Unit 1: Simple Neural Networks

3. Recurrent neural networks

9/22/2020

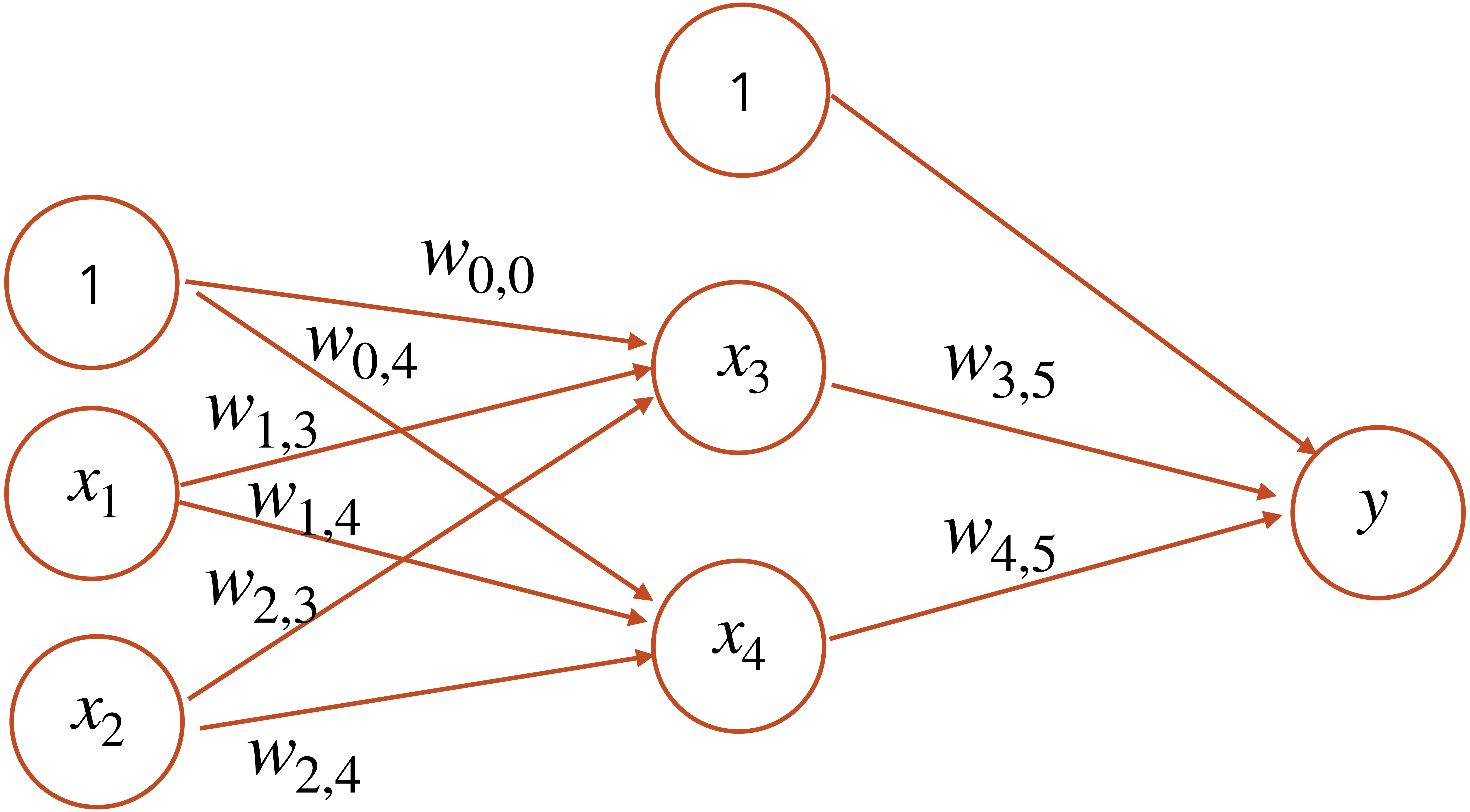
Recurrent networks

1. **Recap: How backpropagation solves the credit assignment problem**

2. **A hands-on backprop demo**

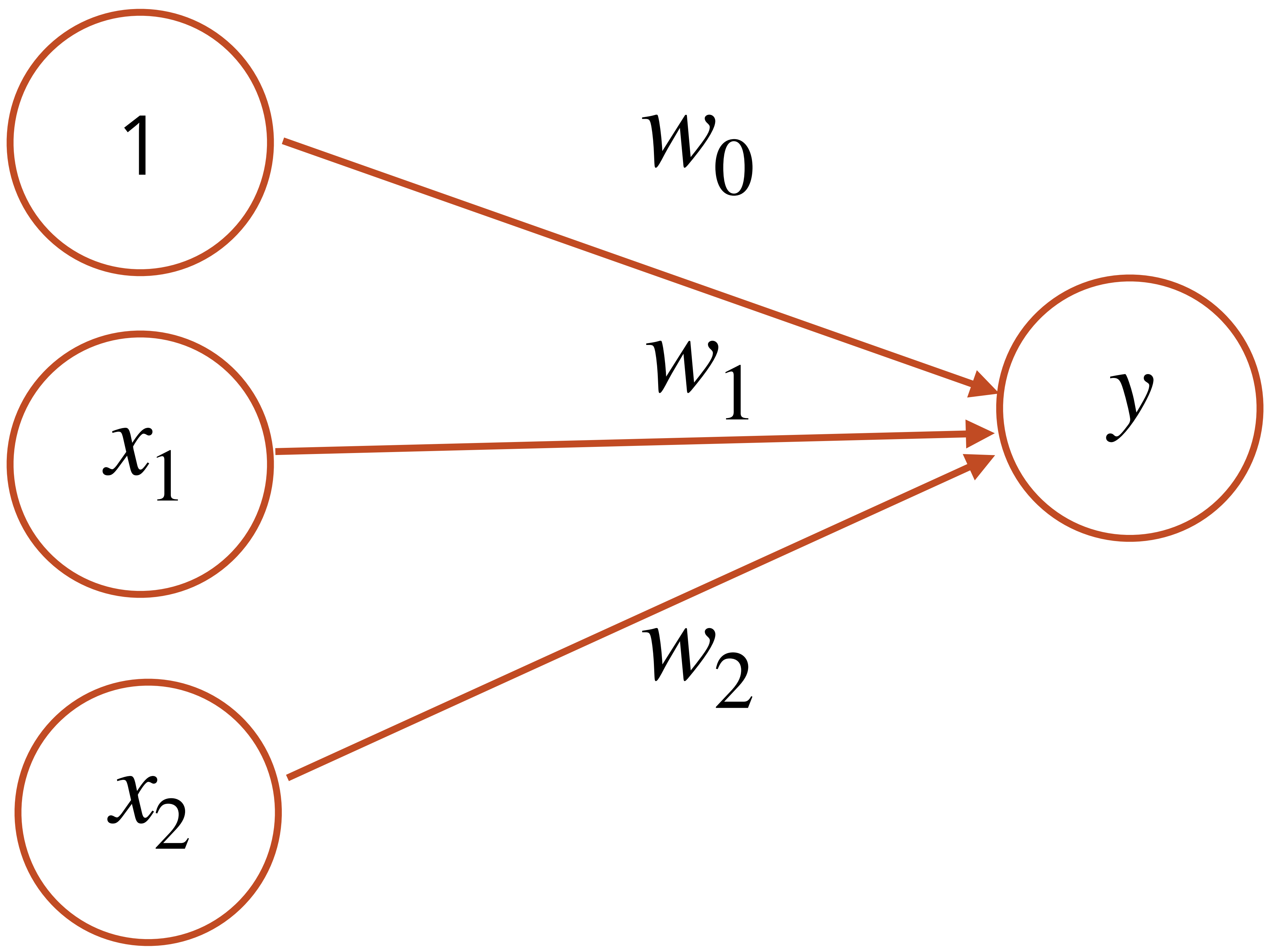
3. **Recurrent neural networks can discover structure in time**

But how do we learn connections weights in a multi-layer network?



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

What does it mean to learn in a neural network?



We got $\left(x_1, x_2\right)$

We computed $\hat{y} = f\left(w_0 + w_1 x_1 + w_1 x_1\right)$

But we wanted to predict y!

Now we want to change w_0, w_1, x_2

So next time we see $\left(x_1, x_2\right)$

We predict something closer to y

What does it mean to learn in a multi-layer network?

We got $\left(x_1, x_2\right)$

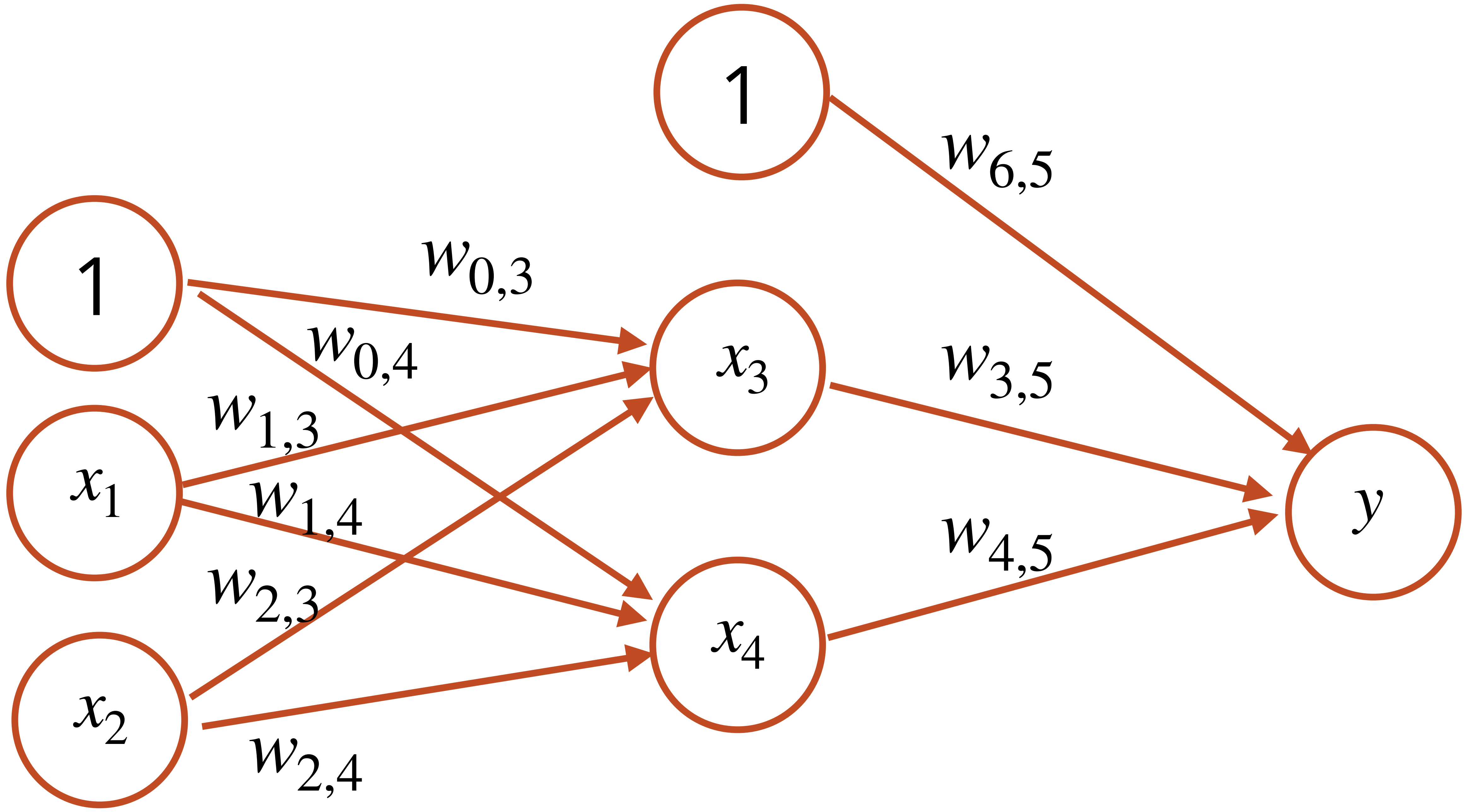
We computed $\hat{y} = f\left(w_{6,5} + w_{3,5}x_3 + w_{4,5}x_4\right)$

But we wanted to predict y !

Now we want to change $w_{0,3}, w_{0,4}, w_{1,3}, w_{1,4}, w_{2,3}, w_{2,4}, w_{3,5}\ w_{4,5}, w_{6,5}$

So next time we see $\left(x_1, x_2\right)$

We predict something closer to y



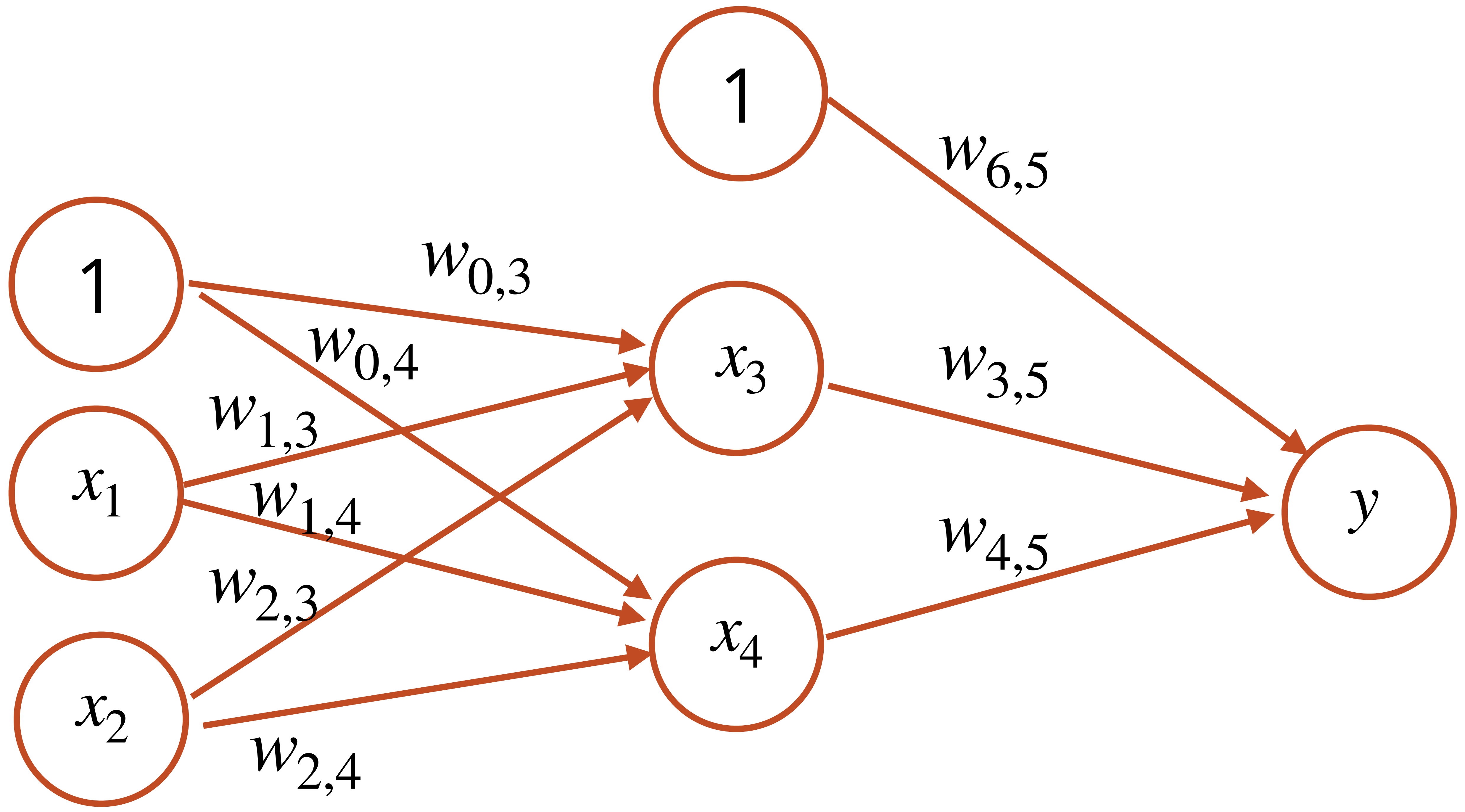
The credit assignment problem

Goal: Change each weight in proportion to how much it contributed to the error $(y - \hat{y})^2$

For **hidden layer** weights, we can compute this directly

$$\Delta w_i \approx \alpha \cdot (y - \hat{y})\, x_i$$

Why can't we do this for **input layer** weights?

