Moderne Methoden der Datenanalyse

Event classification and neural networks

Roger Wolf

23. June 2023

Content of this lecture

- Event classification.
- Classification features, test statistic.
- Linear discriminant analysis (LDA).
- Neuronal networks (NNs):
  - The perceptron.
  - The multilayer perceptron (MLP).
  - From Booleans to real numbers.
  - The MLP as a classifier.
  - Extension to complex boundaries.

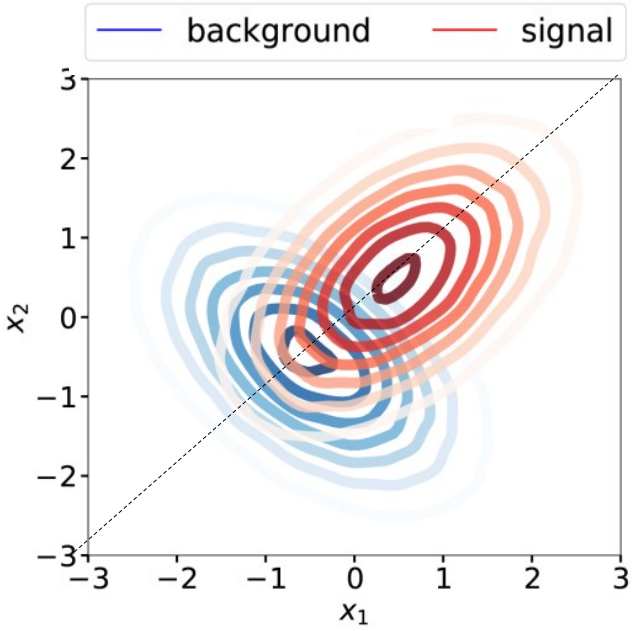
Event classification

- Association of a *sample* (*event*) with one out of a larger set of classes.
  - $N_{classes}=2$ (e.g. „signal“ and „background“) → **binary classification**.
  - $N_{classes}\geq 2$ → **multiclassification**.
  - Transition to an inifnite number of classes → **regression**.

Event classification

- Association of a *sample* (*event*) with one out of a larger set of classes.
  - $N_{classes}=2$ (e.g. „signal“ and „background“) → **binary classification**.
  - $N_{classes}\geq 2$ → **multiclassification**.
  - Transition to an inifnite number of classes → **regression**.

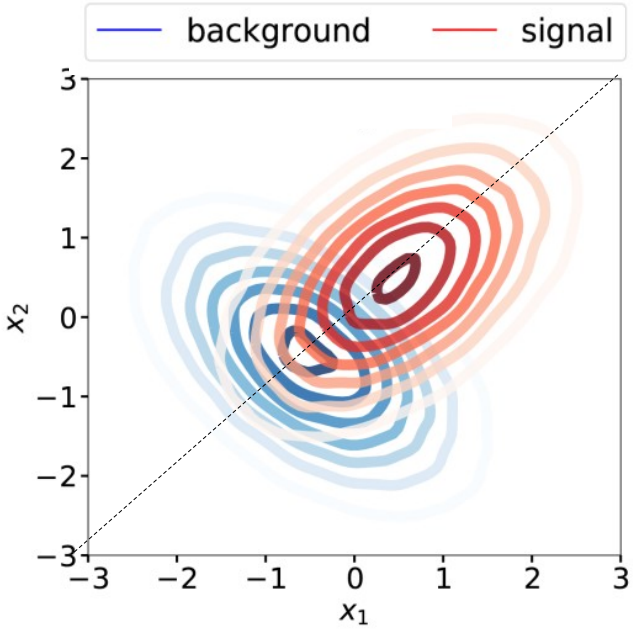
- Example (two-class hypothesis test w/ two inputs):
  - What do you think is the best discriminant in this case?



Event classification

- Association of a *sample* (*event*) with one out of a larger set of classes.
  - $N_{classes}=2$ (e.g. „signal“ and „background“) → **binary classification**.
  - $N_{classes}\geq 2$ → **multiclassification**.
  - Transition to an inifnite number of classes → **regression**.

- Example (two-class hypothesis test w/ two inputs):
  - What do you think is the best discriminant in this case? – Likelihood ratio (LR), with 2D probability densities (PDFs) for „signal“ and „background“.



Classification features

- Classification is performed on the basis of a number of *features*, which can be combined into a *feature vector* to form a *feature space*.

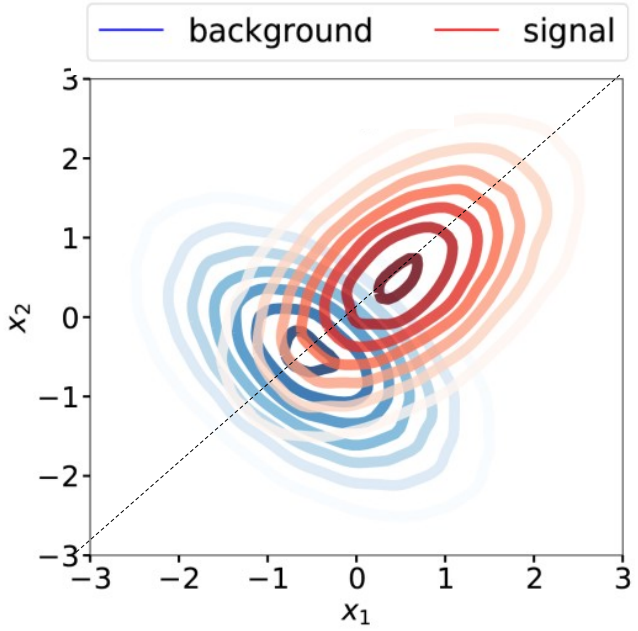
- In general classes cannot be unambiguously distinguished by a single feature, rather several features collectively contribute to the distinction of a class.

Classification features

- Classification is performed on the basis of a number of *features*, which can be combined into a *feature vector* to form a *feature space*.

- In general classes cannot be unambiguously distinguished by a single feature, rather several features collectively contribute to the distinction of a class.

- Example (two-class hypothesis test w/ two inputs):
  - Here the feature vector comprises two features x_1 and x_2.

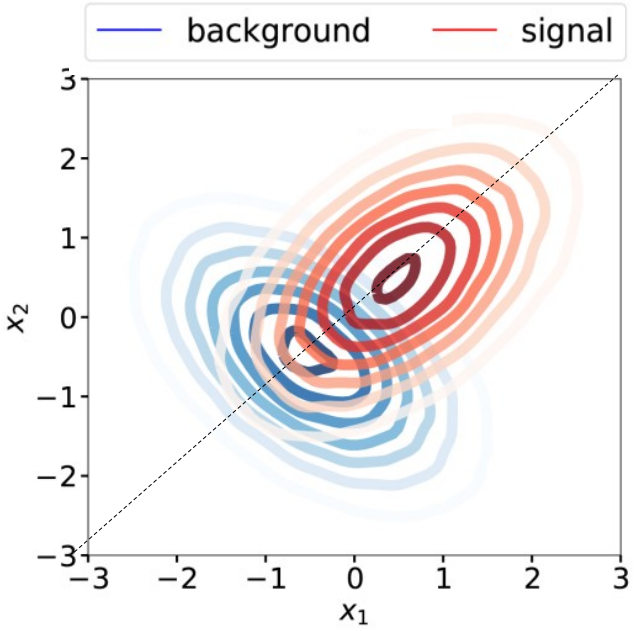


Classification features

- Classification is performed on the basis of a number of *features*, which can be combined into a *feature vector* to form a *feature space*.

- In general classes cannot be unambiguously distinguished by a single feature, rather several features collectively contribute to the distinction of a class.

- Example (two-class hypothesis test w/ two inputs):
  - Here the feature vector comprises two features x_1 and x_2.
  - In particle physics the feature vector usually comprises $\mathcal{O}(10-50)$ (complex) features.

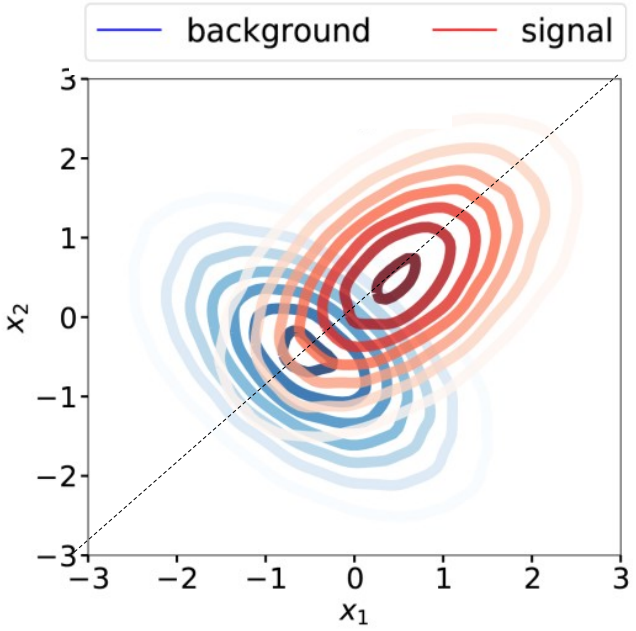


Classification features

- Classification is performed on the basis of a number of *features*, which can be combined into a *feature vector* to form a *feature space*.

- In general classes cannot be unambiguously distinguished by a single feature, rather several features collectively contribute to the distinction of a class.

- Example (two-class hypothesis test w/ two inputs):
  - Here the feature vector comprises two features x_1 and x_2.
  - In particle physics the feature vector usually comprises $\mathcal{O}(10-50)$ (complex) features.
  - In modern *machine learning* (ML) applications (e.g. image recognition) the feature vector can comprise millions of (simple) features.



Test statistic for classification

- In cases with huge feature spaces PDFs cannot be easily dealt with any more (→ curse of dimension).

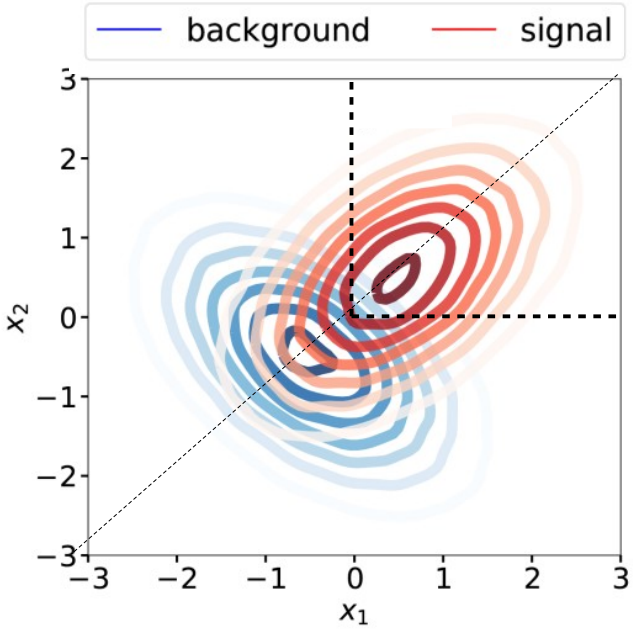
- The association with a given class, under practical aspects, is then based on a usually 1D or higher-dimensional **test statistic**.

Test statistic for classification

- In cases with huge feature spaces PDFs cannot be easily dealt with any more (→ curse of dimension).

- The association with a given class, under practical aspects, is then based on a usually 1D or higher-dimensional **test statistic**.

- Example (two-class hypothesis test w/ two inputs):
  - Simple selection criteria (a.k.a. „cuts“) on individual features (→ here 2D test statistic).

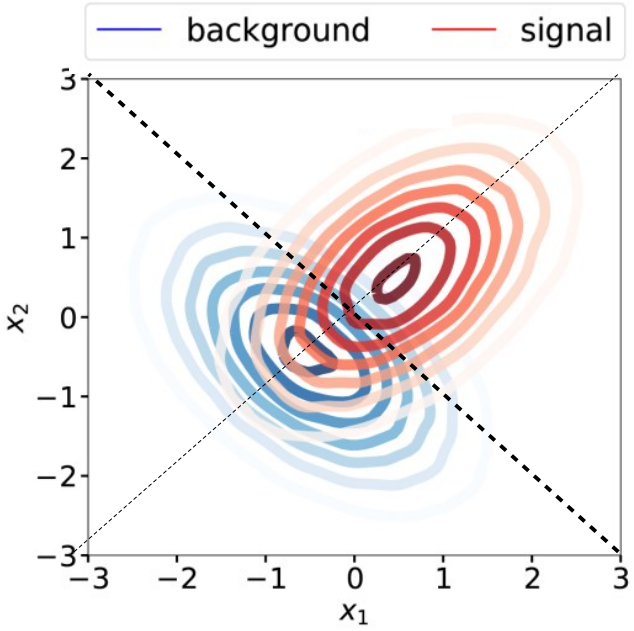


Test statistic for classification

- In cases with huge feature spaces PDFs cannot be easily dealt with any more (→ curse of dimension).

- The association with a given class, under practical aspects, is then based on a usually 1D or higher-dimensional **test statistic**.

- Example (two-class hypothesis test w/ two inputs):
  - Simple selection criteria (a.k.a. „cuts“) on individual features (→ here 2D test statistic).
  - Selection criterion on a linear combination of individual features (→ 1D linear discriminant).

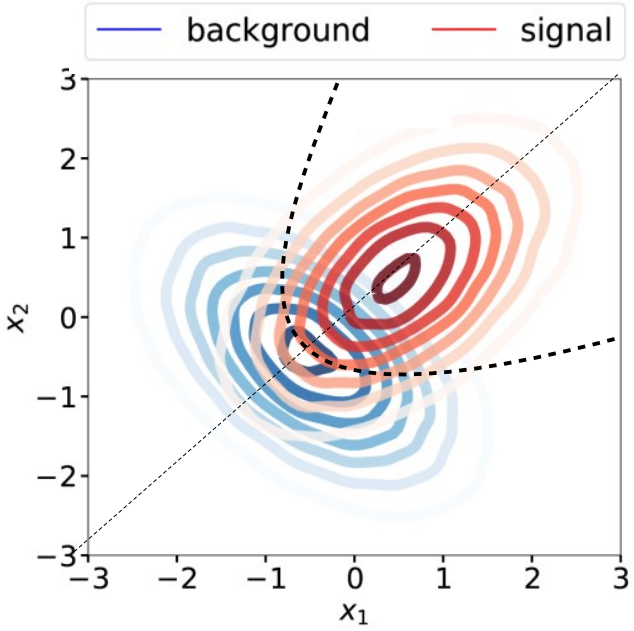


Test statistic for classification

- In cases with huge feature spaces PDFs cannot be easily dealt with any more (→ curse of dimension).

