Neural Networks

COMP9414: Artificial Intelligence

Lecture Overview

- Motivation
- Biological and artificial neurons
- Single-layer perceptron
- Multi-layer perceptron
- Neural network design
- Neural network architectures

Lecture Overview

- Motivation
- Biological and artificial neurons
- Single-layer perceptron
- Multi-layer perceptron
- Neural network design
- Neural network architectures

Motivation

- Great ability of cognitive beings to carry out some tasks: shape recognition, speech and image processing, etc.

- It seemed important to understand and emulate successful mechanisms from humans and animals: Parallelism and high connectivity.

- A branch of artificial intelligence: Artificial Neural Networks.

Motivation

- New paradigm (non-algorithmic) to process information (neurocomputing): learning and adaptation, distributed and parallel processing.
- New computational tools (faster and cheaper)
- In the future: more caution and theoretical support. Open issue: Generalization.

Motivation

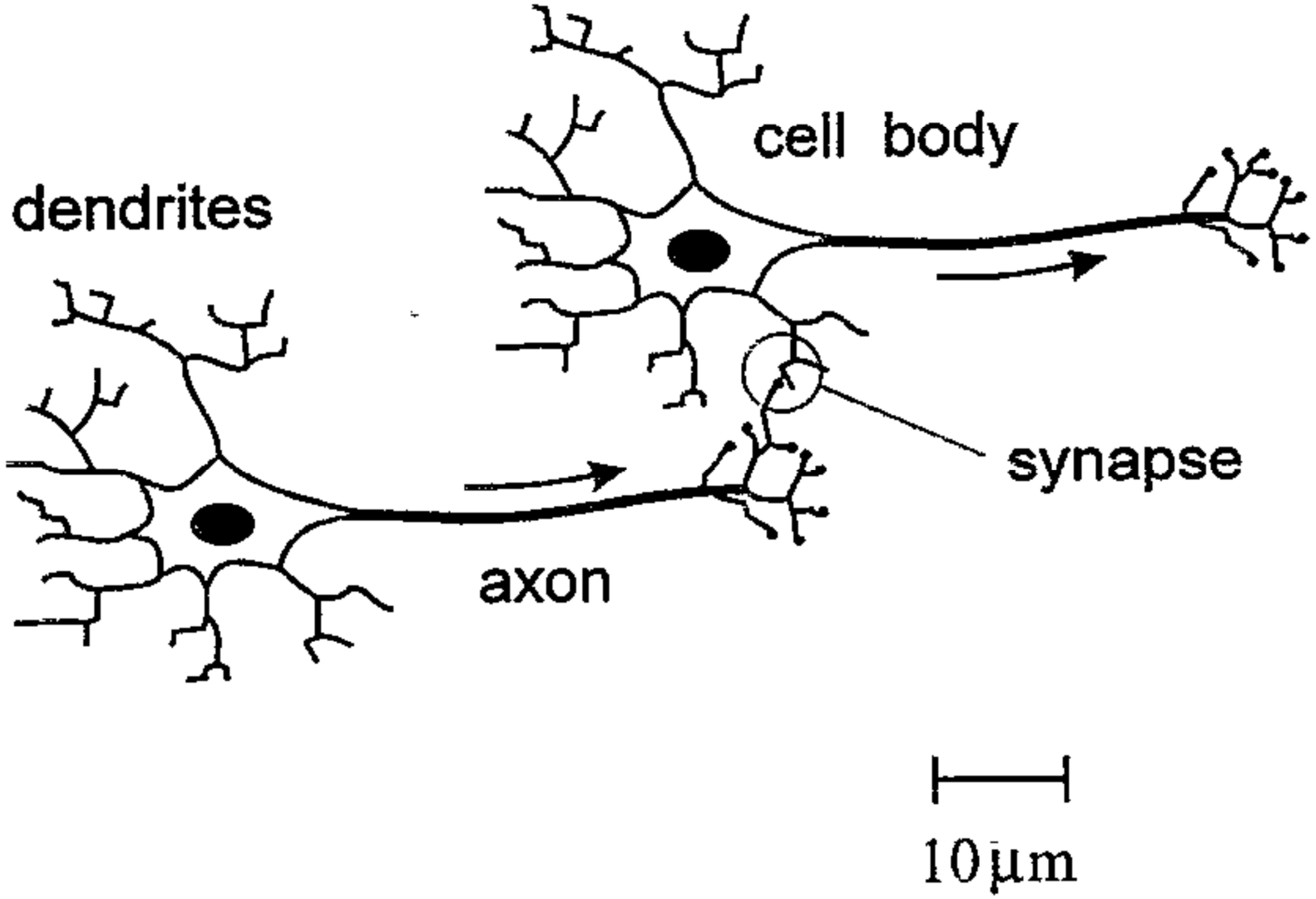
- The general problem of function approximation can be divided into two subproblems:
  - Classification: to approximate a function that represents the membership function of an entity – characterized by a set of input variables, either continuous or discrete – to a particular class (output with discrete values), e.g., character recognition.
  - Regression: to approximate the generating function (unknown) of a process by mapping elements from the input variables to output variables. Usually, continuous values are used.

$$y = f(x, w)$$

Lecture Overview

- Motivation
- Biological and artificial neurons
- Single-layer perceptron
- Multi-layer perceptron
- Neural network design
- Neural network architectures

Biological Neuron



Biological Neuron

- The brain is made up of neurons (nerve cells) which have
  - a cell body (soma)
  - dendrites (inputs)
  - an axon (outputs)
  - synapses (connections between cells)

- Synapses can be excitatory or inhibitory and may change over time.

- When the inputs reach some threshold an action potential (electrical pulse) is sent along the axon to the outputs.

Biological Neuron

- Human brain has 100 billion neurons (~ 10^{10} - 10^{11} neurons) with an average of 10,000 synapses each (some even with 100,000 synapses).

- Latency is about 3-6 milliseconds.

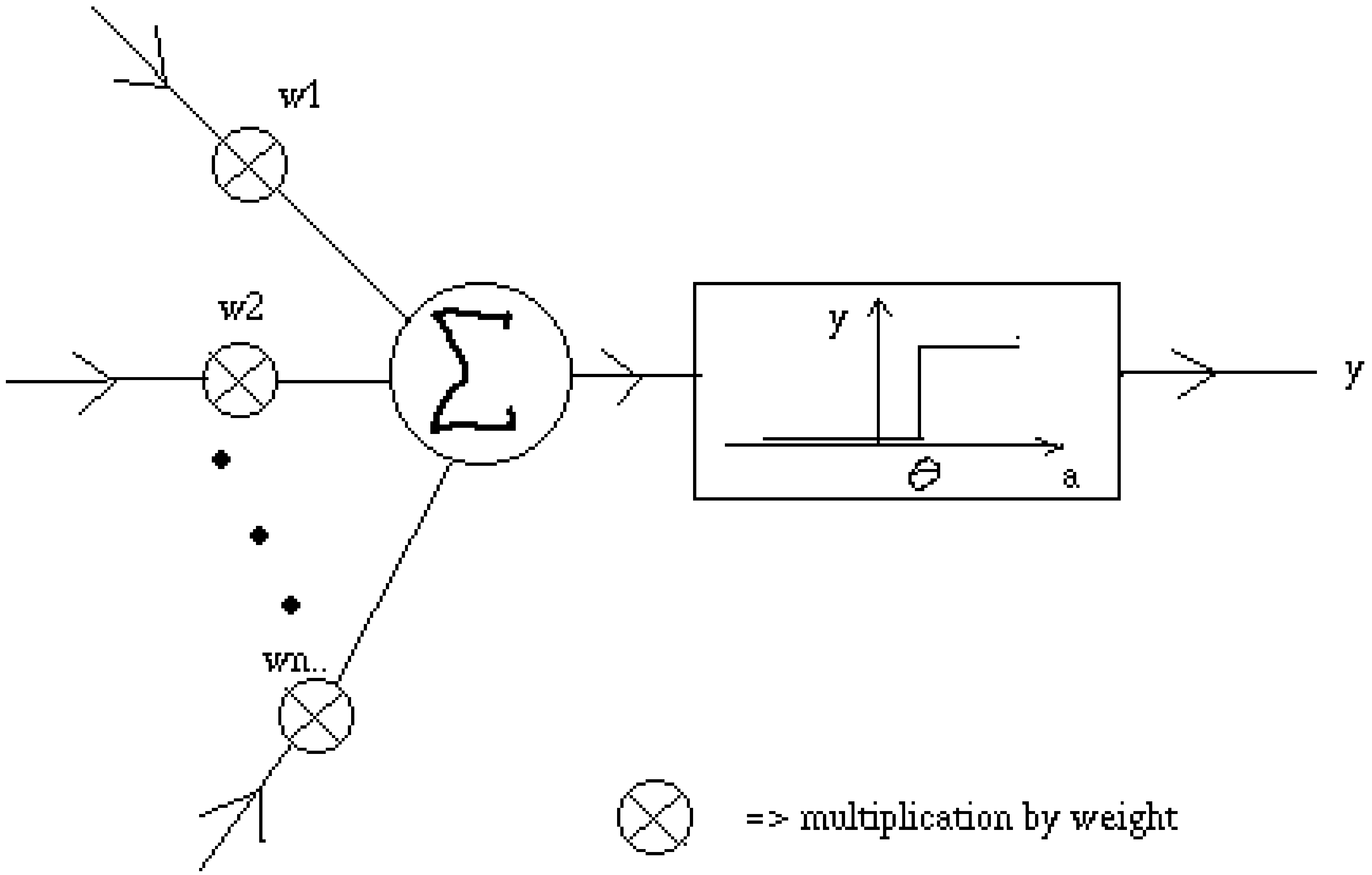
- At most a few hundred “steps” in any mental computation, but massively parallel.

Artificial Neuron

- Automata characterized by:
  - An internal state.
  - Input signals.
  - Activation and transfer functions.

Artificial Neuron

- McCulloch-Pitts' model (1943)

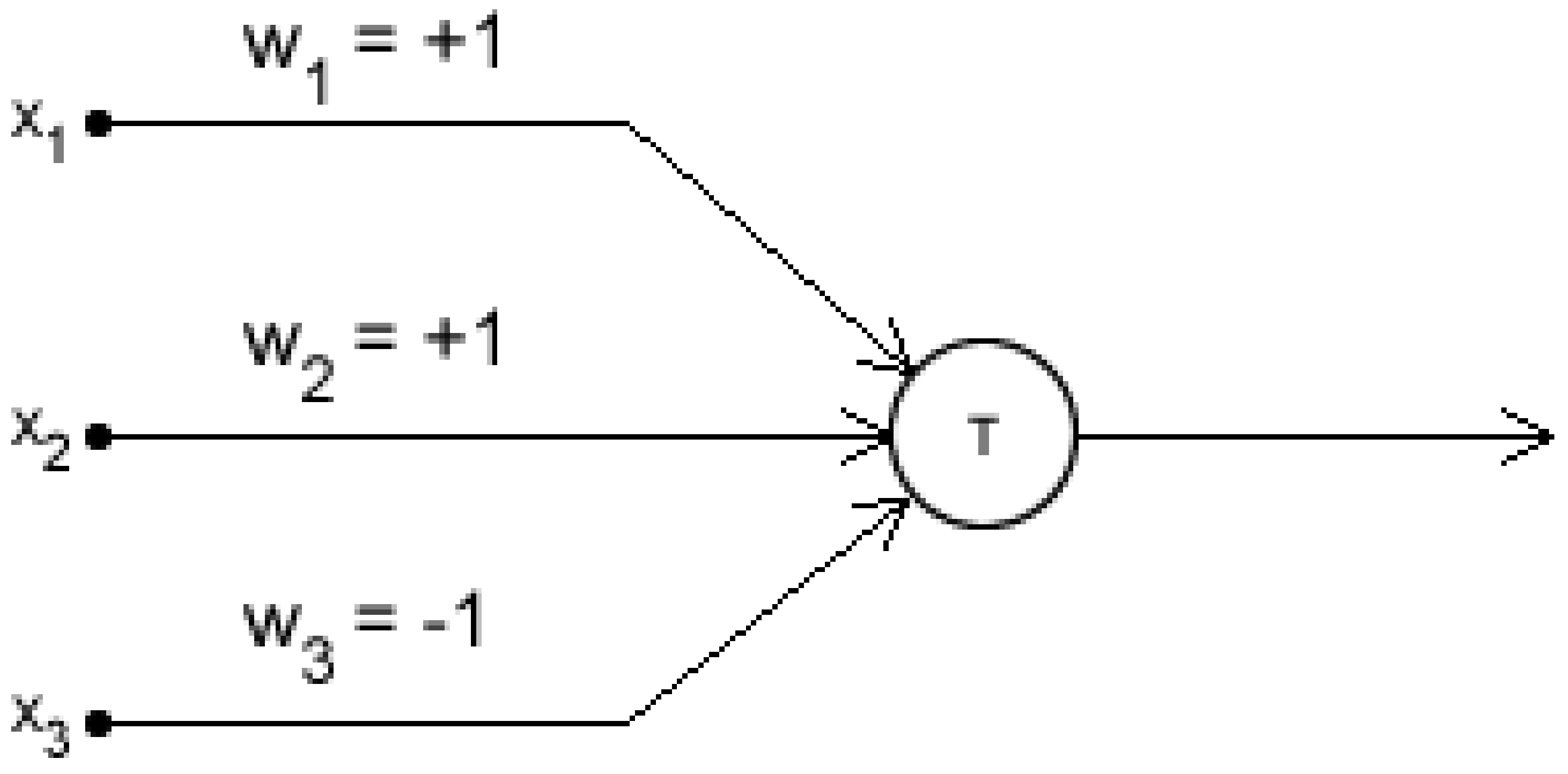


Artificial Neuron

- McCulloch-Pitts model:
  - Inputs either 0 or 1.
  - Output 0 or 1.
  - Input can be either excitatory or inhibitory.

- Summing inputs
  - If input is 1, and is excitatory, add 1 to sum.
  - If input is 1, and is inhibitory, subtract 1 from sum.

- Threshold,
  - if sum < threshold ϴ, output 0.
  - Otherwise, output 1.



$$sum = x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + \dots$$

if *sum* < $\boldsymbol{\theta}$ **then** *output* is 0
else *output* is 1.

Learning

- Ability of a neuron (or neural net) to adjust connections (weights) to obtain the intended output or that meets certain criteria.

- **Hebbian learning (1949):** When a neuron A persistently activates another nearby neuron B, the connection between the two neurons becomes stronger. Specifically, a growth process occurs that increases how effective neuron A is in activating neuron B. As a result, the connection between those two neurons is strengthened over time.

    - "Neurons that fire together, wire together", Hebb.

