Machine Learning with Python

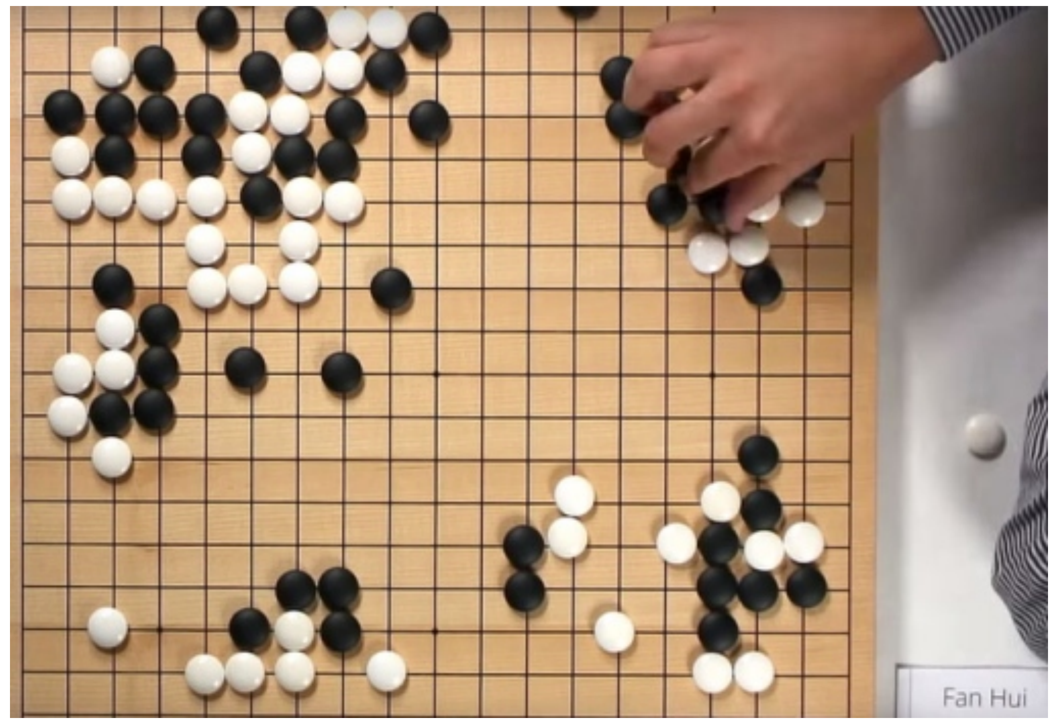
MTH786U/P 2023/24

Lecture 6: Regression with Neural Networks

Nicola Perra, Queen Mary University of London (QMUL)

n.perra@qmul.ac.uk

Deep learning is everywhere

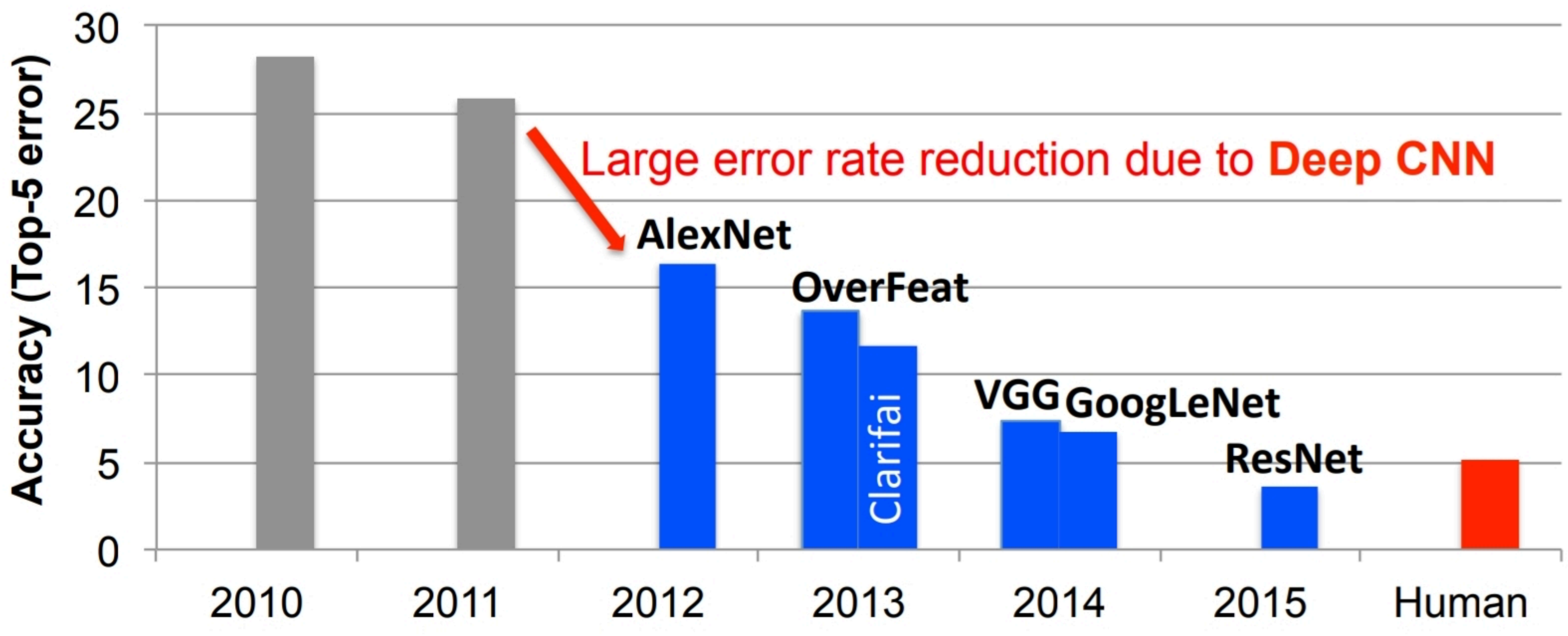


how-old.net

How-Old.net

Deep learning pioneers Bengio, Hinton, LeCun have been awarded the Turing Award 2018

Deep learning is everywhere



*

Many great successes in

- Image classification
- Speech recognition
- Image captioning
- Natural language processing
- Text generation
- and many many more

*O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015.

Neural networks

Image source 1

Image source 2

Image source

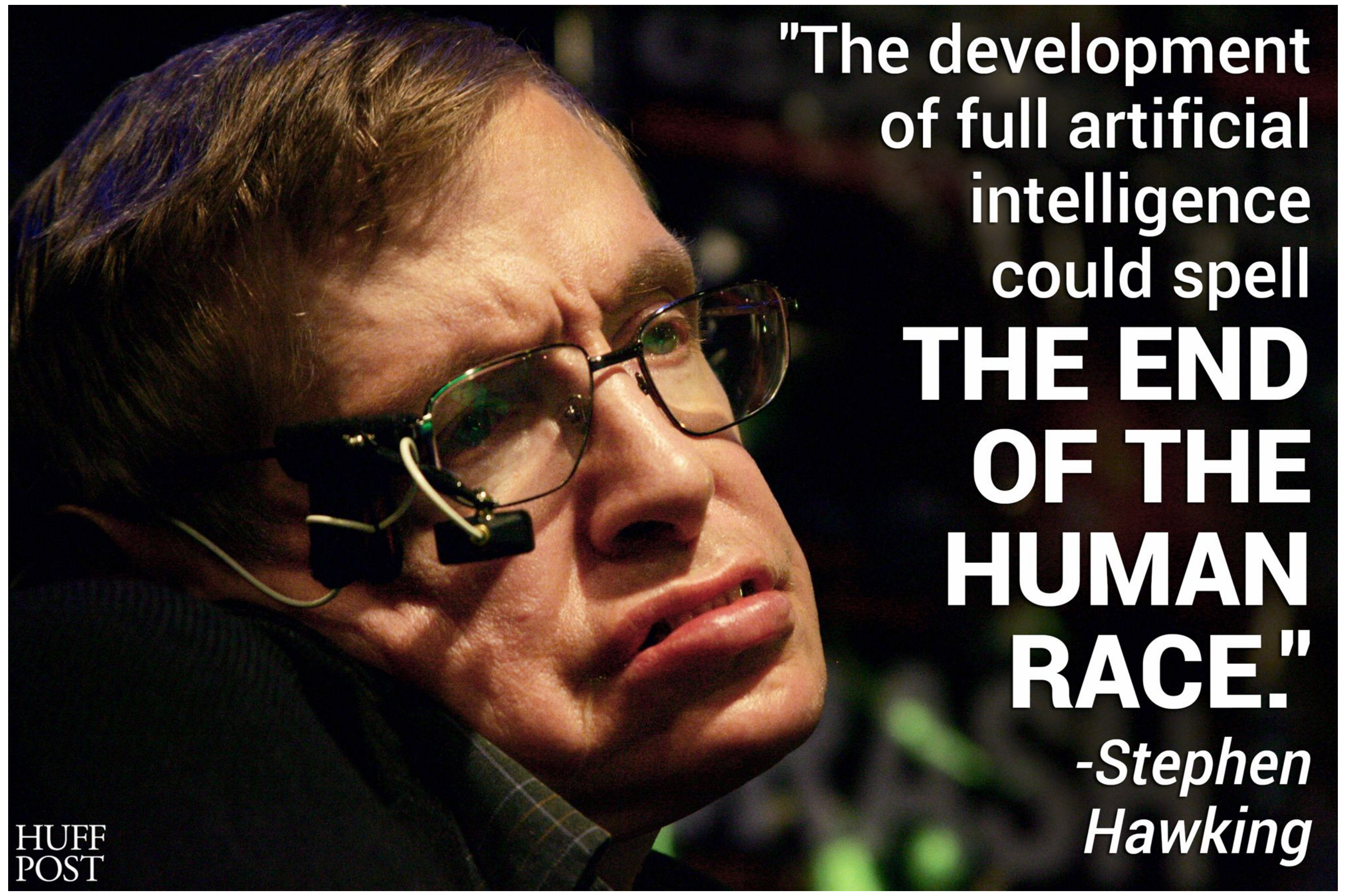


Image source

Neural networks

Source

Companies sponsoring NeurIPS

Source

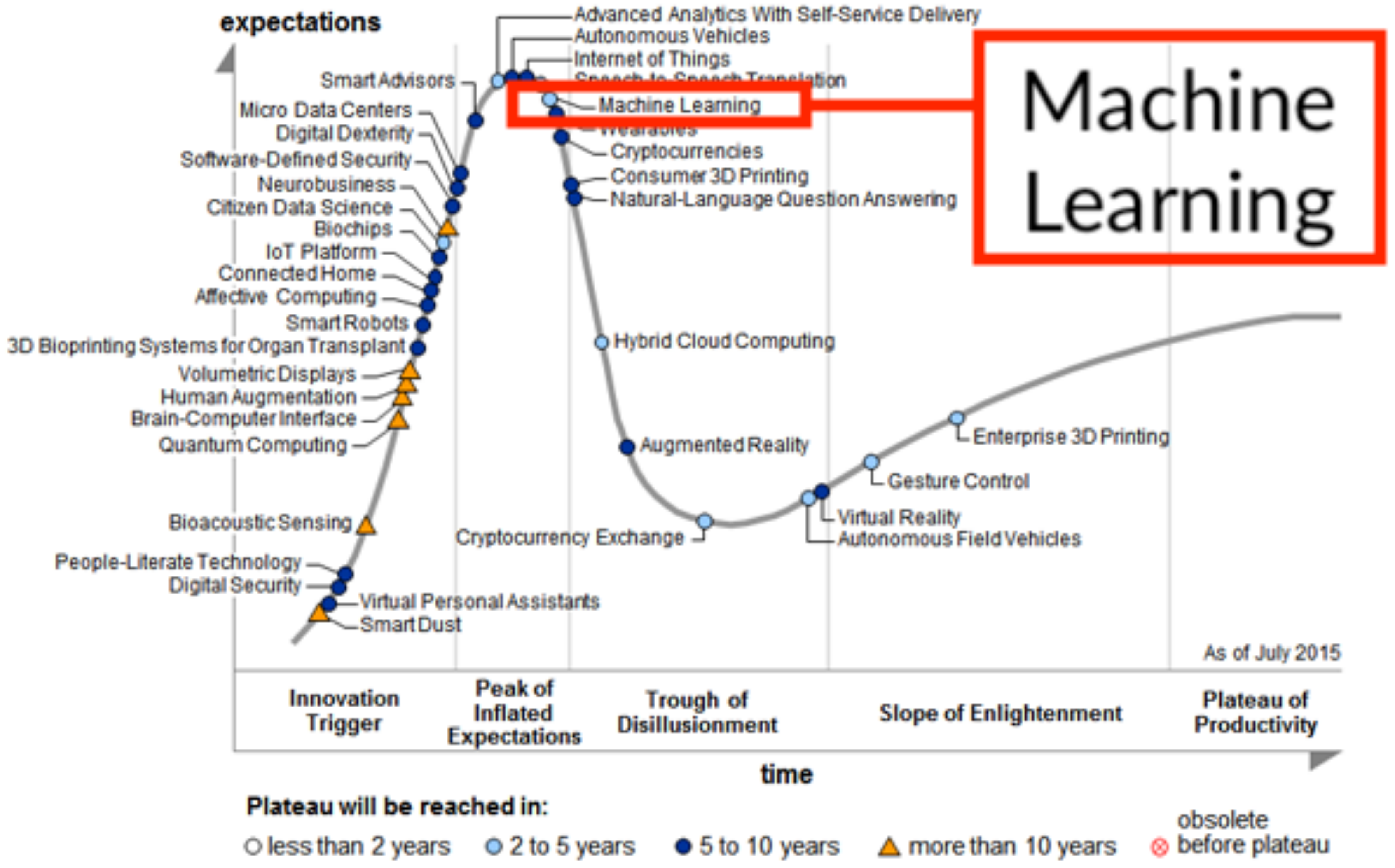
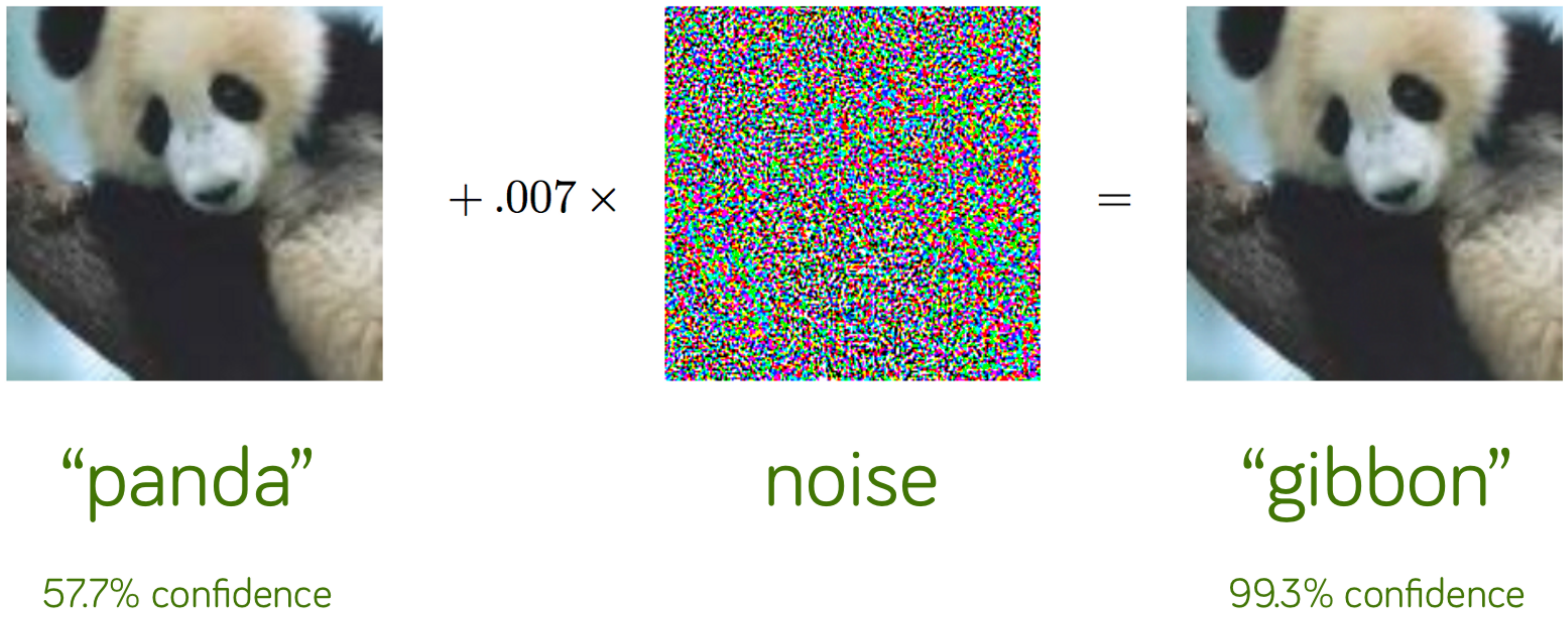


Image source

Deep learning is everywhere

However, many successful strategies are black-boxes with little to no guarantees

Adversarial examples in image classification:



This is why we need mathematics to better understand deep learning

Explaining and Harnessing Adversarial Examples, Goodfellow et al, ICLR 2015.

Deep learning is everywhere

Large Language Models and generative AI are super popular

Deep learning is everywhere

Large Language Models and generative AI are super popular

ChatGPT made a real splash and opened this world to everyone

NI can you write a brief explanation for students in a MSc in Data Analytics of chatGPT?

Certainly! Here's a brief explanation of ChatGPT for students in an MSc in Data Analytics:

ChatGPT: An Overview for Data Analytics Students

ChatGPT is an advanced artificial intelligence (AI) model that falls under the category of Natural Language Processing (NLP). Developed by OpenAI, it is designed to understand and generate human-like text based on the input it receives. Here's how it works:

1. **Training Data**: ChatGPT has been trained on an extensive dataset containing a wide range of text from the internet. This data includes books, articles, websites, and more. This extensive training allows ChatGPT to understand and generate text on a vast array of topics.
2. **Transformers Architecture**: ChatGPT uses a deep learning architecture called a "transformer." This architecture is particularly well-suited for NLP tasks. It allows the model to process and generate text in a context-aware manner, which means it can understand and generate text that takes into account the surrounding context.
3. **Chat-Based Interaction**: Users can interact with ChatGPT by providing text-based input. They can ask questions, seek explanations, or request information on various topics. ChatGPT responds with text-based answers or information.
4. **Language Generation**: ChatGPT is not just about answering questions but also generating human-like text. This makes it versatile for tasks like generating content, summarizing text, and even engaging in creative writing.
5. **Limitations**: It's essential to be aware of ChatGPT's limitations. While it's a powerful tool, it may generate incorrect or biased information, so users need to critically evaluate the responses. It can also be sensitive to how questions are framed, and its responses m

Regenerate

can you make it much shorter?

Absolutely! Here's a concise explanation:

ChatGPT in Data Analytics

ChatGPT is an AI model designed for natural language understanding and generation. It's a valuable tool in data analytics, helping analysts with data queries and reports. However, it's important to verify the accuracy of its responses and be aware of potential biases.

NI

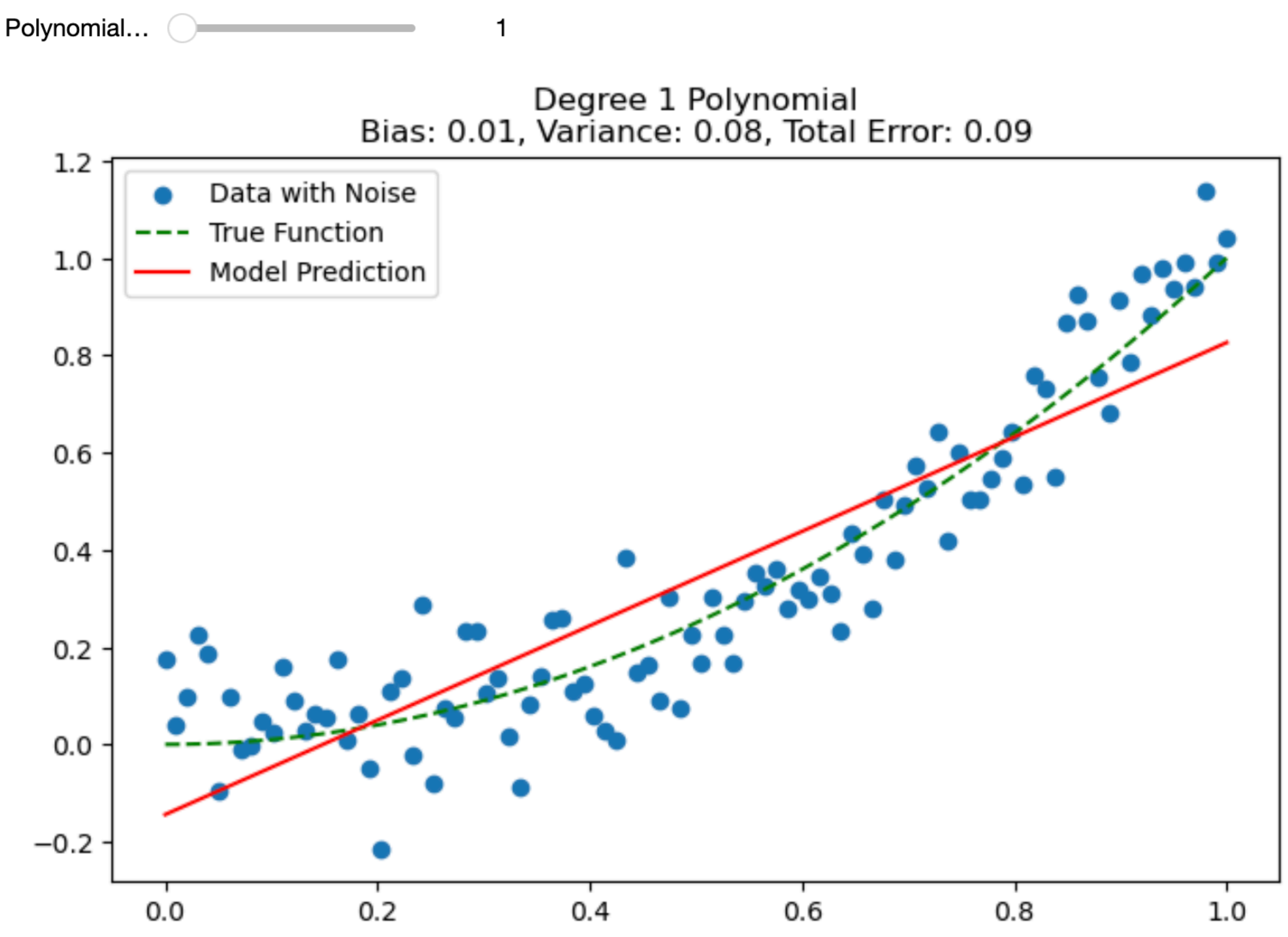
can you write a python widget to show the bias-variance decomposition?

Creating an interactive widget to visualize the bias-variance decomposition in Python can be done using libraries like Matplotlib and ipywidgets. You would need to have these libraries installed. Here's a basic example of how you can create a simple widget for visualizing the bias-variance tradeoff:

python

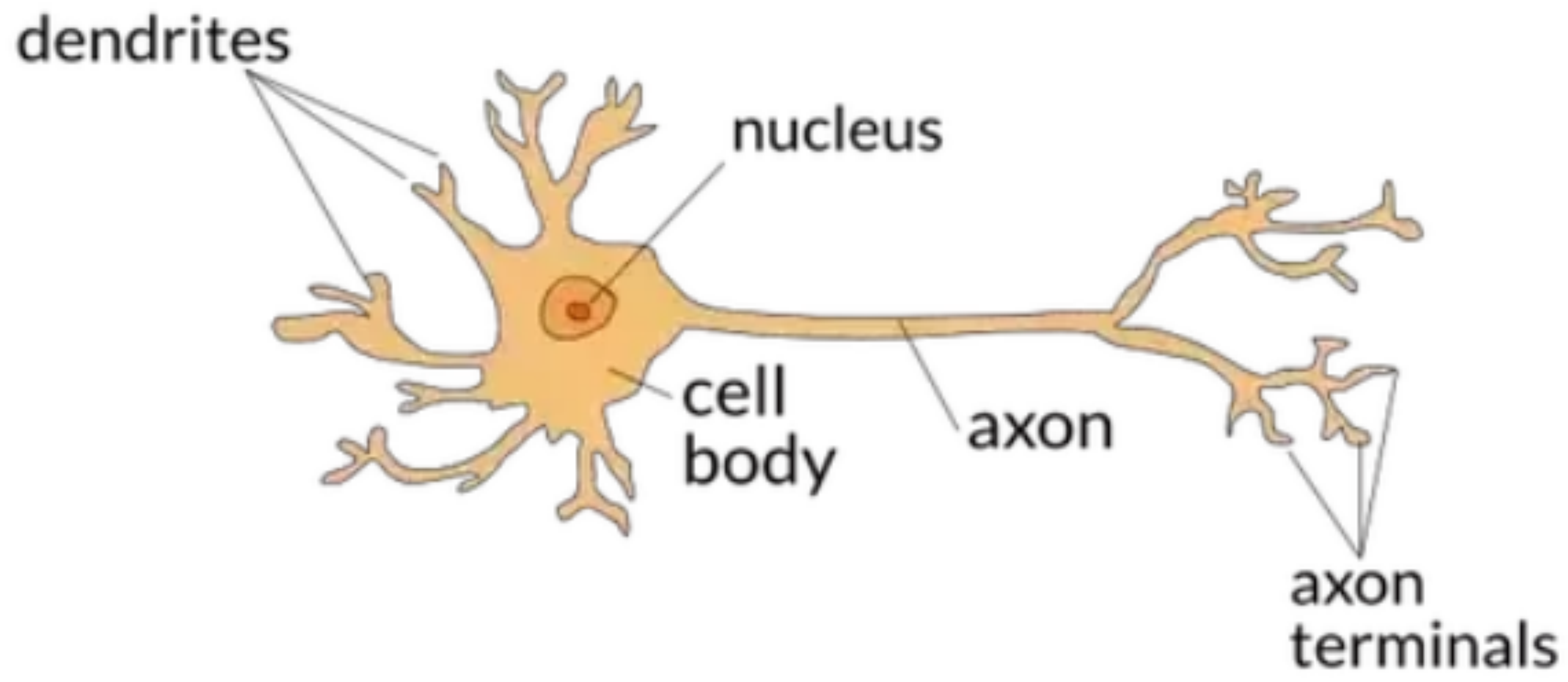
Copy code

```python
import numpy as np
import matplotlib.pyplot as plt
import ipywidgets as widgets
from ipywidgets import interactive

# Generate some sample data
np.random.seed(0)
X = np.linspace(0, 1, 100)
y_true = X**2
y_noise = np.random.normal(0, 0.1, 100)
y = y_true + y_noise

# Function to plot bias-variance tradeoff
def plot_bias_variance(degree):
    # Fit a polynomial of the selected degree
    p = np.polyfit(X, y, degree)
    y_pred = np.polyval(p, X)
```



Polynomial…
1
Degree 1 Polynomial
Bias: 0.01, Variance: 0.08, Total Error: 0.09
Data with Noise
True Function
Model Prediction
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0

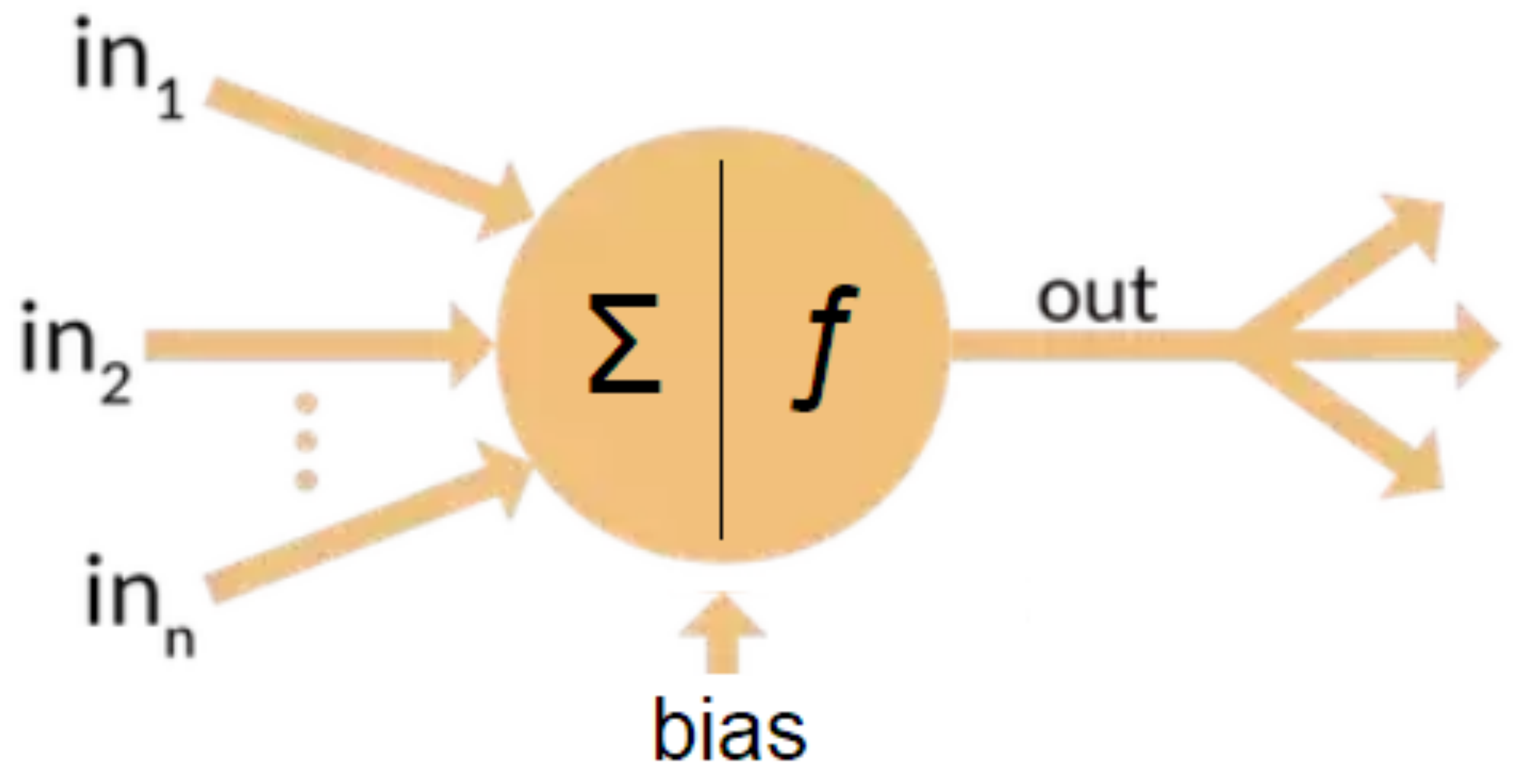
Deep learning motivation



Biological neuron has dendrites to receive signals, a cell body to process them, and an axon to send signals out to other neurons

Source

Deep learning motivation



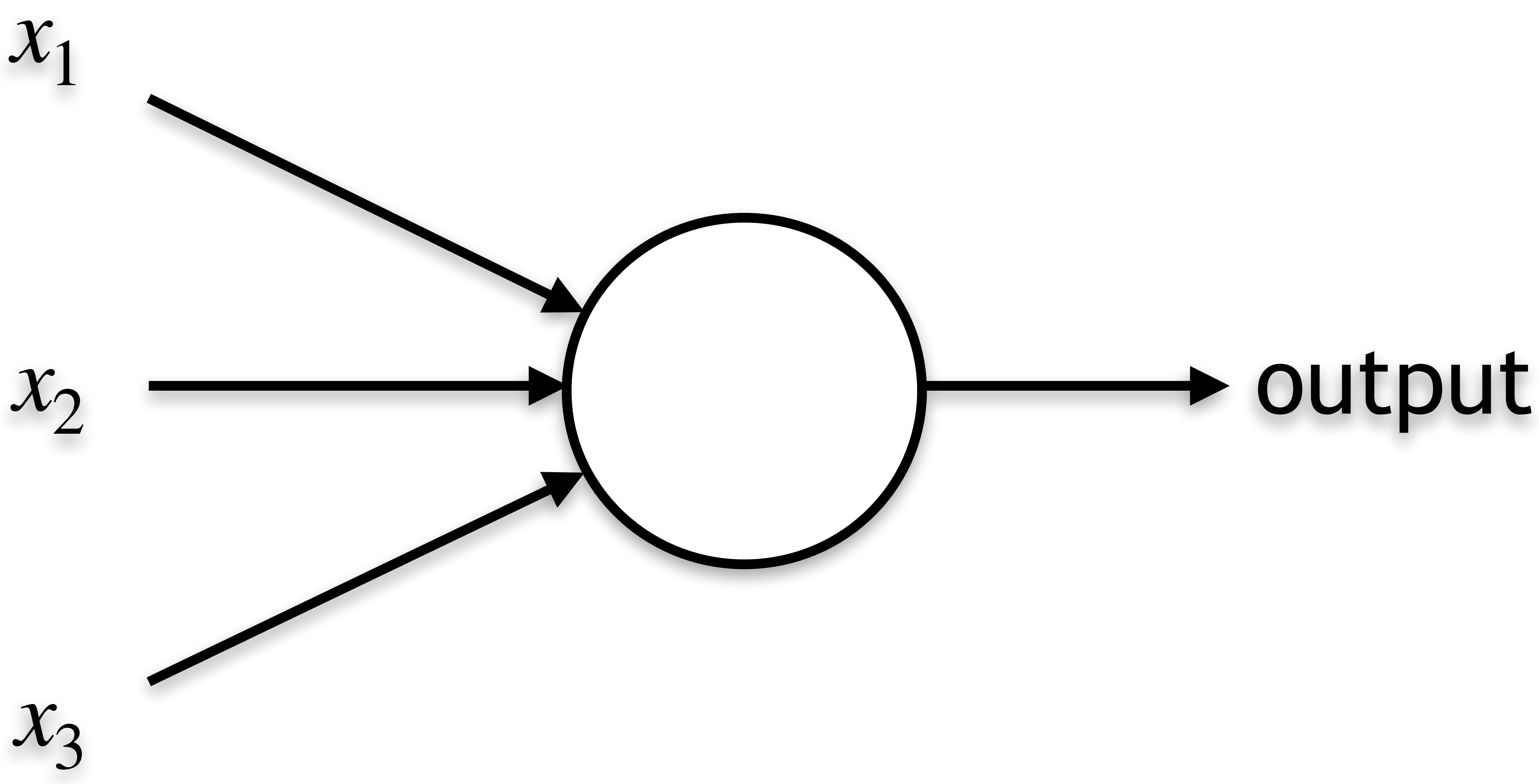
McCulloch, Pitts, Rosenblatt, 1950s - 60s

Artificial neuron has a number of input channels, a processing stage, and one output that can fan out to multiple other artificial neurons.

Source

Perceptron

McCulloch, Pitts, Rosenblatt,
1950s - 60s



Binary inputs x_1, x_2, x_3

Weights w_1, w_2, w_3

Rosenblatt:

$$\text{output} = \begin{cases} 0 & \text{if } \sum_{j=1}^{3} w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_{j=1}^{3} w_j x_j > \text{threshold} \end{cases}$$

Perceptron

Example*:

Download festival is approaching & we really like Tool (the band)

Perceptron

Example*:

Download festival is approaching & we really like Tool (the band)

Now we try to decide whether to go or not

Perceptron

Example*:

Download festival is approaching & we really like Tool (the band)

Now we try to decide whether to go or not

We make our decision by weighing up three factors:

*Inspired by 'Neural networks and deep learning'

Perceptron

Example*:

Download festival is approaching & we really like Tool (the band)

Now we try to decide whether to go or not

We make our decision by weighing up three factors:

- Is the weather good?

*Inspired by 'Neural networks and deep learning'

Perceptron

Example*:

Download festival is approaching & we really like Tool (the band)

Now we try to decide whether to go or not

We make our decision by weighing up three factors:

- Is the weather good?
- Does our partner want to accompany us?

*Inspired by 'Neural networks and deep learning'

Perceptron

Example*:

Download festival is approaching & we really like Tool (the band)

Now we try to decide whether to go or not

We make our decision by weighing up three factors:

- Is the weather good?
- Does our partner want to accompany us?
- Is the festival near public transit (we don’t own a car)?

*Inspired by 'Neural networks and deep learning'

Perceptron

We make our decision by weighing up three factors:

- Is the weather good? $x_1 \in \{0,1\}$
- Does our partner want to accompany us? $x_2 \in \{0,1\}$
- Is the festival near public transit (we don't own a car)? $x_3 \in \{0,1\}$

$$(1 = \text{yes}, \quad 0 = \text{no})$$

Perceptron

We make our decision by weighing up three factors:

- Is the weather good? $x_1 \in \{0,1\}$
- Does our partner want to accompany us? $x_2 \in \{0,1\}$
- Is the festival near public transit (we don't own a car)? $x_3 \in \{0,1\}$

$(1 =$ yes, $\quad 0 =$ no$)$

Suppose we like Tool so much that we would see them without our partner, but we really loathe bad weather

Perceptron

We make our decision by weighing up three factors:

- Is the weather good? $x_1 \in \{0,1\}$
- Does our partner want to accompany us? $x_2 \in \{0,1\}$
- Is the festival near public transit (we don't own a car)? $x_3 \in \{0,1\}$

$$(1 = \text{yes}, \quad 0 = \text{no})$$

We can model decision processes like this with perceptrons:

$$w_1 = 6, w_2 = 2, w_3 = 2$$

$$\text{threshold} = 5$$

Perceptron

We make our decision by weighing up three factors:

- Is the weather good? $x_1 \in \{0,1\}$
- Does our partner want to accompany us? $x_2 \in \{0,1\}$
- Is the festival near public transit (we don't own a car)? $x_3 \in \{0,1\}$

$$\text{output} = \begin{cases} 0 & \text{if } 6x_1 + 2x_2 + 2x_3 \leq 5 \\ 1 & \text{if } 6x_1 + 2x_2 + 2x_3 > 5 \end{cases}$$

Perceptron

We make our decision by weighing up three factors:

- Is the weather good? $x_1 \in \{0,1\}$
- Does our partner want to accompany us? $x_2 \in \{0,1\}$
- Is the festival near public transit (we don't own a car)? $x_3 \in \{0,1\}$

$$\text{output} = \begin{cases} 0 & \text{if } 6x_1 + 2x_2 + 2x_3 \leq 5 \\ 1 & \text{if } 6x_1 + 2x_2 + 2x_3 > 5 \end{cases}$$

Example: the weather is bad, our partner wants to accompany us and the festival is near public transit

$$x_1 = 0, x_2 = 1, x_3 = 1 \quad \Rightarrow \quad \text{output} = 0$$

Perceptron

We make our decision by weighing up three factors:

- Is the weather good? $x_1 \in \{0,1\}$
- Does our partner want to accompany us? $x_2 \in \{0,1\}$
- Is the festival near public transit (we don't own a car)? $x_3 \in \{0,1\}$

$$\text{output} = \begin{cases} 0 & \text{if } 6x_1 + 2x_2 + 2x_3 \leq 5 \\ 1 & \text{if } 6x_1 + 2x_2 + 2x_3 > 5 \end{cases}$$

Example: the weather is good, our partner does not want to accompany us and the festival is not near public transit

$$x_1 = 1, x_2 = 0, x_3 = 0 \quad \Rightarrow \quad \text{output} = 1$$

Perceptron

A perceptron with n inputs can be modelled mathematically as

$$f(x_1, \ldots, x_n) \;:= \begin{cases} 0 & \text{if } \sum_{j=1}^{n} w_j x_j \leq -b \\ 1 & \text{if } \sum_{j=1}^{n} w_j x_j > -b \end{cases}$$

Perceptron

A perceptron with n inputs can be modelled mathematically as

$$f(x_1, \ldots, x_n) \quad := \begin{cases} 0 & \text{if } \sum_{j=1}^{n} w_j x_j \leq -b \\ 1 & \text{if } \sum_{j=1}^{n} w_j x_j > -b \end{cases}$$

$-b =$ threshold

Perceptron

A perceptron with n inputs can be modelled mathematically as

$$f(x_1, \ldots, x_n) = \sigma(w^\top x + b) \; := \begin{cases} 0 & \text{if } \sum_{j=1}^{n} w_j x_j \le -b \\ 1 & \text{if } \sum_{j=1}^{n} w_j x_j > -b \end{cases}$$

$-b =$ threshold

©intmath

Perceptron

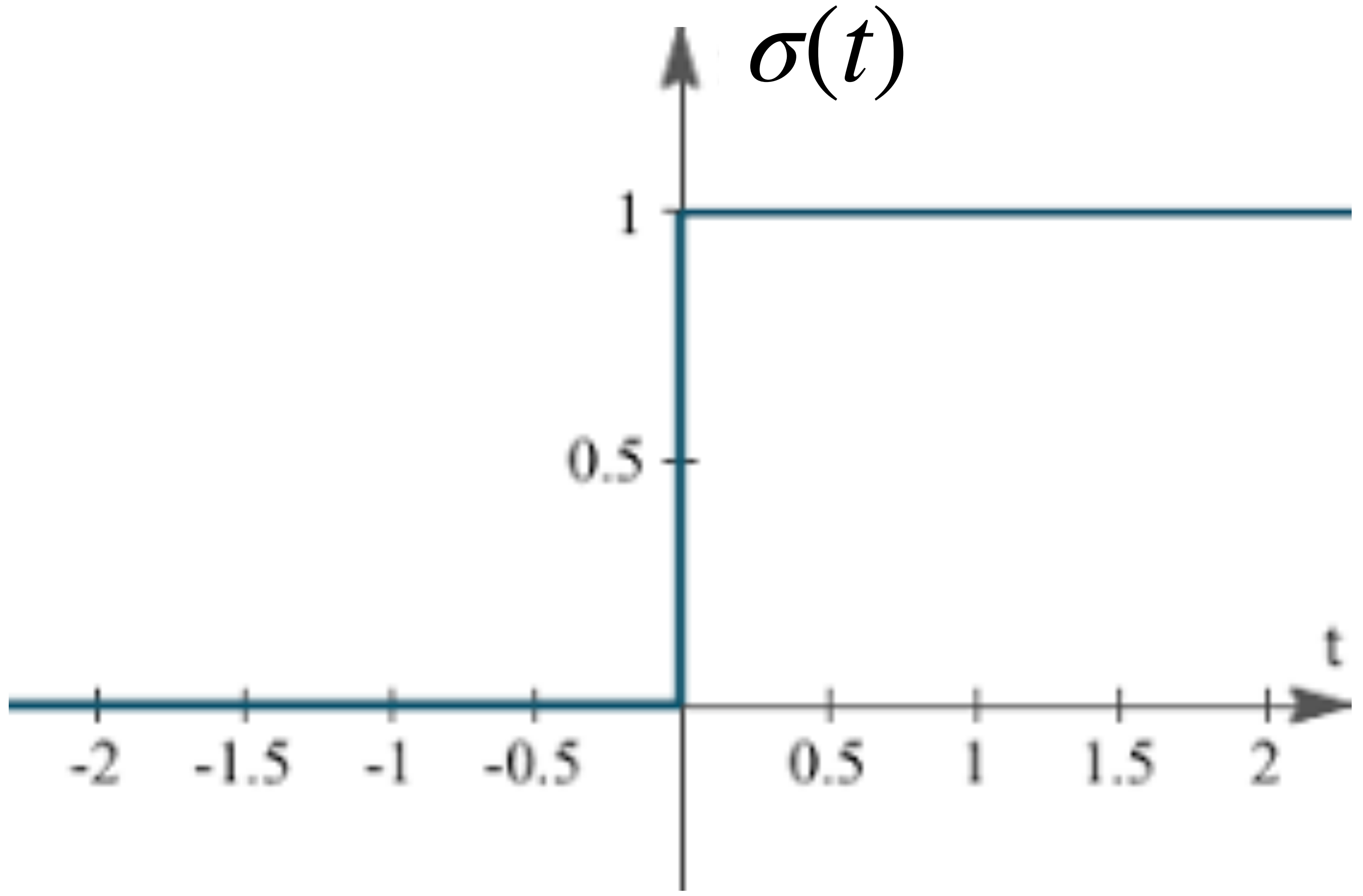
A perceptron with n inputs can be modelled mathematically as

$$f(x_1, \ldots, x_n) = \sigma(w^\top x + b) \ := \begin{cases} 0 & \text{if } \sum_{j=1}^{n} w_j x_j \leq -b \\ 1 & \text{if } \sum_{j=1}^{n} w_j x_j > -b \end{cases}$$

$-b =$ threshold

$$w^\top = (w_1 \quad \cdots \quad w_n) \ , x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

$\sigma(t) =$ Heaviside function



©intmath

Perceptron

$$f(x_1, \ldots, x_n) = \sigma(w^\top x + b)$$

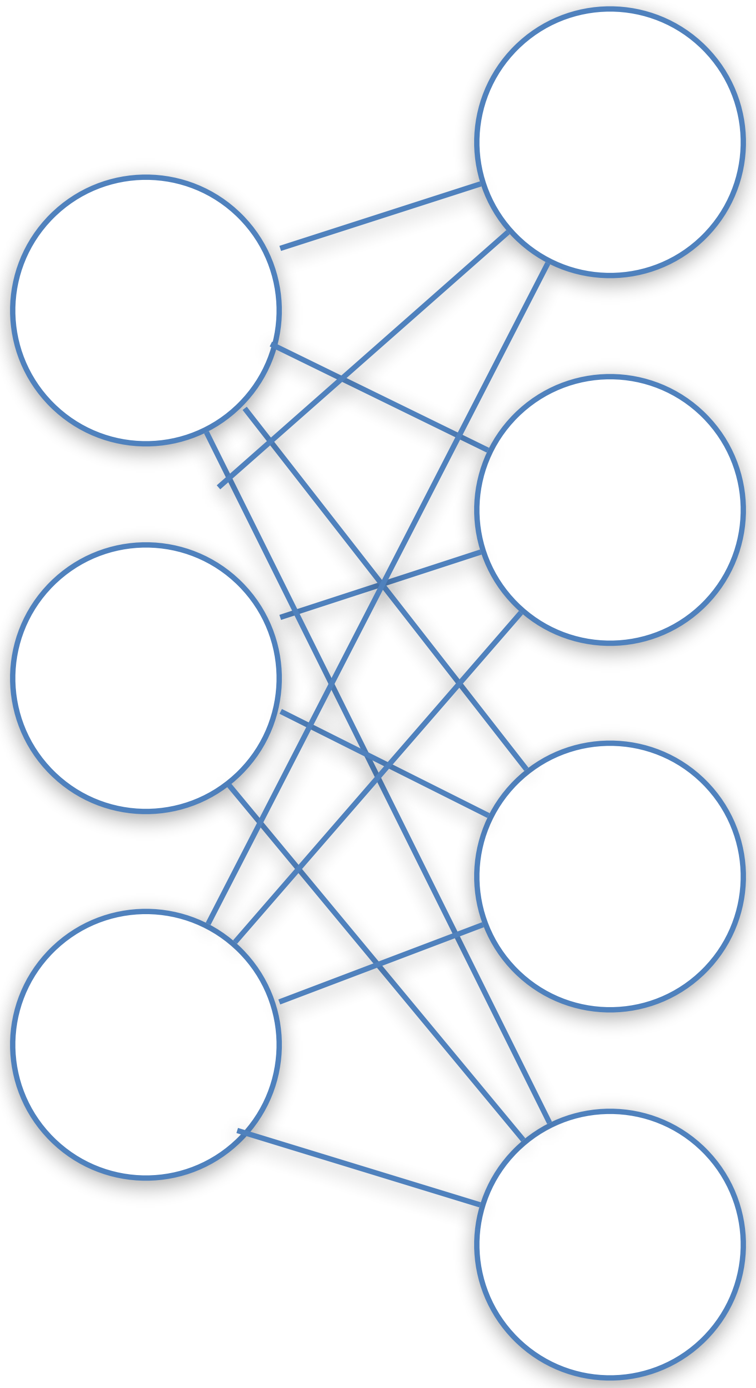
Weights: w

Bias: b

Activation function: σ

Multi-class perceptron

Extend perceptron to also have multiple outputs



Weights: W

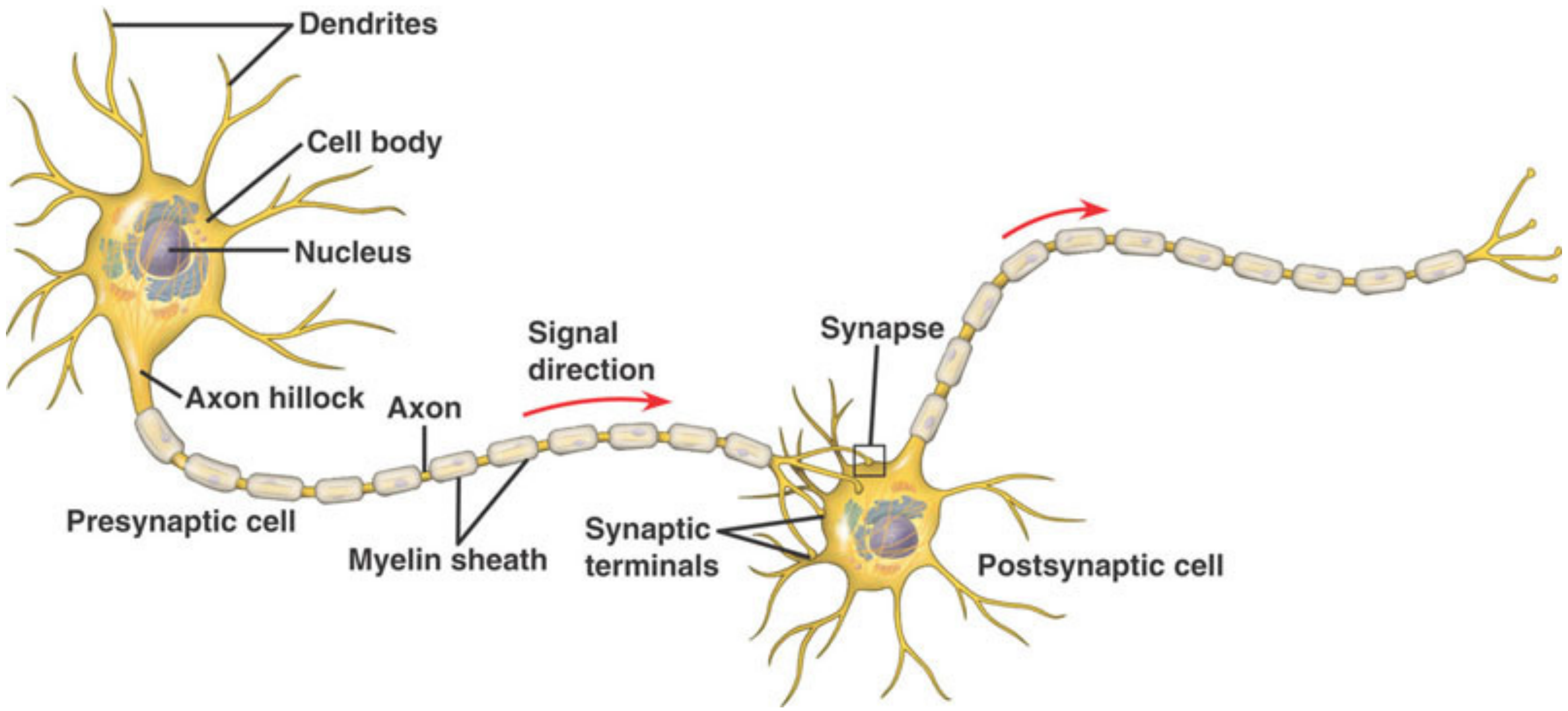
Bias: b

Activation function: σ

$f(x) = \sigma(W^\top x + b)$ $\quad W \in \mathbb{R}^{n \times m}$ $\quad b \in \mathbb{R}^{m}$

Multi-layer perceptron

Multiple neurons communicating:



Weights: W^1, W^2

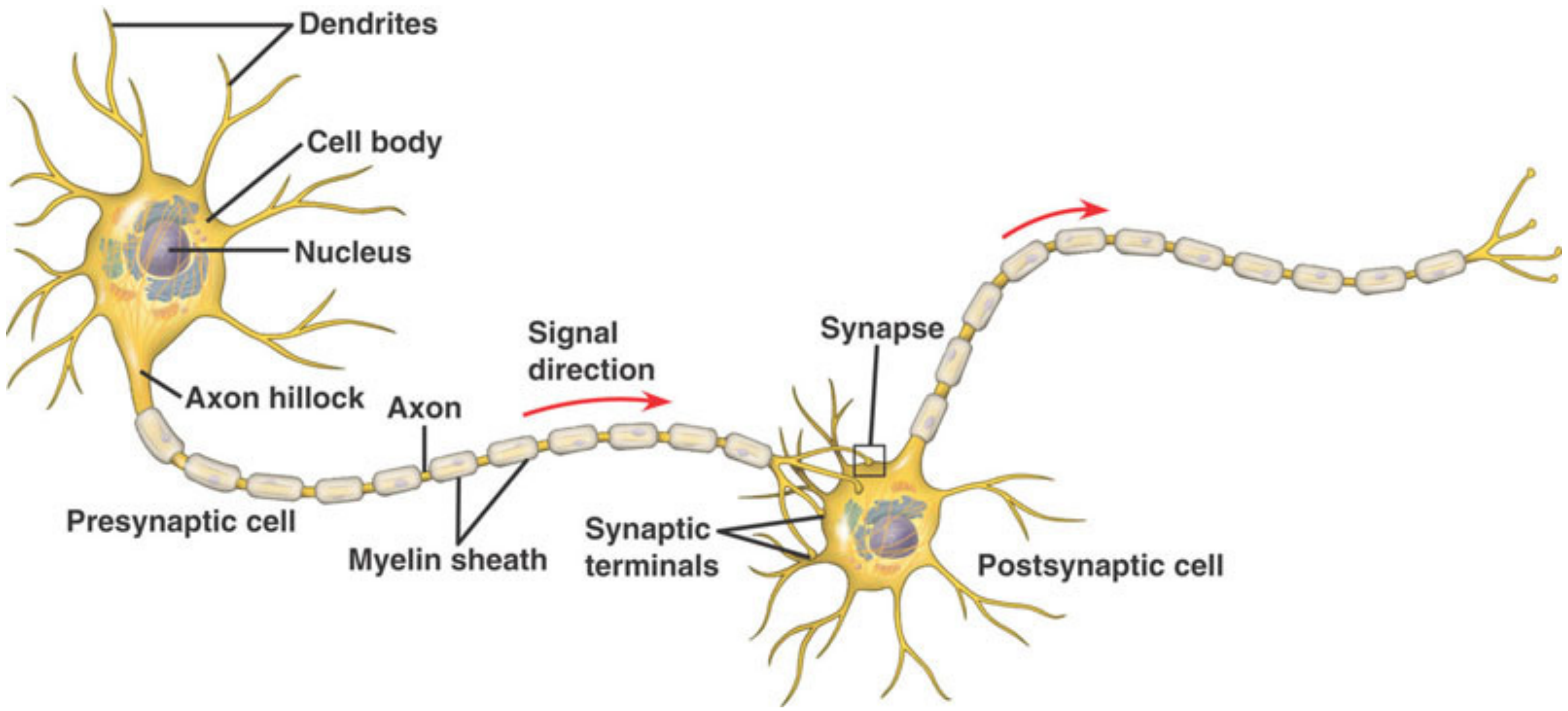
Bias: b^1, b^2

Activation function: σ

2-layer perceptron: $f(x) = \sigma\left((W^2)^\top \sigma\left((W^1)^\top x + b^1\right) + b^2\right)$

Multi-layer perceptron

Multiple neurons communicating:



Weights: $W^1, W^2, \ldots, W^L$

Bias: $b^1, b^2, \ldots, b^L$

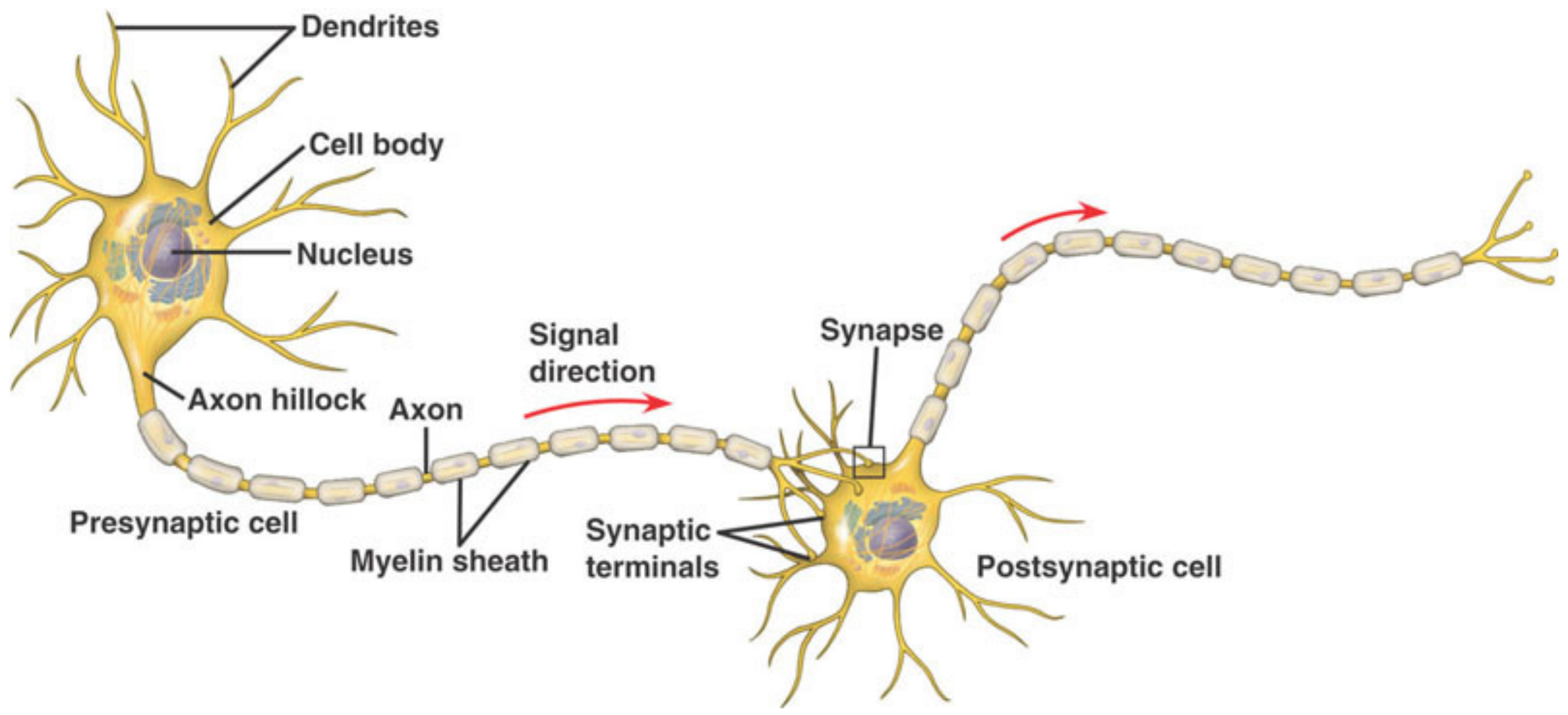
Activation function: σ

No. of layers: L

$$f(x) = \sigma\left((W^L)^\top \sigma\left((W^{L-1})^\top \sigma\left(\ldots\ \sigma\left((W^1)^\top x + b^1\right)\ \ldots\right) + b^{L-1}\right) + b^L\right)$$

Artificial neural network

Multiple neurons communicating:



Weights: $W^1, W^2, \dots, W^L$

Bias: $b^1, b^2, \dots, b^L$

Activation function: σ

No. of layers: L

$$f(x) = \sigma\left((W^L)^\top \sigma\left((W^{L-1})^\top \sigma\left(\dots\ \sigma\left((W^1)^\top x + b^1\right)\ \dots\right) + b^{L-1}\right) + b^L\right)$$

Notation: $f^1(x) := \sigma\left((W^1)^\top x + b^1\right)$, $f^l(x) := \sigma\left((W^l)^\top x + b^l\right)$ and $f(x) := f^L(x)$

Artificial feed-forward neural networks

Summary: $$x^l = \sigma\left((W^l)^\top x^{l-1} + b^l\right)$$

Artificial feed-forward neural networks

Summary: $$x^l = \sigma\left((W^l)^\top x^{l-1} + b^l\right)$$

This can be written as $x^l = f^l(x^{l-1})$ for $f^l(x) := \sigma((W^l)^\top x + b^l)$

Artificial feed-forward neural networks

Summary: $$x^l = \sigma\left((W^l)^\top x^{l-1} + b^l\right)$$

This can be written as $x^l = f^l(x^{l-1})$ for $f^l(x) := \sigma((W^l)^\top x + b^l)$

Then the overall neural network reads as

$$y = f(x) := f^L \circ \cdots \circ f^2 \circ f^1(x)$$

$x =$ input

$y =$ output

$L =$ total no. of layers

where the composition $\circ$ is defined as

$$(f \circ g)(x) := f(g(x))$$

Artificial neural network

How many layers L do we choose?

How do we estimate the parameters?

How do we estimate the parameters?

Choose $w_1 = 6, w_2 = 2, w_3 = 2$

$\text{threshold} = 5$

How do we estimate the parameters?

Choose $w_1 = 6, w_2 = 2, w_3 = 2$

threshold $= 5$

Define $f(x) := \begin{cases} 0 & \text{if } 6x_1 + 2x_2 + 2x_3 \leq 5 \\ 1 & \text{if } 6x_1 + 2x_2 + 2x_3 > 5 \end{cases}$ for $x = (x_1, x_2, x_3)$

How do we estimate the parameters?

