Artificial neural networks

Agenda

- Computational graph
  - Automatic differentiation
- Neurons
- Representing functions by artificial neural networks
  - The back propagation algorithm

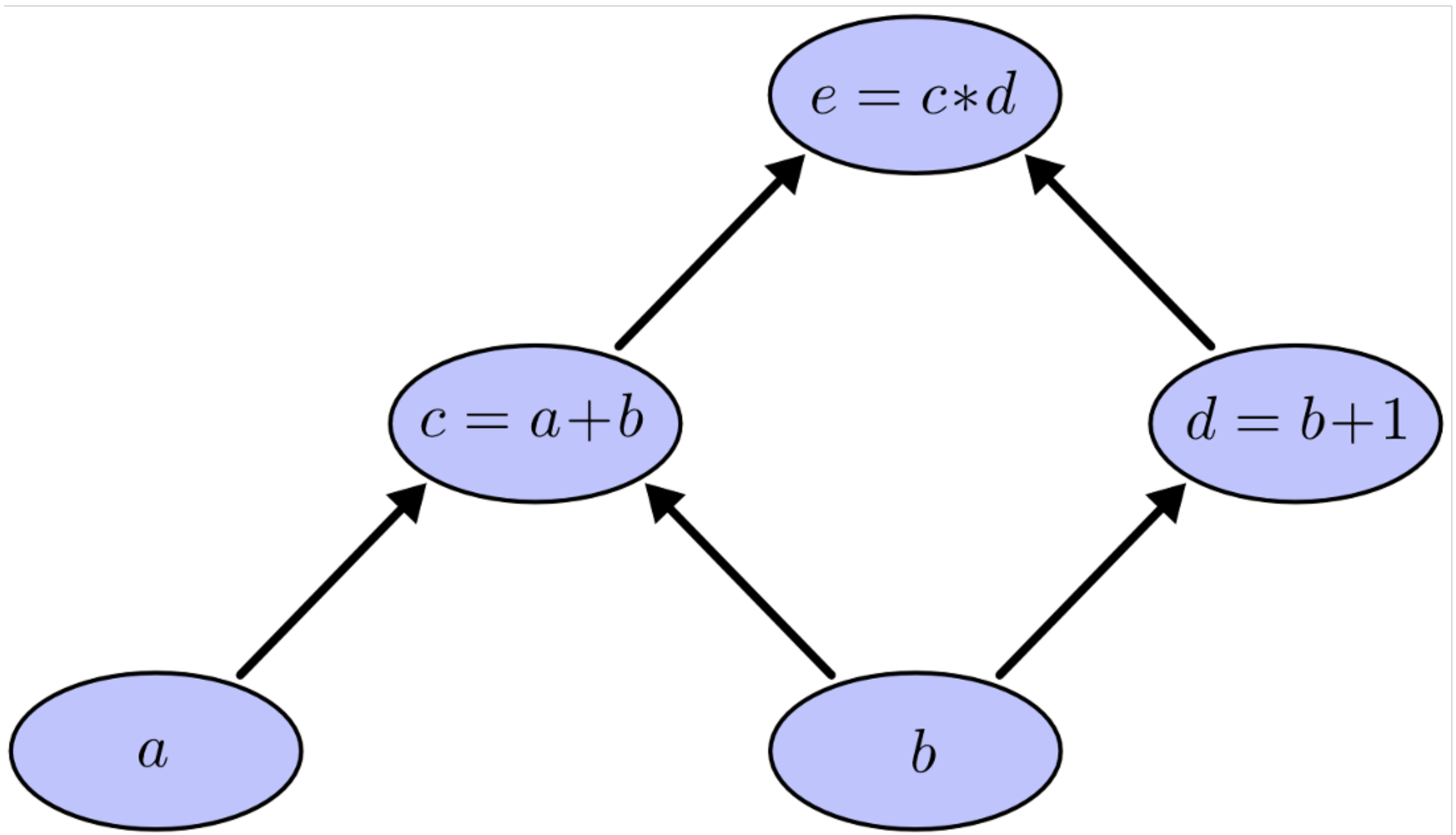
Computational graphs

Computational graphs

A "language" (in terms of graph) to describe a function; e.g., the expression $e = (a + b) \times (b + 1)$ can be described as

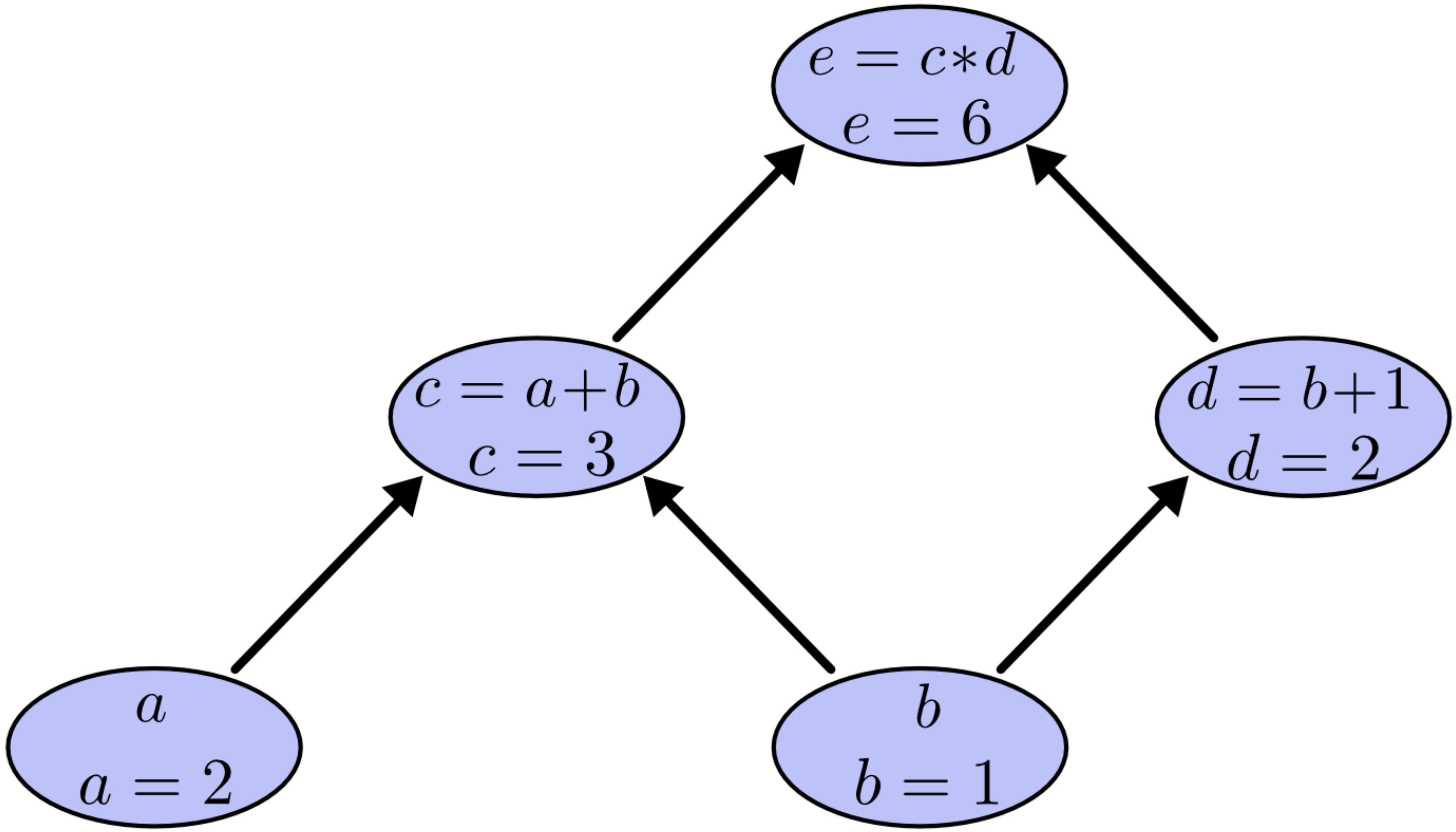
$$c = a + b$$
$$d = b + 1$$
$$e = c \times d$$

