Artificial Neural Networks: Lecture 1

Simple Perceptrons for Classification

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Objectives for today:

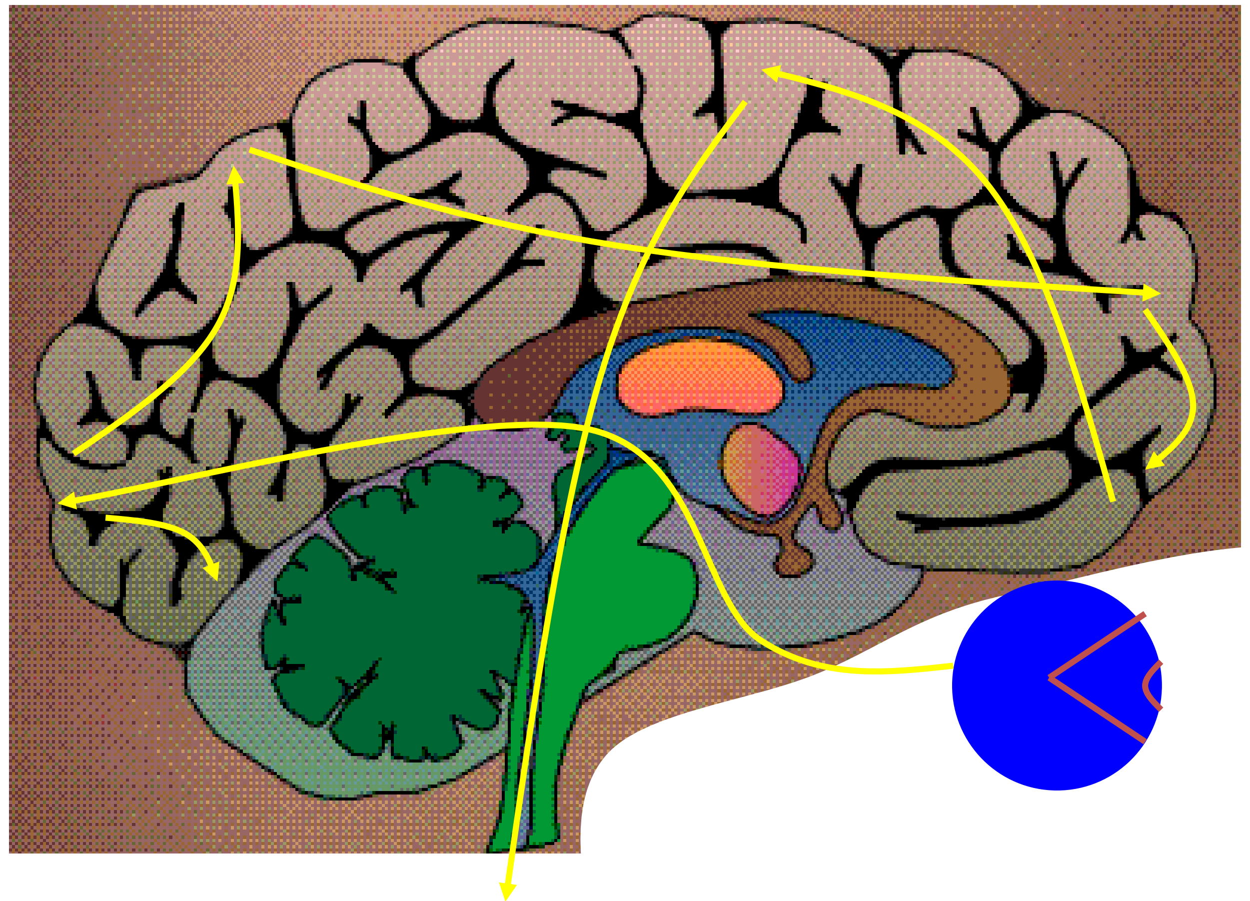
- understand classification as a geometrical problem
- discriminant function of classification
- linear versus nonlinear discriminant function
- perceptron algorithm
- gradient descent for simple perceptrons

