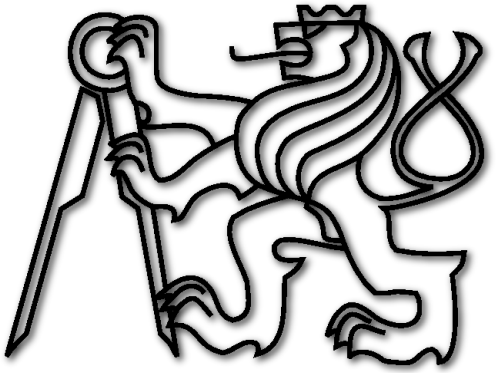
# Artificial Neural Networks
# **The Introduction**

*Jan Drchal*
drchajan@fel.cvut.cz



*Computational Intelligence Group*
*Department of Computer Science and Engineering*
*Faculty of Electrical Engineering*
*Czech Technical University in Prague*

# Information Resources

- https://cw.felk.cvut.cz/doku.php/courses/a4m33bia/start
- M. Šnorek: **Neuronové sítě a neuropočítače.** Vydavatelství ČVUT, Praha, 2002.
- **Courseware**: http://neuron.felk.cvut.cz/courseware/
- R. Rojas: **Neural Networks - A Systematic Introduction**, Springer-Verlag, Berlin, New-York, 1996, http://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/pmwiki/pmwiki.php?n=Books.NeuralNetworksBook
- J. Šíma, R. Neruda: **Theoretical Issues of Neural Networks**, MATFYZPRESS, Prague, 1996, http://www2.cs.cas.cz/~sima/kniha.html
- This presentation is based on materials for 36NAN and 336VD.

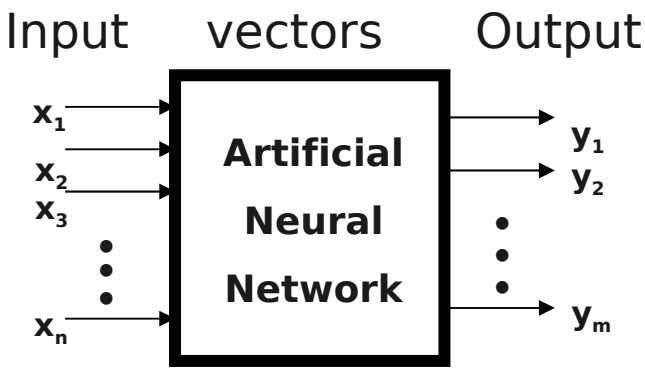
# What Are Artificial Neural Networks (ANNs)?

- Artificial information systems which imitate functions of neural systems of living organisms.
- Non-algorithmic approach to computation → learning, generalization.
- Massively-parallel processing of data using large number of simple computational units (neurons).
- Note: we are still very far from the complexity found in the Nature! But there have been some advances lately – we will see ;)

# ANN: the Black Box

- Function of an ANN can be understood as the transformation T of an input vector **X** to the output vector **Y**

$$\boldsymbol{Y} = T(\boldsymbol{X})$$




- What transformations T can be realized by neural networks ? It is the scientific question since the beginning of the discipline.

# What Can Be Solved by ANNs?

- Function approximation - regression.
- Classification (pattern/sequence recognition).
- Time series prediction.
- Fitness prediction (for optimization).
- Control (i.e. robotics).
- Association.
- Filtering, clustering, compression.

# ANN Learning & Recall

- ANNs work most frequently in two phases:
- **Learning phase** – adaptation of ANN's internal parameters.
- **Evaluation phase (recall)** – use what was learned.

