Artificial Neural Networks

Wulfram Gerstner

EPFL, Lausanne, Switzerland

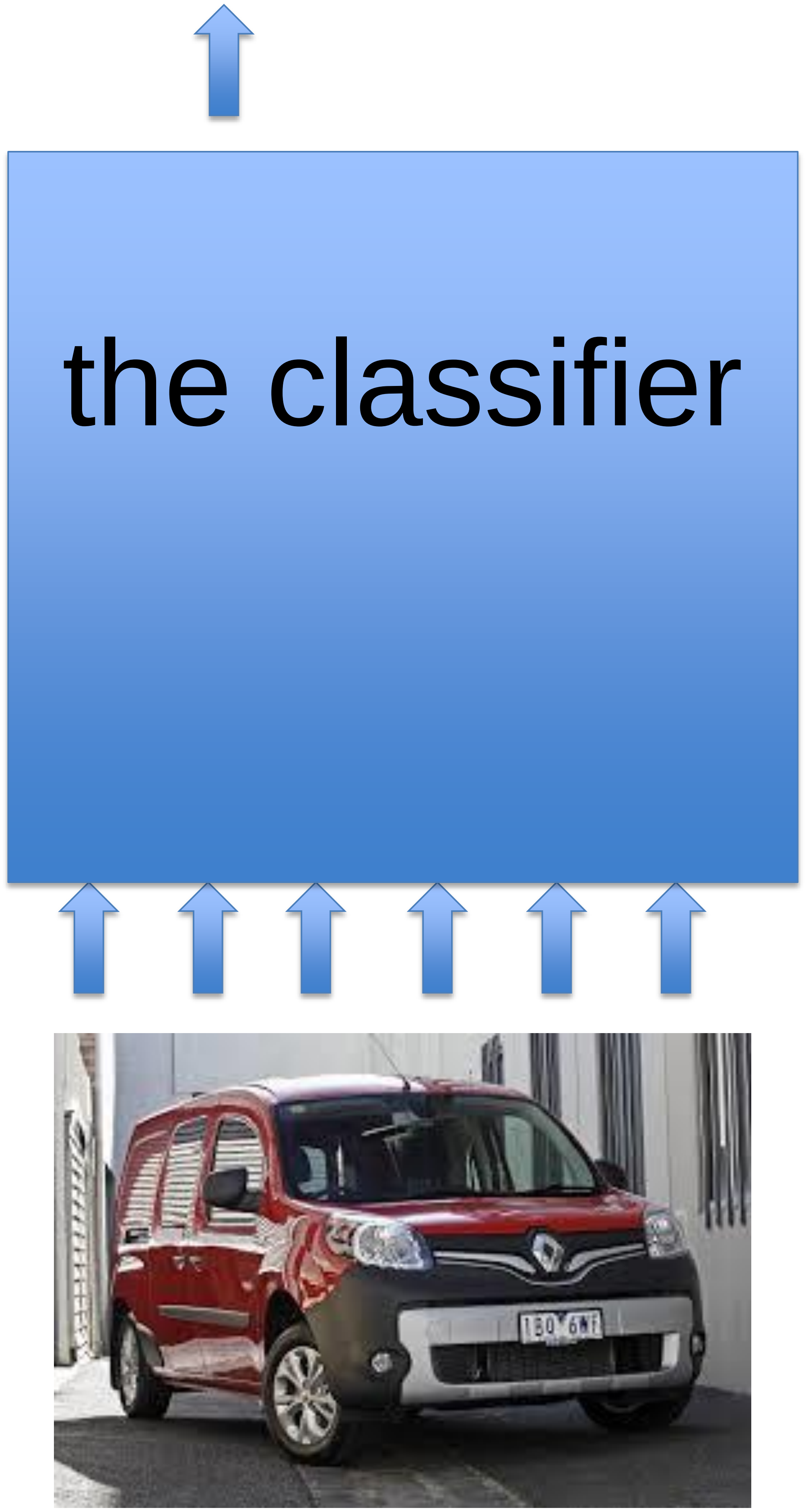
Supervised learning, classification, simple perceptron

1. Classification as a geometric problem

The problem of Classification

car (yes or no)

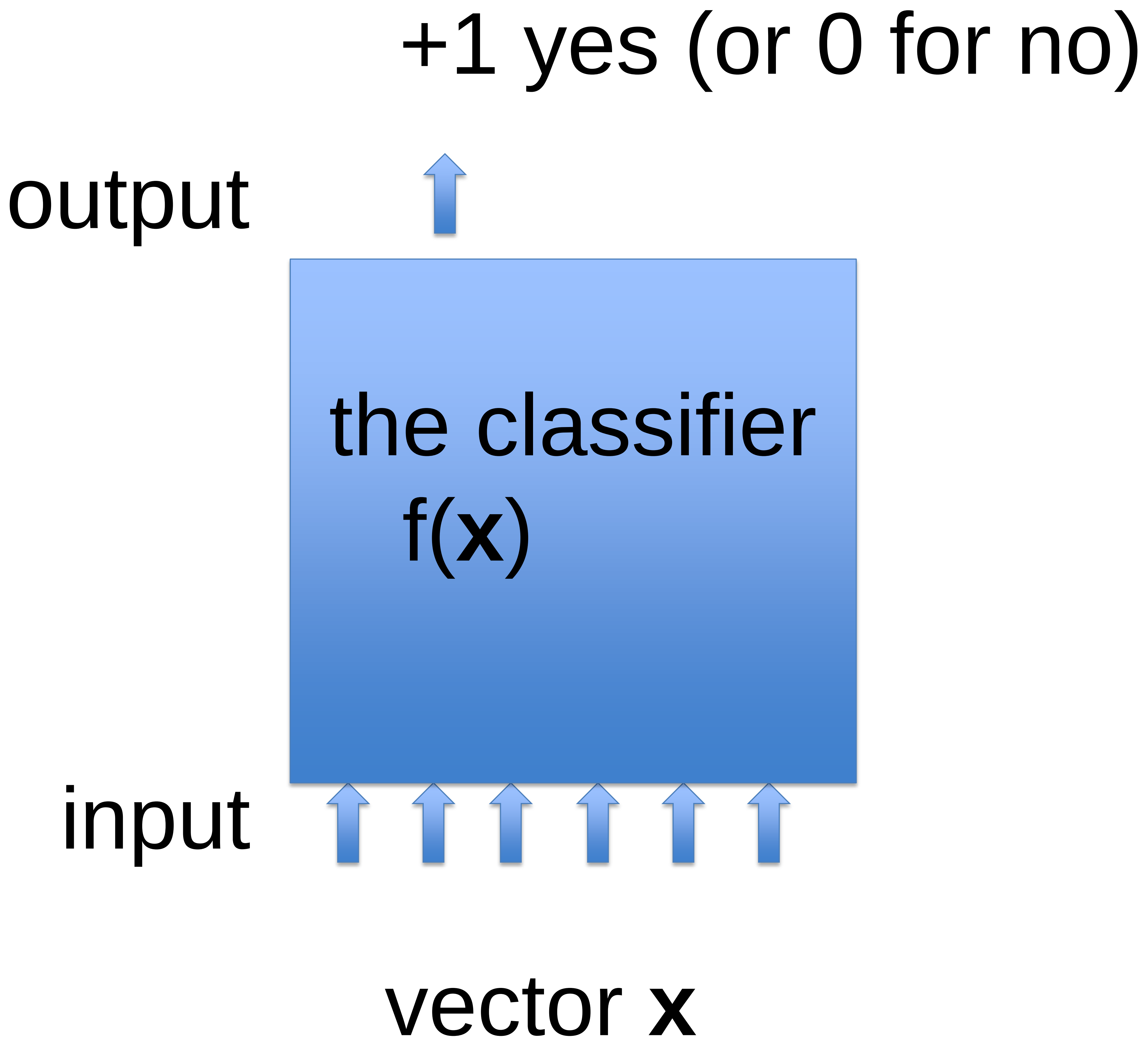
output



the classifier

input

The problem of Classification



+1 yes (or 0 for no)

output

the classifier
f(**x**)

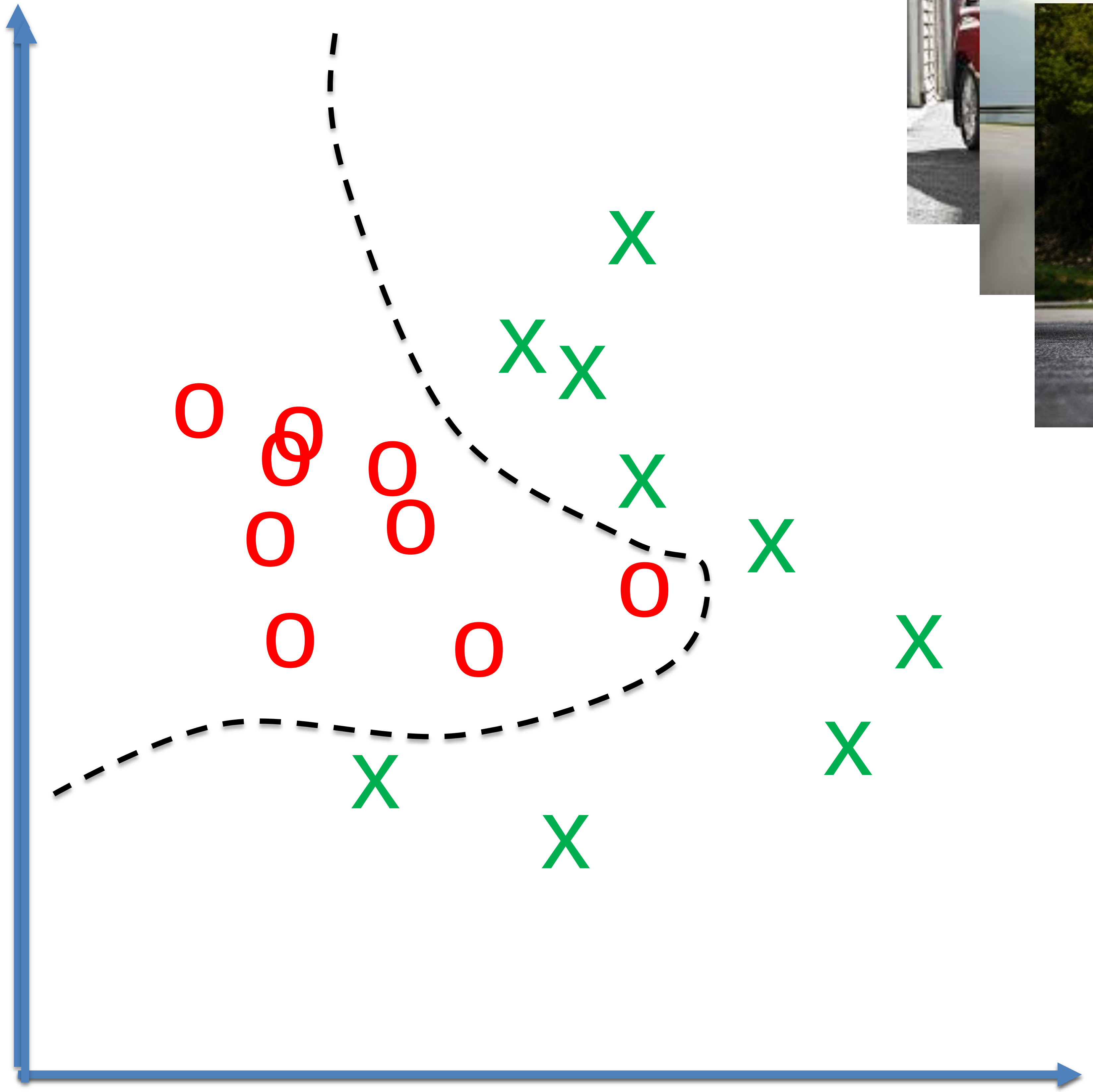
input

vector **x**

Blackboard 1: from images to vector

Blackboard 2: from vectors to classification

Classification as a geometric problem



Blackboard 1:

from images to vector

Blackboard 2:

from vectors to classification

Classification as a geometric problem

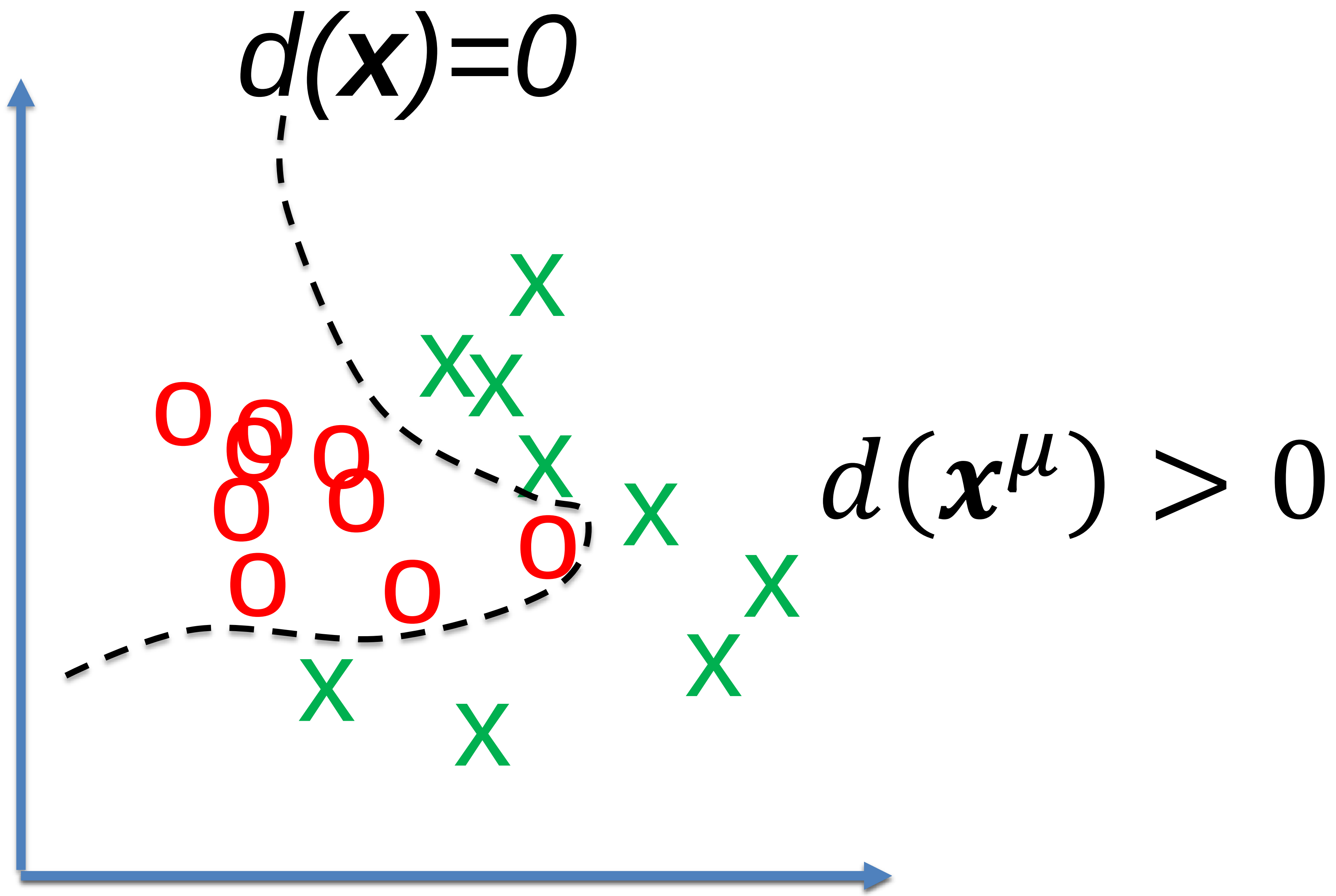
Task of Classification

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

Classification by **discriminant function** *d*(***x***)