The brain: Cortical Areas

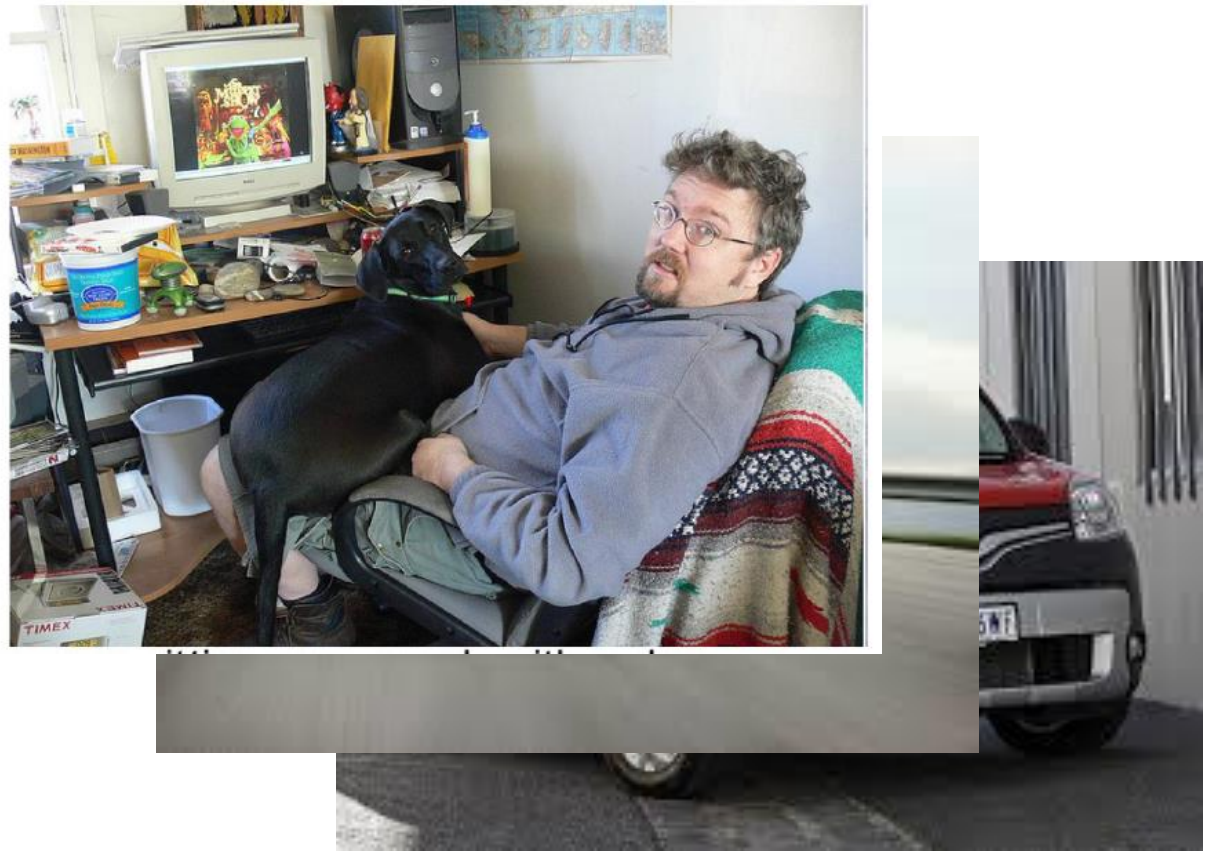
motor
cortex

frontal cortex

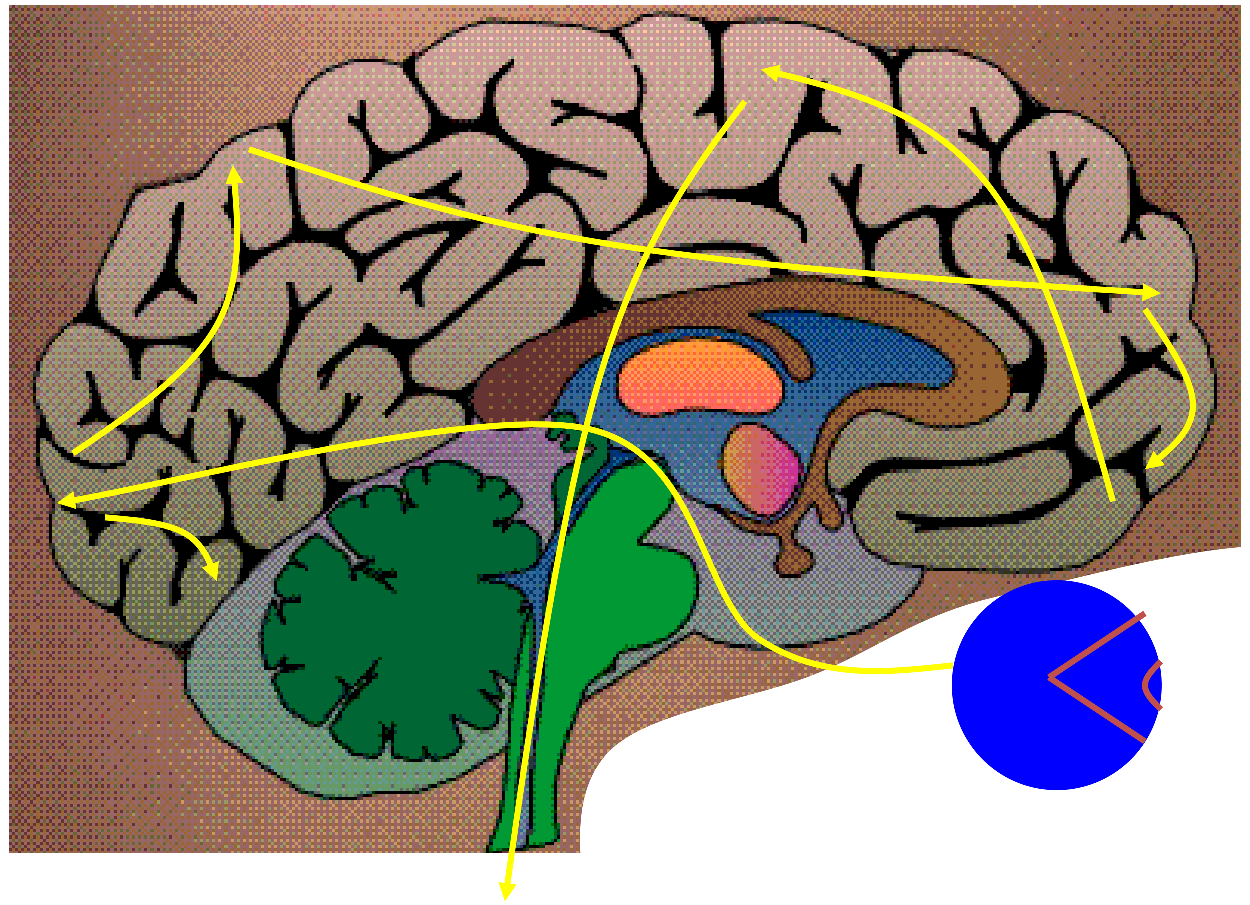
visual
cortex



to
muscles

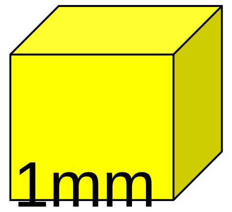


The brain: Cortical Areas



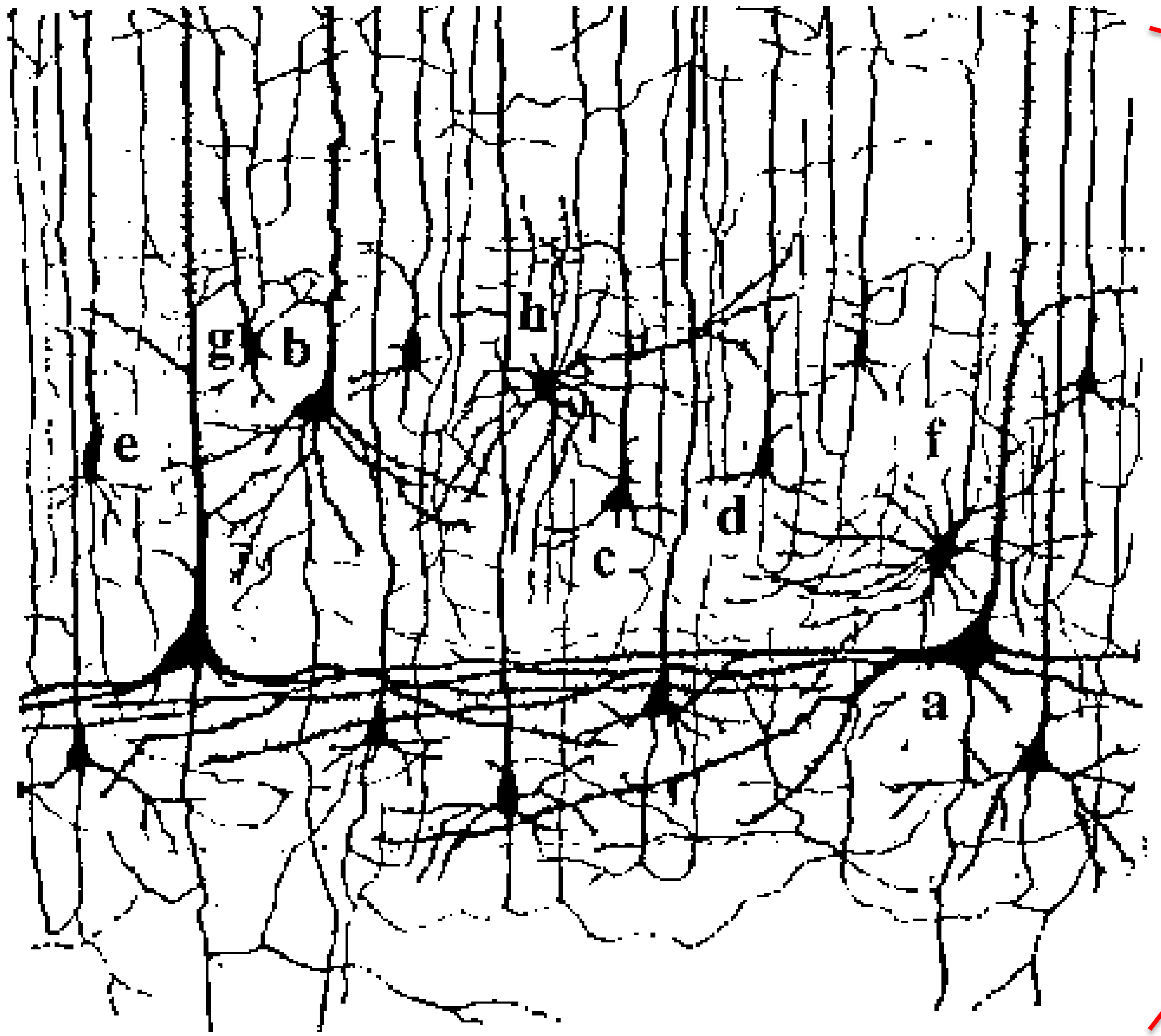
The Brain: zooming in

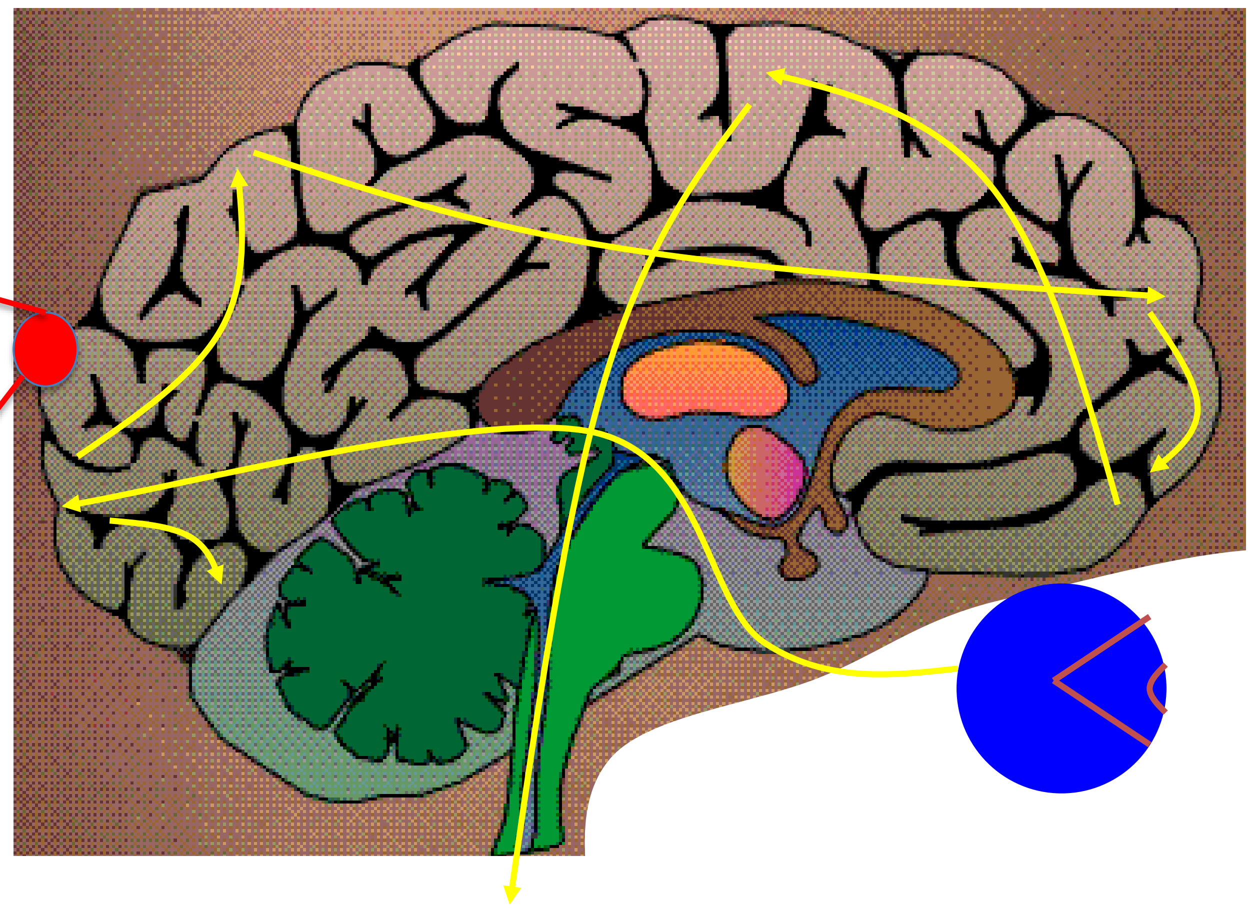
1mm



10 000 neurons

3 km of wire

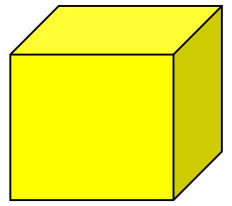




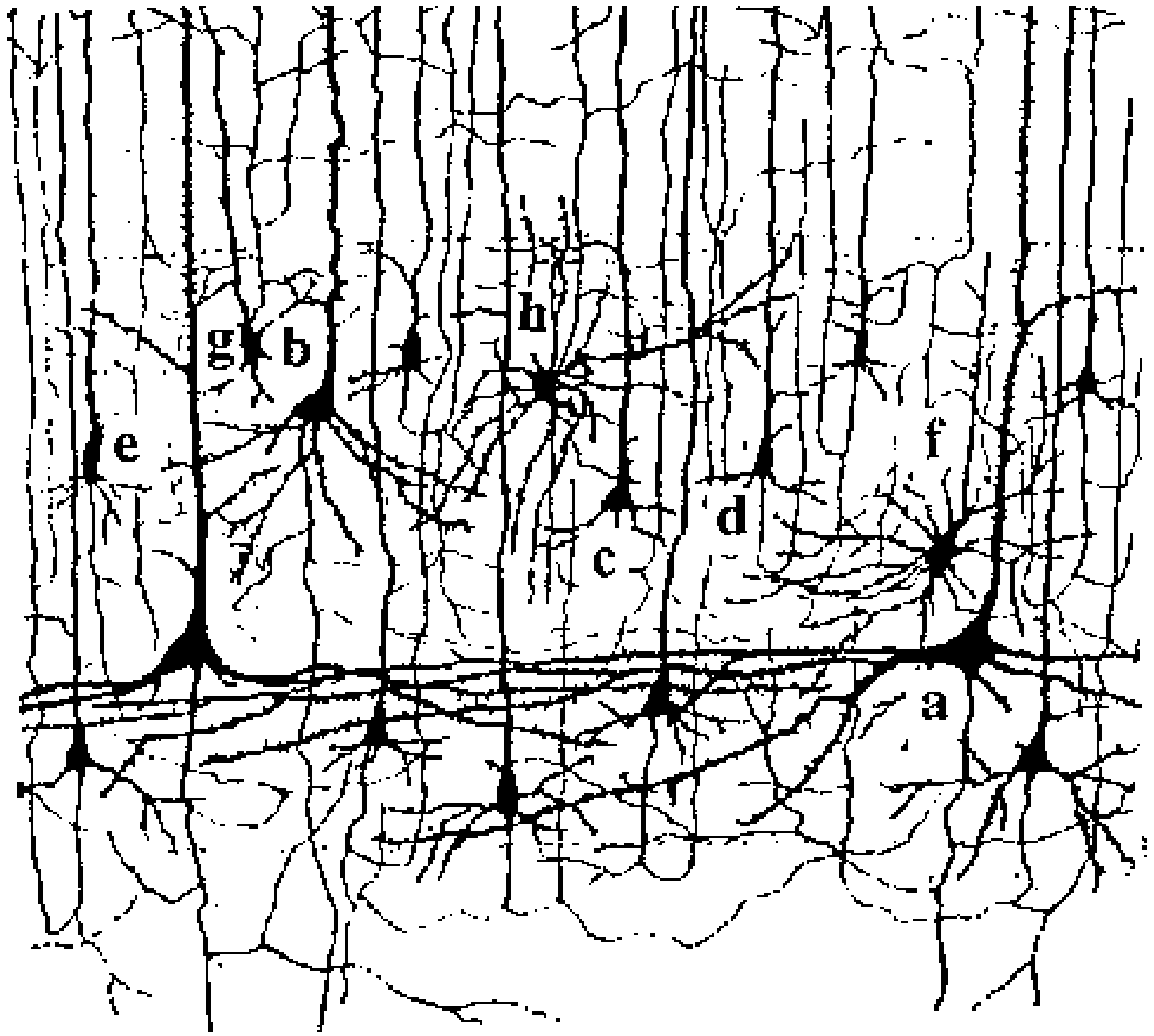
Ramon y Cajal

The brain: a network of neurons

1mm

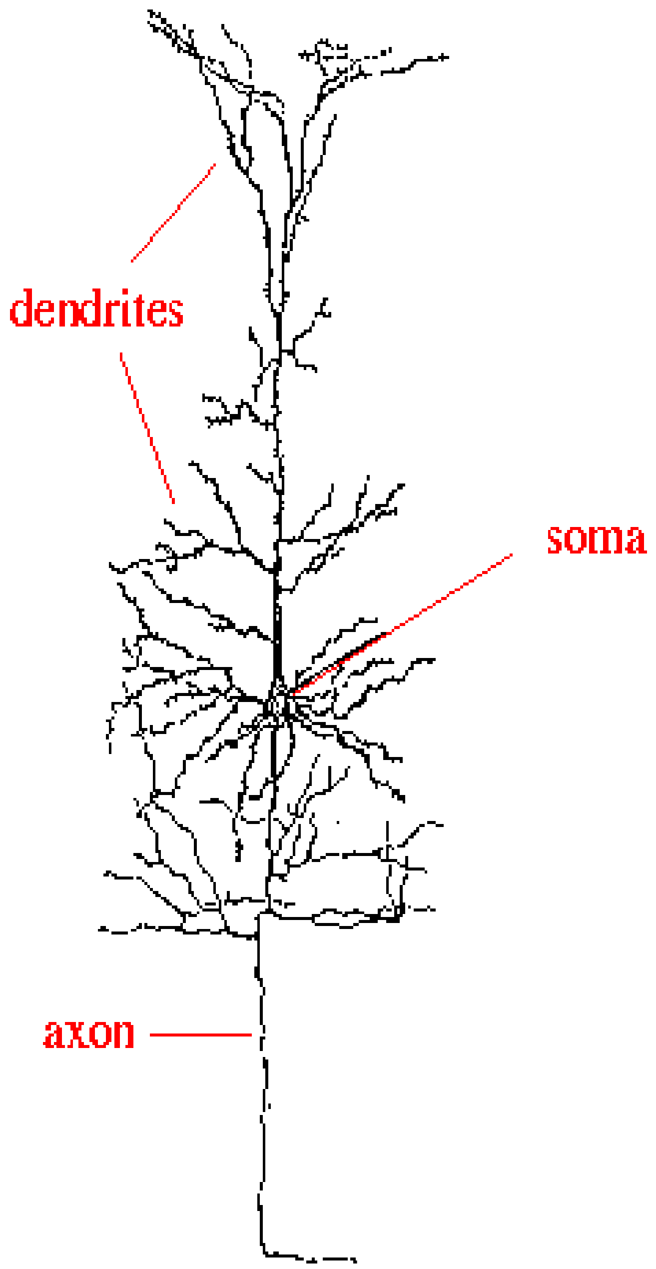


10 000 neurons
3km of wire

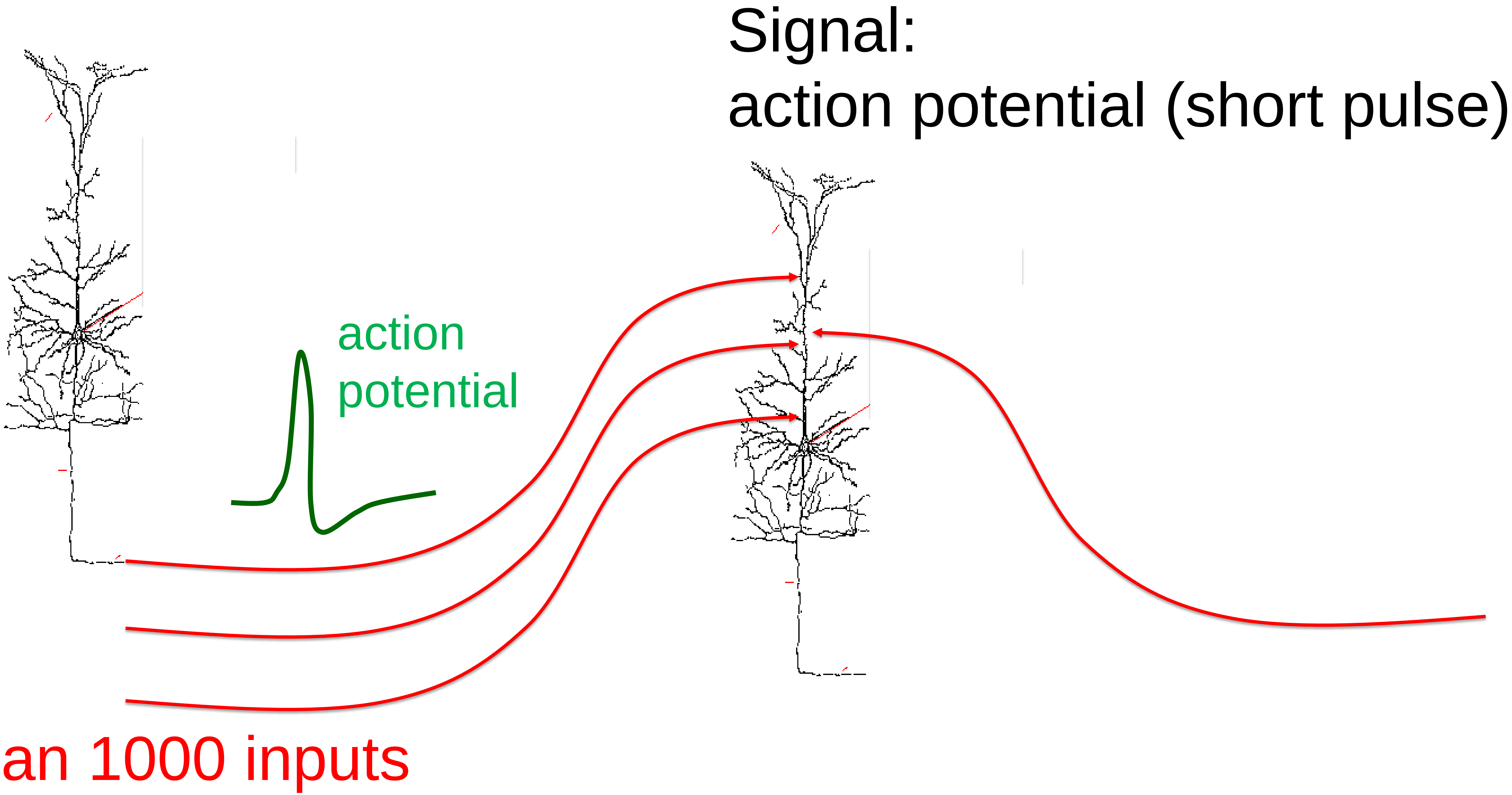


Ramon y Cajal

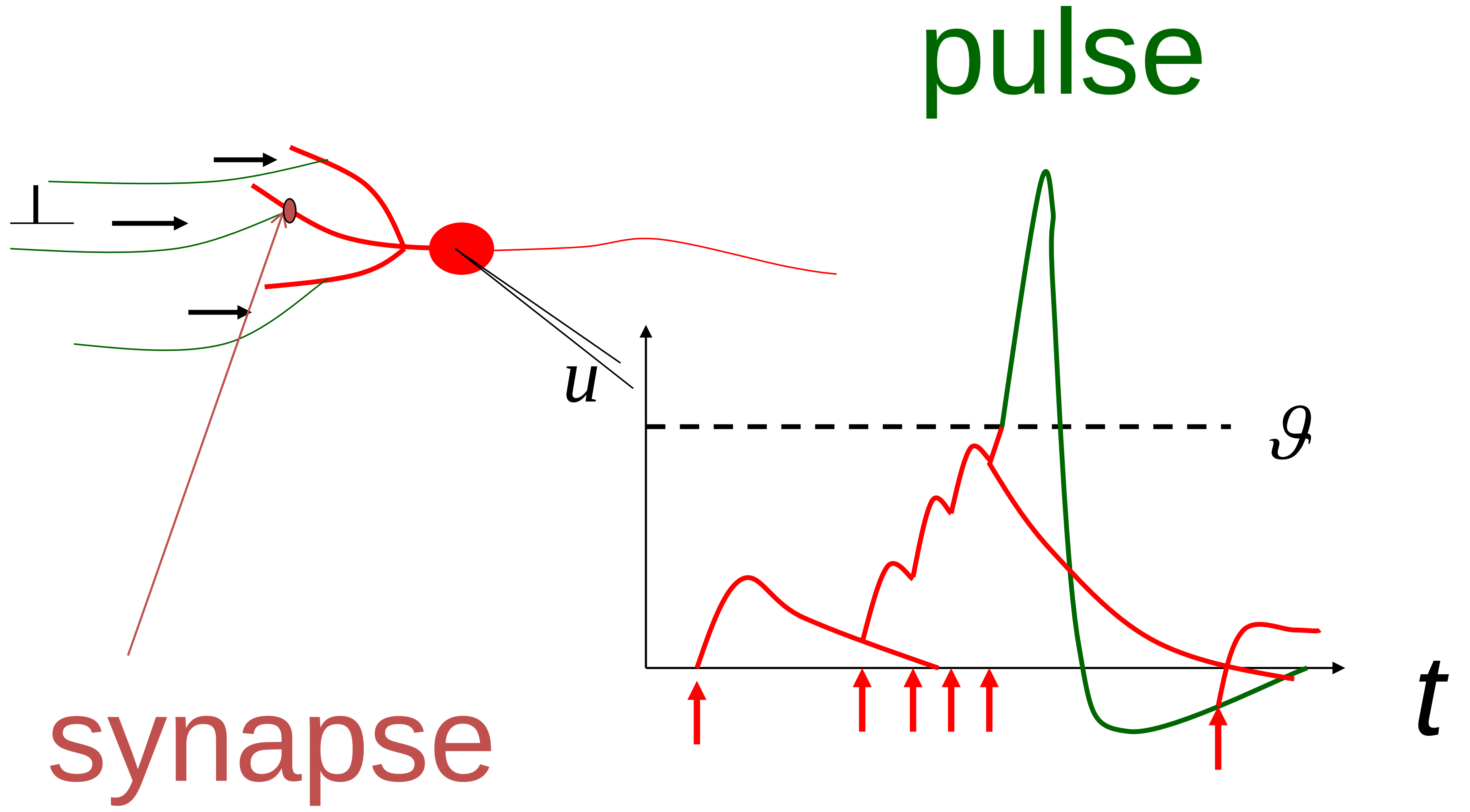
Signal:
Action potential (short pulse)



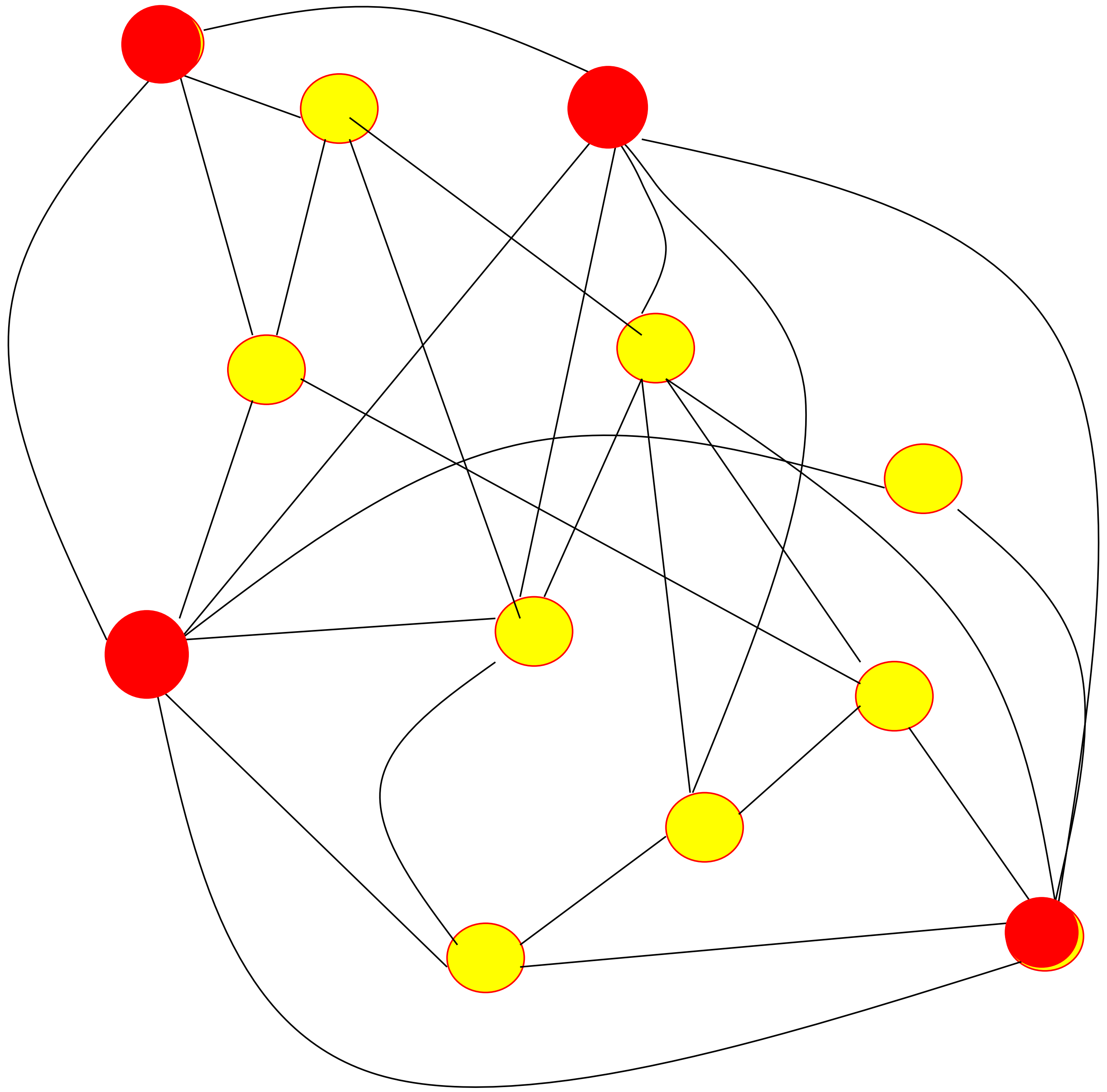
The brain: signal transmission

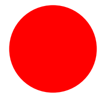


The brain: neurons sum their inputs

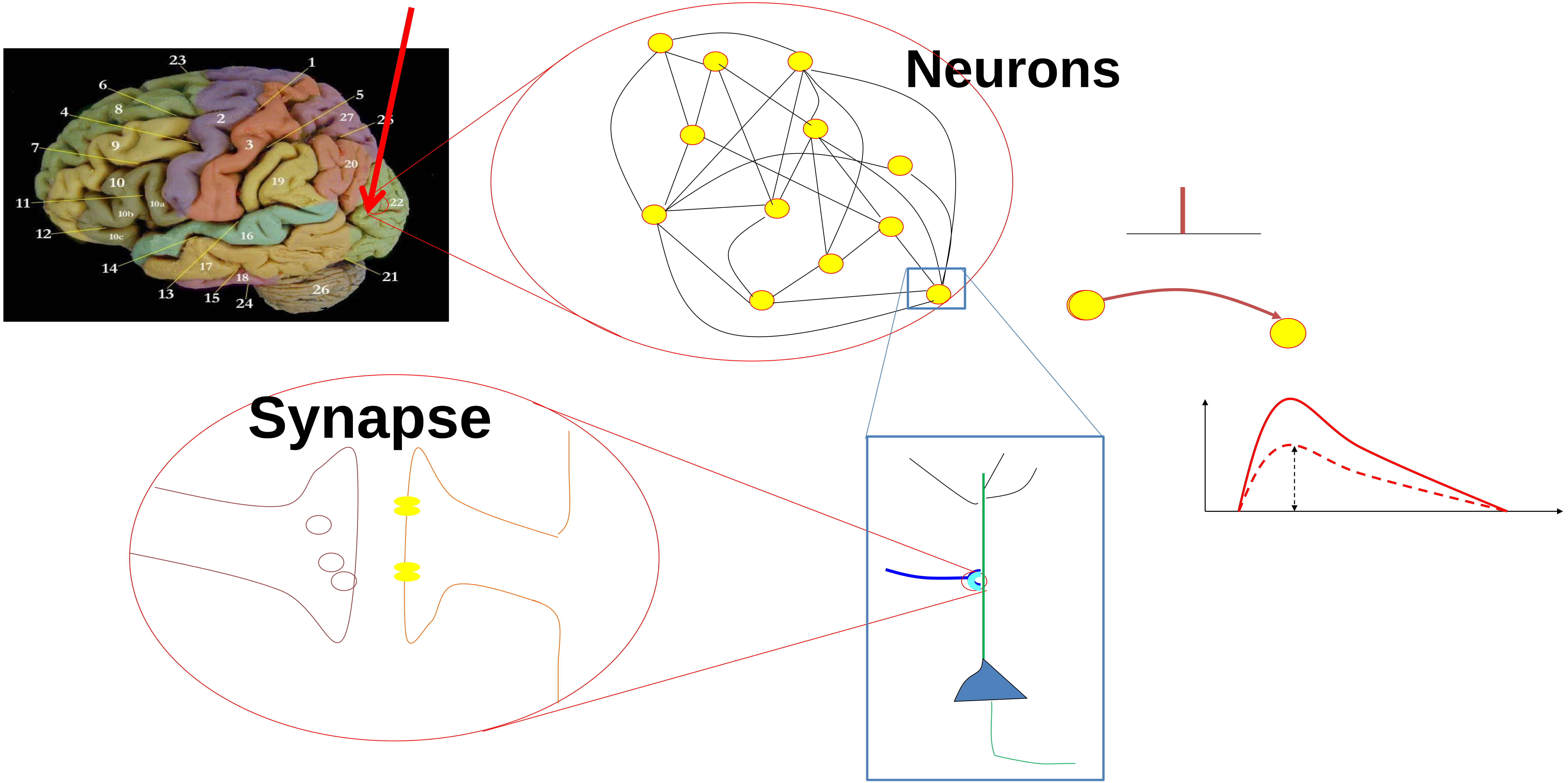


Summary: the brain is a large network of neurons



 Active neuron

Learning in the brain: changes between connections



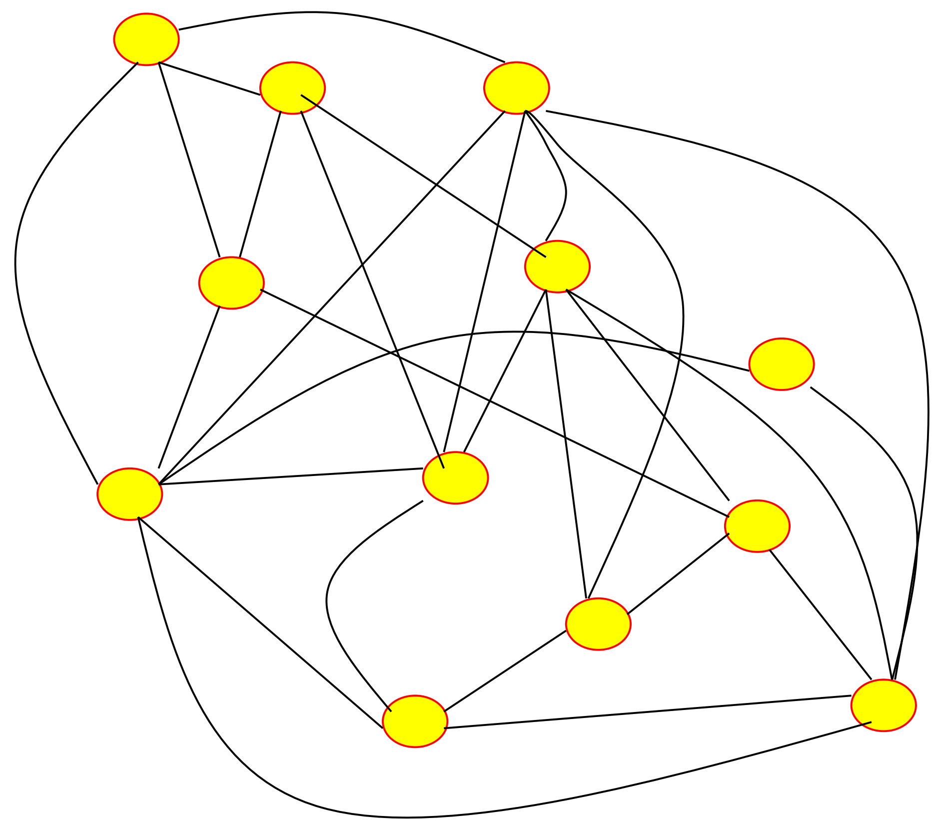
learning = change of connection

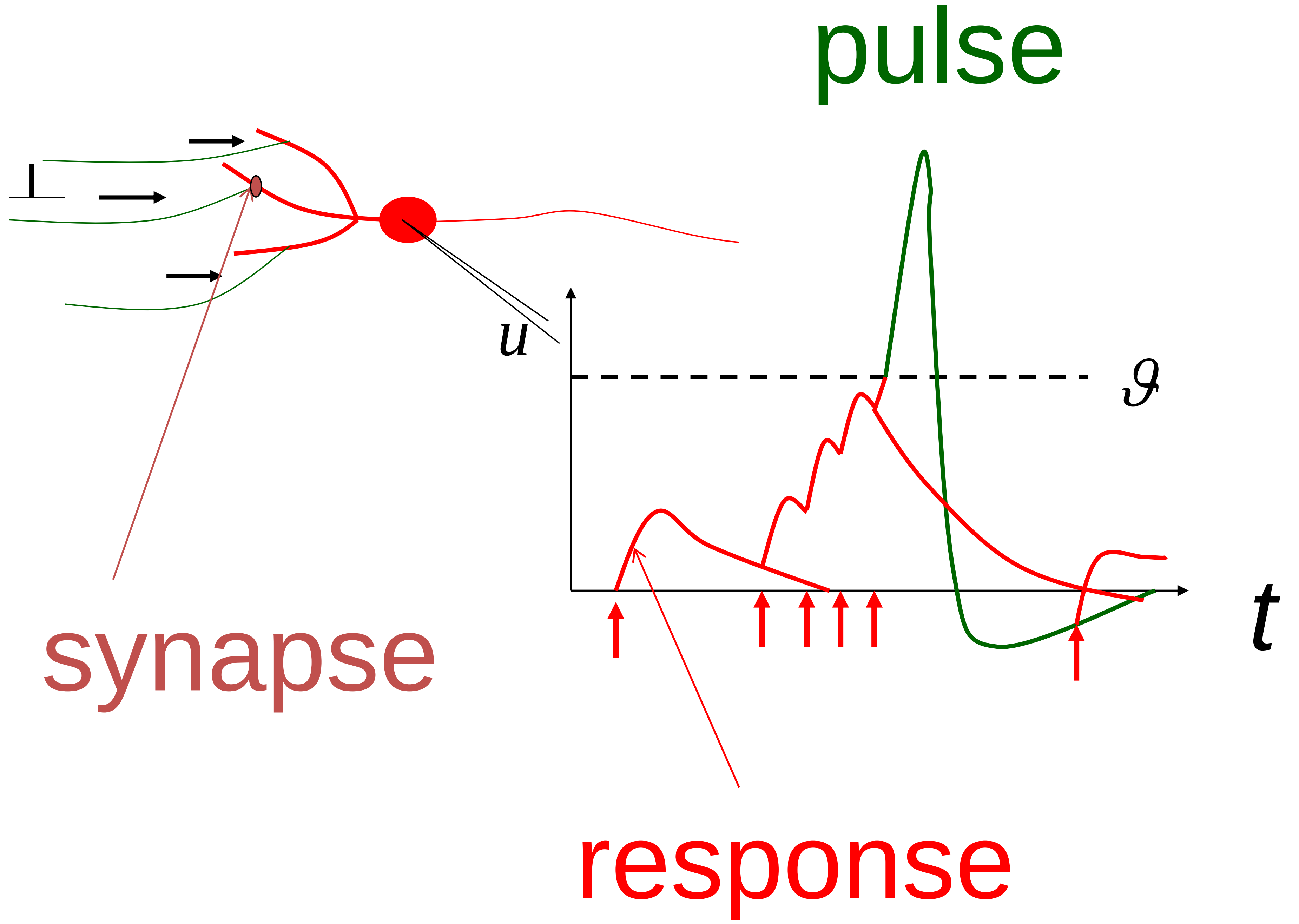
Artificial Neural Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