- The association with a given class, under practical aspects, is then based on a usually 1D or higher-dimensional **test statistic**.

- Example (two-class hypothesis test w/ two inputs):
  - Simple selection criteria (a.k.a. „cuts“) on individual features (→ here 2D test statistic).
  - Selection criterion on a linear combination of individual features (→ 1D linear discriminant).
  - Non-linear discriminant (→ e.g. a neuronal network (NN) with a 1D or higher-dimensional output).



Linear discriminant analysis (LDA)

- For two classes, as shown in Fig. 1, a classification by selection requirements is insufficient.
- But it must be possible to find a suited linear combination in feature space to achieve a more successful classification (see Fig. 2).

Figure 1

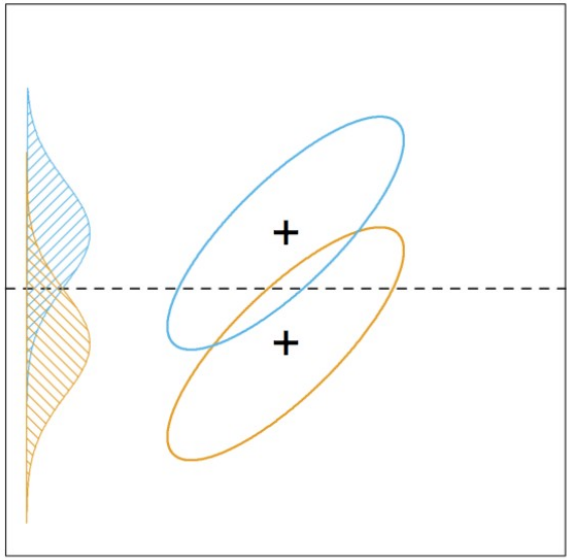
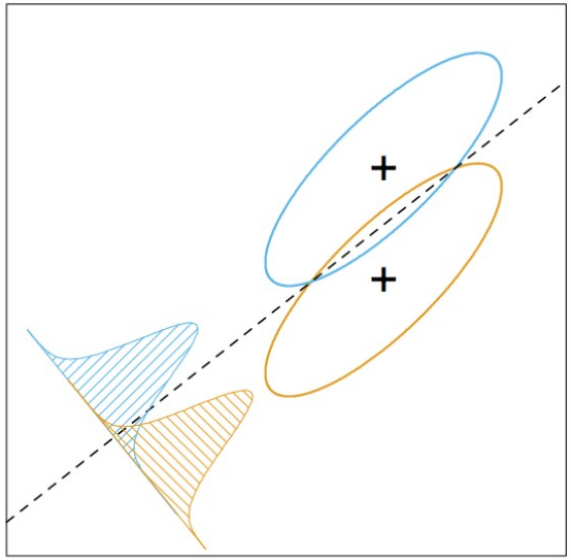


Figure 2



- **Task**: find a test statistic with best possible separation between signal (gold) and background (BG, blue).

Fisher‘s discriminant

- This can be achieved by Fisher‘s discriminant:

$$t(\mathbf{x}) = \sum_{i=1}^{n} a_i x_i$$

- Coefficients $\{a_i\}$ are obtained from the following boundary conditions:
  - The difference of sample means $\mu_s = \overline{x}(\{x_i\}, \{a_i\}|\text{signal})$ and $\mu_b = \overline{x}(\{x_i\}, \{a_i\}|\text{BG})$ should be maximal.
  - The sample variances s_s^2 und s_b^2 should be minimal.
  - Maximize the ratio:

$$J(\mathbf{a}) = \frac{(\mu_s - \mu_b)^2}{s_s^2 + s_b^2}$$

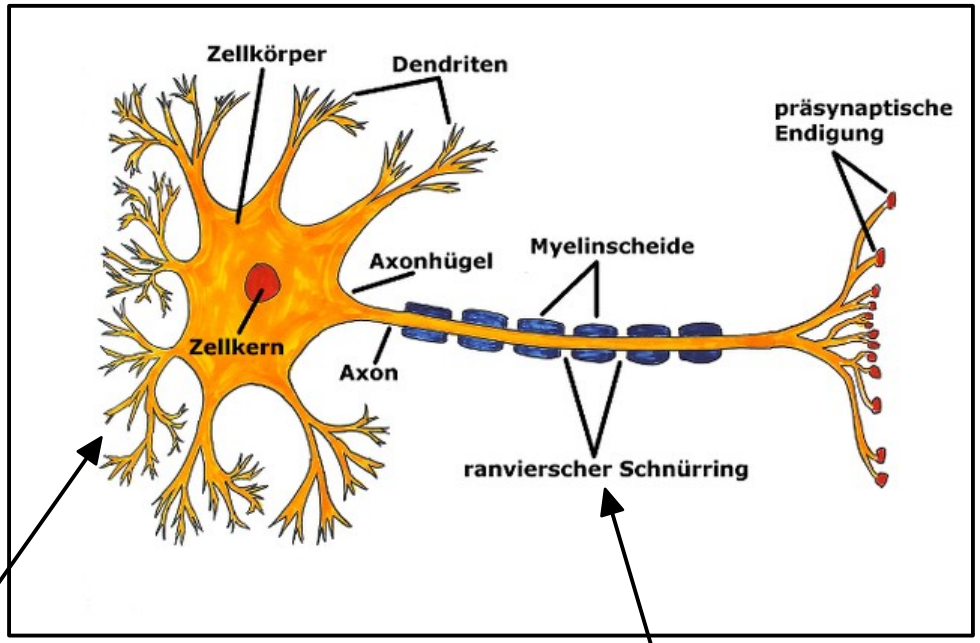
    i.e. determine the coefficients $\{a_i\}$ such that $\nabla_{\mathbf{a}} J(\mathbf{a}) = 0$.

- It can be shown that for linear separable tasks Fisher‘s discriminant is equivalent to the **likelihood ratio** and thus optimal in separating signal from BG.

Event classification with the help of neural networks

- Historically (artificial) neural networks (NNs) originate from the neuro-biological theory of (human) learning:

Schematic view of a nerve cell

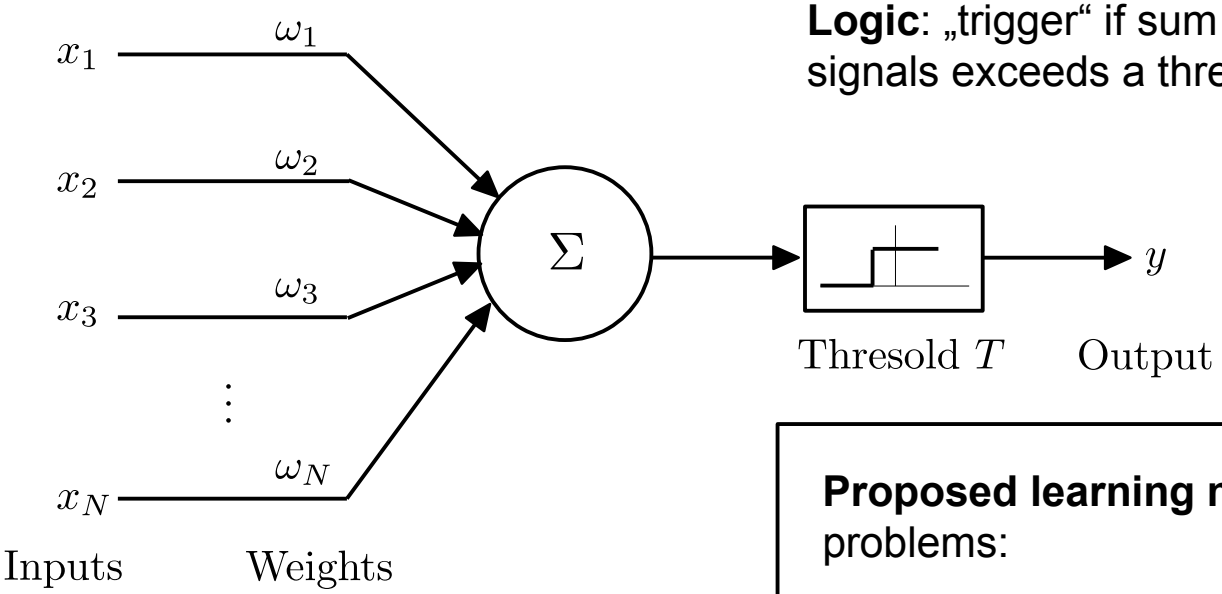


Many occasionally small signals enter here.

If the sum of incoming signals exceeds a threshold the cell „triggers“ an own signal along an axon.

The Perceptron

- After several trials, a corresponding mathematical model has been introduced by Frank Rosenblatt (11.07.1928 – 11.07.1971):



Logic: „trigger“ if sum of incoming signals exceeds a threshold.

$$y = \begin{cases} 1 & \text{if } \sum_i^N \omega_i x_i - T > 0 \\ 0 & \text{else} \end{cases}$$

Proposed learning model for Boolean problems:

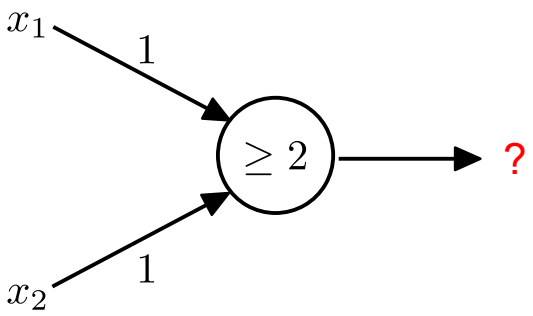
$$\boldsymbol{\omega} \to \boldsymbol{\omega} + \eta \left(d(\mathbf{x}) - y(\mathbf{x}) \right) \mathbf{x}$$

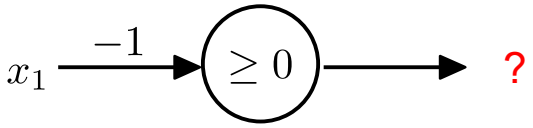
$d(\mathbf{x})$: desired output for x
$y(\mathbf{x})$: actual output for x

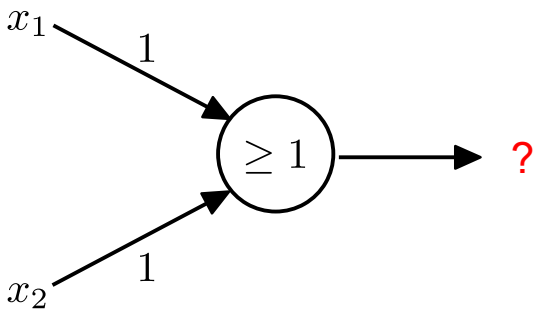
always update $\boldsymbol{\omega}$ if the percepton‘s output is wrong.

Logical operations

- Adapting the weights and thresholds the perceptron can be used to implement any logical operation:





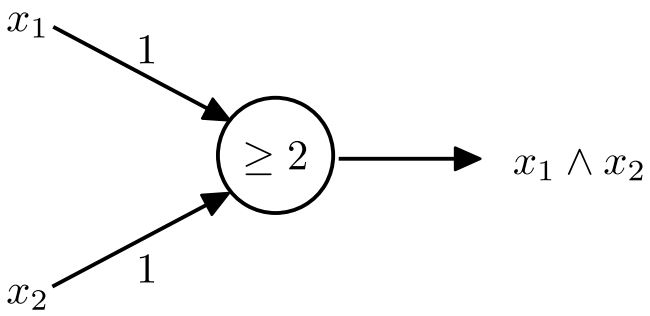


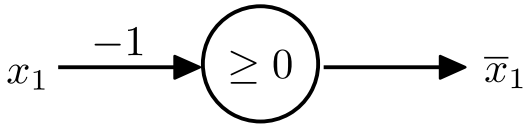
NB:

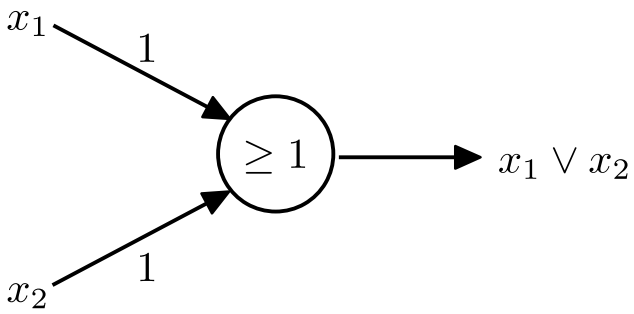
- The values on arrows represent the weights $\{\omega_i\}$;
- The values in circles represent the thresholds T;
- The features $\{x_i\}$ take the values 0 and 1.

Logical operations

- Adapting the weights and thresholds the perceptron can be used to implement any logical operation:





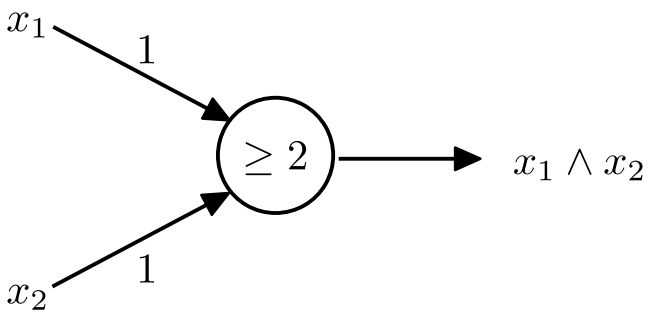


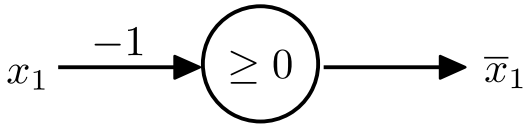
NB:

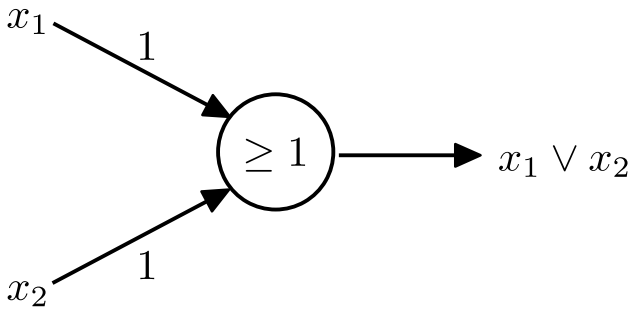
- The values on arrows represent the weights $\{\omega_i\}$;
- The values in circles represent the thresholds T;
- The features $\{x_i\}$ take the values 0 and 1.

