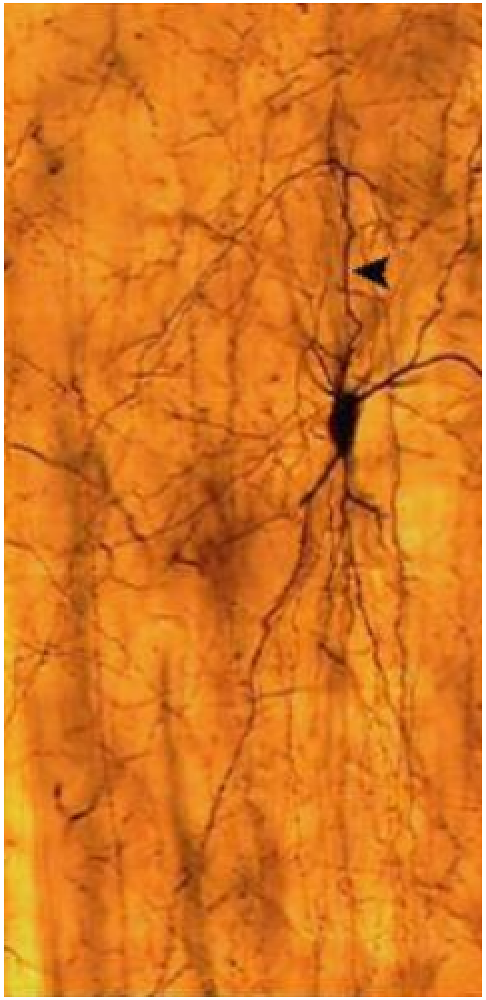
Freie Universität Berlin



Neural Networks, Part 1

Data Visualization and Data Mining

WS 15/16

Talk 1 by Jana Cavojska

Instructors:

Prof. Dr. Agnès Voisard

Daniel Kressner

Overview

1. **What are neural networks?**
2. A single perceptron
3. Perceptron training
4. A simple network
5. Learning via backpropagation
6. Convolutional neural networks
7. 2nd talk: Visualization

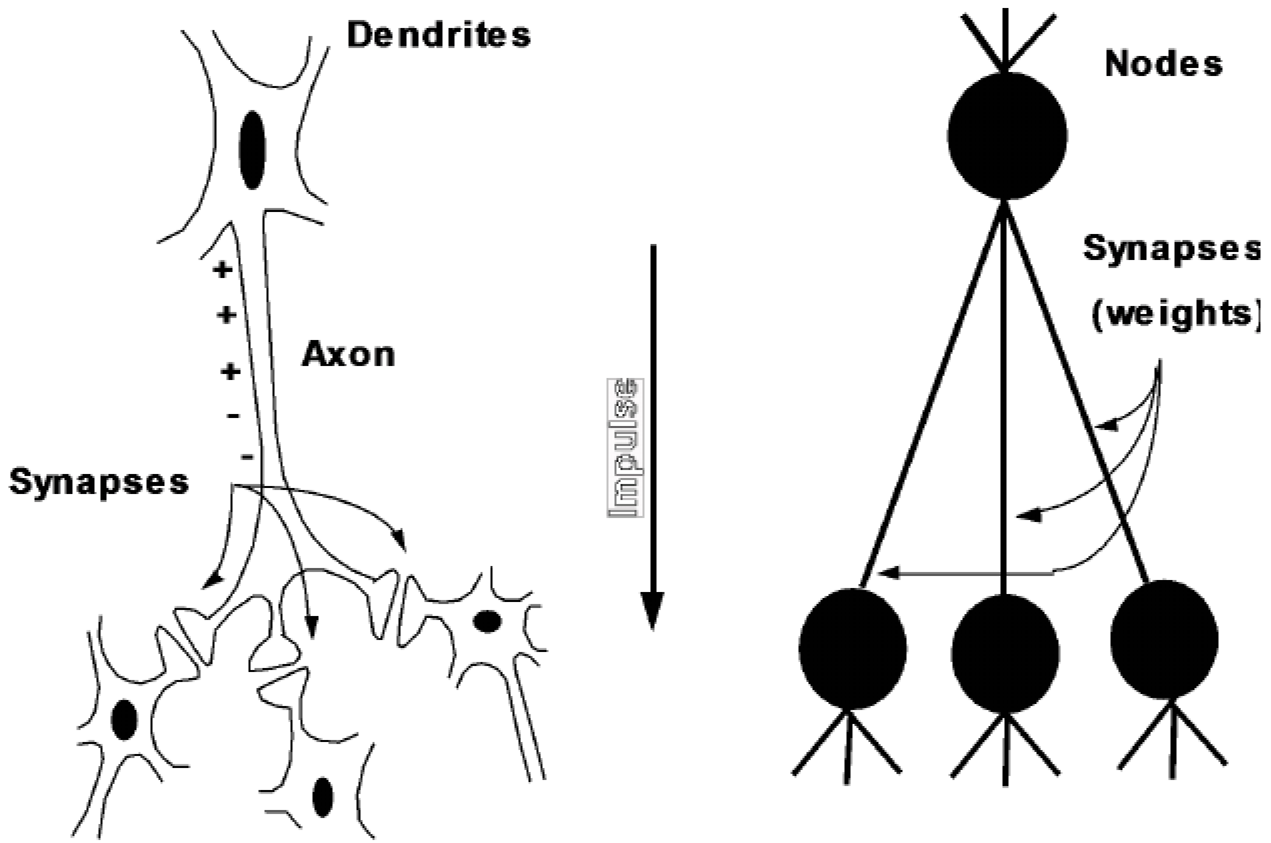
1. What are Neural Networks?

- **classifiers**:
Input data → Class of input data

- consist of layers of neurons (**perceptrons**)

- different **architectures** for different types of problems

Neuron

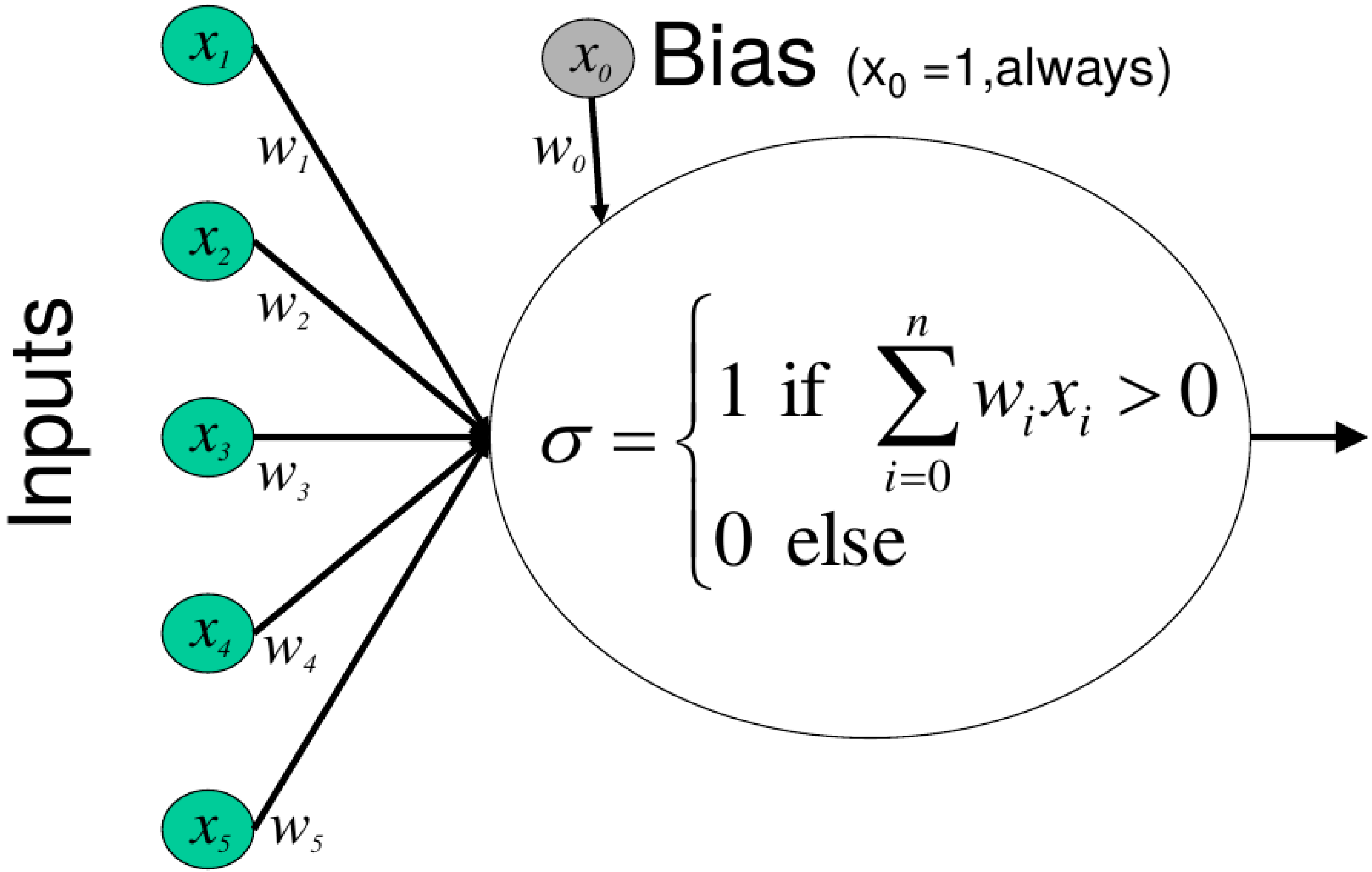


Source: [2]

Overview

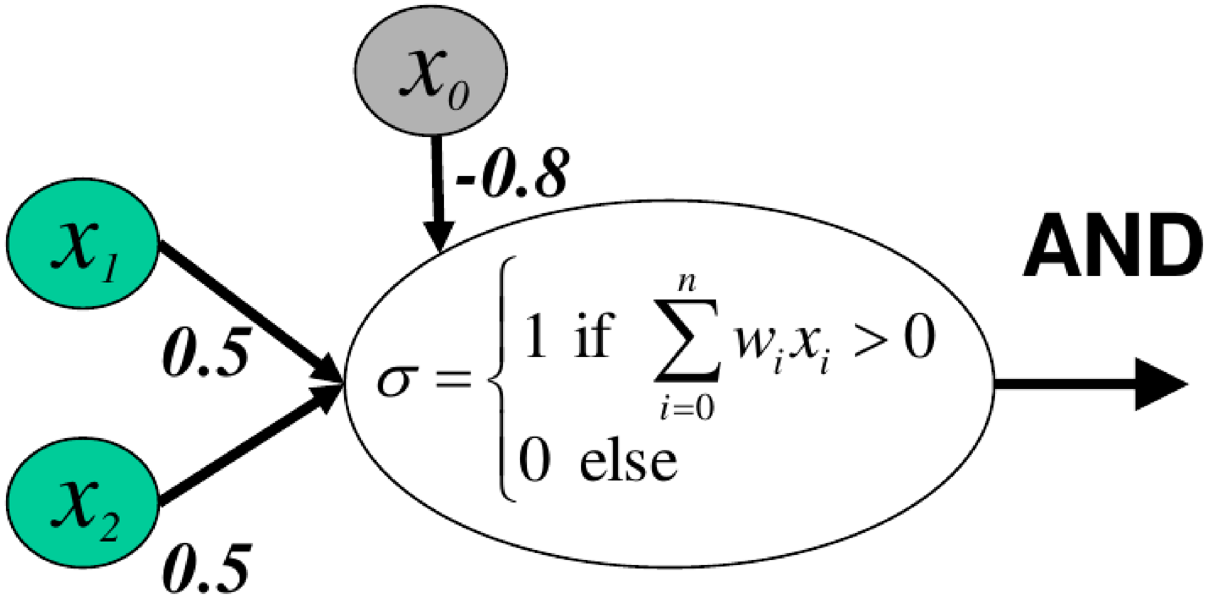
1. What are neural networks?
2. **A single perceptron**
3. Perceptron training
4. A simple network
5. Learning via backpropagation
6. Convolutional neural networks
7. 2nd talk: Visualization

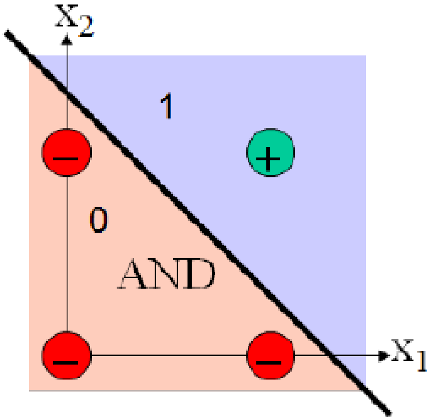
A Single Perceptron



Source: [2]