# Learning Paradigms

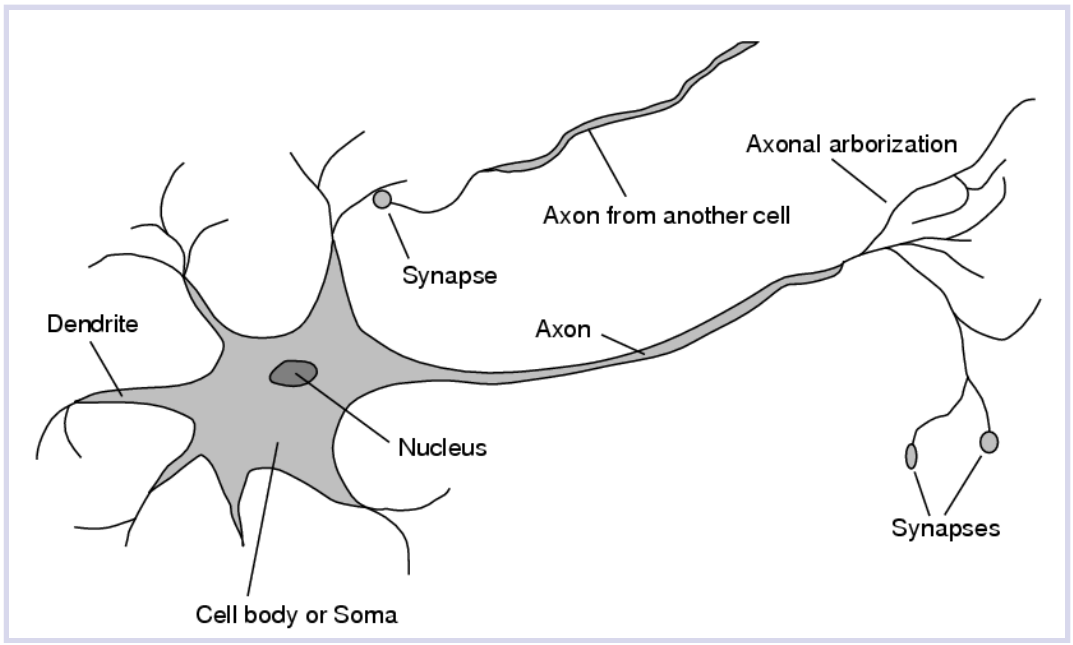
- **Supervised** – teaching by examples, given a set of example pairs $P_i=(x_i,y_i)$ find transformation **T** which approximates $y_i=\boldsymbol{T}(x_i)$ for all $i$.
- **Unsupervised** – self-organization, no teacher.
- **Reinforcement** – Teaching examples not availiable → they are generated by interactions with the environment (mostly control tasks).

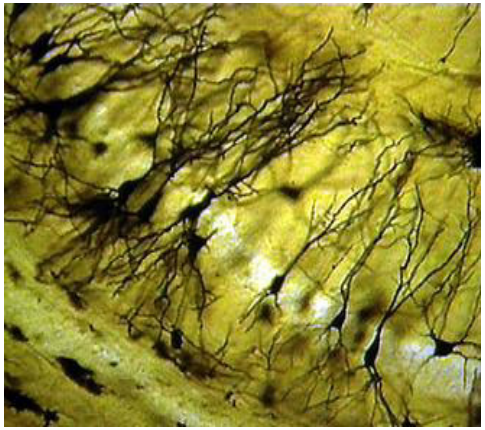
# When to Stop Learning Process?

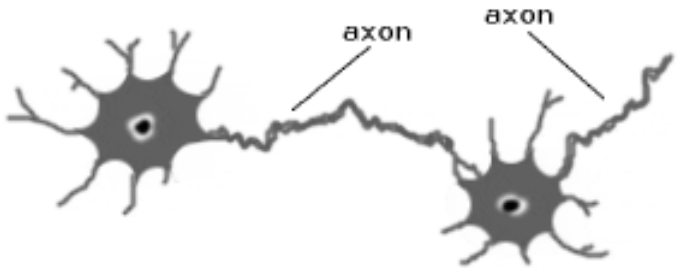
- When for all patterns of a **training set** a given criterion is met:
  - the ANN's error is less than...,
  - stagnation for given number of **epochs**.

- **Training set** – a subset of example pattern set dedicated to learning phase.
- **Epoch** – application of all training set patterns.