Logical operations

- Adapting the weights and thresholds the perceptron can be used to implement any logical operation:

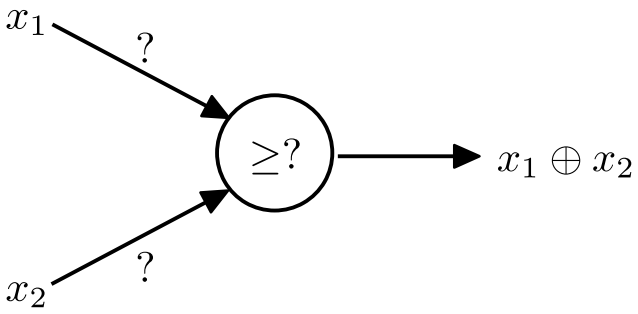






NB:

- The values on arrows represent the weights $\{\omega_i\}$;
- The values in circles represent the thresholds T;
- The features $\{x_i\}$ take the values 0 and 1.
- Any but one: the „**XOR**“

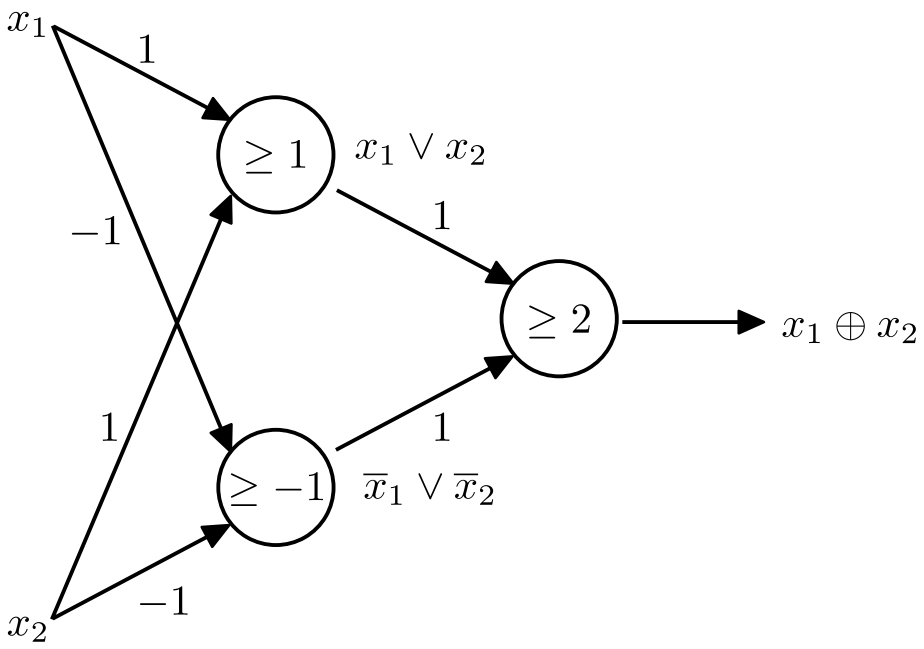


Discussed in Marvin Minsky, Seymour Papert „Perceptrons: An Introduction to Computational Geometry“, 1968 (check review here).

- i.e. a single perceptron is not a *universal computing unit*.

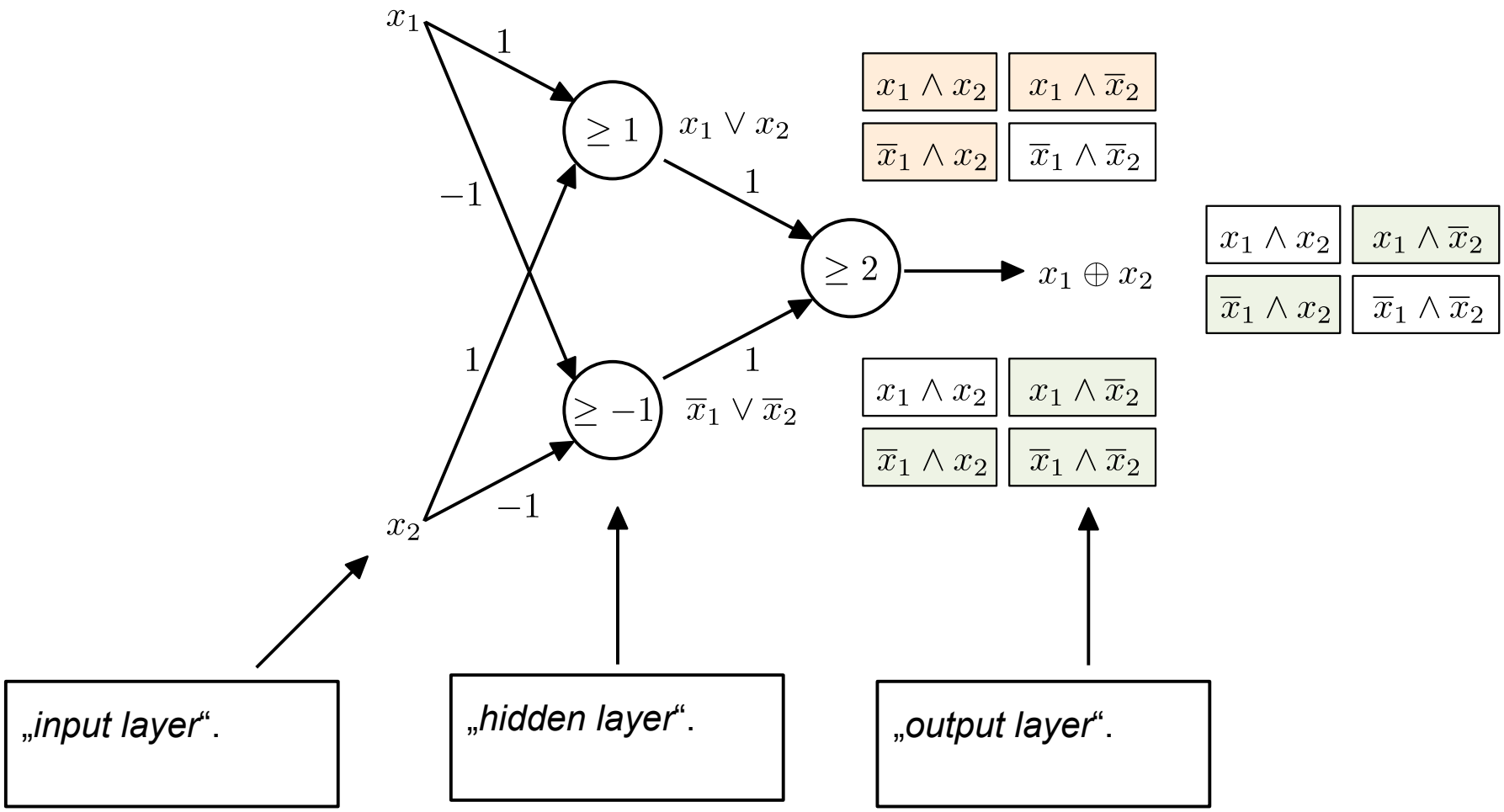
Solution to the XOR problem

- The solution to the XOR problem is to **combine several perceptrons**:



Solution to the XOR problem

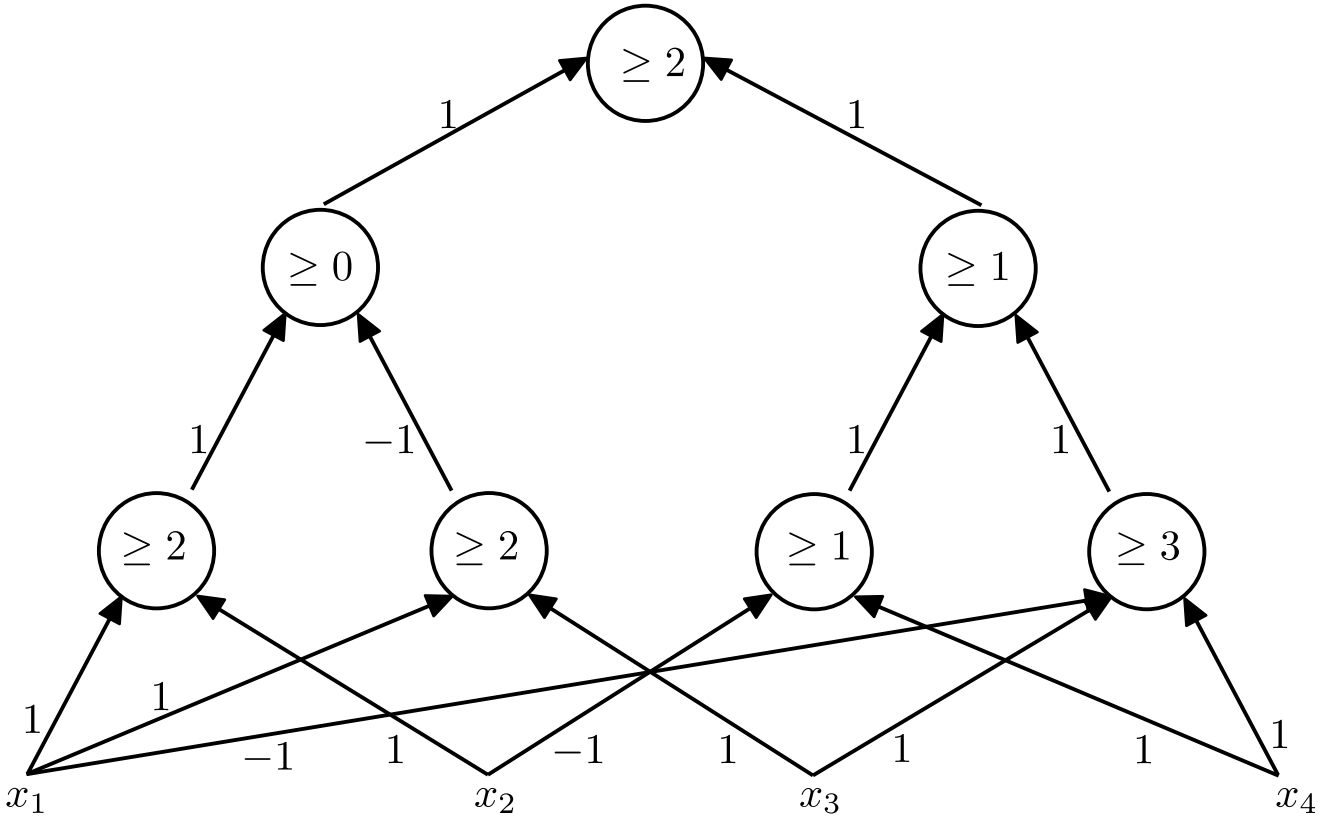
- The solution to the XOR problem is to **combine several perceptrons**:



The multilayer perceptron (MLP)

- Indeed, several layers of combined perceptrons can be used to implement any arbitrarily complex Boolean operation:

What Boolean operation has been implemented here?

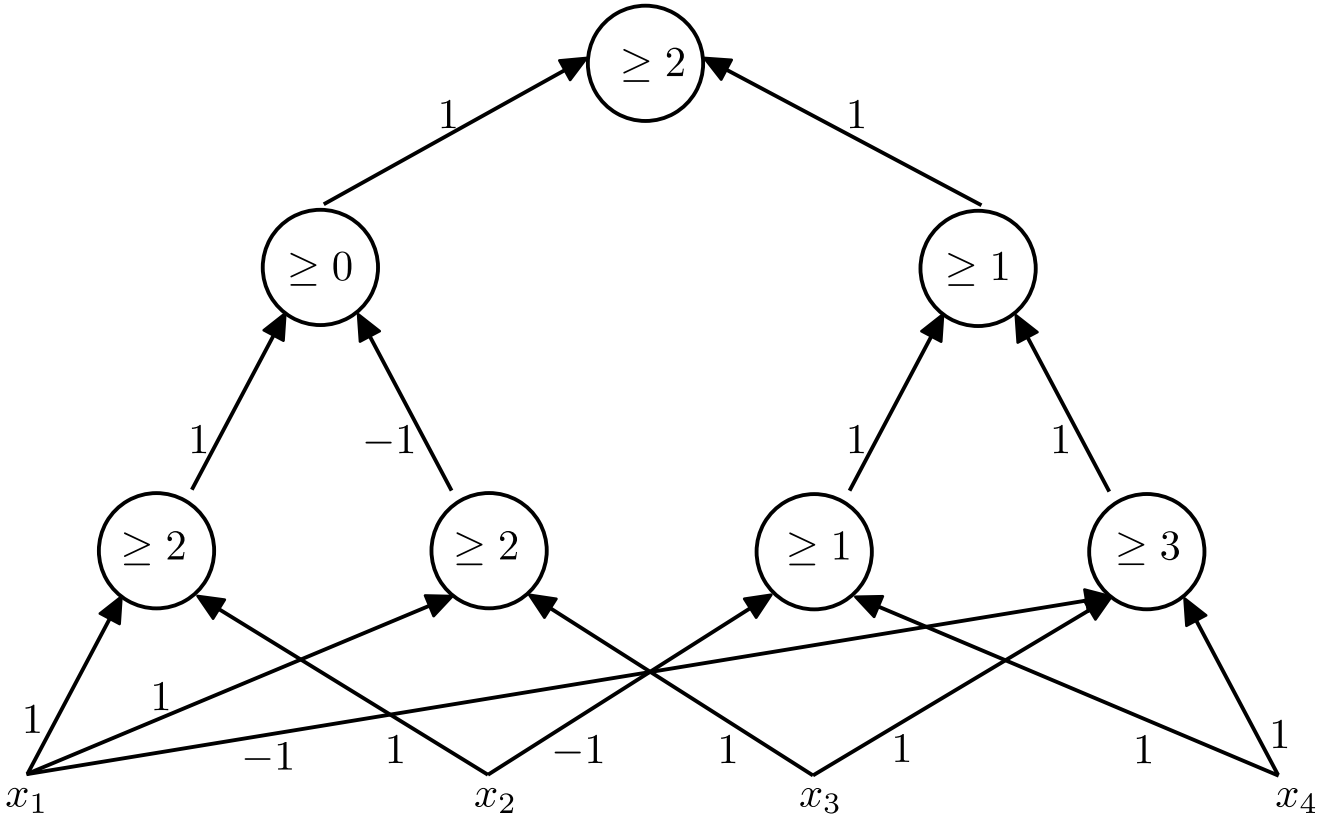


The multilayer perceptron (MLP)