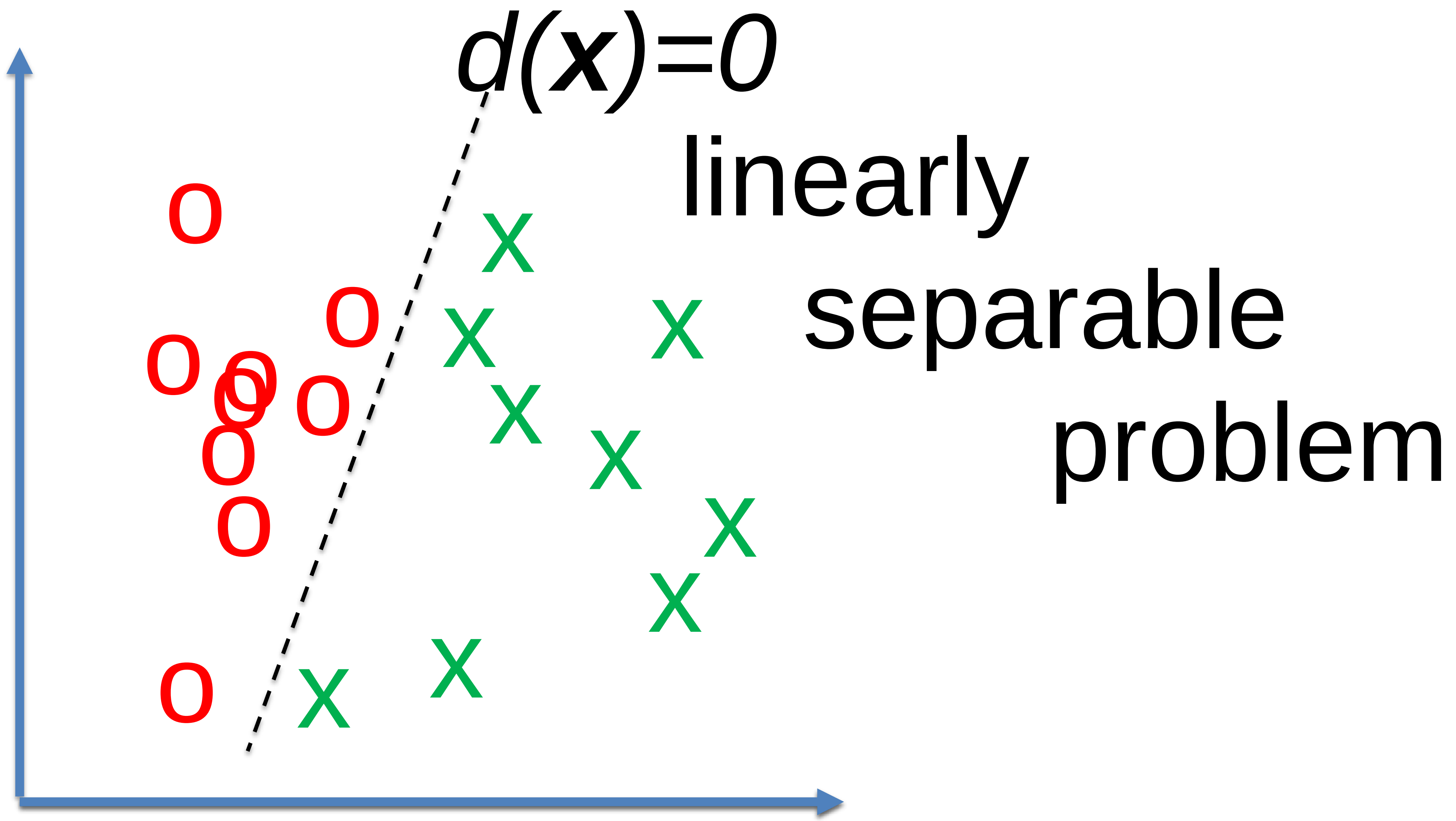
→ *d*(***x***)=*0* on this surface; *d*(***x***)>*0 for all positive examples* ***x***

d(***x***)<*0 for all counter examples* ***x***



d(***x***)=*0*

$d(\boldsymbol{x}^{\mu}) > 0$



d(***x***)=*0*

linearly separable problem

Artificial Neural Networks

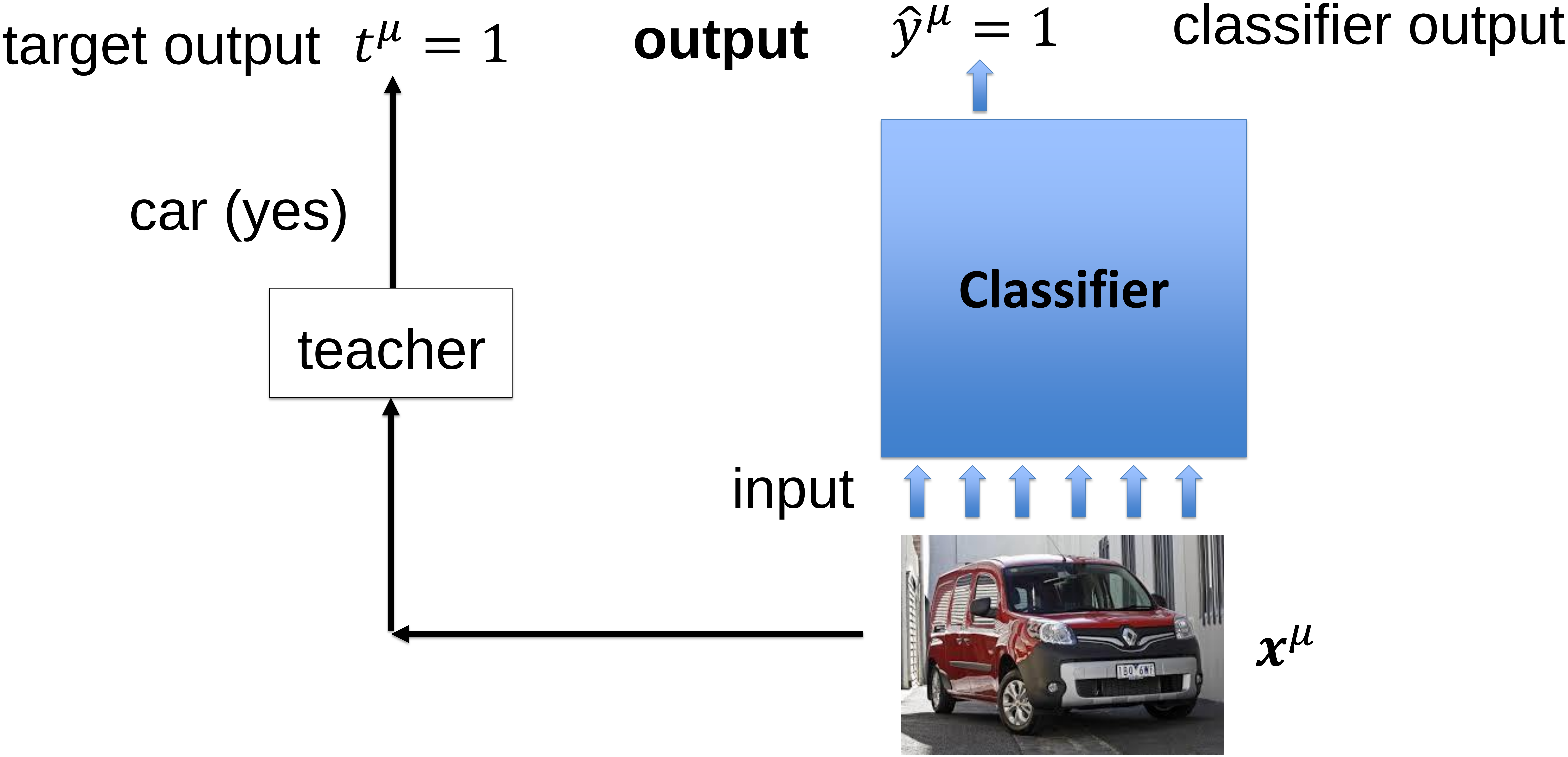
Wulfram Gerstner

EPFL, Lausanne, Switzerland

Supervised learning, classification, simple perceptron

1. Classification as a geometric problem
2. Supervised learning

Data base for Supervised learning



target output $t^{\mu} = 1$

output $\hat{y}^{\mu} = 1$ classifier output

car (yes)

teacher

Classifier

input

$\boldsymbol{x}^{\mu}$

Supervised learning

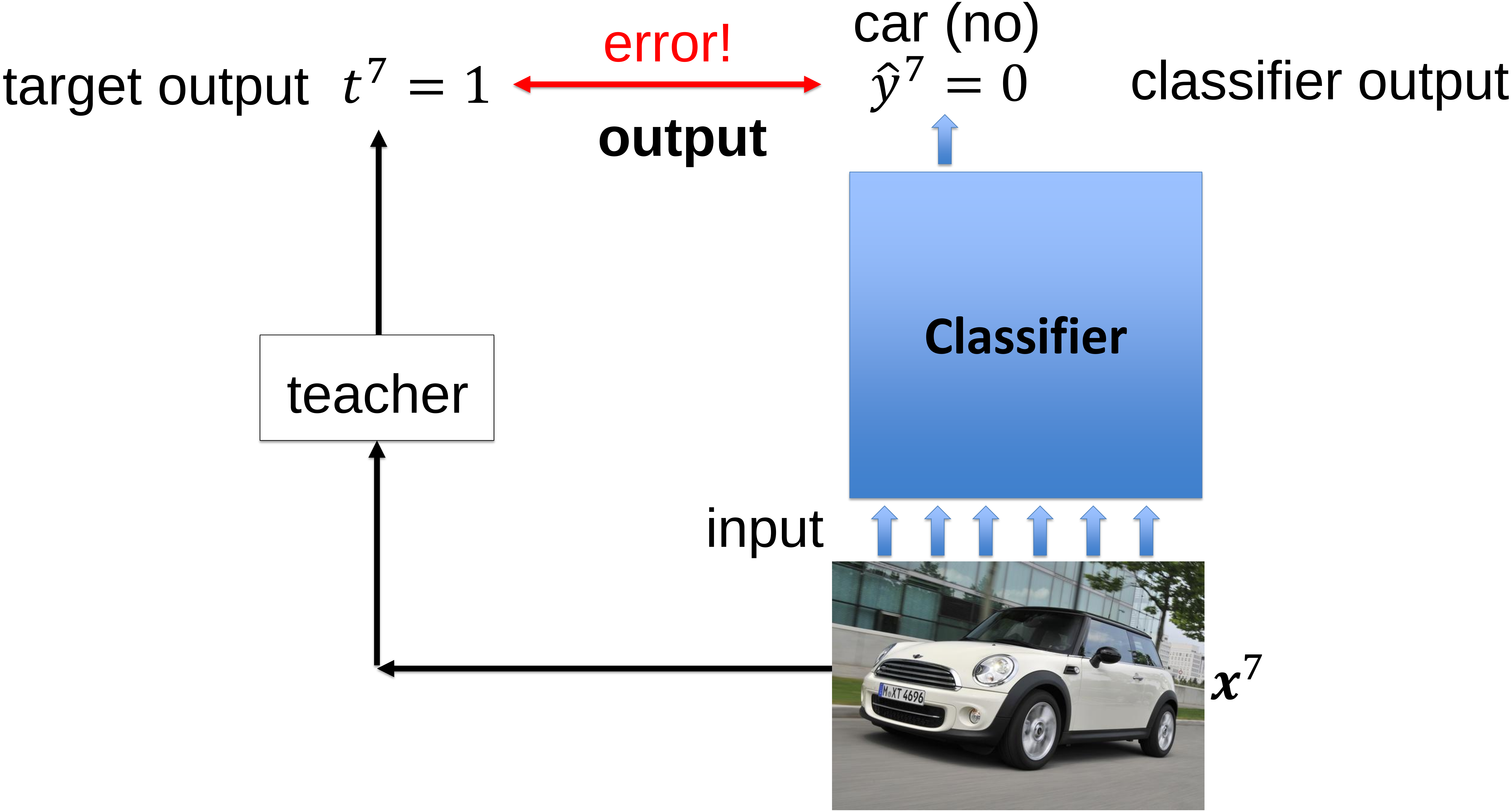
P data points $\{ \quad (\boldsymbol{x}^\mu, t^\mu) \quad , \qquad 1 \le \mu \le P \quad \};$

input target output

$t^\mu = 1$ car =yes

$t^\mu = 0$ car =no

Data base for Supervised learning



car (no)

target output $t^7 = 1$

error!