1. The brain
2. **Artificial Neural Networks**

Modeling: artificial neurons

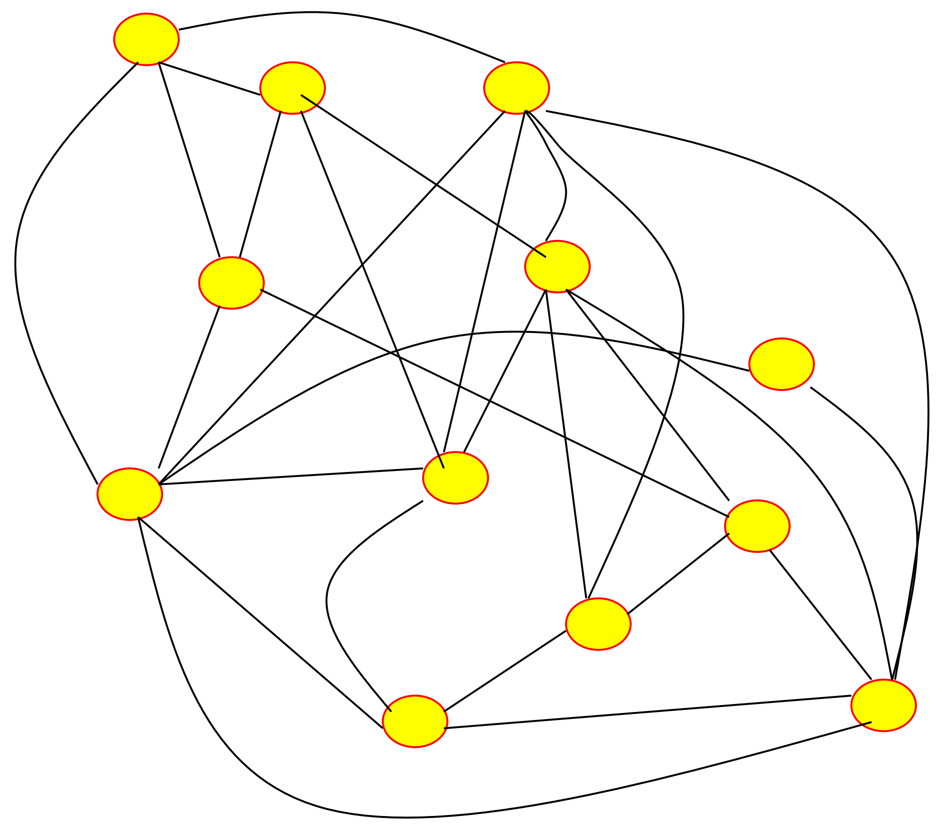




-responses are added
-pulses created at threshold
-transmitted to other

→ Mathematical description

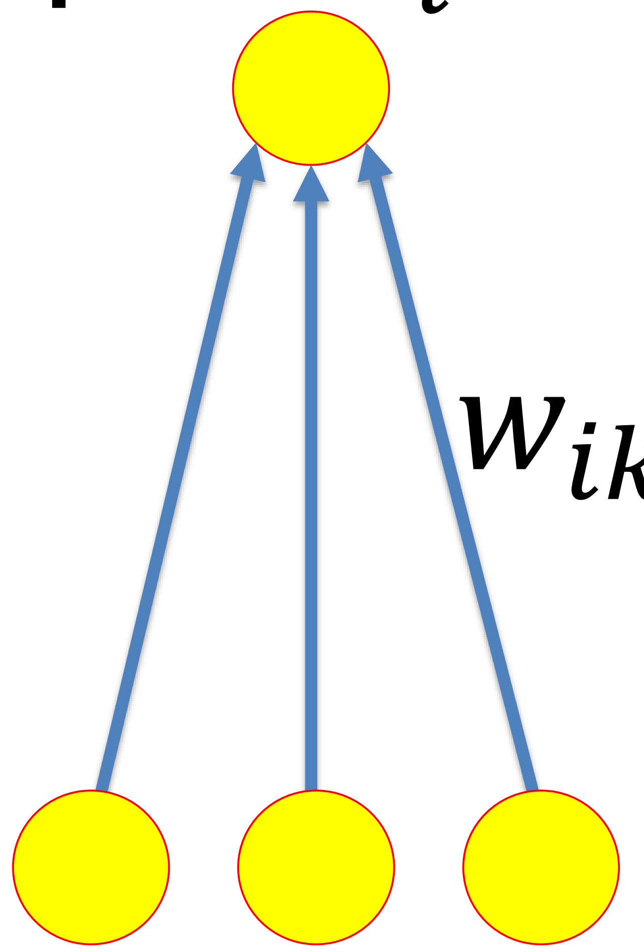
Modeling: artificial neurons



forget spikes: continuous activity x
forget time: discrete updates

nonlinearity/threshold

activity of output $x_i = g\left(\sum_k w_{ik}\, x_k \quad \vartheta\right)$

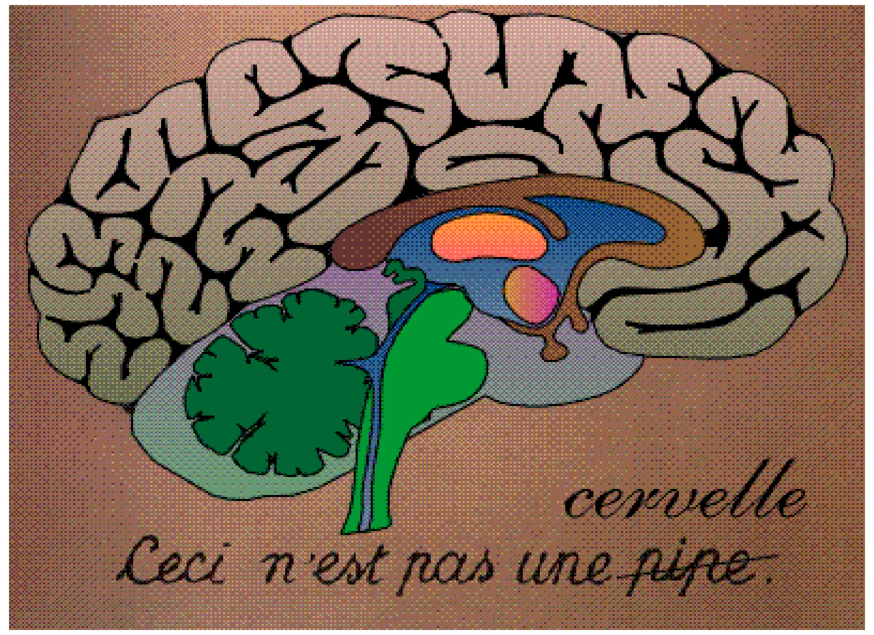


w_{ik}

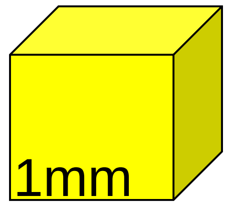
weights = adaptive parameters

activity of inputs x_k

Neurons and Synapses form a big network



Brain



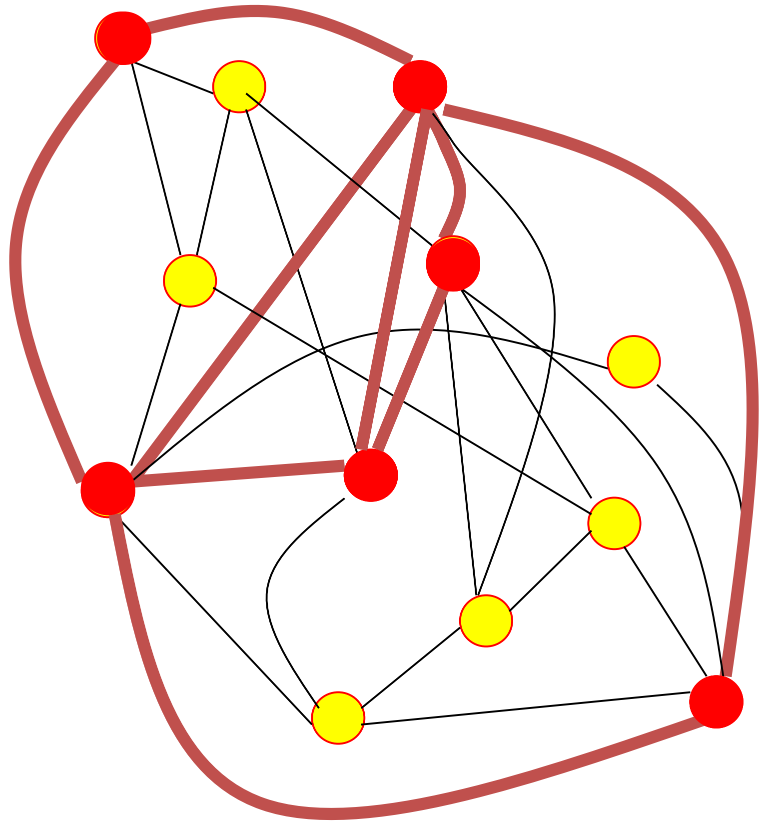
1mm

10 000 neurons
3 km of wire

10 billions neurons

10 000 connexions/neurons

memory in the connections



No separation of processing and memory

Distributed Architecture

Quiz: biological neural networks

[ ] Neurons in the brain have a threshold.
[ ] Learning means a change in the threshold.
[ ] Learning means a change of the connection weights
[ ] The total input to a neuron is the weighted sum of individual inputs
[ ] The neuronal network in the brain is feedforward: it has no recurrent connections

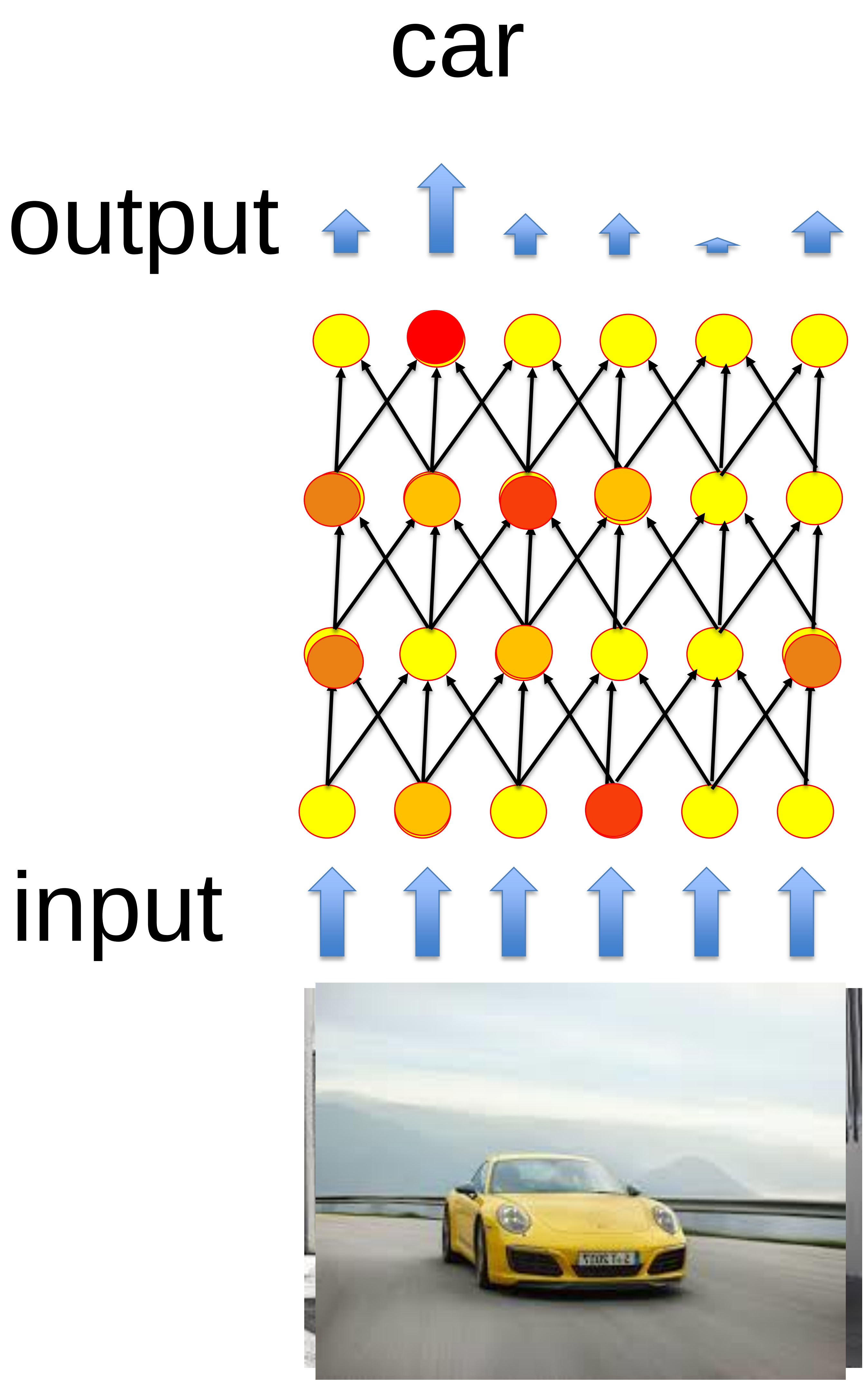
Artificial Neural Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

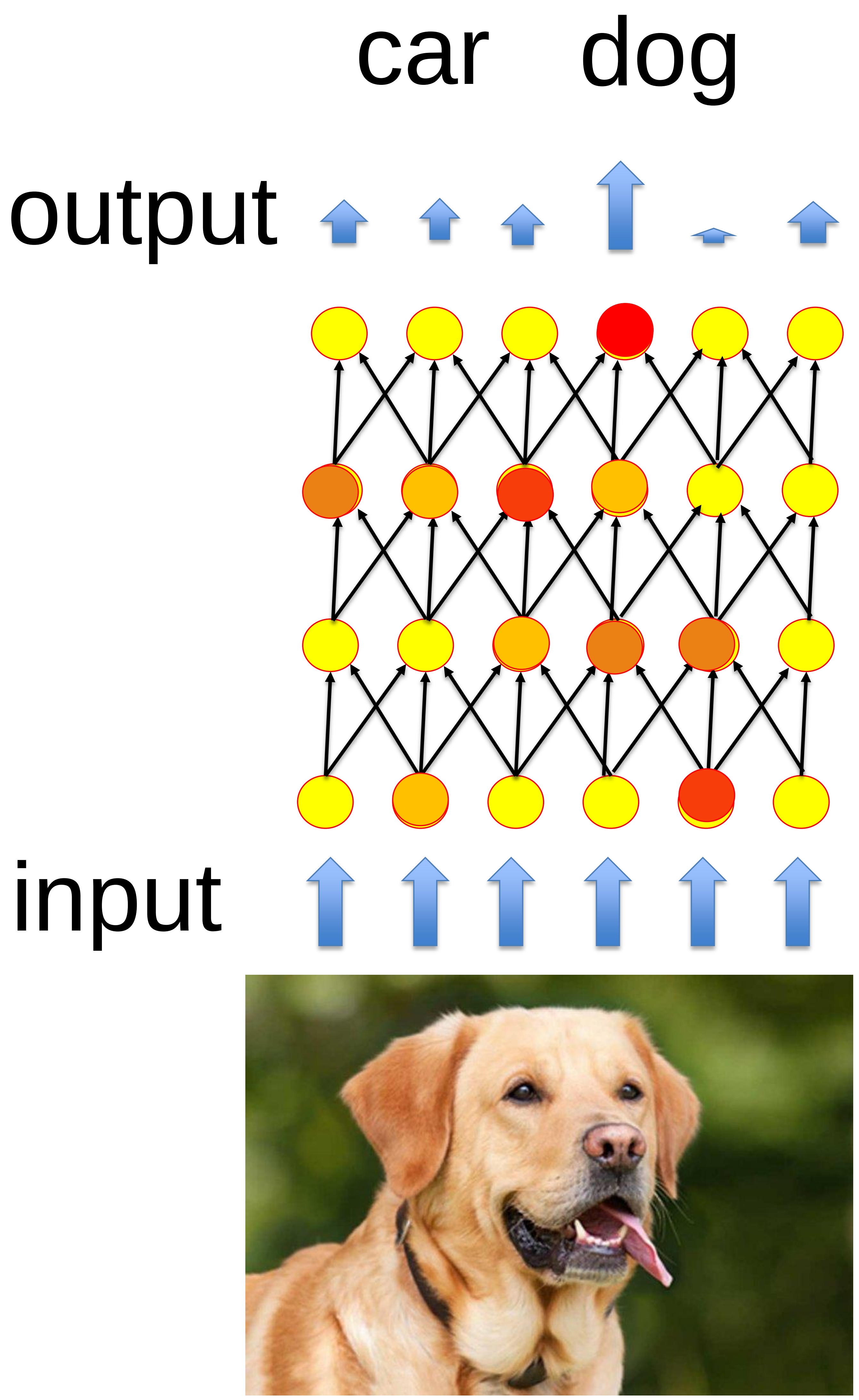
1. The brain
2. **Artificial Neural Networks**
   - **artificial neurons**
   - **artificial neural networks for classification**

Artificial Neural Networks for classification

feedforward network



Artificial Neural Networks for classification



Aim of learning:
Adjust connections such that output class is correct (for each input)

Artificial Neural Networks

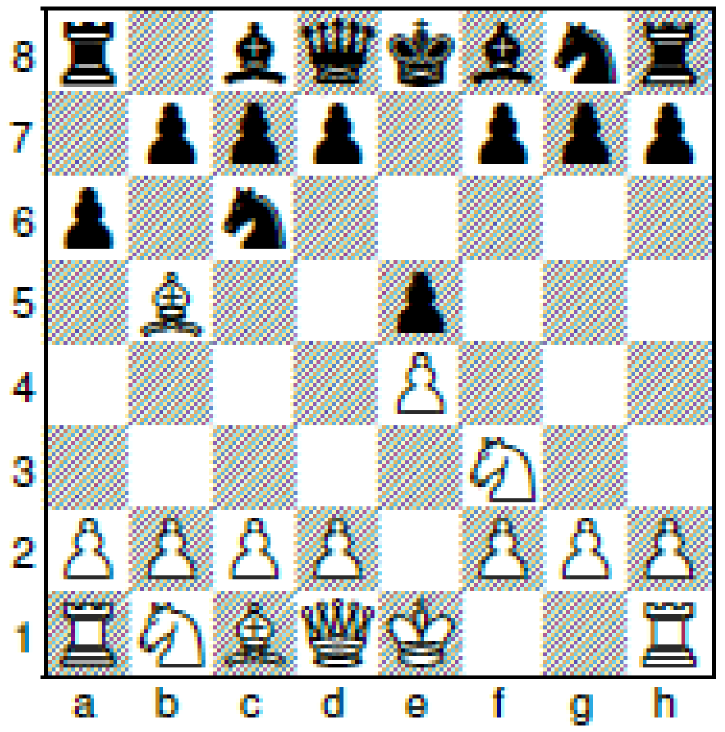
Wulfram Gerstner

EPFL, Lausanne, Switzerland

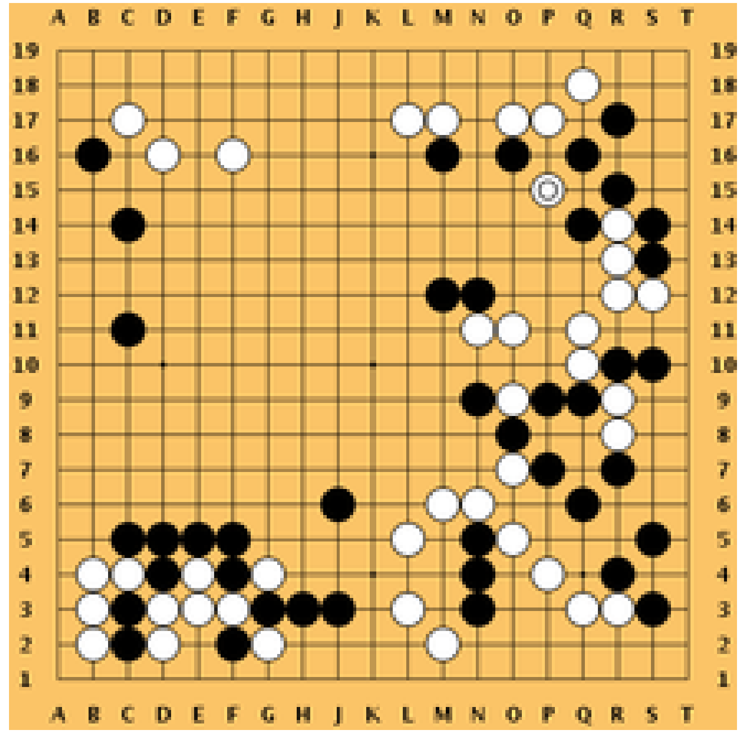
1. The brain
2. Artificial Neural Networks
   - artificial neurons
   - Neural networks for classification
   - **Neural networks for action learning**

Deep reinforcement learning

Chess



Go



Artificial neural network (*AlphaZero*) discovers different strategies by playing against itself.