- Indeed, several layers of combined perceptrons can be used to implement any arbitrarily complex Boolean operation:

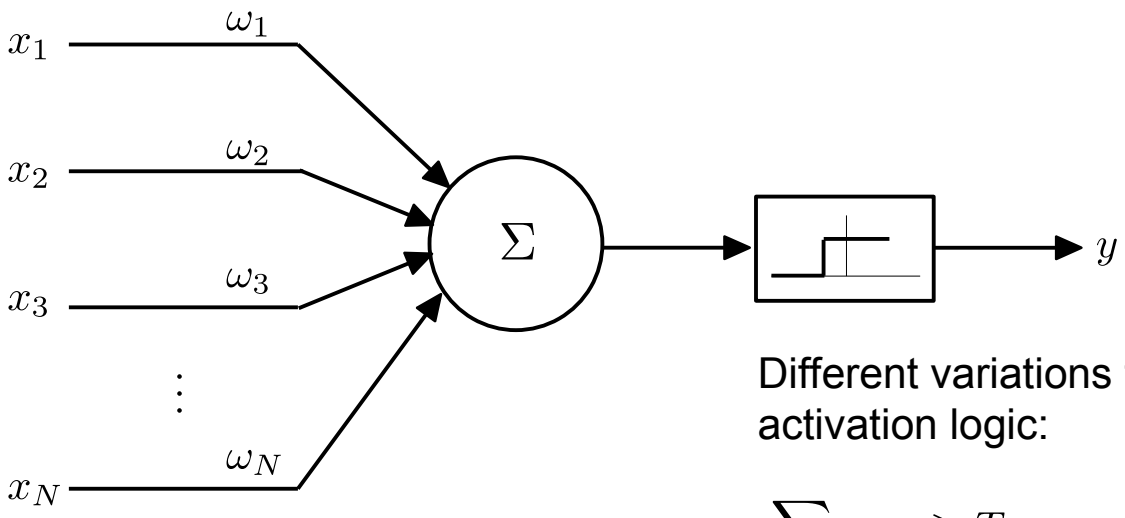
What Boolean operation has been implemented here?

$$\Big((x_1 \wedge x_2) \vee \overline{(x_1 \wedge x_3)}\Big) \wedge \Big((\overline{x}_1 \wedge x_3 \wedge x_4) \vee (\overline{x}_2 \vee x_4)\Big)$$



Boolean → real numbers

- The transition from Boolean to real numbers is indicated below:



$x_1 \dots x_N \in \mathbb{R}$

$\omega_1 \dots \omega_N \in \mathbb{R}$

Unit triggers, if $\sum_i \omega_i x_i \geq T$

Different variations to express the activation logic:

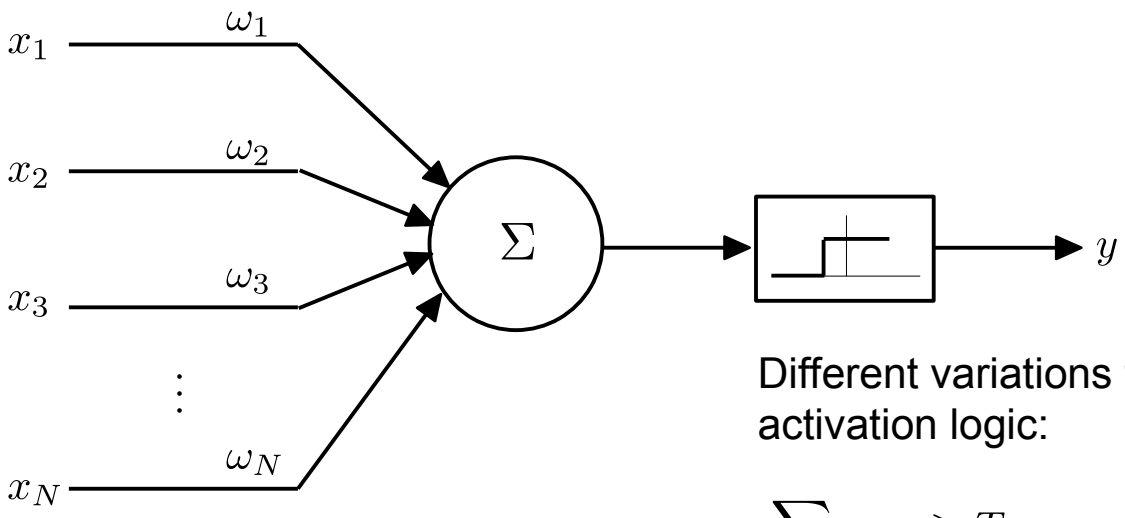
$$\sum_i \omega_i x_i \geq T\,,$$

$$\sum_i \omega_i x_i - T \geq 0\,,$$

$$\theta\left(\sum_i \omega_i x_i - T\right) \quad \text{(Heavyside function)}$$

Boolean → real numbers

- The transition from Boolean to real numbers is indicated below:



Different variations to express the activation logic:

$$\sum_i \omega_i x_i \geq T\,,$$

$$\sum_i \omega_i x_i - T \geq 0\,,$$

$$\theta\left(\sum_i \omega_i x_i - T\right) \quad \text{(Heavyside function)}$$

$x_1 \ldots x_N \in \mathbb{R}$

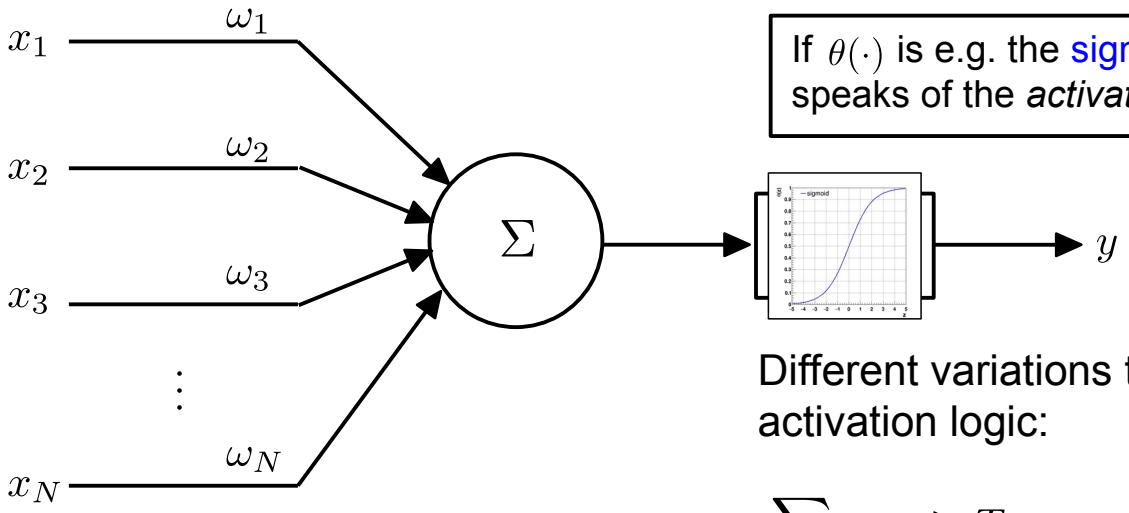
$\omega_1 \ldots \omega_N \in \mathbb{R}$

Unit triggers, if $\sum_i \omega_i x_i \geq T$

$\theta(\cdot)$ could be any real valued function

Boolean → real numbers

- The transition from Boolean to real numbers is indicated below:



If $\theta(\cdot)$ is e.g. the sigmoid function one speaks of the *activation probability*.

Different variations to express the activation logic:

$$\sum_i \omega_i x_i \geq T\,,$$

$$\sum_i \omega_i x_i - T \geq 0\,,$$

$$\theta\left(\sum_i \omega_i x_i - T\right) \quad \text{(Heavyside function)}$$

$x_1 \ldots x_N \in \mathbb{R}$

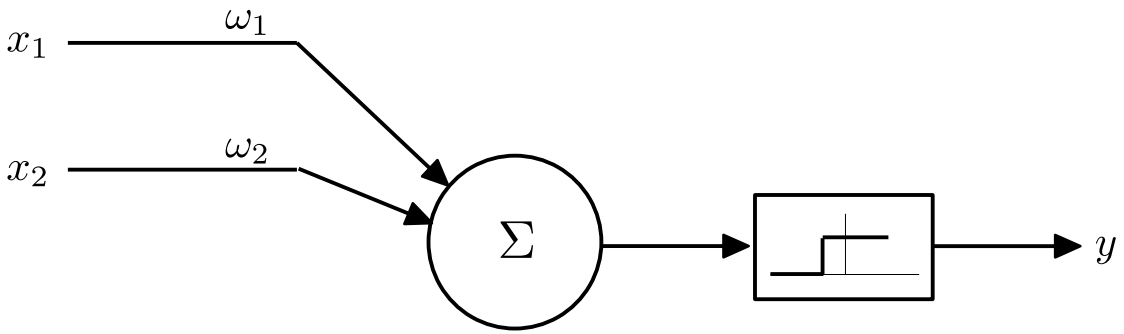
$\omega_1 \ldots \omega_N \in \mathbb{R}$

Unit triggers, if $\sum_i \omega_i x_i \geq T$

$\theta(\cdot)$ could be any real valued function

The perceptron as a classifier

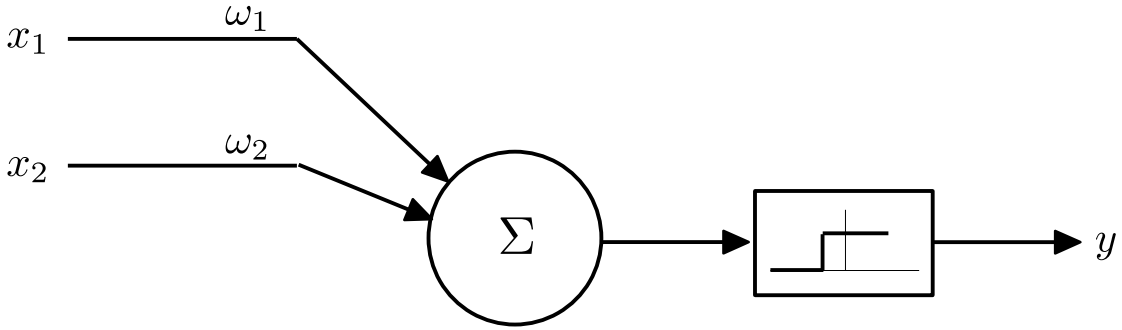
- We will, for a short time, return to the Boolean activation logic.
- Assume two real-valued inputs x_1 und x_2. The unit „triggers“ above a certain threshold T. To what boundary does this correspond to, in the space that is spanned by x_1 and x_2?

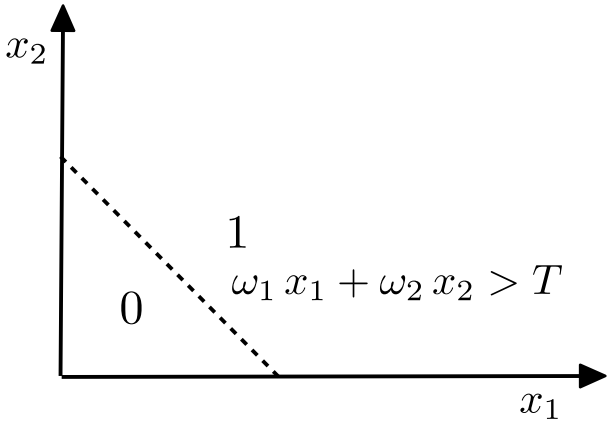


$$y = \begin{cases} 1 & \text{if } \sum_i^2 \omega_i x_i - T > 0 \\ 0 & \text{else} \end{cases}$$

The perceptron as a classifier

- We will, for a short time, return to the Boolean activation logic.
- Assume two real-valued inputs x_1 und x_2. The unit „triggers“ above a certain threshold T.
  To what boundary does this correspond to, in the space that is spanned by x_1 and x_2?



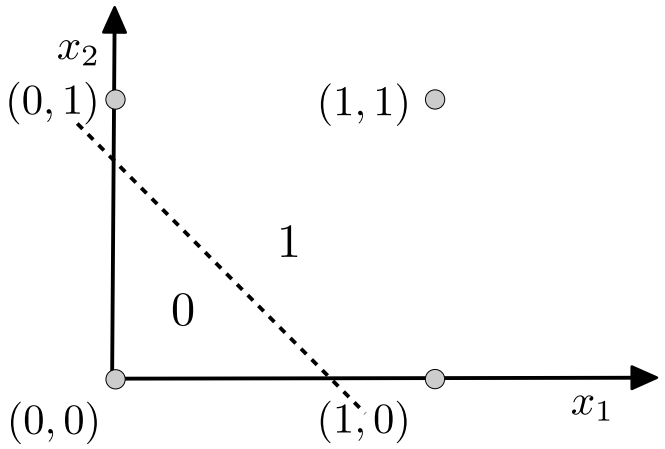


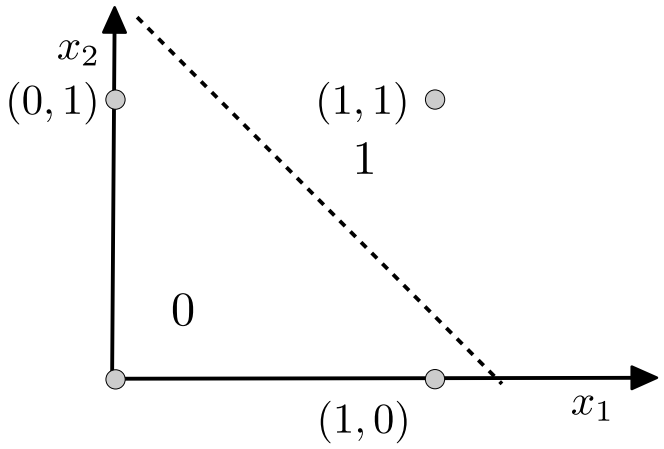
$$y = \begin{cases} 1 & \text{if } \sum_i^2 \omega_i x_i - T > 0 \\ 0 & \text{else} \end{cases}$$

Here the perceptron fullfills the role of linear classification.

