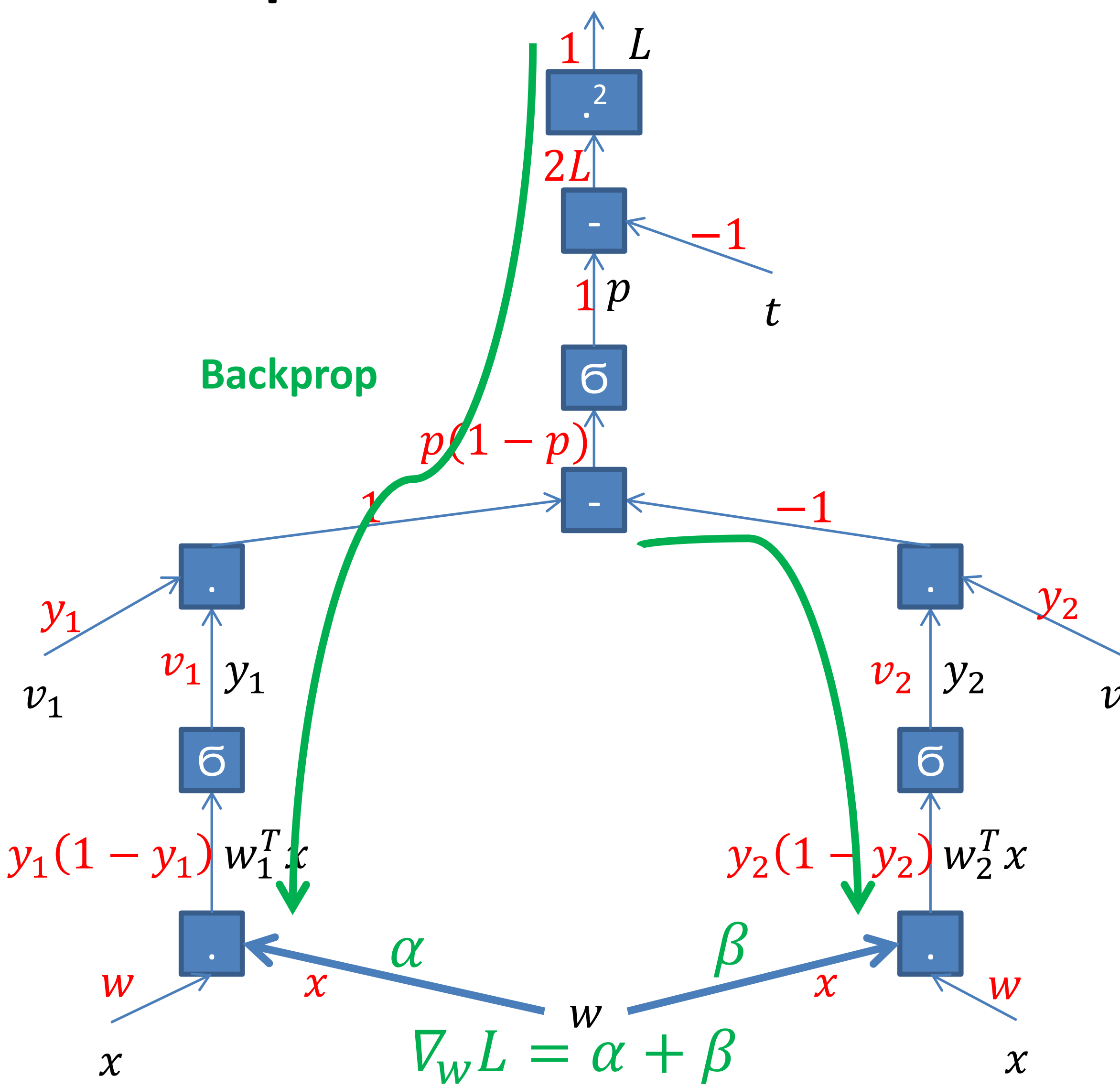
Similarly…



Additional complexity: Shared parameters

So far, each computed output goes to 1 unit only

Similarly…



Additional complexity: Shared parameters

- Why do we add when output/parameter replicated?

Proof

$$L(y', y'')$$

$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial y'} . \frac{\partial y'}{\partial y} + \frac{\partial L}{\partial y''} . \frac{\partial y''}{\partial y}$$

Additional complexity: Shared parameters

- Why do we add when output/parameter replicated?

Proof

$L(y', y'')$

$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial y'} . \frac{\partial y'}{\partial y} + \frac{\partial L}{\partial y''} . \frac{\partial y''}{\partial y}$$

Since $y = y' = y''$, it follows:

Additional complexity: Shared parameters

- Why do we add when output/parameter replicated?

Proof

$L(y', y'')$

$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial y'} . \frac{\partial y'}{\partial y} + \frac{\partial L}{\partial y''} . \frac{\partial y''}{\partial y}$$

Since $y = y' = y''$, it follows:

$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial y'} + \frac{\partial L}{\partial y''}$$

Additional complexity: Shared parameters

- Why do we add when output/parameter replicated?

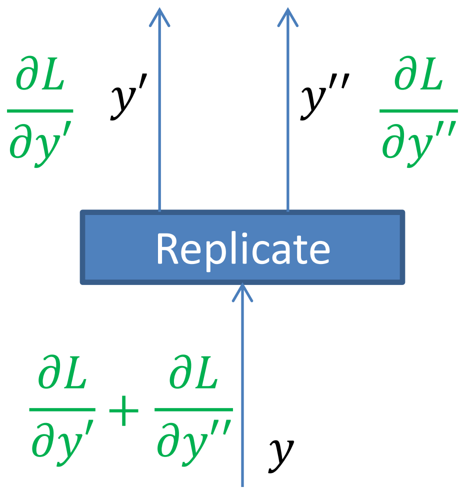
Proof

$$L(y', y'')$$

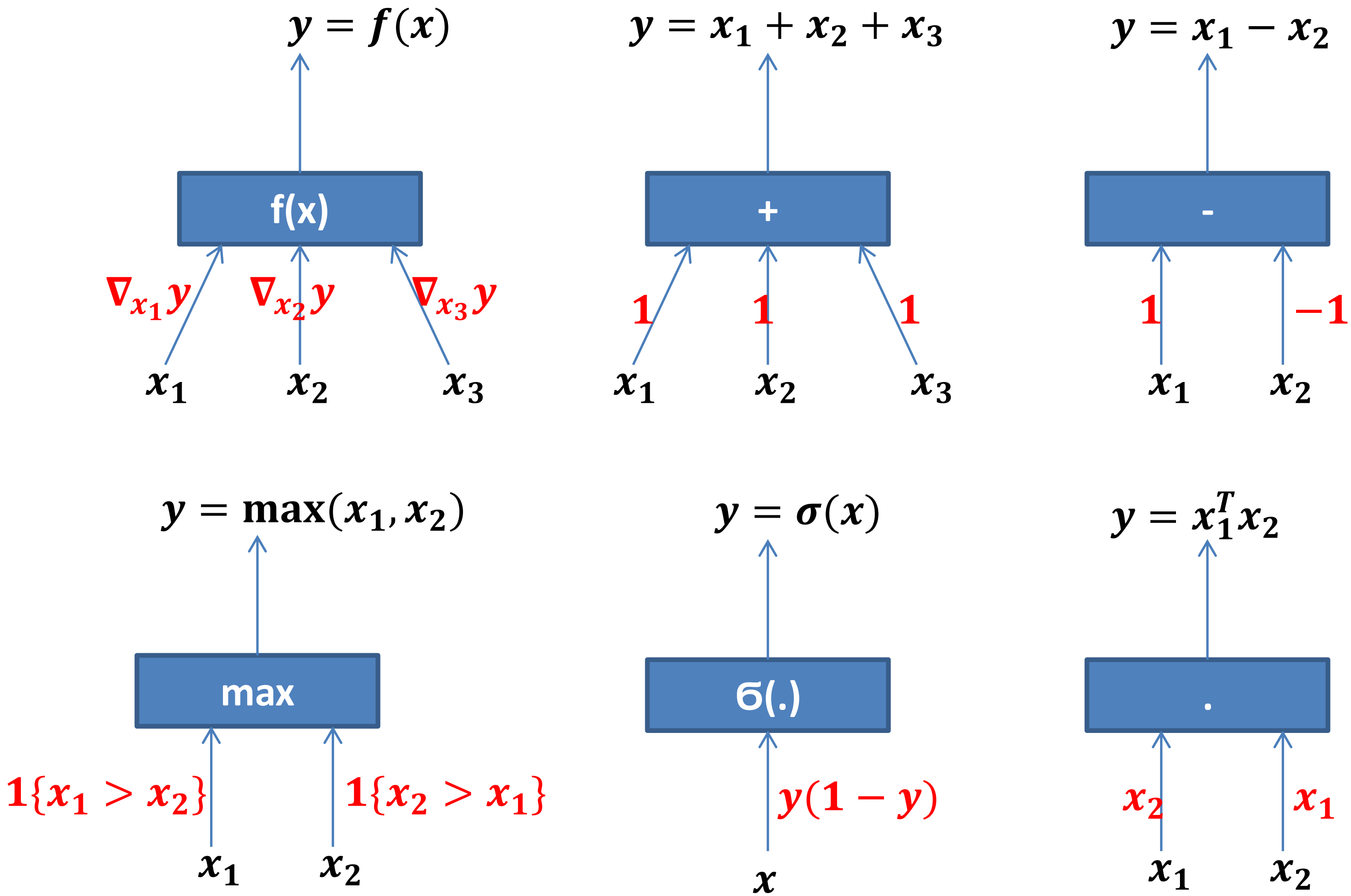
$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial y'} . \frac{\partial y'}{\partial y} + \frac{\partial L}{\partial y''} . \frac{\partial y''}{\partial y}$$

Since $y = y' = y''$, it follows:

$$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial y'} + \frac{\partial L}{\partial y''}$$

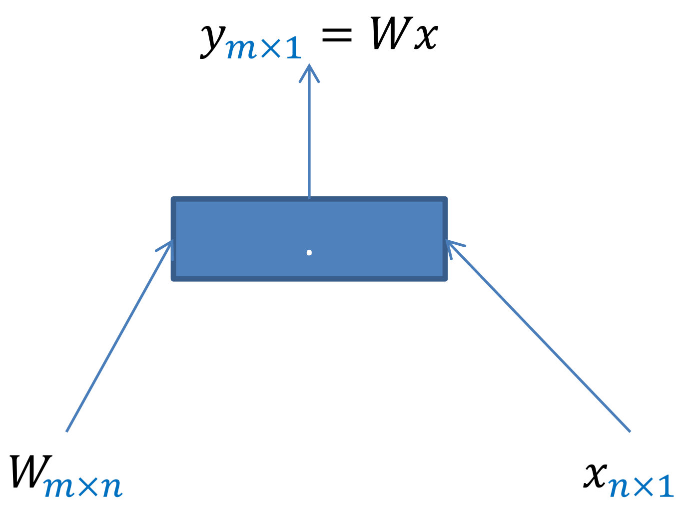


Additional complexity: Dealing with vectors (optional)

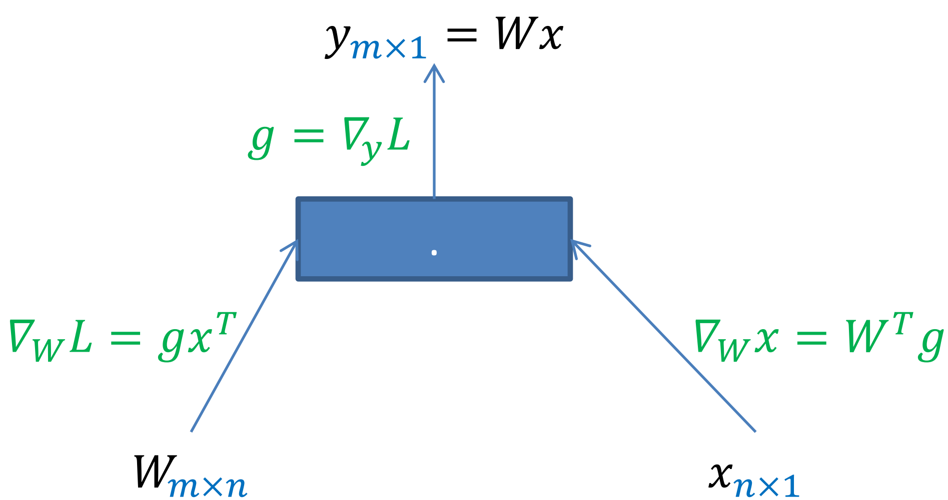


Note: All operations, including backprop, are component-wise!

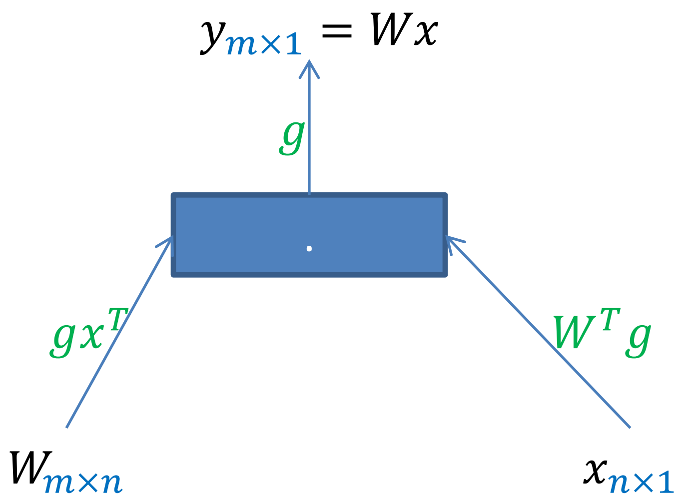
Additional complexity: Dealing with vectors (optional)



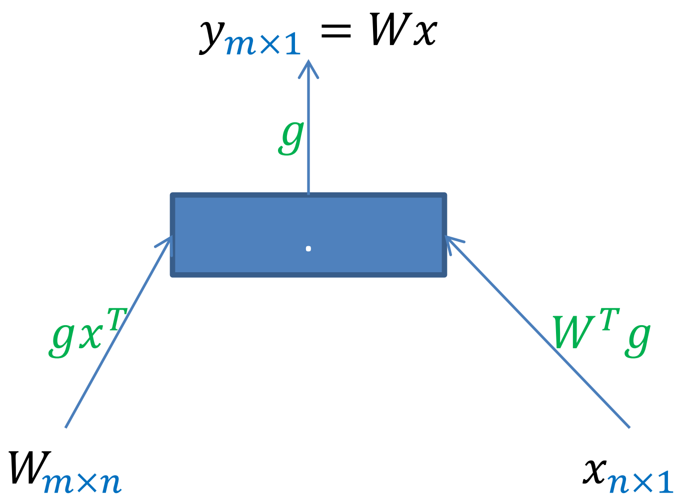
Additional complexity: Dealing with vectors (optional)

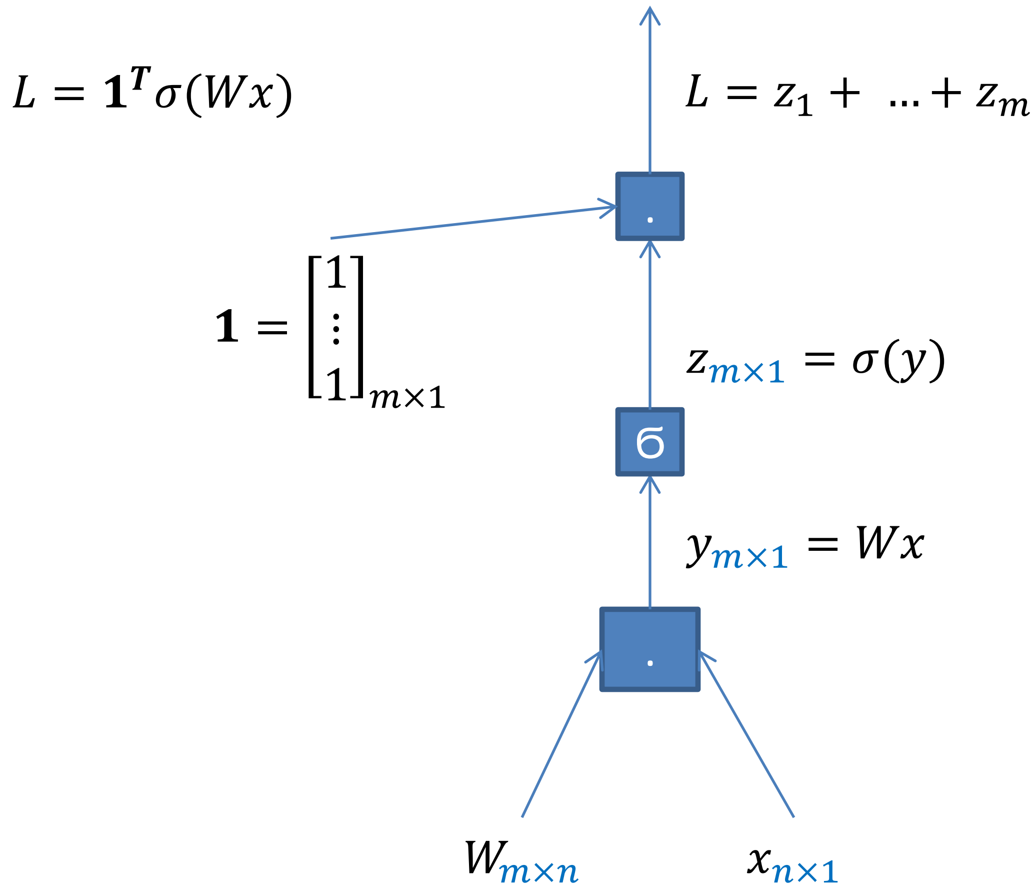


Additional complexity: Dealing with vectors (optional)