# Inspiration: Biological Neurons

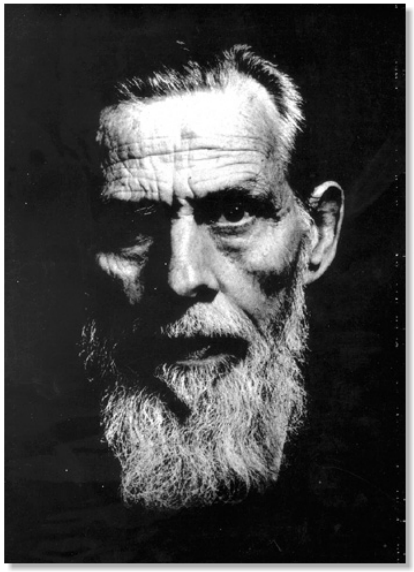




# Neuron Model: History

- Warren McCulloch,  Walter Pitts

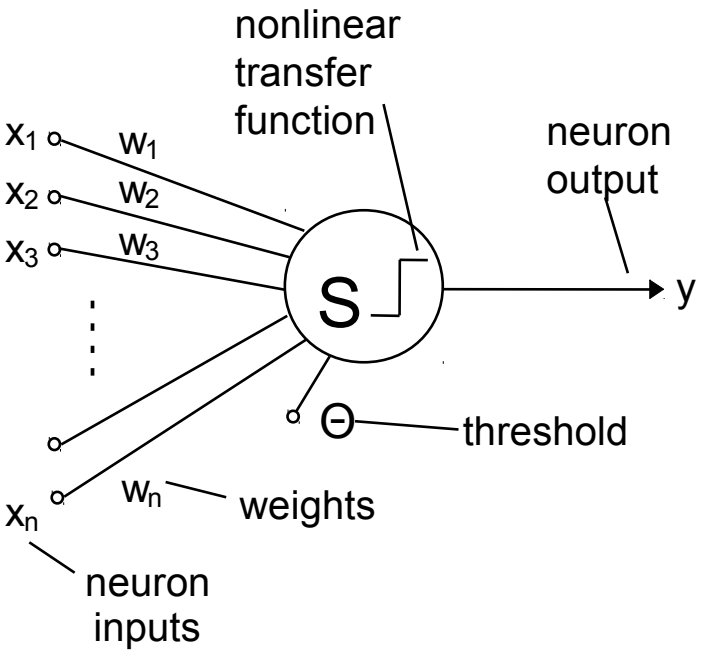


1899 - 1969

1923 - 1969

McCulloch, W. S. and Pitts, W. H. (1943).
**A logical calculus of the ideas immanent in nervous activity.**
Bulletin of Mathematical Biophysics, 5:115-133.

# McCulloch-Pitts (MCP) Neuron

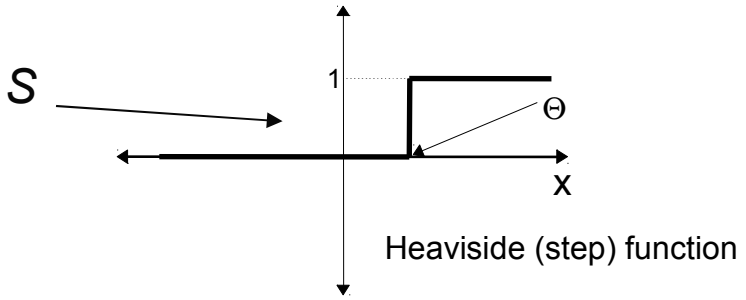



- $y$ neuron's **output (activity)**
- $x_i$ neuron's $i$-th **input** out of total $N$,
- $w_i$ $i$-th synaptic **weight**,
- S (nonlinear) **transfer (activation) function**,
- Θ **threshold, bias** (performs shift).
- The expression in brackets is **the inner potential**.
- They worked with binary inputs only.
- No learning algorithm.

$$y = S\left(\sum_{i=1}^{N} w_i x_i + \Theta\right)$$

OR

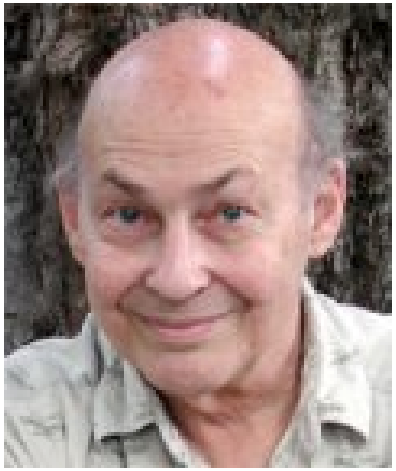
$$y = S\left(\sum_{i=0}^{N} w_i x_i\right)$$ , where $w_0 = \Theta$ and $x_0 = 1$