Computational graphs



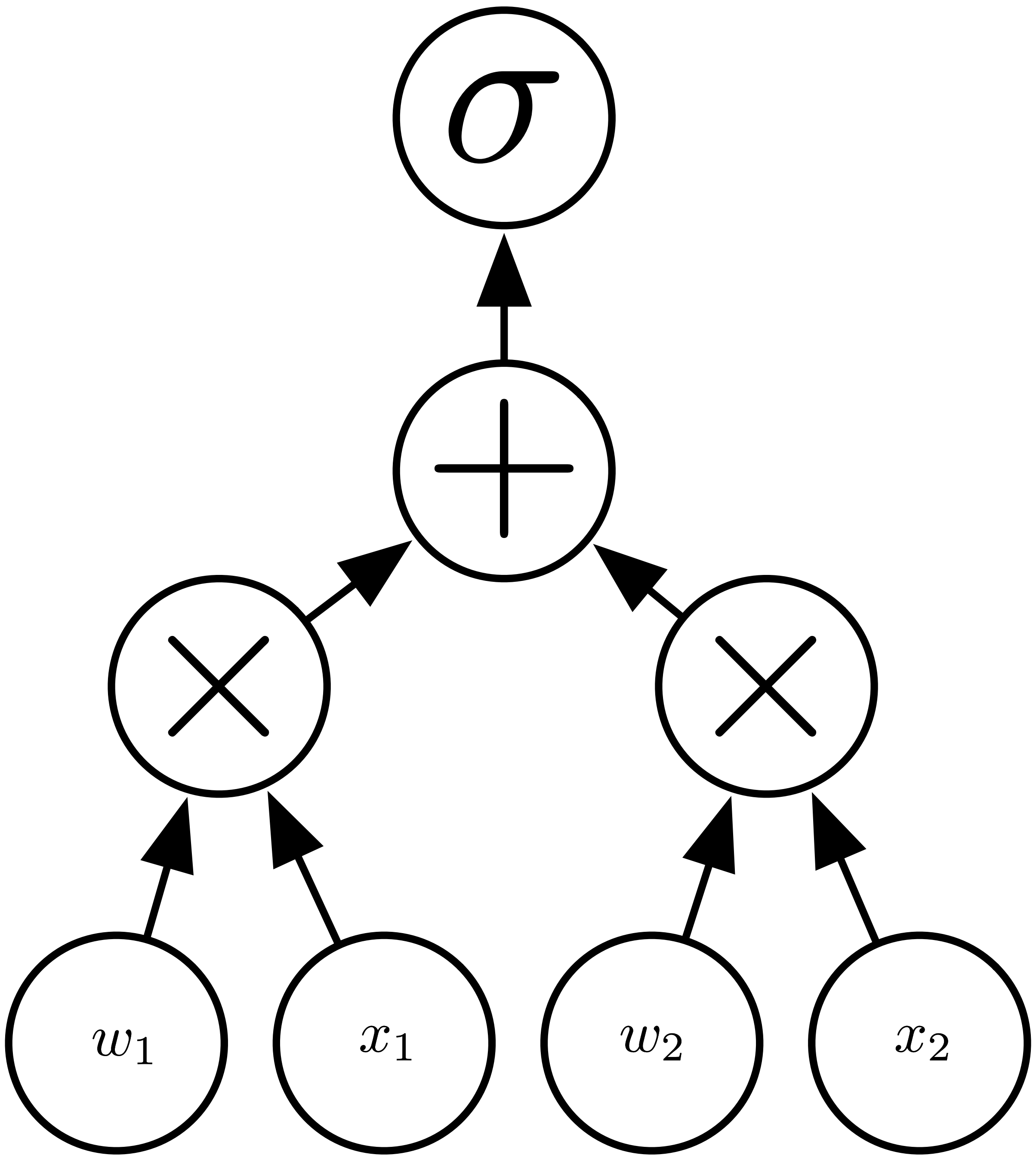
http://colah.github.io/posts/2015-08-Backprop/

Evaluate a computational graph



http://colah.github.io/posts/2015-08-Backprop/

Computational graph for logistic regression



Derivatives on Computational Graphs

Idea: understand derivatives on the edges together with chain rule. For example,

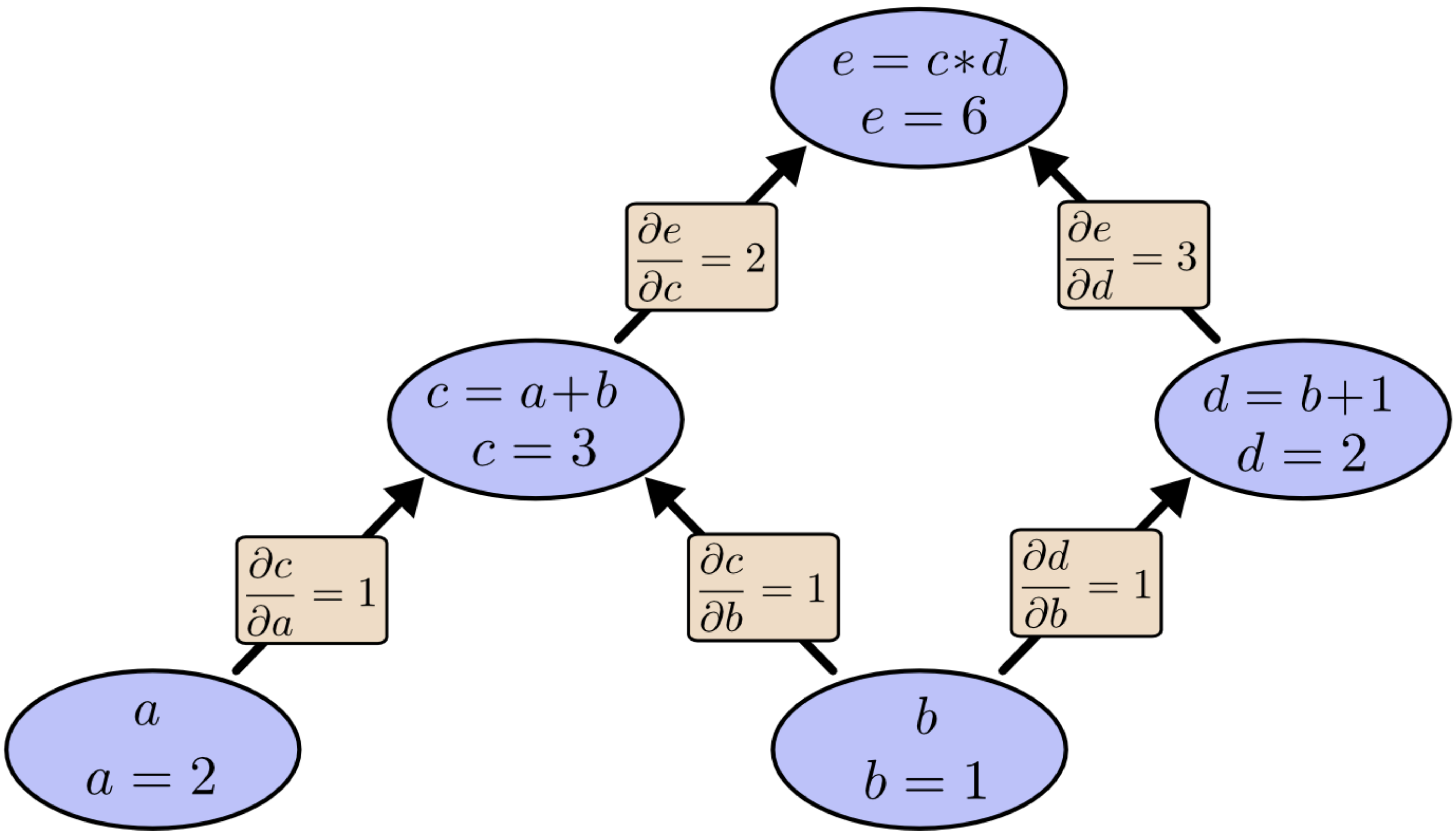
$$\frac{\partial c}{\partial a} = \frac{\partial(a+b)}{\partial a} = 1$$