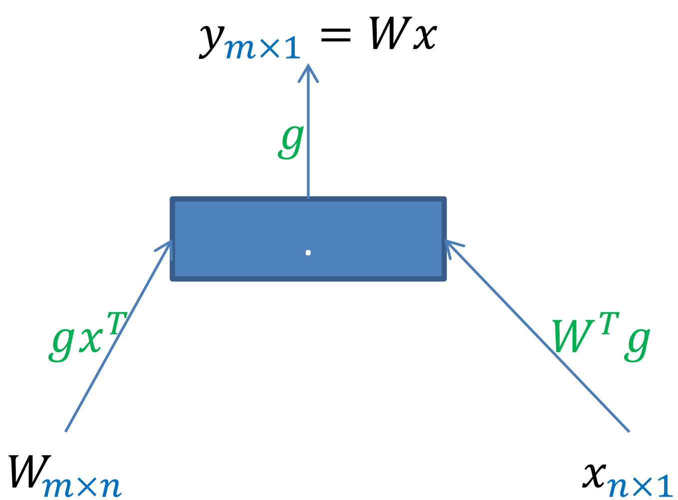


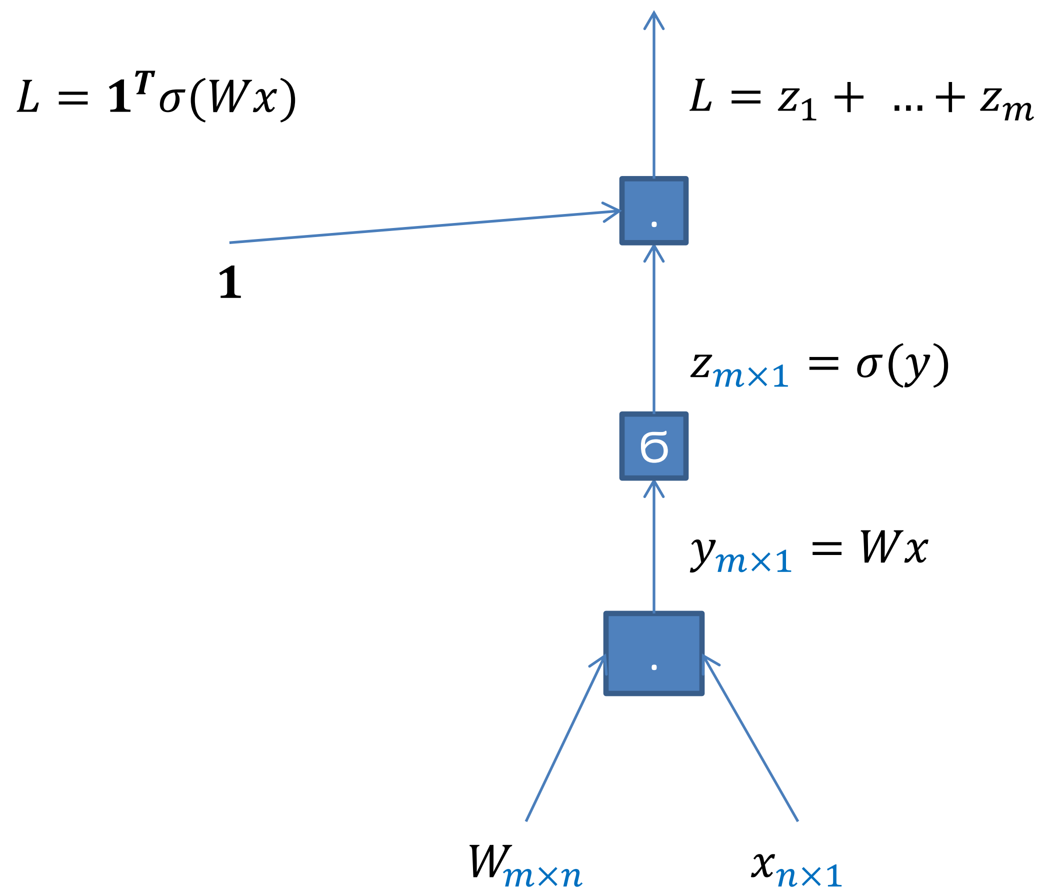
Additional complexity: Dealing with vectors (optional)



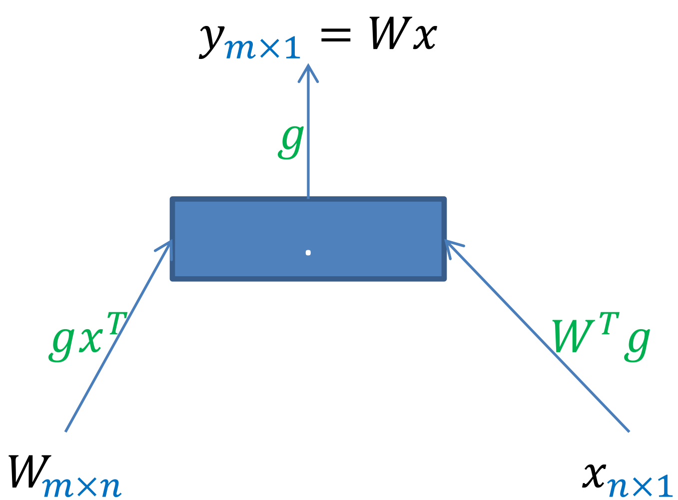


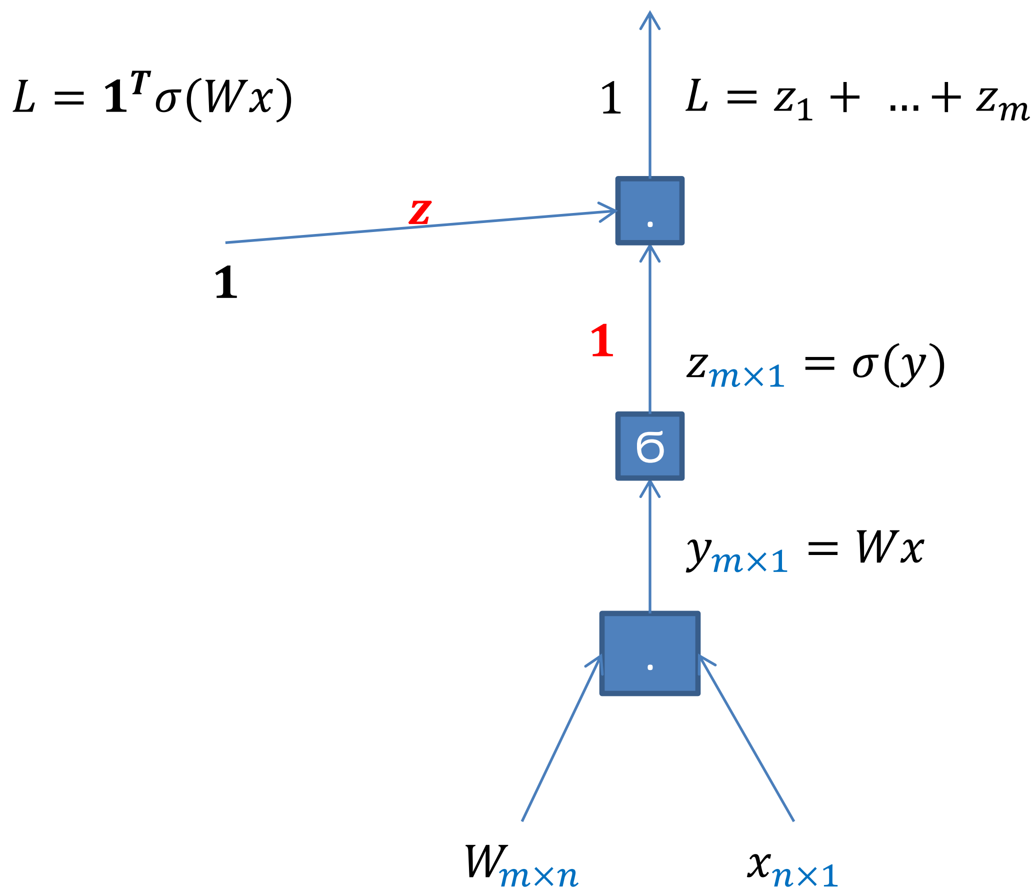
Additional complexity: Dealing with vectors (optional)



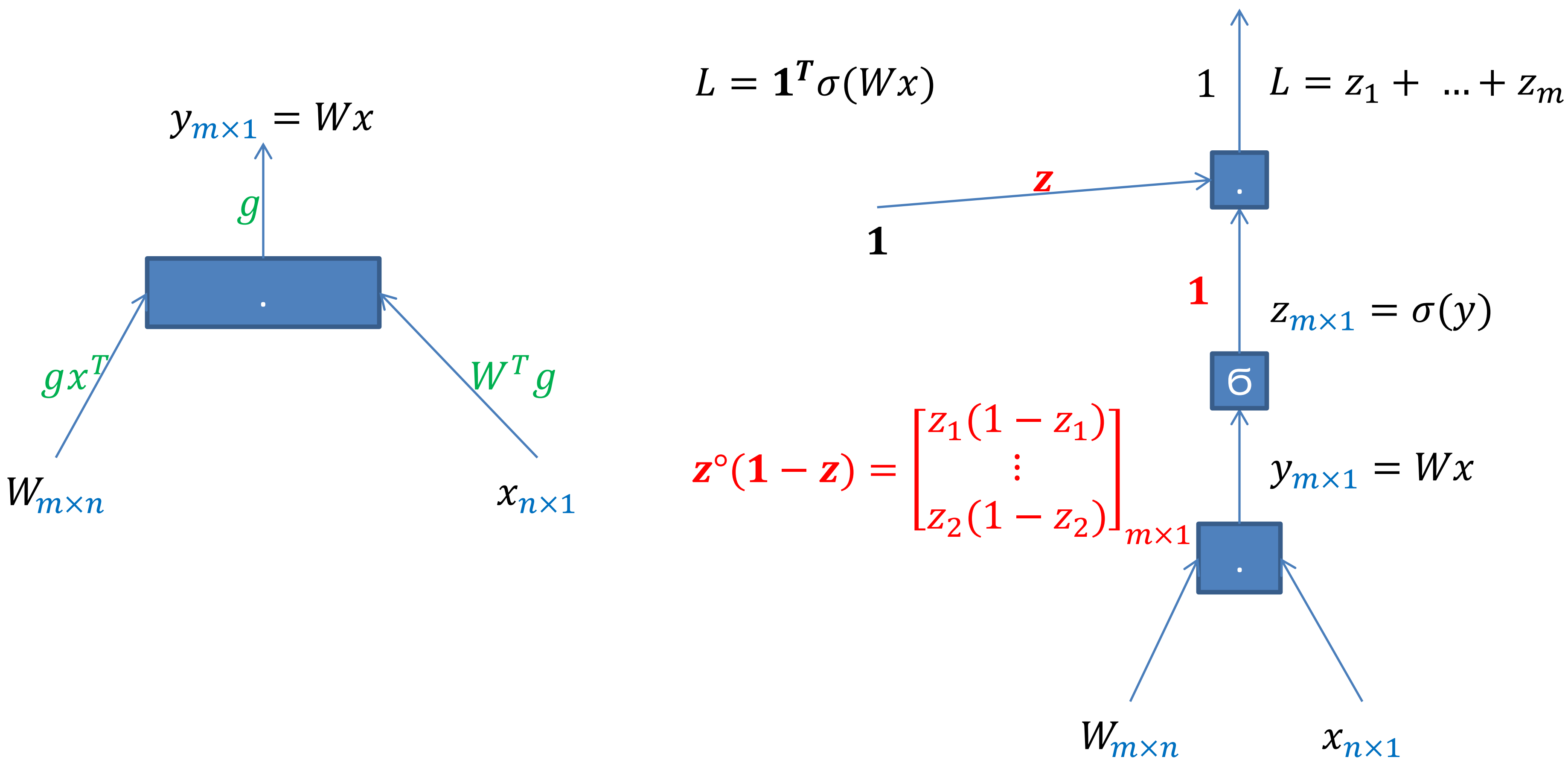


Additional complexity:
Dealing with vectors (optional)

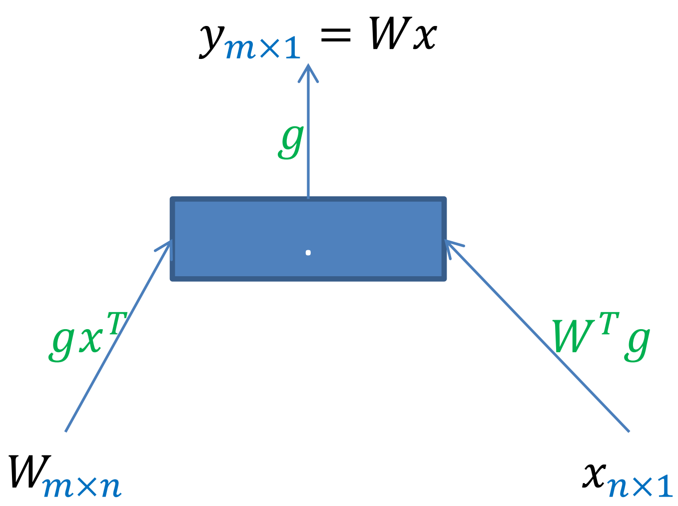


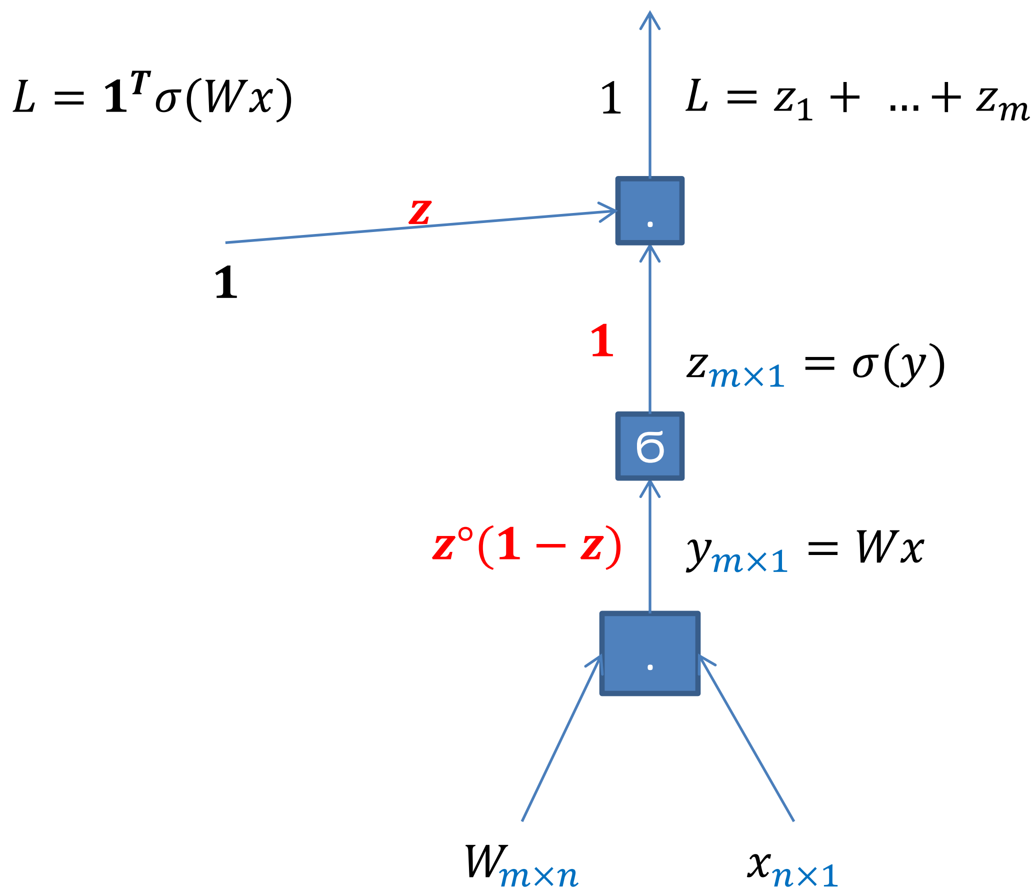


Additional complexity: Dealing with vectors (optional)