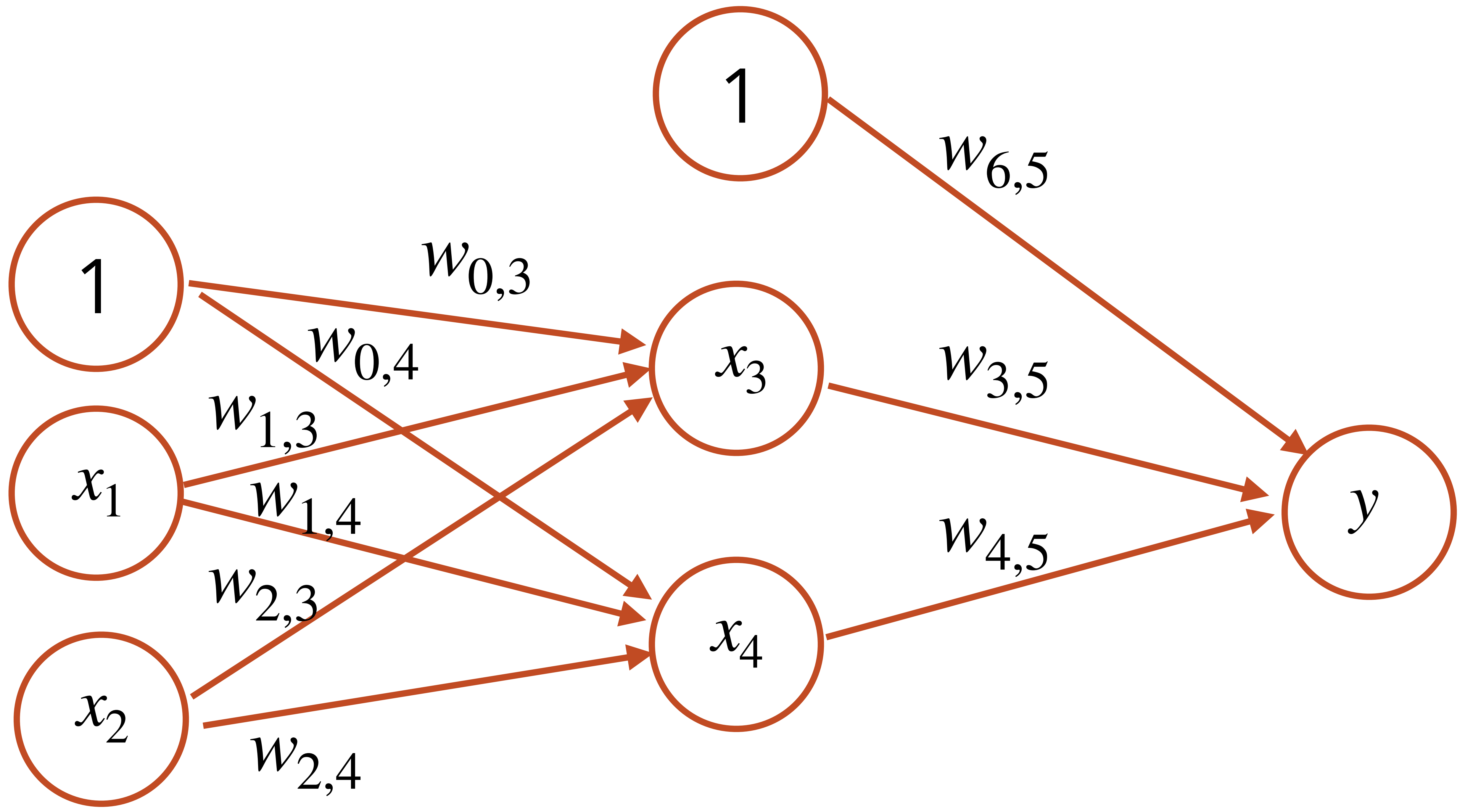


The credit assignment problem

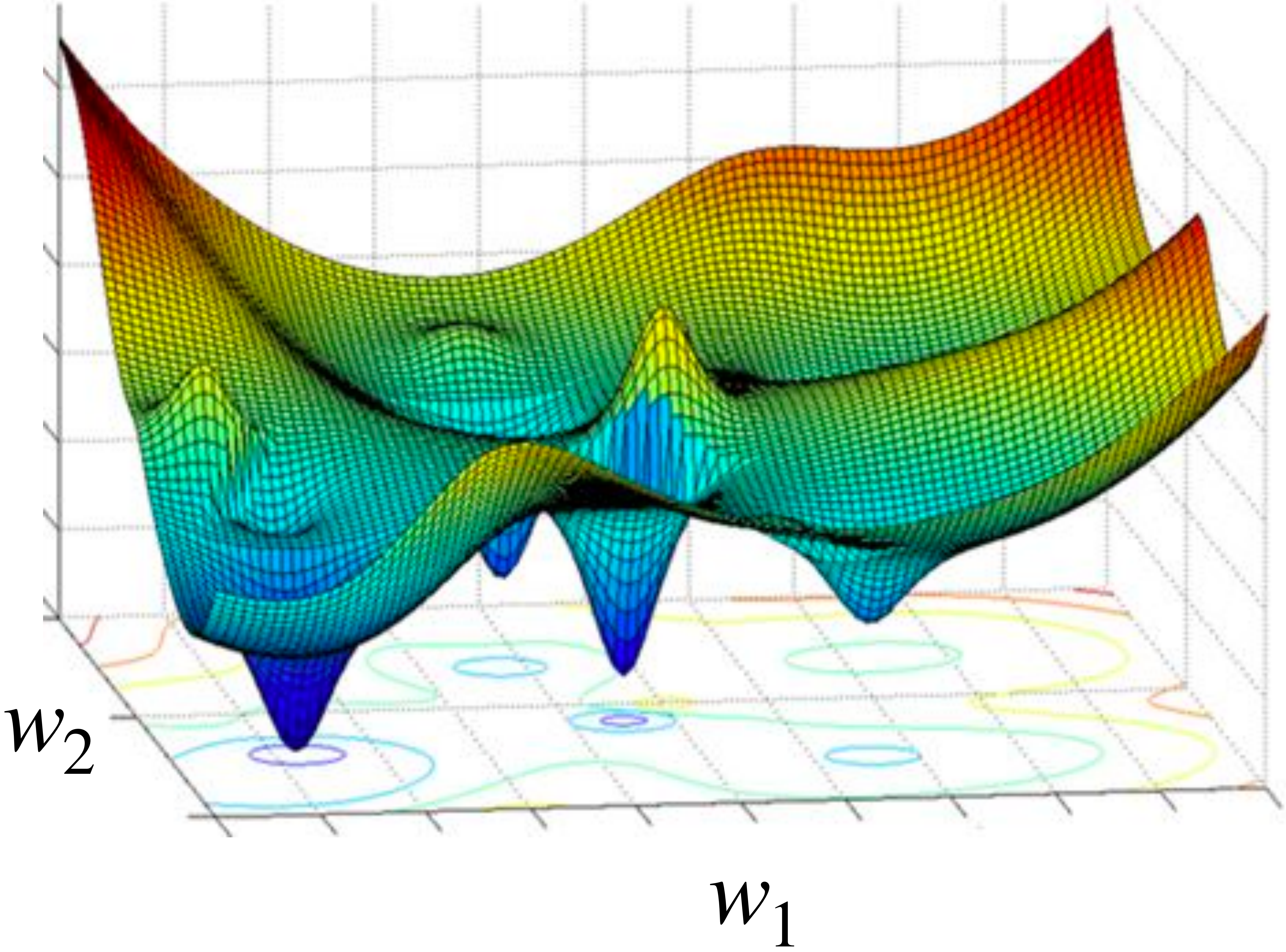
Goal: Change each weight in proportion to how much it contributed to the error $\left(y - \hat{y}\right)^2$

Input layer weights indirectly contribute to their error by directly affecting the activation of **hidden layer units**.

We can compute the contribution of **hidden layer units** to the error.



Gradient Descent



The credit assignment equation

The direction of the error gradient for $w_{3,5}$

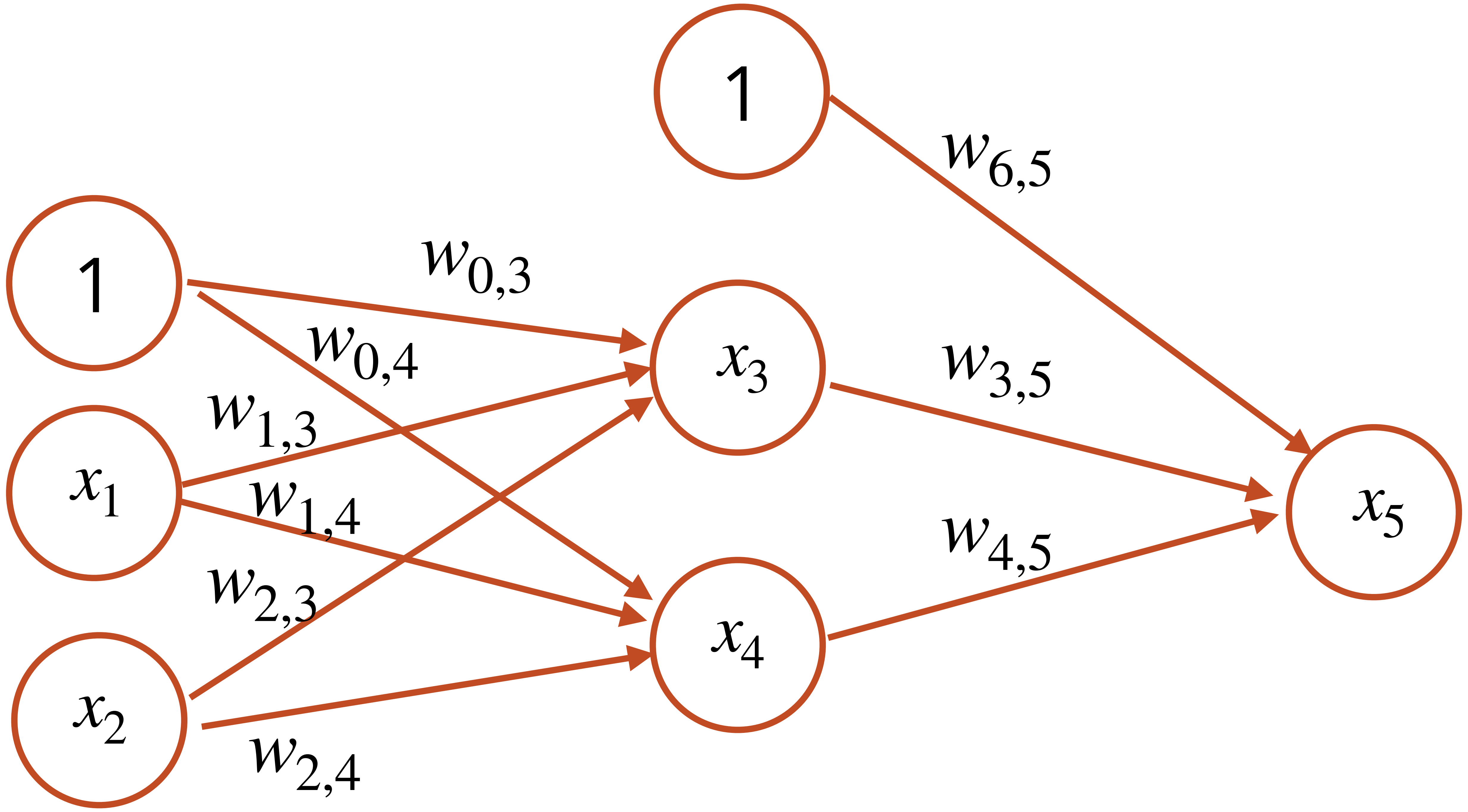
How does $w_{3,5}$ contribute to error?

$w_{3,5}$ Changes input to x_5

x_5 Changes its activation a_{x_5}

a_{x_5} Contributes directly to error

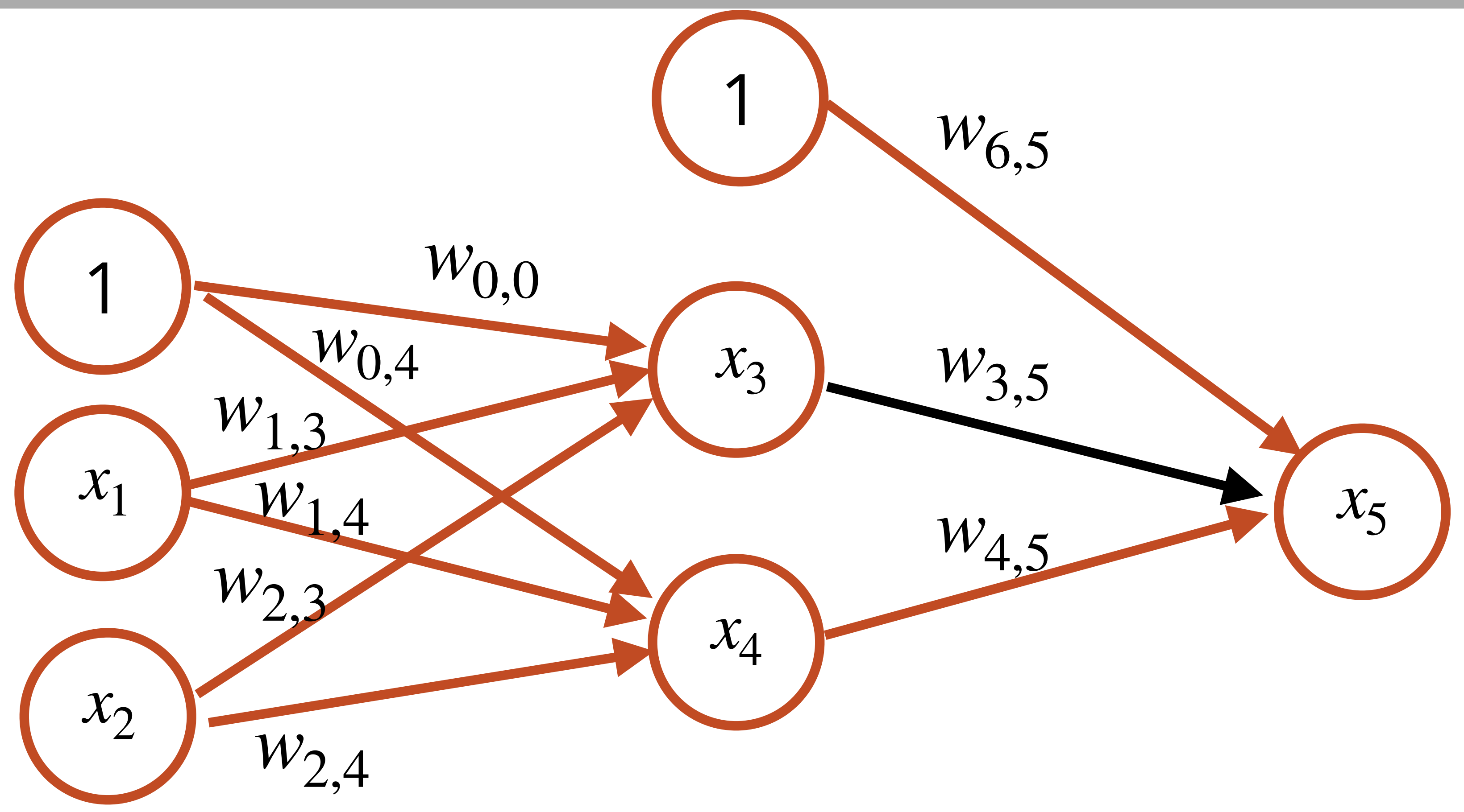
$$\frac{\partial E}{\partial w_{3,5}} = \frac{\partial x_5}{\partial w_{3,5}} \frac{\partial a_{x_5}}{\partial x_5} \frac{\partial E}{\partial a_{x_5}}$$



The chain rule of derivatives says we can multiple these

Updating one weight

$$\frac{\partial E}{\partial w_{3,5}} = \frac{\partial x_5}{\partial w_{3,5}} \frac{\partial a_{x_5}}{\partial x_5} \frac{\partial E}{\partial a_{x_5}}$$



$$\frac{\partial E}{\partial a_{x_5}} = 2\left(y - a_{x_5}\right) \qquad E = \left(y - a_{x5}\right)^2$$

$$\frac{\partial a_{x_5}}{\partial x_5} = \sigma\left(x_5\right)\left(1 - \sigma\left(x_5\right)\right) \qquad \sigma'(x) = \sigma(x)\left(1 - \sigma(x)\right)$$

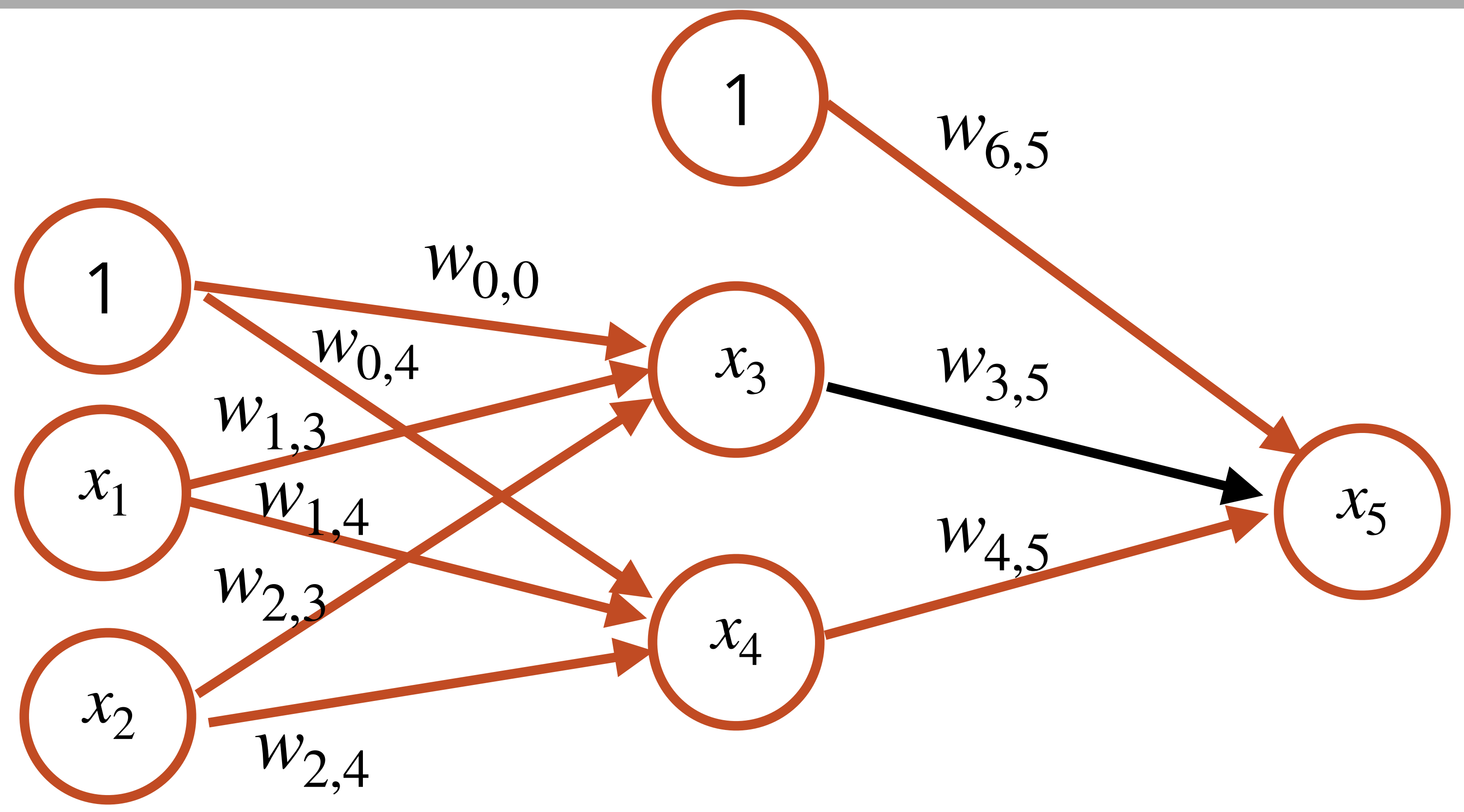
$$\frac{\partial x_5}{\partial w_{3,5}} = a_{x_3} \qquad x_5 = w_{6,5} + w_{3,5} \cdot a_{x_3} + w_{4,5} \cdot a_{x_4}$$

Updating one weight

$$\frac{\partial E}{\partial w_{3,5}} = \frac{\partial x_5}{\partial w_{3,5}} \frac{\partial a_{x_5}}{\partial x_5} \frac{\partial E}{\partial a_{x_5}}$$

$$= 2\left(y - a_{x_5}\right) \sigma'\left(x_5\right) a_{x_3}$$

$$\Delta w_i = \alpha \cdot \left(y - \hat{y}\right) x_i$$



The credit assignment equation

The direction of the error gradient for $w_{1,3}$

How does $w_{1,3}$ contribute to error?