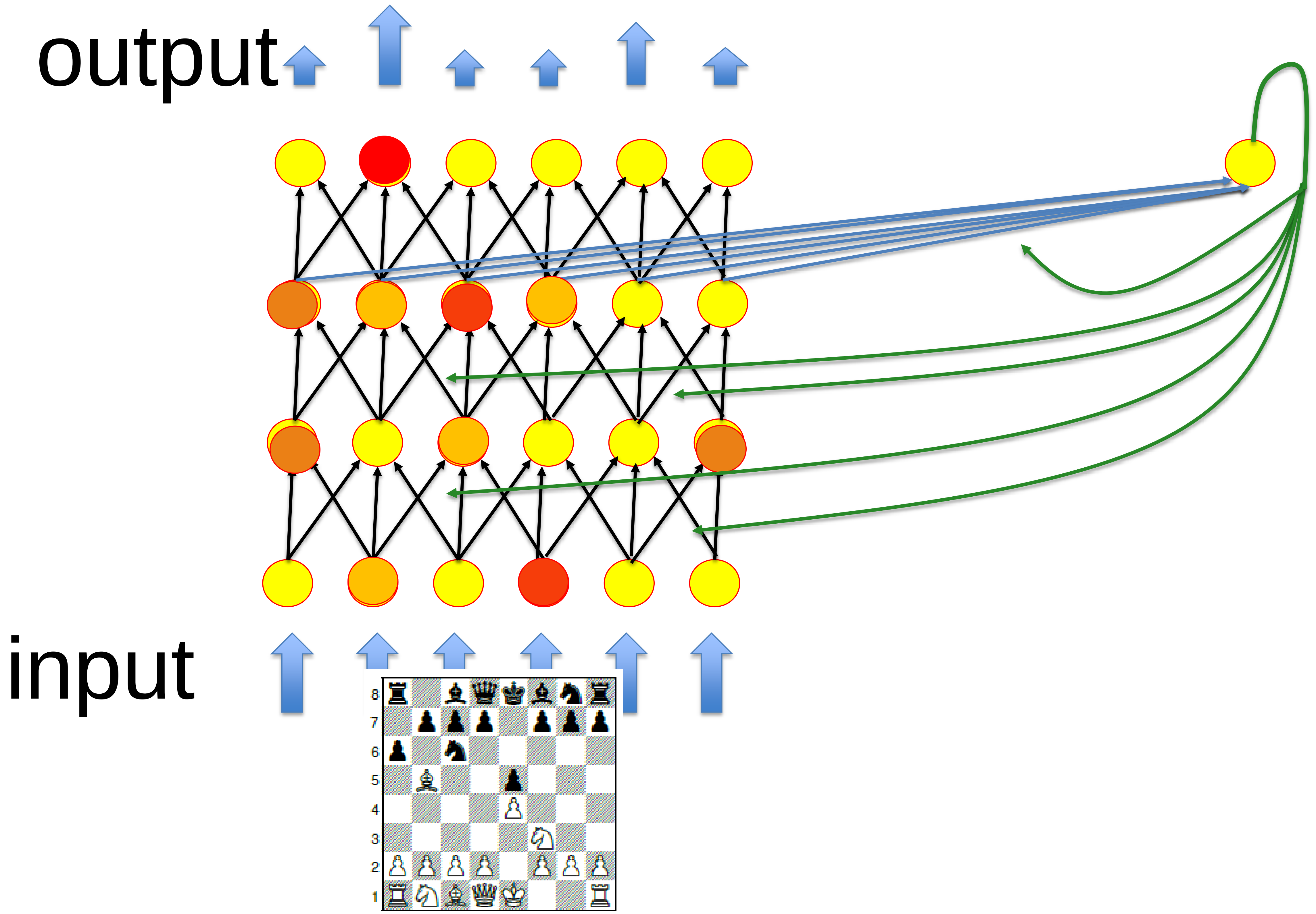
In Go, it beats Lee Sedol

Reinforcement learning: Learning through rewards (win)

Network for choosing action

action:
Advance king

2e output for **value** of action:
probability to win



learning:
- change connections

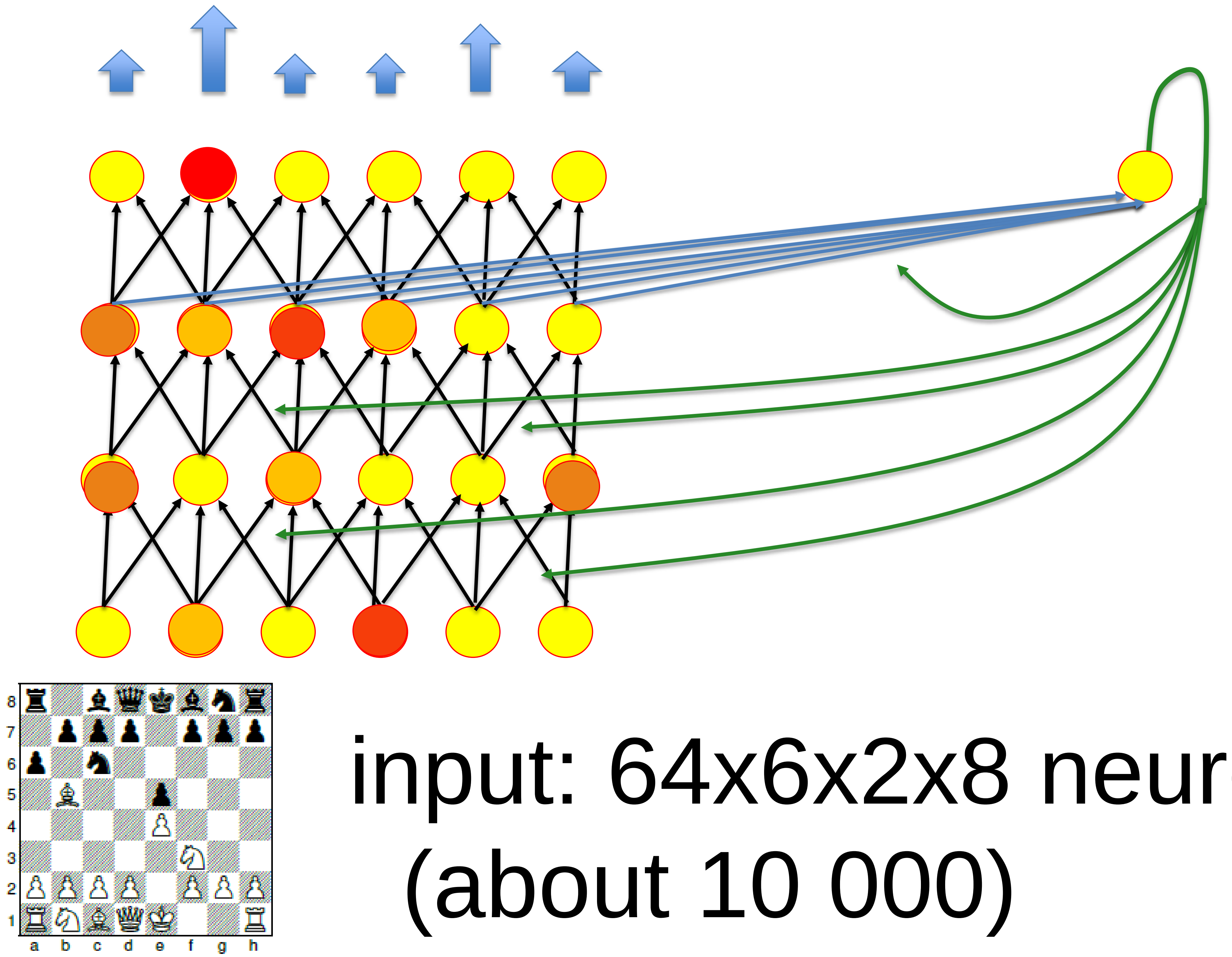
aim:
- Predict value of position
- Choose next action to win

Deep reinforcement learning (alpha zero)

Silver et al. (2017) , Deep Mind

output: 4672 actions

advance king



Training 44Mio games (9 hours)

Planning:
potential sequences
(during 1s before playing next action)

input: 64x6x2x8 neuronss
(about 10 000)

Deep reinforcement learning (alpha zero)

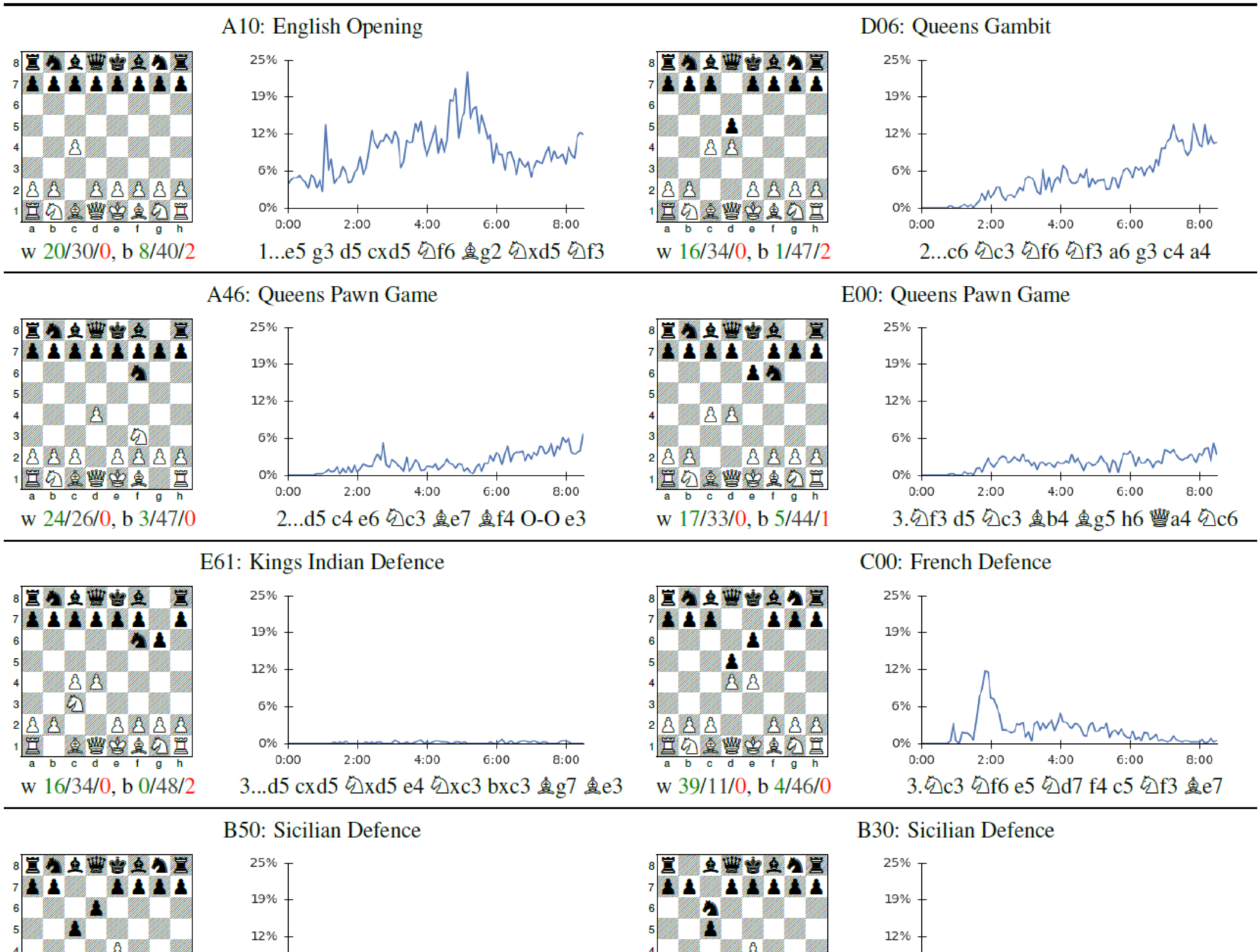
Silver et al. (2017)

Chess:

-discovers classic openings

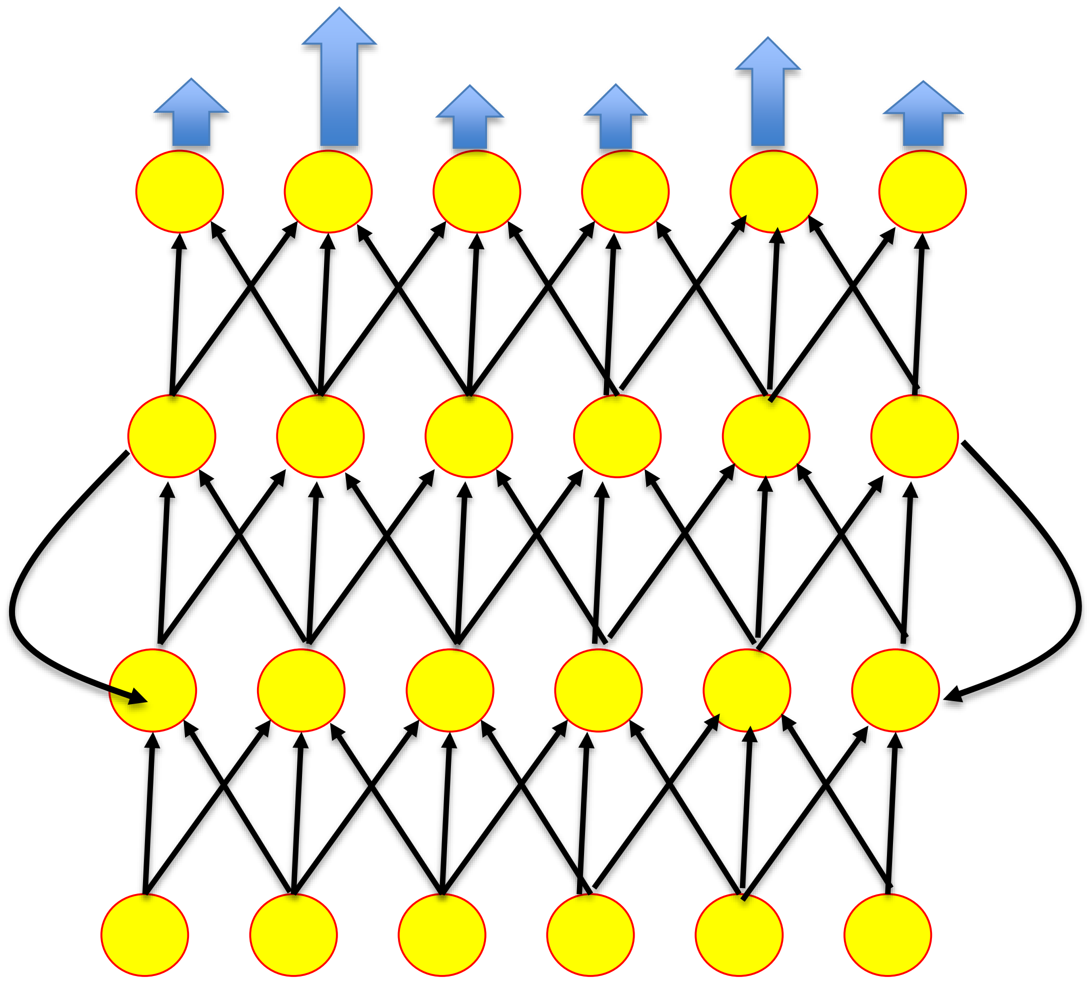
-beats best human players

-bets best classic AI algorithms



Deep networks with recurrent connections

‘a man sitting on a couch with a dog’



Network desribes the image with the words:

‘a man sitting on a couch with a dog’

(Fang et al. 2015)

Quiz: Classification versus Reinforcement Learning

[ ] Classification aims at predicting the correct category such as ‘car’ or ‘dog’
[ ] Classification is based on rewards
[ ] Reinforcement learning is based on rewards
[ ] Reinforcement learning aims at optimal action choices
[ ] Recurrent neural networks are useful for sequences

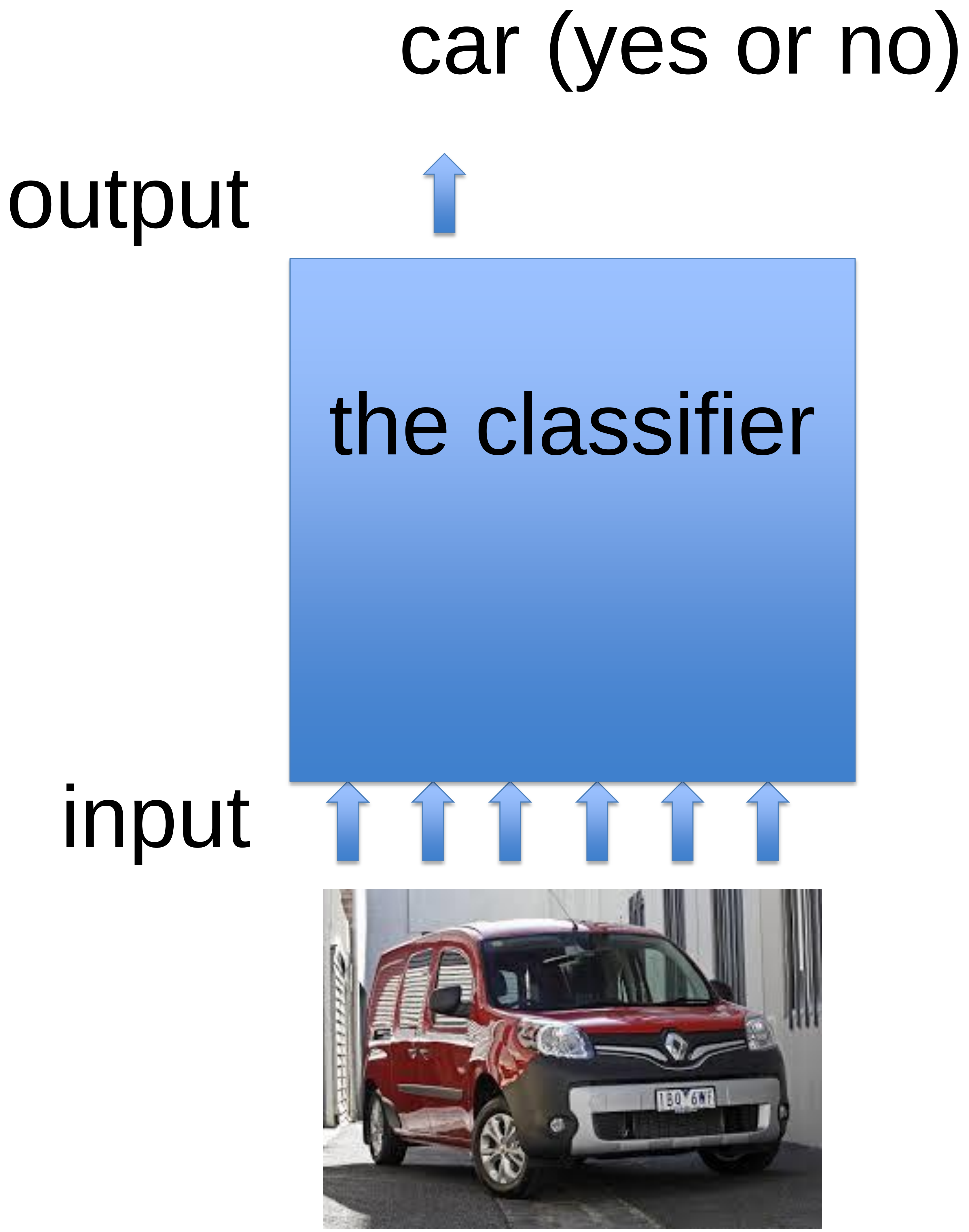
Artificial Neural Networks: Lecture 1
Simple Perceptrons for Classification

Wulfram Gerstner
EPFL, Lausanne, Switzerland

Objectives for today:

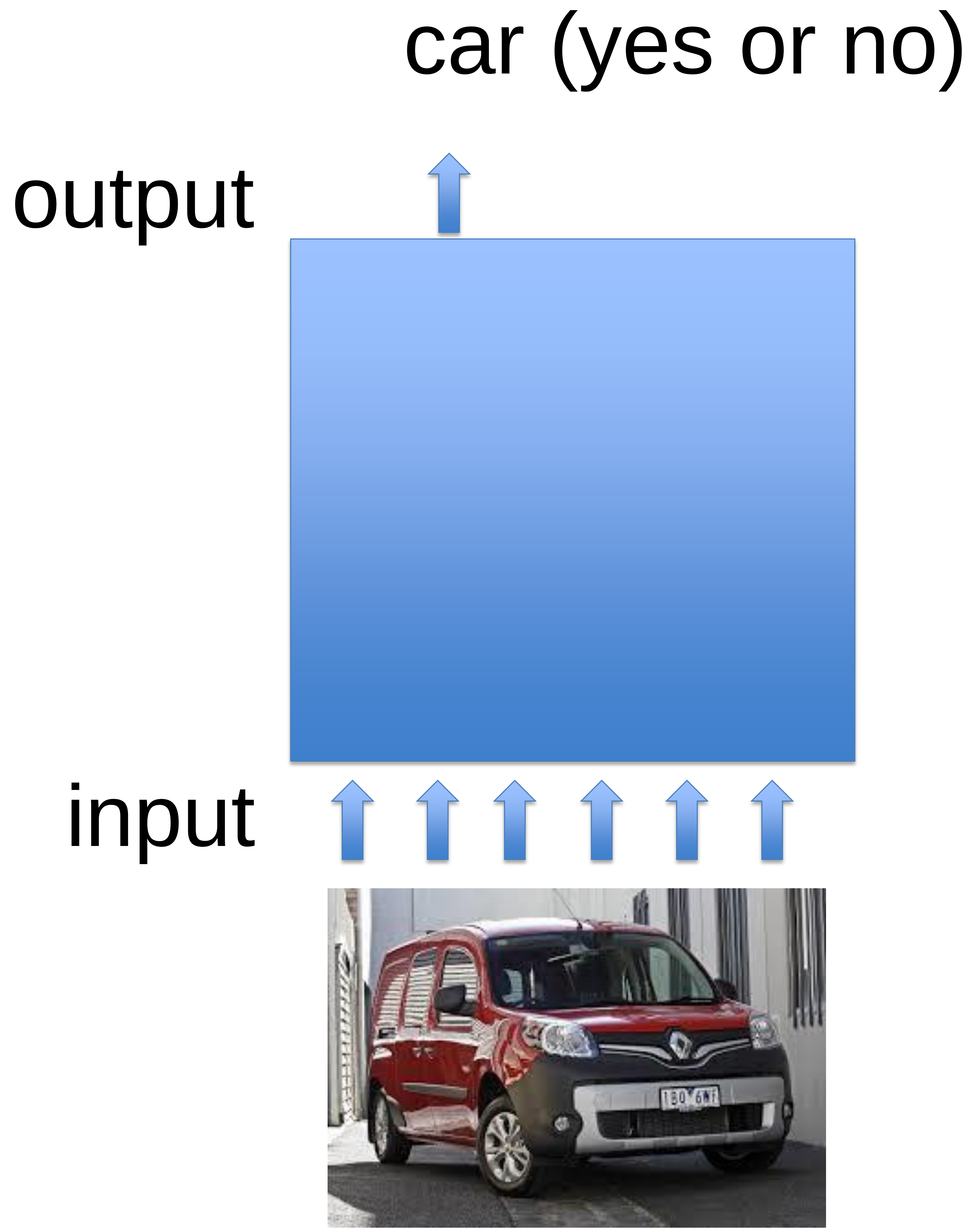
- understand classification as a geometrical problem
- discriminant function of classification
- linear versus nonlinear discriminant function
- perceptron algorithm
- gradient descent for simple perceptrons

1. The problem of Classification



1. The problem of Classification

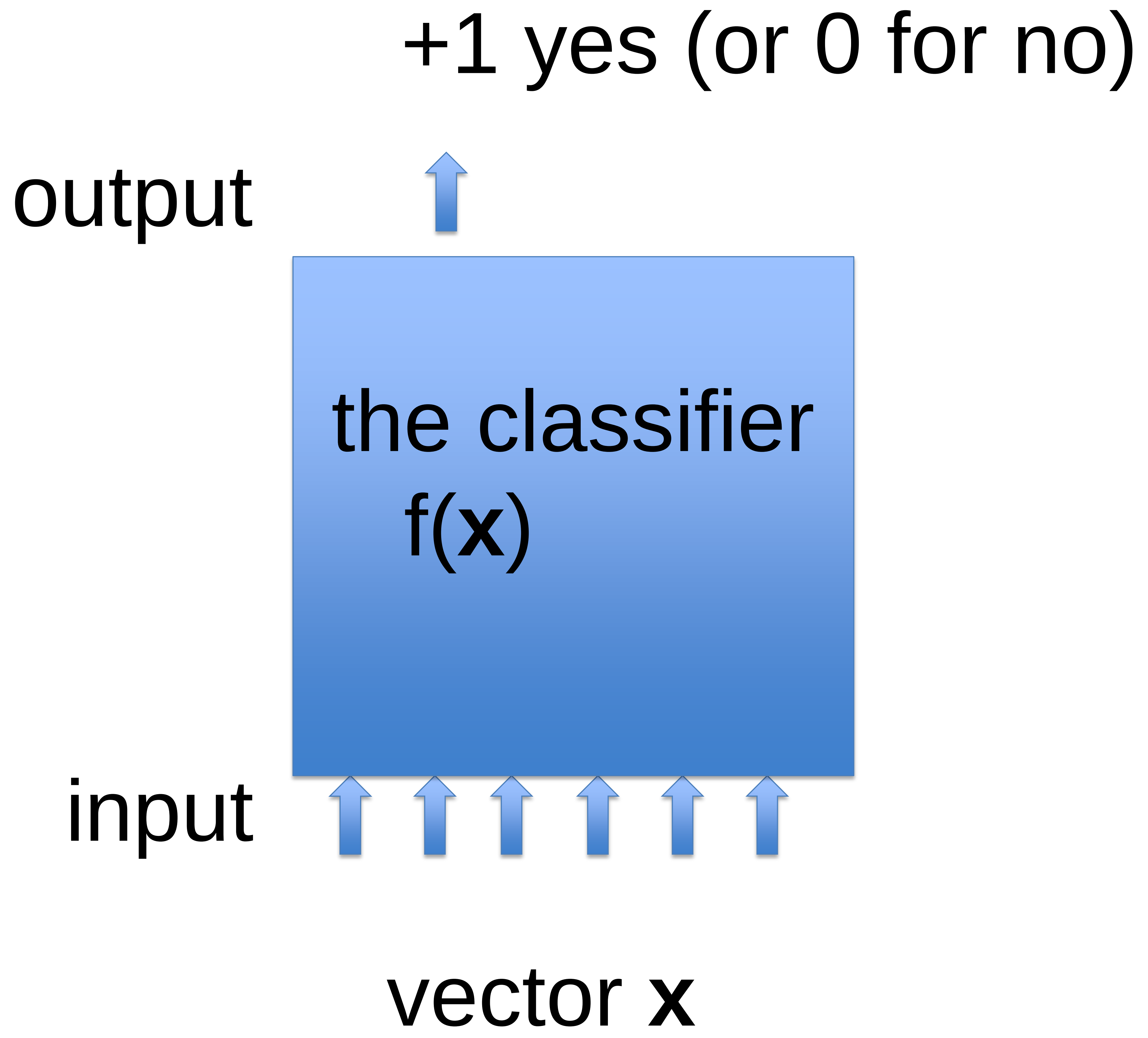
Blackboard 1:
from images to vector



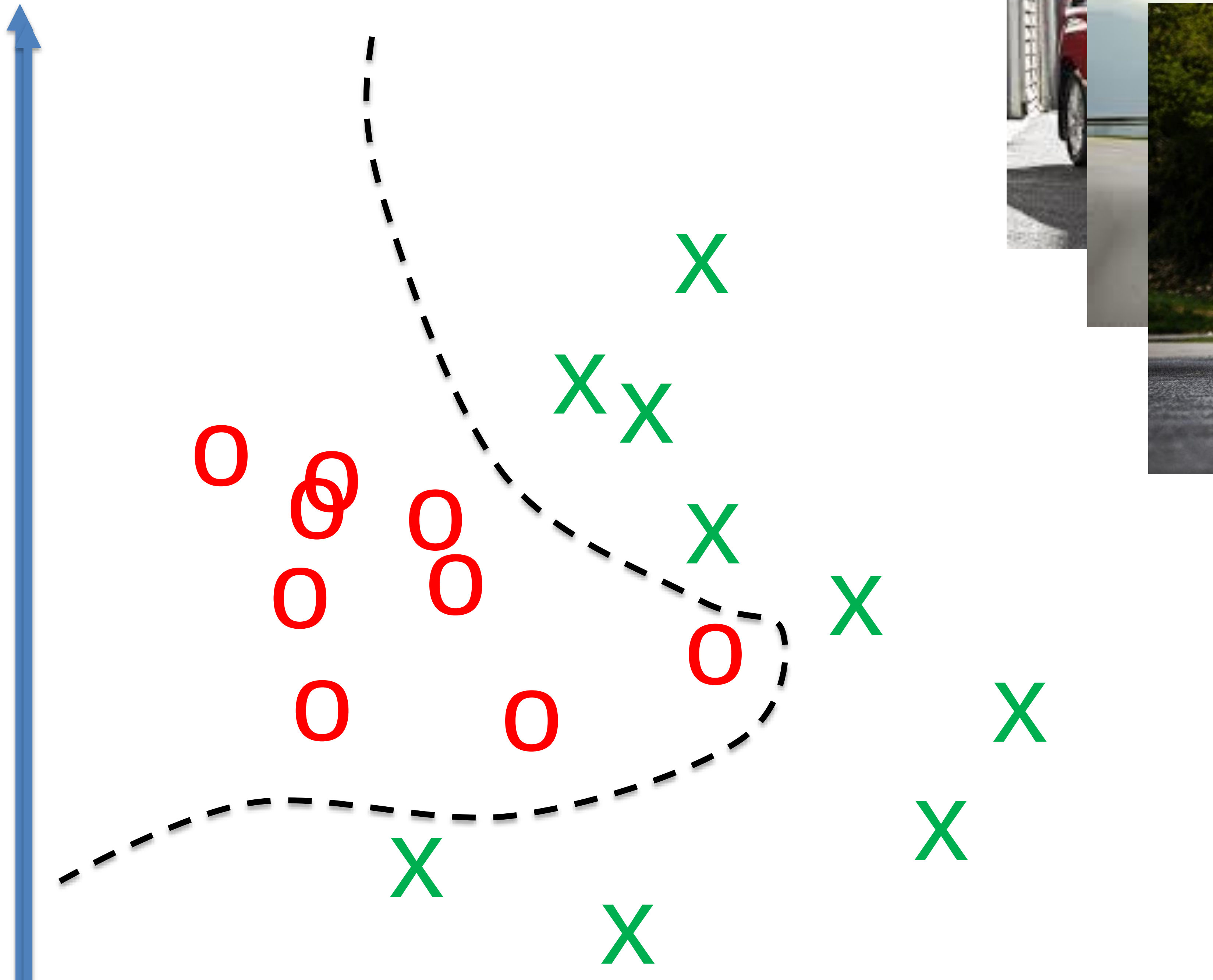
Blackboard 1:

from images to vector

1. The problem of Classification



1. Classification as a geometric problem



Blackboard 2:

from vectors to classification

1. Classification as a geometric problem

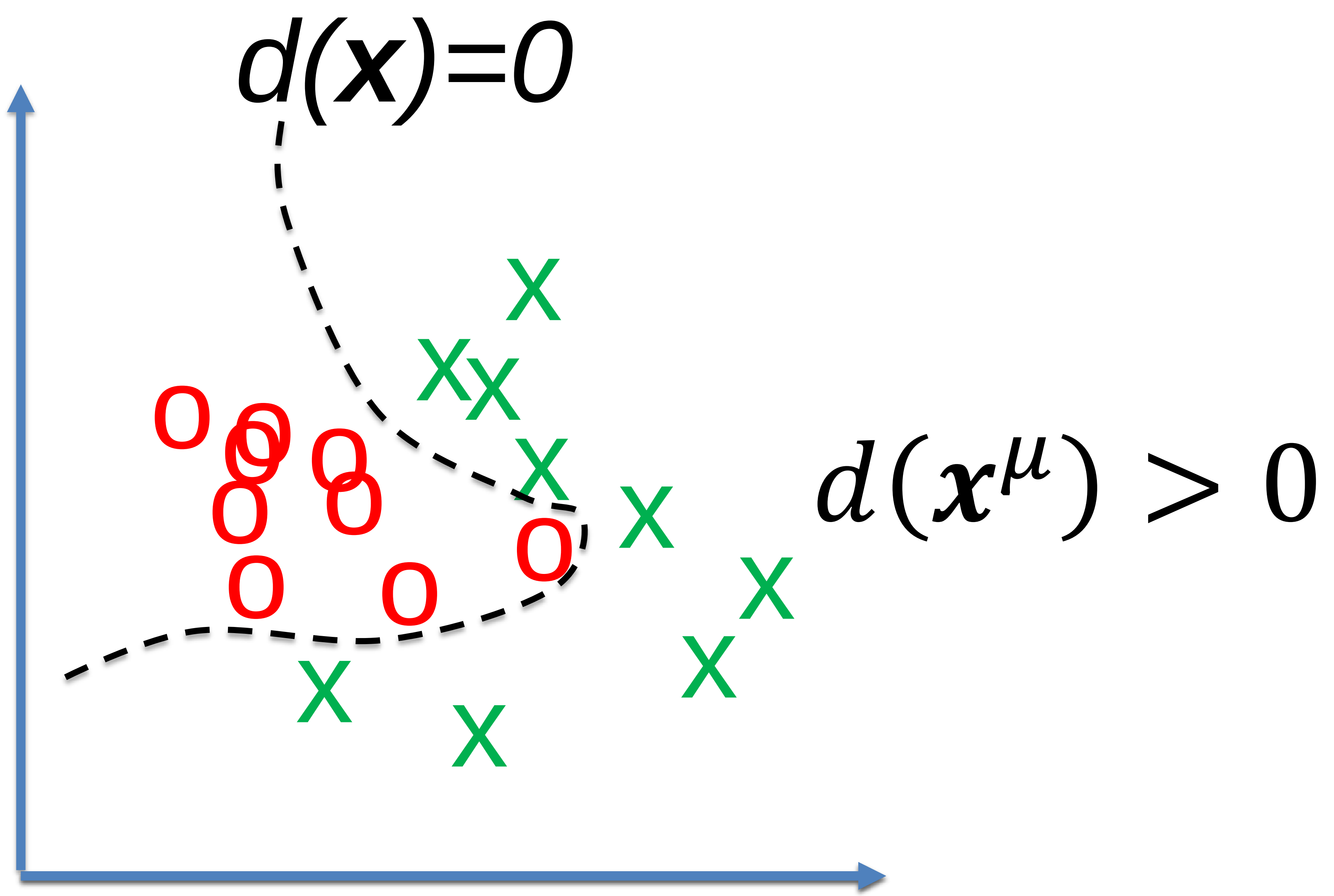
Task of Classification

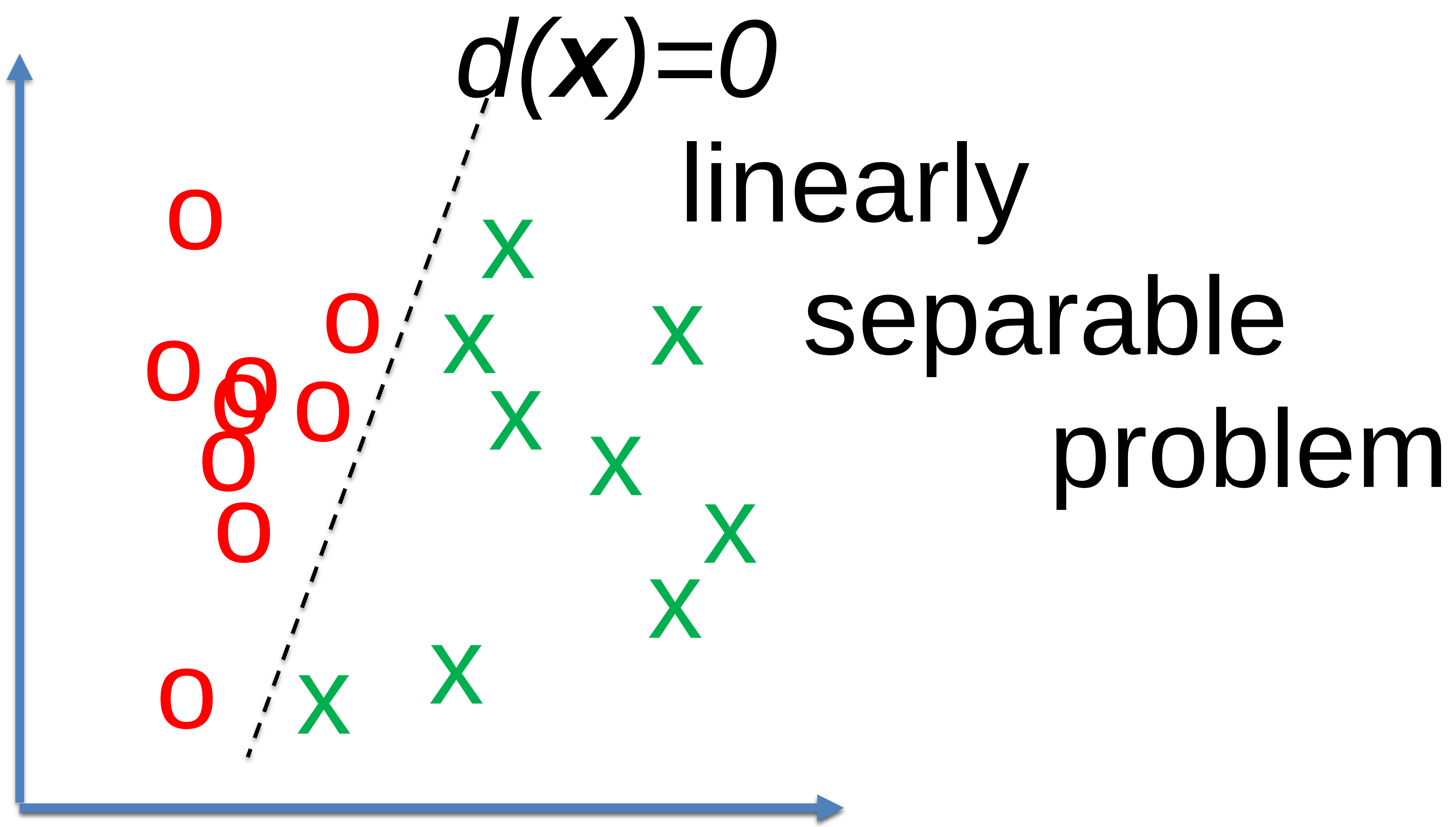
= find a **separating surface** in the high-dimensional input space

Classification by **discriminant function** $d(\mathbf{x})$

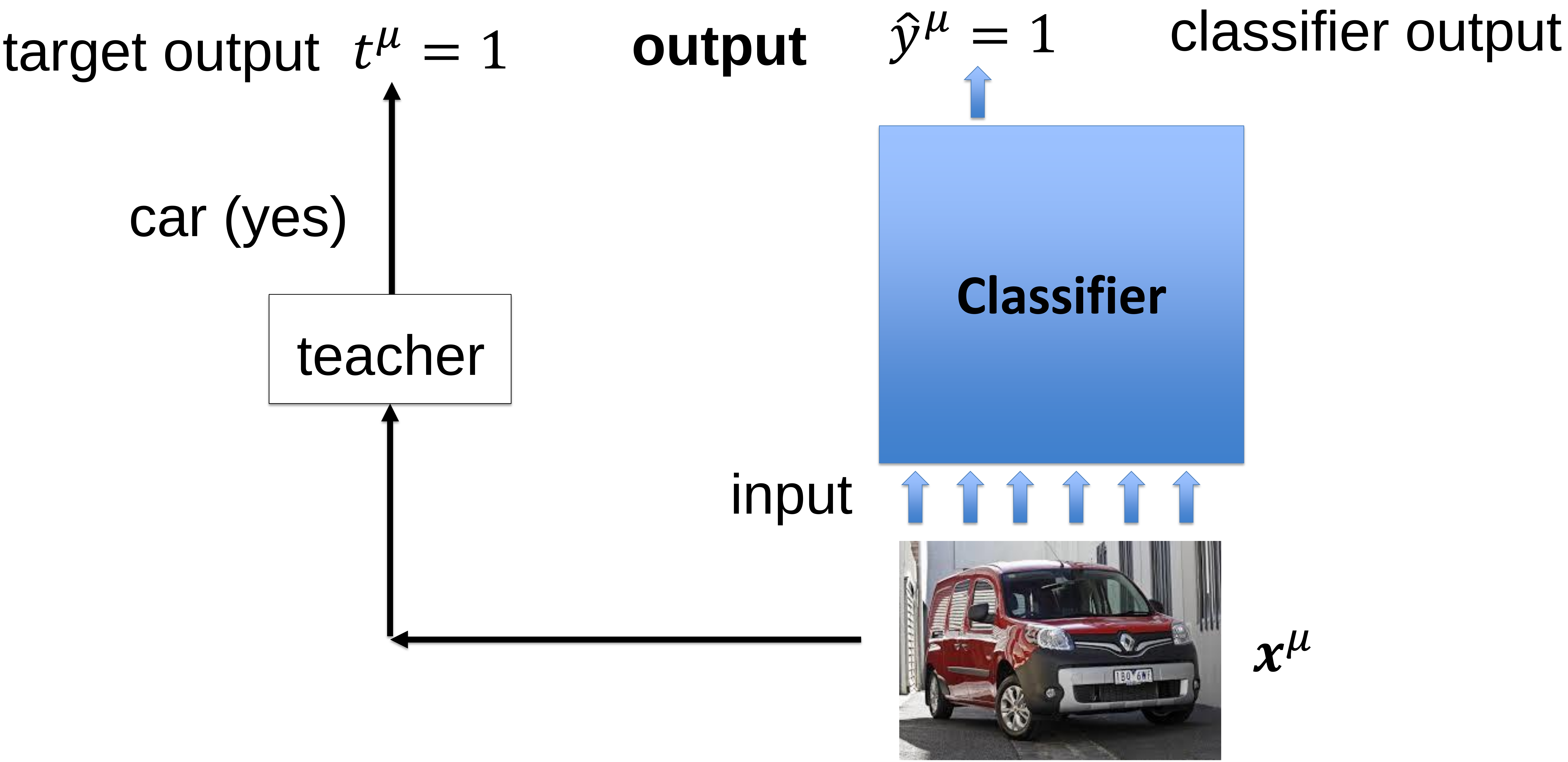
→ $d(\mathbf{x})=0$ on this surface; $d(\mathbf{x})>0$ *for all positive examples* $\mathbf{x}$