Choose $w_1 = 6, w_2 = 2, w_3 = 2$

threshold $= 5$

Define $f(x) := \begin{cases} 0 & \text{if } 6x_1 + 2x_2 + 2x_3 \leq 5 \\ 1 & \text{if } 6x_1 + 2x_2 + 2x_3 > 5 \end{cases}$ for $x = (x_1, x_2, x_3)$

Generate outputs $y_i = f(x_i)$ for inputs $x_1, x_2, \ldots, x_s$

How do we estimate the parameters?

Choose $w_1 = 6, w_2 = 2, w_3 = 2$

threshold $= 5$

Define $f(x) := \begin{cases} 0 & \text{if } 6x_1 + 2x_2 + 2x_3 \leq 5 \\ 1 & \text{if } 6x_1 + 2x_2 + 2x_3 > 5 \end{cases}$ for $x = (x_1, x_2, x_3)$

Generate outputs $y_i = f(x_i)$ for inputs $x_1, x_2, \ldots, x_s$

This is a forward problem:

Weights, threshold → Model function f → Model outputs $y_i = f(x_i)$

How do we estimate the parameters?

What we are interested in (in practice) is the inverse problem:

How do we estimate the parameters?

What we are interested in (in practice) is the inverse problem:

Model inputs x_i and outputs y_i → Weights, threshold

How do we estimate the parameters?

What we are interested in (in practice) is the inverse problem:

Model inputs x_i and outputs y_i → Weights, threshold

Imagine we have some ground truth data that tells us the preferences of potential attendees and we know if they went or no to the concert → "Learn" the weights and threshold

How do we estimate the parameters?

What we are interested in (in practice) is the inverse problem:

Model inputs x_i and outputs y_i → Weights, threshold

Imagine we have some ground truth data that tells us the preferences of potential attendees and we know if they went or no to the concert → "Learn" the weights and threshold

This might help us train the model and predict what other people might do

How do we estimate the parameters?

How do we solve such an inverse problem?

Empirical risk minimisation

Empirical risk minimisation: based on pairs of training data $(x_1, y_1), \ldots, (x_s, y_s)$, find optimal parameters $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Empirical risk minimisation

Empirical risk minimisation: based on pairs of training data $(x_1, y_1), \ldots, (x_s, y_s)$, find optimal parameters $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Hence a nonlinear regression problem:

$$\min_{W^1,\ldots,W^L,b^1,\ldots,b^L} \frac{1}{s} \sum_{i=1}^{s} \| f(x_i) - y_i \|^2$$

Empirical risk minimisation

Empirical risk minimisation: based on pairs of training data $(x_1, y_1), \ldots, (x_s, y_s)$, find optimal parameters $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Hence a nonlinear regression problem:

$$\min_{W^1,\ldots,W^L,b^1,\ldots,b^L} \frac{1}{s} \sum_{i=1}^{s} \ell\left(f(x_i), y_i\right)$$

ℓ = loss-function

Empirical risk minimisation

Empirical risk minimisation: based on pairs of training data $(x_1, y_1), \ldots, (x_s, y_s)$, find optimal parameters $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Hence a nonlinear regression problem:

$$\min_{W^1,\ldots,W^L,b^1,\ldots,b^L} \frac{1}{s} \sum_{i=1}^{s} \ell\left(f(x_i), y_i\right)$$

ℓ = loss-function

Potentially lots of unknowns! It is crucial to set the problem based on the amount of data available

Training neural networks

Supervised training of neural networks is basically like all other supervised training that we have encountered in the module.

Training neural networks

Supervised training of neural networks is basically like all other supervised training that we have encountered in the module.

We formulate cost function and minimise it for pairs of input/output training data

Training neural networks

Supervised training of neural networks is basically like all other supervised training that we have encountered in the module.

We formulate cost function and minimise it for pairs of input/output training data

Example: MSE cost function

$$E := \frac{1}{2s} \sum_{i=1}^{s} \|f(x_i) - y_i\|^2$$

Training neural networks

Supervised training of neural networks is basically like all other supervised training that we have encountered in the module.

We formulate cost function and minimise it for pairs of input/output training data

Example: MSE cost function

$$E := \frac{1}{2s} \sum_{i=1}^{s} \|f(x_i) - y_i\|^2$$

Optimise for parameters $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Training neural networks

$$E := \frac{1}{2s}\sum_{i=1}^{s}\|f(x_i) - y_i\|^2$$

$$= \frac{1}{2s}\sum_{i=1}^{s}\left\| f^L \circ \cdots \circ f^2 \circ f^1(x_i) - y_i \right\|^2$$

How do we determine the optimal parameters $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$?

Training neural networks

$$E := \frac{1}{2s} \sum_{i=1}^{s} \|f(x_i) - y_i\|^2$$

$$= \frac{1}{2s} \sum_{i=1}^{s} \left\| f^L \circ \cdots \circ f^2 \circ f^1(x_i) - y_i \right\|^2$$

How do we determine the optimal parameters $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$?

Let's assume f is differentiable, i.e. ∇f exists

Then we can for example perform gradient descent

Training neural networks

So our variables are $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Training neural networks

So our variables are $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Hence gradient descent means

Training neural networks

So our variables are $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Hence gradient descent means

$$W_k^1 = W_{k-1}^1 - \tau \nabla E(W_{k-1}^1)$$

Training neural networks

So our variables are $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Hence gradient descent means

$$W_k^1 = W_{k-1}^1 - \tau \nabla E(W_{k-1}^1)$$
$$\vdots$$

Training neural networks

So our variables are $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Hence gradient descent means

$$W_k^1 = W_{k-1}^1 - \tau \nabla E(W_{k-1}^1)$$
$$\vdots$$
$$W_k^L = W_{k-1}^L - \tau \nabla E(W_{k-1}^L)$$

Training neural networks

So our variables are $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Hence gradient descent means

$$W_k^1 = W_{k-1}^1 - \tau \nabla E(W_{k-1}^1)$$

$$\vdots$$

$$W_k^L = W_{k-1}^L - \tau \nabla E(W_{k-1}^L)$$

$$b_k^1 = b_{k-1}^1 - \tau \nabla E(b_{k-1}^1)$$

Training neural networks

So our variables are $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Hence gradient descent means

$$W_k^1 = W_{k-1}^1 - \tau \nabla E(W_{k-1}^1)$$

$$\vdots$$

$$W_k^L = W_{k-1}^L - \tau \nabla E(W_{k-1}^L)$$

$$b_k^1 = b_{k-1}^1 - \tau \nabla E(b_{k-1}^1)$$

$$\vdots$$

Training neural networks

So our variables are $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Hence gradient descent means

$$W_k^1 = W_{k-1}^1 - \tau \nabla E(W_{k-1}^1)$$

$$\vdots$$

$$W_k^L = W_{k-1}^L - \tau \nabla E(W_{k-1}^L)$$

$$b_k^1 = b_{k-1}^1 - \tau \nabla E(b_{k-1}^1)$$

$$\vdots$$

$$b_k^L = b_{k-1}^L - \tau \nabla E(b_{k-1}^L)$$

Training neural networks

So our variables are $W^1, W^2, \ldots, W^L, b^1, b^2, \ldots, b^L$

Hence gradient descent means

$$W^1_k = W^1_{k-1} - \tau \nabla E(W^1_{k-1})$$
$$\vdots$$
$$W^L_k = W^L_{k-1} - \tau \nabla E(W^L_{k-1})$$

$$b^1_k = b^1_{k-1} - \tau \nabla E(b^1_{k-1})$$
$$\vdots$$
$$b^L_k = b^L_{k-1} - \tau \nabla E(b^L_{k-1})$$

So, we need to compute lots of partial derivatives!

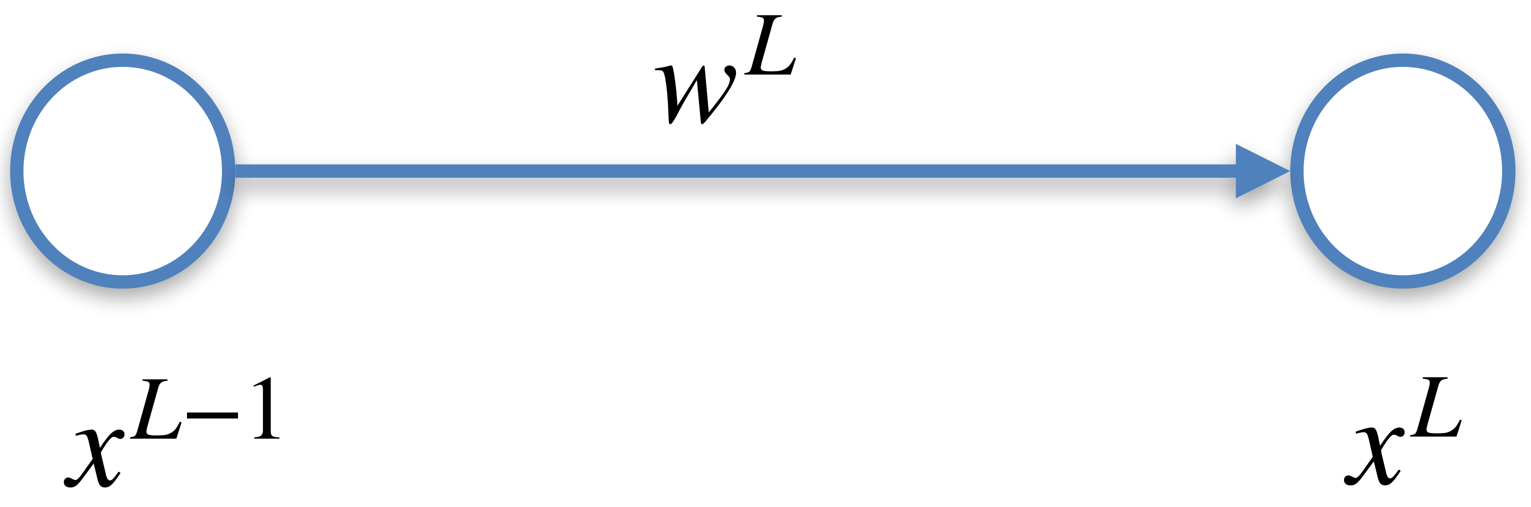
Backpropagation algorithm

Backpropagation algorithm

For simplicity let's consider a simple linear case, without activation function

Backpropagation algorithm

For simplicity let's consider a simple linear case, without activation function

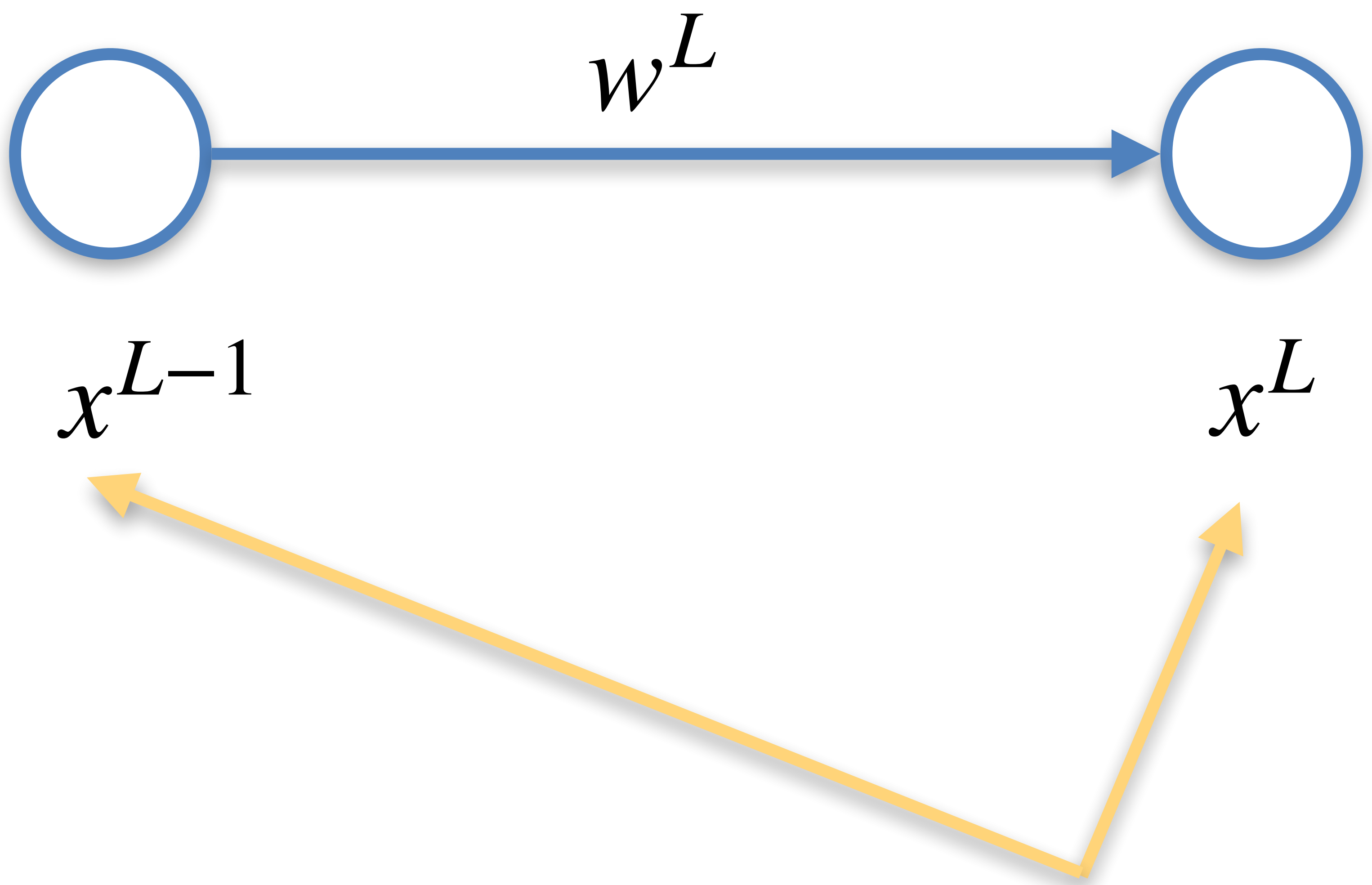


Desired output

y

Backpropagation algorithm

For simplicity let's consider a simple linear case, without activation function



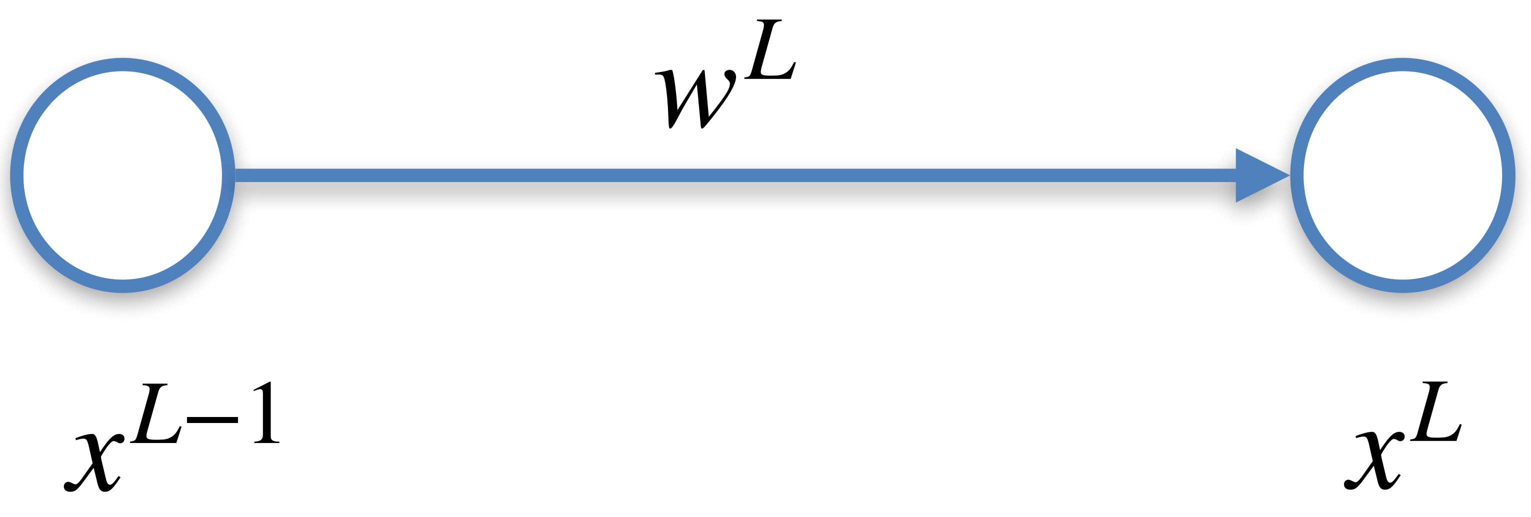
Desired output

y

Are often called "activations"

Backpropagation algorithm

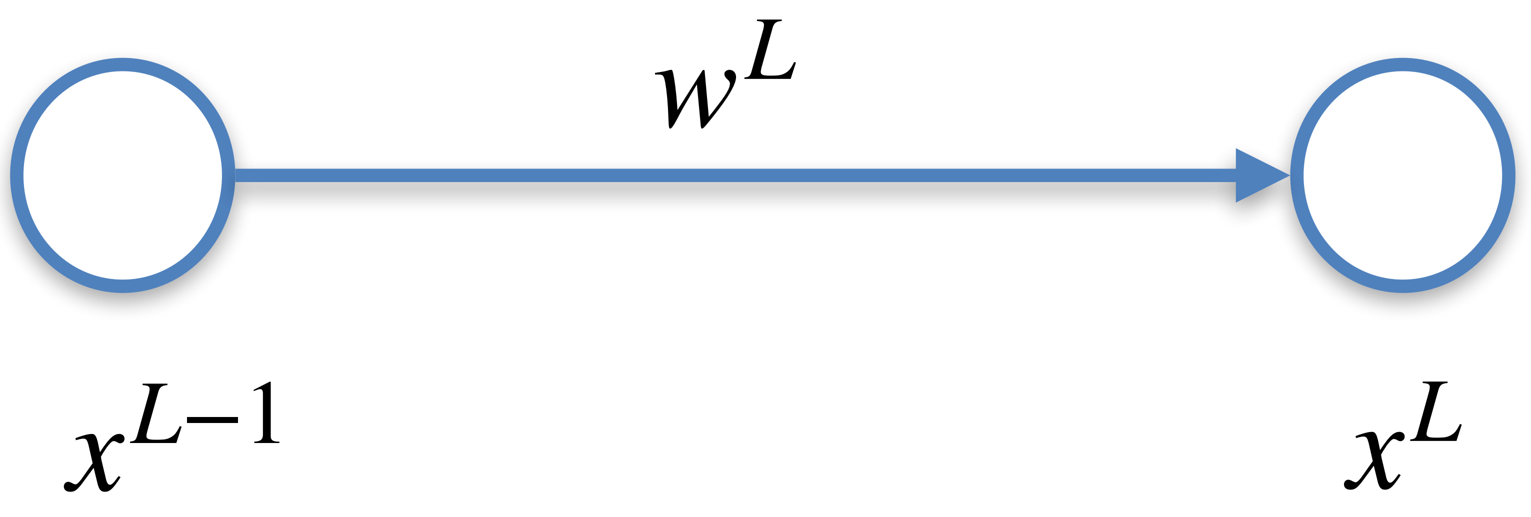
Backpropagation algorithm



Desired output

y

Backpropagation algorithm

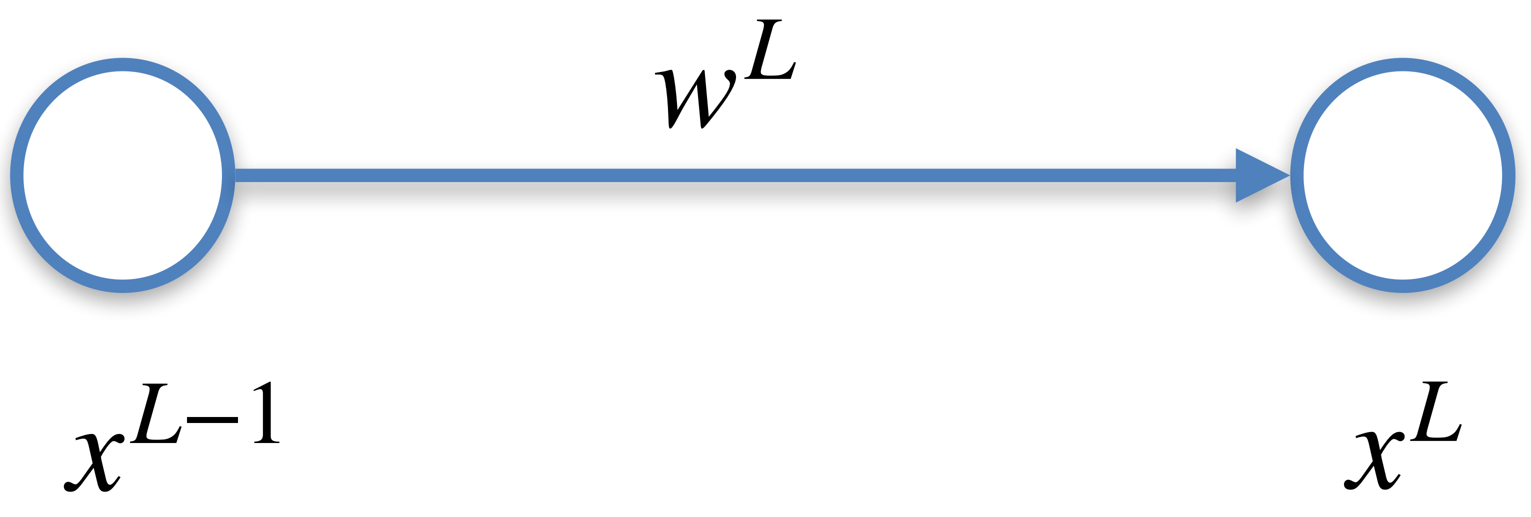


Desired output

y

Imagine that the input is $X^{L-1} = \frac{3}{2}$

Backpropagation algorithm



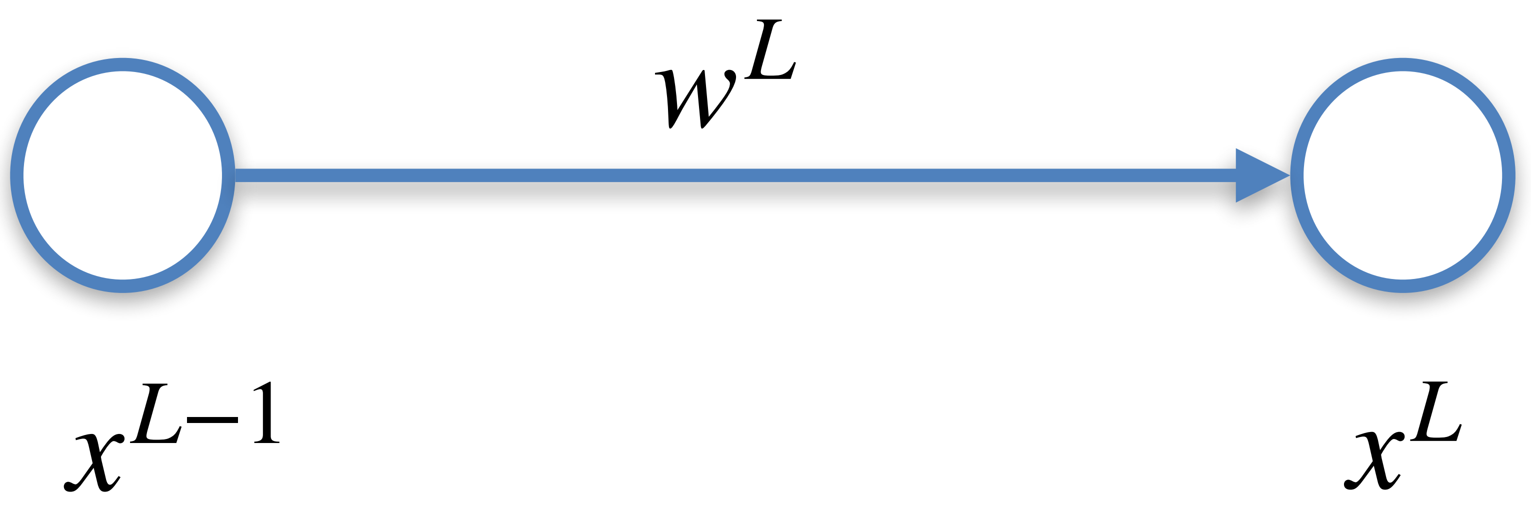
Desired output

y

Imagine that the input is $X^{L-1} = \frac{3}{2}$

The desired output $y = \frac{1}{2}$

Backpropagation algorithm



Desired output

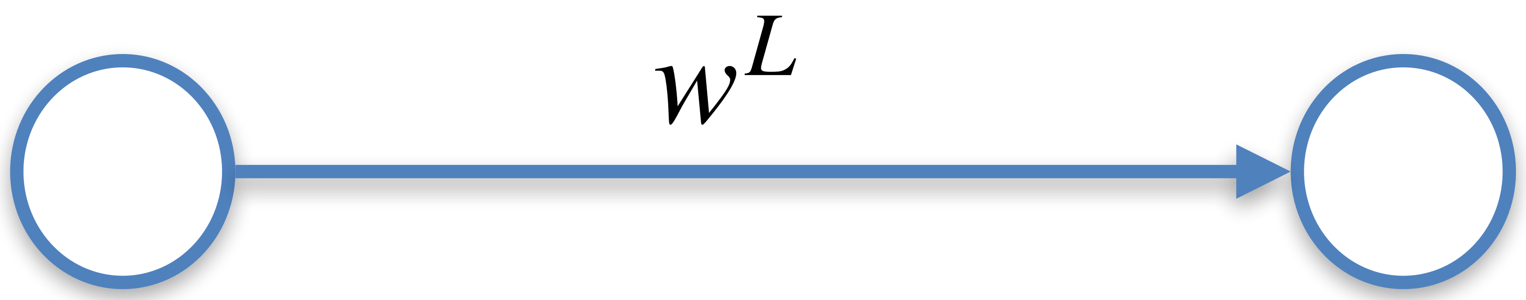
y

Imagine that the input is $X^{L-1} = \frac{3}{2}$

The desired output $y = \frac{1}{2}$

We can initialise gradient descent by setting $w_0^L = 0.8$ and $b_0^L = 1$

Backpropagation algorithm



$x^{L-1} = 1.5$ x^L

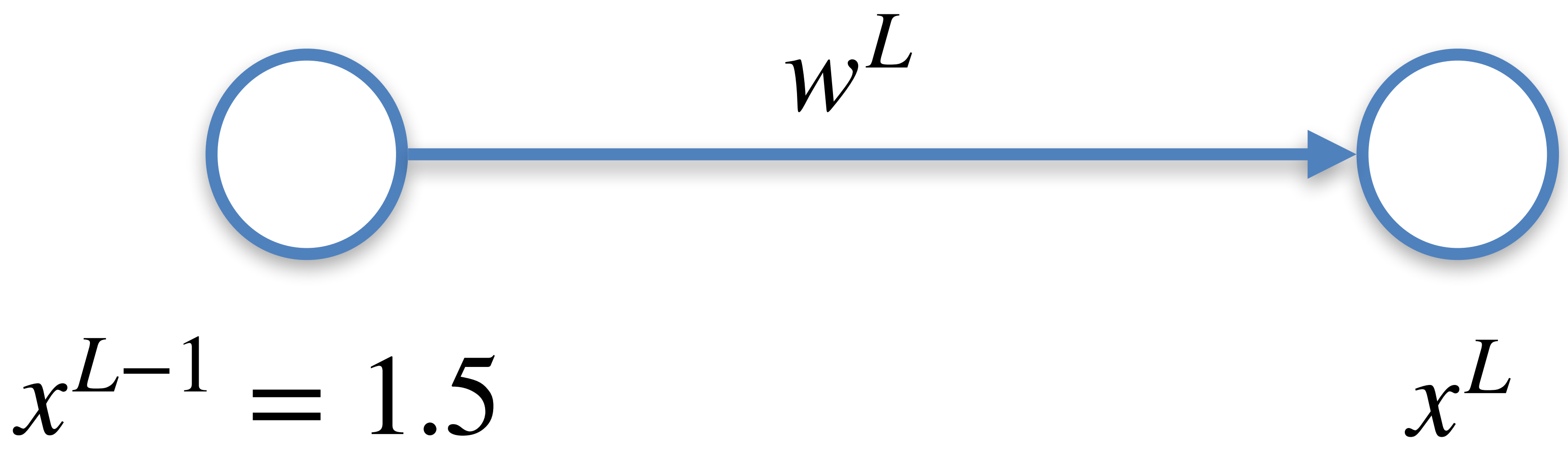
Desired output

$$y = 0.5$$

We know that

$$x^L = w^L x^{L-1} + b^L$$

Backpropagation algorithm



Desired output

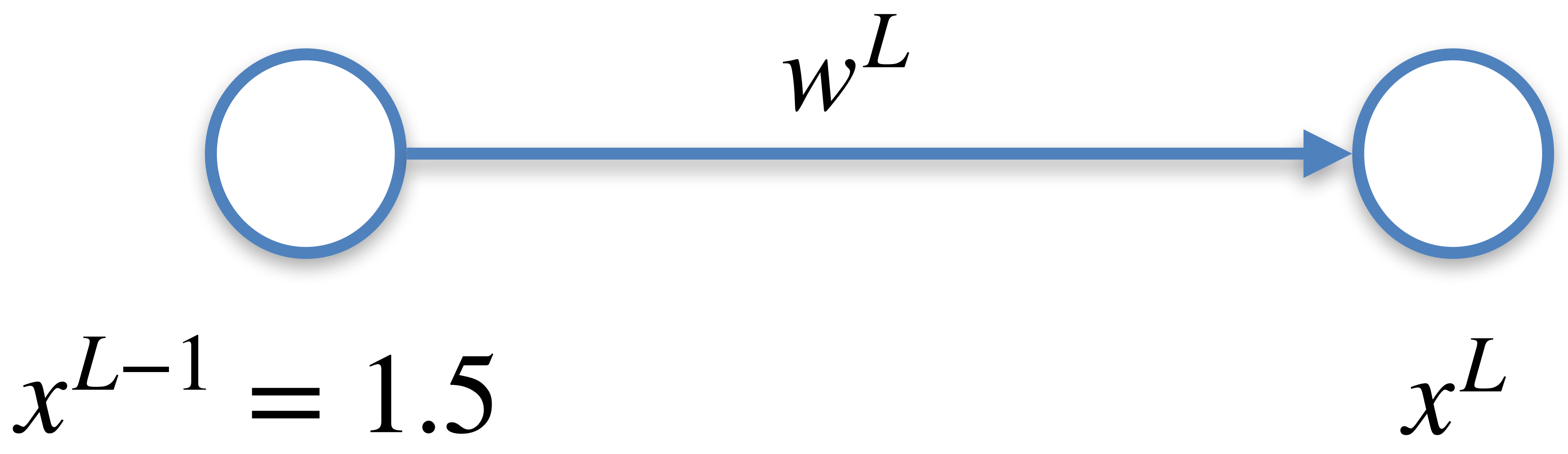
$$y = 0.5$$

We know that

$$x^L = w^L x^{L-1} + b^L$$

Hence with the initial values of w and b we get

Backpropagation algorithm



Desired output

$$y = 0.5$$

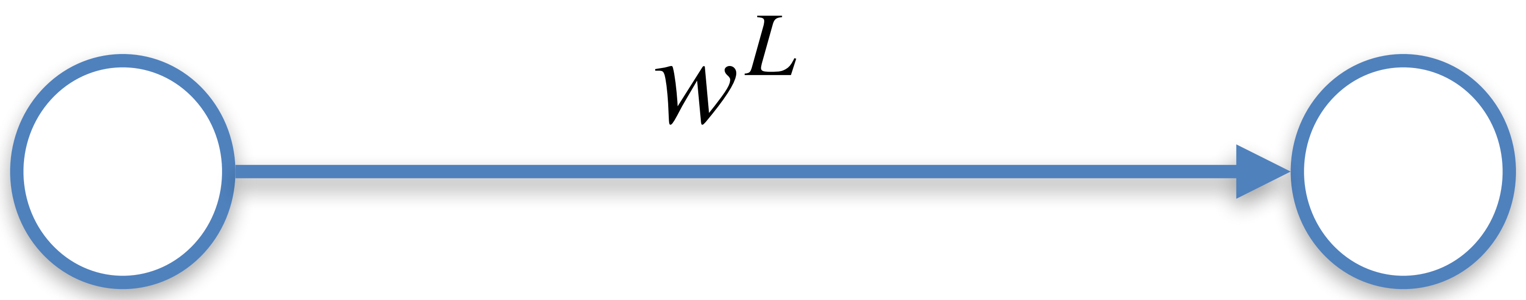
We know that

$$x^L = w^L x^{L-1} + b^L$$

Hence with the initial values of w and b we get

$$x^L = w^L x^{L-1} + b^L = 0.8 \times 1.5 + 1 = 2.2$$

Backpropagation algorithm



Desired output

$$y = 0.5$$

We know that

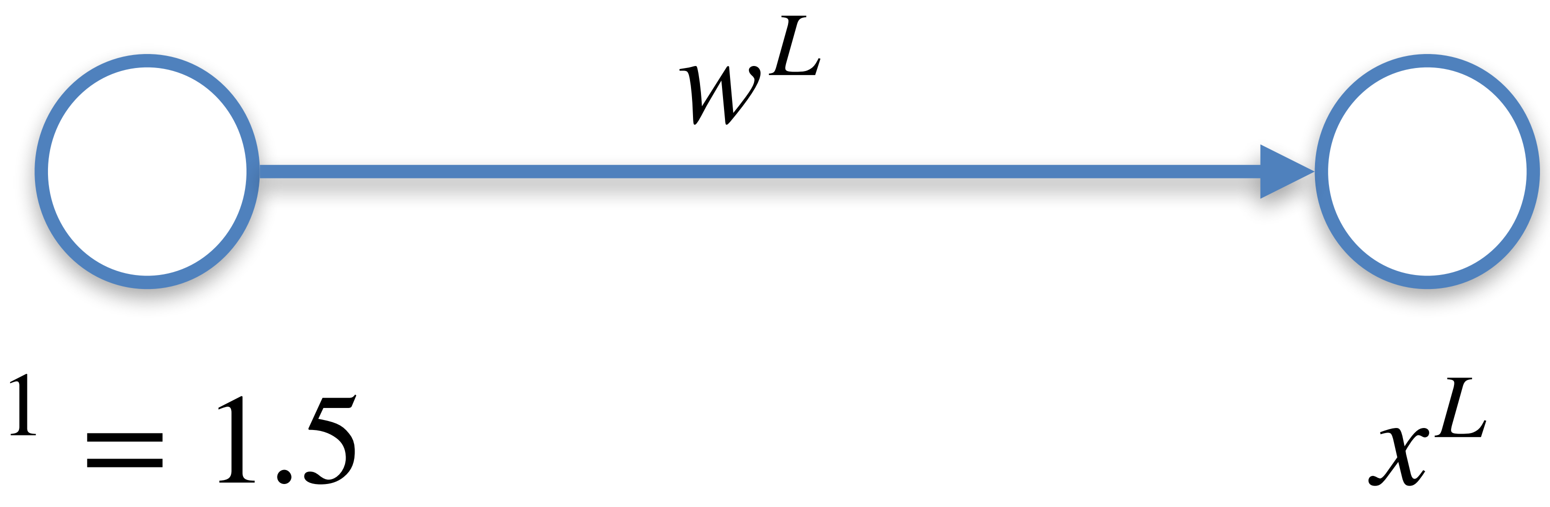
$$x^L = w^L x^{L-1} + b^L$$

Hence with the initial values of w and b we get

$$x^L = w^L x^{L-1} + b^L = 0.8 \times 1.5 + 1 = 2.2$$

We are far from the desired value

Backpropagation algorithm



Desired output

$$y = 0.5$$

We know that

$$x^L = w^L x^{L-1} + b^L$$

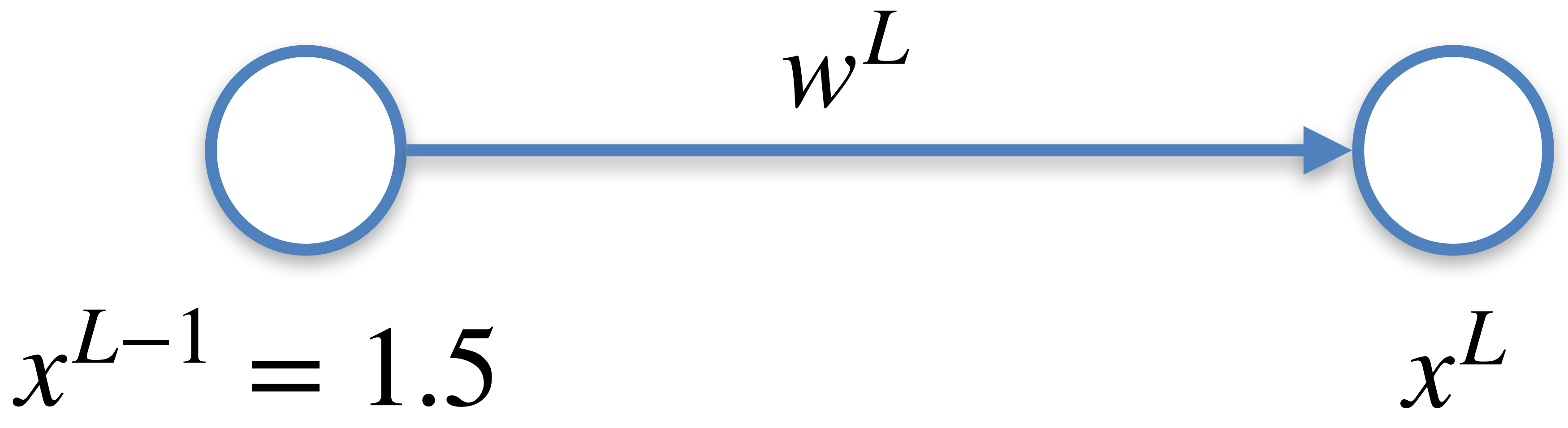
Hence with the initial values of w and b we get

$$x^L = w^L x^{L-1} + b^L = 0.8 \times 1.5 + 1 = 2.2$$

We are far from the desired value

We need to apply gradient descent!

Backpropagation algorithm

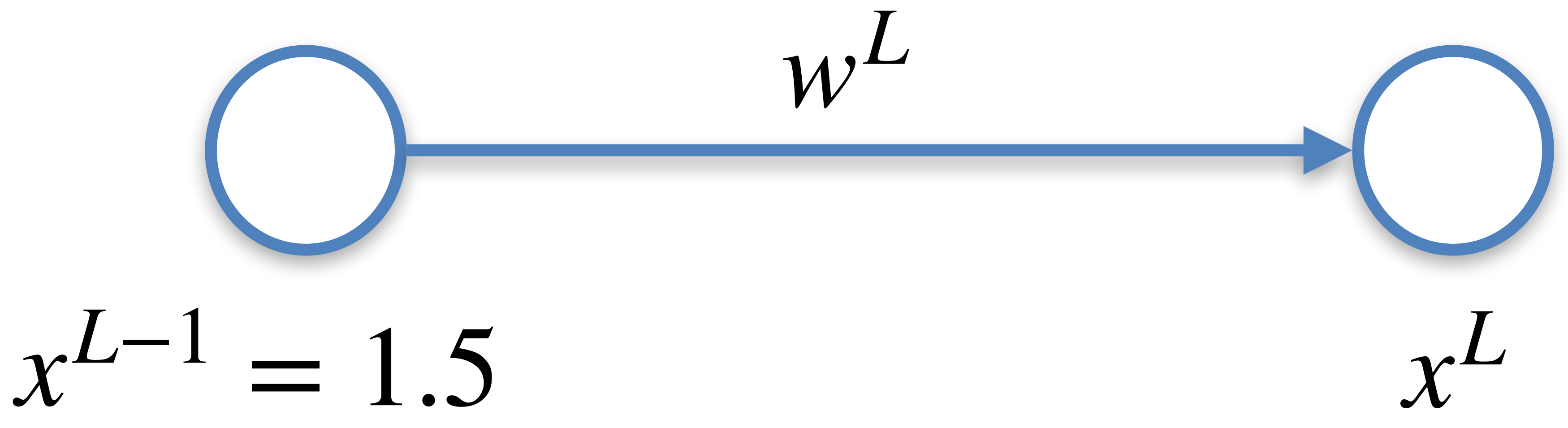


Desired output

$$y = 0.5$$

The variables are w and b and the gradient is then

Backpropagation algorithm



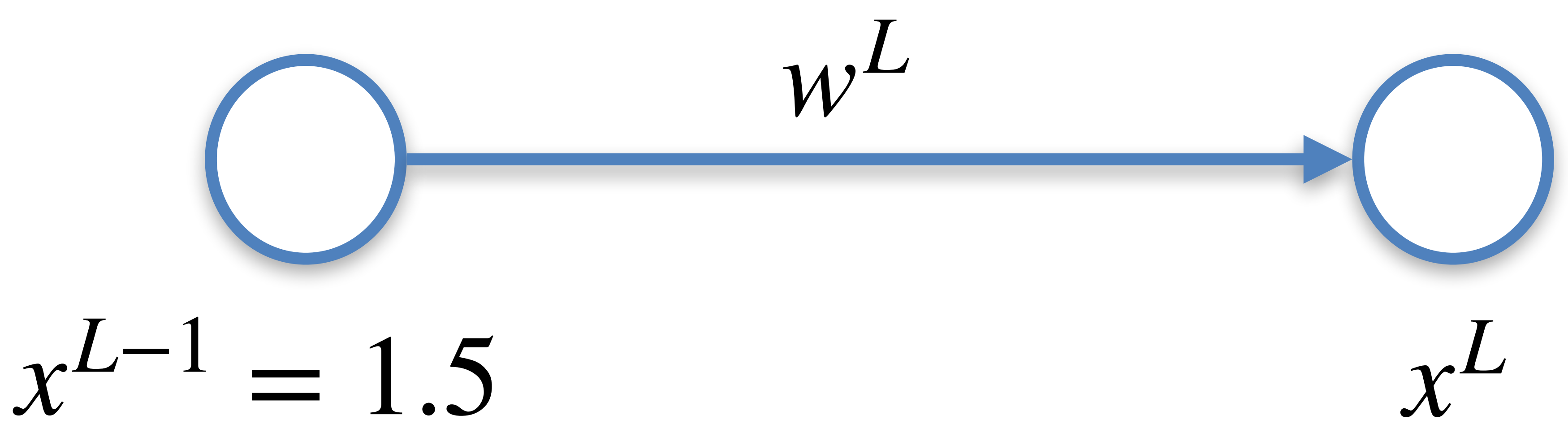
Desired output

$$y = 0.5$$

The variables are w and b and the gradient is then

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial b^L} \right)^\top$$

Backpropagation algorithm



Desired output

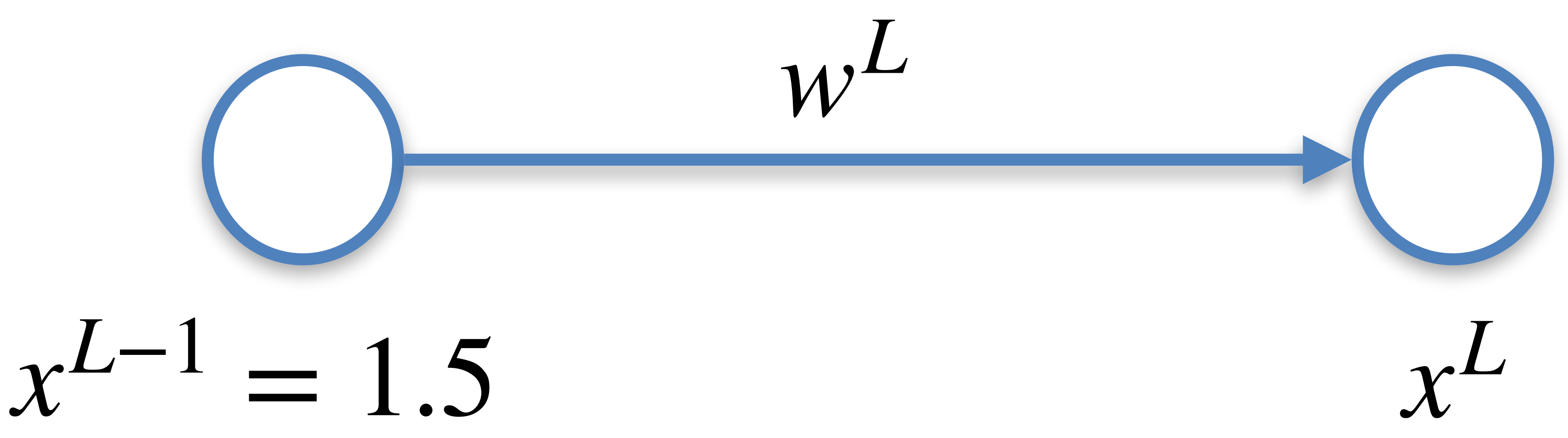
$$y = 0.5$$

The variables are w and b and the gradient is then

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial b^L} \right)^\top$$

$$E = (x^L - y)^2$$

Backpropagation algorithm



Desired output

$$y = 0.5$$

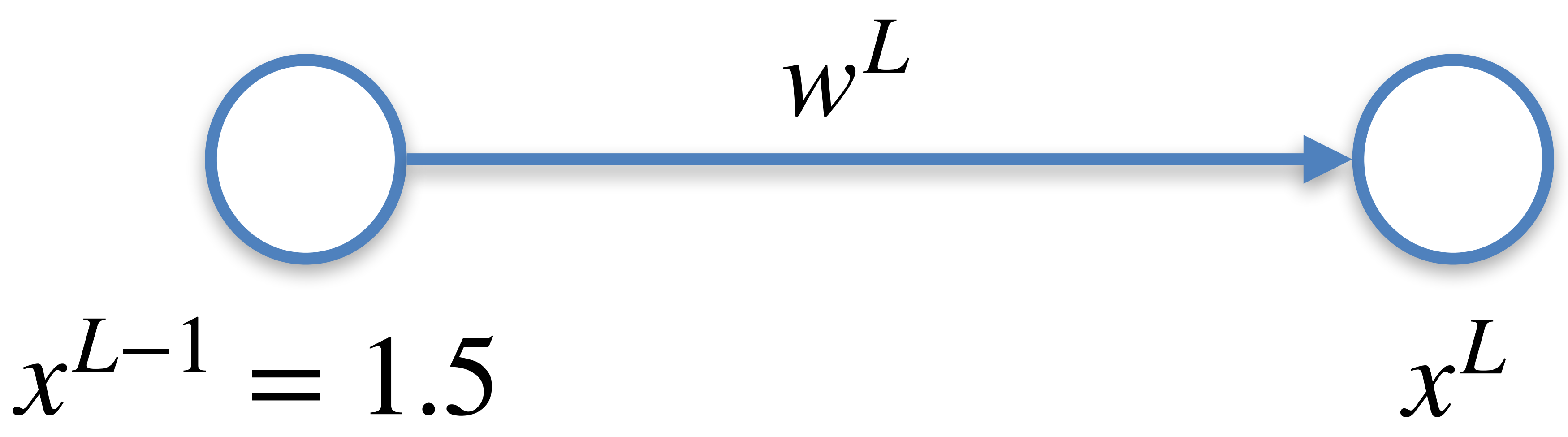
The variables are w and b and the gradient is then

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial b^L} \right)^\top$$

$$E = (x^L - y)^2$$

$$x^L = w^L x^{L-1} + b^L$$

Backpropagation algorithm



Desired output

$$y = 0.5$$

The variables are w and b and the gradient is then

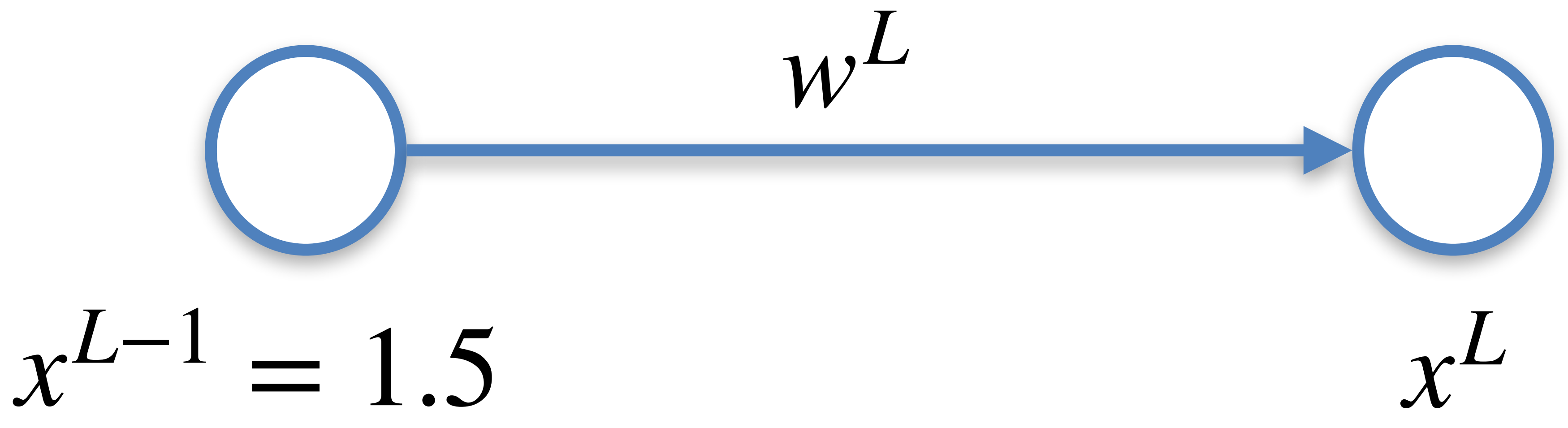
$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial b^L} \right)^\top$$

$$E = (x^L - y)^2$$

$$x^L = w^L x^{L-1} + b^L$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial w^L}$$

Backpropagation algorithm



Desired output

$$y = 0.5$$

The variables are w and b and the gradient is then

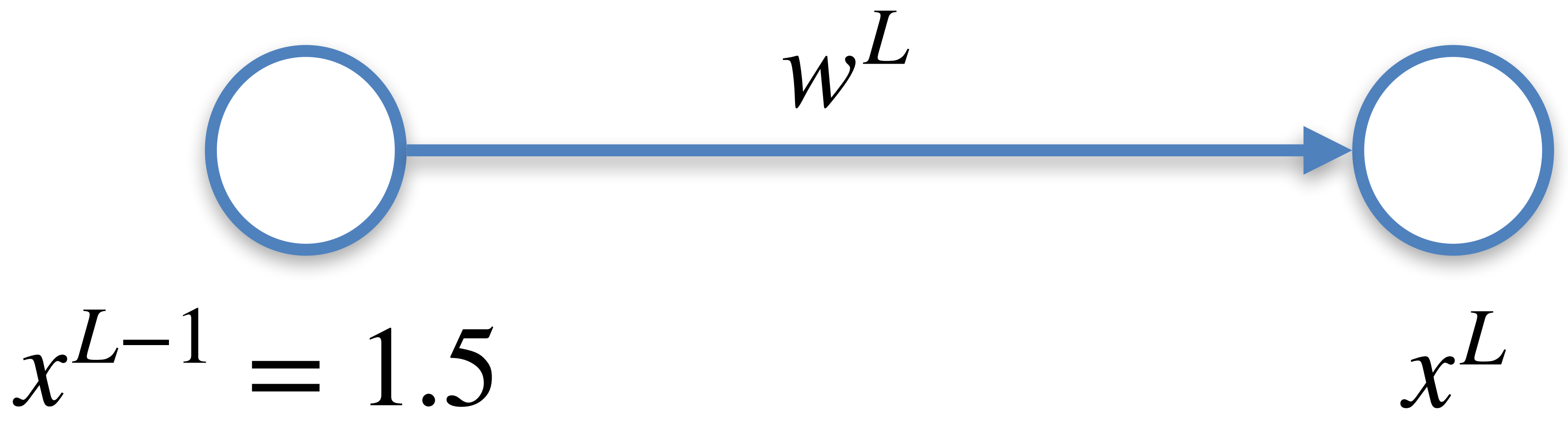
$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial b^L} \right)^\top$$

$$E = (x^L - y)^2$$

$$x^L = w^L x^{L-1} + b^L$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial w^L} \quad = 2(x^L - y)x^{L-1}$$

Backpropagation algorithm



Desired output

$$y = 0.5$$

The variables are w and b and the gradient is then

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial b^L} \right)^\top$$

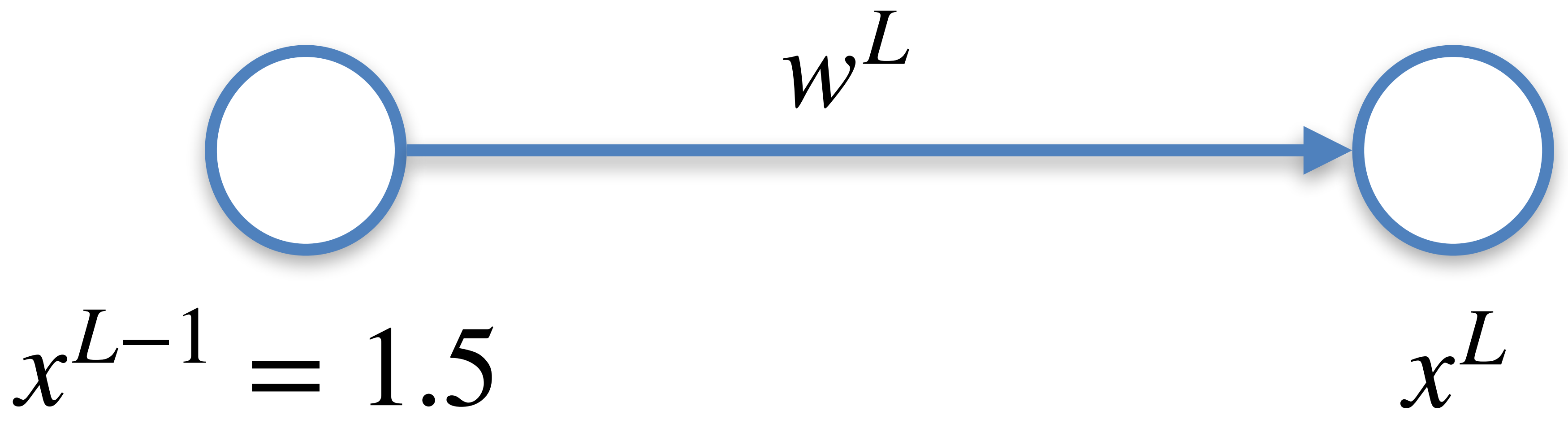
$$E = (x^L - y)^2$$

$$x^L = w^L x^{L-1} + b^L$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L}$$

Backpropagation algorithm



Desired output

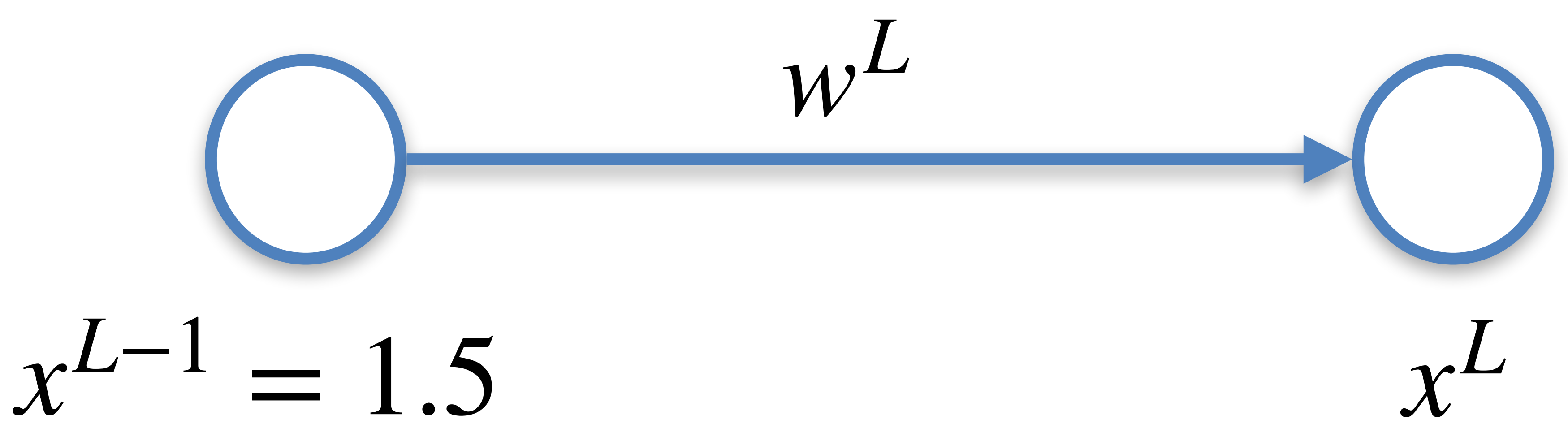
$$y = 0.5$$

The variables are w and b and the gradient is then

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial b^L} \right)^\top \qquad E = (x^L - y)^2 \qquad x^L = w^L x^{L-1} + b^L$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1} \qquad \frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} = 2(x^L - y)$$

Backpropagation algorithm



Desired output

$$y = 0.5$$

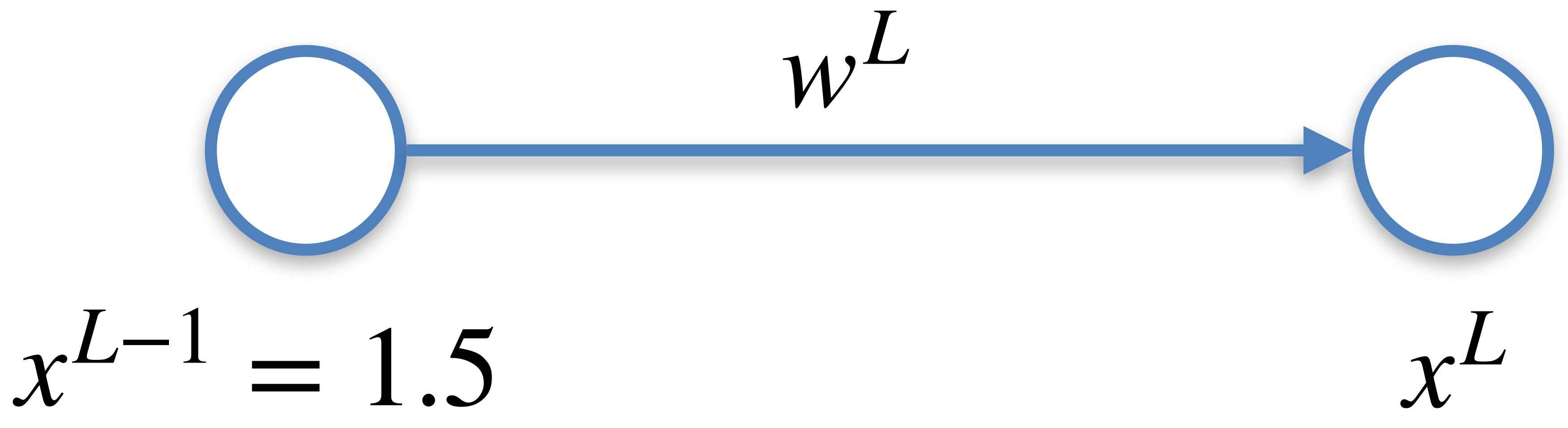
The variables are w and b and the gradient is then

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial b^L} \right)^\top \qquad E = (x^L - y)^2 \qquad x^L = w^L x^{L-1} + b^L$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1} \qquad \frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} = 2(x^L - y)$$

Chain rule!