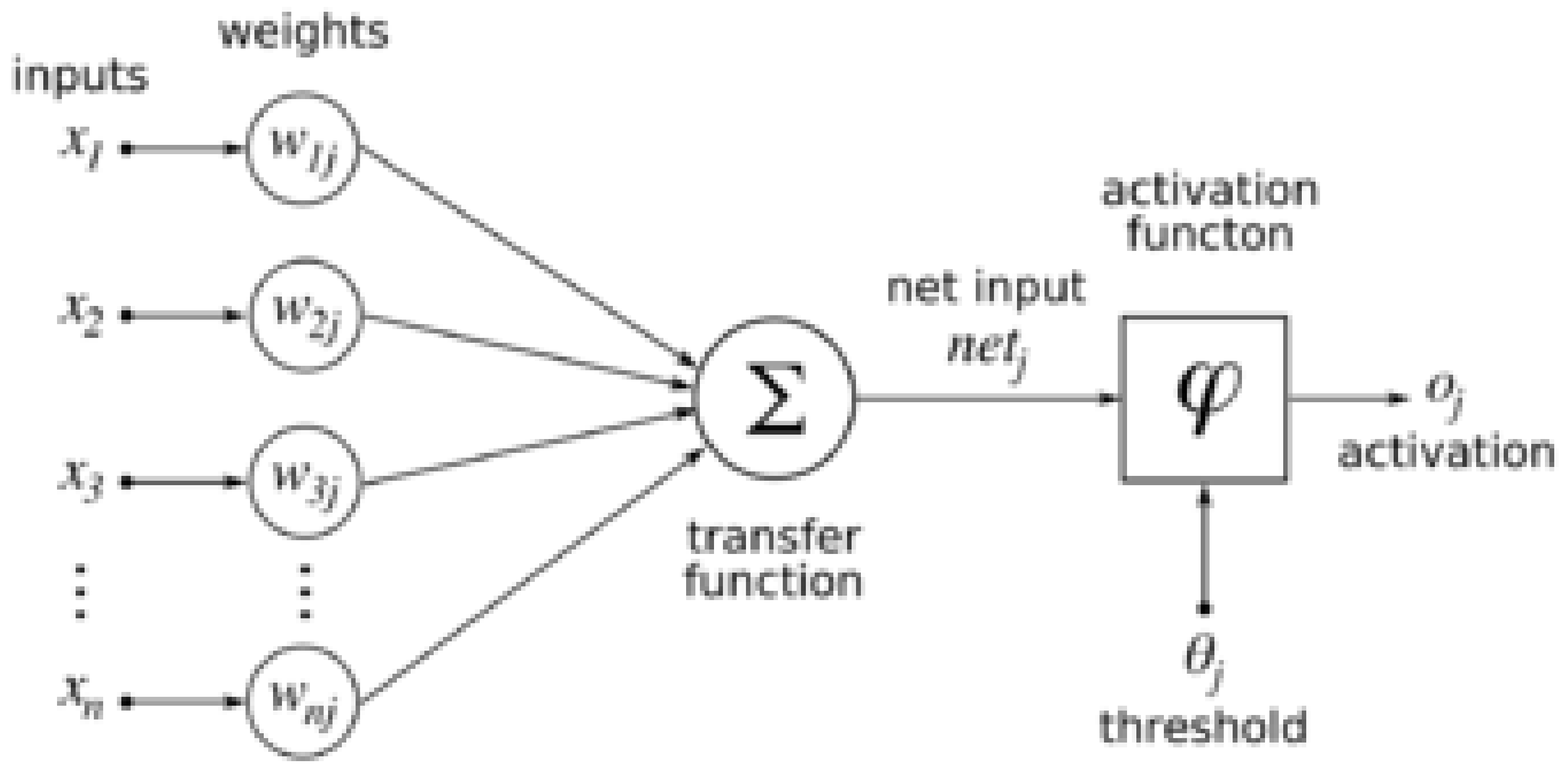
Artificial Neural Networks

- Information processing architecture loosely modelling the brain
- Consists of many interconnected processing units (neurons)
  - Work in parallel to accomplish a global task
- Generally used to model relationships between inputs and outputs or to find patterns in data
- Characterized by (i) number of neurons, (ii) interconnection architecture, (iii) weight values, (iv) activation and transfer functions.

Artificial Neural Networks

❑ ANNs nodes have

- ➢ inputs edges with some weights
- ➢ outputs edges with weights
- ➢ activation level (function of inputs)



- Weights can be positive or negative and may change over time (learning).
- The input function is the weighted sum of the activation levels of inputs.
- The activation level is a non-linear transfer function *g* of this input:

$$\text{activation}_j = g(s_j) = g\left(\sum_i w_{ij} x_i\right)$$

Some nodes are inputs (sensing), some are outputs (action)

Artificial Neural Networks

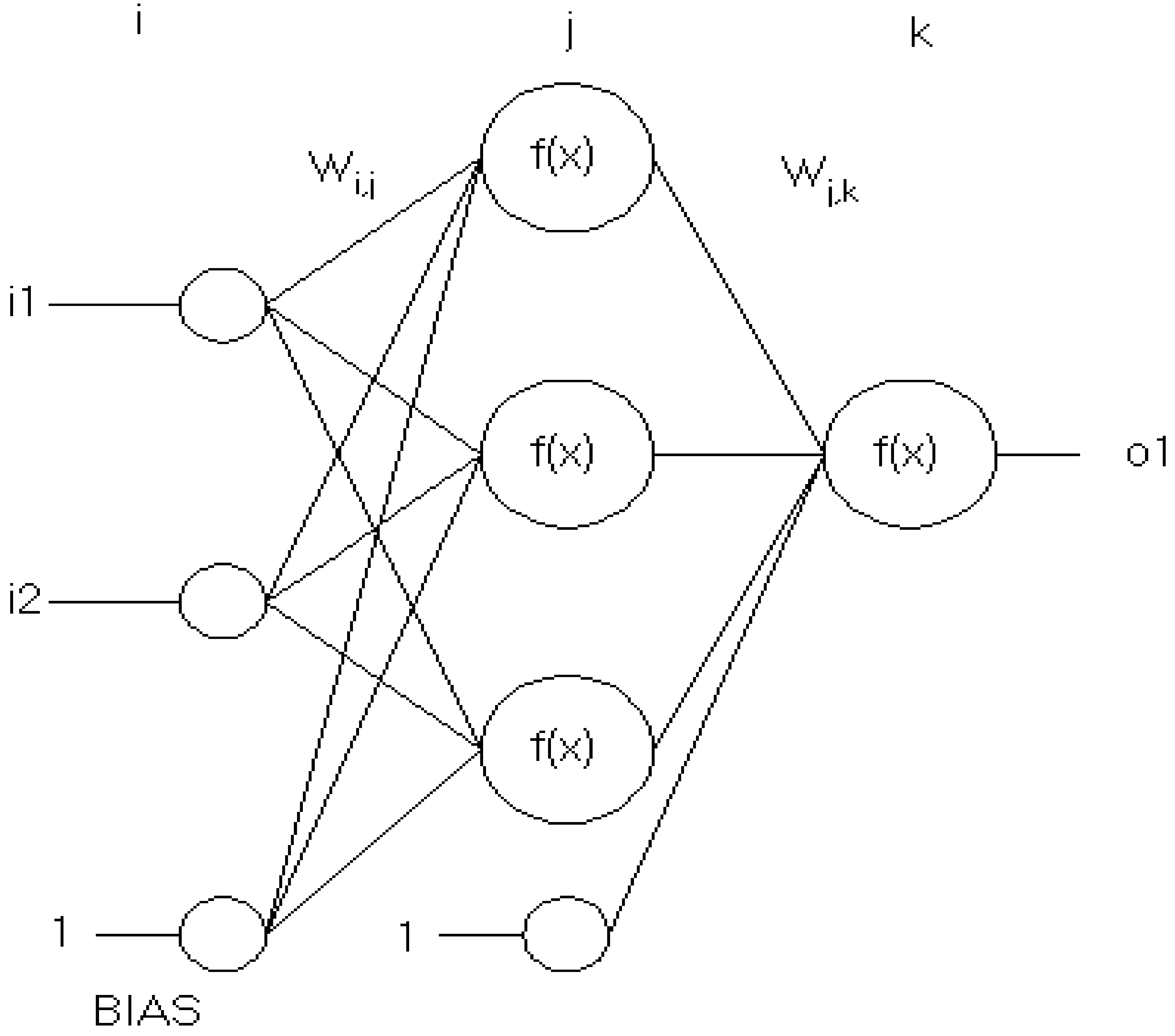


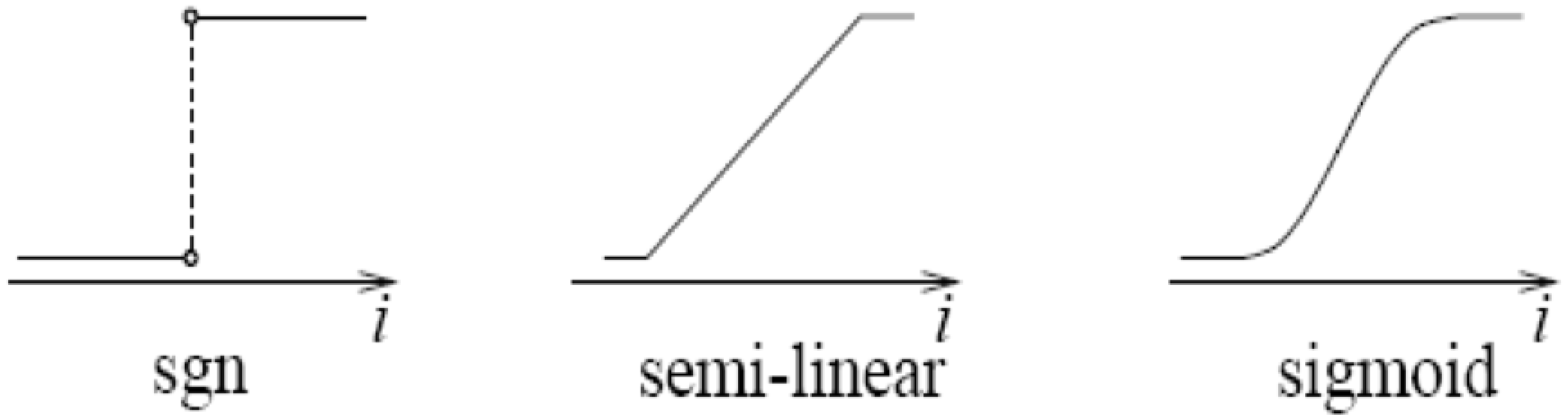
Figure - 1 A simple 2x3x1 NN. The lines connecting the neurons represent the weights. Also shown are the bias nodes used to shift the neuron transfer function and improves the network performance.

Artificial Neural Networks

- Neural networks (NN) might work in two ways:
    - Learning: adapting its weights, architecture, activation and transfer functions.
    - Simulation or recognition: it is used for information processing.
- NN learning: Supervised (through examples), unsupervised.

Activation Functions

Function $g(s)$ takes the weighted sum of inputs and produces output for node, given some threshold.



$$g(s) = \begin{cases} 1 \text{ if } s \ \geq 0 \\ 0 \text{ if } s < 0 \end{cases}$$

Lecture Overview

- Motivation
- Biological and artificial neurons
- Single-layer perceptron
- Multi-layer perceptron
- Neural network design
- Neural network architectures

Single-layer Perceptron

- Frank Rossenblat, 1957
  - Use a logic threshold function for classification tasks.
- Classification and discriminant functions
  - How can we make the classification?
  - Using discriminant functions defined in the vector space we want to classify and produce a value that can be compared.

Single-layer Perceptron

- If we have a function (g_i) for each class (s_i). The classification rule is:

$$u \in s_i \text{ iff } g_i\,(u) > g_j\,(u);\ \ j \neq i$$

- For 2-classes problems, it can be reduced to one function as:

$$g(u) = g_1(u) - g_2(u), \text{ then } u \in s_1\ g(u) > 0 \text{ and } u\ \in\ s_2 \text{ otherwise}$$