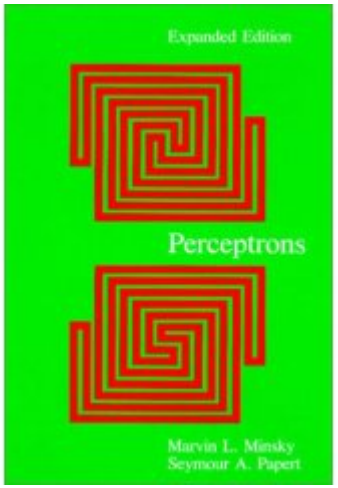



Heaviside (step) function

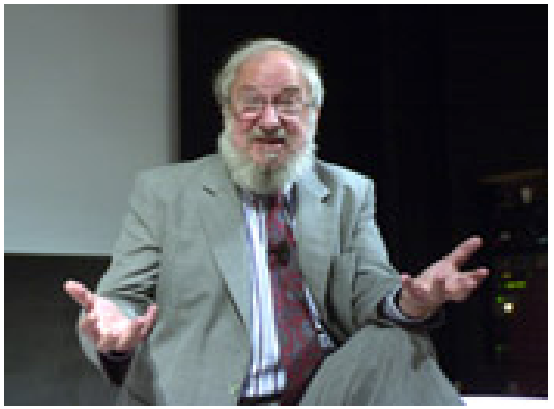
# MP-Perceptron

- MP = Marvin Minsky a Seymour Papert,
  - MIT Research Laboratory of Electronics,
  - In 1969 they published:

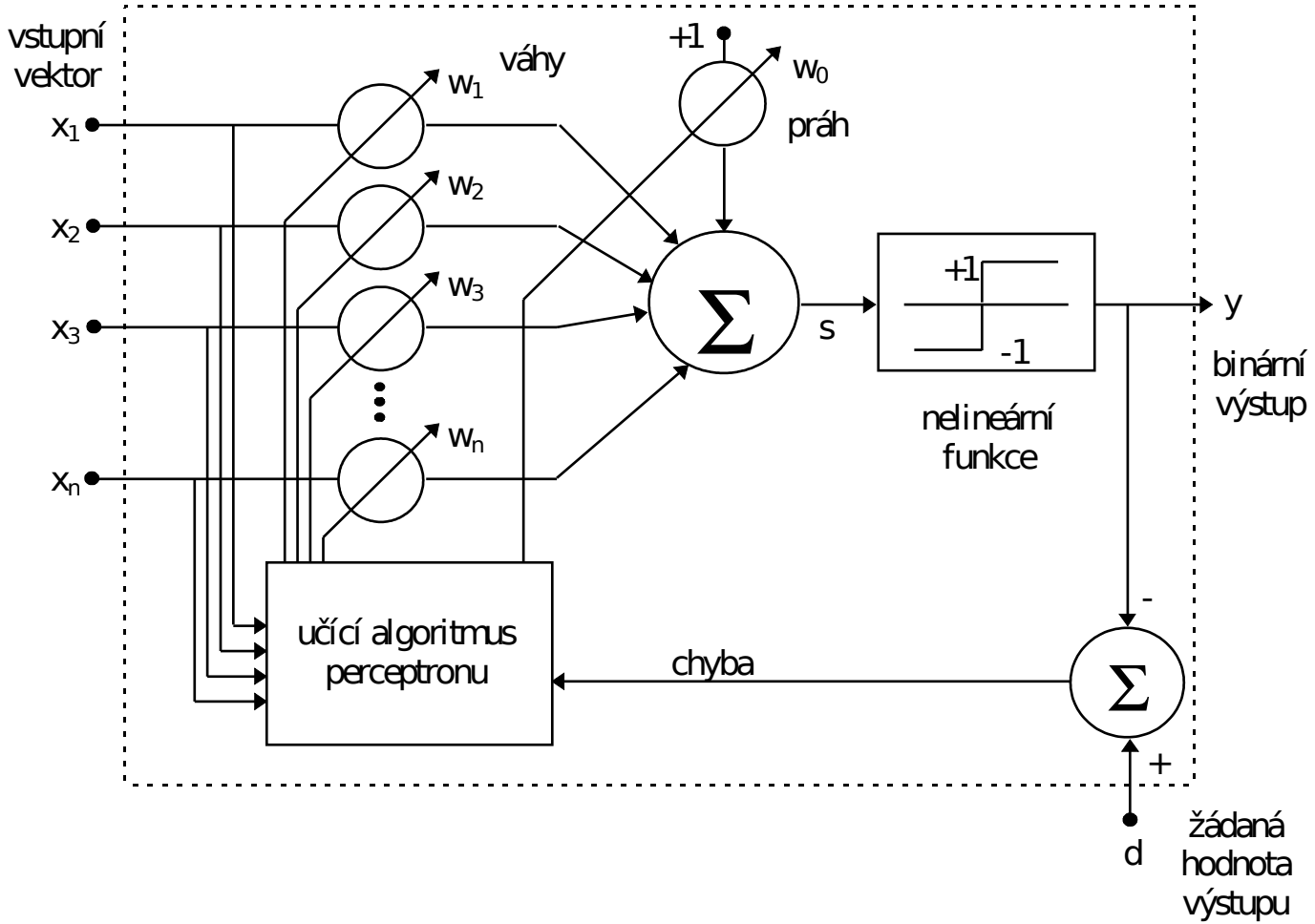





**Perceptrons**



# MP-Perceptron



# MP-Perceptron: Learning

For each example pattern, modify perceptron weights:

$$w_i(t+1) = w_i(t) + \eta\, e(t)\, x_i(t)$$

$$e(t) = d(t) - y(t)$$

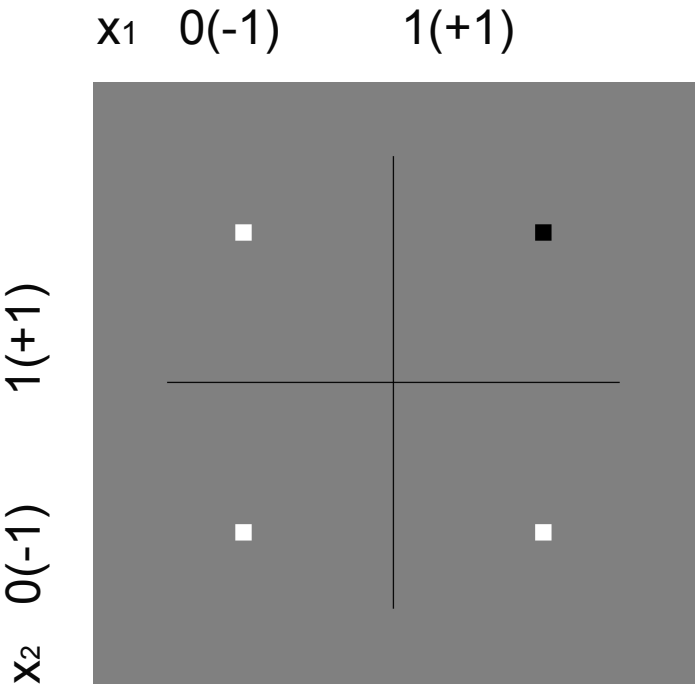
error

where

| | | | |
|---|---|---|---|
| $w_i$ | weight of i-th input | $x_i$ | value of i-th input |
| $\eta$ | learning rate <0;1) | $y$ | neuron output |
| | | $d$ | desired output |

# Demonstration

- Learning AND...