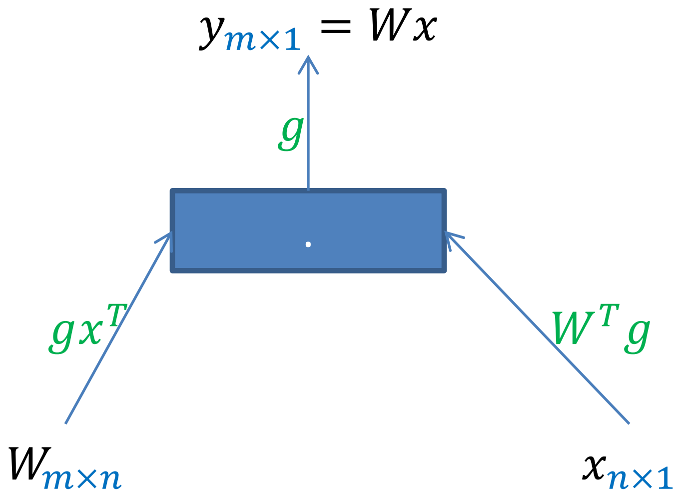


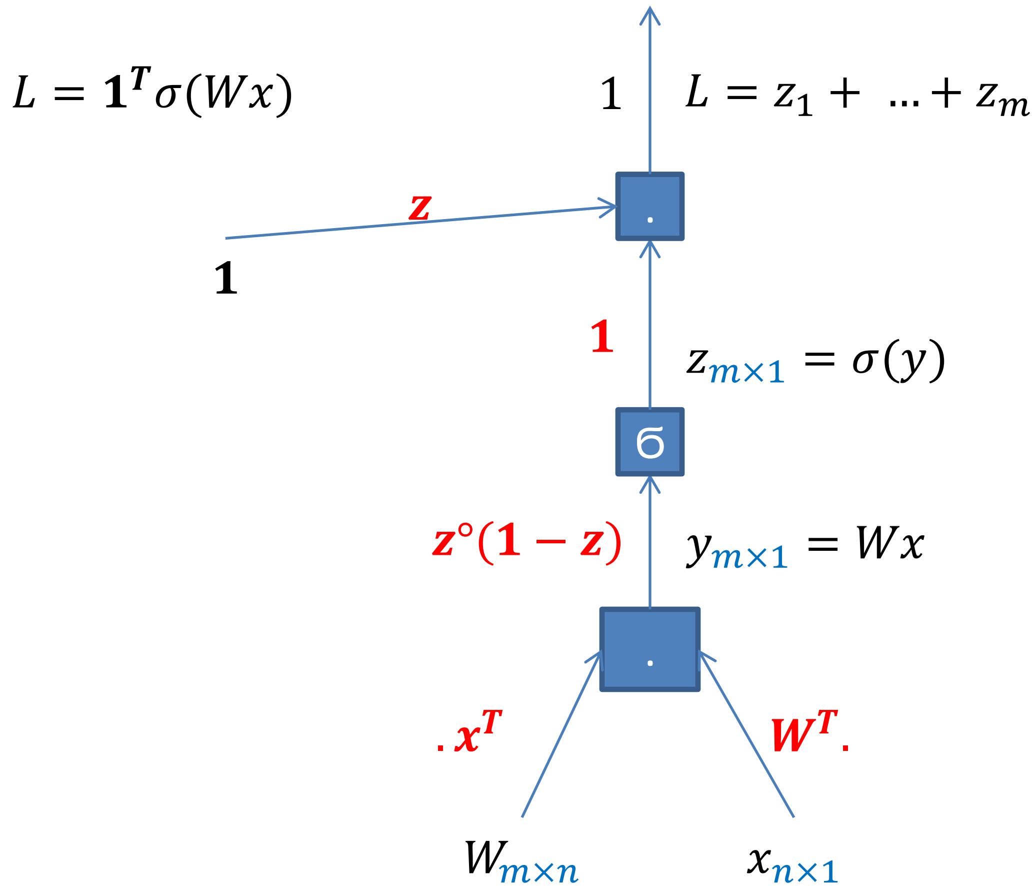
Additional complexity: Dealing with vectors (optional)



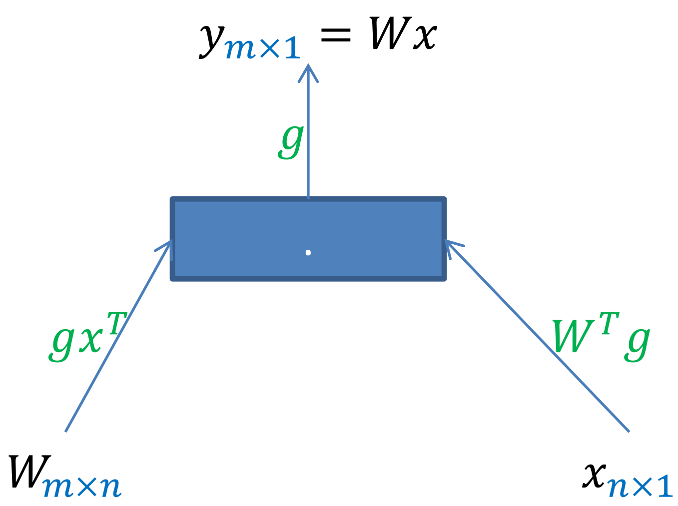


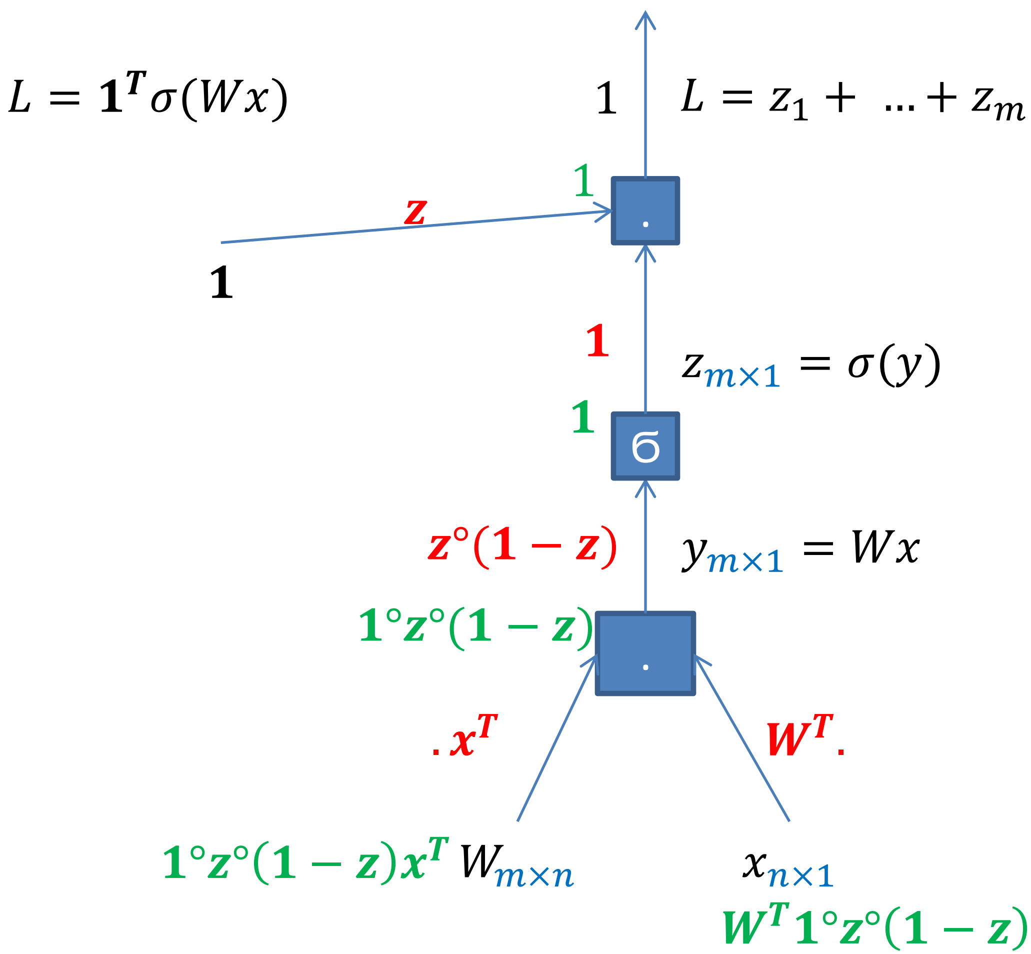
Additional complexity: Dealing with vectors (optional)



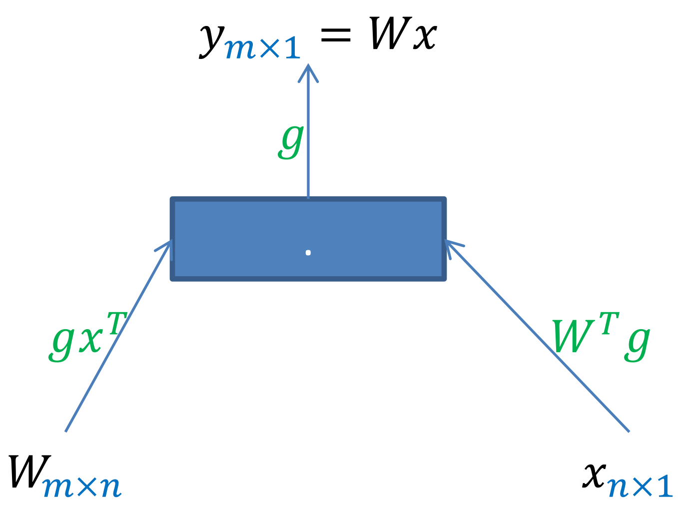


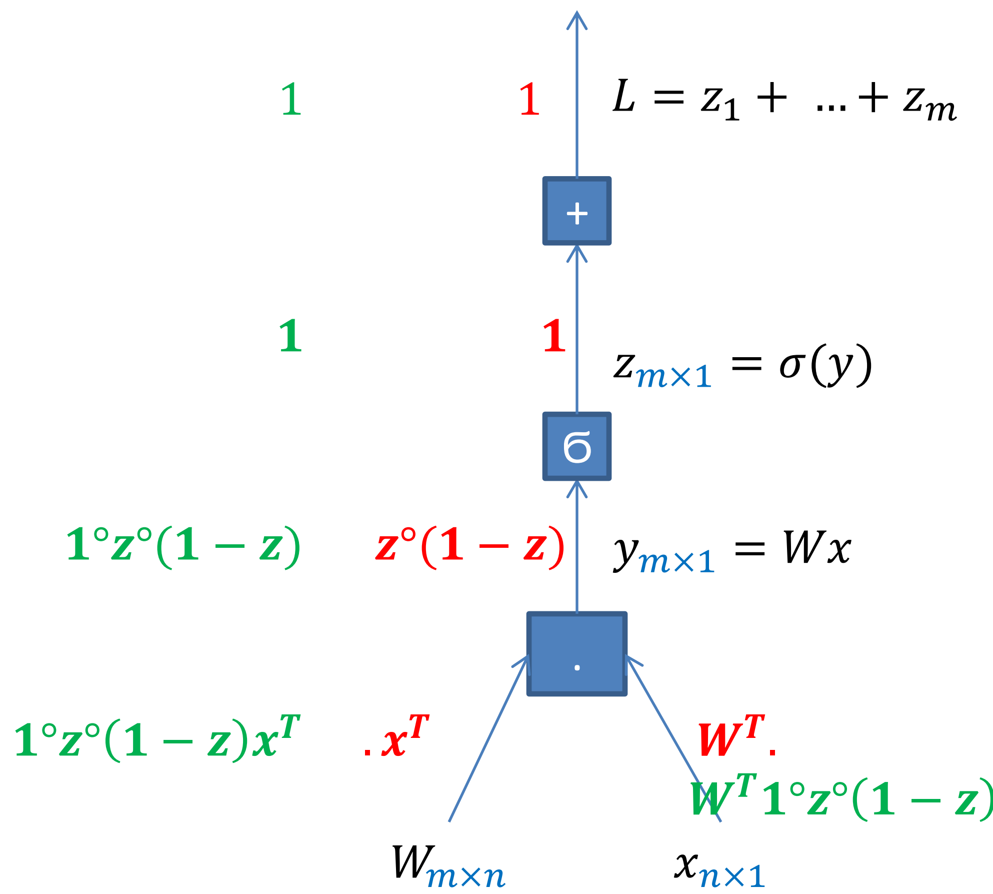
Additional complexity: Dealing with vectors (optional)





Additional complexity: Dealing with vectors (optional)



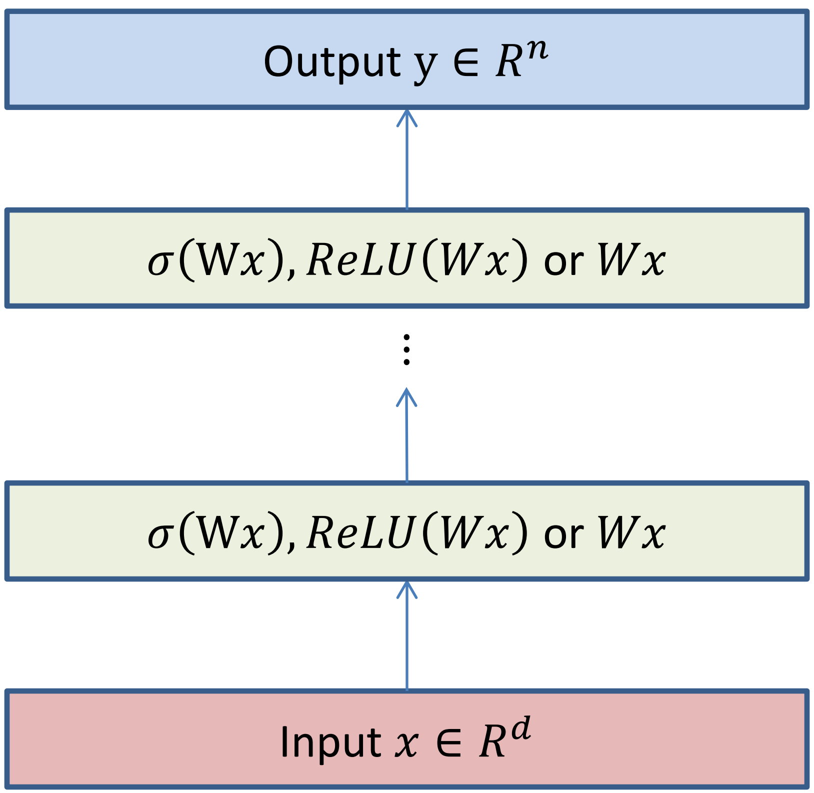


Topics

- Review of backprop
  - Basic operations
  - Class example
- Additional complexity
  - Shared parameters
  - Dealing with vectors (optional)
- Recurrent Neural Net (RNN)
  - Motivation
  - Simple backprop (vector one optional)
  - Demo

Recurrent Neural Net (RNN): Motivation

- Conventional Neural Networks



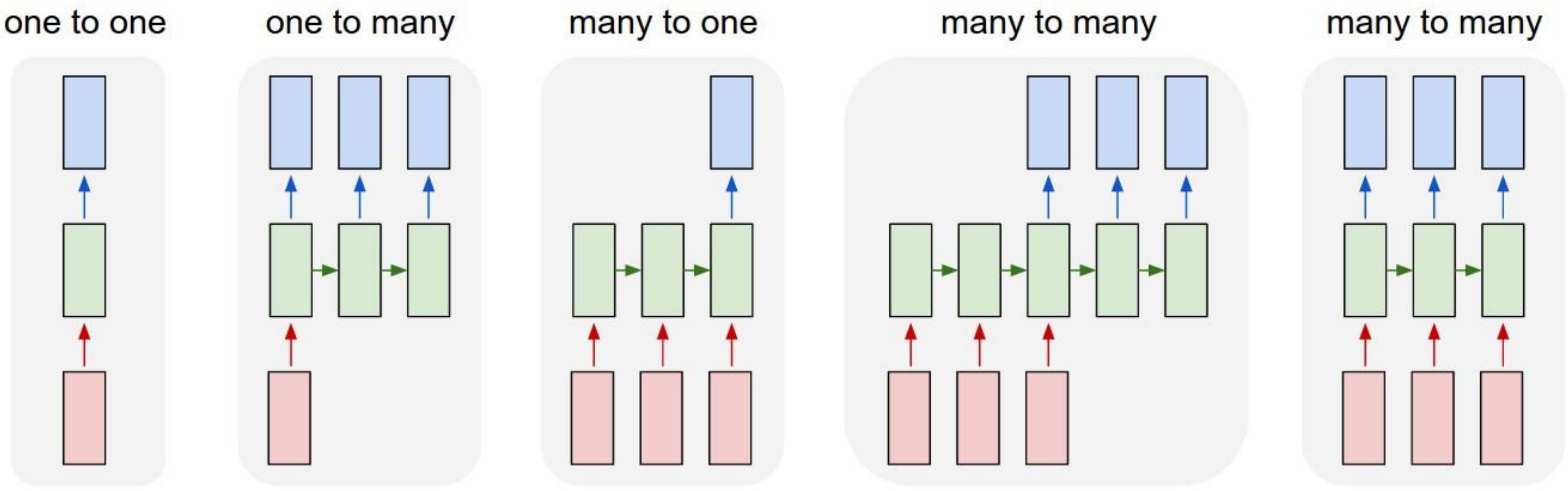
Fixed input and output size
- 1 input (*d* dimensional)
- 1 output (*n* dimensional)

Fixed computing steps
- Independent of input
- Static framework

Typical applications
- Image classification
- Regression

Recurrent Neural Net (RNN): Motivation

- We desire variable input/output size, variable computational steps…

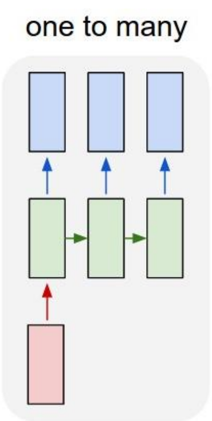


Applications: Image captioning | Sentiment analysis | Machine translation | Text generation

Video classification

Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Neural Net (RNN): Motivation



- Image captioning (one-to-many)

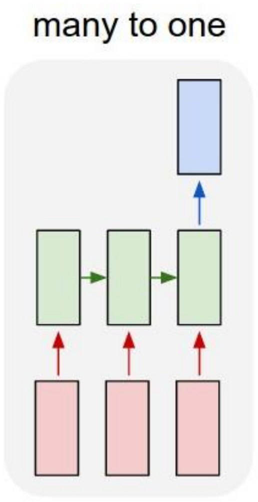
"man in black shirt is playing guitar."