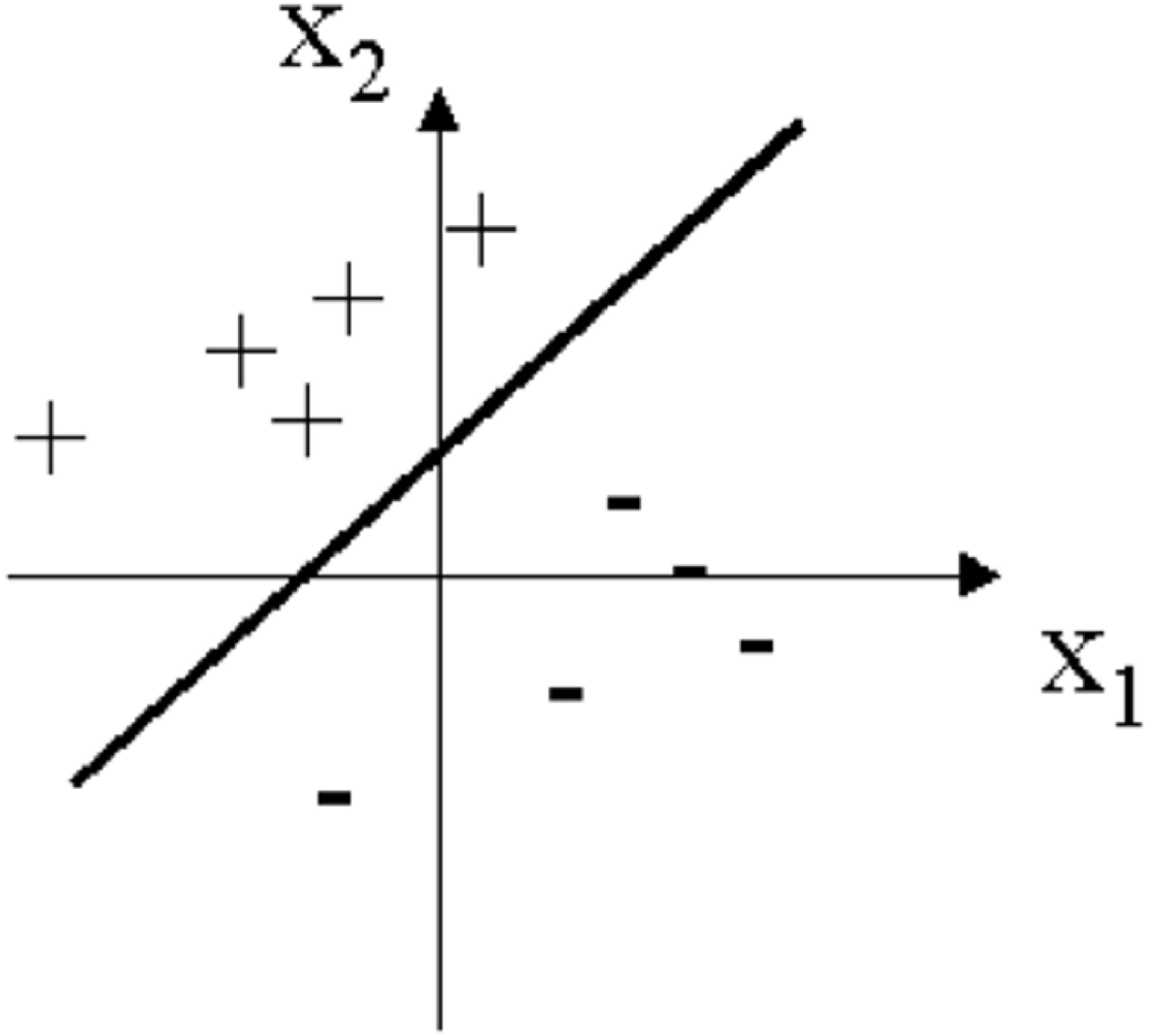
Single-layer Perceptron

- **Linear classification with discriminant function using perceptron:**

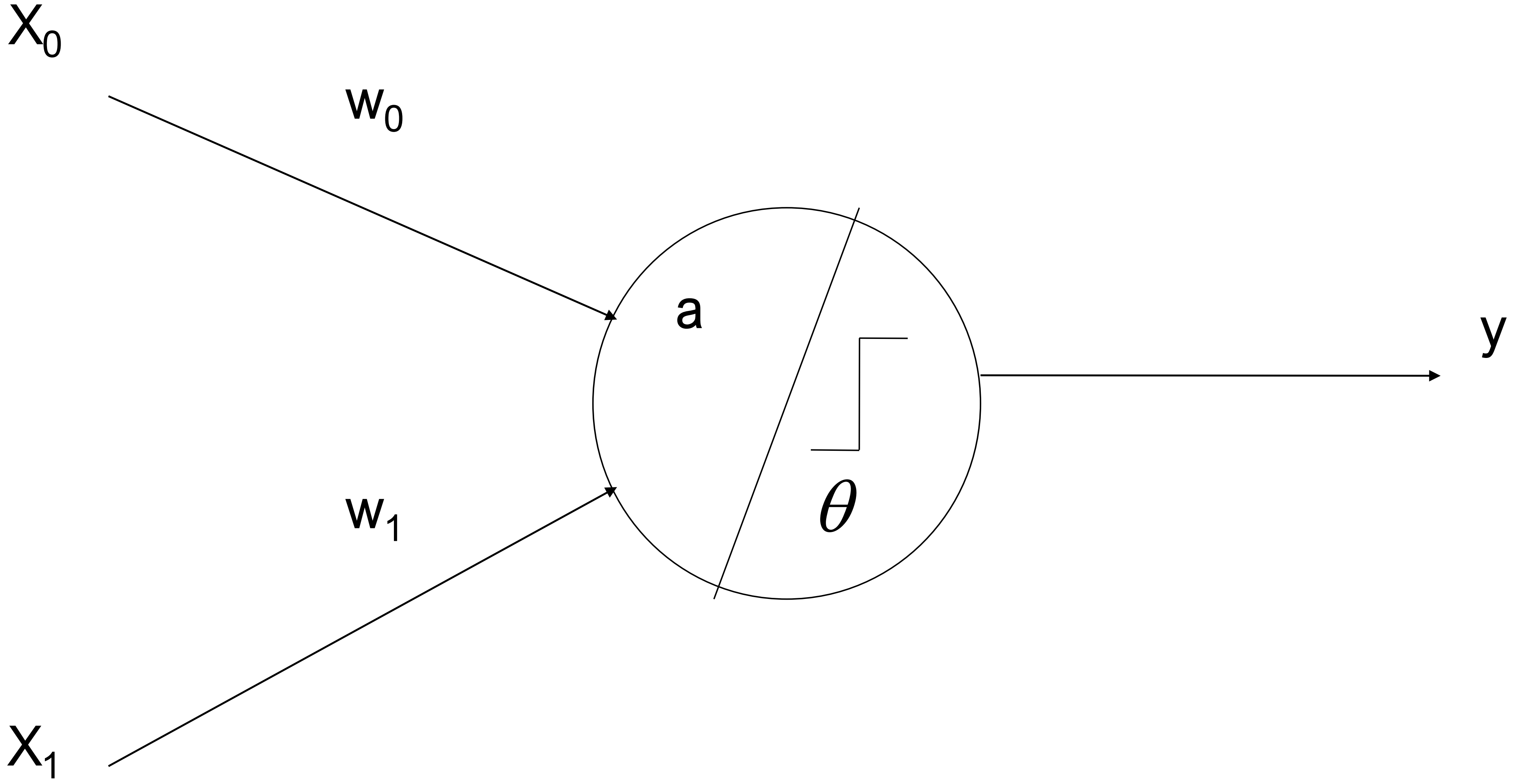
$$g\,(u) = w_1\,u_1 + w_2\,u_2 + \ldots + w_n\,u_n = w \,.\, u$$

- The value associated with g(u) = 0 corresponds to the border, a hyperplane, that divides the two classes.
- Perceptrons are appropriate only for classes linearly separable.
- The learning problem is reduced to finding a hyperplane that separates the two classes.

Single-layer Perceptron

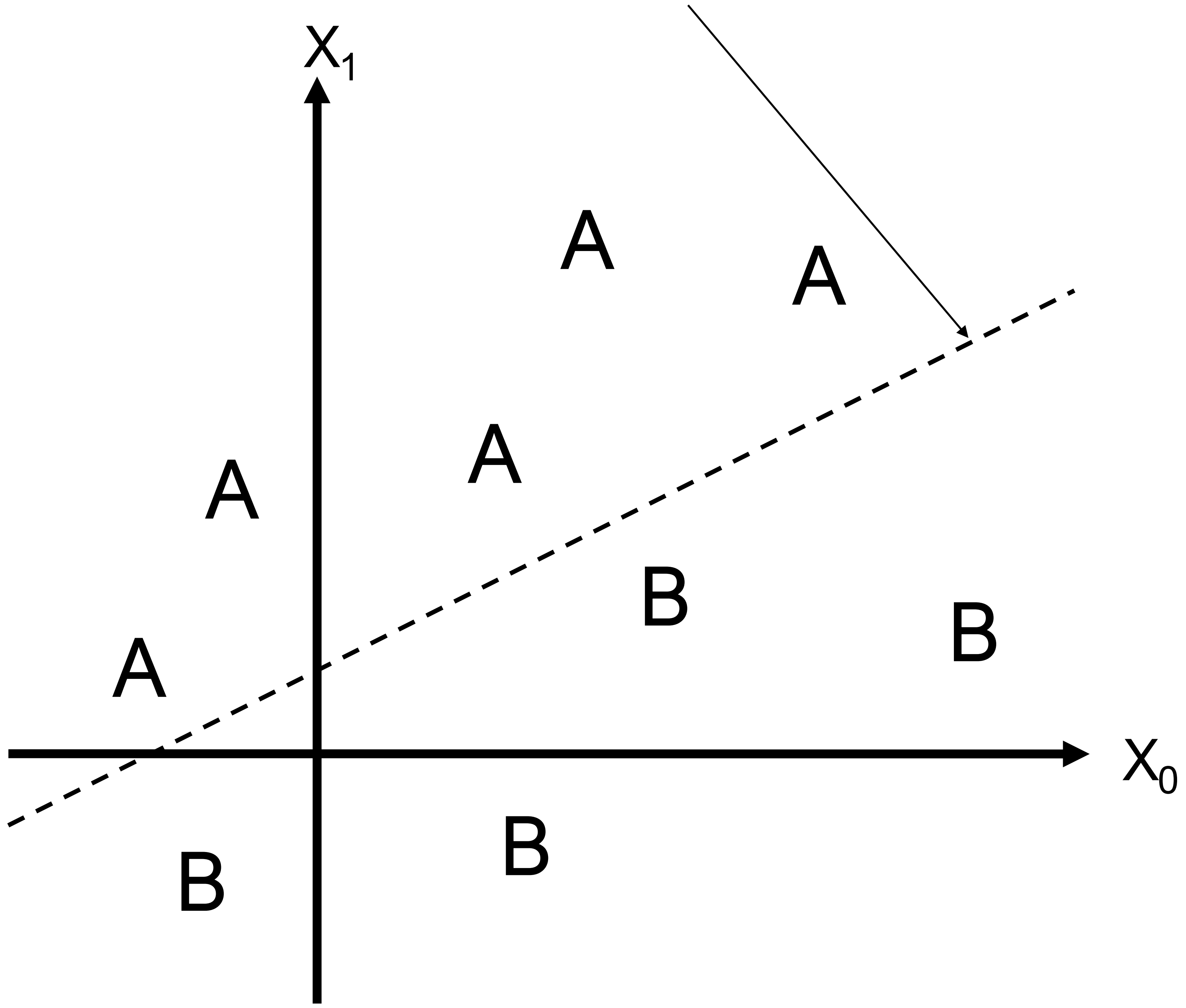


$y = ax + b$



$$X_0\omega_0 + X_1\omega_1 = \theta$$

$$\Rightarrow X_1 = -\frac{\omega_0}{\omega_1}X_0 + \frac{\theta}{\omega_1}$$

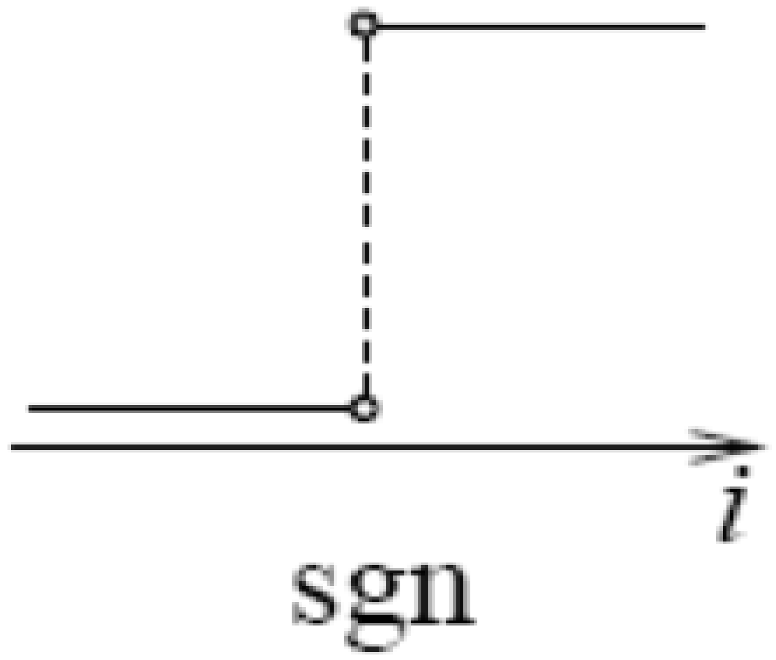


Single-layer Perceptron

- Simplest output function

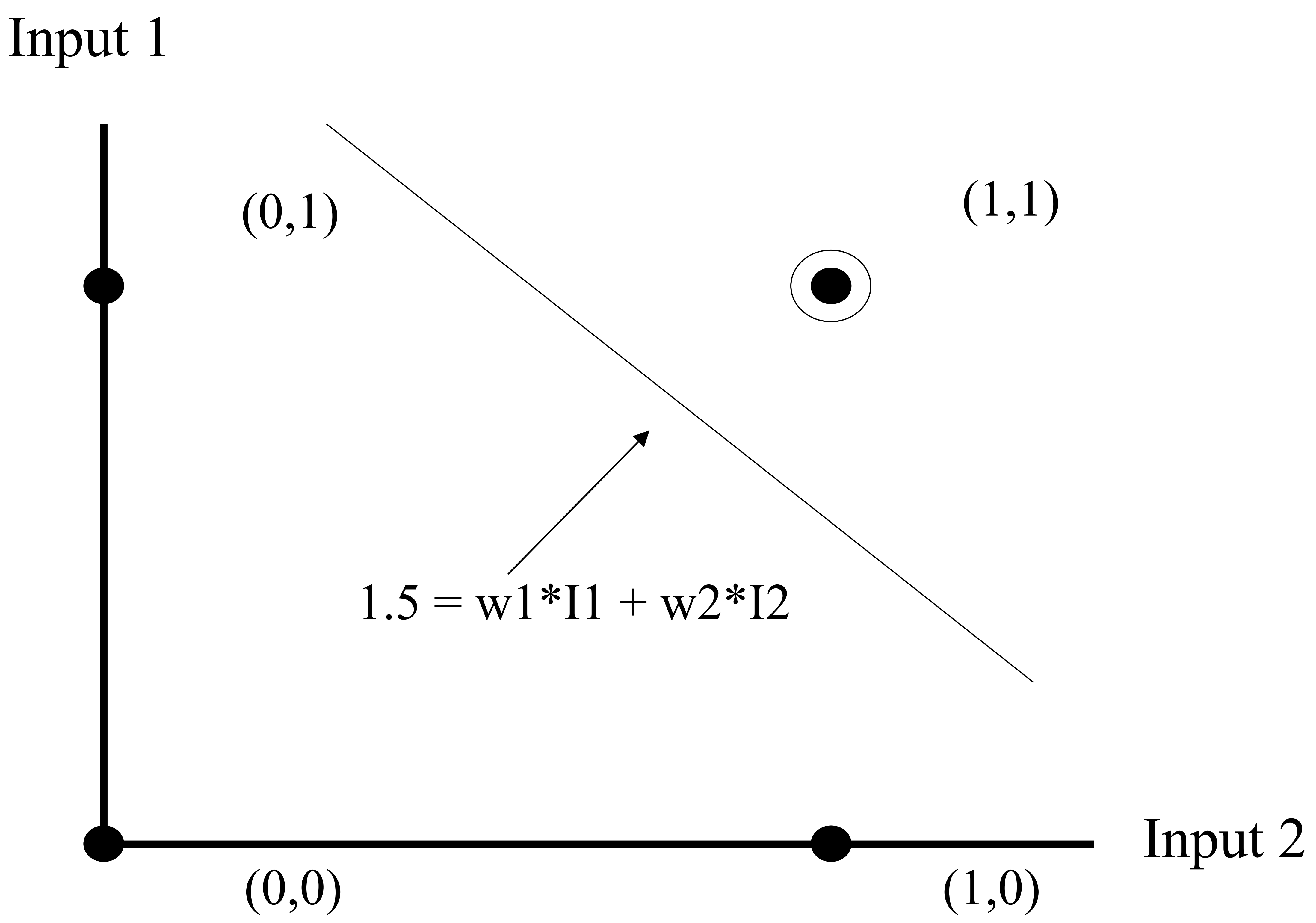
$$y = \operatorname{sgn}\left(\sum_{i=1}^{2} w_i x_i + \theta\right)$$

$$\operatorname{sgn}(s) = \begin{cases} 1 & \text{if } s > 0 \\ -1 & \text{otherwise.} \end{cases}$$