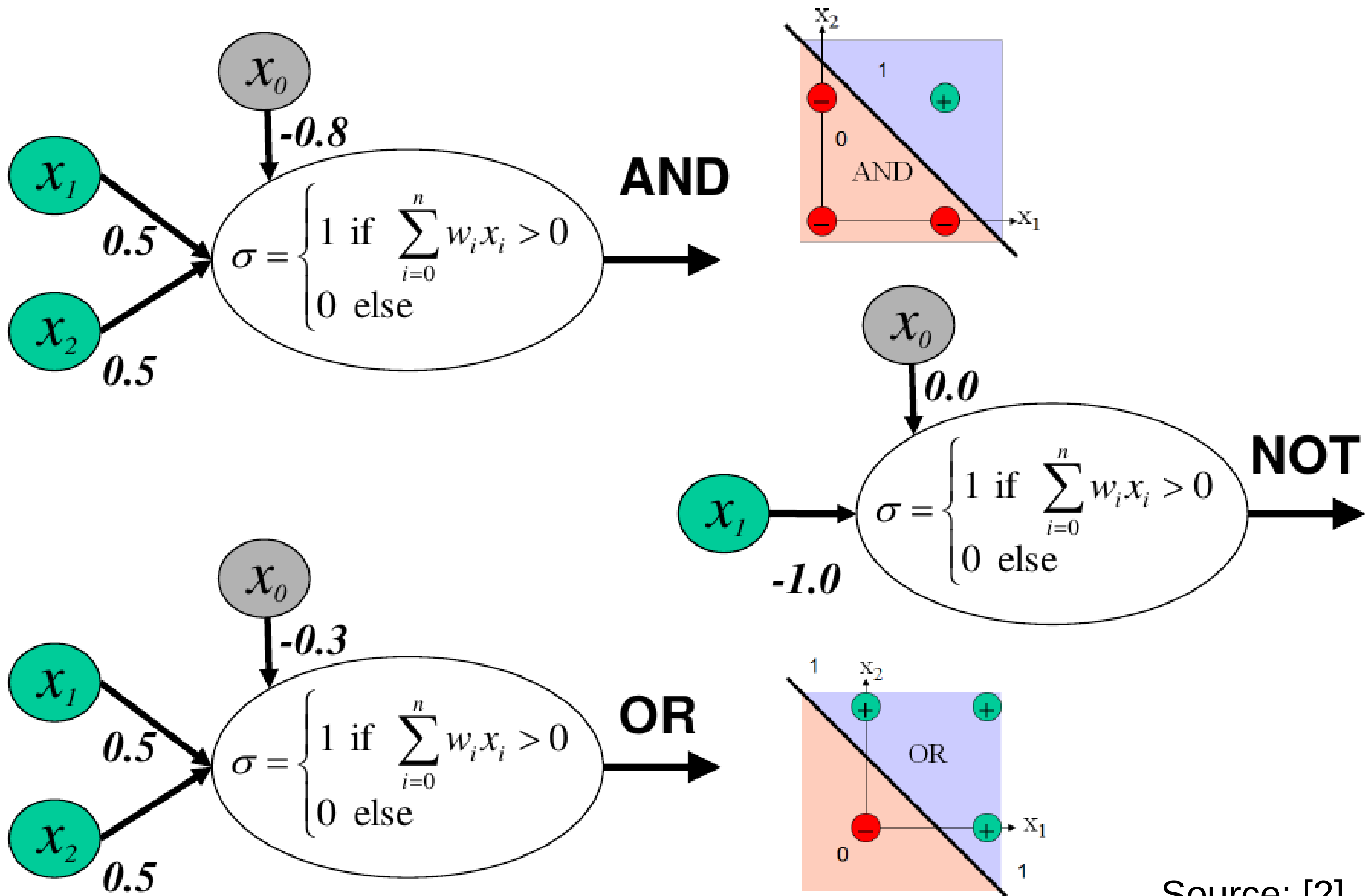
Logical Operators





Source: [2]

Logical Operators

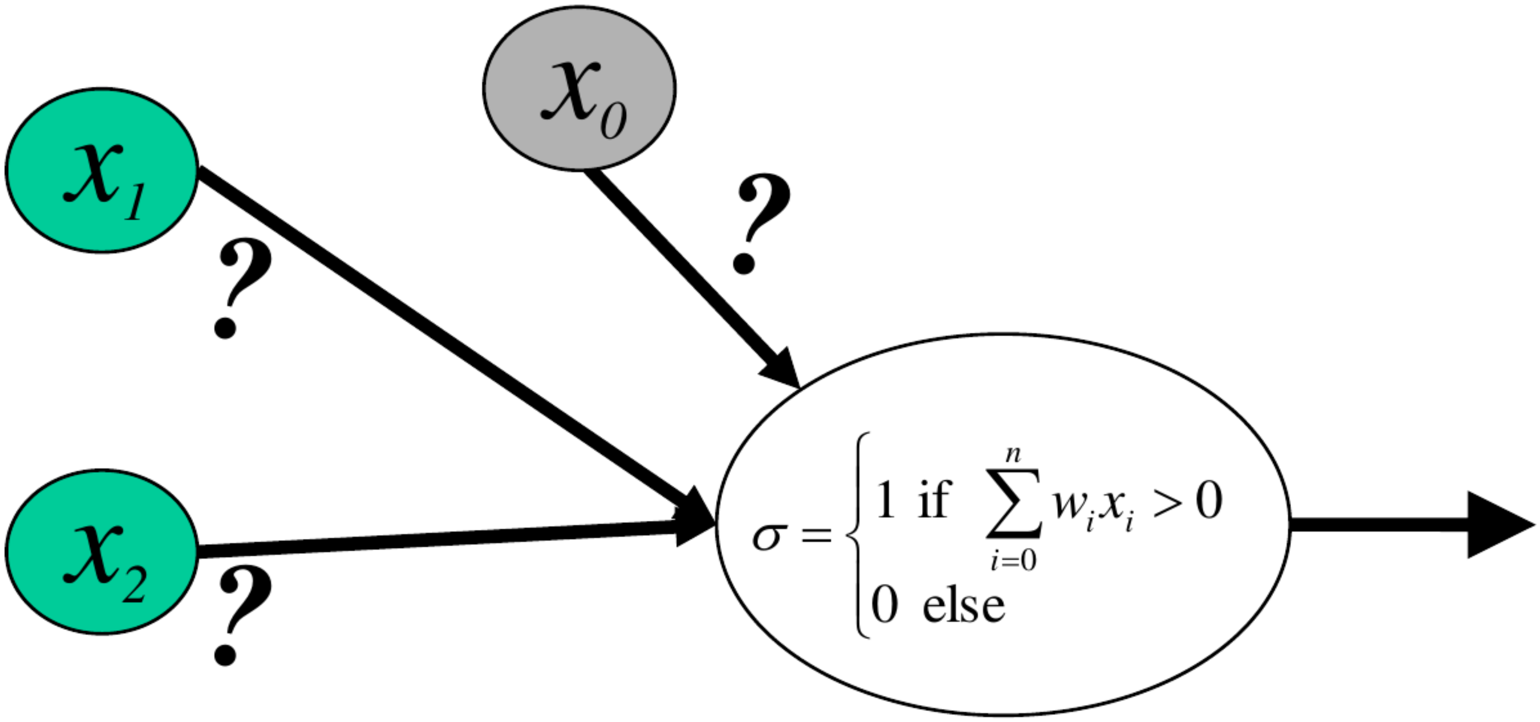


Source: [2]

Overview

1. What are neural networks?
2. A single perceptron
3. **Perceptron training**
4. A simple network
5. Learning via backpropagation
6. Convolutional neural networks
7. 2nd talk: Visualization

Perceptron Training?



Source: [2]

Freie Universität Berlin

Perceptron Training

$$w_i \leftarrow w_i + \Delta w_i$$

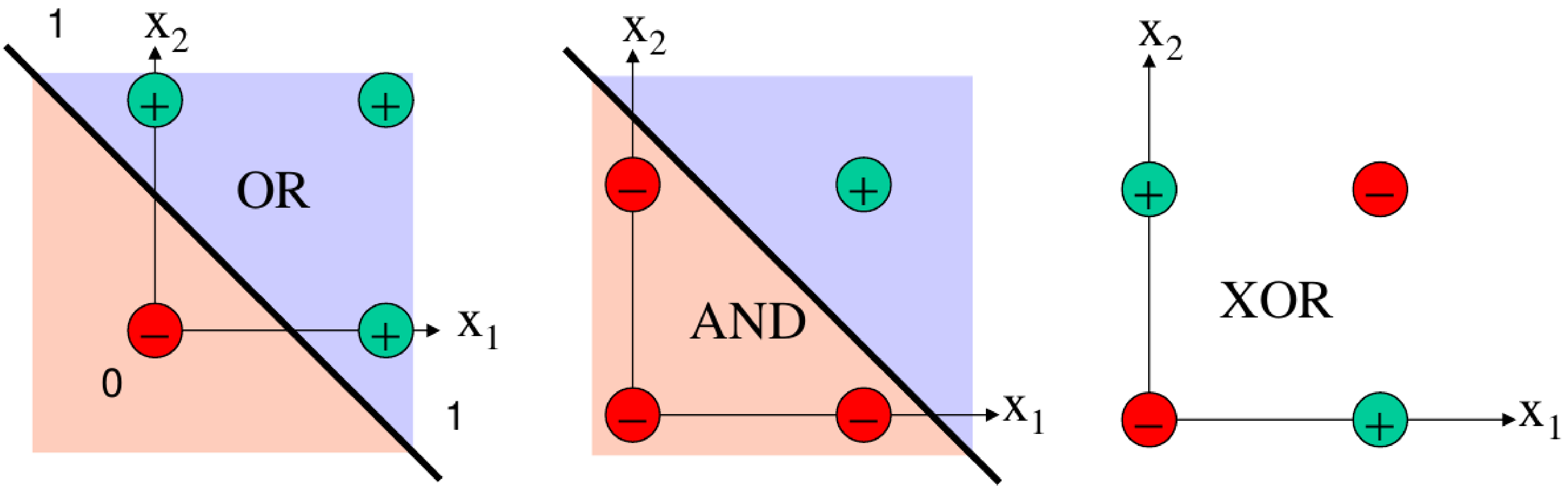
where

$$\Delta w_i = \eta(t - o)x_i$$

learning rate | target value | perceptron output | input value

Source: [2]

Linear Separability

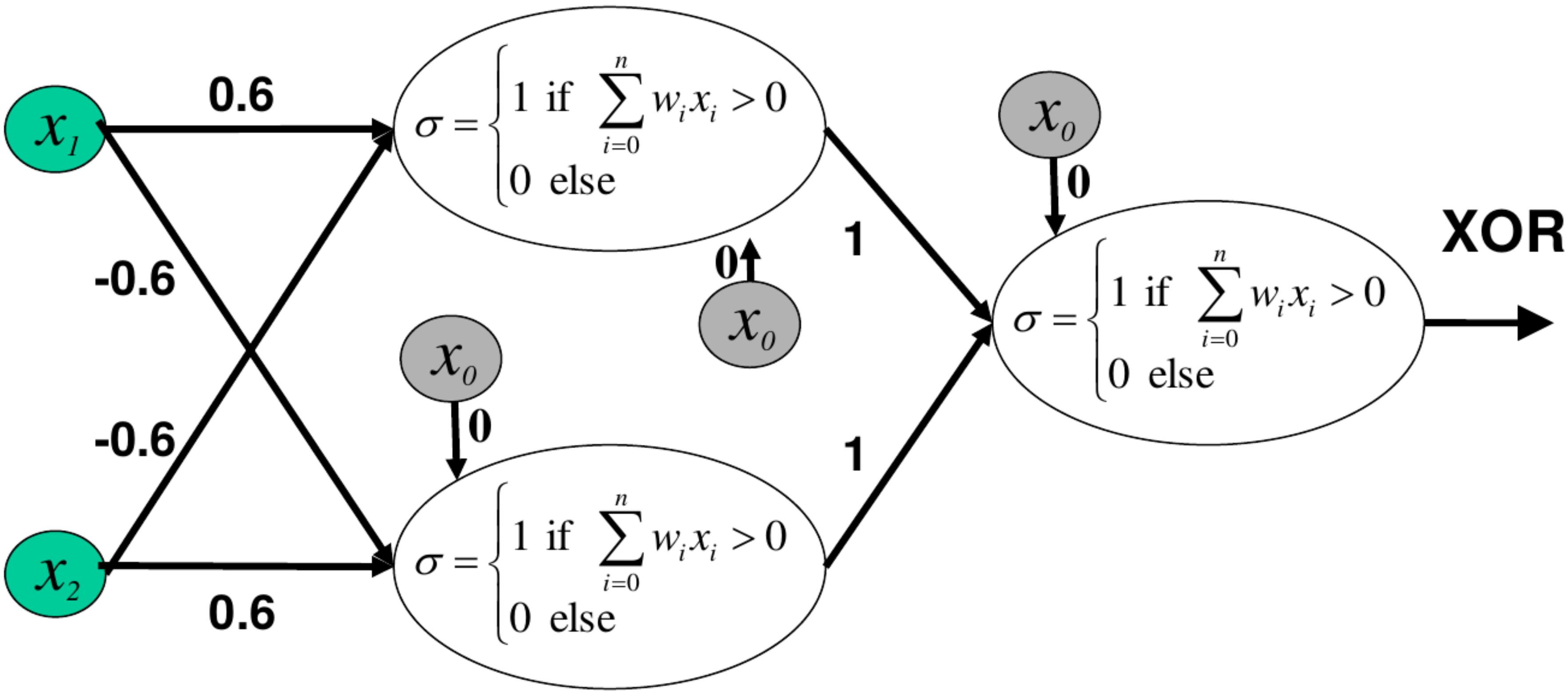


Source: [2]

Overview

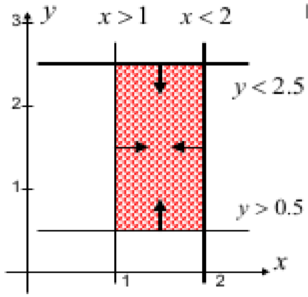
1. What are neural networks?
2. A single perceptron
3. Perceptron training
4. **A simple network**
5. Learning via backpropagation
6. Convolutional neural networks
7. 2nd talk: Visualization

XOR


x_1
x_2
0.6
-0.6
-0.6
0.6
x_0
0
x_0
0
x_0
0
1
1
$\sigma = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 \text{ else} \end{cases}$
$\sigma = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 \text{ else} \end{cases}$
$\sigma = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ 0 \text{ else} \end{cases}$
XOR

Source: [2]

Multiple Neurons



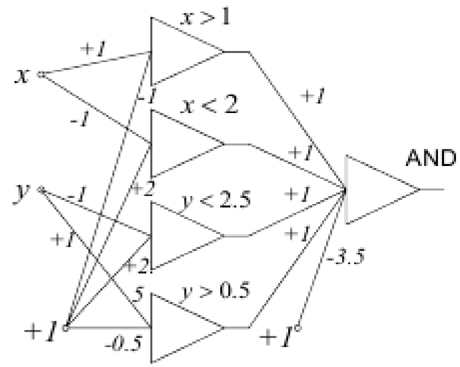
neuron equations:

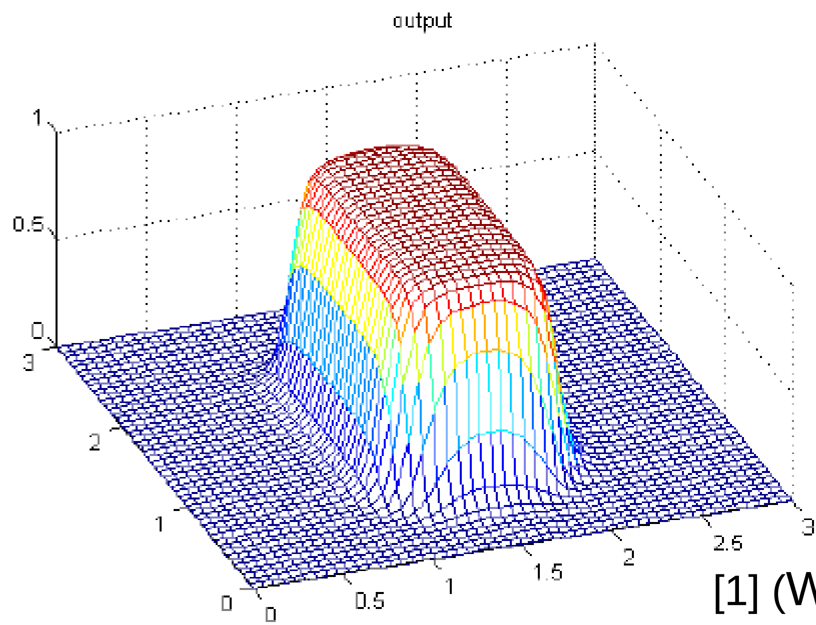
$$x - 1 > 0$$

$$-x + 2 > 0$$

$$-y + 2.5 > 0$$

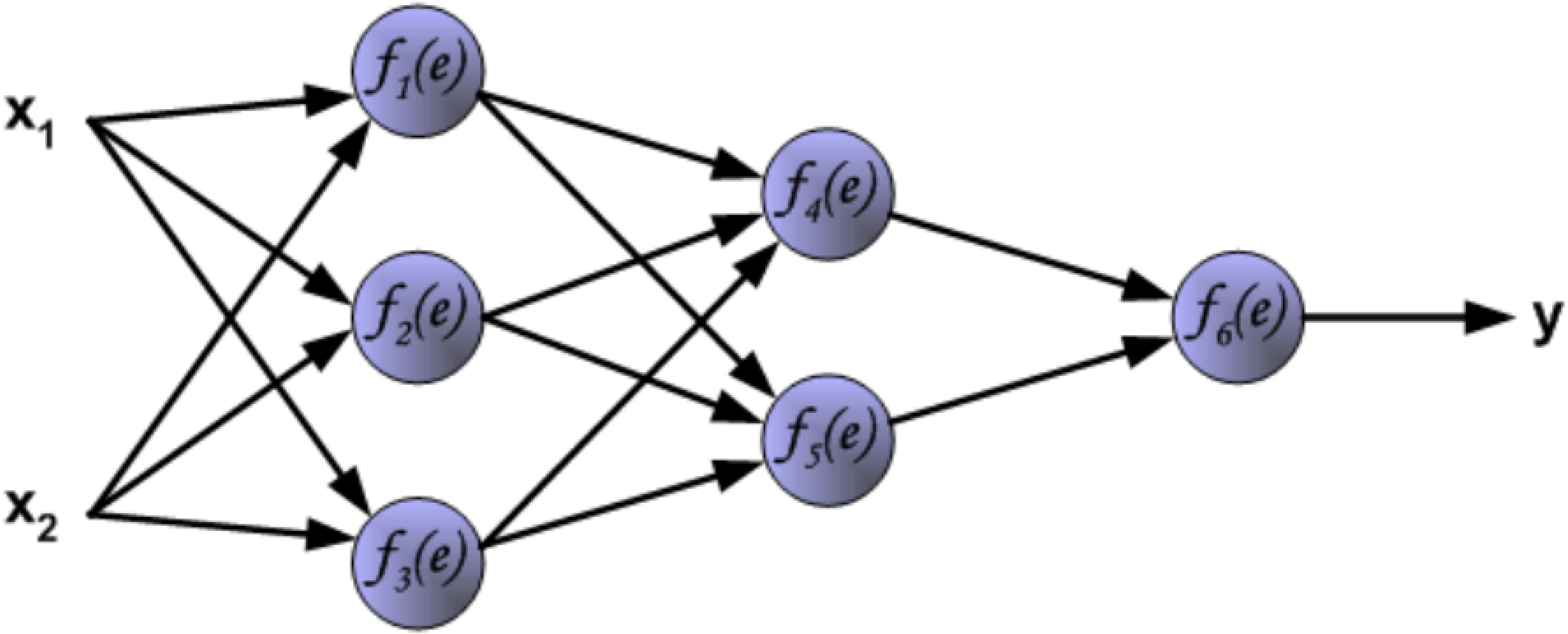
$$y - 0.5 > 0$$





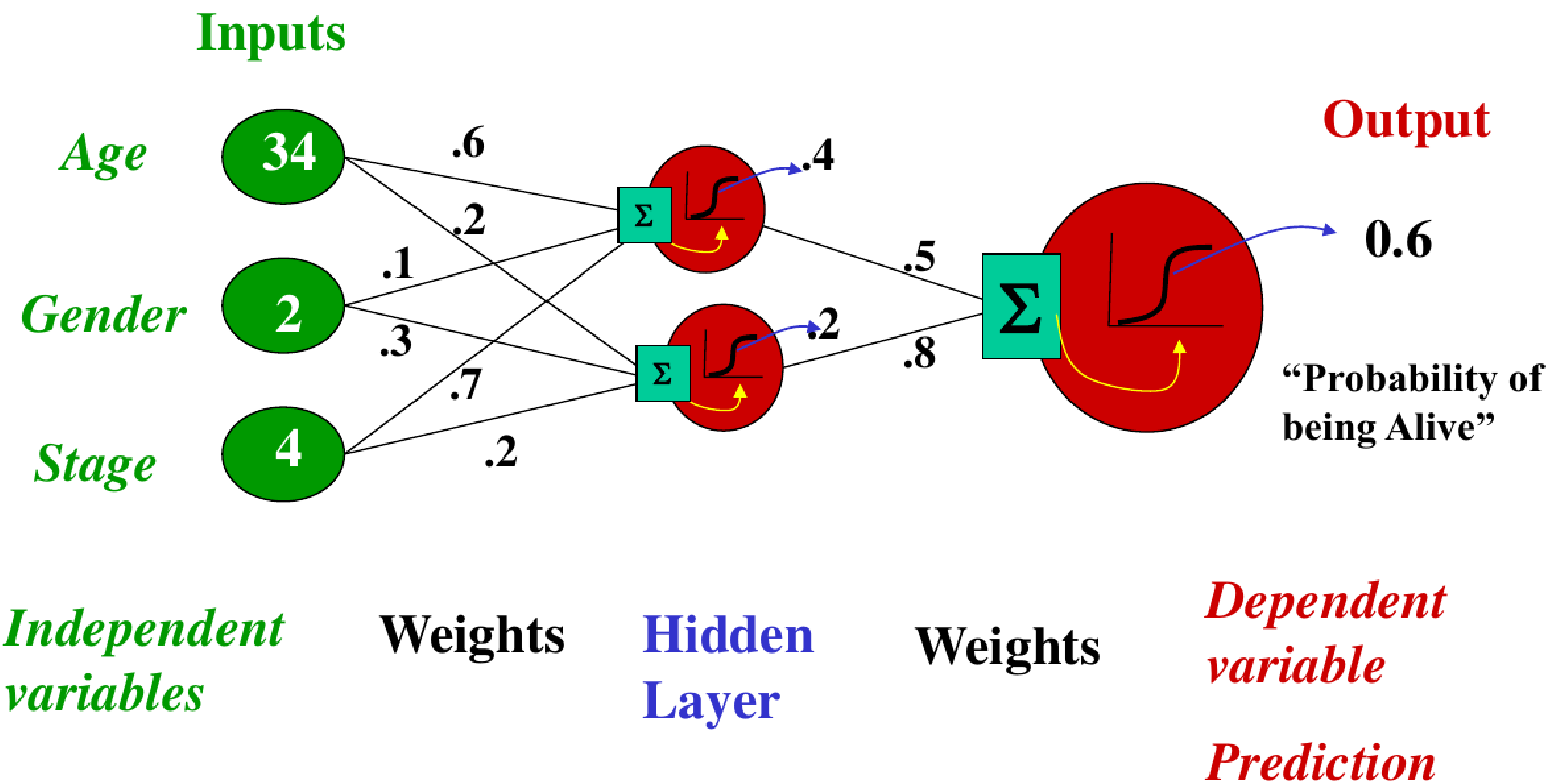
[1] (Wilamowski, 2003)

Example of a Simple Network



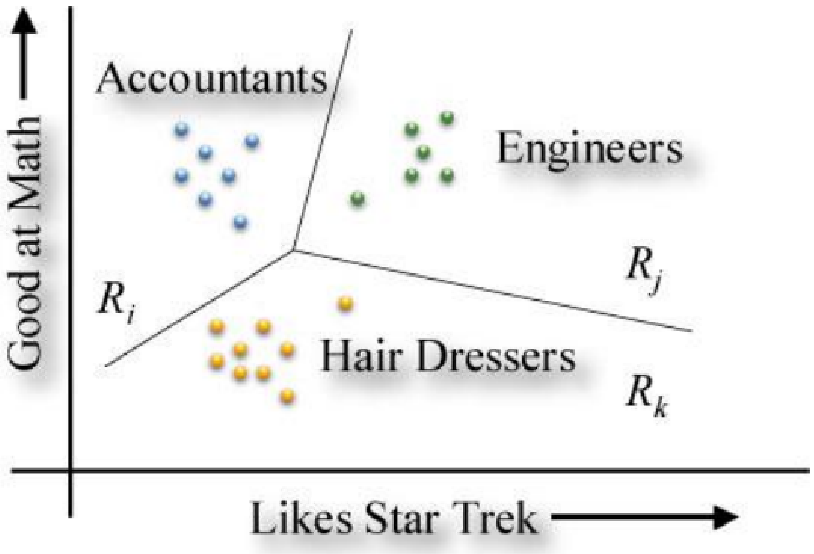
Source: [2]

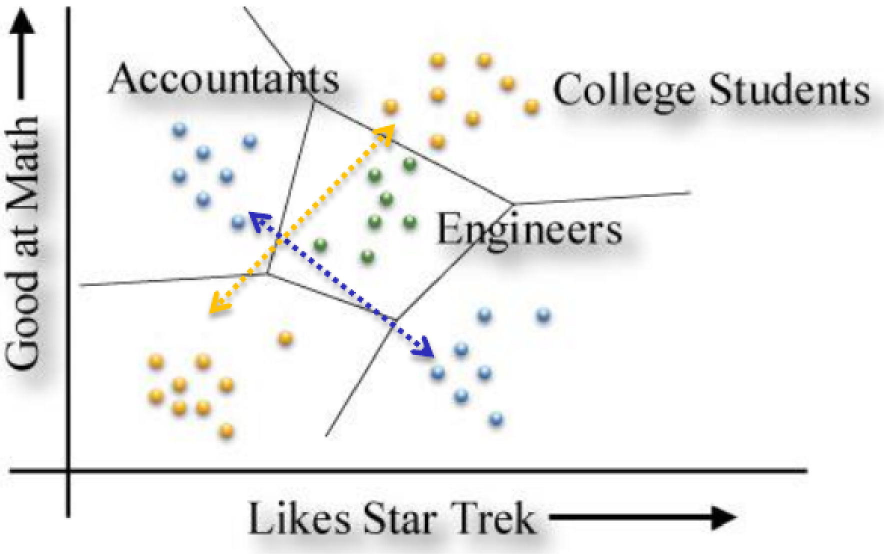
Neural Network Model



Source: [2]

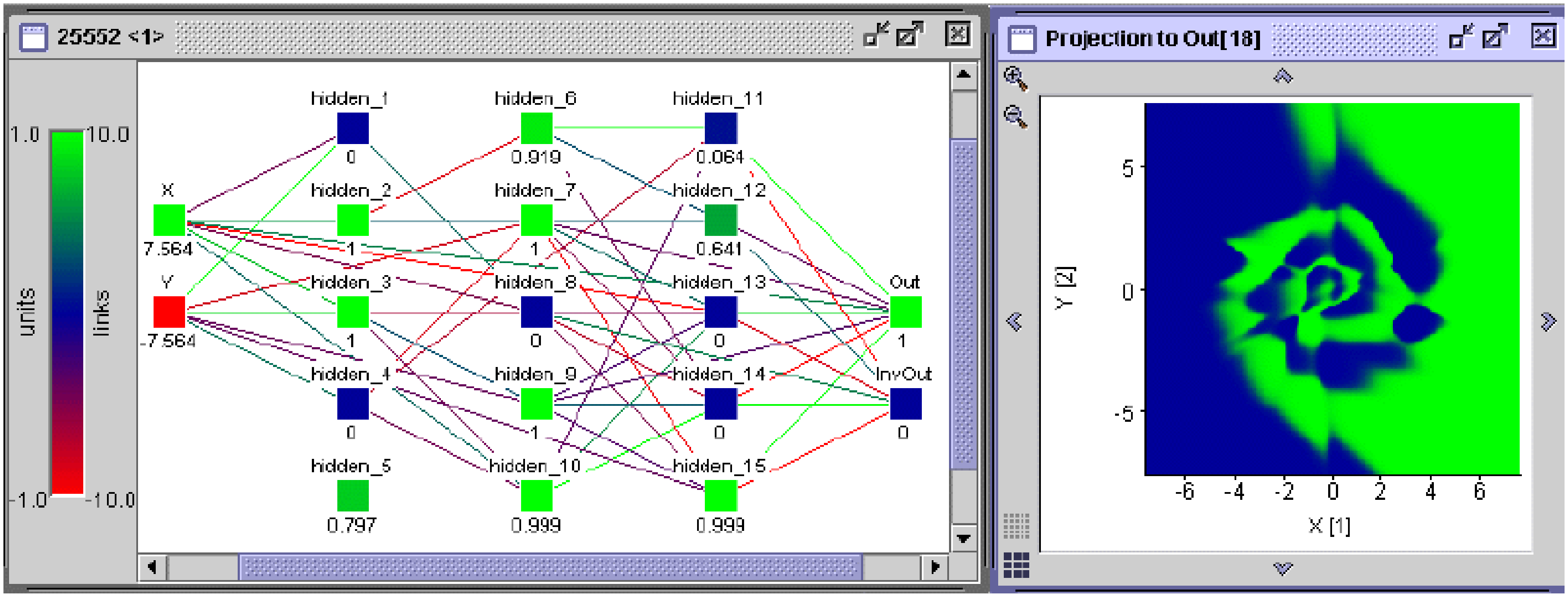
Being creative about Linear Separability