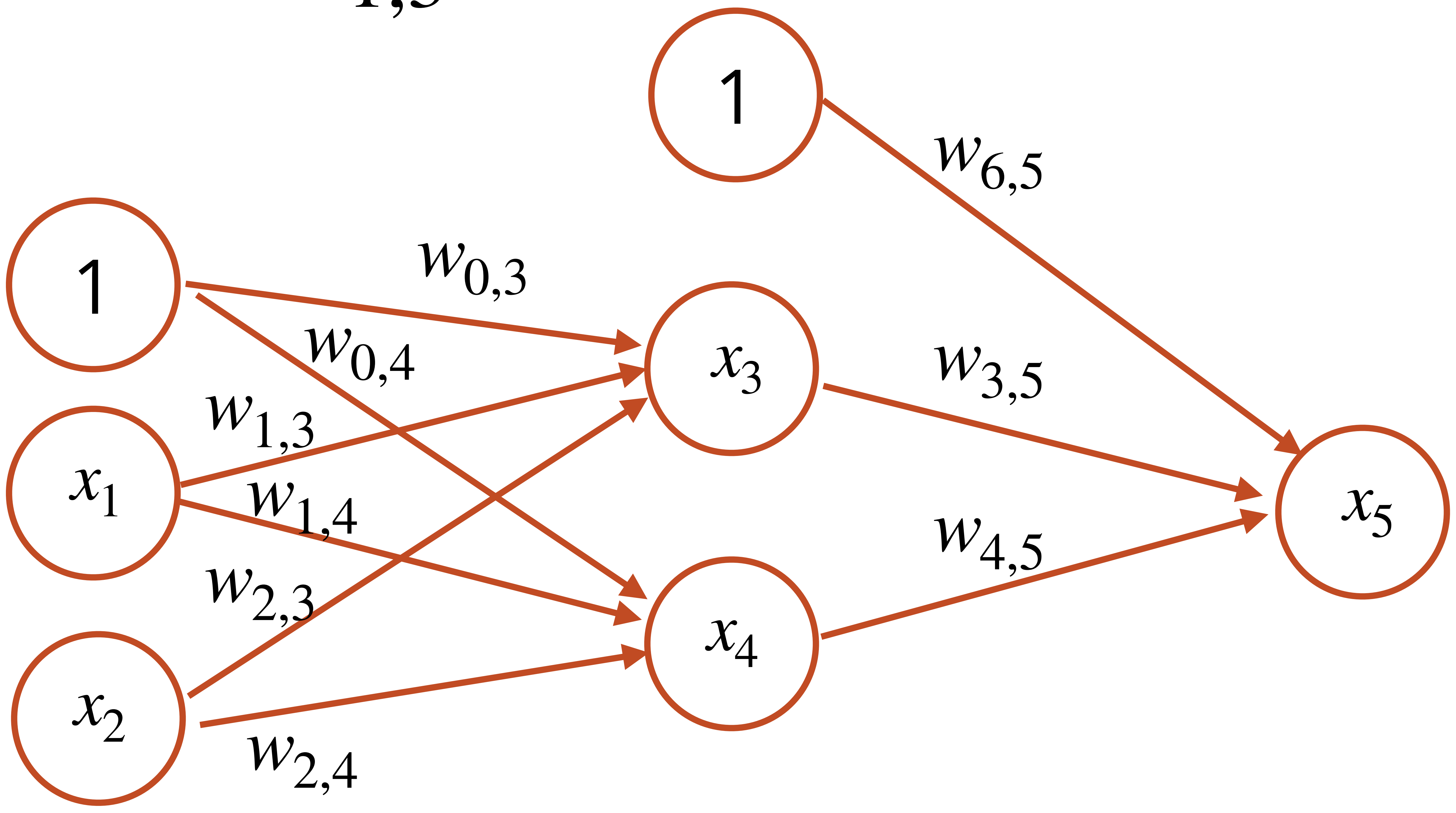
$w_{1,3}$ Changes input to x_3

x_3 Changes its activation a_{x_3}

a_{x_3} Changes the input to x_5

x_5 Changes its activation a_{x_5}

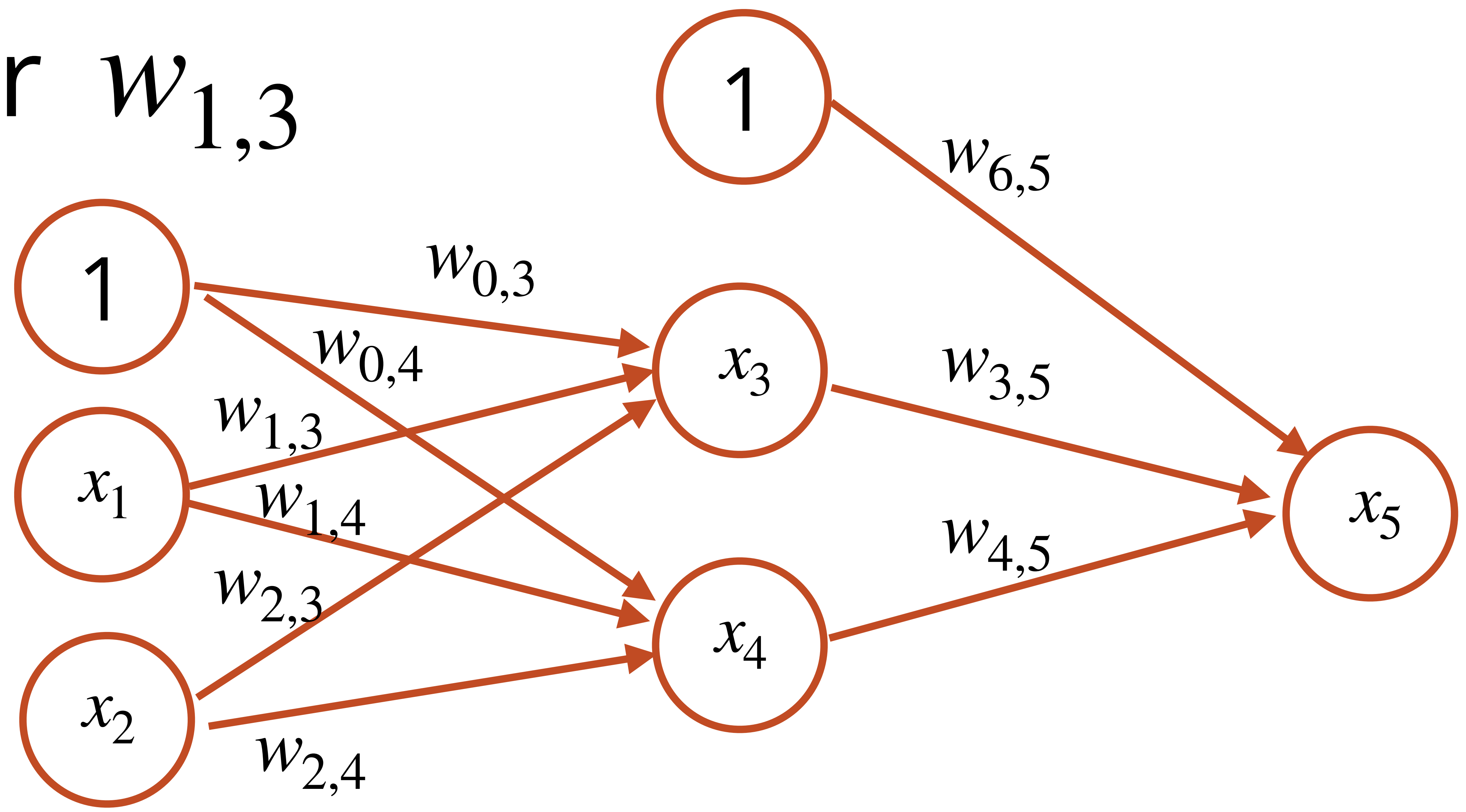
a_{x_5} Contributes directly to error



$$\frac{\partial E}{\partial w_{1,3}} = \frac{\partial x_3}{\partial w_{1,3}} \frac{\partial a_{x_3}}{\partial x_3} \frac{\partial x_5}{\partial a_{x_3}} \frac{\partial a_{x_5}}{\partial x_5} \frac{\partial E}{\partial a_{x_5}}$$

The credit assignment equation

The direction of the error gradient for $w_{1,3}$



How does $w_{1,3}$ contribute to error?

$w_{1,3}$ Changes input to x_3

x_3 Changes its activation a_{x_3}

a_{x_3} Contributes indirectly to the error

$$\frac{\partial E}{\partial w_{1,3}} = \frac{\partial x_3}{\partial w_{1,3}} \frac{\partial a_{x_3}}{\partial x_3} \boxed{\frac{\partial E}{\partial a_{x_3}}}$$

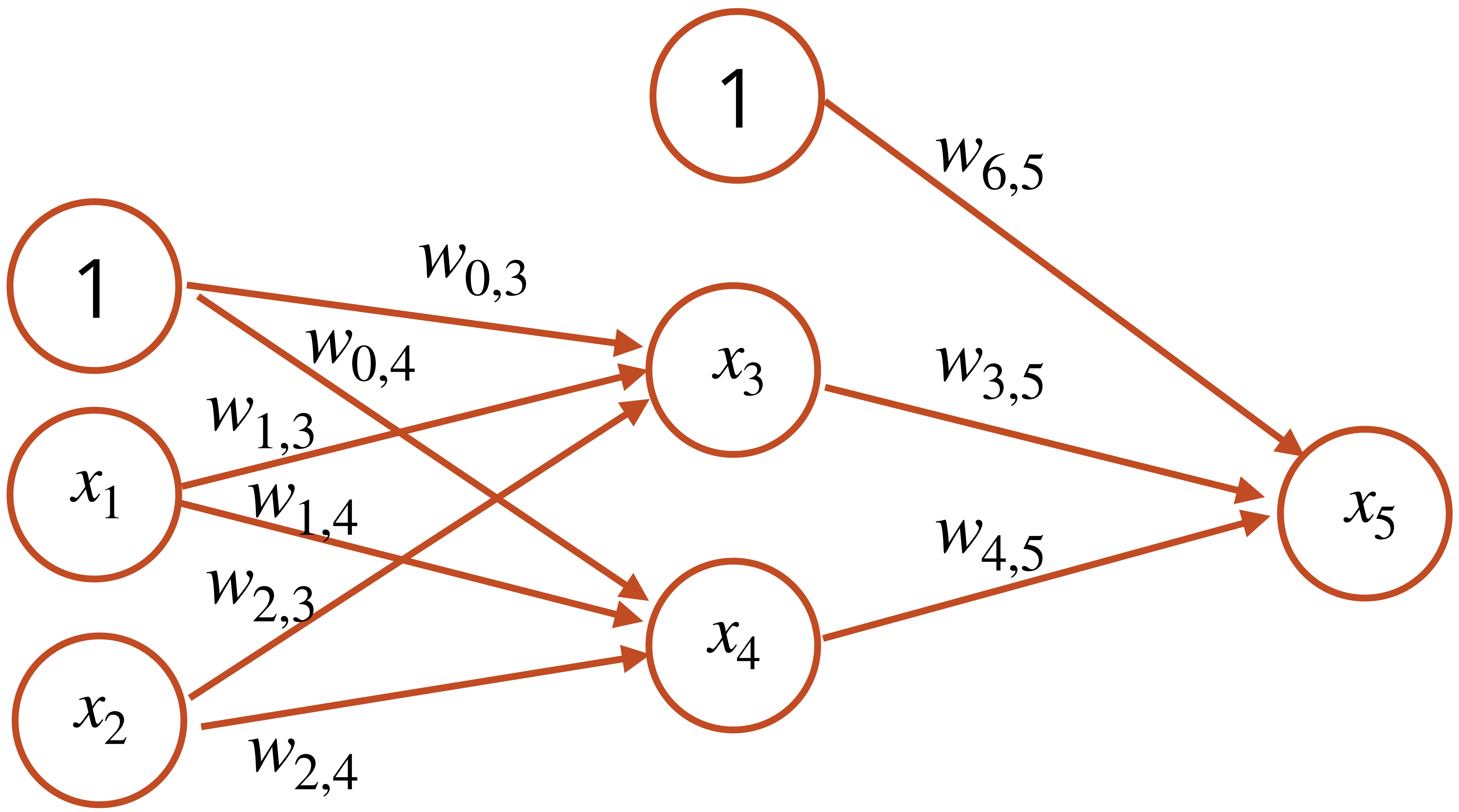
The backpropagation strategy

First compute how much to change weights at the layer closest to the output

$$\Delta w_i = 2\left(y - a_{x_5}\right)\sigma'\left(x_5\right)a_{x_i}$$

Then go back one layer at a time, using the error of the previous layer as the new target

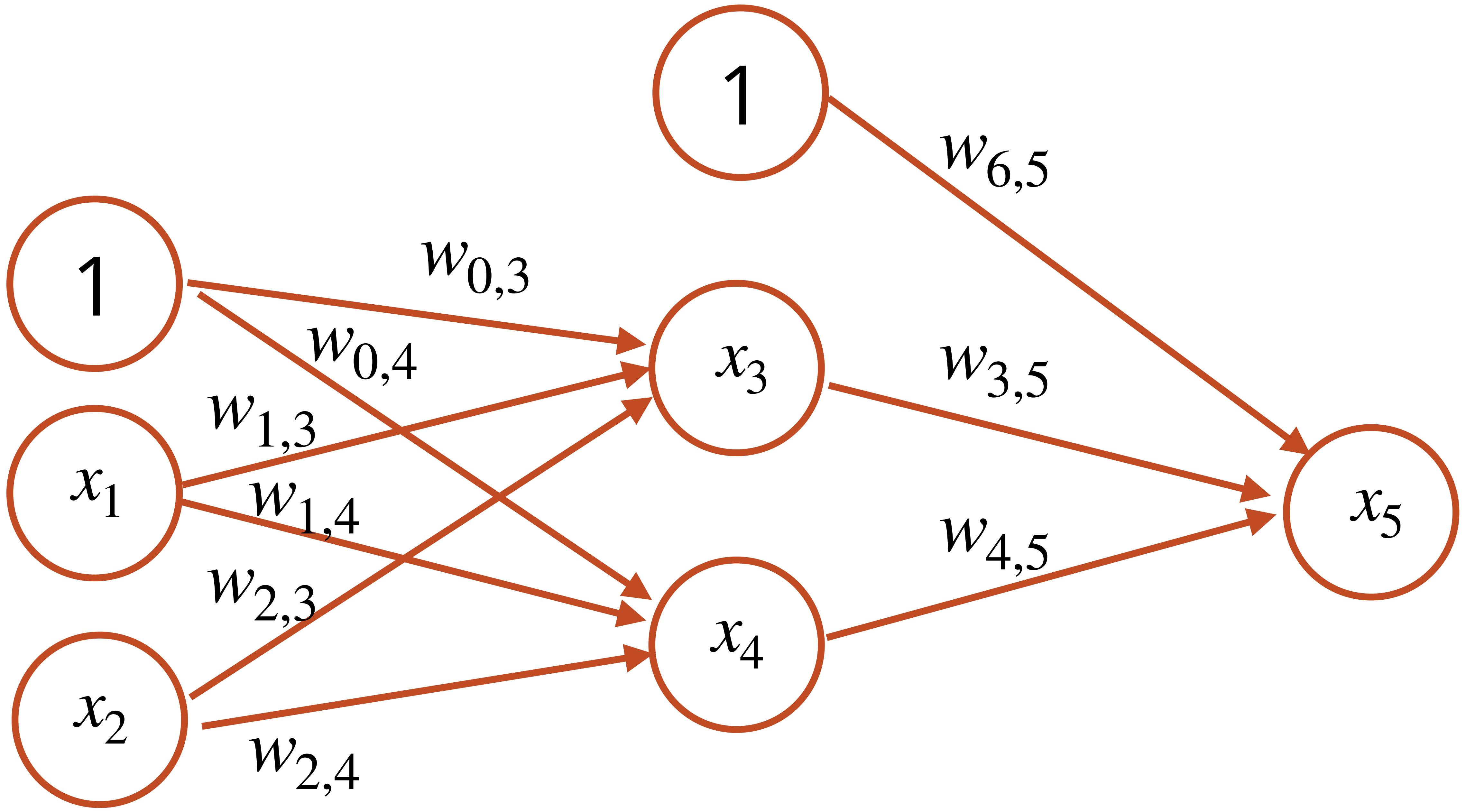
$$\Delta w_i = \sigma'\left(x_i\right)a_{x_j} \cdot \frac{\partial E}{\partial a_{x_j}}$$



Let's try an example

$$\Delta w_i = E'\left(a_{x_5}\right)\sigma'\left(x_5\right)a_{x_i}$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



$$\Delta w_i = \sigma'\left(x_i\right)a_{x_j}\cdot E'\left(a_{x_5}\right)\sigma'\left(x_5\right)w_{x_i,5}$$

Let's try an example

Make a copy of these Google slides:

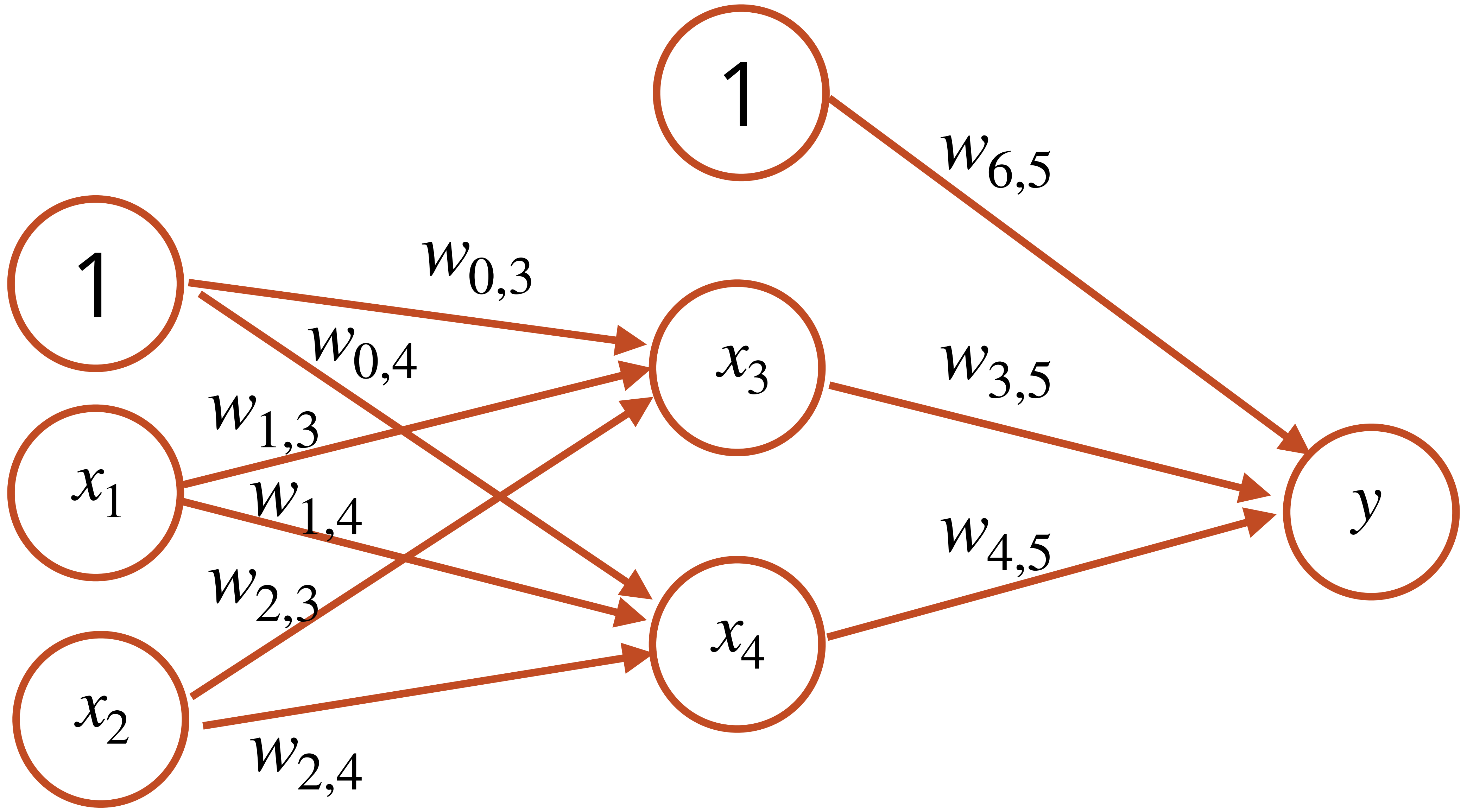
http://bit.ly/backprop-demo

Make a copy of this Google sheet for computation:

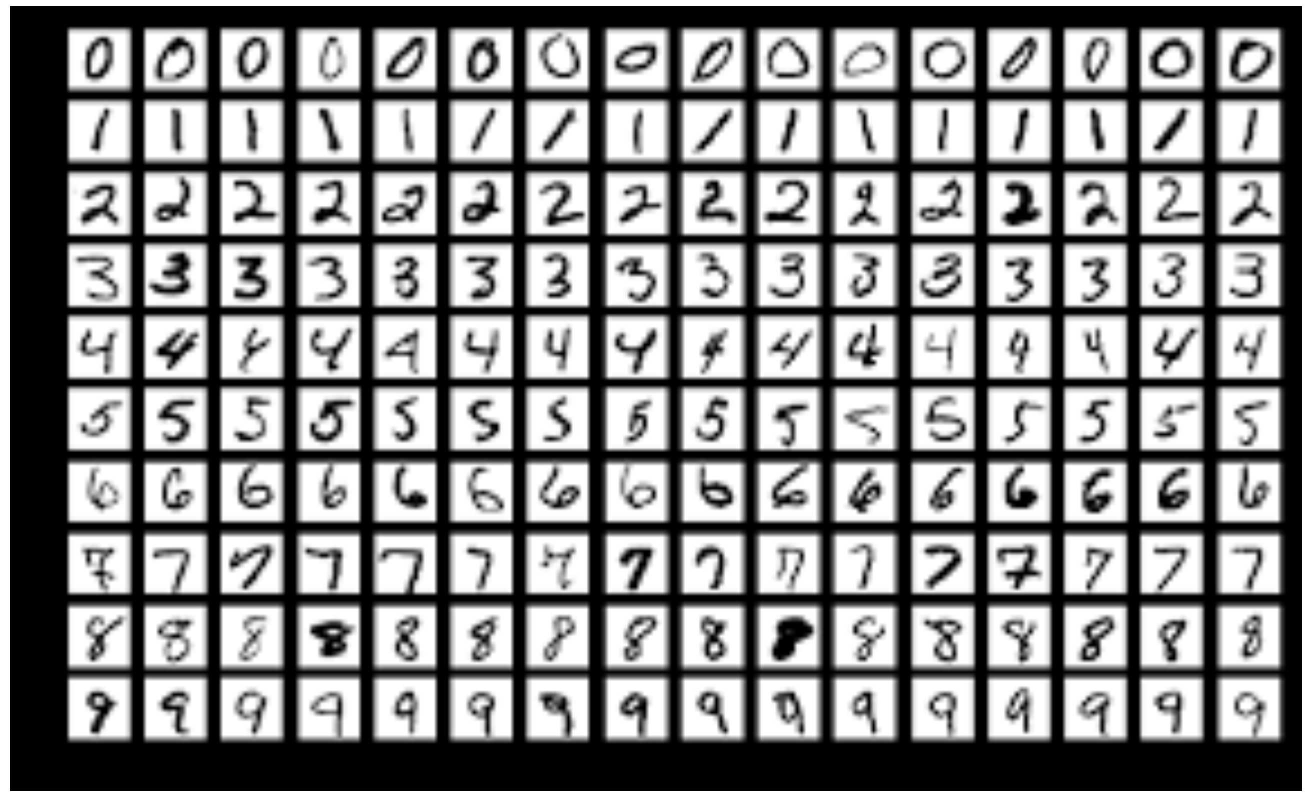
http://bit.ly/backprop-math

What these networks can do

Networks like this one can solve problems where there is structure in co-occurrence



With a little modification, they can also find structure in space (as you'll see in the Homework 2)



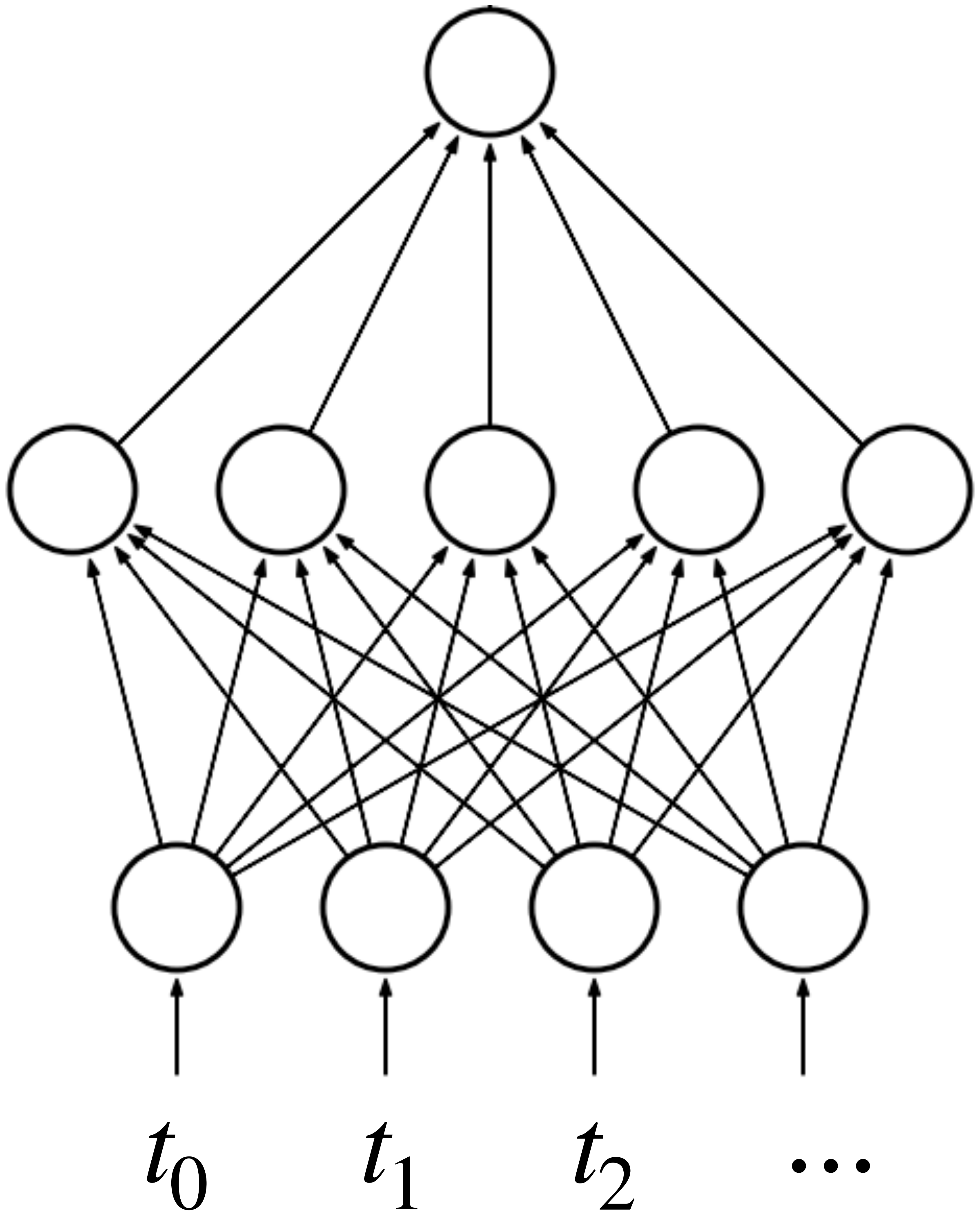
What about structure in time?

Many of the things people learn, and we want machines to learn are about **structure in time**

From our affinity diagram:

- Learning a song
- Learning to knit
- Playing video games
- Learning a dance routine
- Driving
- Cooking
- …

How would we represent structure in a network?

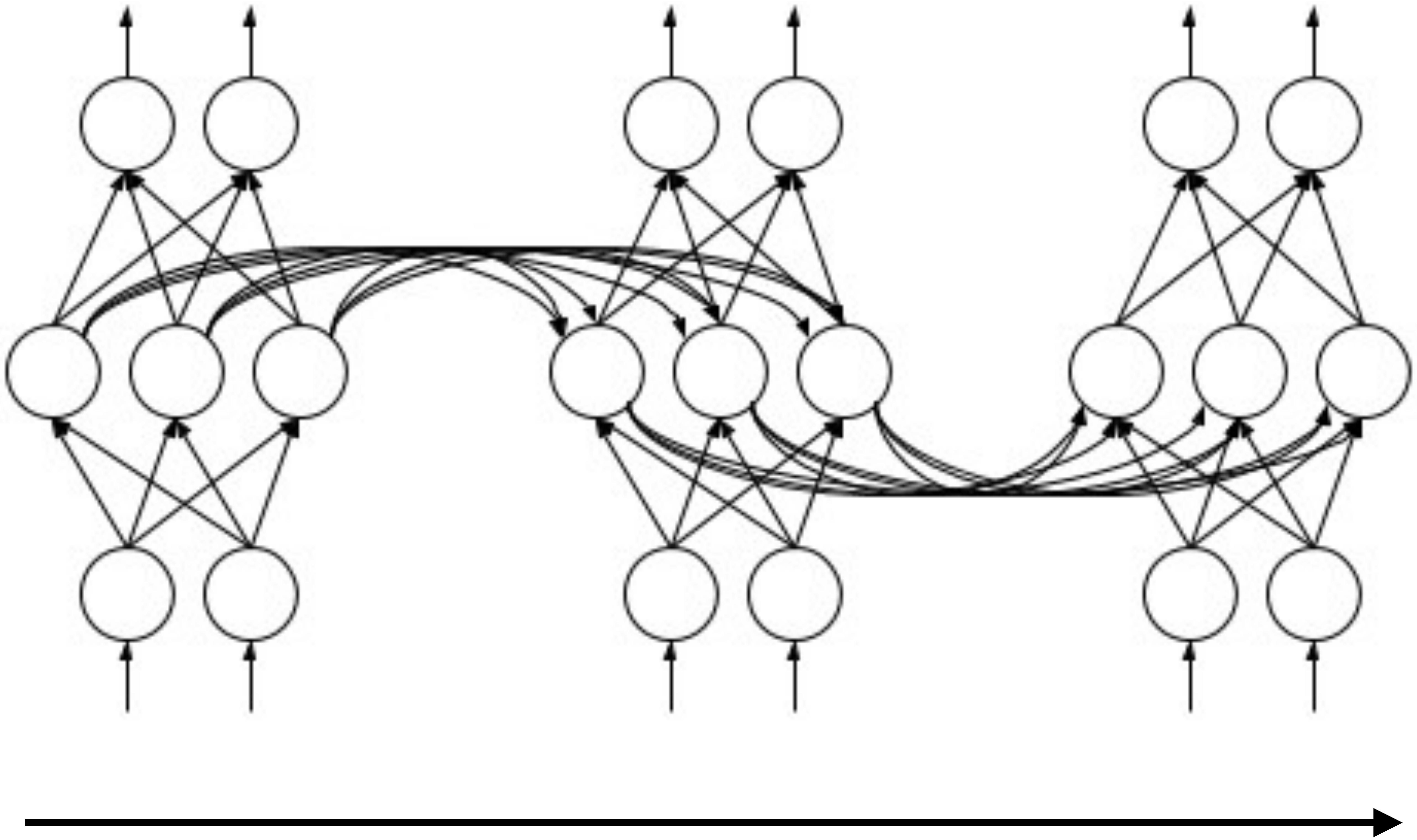


Naive approach: Time as space

Problems:

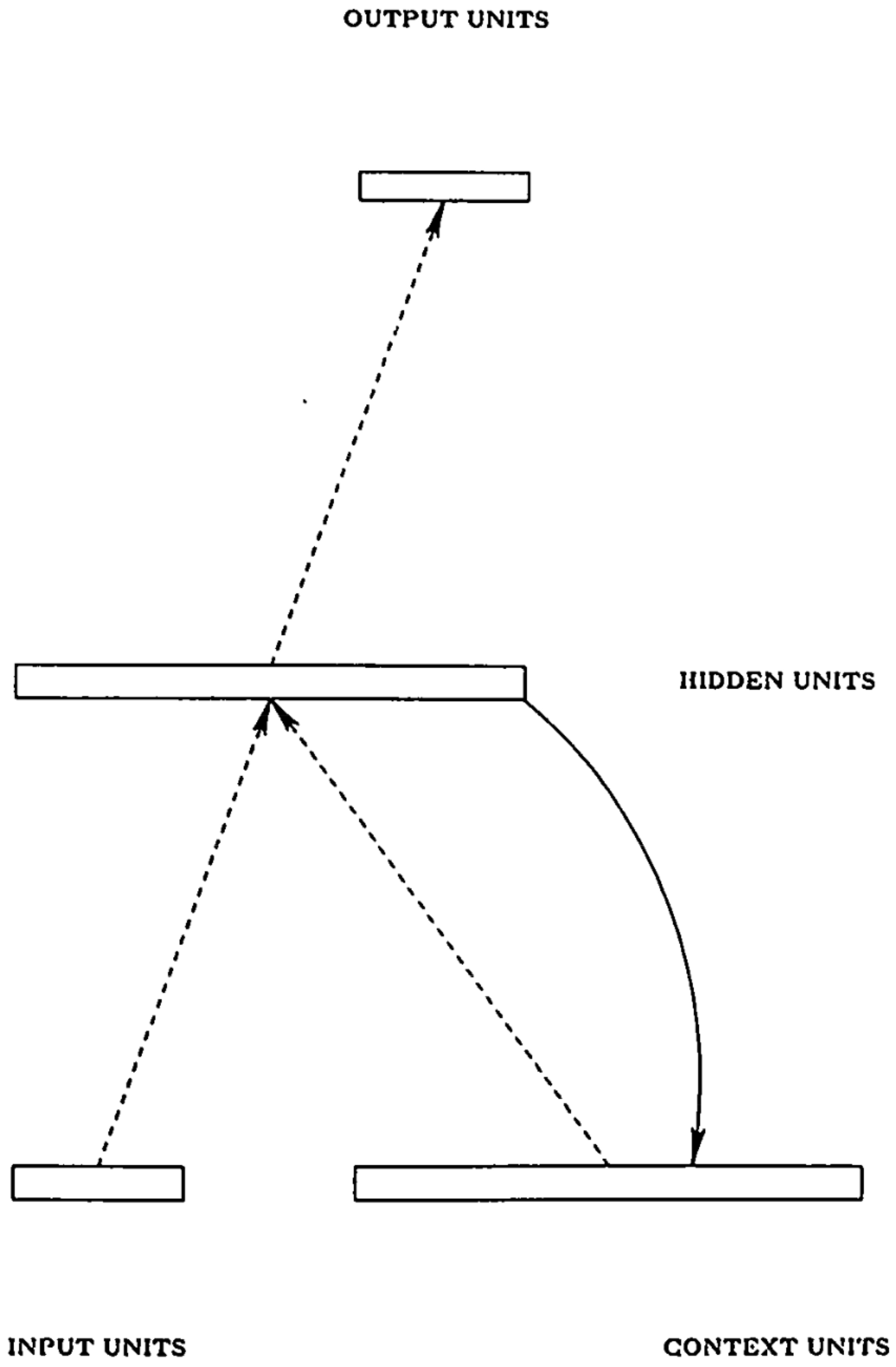
- How big do you make the buffer?
- Two identical patterns translated in time have no natural overlap, e.g. **[0 1 1 0 0 0]** and **[0 0 0 1 1 0]**

Recurrent Neural Networks



Time

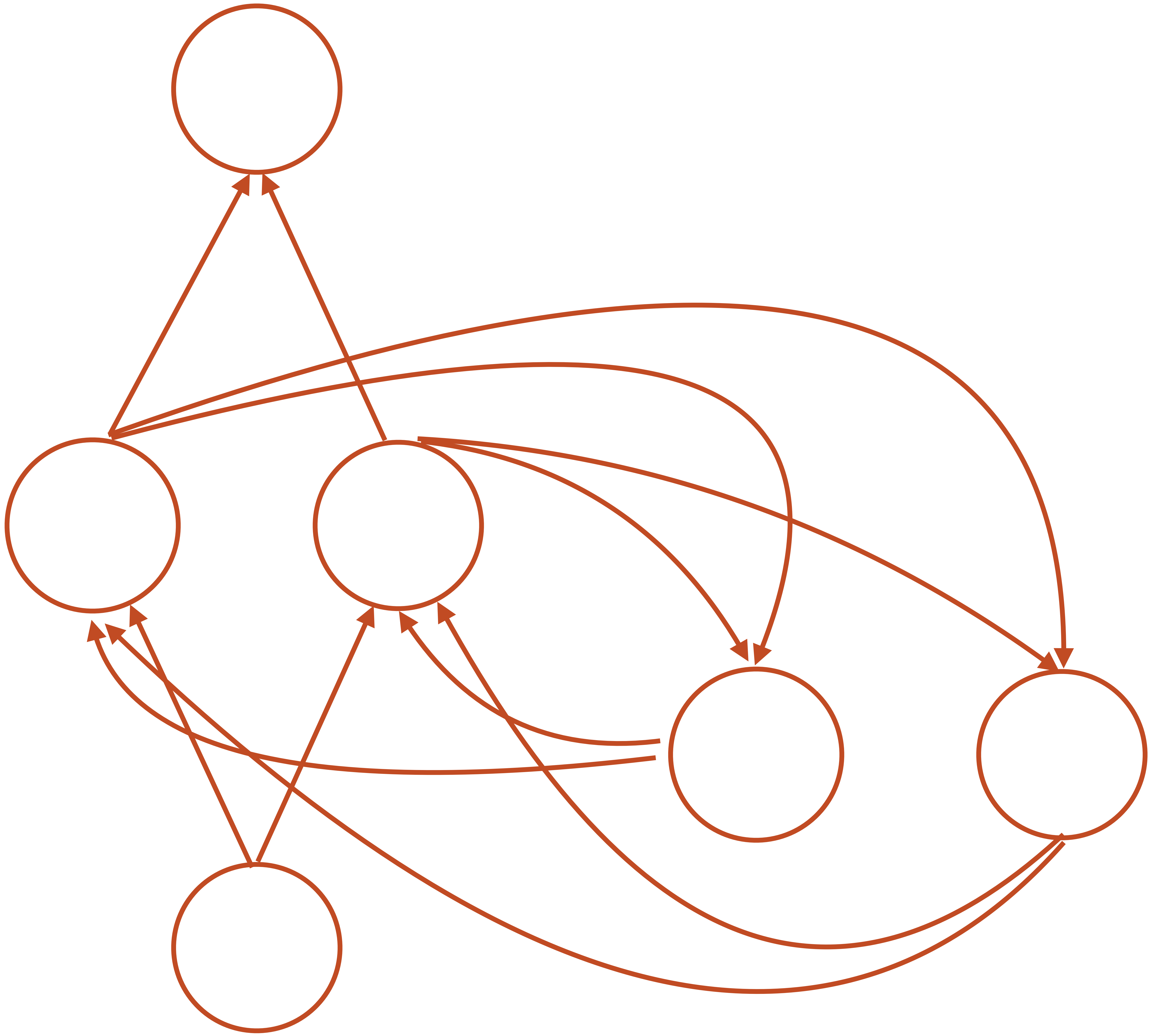
Simple Recurrent Networks (Elman Networks - Elman, 1990)



A set of **context units** that are an exact copy of the hidden layer at $t-1$

The hidden layer at time t gets input from both the **input units** and the **context units**

XOR in an Elman network



Output

0 1 0 0 0 0 1 1 1 1 0 1 0 1 ?