Boolean logic – revisited –

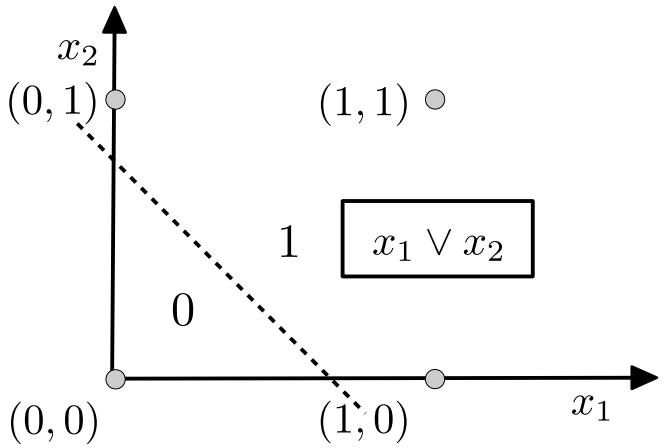
- What Boolean functions are displayed below, according to this logic?

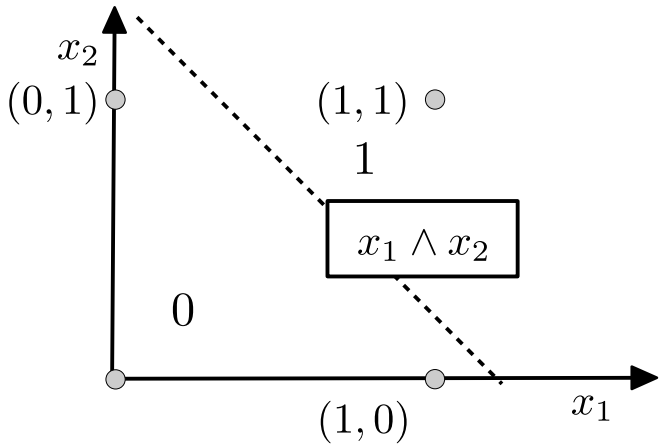




Boolean logic – revisited –

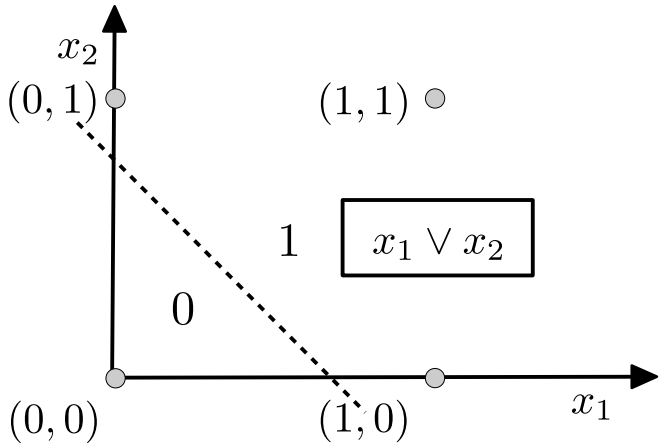
- What Boolean functions are displayed below, according to this logic?

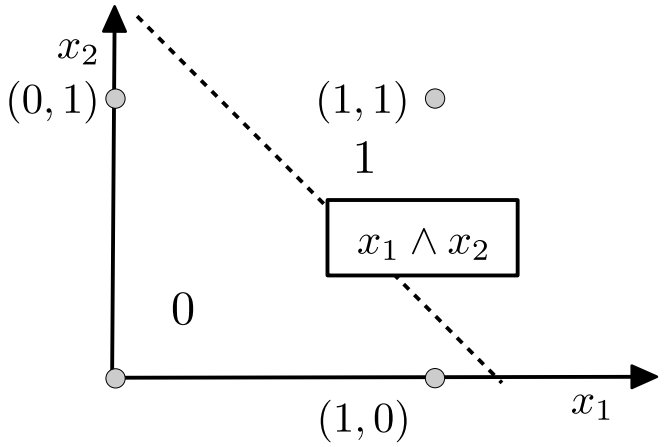




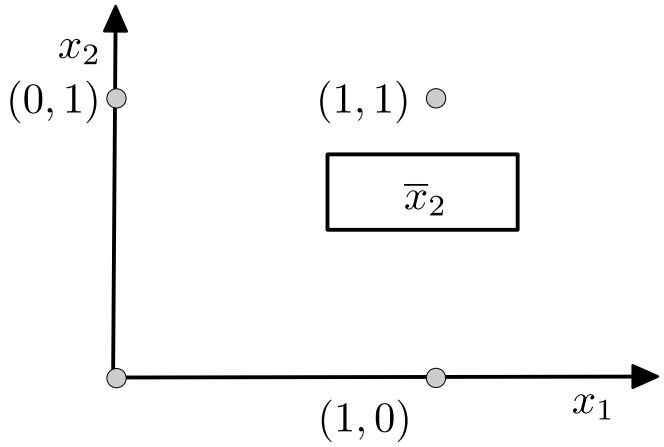
Boolean logic – revisited –

- What Boolean functions are displayed below, according to this logic?



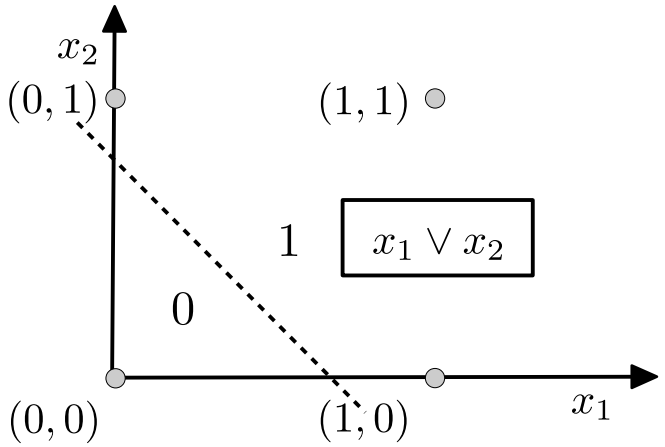


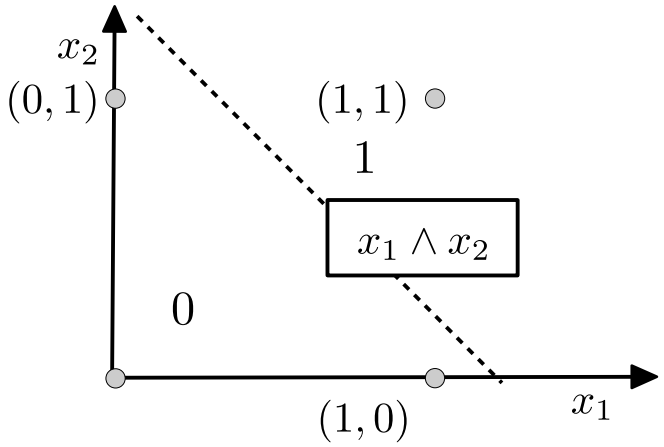
- How would you display the NOT operation ($\overline{x}_2$)?



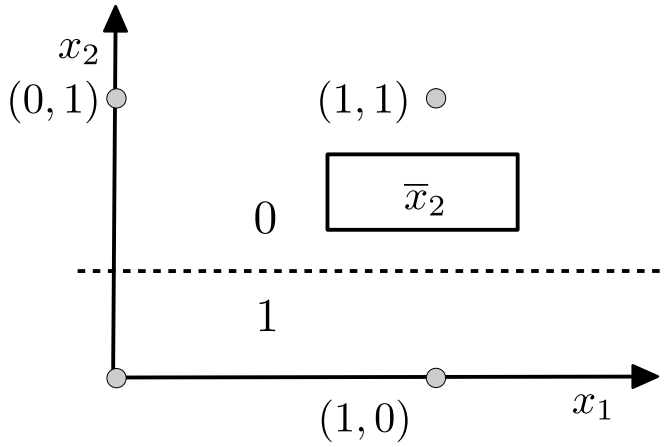
Boolean logic – revisited –

- What Boolean functions are displayed below, according to this logic?





- How would you display the NOT operation ($\overline{x}_2$)?

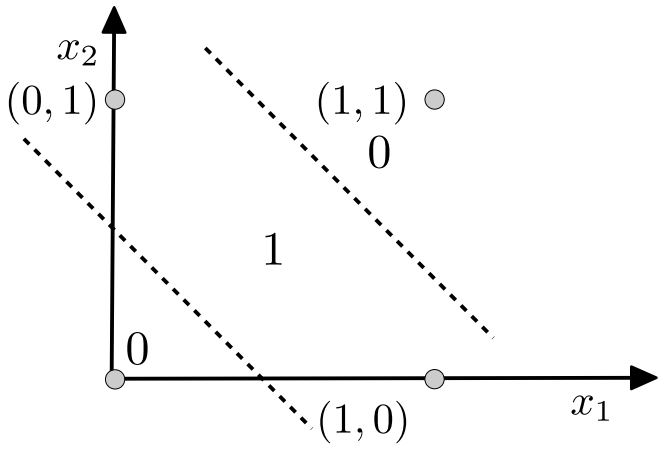


Boolean logic – revisited –

- Why can you not express an „XOR“ based on the logic of a single perceptron?

Boolean logic – revisited –

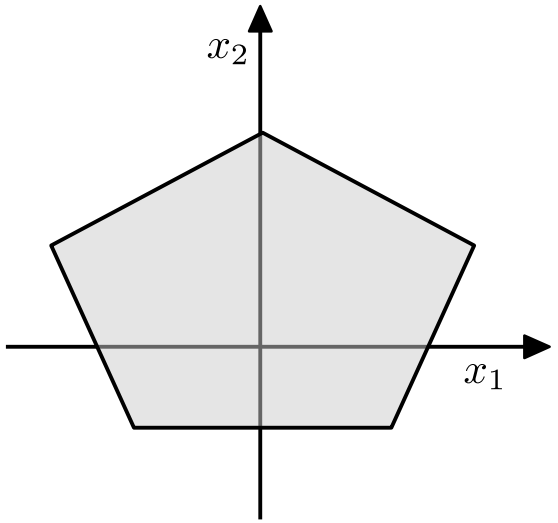
- Why can you not express an „XOR“ based on the logic of a single perceptron?



- An „XOR“ requires a representation with two lines, as shown above.
- With a single Boolean perceptron this is not possible, since it represents only single lines in the space spanned by x_1 and x_2.

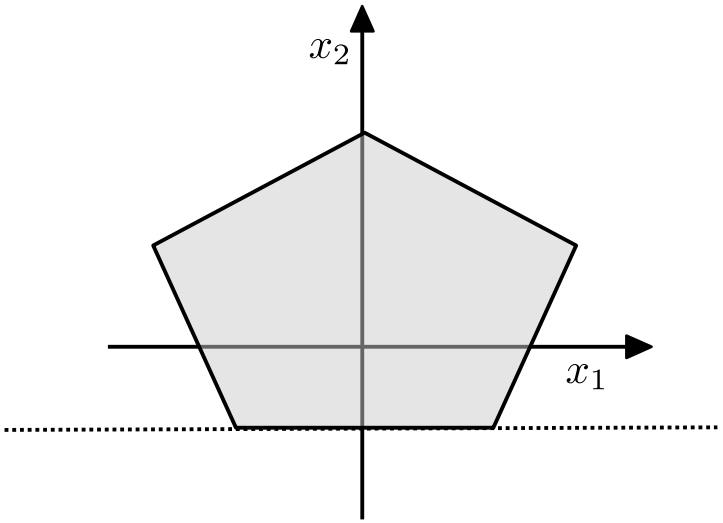
More complex boundaries

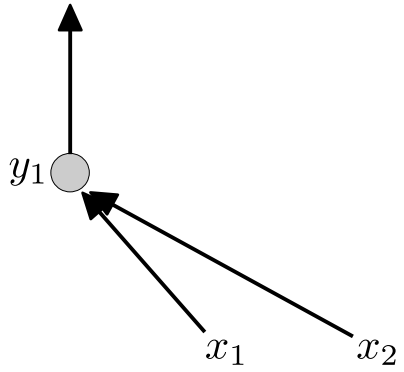
- How would you represent the figure below with the help of Boolean perceptrons?



More complex boundaries

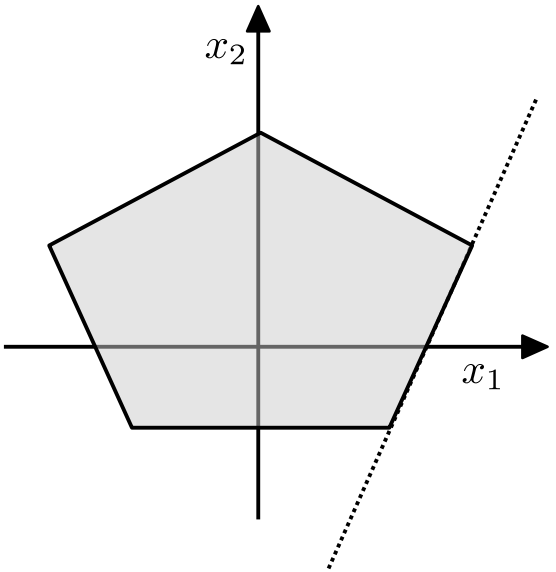
- How would you represent the figure below with the help of Boolean perceptrons?

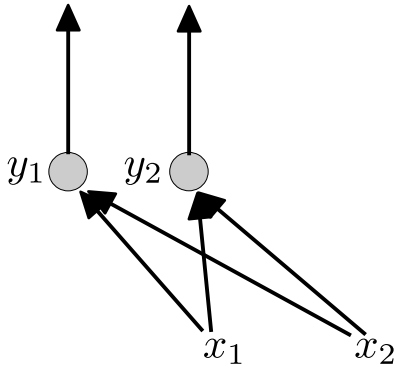




More complex boundaries

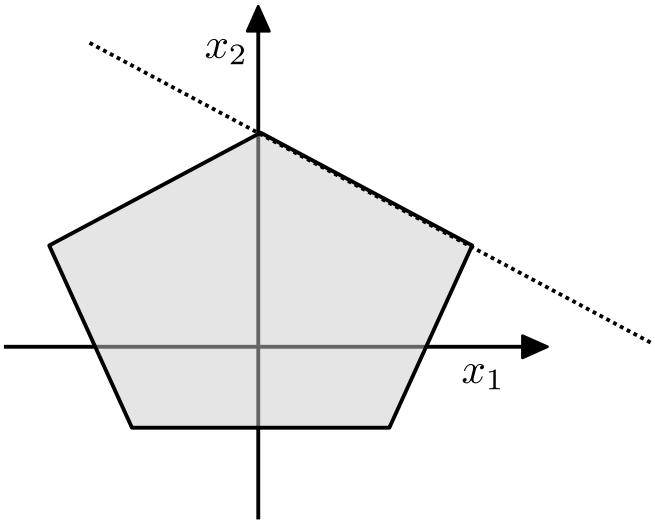
- How would you represent the figure below with the help of Boolean perceptrons?