Source: [2]

Being creative about Linear Separability

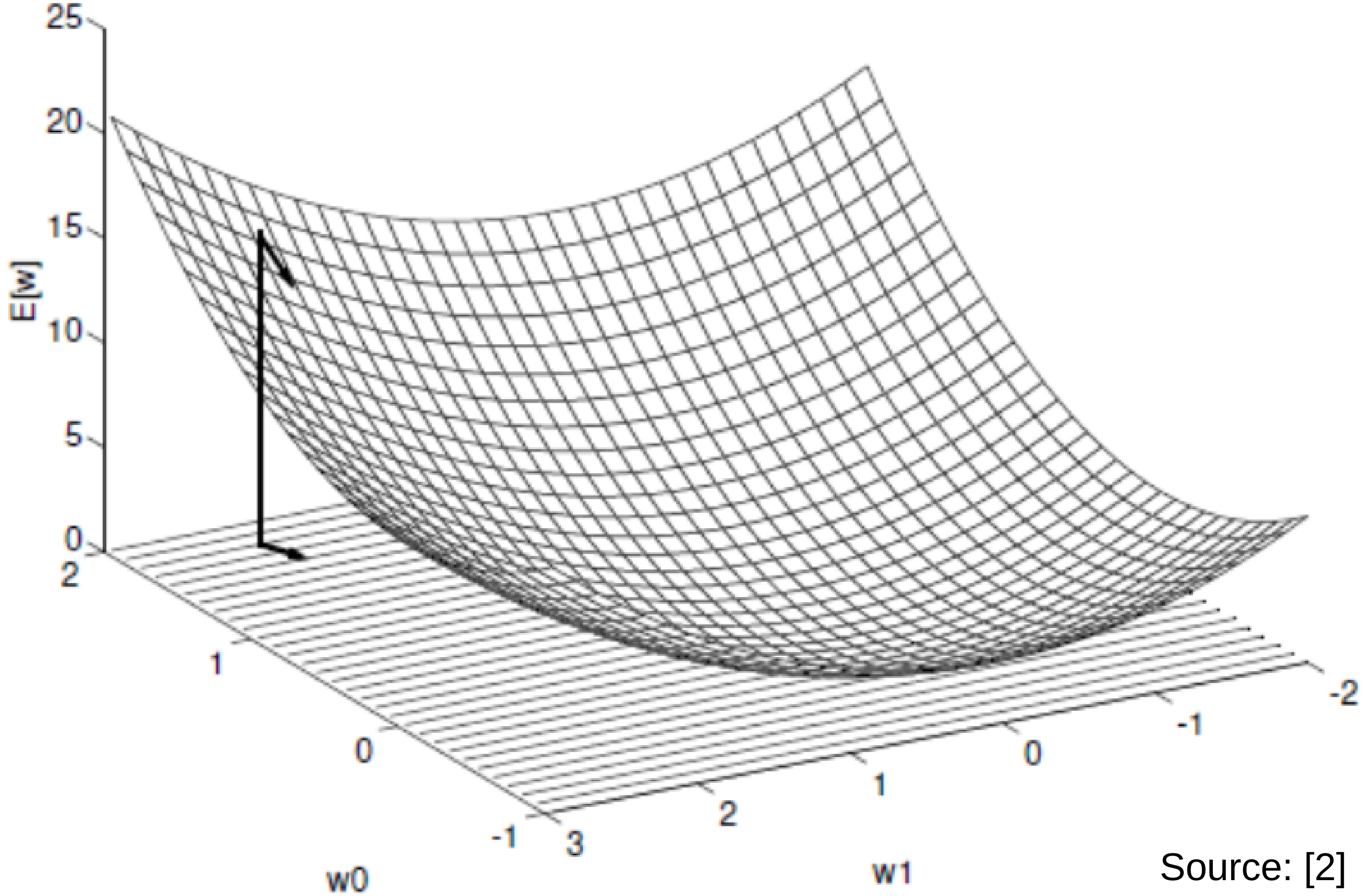


Source: [2]

Overview

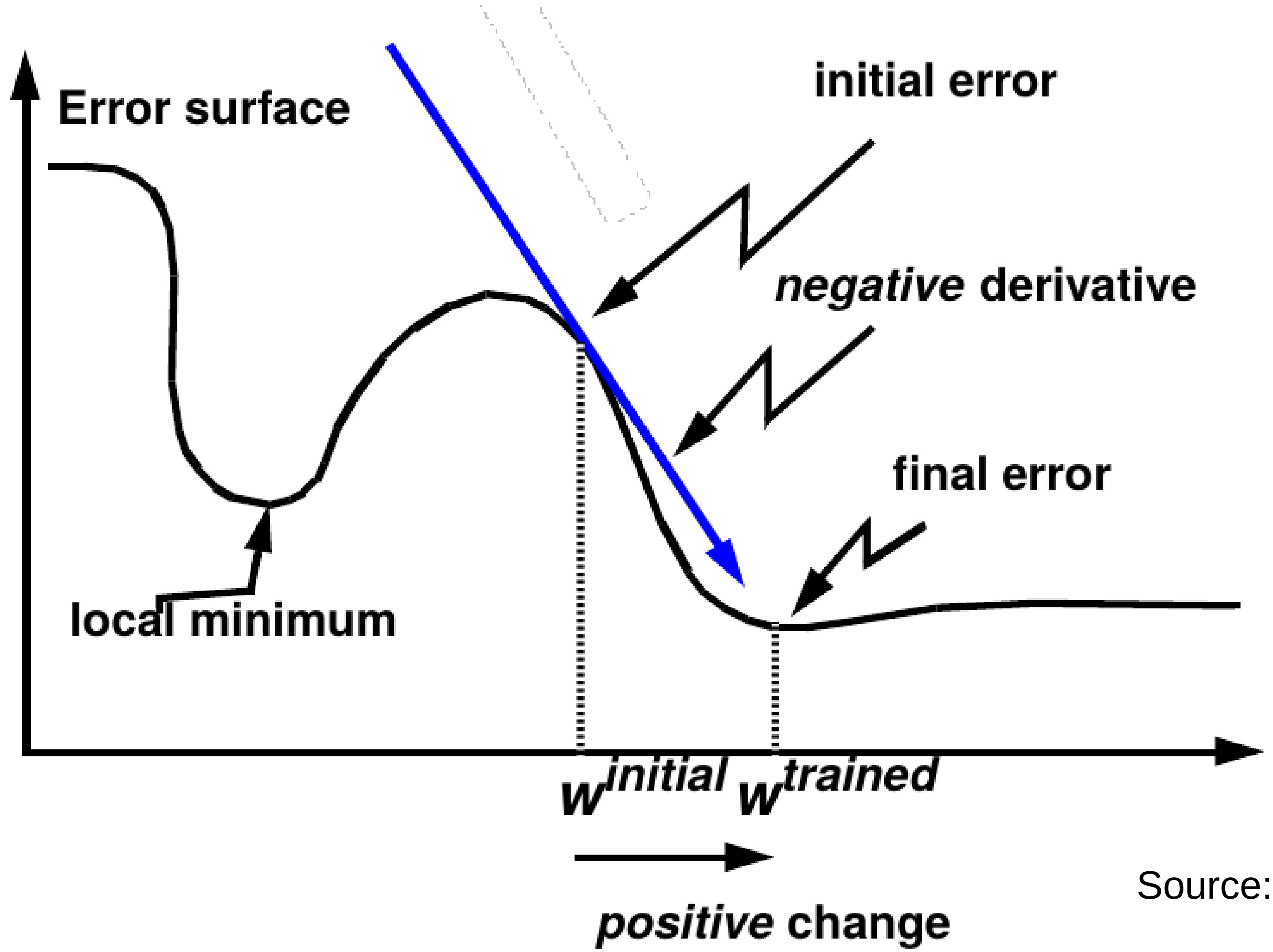
1. What are neural networks?
2. A single perceptron
3. Perceptron training
4. A simple network
5. **Learning via backpropagation**
6. Convolutional neural networks
7. 2nd talk: Visualization

Gradient Descent



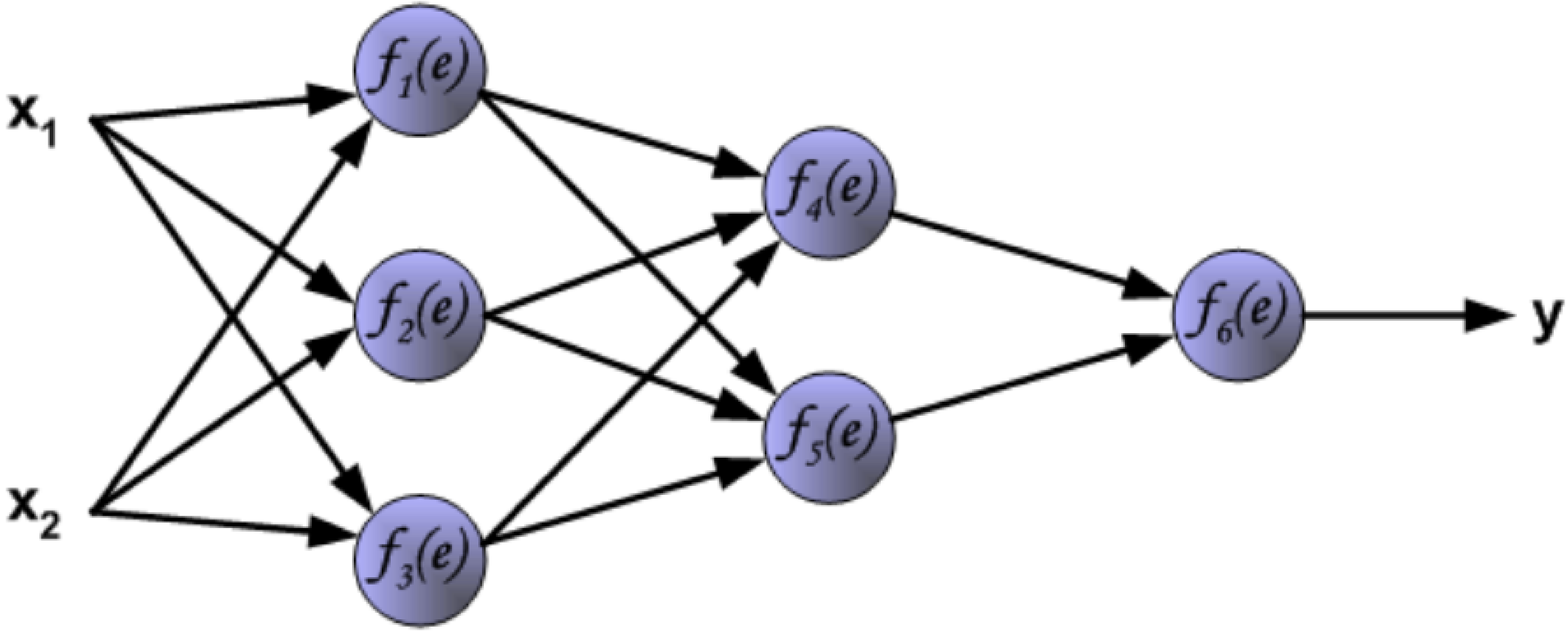
Source: [2]

Gradient Descent



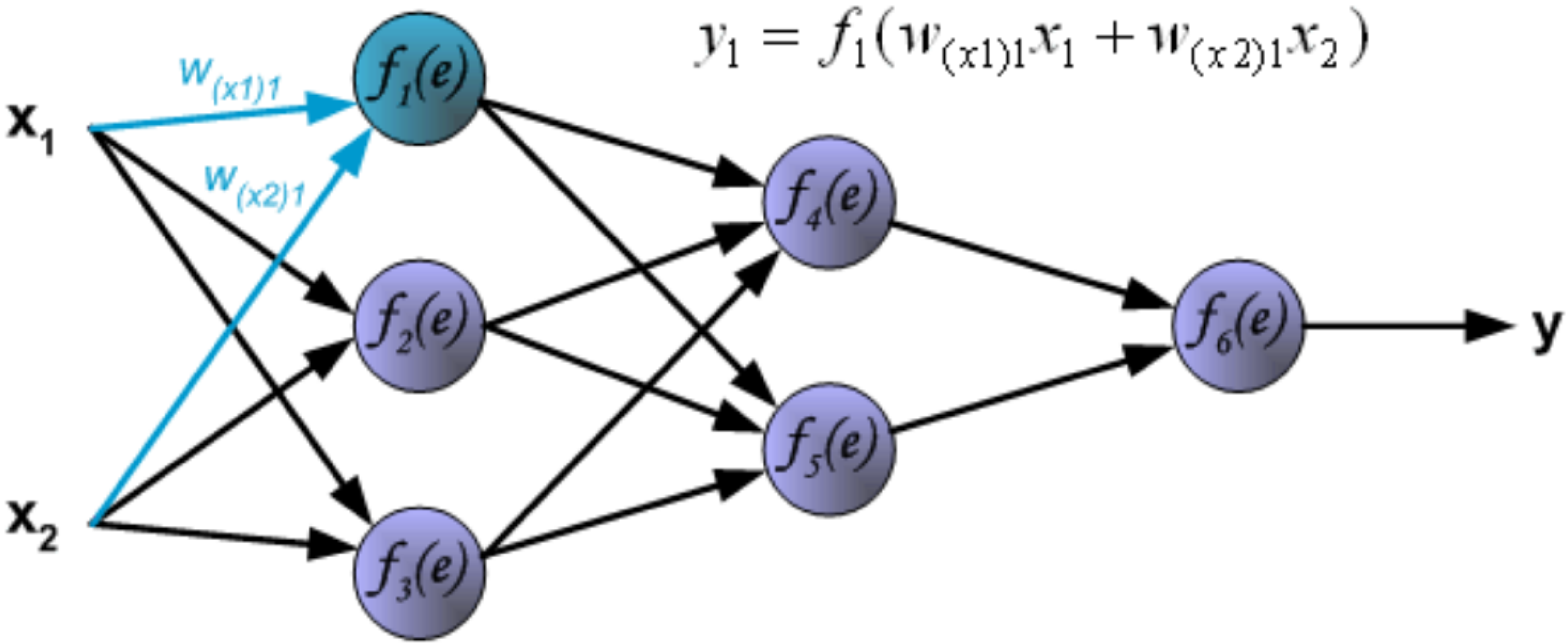
Source: [2]