$d(\mathbf{x})<0$ *for all counter examples* $\mathbf{x}$





2. Data base for Supervised learning



2. Data base for Supervised learning

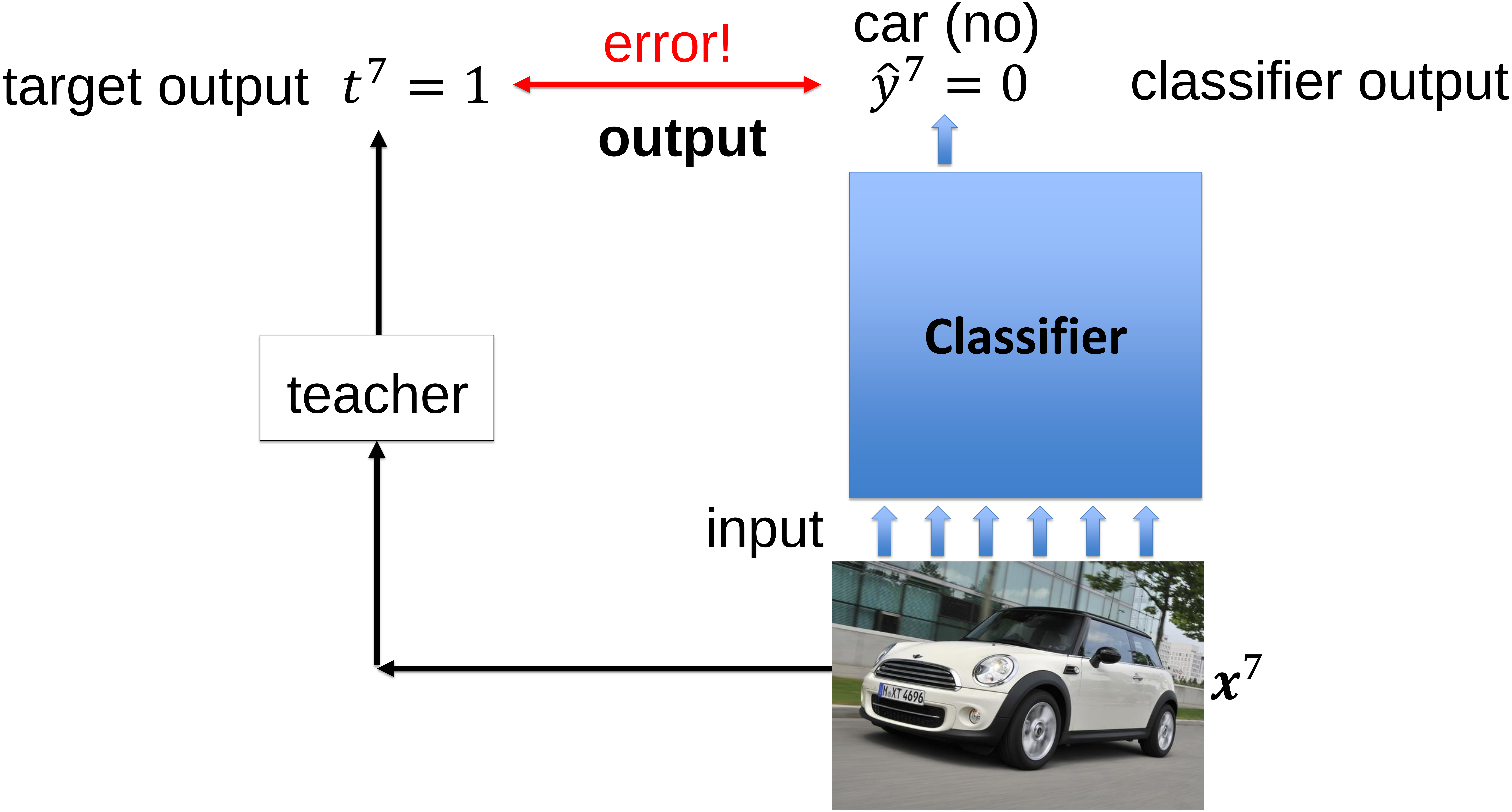
P data points $\{\ (\boldsymbol{x}^{\mu}, t^{\mu})\ ,\quad 1 \le \mu \le P\ \};$

input target output

$t^{\mu} = 1$ car =yes

$t^{\mu} = 0$ car =no

2. Data base for Supervised learning



2. Data base for Supervised learning

P data points $\quad \{ \quad (\boldsymbol{x}^\mu, t^\mu) \quad , \qquad 1 \le \mu \le P \quad \};$

input target output

for each data point $\boldsymbol{x}^\mu$, the classifier gives an output $\hat{y}^\mu$

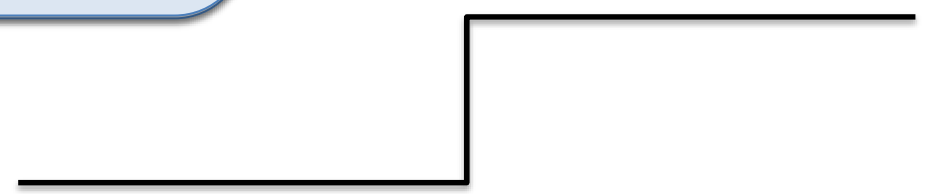
→ use errors $\hat{y}^\mu \neq t^\mu$ **for optimization of classifier**

Remark: for multi-class problems $\mathbf{y}$ and $\boldsymbol{t}$ are vectors

3. Single-Layer networks: simple perceptron

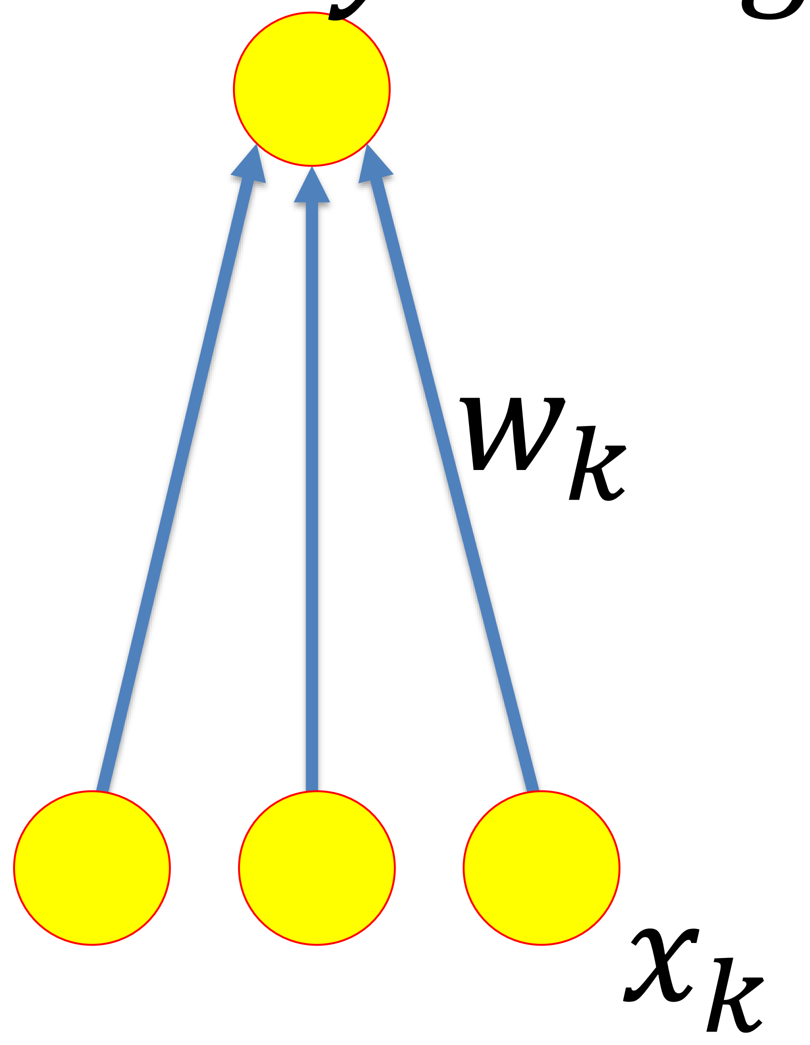
Blackboard 3:
hyperplane

$$\hat{y} = g(a') = \begin{cases} +1 \text{ if } a' > \vartheta \\ 0 \text{ if } a' < \vartheta \end{cases}$$



$$\hat{y} = f(\boldsymbol{x})$$

$$\hat{y} = g\left(\sum_k w_k\, x_k\right)$$



output

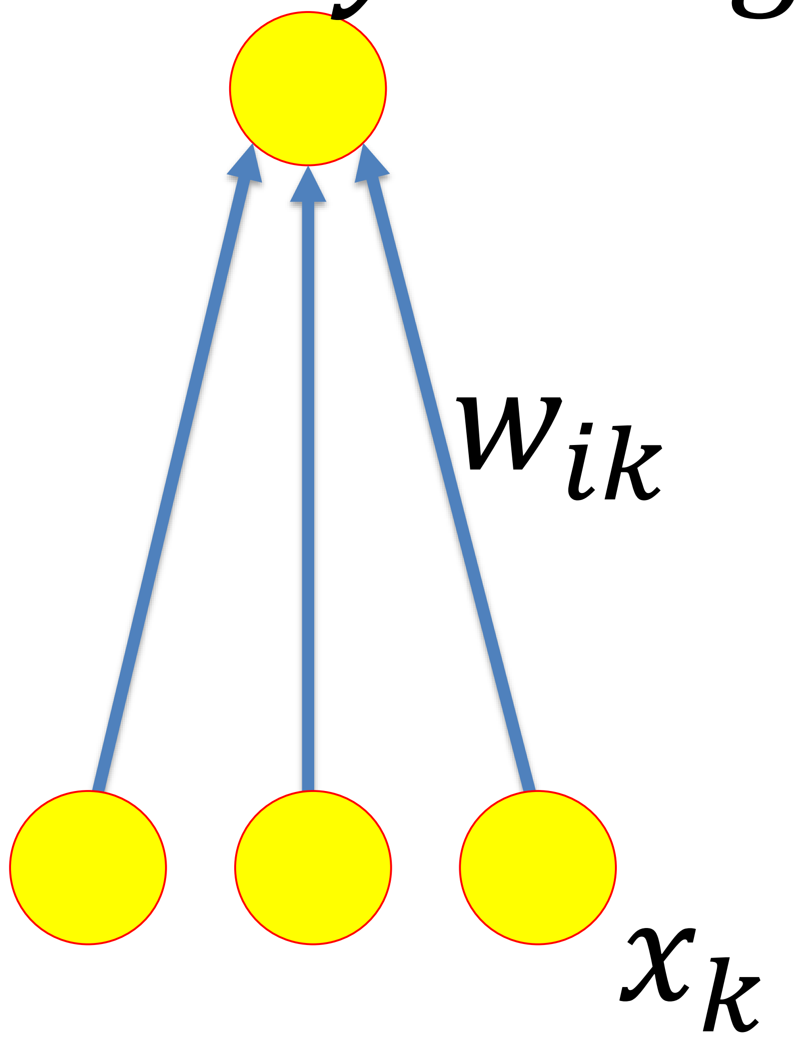
vector $\boldsymbol{x}$

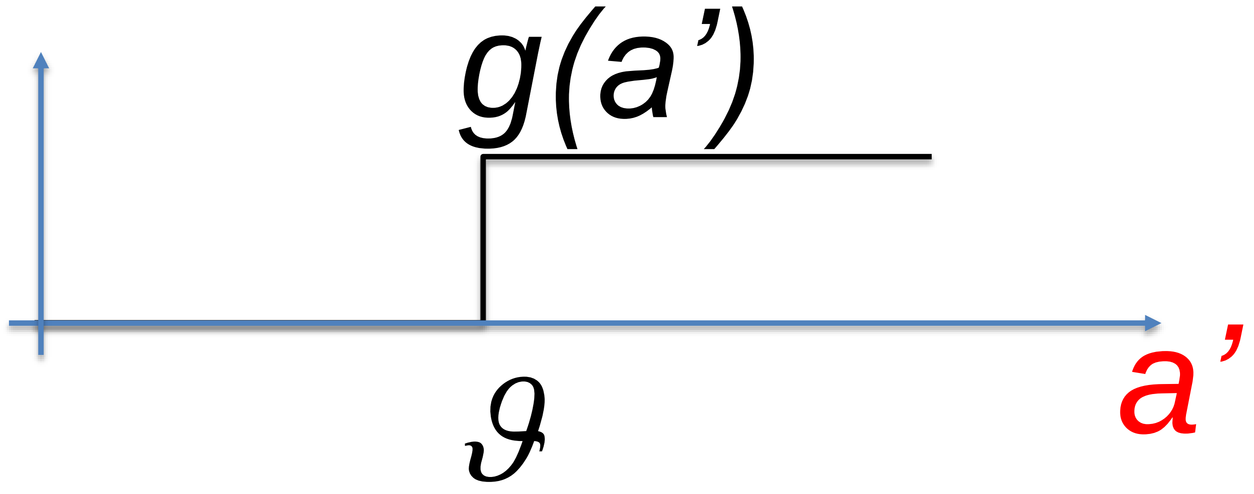
3. Single-Layer networks: simple perceptron

$$\hat{y}^{\mu} = 0.5[1 + sgn(\underbrace{\textstyle\sum_k w_k x_k - \vartheta}_{a})]$$

output

$$\hat{y}^{\mu} = g\left(\underbrace{\sum_k w_k x_k}_{a'}\right)$$





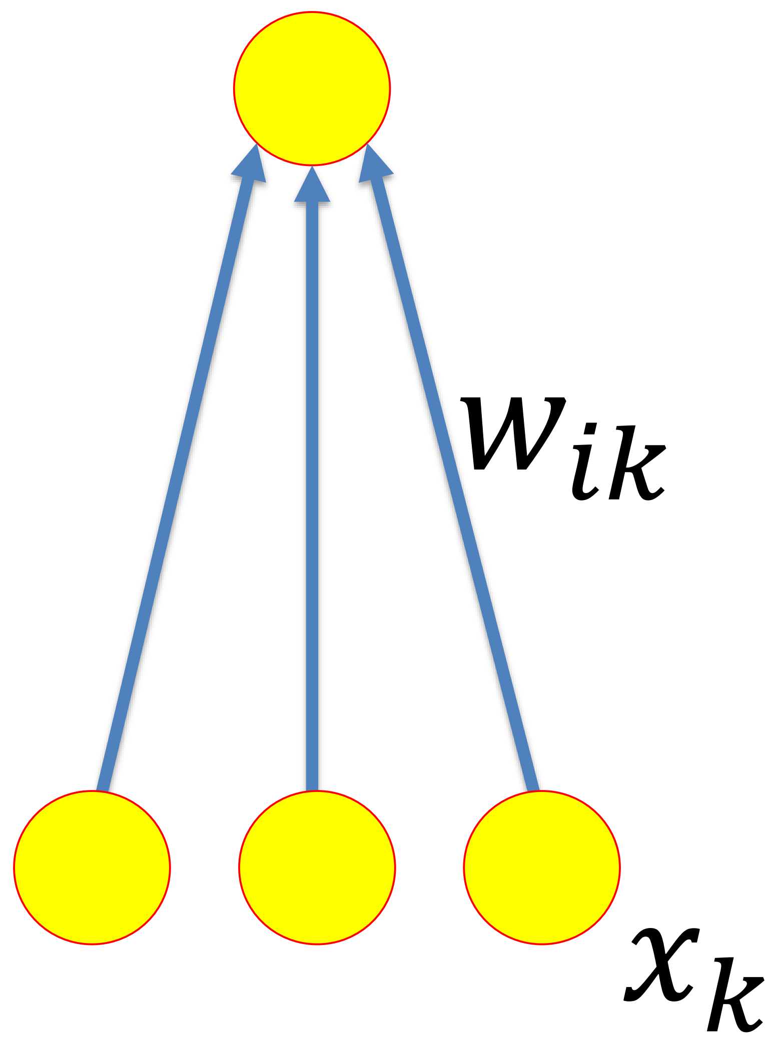
$$g(a') = \begin{cases} 1 & \text{if } a' > \vartheta \\ 0.5 & \text{if } a' = \vartheta \\ 0 & \text{if } a' > \vartheta \end{cases}$$

input

vector $\boldsymbol{x}$

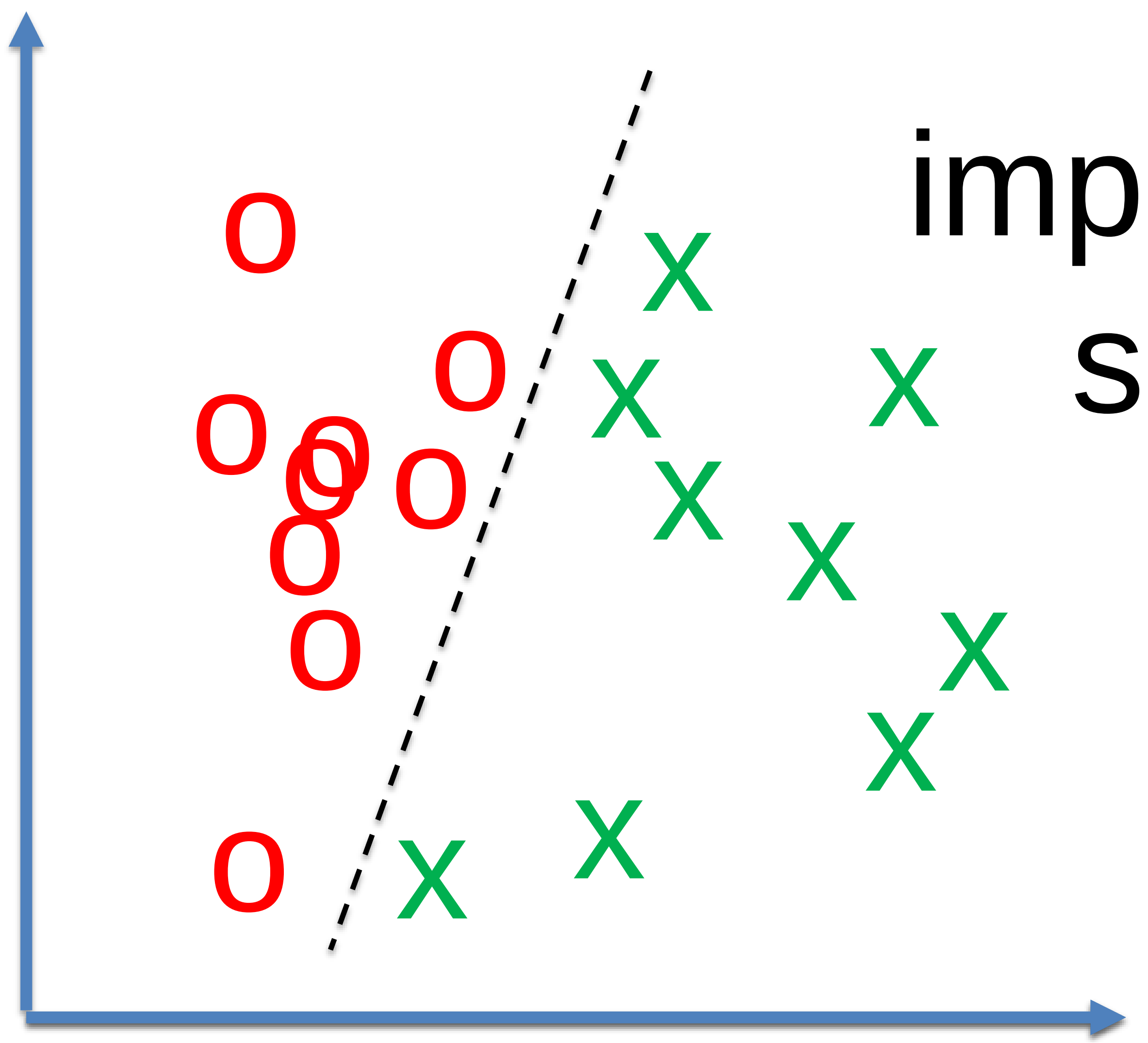
3. Single-Layer networks: simple perceptron

$$\hat{y} \quad = 0.5[1 + sgn(\sum_k w_k\, x_k - \vartheta)]$$



vector $\boldsymbol{x}$

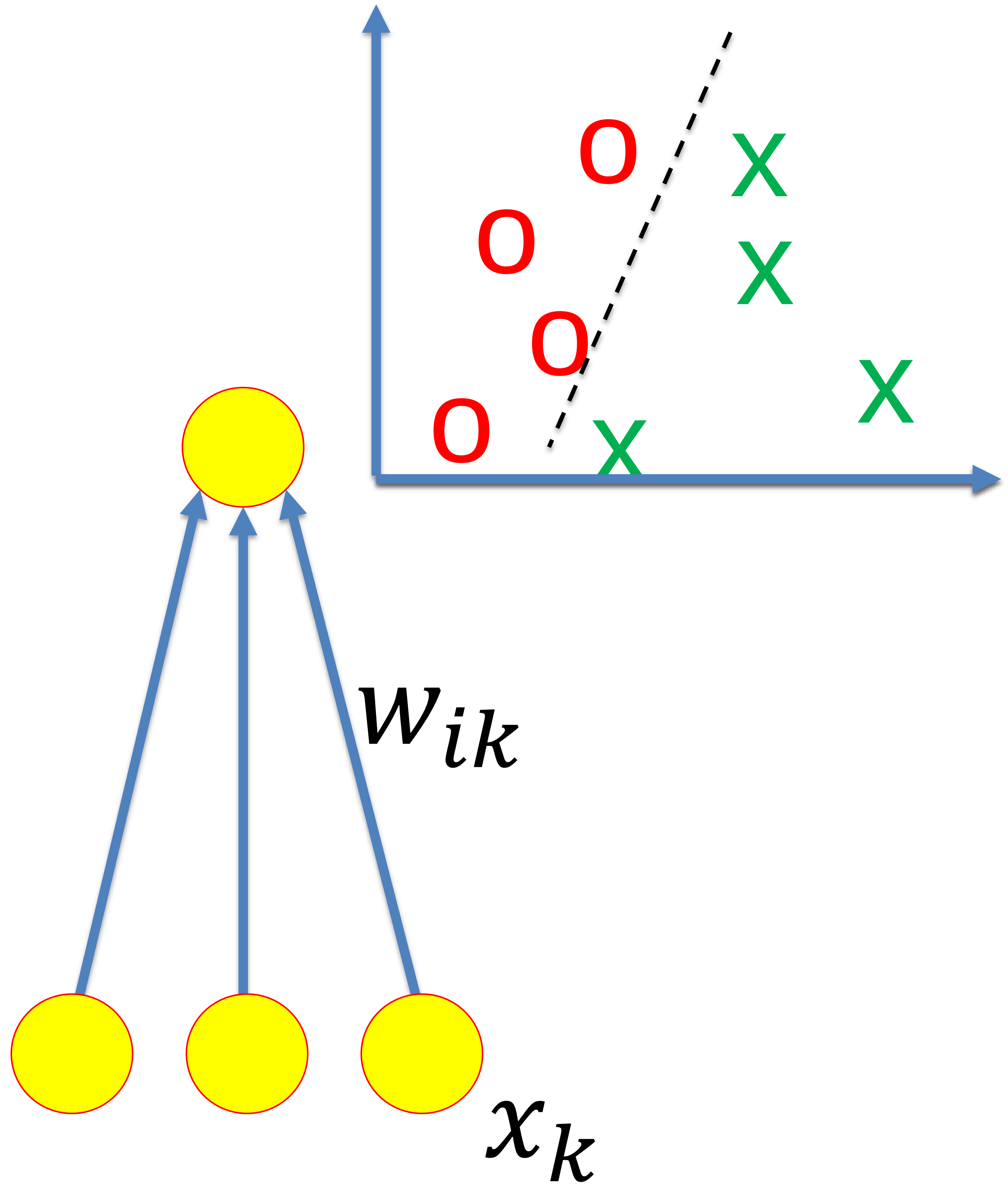
$$d(\boldsymbol{x}) = \sum_k w_k\, x_k - \vartheta = 0$$