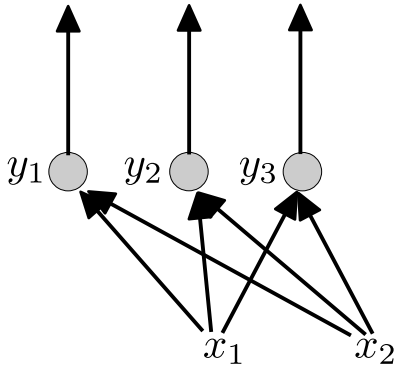




More complex boundaries

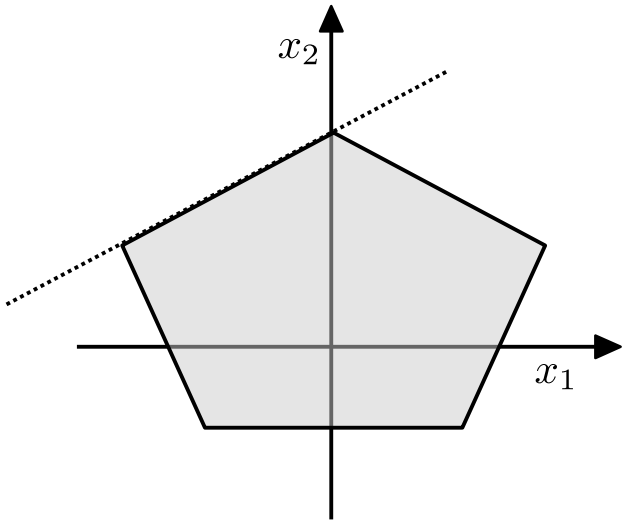
- How would you represent the figure below with the help of Boolean perceptrons?

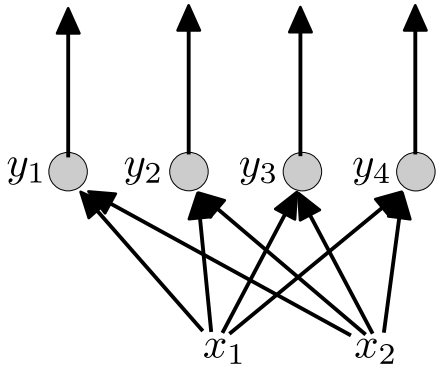




More complex boundaries

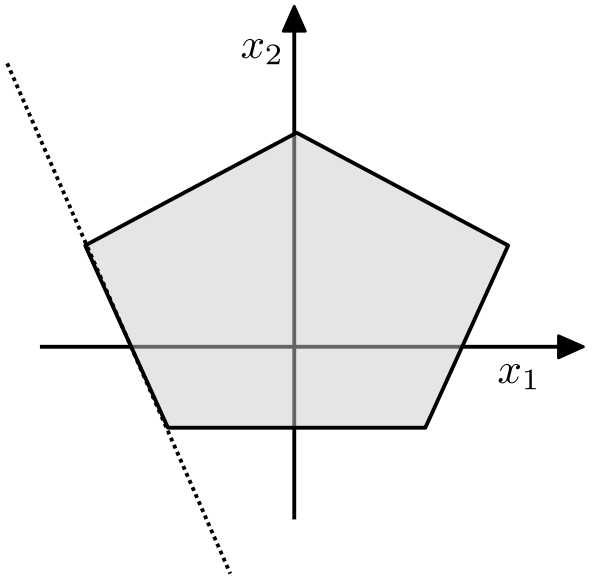
- How would you represent the figure below with the help of Boolean perceptrons?

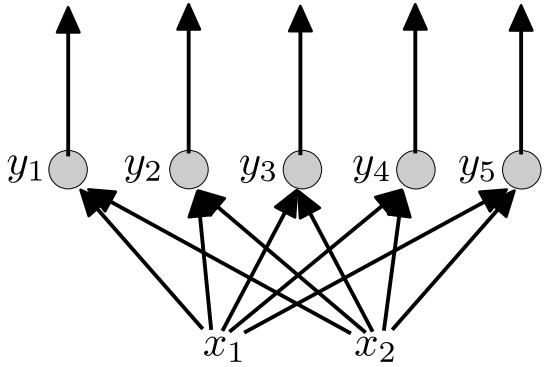




More complex boundaries

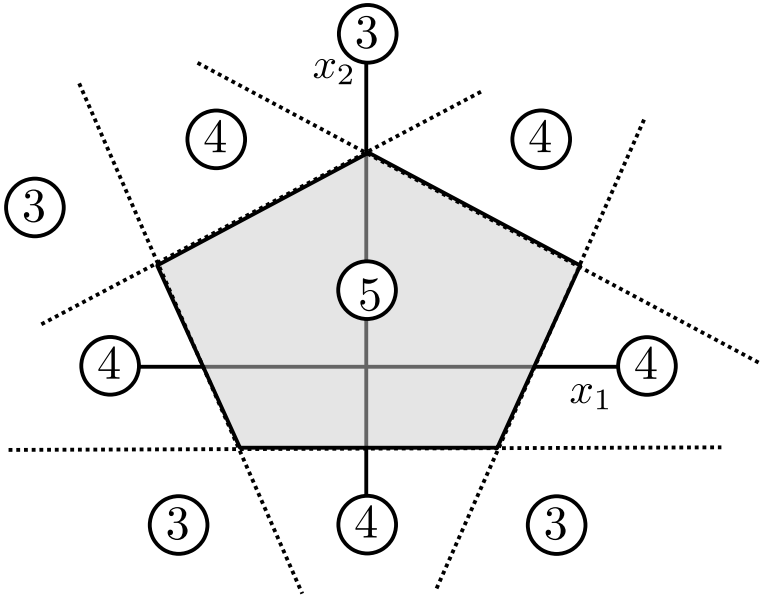
- How would you represent the figure below with the help of Boolean perceptrons?

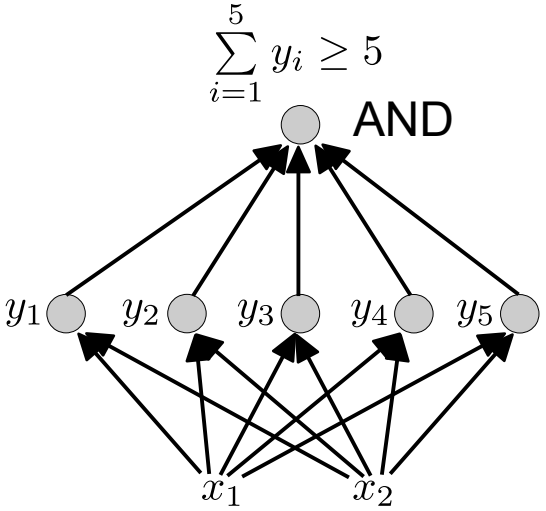




More complex boundaries

- How would you represent the figure below with the help of Boolean perceptrons?

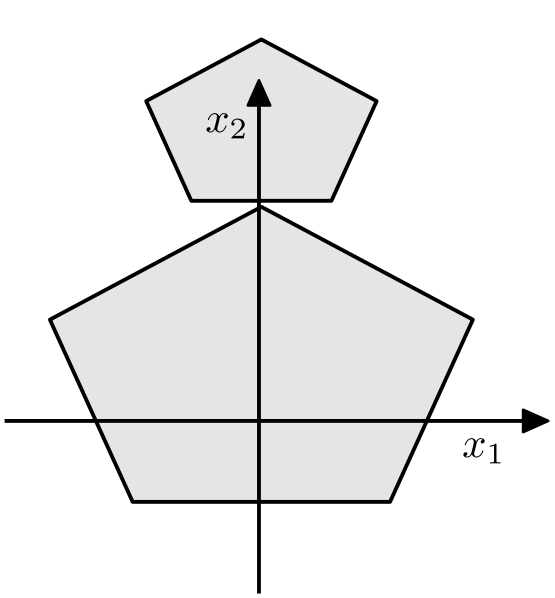


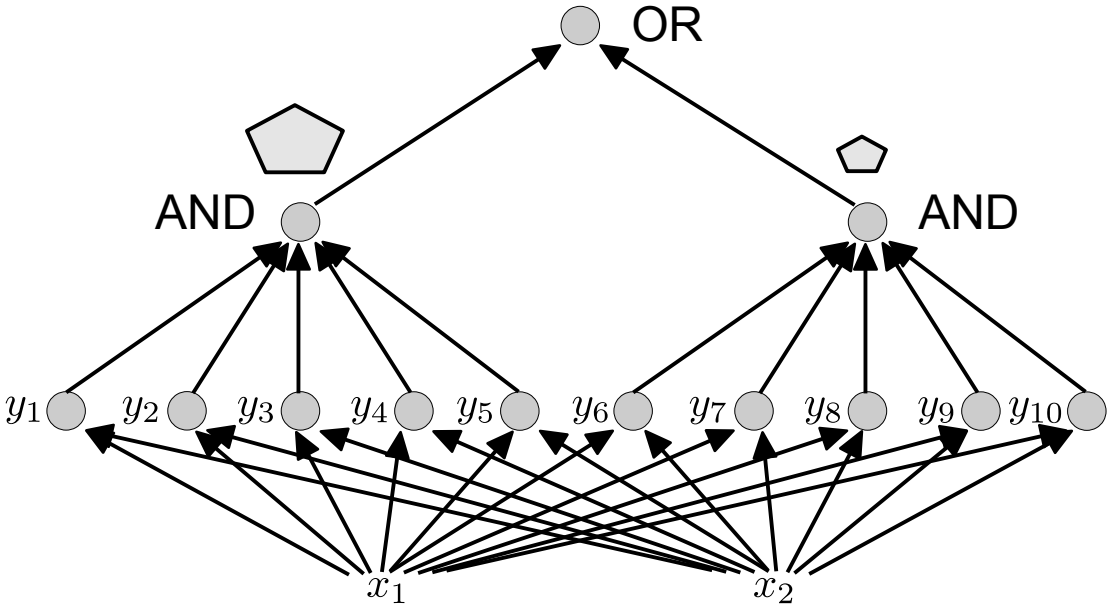


- Normalize the output of each unit y_i to 1 and add.
- Choose threshold of ≥5.

Even more complex boundaries

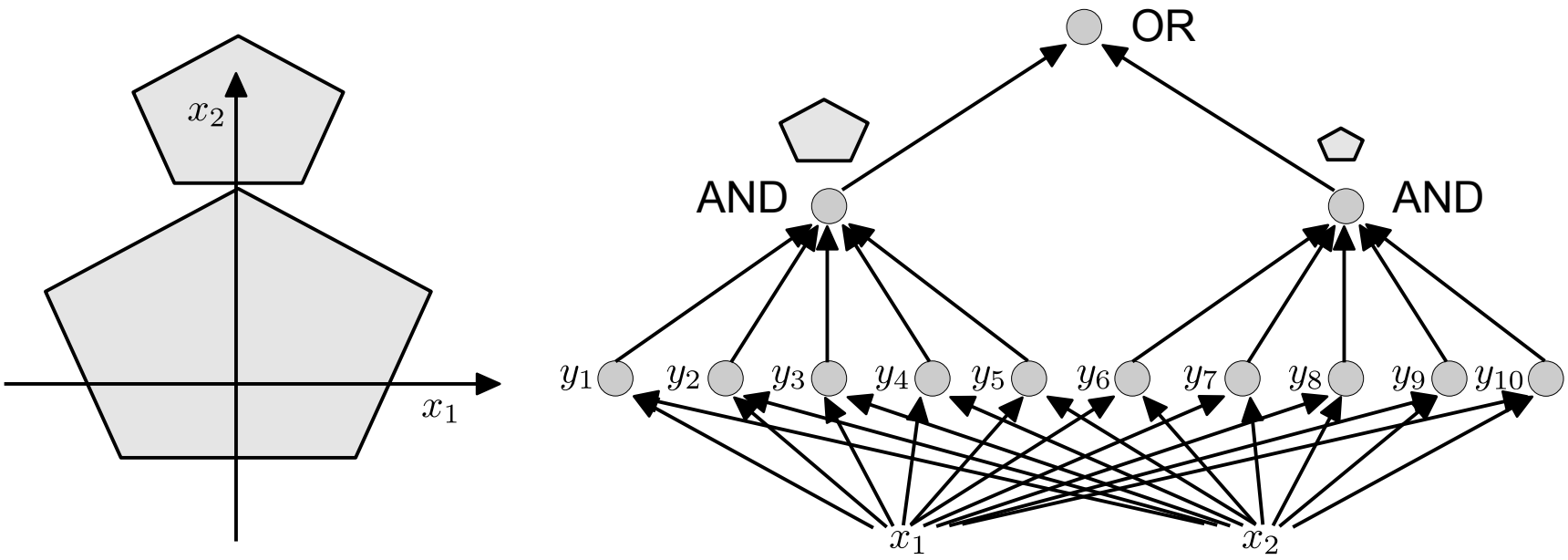
- The figure below would require a third layer of perceptrons:





Even more complex boundaries

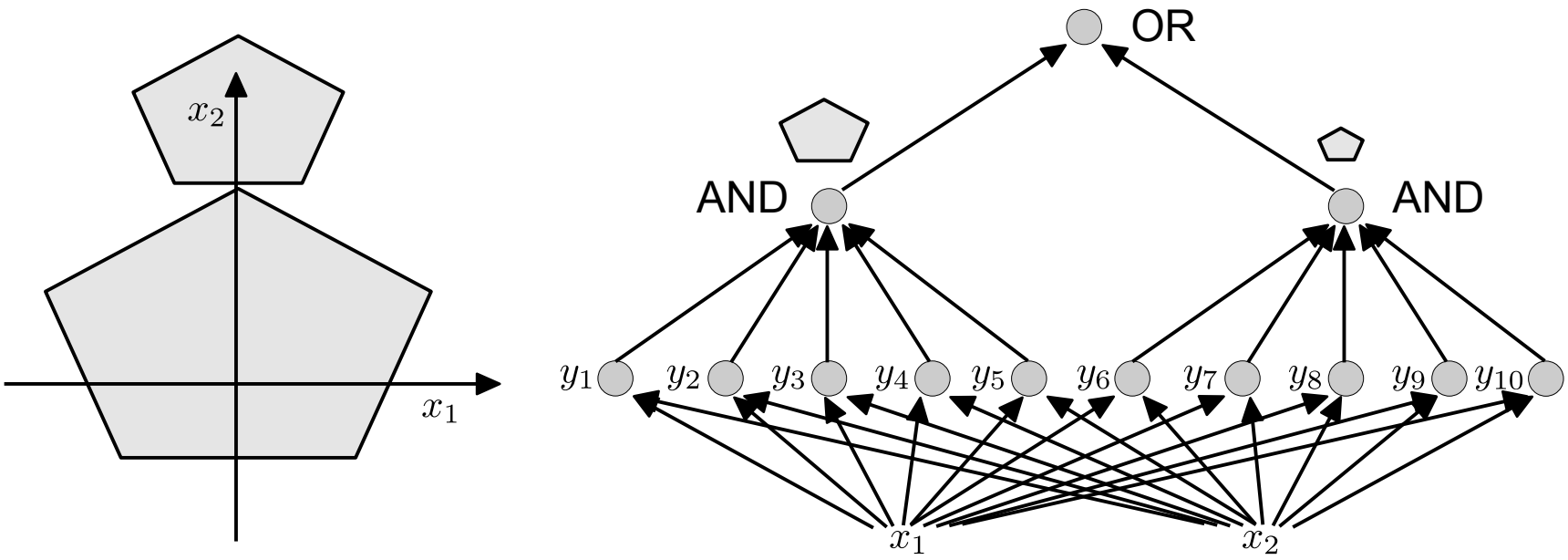
- The figure below would require a third layer of perceptrons:



- Since any abitrary boundary can be approximated by polygones it is possible to describe any abitrary figure with a sufficiently complex network of perceptrons.