Backpropagation algorithm



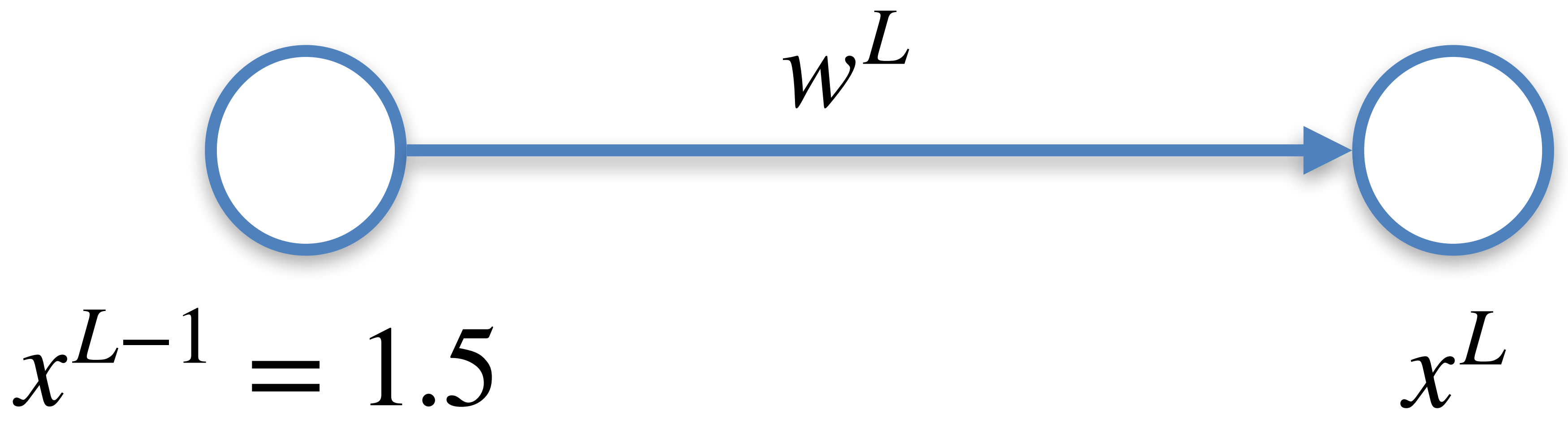
Desired output

$$y = 0.5$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} \quad = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} \quad = 2(x^L - y)$$

Backpropagation algorithm



Desired output

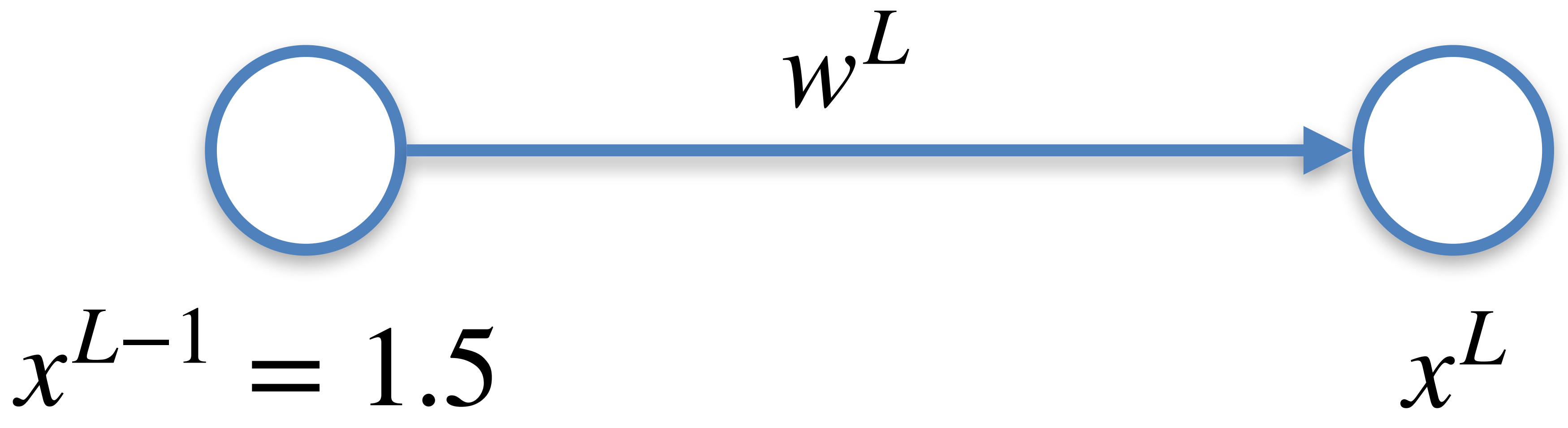
$$y = 0.5$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} = 2(x^L - y)$$

Note how

Backpropagation algorithm



Desired output

$$y = 0.5$$

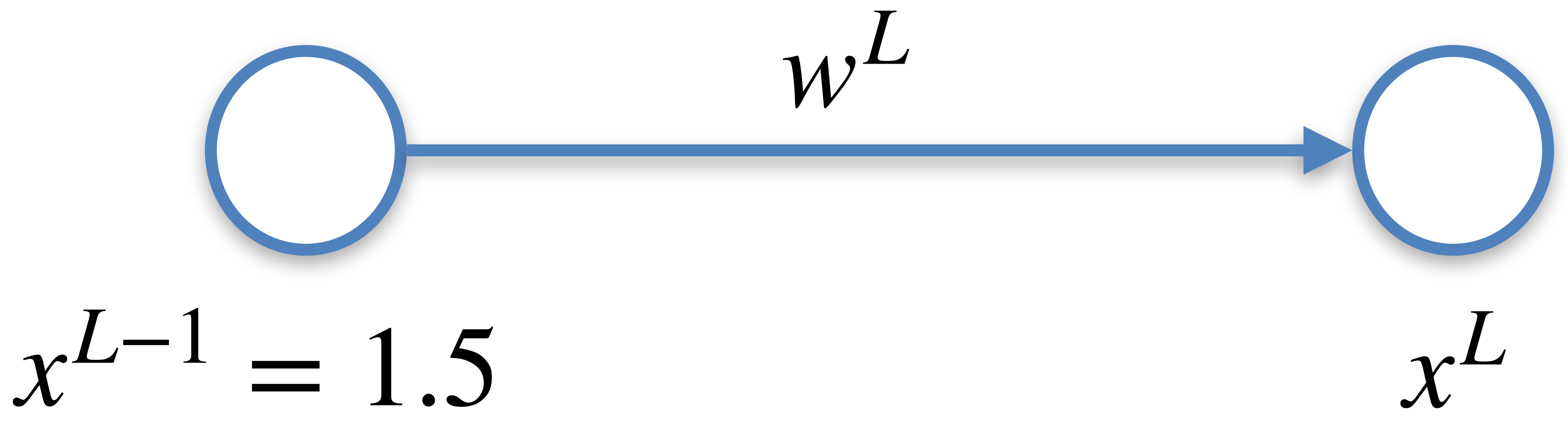
$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} = 2(x^L - y)$$

Note how

1) To compute the gradient we need the value x

Backpropagation algorithm



Desired output

$$y = 0.5$$

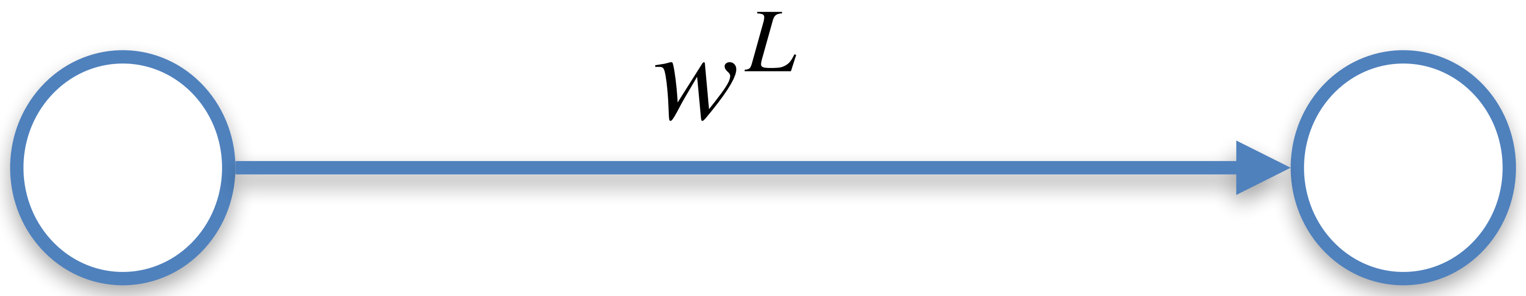
$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} \quad = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} \quad = 2(x^L - y)$$

Note how

1) To compute the gradient we need the value x
2) Hence the first step of the back propagation is the so called forward pass where, given the initial values of w and b, we compute relative inputs and outputs

Backpropagation algorithm



$$x^{L-1} = \frac{3}{2} \qquad x^L$$

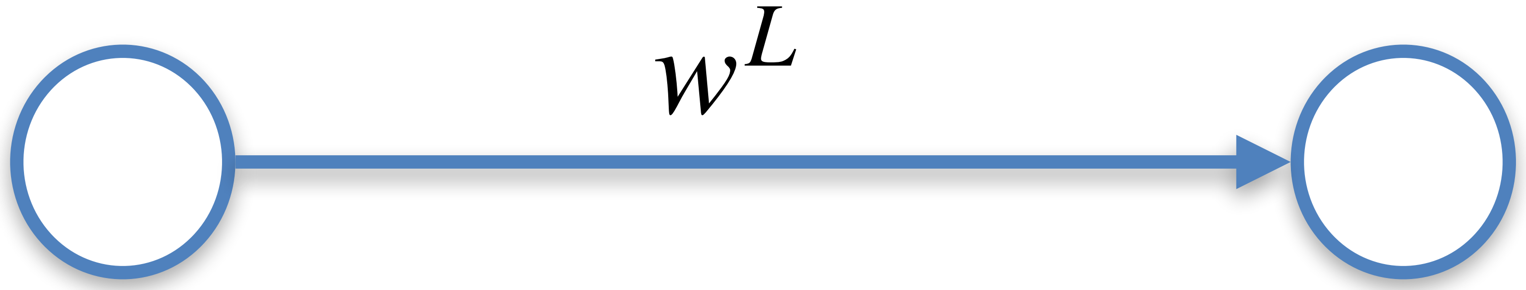
Desired output

$$y = \frac{1}{2}$$

We are ready to update the initial values of the parameters! Let us set $\tau = 0.1$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1} \qquad \frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} = 2(x^L - y)$$

Backpropagation algorithm



Desired output

$$y = \frac{1}{2}$$

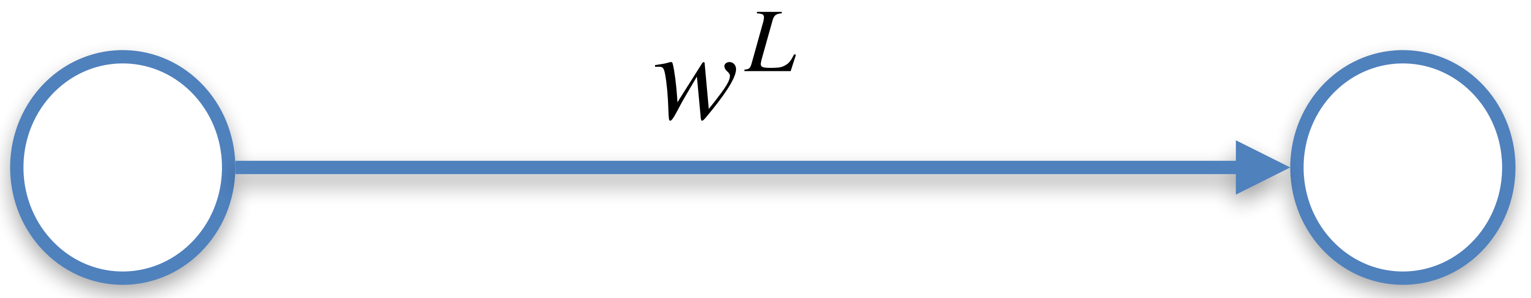
We are ready to update the initial values of the parameters! Let us set $\tau = 0.1$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} = 2(x^L - y)$$

$$w_1^L = w_0^L - \tau \nabla E(w_0^L)$$

Backpropagation algorithm



$$x^{L-1} = \frac{3}{2}$$

Desired output

$$y = \frac{1}{2}$$

We are ready to update the initial values of the parameters! Let us set $\tau = 0.1$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1}$$

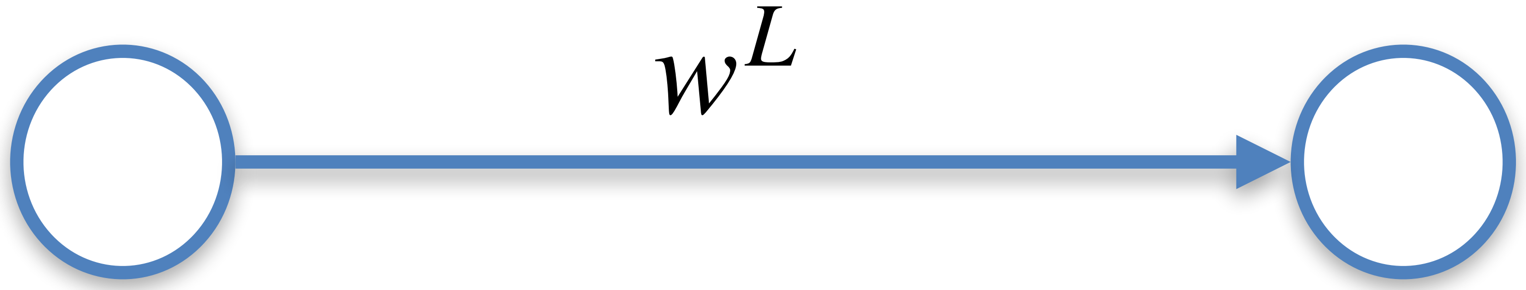
$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} = 2(x^L - y)$$

$$w_1^L = w_0^L - \tau \nabla E(w_0^L)$$

$$w_1^L = 0.8 - \frac{2}{10}\left(2.2 - \frac{1}{2}\right)\frac{3}{2} = 0.29$$

Backpropagation algorithm

w^L



$x^{L-1} = \frac{3}{2}$ $\qquad$ x^L

Desired output

$$y = \frac{1}{2}$$

We are ready to update the initial values of the parameters! Let us set $\tau = 0.1$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} = 2(x^L - y)$$

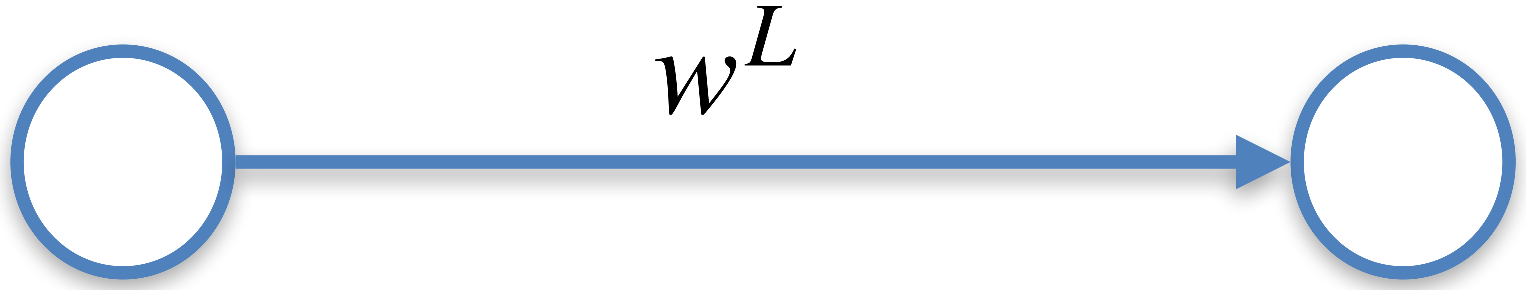
$$w_1^L = w_0^L - \tau \nabla E(w_0^L)$$

$$b_1^L = b_0^L - \tau \nabla E(b_0^L)$$

$$w_1^L = 0.8 - \frac{2}{10}\left(2.2 - \frac{1}{2}\right)\frac{3}{2} = 0.29$$

Backpropagation algorithm

w^L



$x^{L-1} = \frac{3}{2}$ $\qquad$ x^L

Desired output

$$y = \frac{1}{2}$$

We are ready to update the initial values of the parameters! Let us set $\tau = 0.1$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} = 2(x^L - y)$$

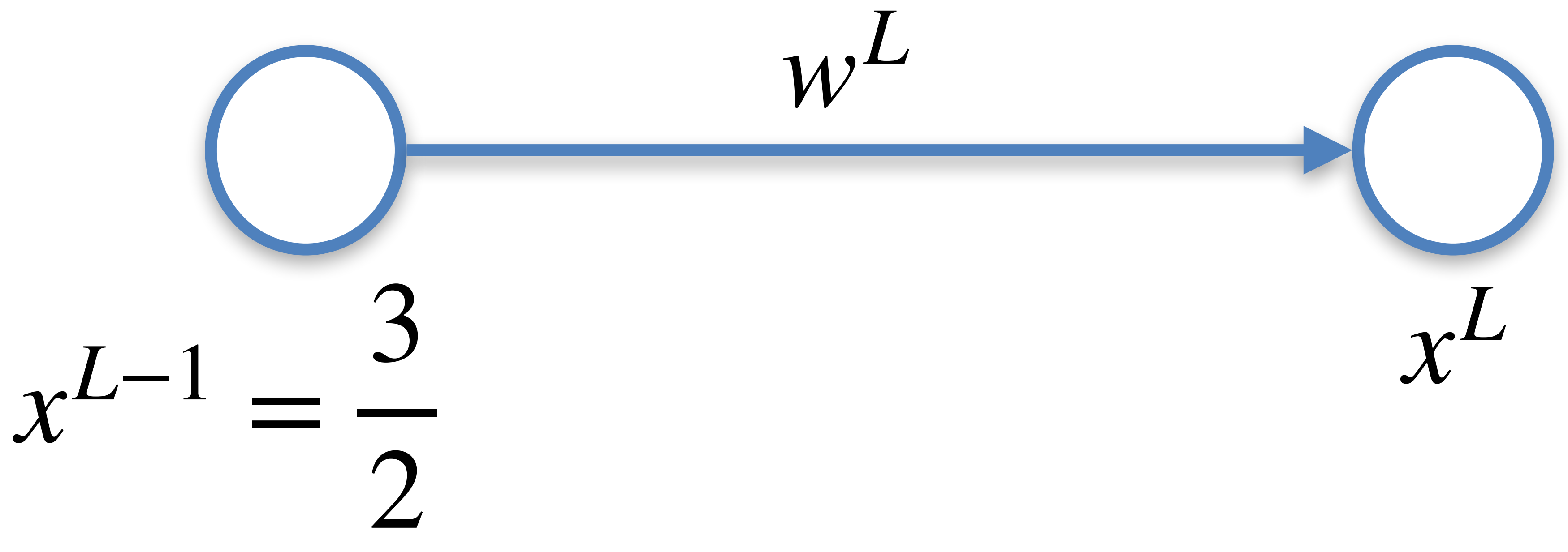
$$w_1^L = w_0^L - \tau \nabla E(w_0^L)$$

$$b_1^L = b_0^L - \tau \nabla E(b_0^L)$$

$$w_1^L = 0.8 - \frac{2}{10}\left(2.2 - \frac{1}{2}\right)\frac{3}{2} = 0.29$$

$$b_1^L = 1 - \frac{2}{10}\left(2.2 - \frac{1}{2}\right) = 0.66$$

Backpropagation algorithm



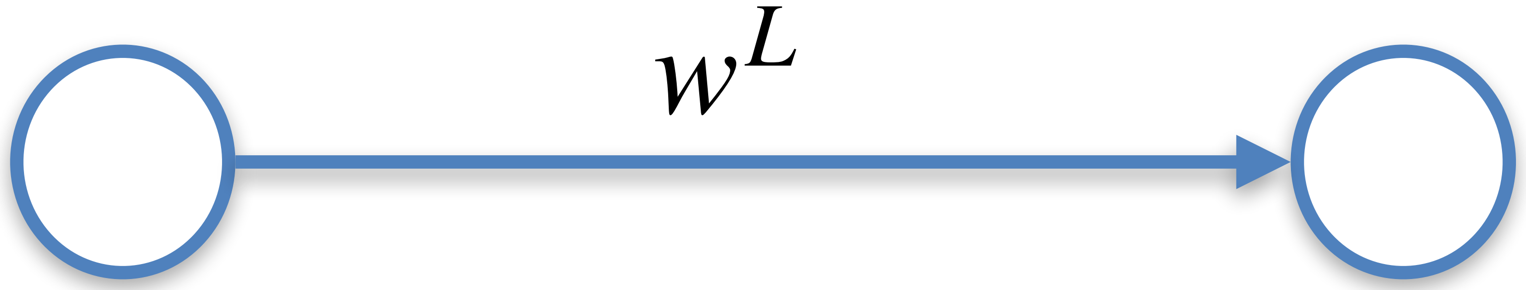
Desired output

$$y = \frac{1}{2}$$

We can progress with the calculation going to the next iteration

k	w_k^L	b_k^L	x^L
1	0.29	0.66	2.20
2	0.11	0.54	1.09
3	0.05	0.50	0.71
4	0.03	0.48	0.57
5	0.02	0.48	0.53
6	0.02	0.48	0.51
7	0.02	0.48	0.50

Backpropagation algorithm



w^L

$x^{L-1} = \frac{3}{2}$ $\quad$ x^L

Desired output

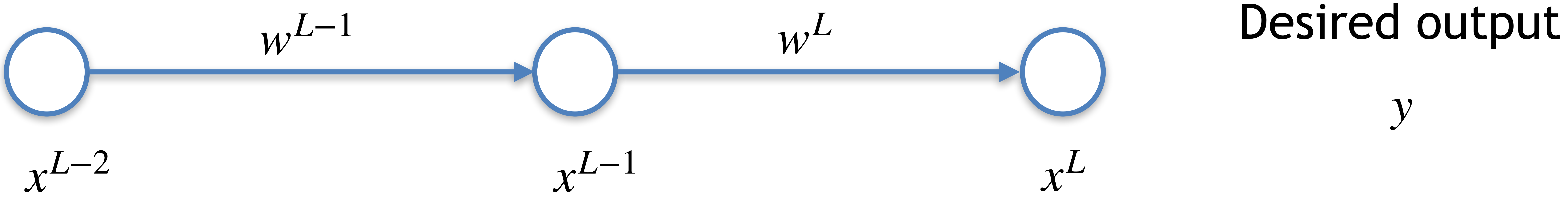
$$y = \frac{1}{2}$$

We can progress with the calculation going to the next iteration

k	w_k^L	b_k^L	x^L
1	0.29	0.66	2.20
2	0.11	0.54	1.09
3	0.05	0.50	0.71
4	0.03	0.48	0.57
5	0.02	0.48	0.53
6	0.02	0.48	0.51
7	0.02	0.48	0.50

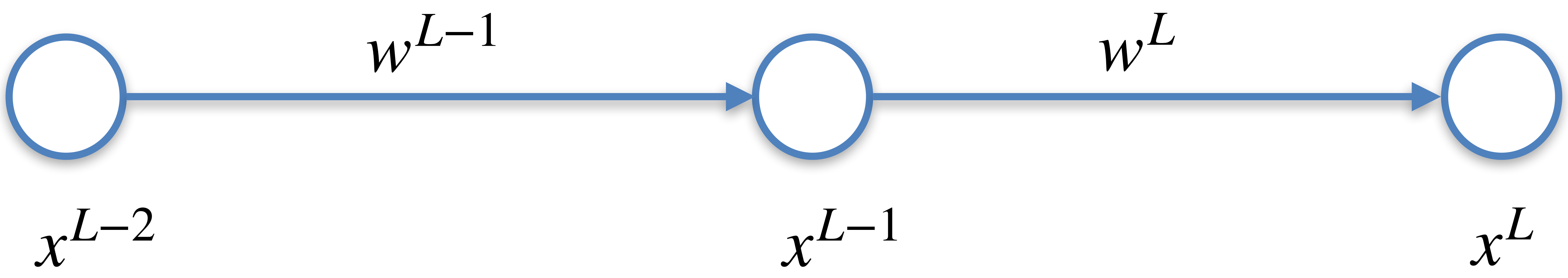
Backpropagation algorithm

What happens if we add a layer?



Backpropagation algorithm

What happens if we add a layer?



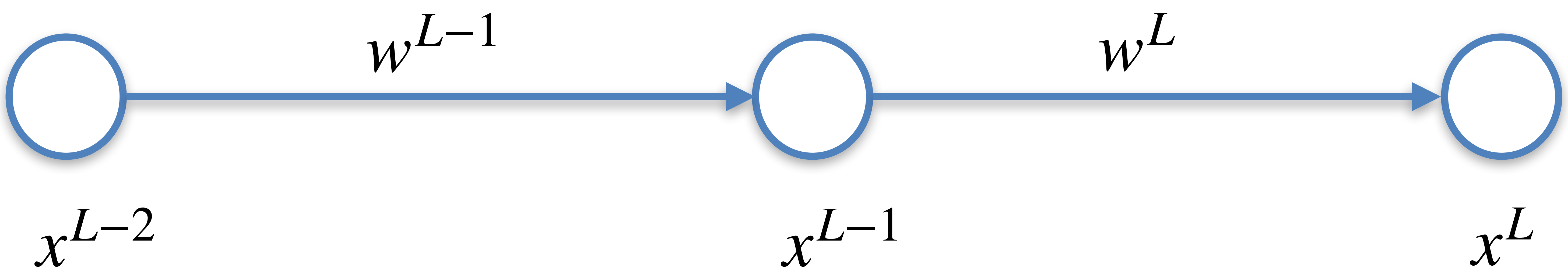
Desired output

y

$$x^L = w^L x^{L-1} + b^L$$

Backpropagation algorithm

What happens if we add a layer?



Desired output

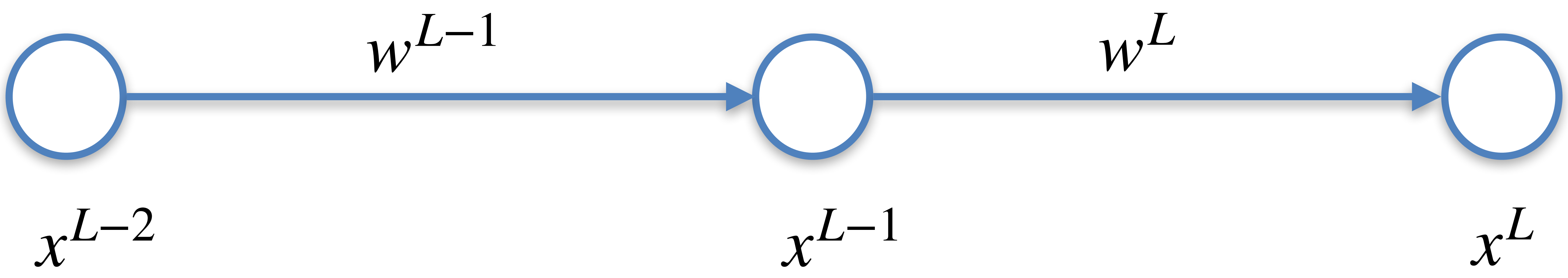
y

$$x^L = w^L x^{L-1} + b^L$$

$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

Backpropagation algorithm

What happens if we add a layer?



Desired output

y

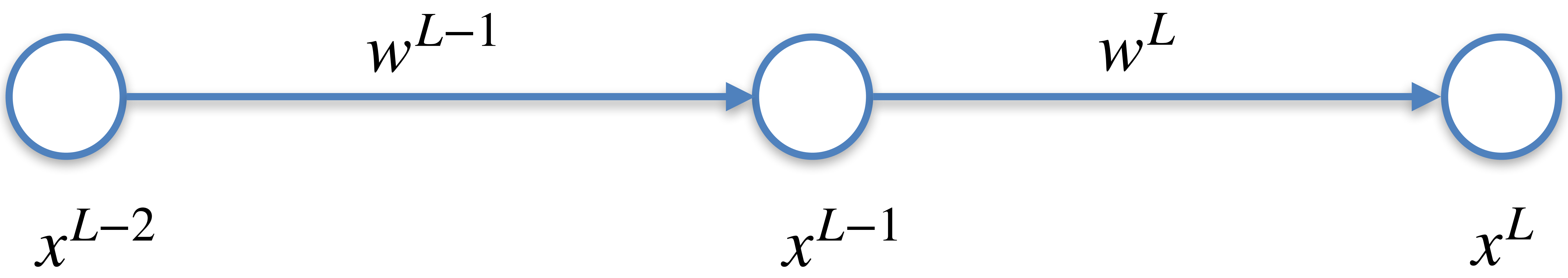
$$x^L = w^L x^{L-1} + b^L$$

$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

$$E = (x^L - y)^2$$

Backpropagation algorithm

What happens if we add a layer?



Desired output

y

Now we are dealing with 4 parameters!

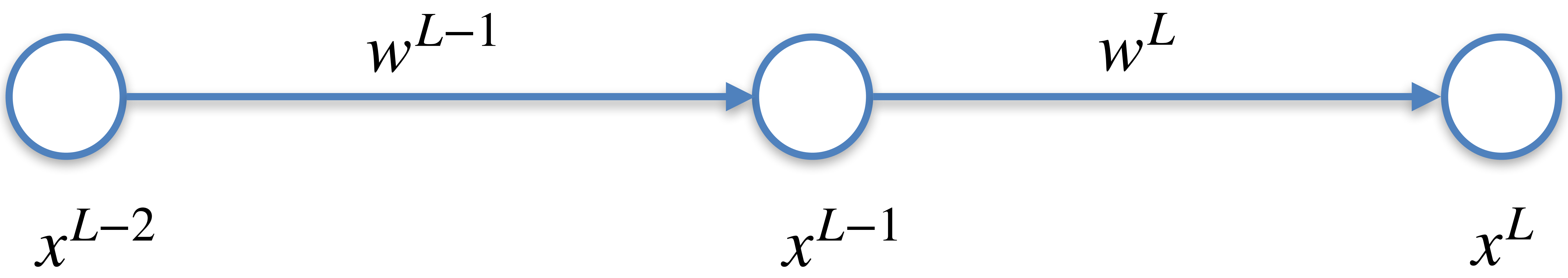
$$x^L = w^L x^{L-1} + b^L$$

$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

$$E = (x^L - y)^2$$

Backpropagation algorithm

What happens if we add a layer?



Desired output

y

Now we are dealing with 4 parameters!

$$x^L = w^L x^{L-1} + b^L$$

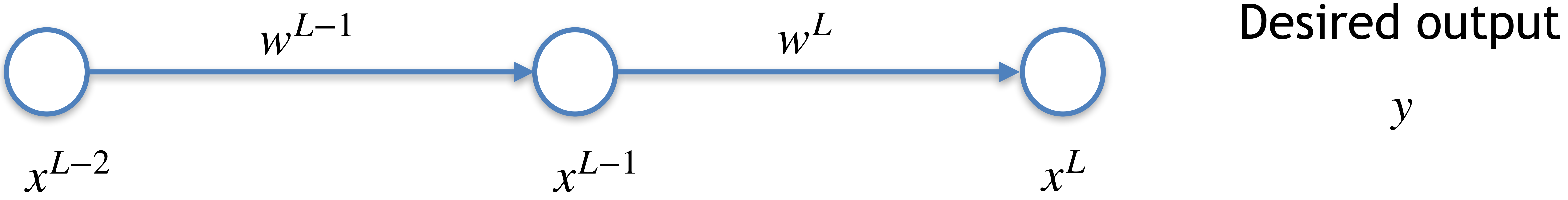
$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

$$E = (x^L - y)^2$$

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial w^{L-1}}, \frac{\partial E}{\partial b^L}, \frac{\partial E}{\partial b^{L-1}} \right)^\top$$

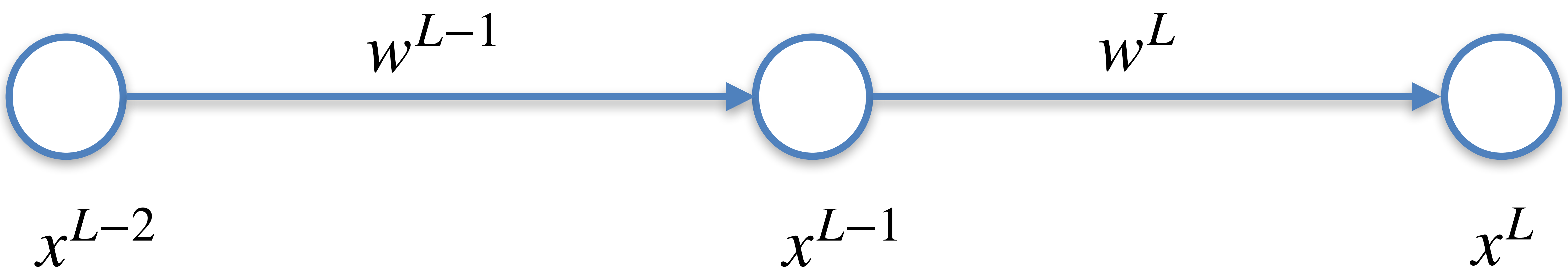
Backpropagation algorithm

What happens if we add a layer?



Backpropagation algorithm

What happens if we add a layer?



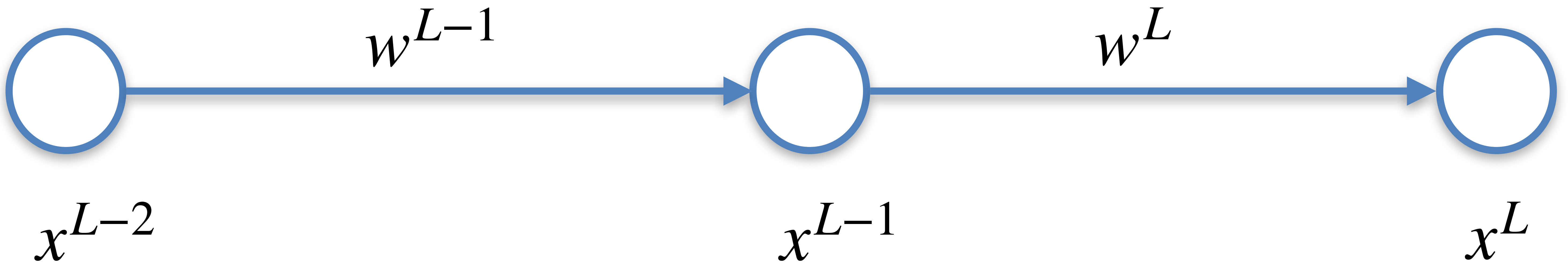
Desired output

y

Despite the number of parameters, the approach is exactly the same:

Backpropagation algorithm

What happens if we add a layer?



Desired output

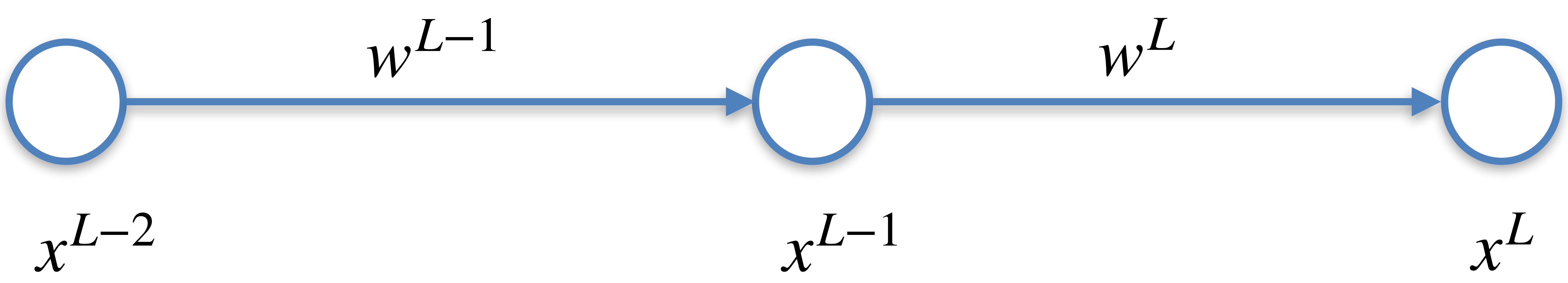
y

Despite the number of parameters, the approach is exactly the same:

1) forward pass to compute the values of the inputs/outputs

Backpropagation algorithm

What happens if we add a layer?



Desired output

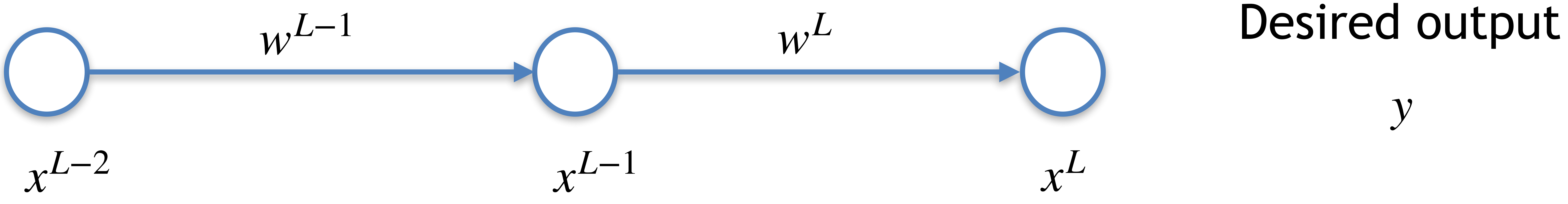
y

Despite the number of parameters, the approach is exactly the same:

1) forward pass to compute the values of the inputs/outputs
2) Backpropagation to get gradients

Backpropagation algorithm

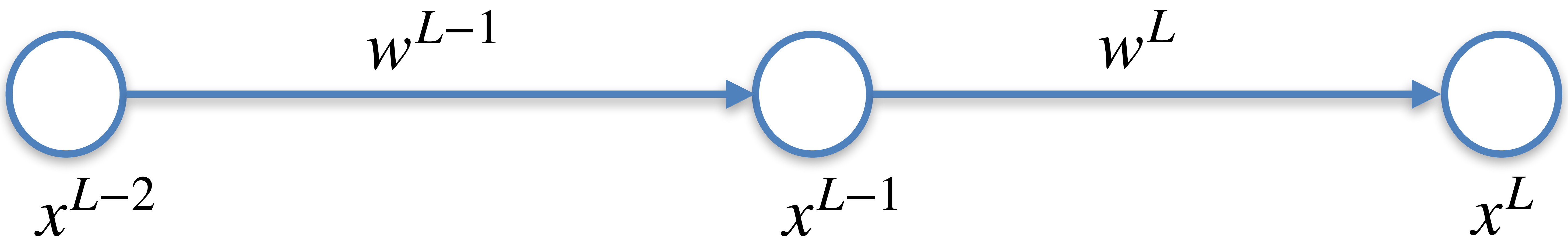
What happens if we add a layer?



Despite the number of parameters, the approach is exactly the same:

1) forward pass to compute the values of the inputs/outputs
2) Backpropagation to get gradients
3) Compute iteration of gradient descent

Backpropagation algorithm



Desired output

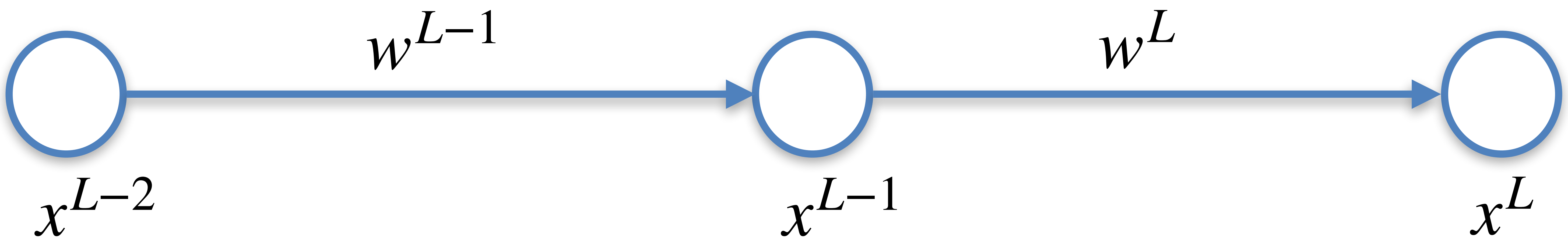
y

$$x^L = w^L x^{L-1} + b^L$$
$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

$$E = (x^L - y)^2$$

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial w^{L-1}}, \frac{\partial E}{\partial b^L}, \frac{\partial E}{\partial b^{L-1}} \right)^\top$$

Backpropagation algorithm



Desired output

y

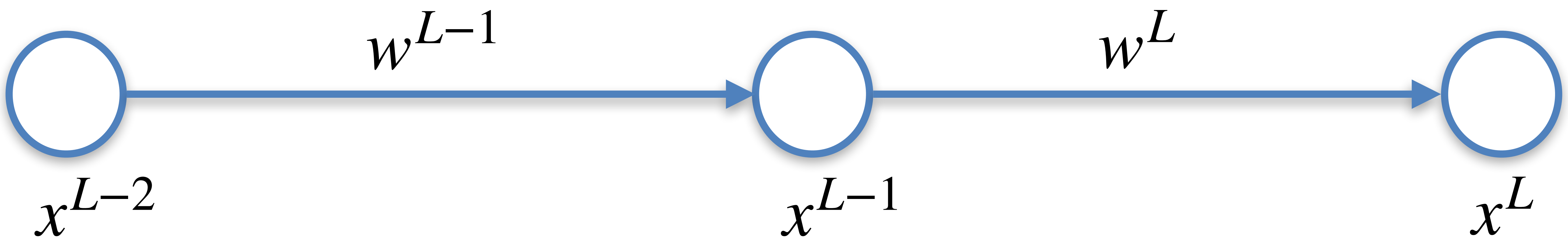
$$x^L = w^L x^{L-1} + b^L$$
$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

$$E = (x^L - y)^2$$

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial w^{L-1}}, \frac{\partial E}{\partial b^L}, \frac{\partial E}{\partial b^{L-1}} \right)^{\mathsf{T}}$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial w^L}$$

Backpropagation algorithm



Desired output

y

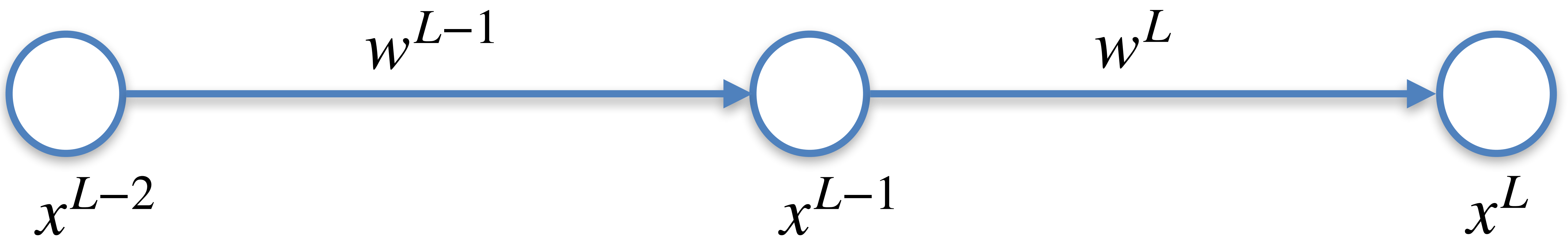
$$x^L = w^L x^{L-1} + b^L$$
$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

$$E = (x^L - y)^2$$

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial w^{L-1}}, \frac{\partial E}{\partial b^L}, \frac{\partial E}{\partial b^{L-1}} \right)^{\mathsf{T}}$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial w^L} \quad = 2(x^L - y) x^{L-1}$$

Backpropagation algorithm



Desired output

y

$$x^L = w^L x^{L-1} + b^L$$

$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

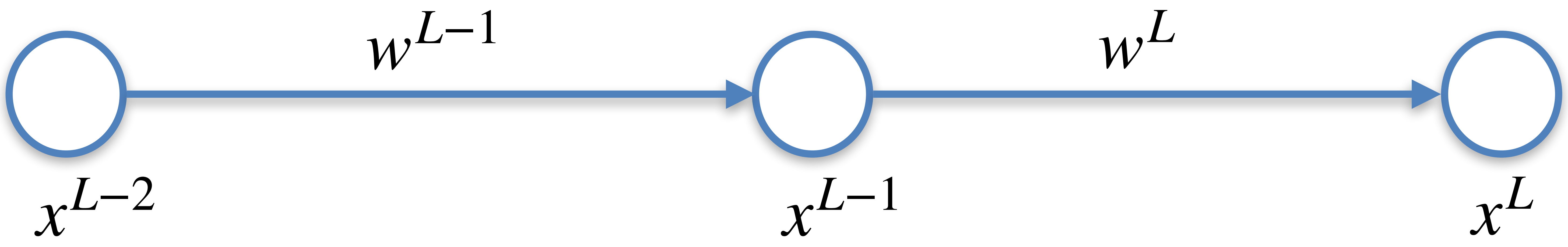
$$E = (x^L - y)^2$$

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial w^{L-1}}, \frac{\partial E}{\partial b^L}, \frac{\partial E}{\partial b^{L-1}} \right)^\top$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial w^{L-1}} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial x^{L-1}} \frac{\partial x^{L-1}}{\partial w^{L-1}}$$

Backpropagation algorithm



Desired output

y

$$x^L = w^L x^{L-1} + b^L$$
$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

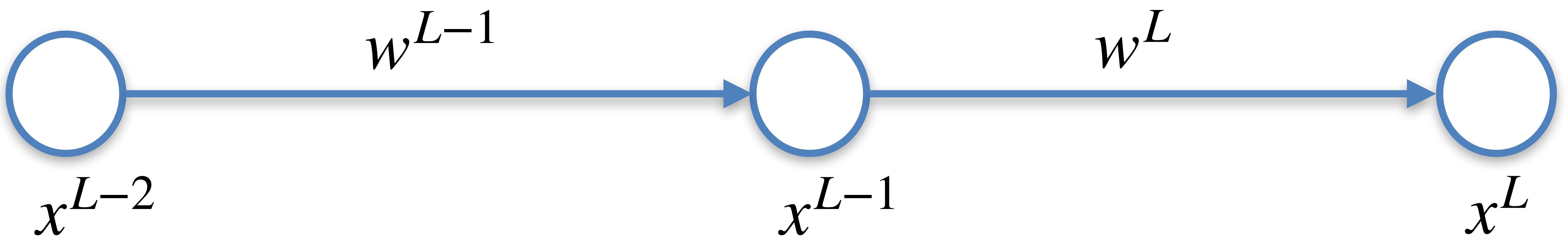
$$E = (x^L - y)^2$$

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial w^{L-1}}, \frac{\partial E}{\partial b^L}, \frac{\partial E}{\partial b^{L-1}} \right)^{\mathsf{T}}$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial w^L} = 2(x^L - y) x^{L-1}$$

$$\frac{\partial E}{\partial w^{L-1}} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial x^{L-1}} \frac{\partial x^{L-1}}{\partial w^{L-1}} = 2(x^L - y) w^L x^{L-2}$$

Backpropagation algorithm



Desired output

y

$$x^L = w^L x^{L-1} + b^L$$
$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

$$E = (x^L - y)^2$$

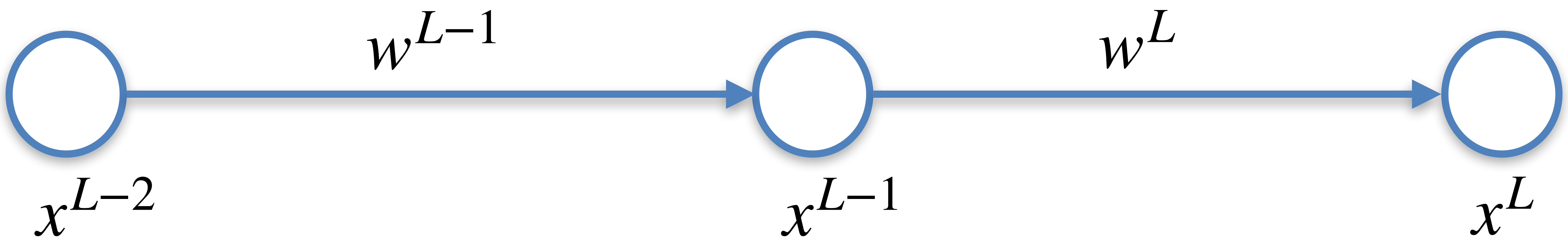
$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial w^{L-1}}, \frac{\partial E}{\partial b^L}, \frac{\partial E}{\partial b^{L-1}} \right)^{\mathsf{T}}$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial w^{L-1}} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial x^{L-1}}\frac{\partial x^{L-1}}{\partial w^{L-1}} = 2(x^L - y)w^L x^{L-2}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L}$$

Backpropagation algorithm



Desired output

y

$$x^L = w^L x^{L-1} + b^L$$
$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

$$E = (x^L - y)^2$$

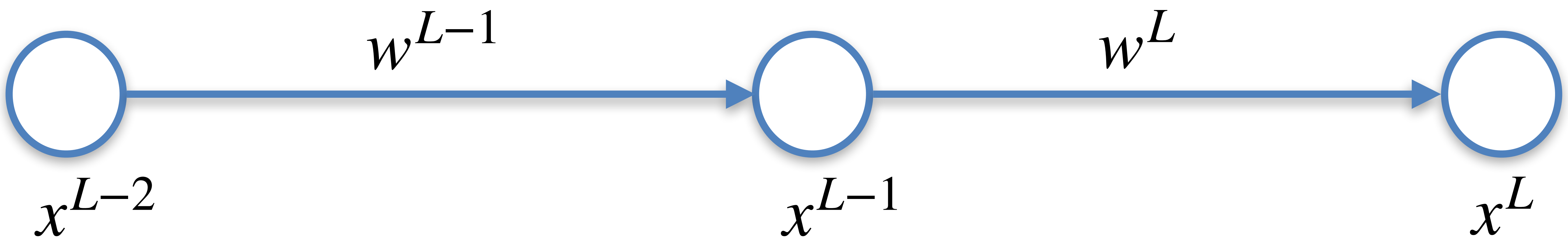
$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial w^{L-1}}, \frac{\partial E}{\partial b^L}, \frac{\partial E}{\partial b^{L-1}} \right)^\top$$

$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial w^{L-1}} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial x^{L-1}} \frac{\partial x^{L-1}}{\partial w^{L-1}} = 2(x^L - y)w^L x^{L-2}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial b^L} = 2(x^L - y)$$

Backpropagation algorithm



Desired output

y

$$x^L = w^L x^{L-1} + b^L$$
$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

$$E = (x^L - y)^2$$

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial w^{L-1}}, \frac{\partial E}{\partial b^L}, \frac{\partial E}{\partial b^{L-1}} \right)^{\mathsf{T}}$$

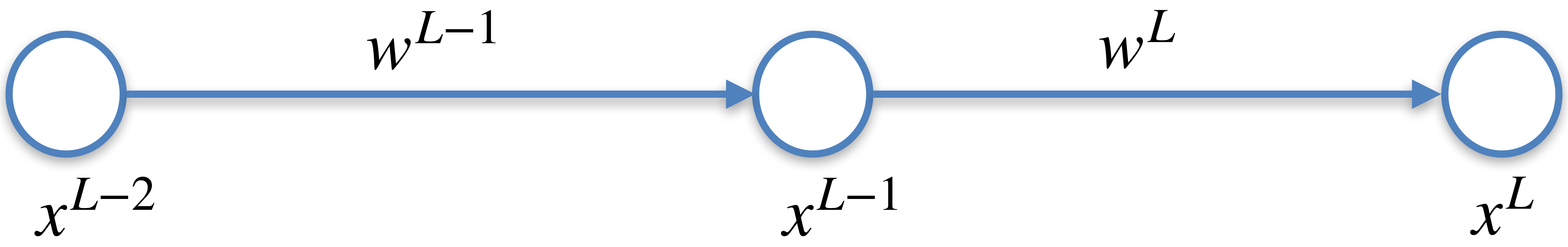
$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial b^L} = 2(x^L - y)$$

$$\frac{\partial E}{\partial w^{L-1}} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial x^{L-1}} \frac{\partial x^{L-1}}{\partial w^{L-1}} = 2(x^L - y)w^L x^{L-2}$$

$$\frac{\partial E}{\partial b^{L-1}} = \frac{\partial E}{\partial x^L} \frac{\partial x^L}{\partial x^{L-1}} \frac{\partial x^{L-1}}{\partial b^{L-1}}$$

Backpropagation algorithm



Desired output

y

$$x^L = w^L x^{L-1} + b^L$$
$$x^{L-1} = w^{L-1} x^{L-2} + b^{L-1}$$

$$E = (x^L - y)^2$$

$$\nabla E = \left(\frac{\partial E}{\partial w^L}, \frac{\partial E}{\partial w^{L-1}}, \frac{\partial E}{\partial b^L}, \frac{\partial E}{\partial b^{L-1}} \right)^{\top}$$

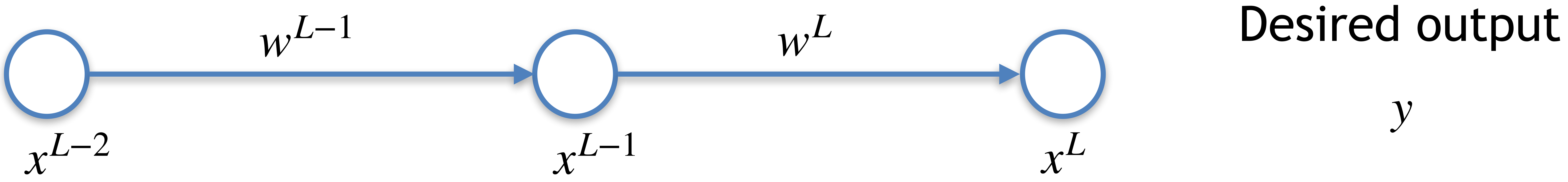
$$\frac{\partial E}{\partial w^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial w^L} = 2(x^L - y)x^{L-1}$$

$$\frac{\partial E}{\partial b^L} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial b^L} = 2(x^L - y)$$

$$\frac{\partial E}{\partial w^{L-1}} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial x^{L-1}}\frac{\partial x^{L-1}}{\partial w^{L-1}} = 2(x^L - y)w^L x^{L-2}$$

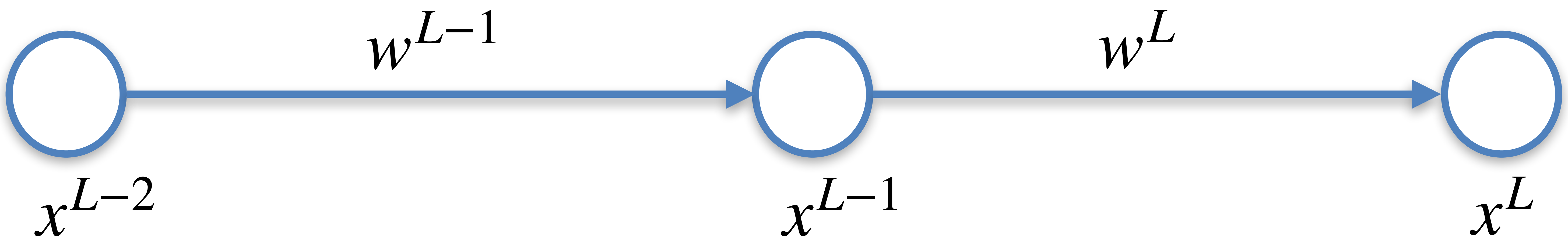
$$\frac{\partial E}{\partial b^{L-1}} = \frac{\partial E}{\partial x^L}\frac{\partial x^L}{\partial x^{L-1}}\frac{\partial x^{L-1}}{\partial b^{L-1}} = 2(x^L - y)w^L$$

Backpropagation algorithm



We can now perform gradient descent!

Backpropagation algorithm



Desired output

y

We can now perform gradient descent!