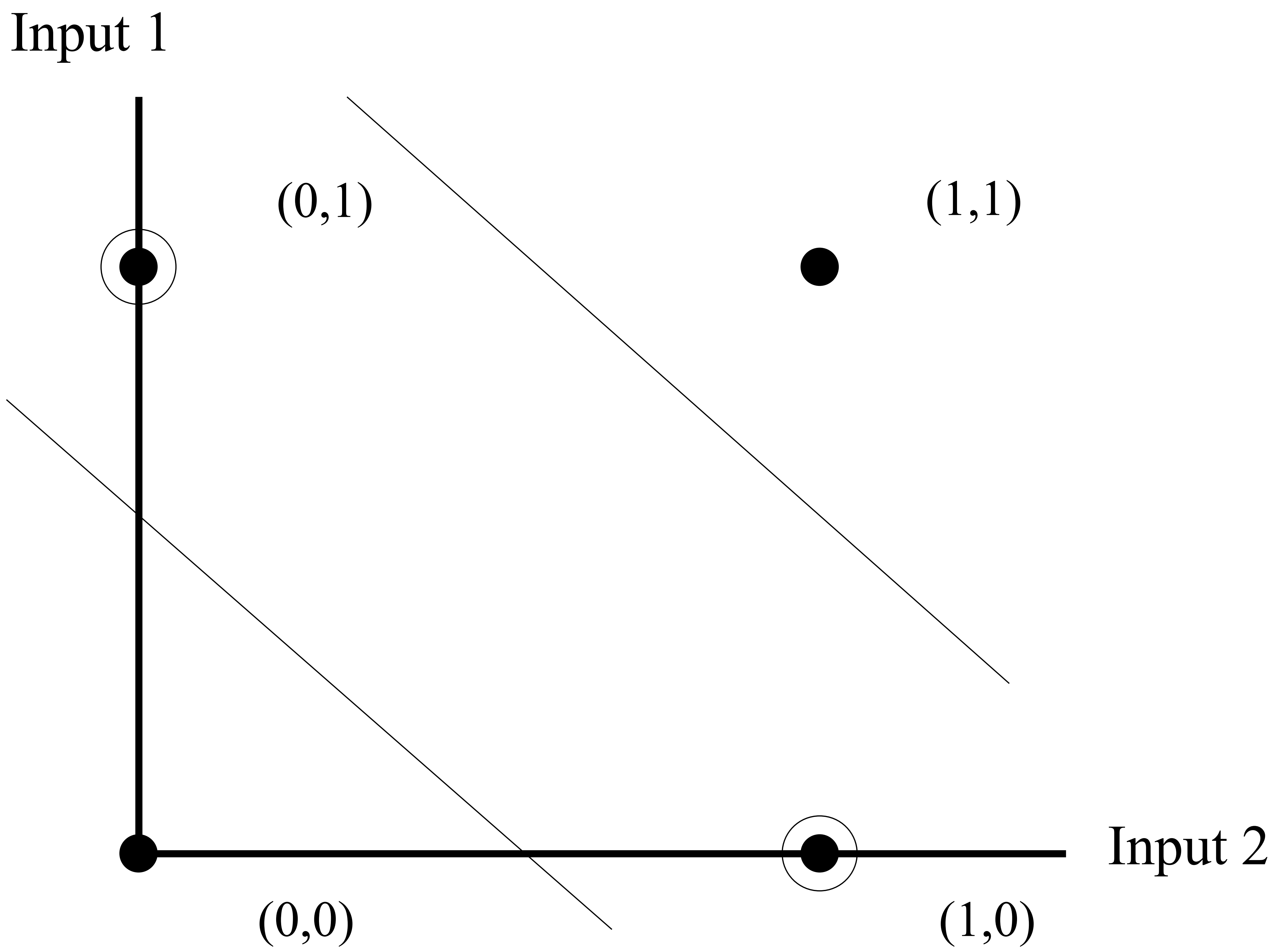
Single-layer Perceptron

- AND gate output



Single-layer Perceptron

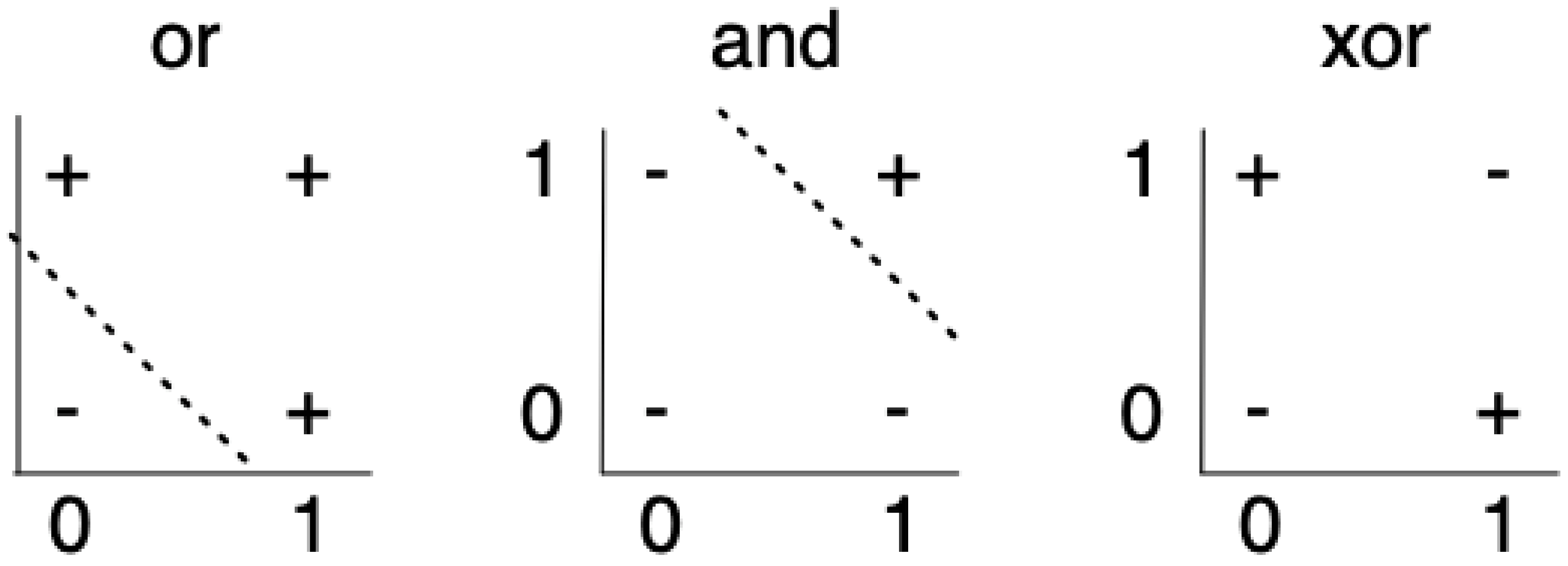
- XOR gate
- => hidden layer needed



Single-layer Perceptron

- **Linearly separable** if there is a hyperplane where classification is true on one side of the hyperplane and false on the other side
- For the sigmoid function, when the hyperplane is:

$$x_1 \cdot w_1 + \cdots + x_n \cdot w_n = 0$$



Single-layer Perceptron

- Learning rule

$\hat{y}(x) = g(X \cdot W^t) \rightarrow$ perceptron output
$y(x) \rightarrow$ target output $\in \{-1, 1\}$

$\Delta W_k = \alpha(y(x_k) - \hat{y}(x_k)) \cdot x_k$
$W_{k+1} = W_k + \Delta W_k$

$\Delta W_k = 0 \rightarrow$ if the perceptron output is correct
$\Delta W_k = +2\alpha x_k \rightarrow$ if $y(x_k) = 1$ and $\hat{y}(x_k) = -1$
$\Delta W_k = -2\alpha x_k \rightarrow$ if $y(x_k) = -1$ and $\hat{y}(x_k) = 1$

$\alpha > 0; \{x_1, ..., x_k\}$ input sequence

Single-layer Perceptron

- Perceptron convergence theorem:
  - For any data set that is linearly separable, the perceptron learning rule is guaranteed to find a solution in a finite number of iterations.

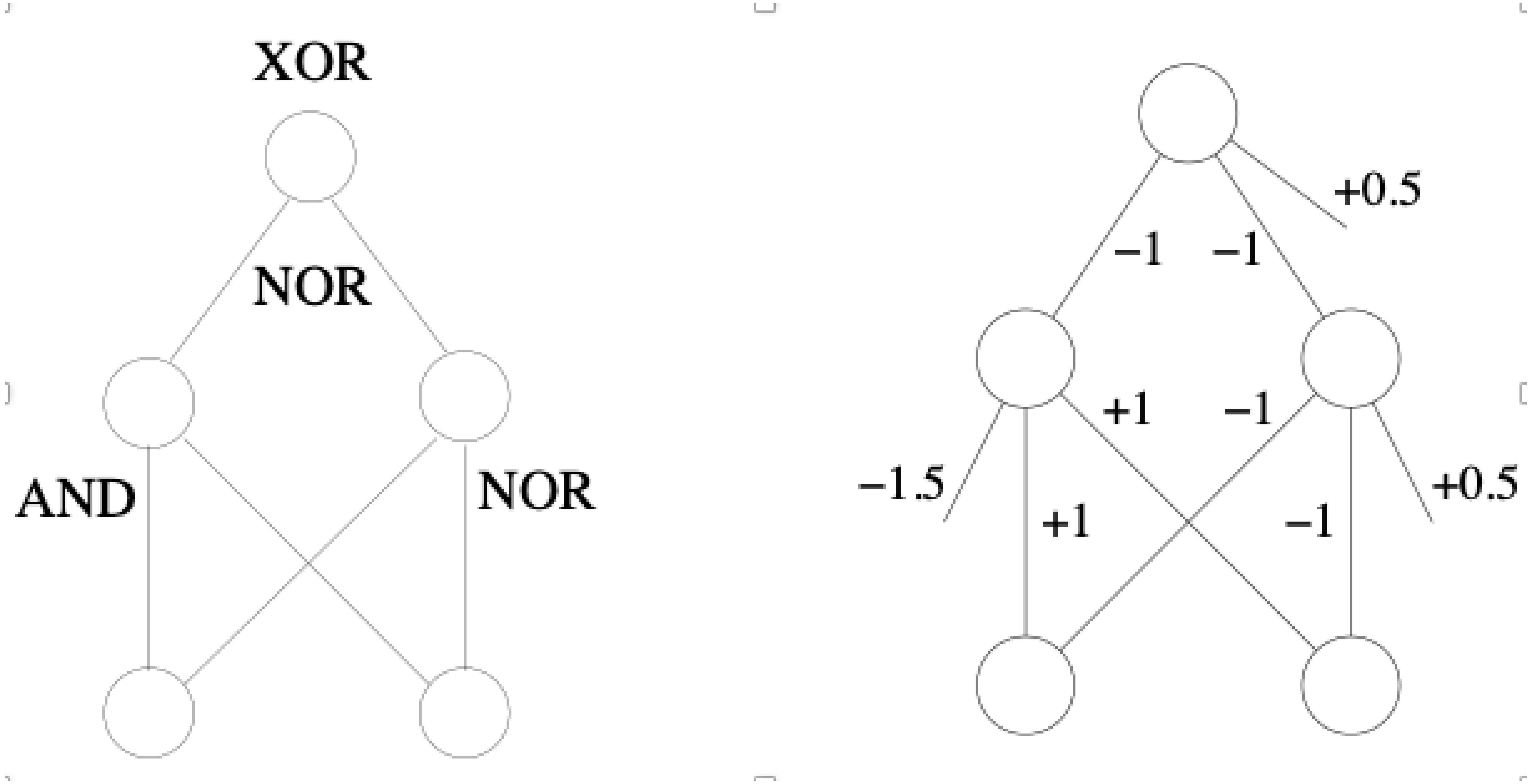
Historical Context

- In 1969, Minsky and Papert published a book highlighting limitations of perceptrons.
  - Funding agencies redirected funding away from neural network research preferring instead logic-based methods such as expert systems.
- Known since 1960s that any logical function could be implemented in a 2-layer neural network with step function activations.
- The problem was how to learn the weights of a multi-layer neural network from training examples.
- Solution found in 1974 by Paul Werbos.
  - Not widely known until rediscovered in 1986 by Rumelhart, Hinton and Williams.

Lecture Overview

- Motivation
- Biological and artificial neurons
- Single-layer perceptron
- Multi-layer perceptron
- Neural network design
- Neural network architectures

Multi-layer Neural Network



- Given an explicit logical function, we can design a multi-layer neural network by hand to compute that function.
- But, if we are just given a set of training data, can we train a multi-layer network to fit these data?

Multi-layer Perceptron

- **Definition:** A network in which its neurons are organised in successive layers. Each layer receives inputs from the previous one (or external input) and sends its outputs to the next layer. There are no internal connections in each layer.

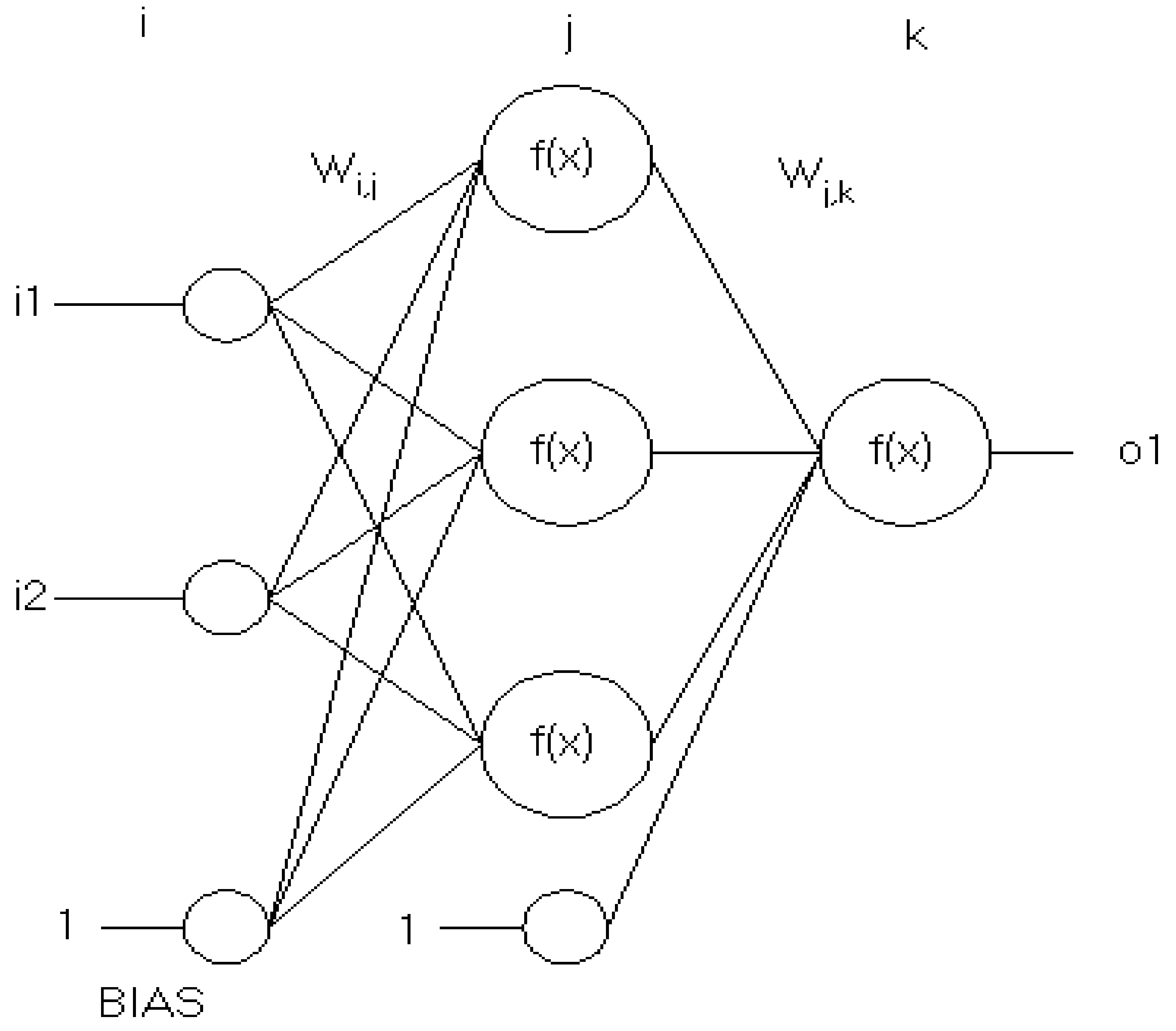
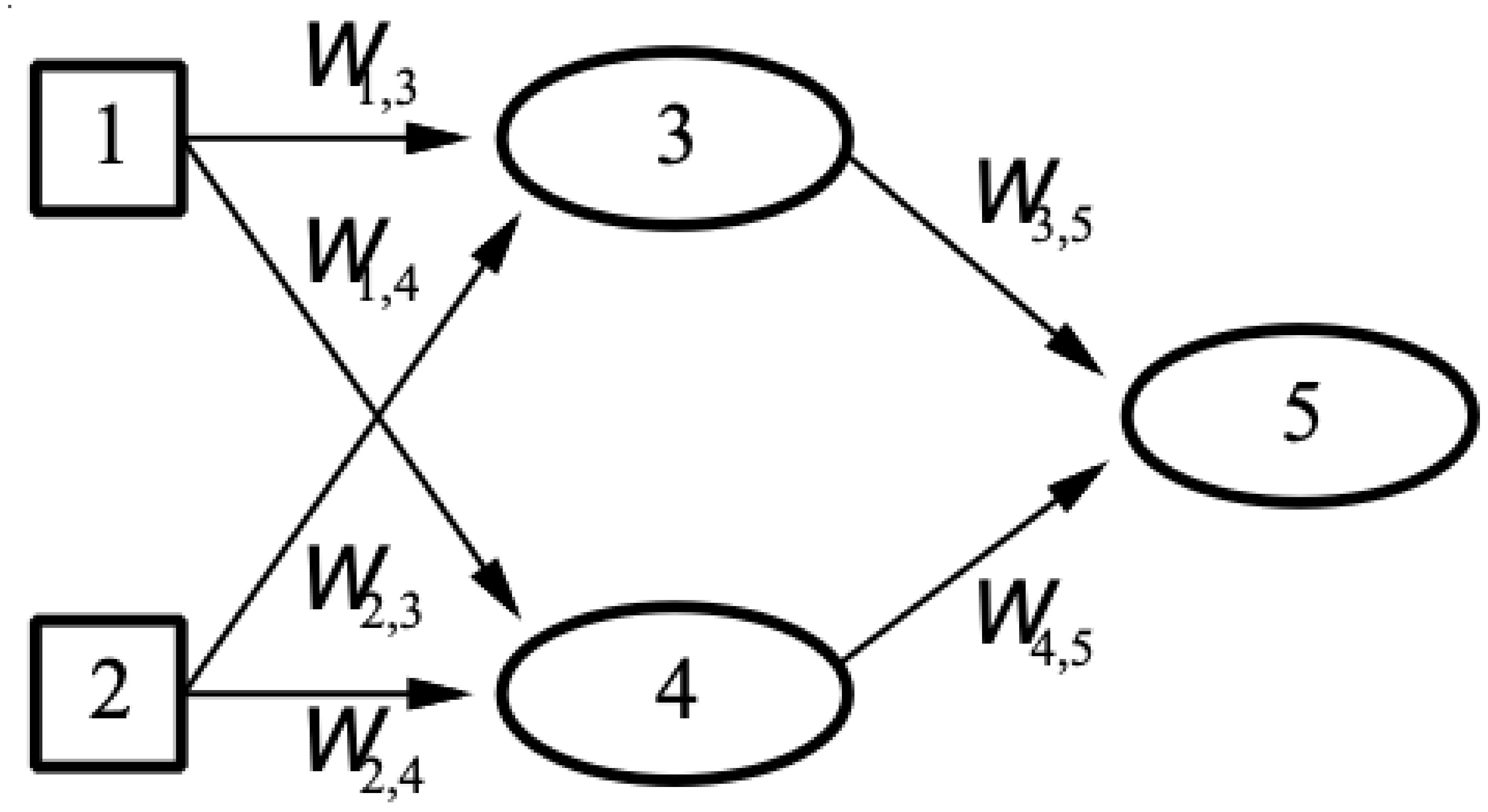


Figure - 1 A simple 2x3x1 NN. The lines connecting the neurons represent the weights. Also shown are the bias nodes used to shift the neuron transfer function and improves the network performance.