$\hat{y}^7 = 0$

classifier output

output

teacher

Classifier

input

x^7

Error in Supervised learning

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

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: Errors can be used to define a ‘Loss function’.

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

Summary: Supervised learning

1. Data base $\{\ (\boldsymbol{x}^{\mu}, t^{\mu})\ ,\quad 1 \leq \mu \leq P\ \};$

input target output

2. A way to measure errors

for $\boldsymbol{x}^{\mu}$ compare classifier output $\hat{y}^{\mu}$ with t^{μ}

$\sum_{\mu} E\,(\hat{y}^{\mu}, t^{\mu})$ Error function/Loss function

3. A method to minimize the errors

Artificial Neural Networks

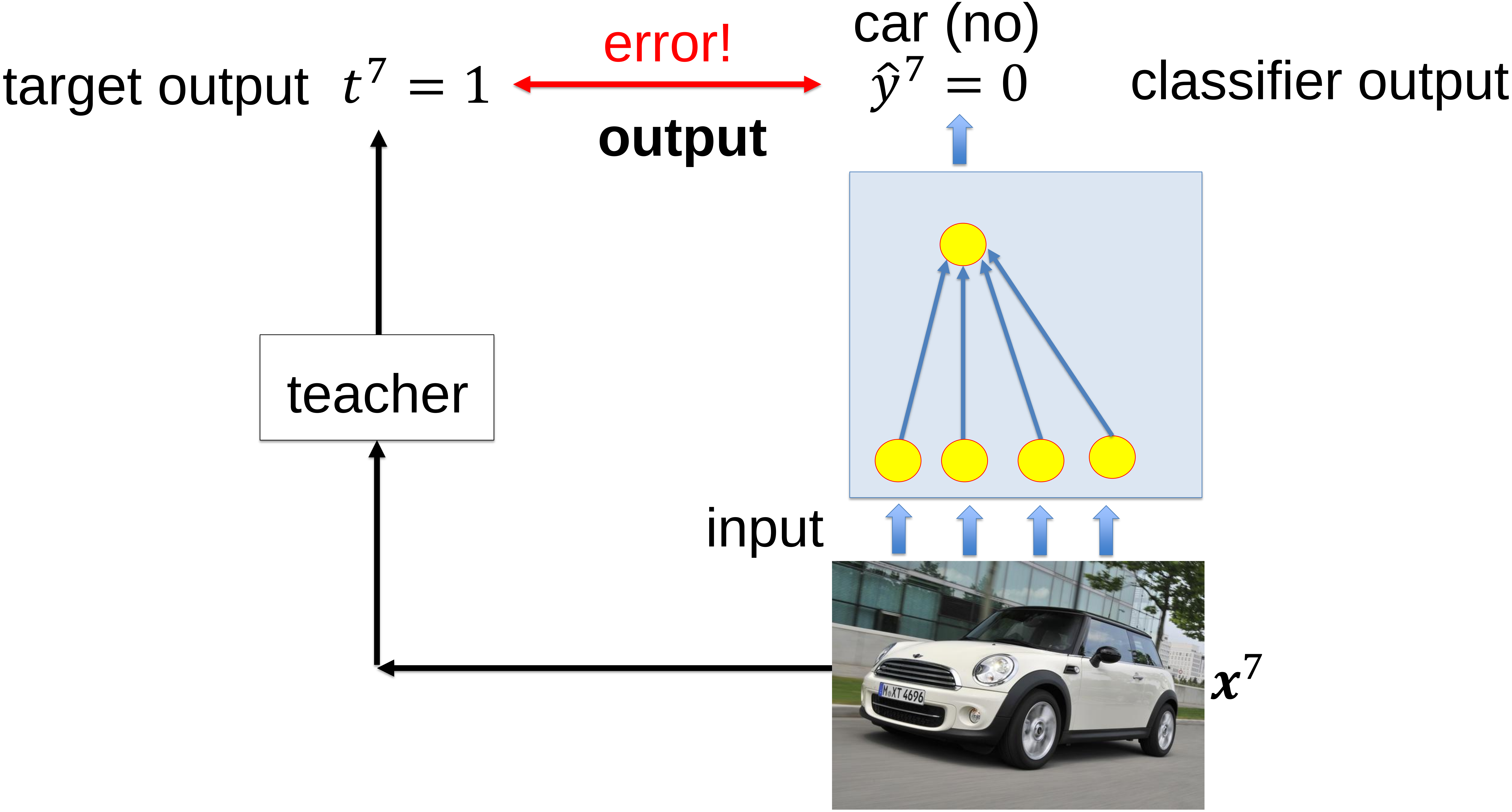
Wulfram Gerstner

EPFL, Lausanne, Switzerland

Supervised learning, classification, simple perceptron

1. Classification as a geometric problem
2. Supervised learning
3. **Gradient descent for a single sigmoidal output unit**

Classifier = neural network with 1 single neuron



target output $t^7 = 1$

error!

output

car (no)

$\hat{y}^7 = 0$

classifier output

teacher

input

$\boldsymbol{x}^7$

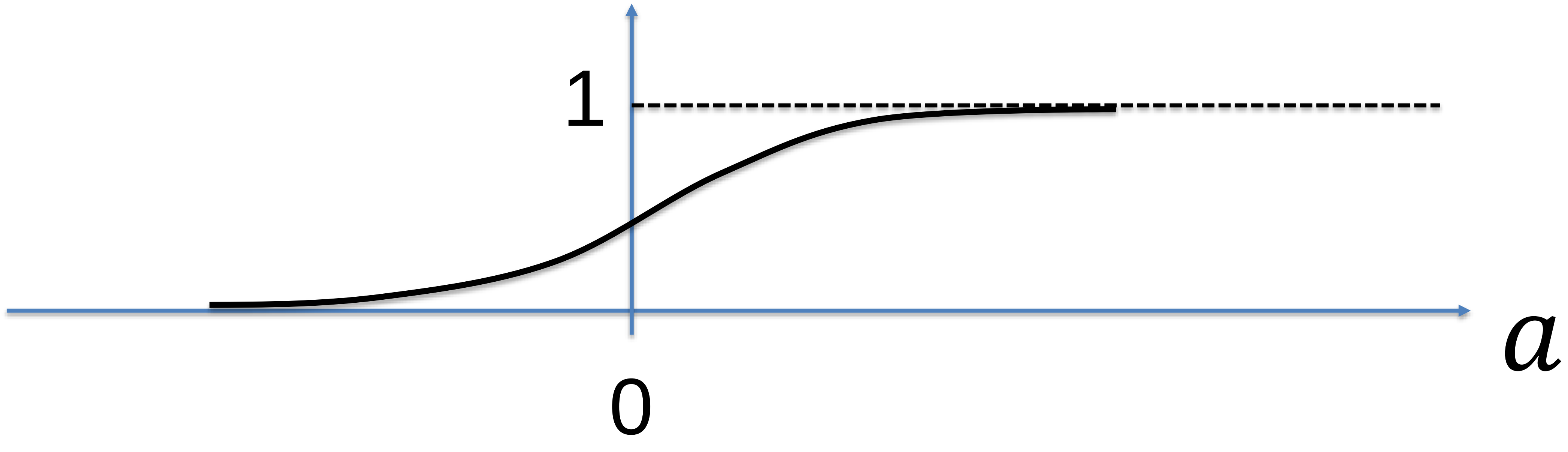
Sigmoidal output unit

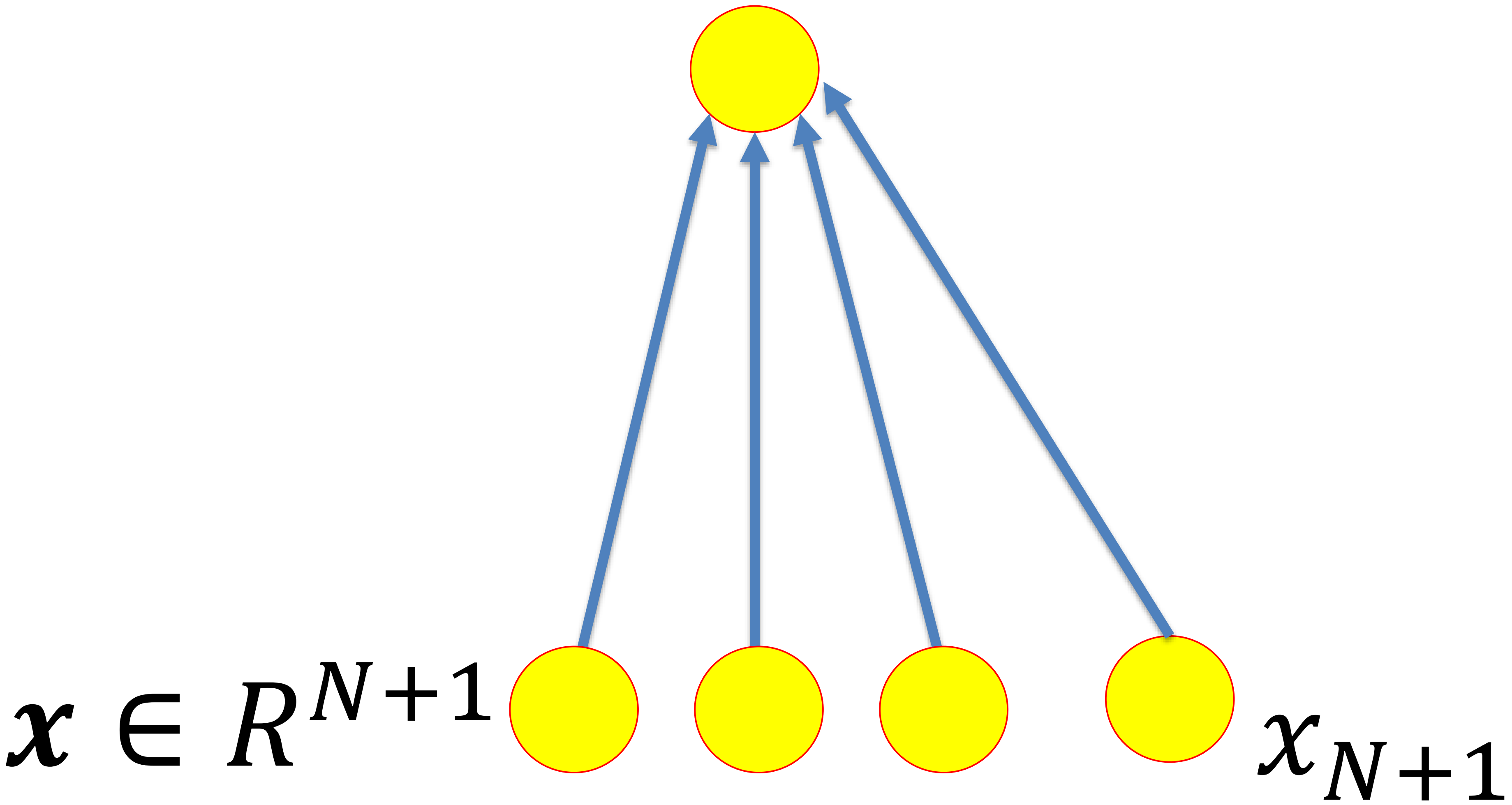
A saturating nonlinear function with a smooth transition from 0 to 1.

