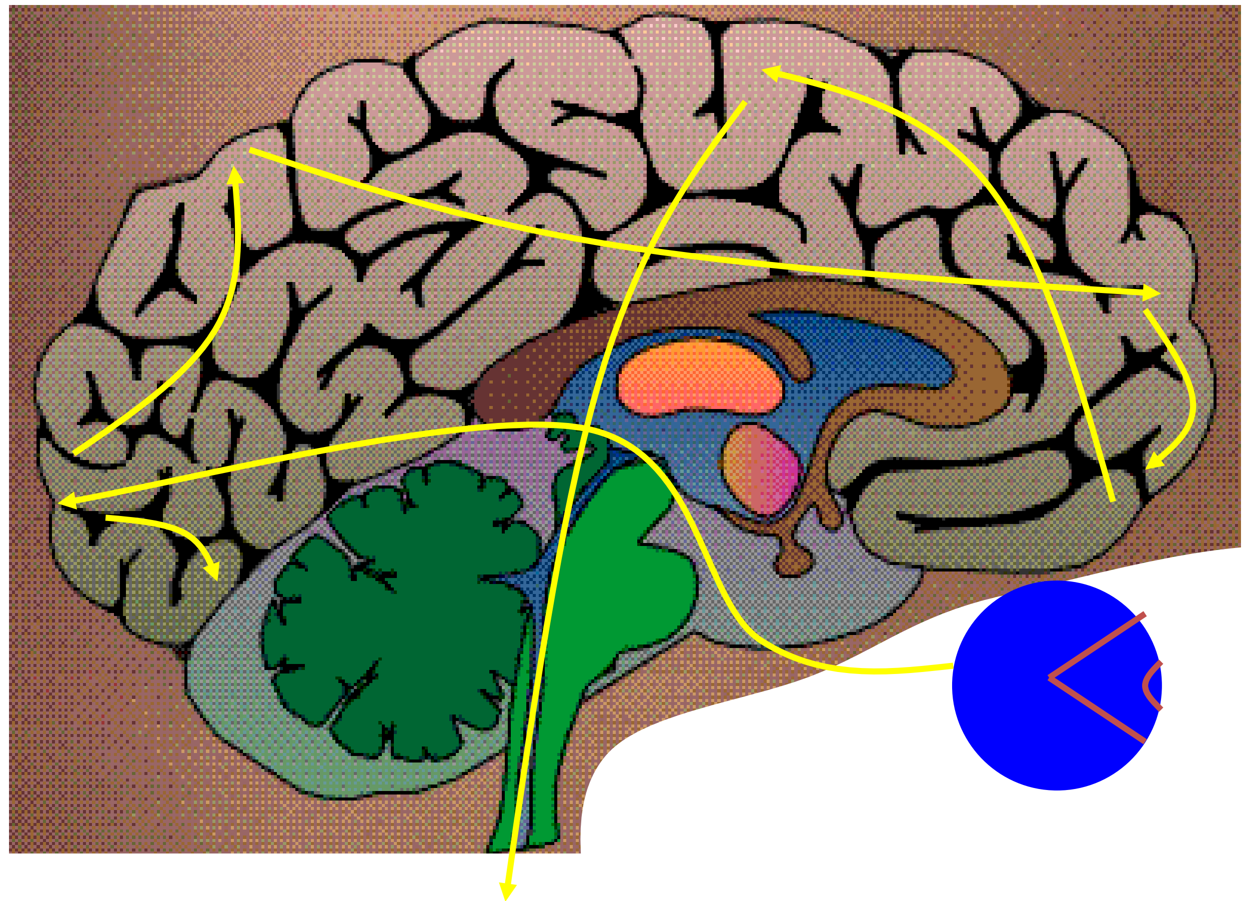
Artificial Neural Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

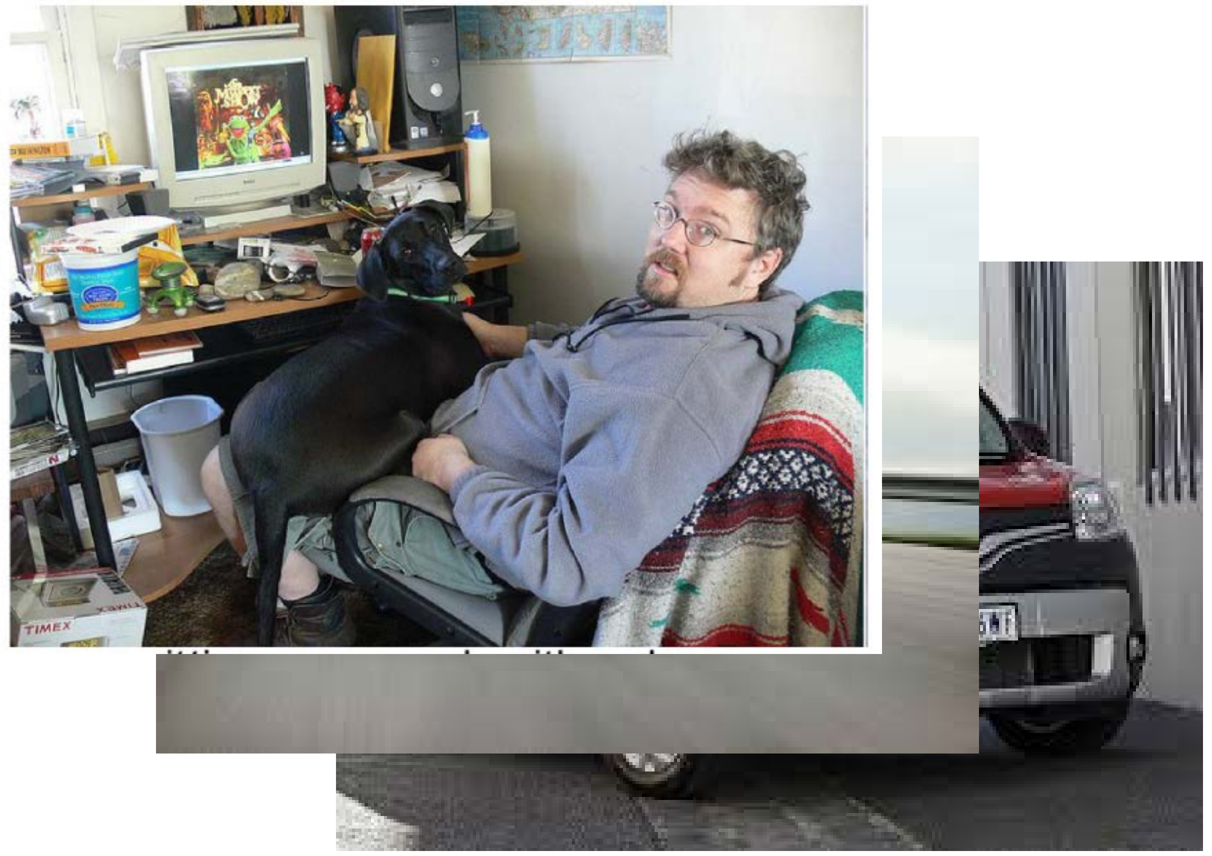
The brain: Cortical Areas

motor
cortex



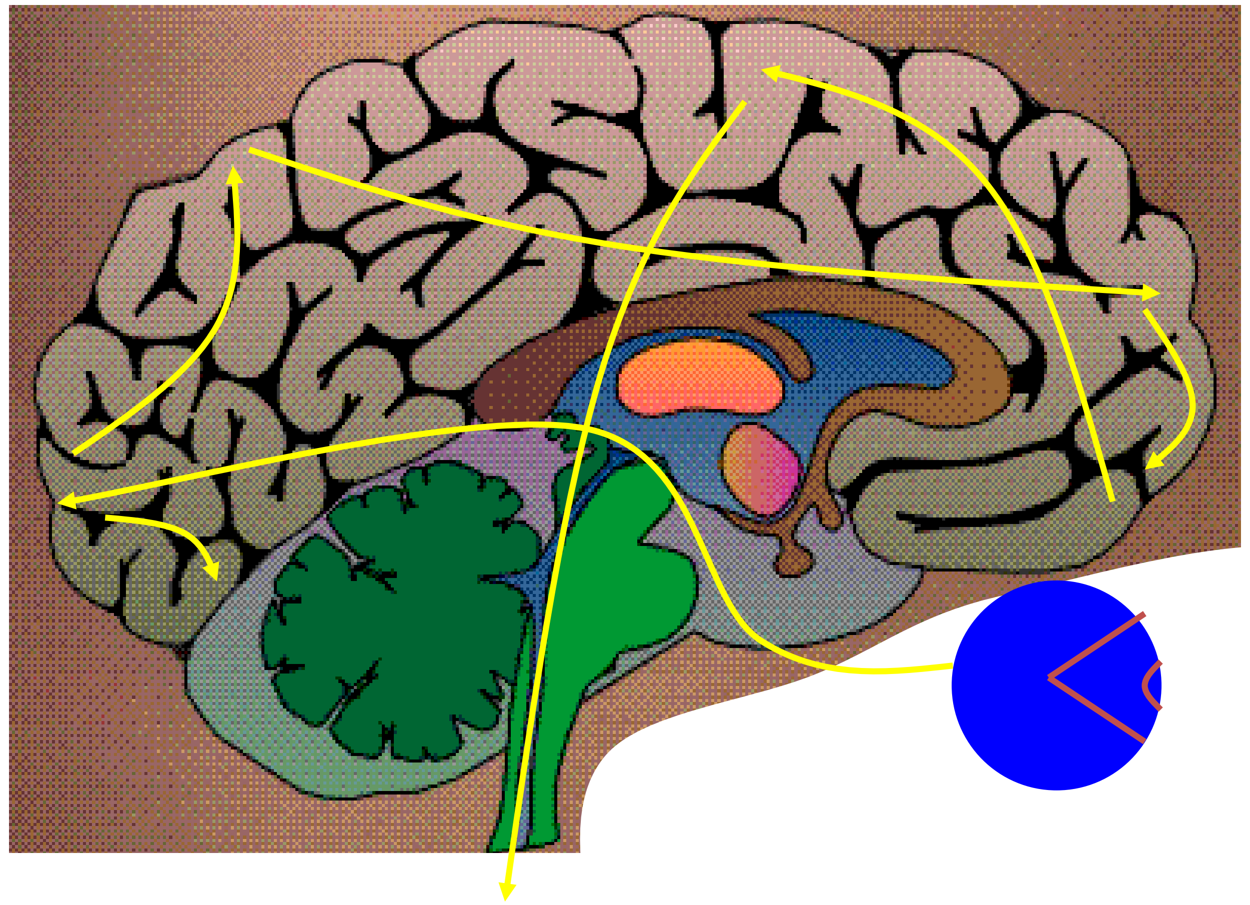
visual
cortex

frontal cortex



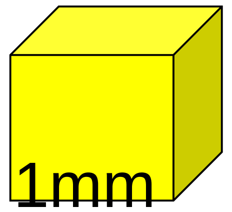
to
muscles

The brain: Cortical Areas



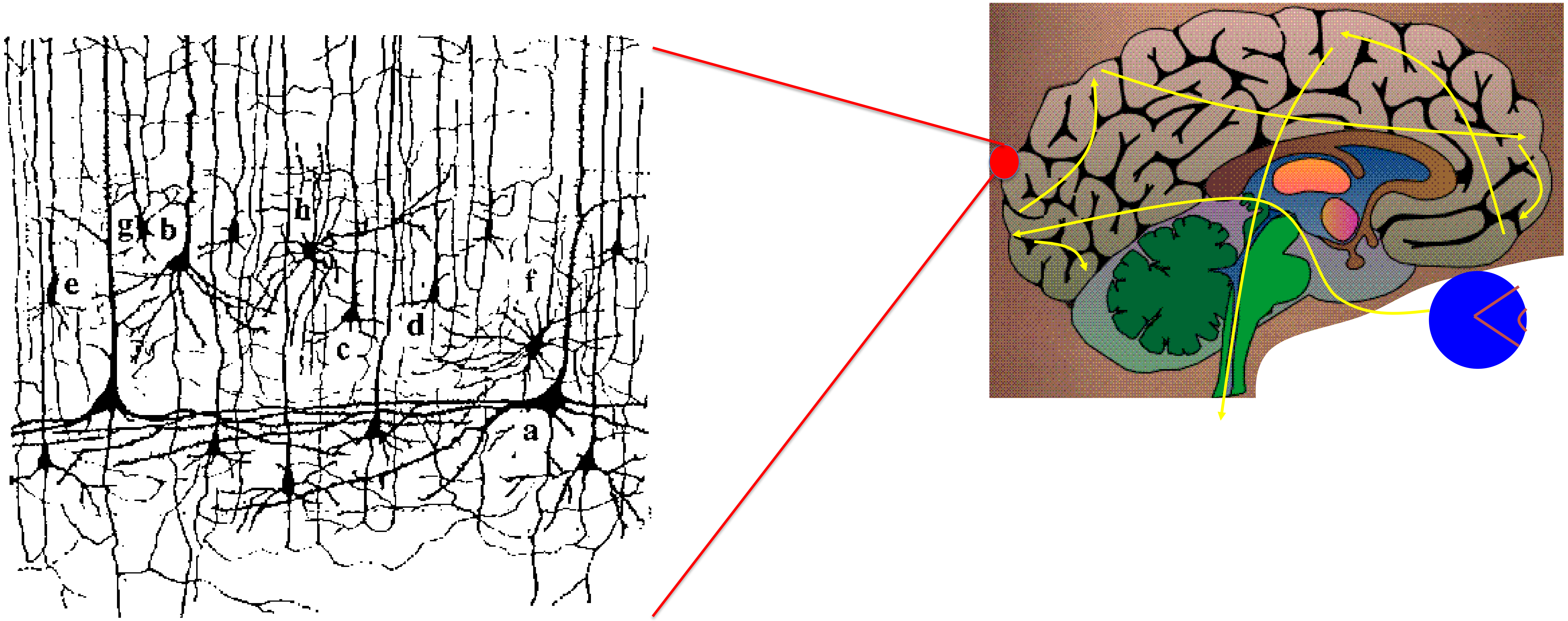
The Brain: zooming in

1mm



10 000 neurons

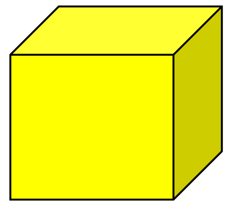
3 km of wire



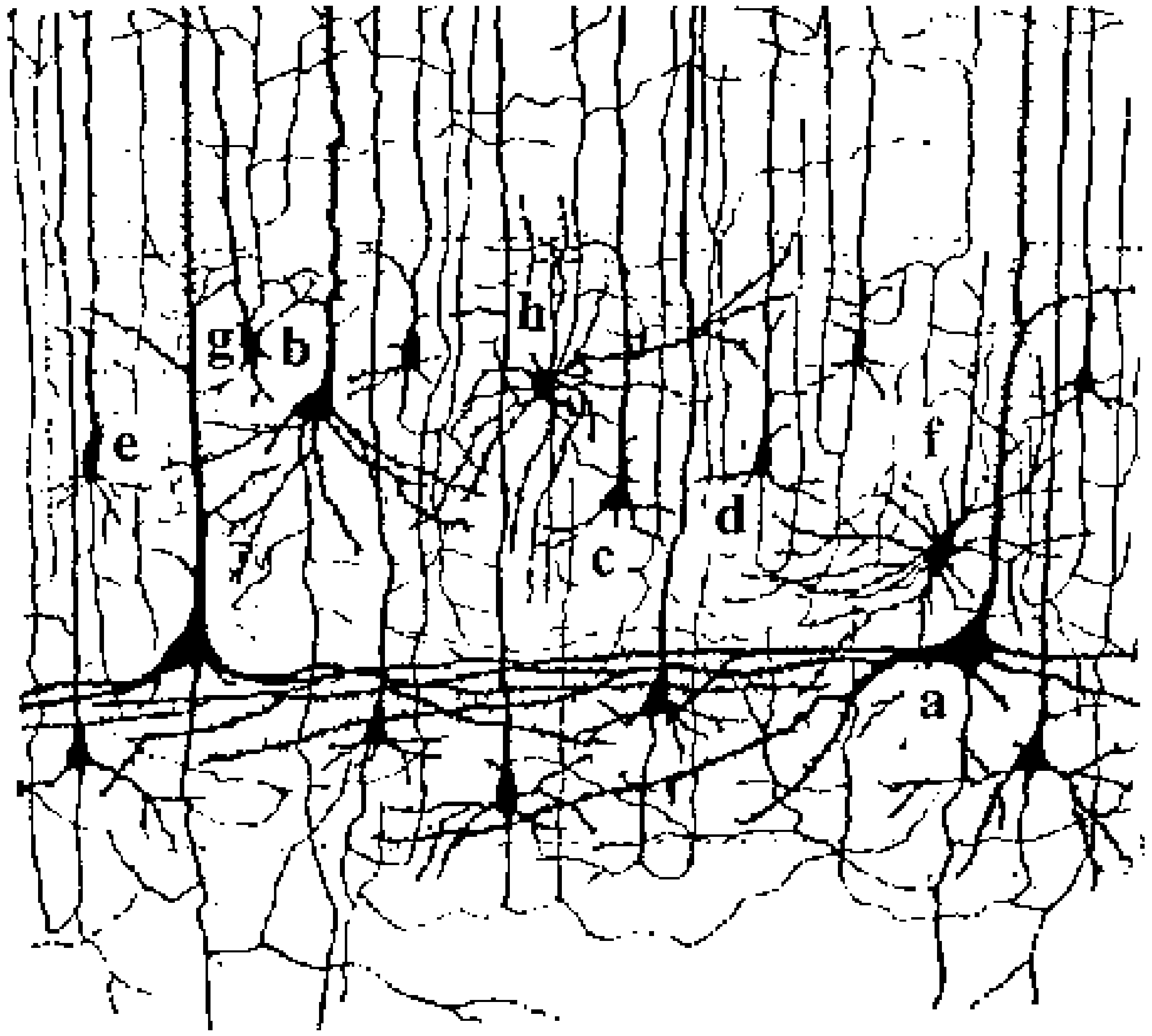
Ramon y Cajal

The brain: a network of neurons

1mm

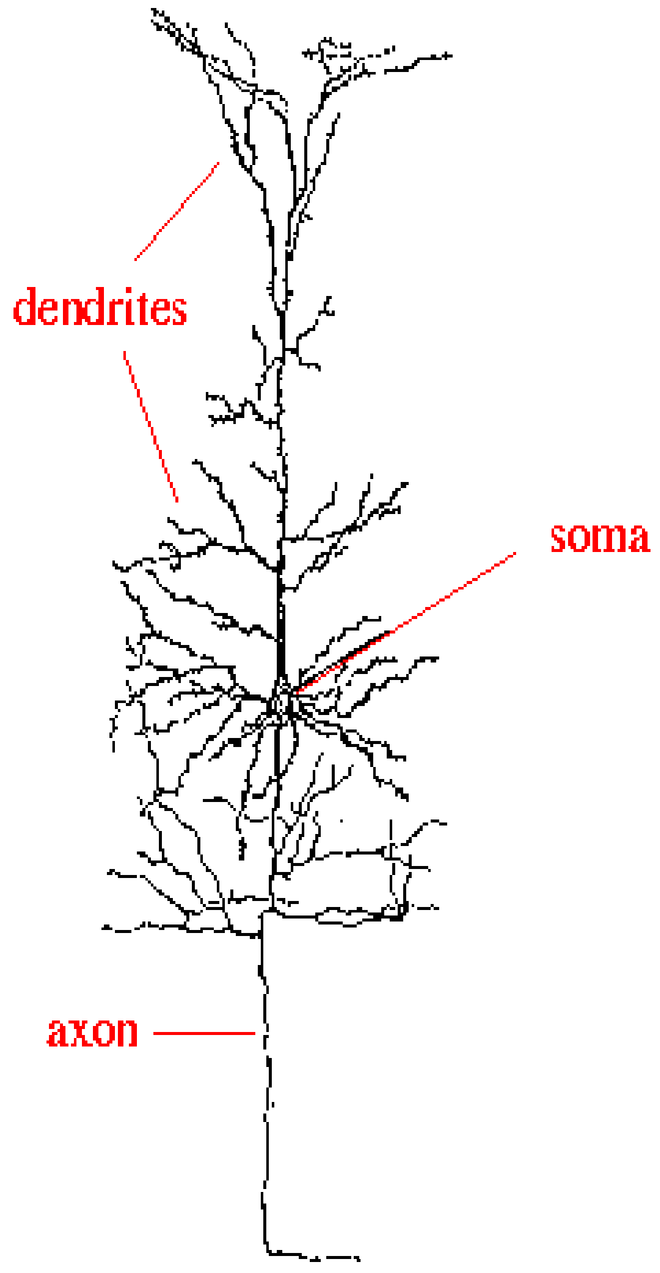


10 000 neurones
3 km de cablage

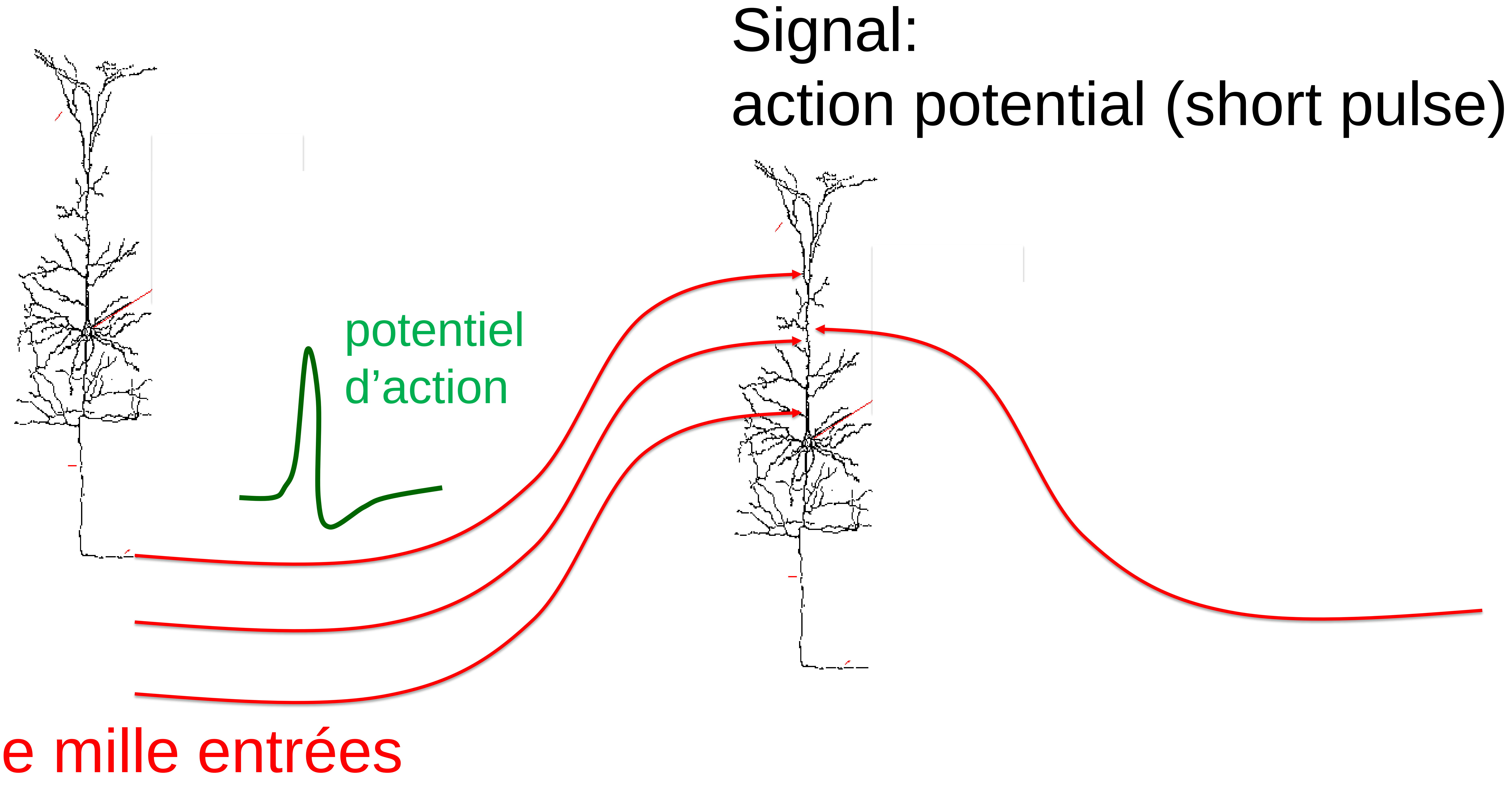


Ramon y Cajal

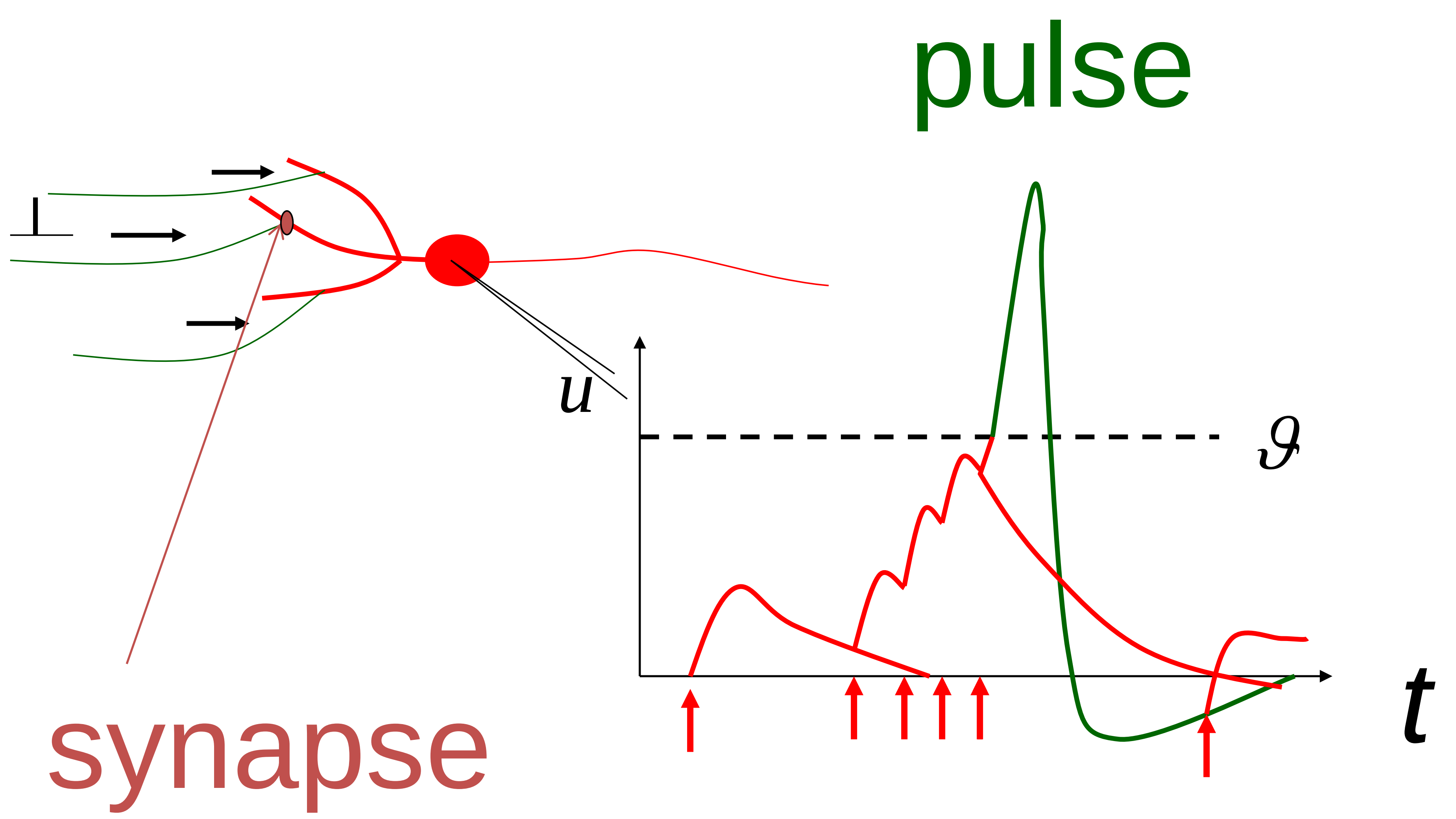
Signal:
Potentiel d’action (impulsion)