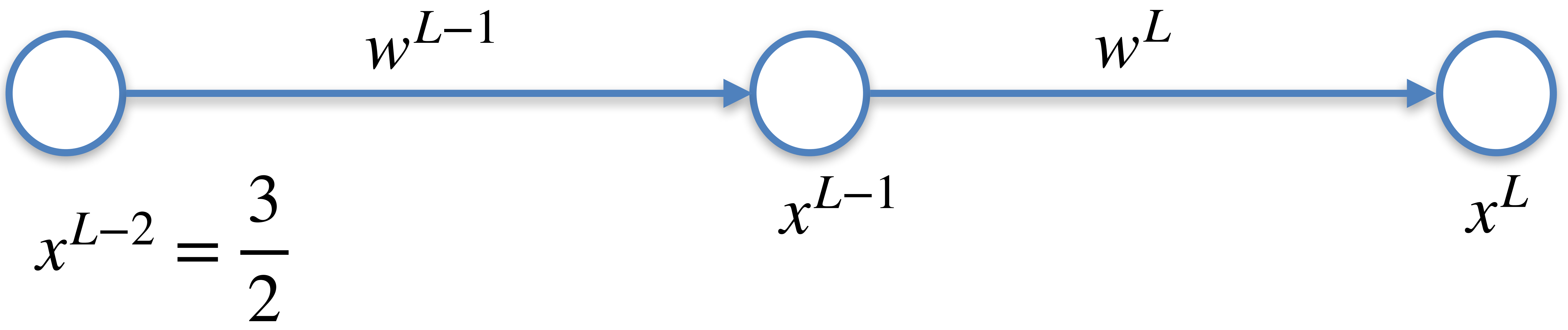
$$w_1^L = w_0^L - \tau \nabla E(w_0^L)$$

$$b_1^L = b_0^L - \tau \nabla E(b_0^L)$$

$$w_1^{L-1} = w_0^{L-1} - \tau \nabla E(w_0^{L-1})$$

$$b_1^{L-1} = b_0^{L-1} - \tau \nabla E(b_0^{L-1})$$

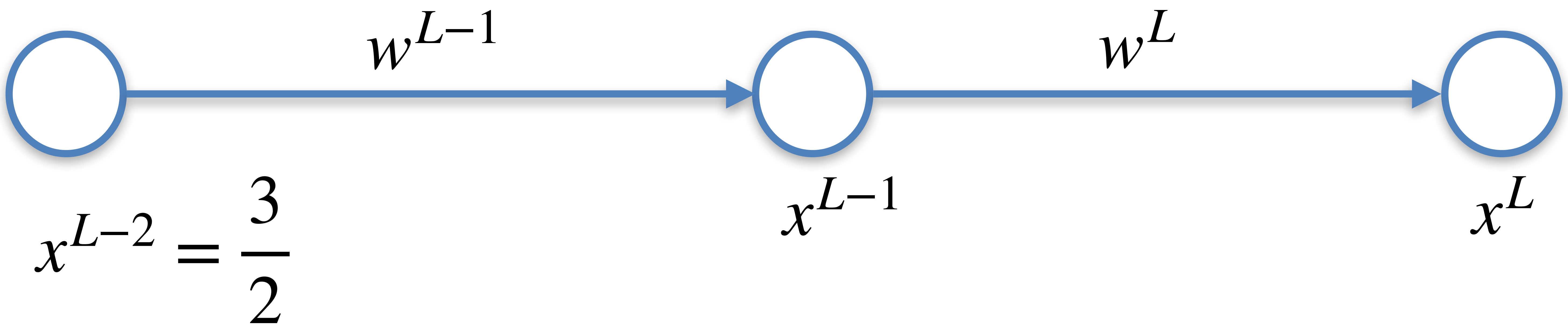
Backpropagation algorithm



Desired output

$$y = \frac{1}{2}$$

Backpropagation algorithm

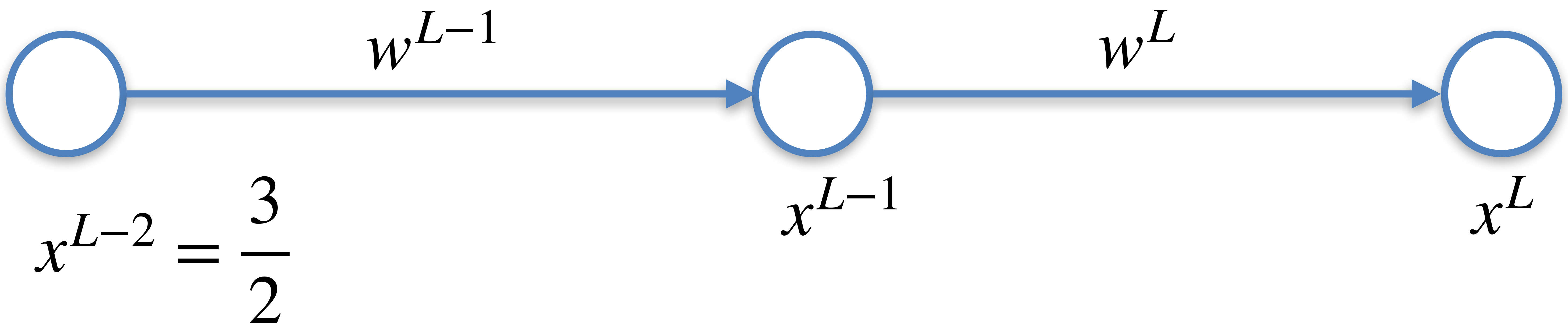


Desired output

$$y = \frac{1}{2}$$

Initialising $\quad w_0^{L-1} = 0.8,\ w_0^{L} = 0.8,\ b_0^{L-1} = 1,\ b_0^{L} = 1,\ \tau = \frac{1}{10}$

Backpropagation algorithm



Desired output

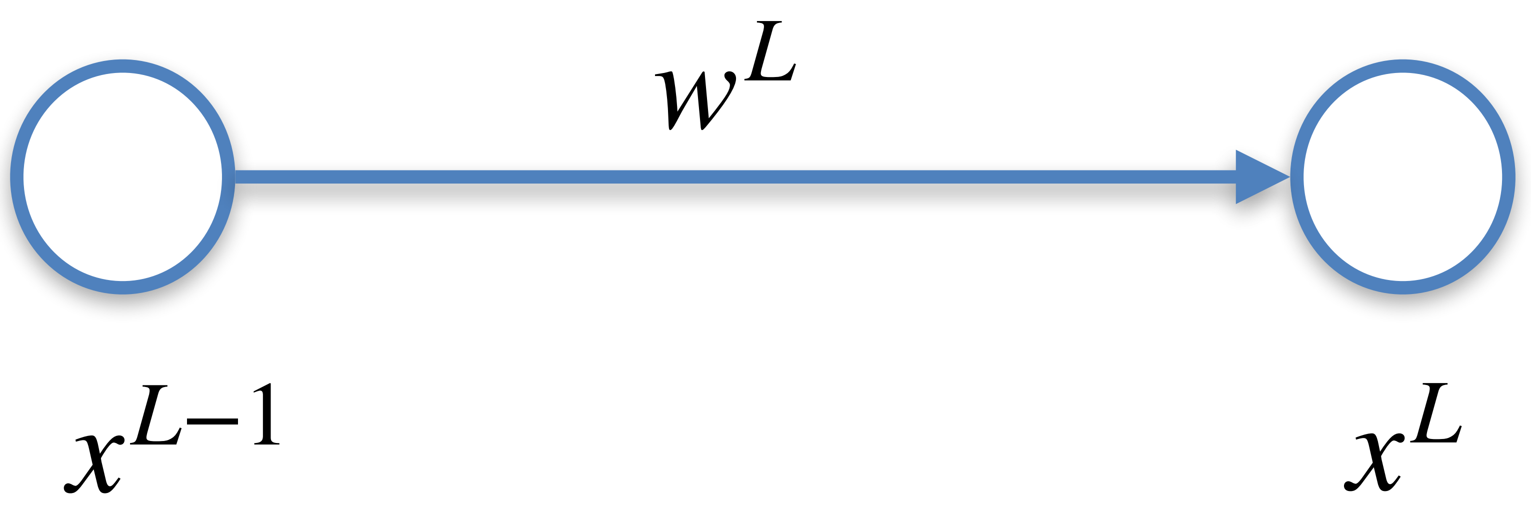
$$y = \frac{1}{2}$$

Initialising $\quad w_0^{L-1} = 0.8,\ w_0^{L} = 0.8,\ b_0^{L-1} = 1,\ b_0^{L} = 1,\ \tau = \frac{1}{10}$

k	w_k^L	b_k^L	w_k^{L-1}	b_k^{L-1}	x^L
1	−0.19	0.55	0.26	0.64	0.35
2	−0.16	0.58	0.25	0.63	0.41
3	−0.15	0.60	0.24	0.63	0.45
4	−0.14	0.61	0.24	0.63	0.47
5	−0.13	0.61	0.24	0.63	0.48
6	−0.13	0.61	0.24	0.63	0.49
7	−0.12	0.62	0.24	0.63	0.49
8	−0.12	0.62	0.24	0.63	0.50

Backpropagation algorithm

We can add the activation function

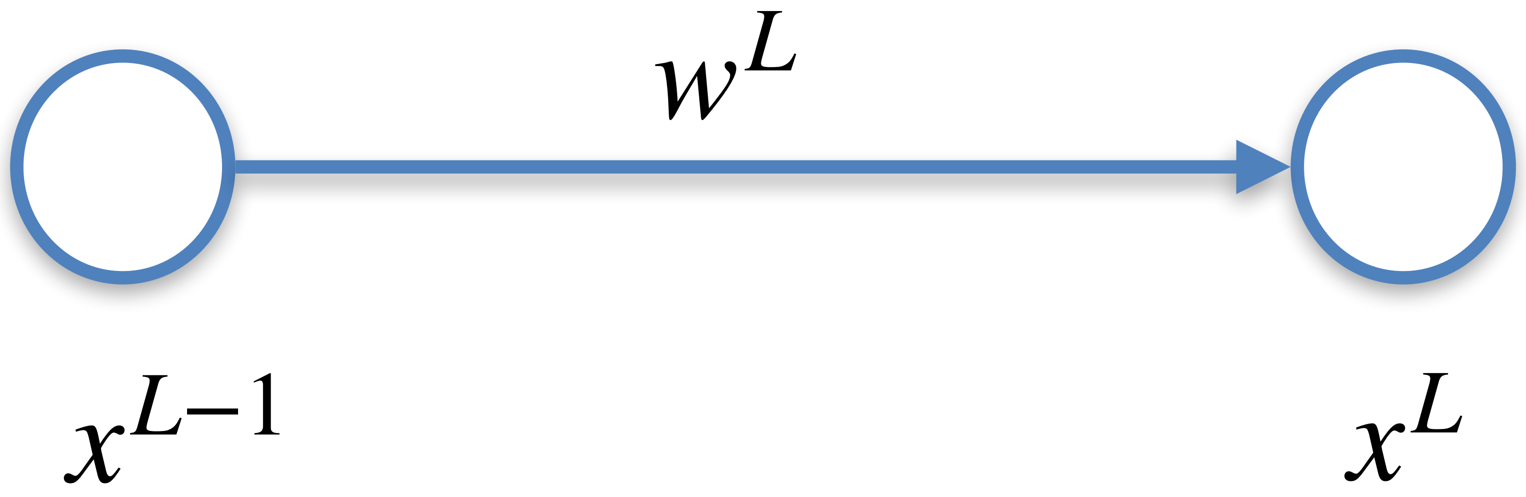


Desired output

y

Backpropagation algorithm

We can add the activation function



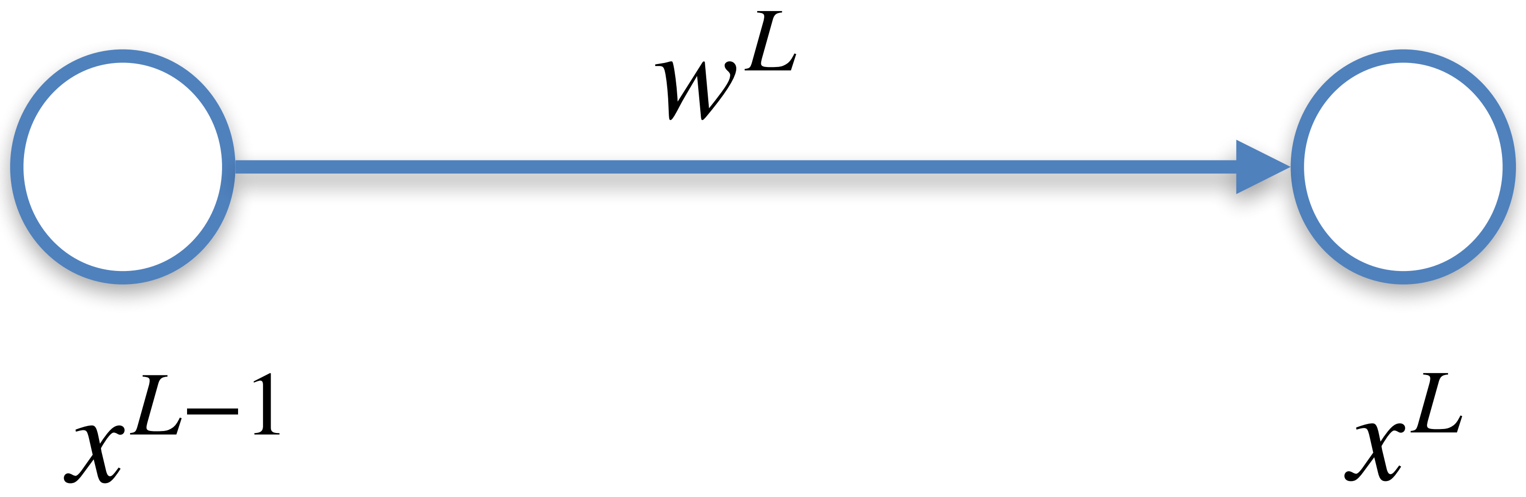
Desired output

y

$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$

Backpropagation algorithm

We can add the activation function



Desired output

y

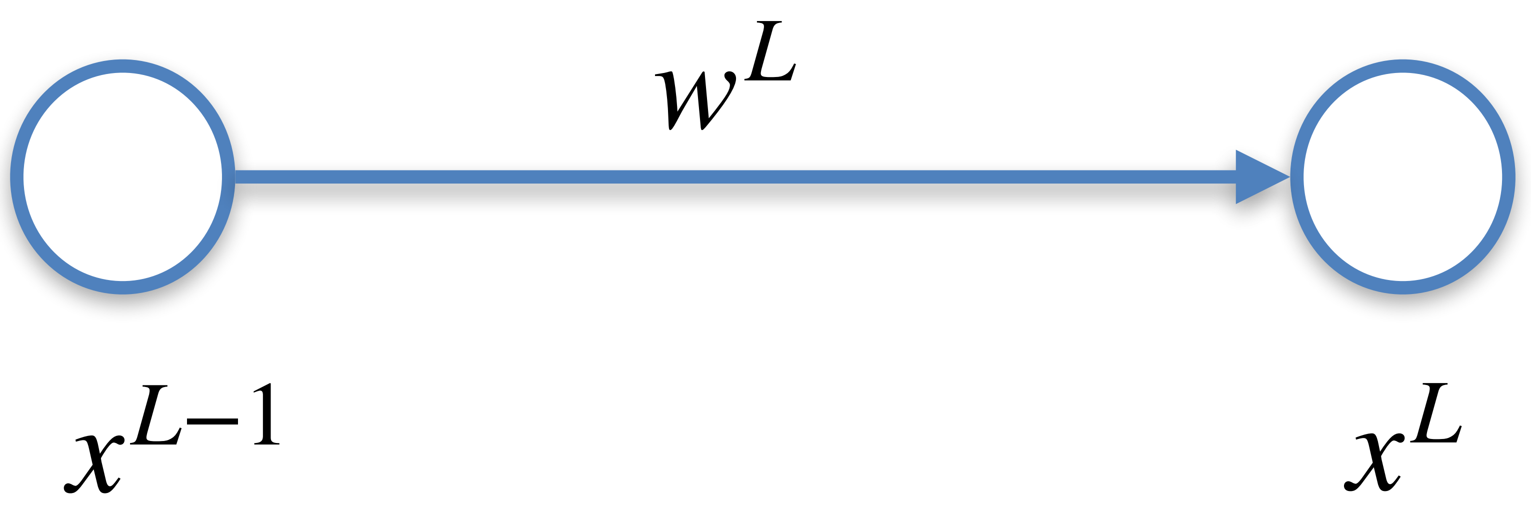
$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$

$$z^L = w^L x^{L-1} + b^L$$

Credits: https://www.youtube.com/watch?v=tIeHLnjs5U8

Backpropagation algorithm

We can add the activation function



Desired output

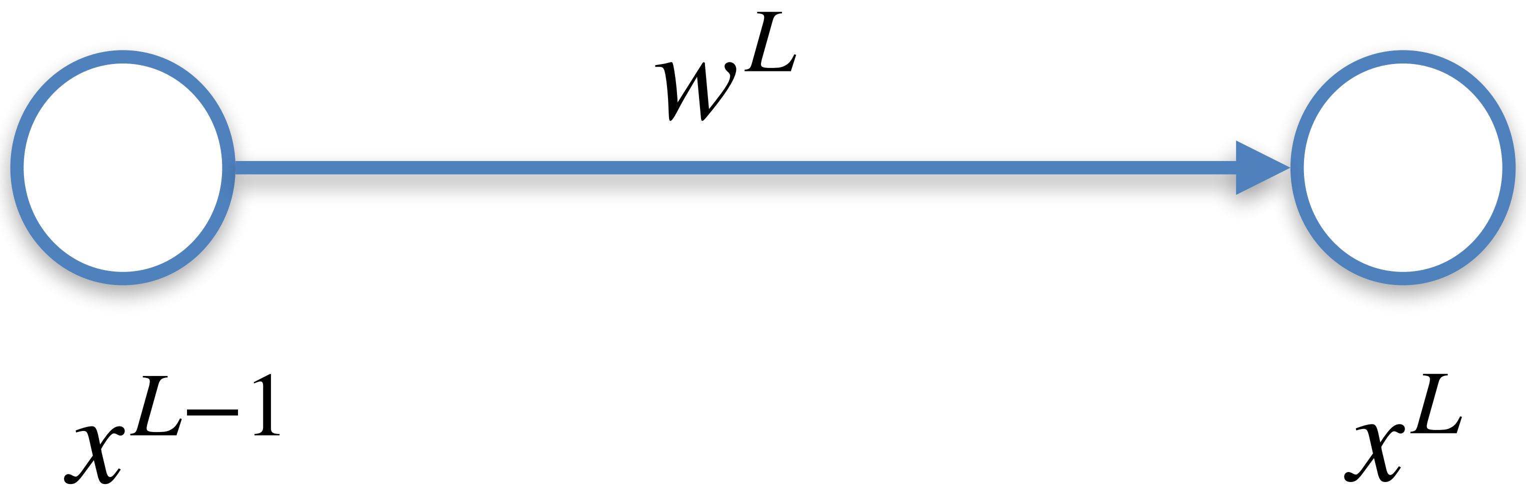
y

$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$

$$z^L = w^L x^{L-1} + b^L$$

Cost function for a single sample $$E_0 = (x^L - y)^2$$

Backpropagation algorithm



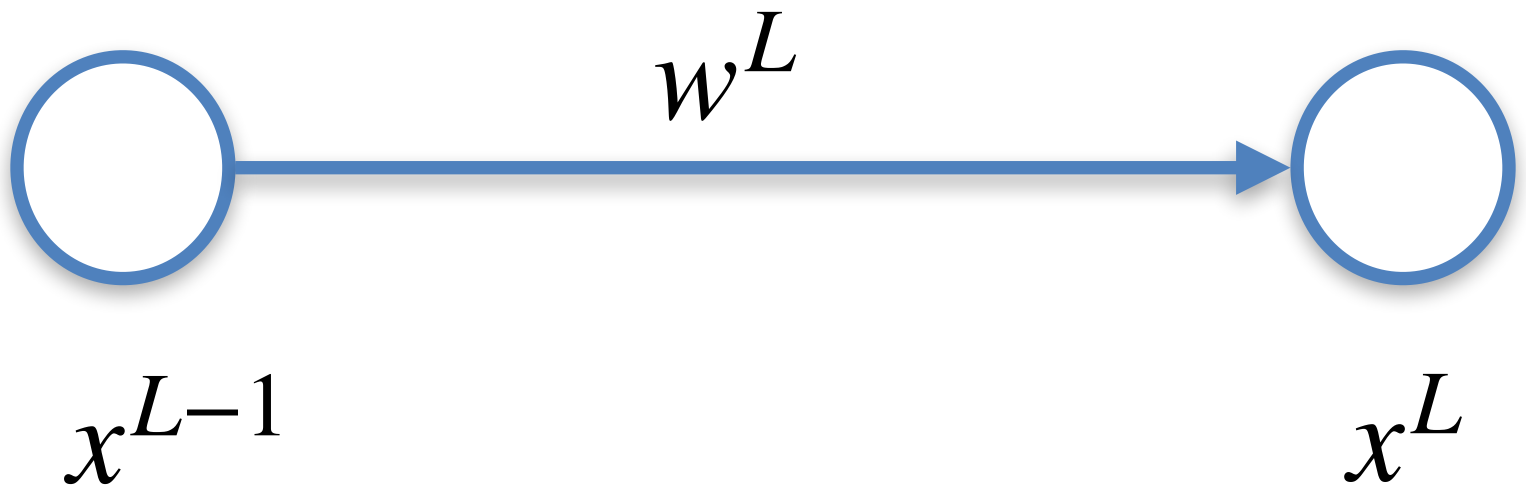
Desired output

y

$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$

$$E_0 = (x^L - y)^2$$

Backpropagation algorithm



Desired output

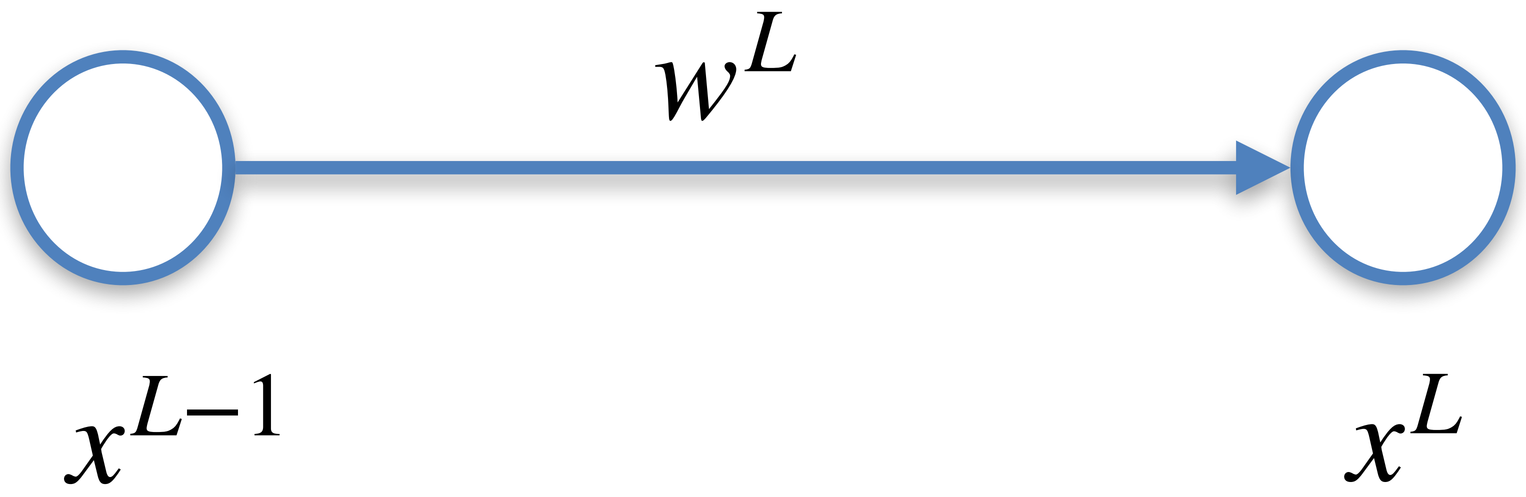
y

$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$

$$E_0 = (x^L - y)^2$$

E_0

Backpropagation algorithm

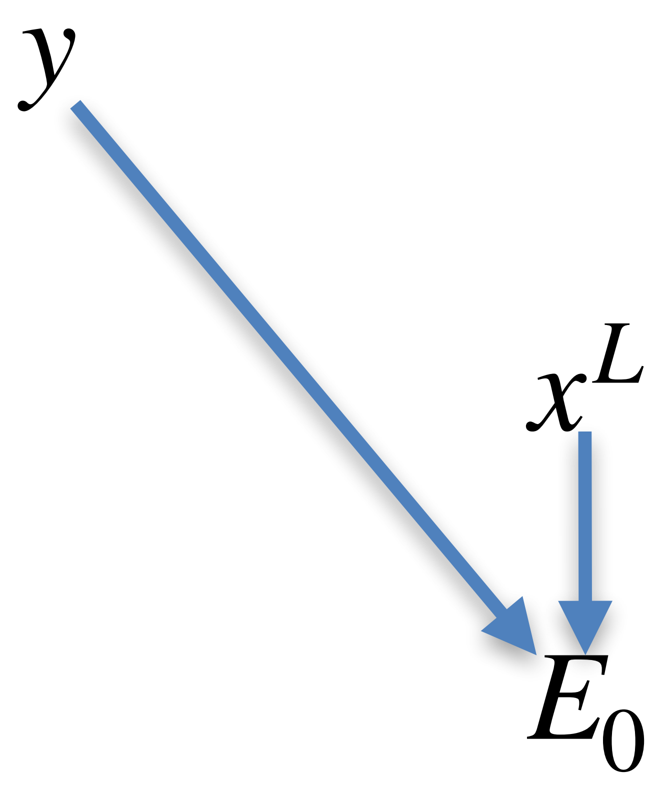


Desired output

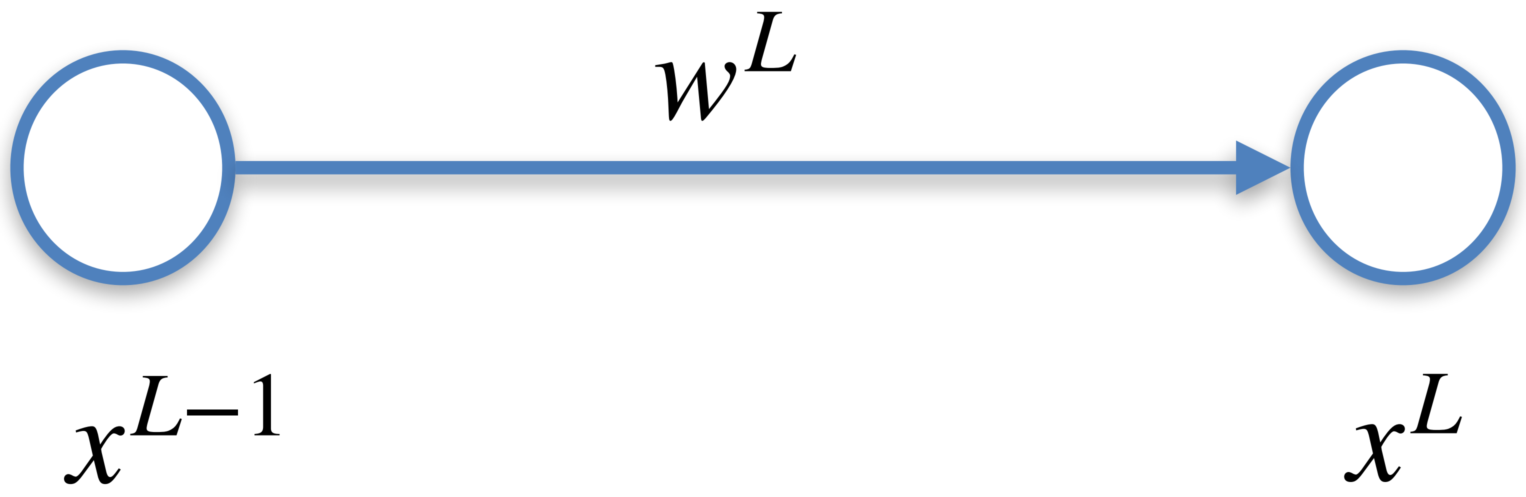
y

$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$

$$E_0 = (x^L - y)^2$$



Backpropagation algorithm

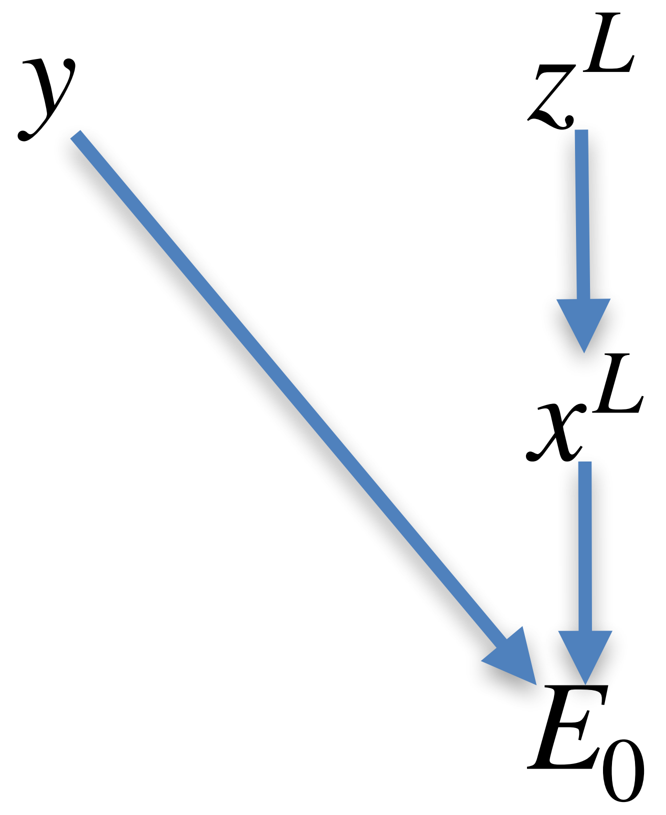


Desired output

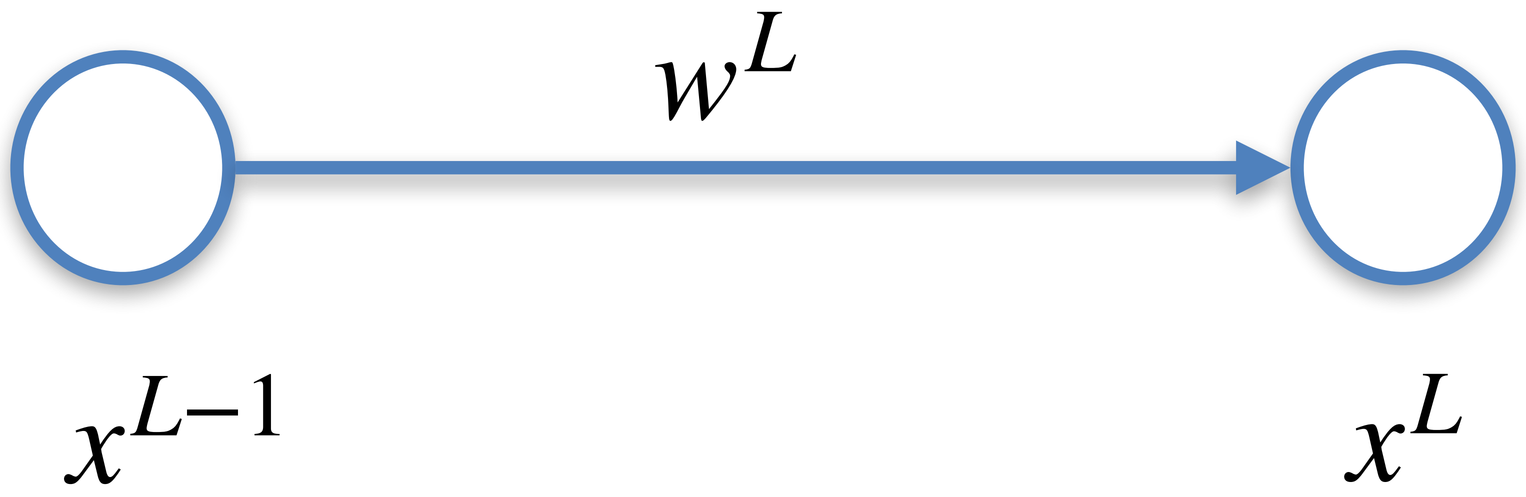
y

$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$

$$E_0 = (x^L - y)^2$$



Backpropagation algorithm

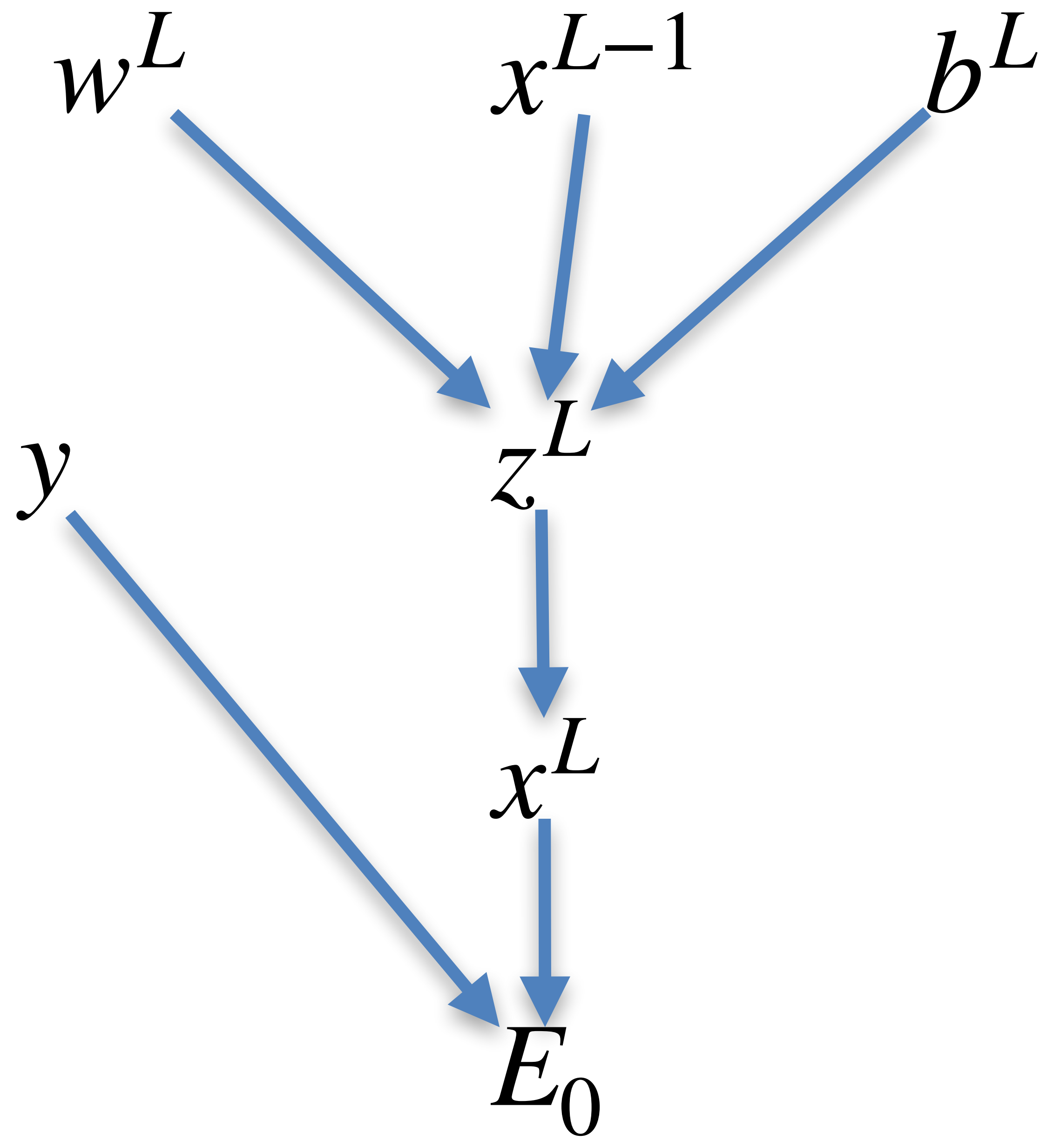


Desired output

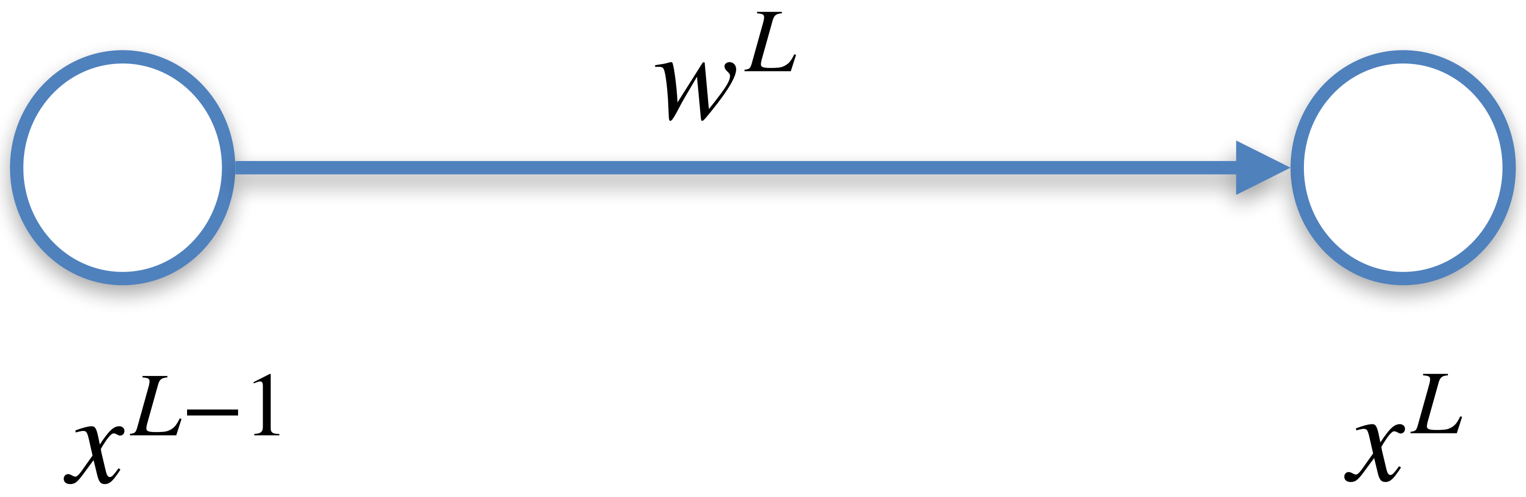
y

$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$

$$E_0 = (x^L - y)^2$$

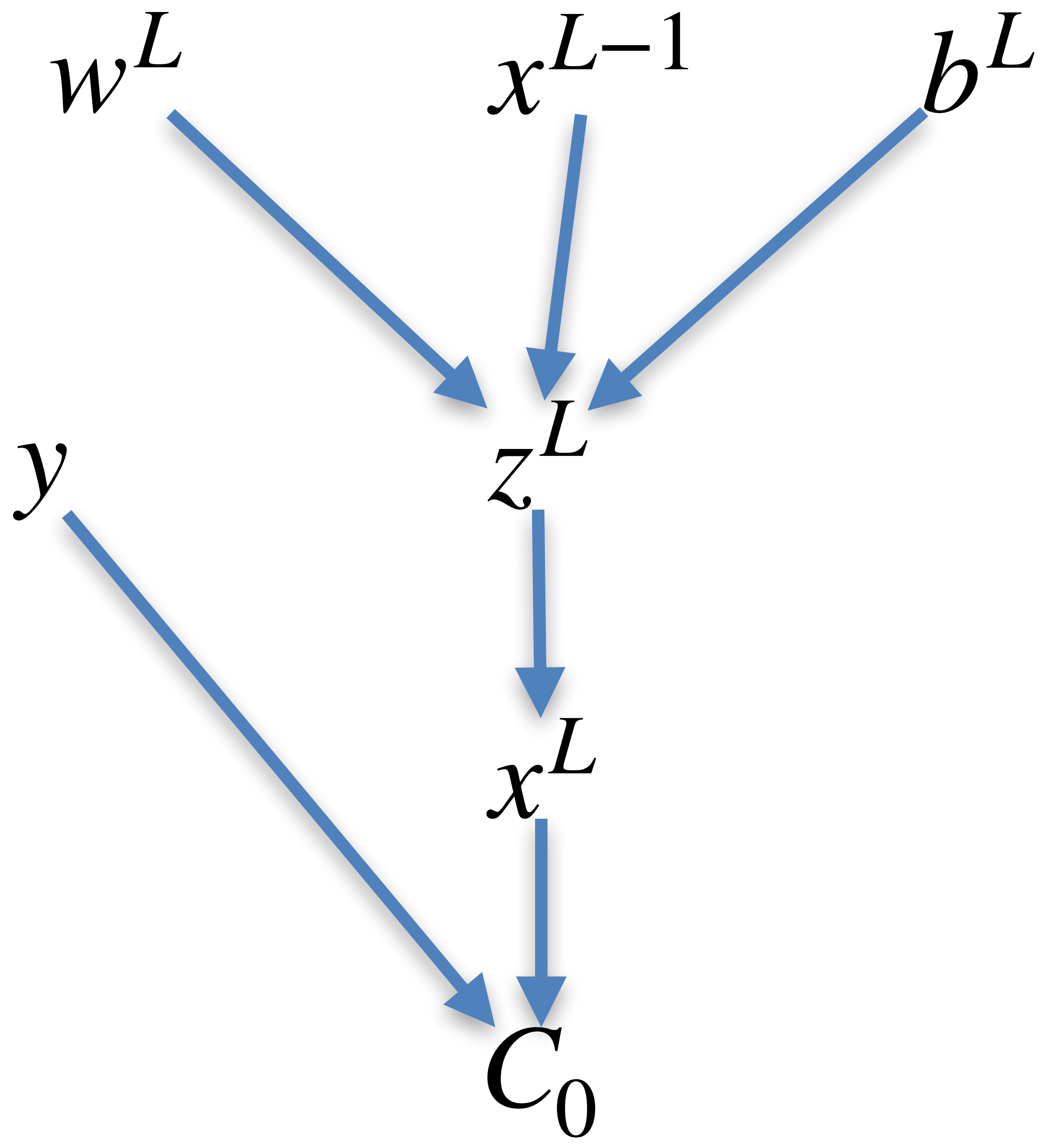


Backpropagation algorithm



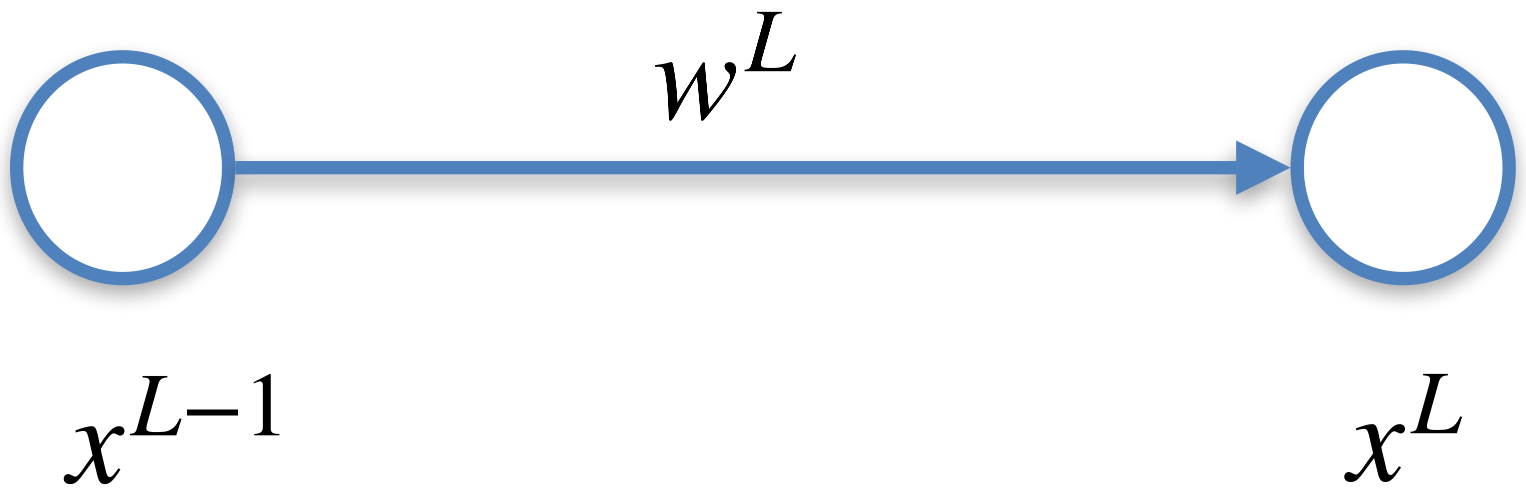
Desired output

y



Credits: https://www.youtube.com/watch?v=tIeHLnjs5U8

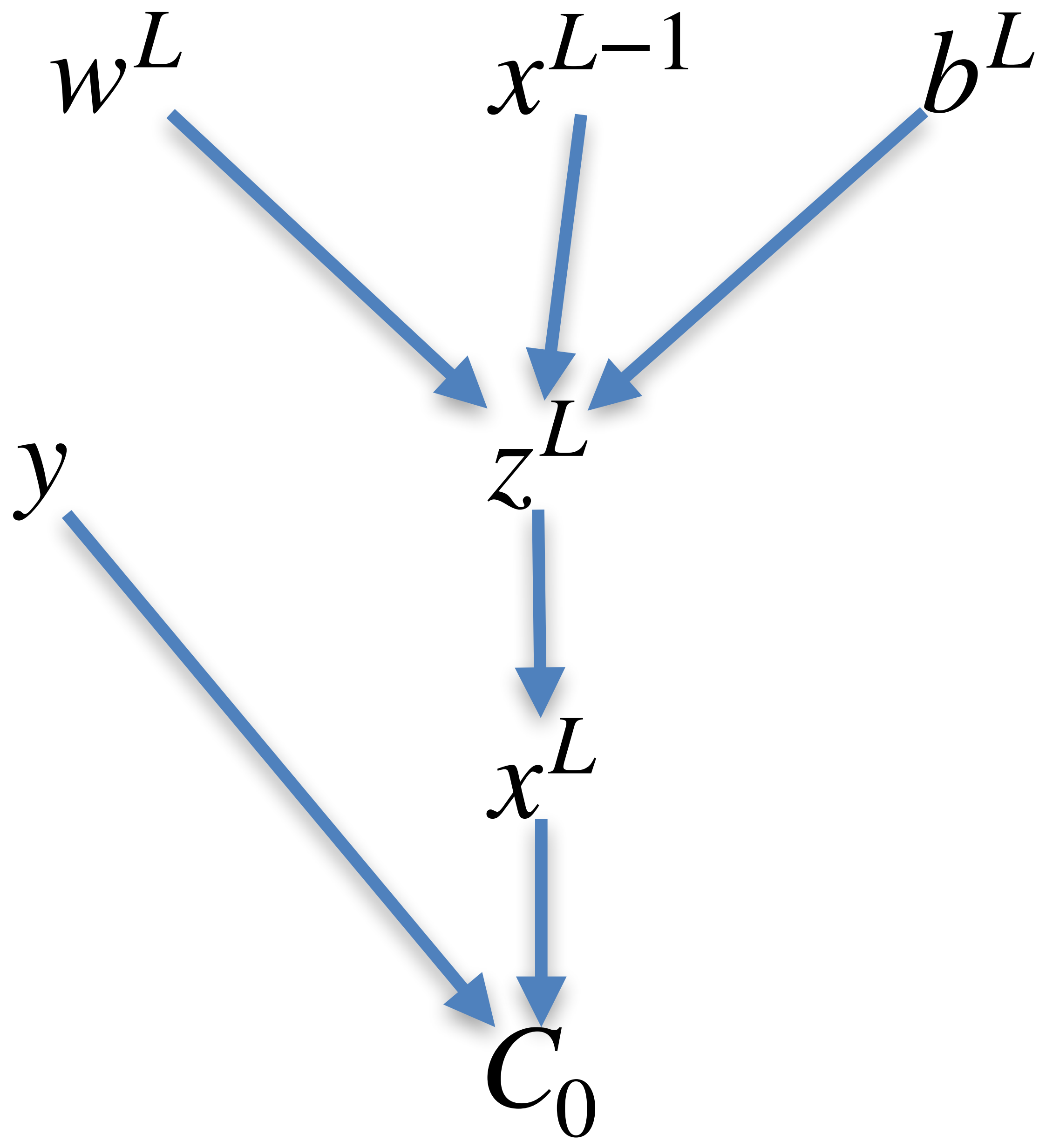
Backpropagation algorithm



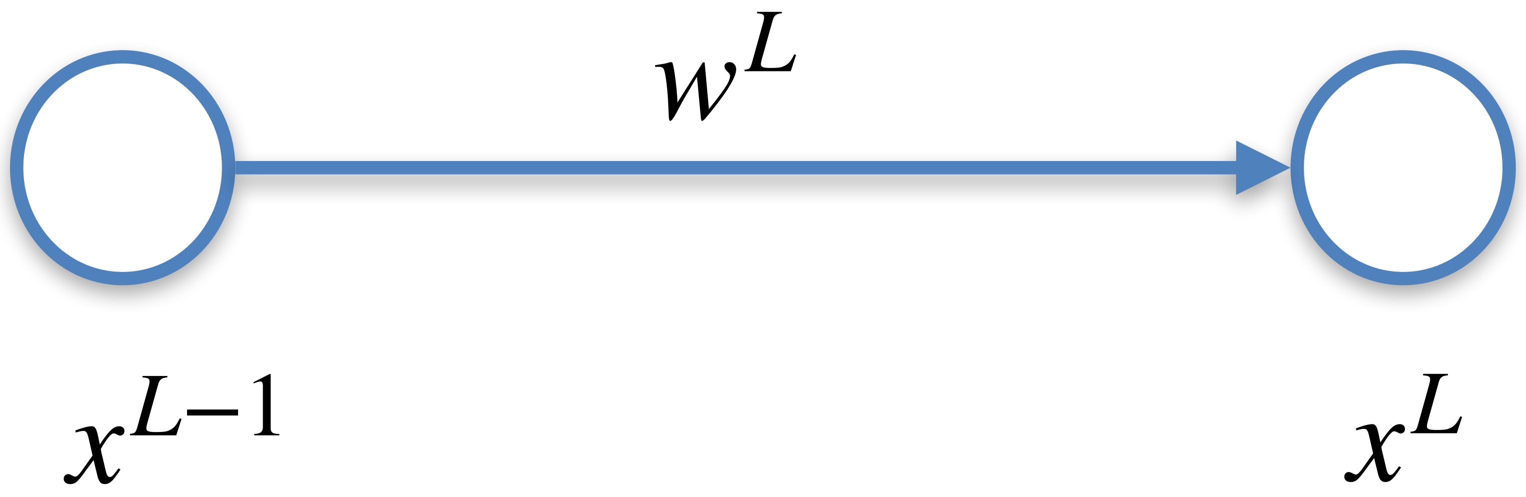
Desired output

y

To apply gradient descent we need to compute $\frac{\partial}{\partial w^L} E_0$



Backpropagation algorithm

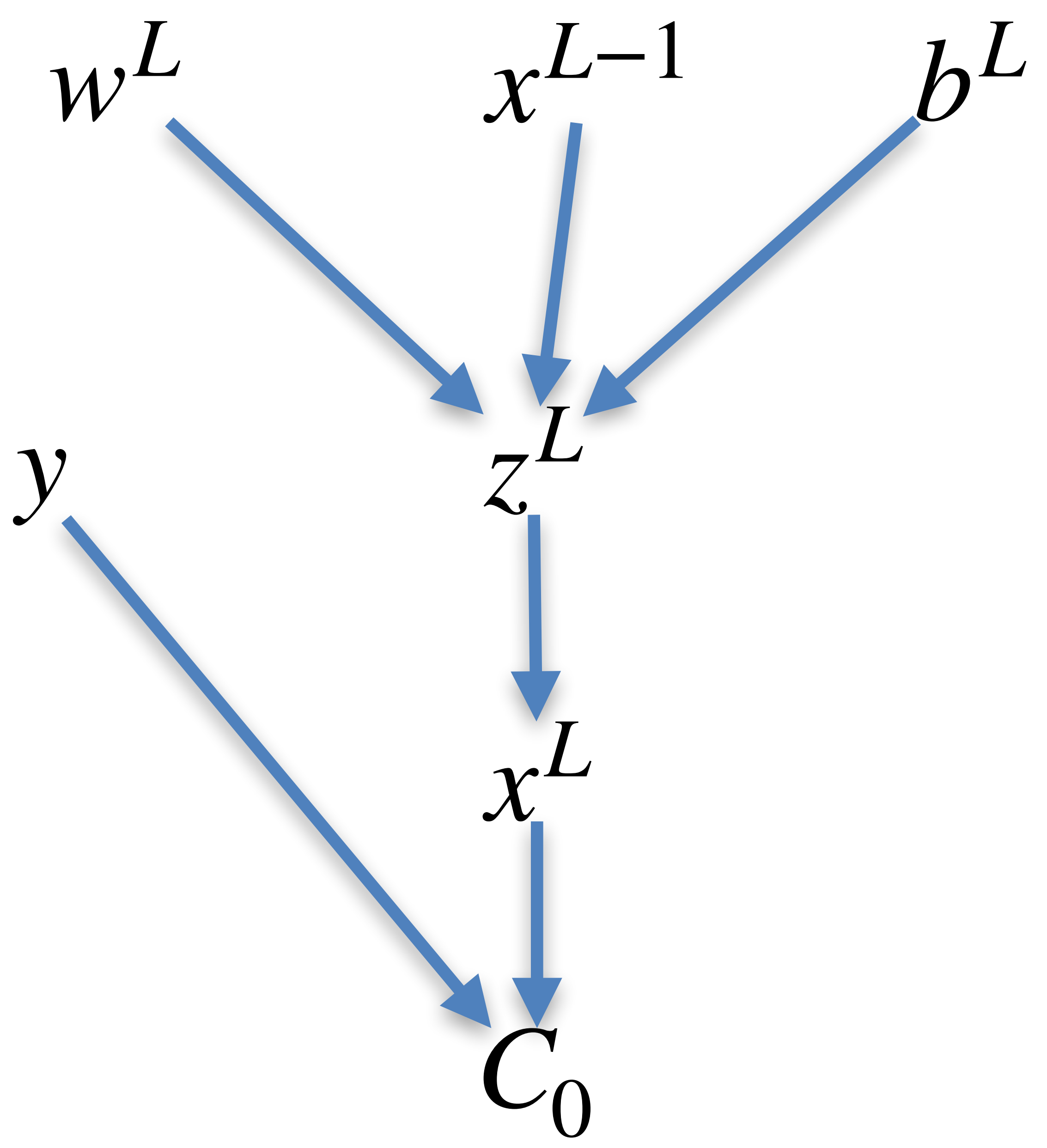


Desired output

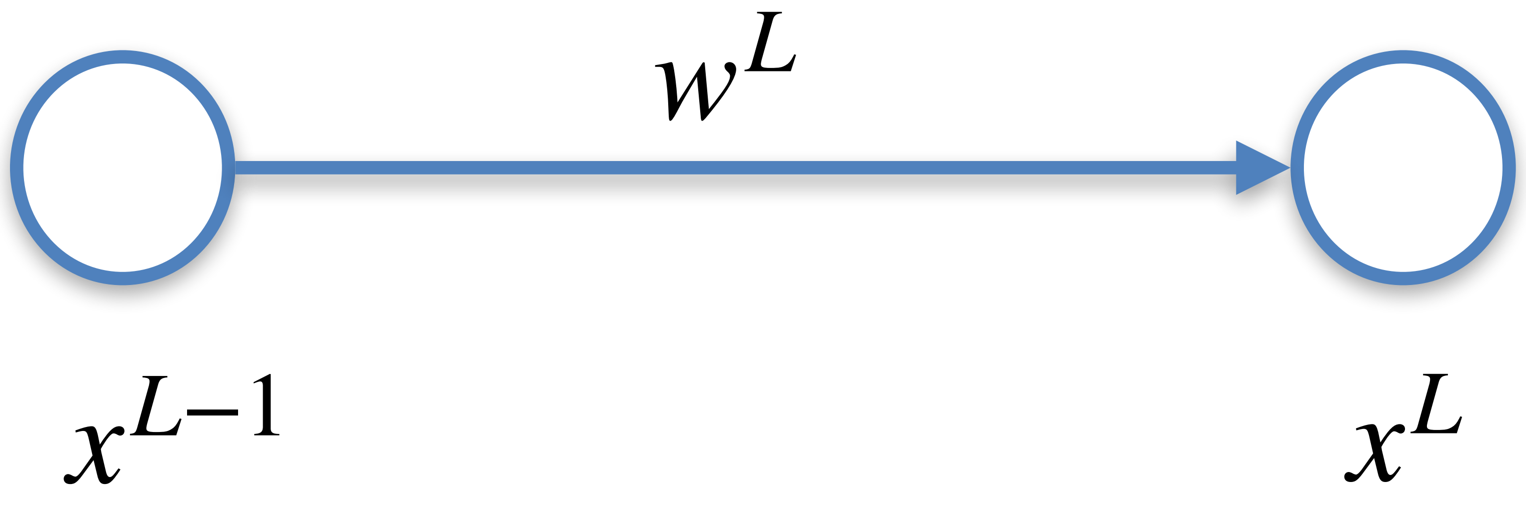
y

To apply gradient descent we need to compute $\frac{\partial}{\partial w^L} E_0$

Which means to evaluate how sensitive is E_0 with respect to w^L?



Backpropagation algorithm

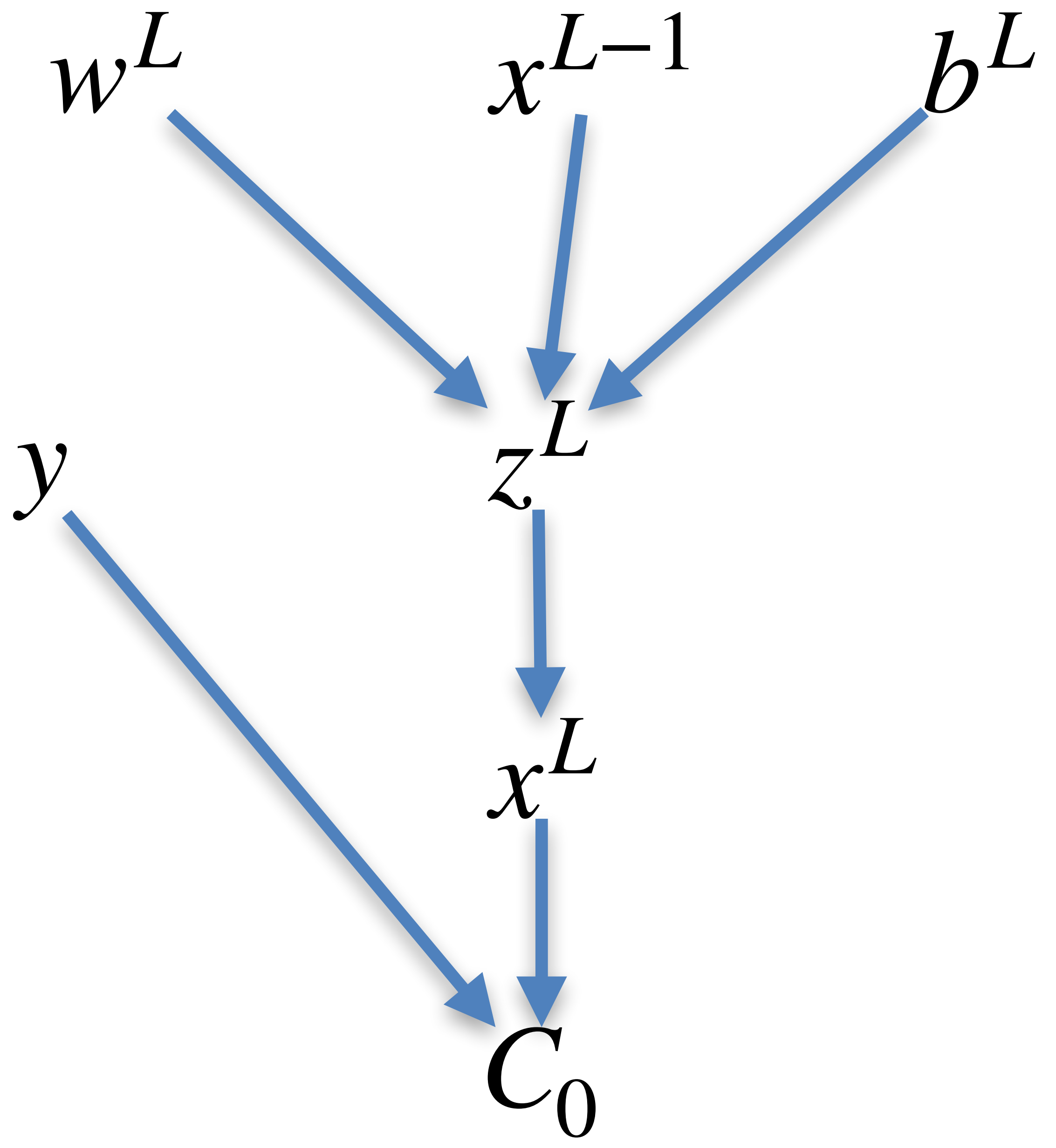


Desired output

y

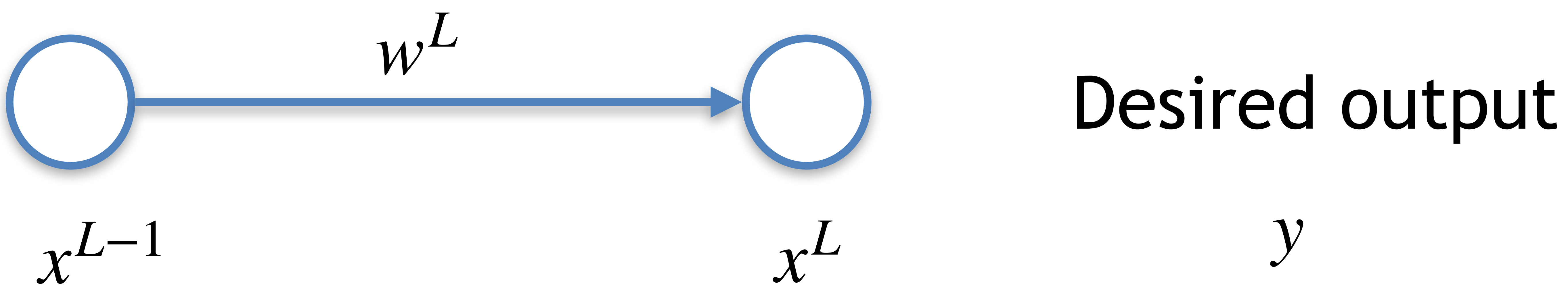
To apply gradient descent we need to compute $\frac{\partial}{\partial w^L} E_0$

Which means to evaluate how sensitive is E_0 with respect to w^L?