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

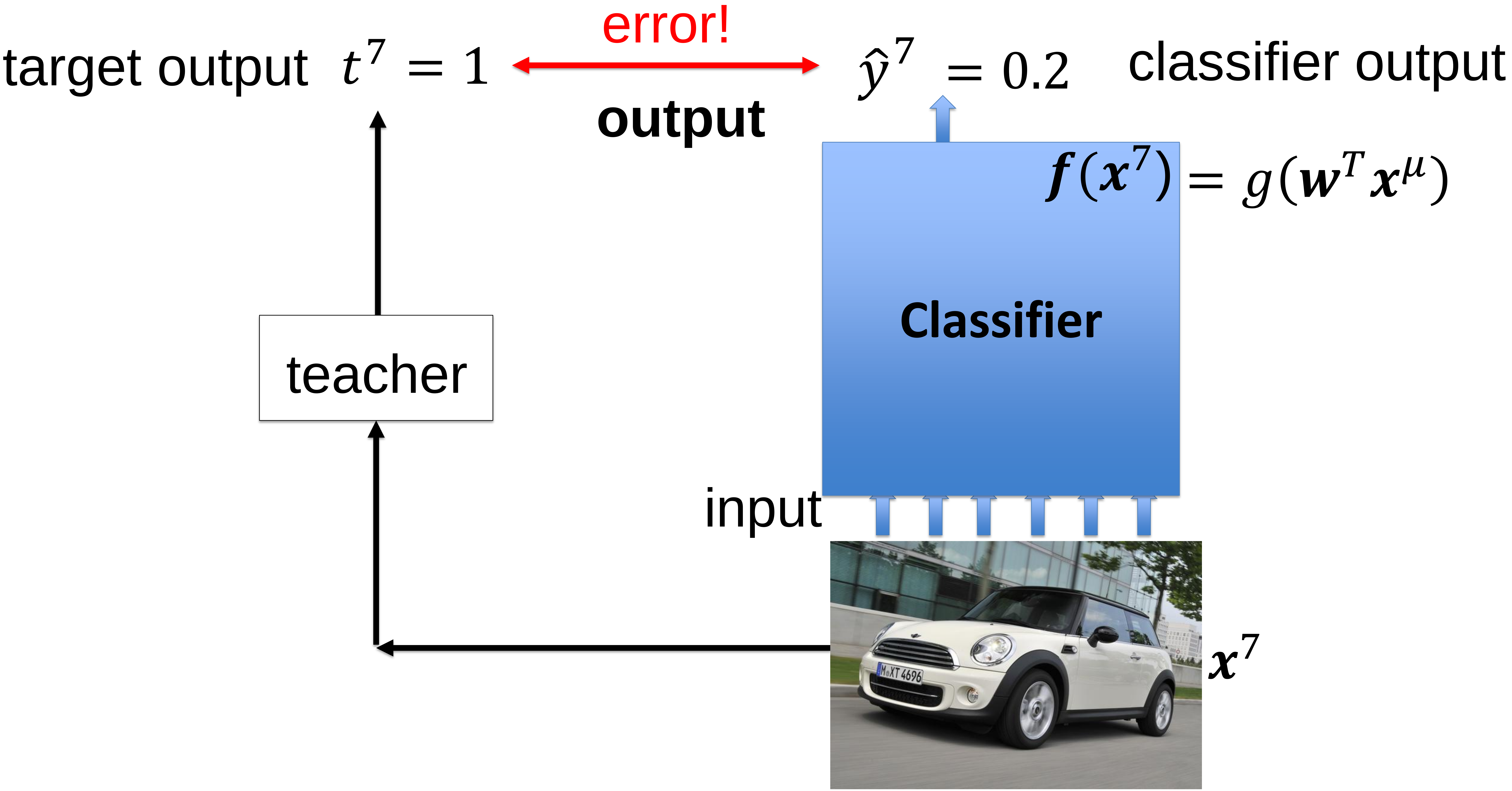
with

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





Supervised learning with sigmoidal output



error!

target output $t^7 = 1$

output

$\hat{y}^7 = 0.2$ classifier output

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

Classifier

teacher

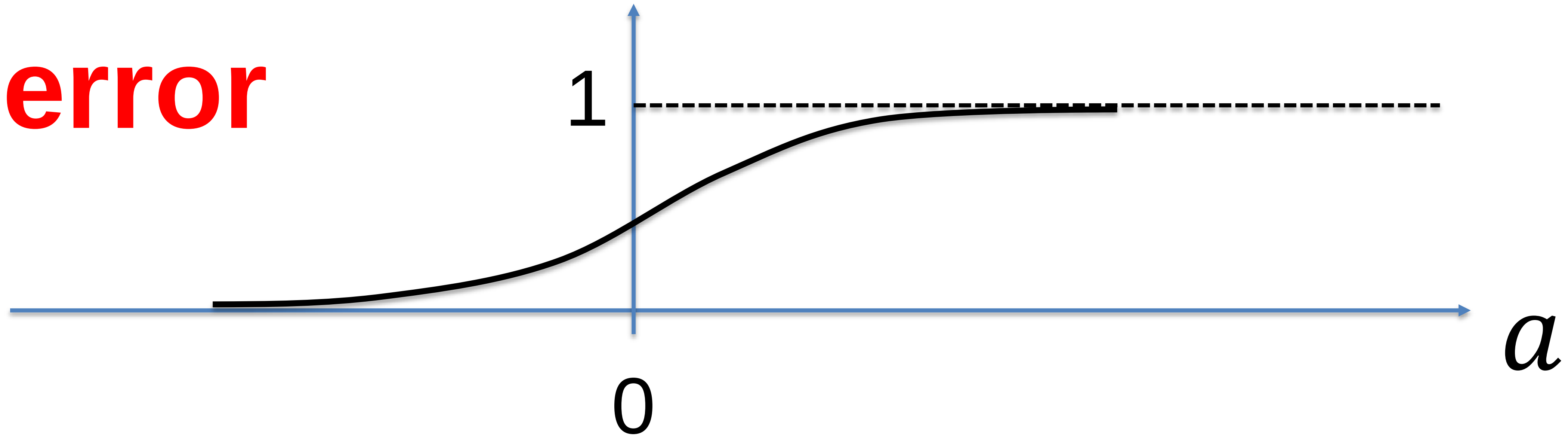
input

$\boldsymbol{x}^7$

Supervised learning with sigmoidal output

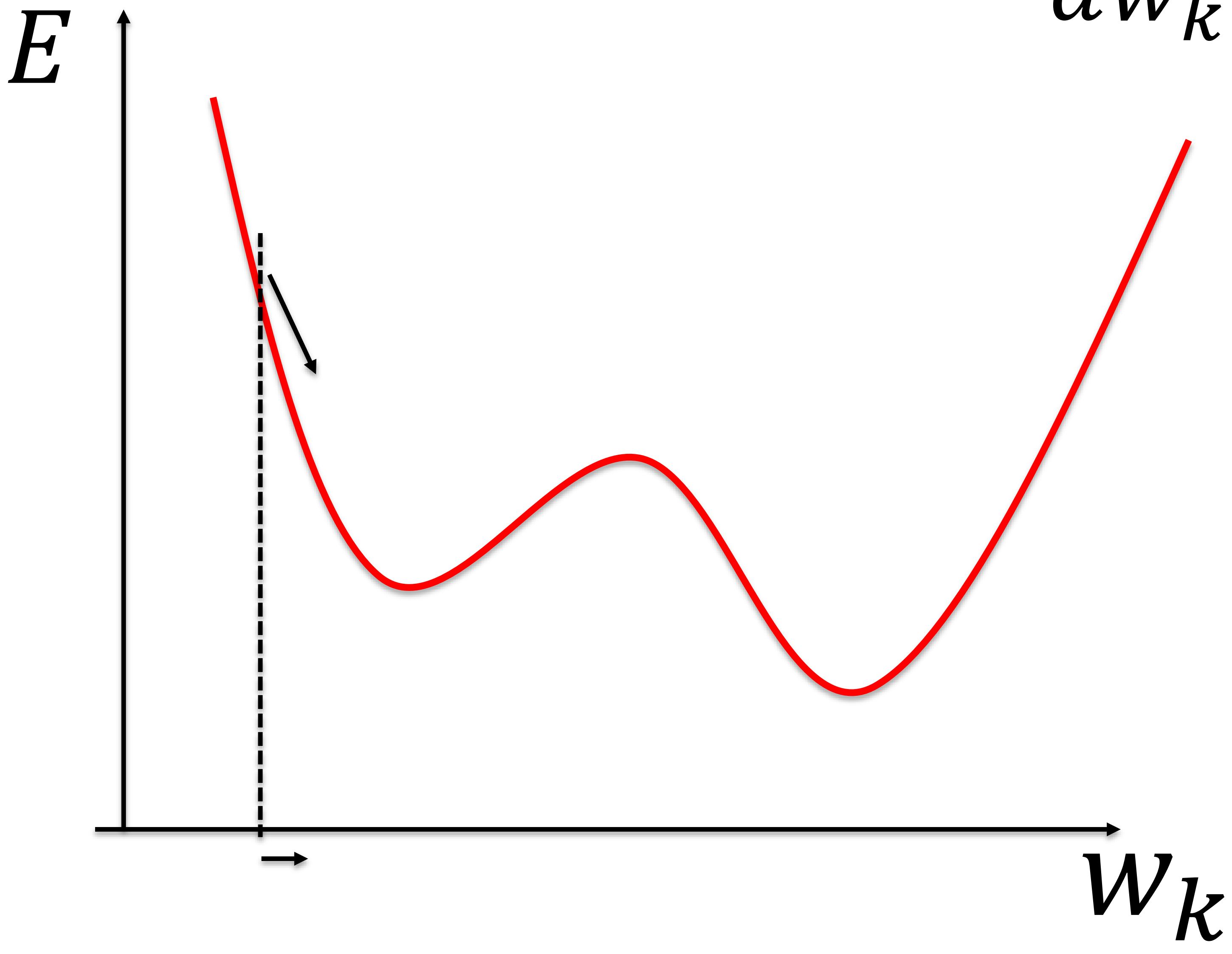
Loss function: define quadratic **error**

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

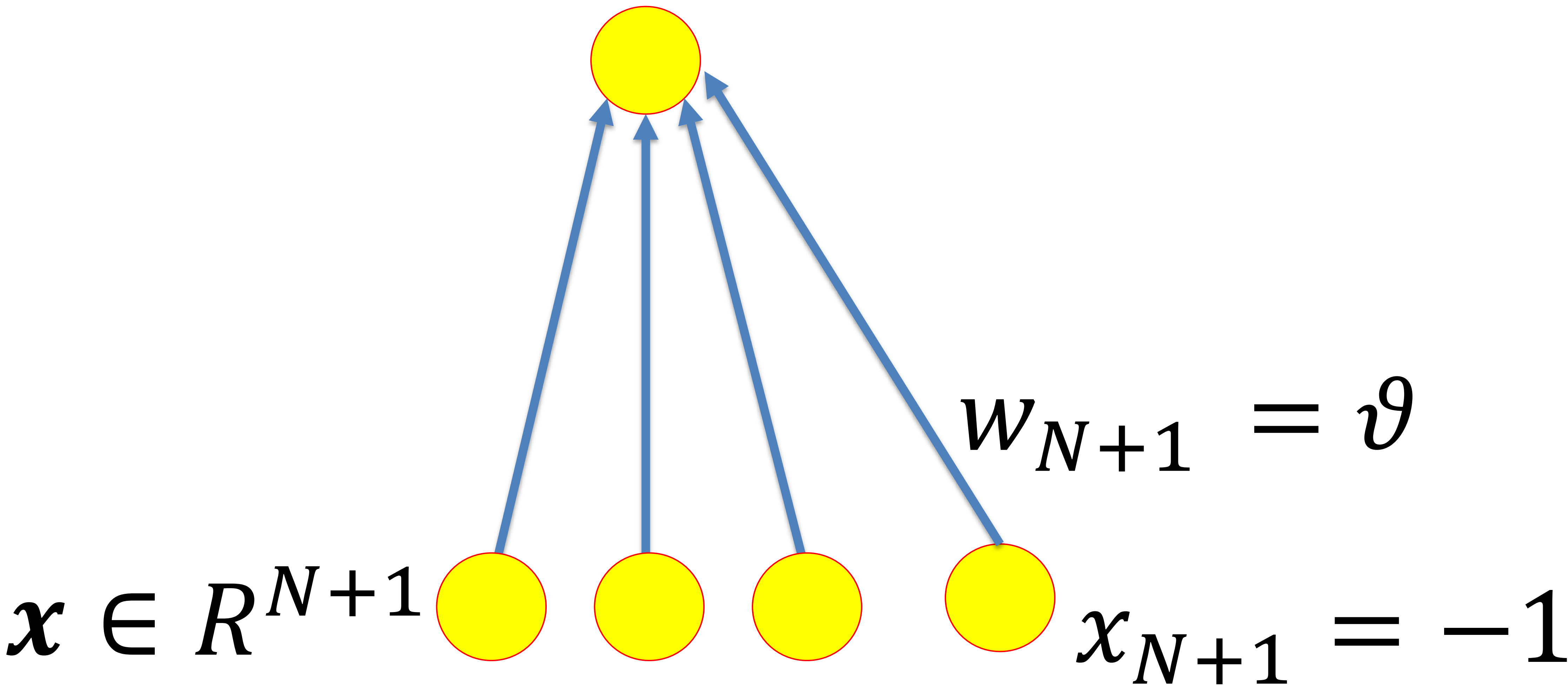


gradient descent

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



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



Gradient descent calculation: ‘batch’ and ‘online’

Batch: one update step **after all** patterns have been applied

Online/Stochastic Gradient Descent (SGD):

- one update step **after each** pattern
- one ‘epoch’ = P patterns have been applied