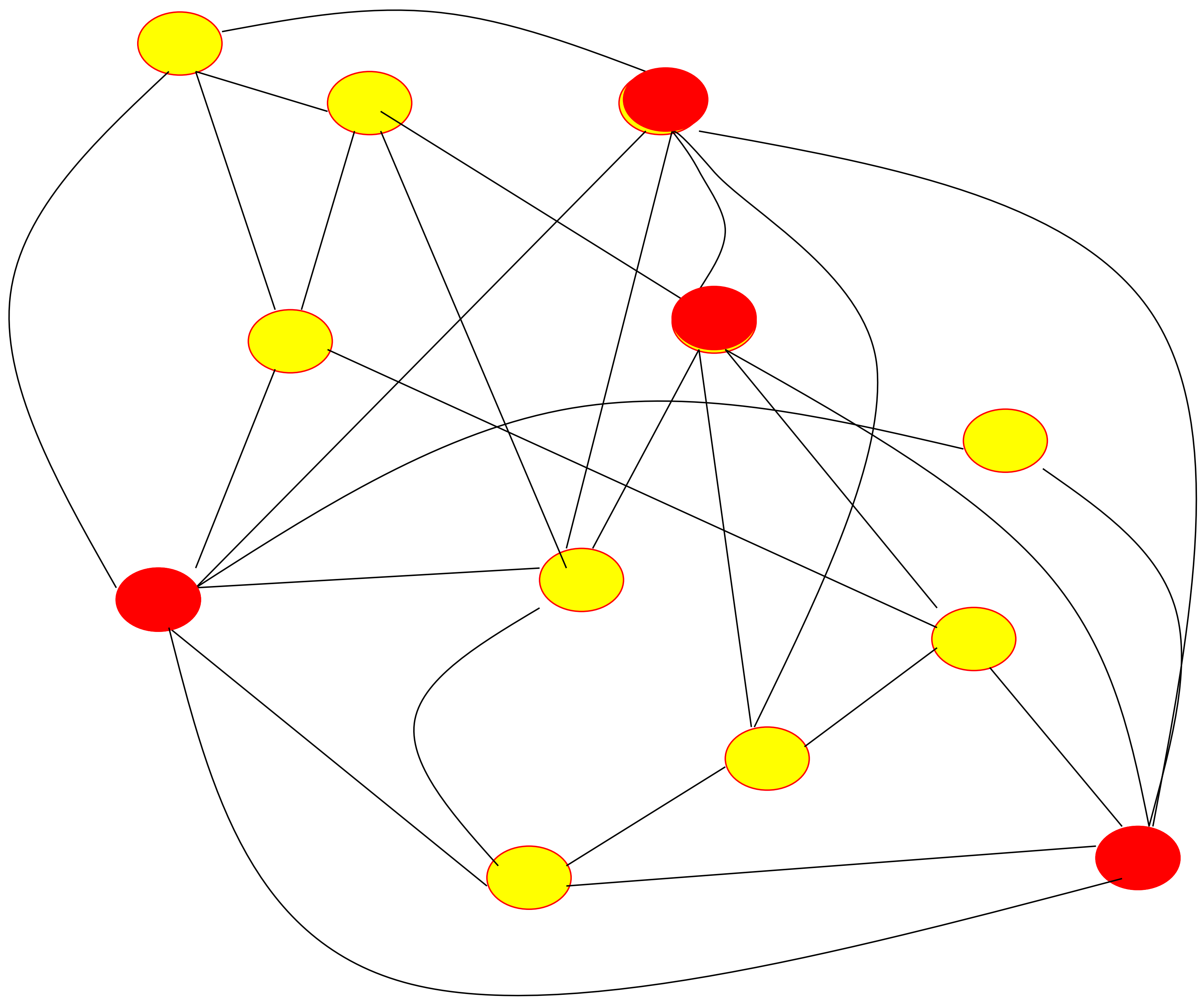
The brain: signal transmission



The brain: neurons sum their inputs

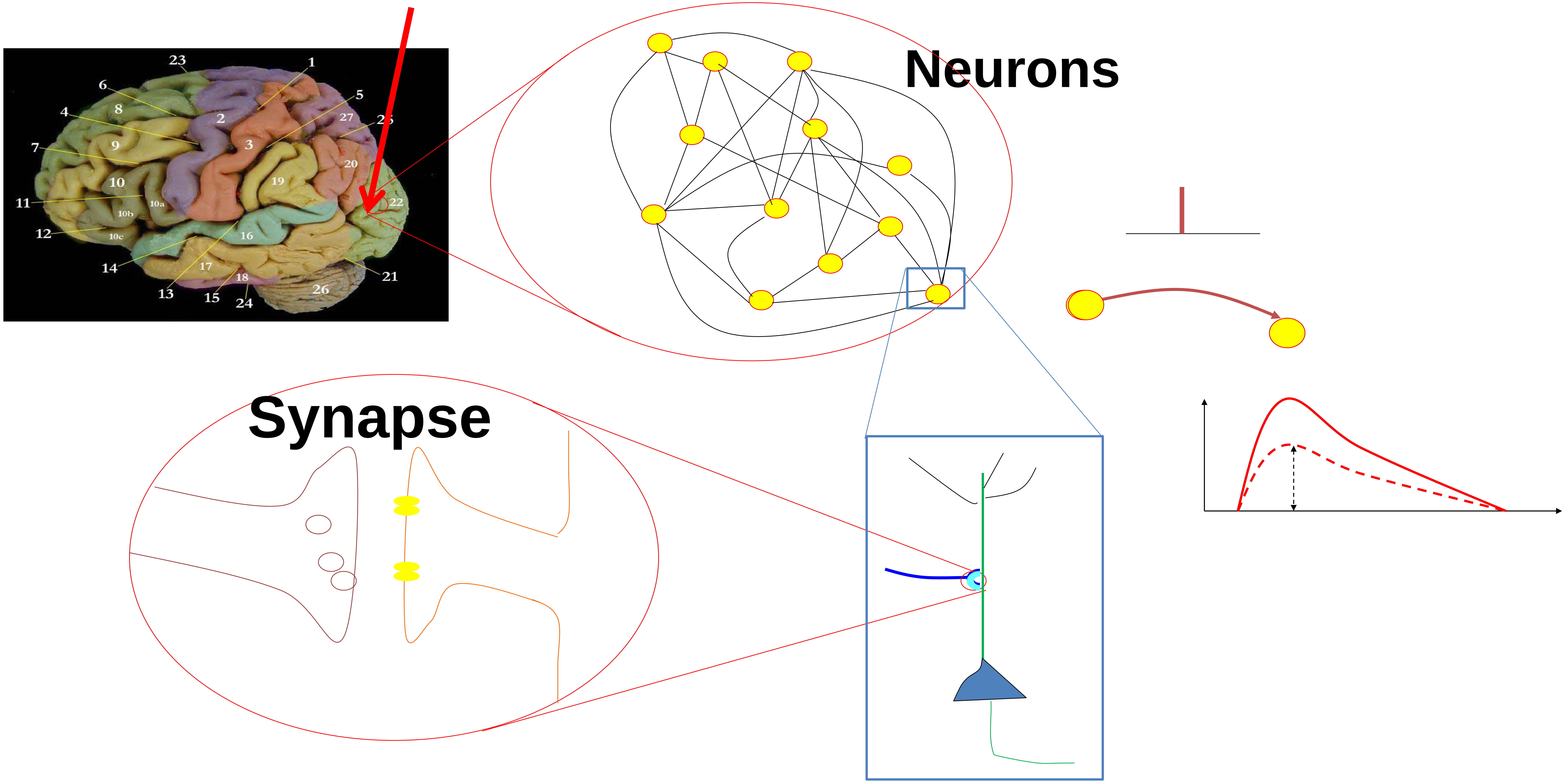


Summary: the brain is a large network of neurons



Active neuron

Learning in the brain: changes between connections



learning = change of connection

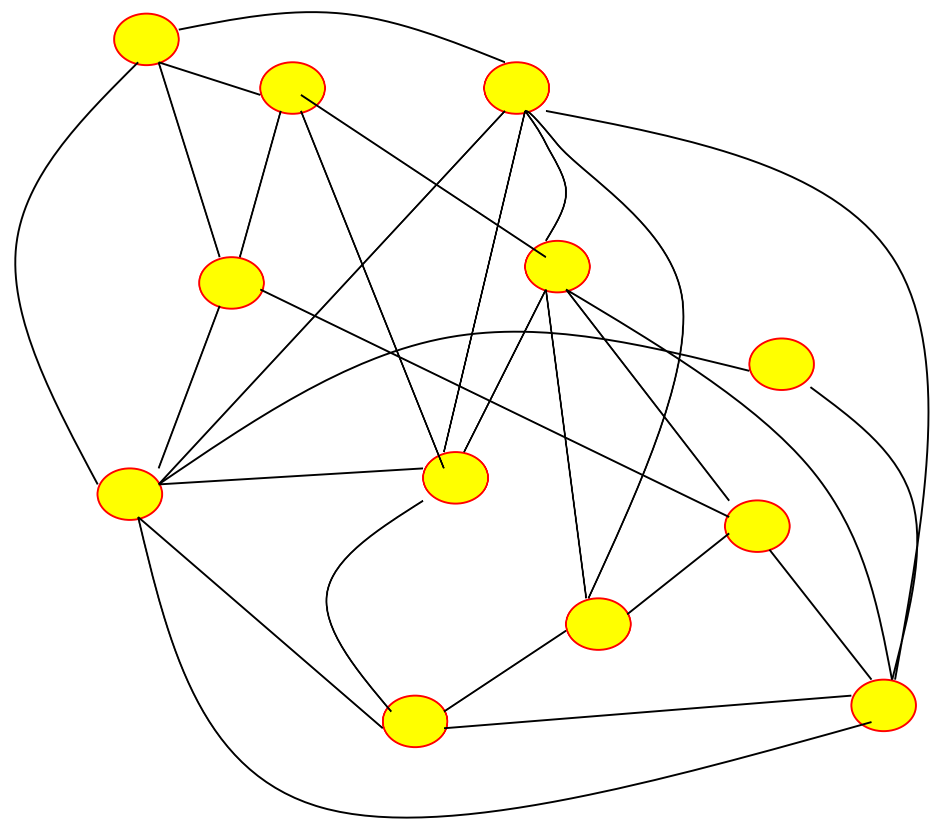
Artificial Neural Networks

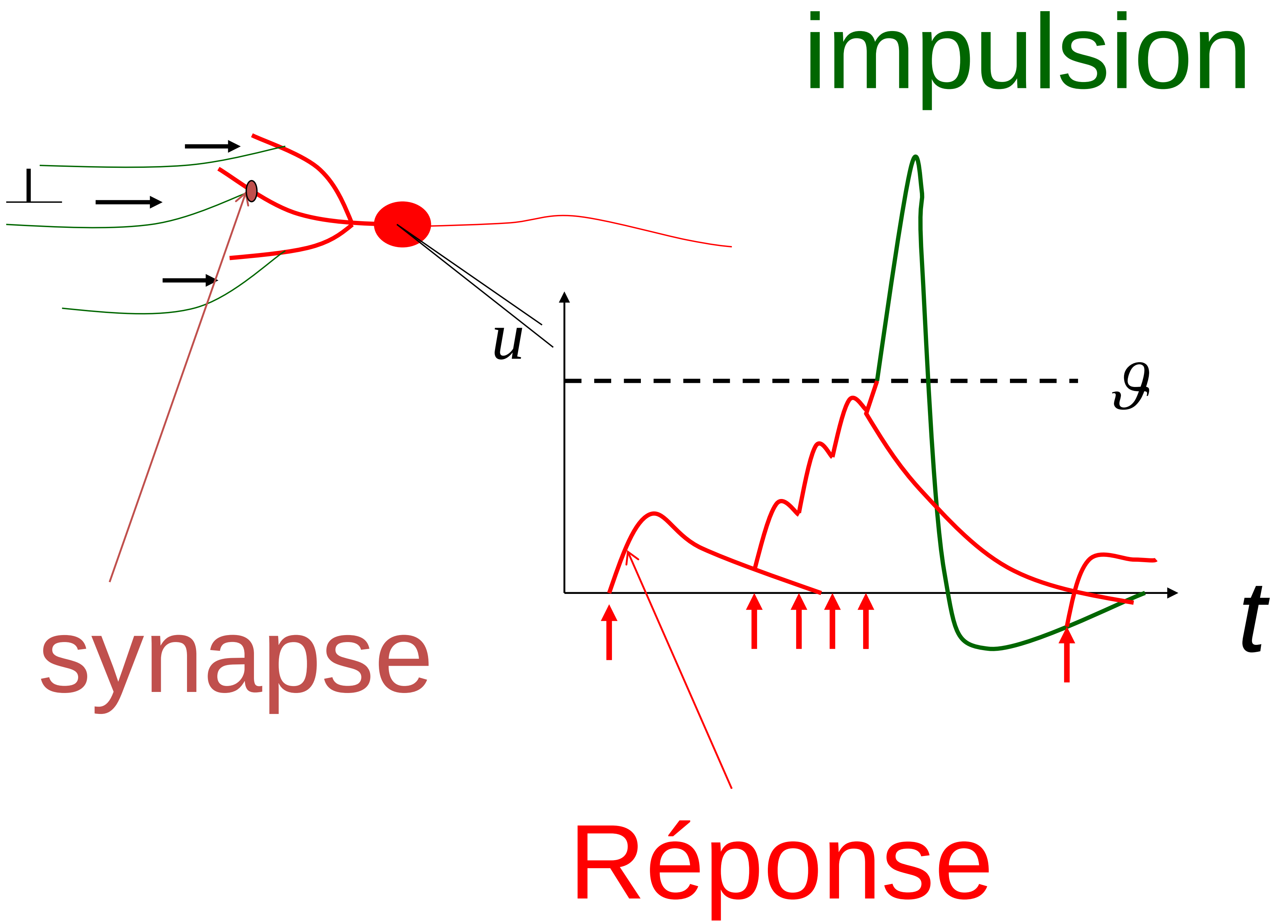
Wulfram Gerstner

EPFL, Lausanne, Switzerland

1. The brain
2. Artificial Neural Networks

Modeling: artificial neurons





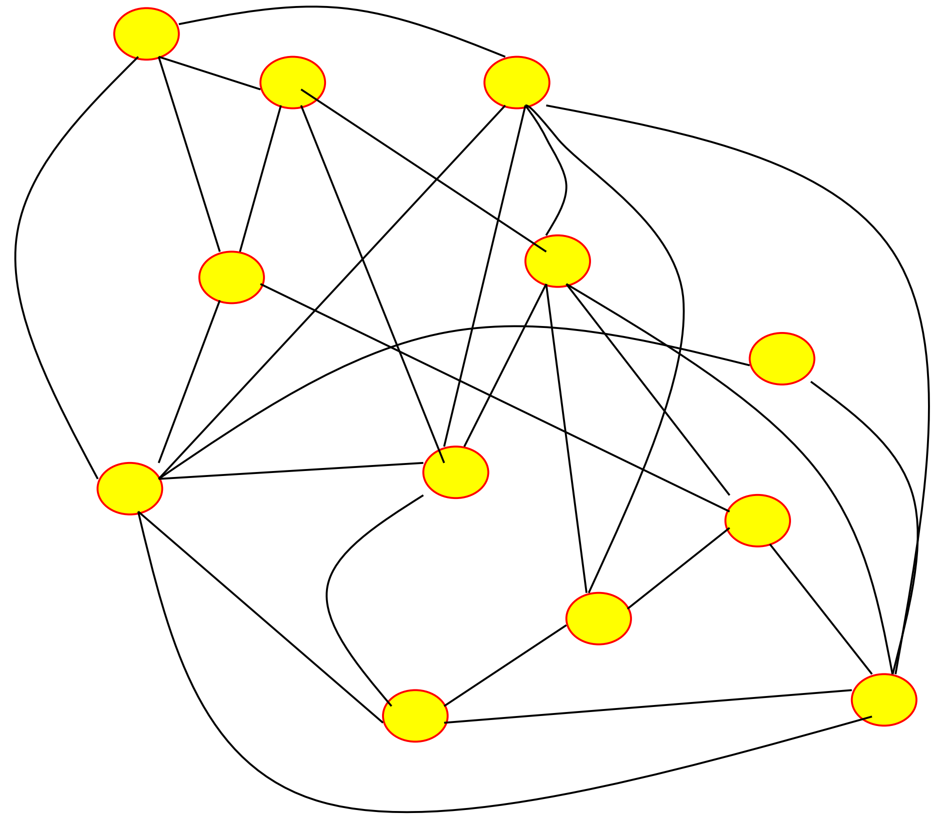
-responses are added
-pulses created at threshold
-transmitted to other

→ Mathematical description

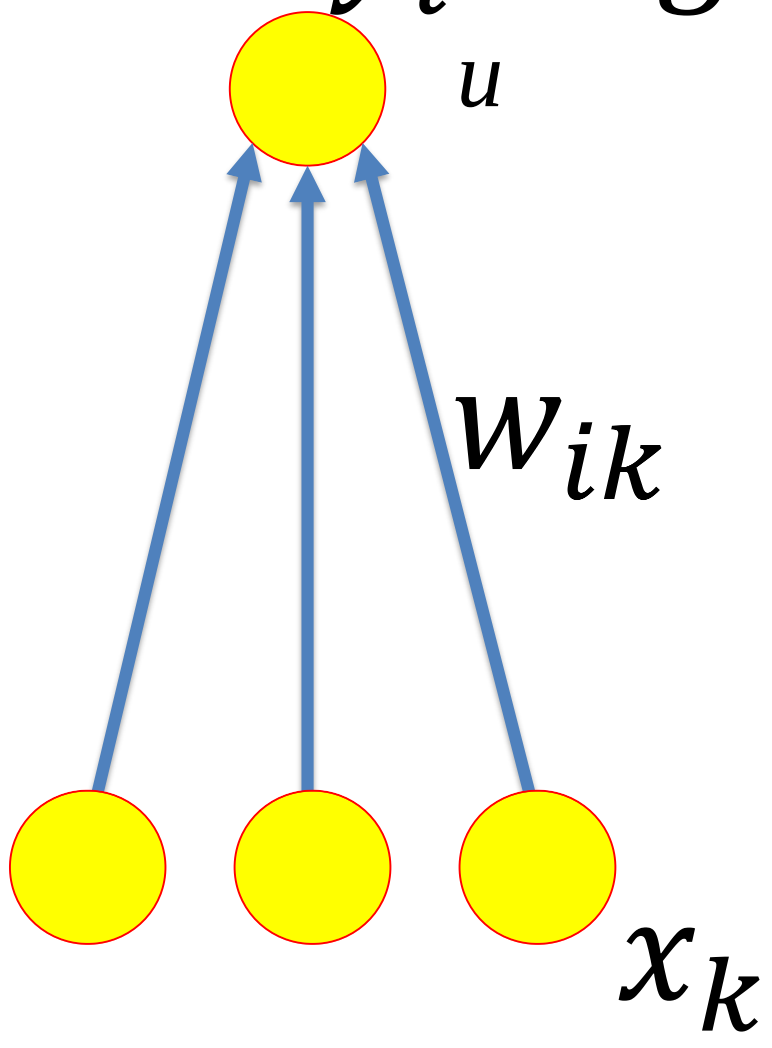
Modeling: artificial neurons

forget spikes: continuous activity x

forget time: discrete updates



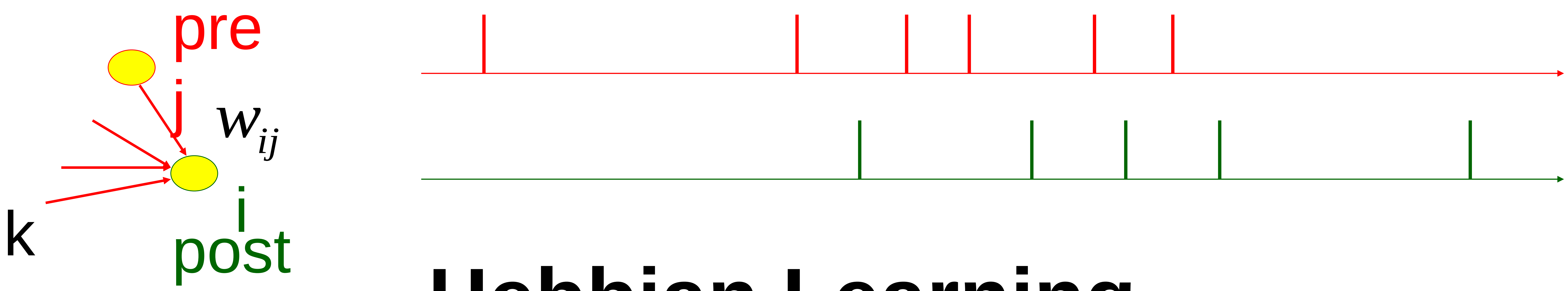
$$y_i = g\left(\sum_k w_{ik}\, x_k\right)$$



weights = adaptive parameters

Learning of connections in biology

Where do the connection weights come from?