$$\frac{\partial e}{\partial c} = \frac{\partial(c \times d)}{\partial c} = d = 2$$

$$\frac{\partial e}{\partial a} = \frac{\partial(c \times d)}{\partial a} = d \times \frac{\partial c}{\partial a} = d = 2$$

http://colah.github.io/posts/2015-08-Backprop/

Derivatives on Computational Graphs



$$\frac{\partial e}{\partial b} = 2 \times 1 + 3 \times 1 = 5$$

http://colah.github.io/posts/2015-08-Backprop/

Artificial Neurons

Artificial Neurons

An artificial neuron is a special case of computation graph that represents

$$y = f\left(b + \sum_{j=1}^{p} w_j x_j\right)$$

- b: bias
- w_j: weight
- f: activation function

unknown parameters to be learned

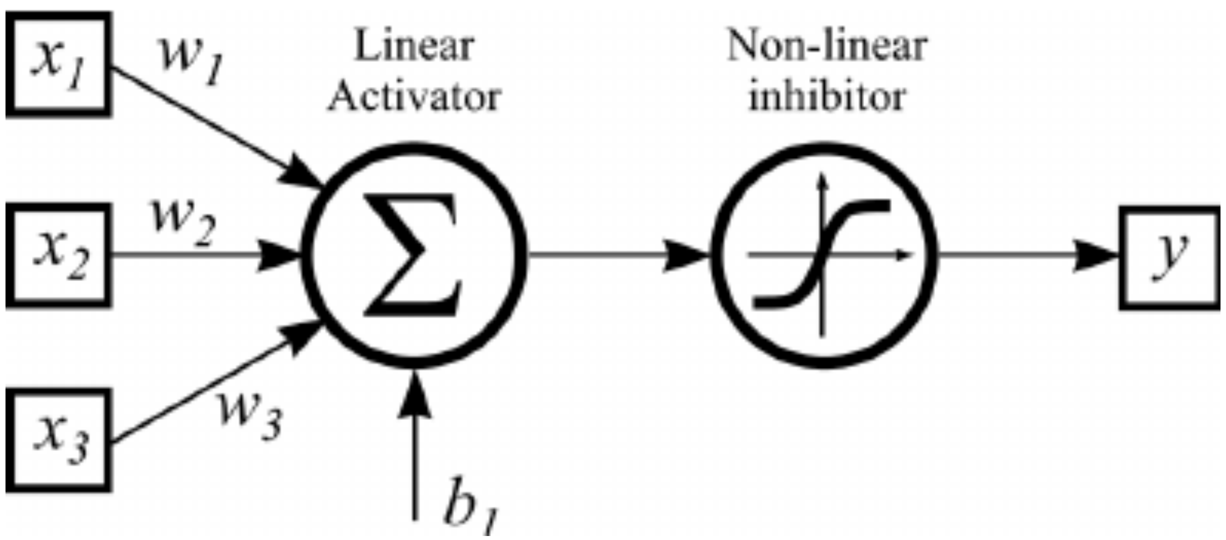
Derivatives of artificial neurons

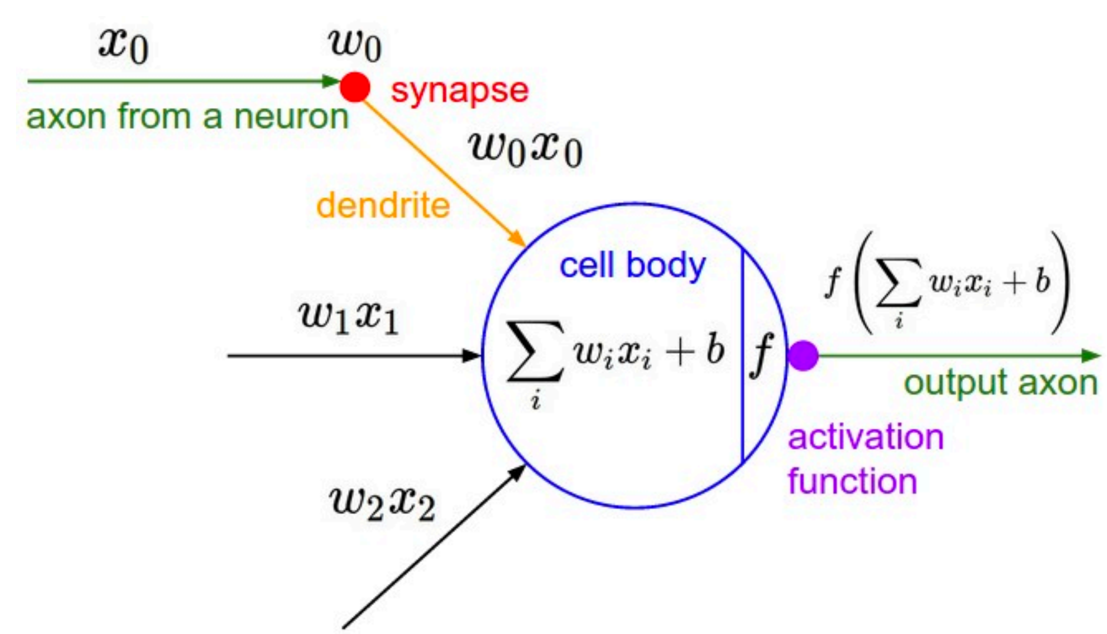
$$\frac{\partial y}{\partial b} = f'\left(b + \sum_{j=1}^{p} w_j x_j\right)$$

$$\frac{\partial y}{\partial w_j} = f'\left(b + \sum_{j=1}^{p} w_j x_j\right) \cdot x_j$$

$$\frac{\partial y}{\partial x_j} = f'\left(b + \sum_{j=1}^{p} w_j x_j\right) \cdot w_j$$