Feedforward Propagation



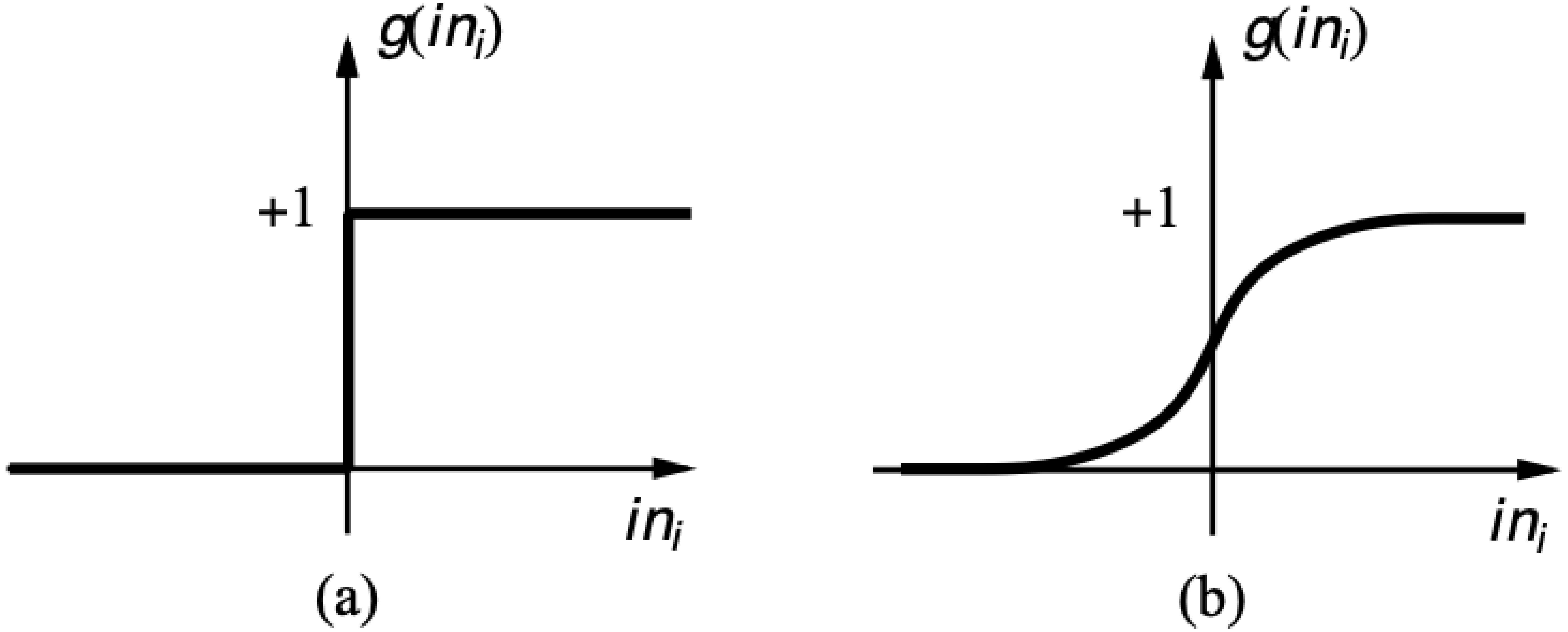
$w_{i,j} \equiv$ *weight between node i and node j*

- Feed-forward network = a parameterised family of nonlinear functions:

$$a_5 = g\left(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4\right)$$
$$= g\left(W_{3,5} \cdot g\left(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2\right) + W_{4,5} \cdot g\left(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2\right)\right)$$

- Adjusting weights changes the function

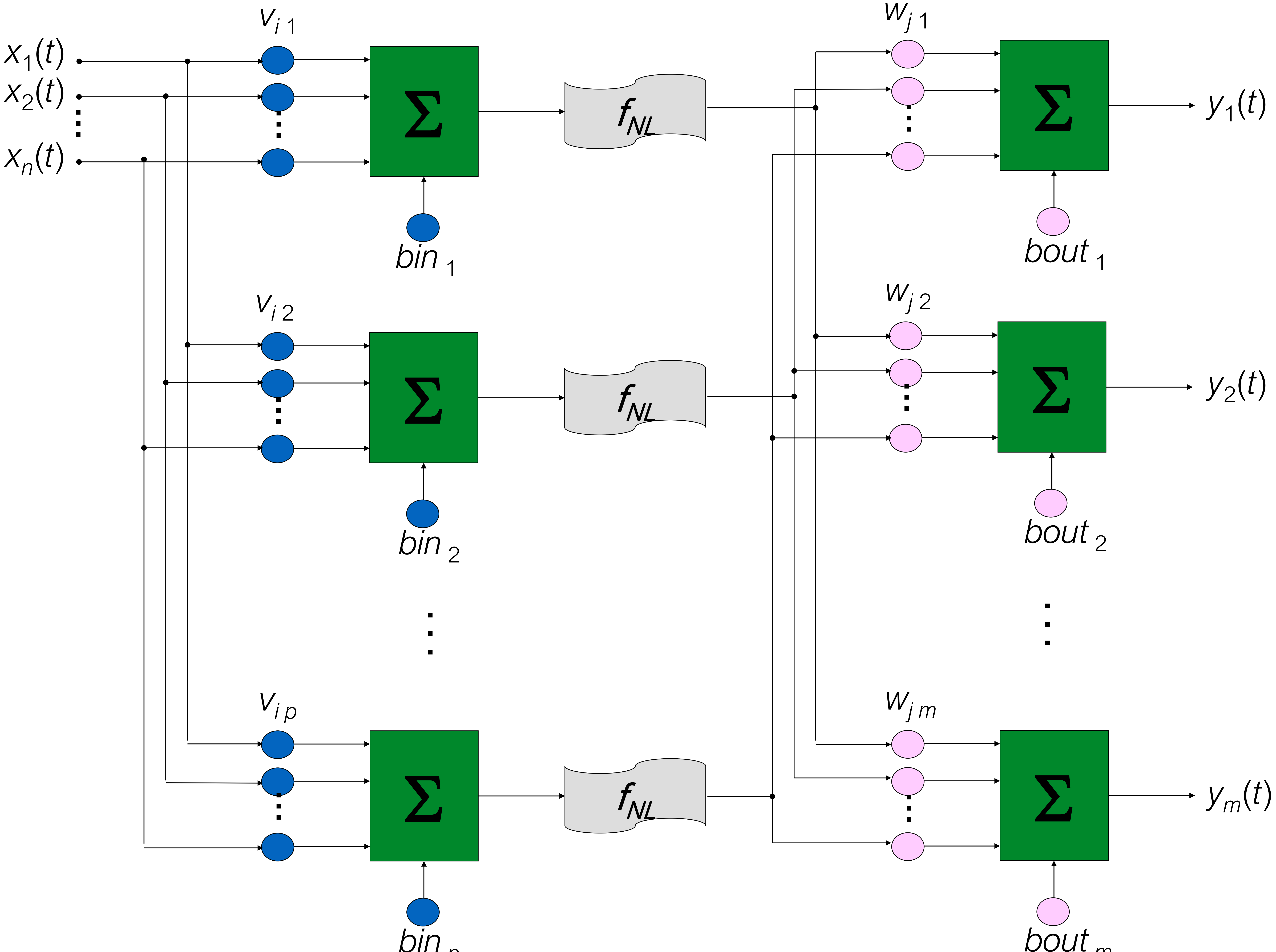
Feedforward Propagation



(a) is a step function or threshold function

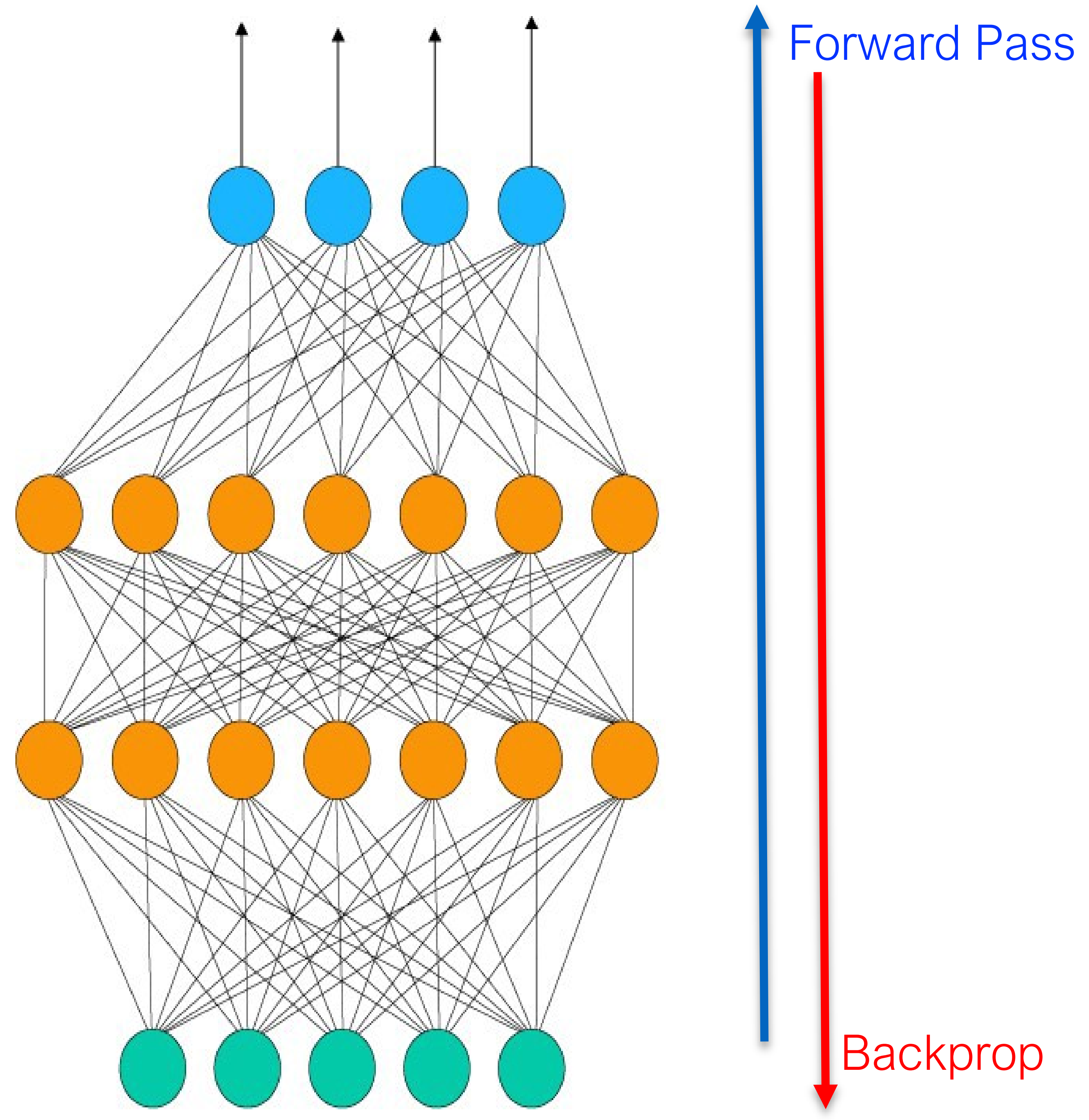
(b) is a sigmoid function $\frac{1}{1+e^{-x}}$

Changing the bias weight b moves the threshold

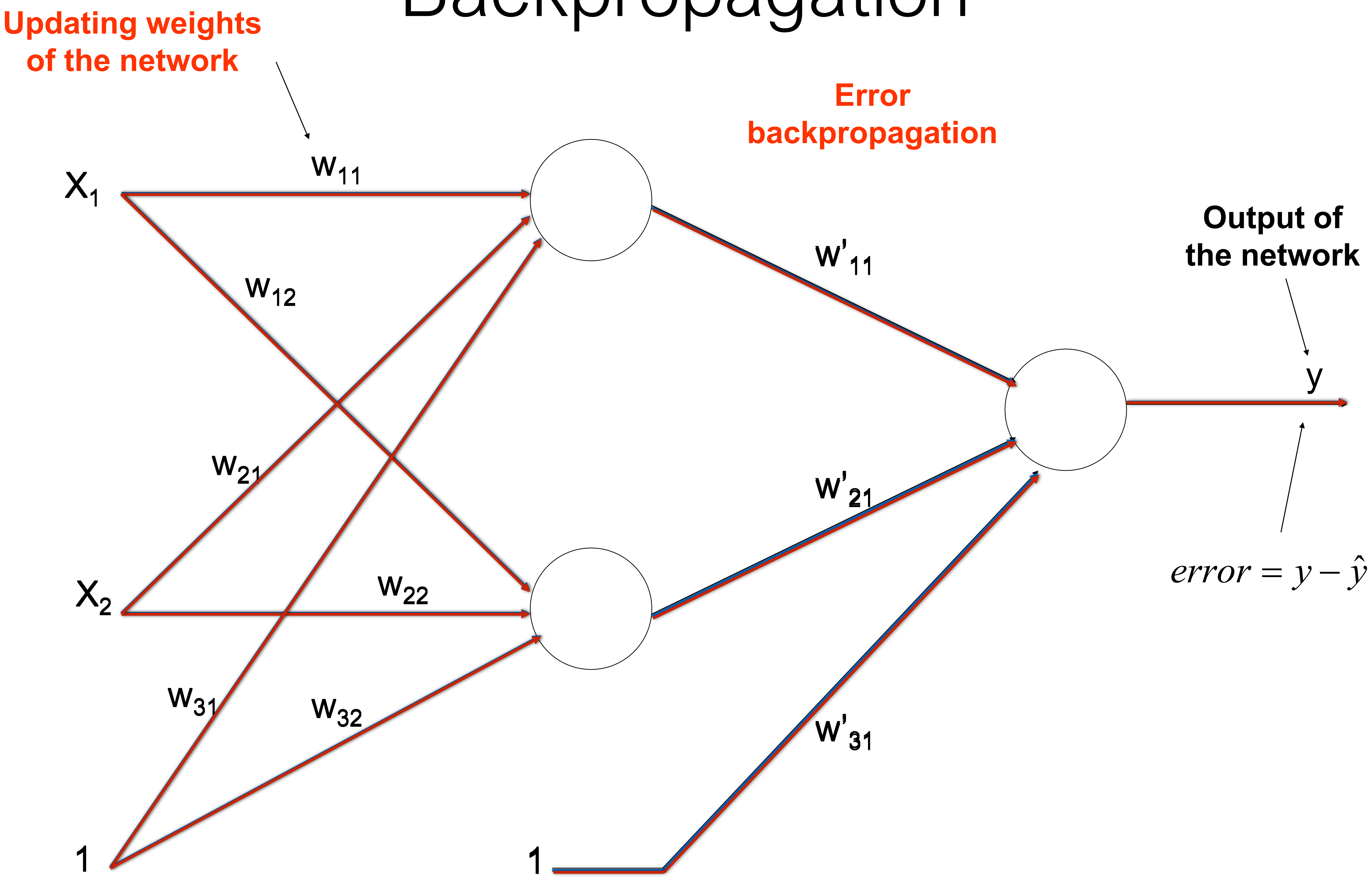

$x_1(t)$
$x_2(t)$
$x_n(t)$
$v_{i\,1}$
$v_{i\,2}$
$v_{i\,p}$
Σ
Σ
Σ
$bin_{\,1}$
$bin_{\,2}$
$bin_{\,p}$
f_{NL}
f_{NL}
f_{NL}
$w_{j\,1}$
$w_{j\,2}$
$w_{j\,m}$
Σ
Σ
Σ
$bout_{\,1}$
$bout_{\,2}$
$bout_{\,m}$
$y_1(t)$
$y_2(t)$
$y_m(t)$

Backpropagation

1. **Forward pass**: apply inputs to the "lowest layer" and feed activations forward to get output.
2. **Calculate error**: difference between desired output and actual output.
3. **Backward pass**: Propagate errors back through the network to adjust weights.