imposes a linear separation

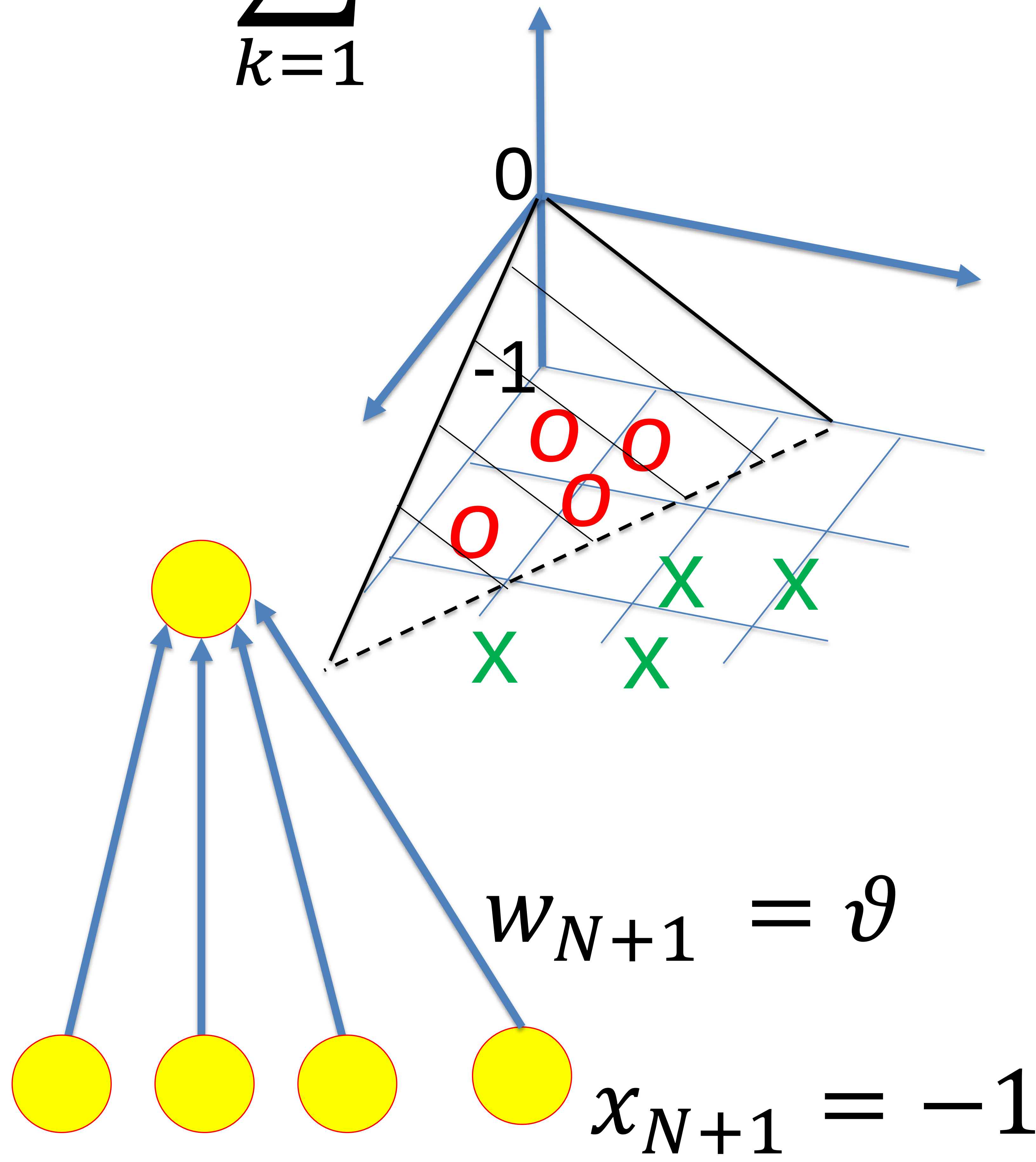
3. remove threshold: add a constant input

$$d(\boldsymbol{x}) = \sum_{k=1}^{N} w_k\, x_k - \vartheta \;= 0$$



$\boldsymbol{x} \in R^N$

$$d(\boldsymbol{x}) = \sum_{k=1}^{N+1} w_k\, x_k \;= 0$$



$\boldsymbol{x} \in R^{N+1}$

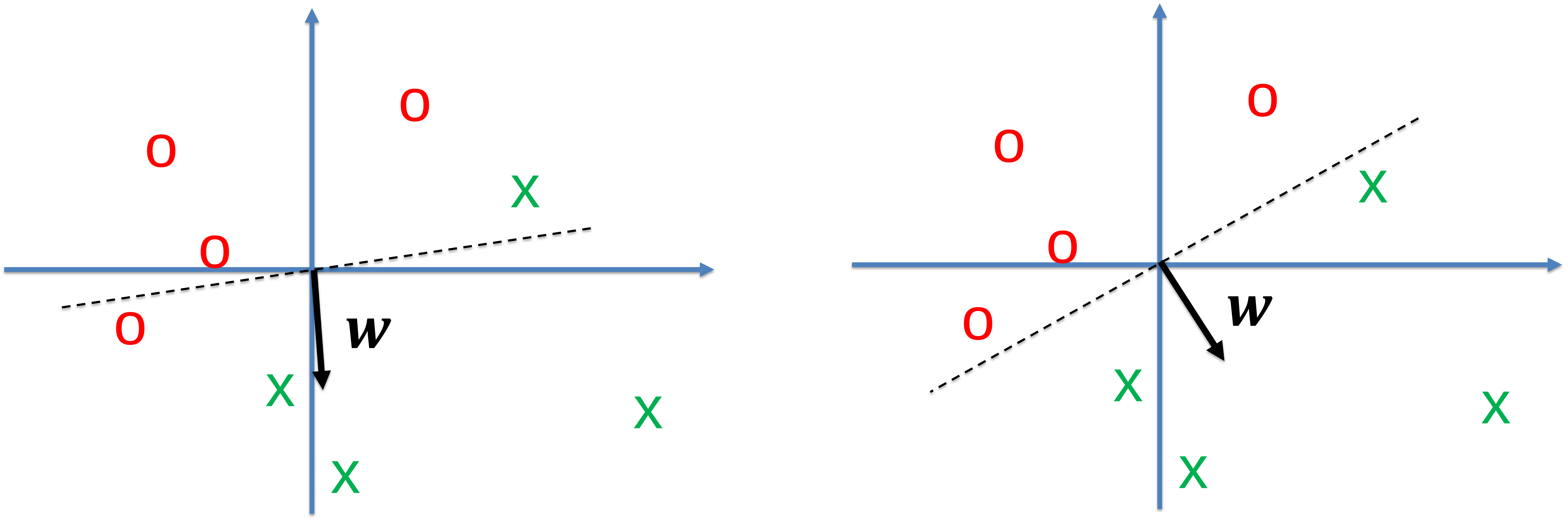
3. Single-Layer networks: simple perceptron

a simple perceptron

- can only solve linearly separable problems
- imposes a separating hyperplane
- for $\vartheta = 0$ hyperplane goes through origin
- threshold parameter ϑ can be removed by adding an input dimension
- in ***N+1*** dimensions hyperplane always goes through origin
- we can **adapt the weight vector** to the problem: this is called ‘learning’

4. Perceptron algorithm: turn weight vector (in N+1 dim.)

$$hyperplane:\ d(\boldsymbol{x}) = \sum_{k=1}^{N+1} w_k\, x_k = \boldsymbol{w}^T \boldsymbol{x} \ = 0$$



4. Perceptron algorithm: turn weight vector

Blackboard 4:
geometry of perceptron algo

$\Delta \boldsymbol{w} \sim \boldsymbol{x}^{\mu}$

Perceptron algo (in N+1 dimensions):

- set $\mu = 1$

(1) cycle many times through patterns

- choose pattern μ
- calculate output

$$\hat{y}^{\mu} = 0.5[1 + sgn(\boldsymbol{w}^T \boldsymbol{x}^{\mu})]$$

- update by

$$\Delta \boldsymbol{w} = \gamma [t^{\mu} - \hat{y}^{\mu}] \boldsymbol{x}^{\mu}$$

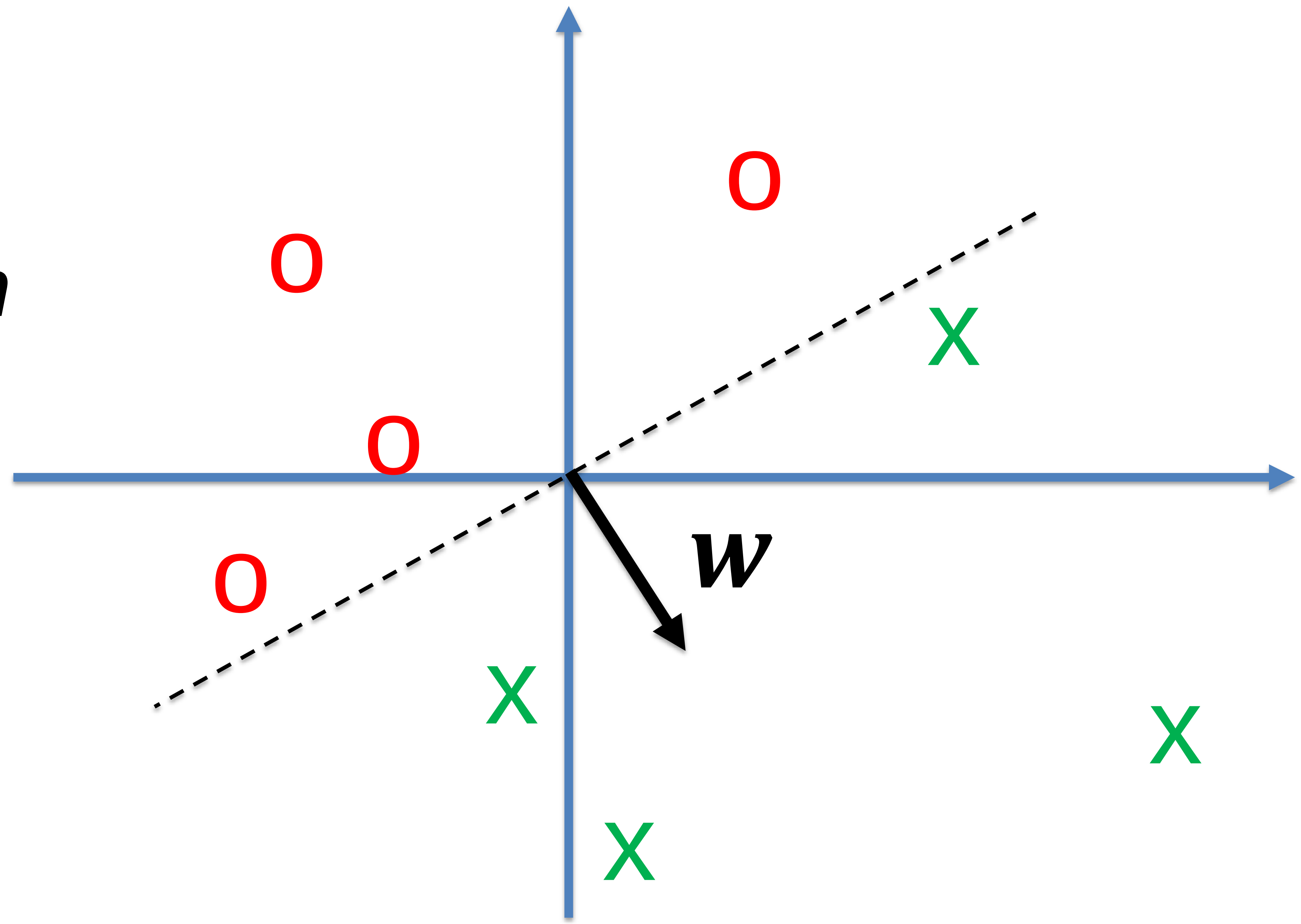
- iterate $\mu \leftarrow (\mu + 1) mod \boldsymbol{P}$, back to (1)

(2) stop if no changes for all ***P*** patterns

4. Perceptron algorithm: theoreom

If the problem is linearly separable, the perceptron algorithm converges in a finite number of steps.

Proof: in many books, e.g.,
Bishop, 1995,
Neural Networks for Pattern Recognition



Quiz: Perceptron algorithm

The **input vector has *N* dimensions** and we apply a perceptron algorithm.

.

[ ] a rotation of the hyperplane in N+1 dimensions implies a change of weight vector.

[ ] An increase of the length of the weight vector implies that the hyperplane does not change in N+1 dimensions

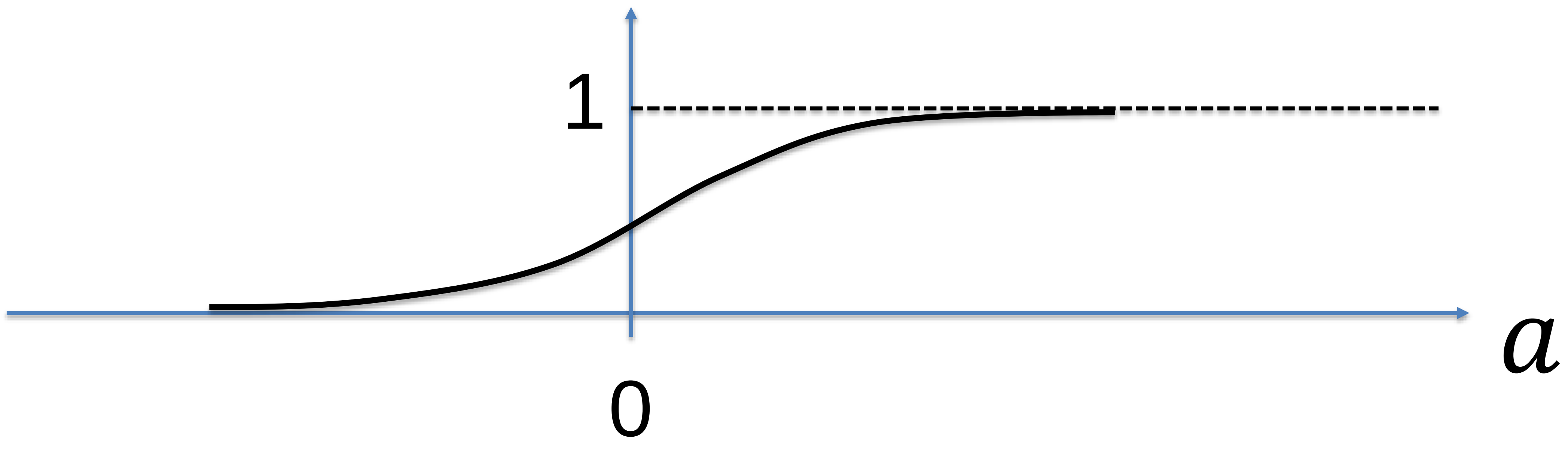
5. Sigmoidal output unit

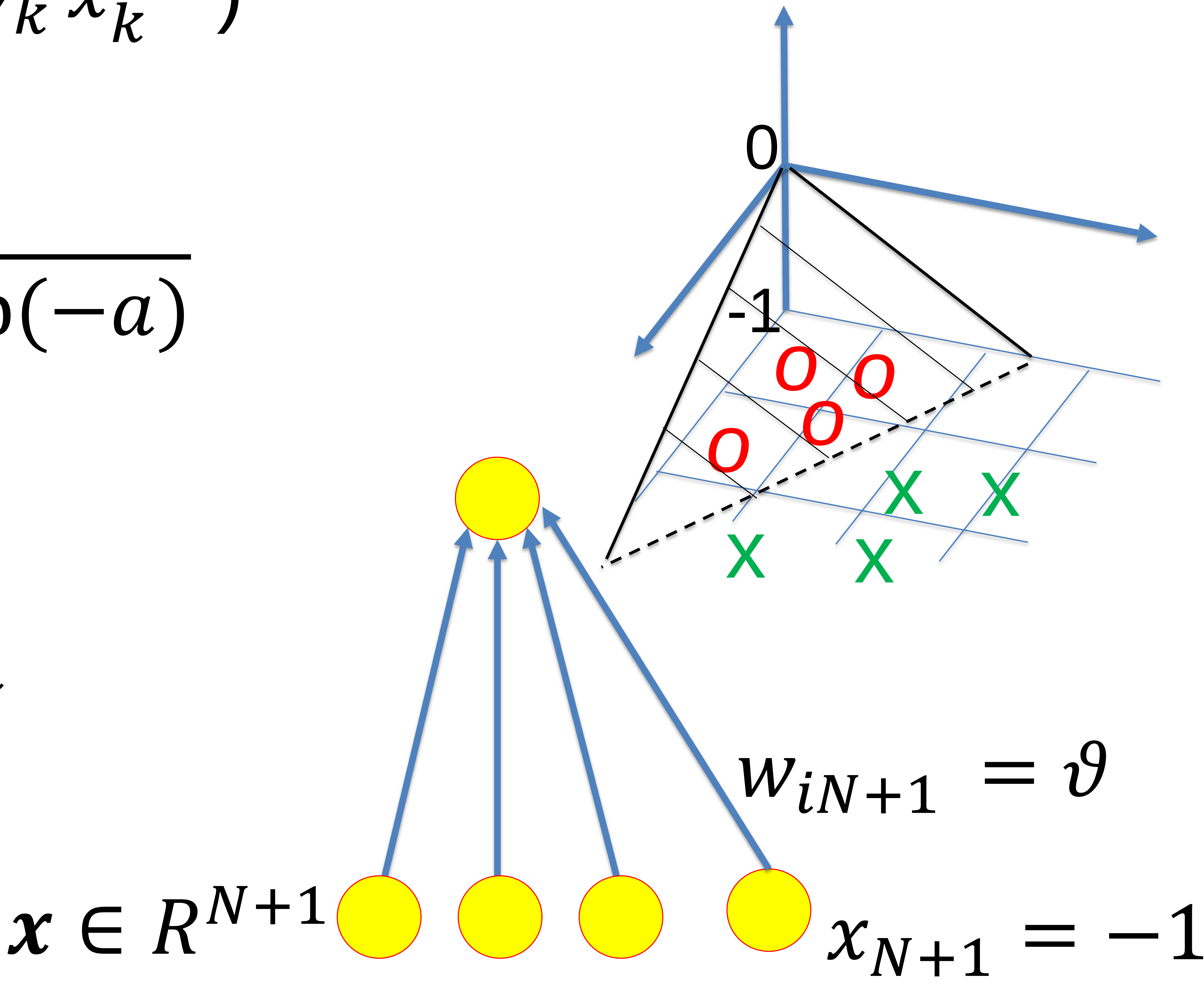
A saturating nonlinear function with a smooth transition from 0 to 1.

$$\hat{y}^{\mu} = g(\boldsymbol{w}^T \boldsymbol{x}^{\mu}) = g\left(\sum_{k=1}^{N+1} w_k \, x_k^{\mu}\right)$$