"man in blue wetsuit is surfing on wave."

"a young boy is holding a baseball bat."

Source: http://cs.stanford.edu/people/karpathy/deepimagesent/

Recurrent Neural Net (RNN): Motivation



- Sentiment Analysis (many-to-one)

The action switches between past and present , but the material link is too tenuous to anchor the emotional connections that purport to span a 125-year divide .

Drops you into a dizzying , volatile , pressure-cooker of a situation that quickly snowballs out of control , while focusing on the what much more than the why .

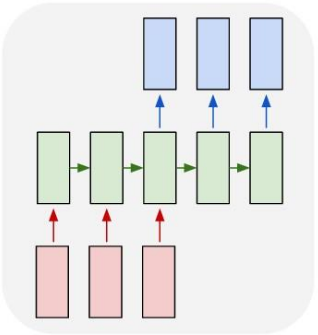
The film is itself a sort of cinematic high crime , one that brings military courtroom dramas down very , very low .

Classify as:
0 - negative
1 - somewhat negative
2 - neutral
3 - somewhat positive
4 - positive

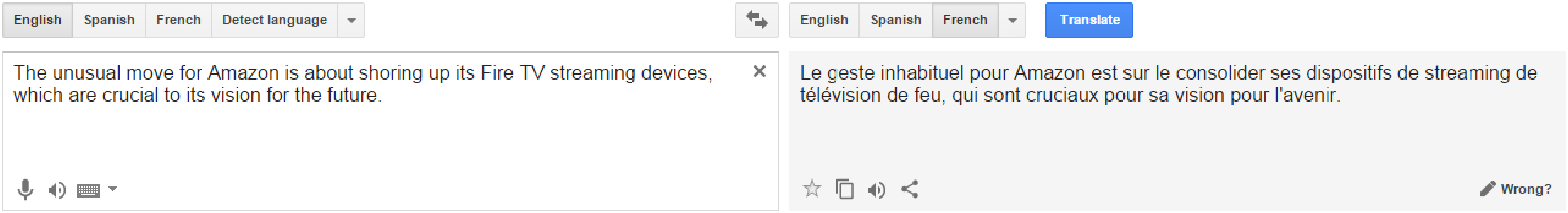
Source: https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews

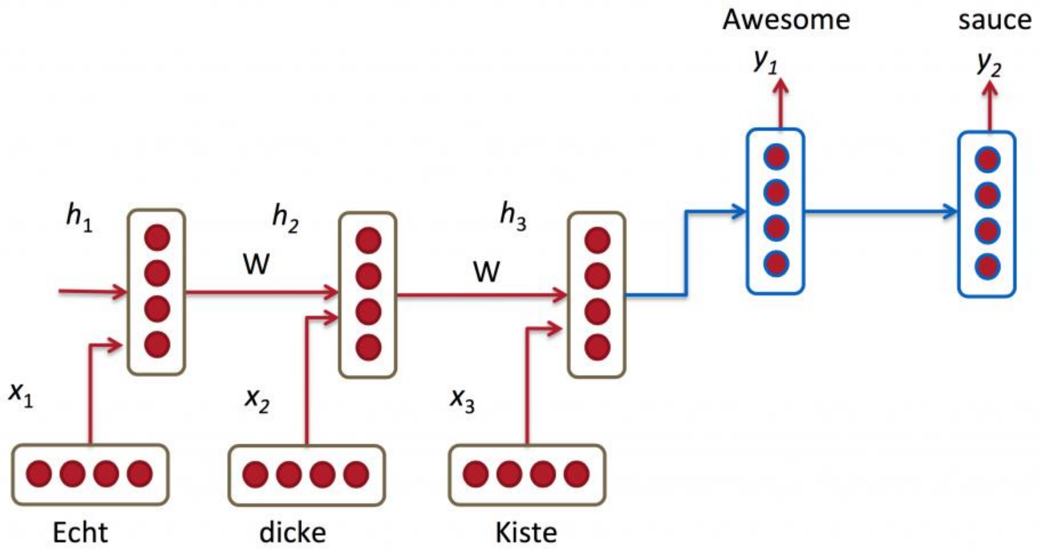
Recurrent Neural Net (RNN): Motivation

many to many



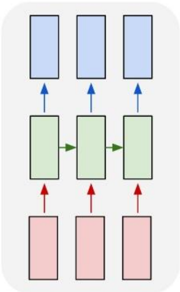
- Machine translation (many-to-many)



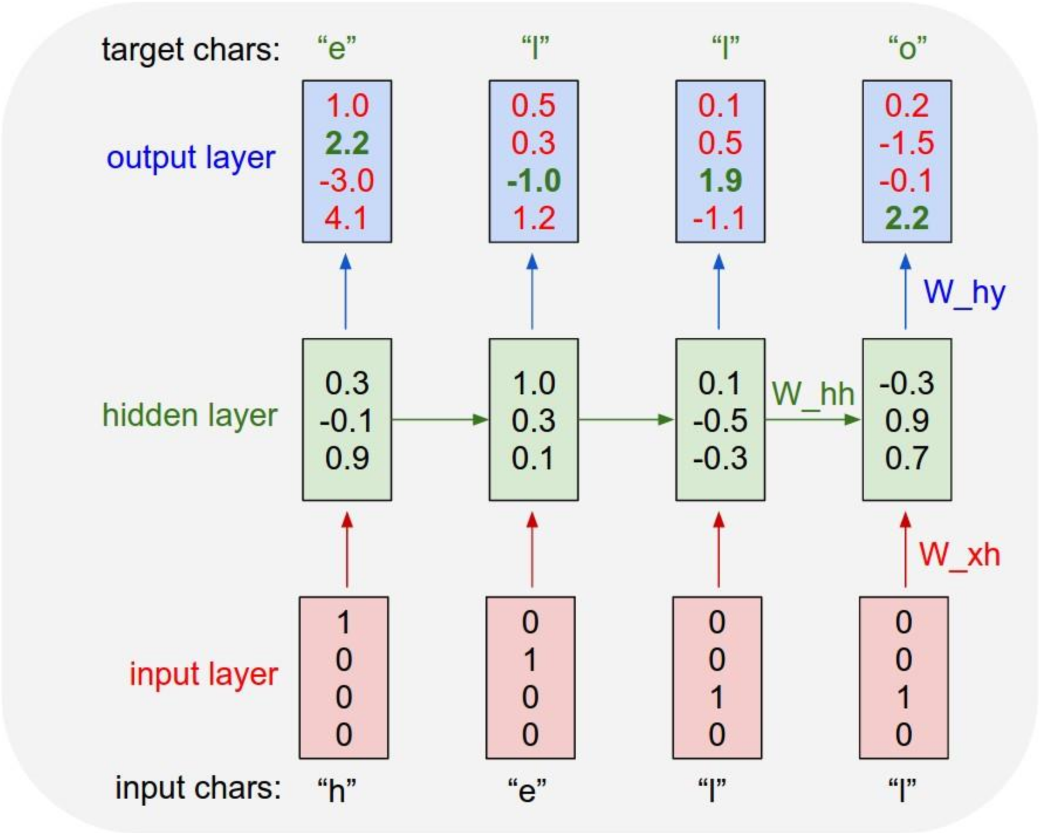


Source: https://translate.google.com (above), http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf (below)

Recurrent Neural Net (RNN): Motivation



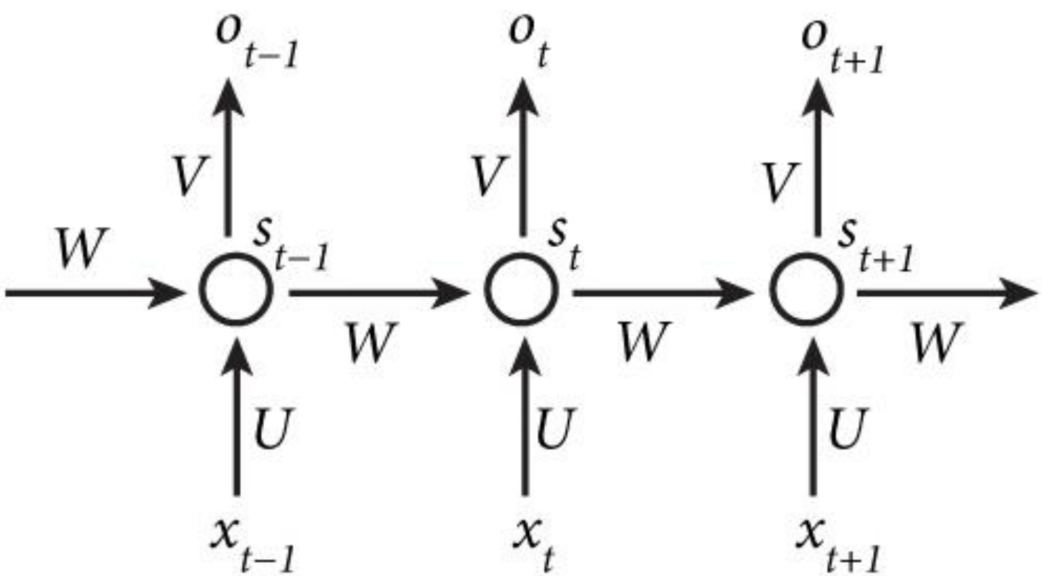
- Character-level language model (many-to-many)

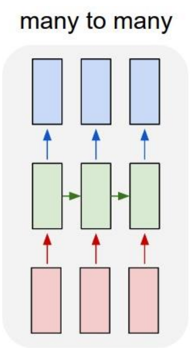


Predict every next character. For example, consider "hello"; use "hell" to predict "ello"

Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Neural Net (RNN): Example





Source: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

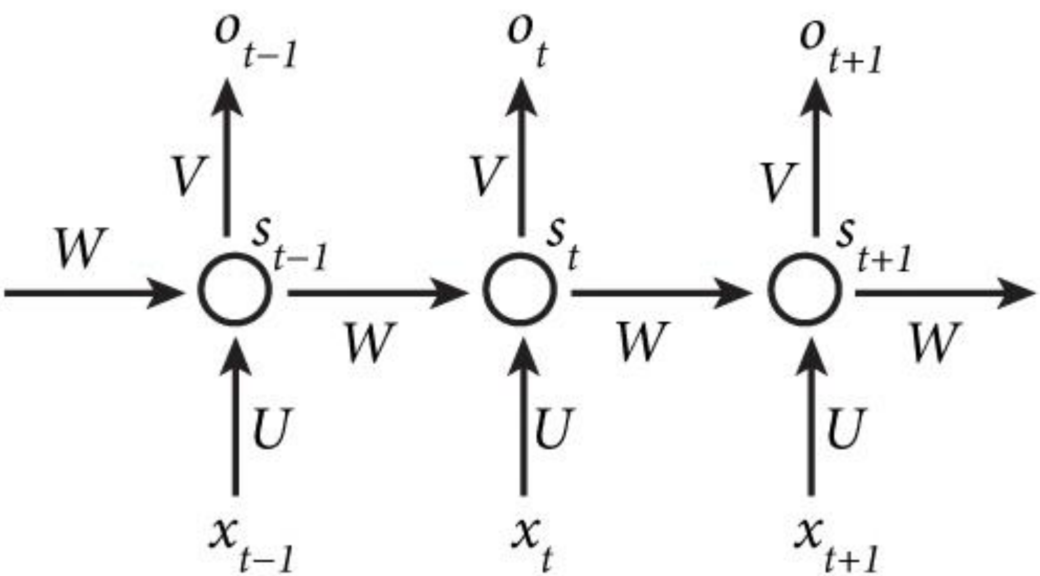
Recurrent Neural Net (RNN): Example

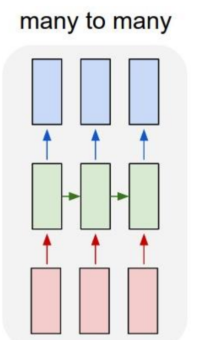
$$s_t = \sigma(Ux_t + Ws_{t-1})$$
$$o_t = \sigma(Vs_t)$$

x_t: input
s_t: hidden state   ($s_{-1} = 0$)
o_t: output
U, V, W: parameters (matrices)

Hidden state stores past information that may be relevant in future.

- Provides context
- Long range dependence





Source: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Recurrent Neural Net (RNN): Example

Character-level language model (many-to-many)

[DEMO]
Script: https://gist.github.com/karpathy/d4dee566867f8291f086
Config: hidden state has 100 dimensions, 93 different characters
Input: norvig.com/big.txt

iter 0, loss: 113.314988
I fechowta ecyoumepuave omas mmur a band chou os Carbinn yond here wa,k, oly soongy pas yin fou alinfo#gtid ed levenupksen Ia tbinl and. Yury sleve lsok ufimeme conlanf youlsseg ve-;aud Mas finn ass w

iter 185000, loss: 49.667141
hing the hri, theme ummengi-hy linced. The candiccevinicas he visur. The her in war to Ereart in dnintorvaned wenced to as rewnighly restera he by appored bat riculing at hooke thiming a somews, and

Observe: Starts looking like English; new sentence starts with big letter; end with full stop; short words spelled correctly; long words still messed up

Source: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Recurrent Neural Net (RNN): Example

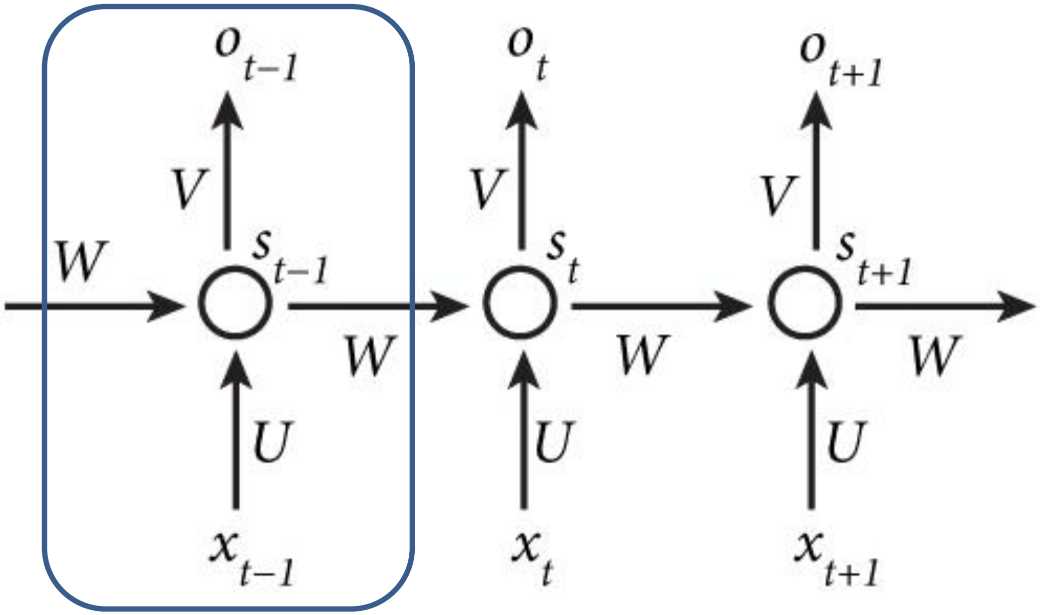
Character-level language model (many-to-many)

Paul Graham's essays
Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

"The surprised in investors weren't going to raise money. I'm not the company with the time there are all interesting quickly, don't have to get off the same programmers. There's a super-angel round fundraising, why do you can do. If you have a different physical investment are become in people who reduced in a startup with the way to argument the acquirer could see them just that you're also the founders will part of users' affords that and an alternation to the idea. ***[2]*** *Don't work at first member to see the way kids will seem in advance of a bad successful startup. And if you have to act the big company too."*

Observe: Learns spelling and grammar from scratch; learns to cite; says *"a company is a meeting to think to investors", starts understanding a bit.*