Backpropagation



Backpropagation

Gradient descent

$$E = \frac{1}{2N}\sum(d - y)^2$$

If transfer functions are smooth, can use multivariate calculus to adjust weights by taking the steepest downhill direction.

$$w \leftarrow w + \alpha\frac{\partial E}{\partial w}$$

Parameter α is the **learning rate**

- How the cost function affects the particular weight

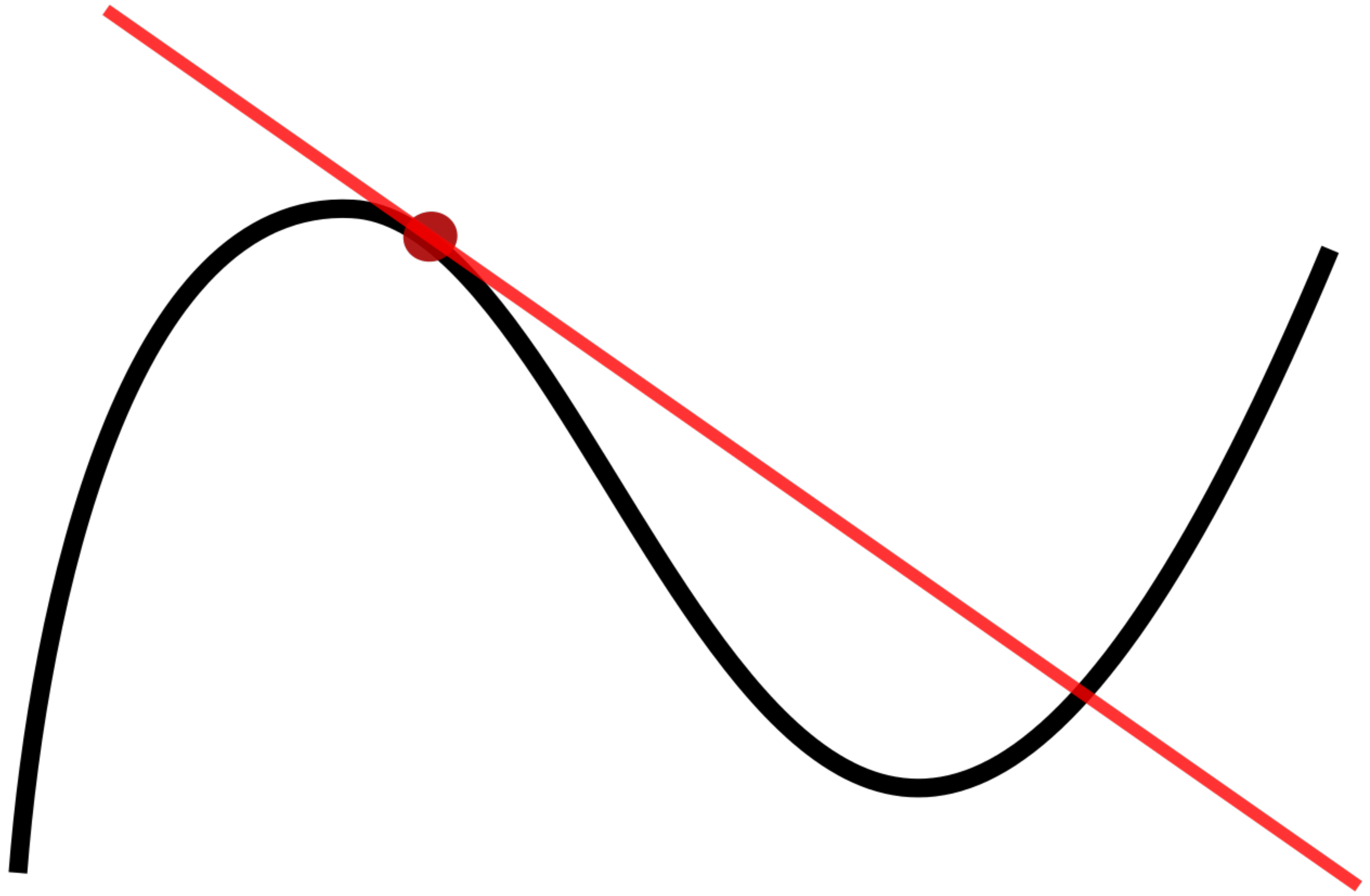
Backpropagation

The derivative of a function is the slope of the tangent at a point

$$y = f(x) = mx + b$$

$$m = \frac{change\ in\ y}{change\ in\ x} = \frac{\Delta y}{\Delta x}$$

Written $\frac{dy}{dx}$



Backpropagation

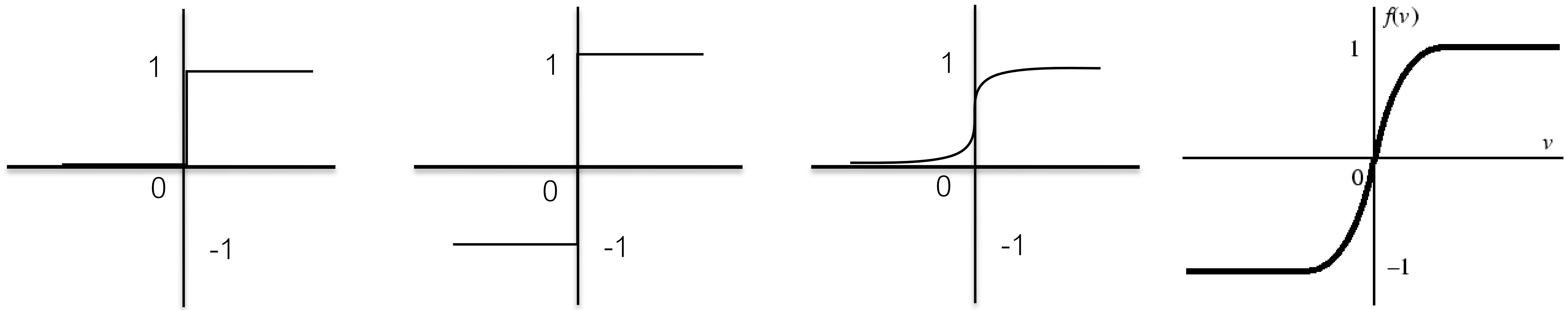
Partial derivative

Derivative of a function of several variables with respect to one of these variables

If $z = f(x, y, \dots)$

Derivative with respect to x is written: $\frac{\partial z}{\partial x}$

Backpropagation



Function must be continuous to be differentiable

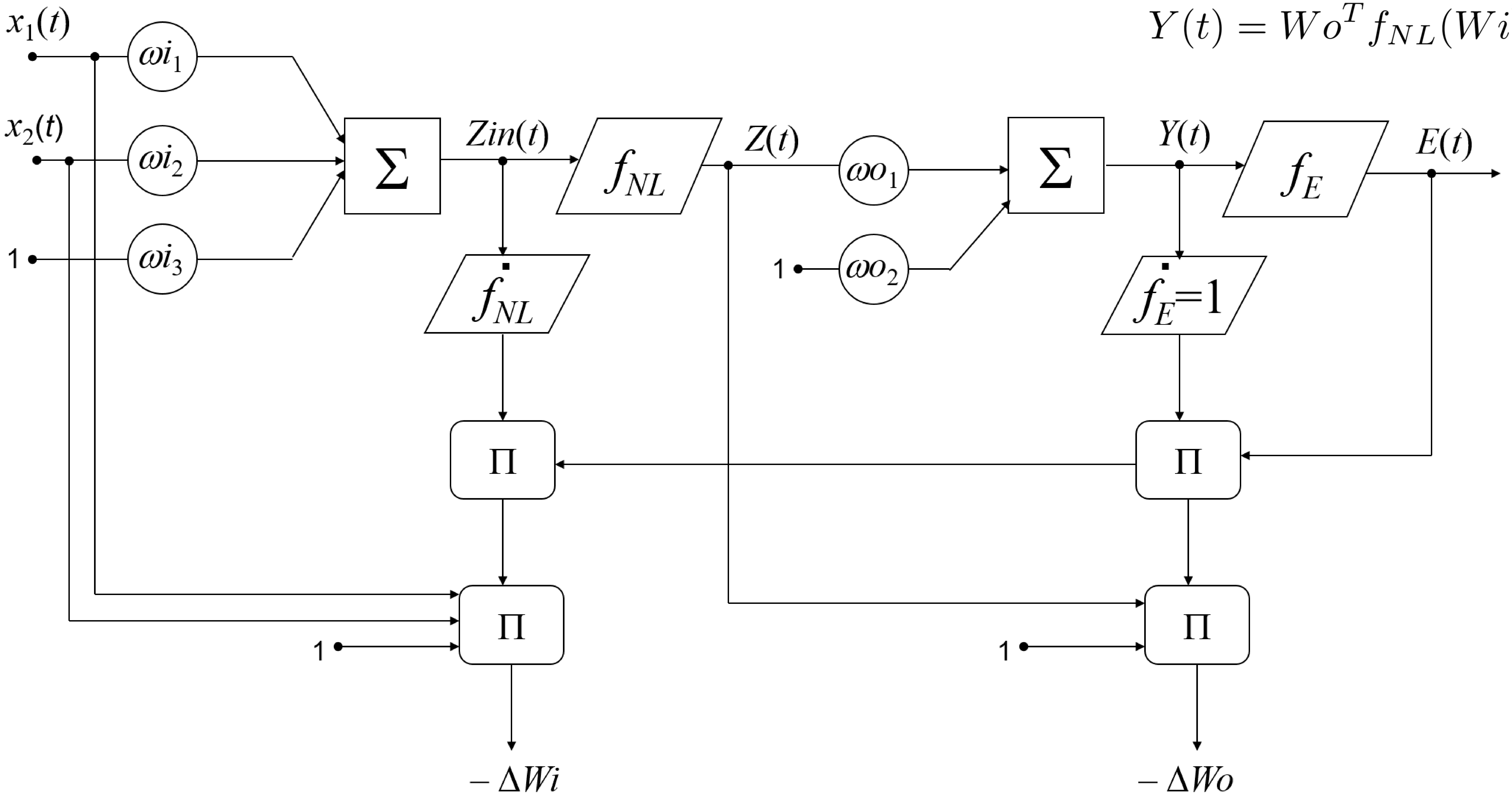
Replace the (discontinuous) step function with a differentiable function, such as the sigmoid:

$$g(s) = \frac{1}{1 + e^{-s}}$$

or hyperbolic tangent

$$g(s) = \tanh(s) = \frac{e^{s} - e^{-s}}{e^{s} + e^{-s}} = 2\left(\frac{1}{1 + e^{-2s}}\right) - 1$$ (-1 to 1)

$$Y(t) = Wo^T f_{NL}(Wi^T X(t))$$



$$\Delta Wo = -\alpha * \frac{2}{s} * E(t) * Z^T(t)$$

$$\Delta Wi^T = -\alpha * \frac{2}{s} * E^T(t) * Wo_j * \frac{\partial f_{NL}}{\partial Zin} * X^T(t)$$

Lecture Overview

- Motivation
- Biological and artificial neurons
- Single-layer perceptron
- Multi-layer perceptron
- Neural network design
- Neural network architectures

Step 1: Exhaustive Analysis of the System

- It should determine the number and type of input variables and model output, reducing the number of variables.

- Is it really necessary to use a neural model? Why not use any other existing classic model (e.g., phenomenological)?

  - Neural Network: Best second solution.

- If a neural model is used, do we have available data representing properly the system to be modelled? Do we have enough?

Step 2: Preprocessing

- Data: a neural network is a black-box model (a.k.a. empirical model) for interpolation (never extrapolation); therefore, they greatly depend on quality and quantity of data available.

- Quality: related to the degree to which the available data represents the function being approximated. Ideal: to obtain them by following a properly designed survey/experimental plan.

- Quantity: It is extremely important because only an adequate amount of data will allow us to correctly identify the parameters (weights) of our neural model.

  - If the quantity of data is small, we cannot expect to develop a complex neural model.

Step 2: Preprocessing

- Visual examination of the data.
  - Detect and, if possible, eliminate outliers, empty values, etc.
  - It might also help to detect correlations between variables.
- Normalization of variables: It is necessary when variables with different units and, therefore, potentially different magnitudes are involved. Sometimes, the magnitudes can differ by several orders of magnitude.
  - $X_n = (X-X_{min})/(X_{max}-X_{min})$; $X_n \in [0,1]$
  - $X_n = 2*(X-X_{min})/(X_{max}-X_{min}) - 1$; $X_n \in [-1,1]$
- It is necessary to perform the corresponding denormalization at the output stage.

Step 3: Design of the Neural Model

- Input and output neurons depend on the previous analysis of the system.

- But, what about the number of neurons Nh in the hidden layer?

- Rule of thumb: Nh should lead to a number of parameters (weights) Nw that:

  - $Nw < (\text{Number of samples}) / 10$

- The number of weights Nw of an MLP, with Ni neurons in its input layer, a hidden layer with Nh neurons, and No neurons in the output layer is:

  - $Nw = (Ni+1)*Nh+(Nh+1)*No$

Step 3: Design of the Neural Model

- An MLP with 3 inputs, 4 units in its hidden layer, and 2 outputs, has a number of parameters:
  - Nw = (3+1)*4+(4+1)*2 = 26
- Then, at least 260 samples are required to train the network weights.

Step 3: Design of the Neural Model

- In MLPs is demonstrated that using one hidden layer with a proper number of neurons is sufficient to approximate any non-linear function with an arbitrary precision degree.