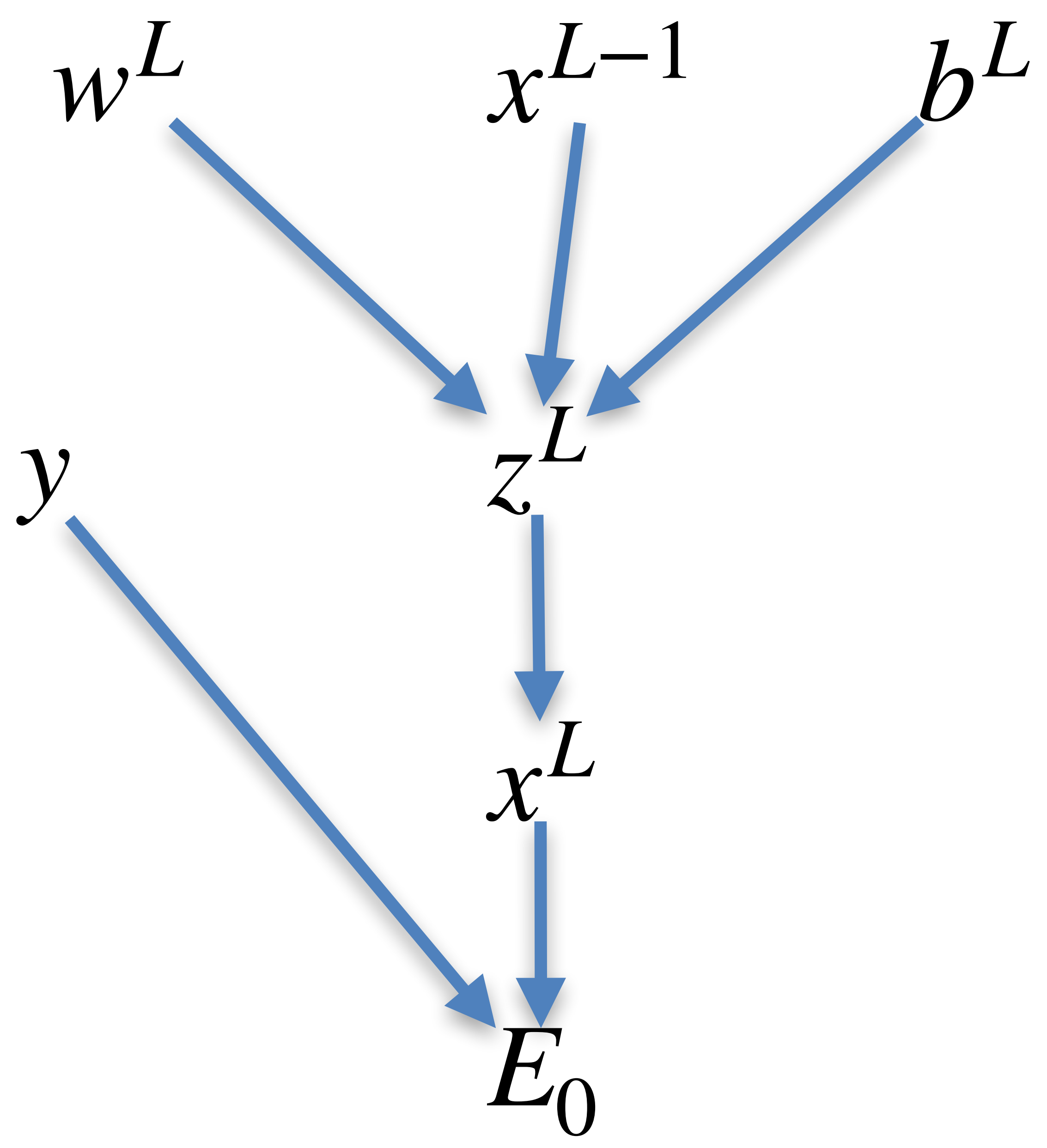
By construction, variations in w will affect z which will affect E

Backpropagation algorithm

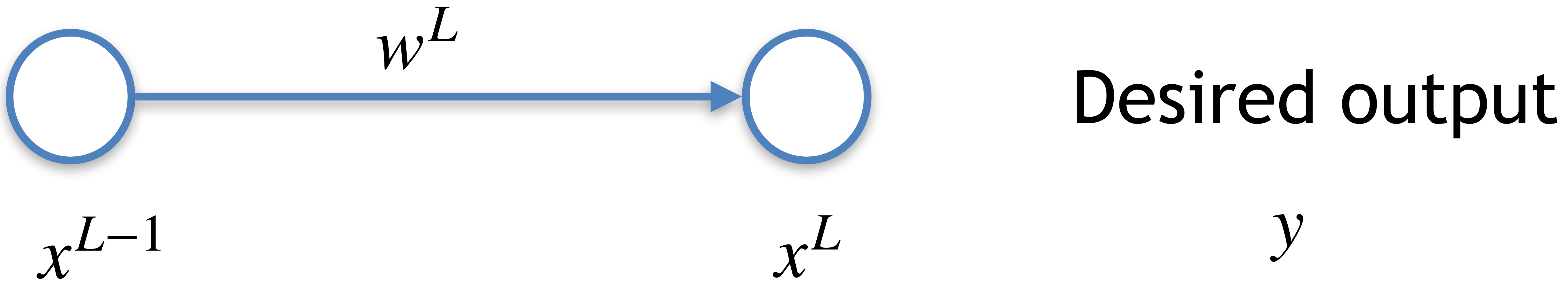


$$\frac{\partial}{\partial w^L}E_0 = \frac{\partial E_0}{\partial x^L}\frac{\partial x^L}{\partial z^L}\frac{\partial z^L}{\partial w^L}$$

Chain rule

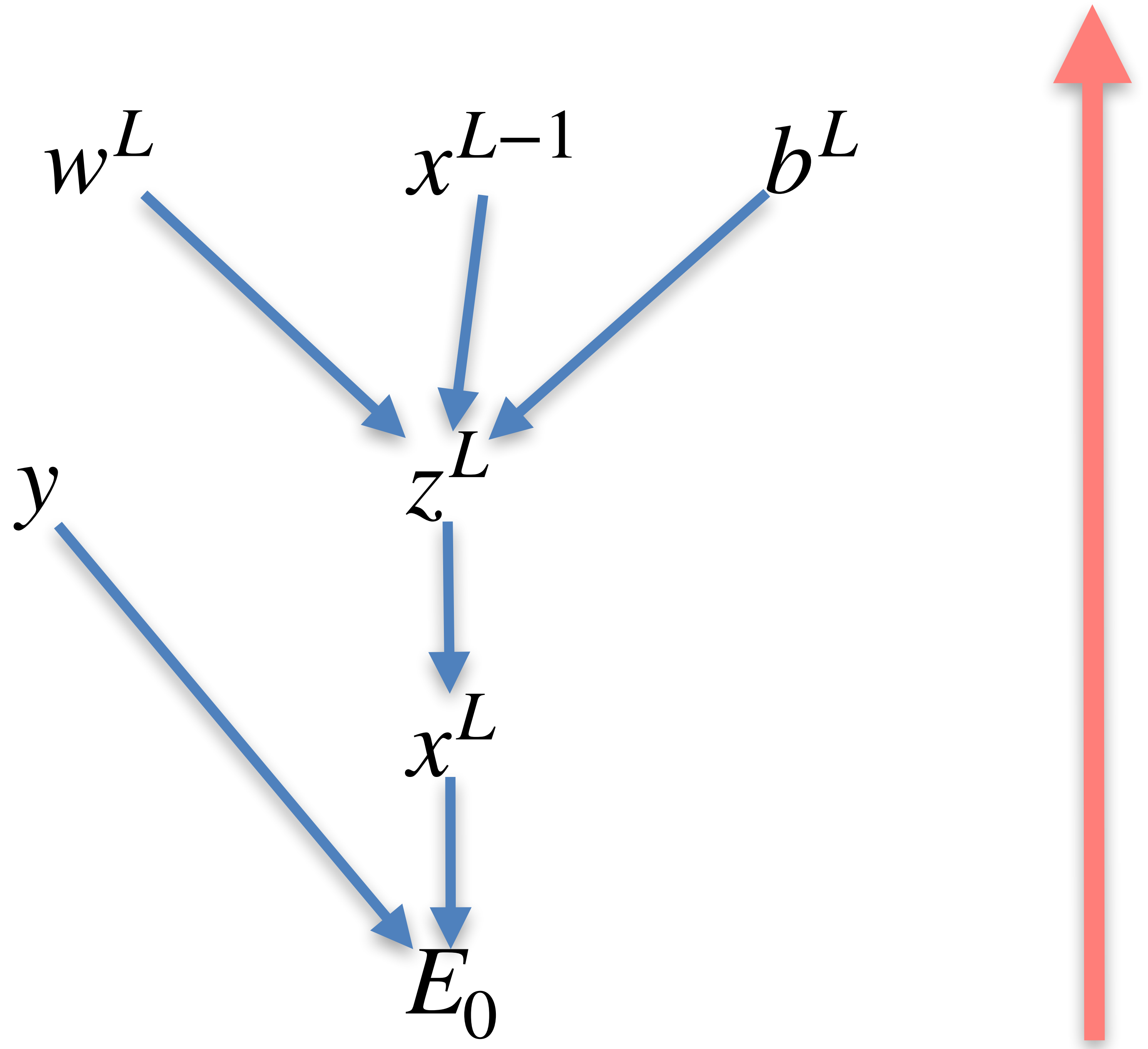


Backpropagation algorithm

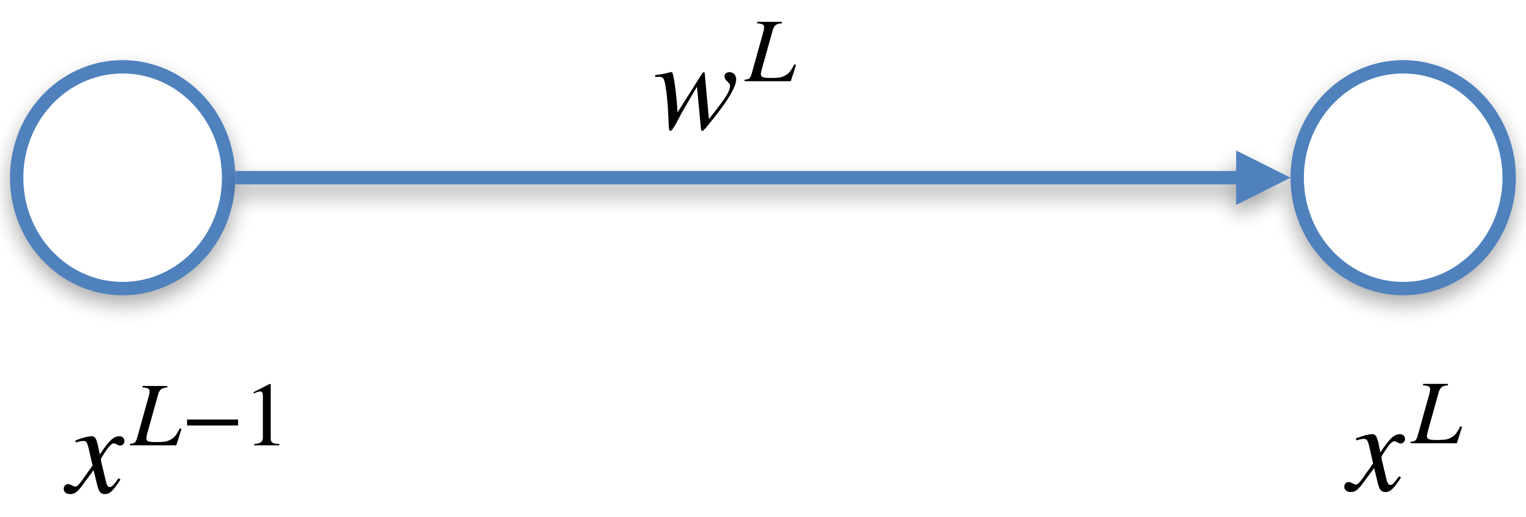


$$\frac{\partial}{\partial w^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial w^L}$$

Chain rule



Backpropagation algorithm

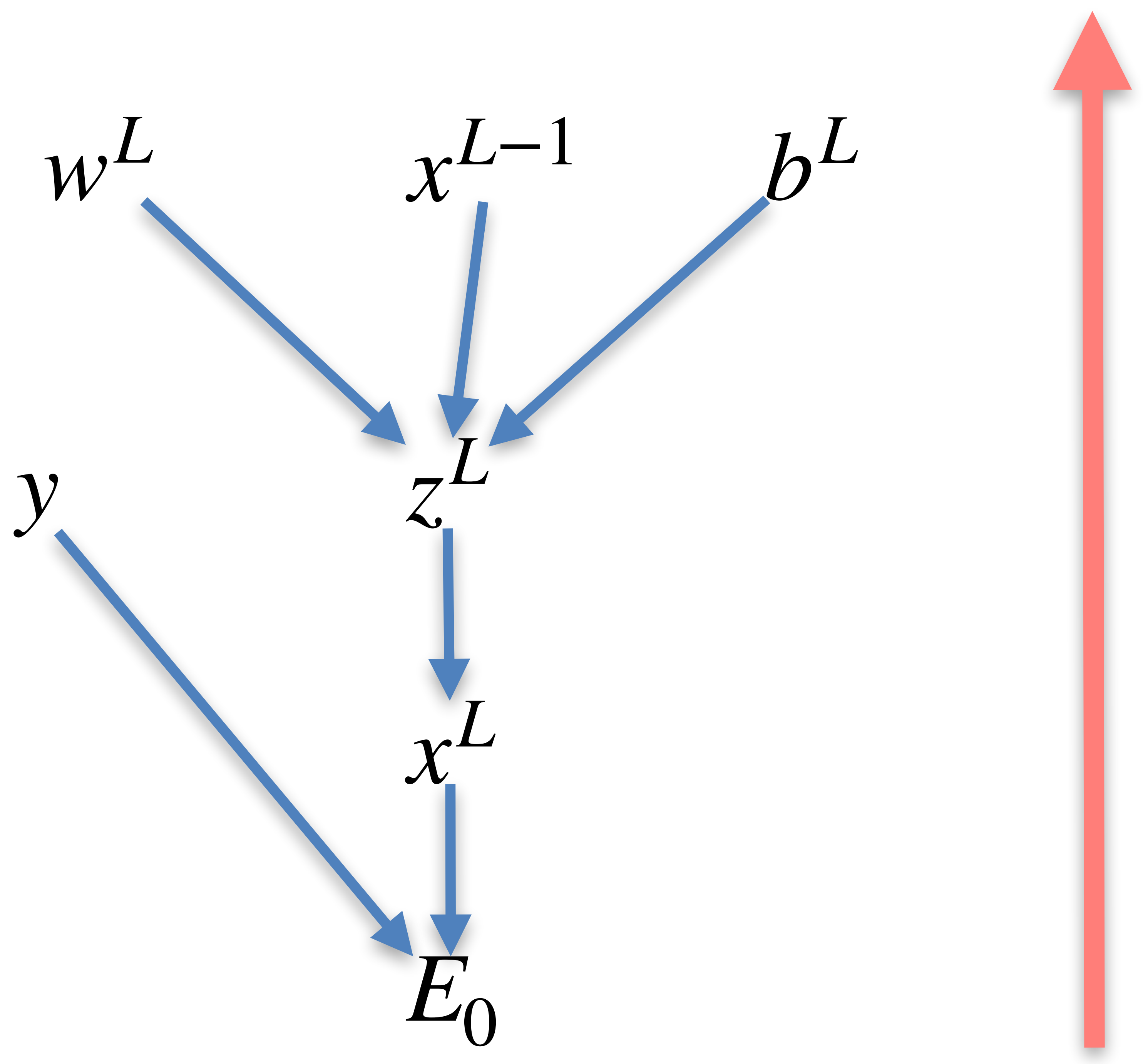


Desired output

y

$$\frac{\partial}{\partial w^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial w^L}$$

Chain rule



It goes backwards: backpropagation!

Backpropagation algorithm

$$E_0 = (x^L - y)^2$$
$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$
$$z^L = w^L x^{L-1} + b^L$$

$$\frac{\partial}{\partial w^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial w^L}$$

Backpropagation algorithm

$$E_0 = (x^L - y)^2$$
$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$
$$z^L = w^L x^{L-1} + b^L$$

$$\frac{\partial}{\partial w^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial w^L}$$

$$\frac{\partial E_0}{\partial x^L} = 2(x^L - y)$$

Backpropagation algorithm

$$E_0 = (x^L - y)^2$$

$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$

$$z^L = w^L x^{L-1} + b^L$$

$$\frac{\partial}{\partial w^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial w^L}$$

$$\frac{\partial E_0}{\partial x^L} = 2(x^L - y)$$

$$\frac{\partial x^L}{\partial z^L} = \sigma'(z^L)$$

Credits: https://www.youtube.com/watch?v=tIeHLnjs5U8

Backpropagation algorithm

$$E_0 = (x^L - y)^2$$
$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$
$$z^L = w^L x^{L-1} + b^L$$

$$\frac{\partial}{\partial w^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial w^L}$$

$$\frac{\partial E_0}{\partial x^L} = 2(x^L - y)$$

$$\frac{\partial x^L}{\partial z^L} = \sigma'(z^L)$$

$$\frac{\partial z^L}{\partial w^L} = x^{L-1}$$

Credits: https://www.youtube.com/watch?v=tIeHLnjs5U8

Backpropagation algorithm

$$E_0 = (x^L - y)^2$$
$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$
$$z^L = w^L x^{L-1} + b^L$$

$$\frac{\partial}{\partial w^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial w^L}$$

$$\frac{\partial E_0}{\partial x^L} = 2(x^L - y) \qquad \frac{\partial x^L}{\partial z^L} = \sigma'(z^L) \qquad \frac{\partial z^L}{\partial w^L} = x^{L-1}$$

$$\frac{\partial}{\partial w^L} E_0 = 2x^{L-1}\sigma'(z^L)(x^L - y)$$

Credits: https://www.youtube.com/watch?v=tIeHLnjs5U8

Backpropagation algorithm

$$E_0 = (x^L - y)^2$$
$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$
$$z^L = w^L x^{L-1} + b^L$$

$$\frac{\partial}{\partial w^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial w^L}$$

$$\frac{\partial E_0}{\partial x^L} = 2(x^L - y)$$

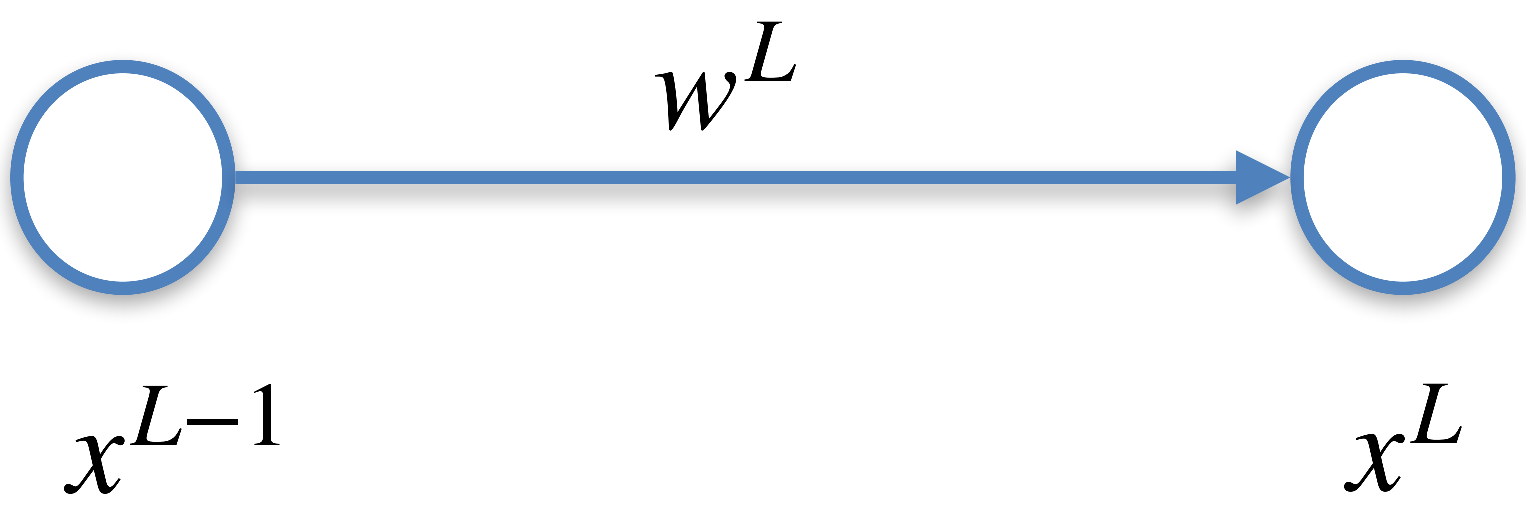
$$\frac{\partial x^L}{\partial z^L} = \sigma'(z^L)$$

$$\frac{\partial z^L}{\partial w^L} = x^{L-1}$$

$$\frac{\partial}{\partial w^L} E_0 = 2x^{L-1}\sigma'(z^L)(x^L - y) \rightarrow \frac{\partial}{\partial w^L} E = \frac{1}{s} \sum_{i=0}^{s-1} \frac{\partial}{\partial w^L} E_i$$

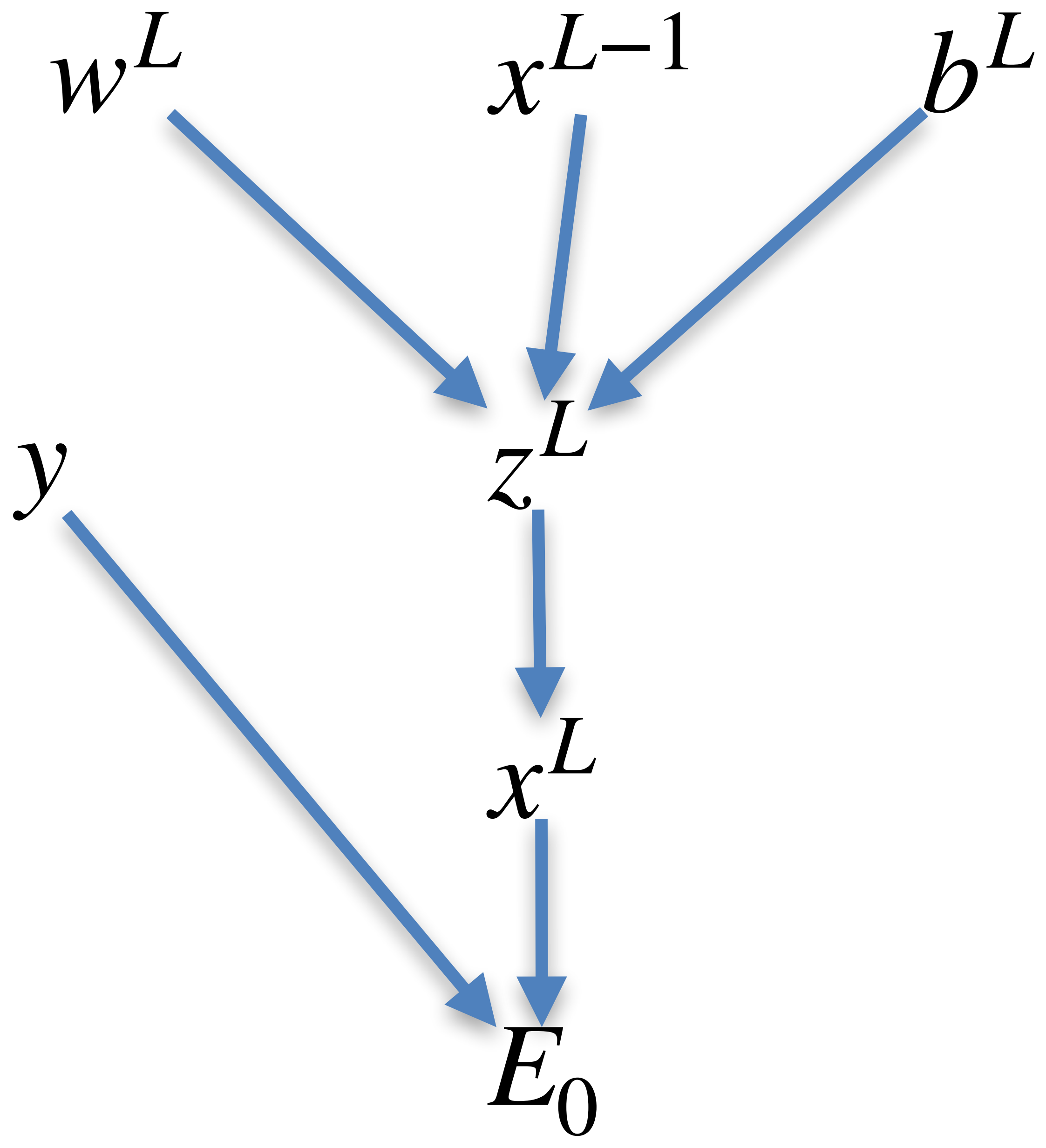
Credits: https://www.youtube.com/watch?v=tIeHLnjs5U8

Backpropagation algorithm

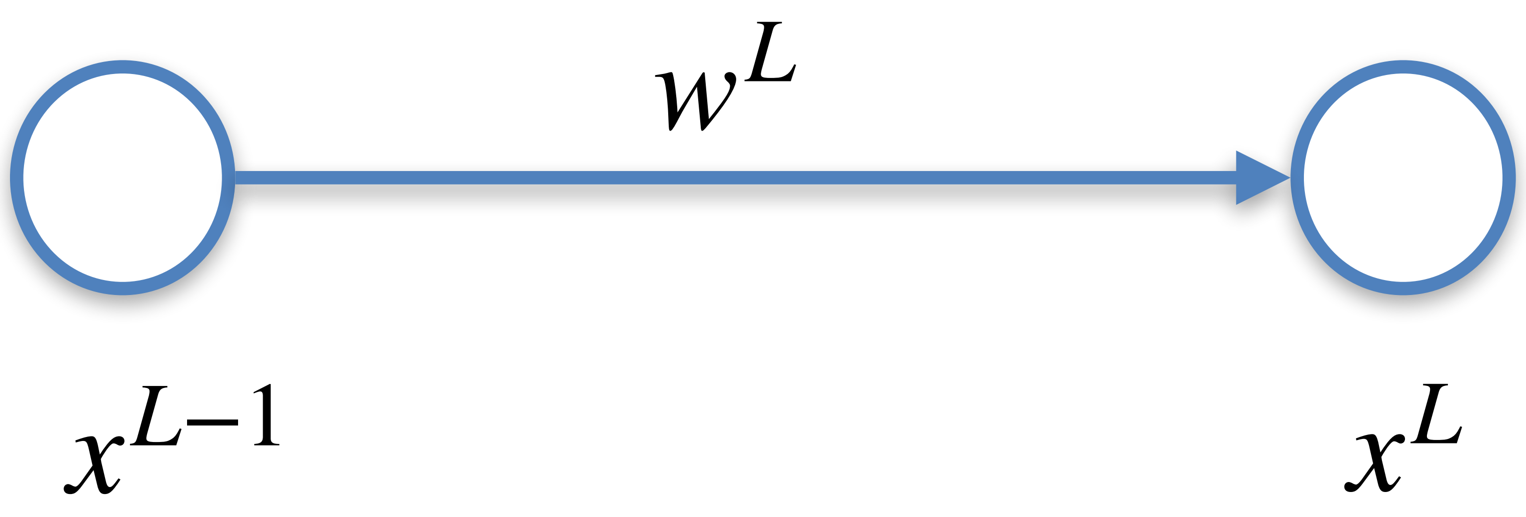


Desired output

y



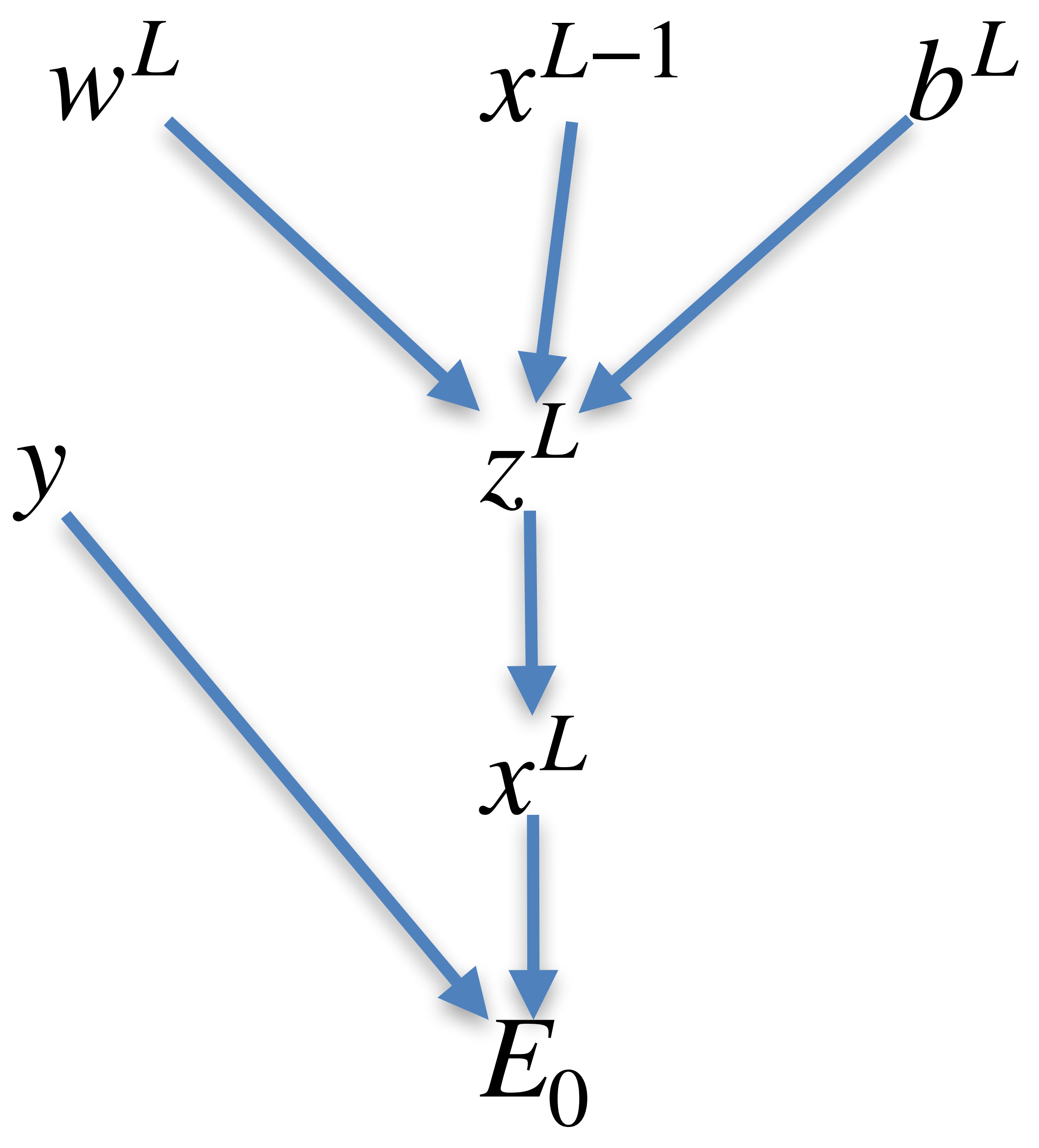
Backpropagation algorithm



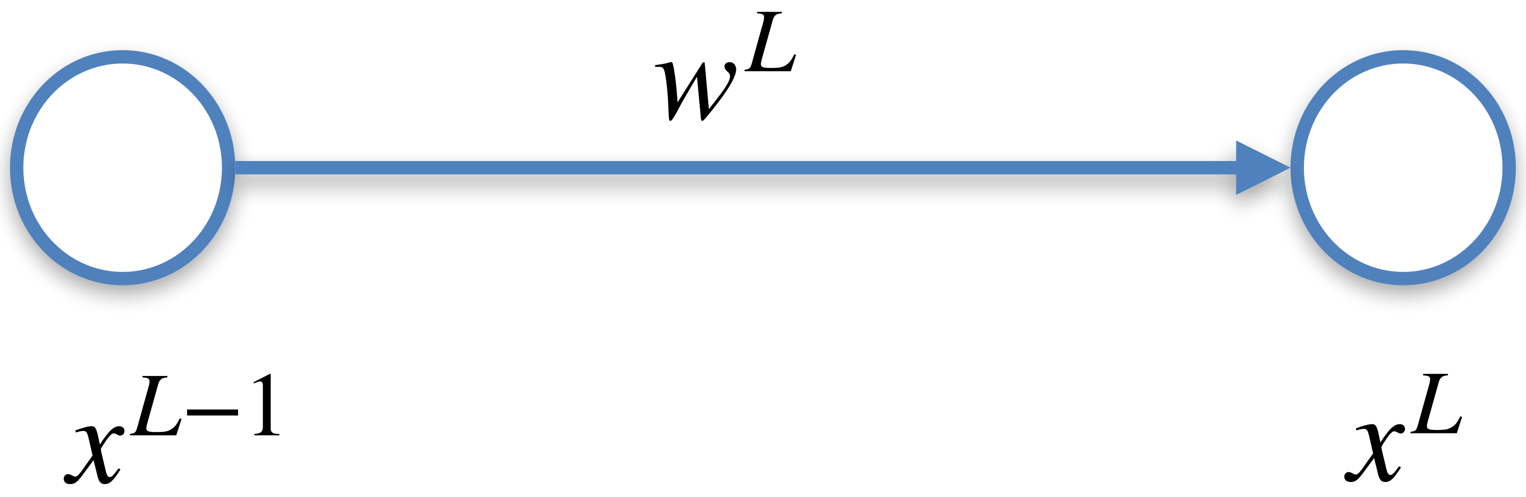
Desired output

y

To apply gradient descent we need to compute $\dfrac{\partial}{\partial b^L} E_0$



Backpropagation algorithm

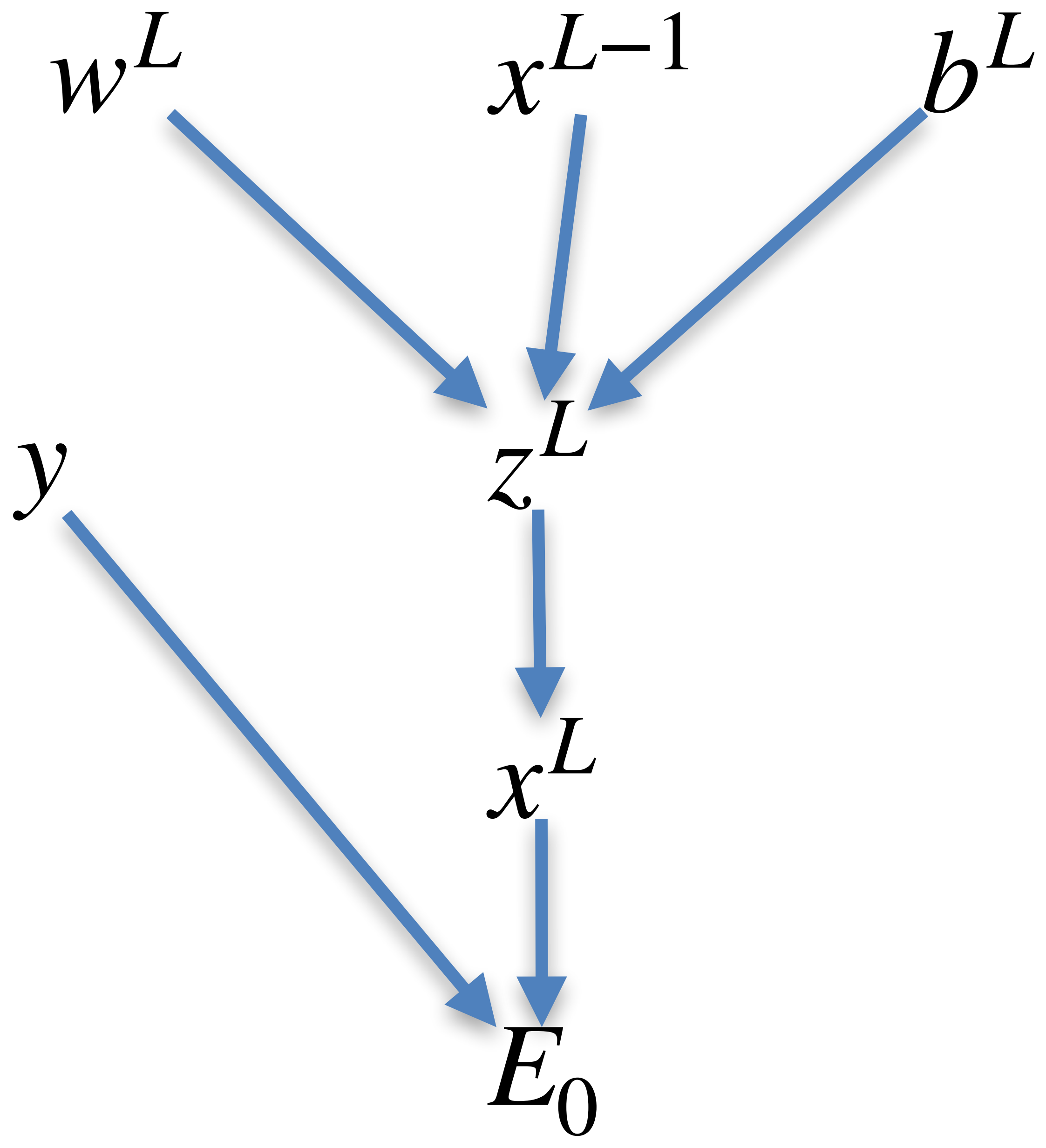


Desired output

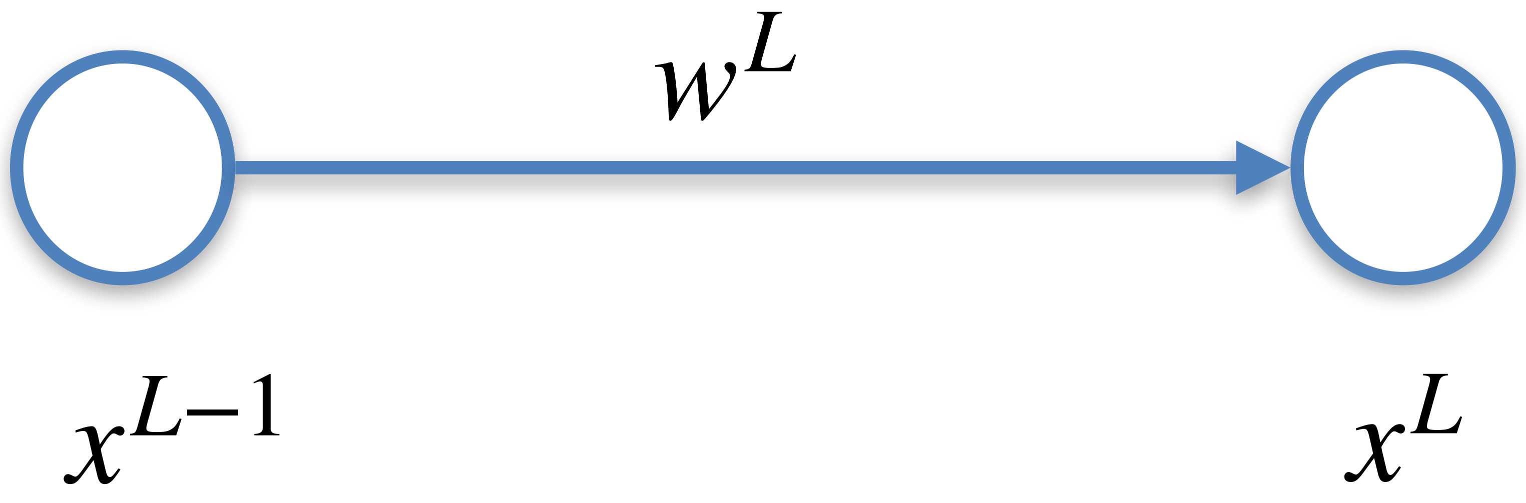
y

To apply gradient descent we need to compute $\frac{\partial}{\partial b^L} E_0$

Which means to evaluate how sensitive is E_0 with respect to b^L?



Backpropagation algorithm



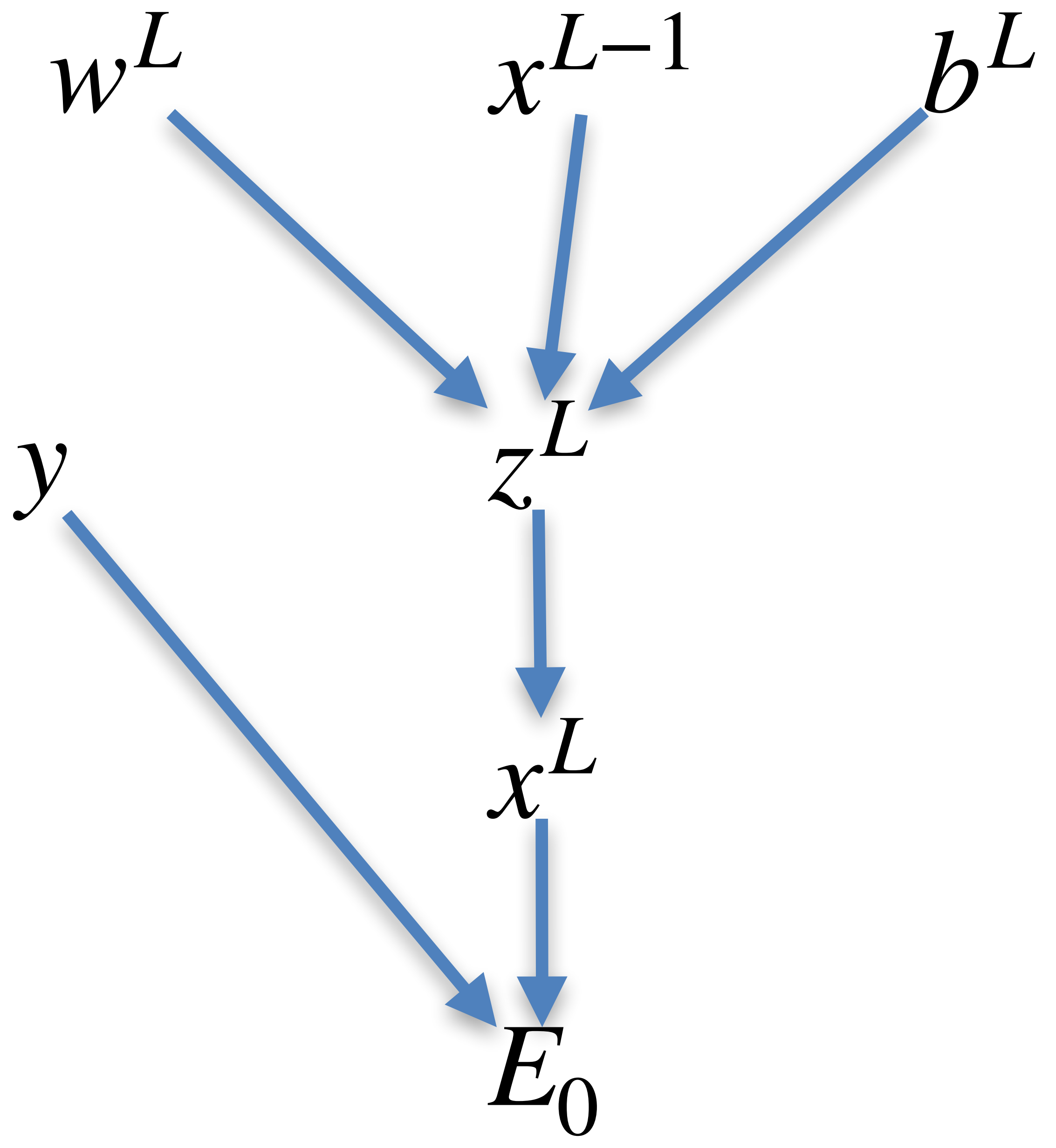
Desired output

y

To apply gradient descent we need to compute $\frac{\partial}{\partial b^L} E_0$

Which means to evaluate how sensitive is E_0 with respect to b^L?

By construction, variations in b will affect z which will affect E



Backpropagation algorithm

$$E_0 = (x^L - y)^2$$
$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$
$$z^L = w^L x^{L-1} + b^L$$

$$\frac{\partial}{\partial b^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial b^L}$$

Backpropagation algorithm

$$E_0 = (x^L - y)^2$$

$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$

$$z^L = w^L x^{L-1} + b^L$$

$$\frac{\partial}{\partial b^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial b^L}$$

$$\frac{\partial E_0}{\partial x^L} = 2(x^L - y)$$

Credits: https://www.youtube.com/watch?v=tIeHLnjs5U8

Backpropagation algorithm

$$E_0 = (x^L - y)^2$$
$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$
$$z^L = w^L x^{L-1} + b^L$$

$$\frac{\partial}{\partial b^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial b^L}$$

$$\frac{\partial E_0}{\partial x^L} = 2(x^L - y)$$

$$\frac{\partial x^L}{\partial z^L} = \sigma'(z^L)$$

Backpropagation algorithm

$$E_0 = (x^L - y)^2$$
$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$
$$z^L = w^L x^{L-1} + b^L$$

$$\frac{\partial}{\partial b^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial b^L}$$

$$\frac{\partial E_0}{\partial x^L} = 2(x^L - y)$$

$$\frac{\partial x^L}{\partial z^L} = \sigma'(z^L)$$

$$\frac{\partial z^L}{\partial b^L} = 1$$

Backpropagation algorithm

$$E_0 = (x^L - y)^2$$
$$x^L = \sigma(w^L x^{L-1} + b^L) = \sigma(z^L)$$
$$z^L = w^L x^{L-1} + b^L$$

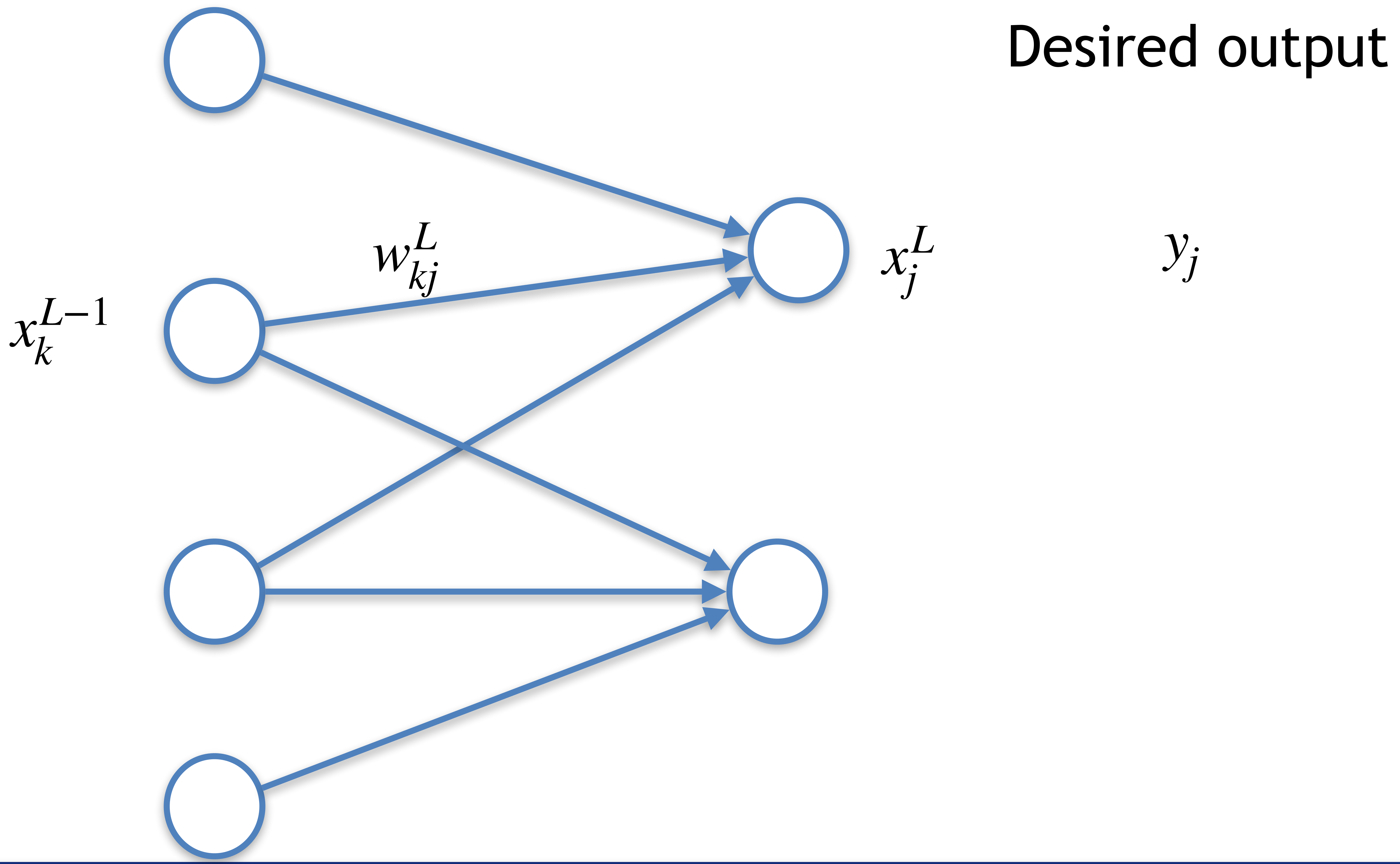
$$\frac{\partial}{\partial b^L} E_0 = \frac{\partial E_0}{\partial x^L} \frac{\partial x^L}{\partial z^L} \frac{\partial z^L}{\partial b^L}$$

$$\frac{\partial E_0}{\partial x^L} = 2(x^L - y) \qquad \frac{\partial x^L}{\partial z^L} = \sigma'(z^L) \qquad \frac{\partial z^L}{\partial b^L} = 1$$

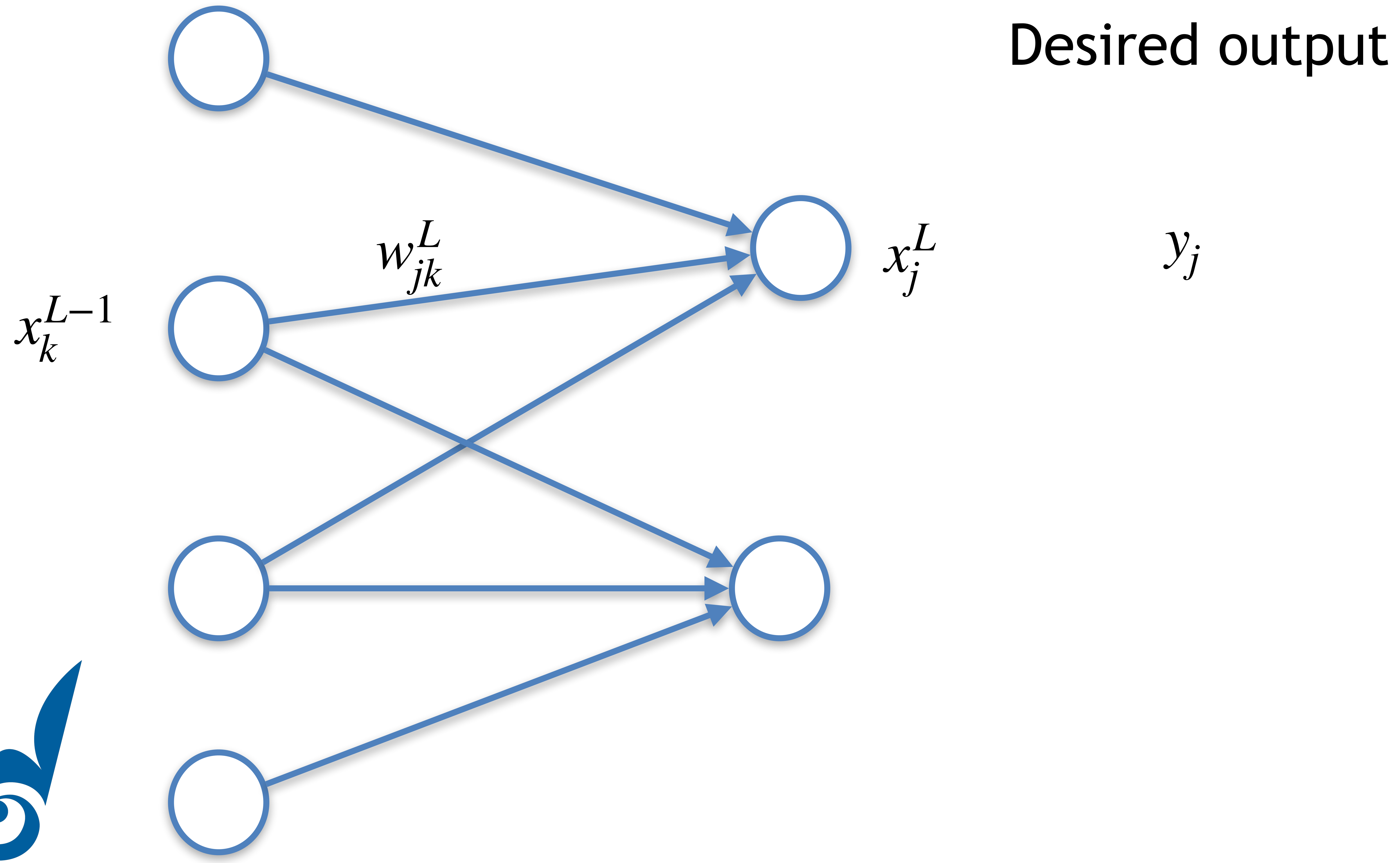
$$\frac{\partial}{\partial b^L} E_0 = 2\sigma'(z^L)(x^L - y) \rightarrow \frac{\partial}{\partial b^L} E = \frac{1}{s} \sum_{i=0}^{s-1} \frac{\partial}{\partial b^L} E_i$$

What about a general neural network?!

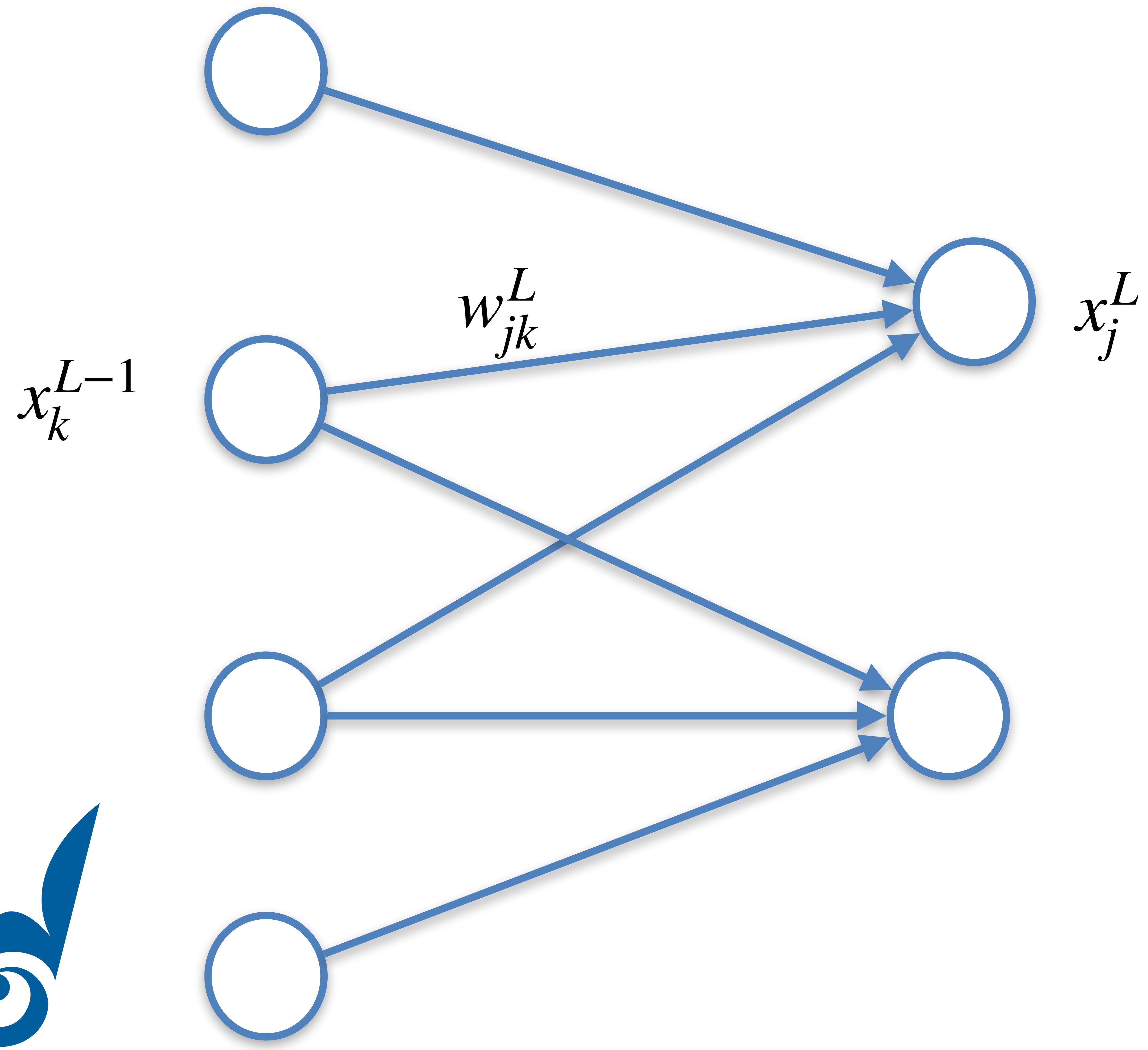
Backpropagation algorithm



Backpropagation algorithm



Backpropagation algorithm

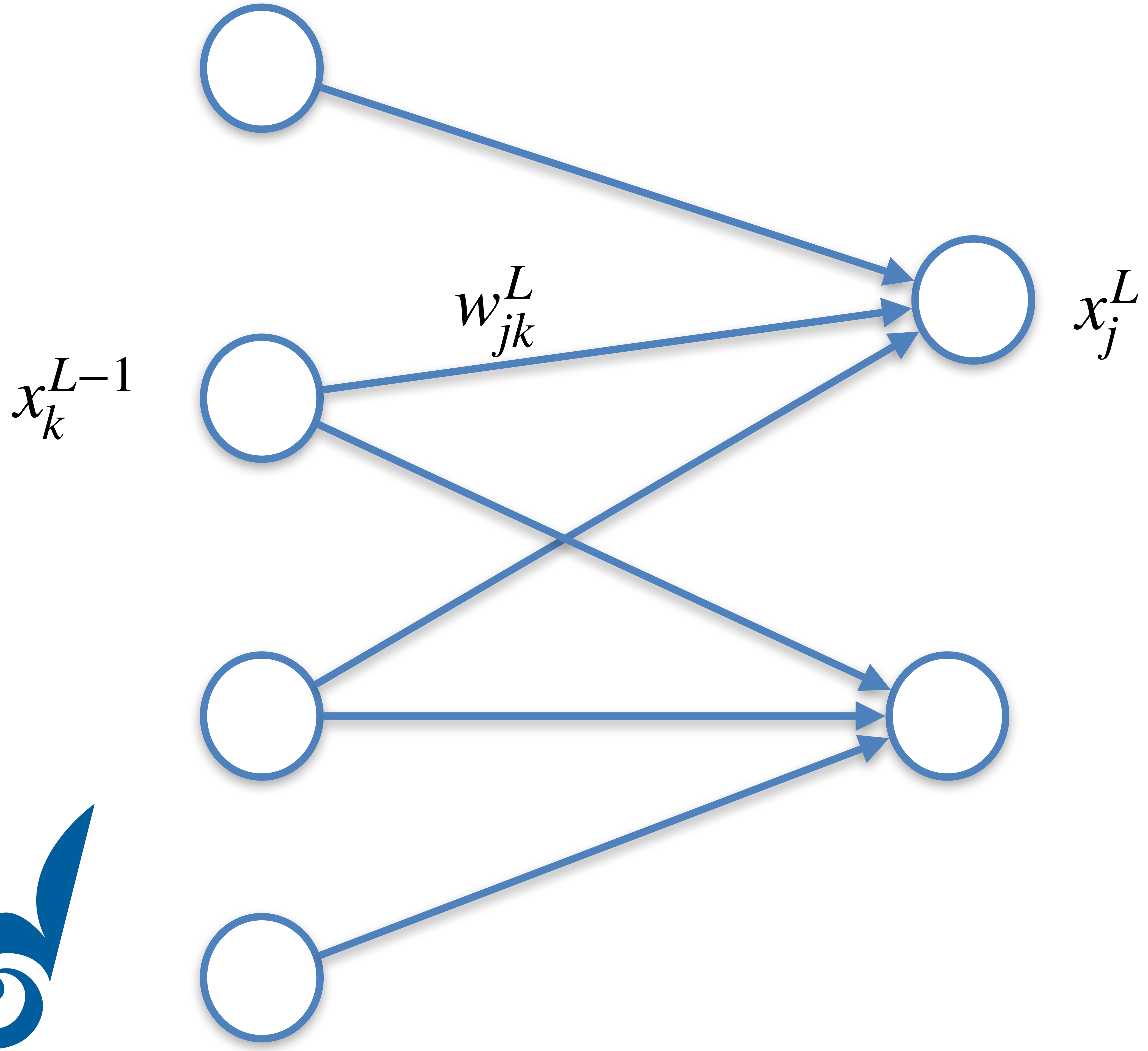


Desired output

y_j

$$E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

Backpropagation algorithm



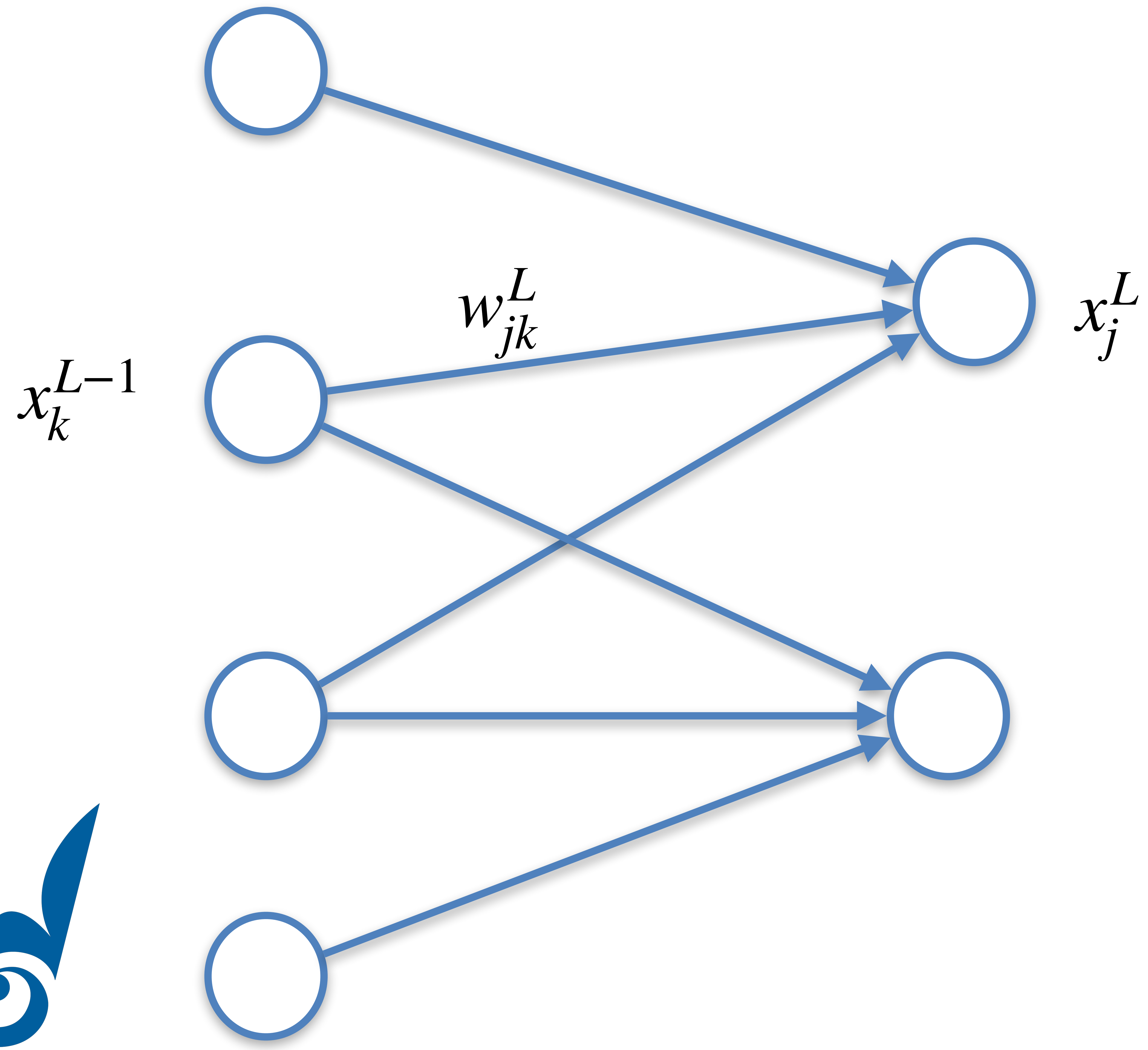
Desired output

y_j

$$E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

$$z_j^L = \sum_v w_{jv}^L x_v^{L-1} + b_j^L$$

Backpropagation algorithm



Desired output

y_j

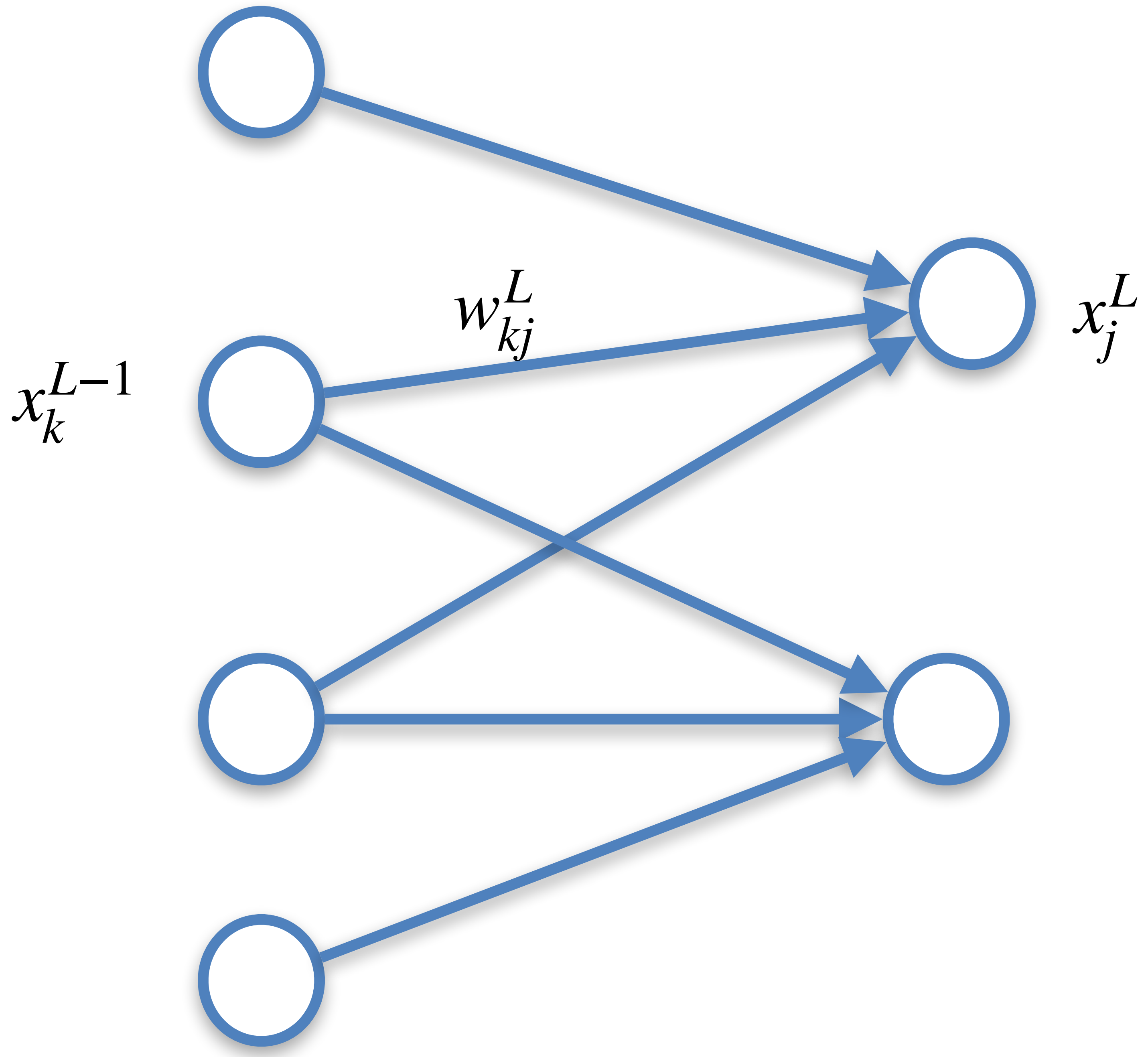
$$E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

$$z_j^L = \sum_v w_{jv}^L x_v^{L-1} + b_j^L$$

$$x_j^L = \sigma(z_j^L)$$

Backpropagation algorithm

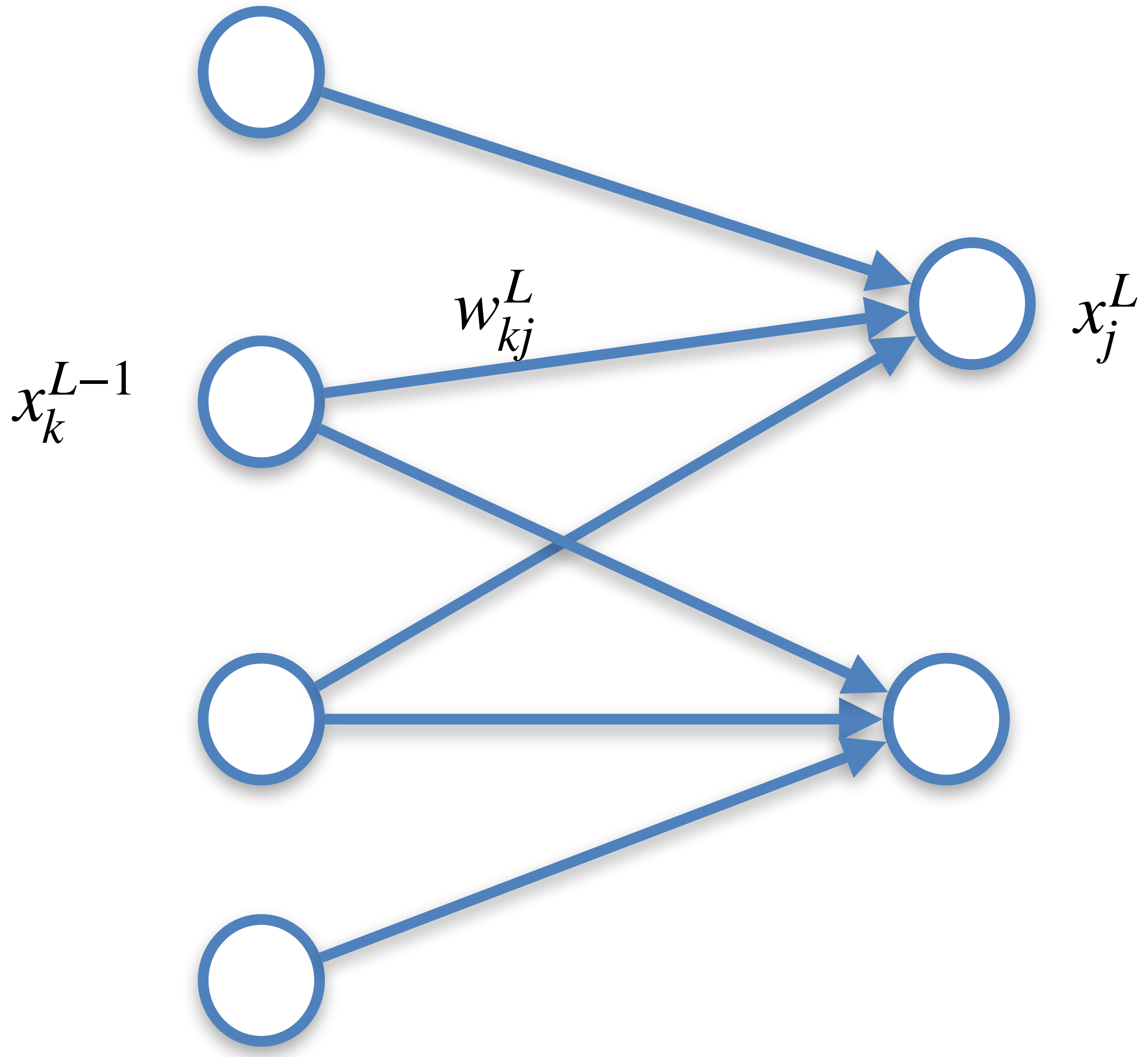
As before we need to compute the gradients