Recurrent Neural Net (RNN): Example



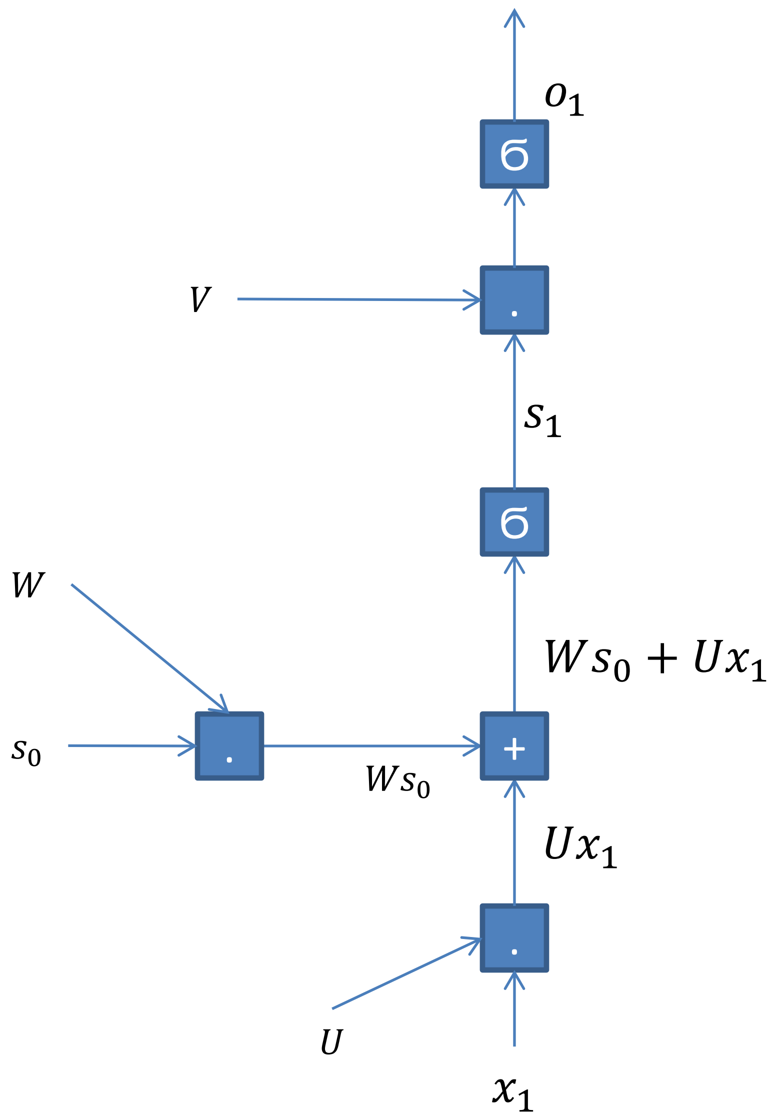
$$s_t = \sigma(Ux_t + Ws_{t-1})$$

$$o_t = \sigma(Vs_t)$$

x_t: input
s_t: hidden state $(s_{-1} = 0)$
o_t: output
U, V, W: parameters (matrices)

Source: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Recurrent Neural Net (RNN): Example



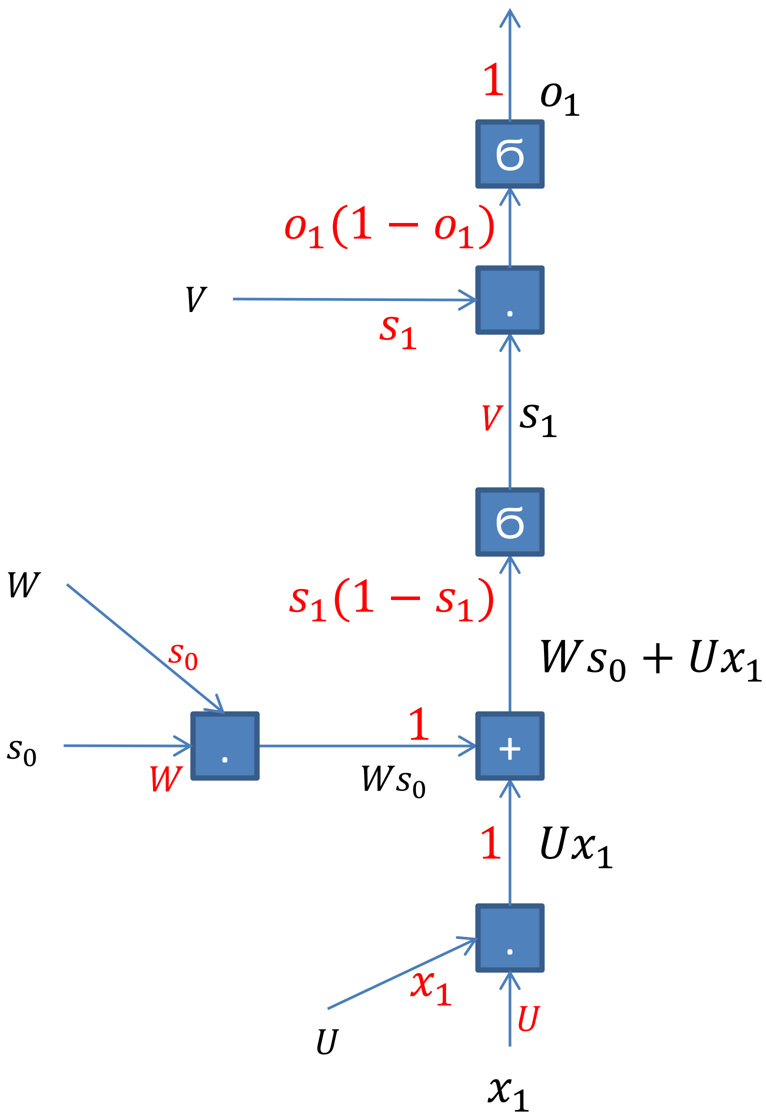
$$s_t = \sigma(Ux_t + Ws_{t-1})$$

$$o_t = \sigma(Vs_t)$$

x_t: input
s_t: hidden state $(s_{-1} = 0)$
o_t: output
U, V, W: parameters

$L = \sum_t o_t$: loss (assume)

Recurrent Neural Net (RNN): Example



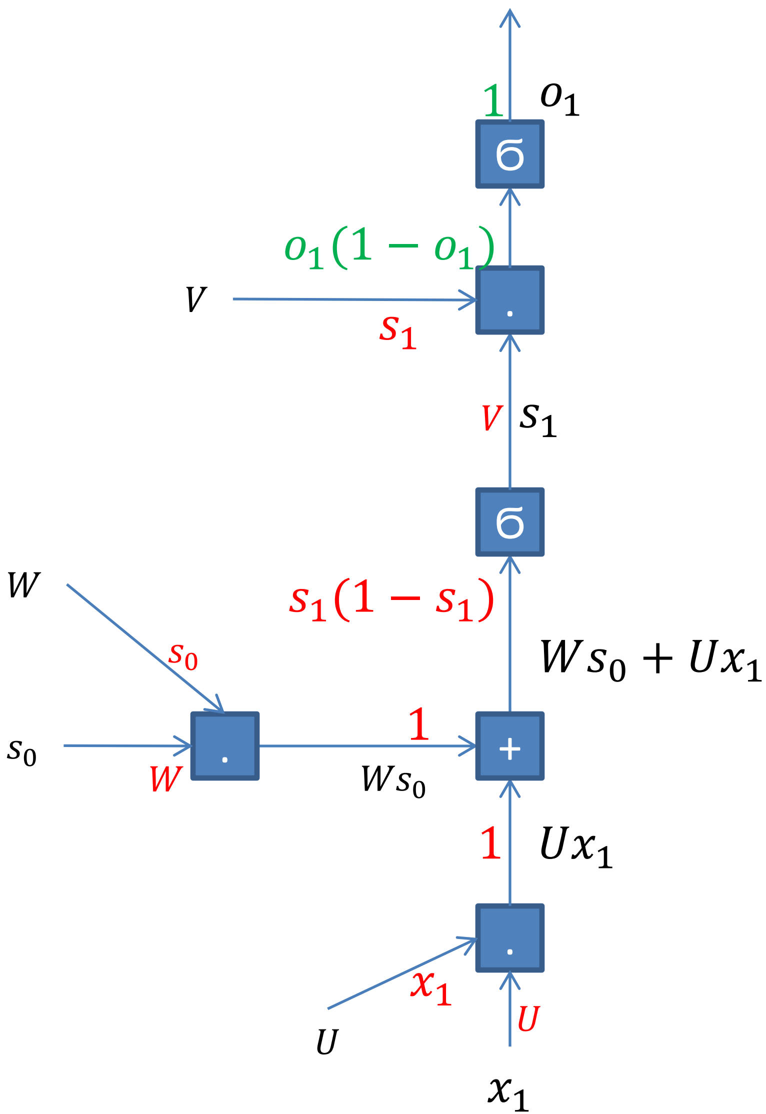
$$s_t = \sigma(Ux_t + Ws_{t-1})$$

$$o_t = \sigma(Vs_t)$$

x_t: input
s_t: hidden state $(s_{-1} = 0)$
o_t: output
U, V, W: parameters

$L = \sum_t o_t$: loss (assume)

Recurrent Neural Net (RNN): Example



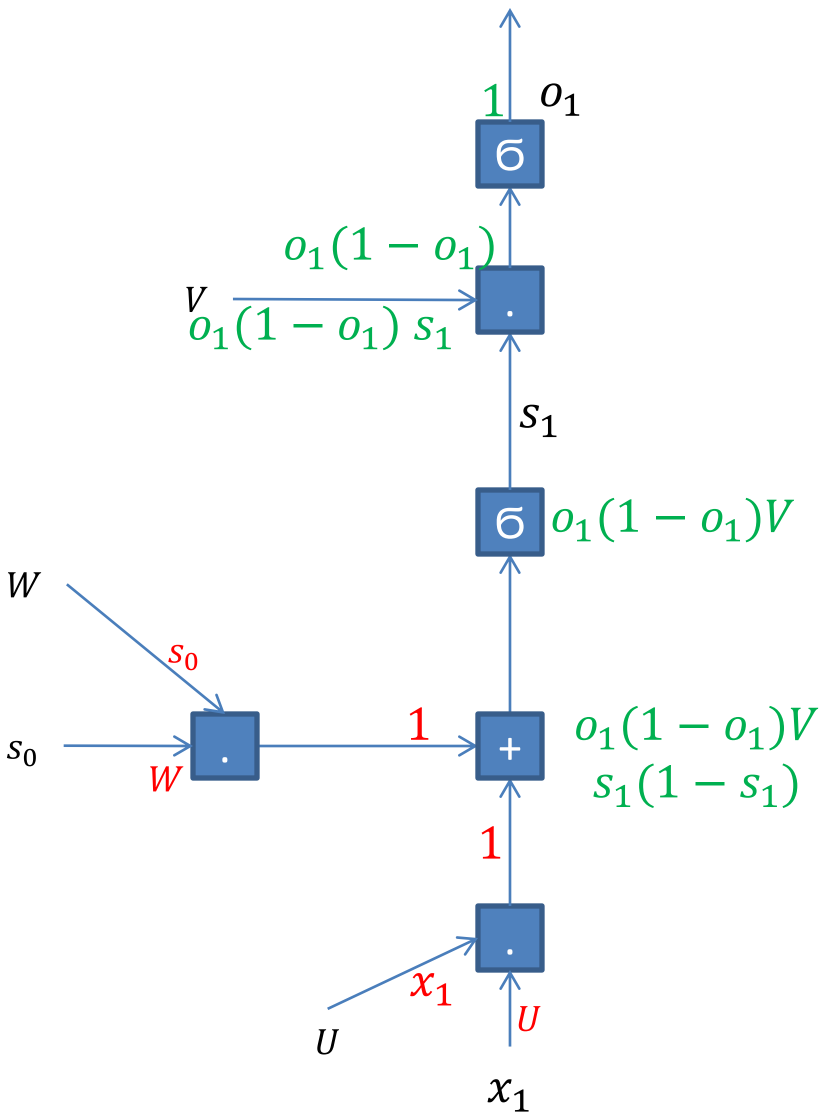
$$s_t = \sigma(Ux_t + Ws_{t-1})$$

$$o_t = \sigma(Vs_t)$$

x_t: input
s_t: hidden state ($s_{-1} = 0$)
o_t: output
U, V, W: parameters

$L = \sum_t o_t$: loss (assume)

Recurrent Neural Net (RNN): Example



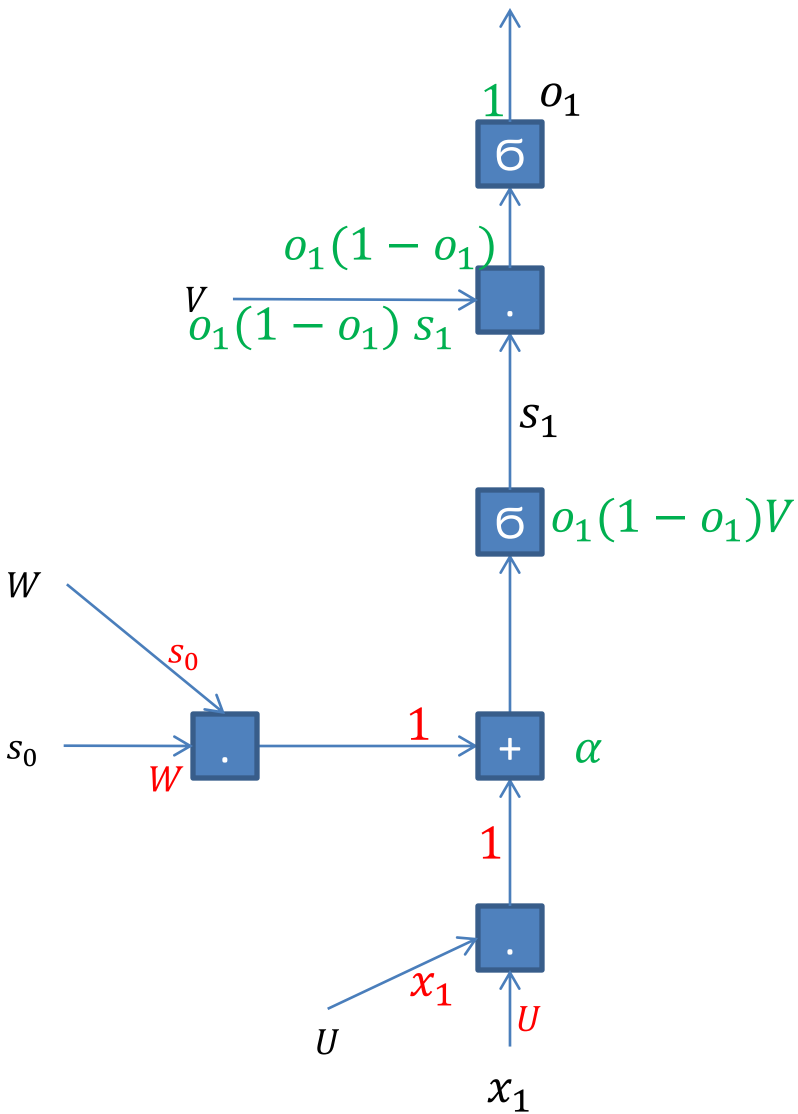
$s_t = \sigma(Ux_t + Ws_{t-1})$

$o_t = \sigma(Vs_t)$

x_t: input
s_t: hidden state $(s_{-1} = 0)$
o_t: output
U, V, W: parameters

$L = \sum_t o_t$: loss (assume)

Recurrent Neural Net (RNN): Example



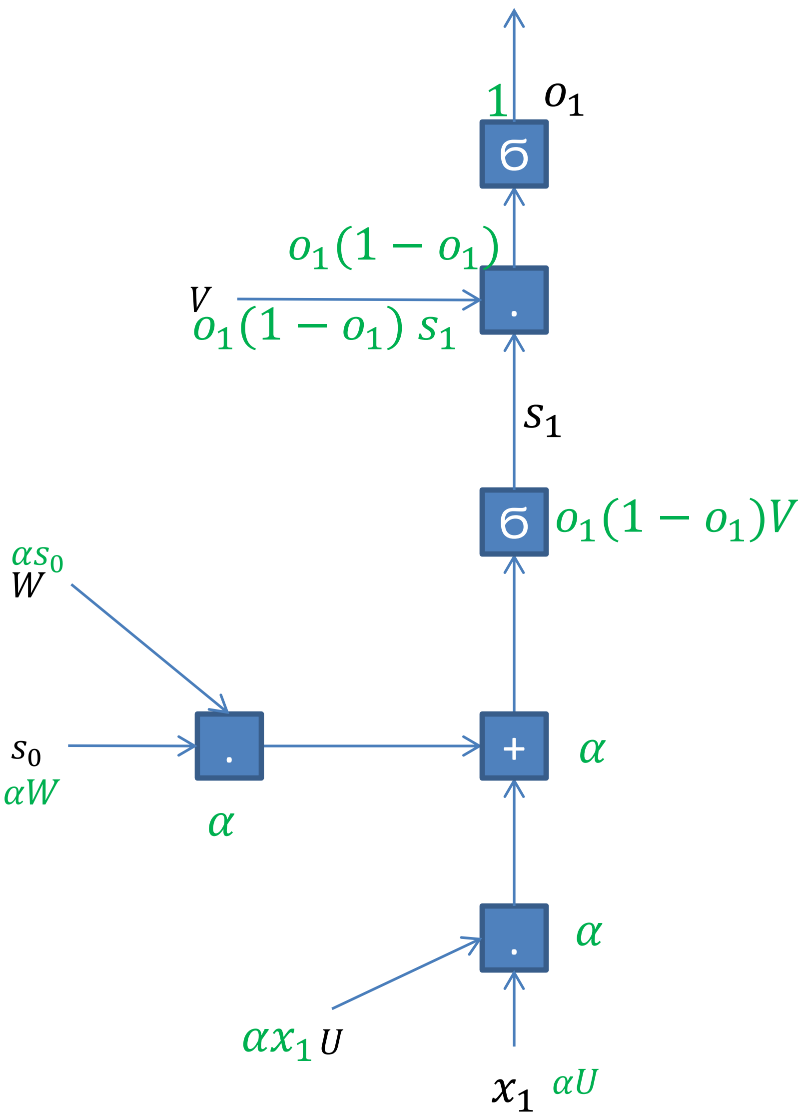
$s_t = \sigma(Ux_t + Ws_{t-1})$

$o_t = \sigma(Vs_t)$

x_t: input
s_t: hidden state $(s_{-1} = 0)$
o_t: output
U, V, W: parameters

$L = \sum_t o_t$: loss (assume)