Training using Backpropagation



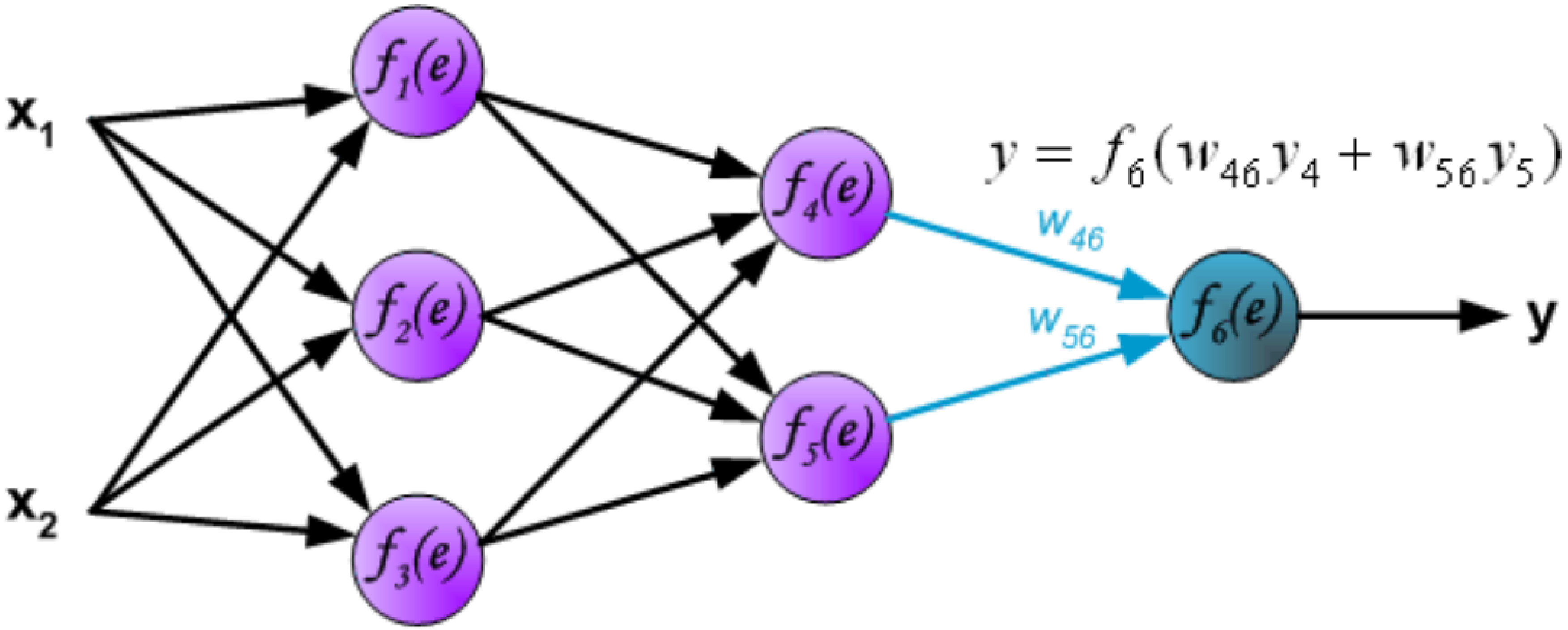
Source: [3]

Training using Backpropagation



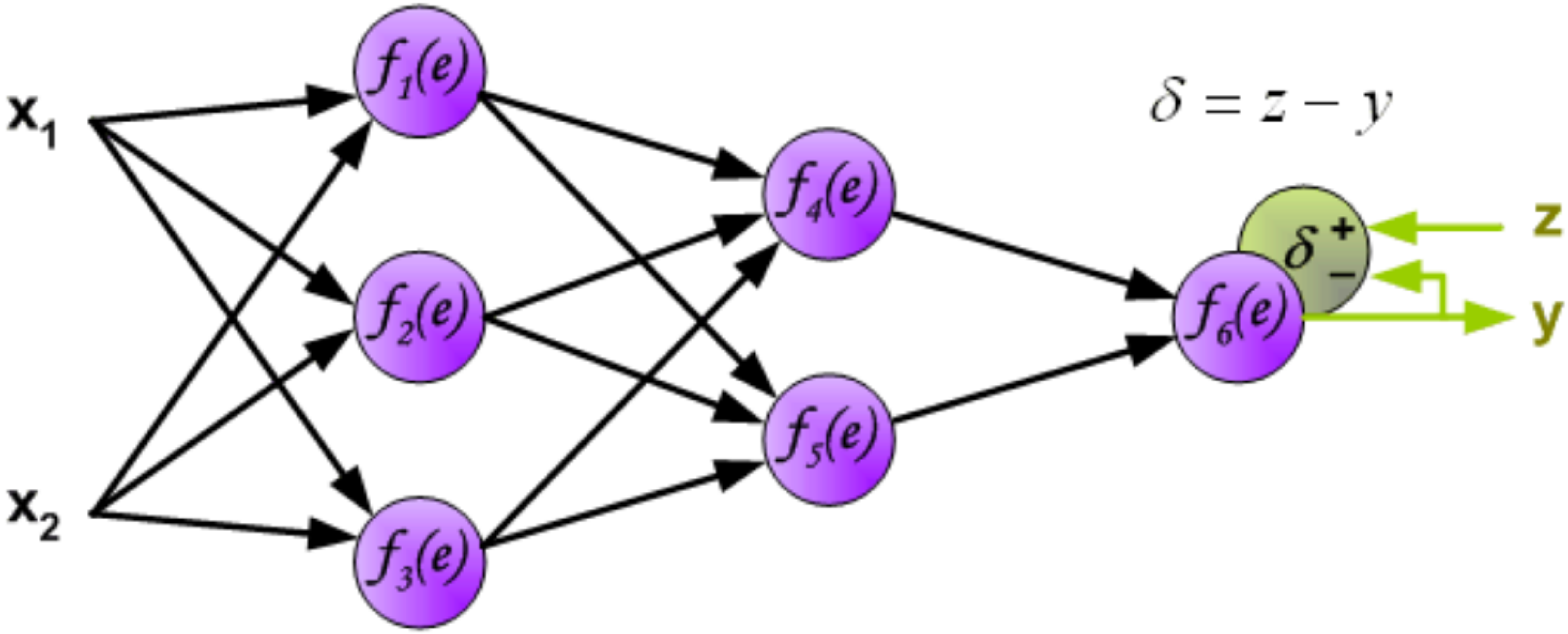
Source: [3]

Training using Backpropagation



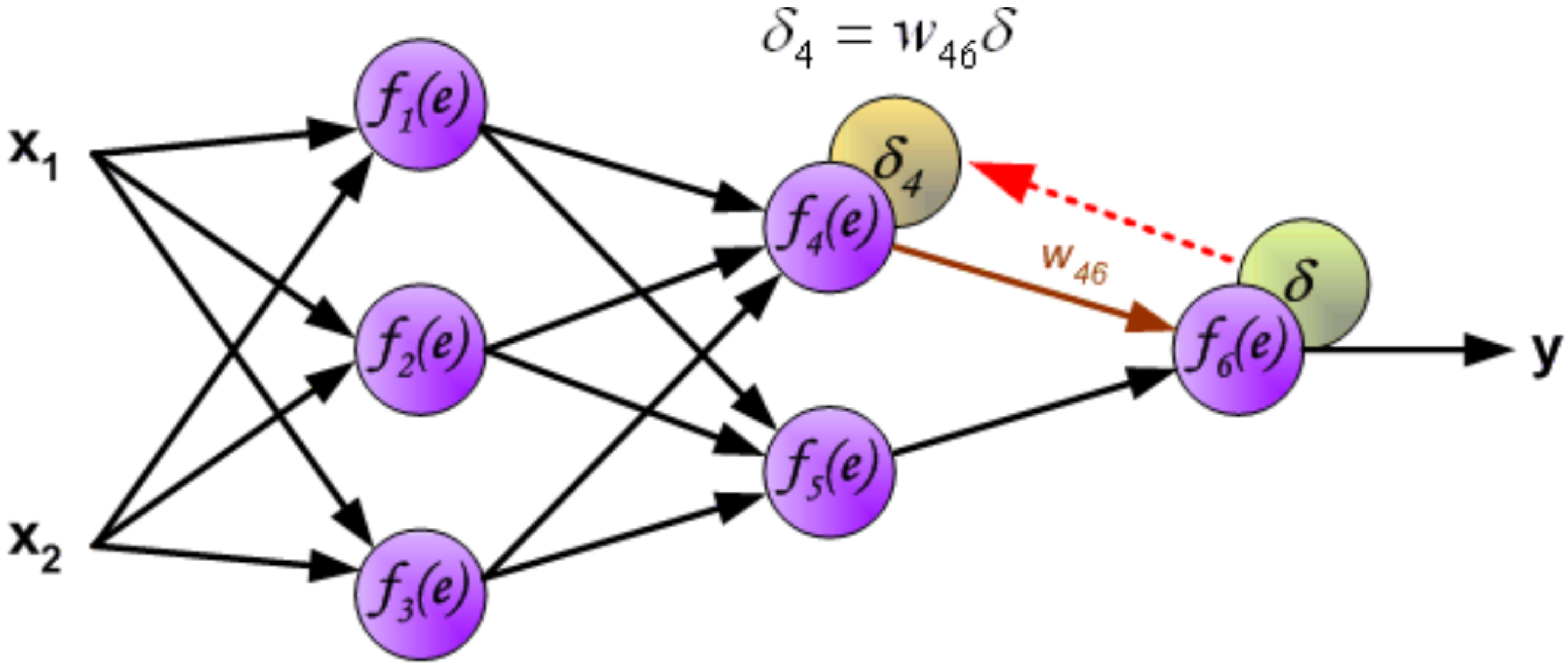
Source: [3]

Training using Backpropagation



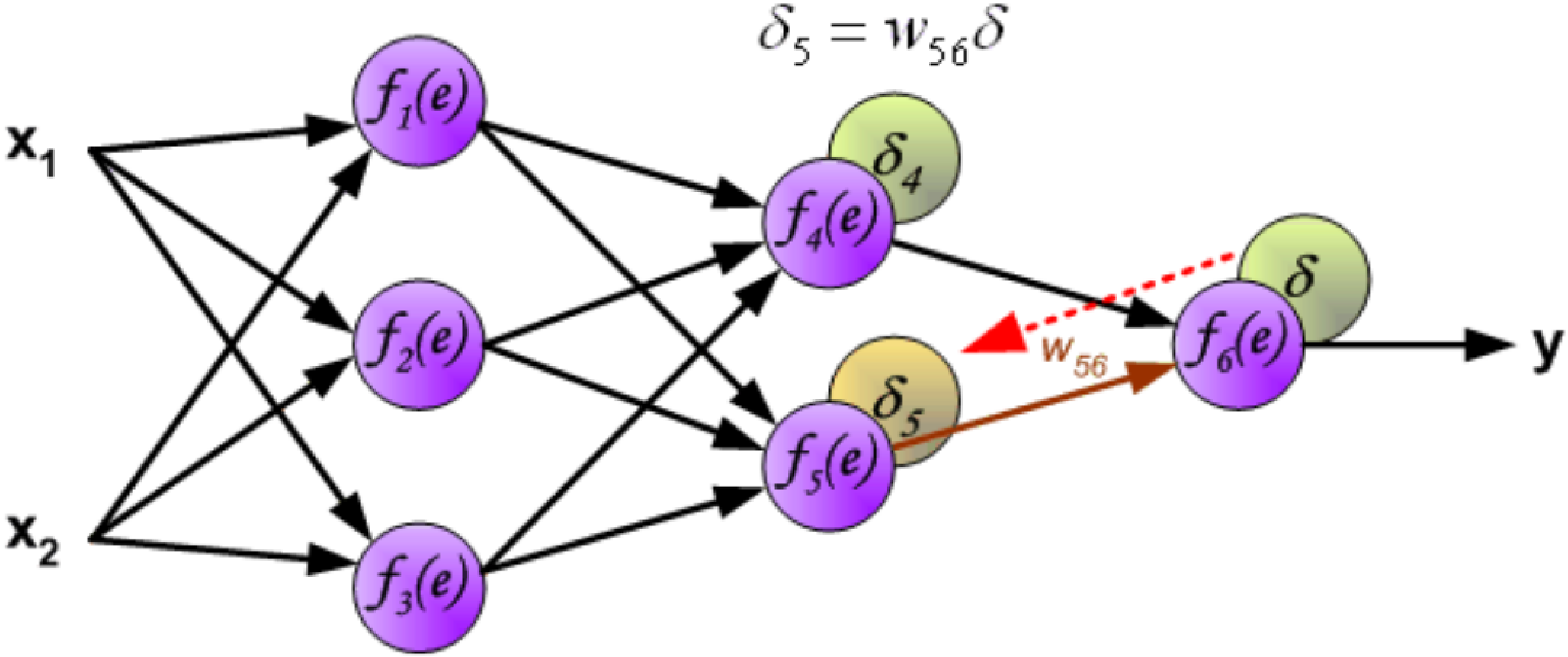
Source: [3]

Training using Backpropagation



Source: [3]