Even more complex boundaries

- The figure below would require a third layer of perceptrons:

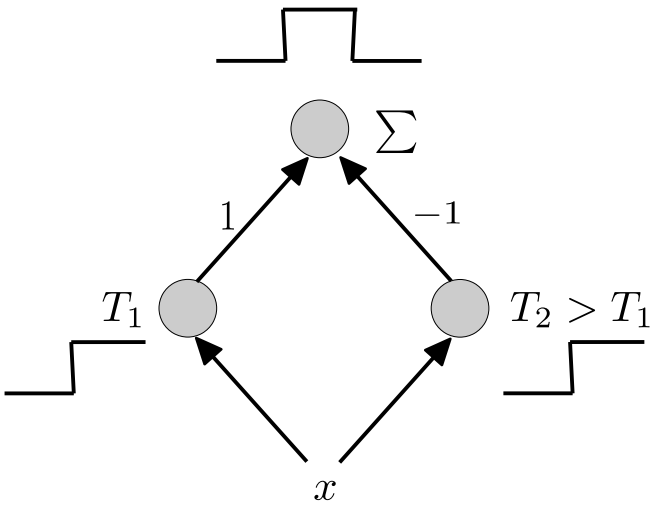


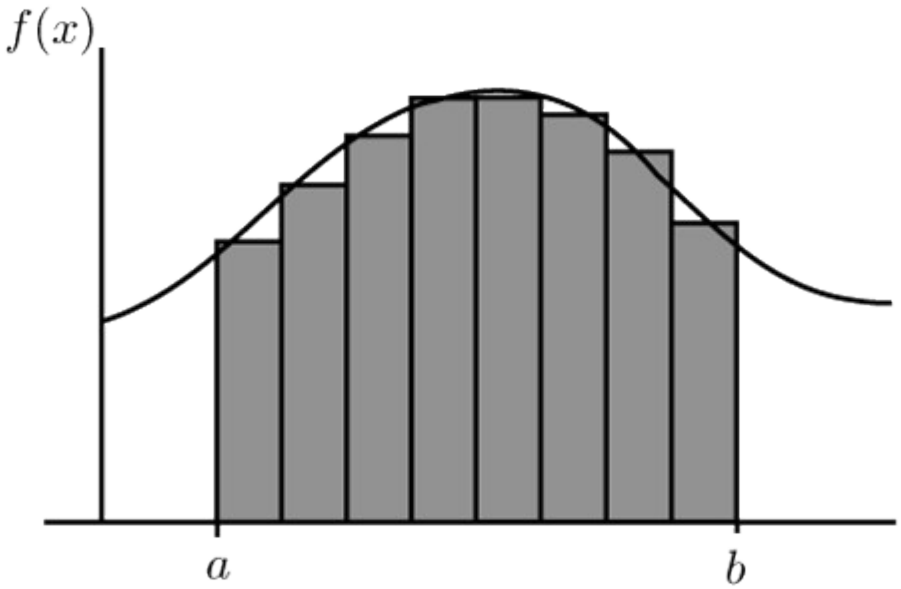
- Since any abitrary boundary can be approximated by polygones it is possible to describe any abitrary figure with a sufficiently complex network of perceptrons.

- Thus an MLP is in principle capable of describing any kind of classification in any hyperspace to arbitrary precision.

Approximation of an abitrary function

- The following unit represents a **step function**:





- With a group of such computing units it is possible to approximate any arbitrary function to abitrary precision.

Recap: activation function

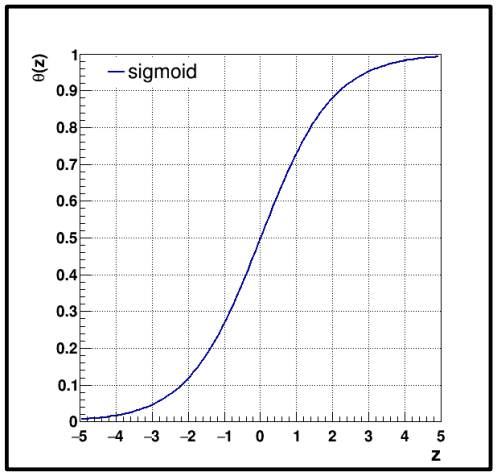
- We discussed that the activation function could be any function.
- A few popular examples are given below:

Rectified linear unit (**ReLU**):

$$\theta(z) = \begin{cases} z & z \geq 0 \\ 0 & \text{sonst} \end{cases}$$

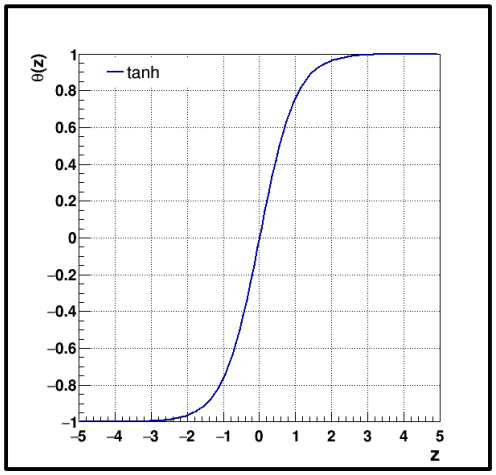
Sigmoid:

$$\theta(z) = \frac{1}{1+\exp(-z)}$$



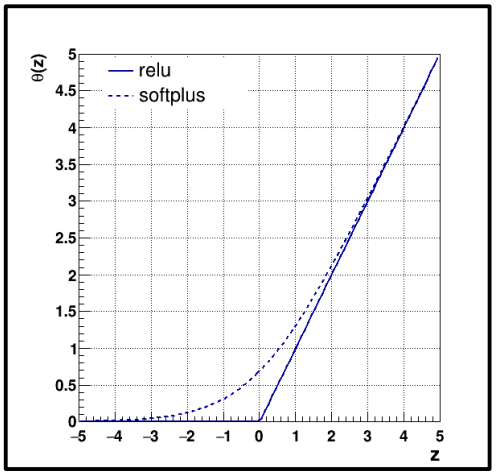
tanh:

$$\theta(z) = \tanh(z)$$



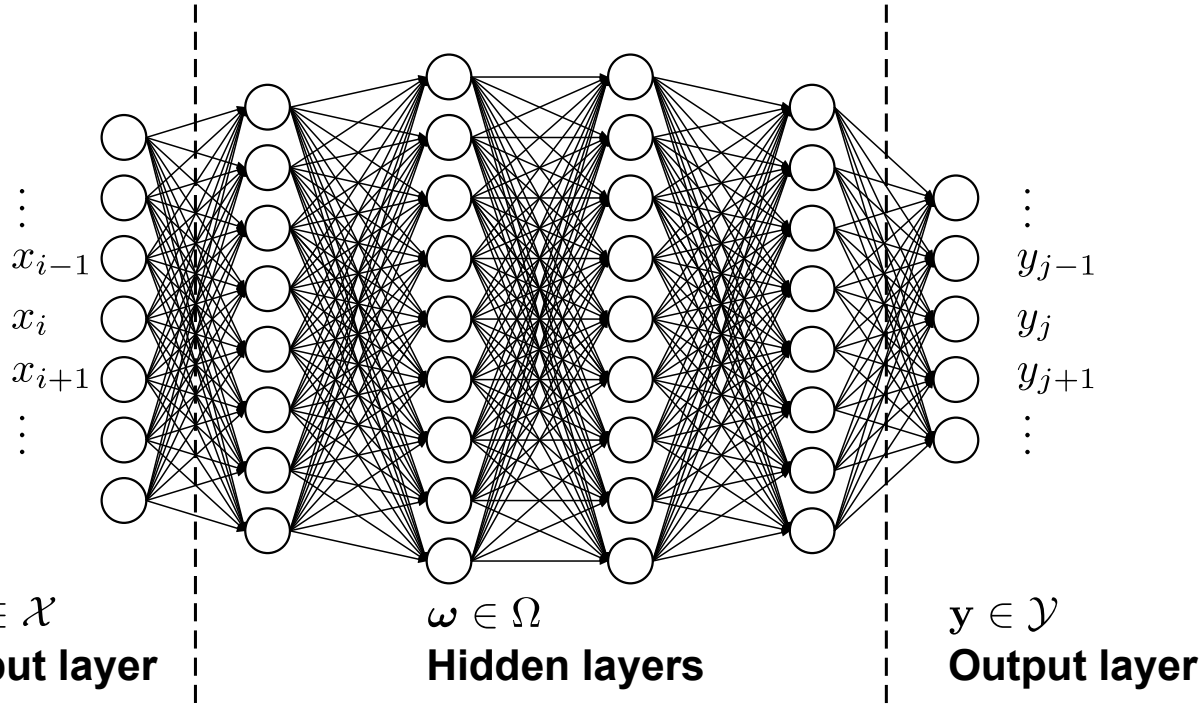
Softplus:

$$\theta(z) = \log(1+\exp(z))$$



Recap: multilayer perceptron (MLP)

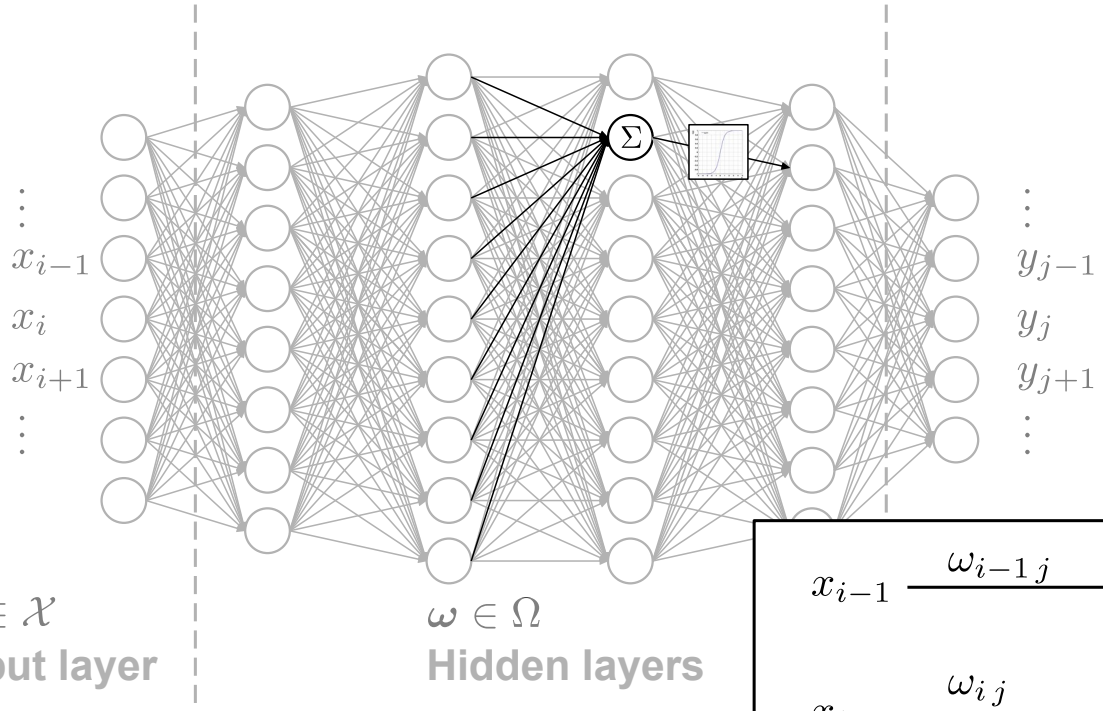
- We have discussed that the ability to express Boolean relations increases when using more than one perceptron, organized in layers → **multilayer perceptron (MLP)**:



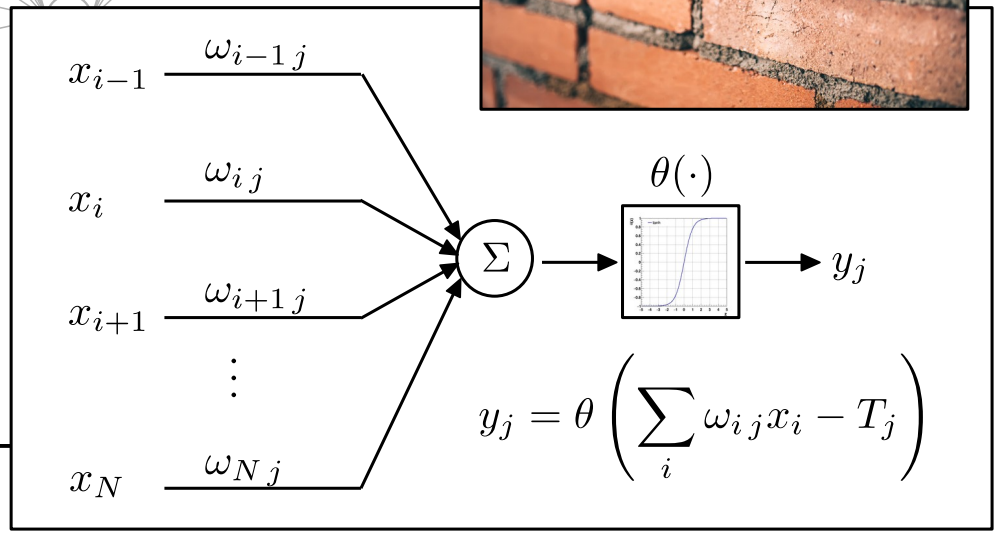
- For the time being, we will discuss the MLP in form of the **fully-connected feed-forward NN**.