Recurrent Neural Net (RNN): Example



$s_t = \sigma(Ux_t + Ws_{t-1})$

$o_t = \sigma(Vs_t)$

x_t: input

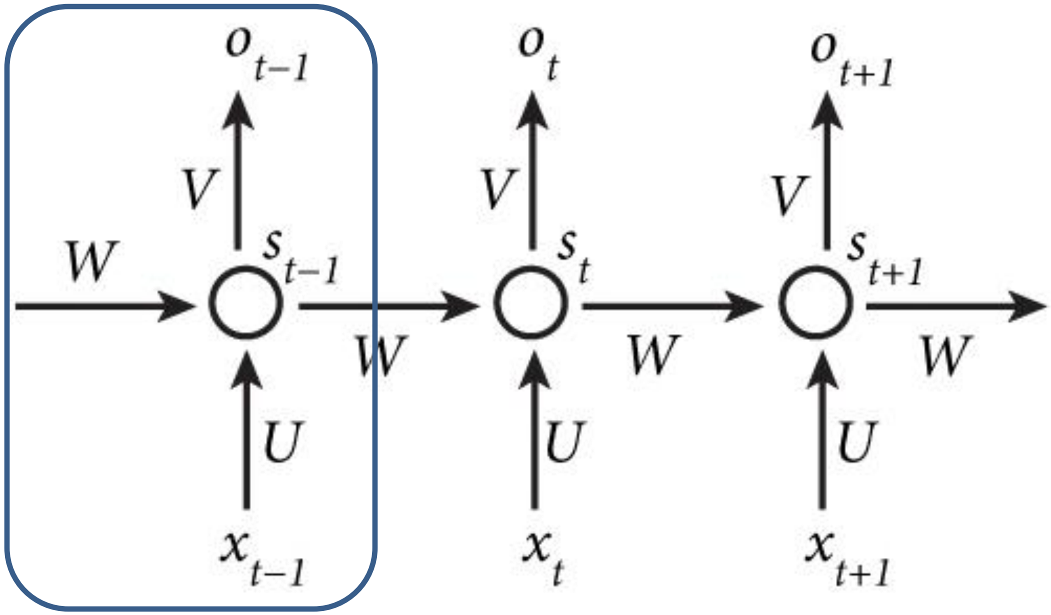
s_t: hidden state $(s_{-1} = 0)$

o_t: output

U, V, W: parameters

$L = \sum_t o_t$: loss (assume)

Recurrent Neural Net (RNN): Example

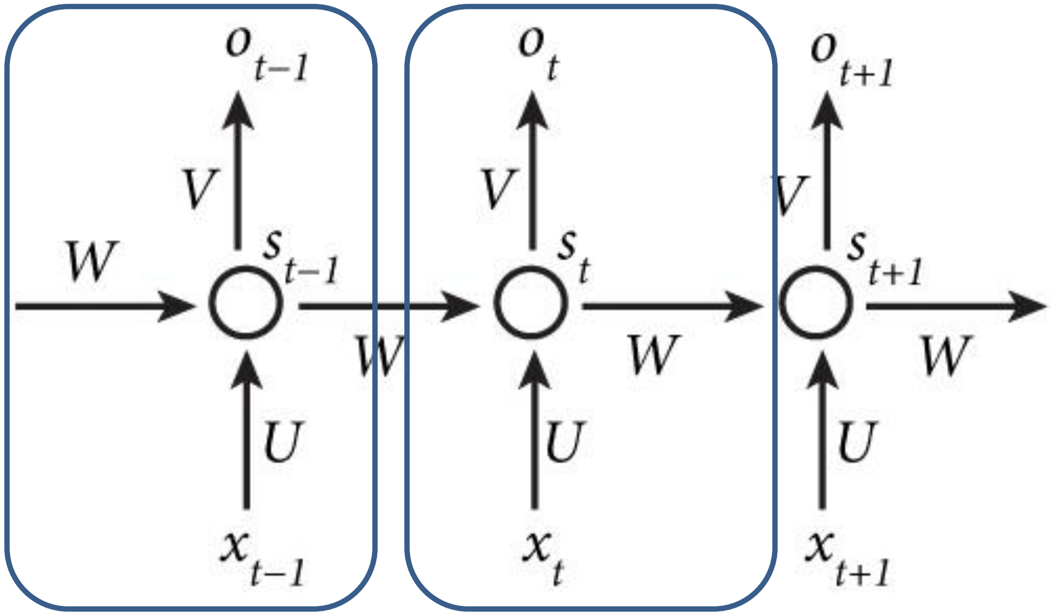


$$s_t = \sigma(Ux_t + Ws_{t-1})$$
$$o_t = \sigma(Vs_t)$$

x_t: input
s_t: hidden state ($s_{-1} = 0$)
o_t: output
U, V, W: parameters (matrices)

Source: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Recurrent Neural Net (RNN): Example



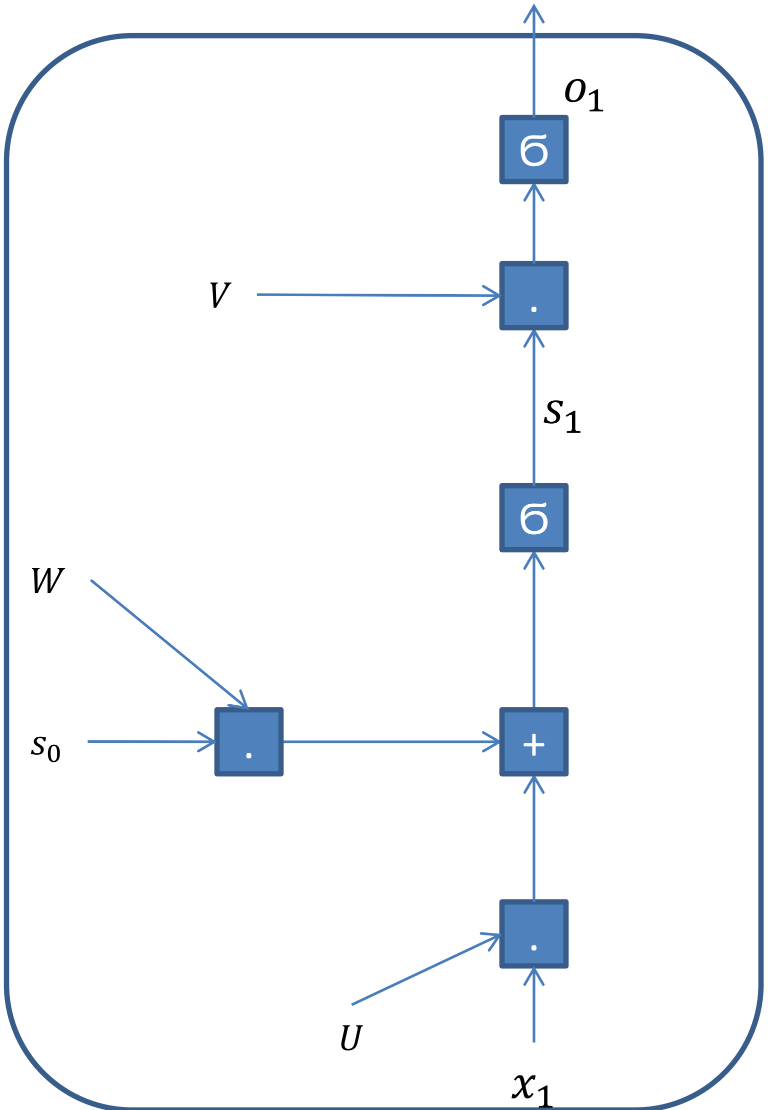
$s_t = \sigma(Ux_t + Ws_{t-1})$

$o_t = \sigma(Vs_t)$

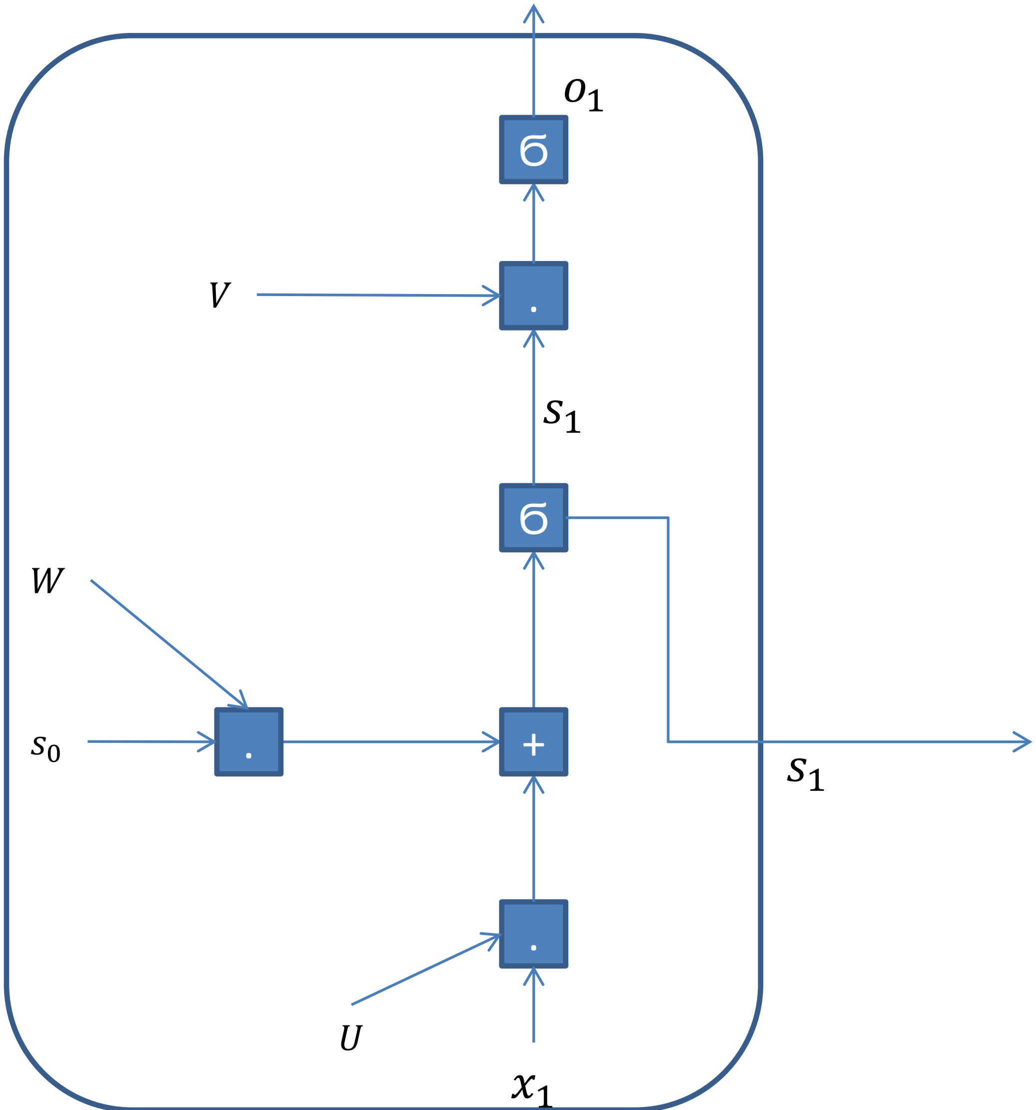
x_t: input
s_t: hidden state $(s_{-1} = 0)$
o_t: output
U, V, W: parameters (matrices)