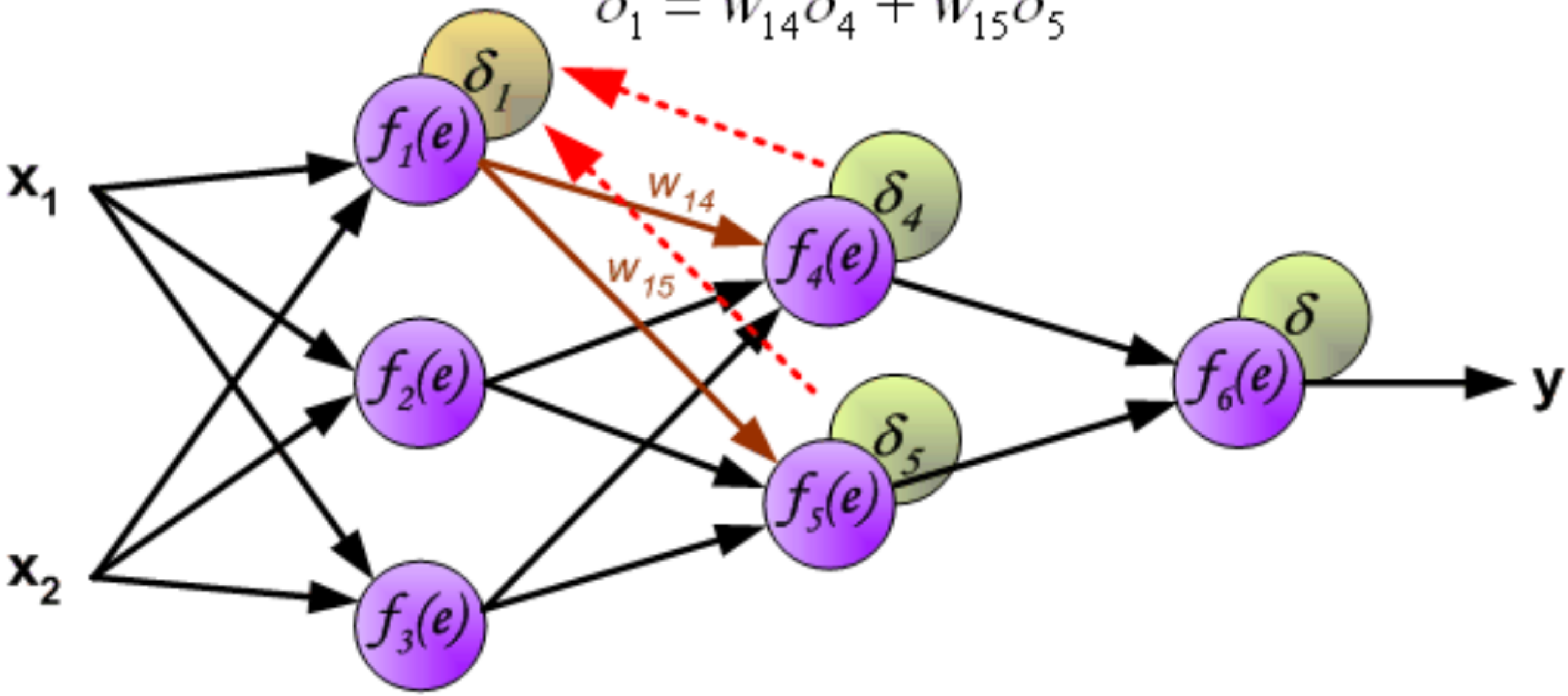
Training using Backpropagation



Source: [3]

Training using Backpropagation

$$\delta_1 = w_{14}\delta_4 + w_{15}\delta_5$$

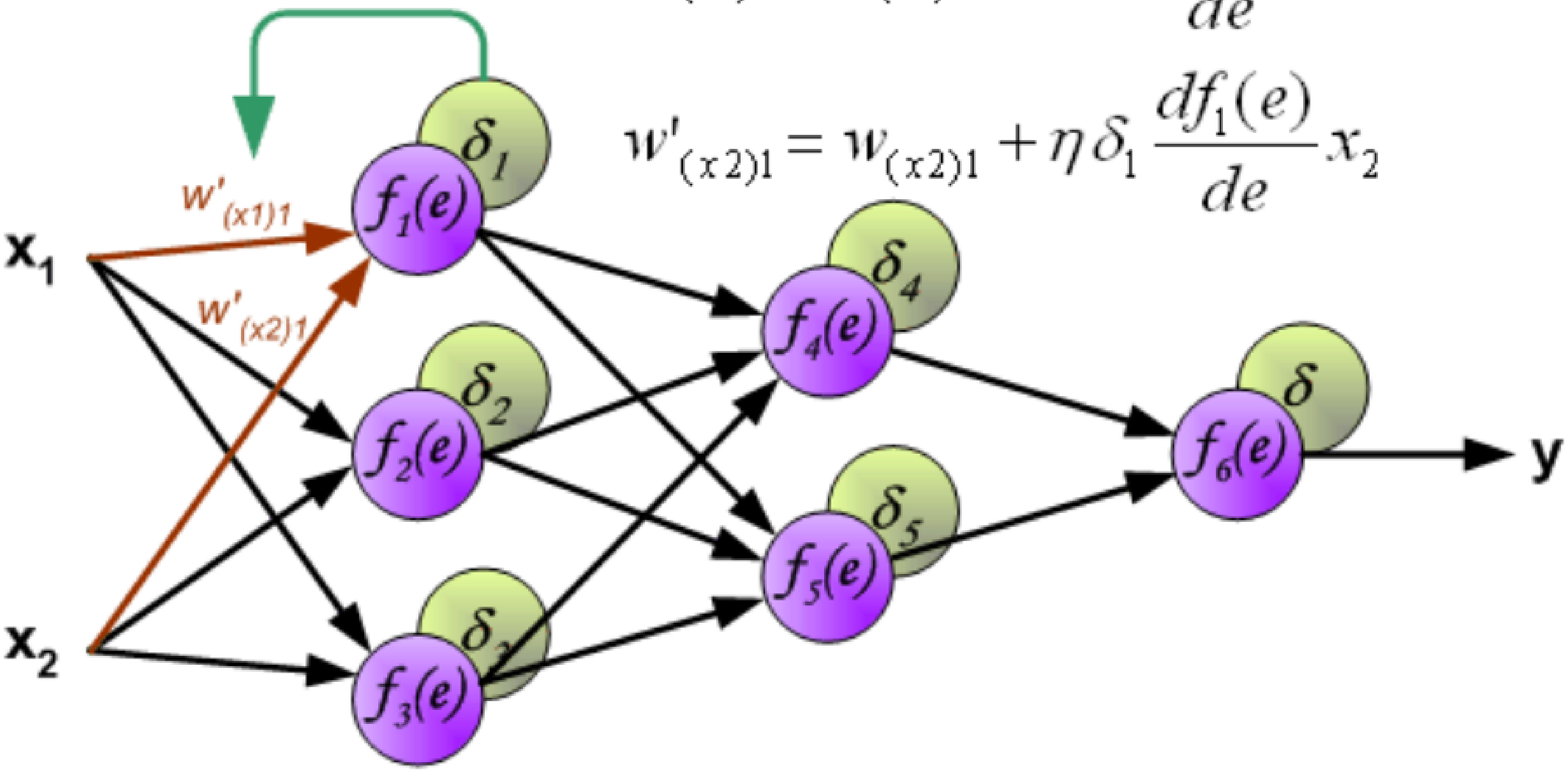


Source: [3]

Training using Backpropagation

$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$

$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$

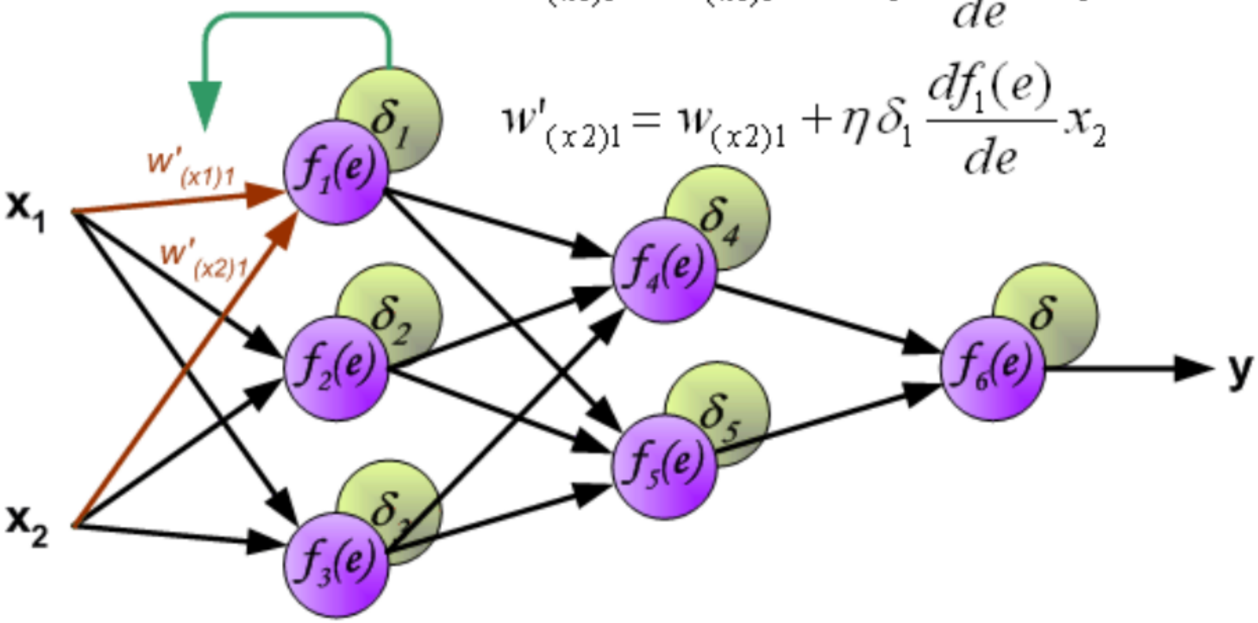


Source: [3]

Training using Backpropagation

$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$

$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$



As vector product, for all weights leading from the input x_1 to all first layer neurons:

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \end{bmatrix}^T \times \begin{bmatrix} \delta_1 \frac{df_1(e)}{de} & \delta_2 \frac{df_2(e)}{de} & \dots & \delta_n \frac{df_n(e)}{de} \end{bmatrix}$$

Source: [3]

Using matrices

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ & & & \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

Source: [6]