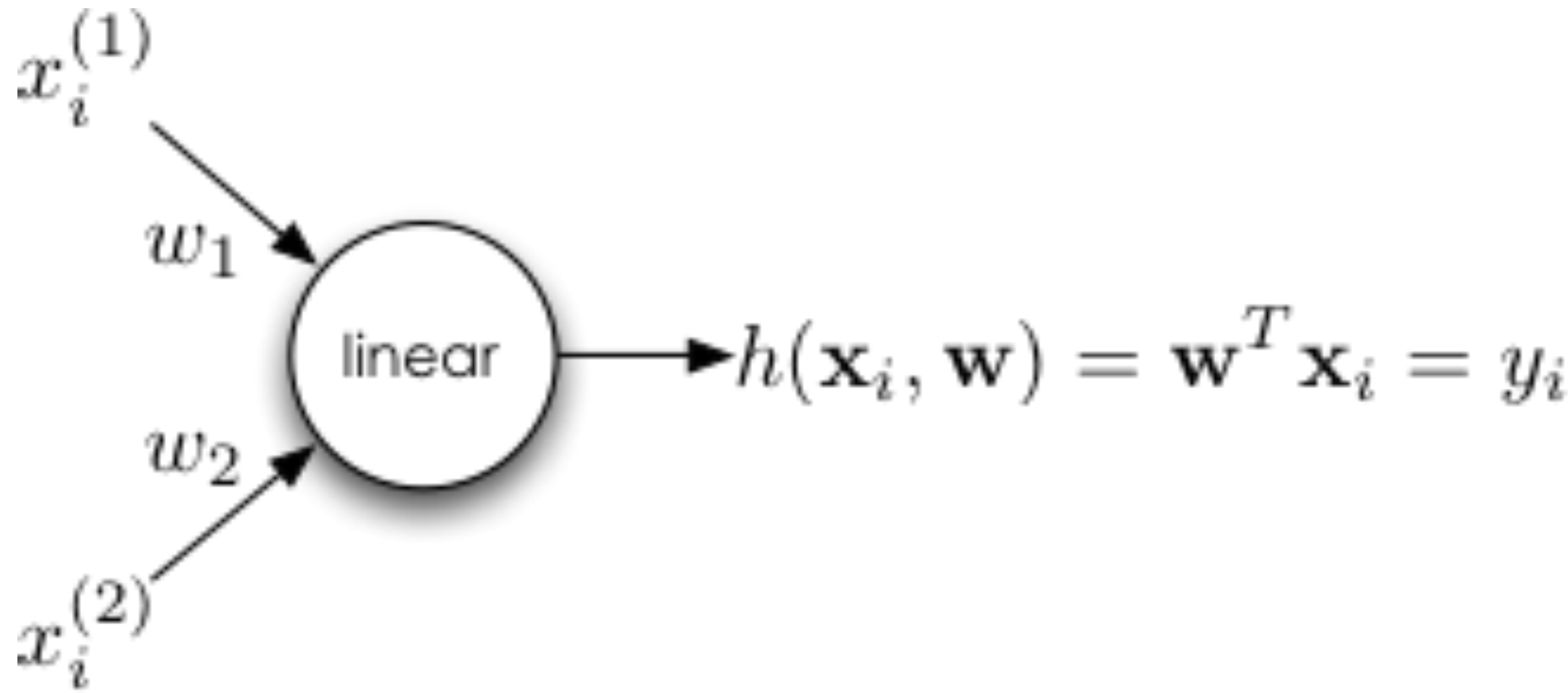
Artificial Neurons





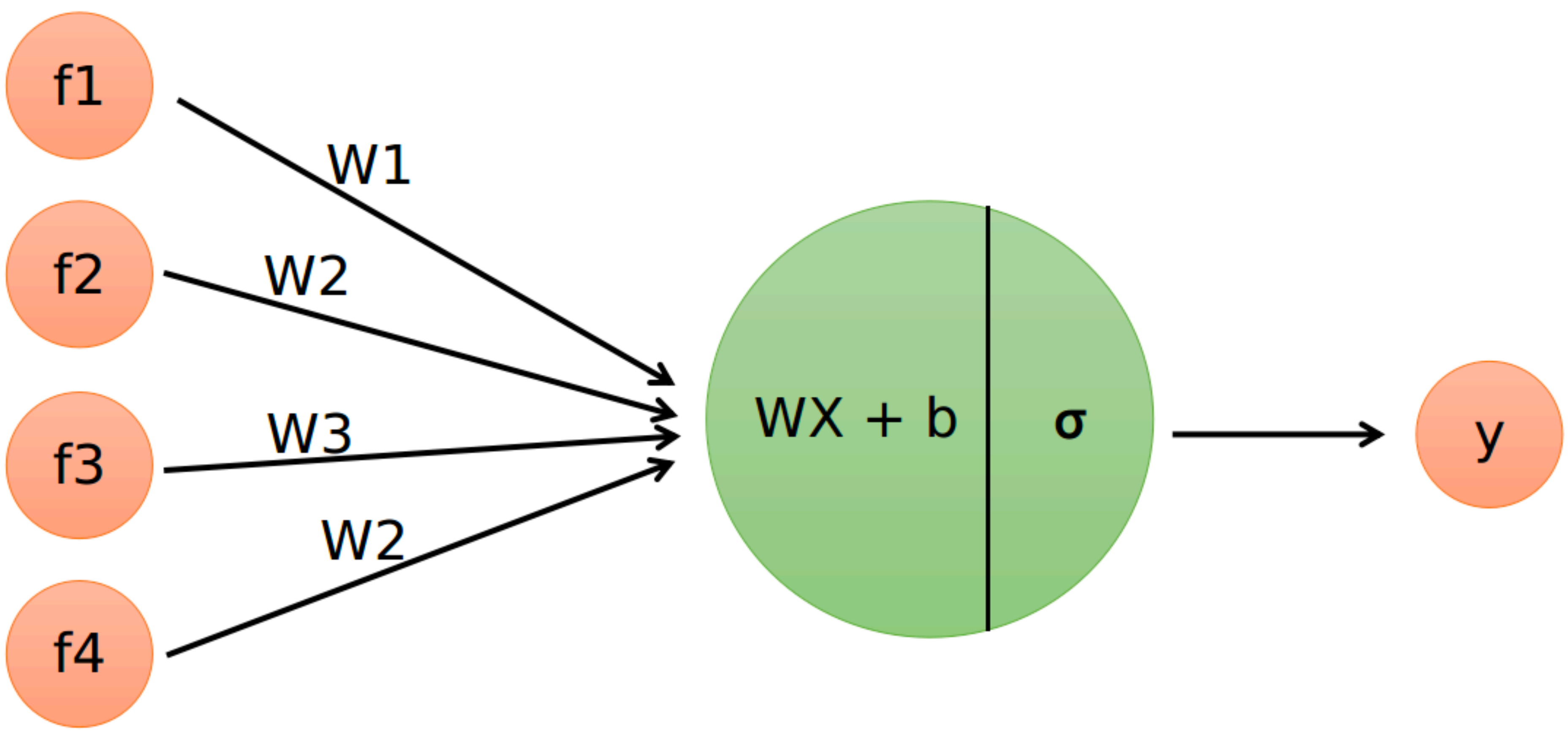
- Computational Methods and Optimization
- https://cs231n.github.io/convolutional-networks/

Example: linear regression



http://www.briandolhansky.com/blog/artificial-neural-networks-linear-regression-part-1