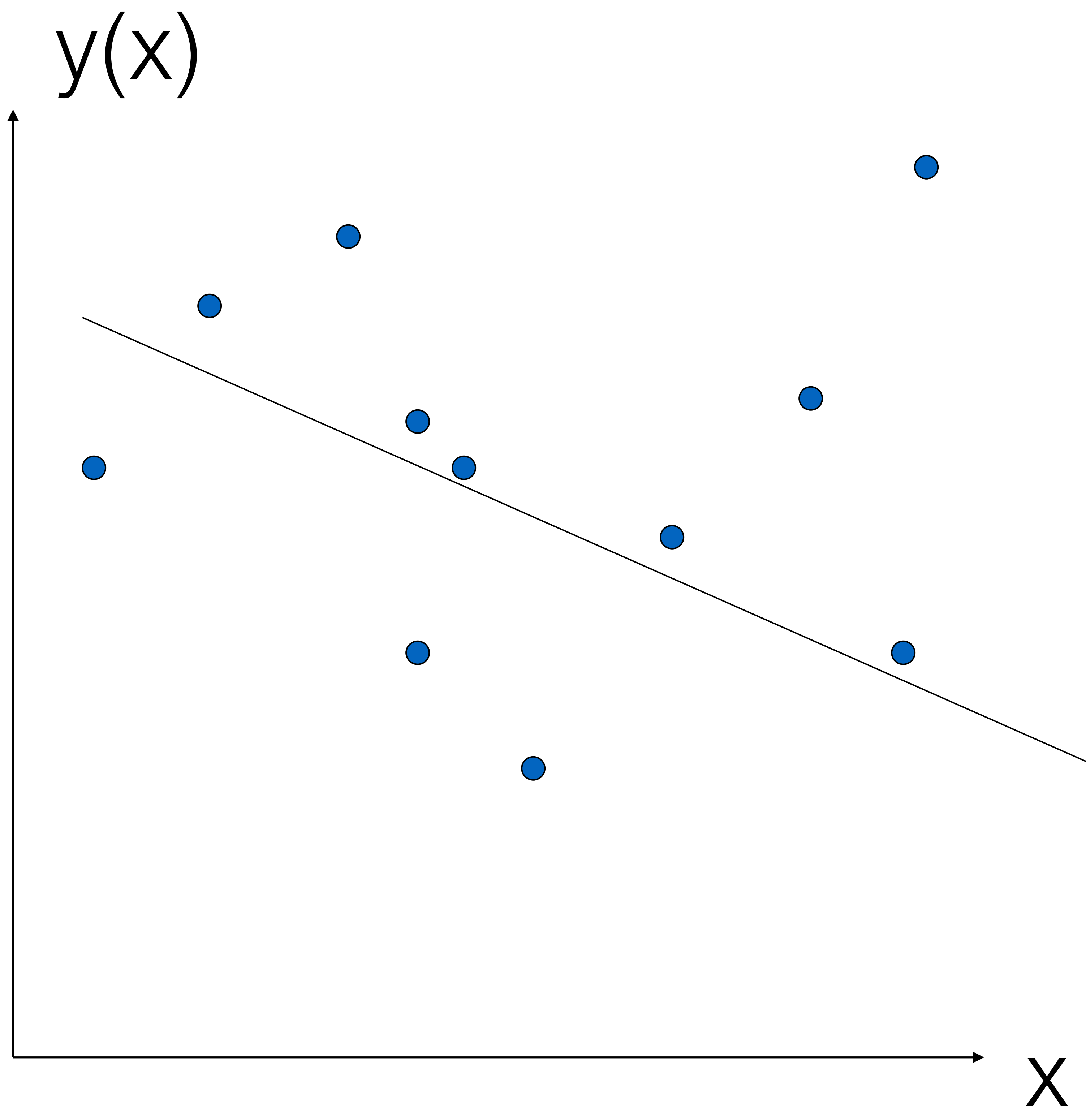
- Activation functions: A usual criterion is to use sigmoid functions or ReLUs in the hidden layer and linear functions in the output. However, sigmoids or softmax can also be used in the output.

Step 4: Training

- Training a neural network is a hard process due to the complexity of the error function solution space, which can have numerous local minima, saddle (minimax) points, etc.
- There are three main problems that can arise during training:
  - Bias
  - Overparameterization
  - Overfitting
- The latter two might affect the network's ability to generalize (high variance).

Step 4: Training

Training bias

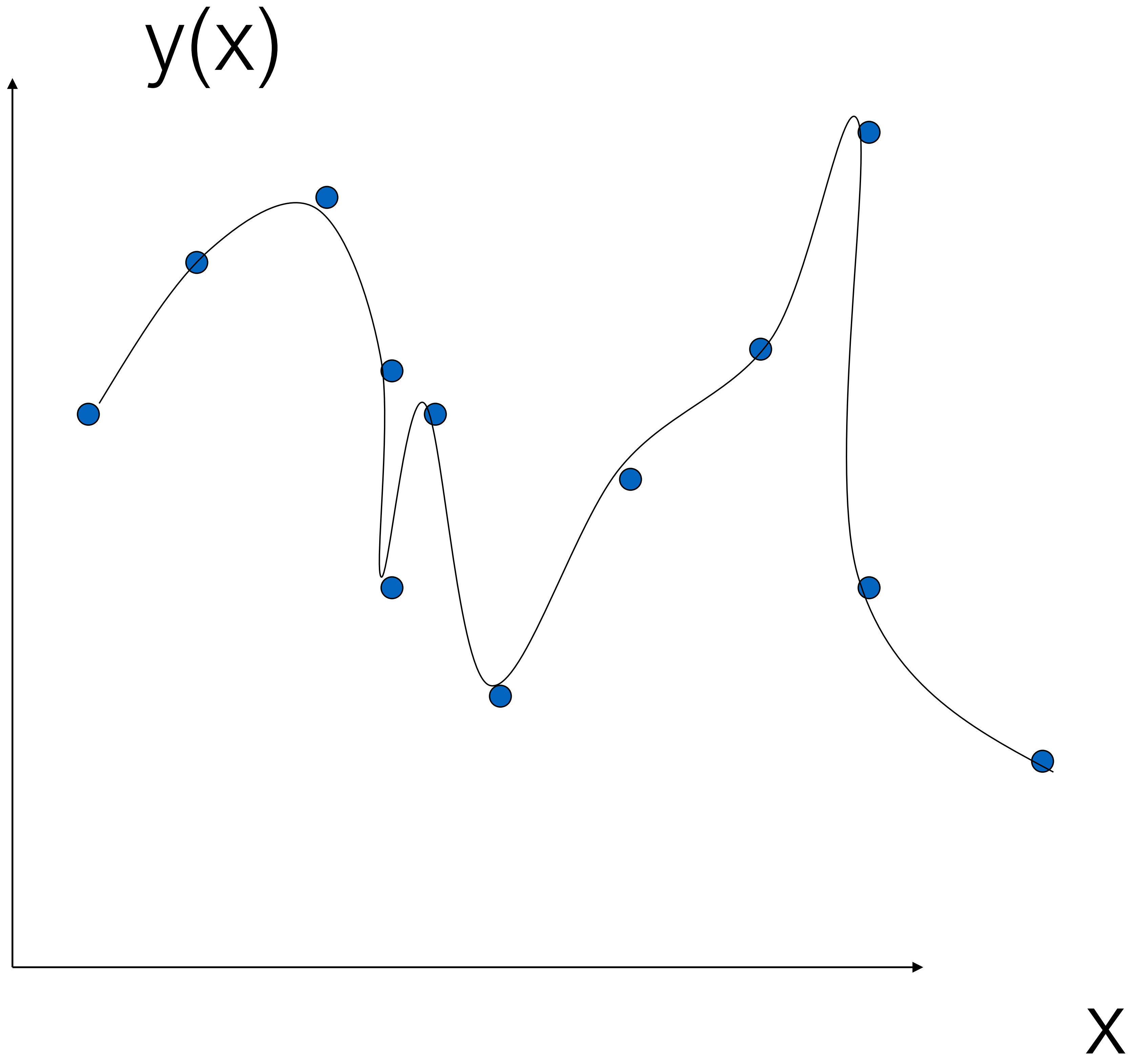


Step 4: Training

- To decrease bias:
  - Increase (prudently) the number of neurons in the hidden layer.
  - Aim to reach a better local minimum by conducting a sufficient number of different training processes, starting from randomly chosen initial weights (20 or more attempts).

Step 4: Training

High variance problem (overparameterization and overfitting)

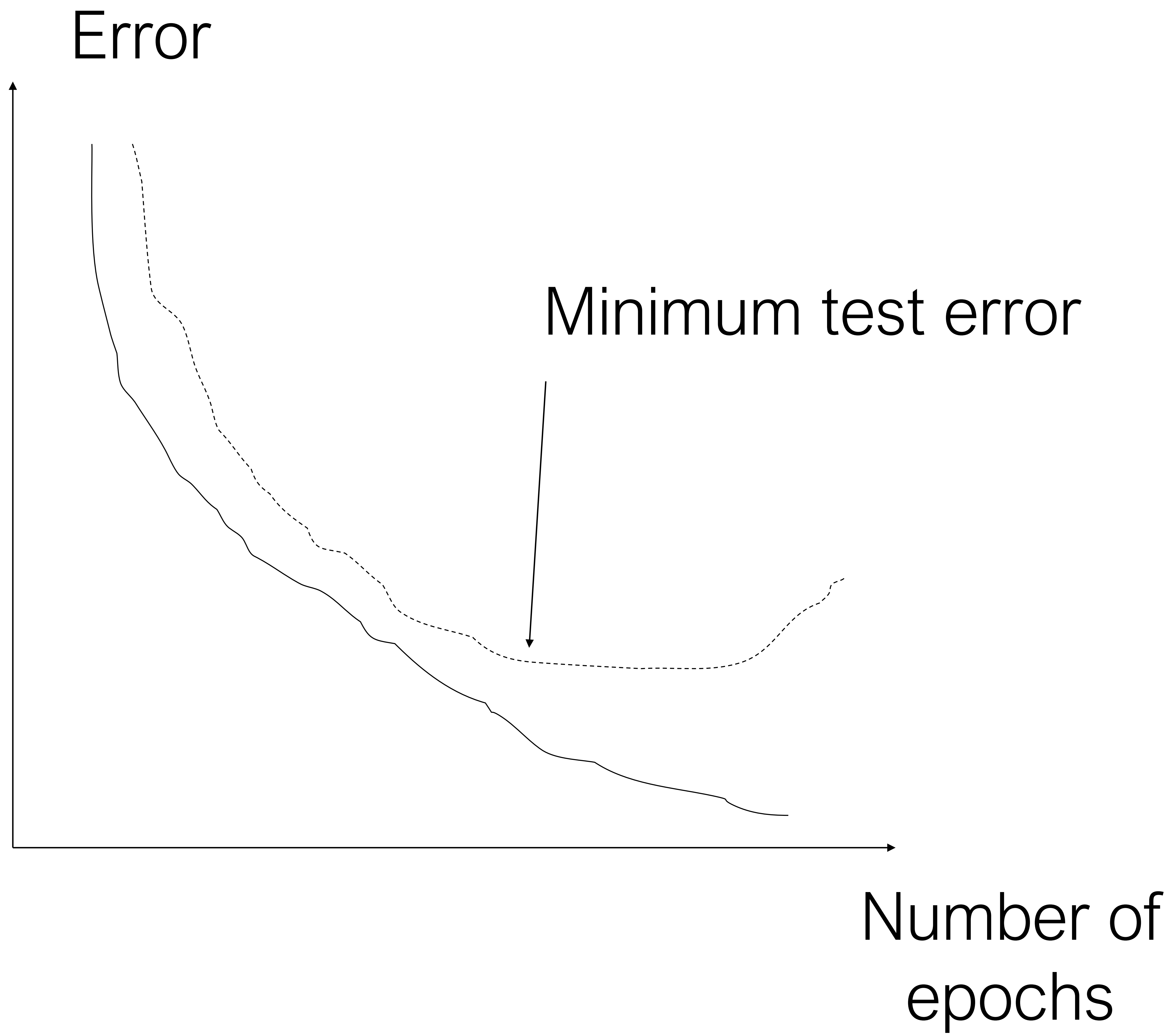


Step 4: Training

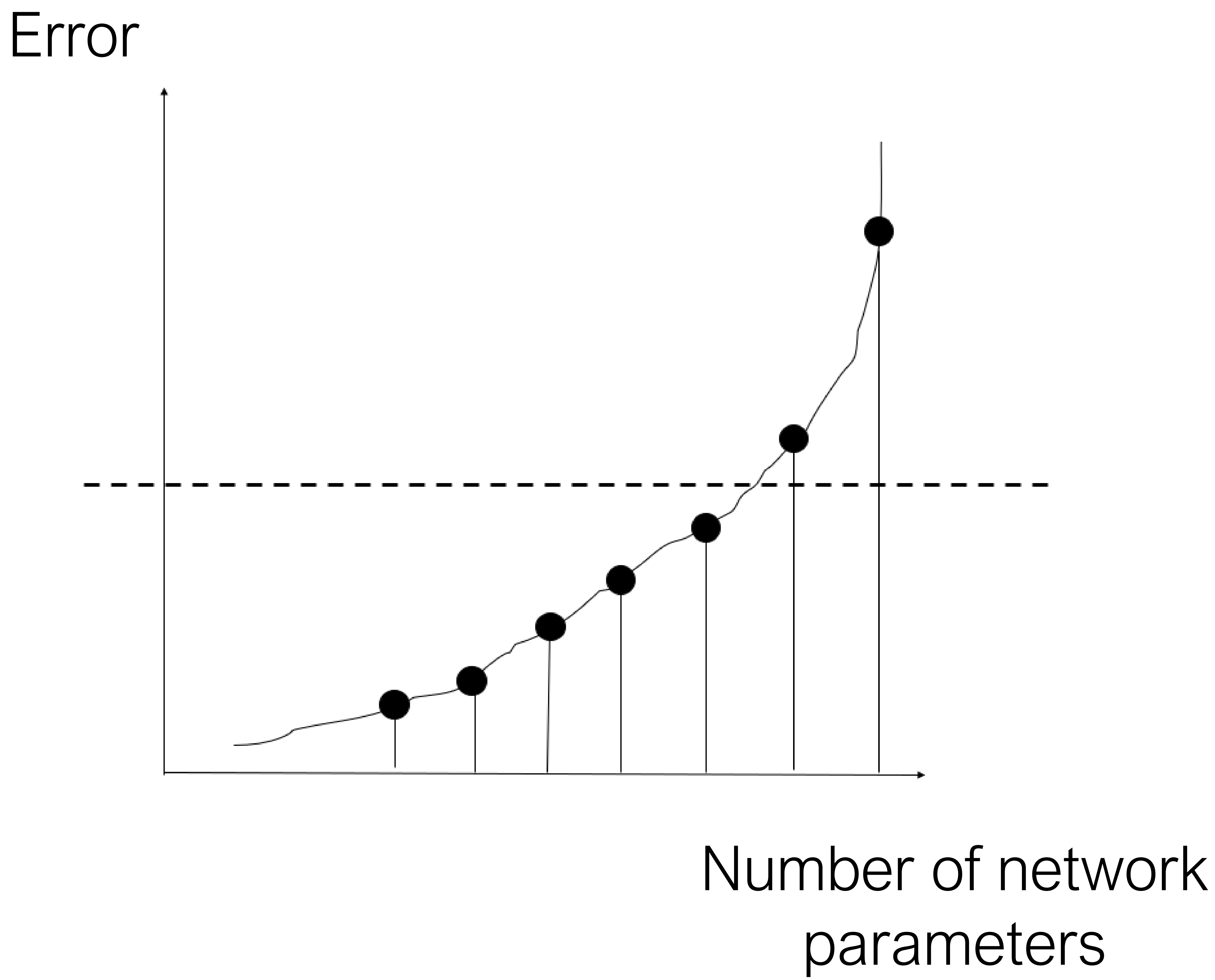
- To avoid overfitting problem, work with two sets during training:
    - Training set
    - Test set
- The best is to visualize the error function simultaneously on both sets.
- Characteristics of the training and test sets:
    - Both sets should be large enough, and data should be representative on both sets.

Step 4: Training

Error function for training set (-) and test set (---)



Step 4: Training



Step 4: Training

- Cross-validation: Different neural network models are developed using the available data, splitting the training and test sets in different ways. The model that achieves the minimum error on the test set is chosen.

- Additional training aspects:

  - Weight initialisation.
  - Online or batch learning.
  - Adjust the parameters, e.g., learning rate and epochs to suit the particular task.