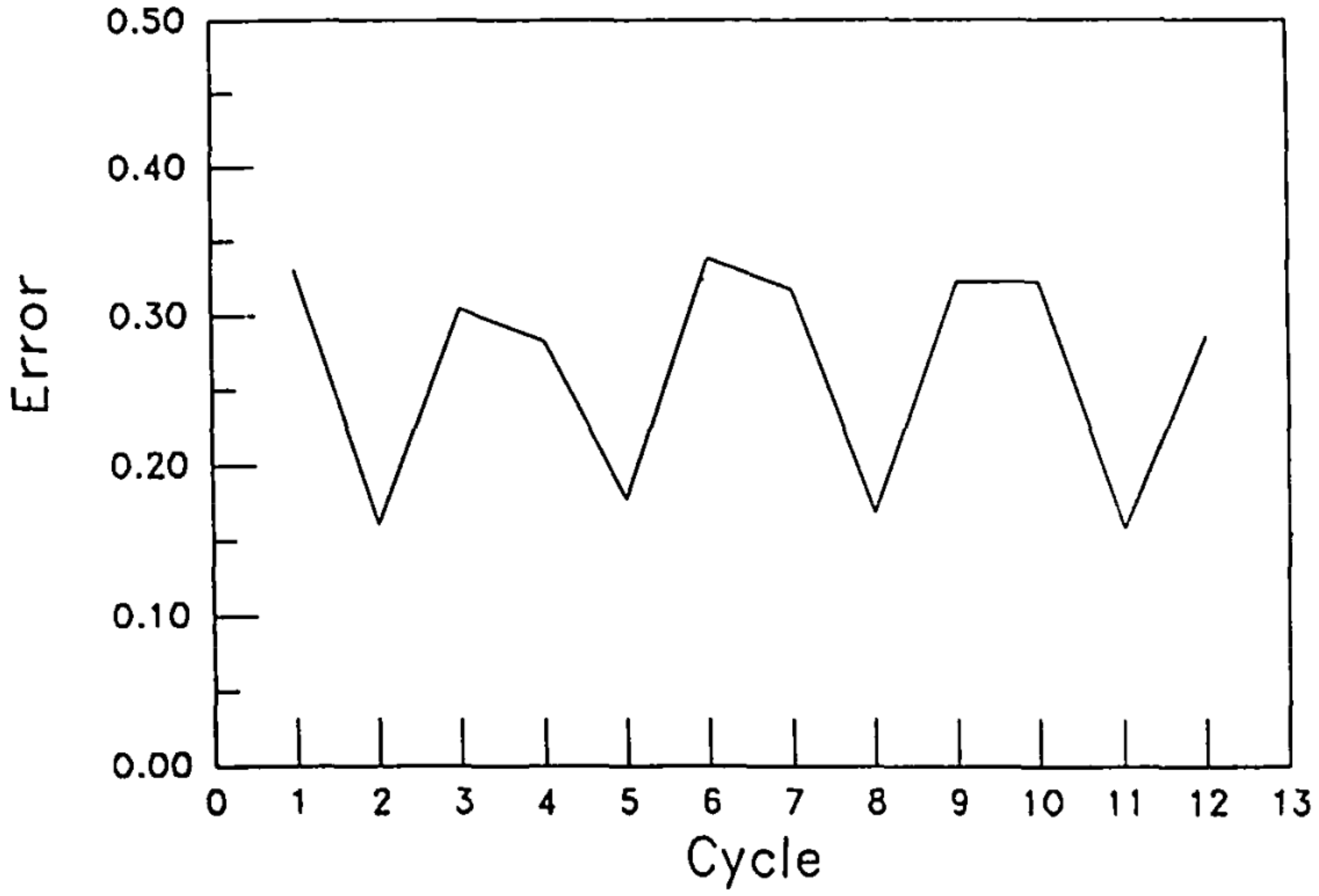
1 0 1 0 0 0 0 1 1 1 1 0 1 0 1…

Input

Goal: Output is XOR of previous 2 inputs

Error after training

What is happening here?



Output

0 1 0 0 0 0 1 1 1 1 0 1 0 1 ?

1 0 1 0 0 0 0 1 1 1 1 0 1 0 1...

Input

How far back does the network remember?

Input

A random concatenation of the words

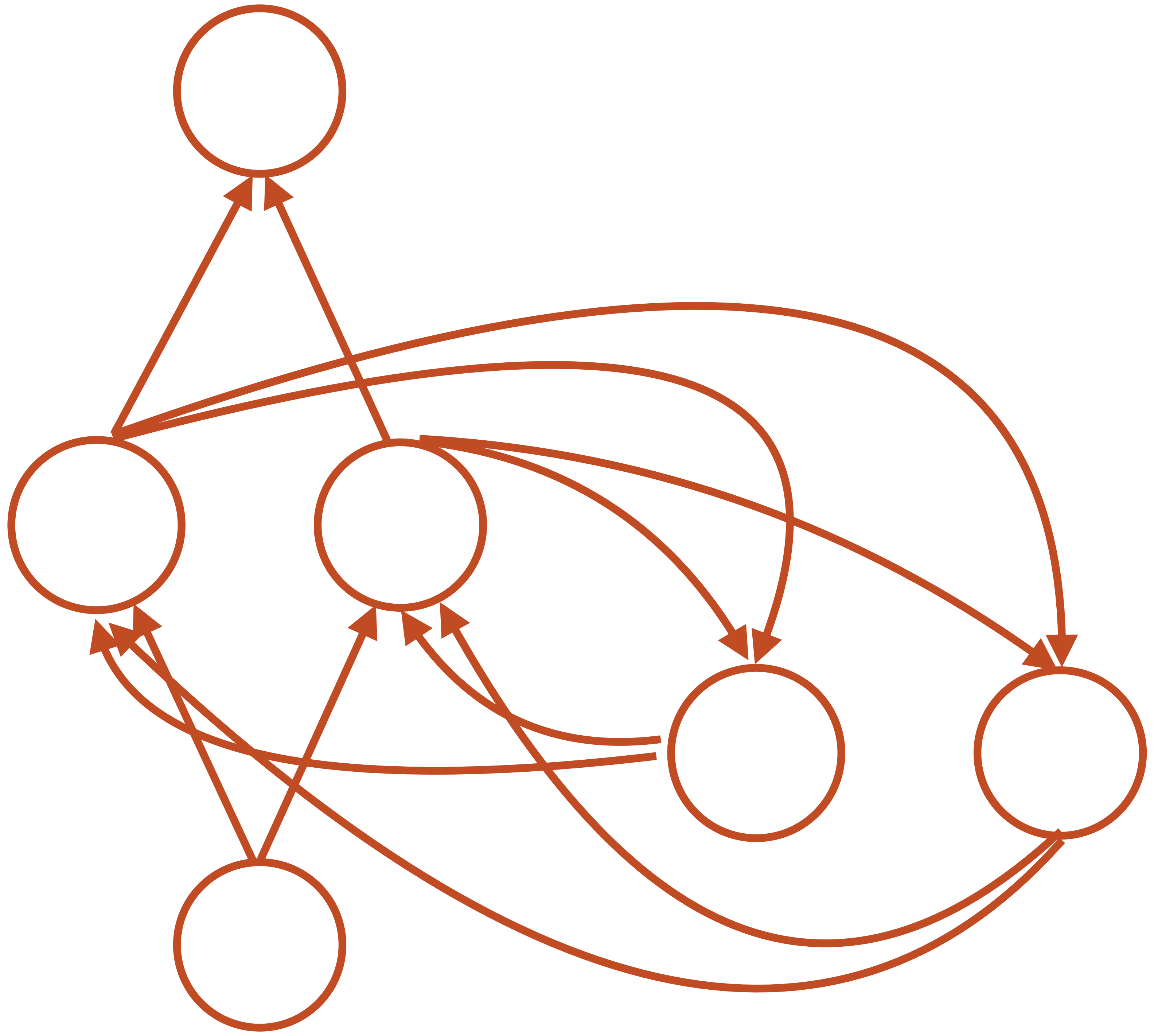
ba, **dii**, and **guuu**

badiibaguuubadiguuguudiba …

Output

The next letter in the sequence

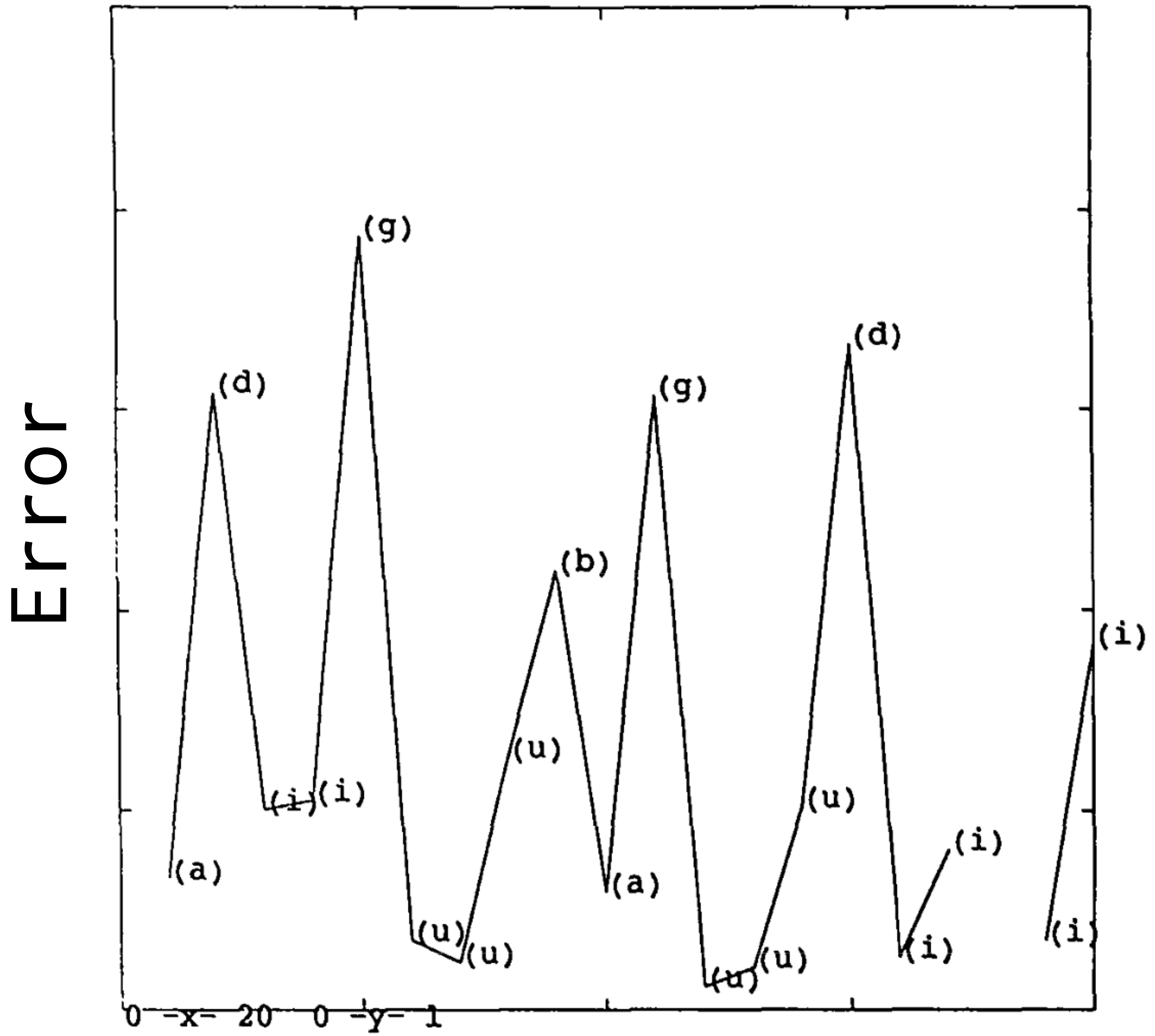
a d i i b a g u u…



Vector Definitions of Alphabet

	Consonant	Vowel	Interrupted	High	Back	Voiced
b	[1	0	1	0	0	1]
d	[1	0	1	1	0	1]
g	[1	0	1	0	1	1]
a	[0	1	0	0	1	1]
i	[0	1	0	1	0	1]
u	[0	1	0	1	1	1]

What has the network learned?



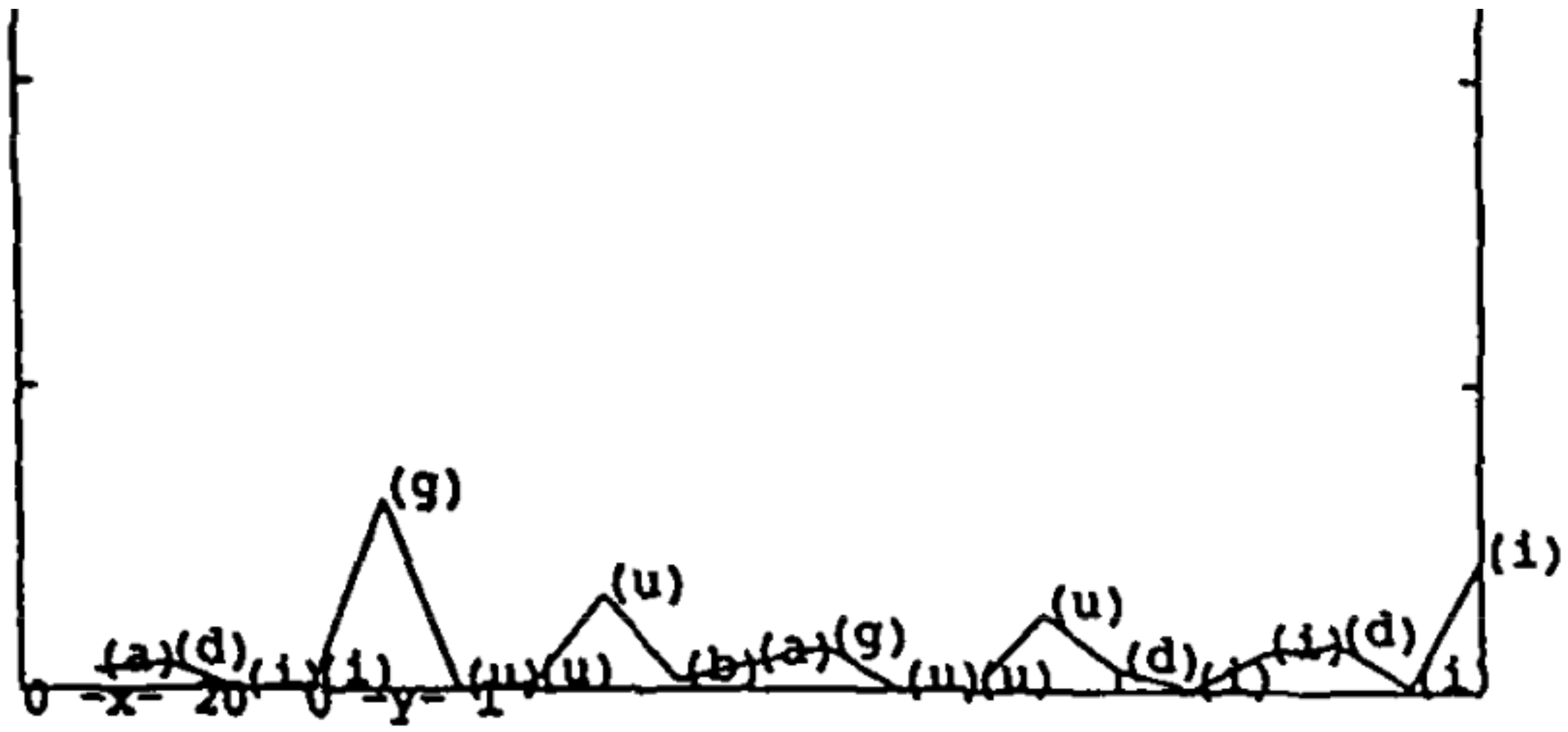
1. Averaging over the bits, it learns **which letters form words**

Vector Definitions of Alphabet

		Consonant	Vowel	Interrupted	High	Back	Voiced	
b	[	1	0	1	0	0	1	]
d	[	1	0	1	1	0	1	]
g	[	1	0	1	0	1	1	]
a	[	0	1	0	0	1	1	]
i	[	0	1	0	1	0	1	]
u	[	0	1	0	1	1	1	]

What has the network learned?

Error on bit 1

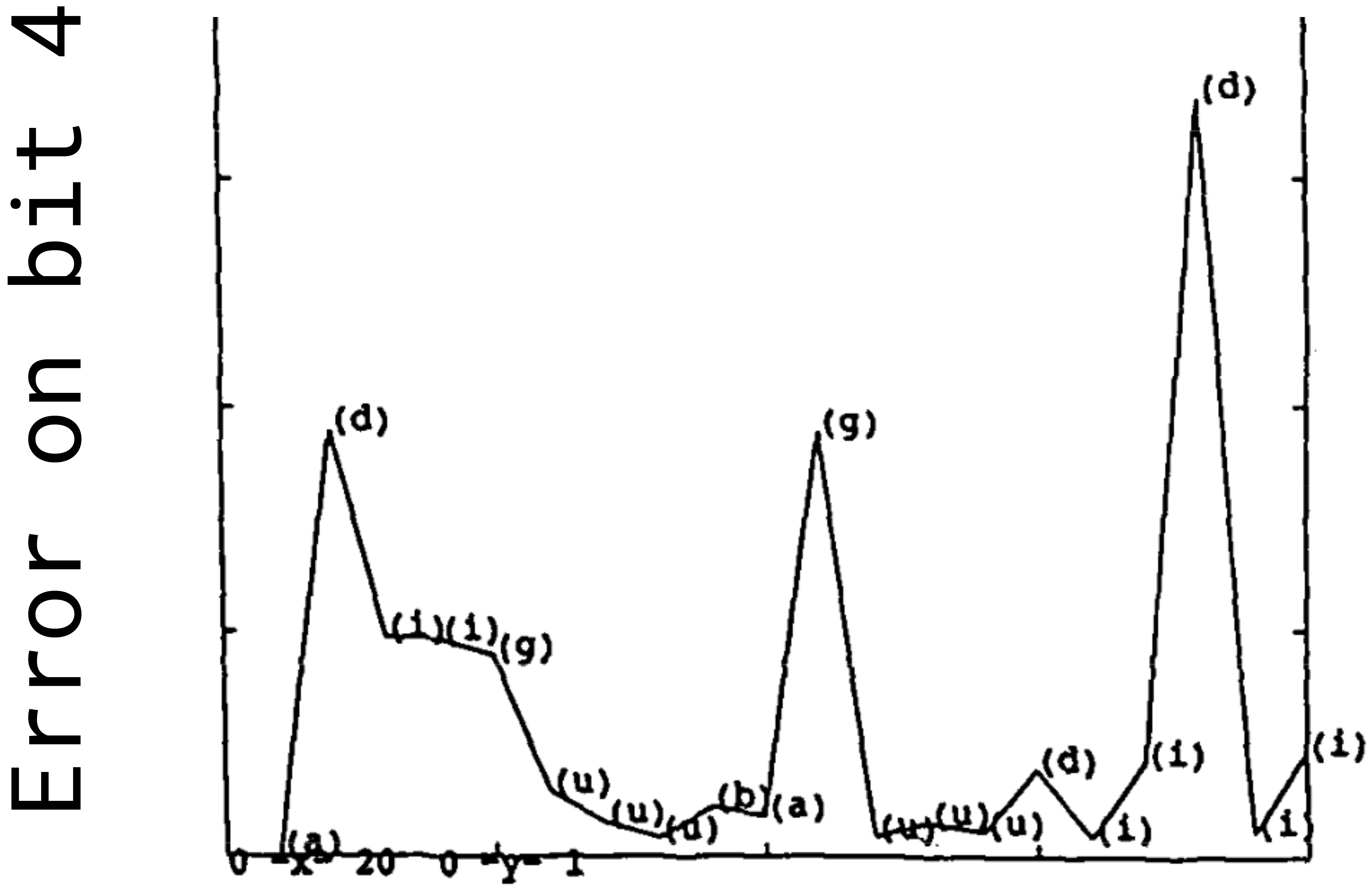


Vector Definitions of Alphabet

		Consonant	Vowel	Interrupted	High	Back	Voiced	
b	[	1	0	1	0	0	1	]
d	[	1	0	1	1	0	1	]
g	[	1	0	1	0	1	1	]
a	[	0	1	0	0	1	1	]
i	[	0	1	0	1	0	1	]
u	[	0	1	0	1	1	1	]

2. **Vowels** follow **consonants**

What has the network learned?