The fully-connected feed-forward (artificial) NN (ANN)

- **Fully-connected**: All nodes of consecutive layers are *connected* with each other.
- **Feed-forward**: Inputs are propagated only in *forward* direction.

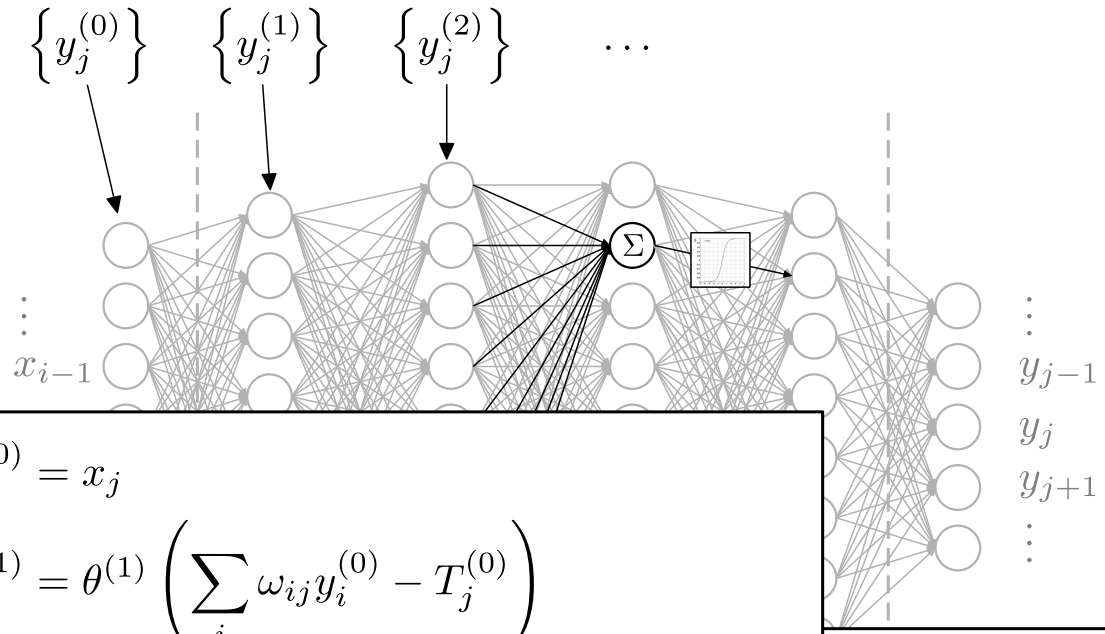


Building blocks are single perceptrons (also called **nodes**):



Priv.-Doz. Dr. Roger Wolf
http://ekpwww.physik.uni-karlsruhe.de/~rwolf/

Mathematical notation



$$y_j^{(0)} = x_j$$

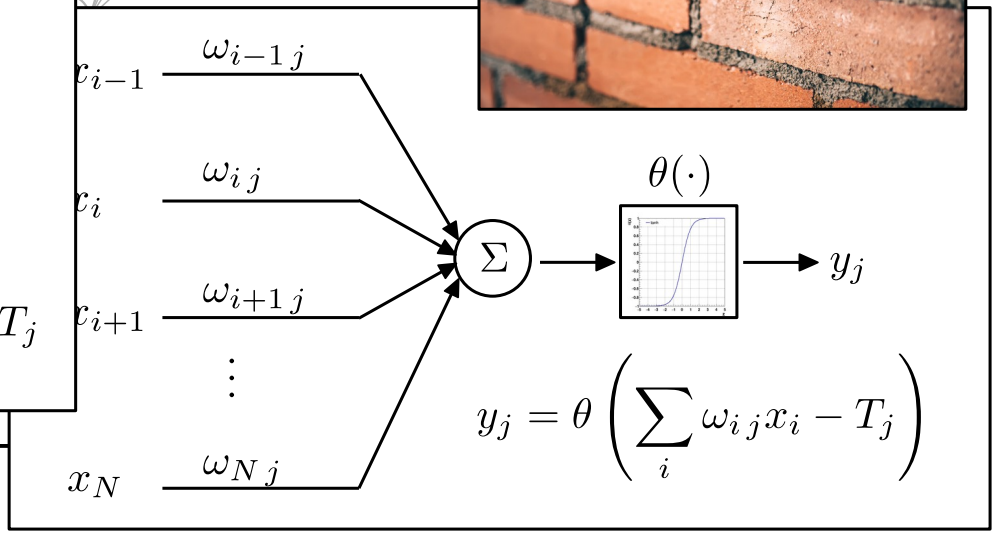
$$y_j^{(1)} = \theta^{(1)}\left(\sum_i \omega_{ij} y_i^{(0)} - T_j^{(0)}\right)$$

$$y_j^{(2)} = \theta^{(2)}\left(\sum_i \omega_{ij} y_i^{(1)} - T_j^{(1)}\right)$$

$$\vdots$$

$$y_j^{(k)} = \theta^{(k)}\left(z_j^{(k)}\right) \quad z_j^{(k)} = \sum_i \omega_{ij} y_i^{(k-1)} - T_j$$

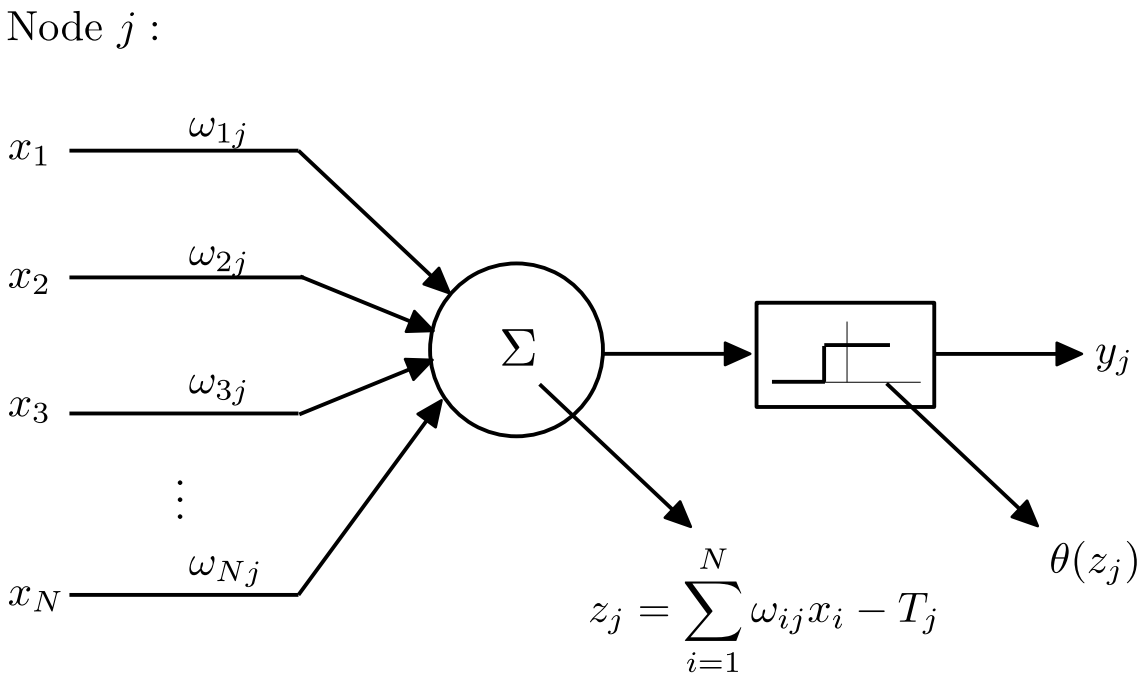
Building blocks are single perceptrons (also called **nodes**):



Priv.-Doz. Dr. Roger Wolf
http://ekpwww.physik.uni-karlsruhe.de/~rwolf/

NB: Weights, thresholds, biases

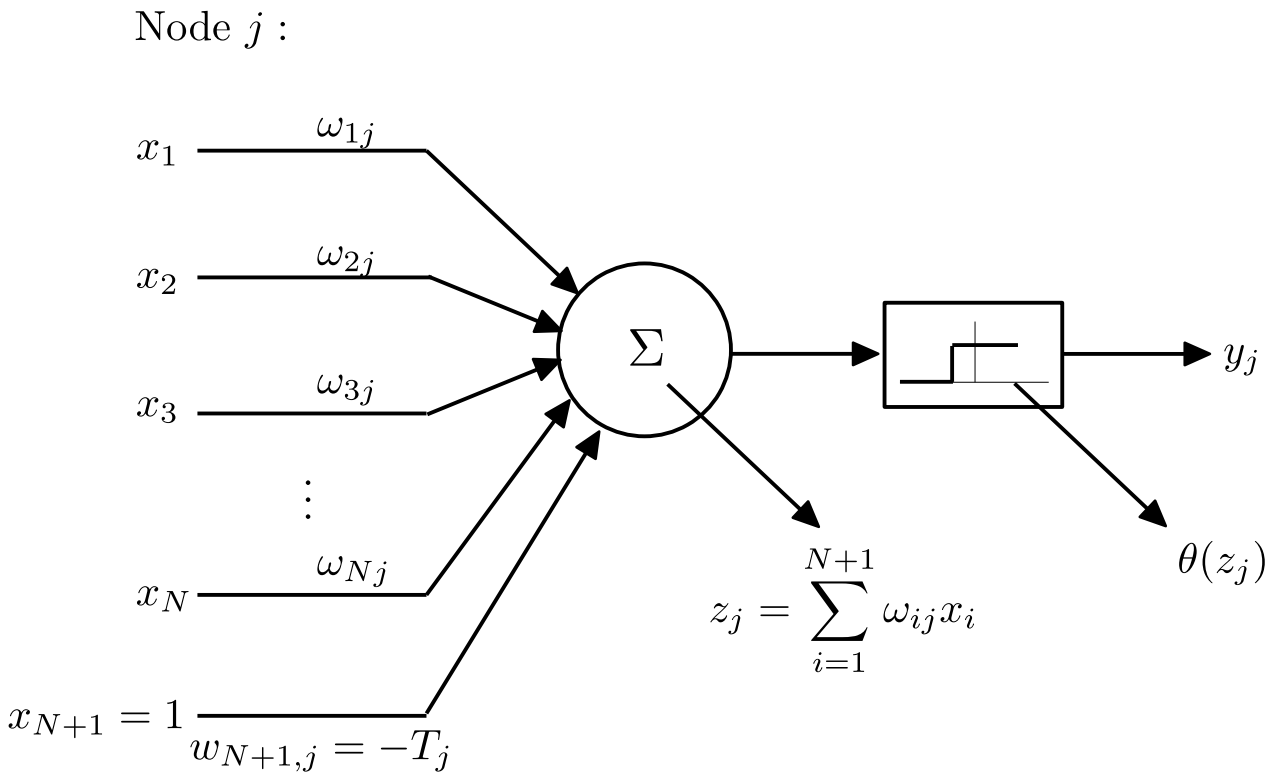
- The MLP is a well-defined multi-dimensional function of the input features $\{x_i\}$, weights $\{\omega_{ij}\}$, and thresholds $\{T_j\}$:



- Often you can see the thresholds $\{T_j\}$ called **biases** and abbreviated by $\{b_j\}$.

NB: Weights, thresholds, biases

- The MLP is a well-defined multi-dimensional function of the input features $\{x_i\}$, weights $\{\omega_{ij}\}$, and thresholds $\{T_j\}$:

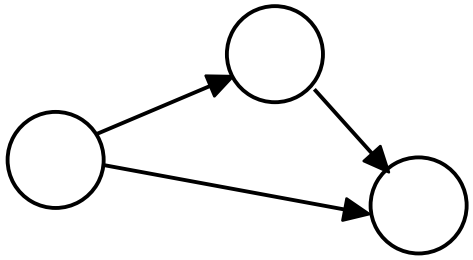


- We will use this fully equivalent notation for clarity of fomulars, in the following.

Depth

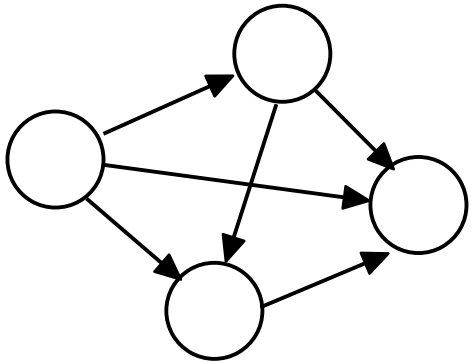
- A feed-forward NN can be understood as a **directed graph** of depth d.
- A directed graph has *sources* and *drains*. The depth of a graph is the longest path between a source and a drain.

Example 1:



Depth?

Example 2:



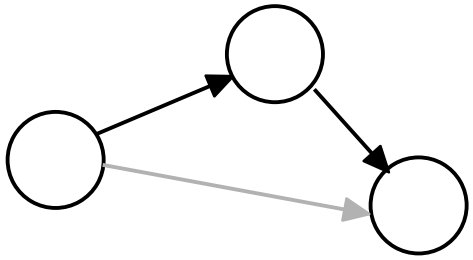
Depth?

Depth

- A feed-forward NN can be understood as a **directed graph** of depth d.

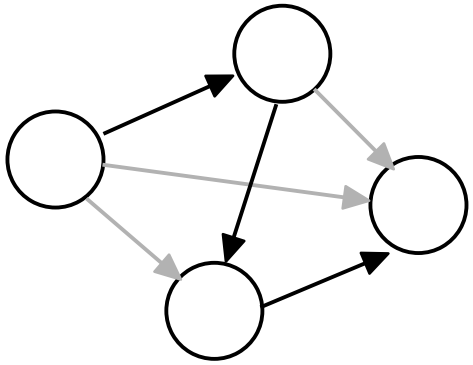
- A directed graph has *sources* and *drains*. The depth of a graph is the longest path between a source and a drain.

Example 1:



Depth? – 2

Example 2:



Depth? – 3

- An NN with a depth of $d > 2$ (i.e. an ANN with more than 2 hidden layers) we call *deep*.

Backup