In both cases, we cycle several times over all patterns

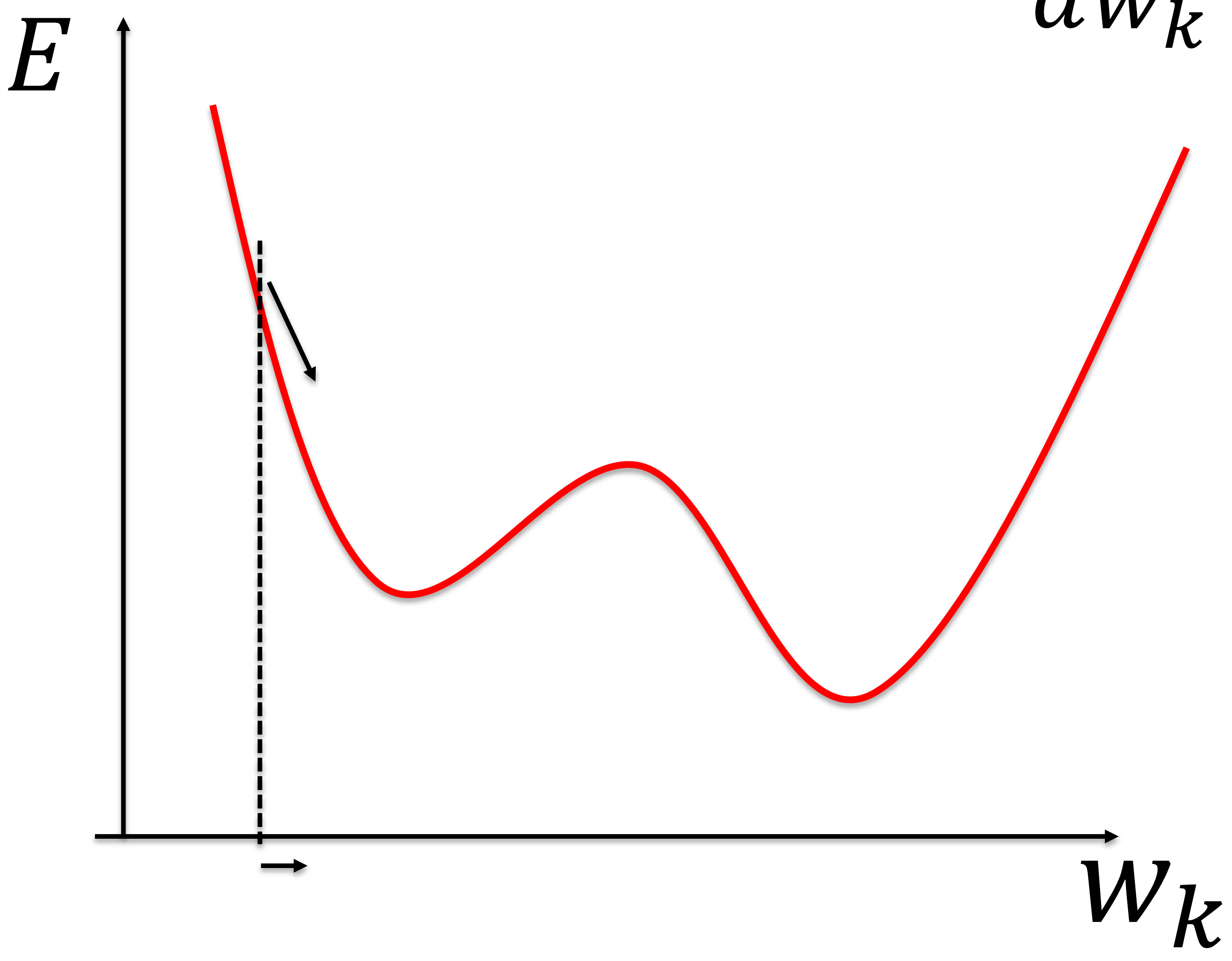
Gradient descent

Quadratic **error**

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

gradient descent

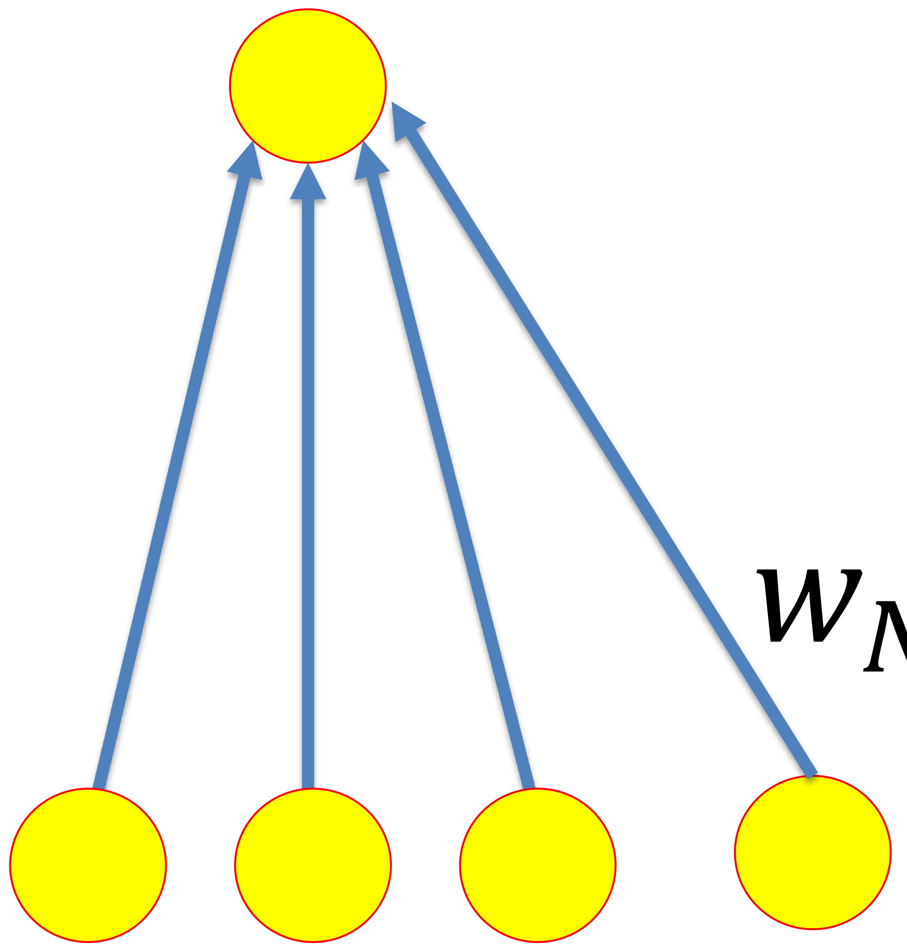
$$w_k = -\gamma \frac{dE}{dw_k}$$



Exercise 1 now:

- *calculate gradient*
- *limit to one pattern*
- *geometric interpretation?*

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



$w_{N+1} = \vartheta$

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

$x_{N+1} = -1$

Artificial Neural Networks (Gerstner). Exercises for week

Week 1: Simple Perceptrons, Geometric interpretation, Discriminant

Lecture continues at 14h15

1. Gradient of quadratic error function

We define the mean square error in a data base with P patterns as

$$E^{\mathrm{MSE}}(\mathbf{w}) = \frac{1}{2}\frac{1}{P}\sum_{\mu}[t^{\mu} - \hat{y}^{\mu}]^2 \qquad (1)$$

Exercise 1 now

- calculate gradient

- apply only 1 pattern

- geometry/vector?

where the output is

$$\hat{y}^{\mu} = g(a^{\mu}) = g(\mathbf{w}^T\mathbf{x}^{\mu}) = g(\sum_k w_k x_k^{\mu})$$

and the input is the pattern $\mathbf{x}^{\mu}$ with components $x_1^{\mu} \dots x_N^{\mu}$.

(a) Calculate the update of weight w_j by gradient descent (batch rule)

$$\Delta w_j = -\eta \frac{dE}{dw_j} \qquad (3)$$

Hint: Apply chain rule

(b) Rewrite the formula by taking one pattern at a time (stochastic gradient descent). What is the difference to the batch rule? What is the geometric interpretation? ~~Compare with the perceptron algorithm~~!

Stochastic gradient descent algorithm (for simple perceptron)

Gradient Descent: Simple Perceptron (in N+1 dimensions)

```
  - set $\boldsymbol{\gamma} = 0.01$   (learning rate; P patterns in total, index μ)
  - choose M (number of epochs)
 (1) For counter k < P M
    - randomly choose pattern $\mu$
    - calculate output          $\hat{y}^{\mu} = g(\boldsymbol{w}^{T}\boldsymbol{x}^{\mu})$
    - update by                 $\Delta\boldsymbol{w} = \gamma[t^{\mu} - \hat{y}^{\mu}]g'\boldsymbol{x}^{\mu}$
    - increase counter k← k+1
 (2a) stop if change during last P patterns was acceptably small
 (2b) else, decrease $\gamma$ , reset k to k=1 and return to (1)
```

Artificial Neural Networks

Wulfram Gerstner

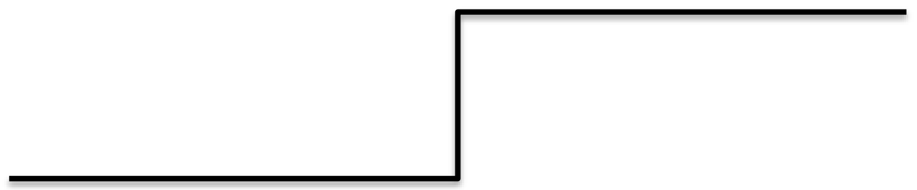
EPFL, Lausanne, Switzerland

Supervised learning, classification, simple perceptron

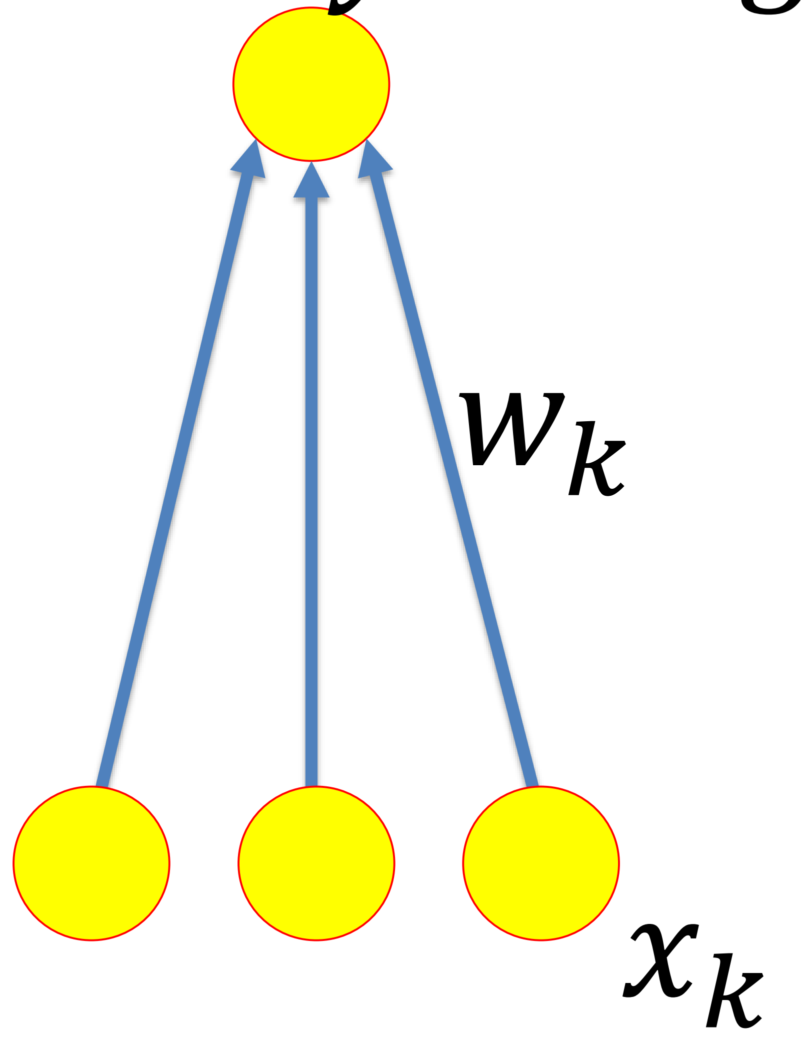
1. Classification as a geometric problem
2. Supervised learning
3. Gradient Descent for a single sigmoidal unit
4. Simple Perceptron (threshold unit)

3. Single-Layer threshold network: simple perceptron

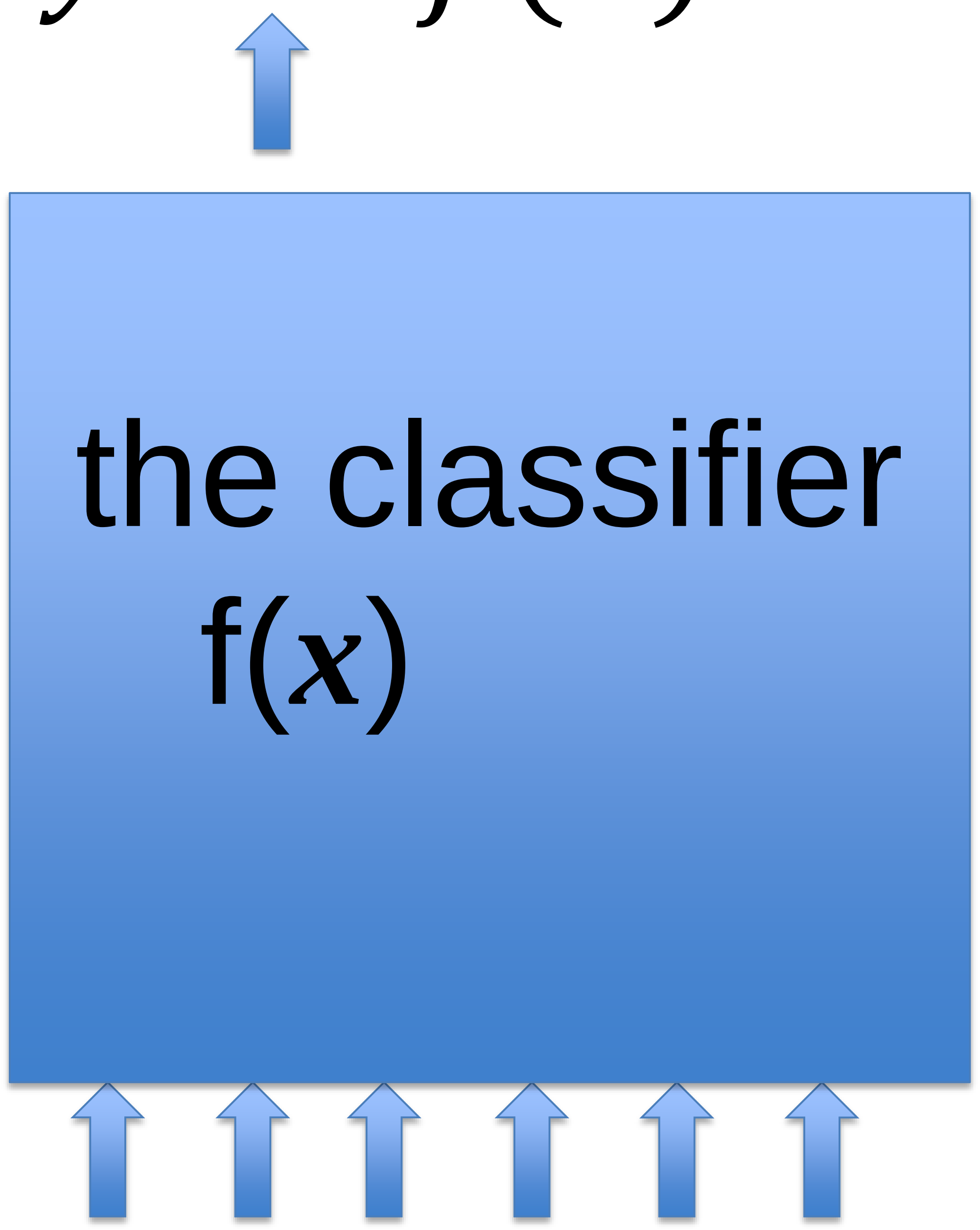
$$\hat{y} = g(a') = \begin{cases} +1 \text{ if } a' > \vartheta \\ 0 \text{ if } a' < \vartheta \end{cases}$$