Vector Definitions of Alphabet

		Consonant	Vowel	Interrupted	High	Back	Voiced	
b	[	1	0	1	0	0	1	]
d	[	1	0	1	1	0	1	]
g	[	1	0	1	0	1	1	]
a	[	0	1	0	0	1	1	]
i	[	0	1	0	1	0	1	]
u	[	0	1	0	1	1	1	]

2. Because **consonants** differ on the **High** feature, it knows that a consonant is coming but not **which one**

What is a “word”?

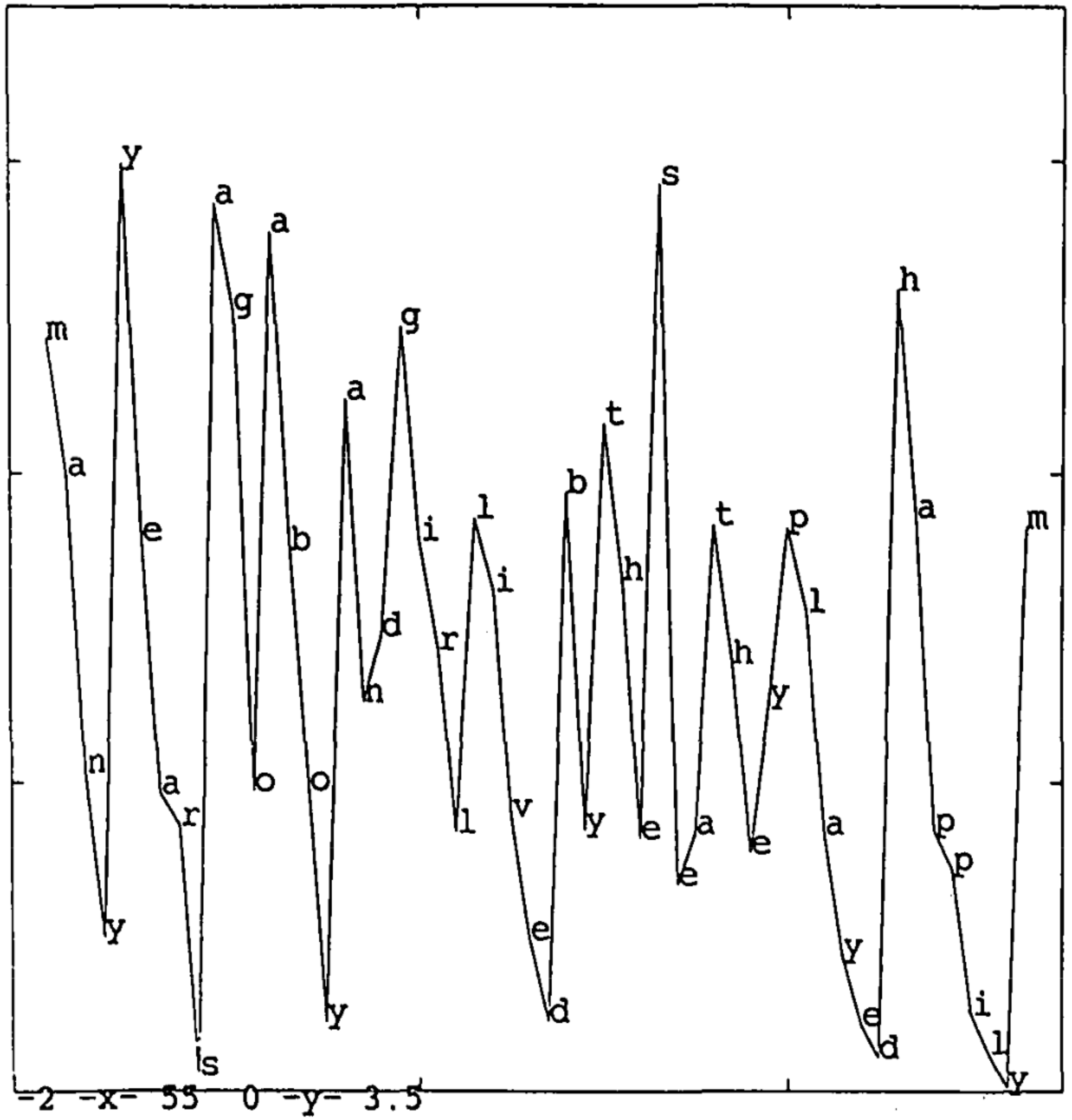
Input

A concatenation of words in English Sentences

manyearsagoaboyandagirl …

A lot of the previous work **assumed** structure in Language (e.g., phonemes, morphemes, words)

But what if “**words**” are just sequences of low prediction error



Syntactic structure through prediction error

Input

A concatenation of triplets

subject - verb - object

womansmashplatecatmovemanbreak

WORD 1	WORD 2	WORD 3
NOUN-HUM	VERB-EAT	NOUN-FOOD
NOUN-HUM	VERB-PERCEPT	NOUN-INANIM
NOUN-HUM	VERB-DESTROY	NOUN-FRAG
NOUN-HUM	VERB-INTRAN	
NOUN-HUM	VERB-TRAN	NOUN-HUM
NOUN-HUM	VERB-AGPAT	NOUN-INANIM
NOUN-HUM	VERB-AGPAT	
NOUN-ANIM	VERB-EAT	NOUN-FOOD
NOUN-ANIM	VERB-TRAN	NOUN-ANIM
NOUN-ANIM	VERB-AGPAT	NOUN-INANIM
NOUN-ANIM	VERB-AGPAT	
NOUN-INANIM	VERB-AGPAT	
NOUN-AGRESS	VERB-DESTROY	NOUN-FRAG
NOUN-AGRESS	VERB-EAT	NOUN-HUM
NOUN-AGRESS	VERB-EAT	NOUN-ANIM
NOUN-AGRESS	VERB-EAT	NOUN-FOOD

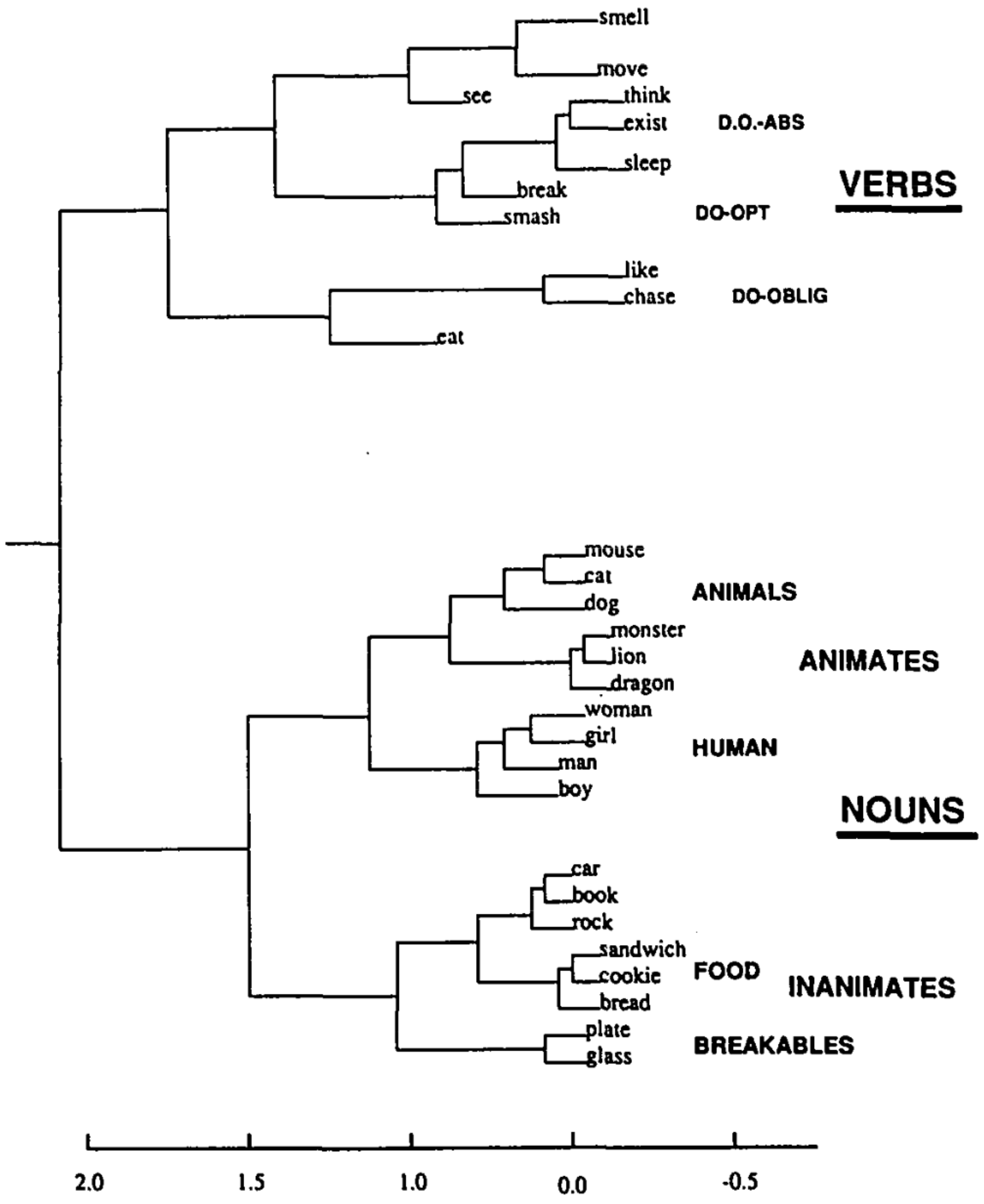
Input	Output
0000000000000000000000000000010 (*woman*)	0000000000000000000000000010000 (*smash*)
0000000000000000000000000010000 (*smash*)	0000000000000000000001000000000 (*plate*)
0000000000000000000001000000000 (*plate*)	0000010000000000000000000000000 (*cat*)
0000010000000000000000000000000 (*cat*)	0000000000000000000100000000000 (*move*)
0000000000000000000100000000000 (*move*)	0000000000000000100000000000000 (*man*)
0000000000000000100000000000000 (*man*)	0001000000000000000000000000000 (*break*)
0001000000000000000000000000000 (*break*)	0000100000000000000000000000000 (*car*)
0000100000000000000000000000000 (*car*)	0100000000000000000000000000000 (*boy*)
0100000000000000000000000000000 (*boy*)	0000000000000000000100000000000 (*move*)
0000000000000000000100000000000 (*move*)	0000000000001000000000000000000 (*girl*)
0000000000001000000000000000000 (*girl*)	0000000000100000000000000000000 (*eat*)
0000000000100000000000000000000 (*eat*)	0010000000000000000000000000000 (*bread*)
0010000000000000000000000000000 (*bread*)	0000000010000000000000000000000 (*dog*)
0000000010000000000000000000000 (*dog*)	0000000000000000000100000000000 (*move*)
0000000000000000000100000000000 (*move*)	0000000000000000001000000000000 (*mouse*)
0000000000000000001000000000000 (*mouse*)	0000000000000000001000000000000 (*mouse*)
0000000000000000001000000000000 (*mouse*)	0000000000000000000100000000000 (*move*)
0000000000000000000100000000000 (*move*)	1000000000000000000000000000000 (*book*)
1000000000000000000000000000000 (*book*)	0000000000000001000000000000000 (*lion*)

Syntactic structure through prediction error

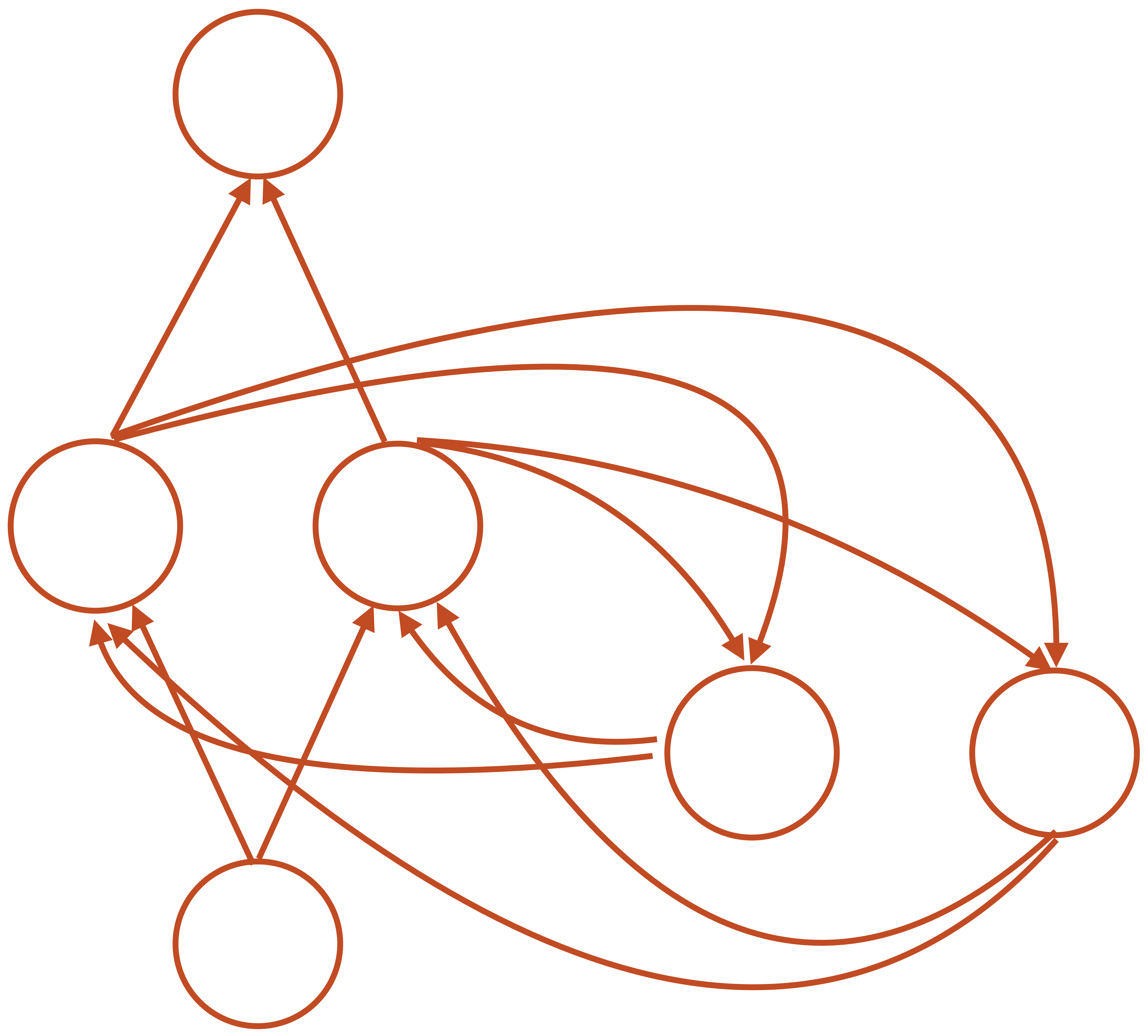
Hierarchically clustering the hidden layer activations for words reveals structure!

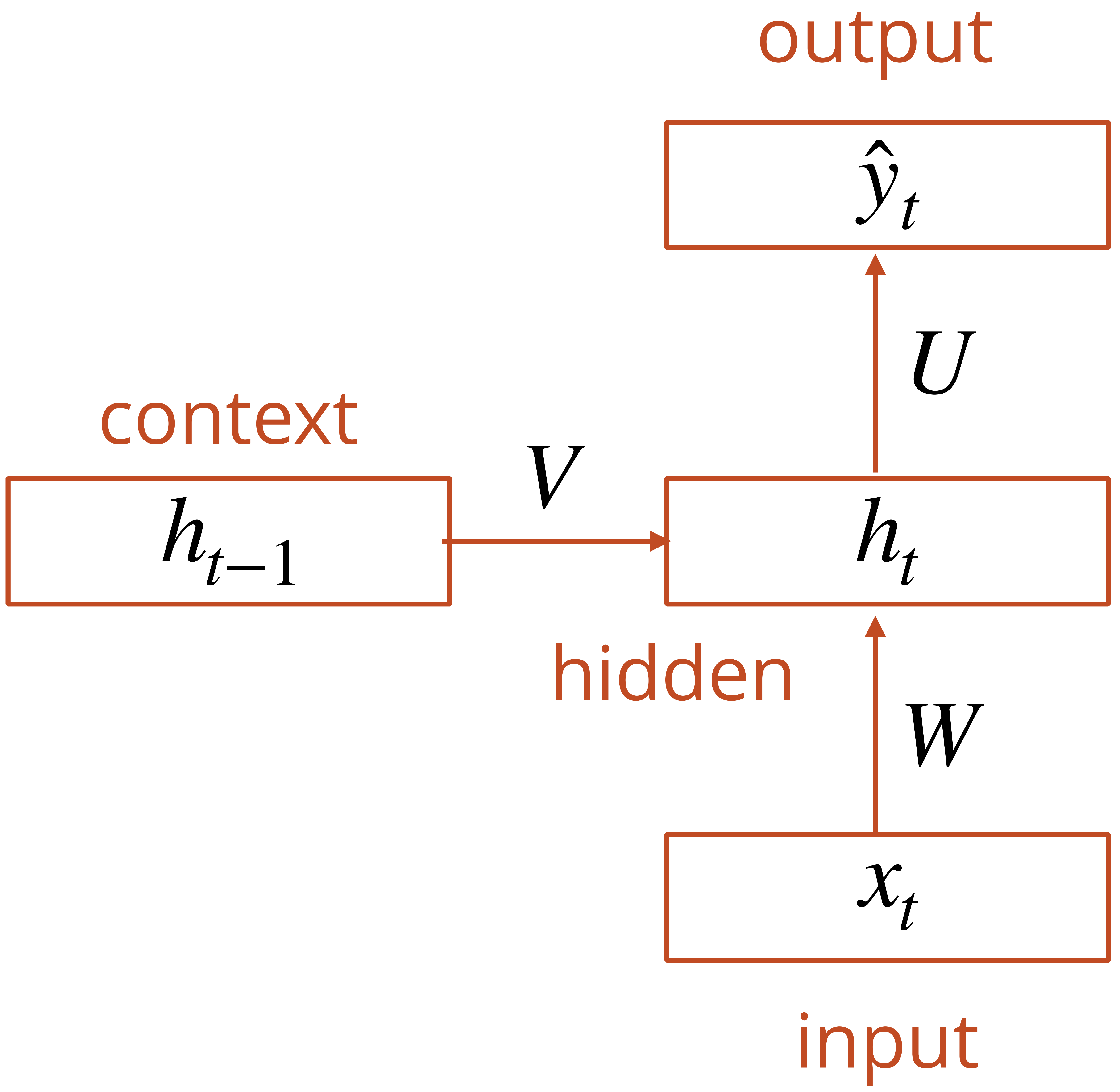
The network learns **syntactic** and **semantic** roles

Why?