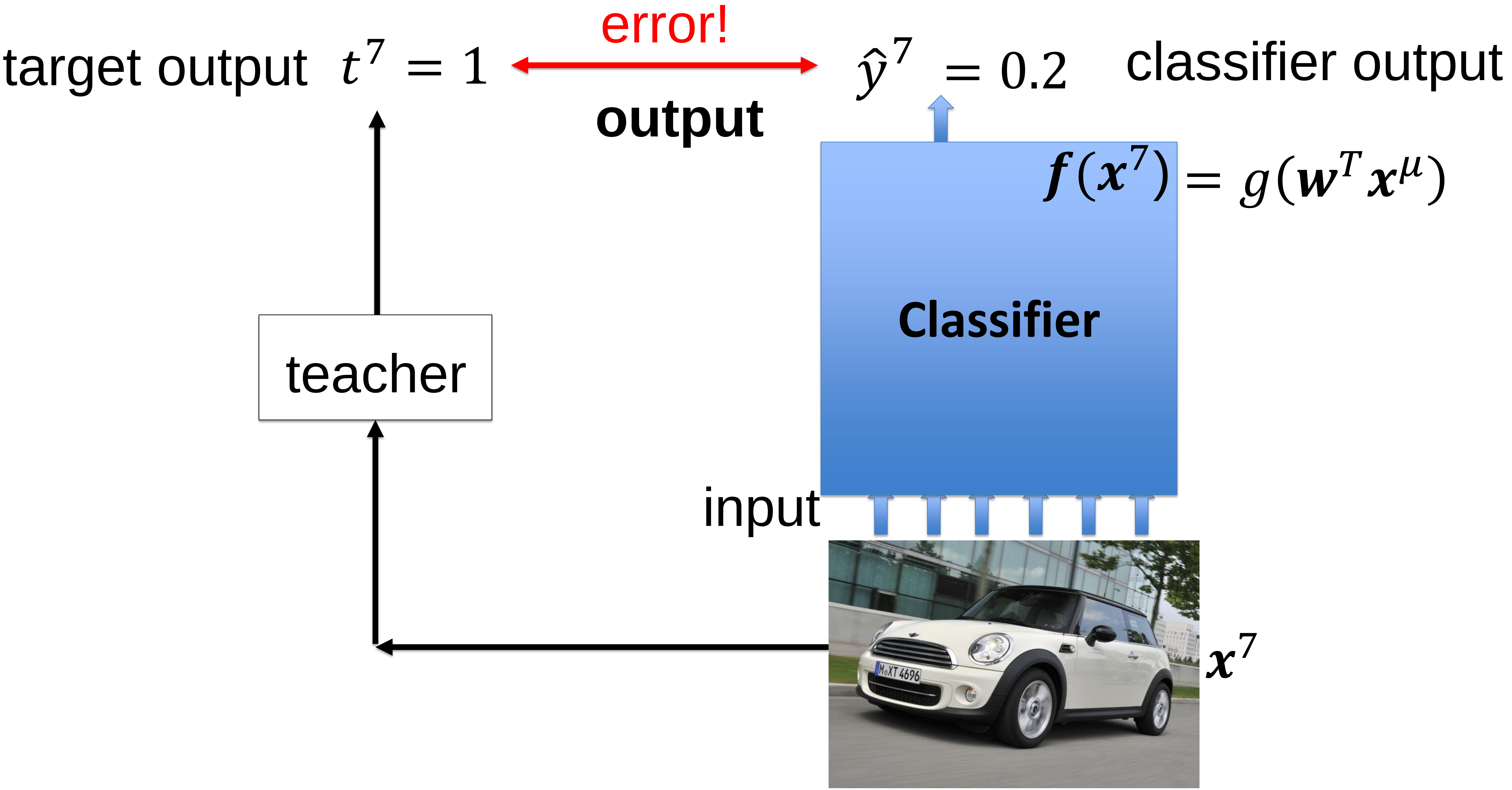
with

$$g(a) = \frac{\exp(a)}{1 + \exp(a)} = \frac{1}{1 + \exp(-a)}$$





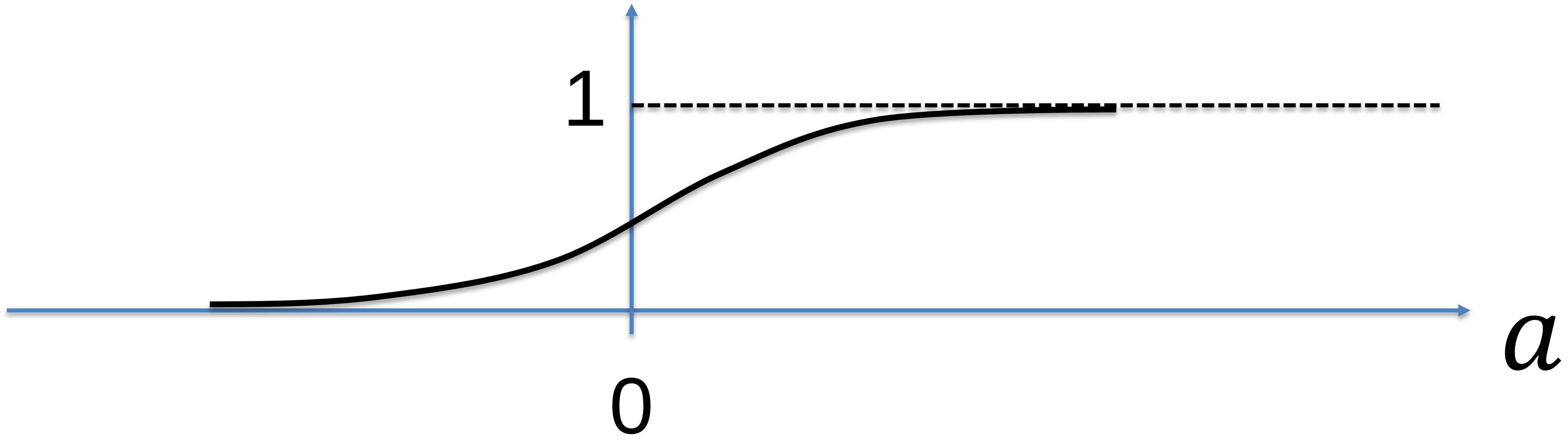
5. Supervised learning with sigmoidal output



5. Supervised learning with sigmoidal output

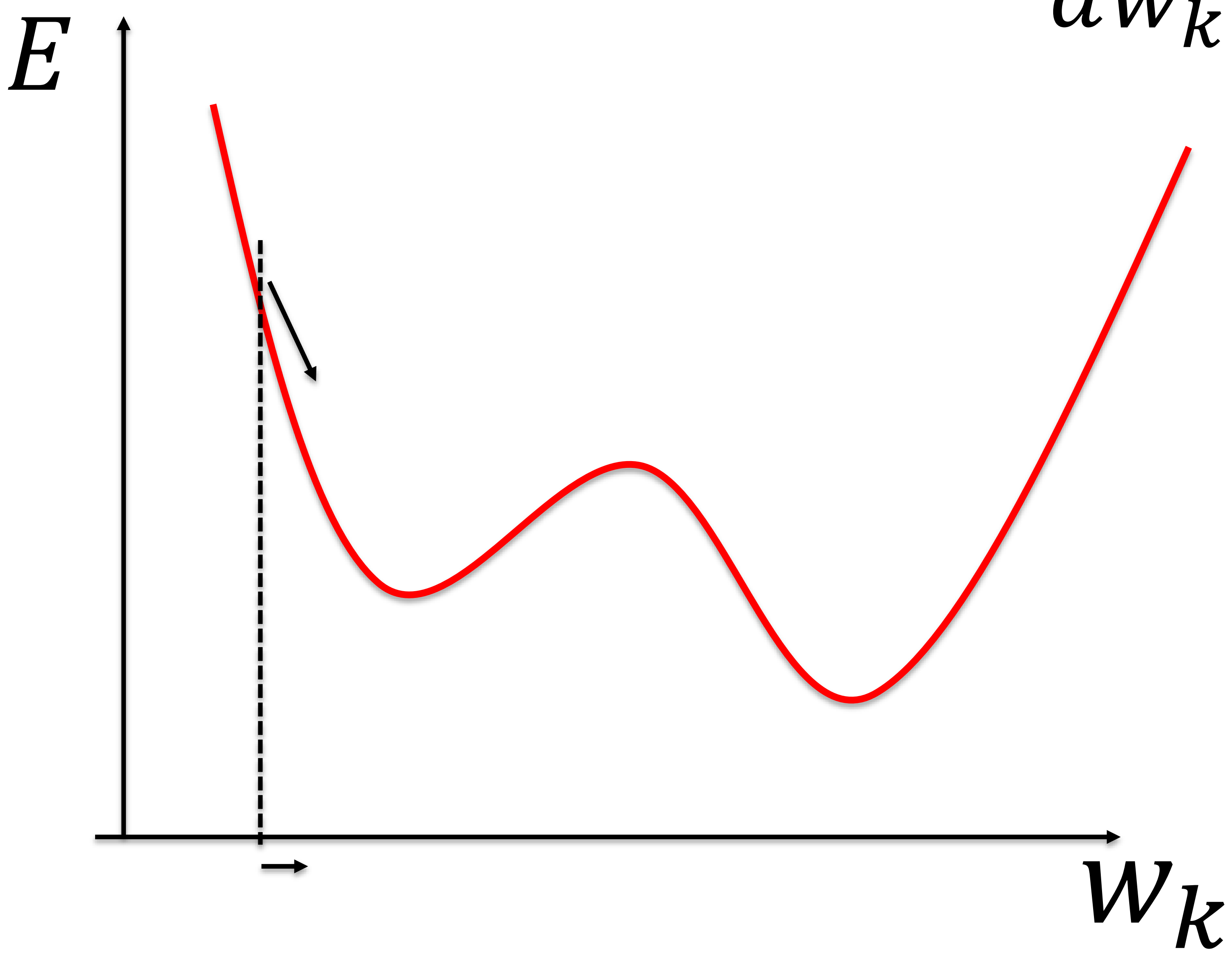
define **error**

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{\mu=1}^{P}\left[t^{\mu} - \hat{y}^{\mu}\right]^2$$

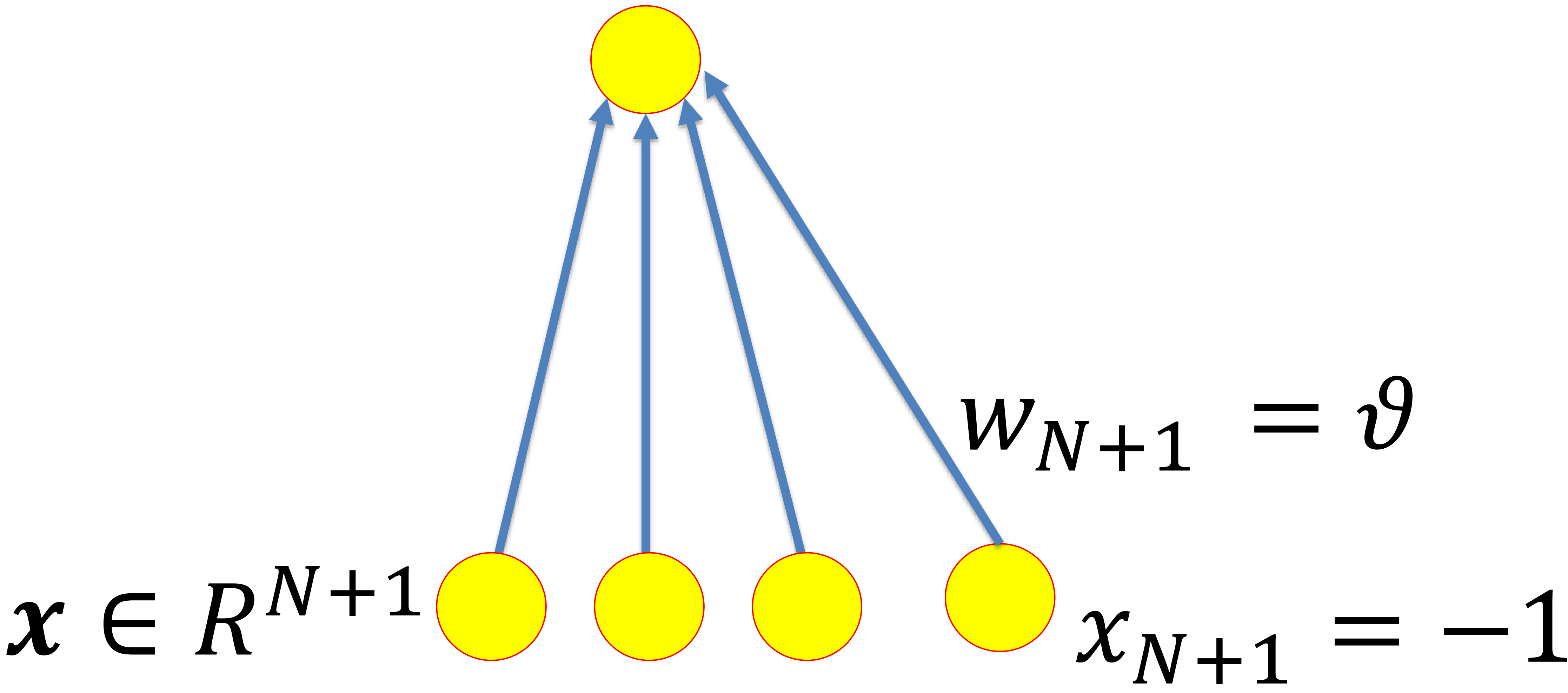


gradient descent

$$\Delta w_k = -\gamma \frac{dE}{dw_k}$$



$$\hat{y}^{\mu} = g(\boldsymbol{w}^T \boldsymbol{x}^{\mu})$$



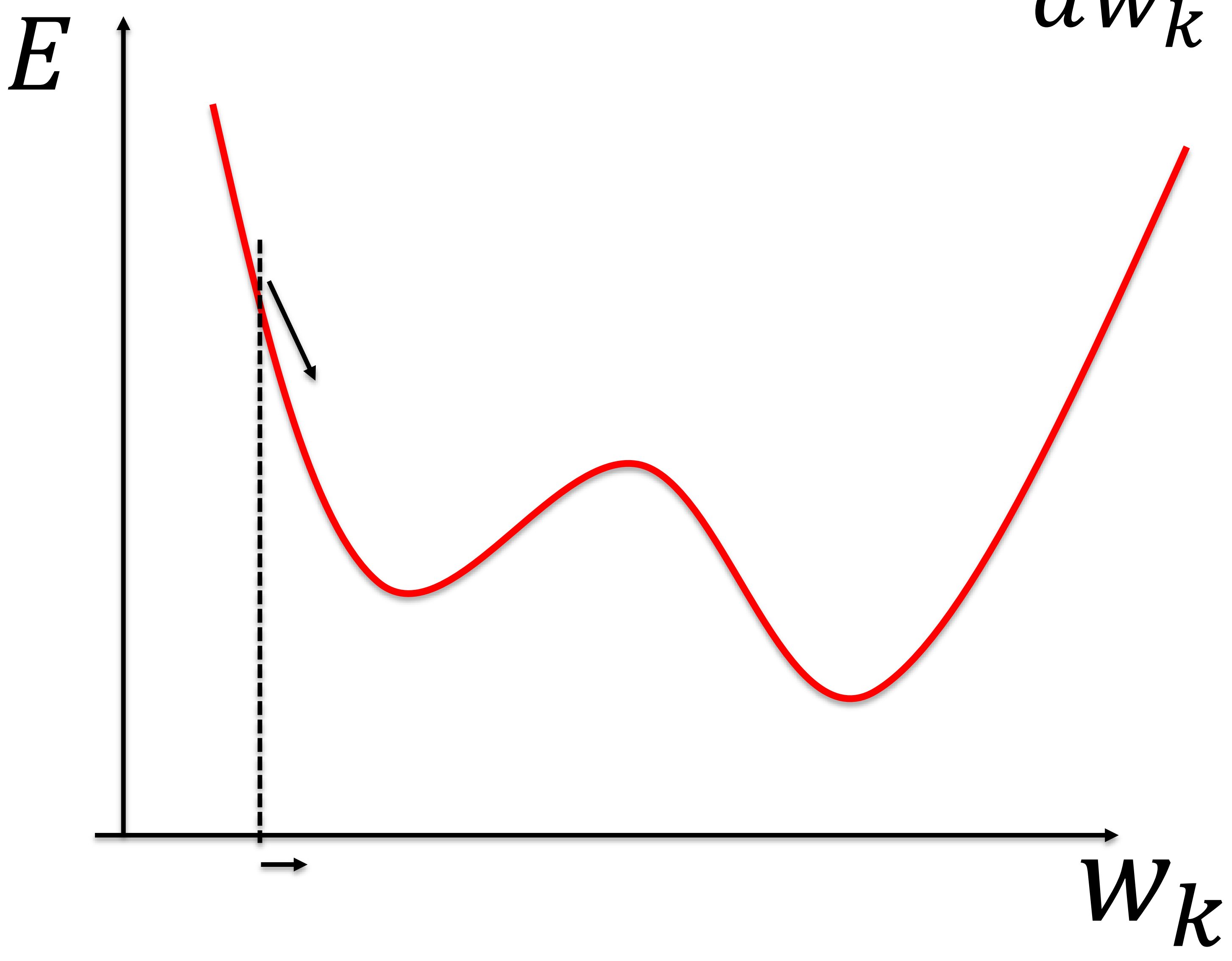
6. gradient descent

Quadratic **error**

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{\mu=1}^{P}\left[t^{\mu} - \hat{y}^{\mu}\right]^{2}$$

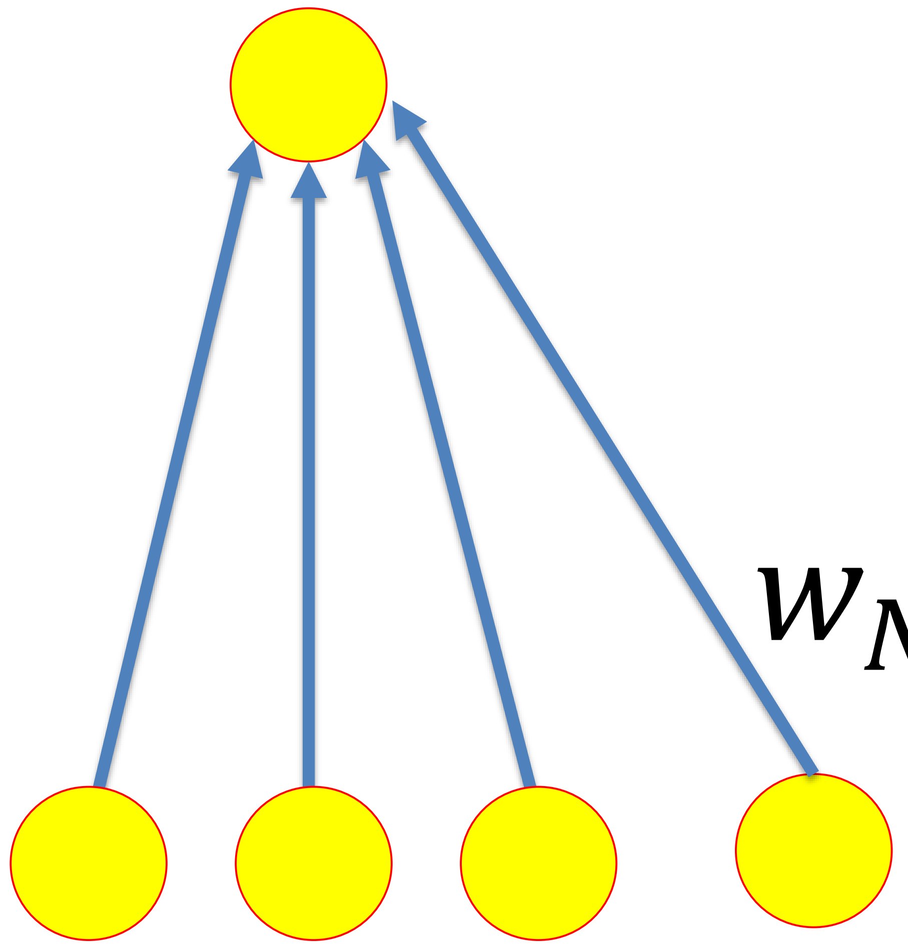
gradient descent

$$w_k = -\gamma \frac{dE}{dw_k}$$



Exercise 1.1 now:
- *calculate gradient (1 pattern)*
- *geometrical interpretation?*

$$\hat{y}^{\mu} = g(\boldsymbol{w}^{T}\boldsymbol{x}^{\mu})$$



$$w_{N+1} = \vartheta$$

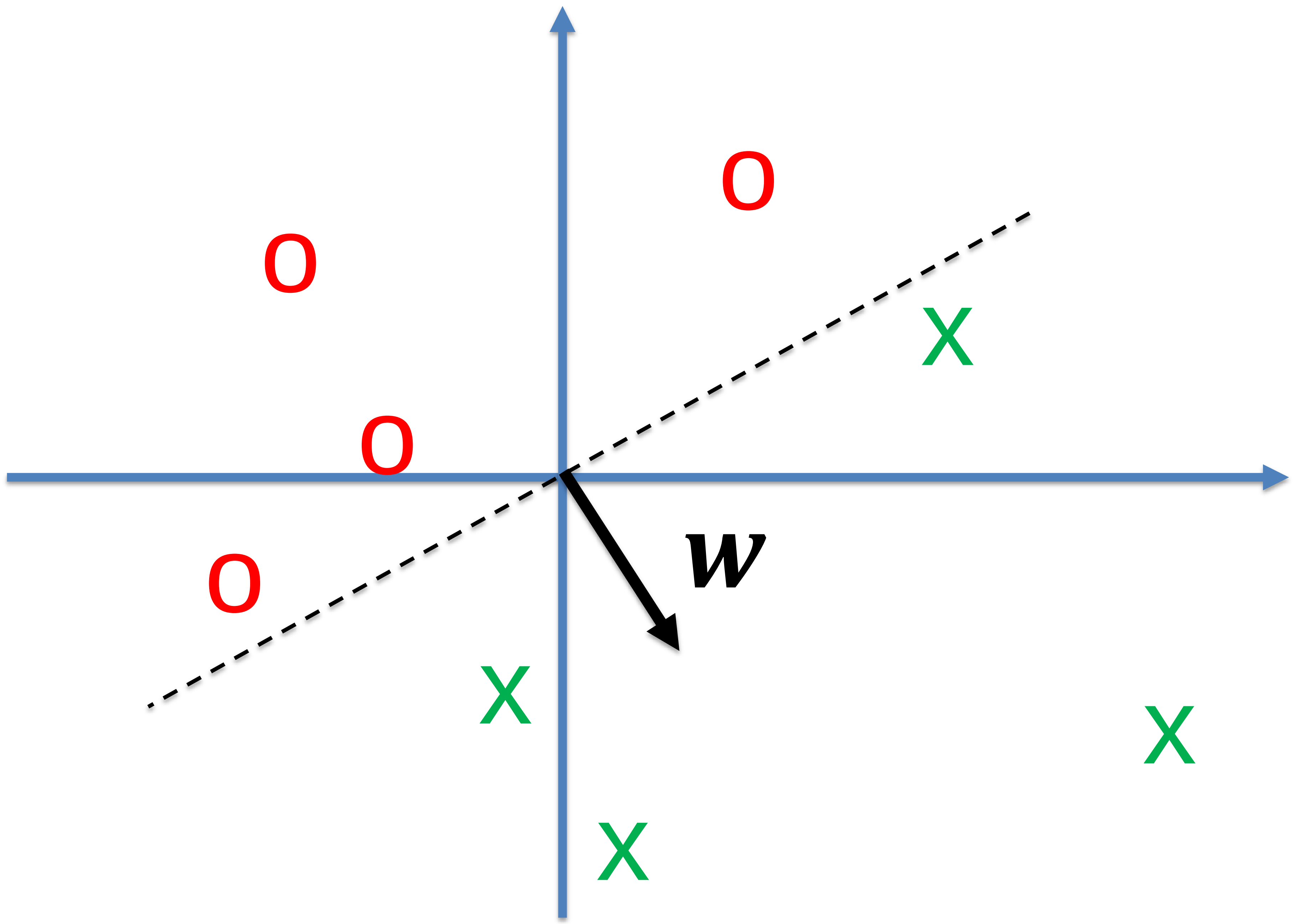
$$\boldsymbol{x} \in R^{N+1} \qquad x_{N+1} = -1$$

6. Gradient descent algorithm

After presentation of pattern $\boldsymbol{x}^\mu$ update the weight vector by

$$\Delta \boldsymbol{w} = \gamma \delta(\mu) \boldsymbol{x}^\mu$$

- amount of change depends on $\delta(\mu)$, i.e., the (signed) output mismatch for this data point
- change implemented even if 'correctly' classified
- change proportional to $\boldsymbol{x}^\mu$
- compare with perceptron algorithm



Learning outcome and conclusions for today:

- understand classification as a geometrical problem
- discriminant function of classification
- linear versus nonlinear discriminant function
- perceptron algorithm
- gradient descent for simple perceptrons
- learning as rotation of a hyperplane