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



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



the classifier f($\boldsymbol{x}$)

vector $\boldsymbol{x}$

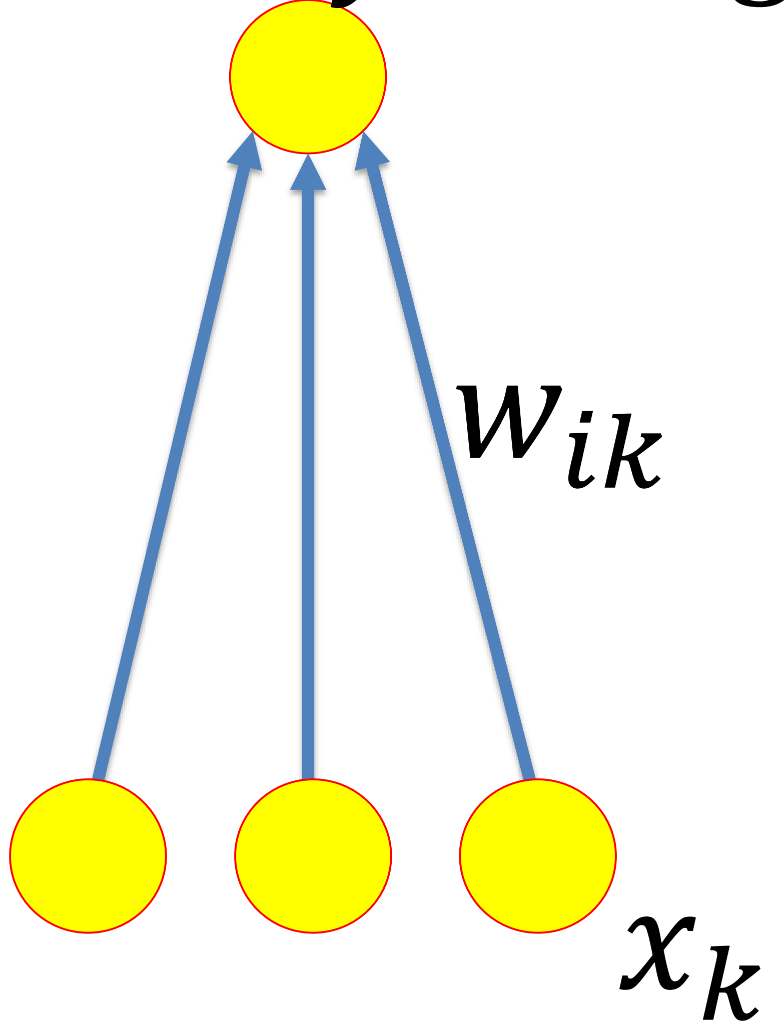
Blackboard 3: Geometry of perceptron: hyperplane

Single-Layer networks: simple perceptron

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

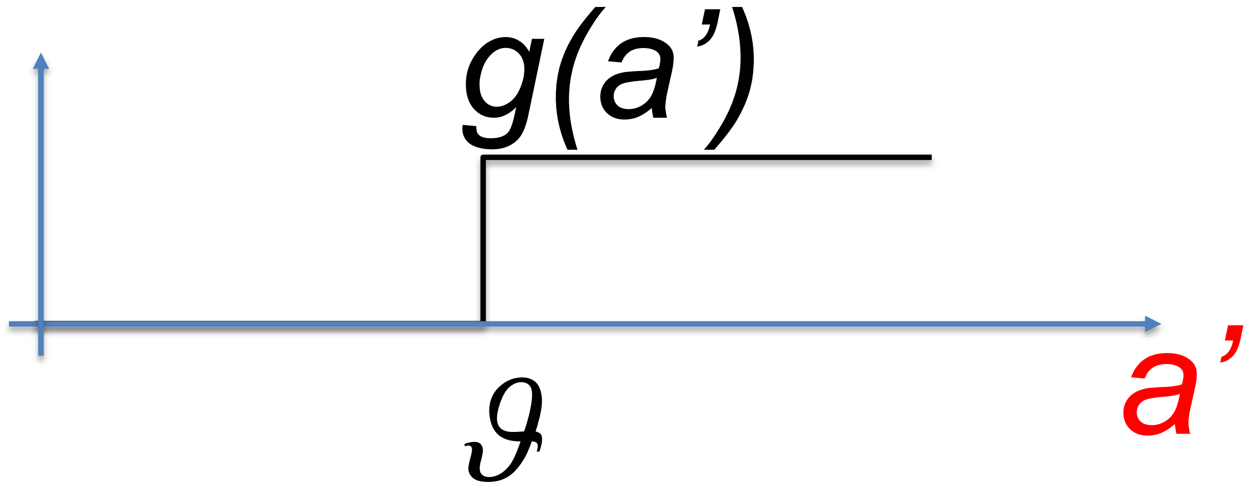
output

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



input

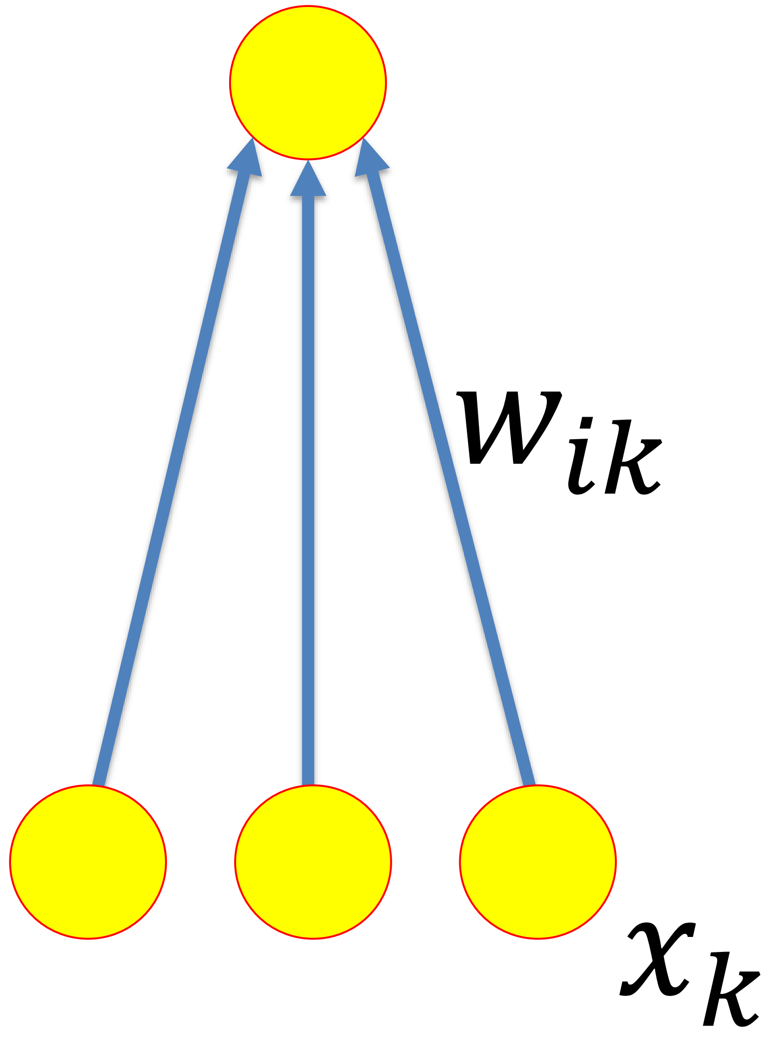
vector $\boldsymbol{x}$



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

Single-Layer networks: simple perceptron

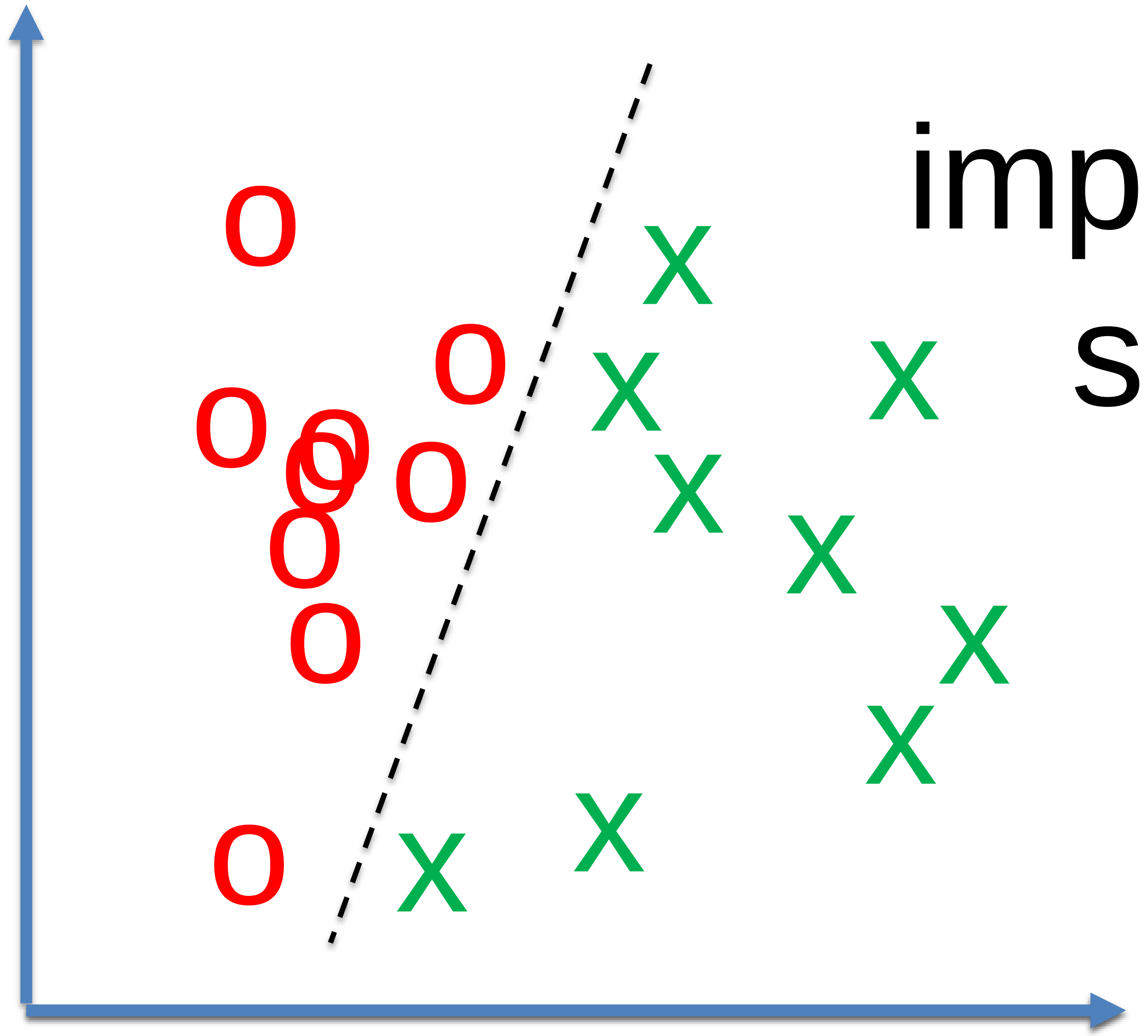
$$\hat{y} \quad = 0.5[1 + sgn(\sum_k w_k \, x_k - \vartheta)]$$