Backpropagation algorithm

As before we need to compute the gradients

For the last layer, for example, before we computed

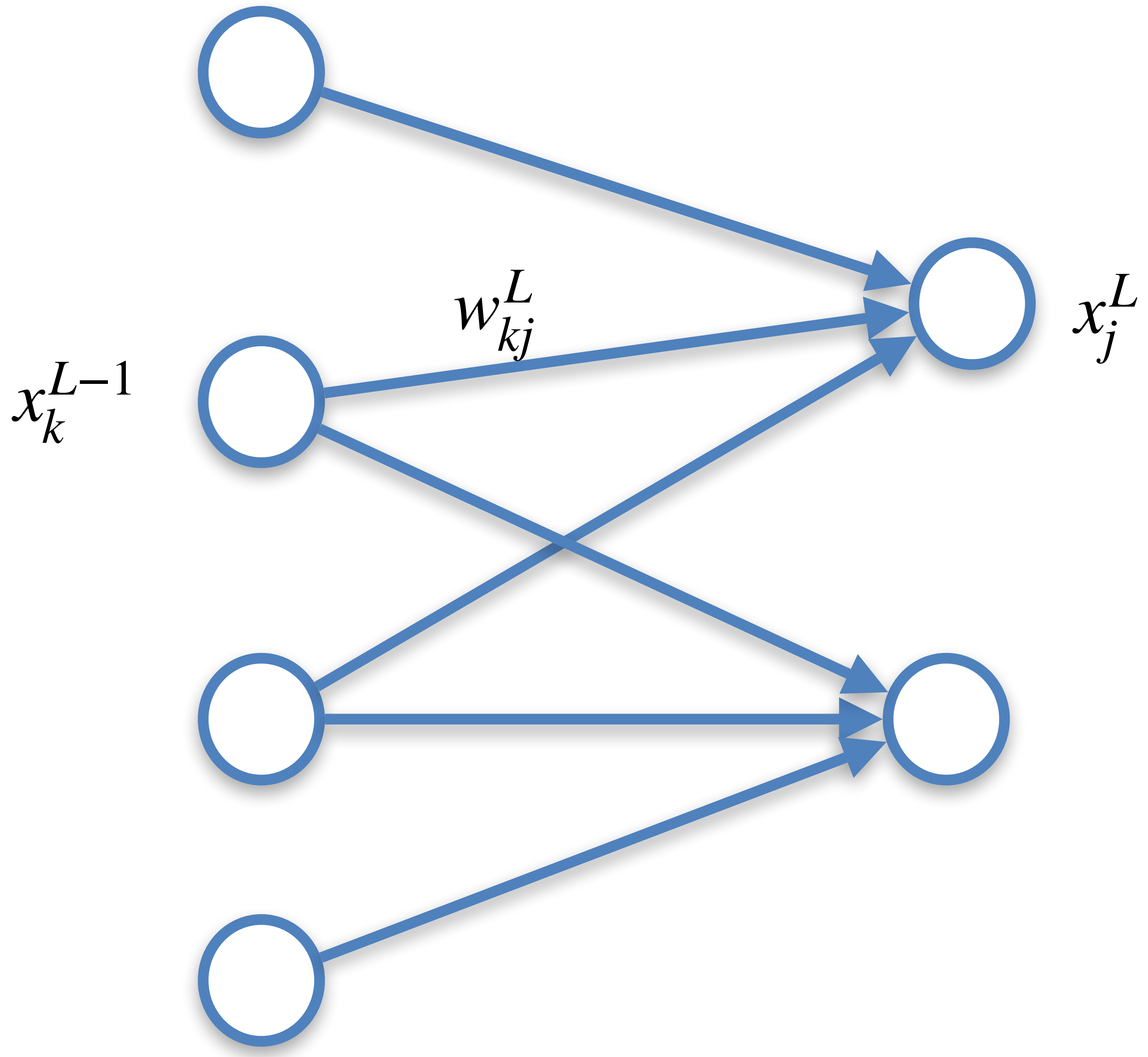


Backpropagation algorithm

As before we need to compute the gradients

For the last layer, for example, before we computed

$$\frac{\partial E}{\partial w^L}$$



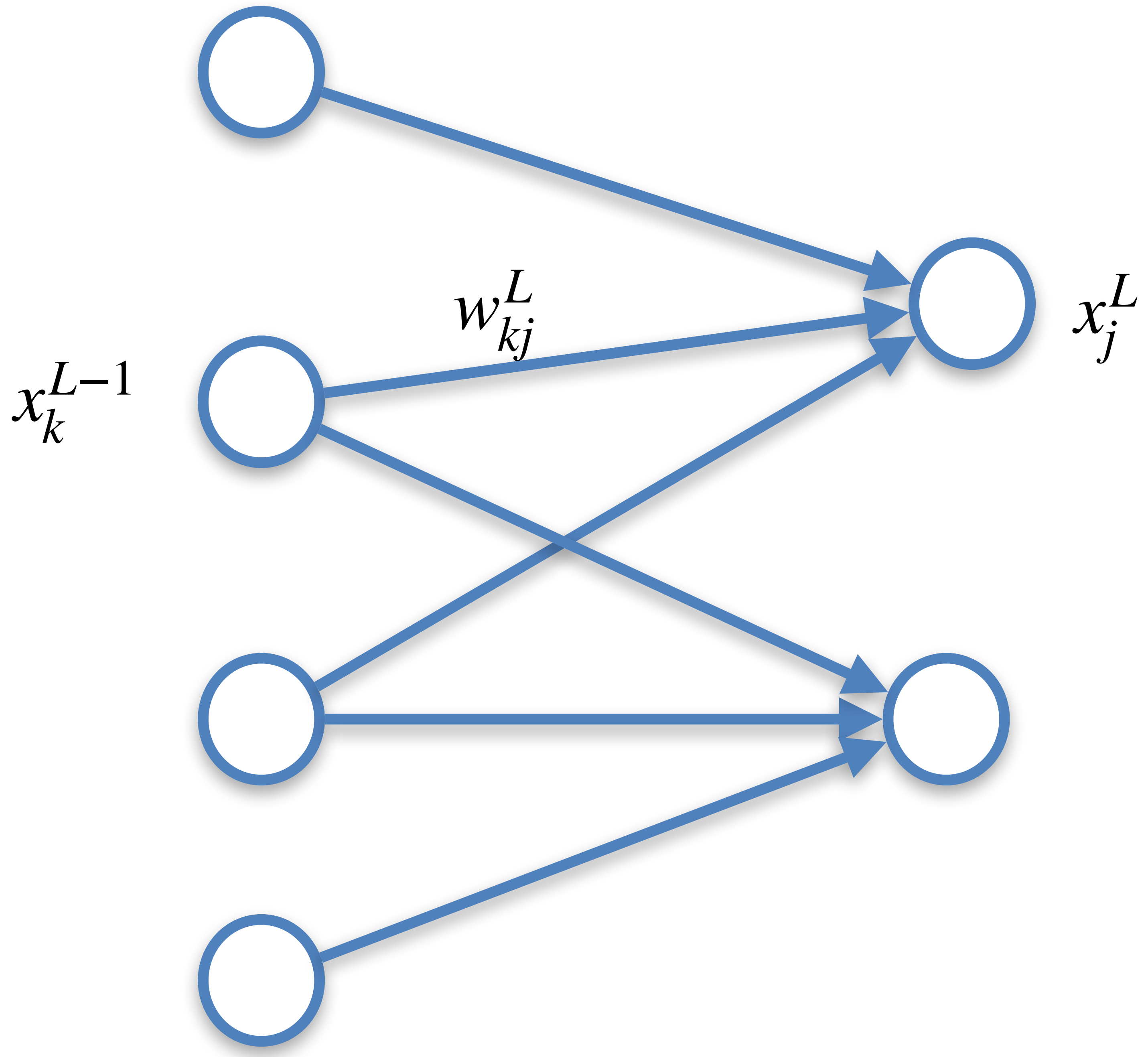
Backpropagation algorithm

As before we need to compute the gradients

For the last layer, for example, before we computed

$$\frac{\partial E}{\partial w^L}$$

Now w is a matrix!



Backpropagation algorithm

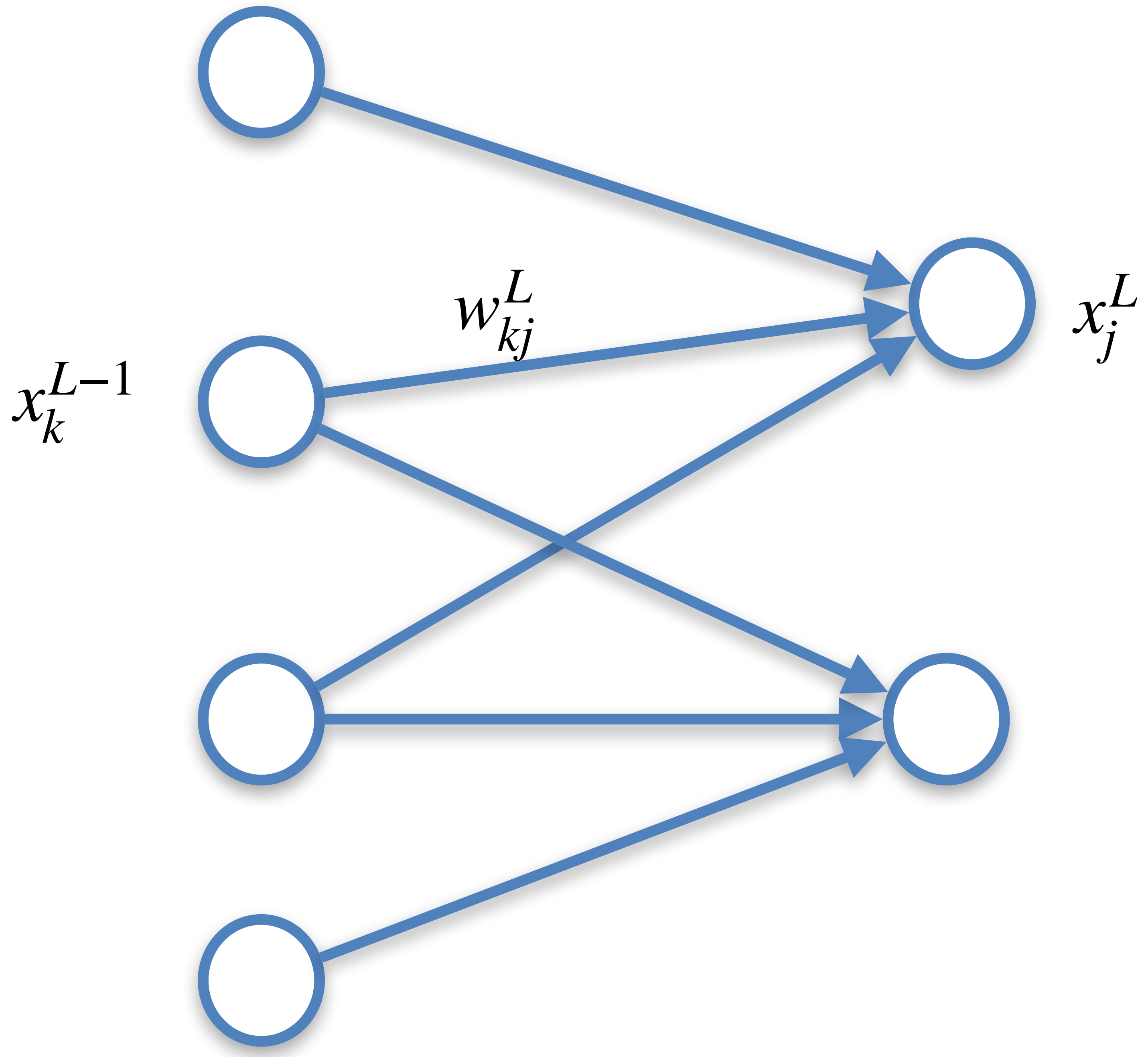
As before we need to compute the gradients

For the last layer, for example, before we computed

$$\frac{\partial E}{\partial w^L}$$

Now w is a matrix!

$$\frac{\partial E}{\partial w^L} \rightarrow \frac{\partial E}{\partial w^L_{ij}}$$



Backpropagation algorithm

As before we need to compute the gradients

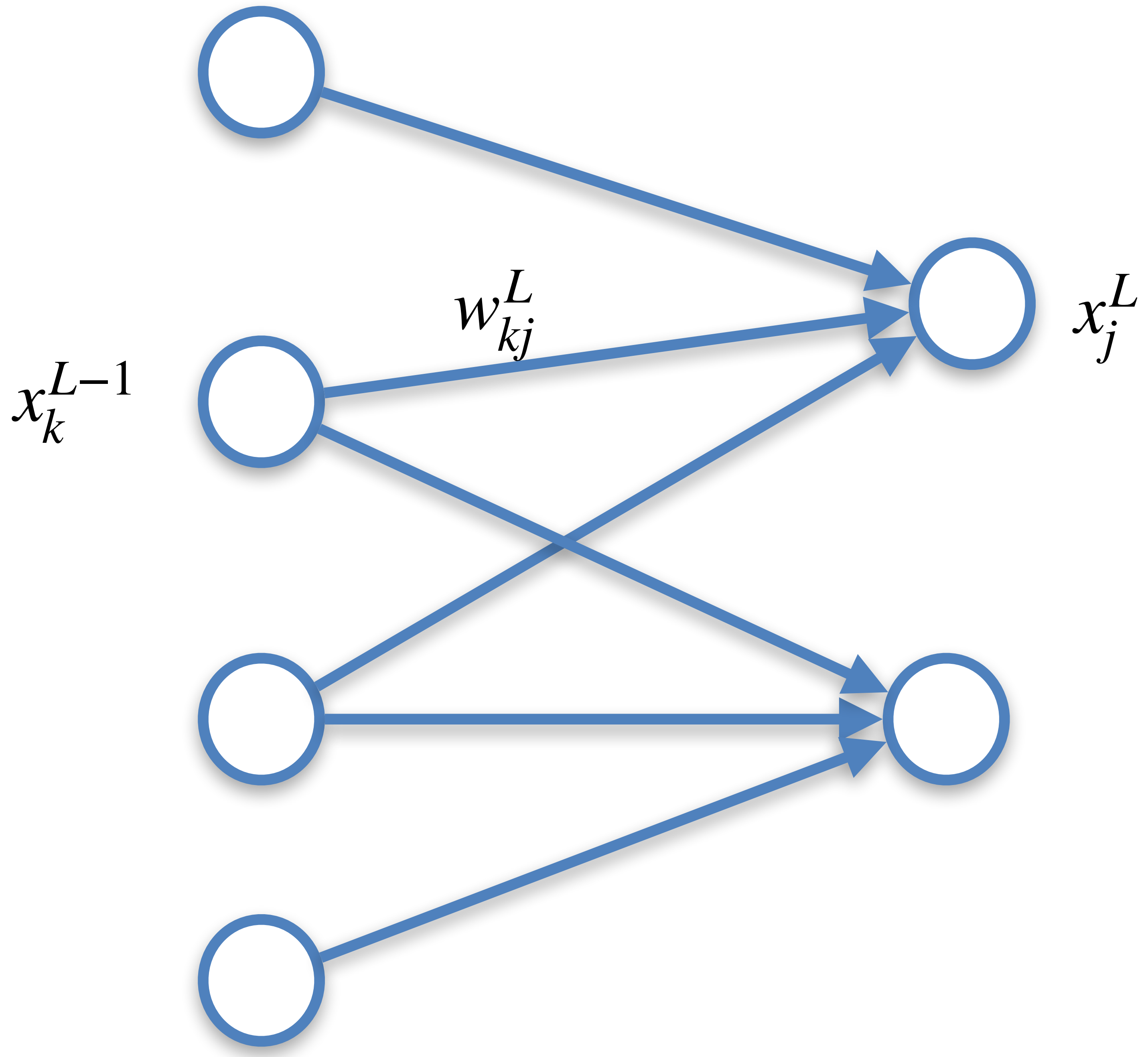
For the last layer, for example, before we computed

$$\frac{\partial E}{\partial w^L}$$

Now w is a matrix!

$$\frac{\partial E}{\partial w^L} \to \frac{\partial E}{\partial w^L_{ij}}$$

We also need to consider the bias



Backpropagation algorithm

As before we need to compute the gradients

For the last layer, for example, before we computed

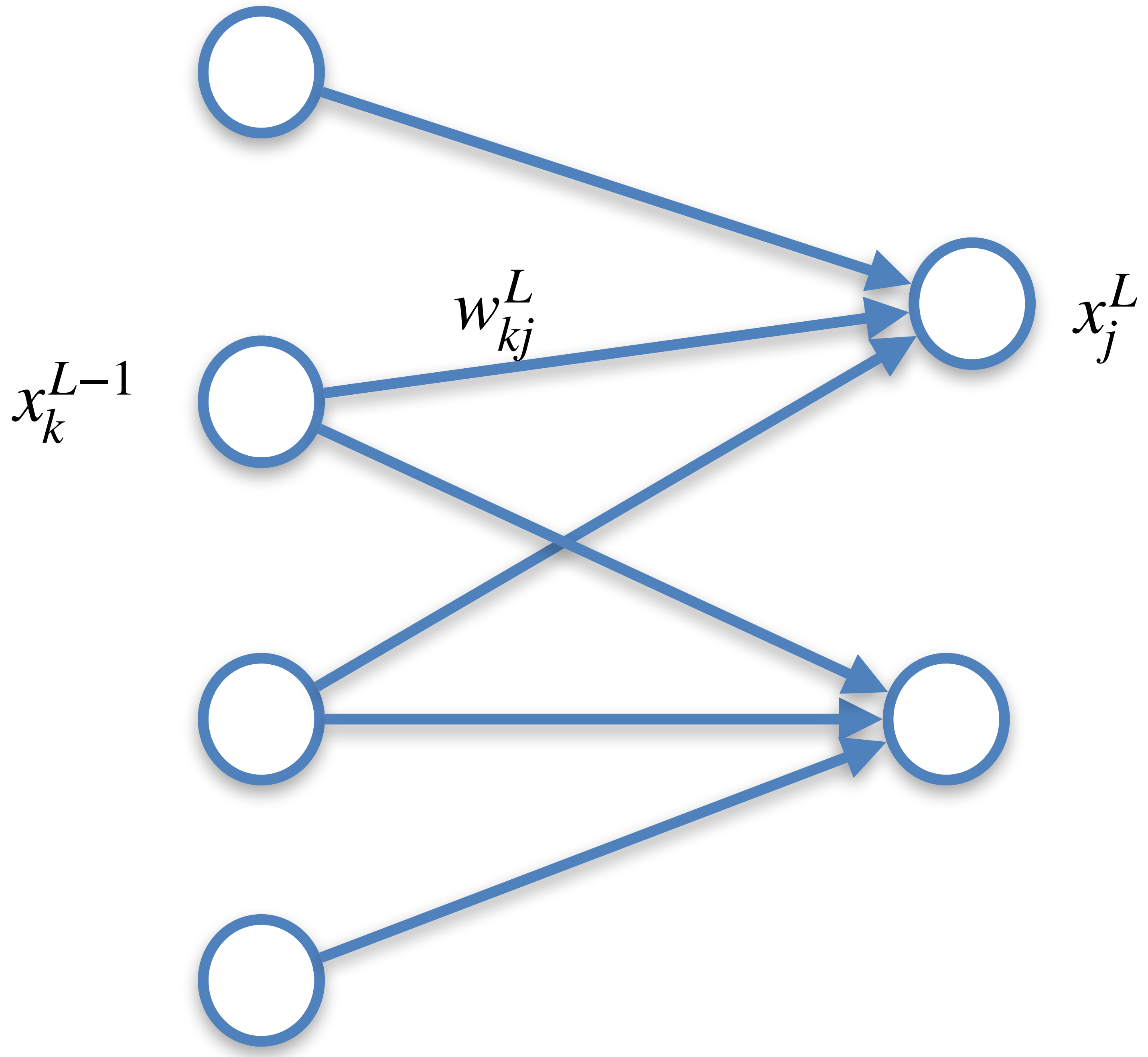
$$\frac{\partial E}{\partial w^L}$$

Now w is a matrix!

$$\frac{\partial E}{\partial w^L} \rightarrow \frac{\partial E}{\partial w^L_{ij}}$$

We also need to consider the bias

$$\frac{\partial E}{\partial b^L}$$



Backpropagation algorithm

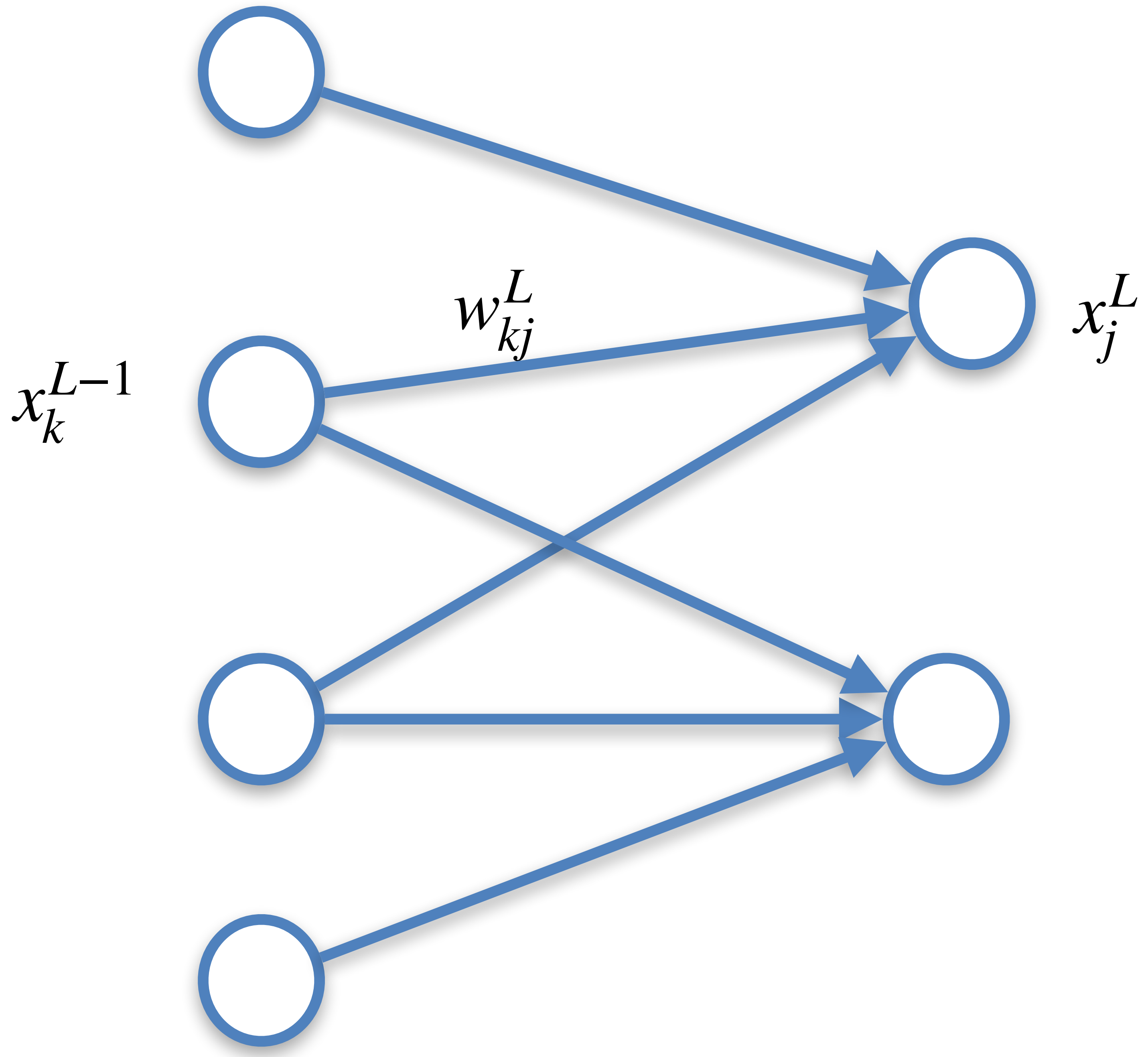
As before we need to compute the gradients

For the last layer, for example, before we computed

$$\frac{\partial E}{\partial w^L}$$

Now w is a matrix!

$$\frac{\partial E}{\partial w^L} \rightarrow \frac{\partial E}{\partial w^L_{ij}}$$



We also need to consider the bias

$$\frac{\partial E}{\partial b^L}$$

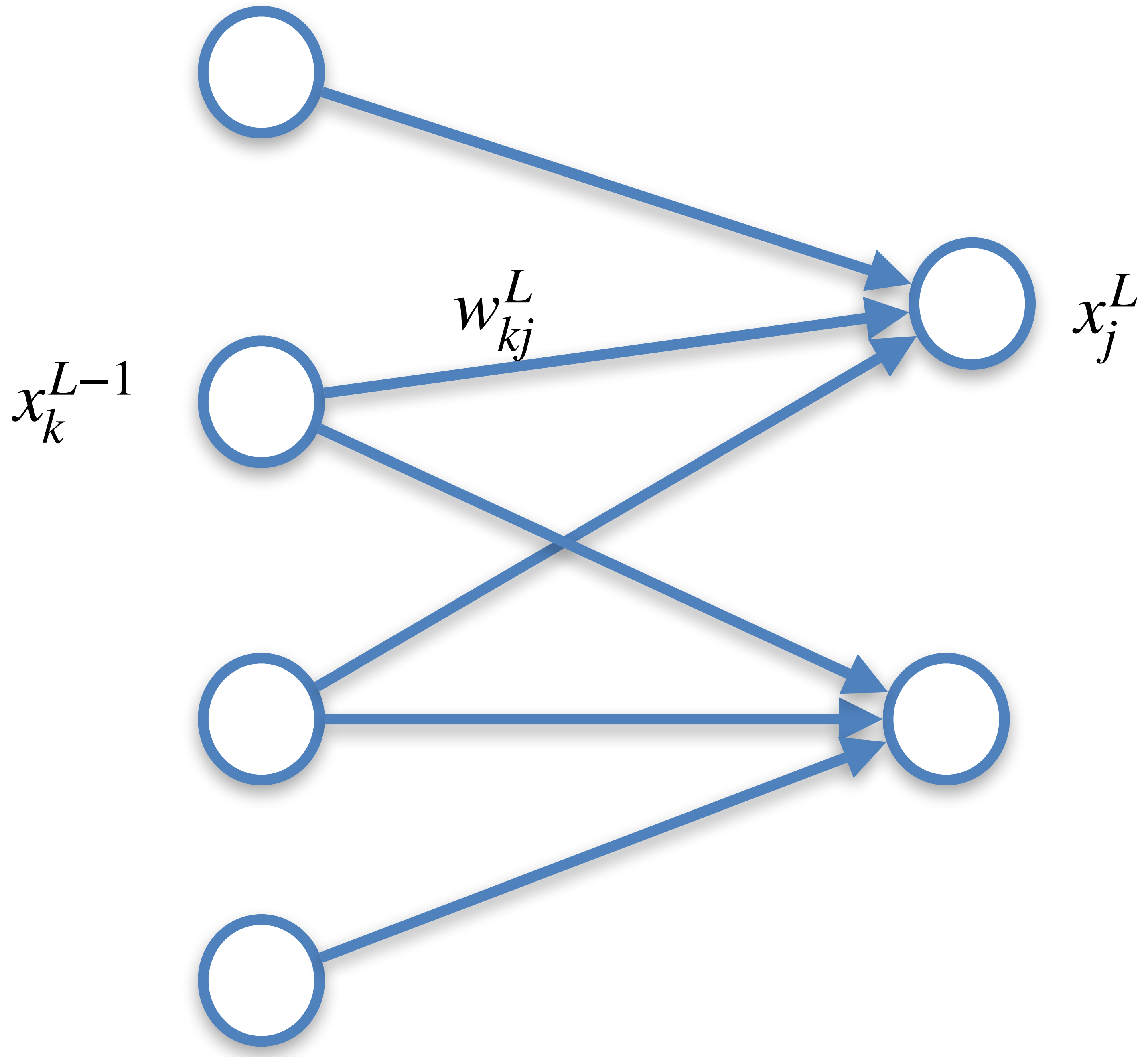
Now the bias is now a vector!

Backpropagation algorithm

As before we need to compute the gradients

For the last layer, for example, before we computed

$$\frac{\partial E}{\partial w^L}$$ Now w is a matrix! $$\frac{\partial E}{\partial w^L} \rightarrow \frac{\partial E}{\partial w^L_{ij}}$$



We also need to consider the bias

$$\frac{\partial E}{\partial b^L}$$ Now the bias is now a vector! $$\frac{\partial E}{\partial b^L} \rightarrow \frac{\partial E}{\partial \mathbf{b}^L}$$

Interested in seeing how these gradients are computed?
Interested in seeing how backprogation works for real?

Make your choice!

Interested in seeing how these gradients are computed?
Interested in seeing how backprogation works for real?

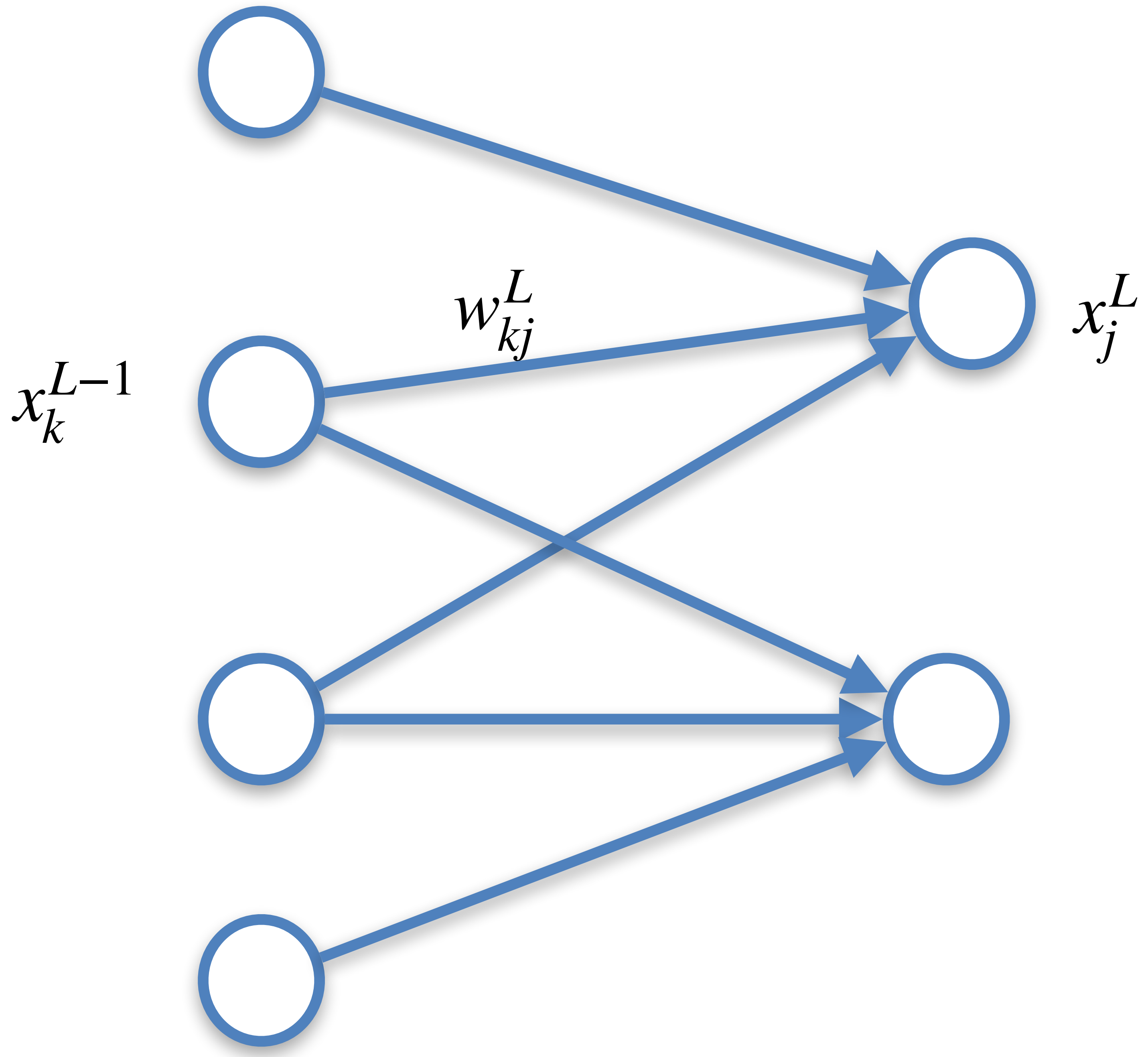
Make your choice!

If the answer is no skip to slide 73

Backpropagation algorithm

In the last layer we have

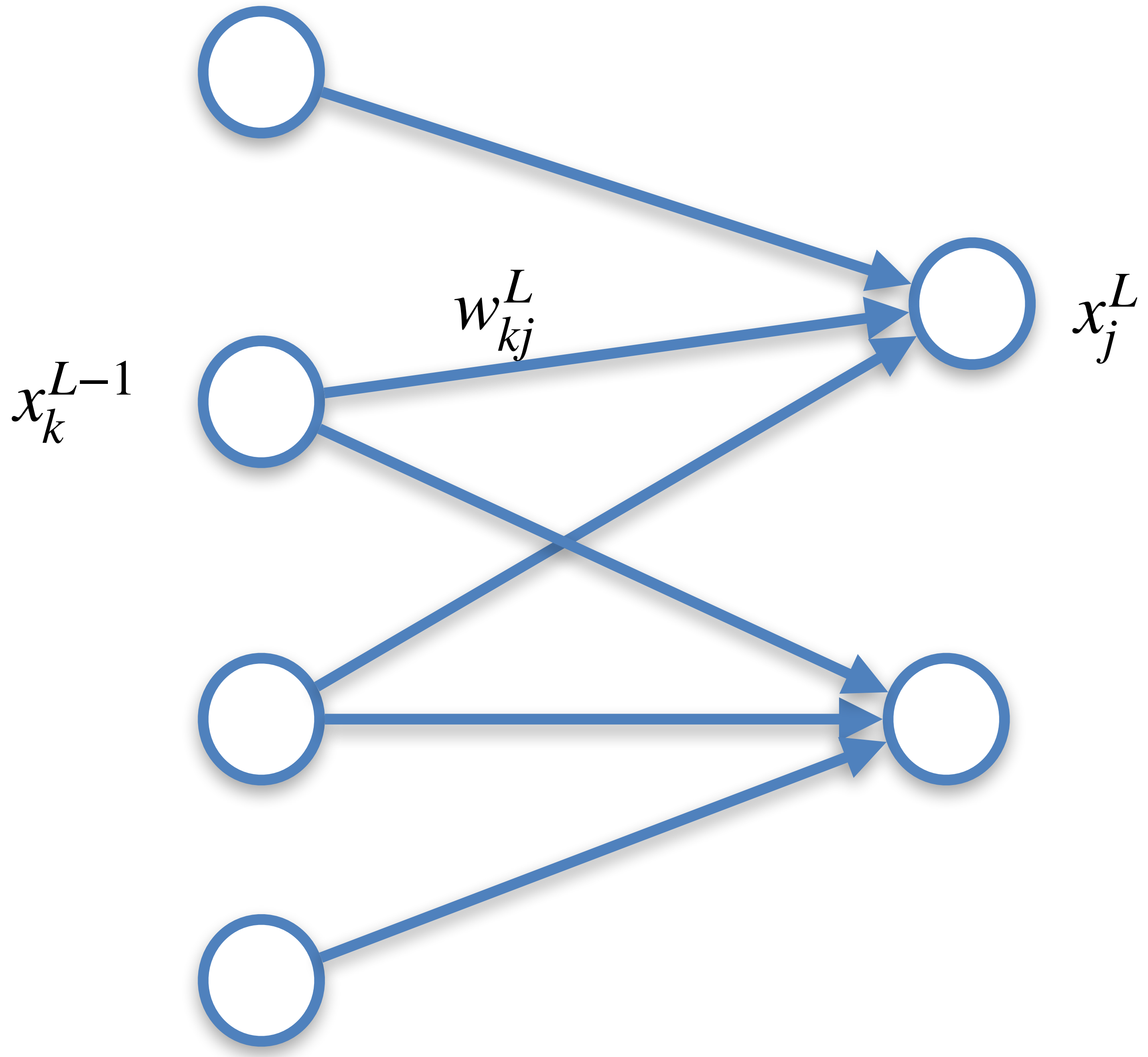
$$x_j^L = \sigma(z_j^L)$$



Backpropagation algorithm

In the last layer we have

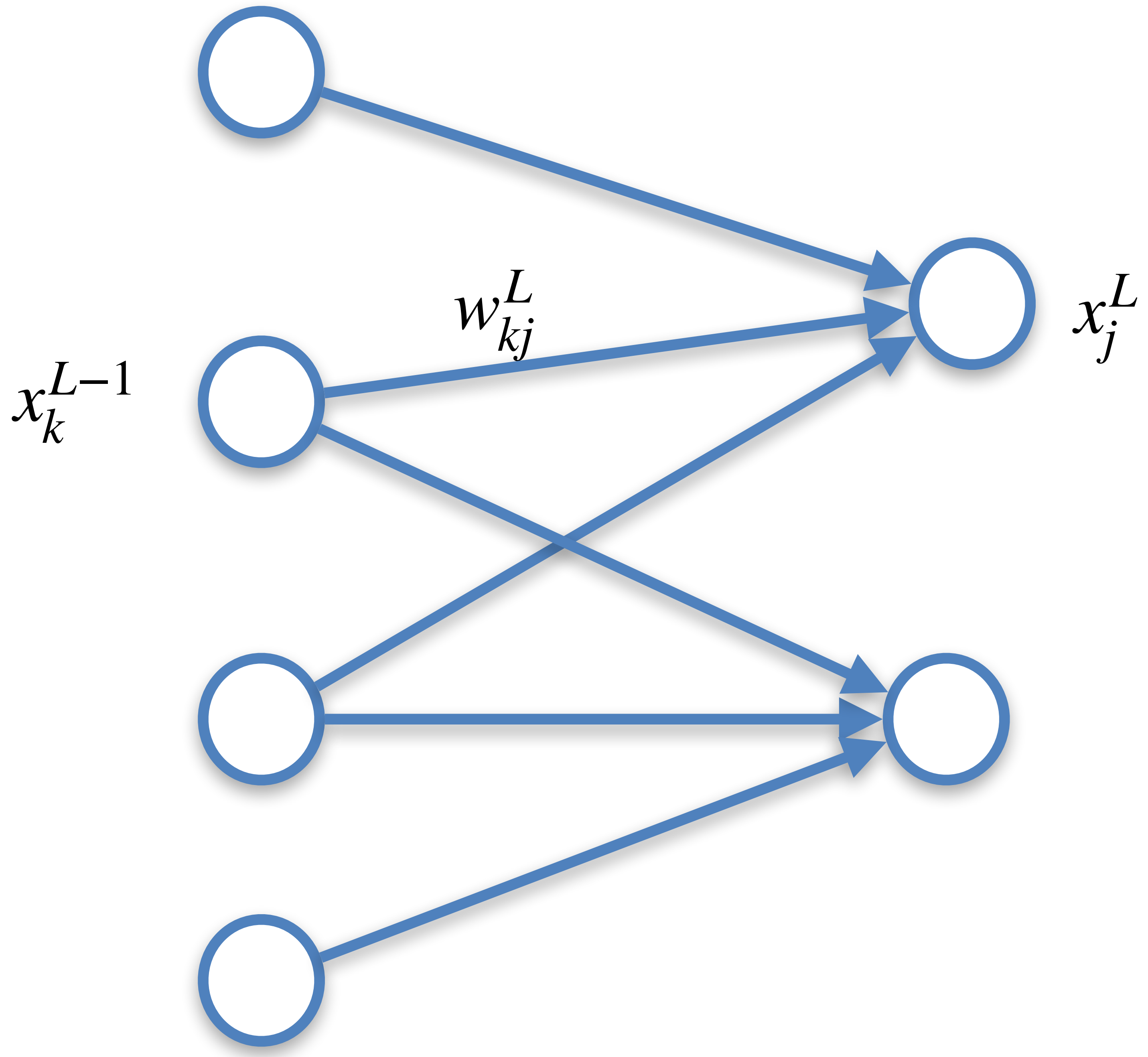
$$x_j^L = \sigma(z_j^L) \qquad z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L$$



Backpropagation algorithm

In the last layer we have

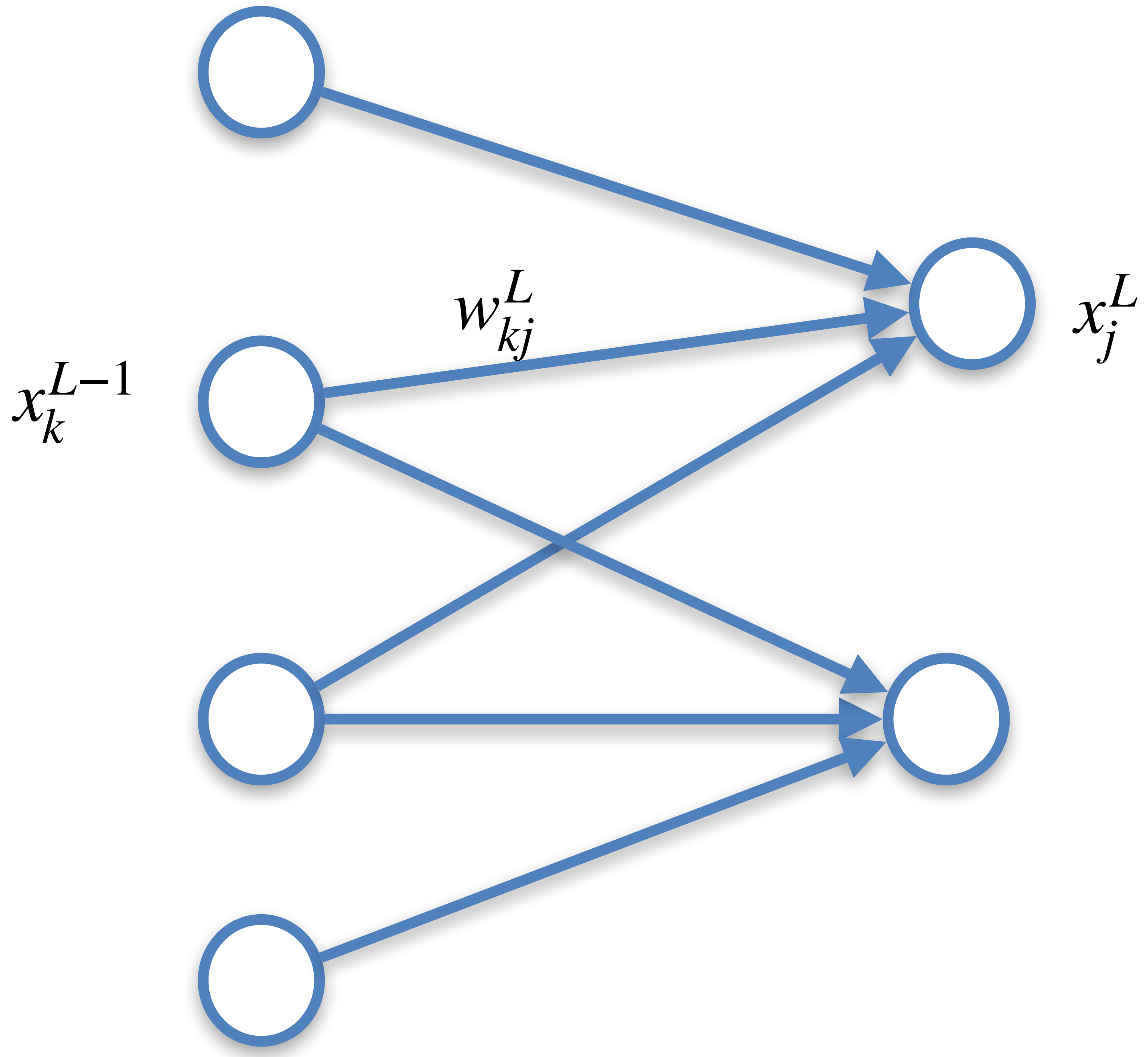
$$x_j^L = \sigma(z_j^L) \qquad z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



Backpropagation algorithm

In the last layer we have

$$x_j^L = \sigma(z_j^L) \qquad z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

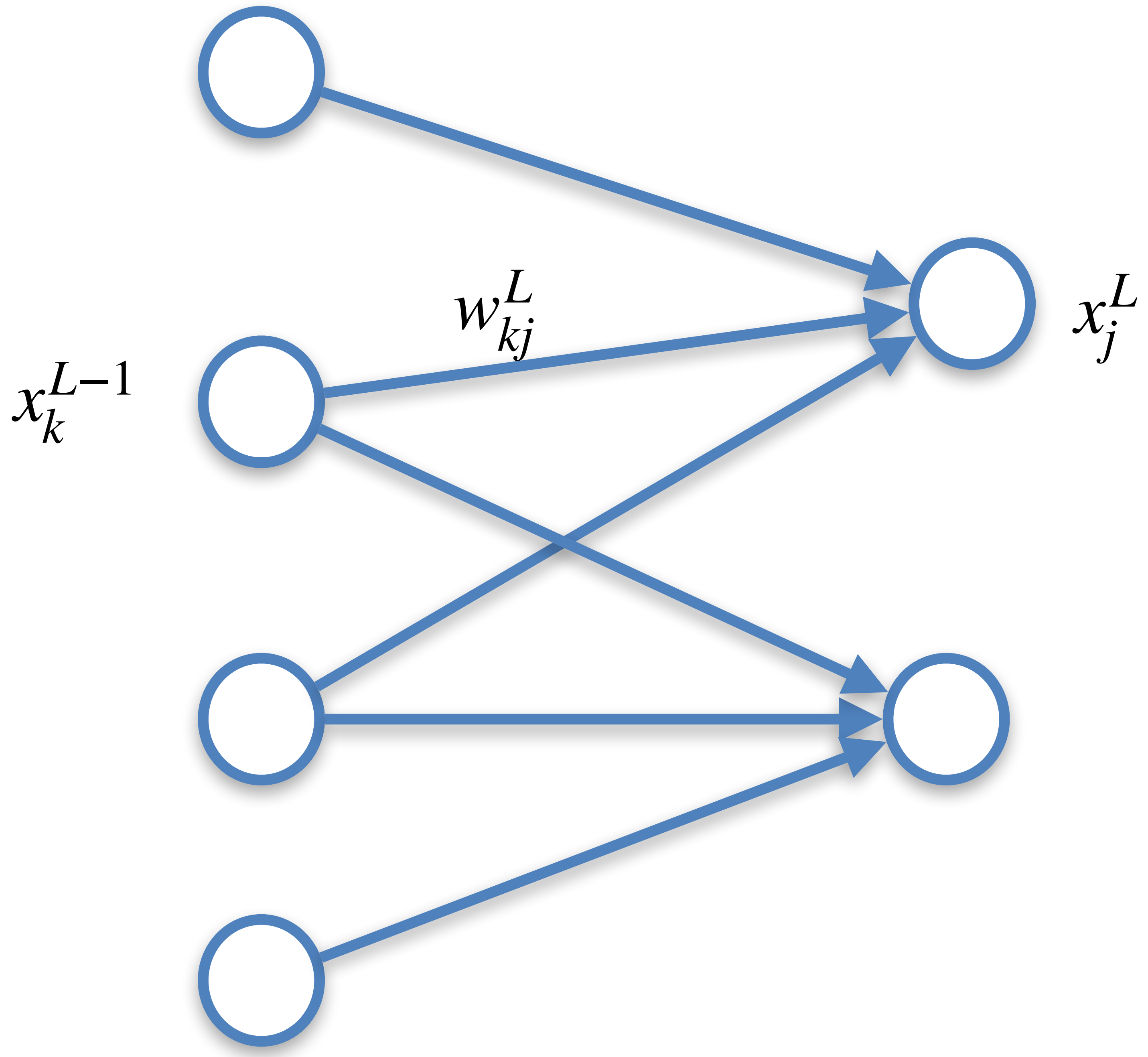


To compute the gradients we need the chain rule as before

Backpropagation algorithm

In the last layer we have

$$x_j^L = \sigma(z_j^L) \qquad z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



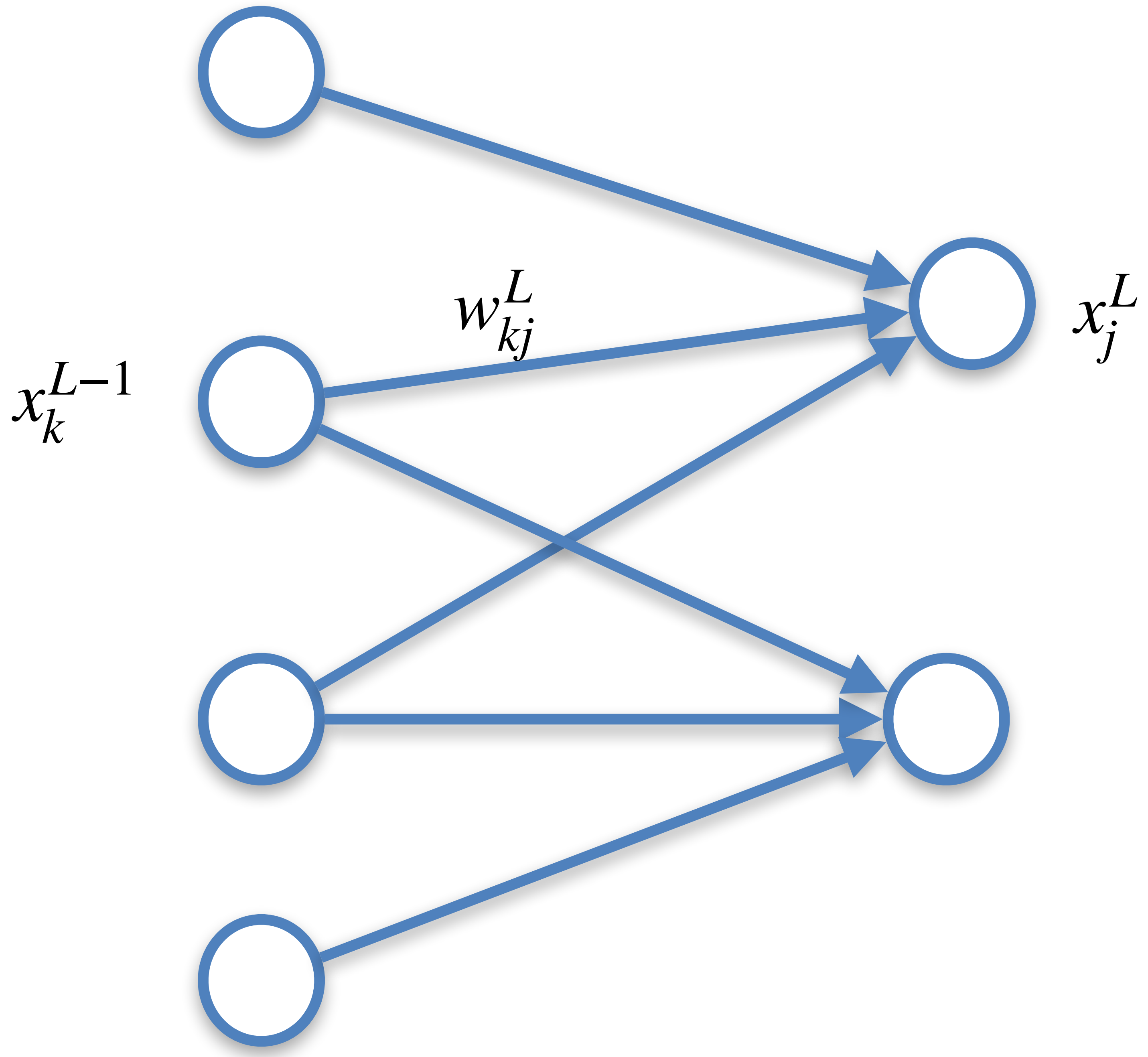
To compute the gradients we need the chain rule as before

We start the derivation by defining δ_j^L as

Backpropagation algorithm

In the last layer we have

$$x_j^L = \sigma(z_j^L) \qquad z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



To compute the gradients we need the chain rule as before

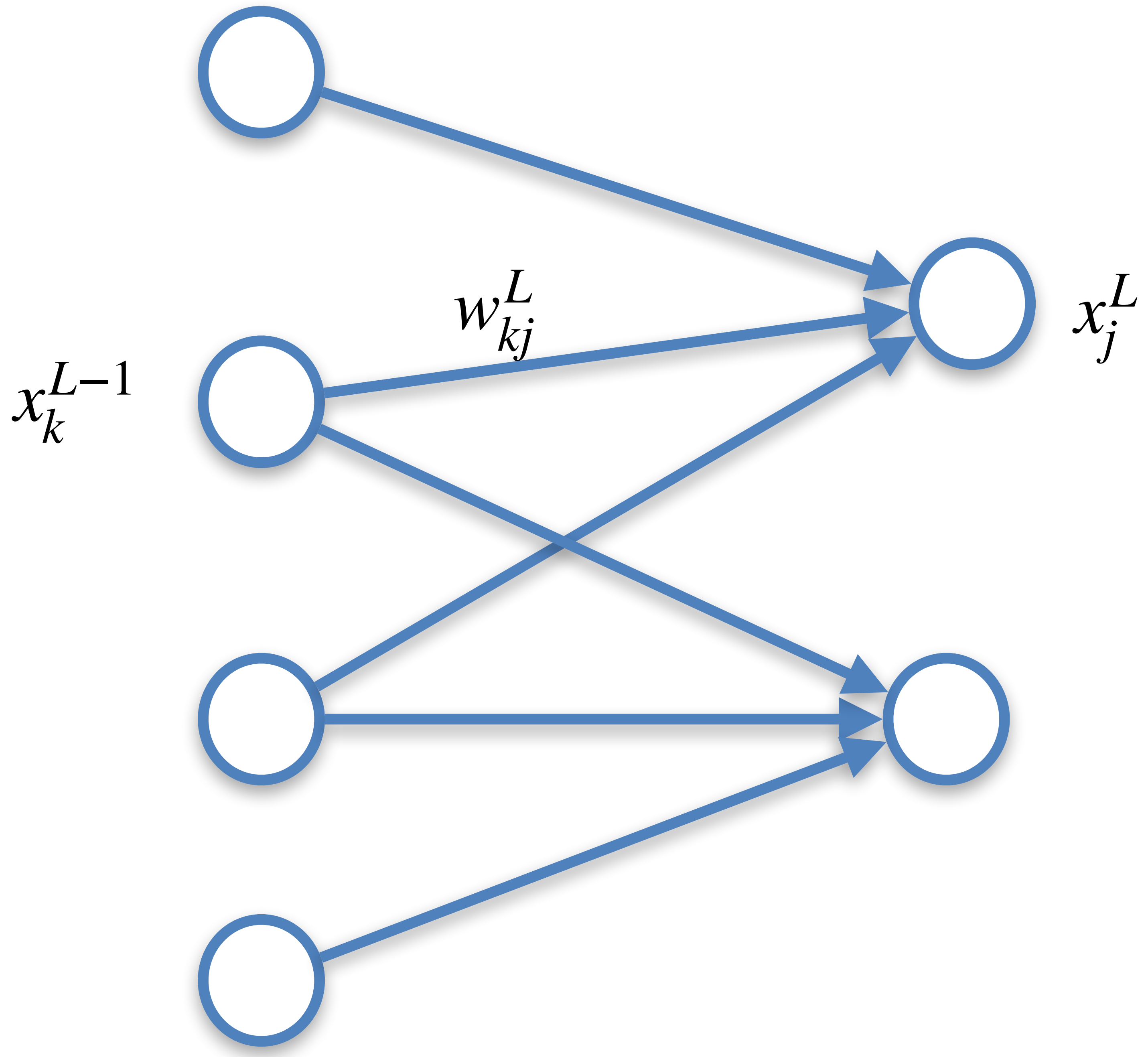
We start the derivation by defining δ_j^L as

$$\frac{\partial E}{\partial z_j^L} = \delta_j^L$$

Backpropagation algorithm

In the last layer we have

$$x_j^L = \sigma(z_j^L) \qquad z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