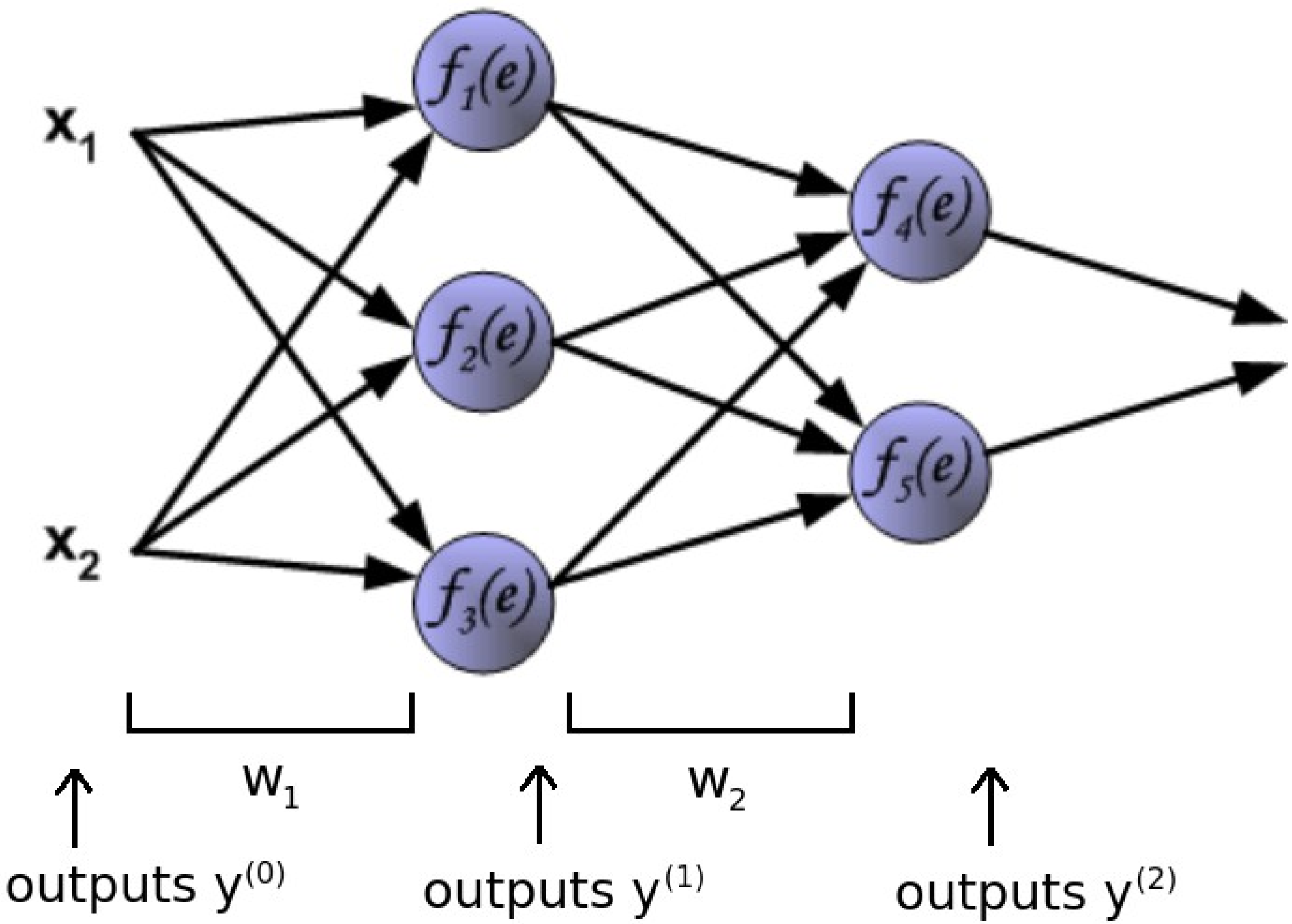
Backpropagating the error

$$E = (z - y)^2$$

$$\delta = \frac{\cancel{2}(z - y)}{\cancel{2}}$$

$$\delta = z - y$$

Computing weight corrections: 2-layer-network



Computing weight corrections: 2-layer-network

$$\delta^{(2)} = D_2 \cdot (z - y^{(2)})$$

$$\Delta w_2 = -\eta \delta^{(2)} y^{(1)}$$

$$\delta^{(1)} = D_1 w_2 \delta^{(2)}$$

$$\Delta w_1 = -\eta \delta^{(1)} y^{(0)}$$

D_i - matrix of derivatives of the activation function in the i‘th layer

w_i - matrix of weights in the i‘th layer

$y^{(i)}$ - actual output values of the i‘th layer

z - correct output values of the last layer

Computing weight corrections: 2-layer-network

$$\delta^{(2)} = D_2 \cdot (z - y^{(2)})$$

$$\Delta w_2 = -\eta \delta^{(2)} y^{(1)}$$

$$\delta^{(1)} = D_1 w_2 \delta^{(2)}$$

$$\Delta w_1 = -\eta \delta^{(1)} y^{(0)}$$

$$w_i' = w_i + \Delta w_i$$

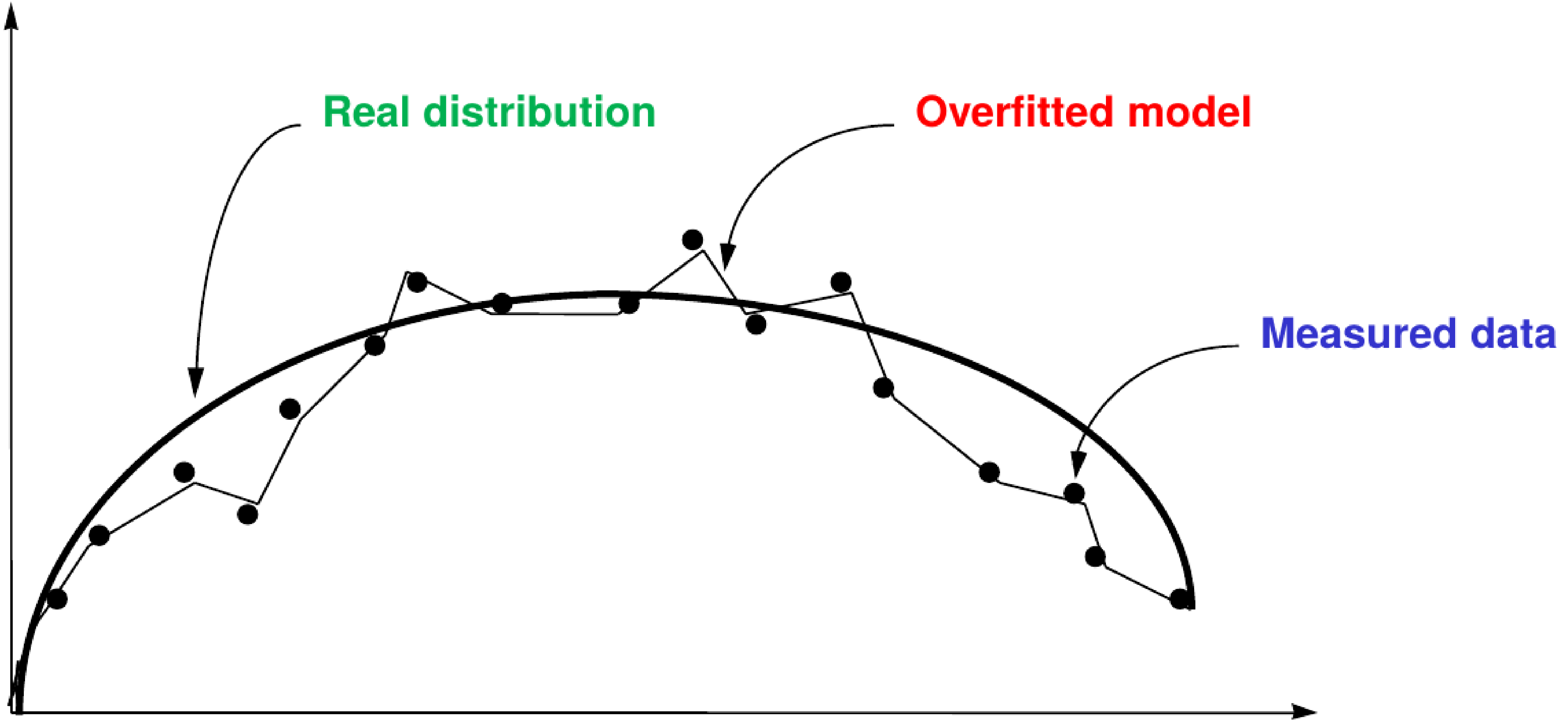
D_i - matrix of derivatives of the activation function in the i‘th layer

w_i - matrix of weights in the i‘th layer

$y^{(i)}$ - actual output values of the i‘th layer

z - correct output values of the last layer

Problem: Overfitting

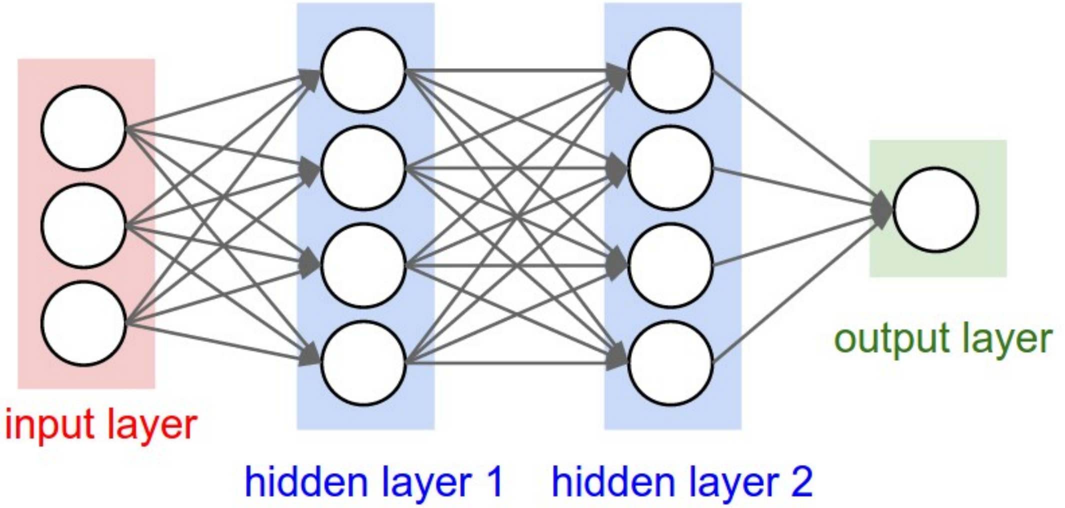


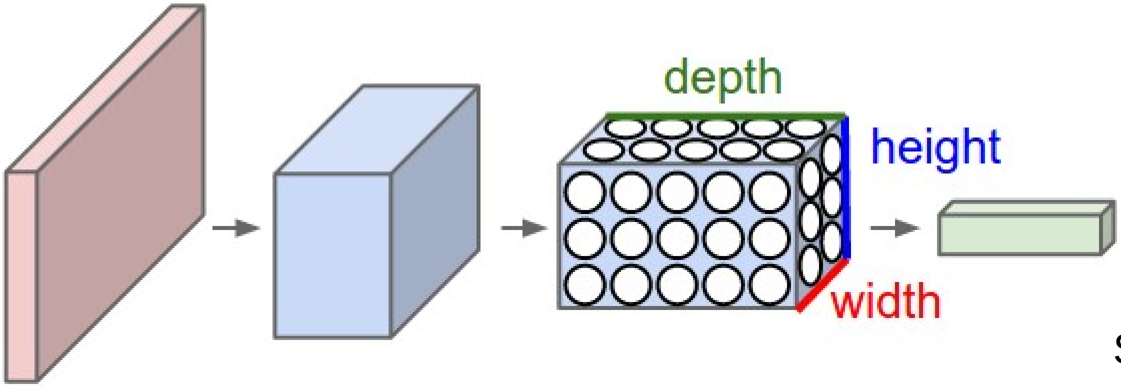
Source: [3]

Overview

1. What are neural networks?
2. A single perceptron
3. Perceptron training
4. A simple network
5. Learning via backpropagation
6. **Convolutional neural networks**
7. 2nd talk: Visualization

Convolutional Neural Networks (CNN)

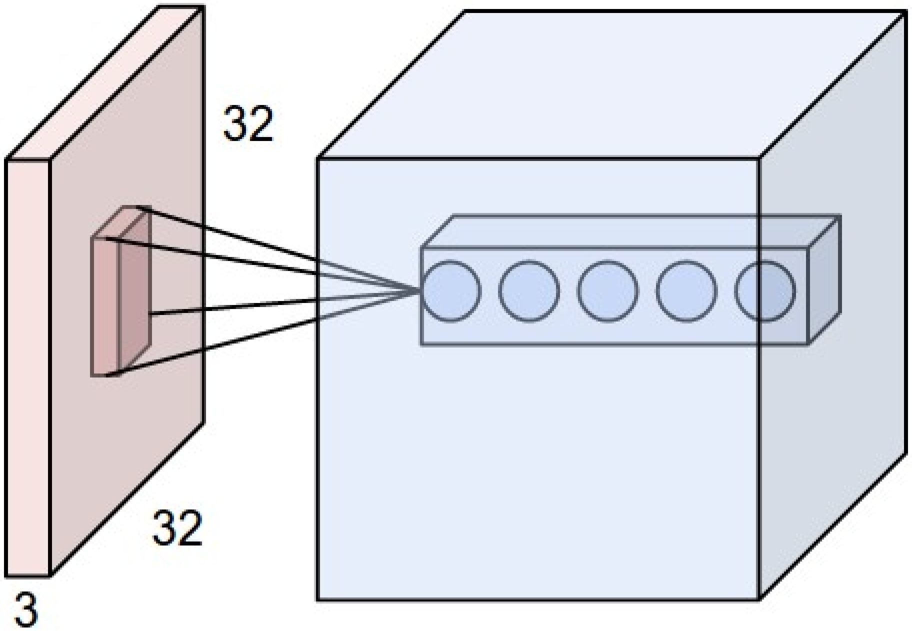




Source: [4]

Convolutional Neural Networks (CNN)

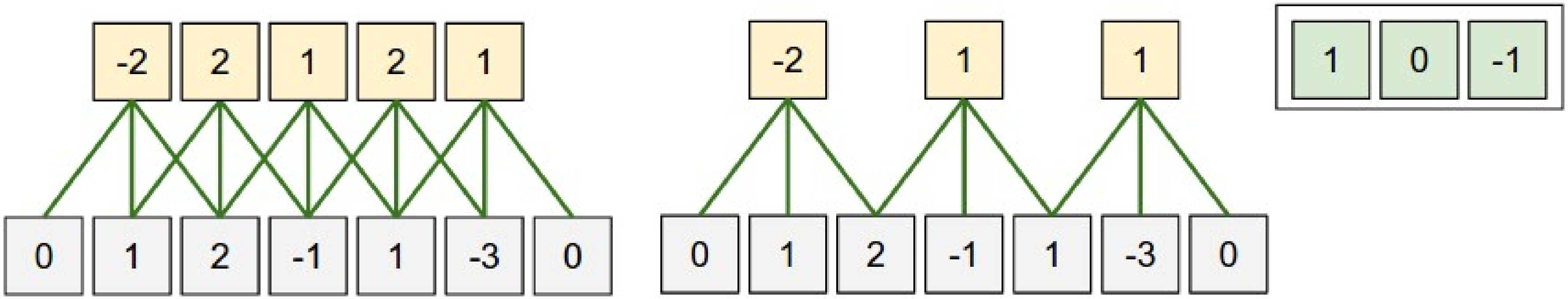
Complexity Reduction



Source: [4]

Convolutional Neural Networks (CNN)

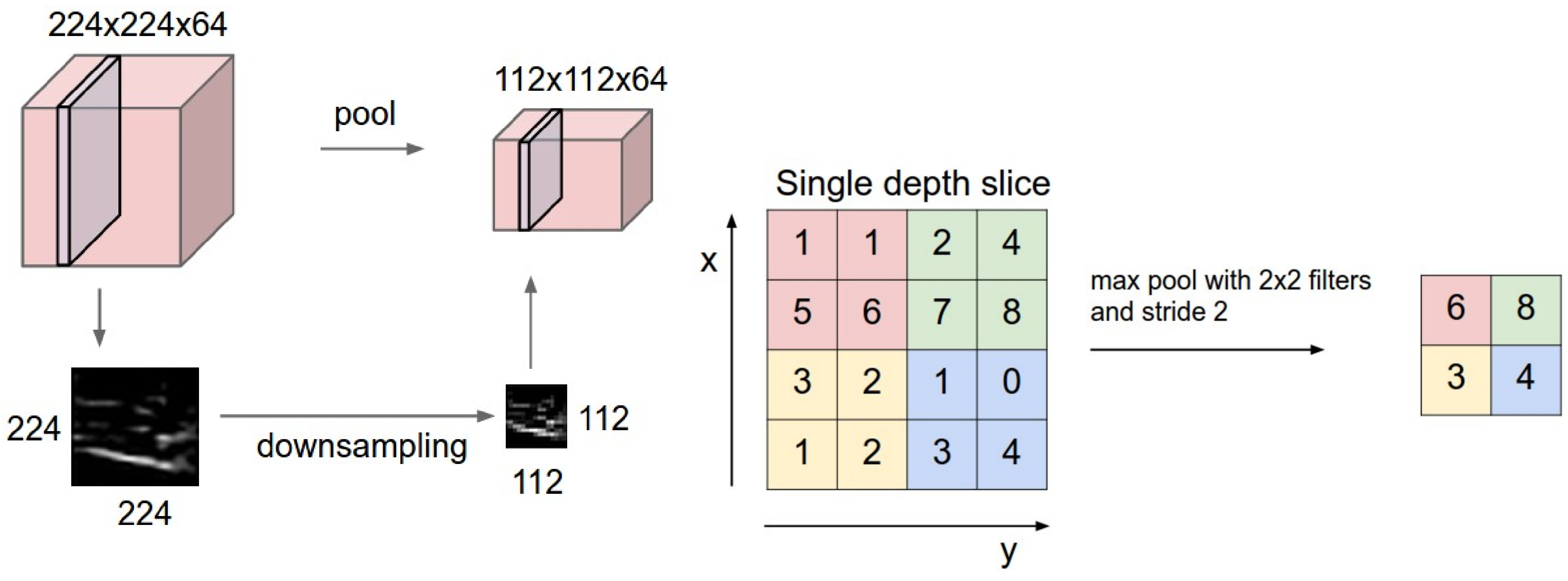
Complexity Reduction



Source: [4]