Example: logistic regression



https://towardsdatascience.com/a-logistic-regression-from-scratch-3824468b1f88

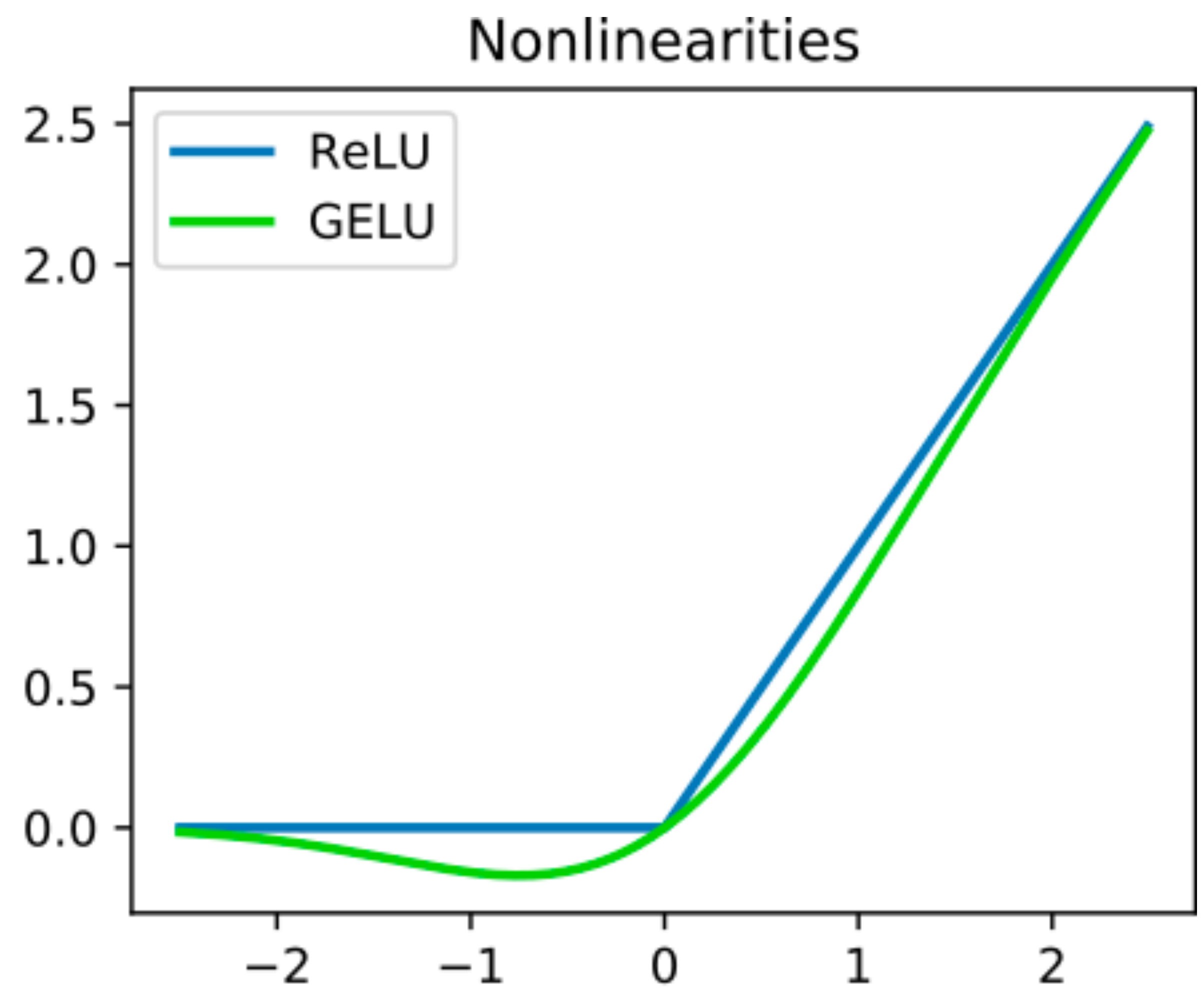
Rectifier activation function

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$= \max(0, x)$$

$$\triangleq x^{+}$$

$$ReLU'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

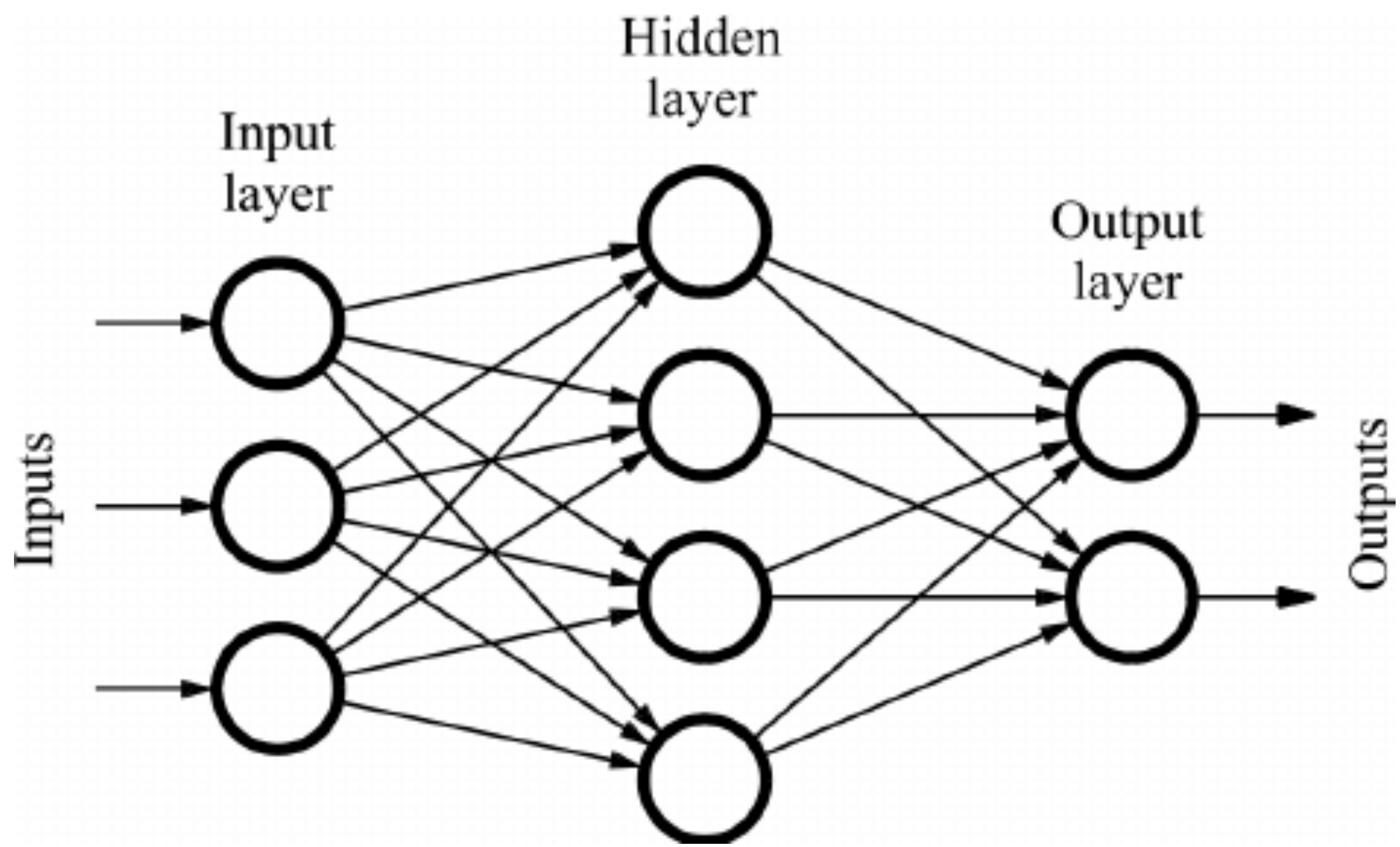


https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

Artificial neural networks

Directed graph of neurons