white square 0, black 1

0 encoded as -1, 1 as +1

-1 AND -1 = false

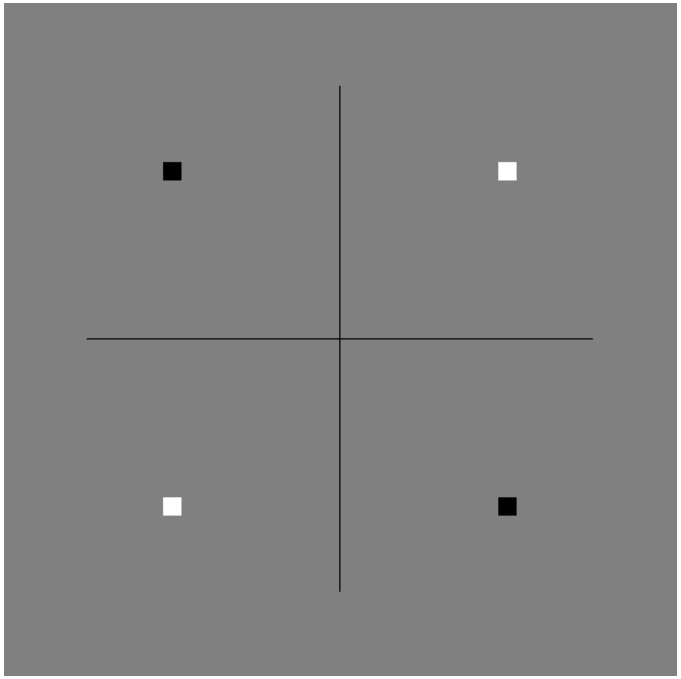
-1 AND +1 = false

+1 AND -1 = false

+1 AND +1 = true

# Minsky-Papert's Blunder 1/2

- Both have exactly proved, that this ANN (understand: perceptron) is not convenient even for implementation of so simple logic function as XOR.