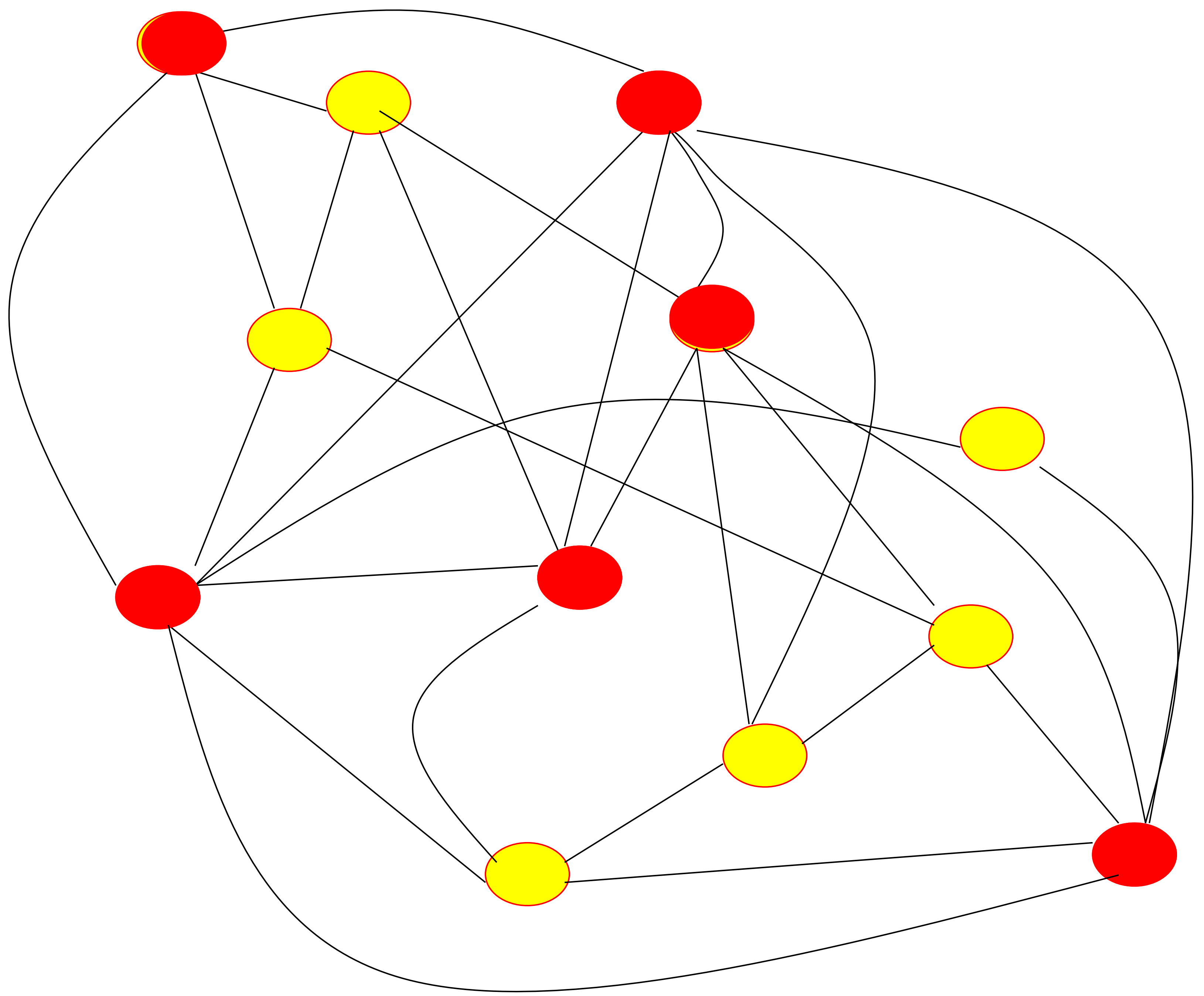
Hebbian Learning

When an axon of cell **j** repeatedly or persistently takes part in firing cell **i**, then j's efficiency as one of the cells firing i is increased

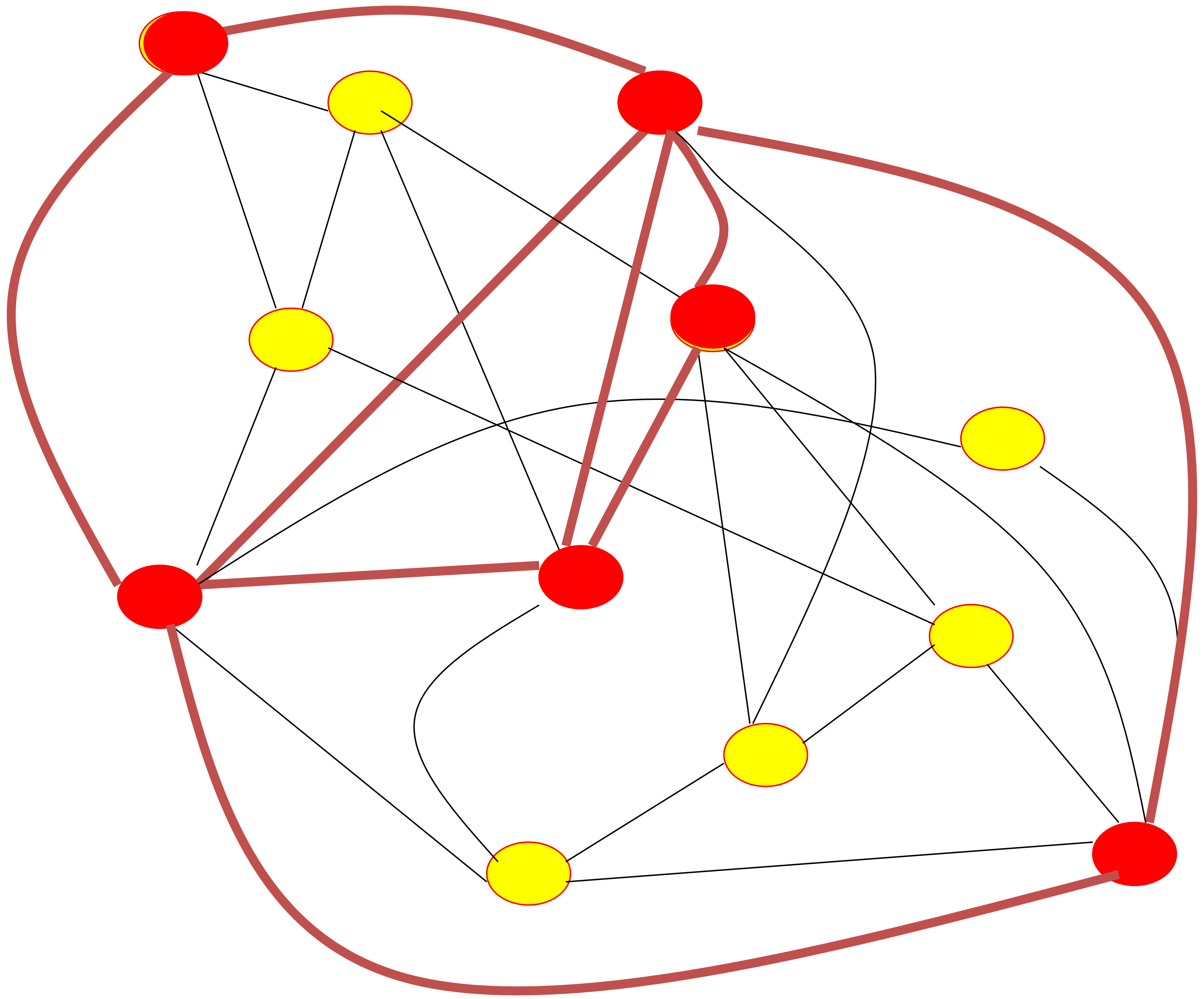
Hebb, 1949

- local rule
- simultaneously active neurons

Hebbian Learning of Associations



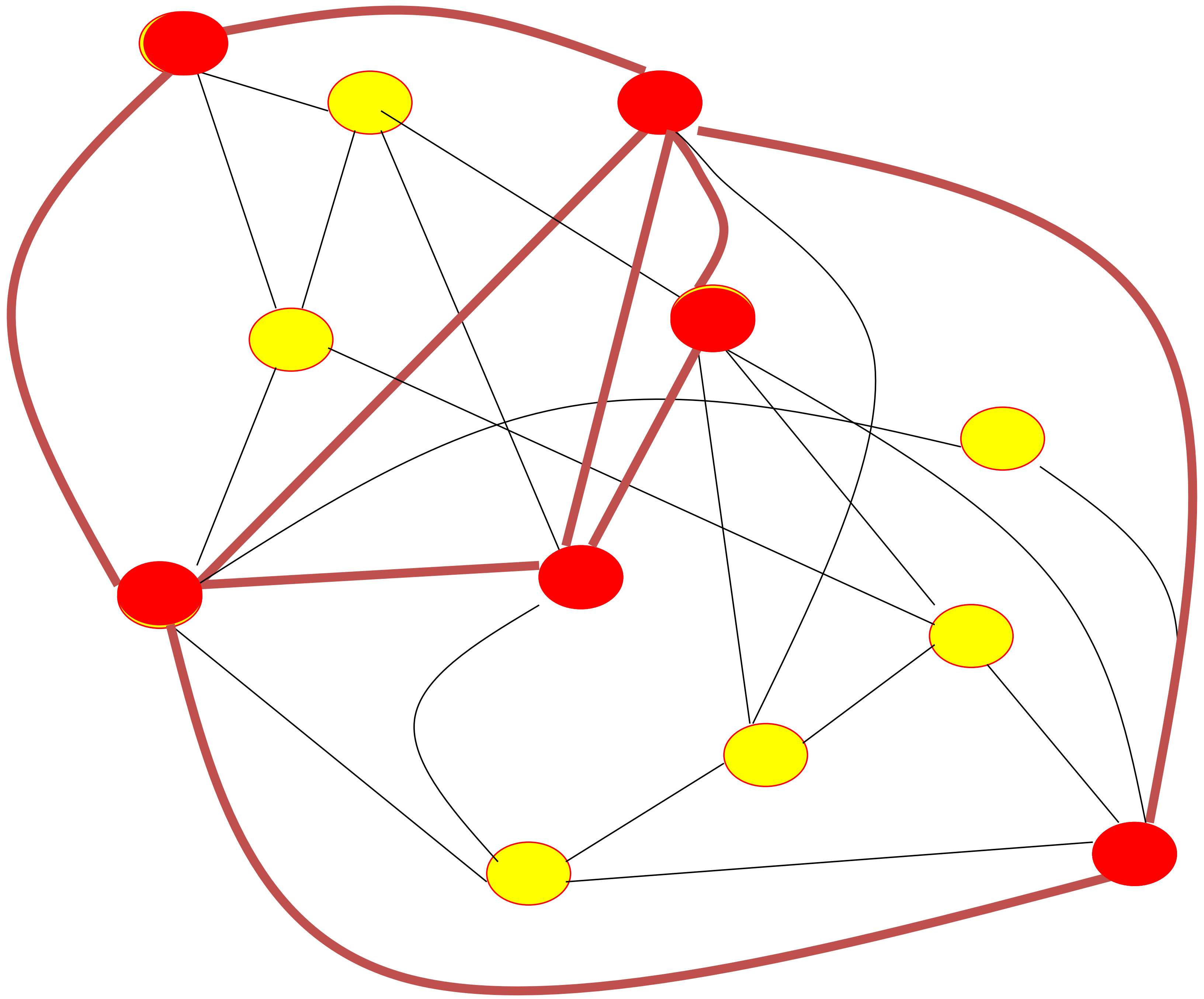
Hebbian Learning of Associations

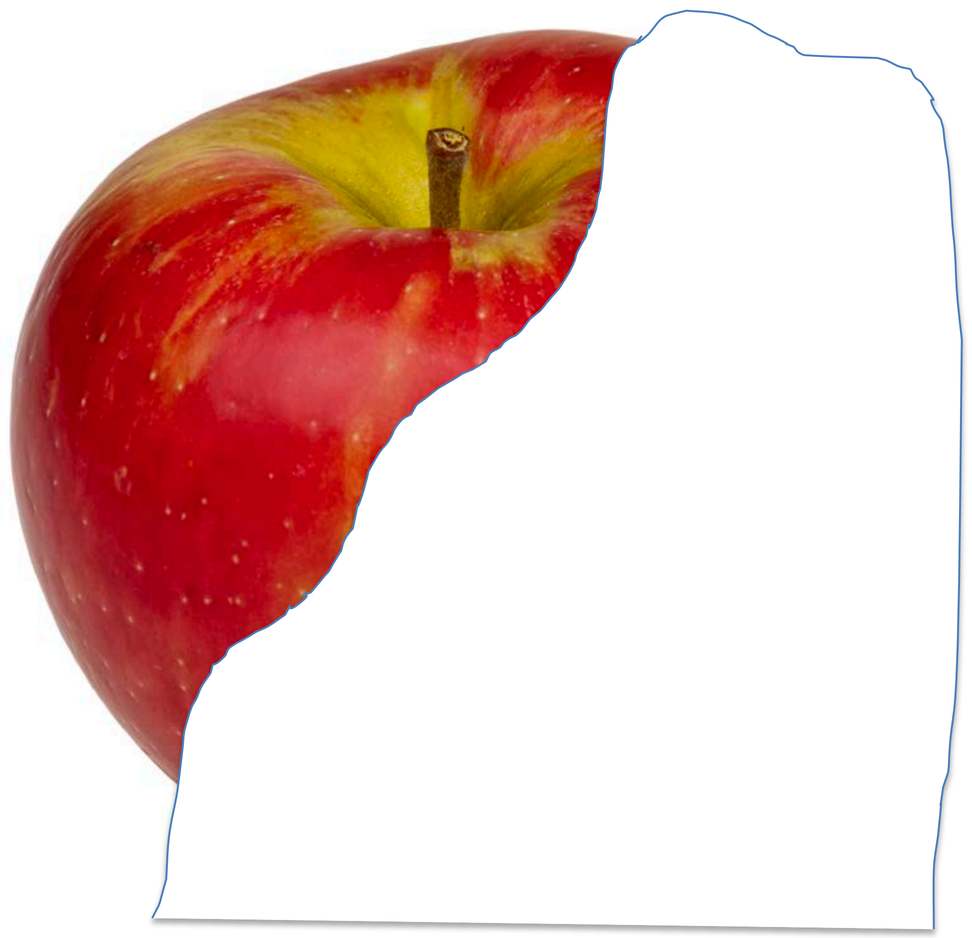


item memorized by change of synaptic weights

Hebbian Learning: Associative Recall

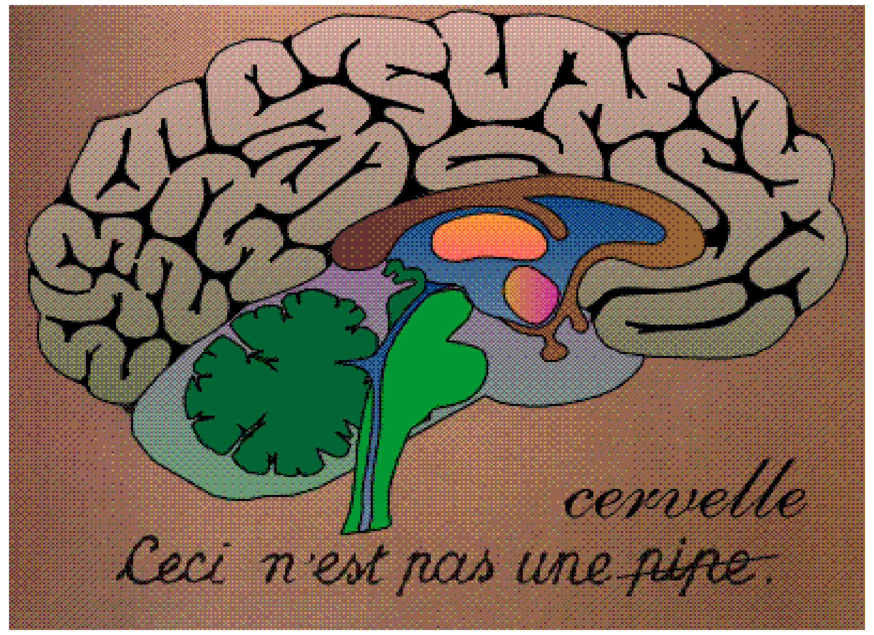
Recall:
Partial info



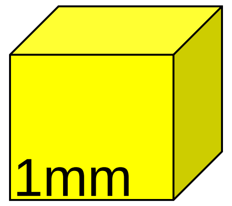


item recalled

Neurons and Synapses form a big network



Brain



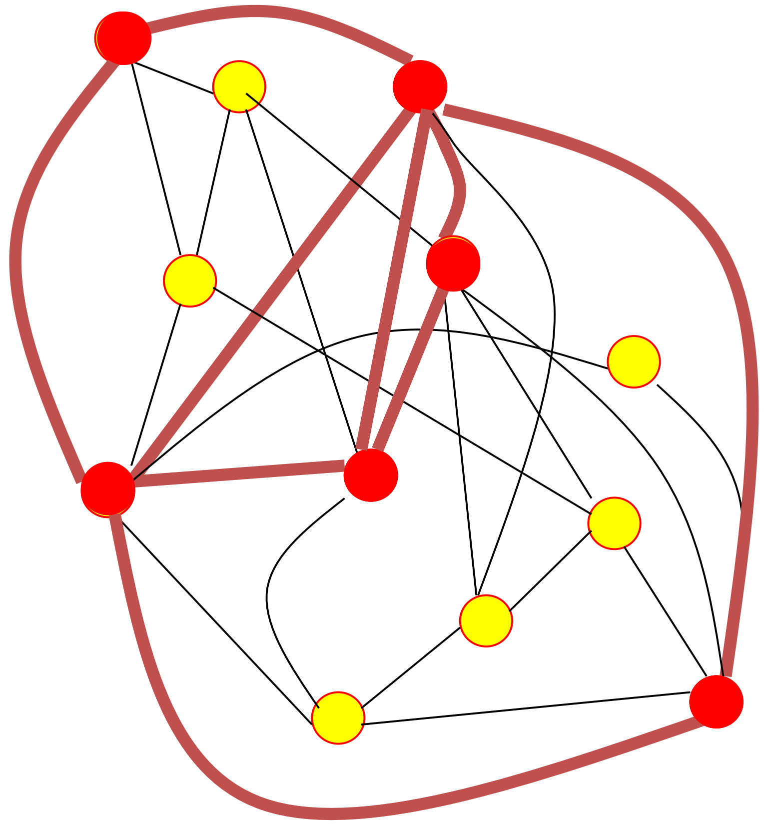
1mm

10 000 neurons
3 km of wire

10 billions neurons

10 000 connexions/neurons

memory in the connections



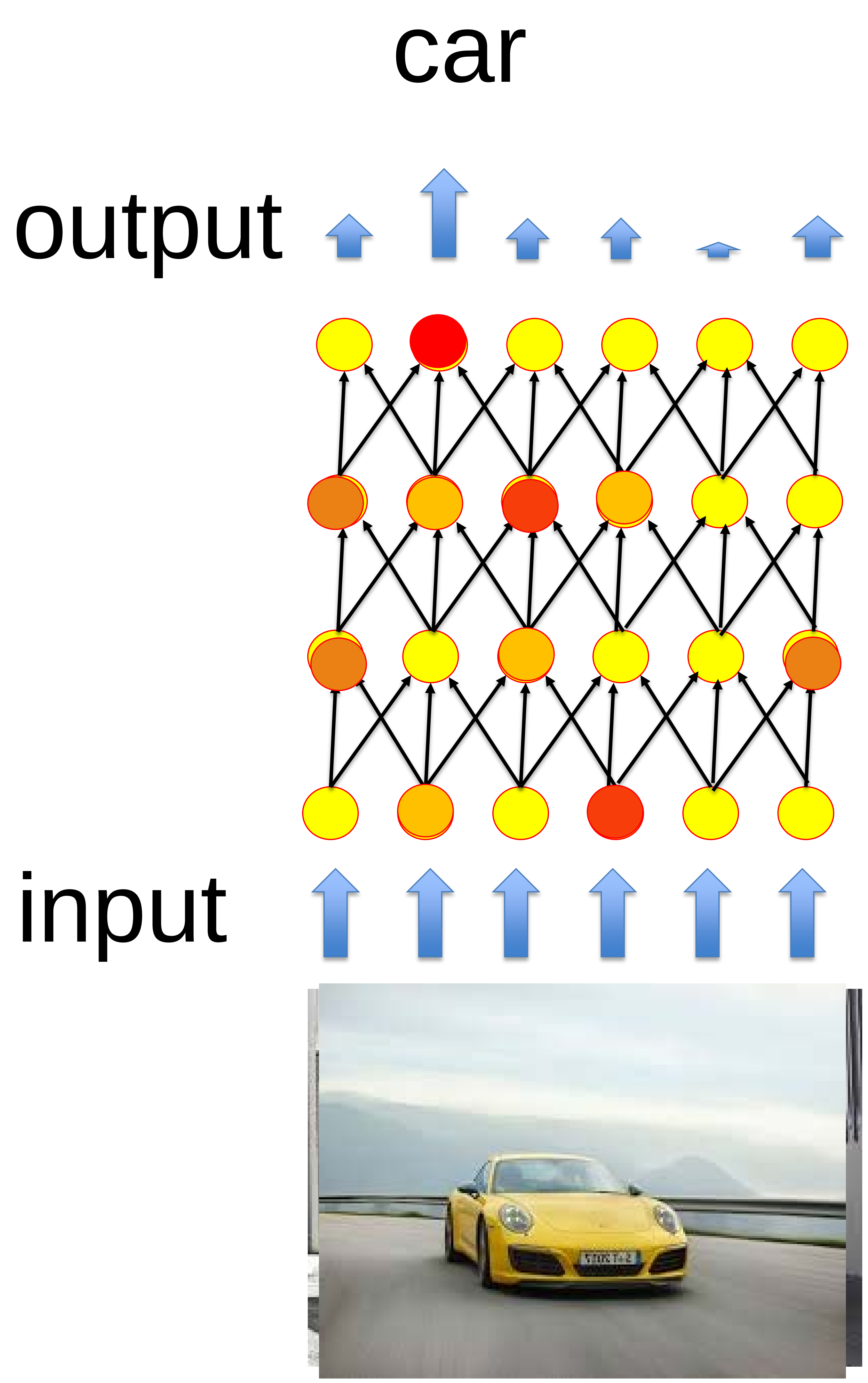
Distributed Architecture

No separation of processing and memory

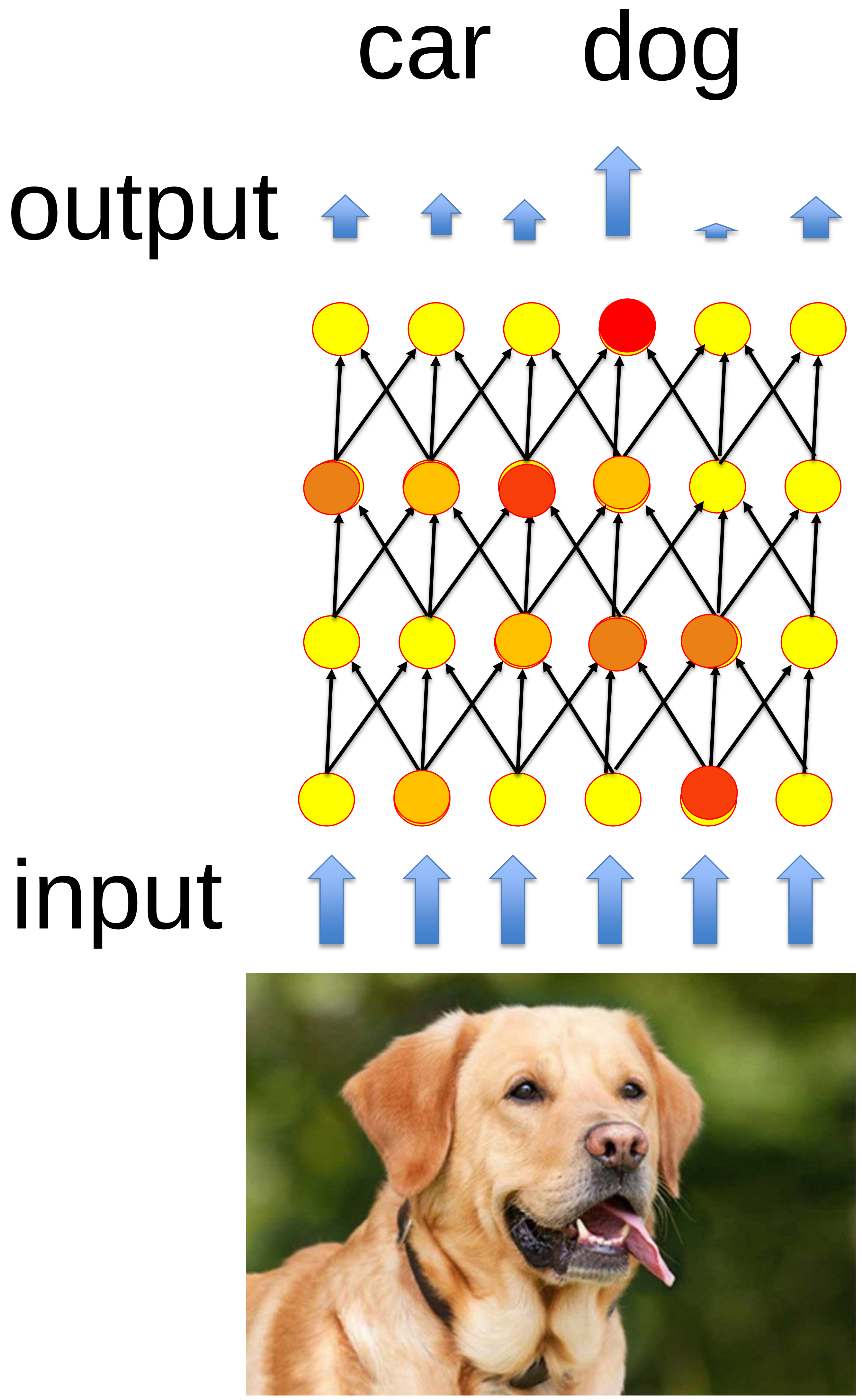
Quiz: biological neural networks

[ ] 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 has no recurrent connections and no feedback connections