Example: feed-forward networks

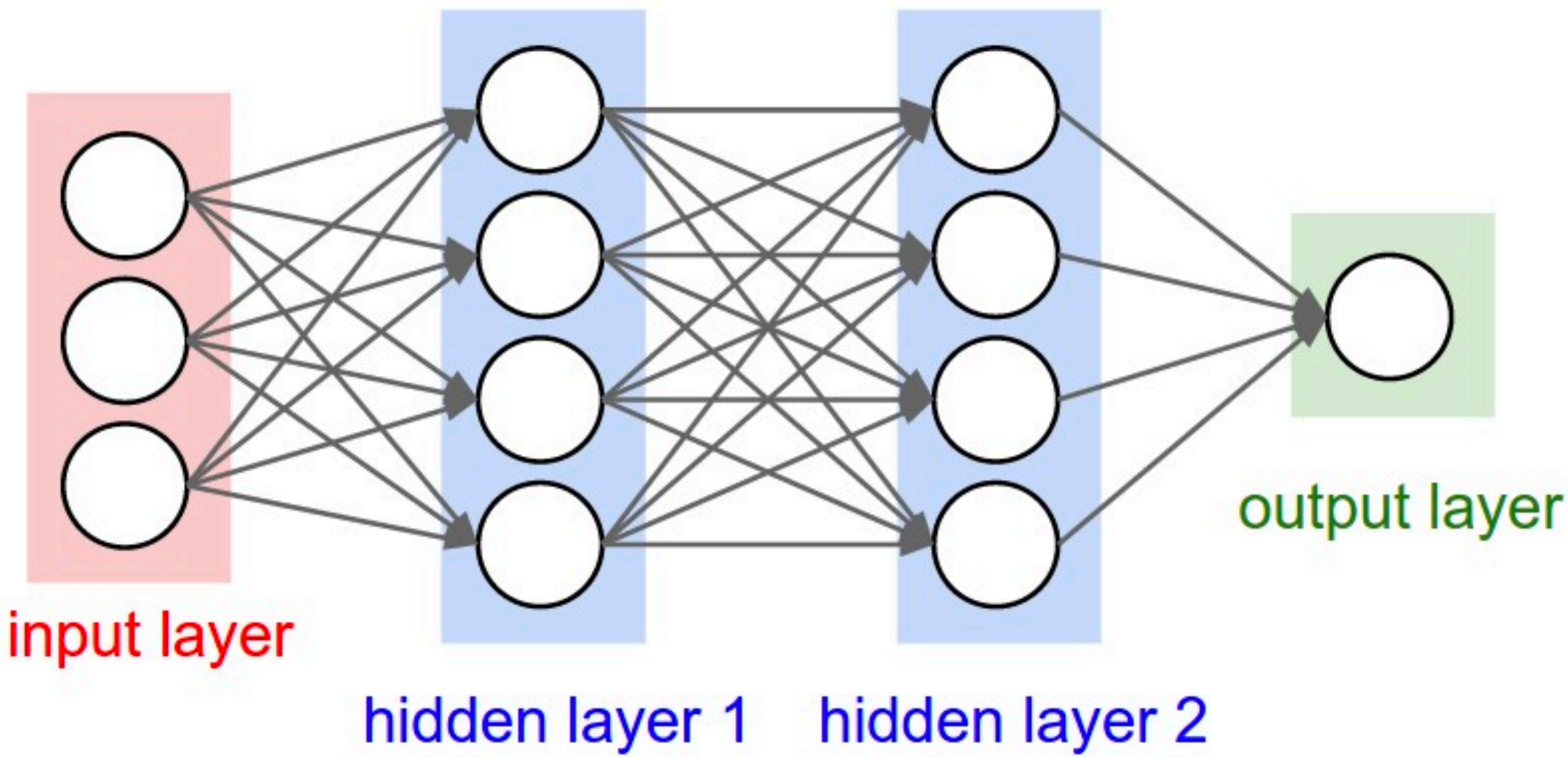


Feed-forward networks

- An ANN in which the connection between neurons does not form a cycle. It is the simplest ANN structure as information is only processed in one direction and never backwards.

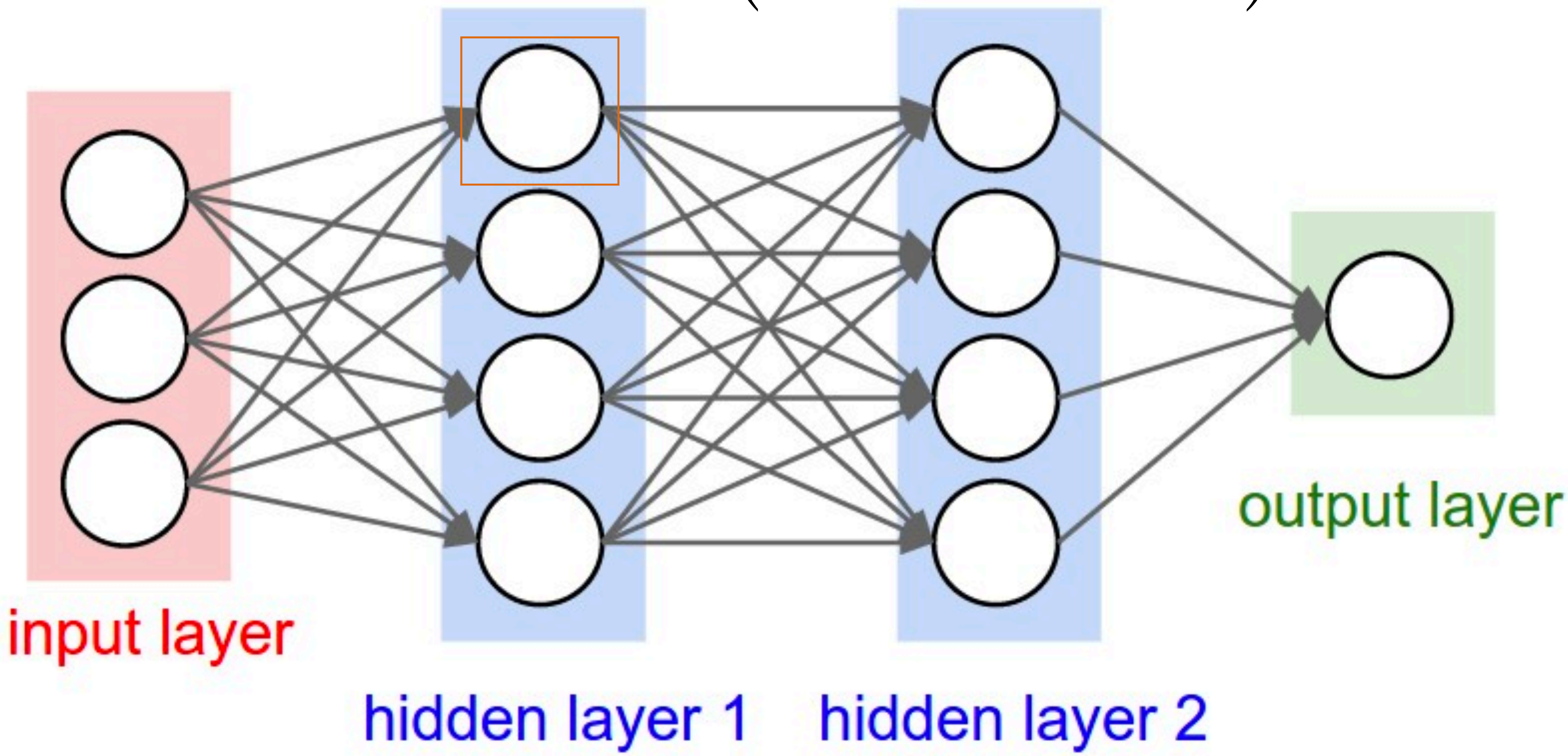
- The neurons are usually arranged layer-by-layer.

Example: feed-forward networks (deeper)



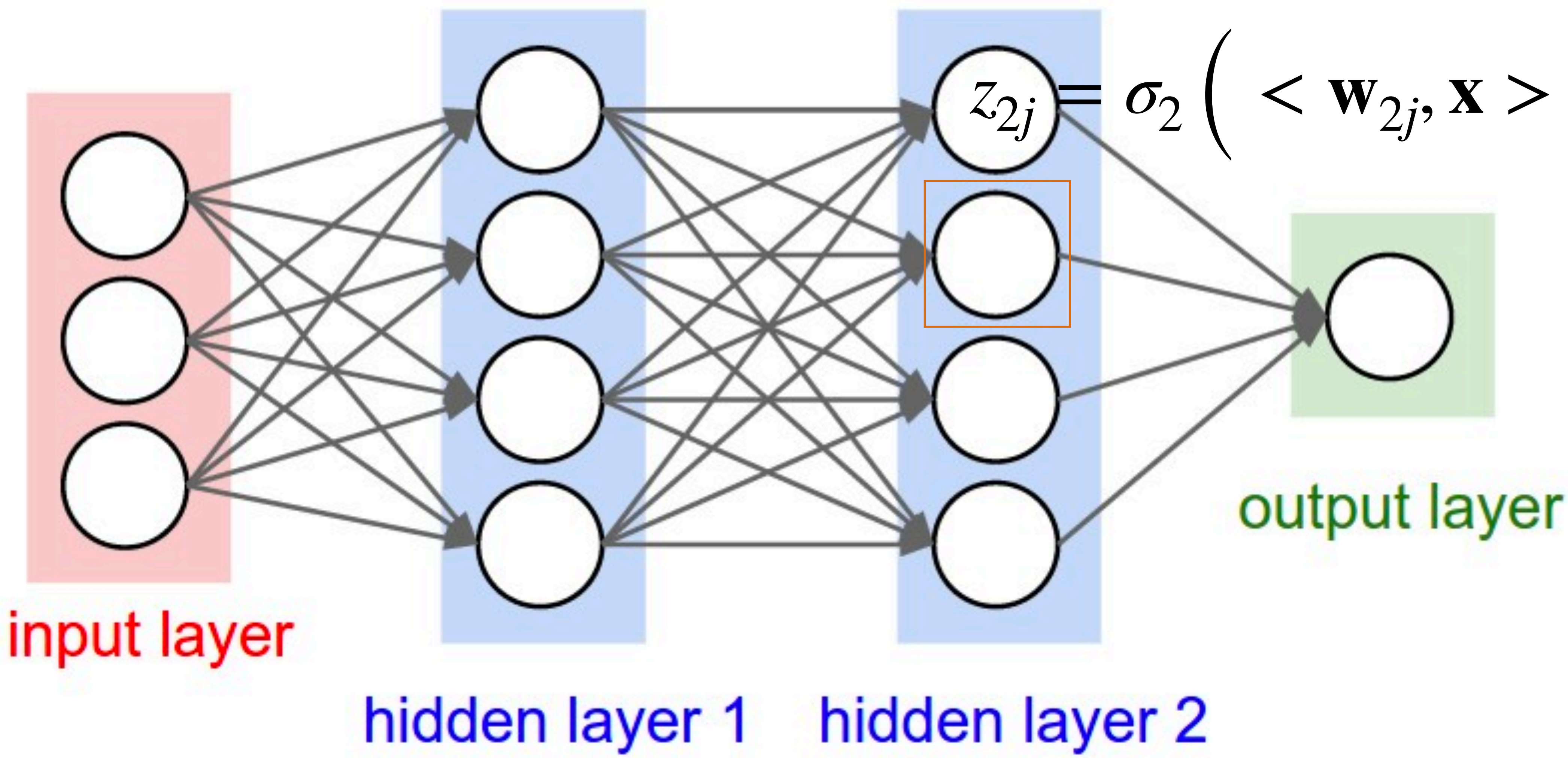
https://cs231n.github.io/convolutional-networks/