-1 XOR -1 = *false*

-1 XOR +1 = *true*

+1 XOR -1 = *true*

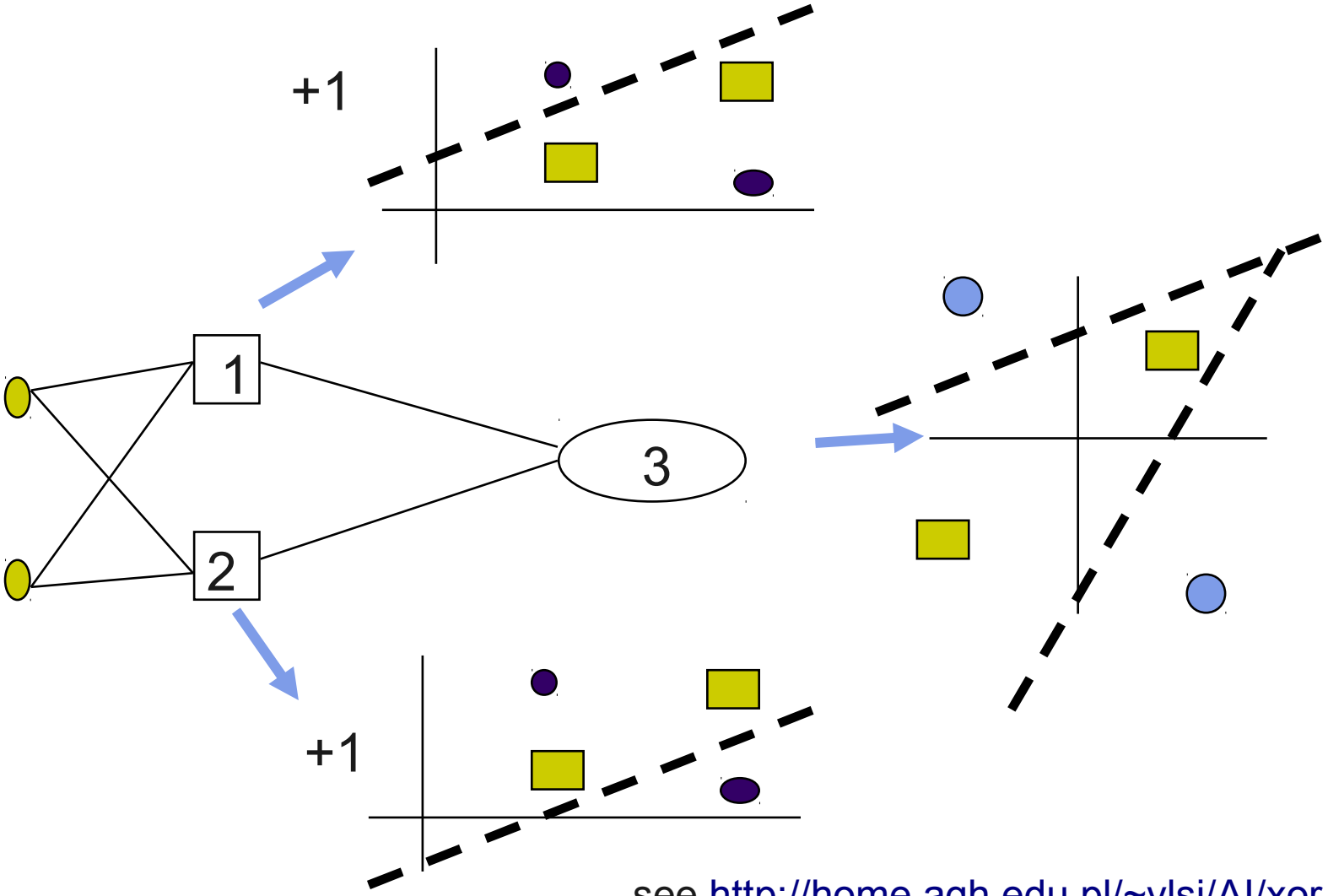
+1 XOR +1 = *false*

Impossible to separate
the classes by a single line!

# Minsky-Pappert's Blunder 2/2

- They have previous correct conclusion incorrectly generalized for all ANNs. The history showed, that they are responsible for more than two decades delay in the ANN investigations.
- **Linear non-separability!**