Source: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

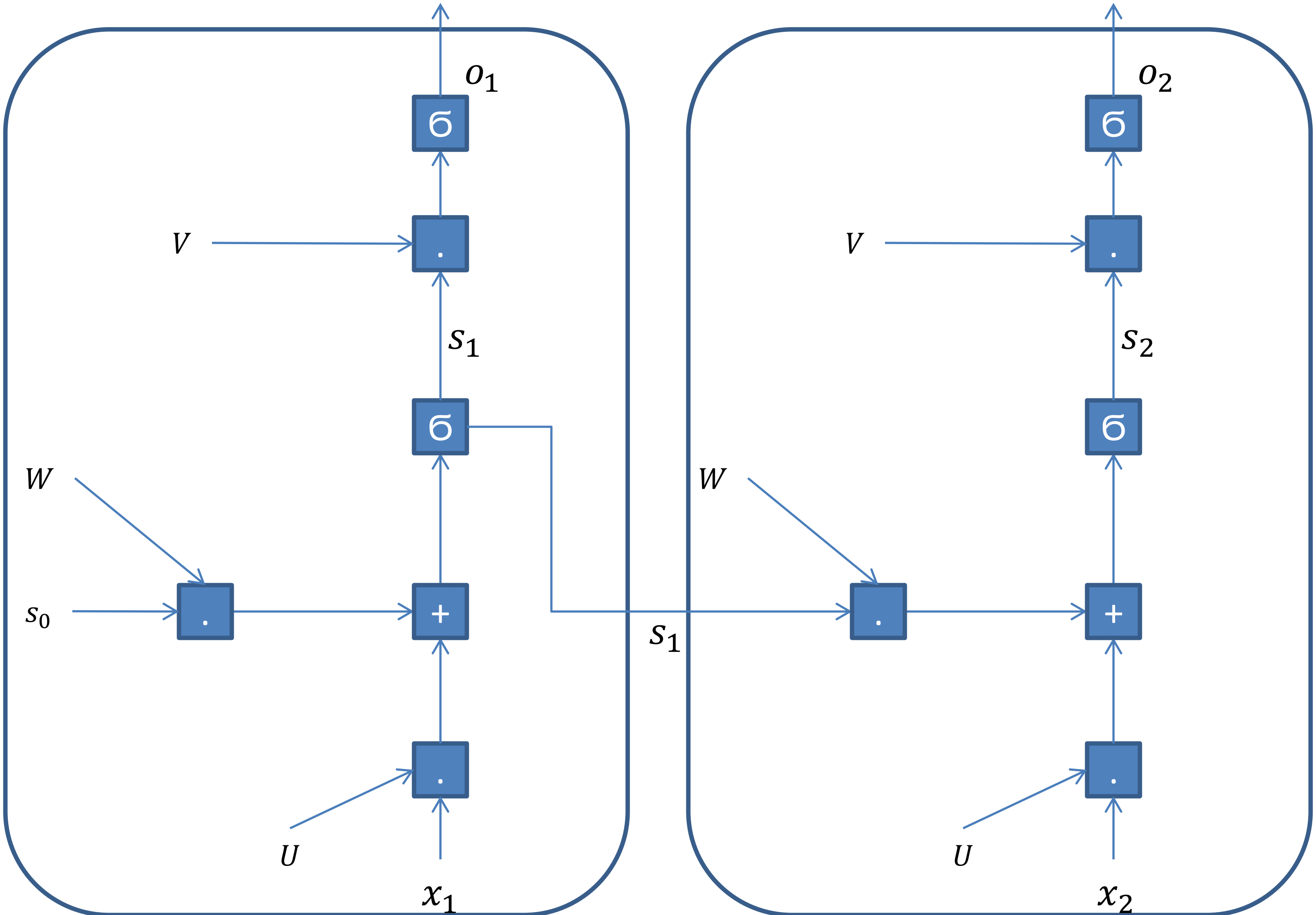
Recurrent Neural Net (RNN): Example



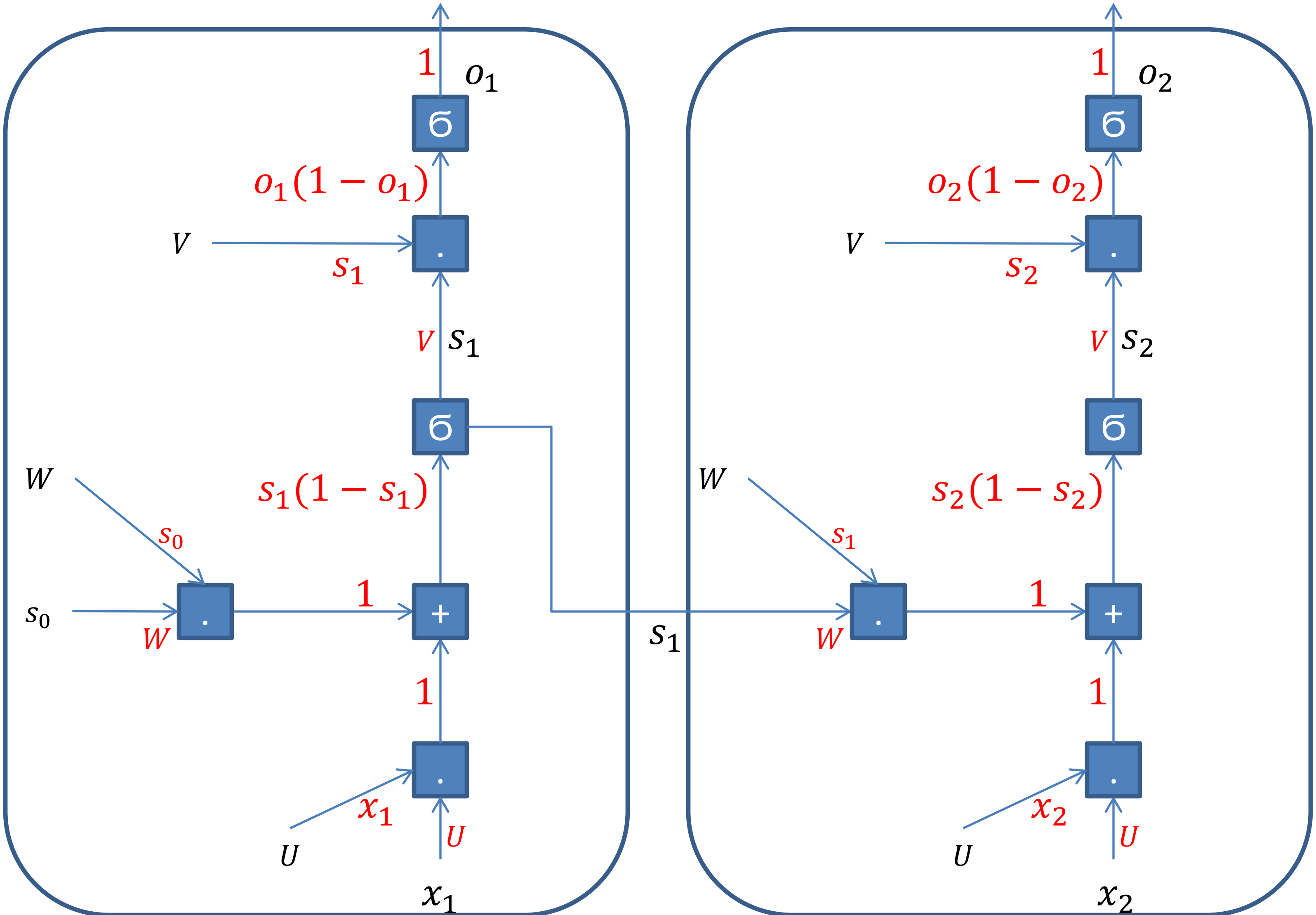
Recurrent Neural Net (RNN): Example



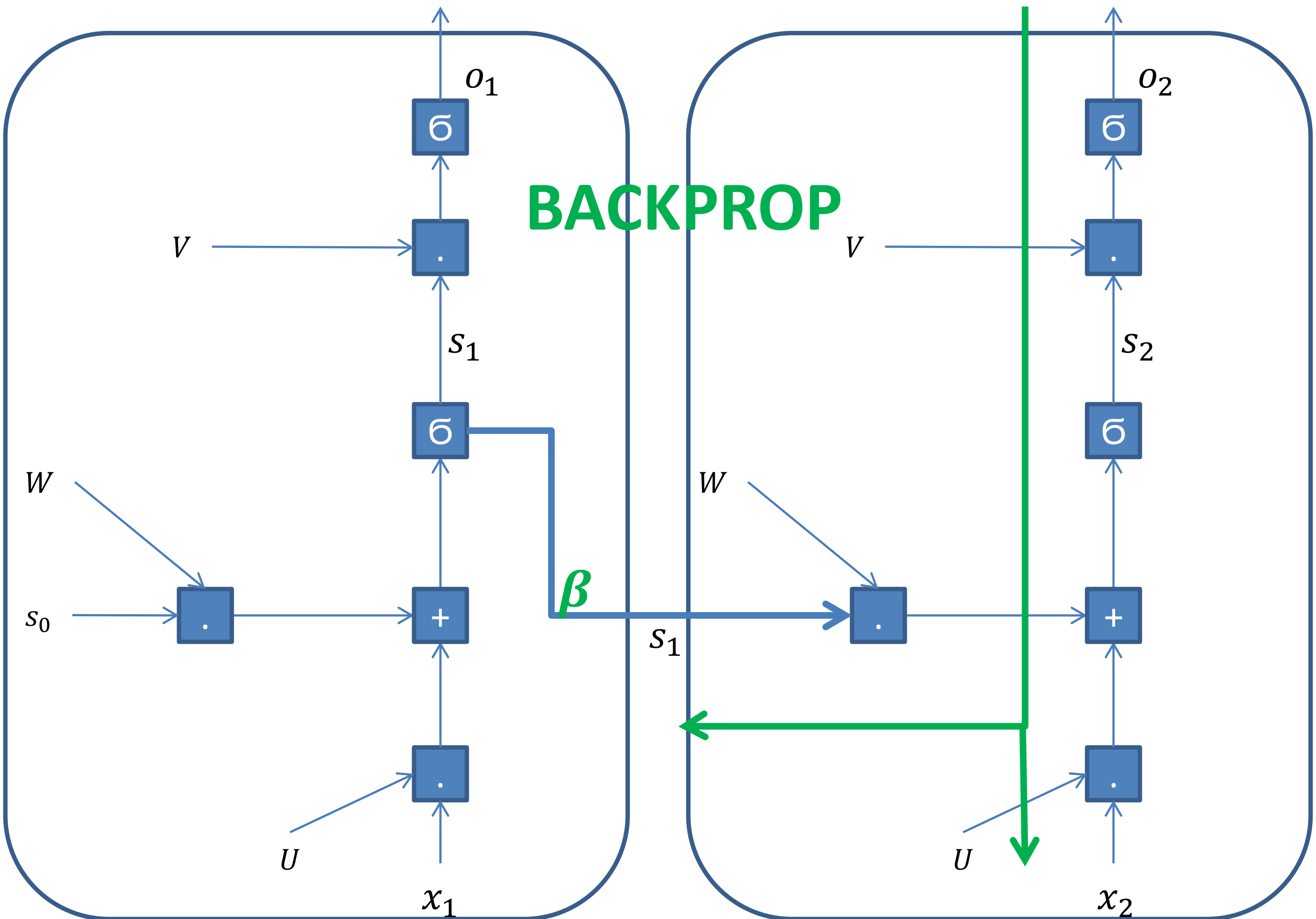
Recurrent Neural Net (RNN): Example



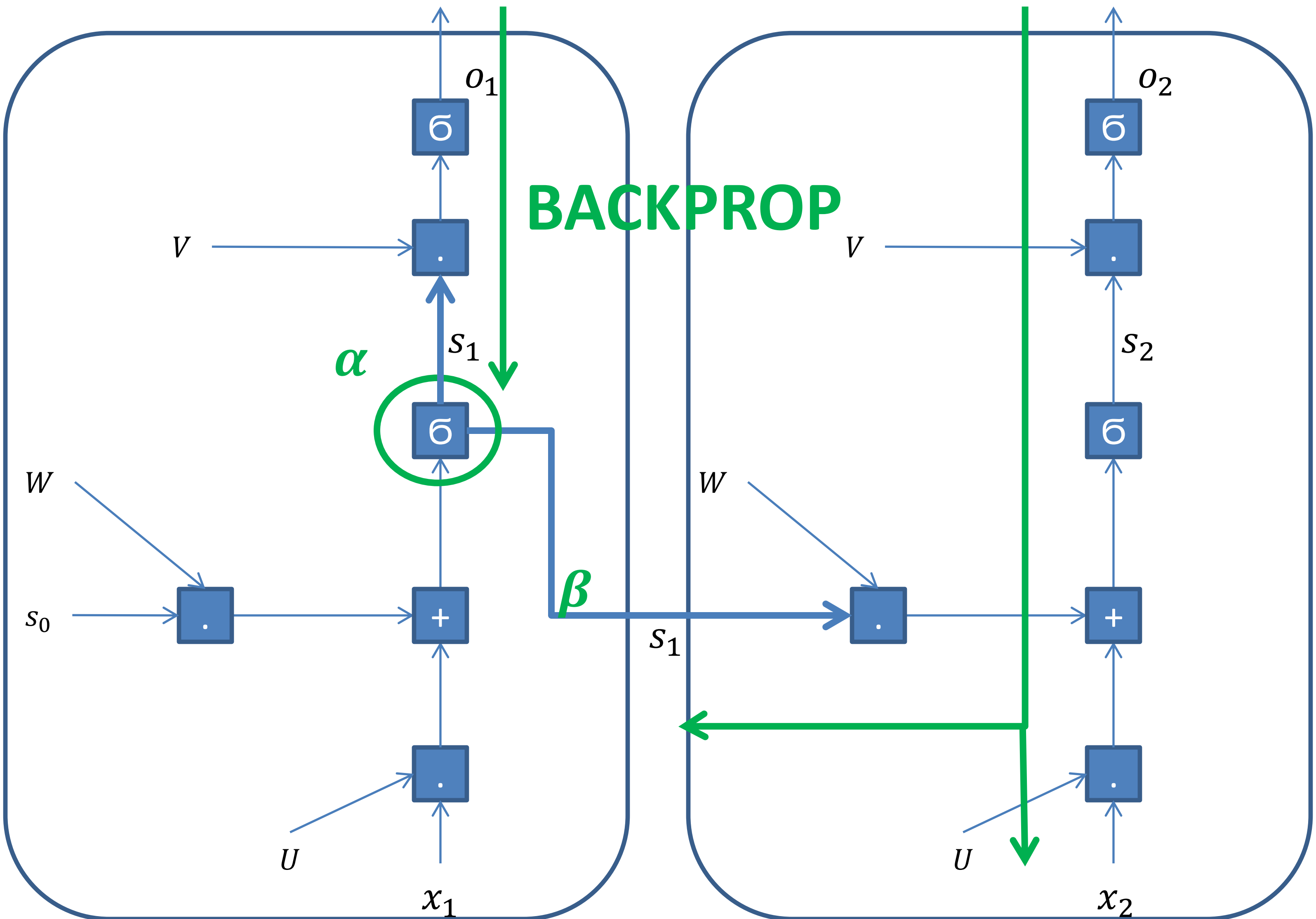
Recurrent Neural Net (RNN): Example



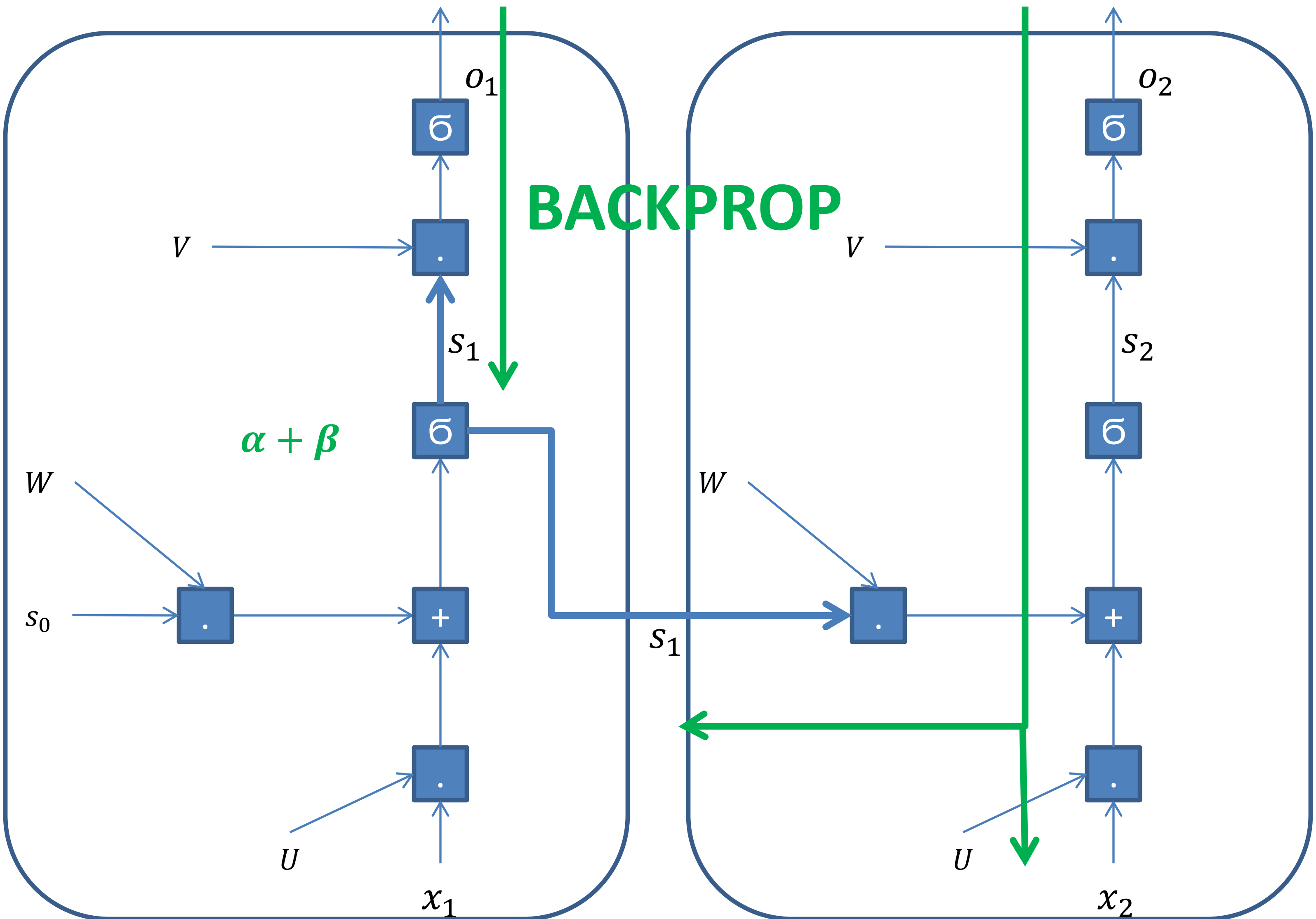
Recurrent Neural Net (RNN): Example



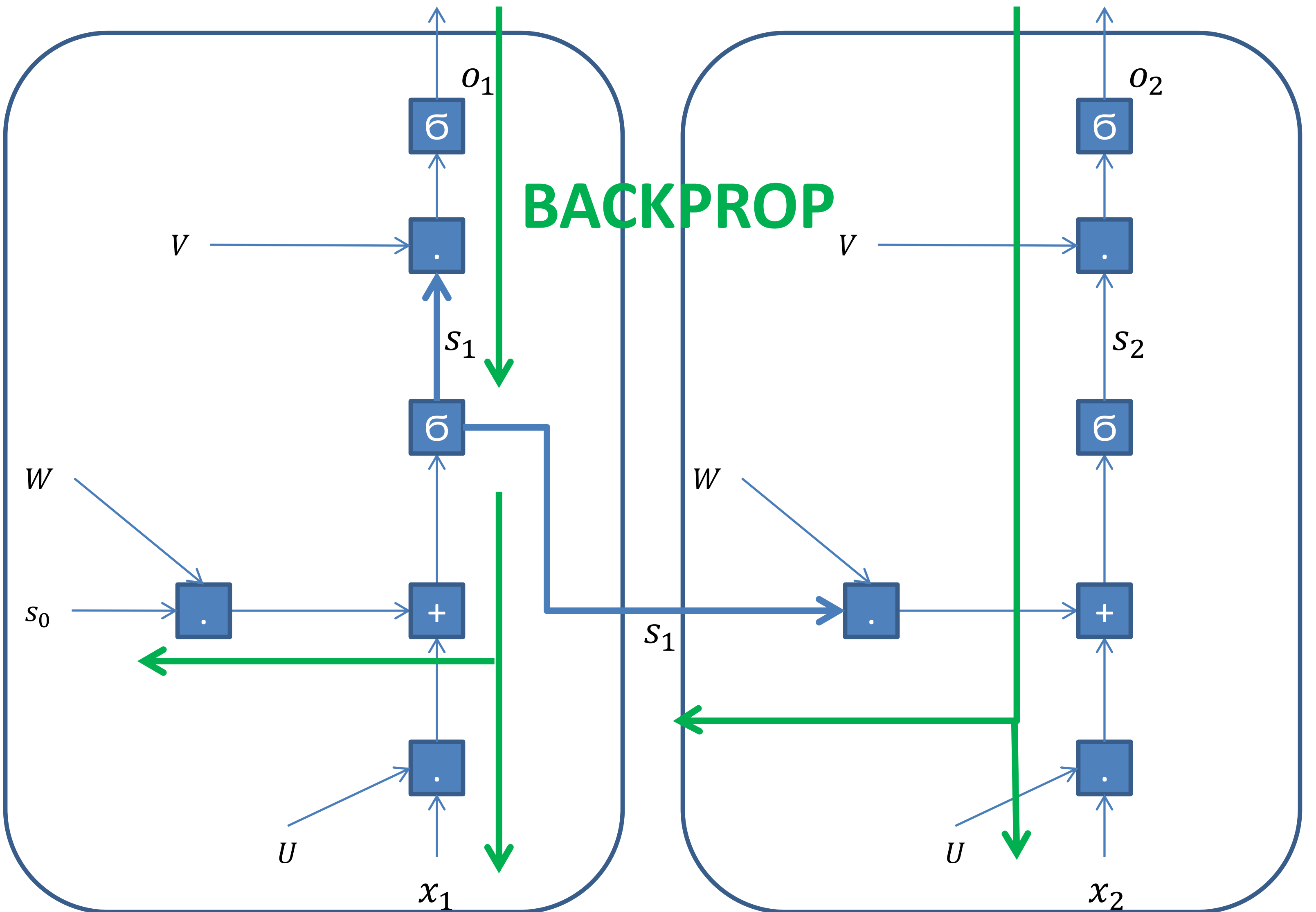
Recurrent Neural Net (RNN): Example