vector $\boldsymbol{x}$

Discriminant function

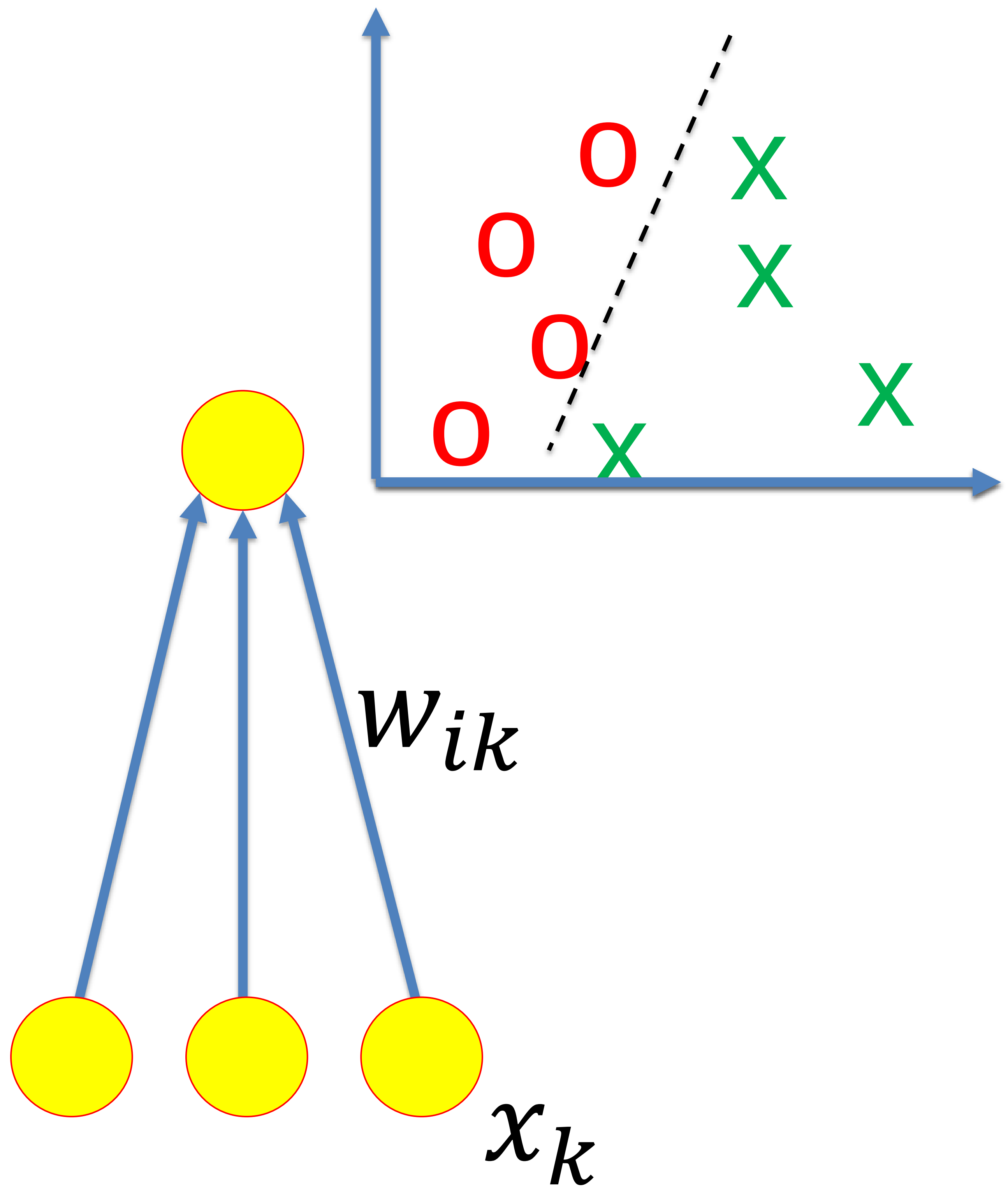
$$d(\boldsymbol{x}) = \sum_k w_k \, x_k - \vartheta = 0$$



imposes a linear separation

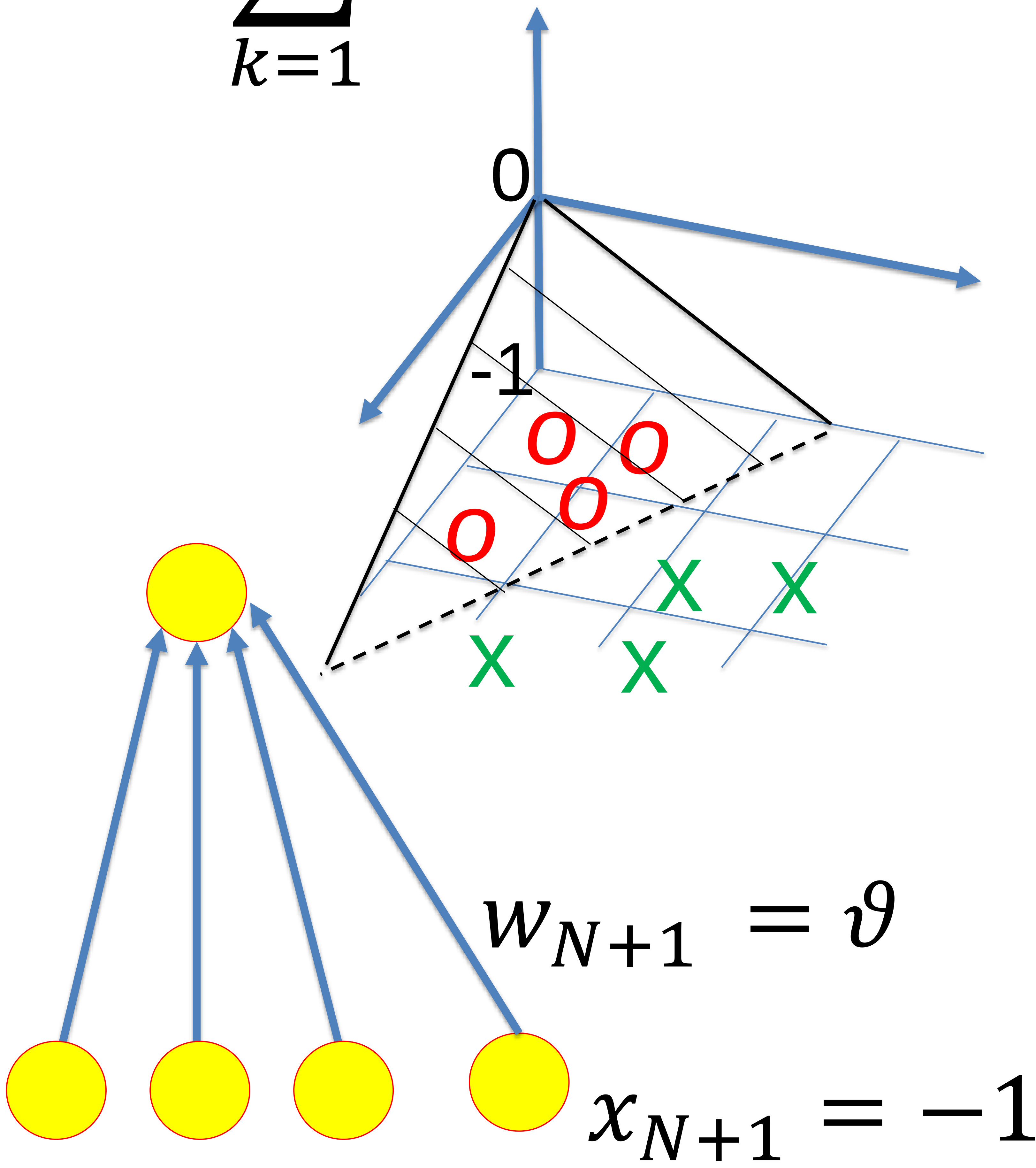
remove threshold: add a constant input

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



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

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



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

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’

Wulfram Gerstner

EPFL, Lausanne, Switzerland

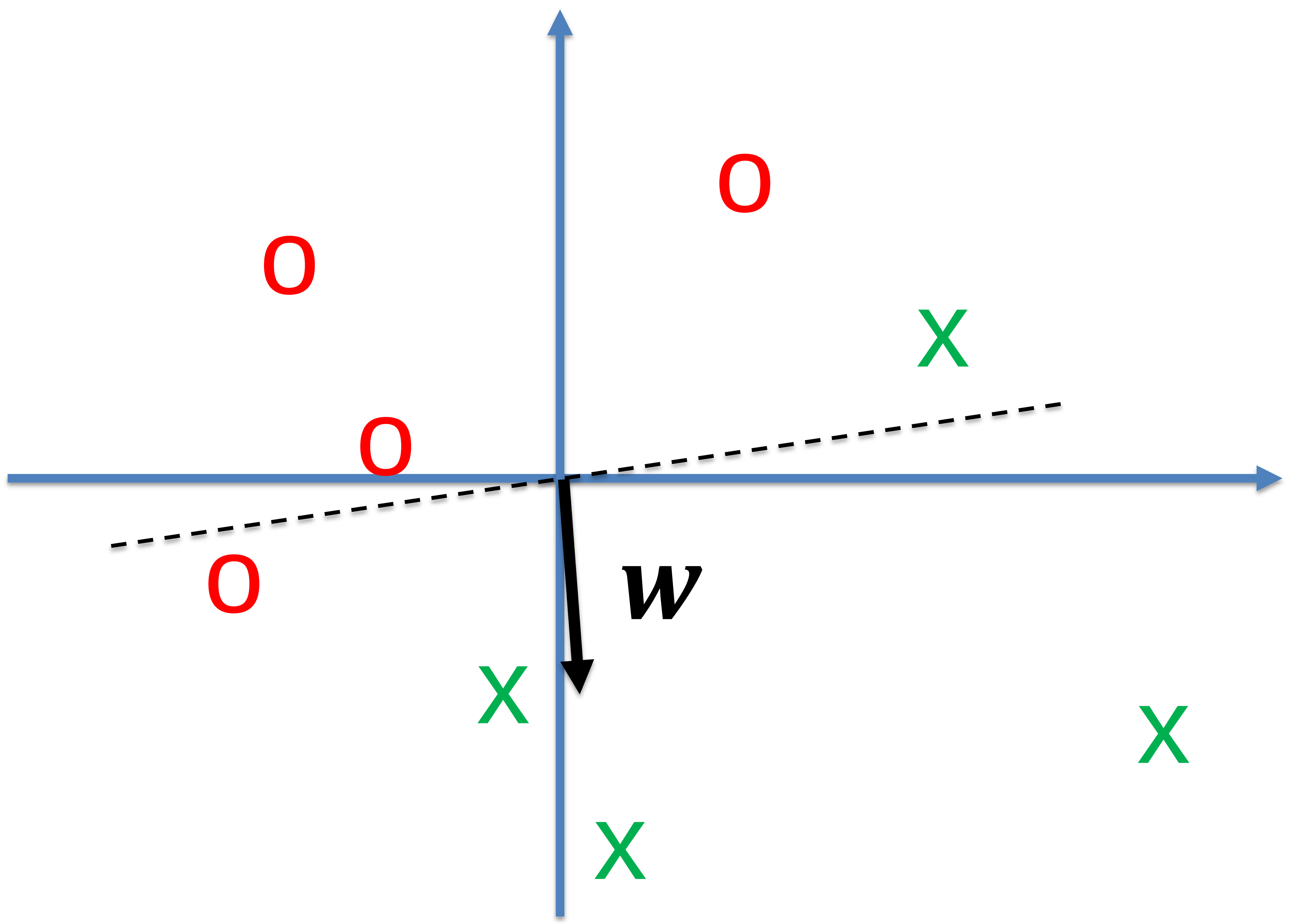
Artificial Neural Networks

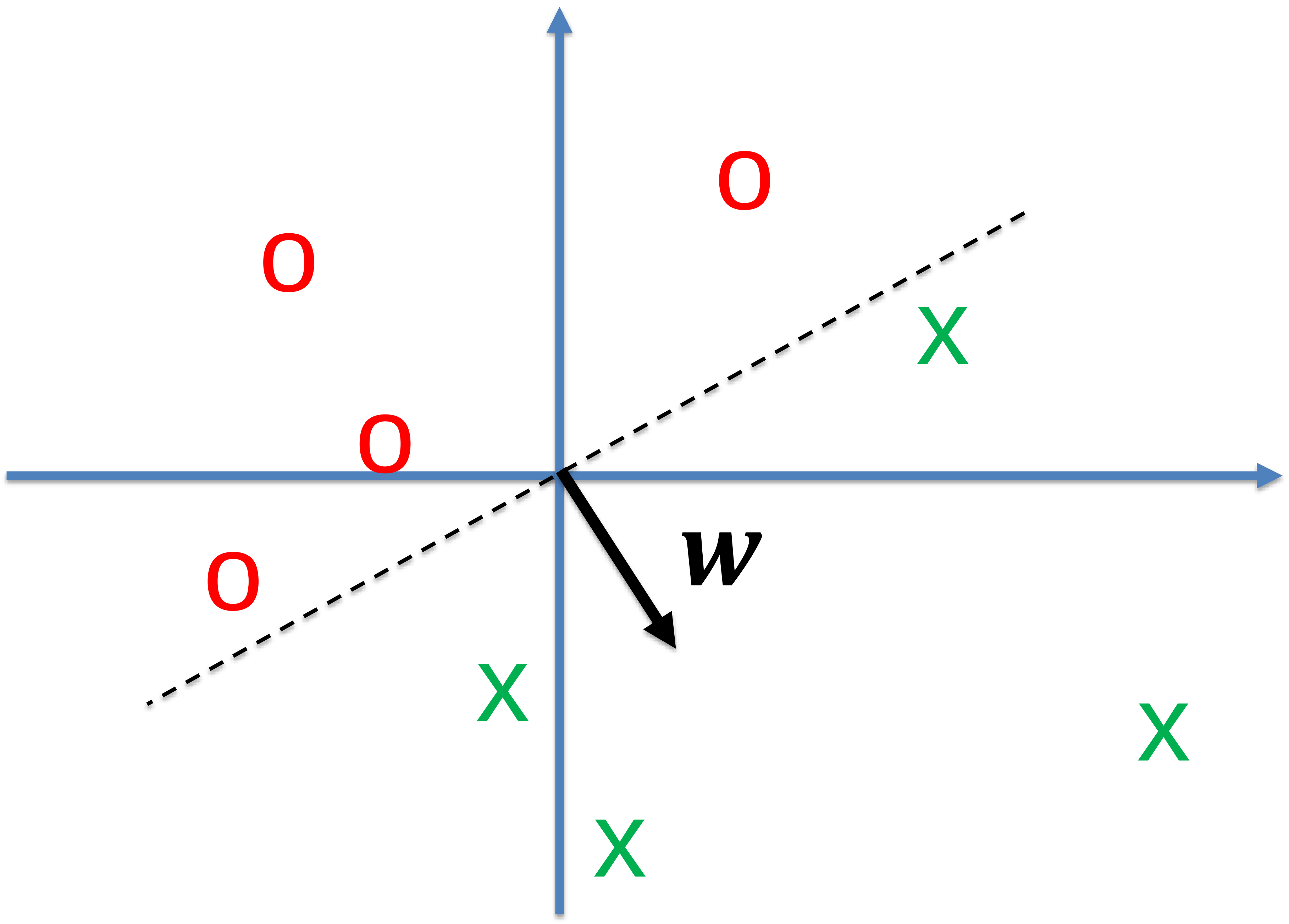
Supervised learning, classification, simple perceptron