Artificial Neural Networks for classification



Artificial Neural Networks for classification



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

Artificial Neural Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

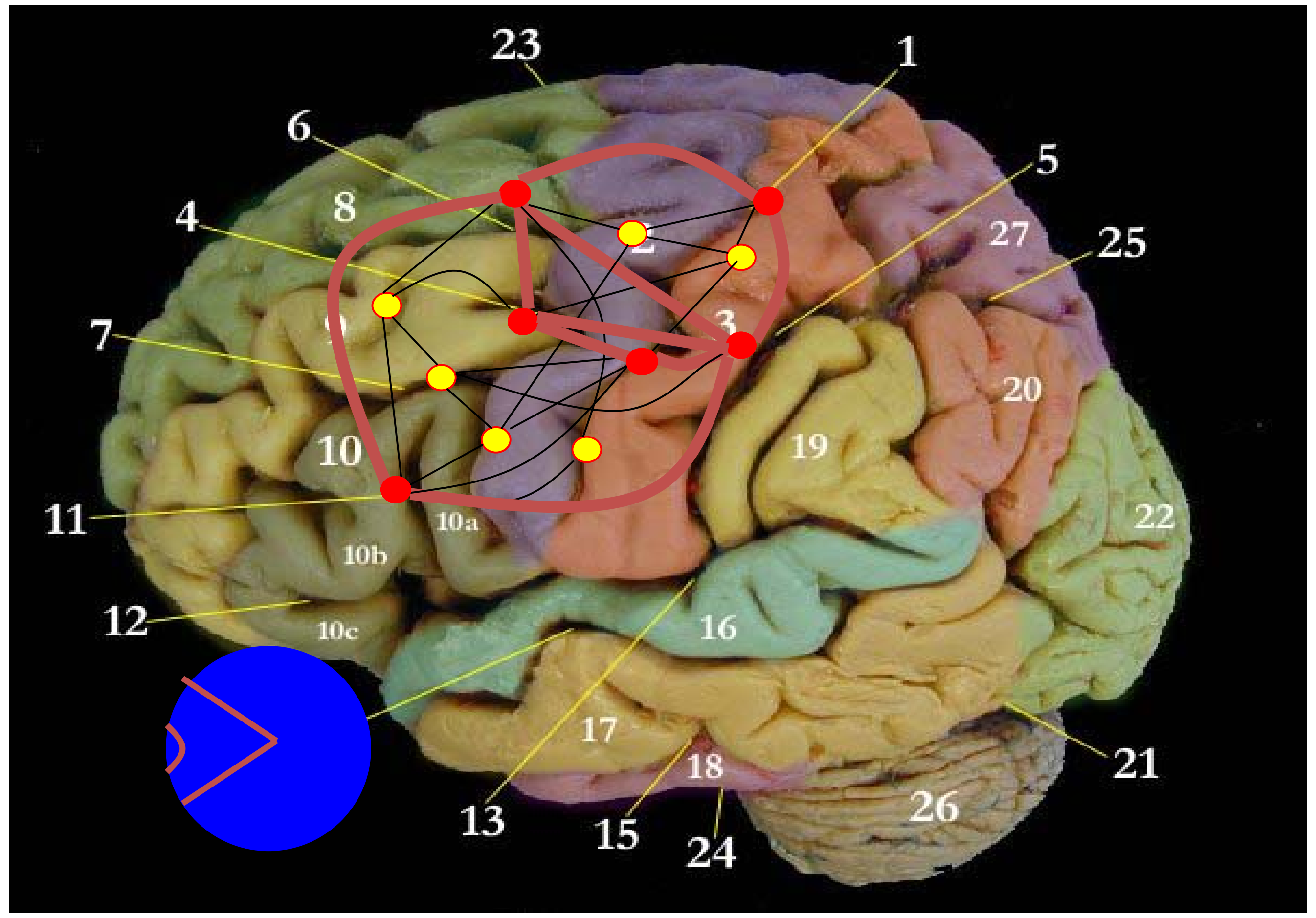
1. The brain
2. Artificial Neural Networks
   - for classification
   - for action learning

Artificial Neural Networks for action learning

Coactivation of 2 neurones:

- Connections strengthened
- Facilitates to repeat same action

Even the mistakes?

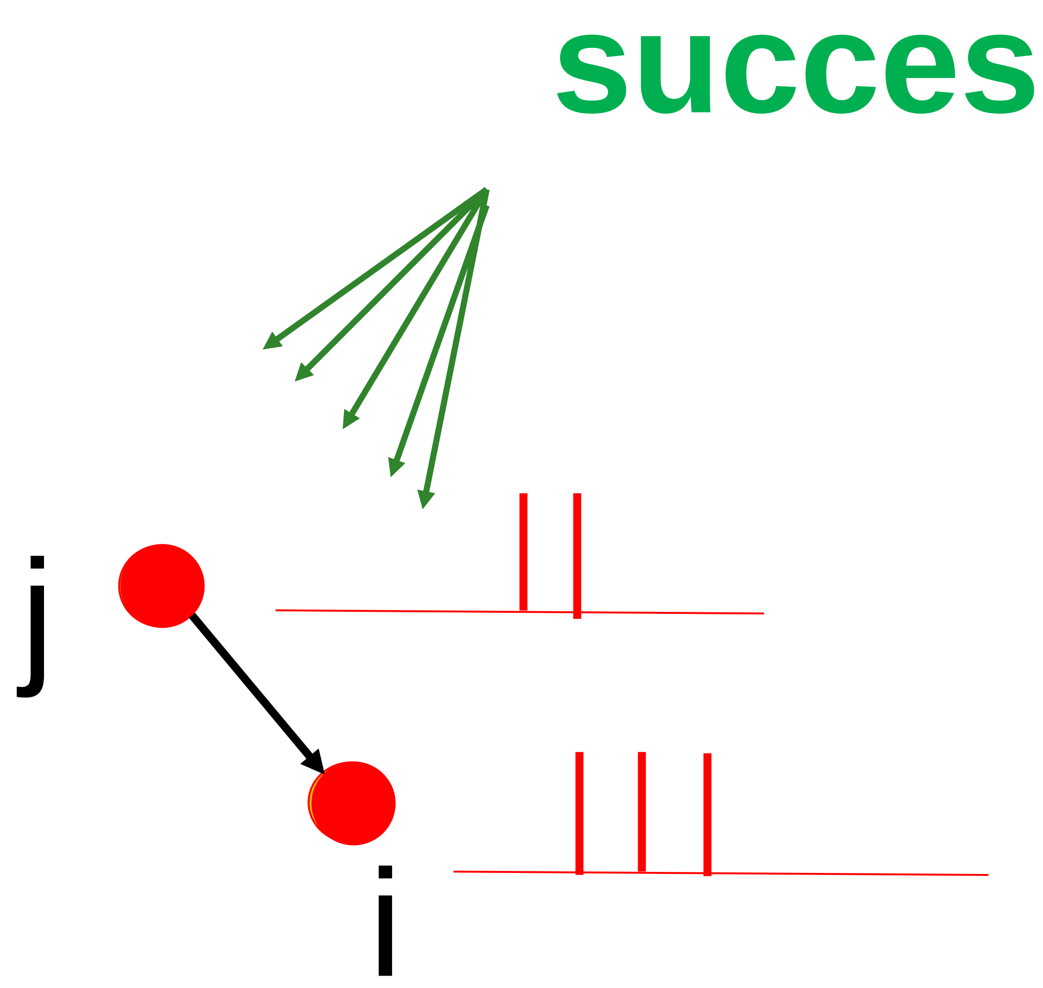


Missing:

Value of action

- ‘goodie’ for do
- ‘success’
- ‘compliment’

Modeling – the role of reward



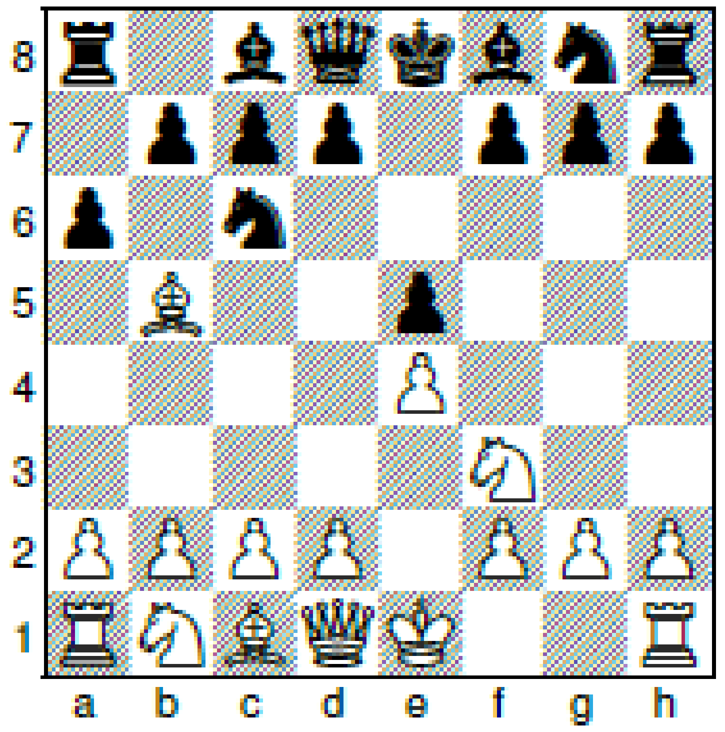
Three factors for changing a connection

- activity of neuron j
- activity of neurone i
- success

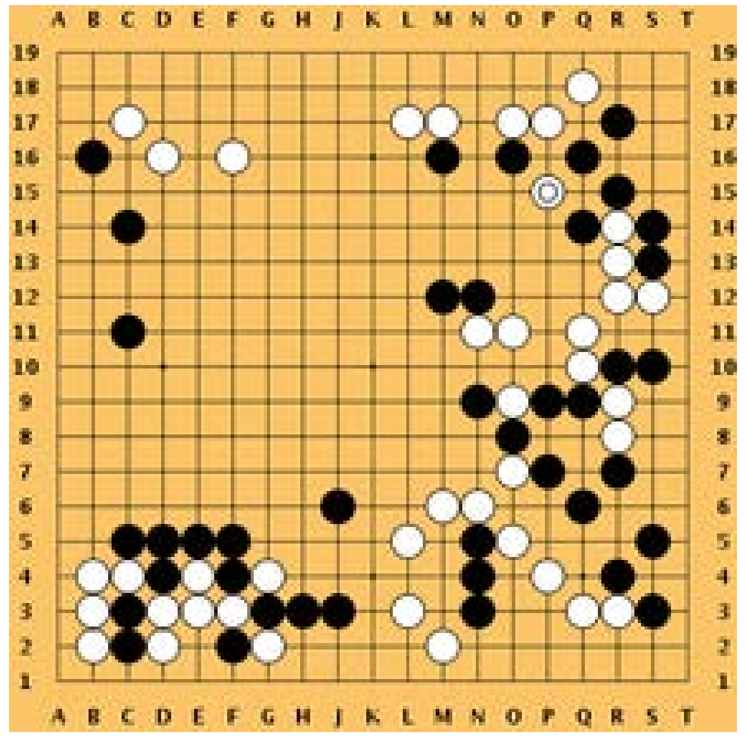
Barto 1985, Schultz et al. 1997; Waelti et al., 2001; Reynolds and Wickens 2002; Lisman et al. 2011

Deep reinforcement learning

Chess



Go



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

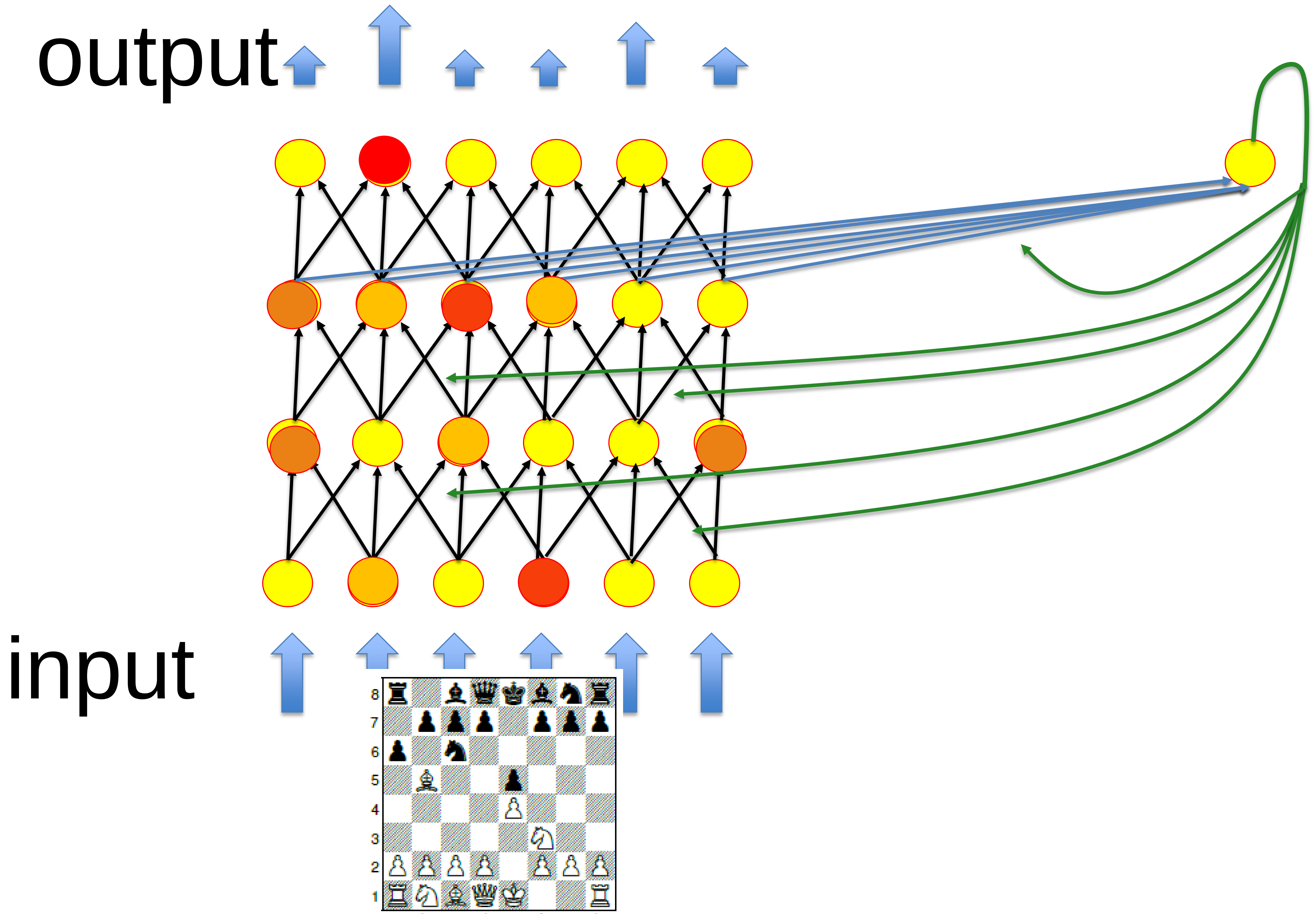
In Go, it beats Lee Sedol

Deep reinforcement learning

Network for choosing action

action:
Advance king

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



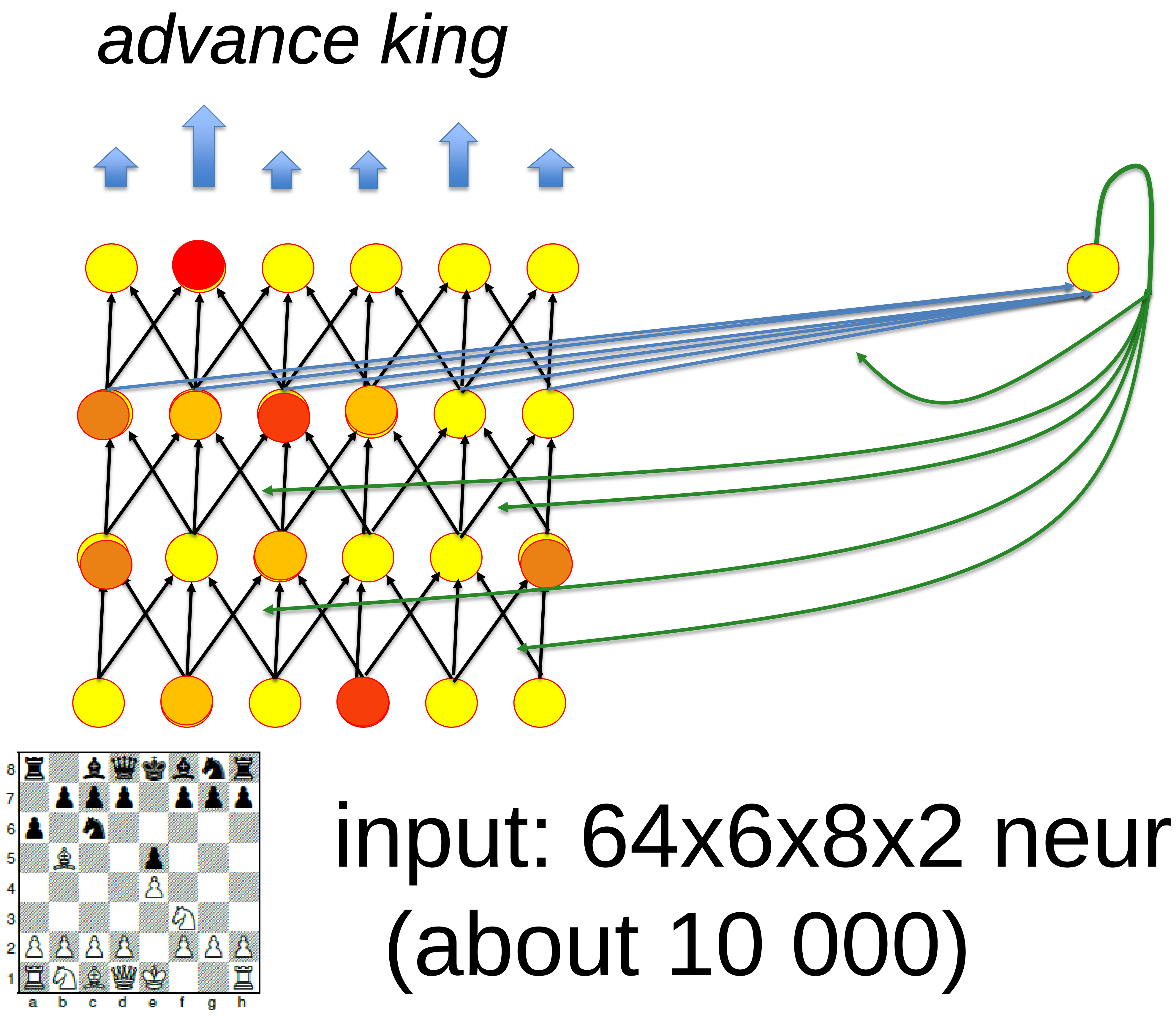
learning:
- change connections

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

Deep reinforcement learning (alpha zero)

Silver et al. (2017) , Deep Mind

output: 4672 actions



input: 64x6x8x2 neuronss
(about 10 000)

Training 44Mio games (9 hours)

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

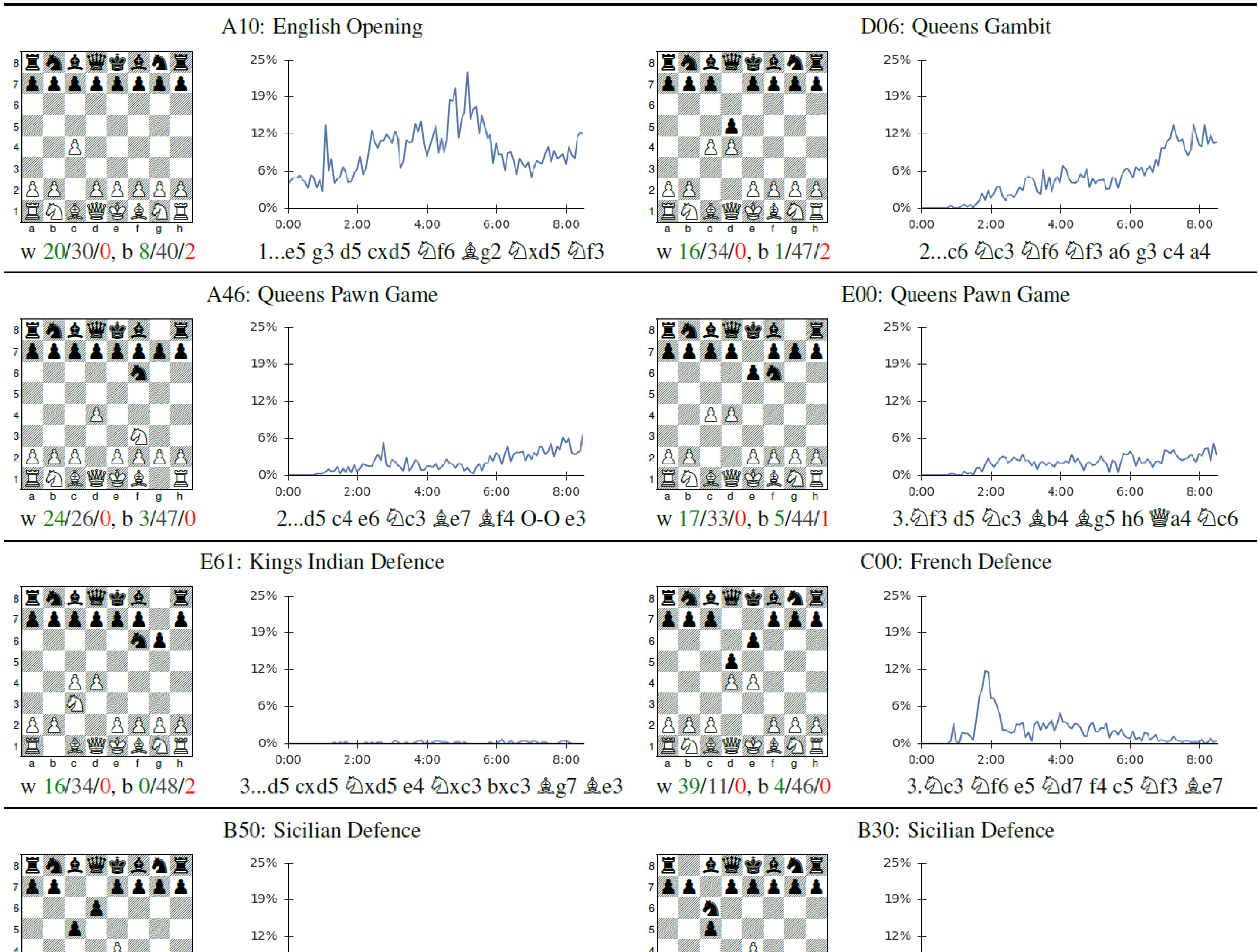
Deep reinforcement learning (alpha zero)

Chess:

-discovers classic openings

-beats best human players

-bets best classic AI algorithms



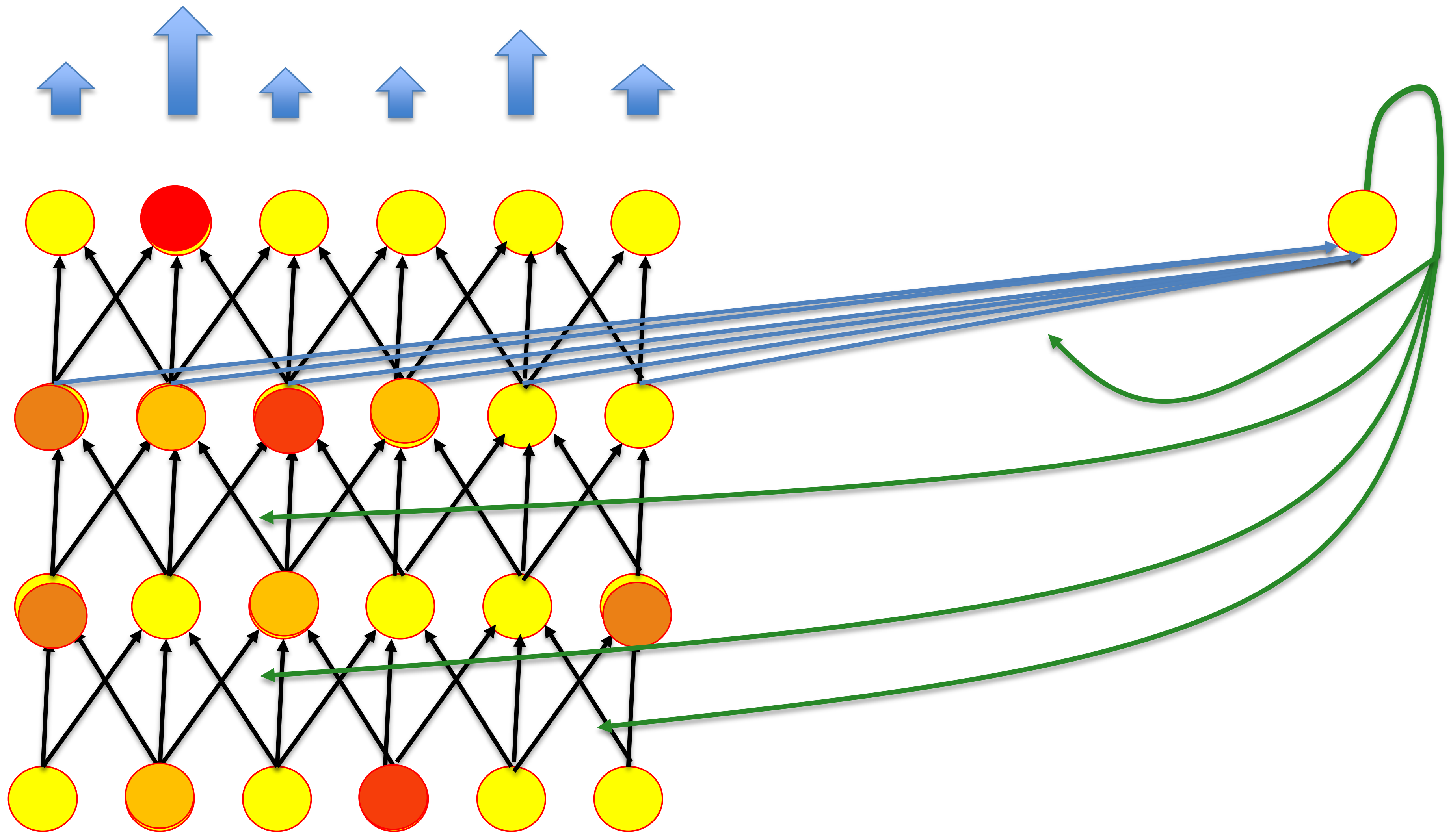
Self-driving cars

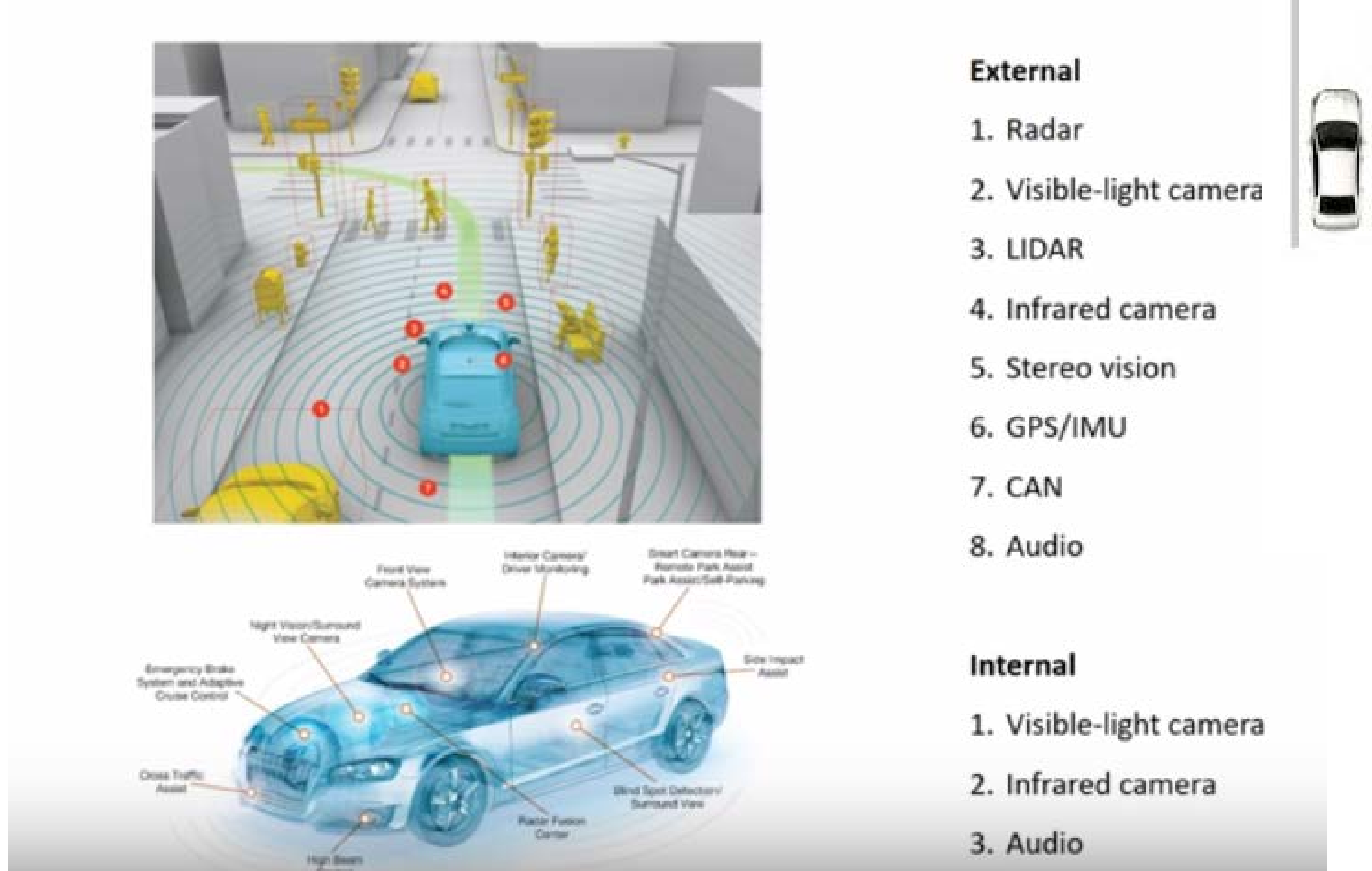
https://selfdrivingcars.mit.edu/

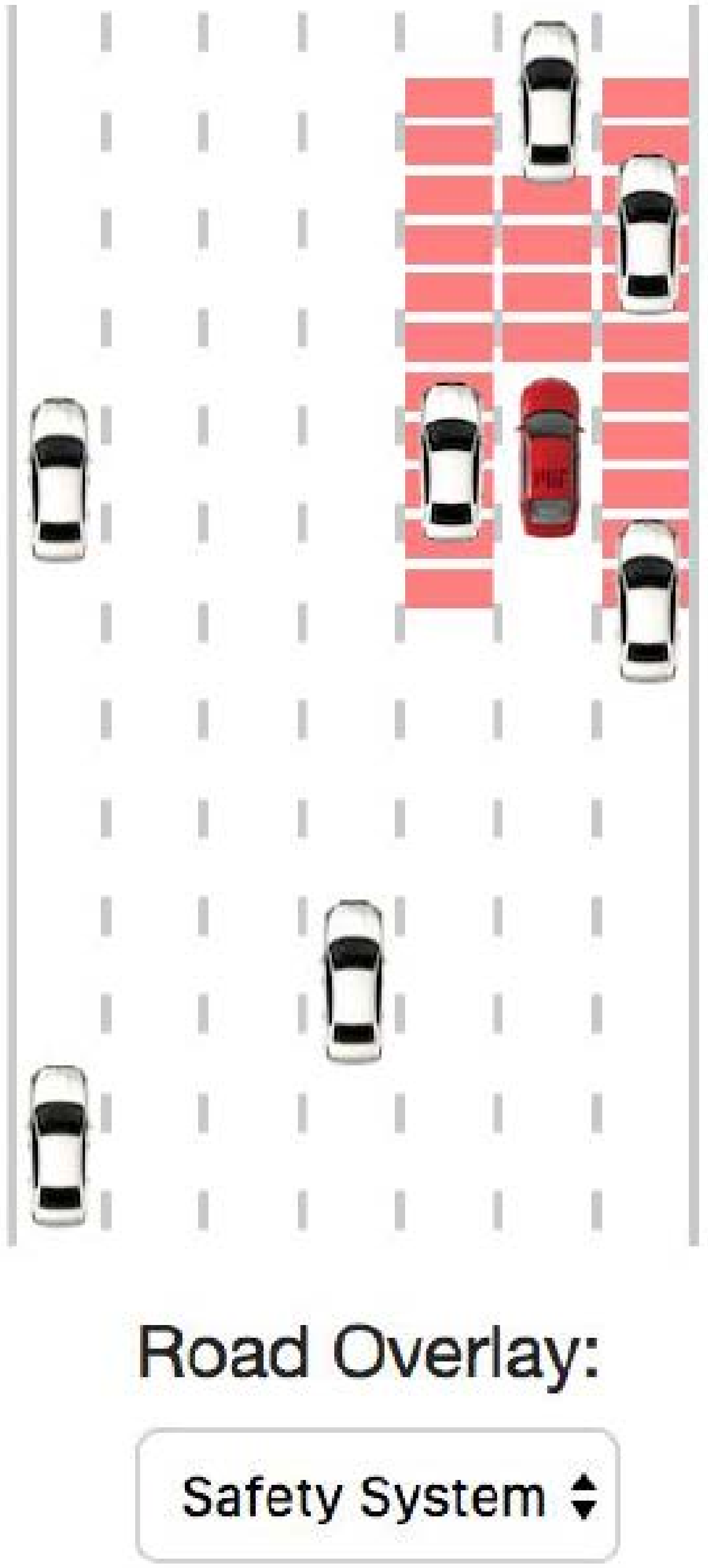
Lex Friedman, MIT

advance and accerate

Value: security, duration of travel







Artificial Neural Networks

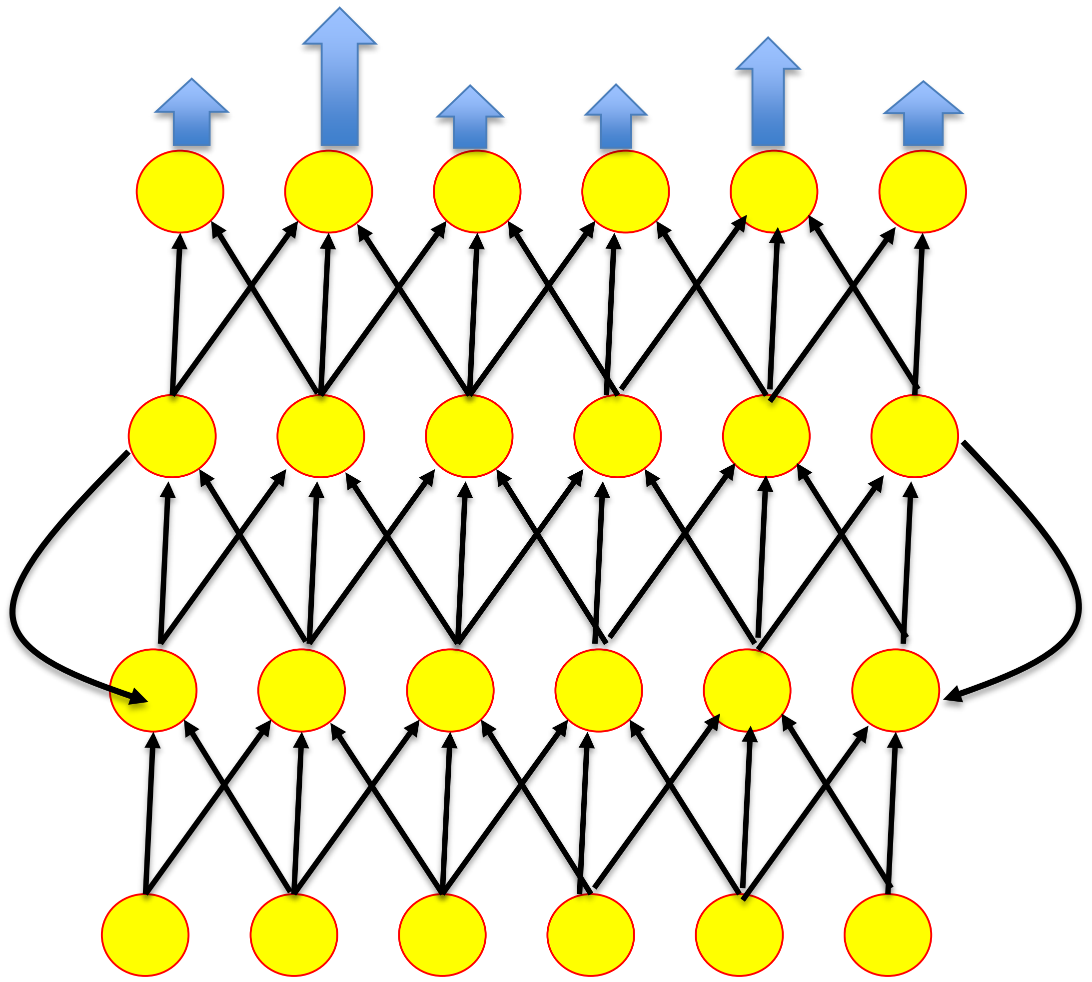
Wulfram Gerstner

EPFL, Lausanne, Switzerland

1. The brain
2. Artificial Neural Networks
   - for classification
   - for action learning
   - for sequences (music, translation, speech)

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 is based on rewards
[ ] Reinforcement learning is based on rewards
[ ] Reinforcement learning aims at optimal action choices
[ ] Classification aims at predicting the correct category such as ‘car’ or ‘dog’

Artificial Neural Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

1. The brain
2. Artificial Neural Networks
   - for classification
   - for action learning
   - for sequences
3. Overview of class

Artificial Neural Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

1. Simple perceptrons for classification
2. Backprop and multilayer perceptron
3. Statistical Classification by deep networks
4. Deep learning:
   regularization and tricks of the trade
5. Complements to deep learning
6. Sequence predictions and LSTMs
7. Convolutional networks
8. Reinforcement learning1: TD learning
9. Reinforcement learning2: SARSA
10. Reinforcement learning3: Policy Gradient
11. Deep Reinforcement learning
12. Applications

Artificial Neural Networks

Wulfram Gerstner
EPFL, Lausanne, Switzerland

1. Simple perceptrons for classification
2. Backprop and multilayer perceptron
3. Statistical Classification by deep networks ← miniproject1
4. Deep learning:
   regularization and tricks of the trade
5. Complements to deep learning
6. Sequence predictions and LSTMs ← miniproject2
7. Convolutional networks
8. Reinforcement learning1: TD learning
9. Reinforcement learning2: SARSA
10. Reinforcement learning3: Policy Gradient ← miniproject3
11. Deep Reinforcement learning
12. Applications

Miniprojects (MPs): we use software package ‘Keras’

- hand in 2 (or 3) out of 3 projects
- graded on a scale of 1-6
- average grade of MPs counts 1/3 toward final grade
- we do fraud detection interviews
- you get three weeks for each MP
- MP can be done alone or in group of two students
- interview for final MP is in first week after end of classes

→ **plan ahead!!**

Written exam:

- counts 2/3 toward final grade
- no tools allowed (no calculator, no cell phone, no paper, no book)
- ‘mathy’

Exercise sessions as follows:

- hand-out of exercise sheet ***n*** Friday of week ***n***
- You work on it until Thursday of week ***n+1***
- Solutions posted at noon, Thursday ***n+1***
- Friday week ***n+1*** by 10:00 am: you vote for the one exercise that you want to be explained
- Friday week ***n+1*** at 12:00 am. This exercise is explained on the blackboard + Q&A

  \+ Q&A for miniprojects

TA's:
- Dr. Johannni Brea
- Dane Corneil
- Florian Colombo
- Teo Stocco

Artificial Neural Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

- The math is developed on the blackboard
- There are no written course notes!!
- All of the contents are standard textbook material

Choose a textbook that you like:

For first half of class:

- *Pattern Recognition and Machine Learning*, C.M Bishop, 2006
- *Neural Networks for Pattern Recognition*, C.M. Bishop, 1995
- *Deep Learning*, Ian Goodfellow et al., 2017 (also online)

For second half of class:

- *Reinforcement learning,* R. Sutton+ A. Barto (2^{nd} ed, online)

Artificial Neural Networks

Prerequisits:

CS433, Pattern Classification and Machine Learning
(Profs Jaggi+Urbanke)

Rules:

If you have taken this class: please ask many questions

If you have not taken this class: please do not complain

Artificial Neural Networks

Learning outcomes:

- apply learning in deep networks to real data
- assess/evaluate performance of learning algorithms
- Elaborate relations between different mathematical concepts of learning
- judge limitations of learning algorithms
- propose models for learning in deep networks

Transversal skills:

Access and evaluate appropriate sources of information
Manage priorities
work through difficulties, write a technical report

Artificial Neural Networks

Work load:

4 credit course → 6 hours per week for 18 weeks

(1 ECTS = 27 hours of work)

Questions?

… before we start

TA’s this year:

- Dr. Johannni Brea
- Dane Corneil
- Florian Colombo
- Teo Stocco

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

car (yes or no)

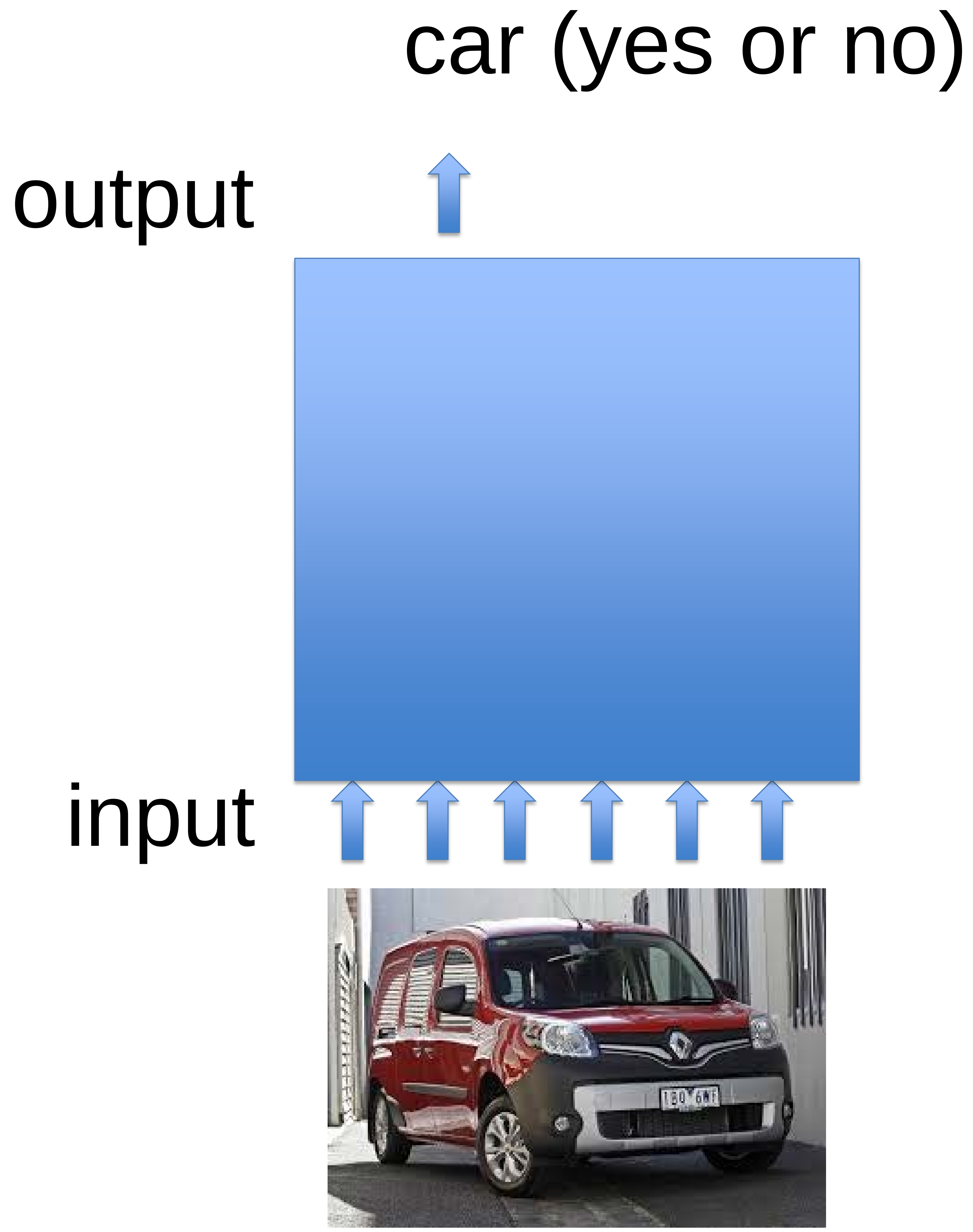
output

the classifier

input

1. The problem of Classification

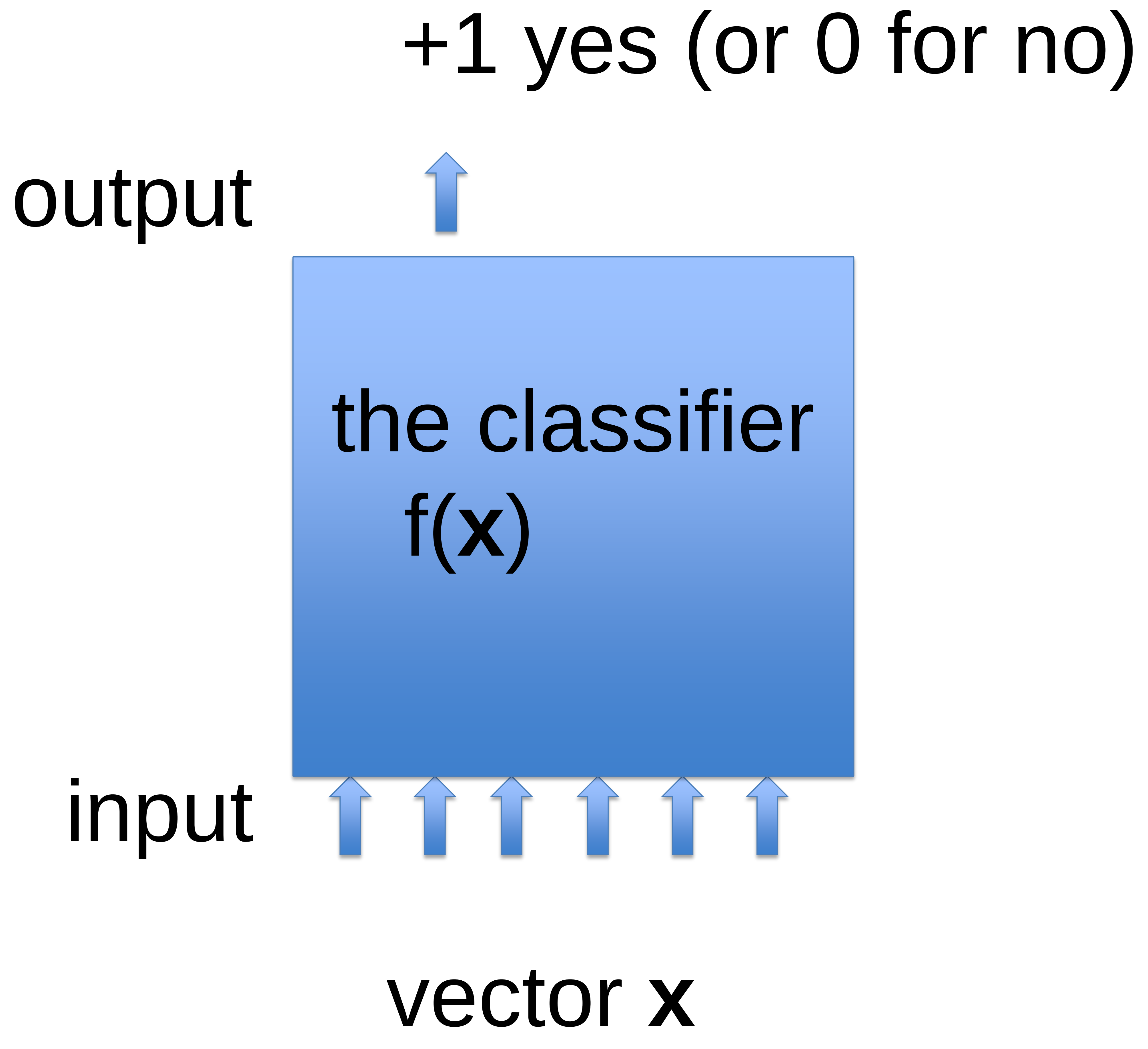
Blackboard 1:
from images to vector



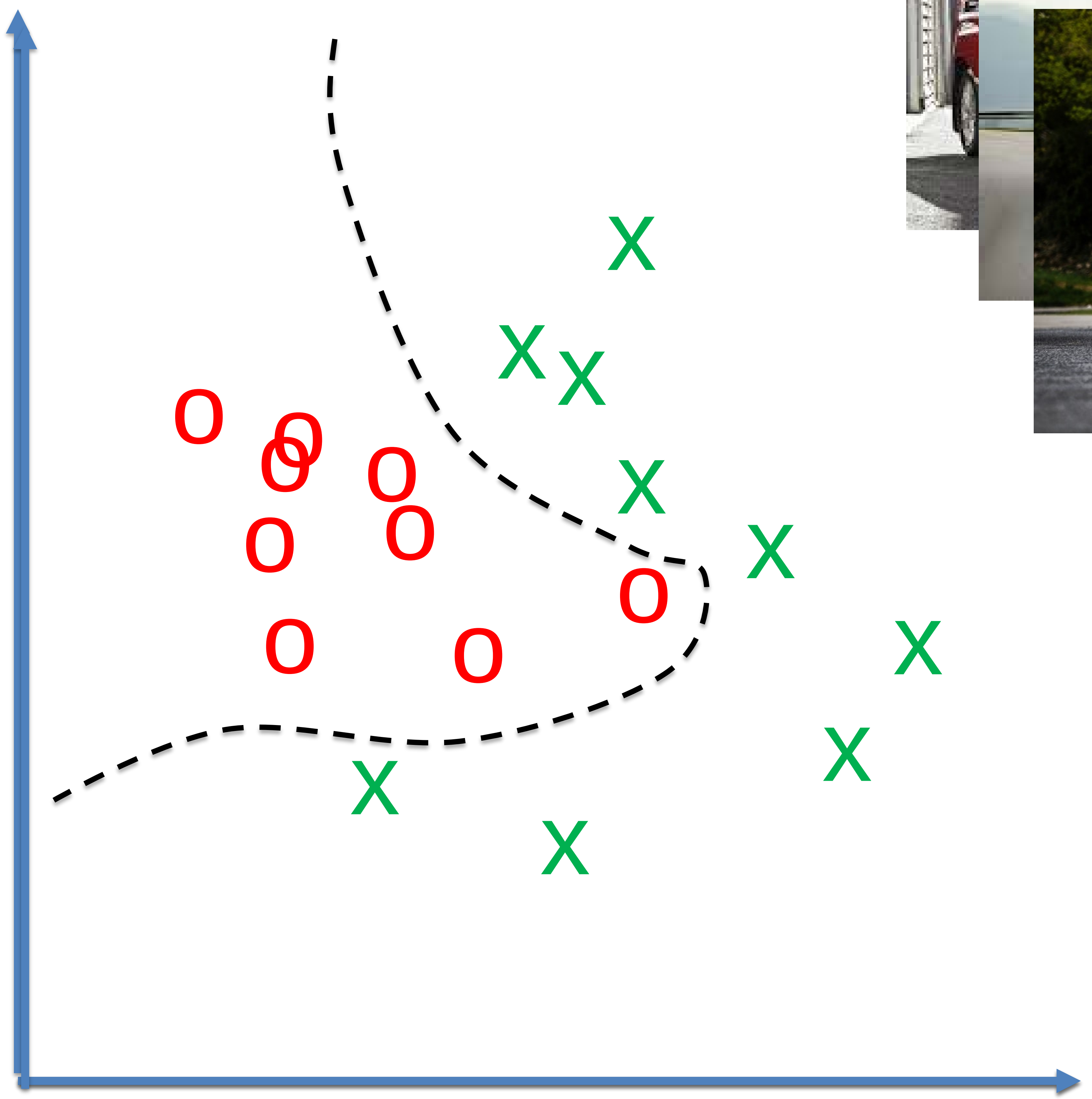
Blackboard 1:

from images to vector

1. The problem of Classification



1. Classification as a geometric problem



Blackboard 2:

from vectors to classification

Blackboard 2:

from vectors to classification

1. Classification as a geometric problem

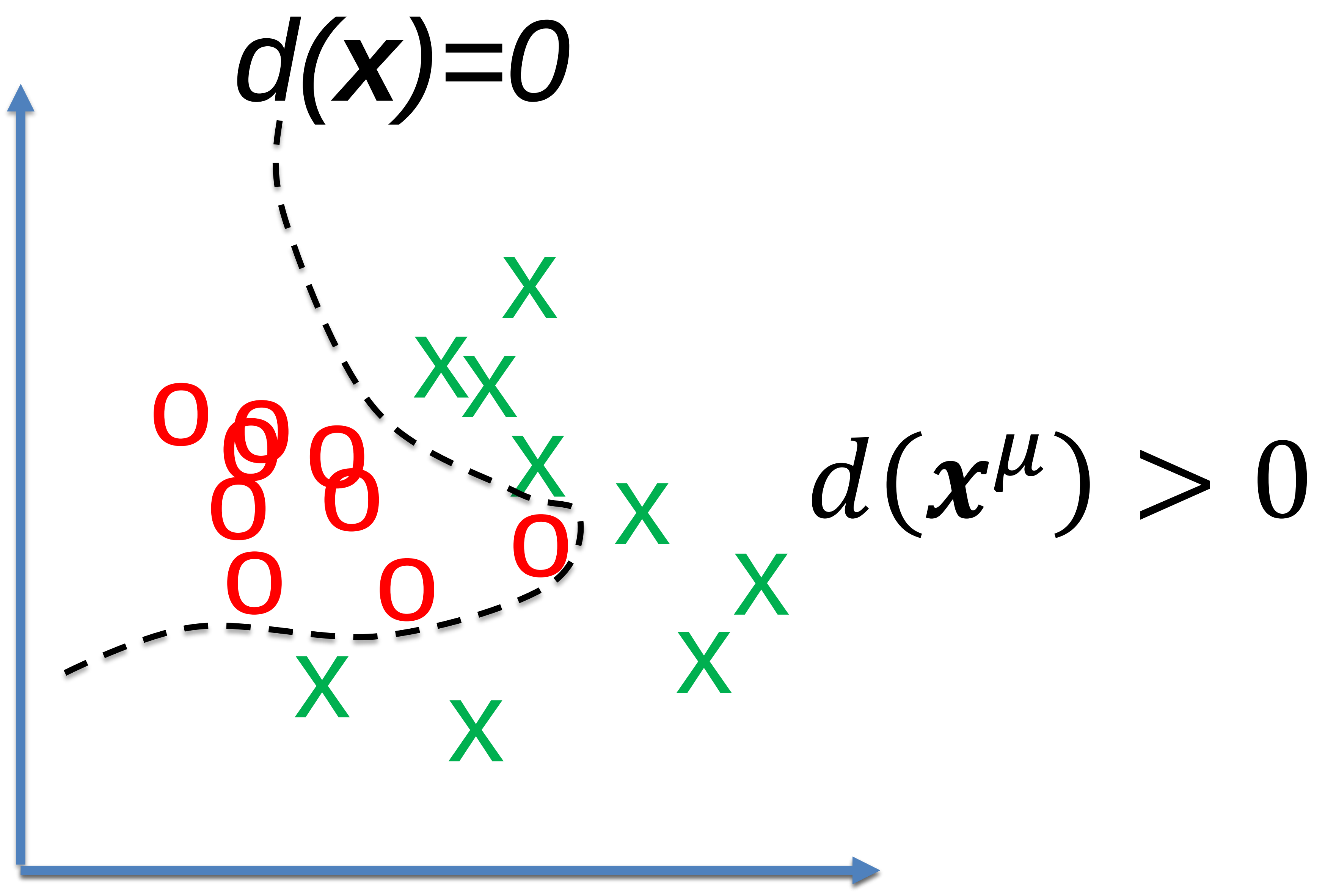
Task of Classification

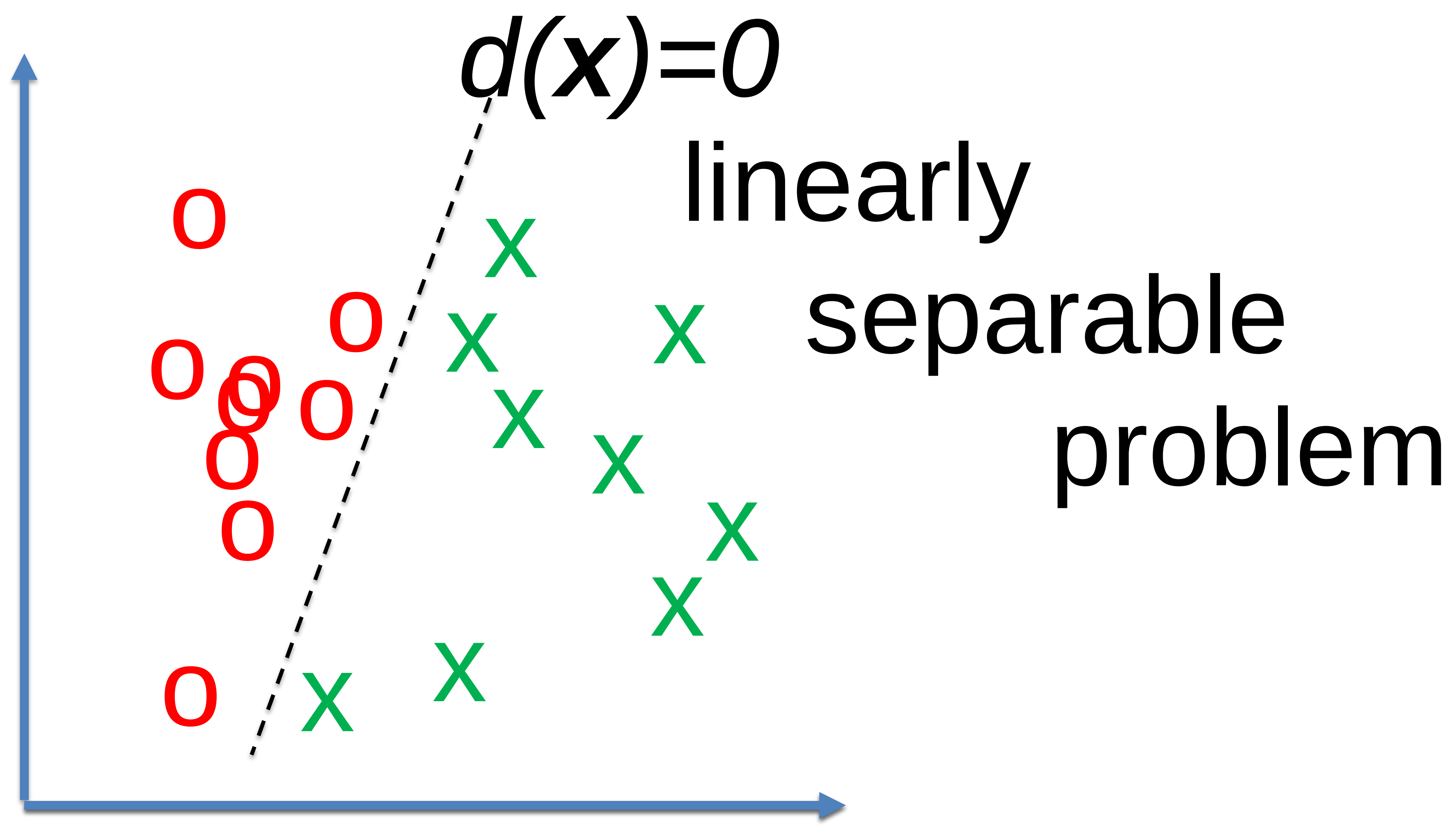
= find a **separating surface** in the high-dimensional input space

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

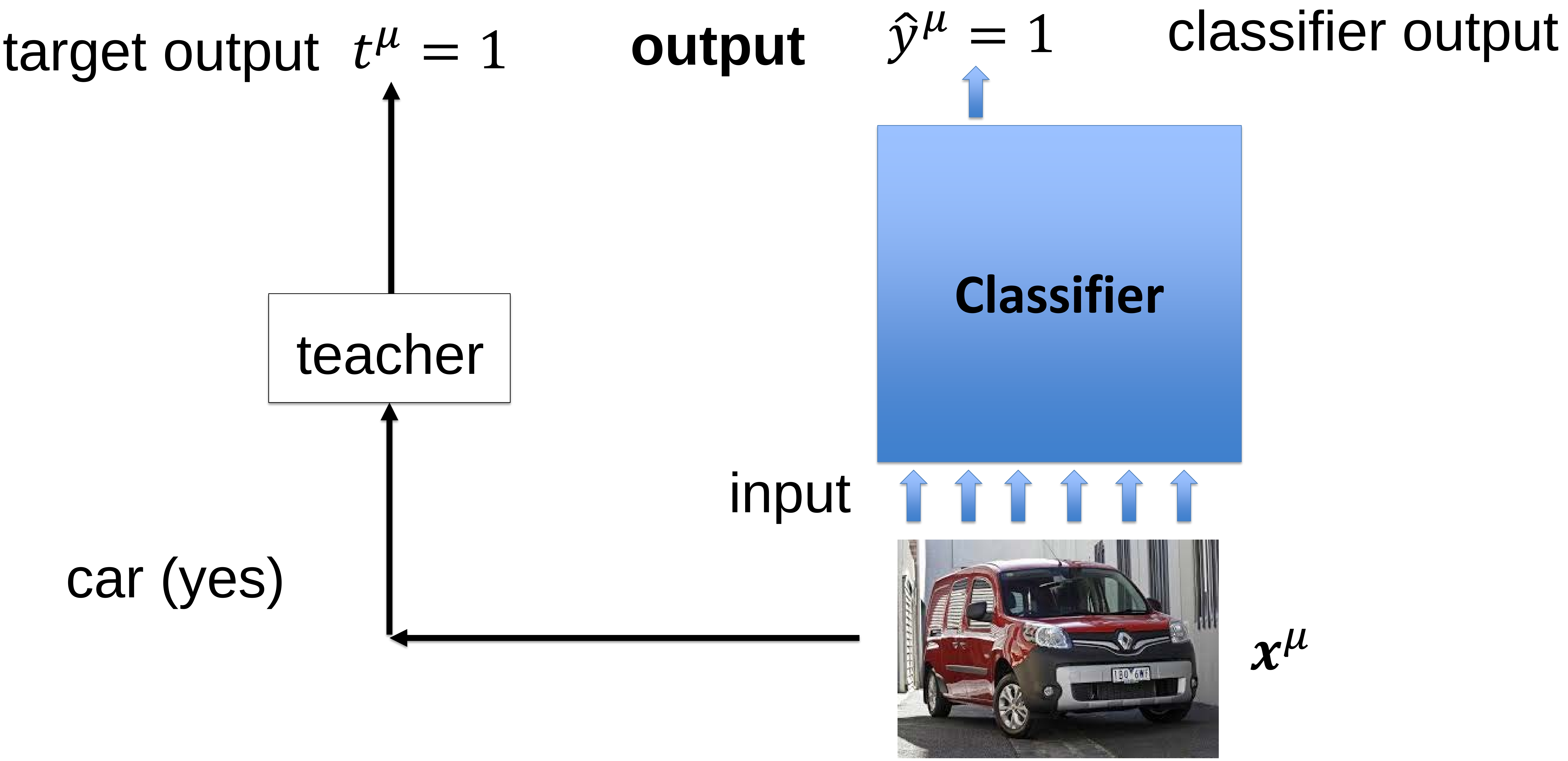
→ $d(\mathbf{x})=0$ on this surface; $d(\mathbf{x})>0$ *for all positive examples* $\mathbf{x}$

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





2. Data base for Supervised learning



2. Data base for Supervised learning

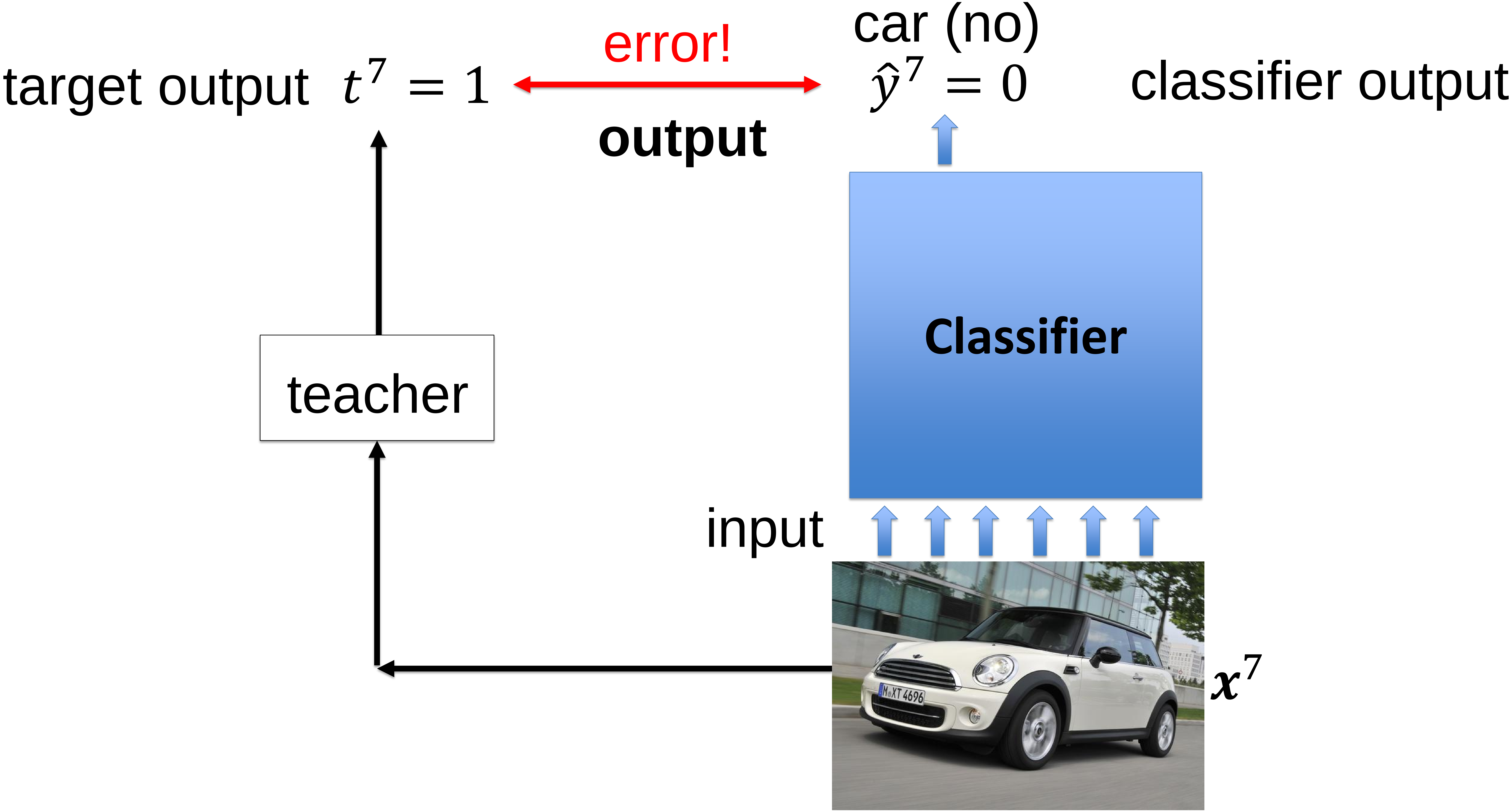
P data points $\{ \quad (\boldsymbol{x}^{\mu}, t^{\mu}) \quad , \qquad 1 \le \mu \le P \quad \};$

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

input target output

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

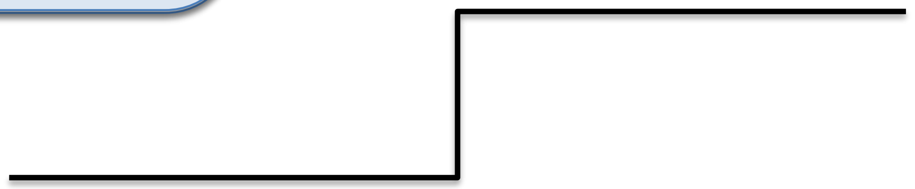
→ **use errors** $\hat{y}^\mu \neq t^\mu$ **for optimization of classifier**

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

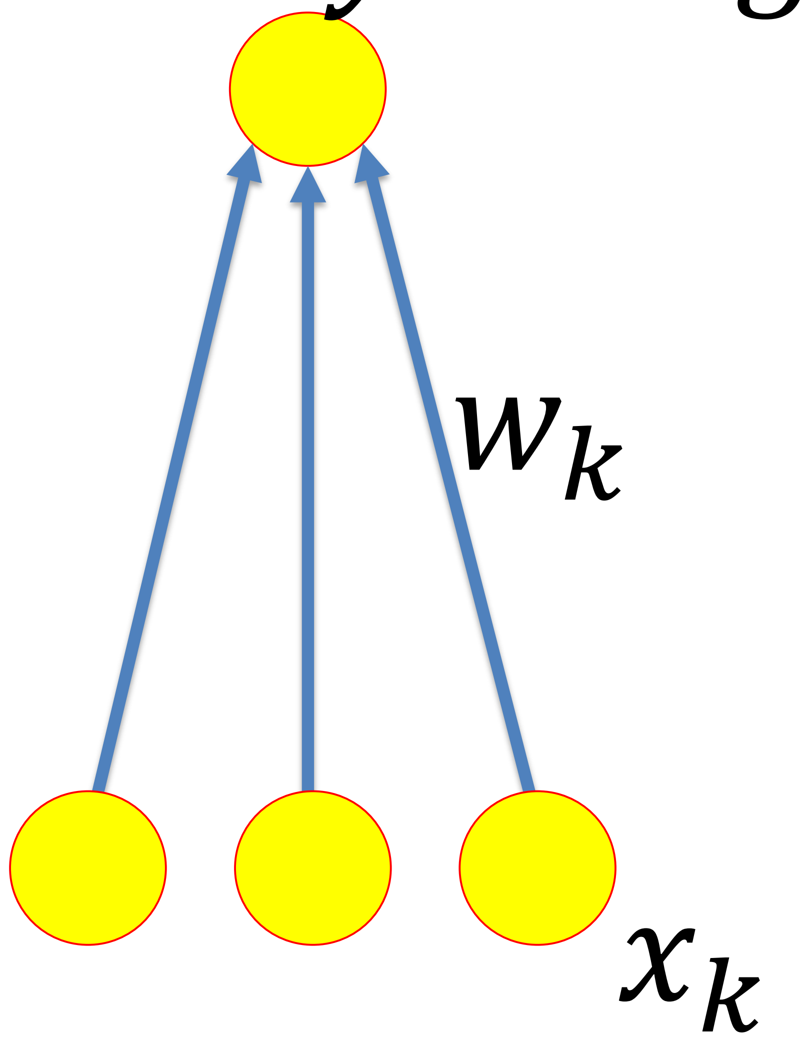
3. Single-Layer networks: simple perceptron

Blackboard 3:
hyperplane

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



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



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

vector $\boldsymbol{x}$

Blackboard 3:

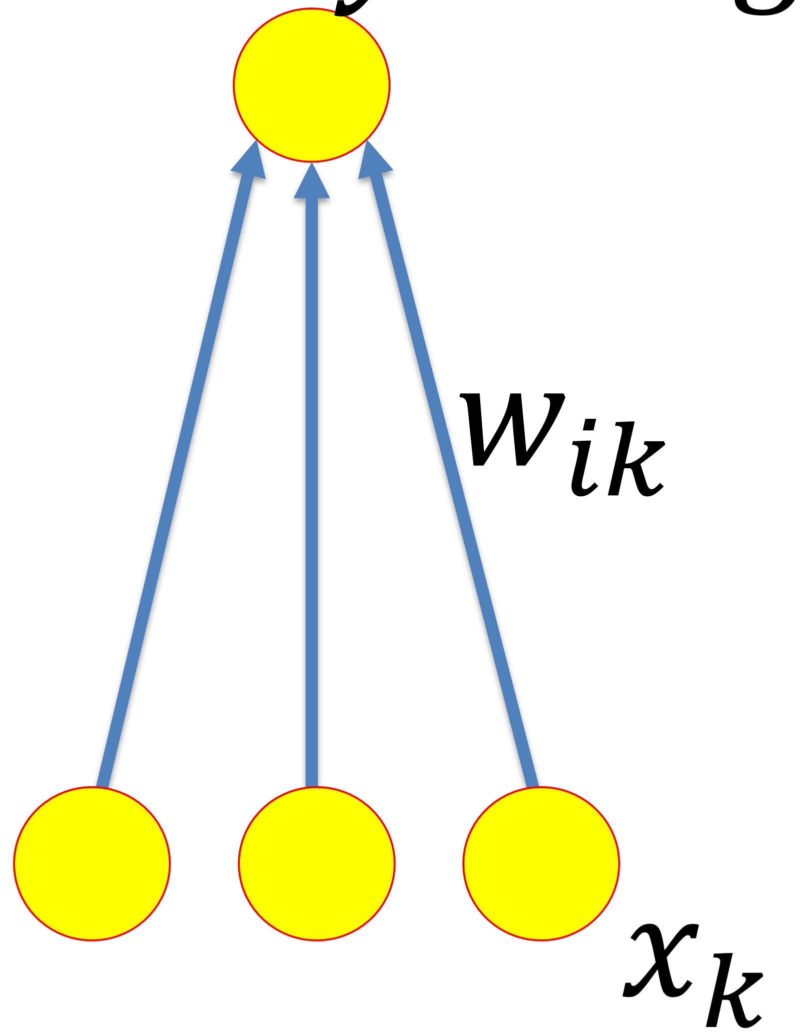
hyperplane

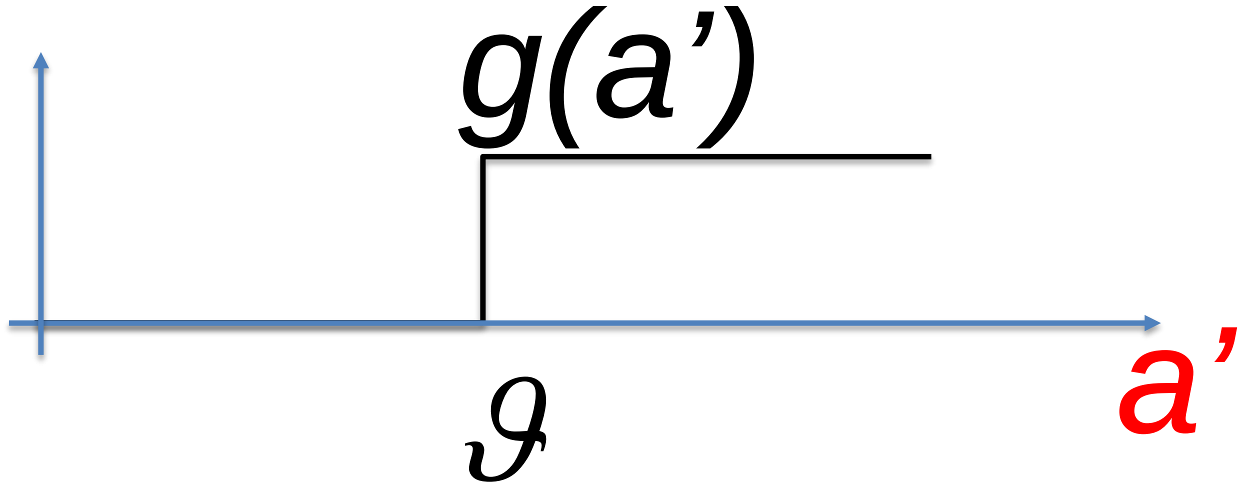
3. Single-Layer networks: simple perceptron