$$z_{1j} = \sigma_1\left(< \mathbf{w}_{1j}, \mathbf{x} > + b_{1j} \right)$$



T547

$$z_{2j} = \sigma_2\Big(< \mathbf{w}_{2j}, \mathbf{x} > + b_{2j} \Big)$$



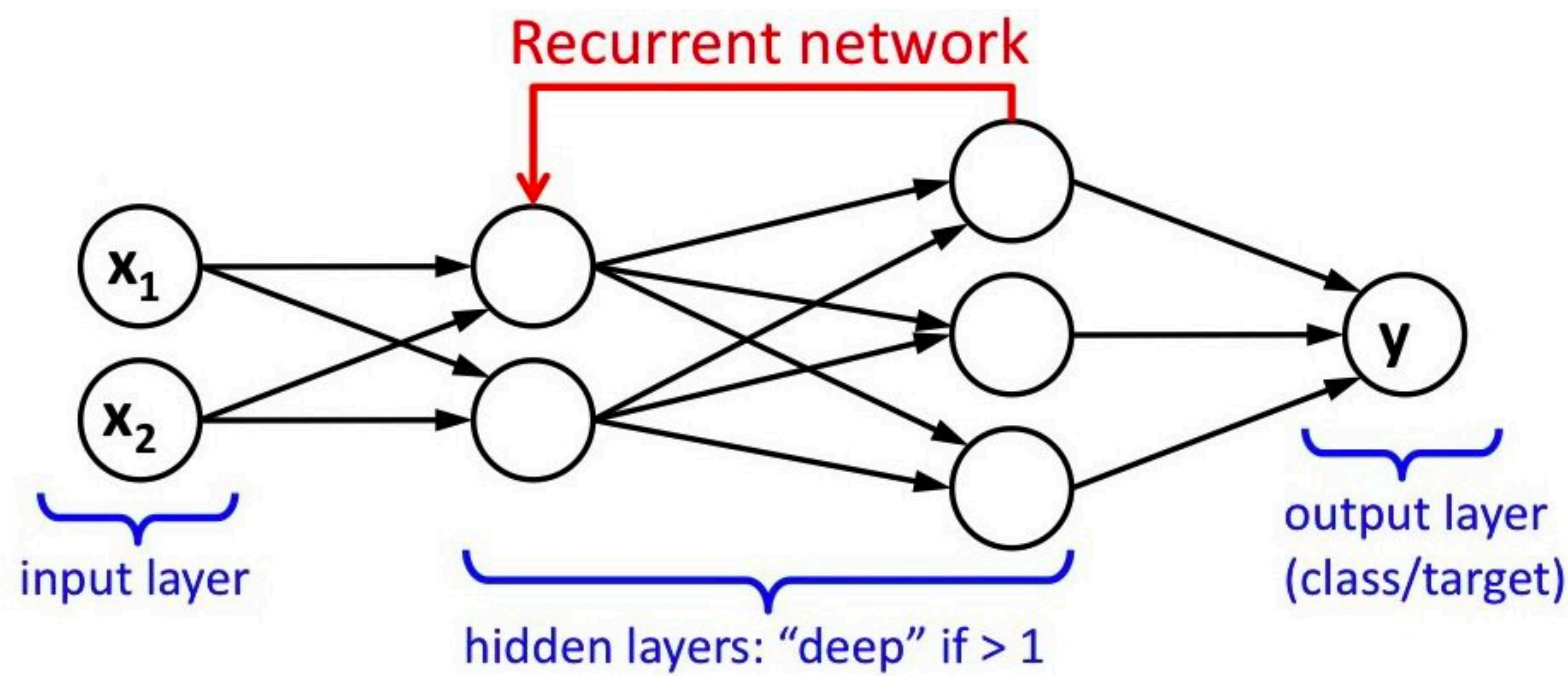
T547

$$
\begin{aligned}
\mathbf{z}_1 &= \sigma_1\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right) \\
\mathbf{z}_2 &= \sigma_2\left(\mathbf{W}_2\mathbf{z}_1 + \mathbf{b}_2\right) \\
&\vdots \\
\mathbf{z}_\ell &= \sigma_\ell\left(\mathbf{W}_\ell\mathbf{z}_{\ell-1} + \mathbf{b}_\ell\right) \\
f(\mathbf{x}) &= \sigma_f\left(\mathbf{W}_f\mathbf{z}_\ell + \mathbf{b}_f\right)
\end{aligned}
$$

The empirical risk of a feed-forward networks becomes

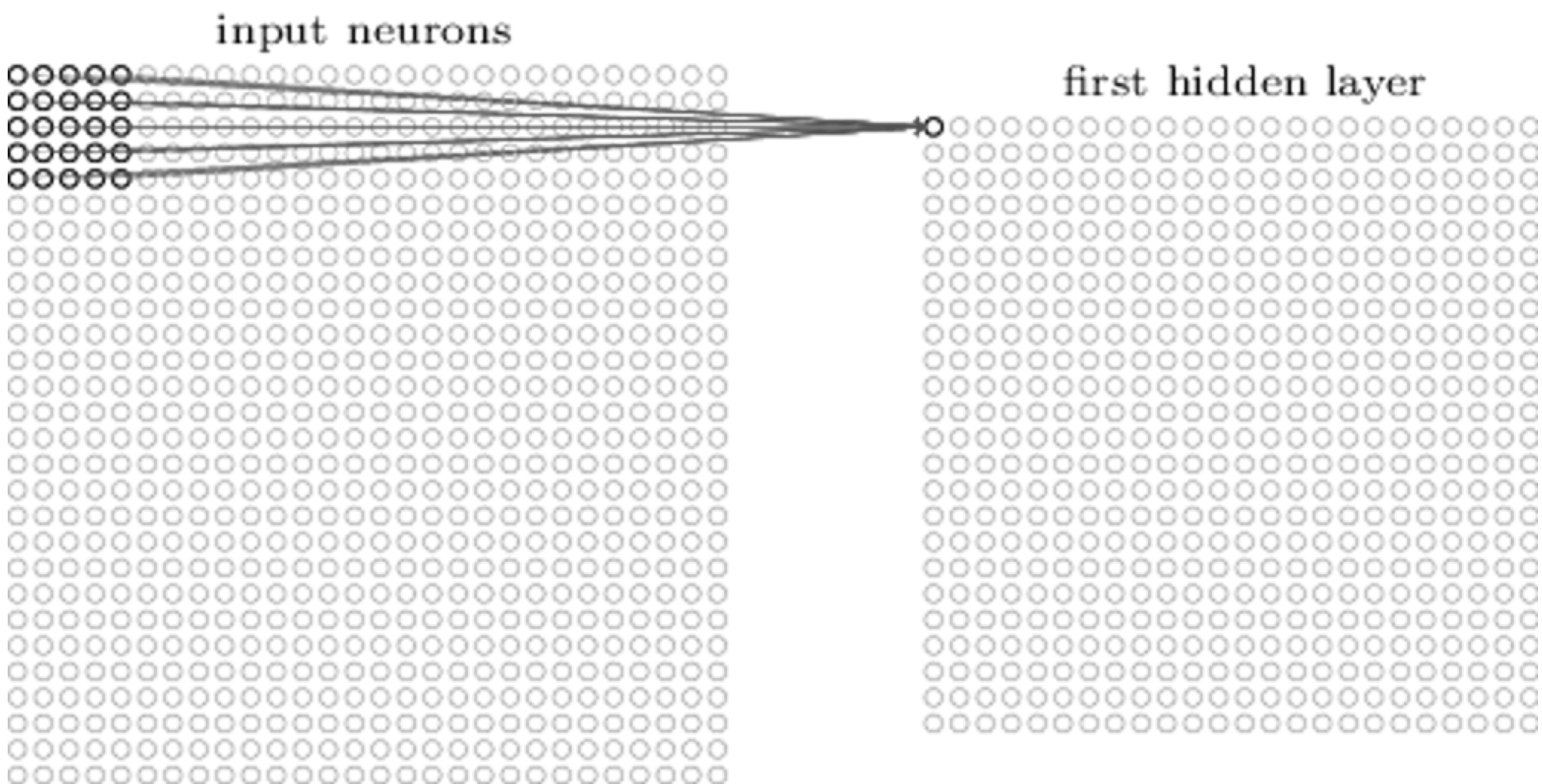
$$
R = \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, f(\mathbf{x}_i)\right)
$$