To compute the gradients we need the chain rule as before

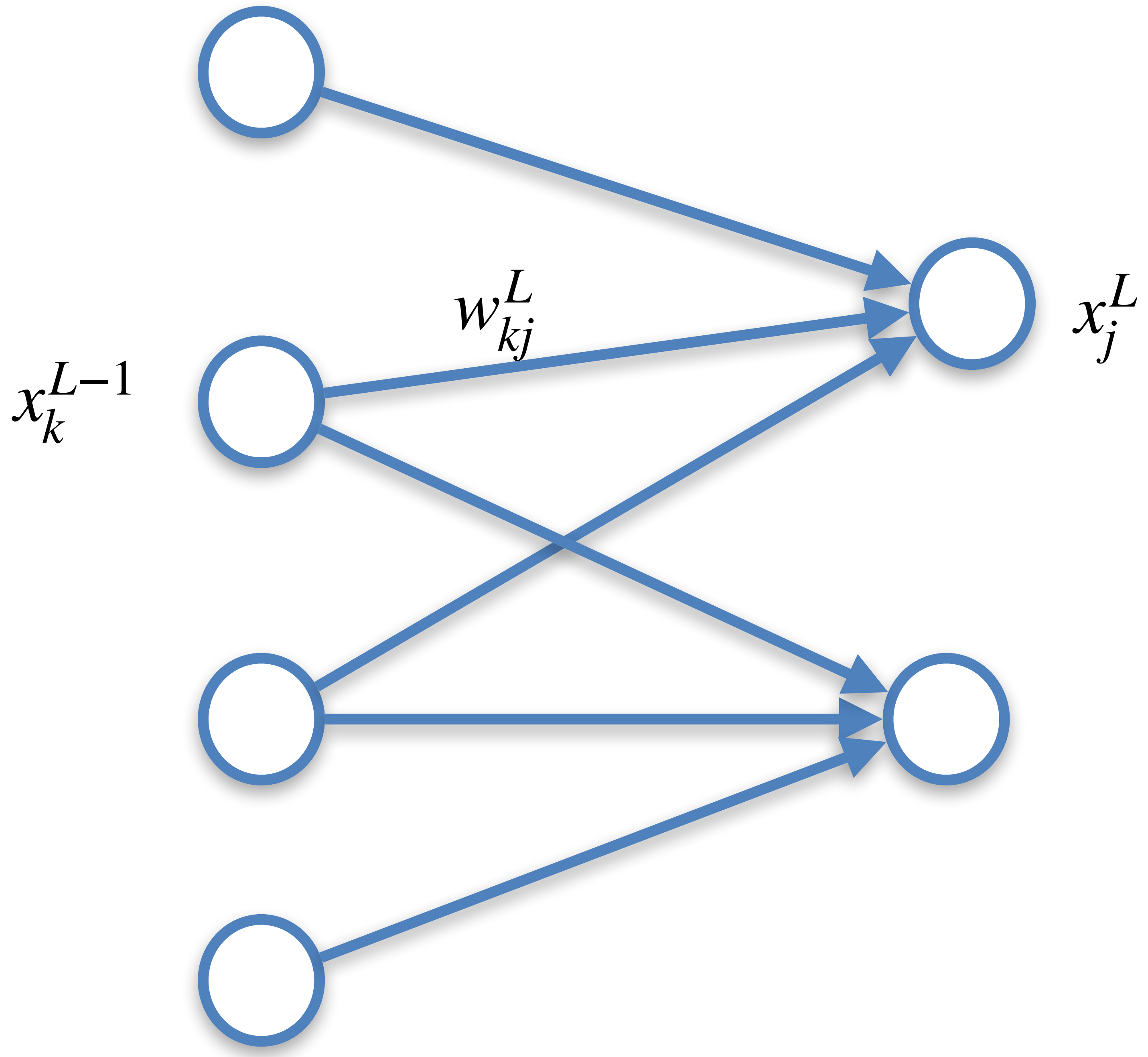
We start the derivation by defining δ_j^L as

$$\frac{\partial E}{\partial z_j^L} = \delta_j^L = \frac{\partial E}{\partial x_j^L}\frac{\partial x_j^L}{\partial z_j^L}$$

Backpropagation algorithm

In the last layer we have

$$x_j^L = \sigma(z_j^L) \qquad z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



To compute the gradients we need the chain rule as before

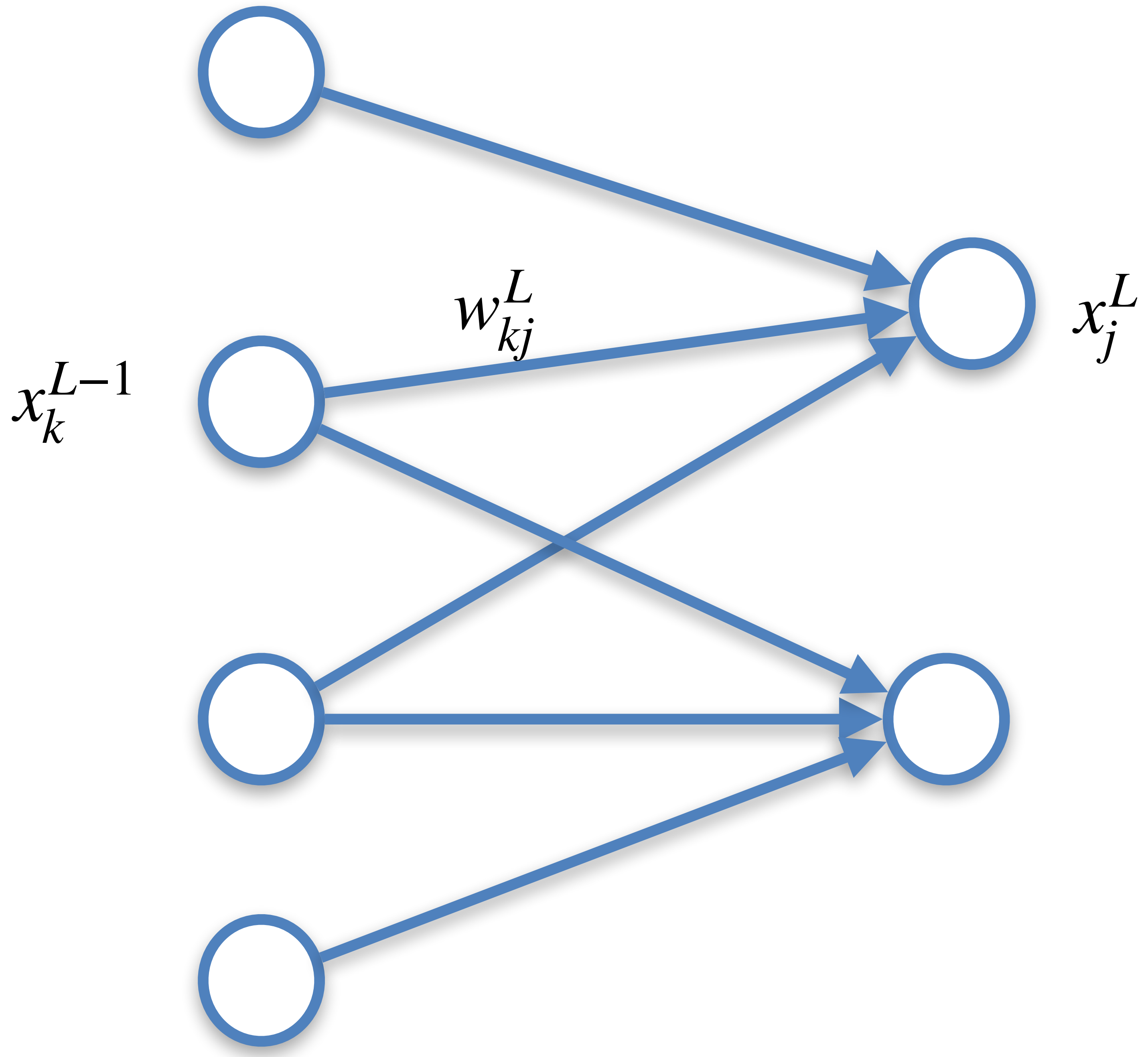
We start the derivation by defining δ_j^L as

$$\frac{\partial E}{\partial z_j^L} = \delta_j^L = \frac{\partial E}{\partial x_j^L}\frac{\partial x_j^L}{\partial z_j^L}$$

Backpropagation algorithm

In the last layer we have

$$x_j^L = \sigma(z_j^L) \qquad z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



To compute the gradients we need the chain rule as before

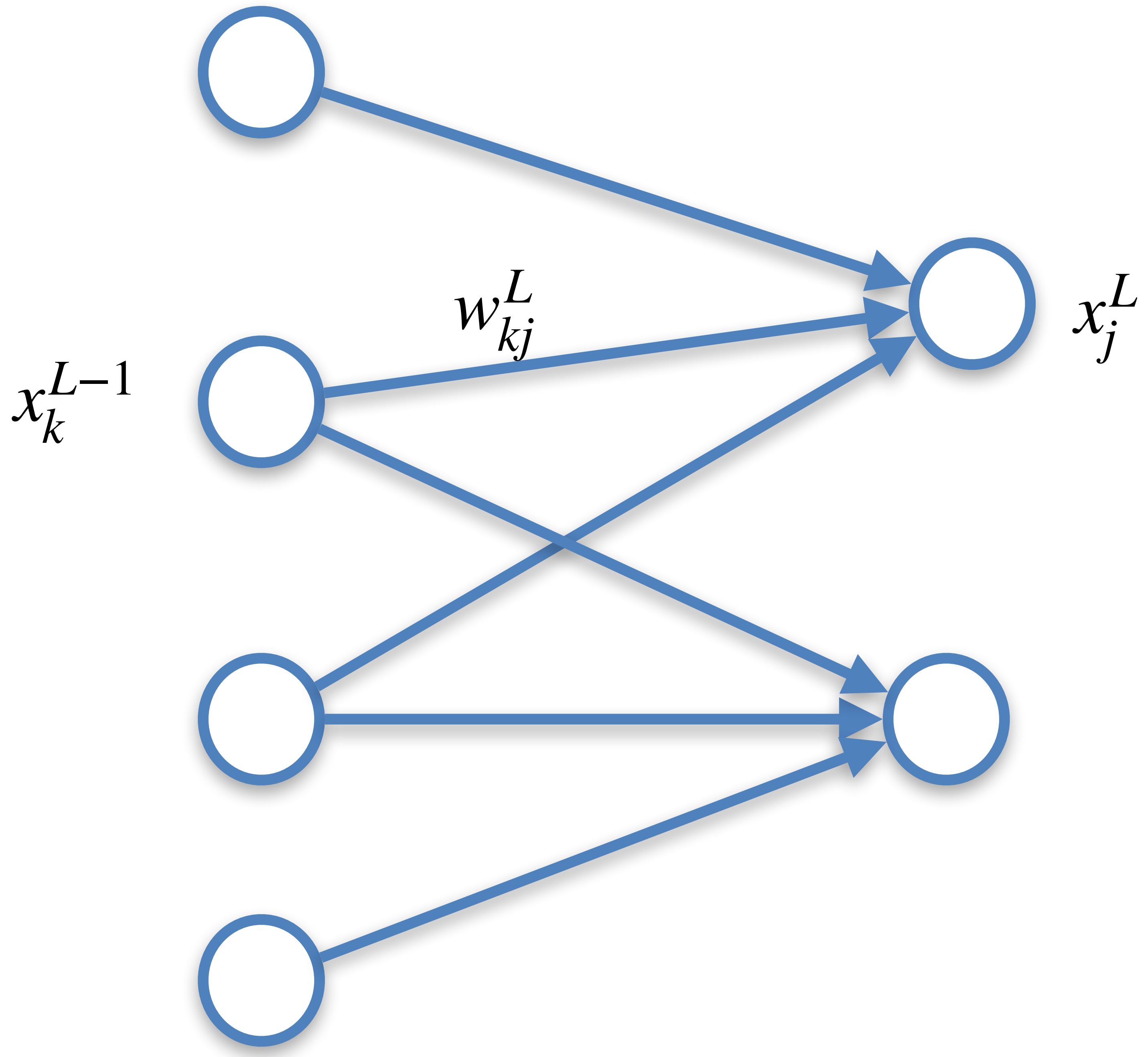
We start the derivation by defining δ_j^L as

$$\frac{\partial E}{\partial z_j^L} = \delta_j^L = \frac{\partial E}{\partial x_j^L}\frac{\partial x_j^L}{\partial z_j^L} = (x_j^L - y_j)$$

Backpropagation algorithm

In the last layer we have

$$x_j^L = \sigma(z_j^L) \qquad z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



To compute the gradients we need the chain rule as before

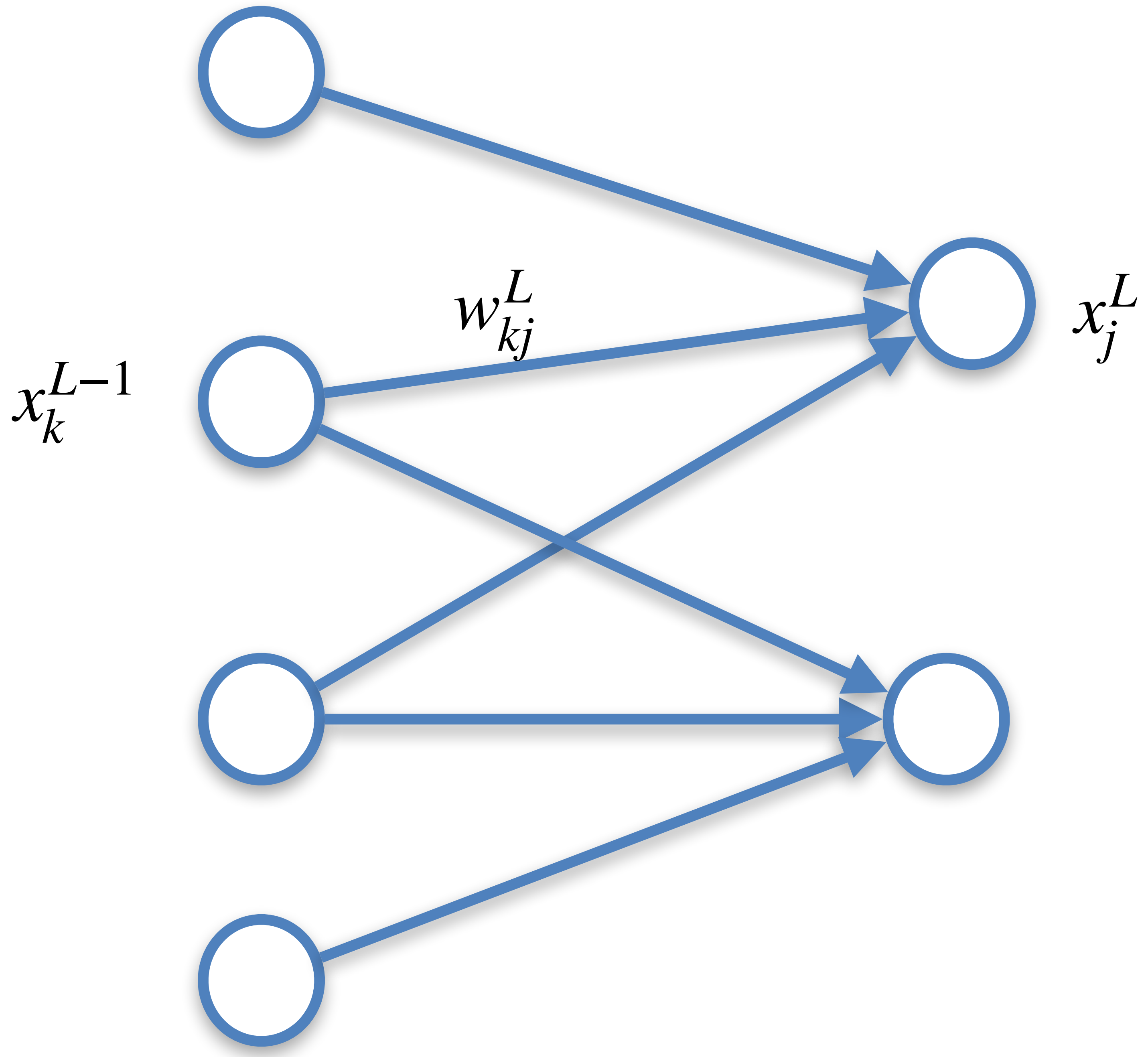
We start the derivation by defining δ_j^L as

$$\frac{\partial E}{\partial z_j^L} = \delta_j^L = \frac{\partial E}{\partial x_j^L}\frac{\partial x_j^L}{\partial z_j^L} = (x_j^L - y_j)$$

Backpropagation algorithm

In the last layer we have

$$x_j^L = \sigma(z_j^L) \qquad z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



To compute the gradients we need the chain rule as before

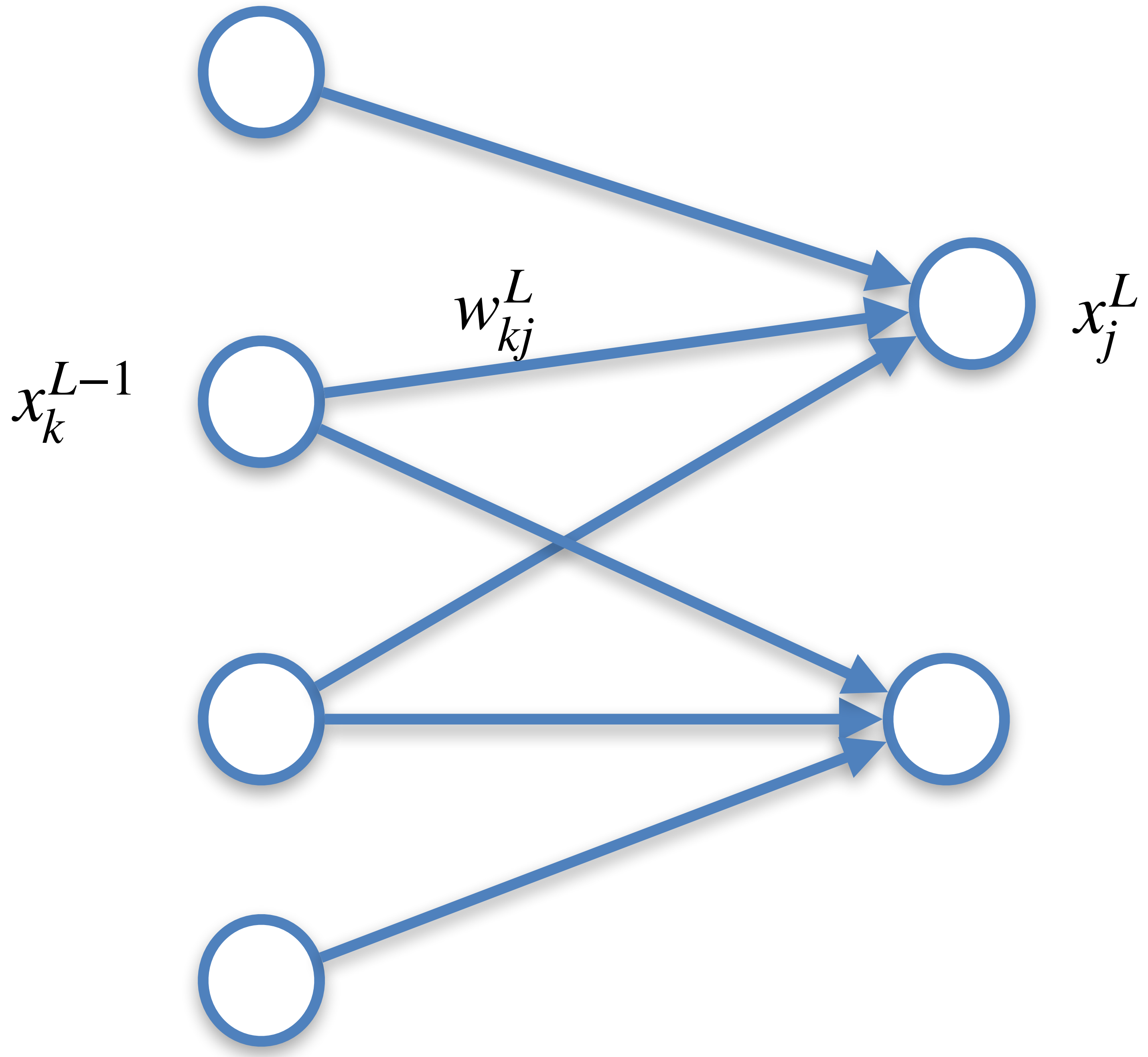
We start the derivation by defining δ_j^L as

$$\frac{\partial E}{\partial z_j^L} = \delta_j^L = \frac{\partial E}{\partial x_j^L}\frac{\partial x_j^L}{\partial z_j^L} = (x_j^L - y_j)\,\sigma'(z_j^L)$$

Backpropagation algorithm

In the last layer we have

$$x_j^L = \sigma(z_j^L) \qquad z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



To compute the gradients we need the chain rule as before

We start the derivation by defining δ_j^L as

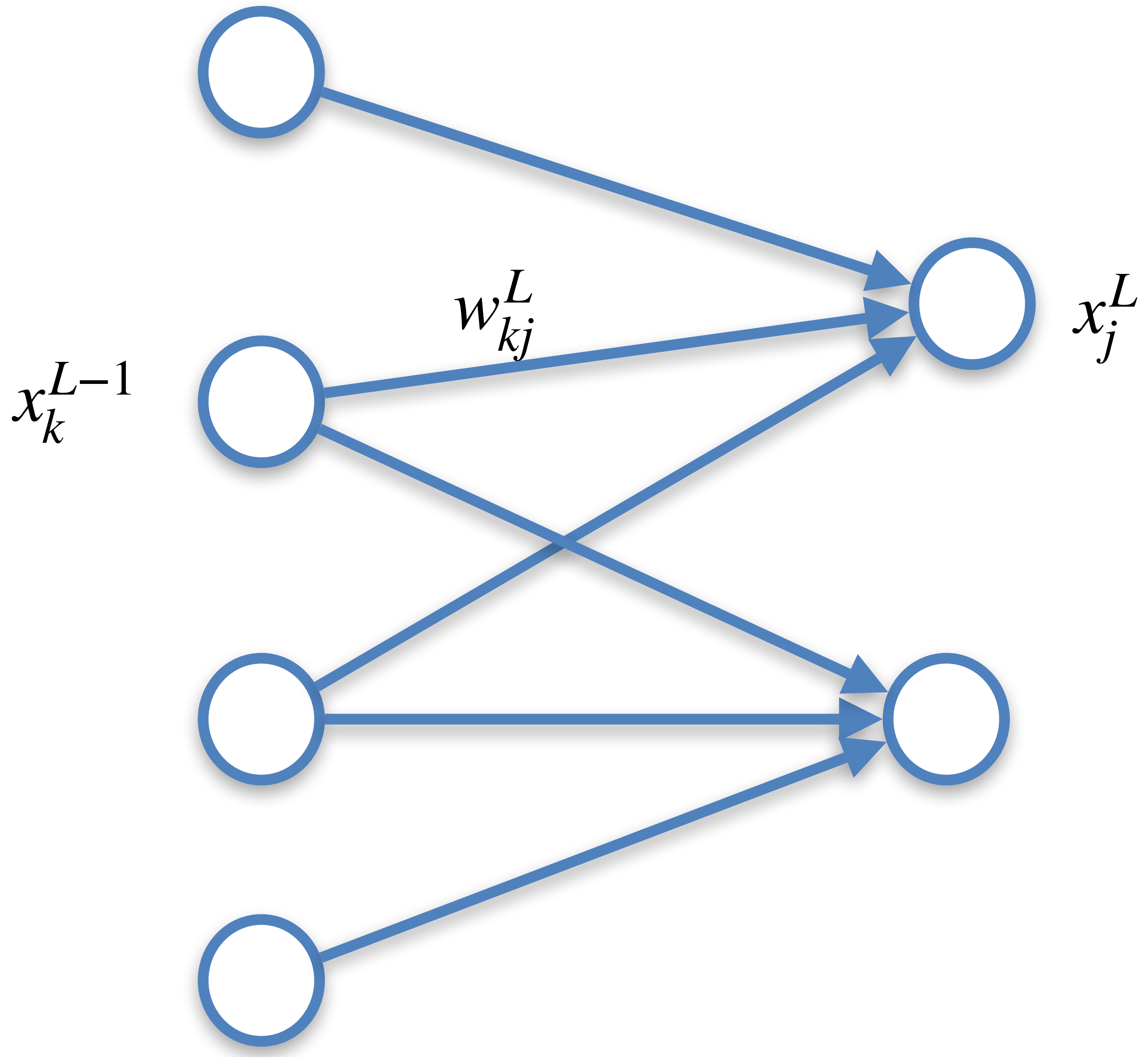
$$\frac{\partial E}{\partial z_j^L} = \delta_j^L = \frac{\partial E}{\partial x_j^L}\frac{\partial x_j^L}{\partial z_j^L} = (x_j^L - y_j)\,\sigma'(z_j^L)$$

This is the first step only since we saw that

$$\frac{\partial}{\partial w^L}E_0 = \frac{\partial E_0}{\partial x^L}\frac{\partial x^L}{\partial z^L}\frac{\partial z^L}{\partial w^L}$$

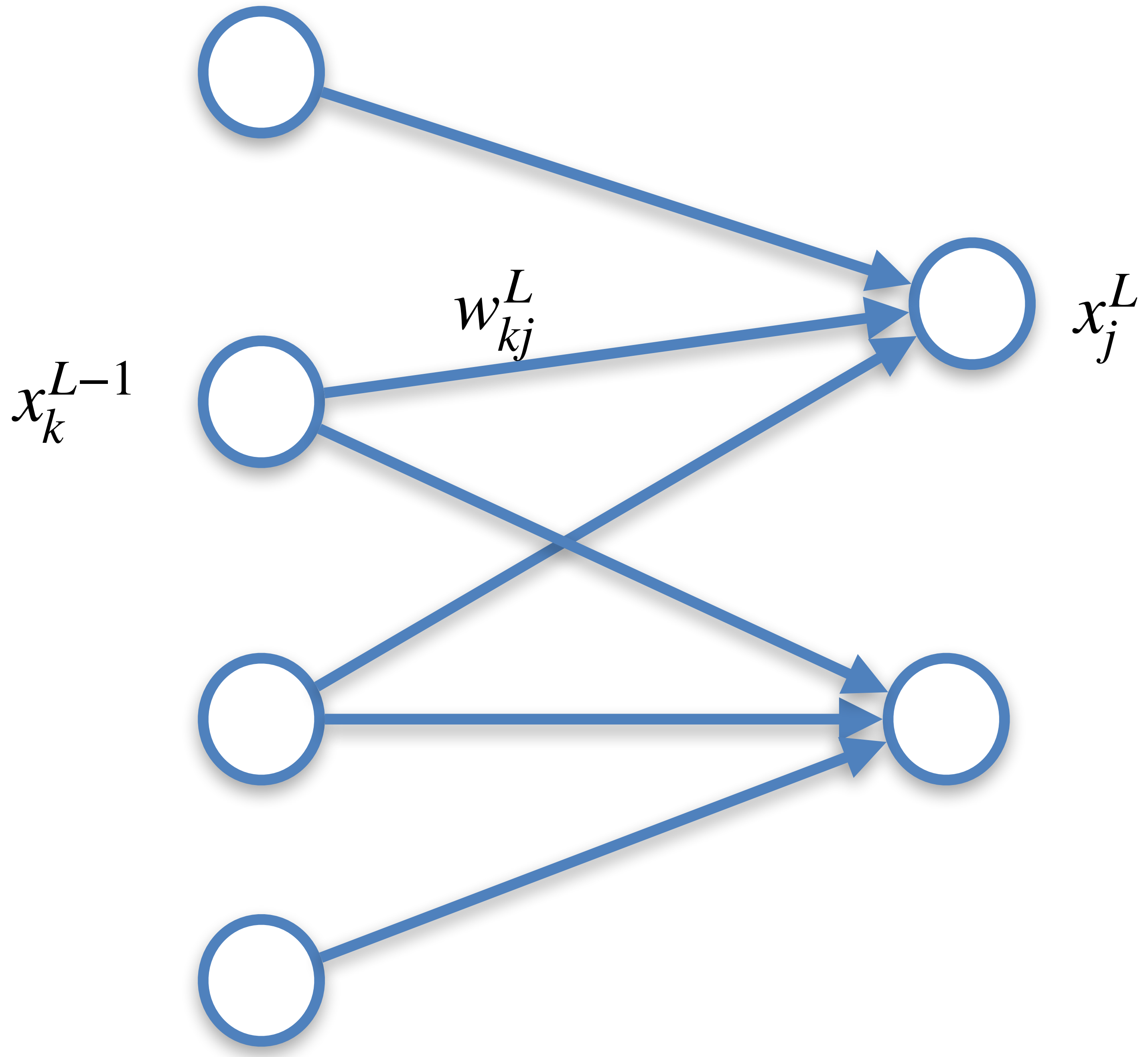
Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



Backpropagation algorithm

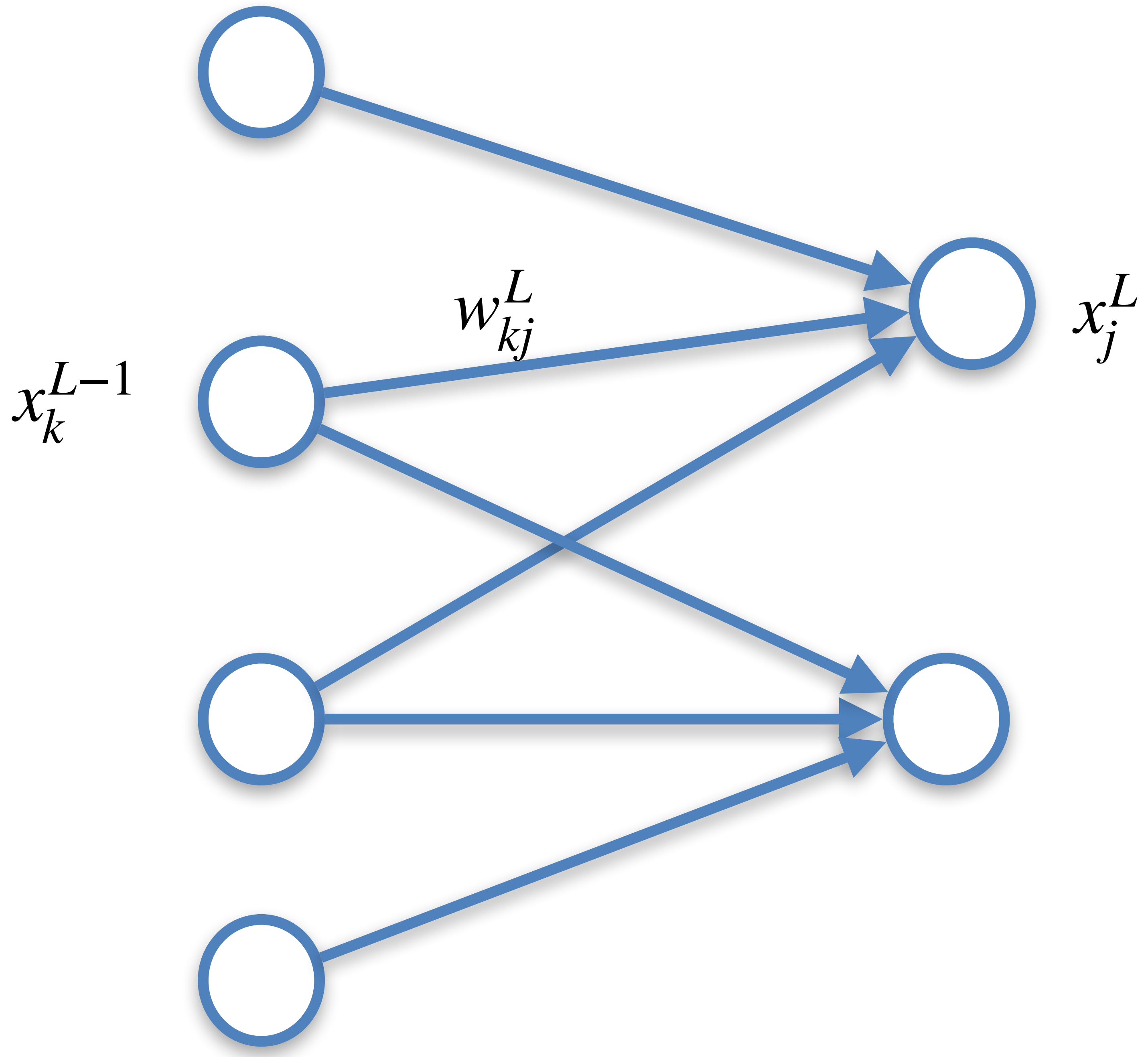
$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



What is the value of delta in any previous layer??

Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



What is the value of delta in any previous layer??

$$\frac{\partial E}{\partial z_j^l} = \delta_j^l \quad \text{for all } l = L-1, \ldots, 1.$$

Backpropagation algorithm

Here it is where the idea of backpropagation comes into focus

Backpropagation algorithm

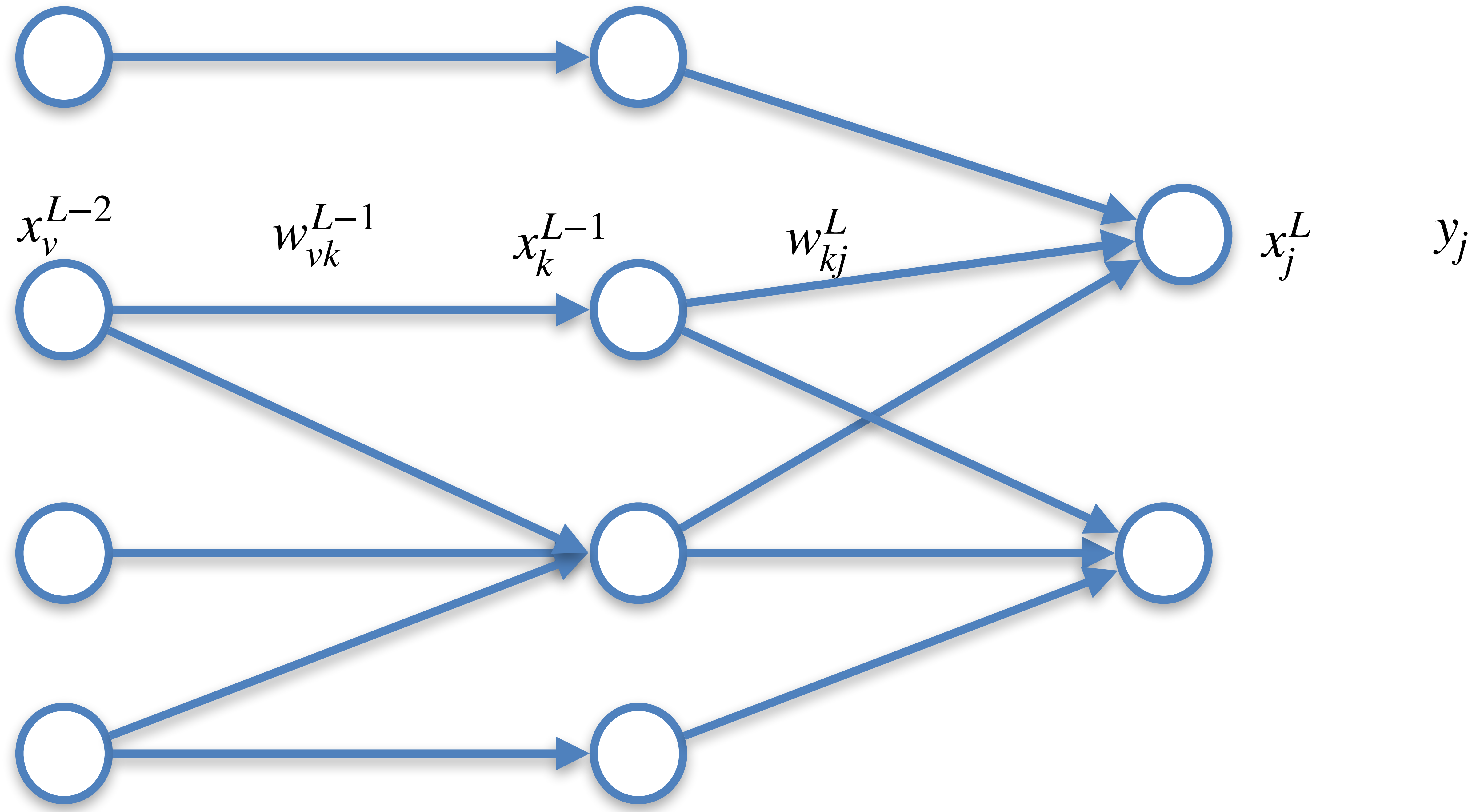
Here it is where the idea of backpropagation comes into focus

Idea: we can compute the delta value in each generic layer by looking at the layer before!

Backpropagation algorithm

Here it is where the idea of backpropagation comes into focus

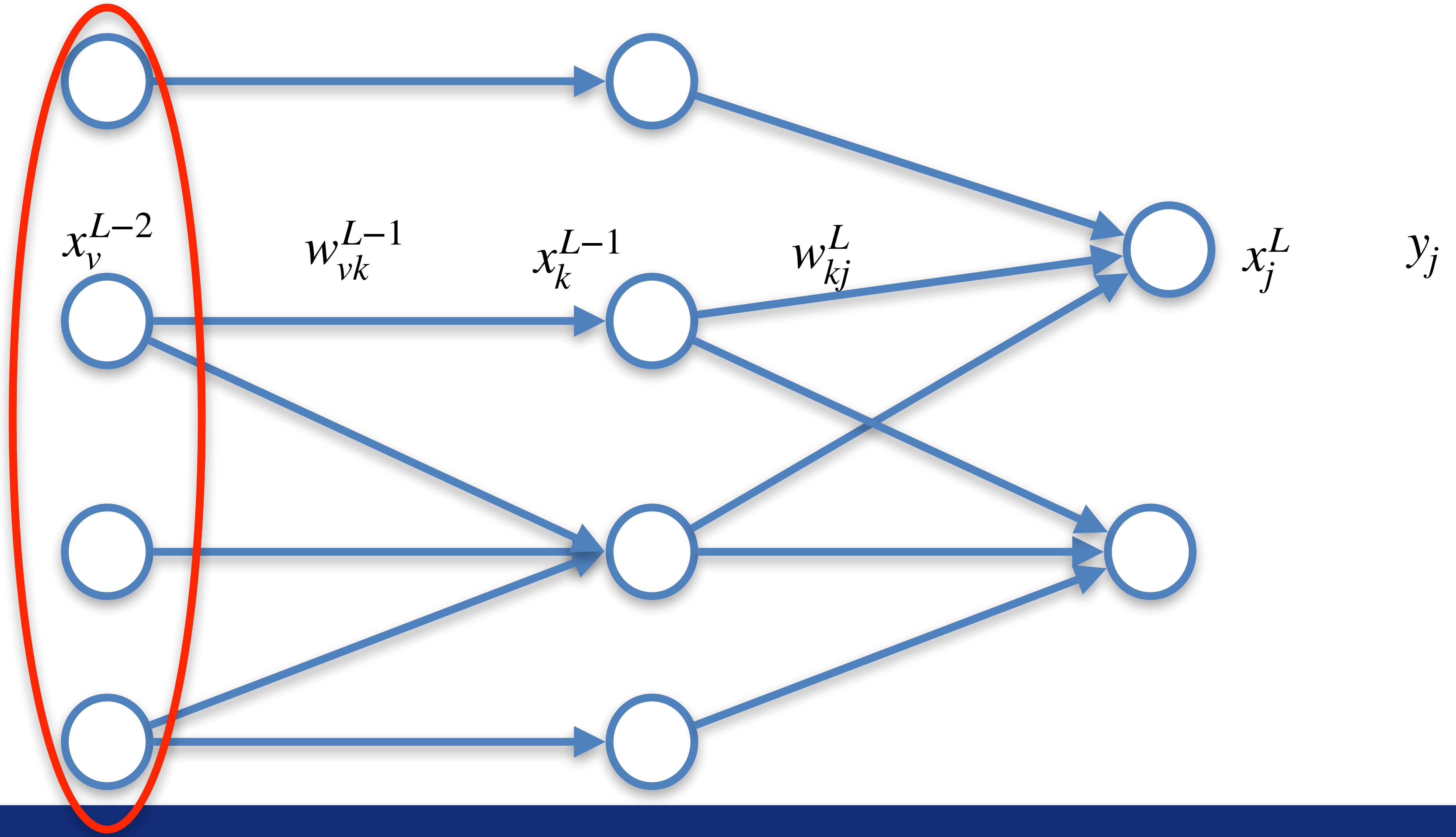
Idea: we can compute the delta value in each generic layer by looking at the layer before!



Backpropagation algorithm

Here it is where the idea of backpropagation comes into focus

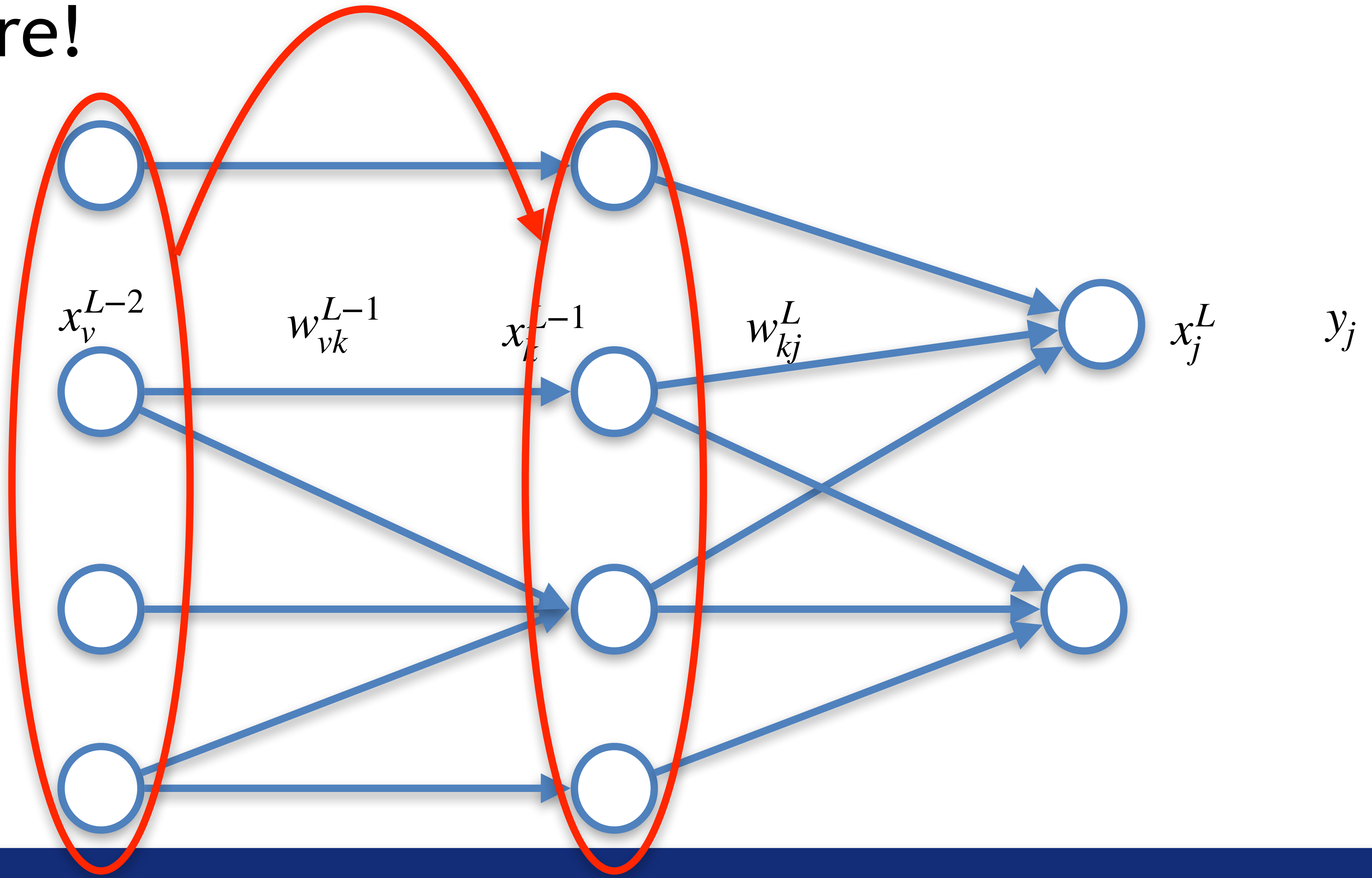
Idea: we can compute the delta value in each generic layer by looking at the layer before!



Backpropagation algorithm

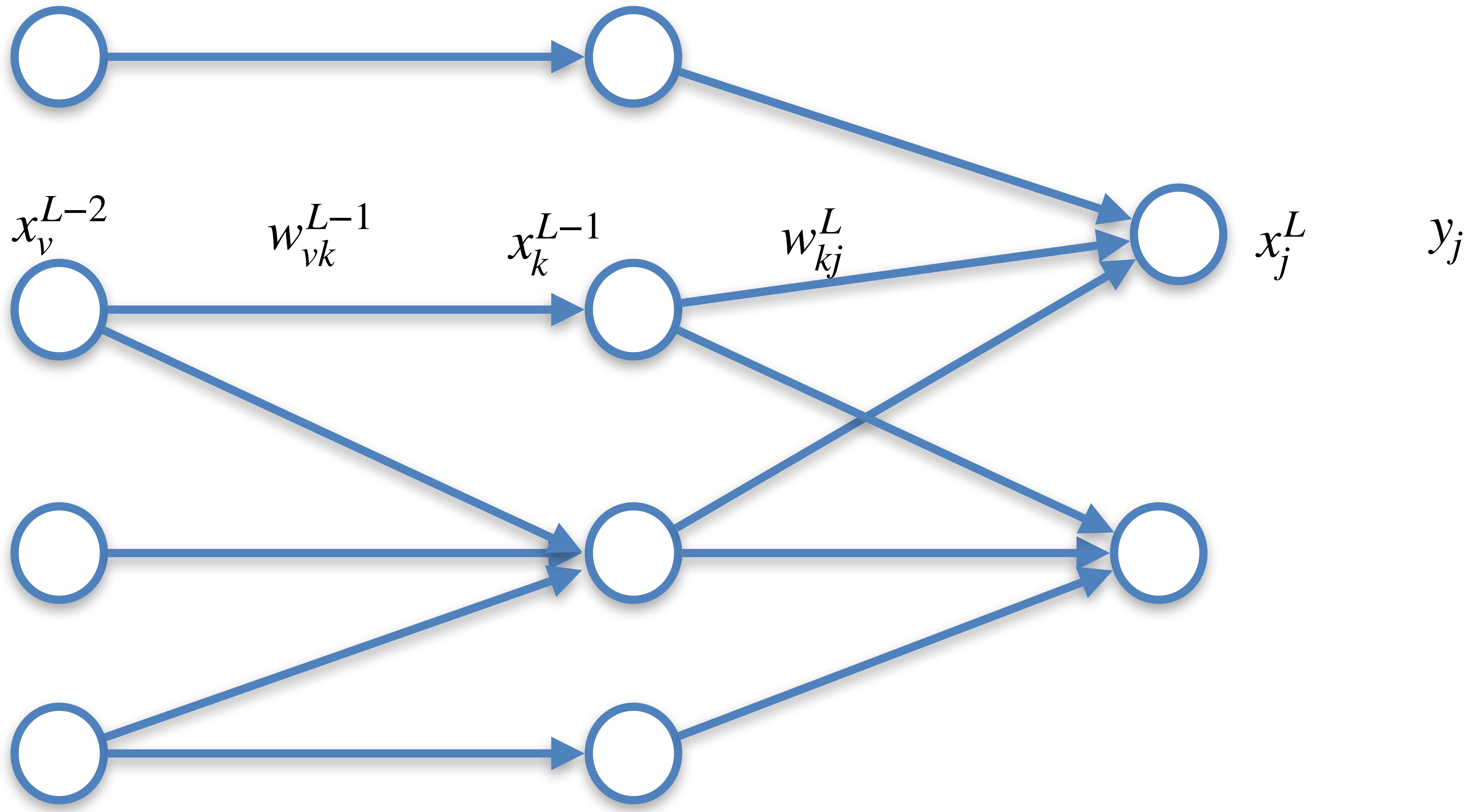
Here it is where the idea of backpropagation comes into focus

Idea: we can compute the delta value in each generic layer by looking at the layer before!



Backpropagation algorithm

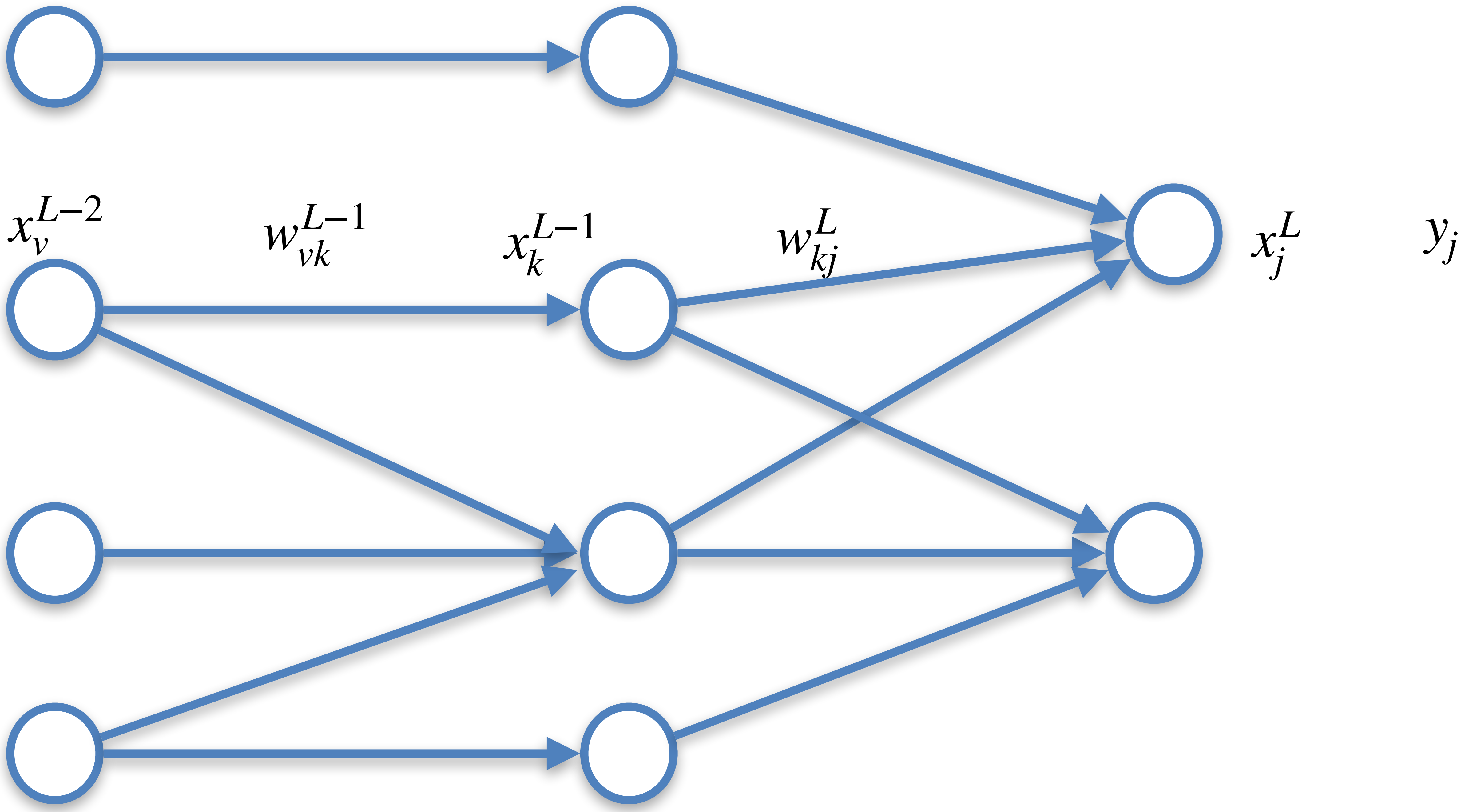
$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$



Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

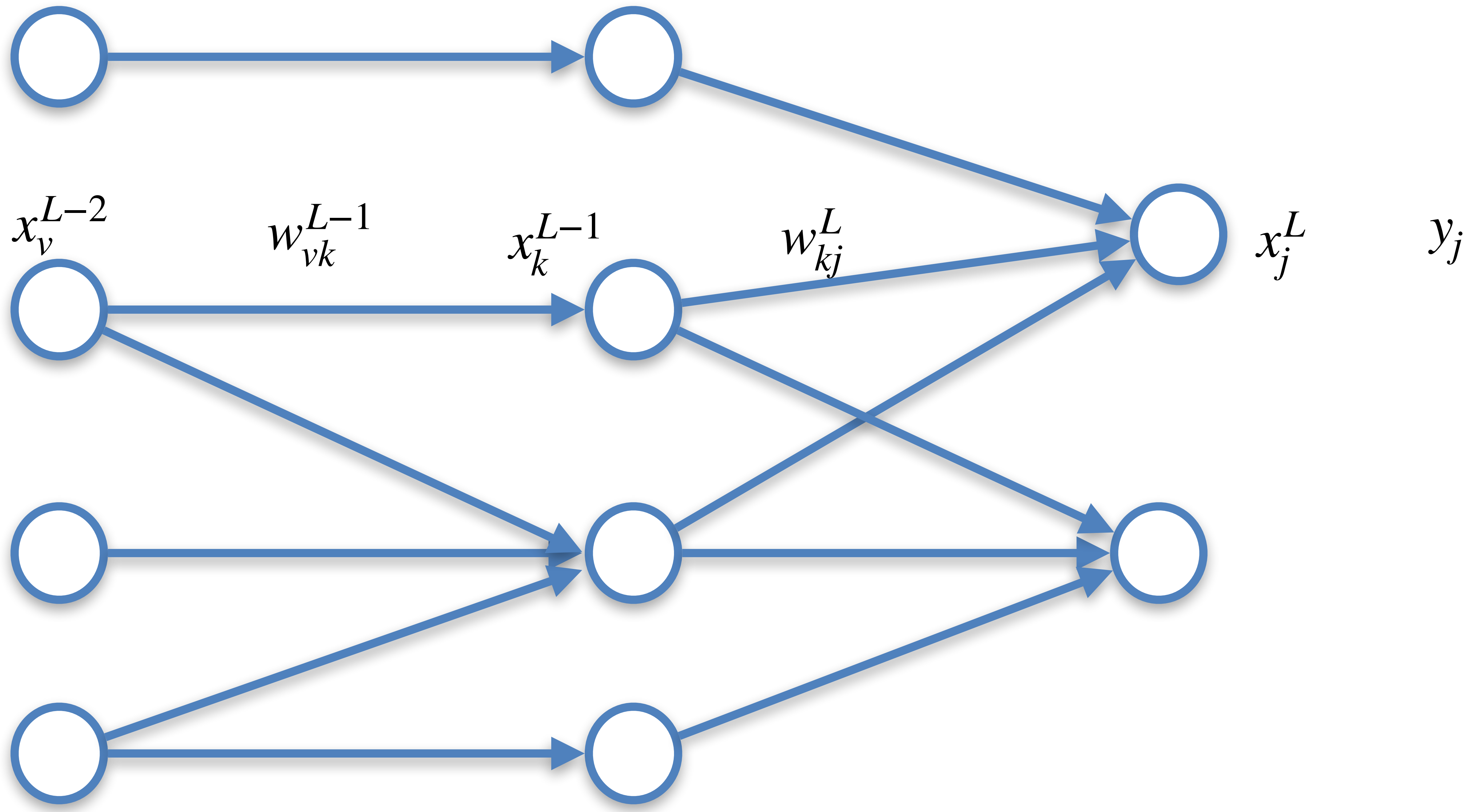
$$\frac{\partial E}{\partial z_j^l} = \delta_j^l$$



Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

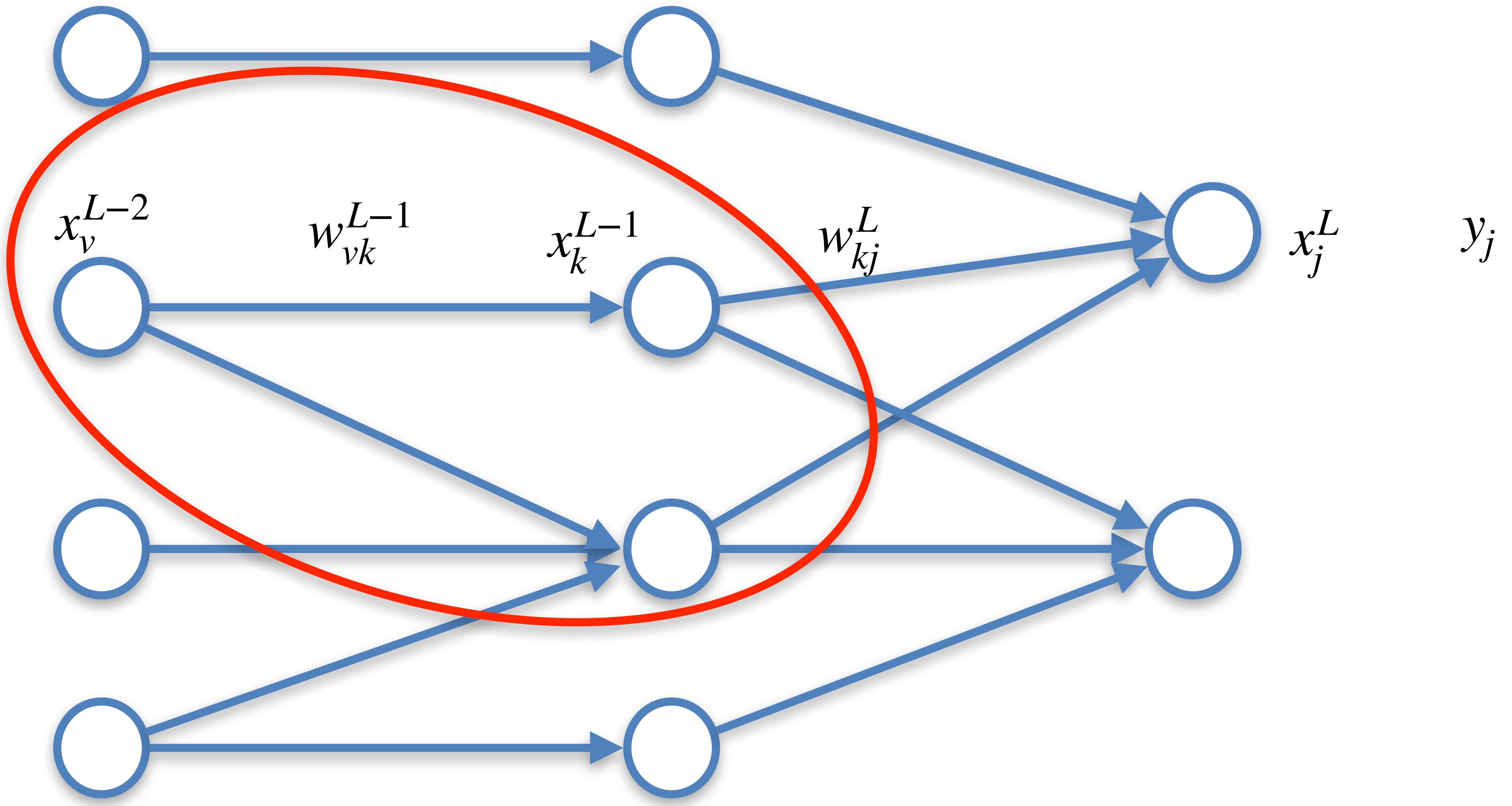
$$\frac{\partial E}{\partial z_j^l} = \delta_j^l = \sum_v \frac{\partial E}{\partial z_v^{l+1}} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$



Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

$$\frac{\partial E}{\partial z_j^l} = \delta_j^l = \sum_v \frac{\partial E}{\partial z_v^{l+1}} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$



Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

$$\frac{\partial E}{\partial z_j^l} = \delta_j^l = \sum_v \frac{\partial E}{\partial z_v^{l+1}} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

$$\frac{\partial E}{\partial z_j^l} = \delta_j^l = \sum_v \frac{\partial E}{\partial z_v^{l+1}} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

$$= \sum_v \delta_v^{l+1}$$

Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

$$\frac{\partial E}{\partial z_j^l} = \delta_j^l = \sum_v \frac{\partial E}{\partial z_v^{l+1}} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

$$= \sum_v \delta_v^{l+1} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

$$\frac{\partial E}{\partial z_j^l} = \delta_j^l = \sum_v \frac{\partial E}{\partial z_v^{l+1}} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

$$z_v^{l+1} = \sum_i w_{iv}^{l+1} x_i^l + b_v^{l+1}$$

$$x_i^l = \sigma(z_i^l)$$

$$= \sum_v \delta_v^{l+1} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

$$\frac{\partial E}{\partial z_j^l} = \delta_j^l = \sum_v \frac{\partial E}{\partial z_v^{l+1}} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

$$z_v^{l+1} = \sum_i w_{iv}^{l+1} x_i^l + b_v^{l+1}$$

$$x_i^l = \sigma(z_i^l)$$

$$= \sum_v \delta_v^{l+1} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

$$\frac{\partial z_v^{l+1}}{\partial z_j^l}$$

Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

$$\frac{\partial E}{\partial z_j^l} = \delta_j^l = \sum_v \frac{\partial E}{\partial z_v^{l+1}} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

$$= \sum_v \delta_v^{l+1} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

$$z_v^{l+1} = \sum_i w_{iv}^{l+1} x_i^l + b_v^{l+1}$$

$$x_i^l = \sigma(z_i^l)$$

$$\frac{\partial z_v^{l+1}}{\partial z_j^l} = \sum_i w_{iv}^{l+1} \frac{\partial x_i^l}{\partial z_j^l}$$

Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

$$\frac{\partial E}{\partial z_j^l} = \delta_j^l = \sum_v \frac{\partial E}{\partial z_v^{l+1}} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

$$z_v^{l+1} = \sum_i w_{iv}^{l+1} x_i^l + b_v^{l+1}$$

$$x_i^l = \sigma(z_i^l)$$

$$= \sum_v \delta_v^{l+1} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

$$\frac{\partial z_v^{l+1}}{\partial z_j^l} = \sum_i w_{iv}^{l+1} \frac{\partial x_i^l}{\partial z_j^l} = w_{jv}^{l+1} \sigma'(z_j^l)$$

Backpropagation algorithm

$$z_j^L = \sum_v w_{vj}^L x_v^{L-1} + b_j^L \qquad x_j^L = \sigma(z_j^L) \qquad E_0 = \frac{1}{2}\sum_i (x_i^L - y_i)^2$$

$$\frac{\partial E}{\partial z_j^l} = \delta_j^l = \sum_v \frac{\partial E}{\partial z_v^{l+1}} \frac{\partial z_v^{l+1}}{\partial z_j^l}$$

$$z_v^{l+1} = \sum_i w_{iv}^{l+1} x_i^l + b_v^{l+1}$$

$$x_i^l = \sigma(z_i^l)$$

$$= \sum_v \delta_v^{l+1} \frac{\partial z_v^{l+1}}{\partial z_j^l} = \sum_v \delta_v^{l+1} w_{jv}^{l+1} \sigma'(z_j^l)$$

$$\frac{\partial z_v^{l+1}}{\partial z_j^l} = \sum_i w_{iv}^{l+1} \frac{\partial x_i^l}{\partial z_j^l} = w_{jv}^{l+1} \sigma'(z_j^l)$$

Backpropagation algorithm

So, we got

$$\delta_j^l = \sum_v w_{jv}^{l+1} \delta_v^{l+1} \sigma'(z_j^l)$$

Backpropagation algorithm

So, we got

$$\delta_j^l = \sum_v w_{jv}^{l+1} \delta_v^{l+1} \sigma'(z_j^l)$$

In matrix form we get

Backpropagation algorithm

So, we got

$$\delta_j^l = \sum_v w_{jv}^{l+1} \delta_v^{l+1} \sigma'(z_j^l)$$

In matrix form we get $$\boldsymbol{\delta}^l = \mathbf{w}^{l+1} \boldsymbol{\delta}^{l+1} \odot \sigma(\mathbf{z}^{\mathbf{l}})$$

Backpropagation algorithm

So, we got

$$\delta_j^l = \sum_v w_{jv}^{l+1}\delta_v^{l+1}\sigma'(z_j^l)$$

In matrix form we get $\boldsymbol{\delta}^l = \mathbf{w}^{l+1}\boldsymbol{\delta}^{l+1} \odot \sigma(\mathbf{z}^\mathbf{l})$

$\odot =$ Hadamard product

(Element-wise multiplication)

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

$$\frac{\partial E}{\partial w_{ij}^{l}} = \frac{\partial E}{\partial z_{j}^{l}} \frac{\partial z_{j}^{l}}{\partial w_{ij}^{l}}$$

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

$$\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{ij}^l} = \delta_j^l \frac{\partial z_j^l}{\partial w_{ij}^l}$$

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

$$\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{ij}^l} = \delta_j^l \frac{\partial z_j^l}{\partial w_{ij}^l}$$

$$z_j^l = \sum_i w_{ij}^l x_i^{l-1} + b_j^l$$

$$x_j^l = \sigma(z_j^l)$$

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

$$\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{ij}^l} = \delta_j^l \frac{\partial z_j^l}{\partial w_{ij}^l} = \delta_j^l x_i^{l-1}$$

$$z_j^l = \sum_i w_{ij}^l x_i^{l-1} + b_j^l$$

$$x_j^l = \sigma(z_j^l)$$

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

$$\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{ij}^l} = \delta_j^l \frac{\partial z_j^l}{\partial w_{ij}^l} = \delta_j^l x_i^{l-1}$$

$$z_j^l = \sum_i w_{ij}^l x_i^{l-1} + b_j^l$$

$$x_j^l = \sigma(z_j^l)$$

In matrix form

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

$$\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{ij}^l} = \delta_j^l \frac{\partial z_j^l}{\partial w_{ij}^l} = \delta_j^l x_i^{l-1}$$

$$z_j^l = \sum_i w_{ij}^l x_i^{l-1} + b_j^l$$

$$x_j^l = \sigma(z_j^l)$$

In matrix form

$$\frac{\partial E}{\partial \mathbf{w}^l} = \mathbf{x}^{l-1}(\boldsymbol{\delta}^l)^\top$$

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

$$\frac{\partial E}{\partial b_j^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l}$$

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

$$\frac{\partial E}{\partial b_j^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l \frac{\partial z_j^l}{\partial b_j^l}$$

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

$$\frac{\partial E}{\partial b_j^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l \frac{\partial z_j^l}{\partial b_j^l}$$

$$z_j^l = \sum_i w_{ij}^l x_i^{l-1} + b_j^l$$

$$x_j^l = \sigma(z_j^l)$$

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

$$\frac{\partial E}{\partial b_j^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l \frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l$$

$$z_j^l = \sum_i w_{ij}^l x_i^{l-1} + b_j^l$$

$$x_j^l = \sigma(z_j^l)$$

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

$$\frac{\partial E}{\partial b_j^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l \frac{\partial z_j^l}{\partial b_j^l} \quad = \delta_j^l$$

$$z_j^l = \sum_i w_{ij}^l x_i^{l-1} + b_j^l$$

$$x_j^l = \sigma(z_j^l)$$

In matrix form

Backpropagation algorithm

This is ok, but to perform gradient descent we need the variation of the cost function for w ans b!

$$\frac{\partial E}{\partial b_j^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l \frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l$$

$$z_j^l = \sum_i w_{ij}^l x_i^{l-1} + b_j^l$$

$$x_j^l = \sigma(z_j^l)$$

In matrix form

$$\frac{\partial E}{\partial \mathbf{b}^l} = \boldsymbol{\delta}^l$$

Backpropagation algorithm

If you have followed, congratulations, you have looked at of one of the most historic algorithm in machine learning!