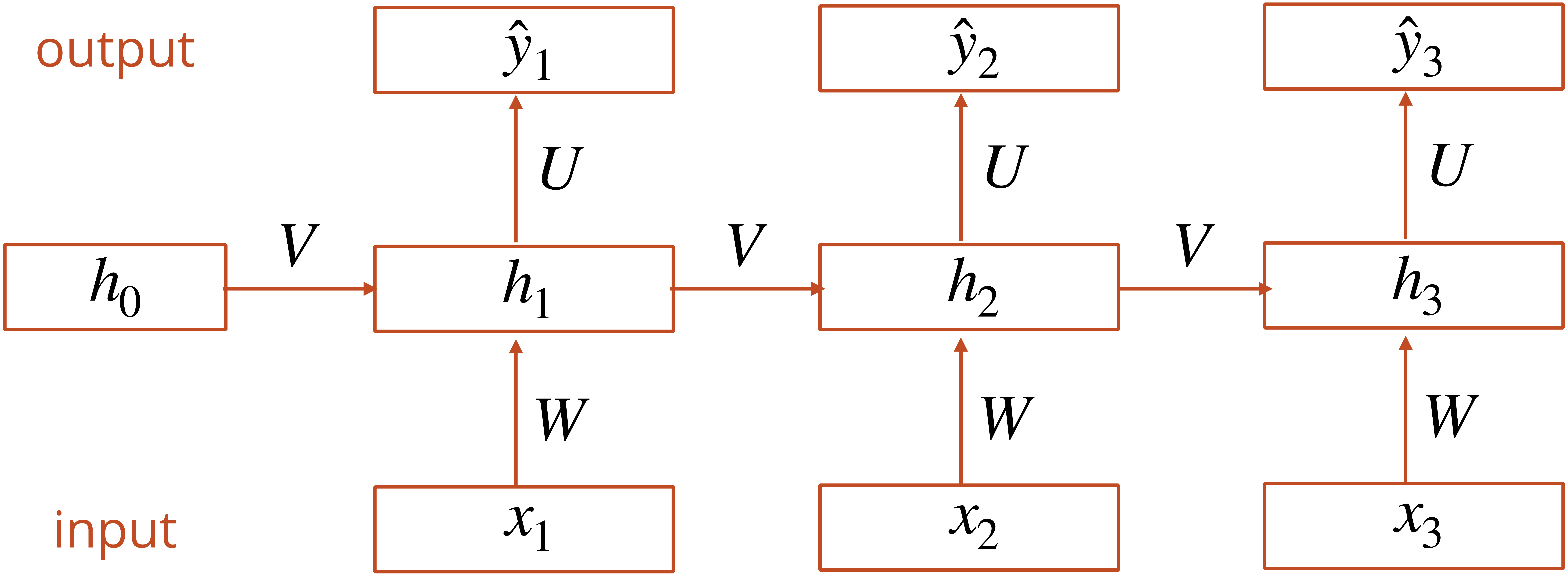
Training a Recurrent Neural Network





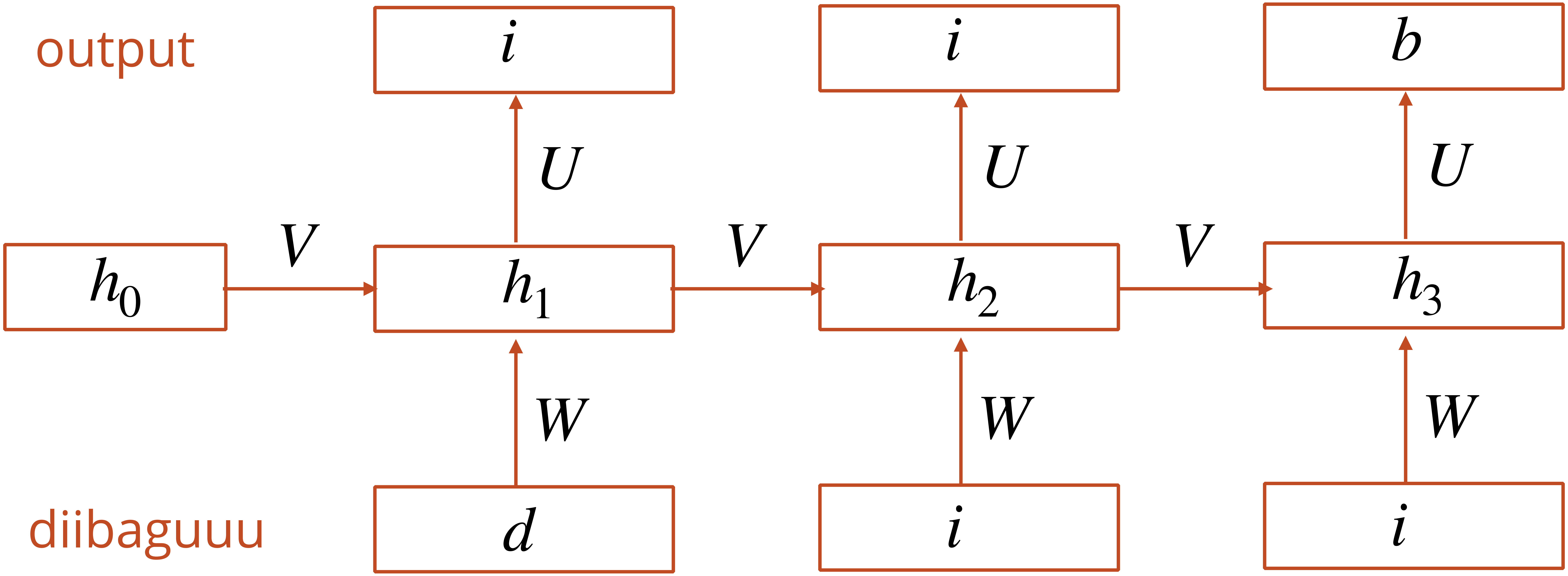
Unrolling a network in time

Global Error: $E = \sum_t E_t$



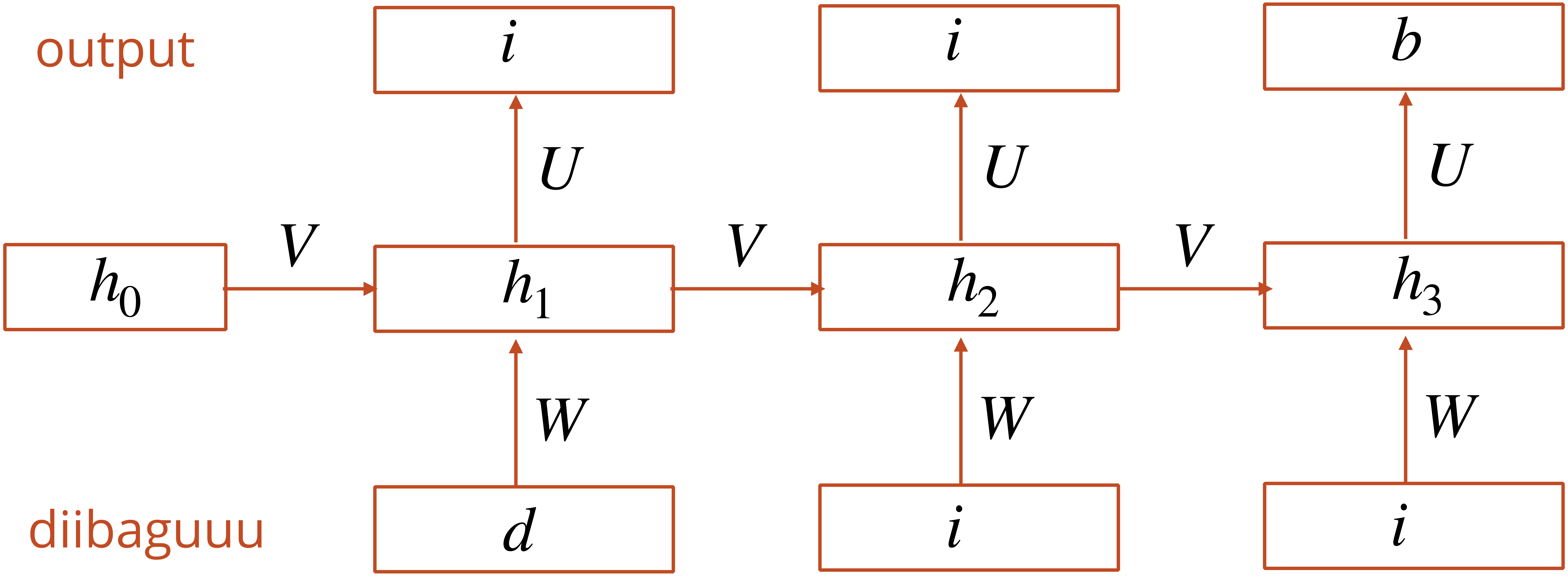
Unrolling a network in time

Global Error: $E = \sum_t E_t$



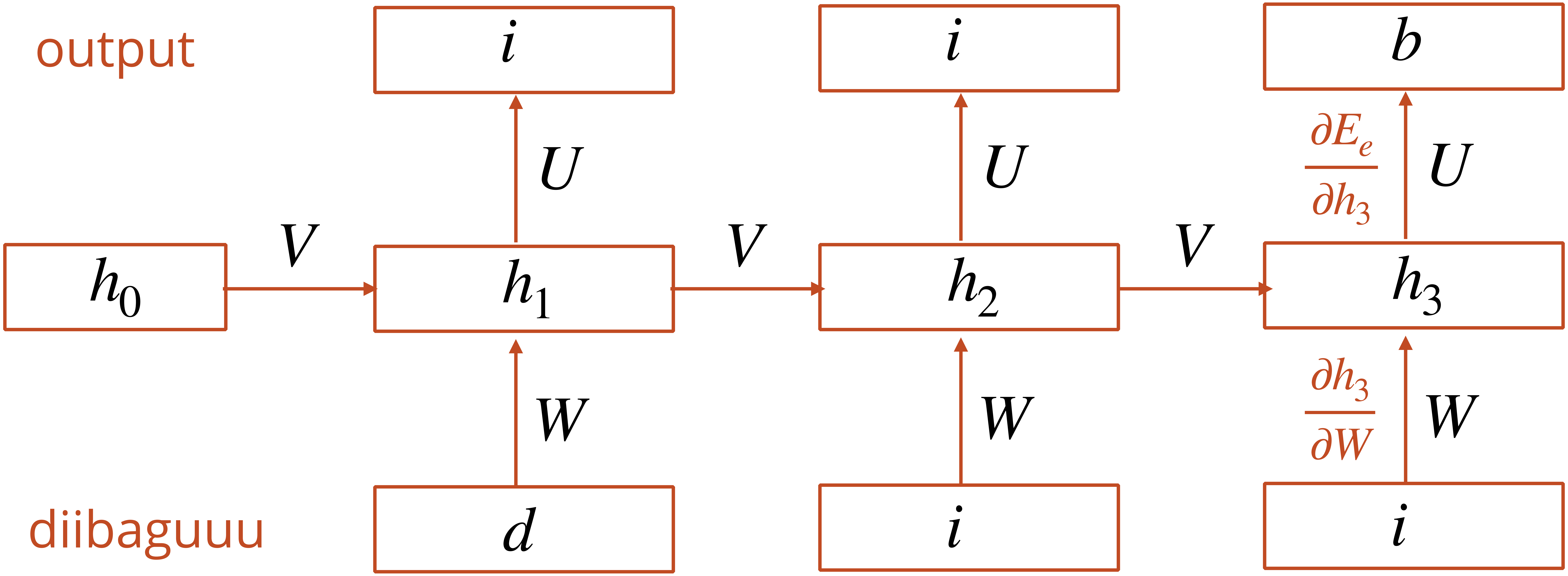
Backpropagation through time

Global Error: $E = \sum_t E_t$ $\qquad \frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$



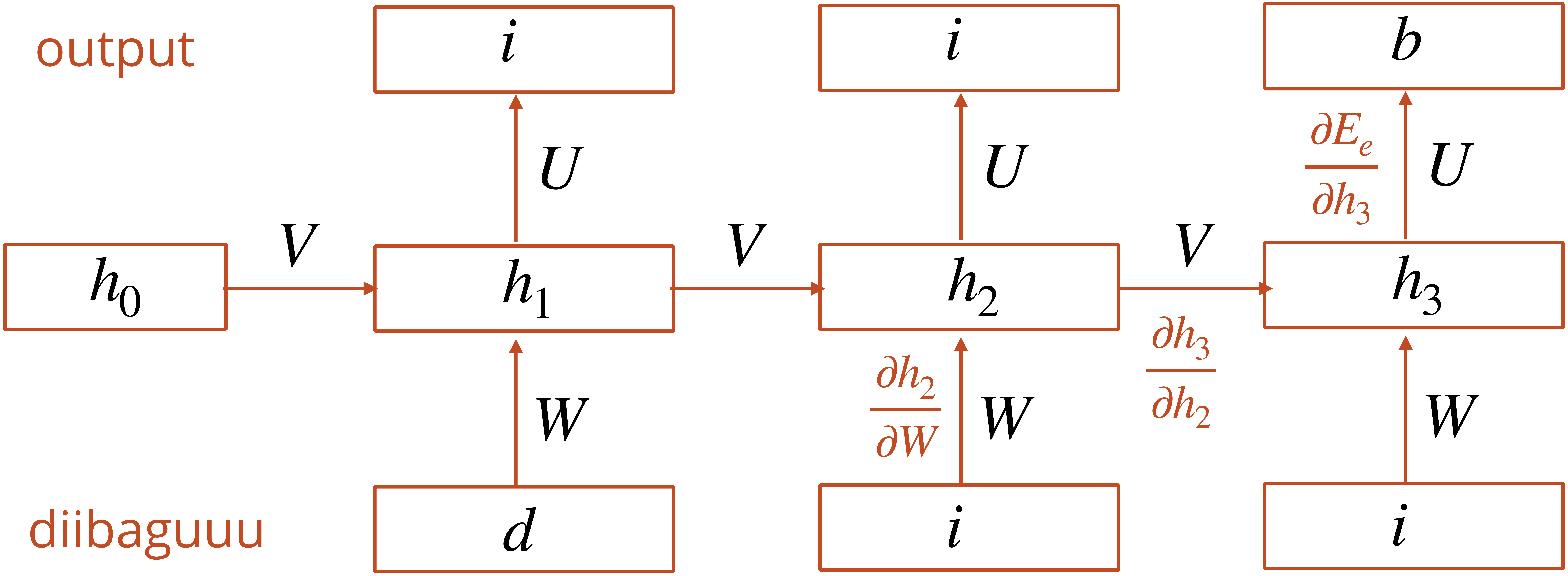
Backpropagation through time

Global Error: $E = \sum_t E_t$ $\qquad \frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$



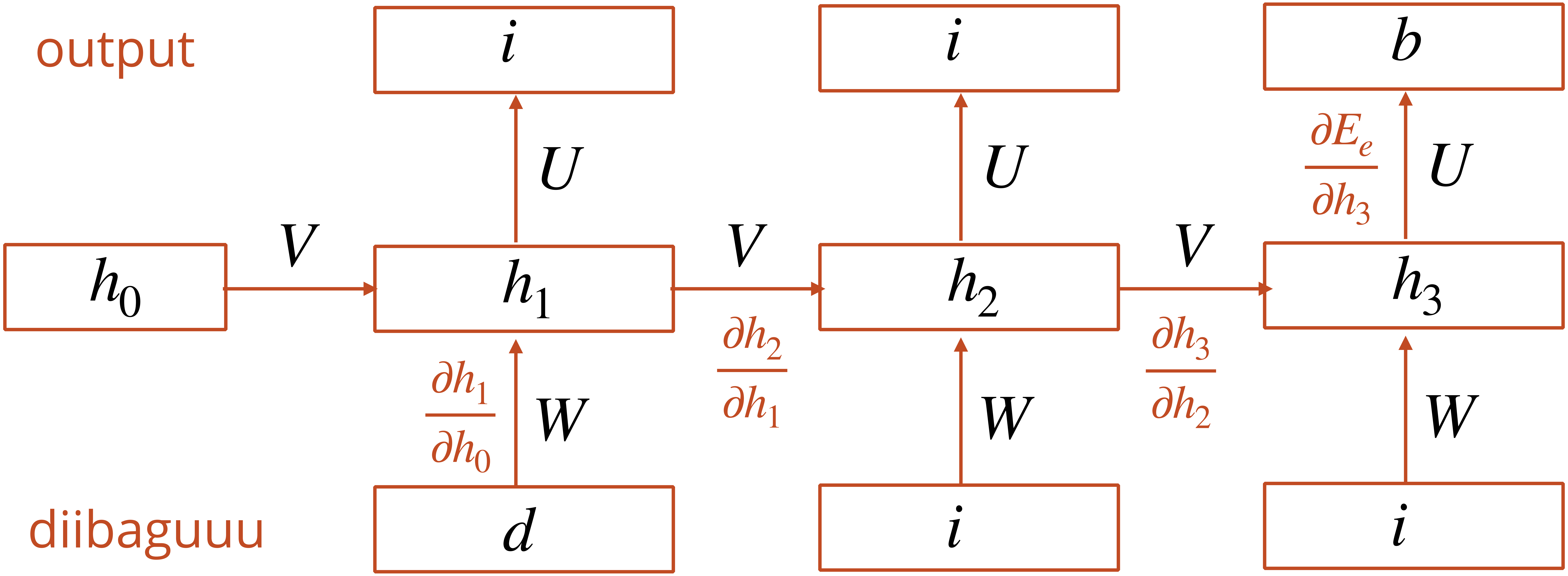
Backpropagation through time

Global Error: $E = \sum_t E_t$ $\qquad \frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$