Convolutional Neural Networks (CNN)

Max-Pooling



Source: [4]

Convolutional Neural Networks (CNN)

Architecture examples

- LeNet	(1990's)
- AlexNet	(2012)
- ZF Net	(2013)
- GoogLeNet	(2014)
- VGGNet	(2014)
- Inception-v3	(2014)
- ResNet	(2015)
- DenseNet	(2017)

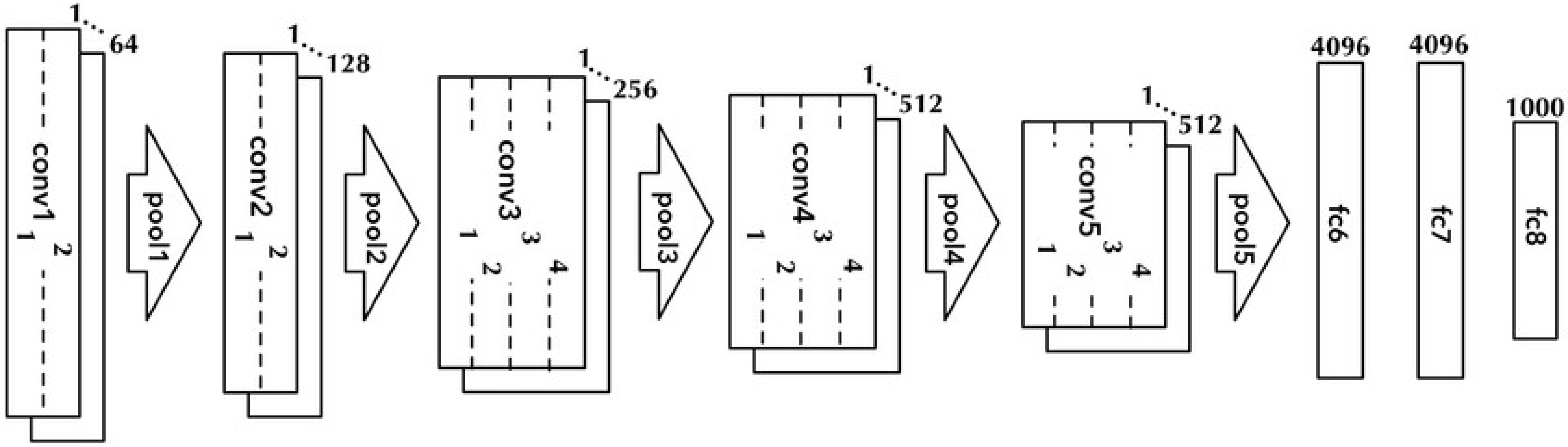
Source: [4]

Convolutional Neural Network: VGG

One of the winners of the Imagenet challenge 2014

Trained on half a million images, recognizes 1000 object classes

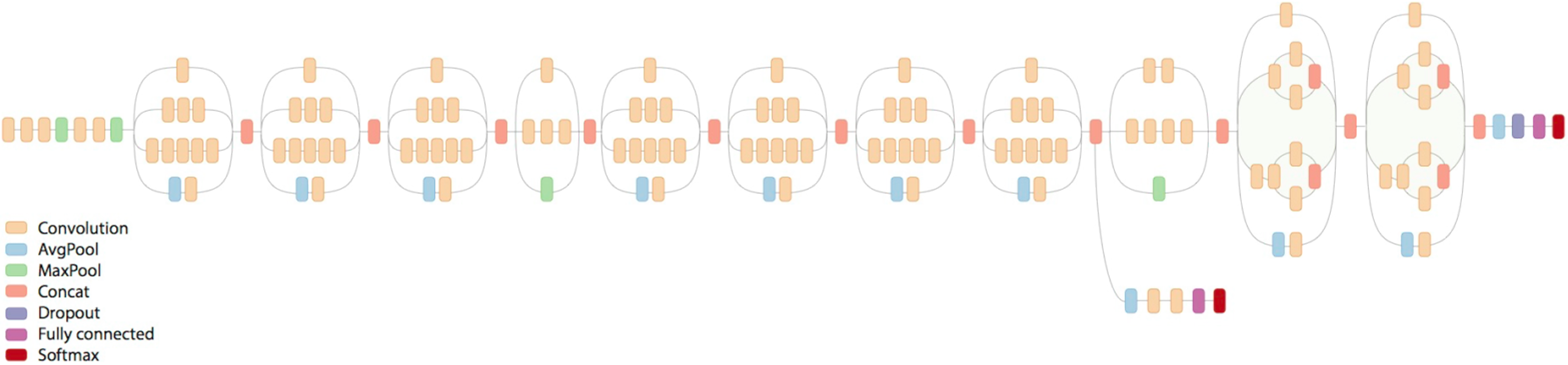
Error rate of **7.3 %**



Source: [7],[8],[9]

Convolutional Neural Network: Inception-v3

Developed in 2015, based on the winner of the ImageNet challenge 2014
trained on half a million images, recognizes 1000 object classes
Error rate of **5.64 %**



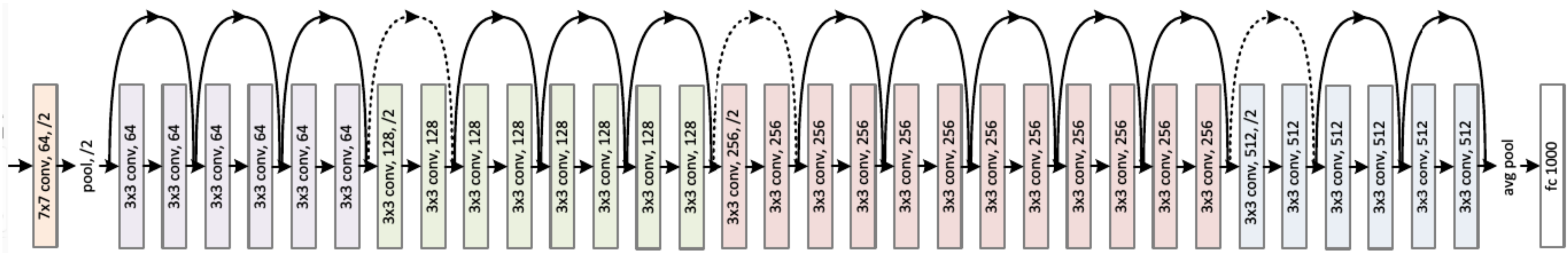
Source: [10],[11],[12],[13,[14]

Convolutional Neural Network: ResNet

Winner of the Imagenet challenge 2015

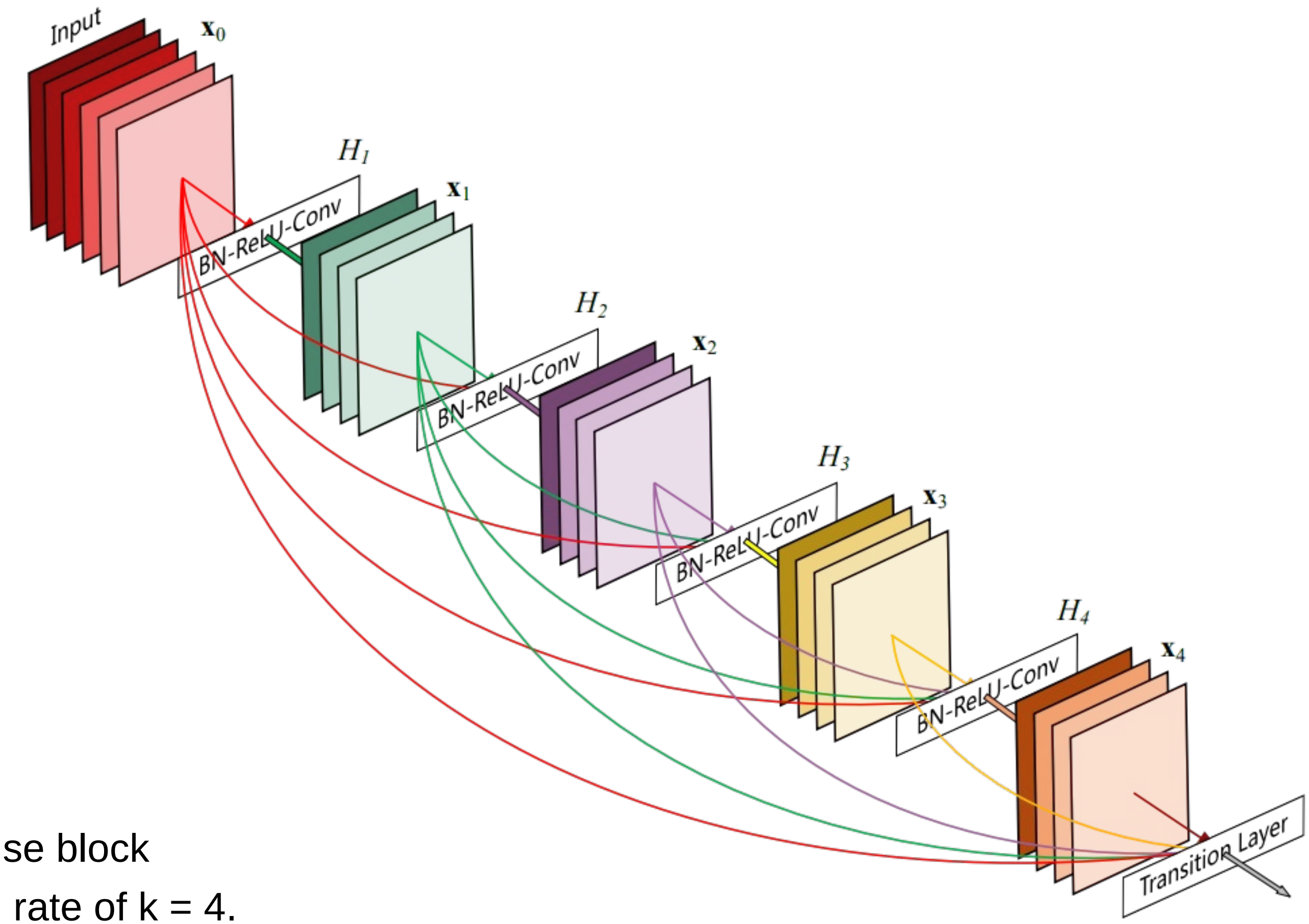
Trained on half a million images, recognizes 1000 object classes

Error rate of **3.57 %**



Source: [15],[16]

Convolutional Neural Network: DenseNet



A 5-layer dense block
with a growth rate of k = 4.
Each layer takes all preceding feature-maps as input.

Source: [17]

Overview

1. What are neural networks?
2. A single perceptron
3. Perceptron training
4. A simple network
5. Learning via backpropagation
6. Convolutional neural networks
7. **2nd talk: Visualization**

Why visualize Neural Networks?

- because it‘s fun

- … also, it helps us understand these networks better

Thanks!

Sources

- [1] Bogdan M. Wilamowski. „Neural Network Architectures and Learning“. In: Conference: Industrial Technology, 2003 IEEE International Conference on, Volume: 1
- [2] „Vertiefung: Neural Networks for Secondary Structure Prediction“. URL: http://medicalbioinformatics.de/downloads/lectures/Algorithmische_BioInformatik/WS13-14/algbioinf_ws13-14_woche9_2.pdf.
- [3] „Vertiefung: Neural Networks for Secondary Structure Prediction, Prediction Methods for Special Secondary Structures“. URL: http://medicalbioinformatics.de/downloads/lectures/Algorithmische_BioInformatik/WS13-14/algbioinf_ws13-14_woche10_1.pdf
- [4] „Convolutional Neural Networks (CNNs / ConvNets)“. URL: http://cs231n.github.io/convolutional-networks/
- [5] „Chapter 15: Visual Processing: Cortical Pathways“. URL: http://neuroscience.uth.tmc.edu/s2/chapter15.html.
- [6] „Neural Network Architectures“. URL: http://de.mathworks.com/help/nnet/ug/neural-network-architectures.html
- (Alle Weblinks zuletzt abgerufen am 18.11.2015 um 14:27 Uhr)

Sources

•[7] ImageNet Large Scale Visual Recognition Challenge 2014: http://image-net.org/challenges/LSVRC/2014/(last accessed Feb. 10th, 2017)

•[8] VGG: http://www.robots.ox.ac.uk/~vgg/research/very_deep/(last accessed Feb. 10th, 2017)

•[9] Architecture of the VGG network: https://www.researchgate.net/figure/303993422_fig1_Figure-1-Architecture-of-the-VGG-network (last accessed on Jun 27th, 2017)

•[10] ImageNet Large Scale Visual Recognition Challenge 2015: http://image-net.org/challenges/LSVRC/2015/(last accessed Feb. 10th, 2017)

•[11] Inception-v3: https://research.googleblog.com/2016/03/train-your-own-image-classifier-with.html?m=1(last accessed Feb. 10th, 2017)

•[12] More about Inception-v3: https://pseudoprofound.wordpress.com/2016/08/28/notes-on-the-tensorflow-implementation-of-inception-v3/ (last accessed Feb. 10th, 2017)

•[13] ImageNet Large Scale Visual Recognition Challenge 2014: Workshop: http://www.image-net.org/challenges/LSVRC/2014/eccv2014 (last accessed Feb. 10th, 2017)

•[14] Architecture of the Inception-v3 network: https://research.googleblog.com/2016/03/train-your-own-image-classifier-with.html (last accessed on Jun 28th, 2017)

•[15] ResNet: https://arxiv.org/abs/1512.03385 (last accessed Feb. 10th, 2017)

•[16] Architecture of the ResNet network: https://arxiv.org/pdf/1512.03385.pdf (last accessed on Jun 27th, 2017)

•[17] Densely Connected Convolutional Networks: https://arxiv.org/pdf/1608.06993.pdf (last accessed Jan 26th, 2018)