Example: recurrent neural networks



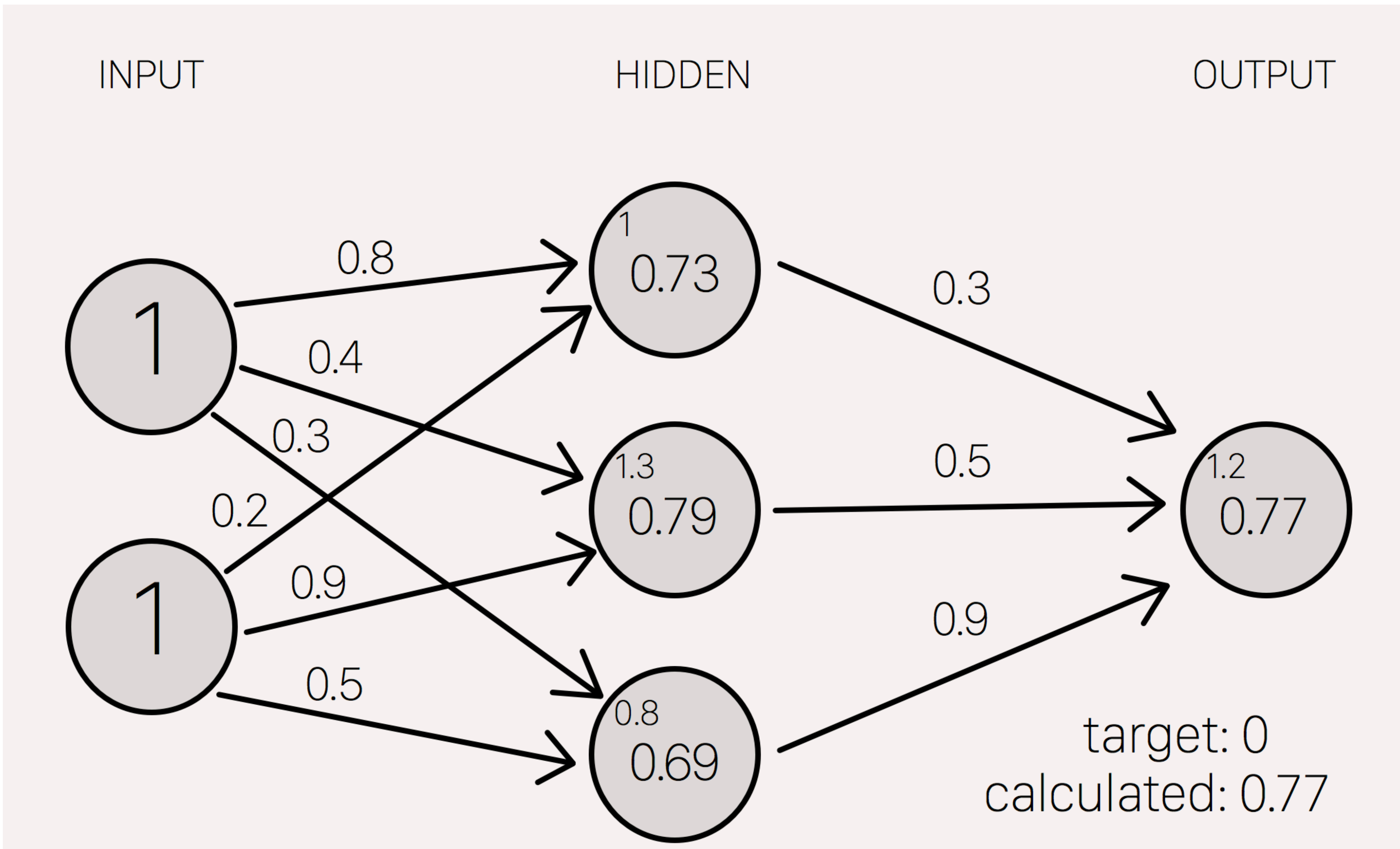
https://bit.ly/2HIXgH9

Example: convolutional neural networks



Granulated deep learning and Z-numbers in motion detection and object recognition

Forward evaluation



https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/

Recap: feed–forward networks

$$
\begin{aligned}
\mathbf{z}_1 &= \sigma_1\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right) \\
\mathbf{z}_2 &= \sigma_2\left(\mathbf{W}_2\mathbf{z}_1 + \mathbf{b}_2\right) \\
&\vdots \\
\mathbf{z}_\ell &= \sigma_\ell\left(\mathbf{W}_\ell\mathbf{z}_{\ell-1} + \mathbf{b}_\ell\right) \\
f(\mathbf{x}) &= \sigma_{\ell+1}\left(\mathbf{W}_{\ell+1}\mathbf{z}_\ell + \mathbf{b}_f\right)
\end{aligned}
$$

The empirical risk of a feed–forward networks becomes

$$
R = \frac{1}{n}\sum_{i=1}^{n} L\left(y_i, f(\mathbf{x}_i)\right)
$$

Automatic differentiation by <u>backpropagation</u>

Obtain ∇R automatically by chain rules:

$$\frac{\partial R}{\partial \mathbf{W}_{\ell+1}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial L\left(y_i, f(\mathbf{x}_i)\right)}{\partial f} \frac{\partial f}{\partial \mathbf{W}_{\ell+1}},$$

$$\frac{\partial R}{\partial \mathbf{W}_{\ell}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial L\left(y_i, f(\mathbf{x}_i)\right)}{\partial f} \frac{\partial f}{\partial \mathbf{z}_{\ell}} \frac{\partial \mathbf{z}_{\ell}}{\mathbf{W}_{\ell}},$$

$$\frac{\partial R}{\partial \mathbf{W}_{\ell-1}} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial L\left(y_i, f(\mathbf{x}_i)\right)}{\partial f} \frac{\partial f}{\partial \mathbf{z}_{l}} \frac{\partial \mathbf{z}_{\ell}}{\partial \mathbf{z}_{\ell-1}} \frac{\partial \mathbf{z}_{\ell-1}}{\partial \mathbf{W}_{\ell-1}},$$

$$\vdots$$

Why does deep learning so successful?

- Universal approximation
- ReLU networks are universal approximations via piecewise linear or constant functions
- Overparameterization in deep learning does not lead to overfitting
- Gradient descent finds global minima of deep neural networks

Summary

- Computational graphs are graphical representations of mathematical functions equipped with automatic differentiations
- Neural networks are special cases of computational graphs
- Theoretical justifications for the success of deep neural networks