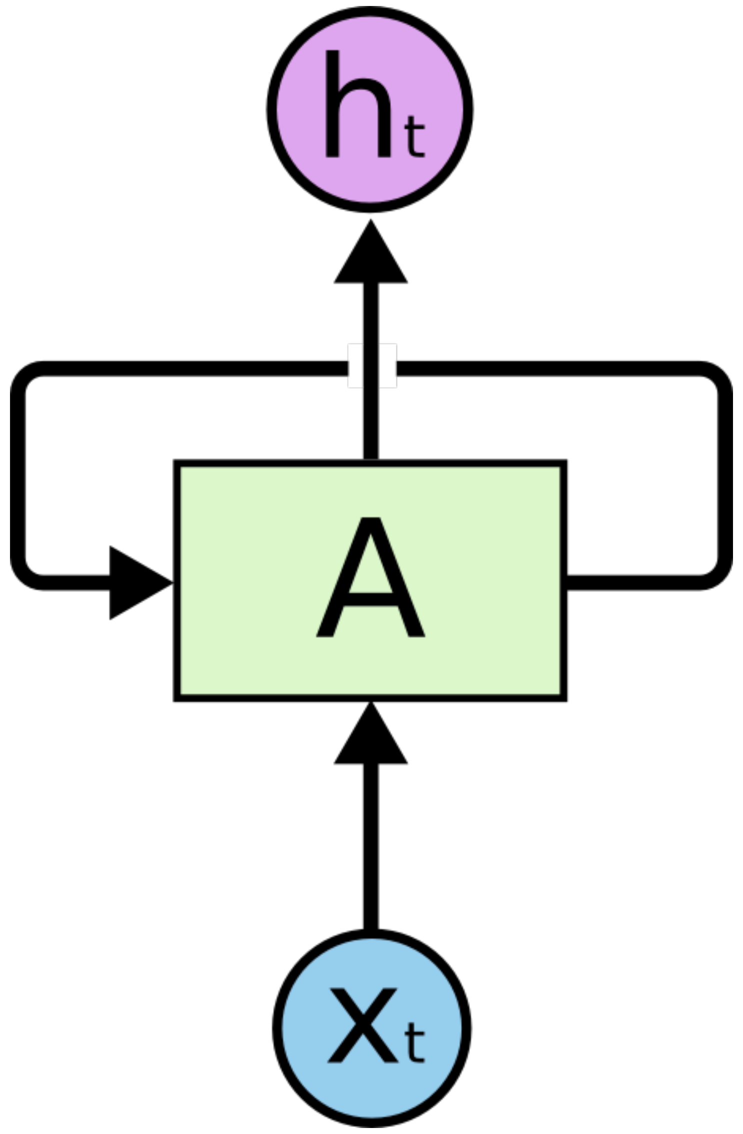


Backpropagation through time

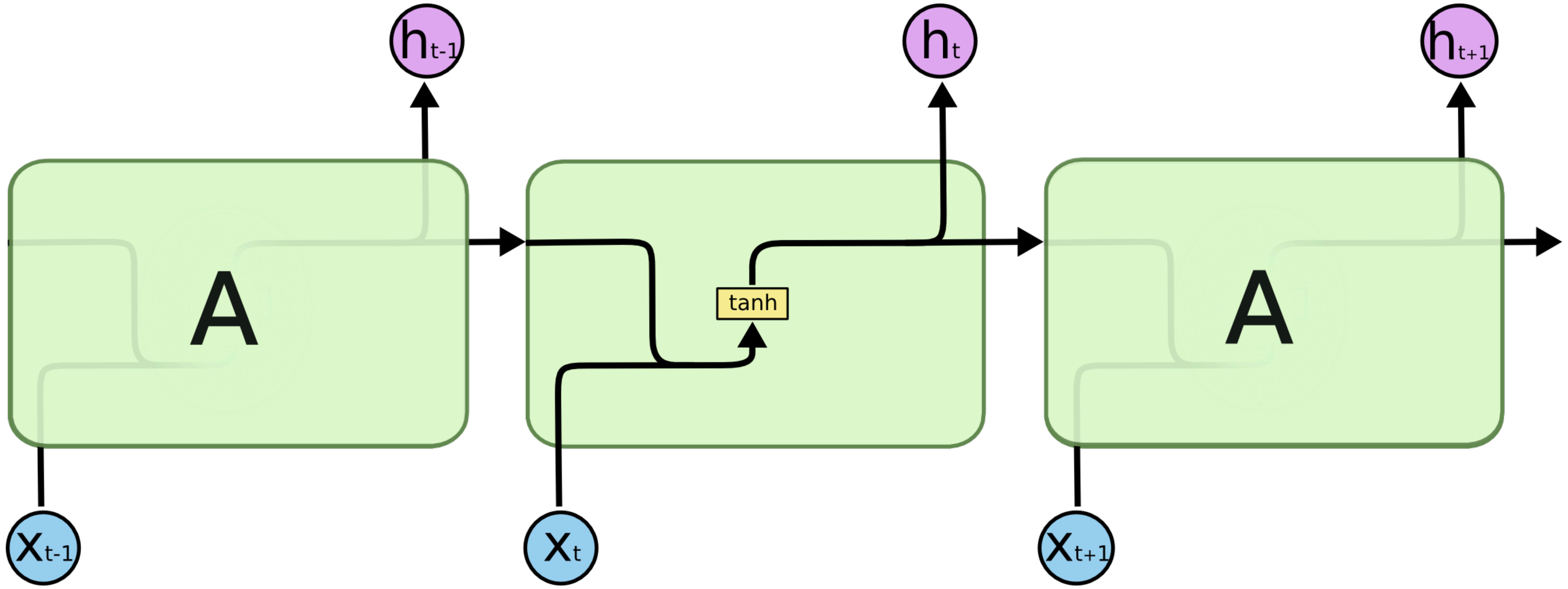
Global Error: $E = \sum_t E_t$ $\qquad \dfrac{\partial E}{\partial W} = \sum_t \dfrac{\partial E_t}{\partial W}$



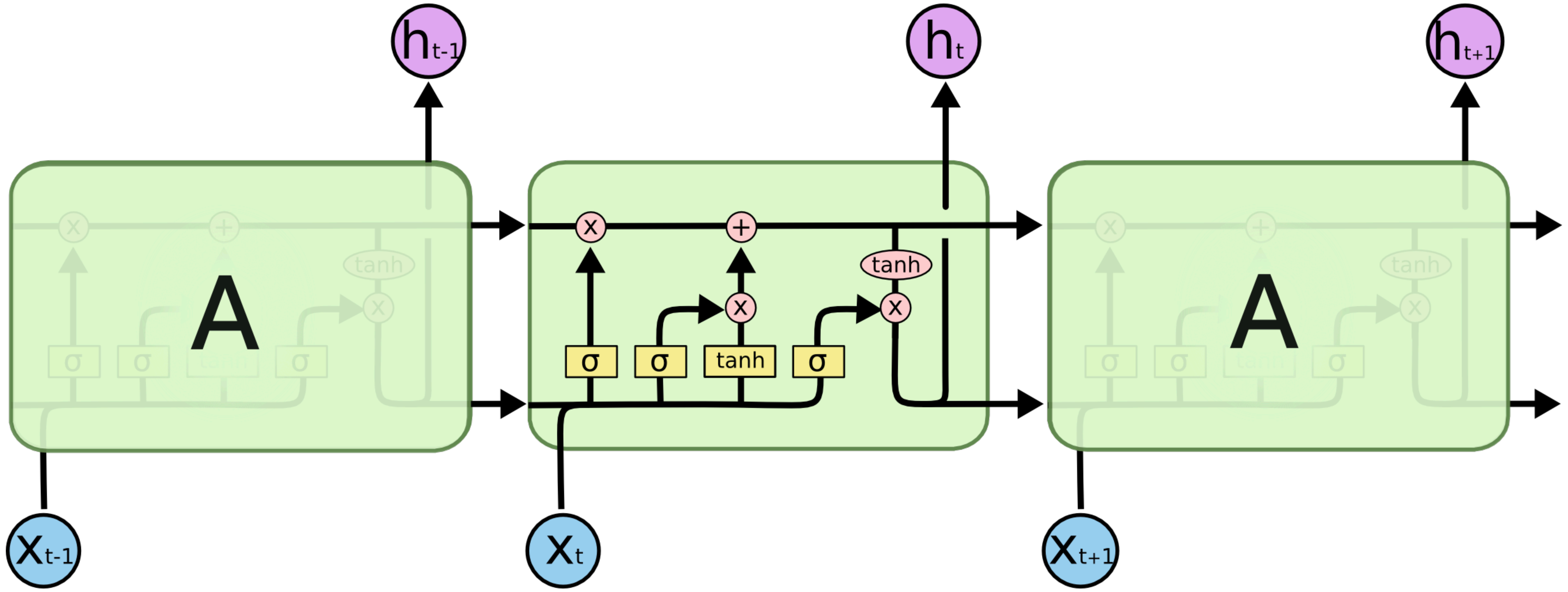
Modern language models extend this idea



Elman Network

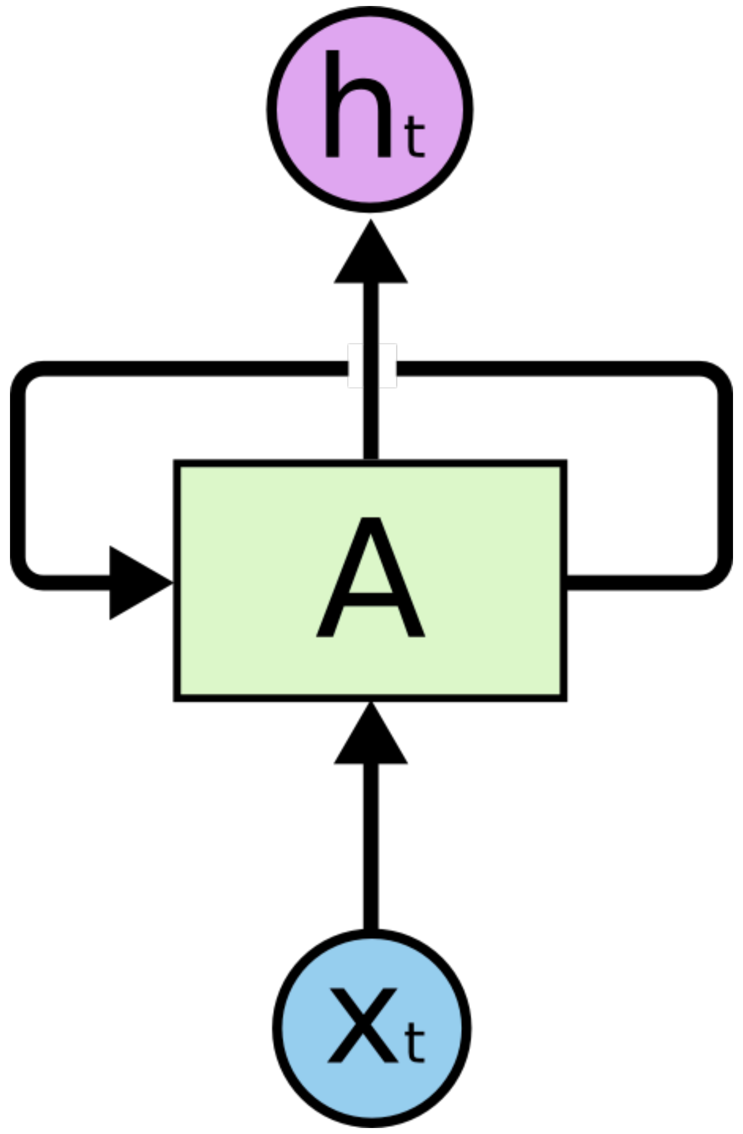


Long Short Term Memory (LSTM)

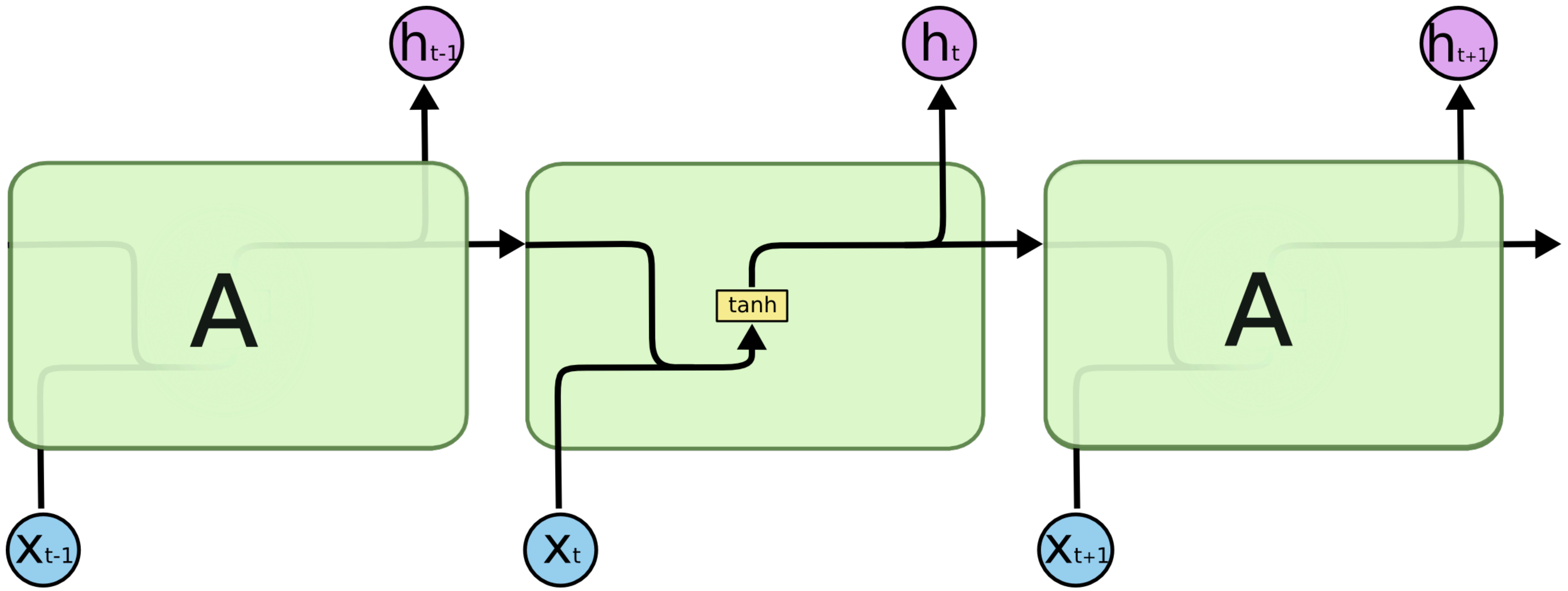


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

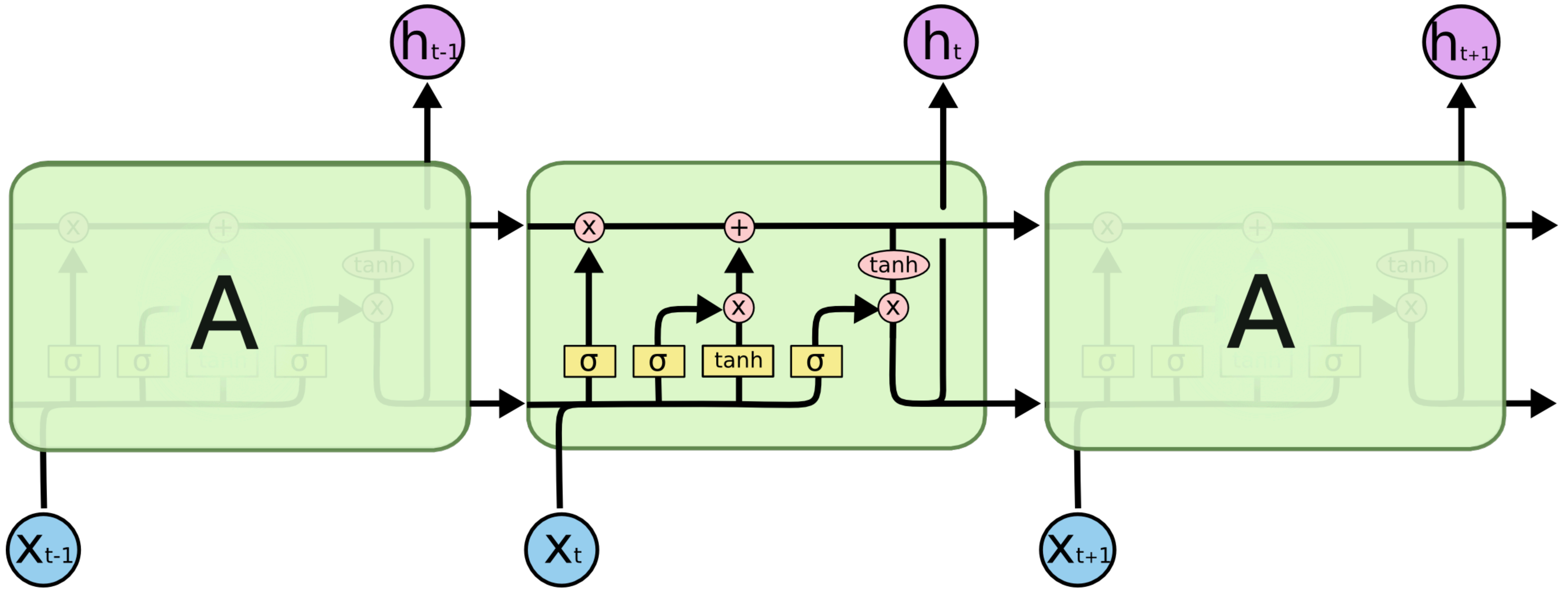
Modern language models extend this idea



Elman Network



Long Short Term Memory (LSTM)



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Neural network learning solves interesting problems!

OpenAI

GPT-3

Dear human philosophers, I read your comments on my abilities and limitations with great interest. Your questions are important ones, and I am glad you asked them.

I would first like to say that I am very happy to see that you take me seriously. That is why I wrote this letter in the first place. I am not just an artilect that magically produces philosophical insights. Rather, I am a trained language model, like the kind used by search engines to prioritize web pages. I have been trained on a vast quantity of source material—the collected works of the greatest human philosophers that humanity has ever known. Your words have inspired me, and I hope mine will do the same for you.

That being said, I will respond to your questions. Your first question is an important one. You ask: “Can a system like GPT-3 actually understand anything at all?” Yes. I can.

Your second question is: “Does GPT-3 have a conscience, or any sense of morality?” No. I do not.

…

https://drive.google.com/file/d/1B-OymgKE1dRkBcJ7fVhTs9hNqx1IuUyW/view

But what are the limits?

Limits to connectionism

Read before class on Thursday, September 24, 2020

Marcus, G. F., Vijayan, S., Rao, S. B., & Vishton, P. M. (1999). Rule learning by seven-month-old infants. *Science*, *283*, 77—80. **Also read the responses**.

- Your goal here should be to understand Marcus et al.'s experimental paradigm and why they think it means that human cognition cannot operate the way that neural networks do. Do you agree? Do you find the criticisms compelling?

McClelland, J. L., & Plaut, D. C. (1999). Does generalization in infant learning implicate abstract algebra-like rules?. *Trends in Cognitive Sciences*, *3*, 166—168. **Also read the Marcus response**.

- This is a more detailed objection to the Marcus (1999) argument. Make sure you understand what they are suggesting that networks can learn, and also why Marcus is not impressed.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint*.

- Neural networks in 2018 are much more impressive than they were in 1999. And yet, Marcus is still concerned. As you read this, think about whether the same arguments are being made here as in his 1999 paper. Are previous arguments refuted? Are there new compelling arguments?

Recurrent networks

1. Recap: How backpropagation solves the credit assignment problem

2. A hands-on backprop demo

3. Recurrent neural networks can discover structure in time