# XOR Solution - Add a Layer

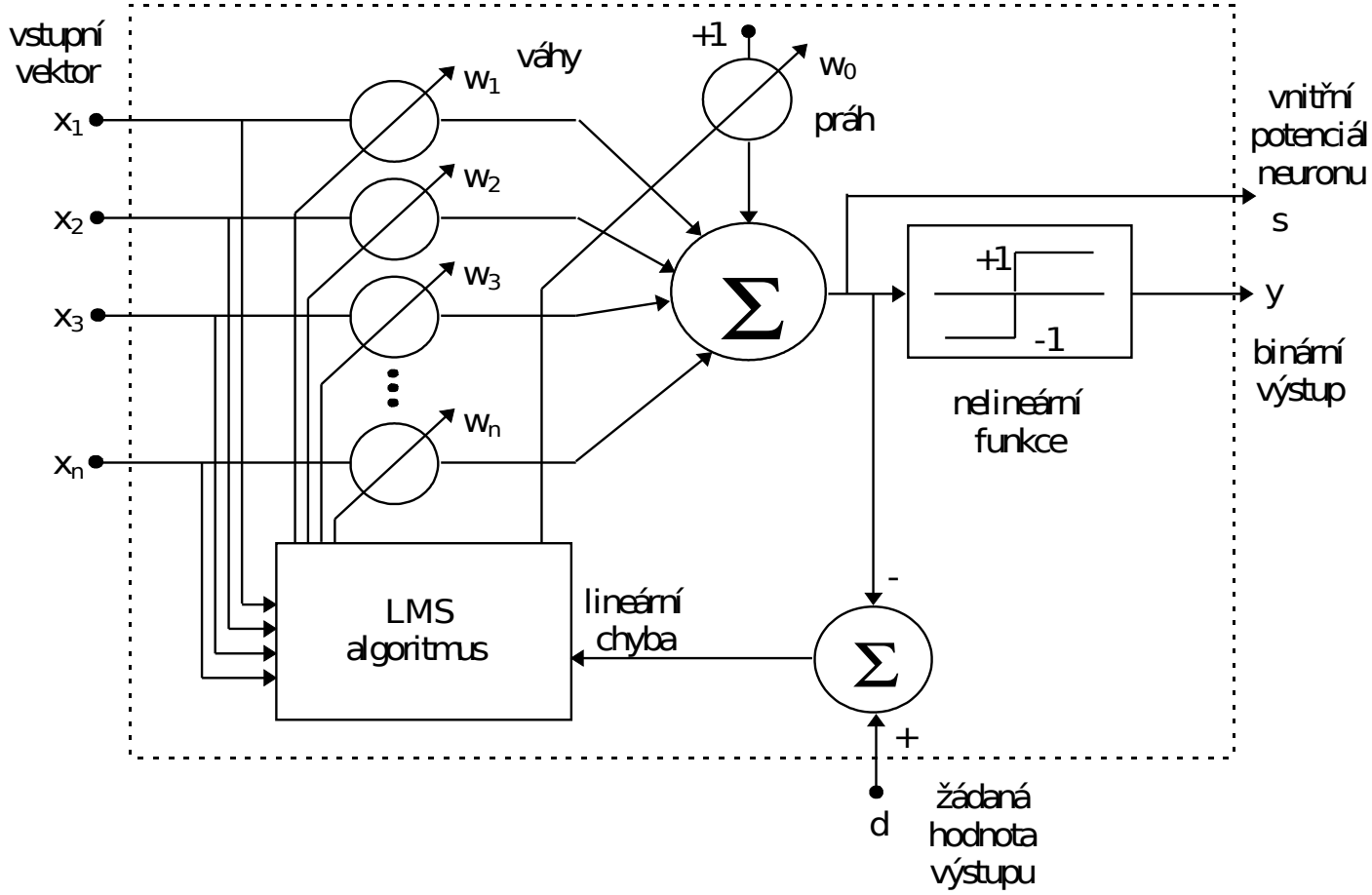



see http://home.agh.edu.pl/~vlsi/AI/xor_t/en/main.htm

# Multi-Class Classification? 1-of-N Encoding

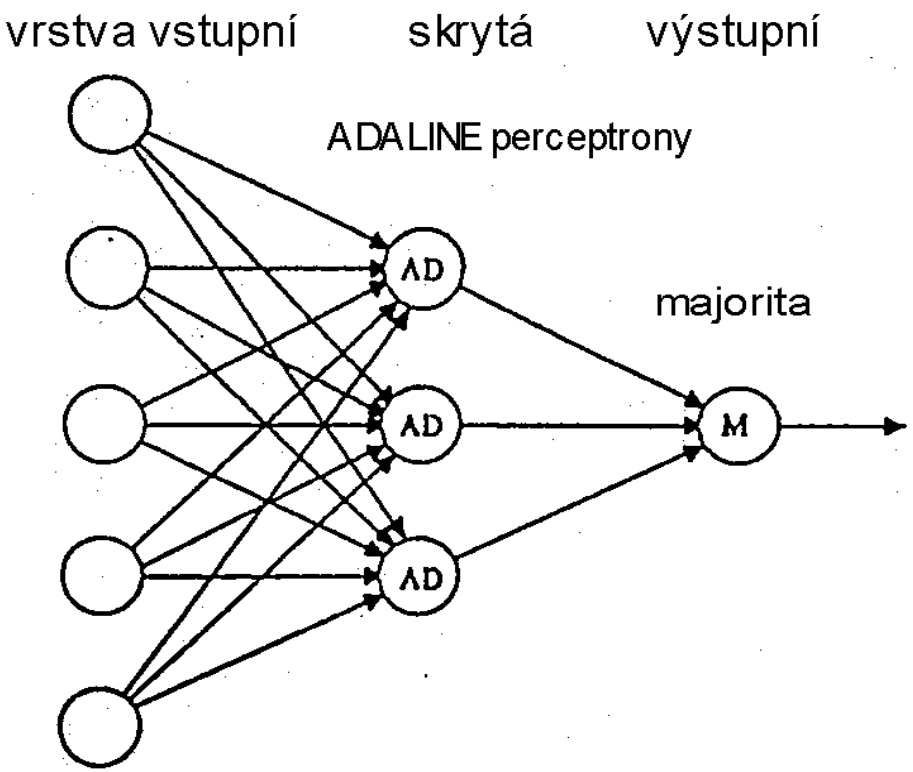
- Perceptron can only handle two-class outputs. What can be done?
- Answer: for N-class problems, learn N perceptrons:
  - *Perceptron 1 learns "Output=1" vs "Output ≠ 1"*
  - *Perceptron 2 learns "Output=2" vs "Output ≠ 2"*
  - *:*
  - *Perceptron N learns "Output=N" vs "Output ≠ N"*
- To classify an output for a new input, just predict with each perceptron and find out which one puts the prediction the furthest into the positive region.

# ADALINE - **AdA**ptive **Li**near **Ne**uron




http://www.learnartificialneuralnetworks.com/perceptronadaline.html

# MADALINE: Architecture




Note: We leanr only weights at ADALINE inputs.