1. Classification as a geometric problem
2. Supervised learning
3. Gradient descent: Single-layer sigmoidal unit
4. Simple Perceptron (threshold unit)
5. Perceptron Algorithm

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

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





idea: ‘turn weight vector’

Perceptron algorithm

geometry of perceptron algorithm:
turn weight vector $\Delta \boldsymbol{w} \sim \boldsymbol{x}^{\mu}$

Perceptron algo (in N+1 dimensions):

```
- set $\gamma = 0.1$
(1) cycle many times through all patterns
    - choose pattern $\mu$
    - 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 perceptron algorithm: Turn weight vector

output

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

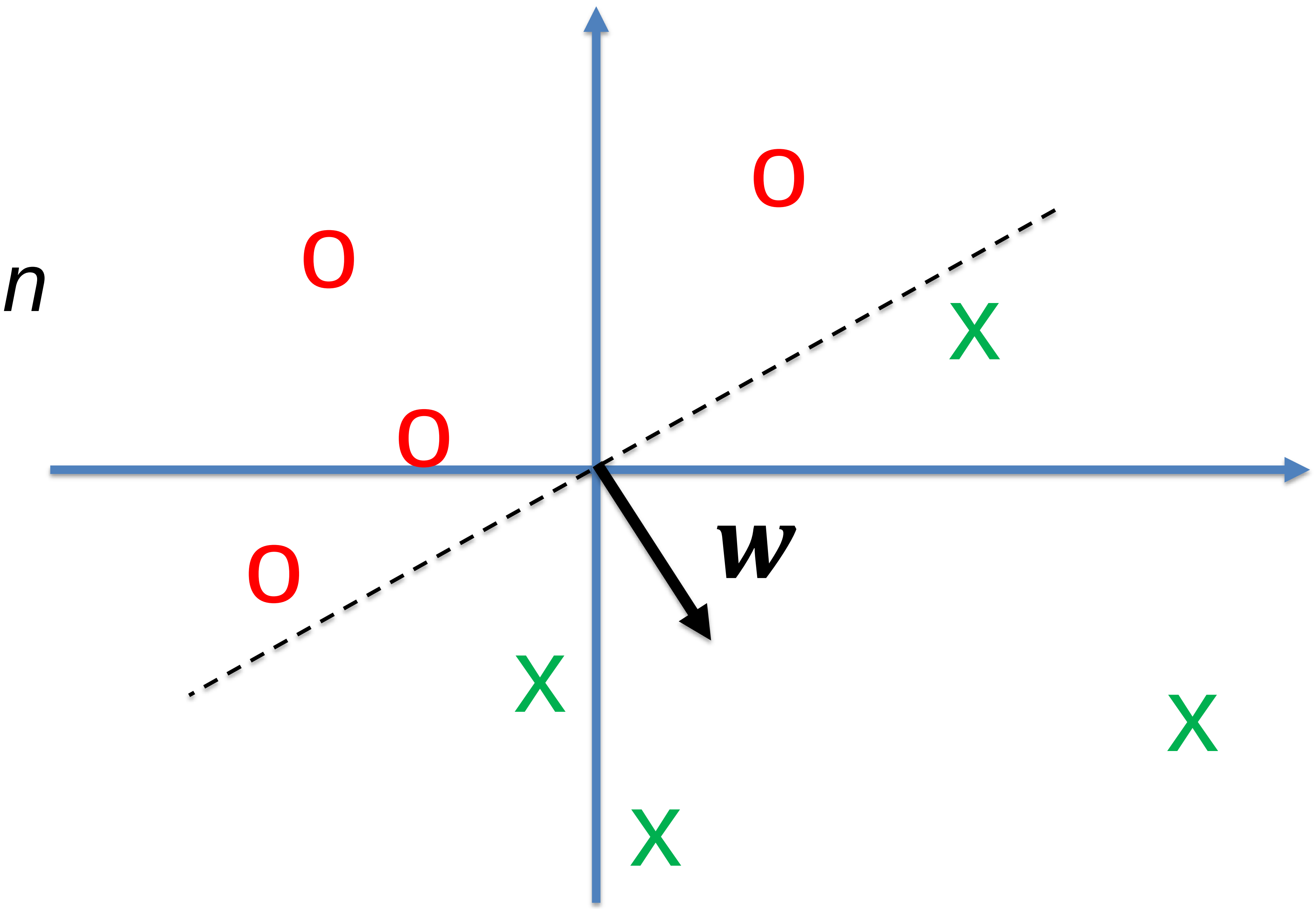
update

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

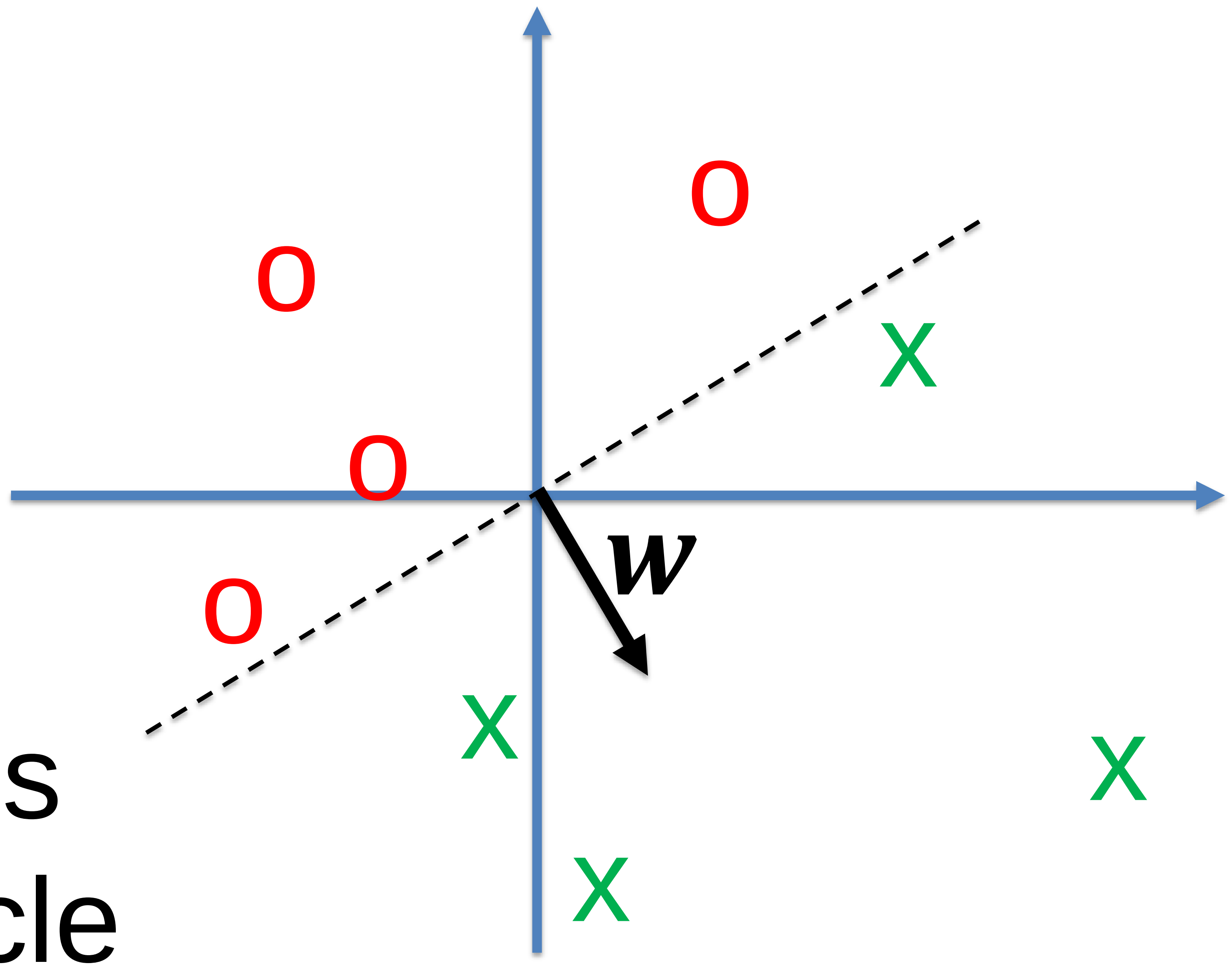
Perceptron algorithm: theoreom

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*



Summary: Perceptron algorithm



- Perceptron algorithm can solve linearly separable problems

- Cycle several times though all patterns until nothing changes during a full cycle

- Update proportional to weight vector $\Delta \boldsymbol{w} \sim \boldsymbol{x}^{\mu}$

- Proof shows: - initial value of $\boldsymbol{w}$ not important
  - learning rate γ not important

  Reason: length of $\boldsymbol{w}$ grows, but only direction matters

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 the separating hyperplane implies a rotation of the hyperplane in N+1 dimensions.

In the following change of length means $\boldsymbol{w} + \Delta\boldsymbol{w} = \beta\boldsymbol{w}$ i.e., same direction

[ ] 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

Compare Perceptron algo / online gradient descent (single layer)

Single unit (in N+1 dimensions), threshold/sigmoidal

```
  - set $\gamma$   (small learning rate; P patterns in total, index $\mu$)
  - choose M (number of epochs)
 (1) For counter k < P M
    - randomly choose pattern $\mu$
    - calculate output      $\hat{y}^{\mu} = g(\boldsymbol{w}^T \boldsymbol{x}^{\mu})$
    - update by             $\Delta \boldsymbol{w} = \gamma F[t^{\mu}, \hat{y}^{\mu}] \boldsymbol{x}^{\mu}$
```

For both algorithms (perceptron algo/stoch. gradient descent):

Mismatch in output → rotation of hyperplane (in N+1 dimensions)

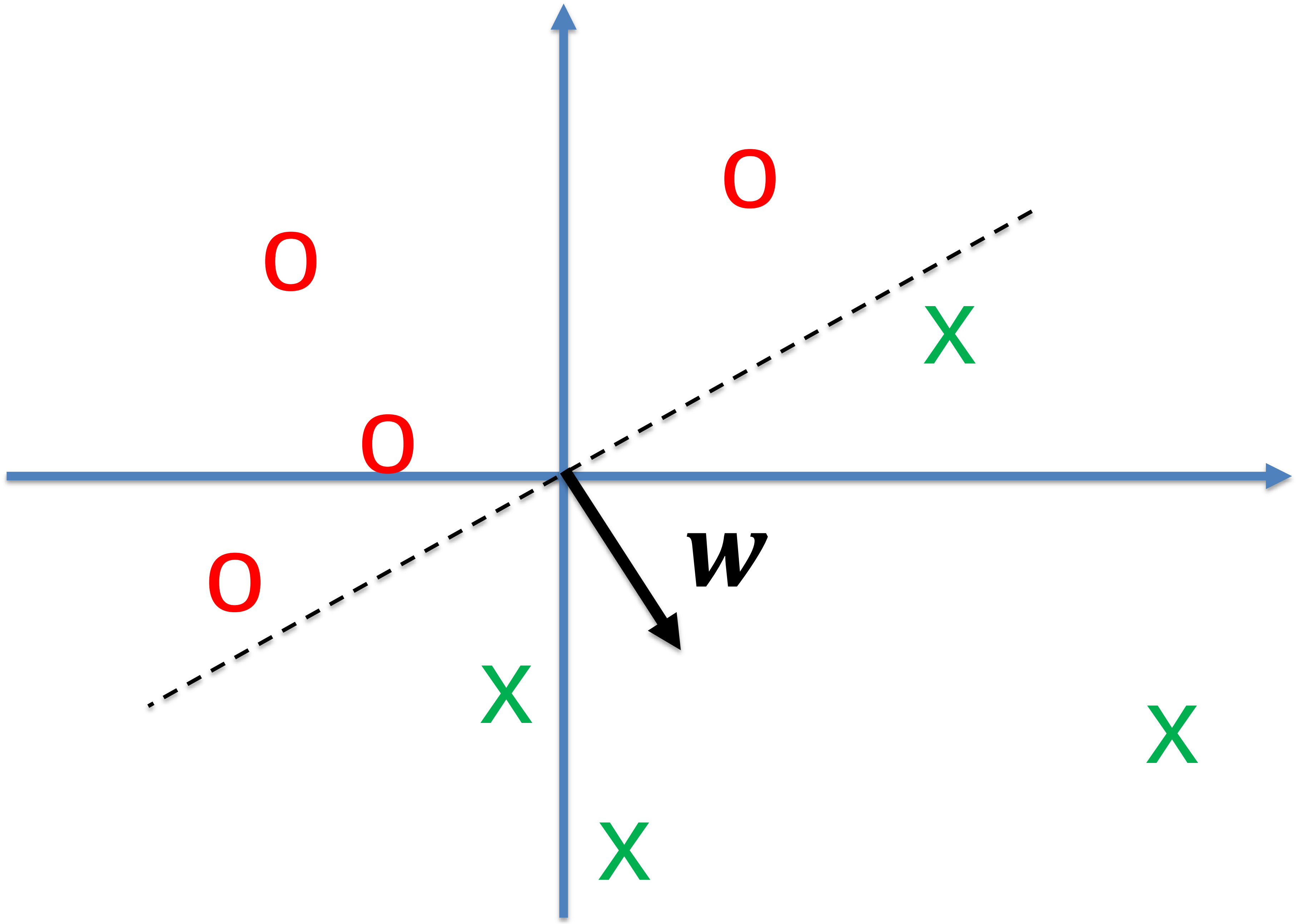
Gradient descent algorithm (stochastic gradient descent)

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

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

$$\delta(\mu) = [t^{\mu} - \hat{y}^{\mu}] g'$$

- amount of change depends on $\delta(\mu)$, prop. to the (signed) output mismatch for this data point
- change implemented even if ‘correctly’ classified
- change proportional to $\boldsymbol{x}^{\mu}$
- similar to perceptron algorithm (see next section)



Learning outcome and conclusions for today:

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

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