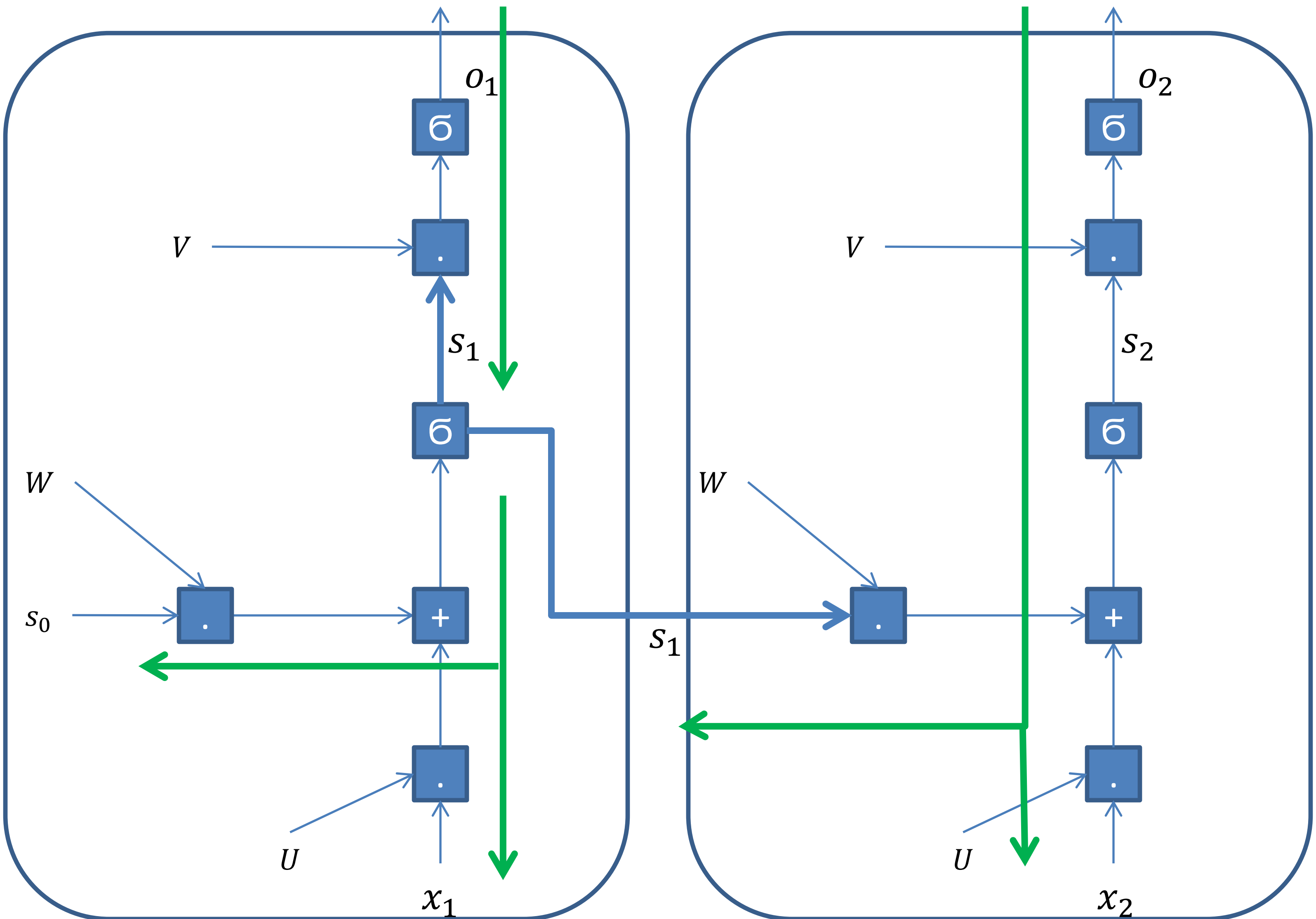
Recurrent Neural Net (RNN): Example



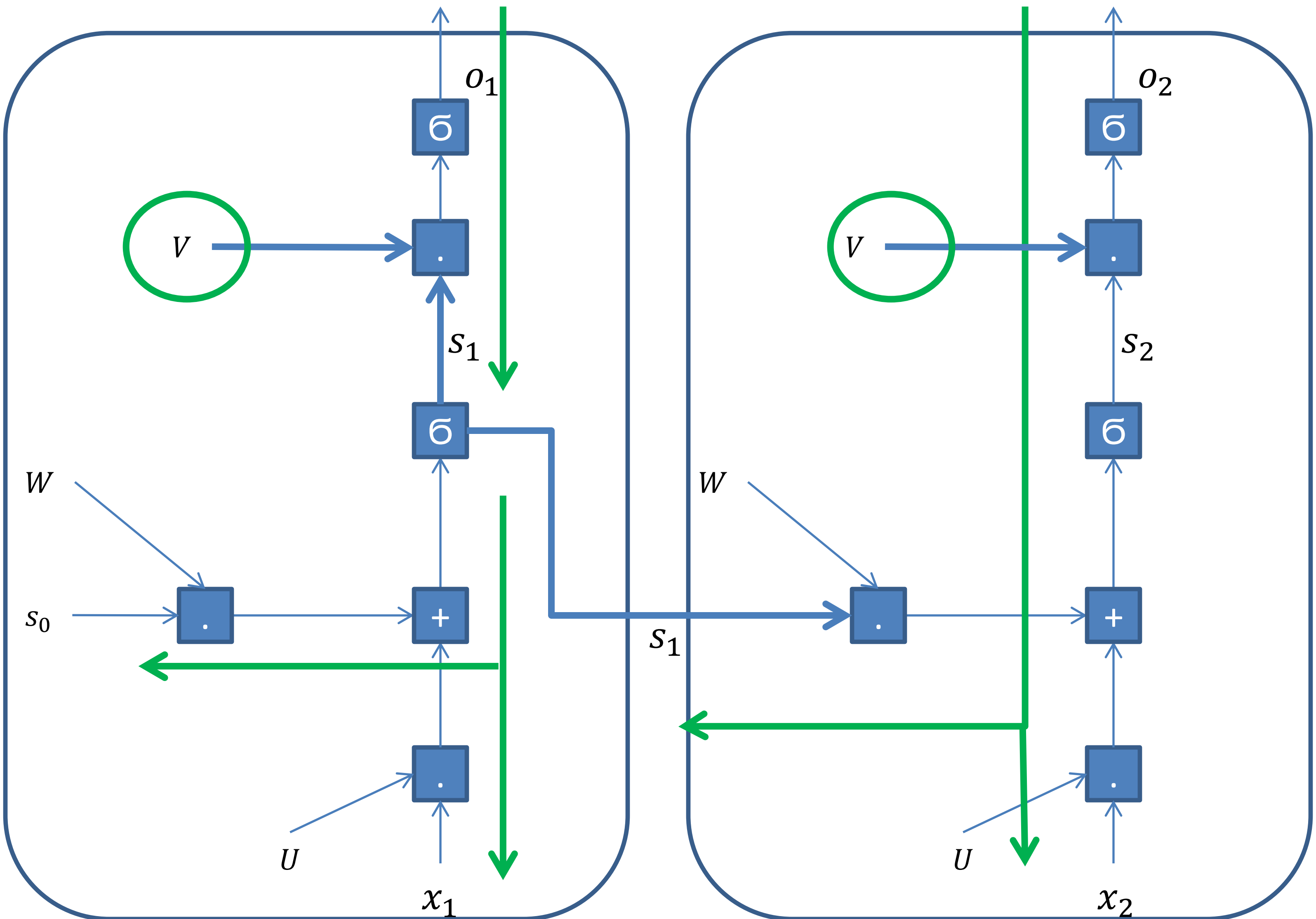
Recurrent Neural Net (RNN): Example



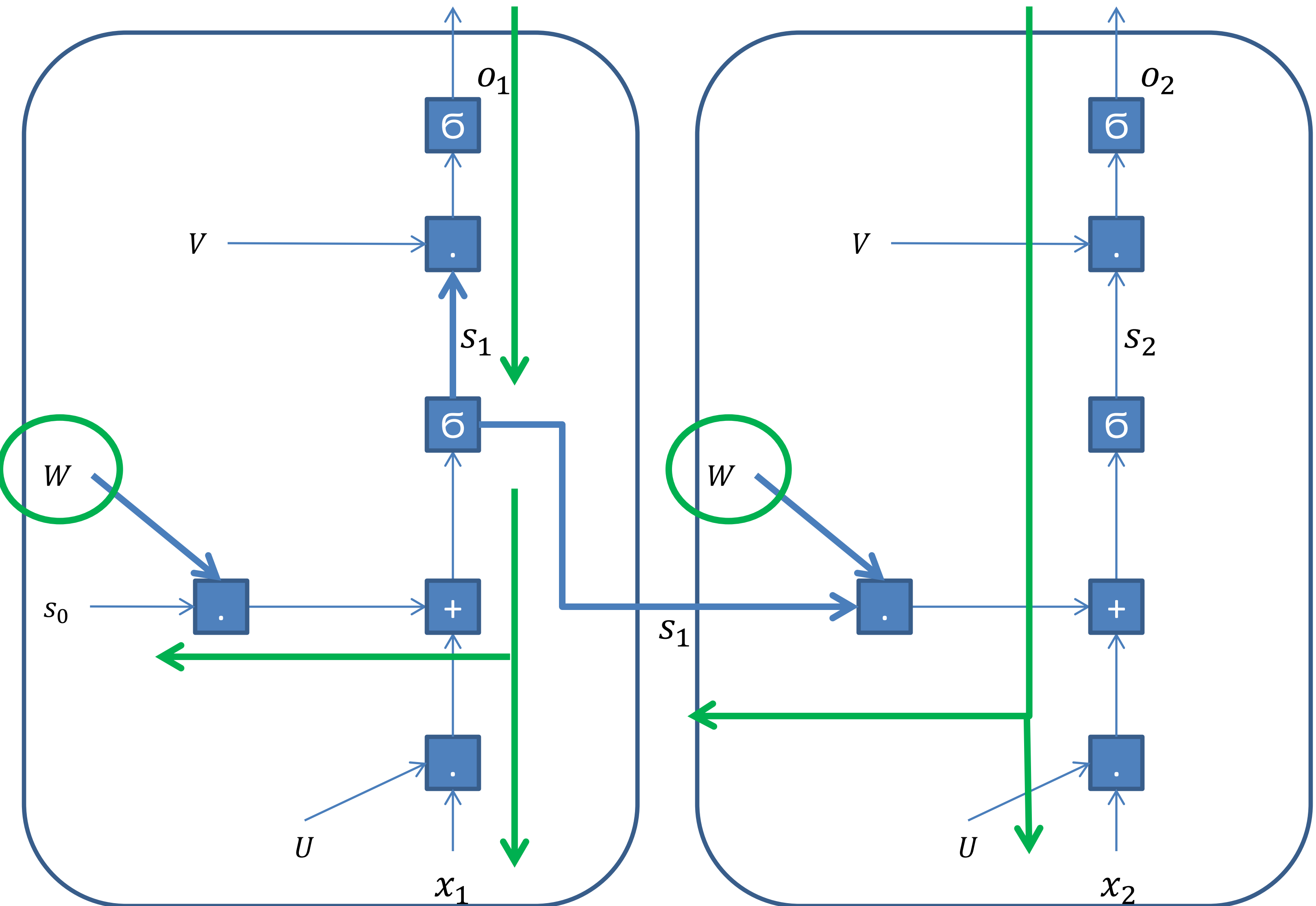
Recurrent Neural Net (RNN): Example



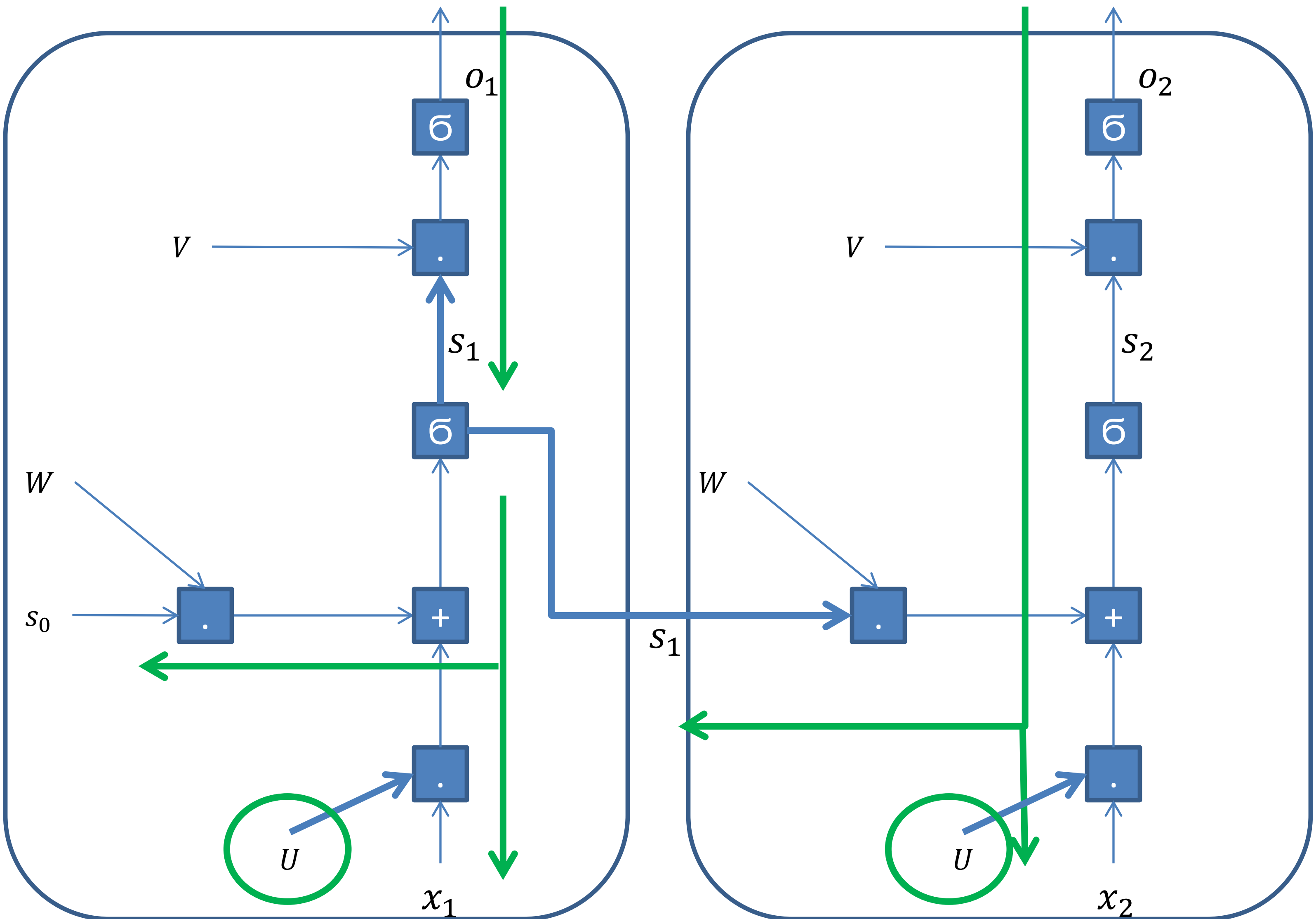
Recurrent Neural Net (RNN): Example



Recurrent Neural Net (RNN): Example



Recurrent Neural Net (RNN): Example



Recurrent Neural Net (RNN): Example (optional)

We assumed U,V and W were scalars

Work through the backprop when U, V and W are matrices.