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

output

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





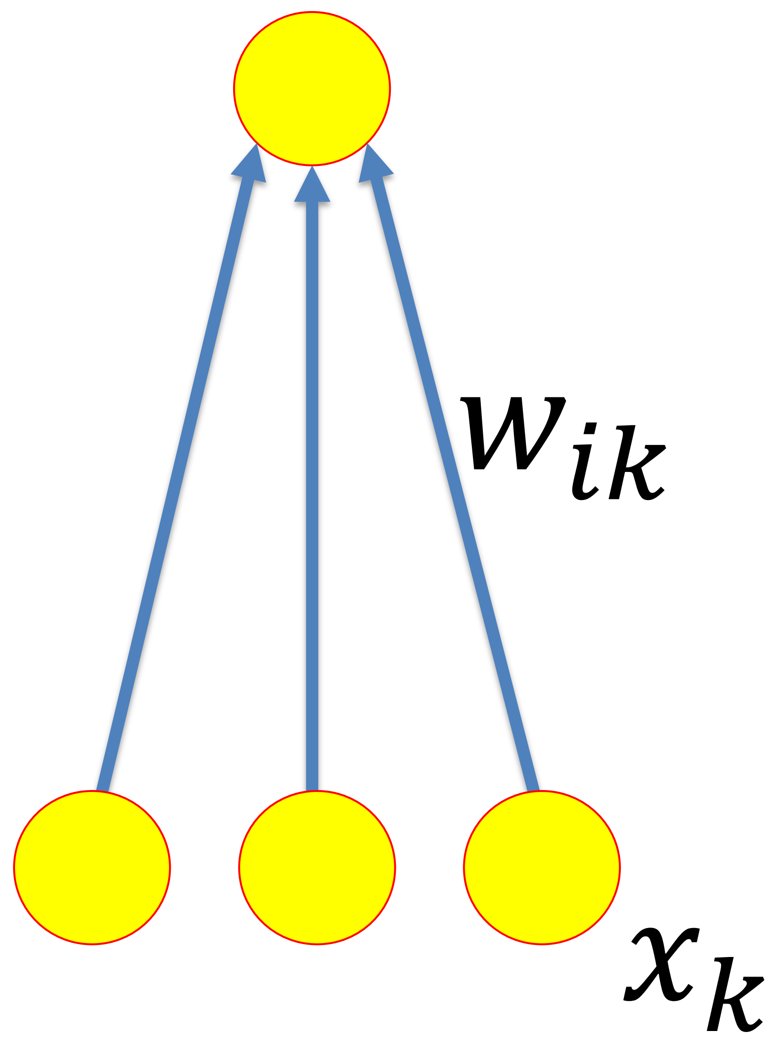
$$g(a') = \begin{cases} 1 & \text{if } a' > \vartheta \\ 0.5 & \text{if } a' = \vartheta \\ 0 & \text{if } a' > \vartheta \end{cases}$$

input

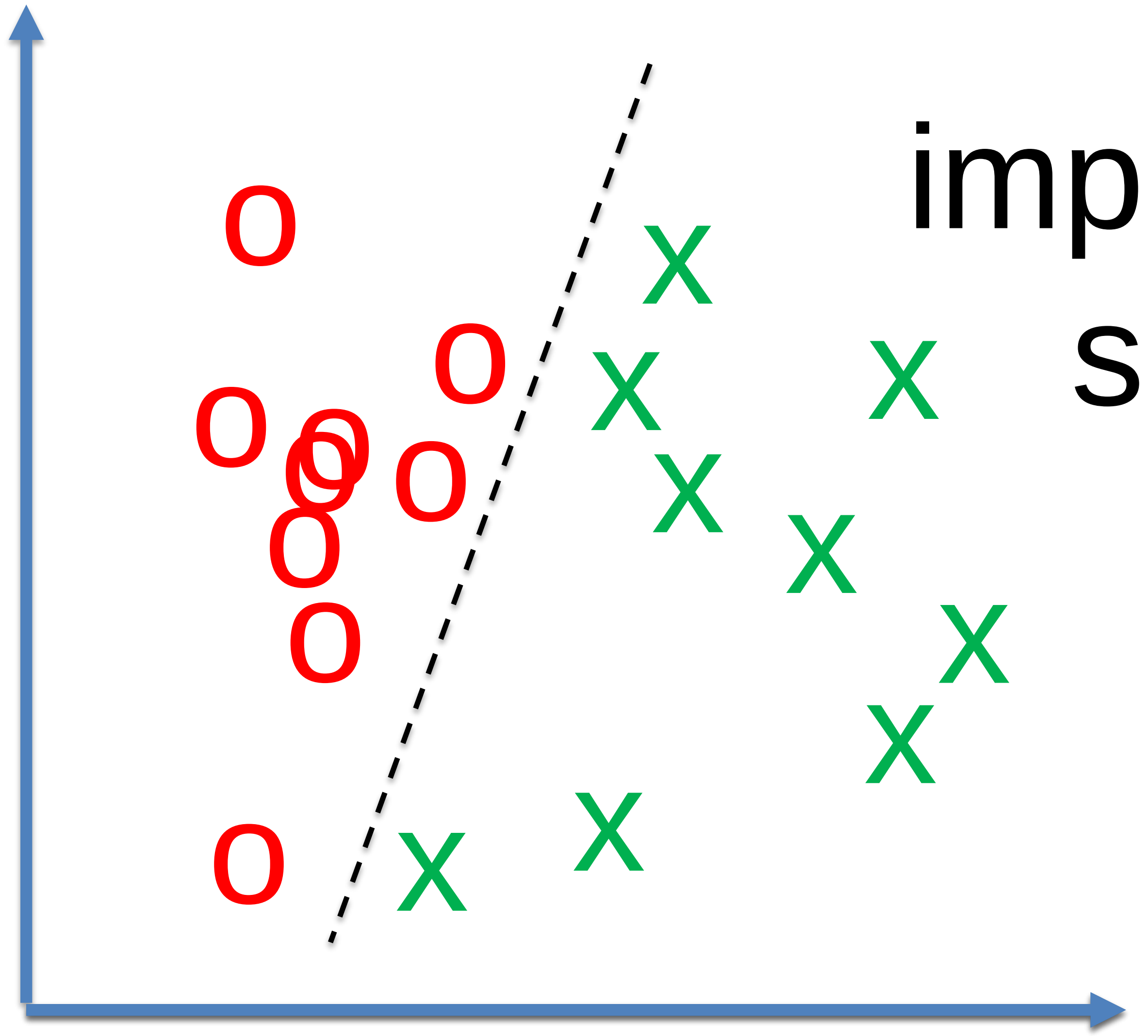
vector $\boldsymbol{x}$

3. Single-Layer networks: simple perceptron

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



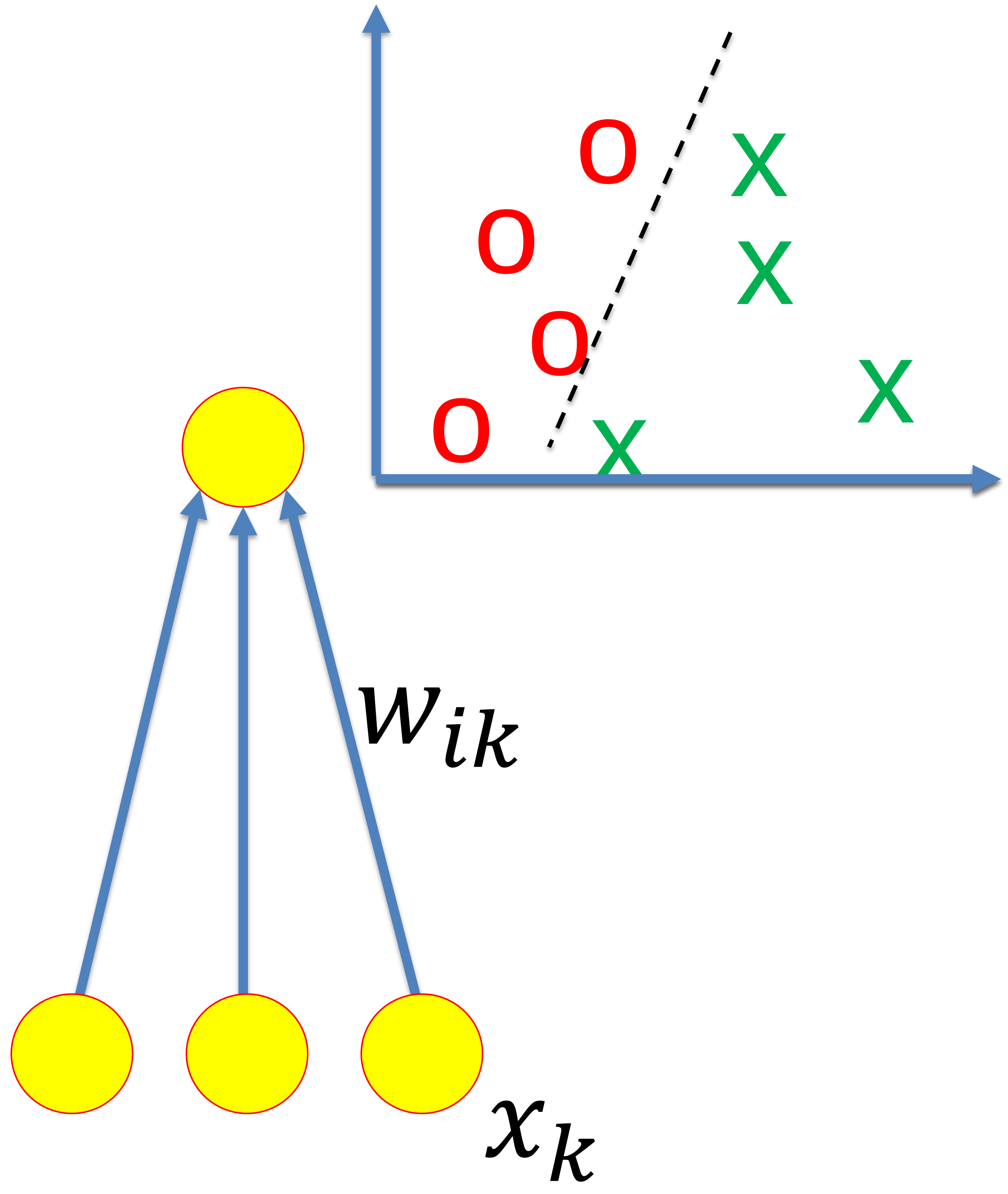
$$d(\boldsymbol{x}) = \sum_k w_k \, x_k - \vartheta = 0$$



imposes a linear separation

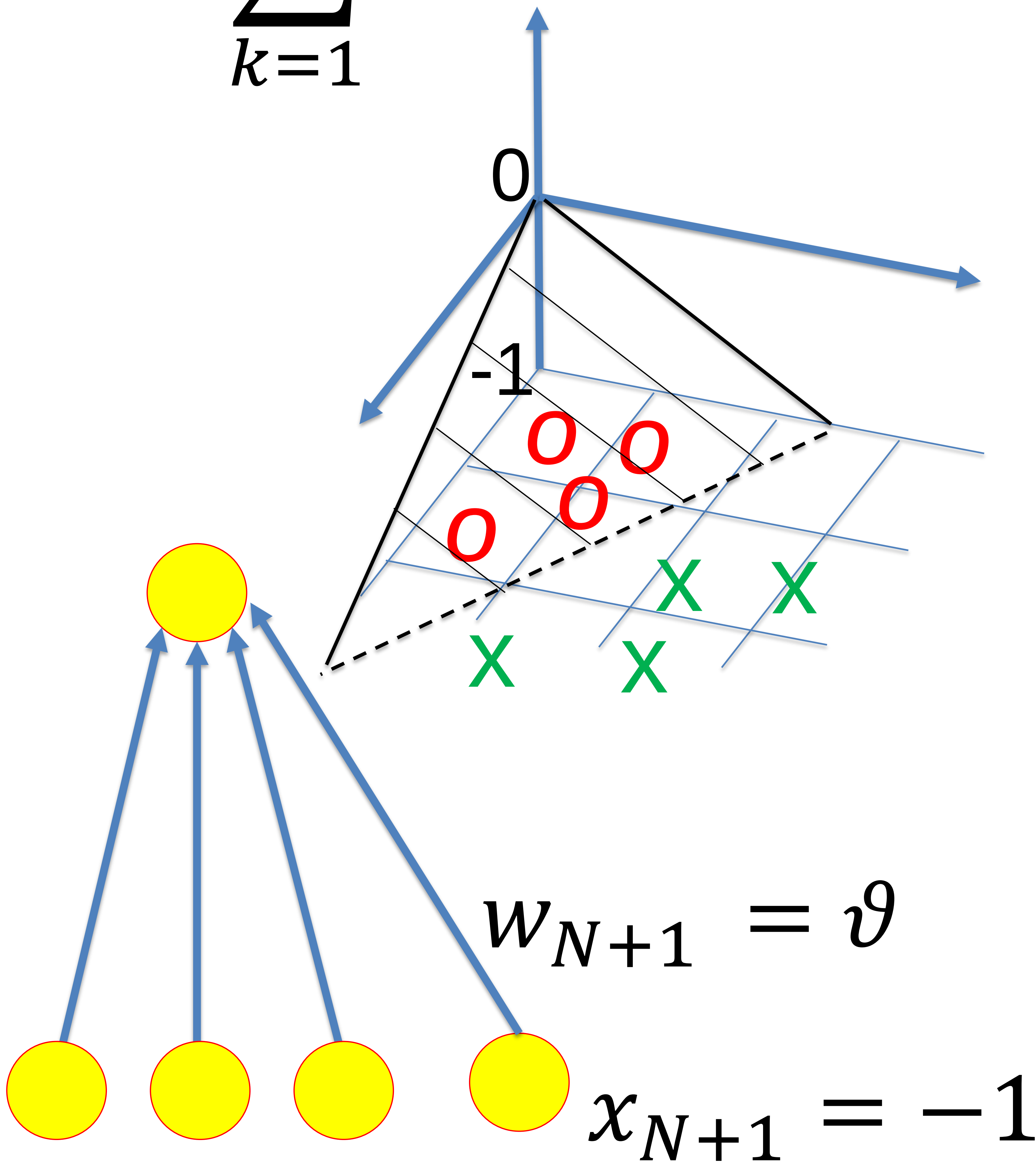
3. remove threshold: add a constant input

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



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

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



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

3. Single-Layer networks: simple perceptron

a simple perceptron

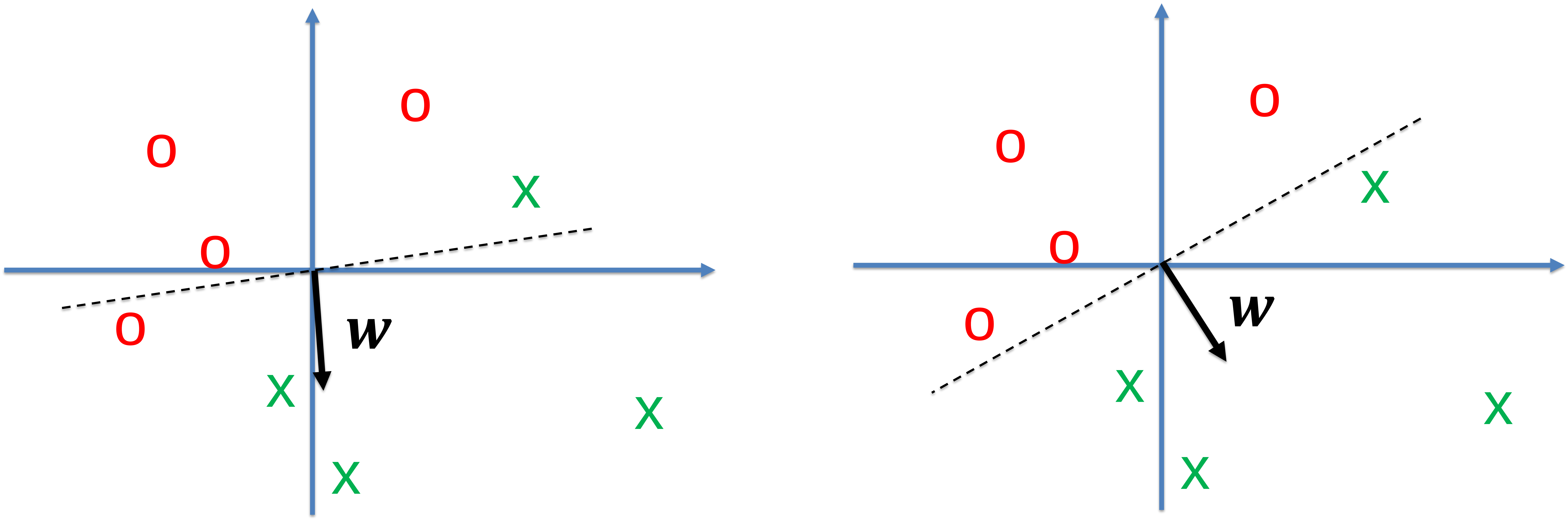
- can only solve linearly separable problems
- imposes a separating hyperplane
- for $\vartheta = 0$ hyperplane goes through origin
- threshold parameter ϑ can be removed by adding an input dimension

→ in ***N+1*** dimensions hyperplane always goes through origin

- We can **adapt the weight vector** to the problem

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)

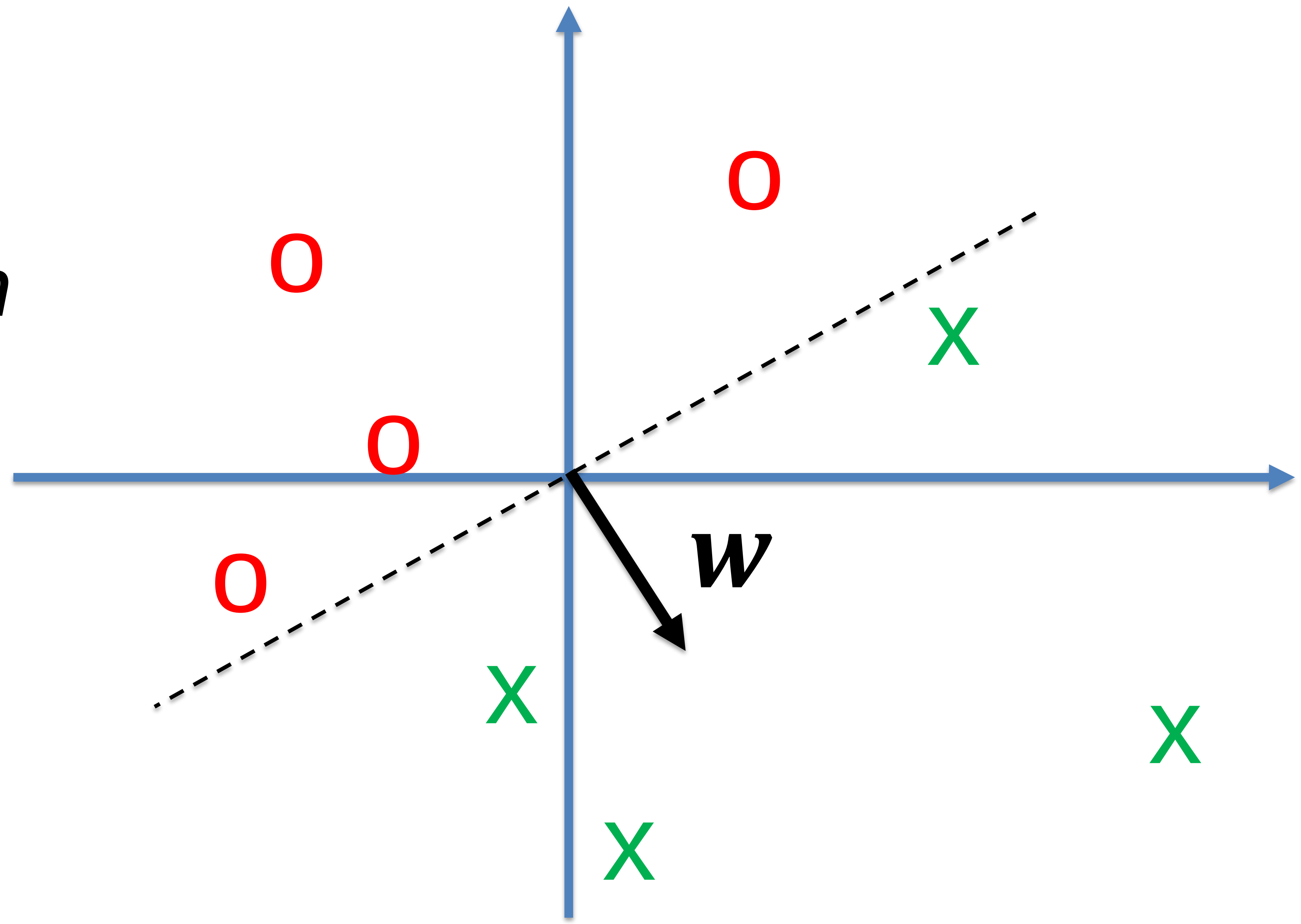
(2) stop if no changes for all ***P*** patterns

Blackboard 4:
geometry of the perc. algo

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 change of parameters corresponds always to a rotation of the separating hyperplane in N dimensions.

[ ] A change of parameters corresponds always to a rotation of the separating hyperplane in N+1 dimensions.

[ ] An increase of the length of the weight vector implies an increase of the distance of the hyperplane from the origin in N dimensions.