Backpropagation algorithm

If you have followed, congratulations, you have looked at of one of the most historic algorithm in machine learning!

$$\boldsymbol{\delta}^L = (\mathbf{x}^L - \mathbf{y}) \odot \sigma'(\mathbf{z}^L)$$

For more details please check "Deep learning: an introduction for applied mathematicians" by Catherine F. Higham and Desmond J. Higham, Section 5

Backpropagation algorithm

If you have followed, congratulations, you have looked at of one of the most historic algorithm in machine learning!

$$\boldsymbol{\delta}^L = (\mathbf{x}^L - \mathbf{y}) \odot \sigma'(\mathbf{z}^L)$$

$$\boldsymbol{\delta}^l = \mathbf{w}^{l+1}\boldsymbol{\delta}^{l+1} \odot \sigma(\mathbf{z}^{\mathbf{l}}) \qquad \text{for all } l = L-1, \ldots, 1 .$$

For more details please check "Deep learning: an introduction for applied mathematicians" by Catherine F. Higham and Desmond J. Higham, Section 5

Backpropagation algorithm

If you have followed, congratulations, you have looked at of one of the most historic algorithm in machine learning!

$$\boldsymbol{\delta}^L = (\mathbf{x}^L - \mathbf{y}) \odot \sigma'(\mathbf{z}^L)$$

$$\boldsymbol{\delta}^l = \mathbf{w}^{l+1}\boldsymbol{\delta}^{l+1} \odot \sigma(\mathbf{z}^{\mathbf{l}})$$

for all $l = L - 1, \dots, 1$.

$$\frac{\partial E}{\partial \mathbf{w}^l} = \mathbf{x}^{l-1}(\boldsymbol{\delta}^l)^\top$$

For more details please check "Deep learning: an introduction for applied mathematicians" by Catherine F. Higham and Desmond J. Higham, Section 5

Backpropagation algorithm

If you have followed, congratulations, you have looked at of one of the most historic algorithm in machine learning!

$$\boldsymbol{\delta}^L = (\mathbf{x}^L - \mathbf{y}) \odot \sigma'(\mathbf{z}^L)$$

$$\boldsymbol{\delta}^l = \mathbf{w}^{l+1}\boldsymbol{\delta}^{l+1} \odot \sigma(\mathbf{z}^{\mathbf{l}})$$

for all $l = L-1, \ldots, 1$.

$$\frac{\partial E}{\partial \mathbf{w}^l} = \mathbf{x}^{l-1}(\boldsymbol{\delta}^l)^\top$$

$$\frac{\partial E}{\partial \mathbf{b}^l} = \boldsymbol{\delta}^l$$

For more details please check "Deep learning: an introduction for applied mathematicians" by Catherine F. Higham and Desmond J. Higham, Section 5

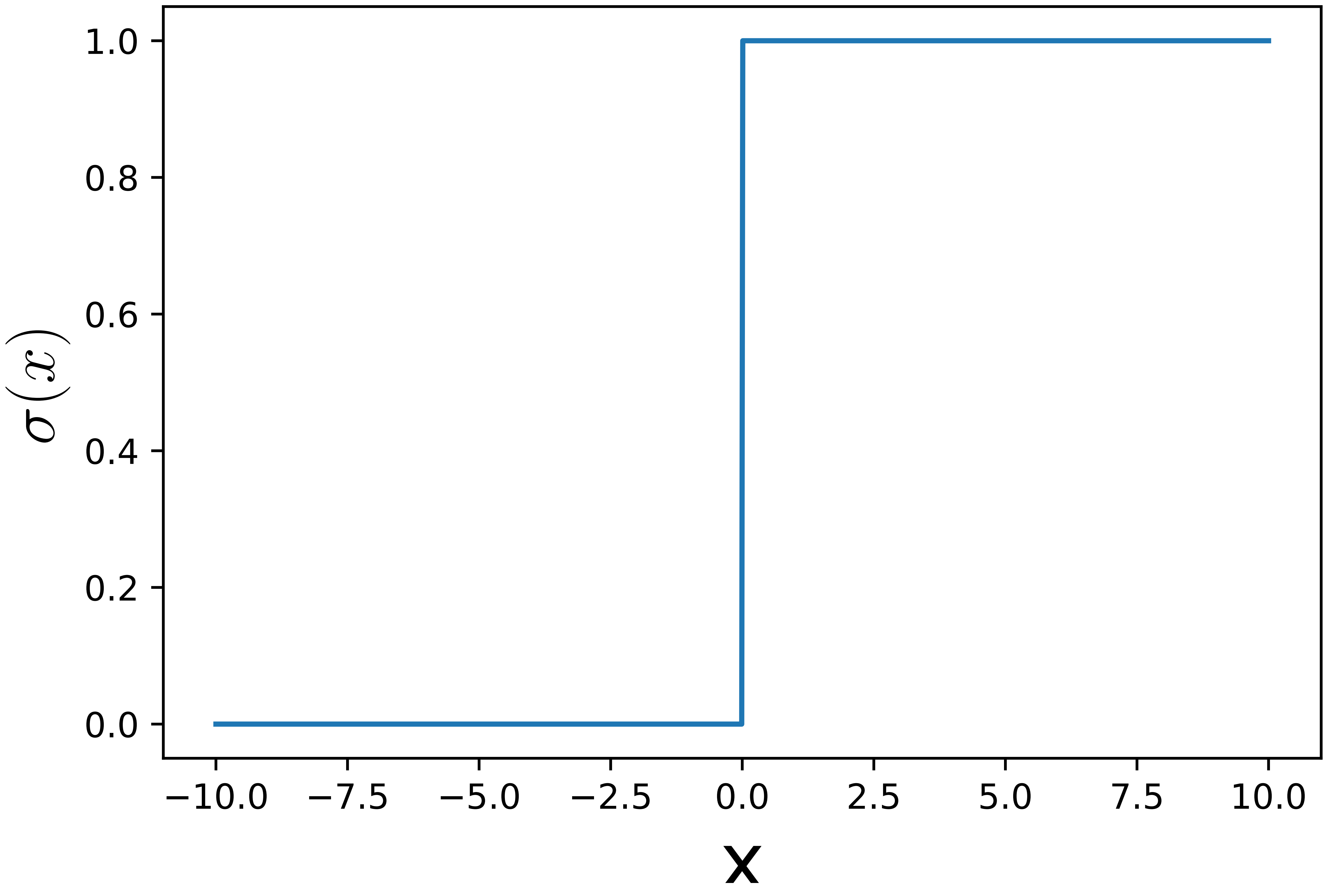
ACTIVATION FUNCTIONS

Popular activation functions

Activation function examples:

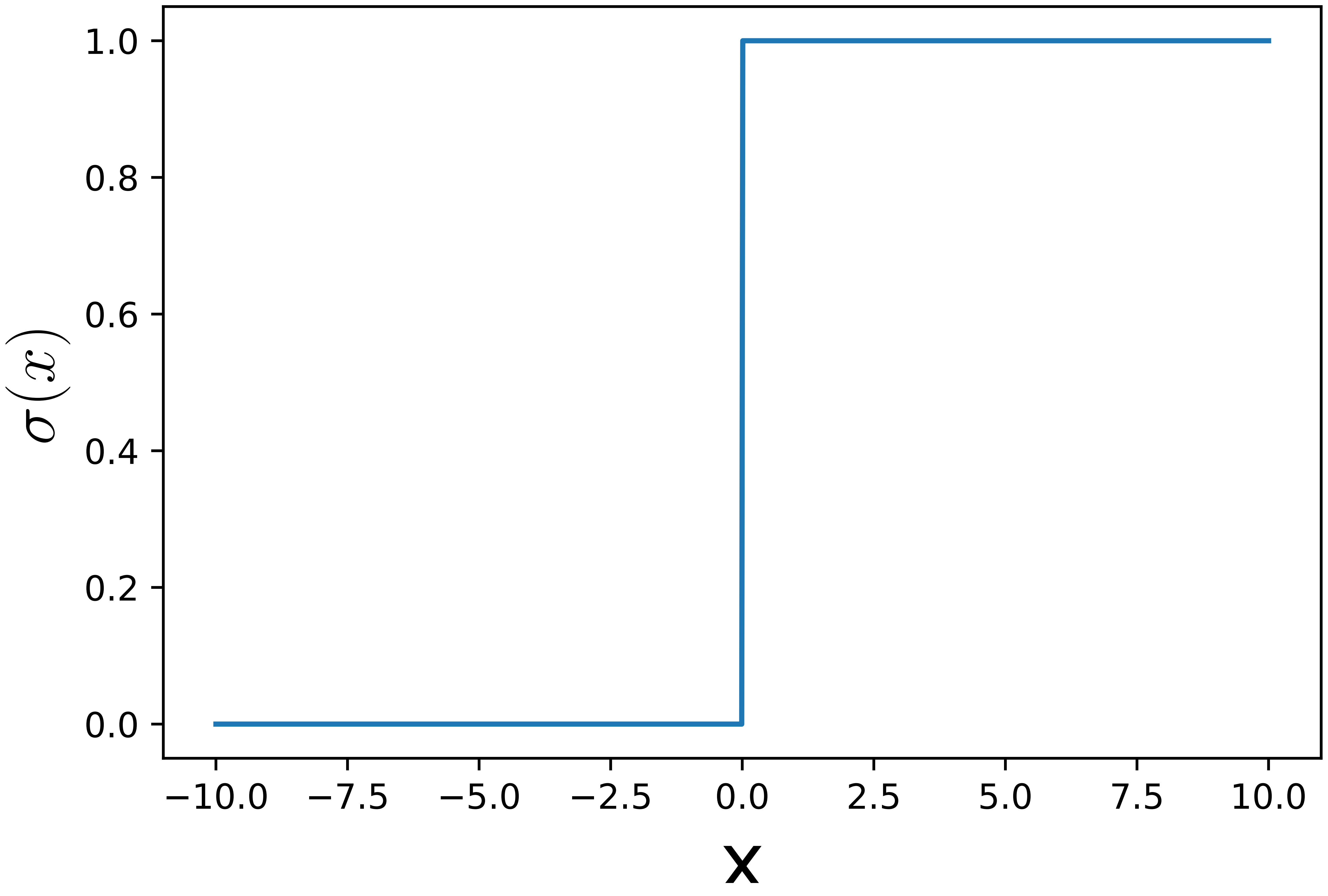
Popular activation functions

Activation function examples: $\sigma(x) = \text{Heaviside}(x)$



Popular activation functions

Activation function examples: $\sigma(x) = \text{Heaviside}(x)$

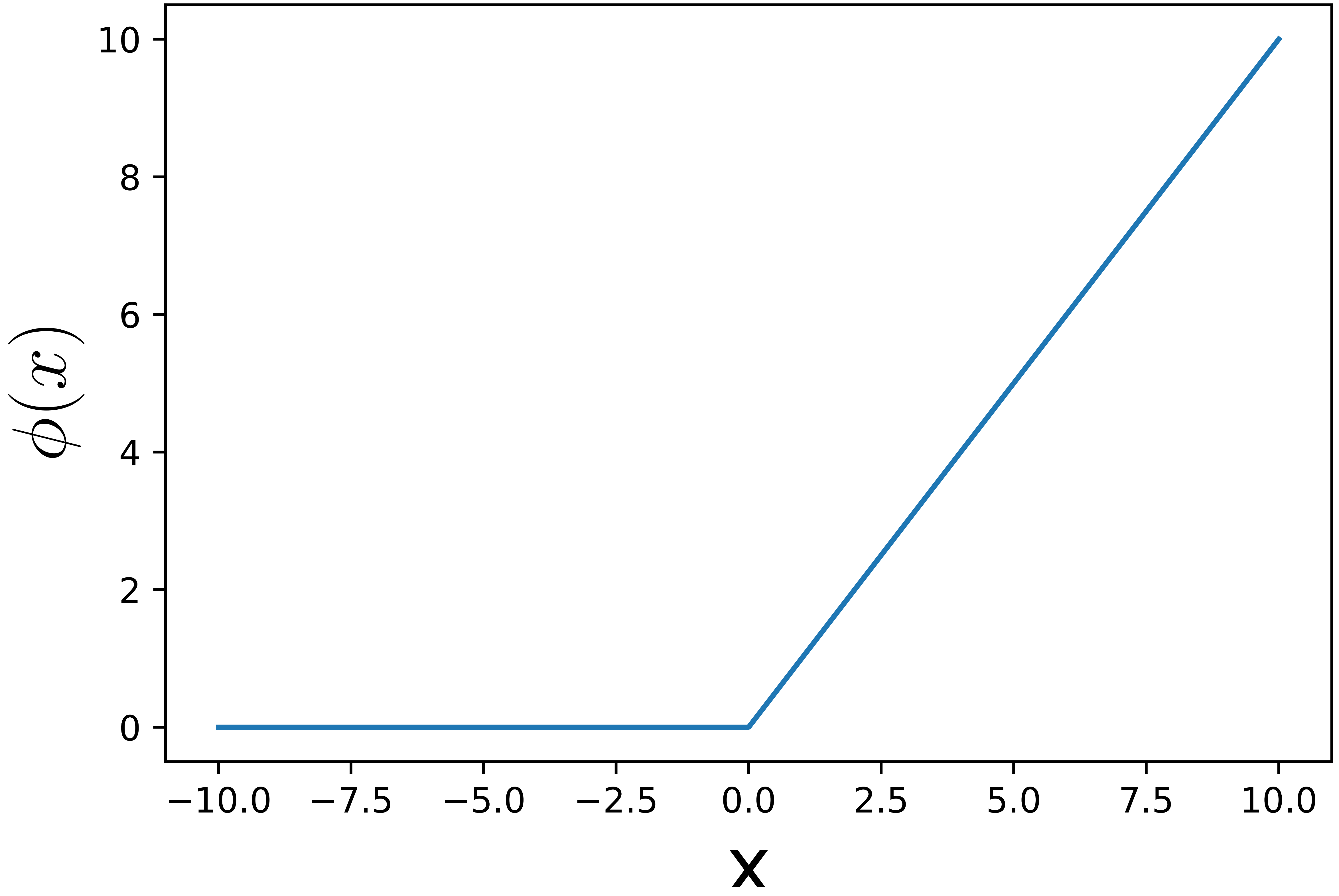


Output, binary

Popular activation functions

Activation function examples:

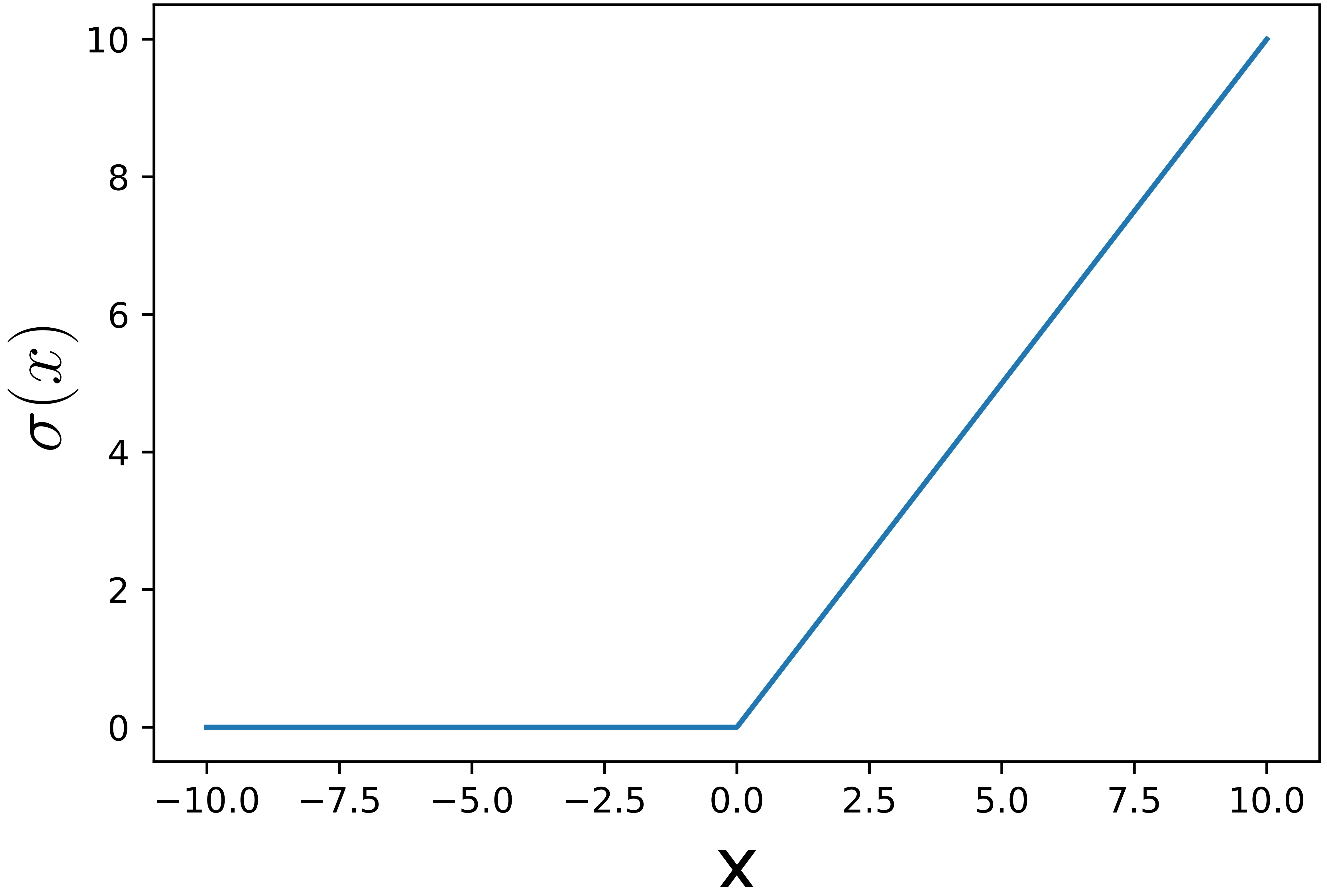
Rectifier: $\sigma(x) = \max(0, x)$



Popular activation functions

Activation function examples:

Rectifier: $\sigma(x) = \max(0, x)$



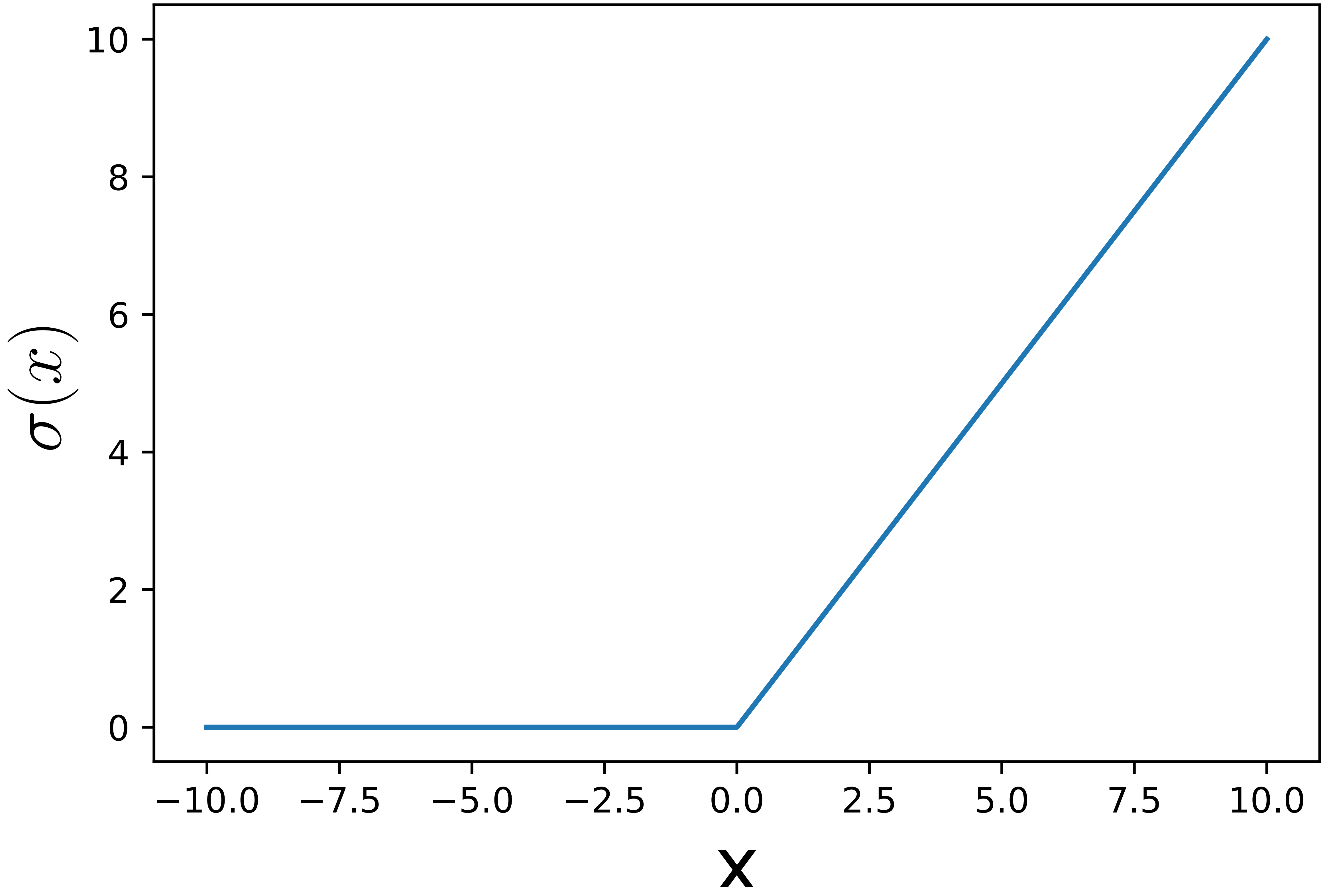
Rectified linear unit (ReLU):

$$\Rightarrow \; f^l(x) = \max\left(0, (W^l)^\top x^{l-1} + b^l\right)$$

Popular activation functions

Activation function examples:

Rectifier: $\sigma(x) = \max(0,x)$



Rectified linear unit (ReLU):

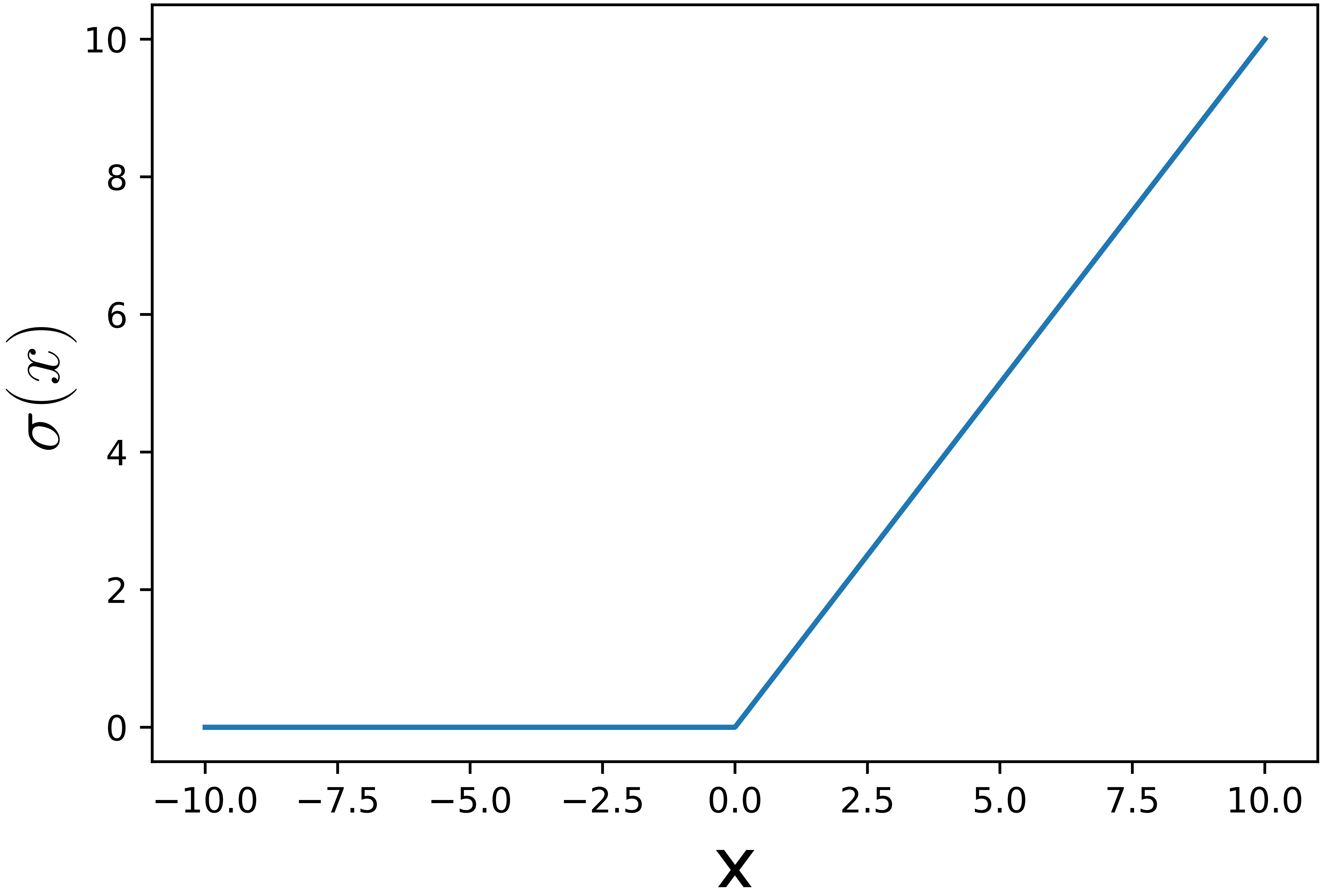
$$\Rightarrow \quad f^l(x) = \max\left(0, (W^l)^\top x^{l-1} + b^l\right)$$

(component-wise application)

Popular activation functions

Activation function examples:

Rectifier: $\sigma(x) = \max(0, x)$ Eliminates negative values



Rectified linear unit (ReLU):

$$\Rightarrow \quad f^l(x) = \max\left(0, (W^l)^\top x^{l-1} + b^l\right)$$

(component-wise application)

Popular activation functions

Many other activation functions possible:

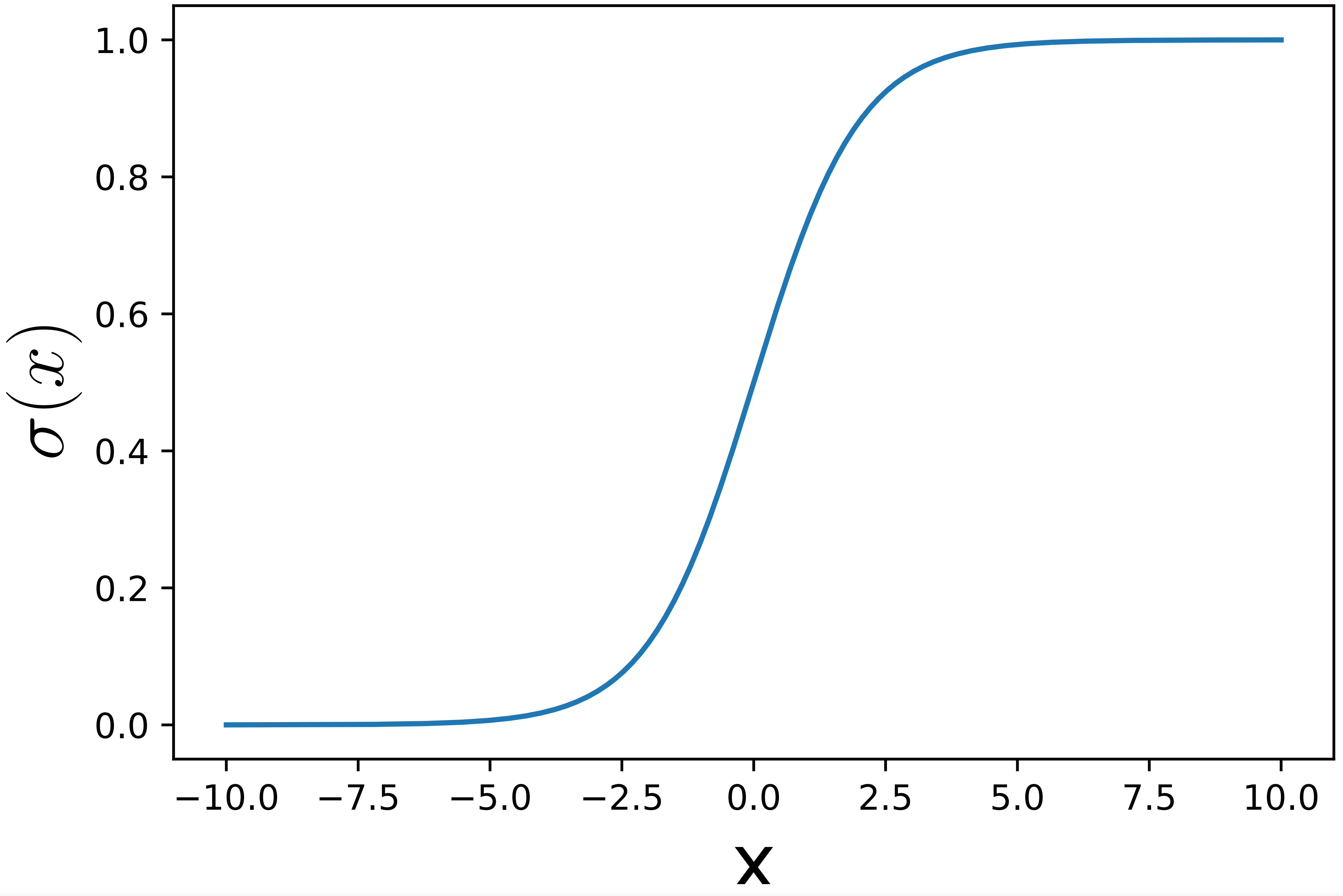
Smooth heavyside

Popular activation functions

Many other activation functions possible:

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

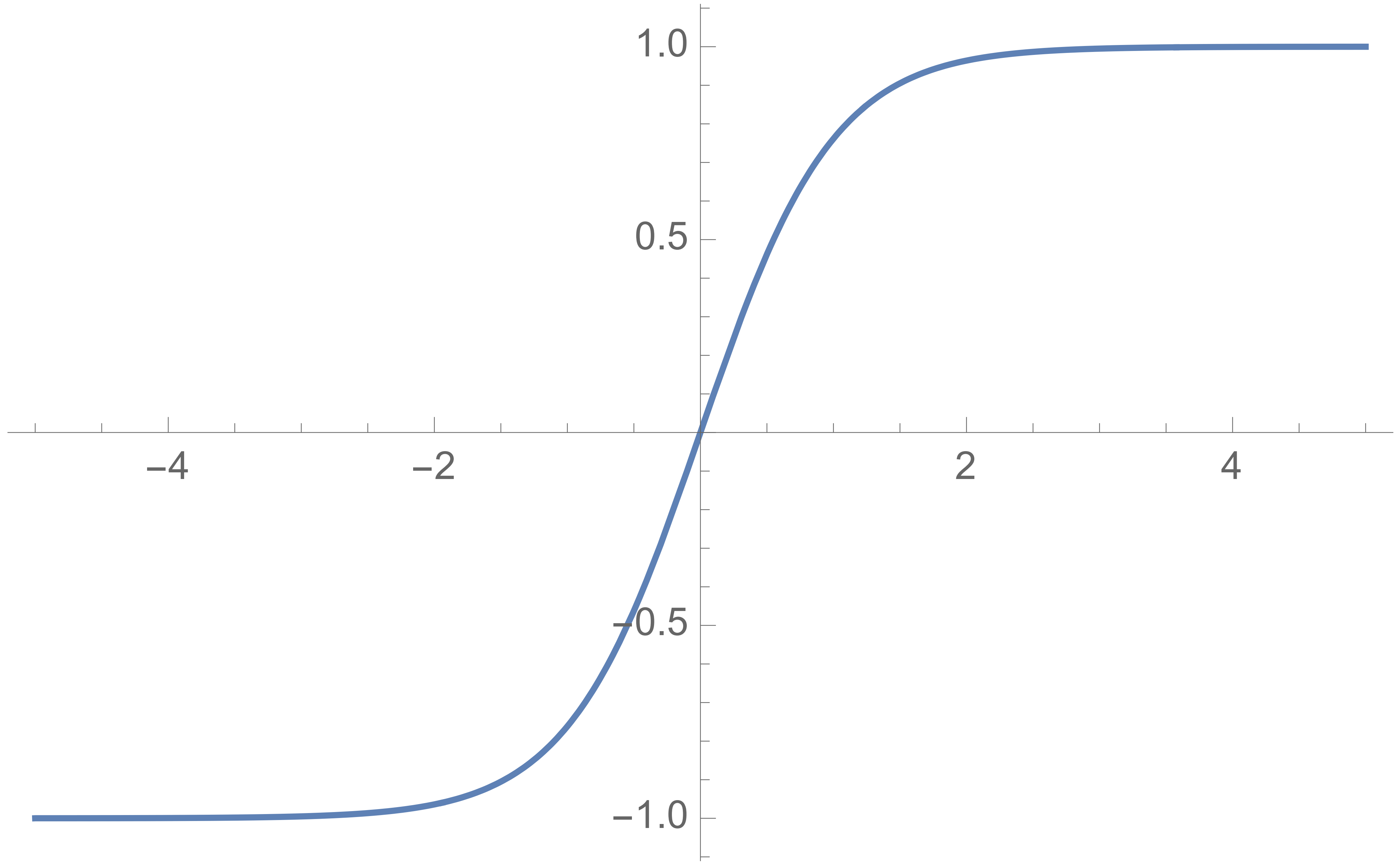
(Sigmoid/logistic function)



Smooth heavyside

Popular activation functions

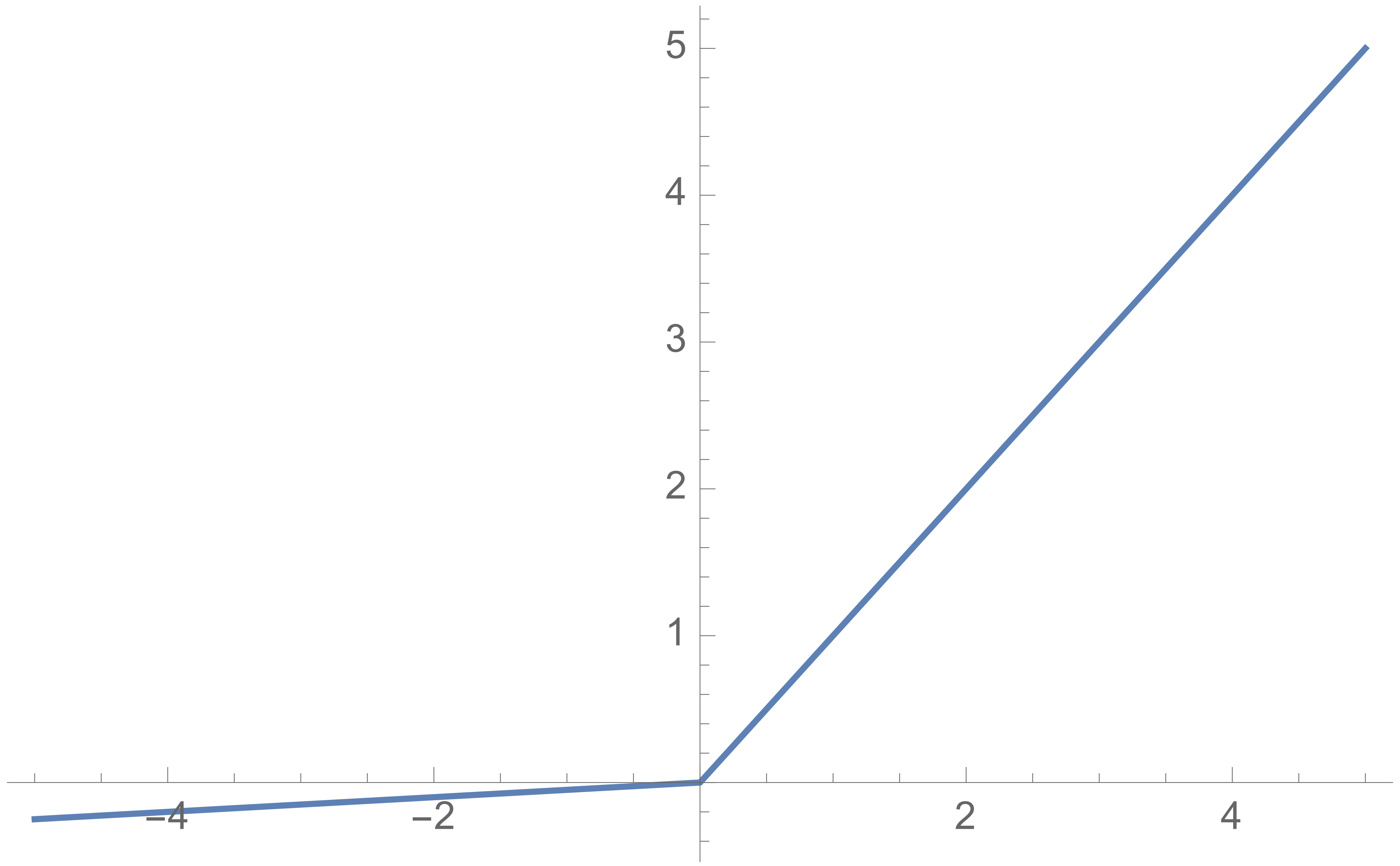
Hyperbolic tangent if you want to allow negative values



$$\sigma(x) = \tanh(x) := \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\phi(2x) - 1 \quad \text{for} \quad \phi(x) = \frac{1}{1 + e^{-x}}$$

Popular activation functions

Leaky rectifier (when you want to allow some small negative values on the x)



$\alpha = 0.05$

$$\sigma_\alpha(x) = \max(\alpha x, x)$$

Popular activation functions

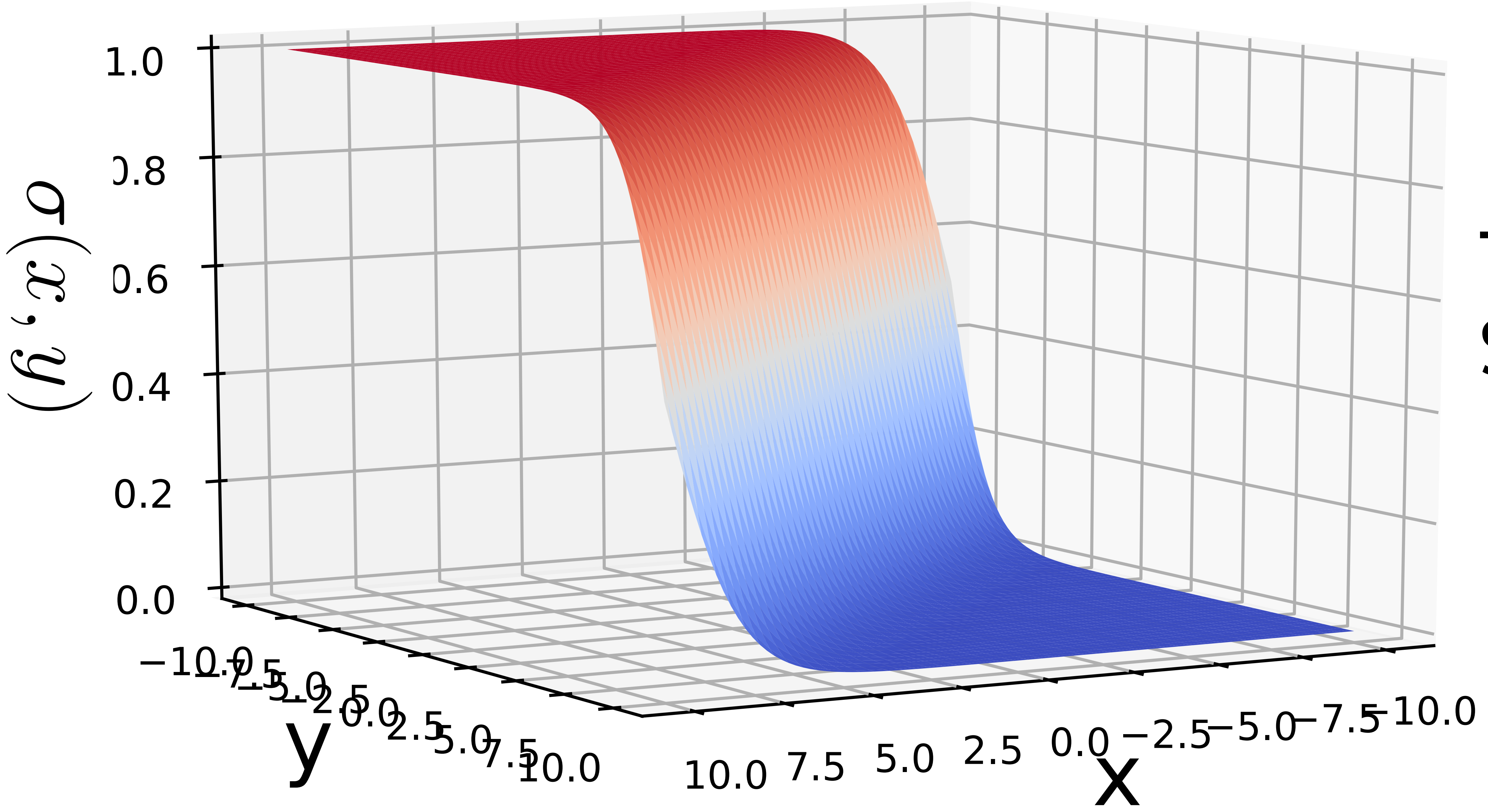
Many other activation functions possible:

Takes a vector, gives you a vector
Sums overall al j is one

Popular activation functions

Many other activation functions possible:

$$\sigma(x)_j = \frac{\exp(x_j)}{\sum_{i=1}^{K} \exp(x_i)}$$ (Softmax function)



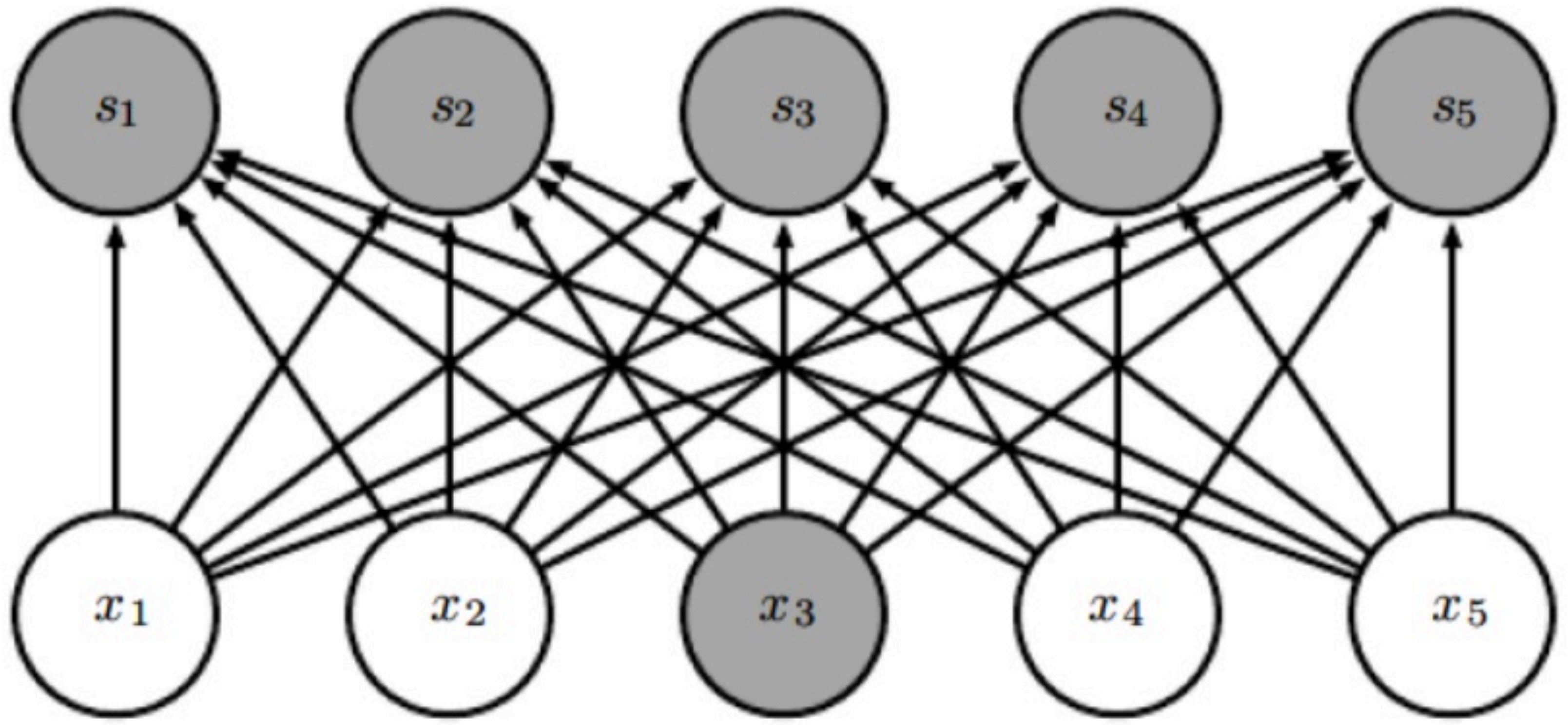
Takes a vector, gives you a vector
Sums overall al j is one

CONVOLUTIONAL NEURAL NETWORKS

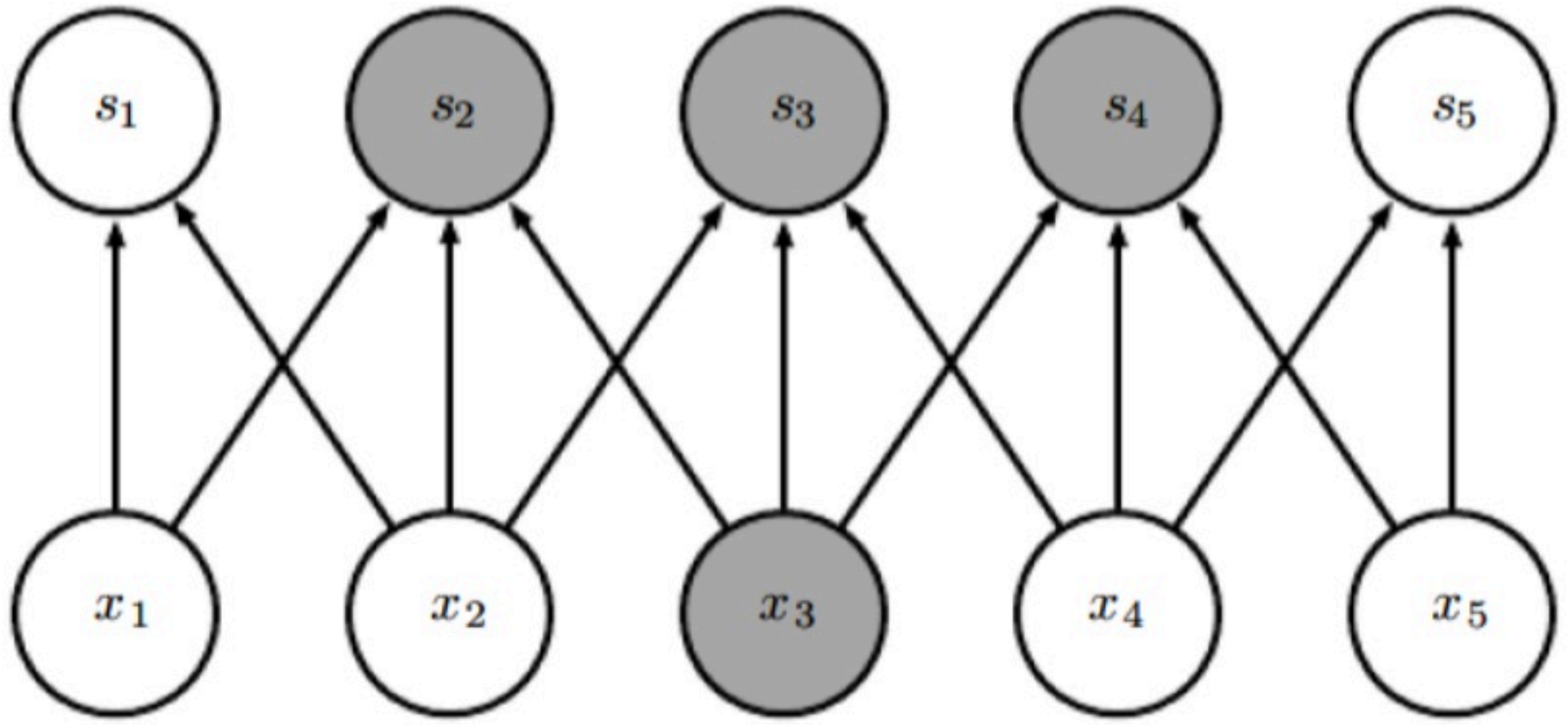
Convolutional neural network

Basic idea: reduce no. of network connections

From



To



Convolutional neural network

We reduce the no. of network connections by restricting our matrices W to a special class of linear operators: convolutions!

Convolutional neural network

We reduce the no. of network connections by restricting our matrices W to a special class of linear operators: convolutions!

Continuous convolution operator: $$(x * y)(\tau) := \int_{\mathbb{R}^n} x(t)\, y(t-\tau)\, dt$$

Convolutional neural network

We reduce the no. of network connections by restricting our matrices W to a special class of linear operators: convolutions!

Continuous convolution operator: $$(x * y)(\tau) := \int_{\mathbb{R}^n} x(t)\, y(t-\tau)\, dt$$

Discrete version: $$(x * y)[\,j\,] := \sum_{i=1}^{m} x[i]\, y[i-j]$$ one-dimensional

Convolutional neural network

We reduce the no. of network connections by restricting our matrices W to a special class of linear operators: convolutions!

Continuous convolution operator: $$(x * y)(\tau) := \int_{\mathbb{R}^n} x(t)\, y(t-\tau)\, dt$$

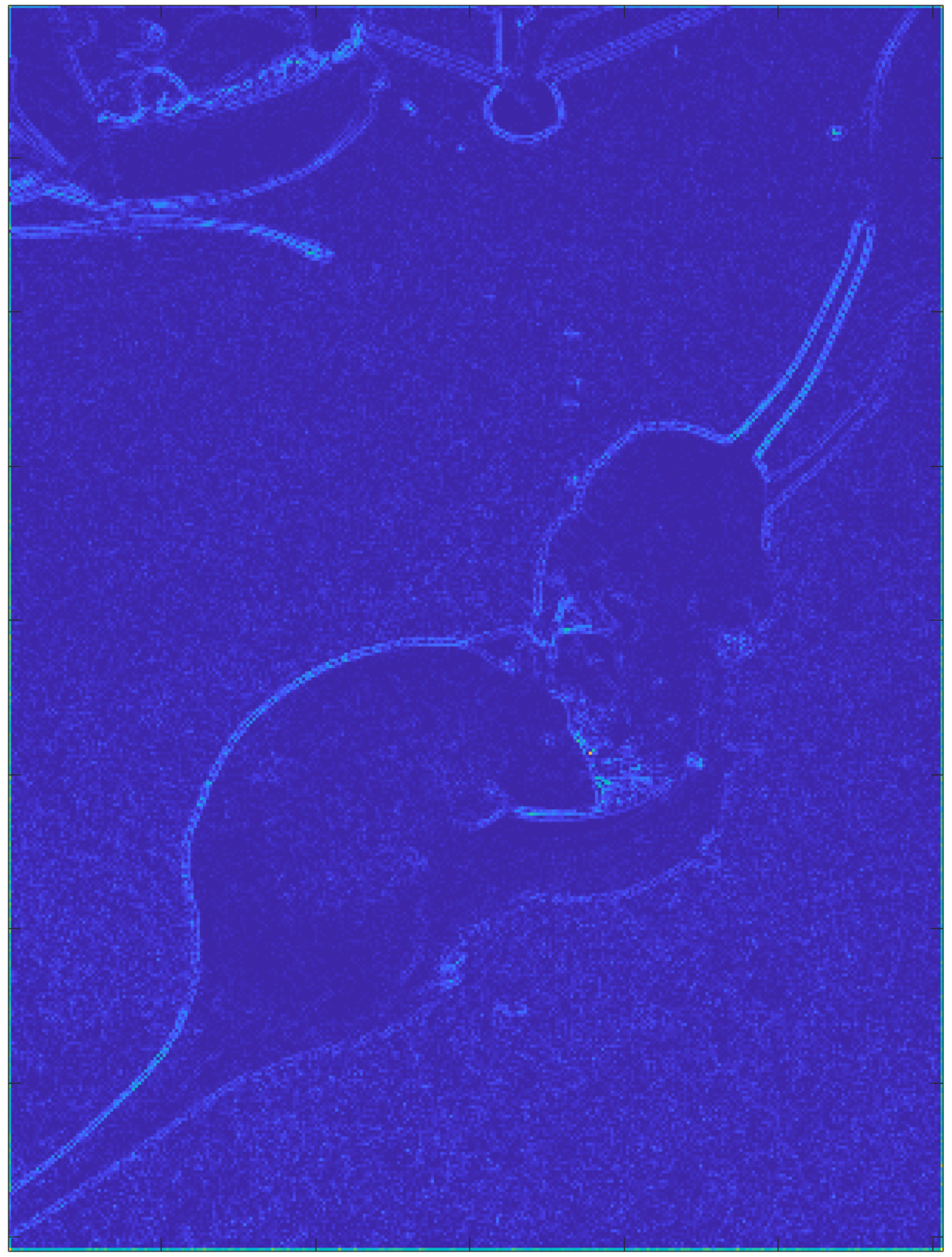
Discrete version: $$(x * y)[j] := \sum_{i=1}^{m} x[i]\, y[i-j]$$ one-dimensional

$$(x * y)[p,q] := \sum_{i=1}^{m} \sum_{j=1}^{n} x[i,j]\, y[i-p, j-q]$$ two-dimensional

Convolutional neural network

Example applied to a RGB coordinate system (a picture!):

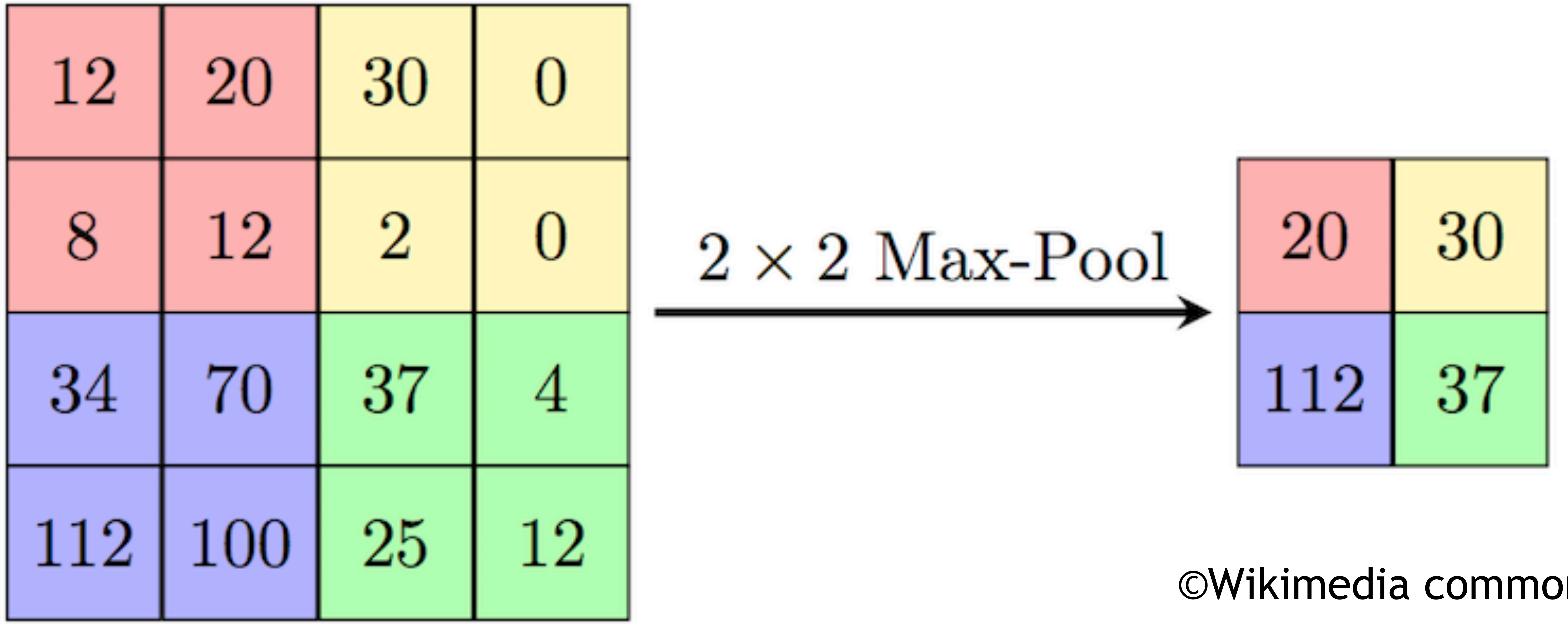
$$* \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} =$$



The convolution “kills” a lot of data, but keeps some key components

Convolutional neural network

Another example: max-pooling



©Wikimedia commons