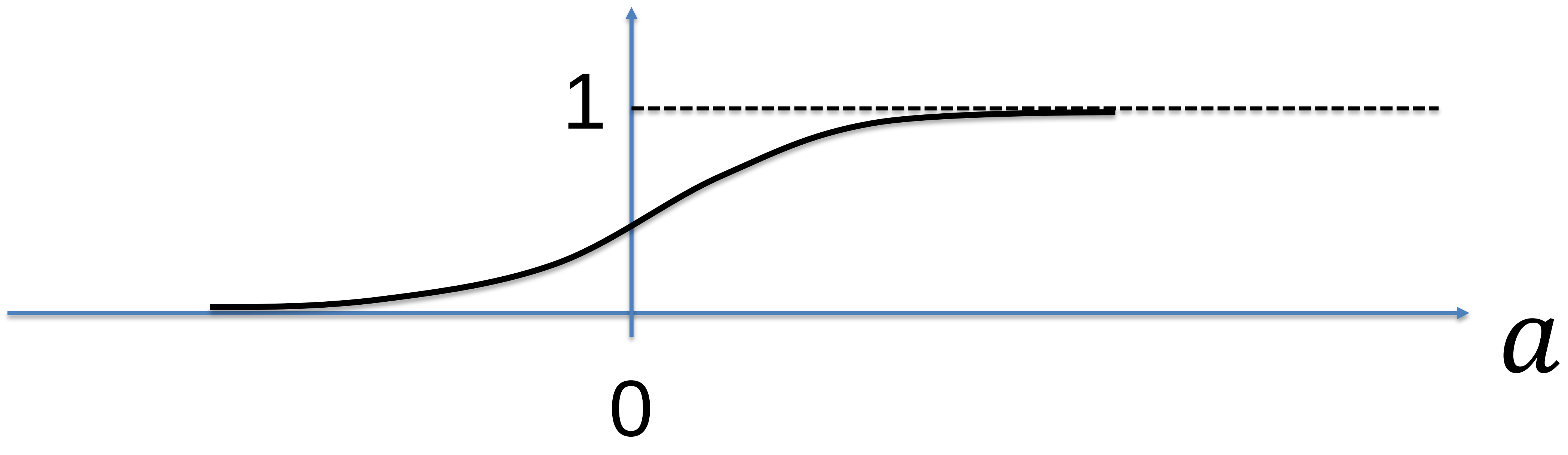
[ ] An increase of the length of the weight vector implies that the hyperplane does not change in N dimensions

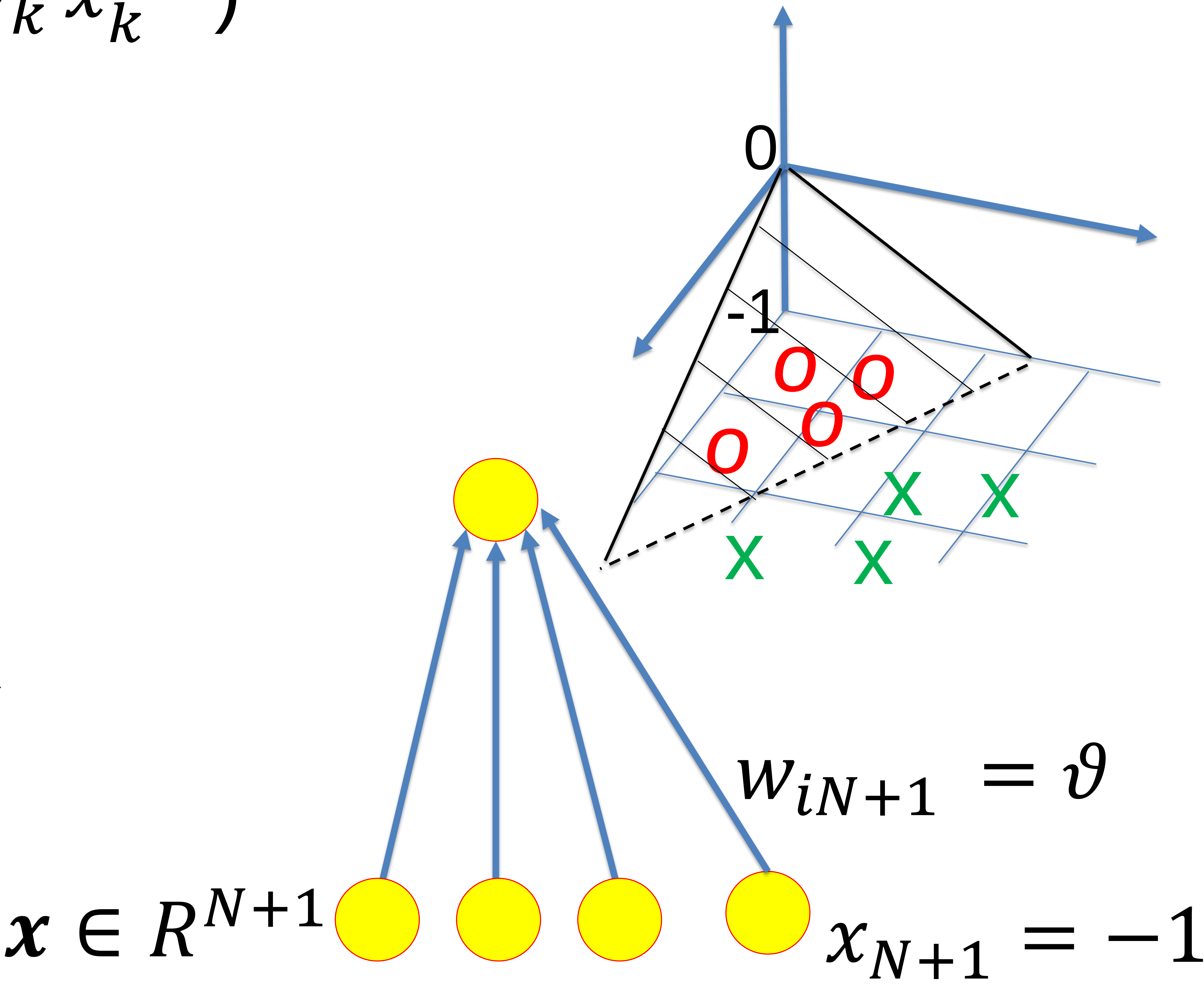
[ ] An increase of the length of the weight vector implies that the hyperplane does not change in N+1 dimensions

5. Sigmoidal output unit

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

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





5. Supervised learning with sigmoidal output

target output $t^7 = 1$ ⟷ error! ⟷ $\hat{y}^7 = 0.2$ classifier output

output

$\boldsymbol{f}(\boldsymbol{x}^7) = g(\boldsymbol{w}^T \boldsymbol{x}^\mu)$

Classifier

teacher

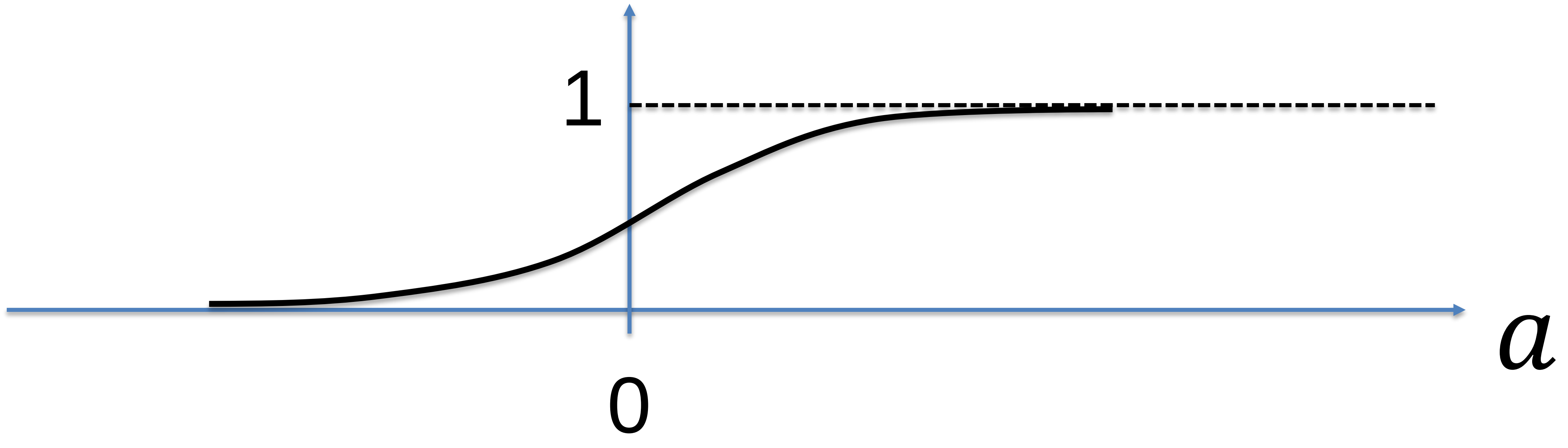
input

$\boldsymbol{x}^7$

5. Supervised learning with sigmoidal output

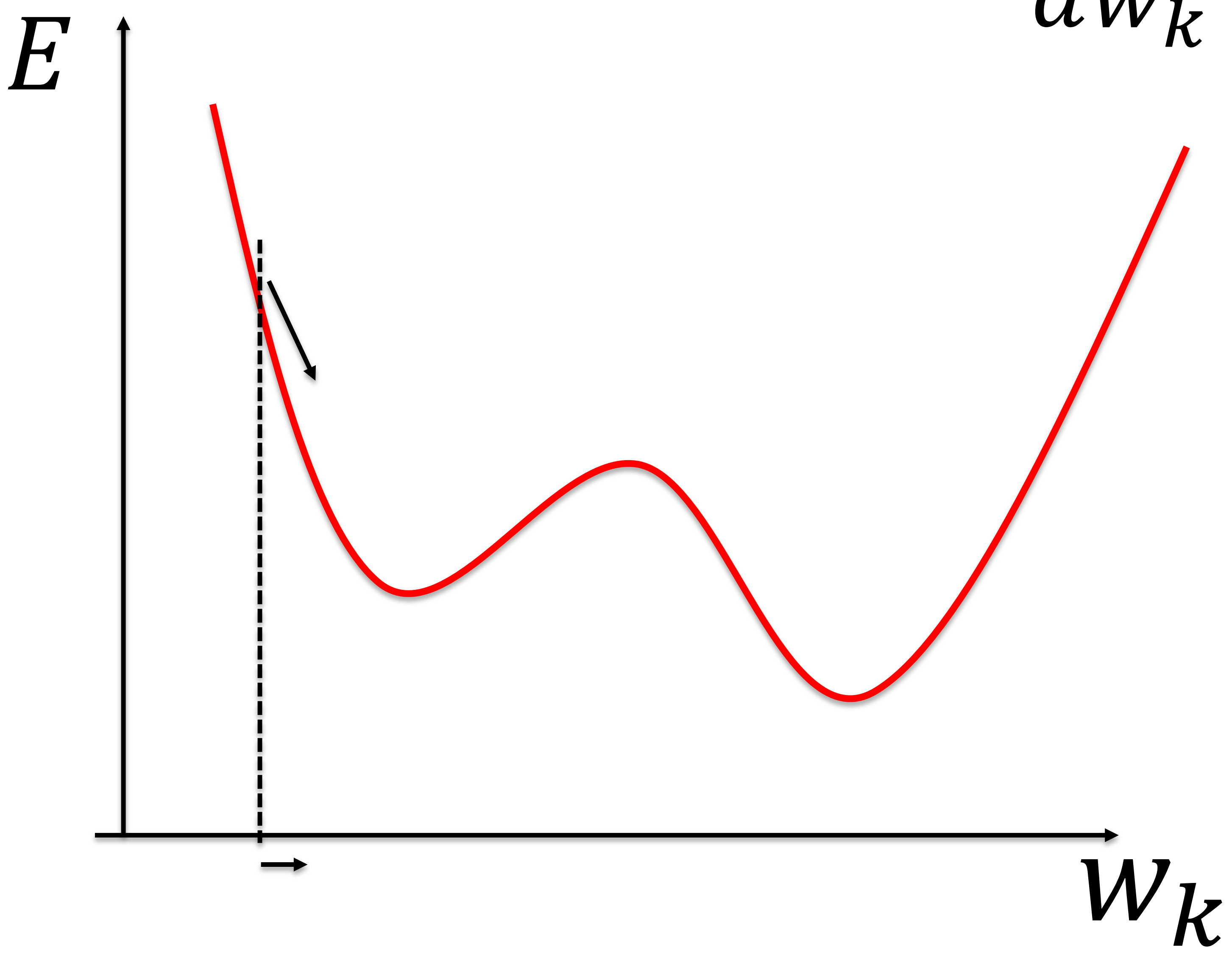
define error

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

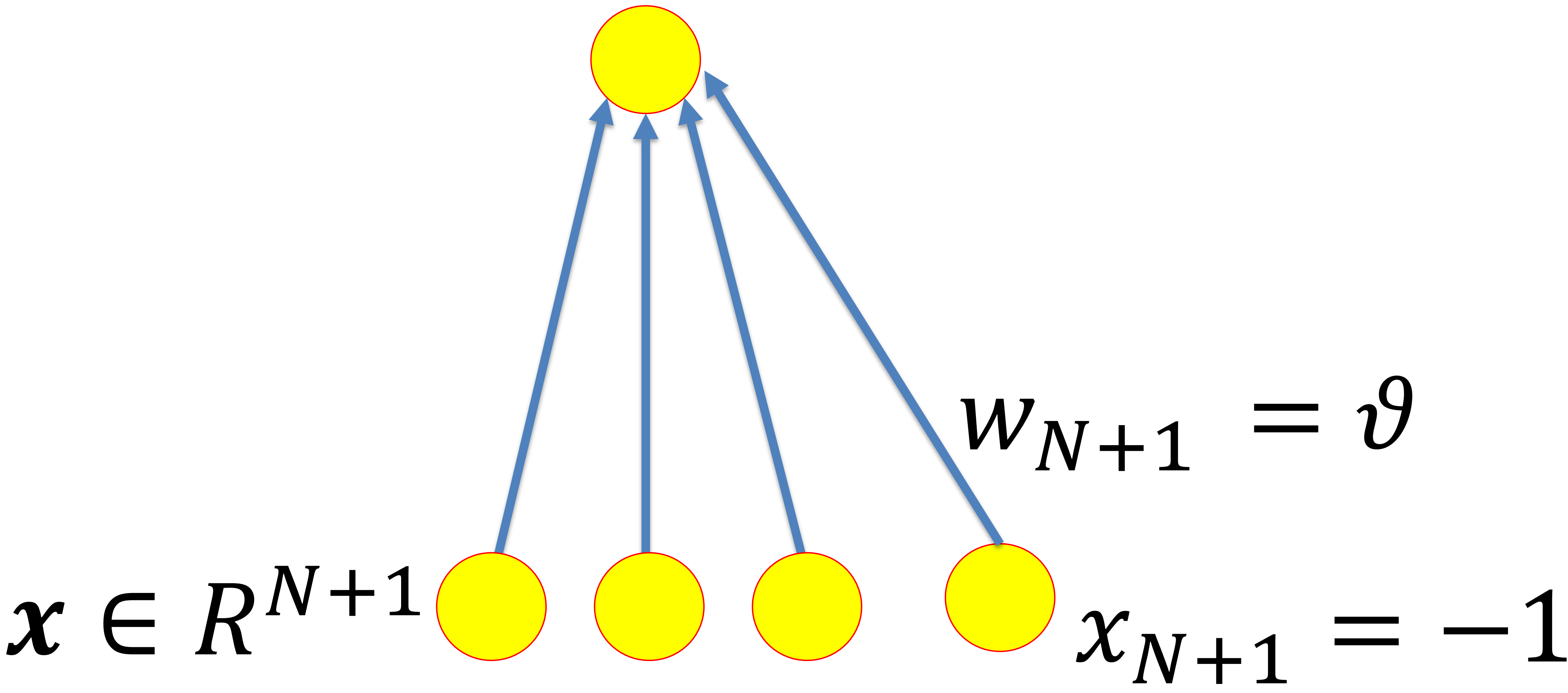


gradient descent

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



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



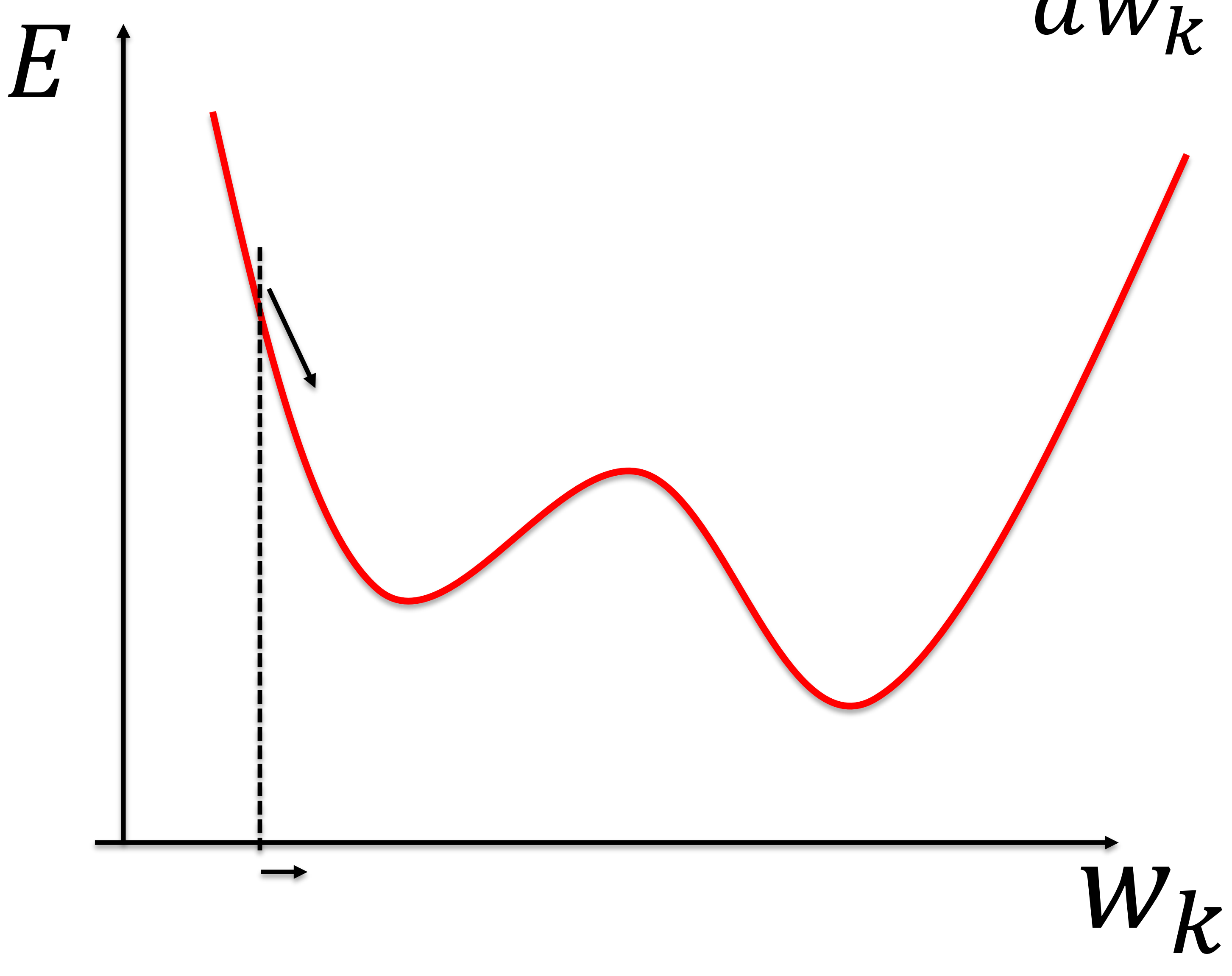
6. gradient descent

Quadratic **error**

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

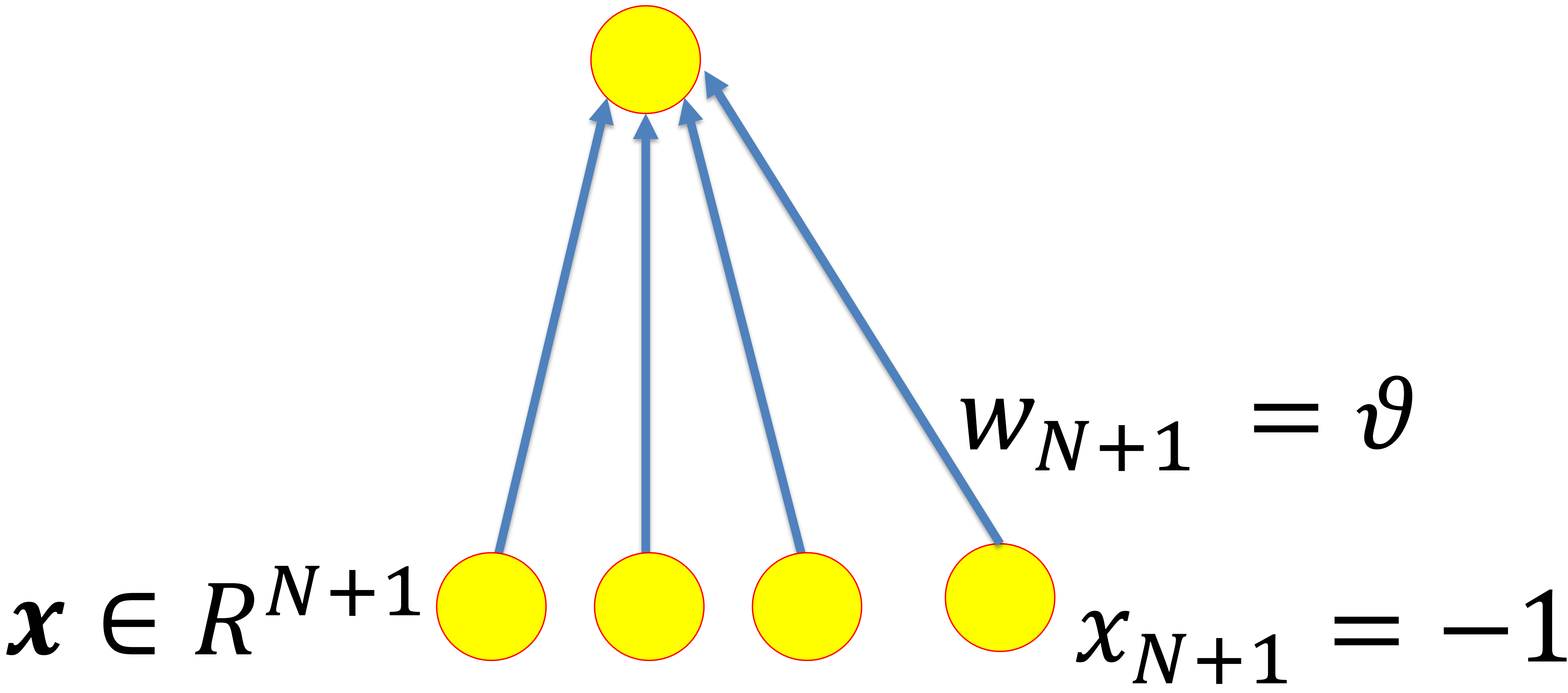
gradient descent

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



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

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



Exercise 1.1 now:

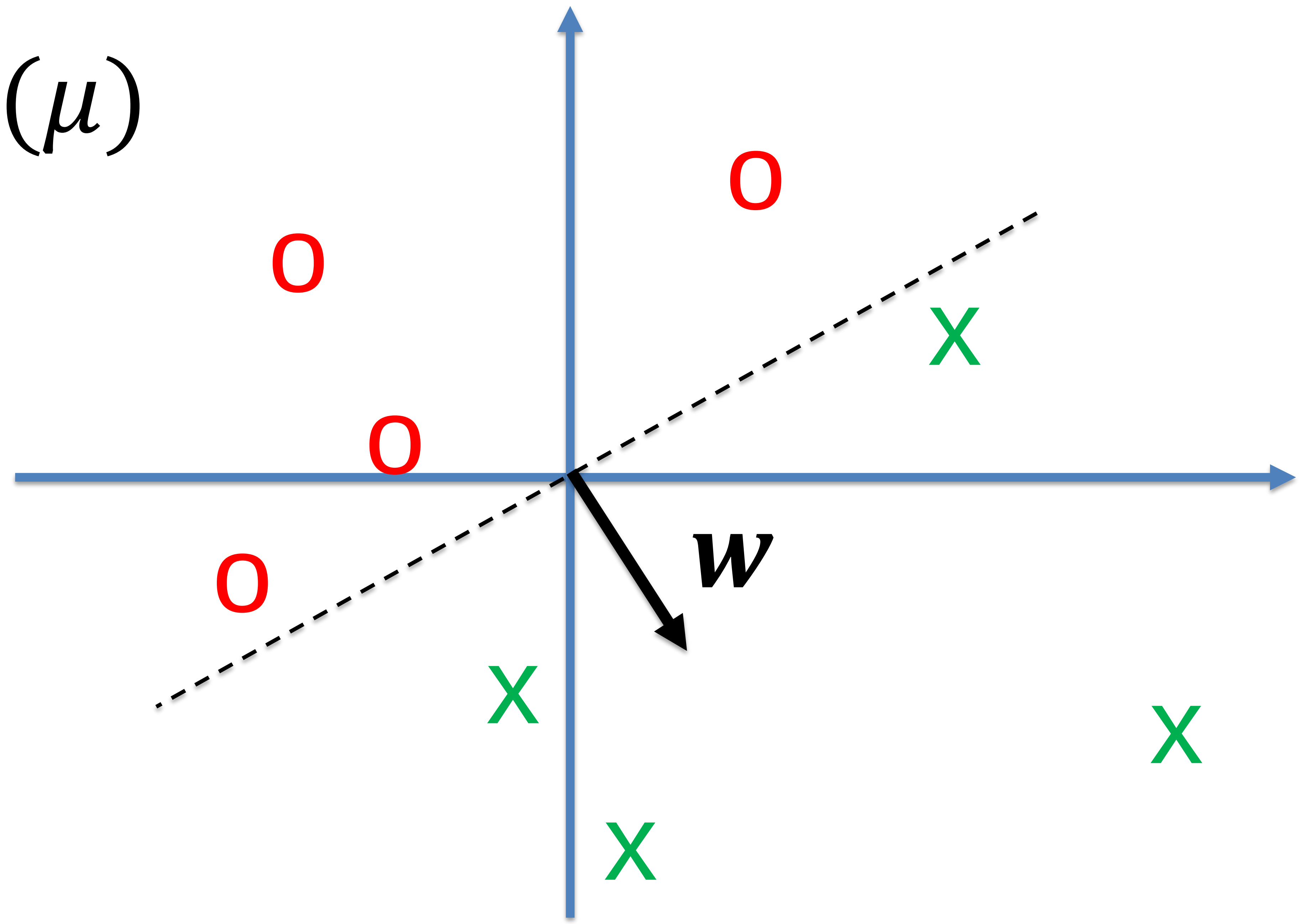
- calculate gradient (1 pattern)

- geometrical interpretation?

6. Gradient descent algorithm

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

- stepsize depends on (signed) output mismatch $\delta(\mu)$ for this data point
- change implemented even if ‘correctly classified
- compare with perceptron algorithm



Summary for today:

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

Reading for this week:

Bishop, Ch. 4.1.7 of

Pattern recognition and Machine Learning

or

Bishop, Ch. 3.1-3.5 of

Neural networks for pattern recognition

Motivational background reading:

Silver et al. 2017, Archive

Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm

Goodfellow et al., Ch. 1 of

Deep Learning

The END