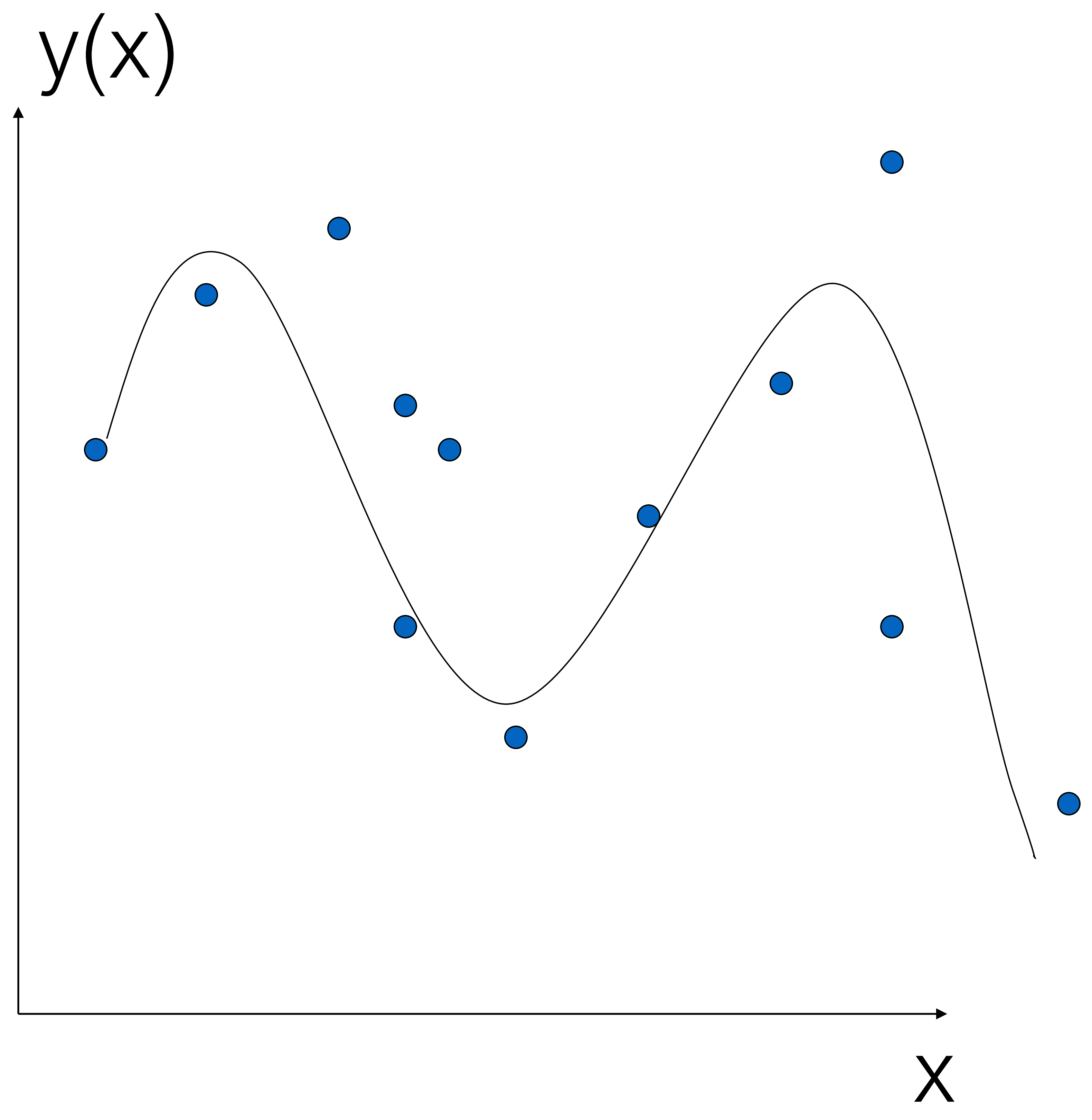
Step 5: Generalization

- To test the generalization capability of the network, that is, its performance on a different (never seen) set of data, a small (but representative) third set might be reserved, the generalization set.

- This set should also be representative of the phenomenon being modelled as the previous sets (training and test).

Step 5: Generalization

Approximation of the underlying function



Lecture Overview

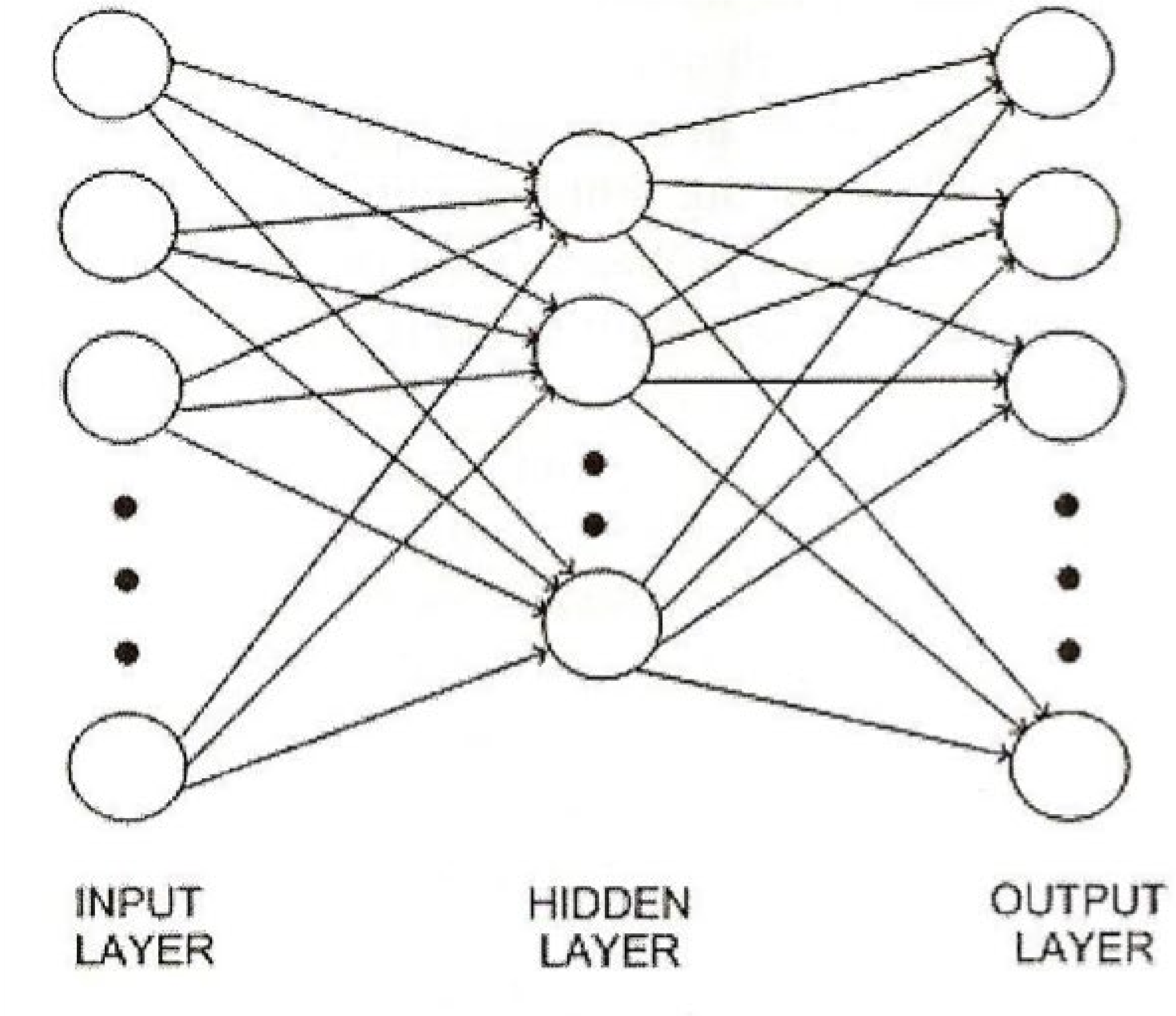
- Motivation
- Biological and artificial neurons
- Single-layer perceptron
- Multi-layer perceptron
- Neural network design
- Neural network architectures

Neural Network Architectures

- Two main network structures

1. Feed-Forward Network

2. Recurrent Network



Neural Network Architectures

Feed-forward network has connections only in one direction:

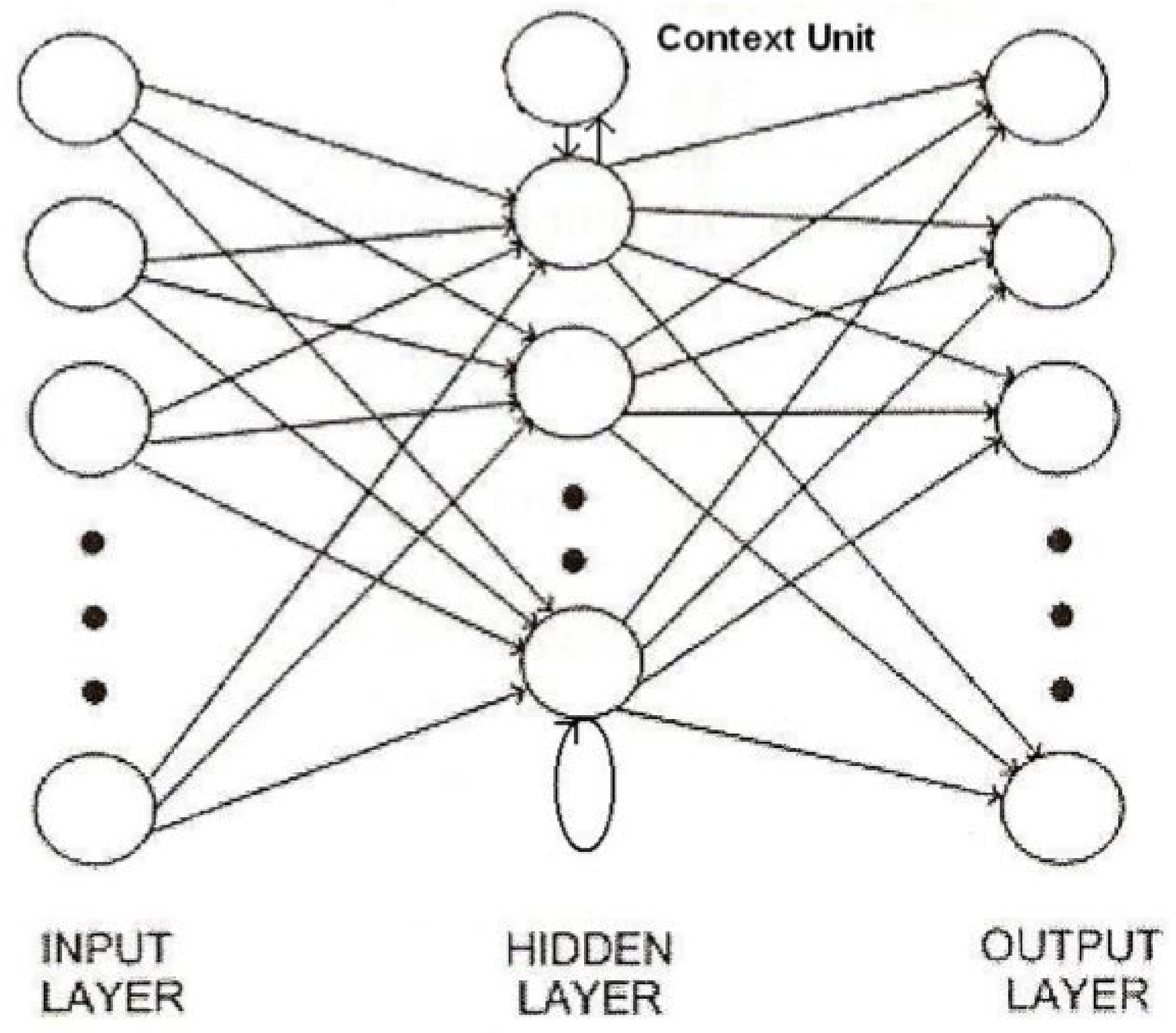
- Every node receives input from “upstream” nodes; delivers output to “downstream” nodes.
    - No loops.
- Represents a function of its current input.
    - It has no internal state other than the weights themselves.

Neural Network Architectures

- Two main network structures

1. Feed-Forward Network

2. Recurrent Network



Neural Network Architectures

Recurrent network feeds outputs back into its own inputs:

- Activation levels of network form a dynamical system.
  - It may reach a stable state or exhibit oscillations or even chaotic behaviour.
- Response of network to an input depends on its initial state.
  - This may depend on previous inputs.
- Can support short-term memory.

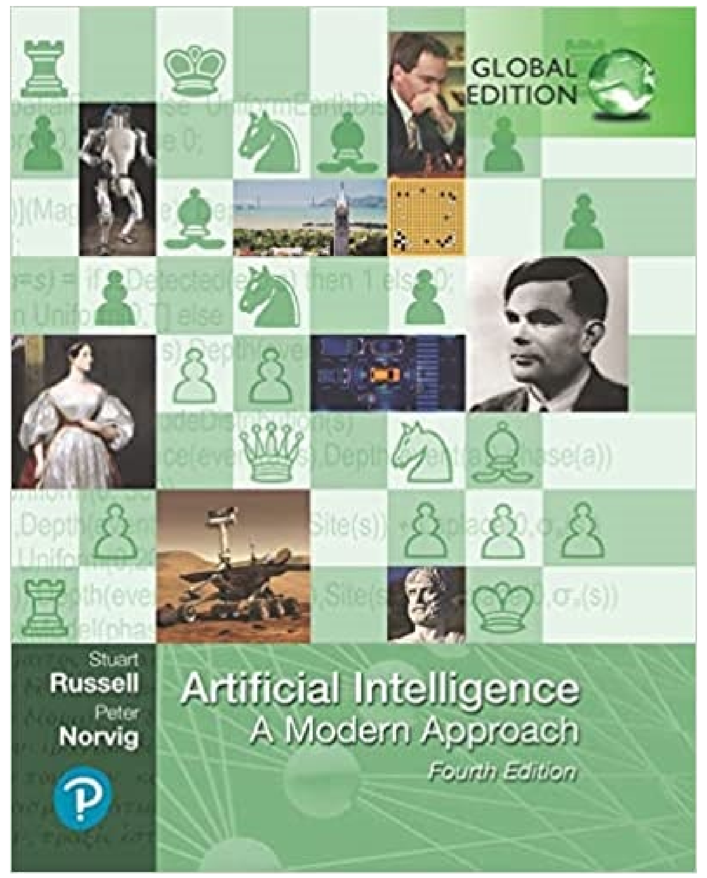
Deep Learning Architectures

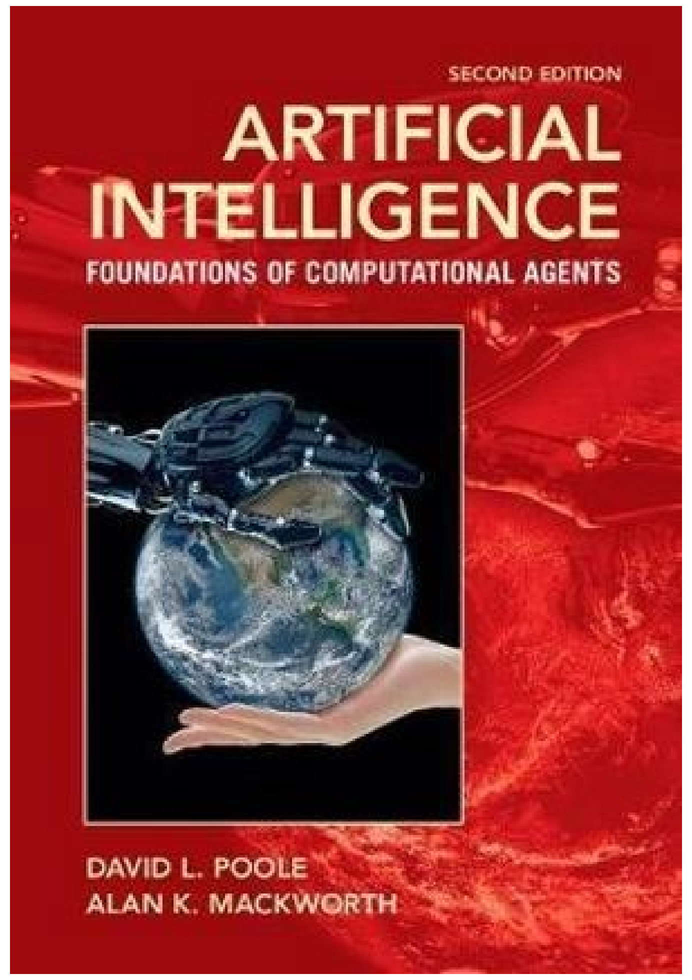
- Multiple layers form a hierarchical model, known as **deep learning.**
  - **Convolutional neural networks** are specialised for vision tasks.
  - **Recurrent neural networks** are used for time series.

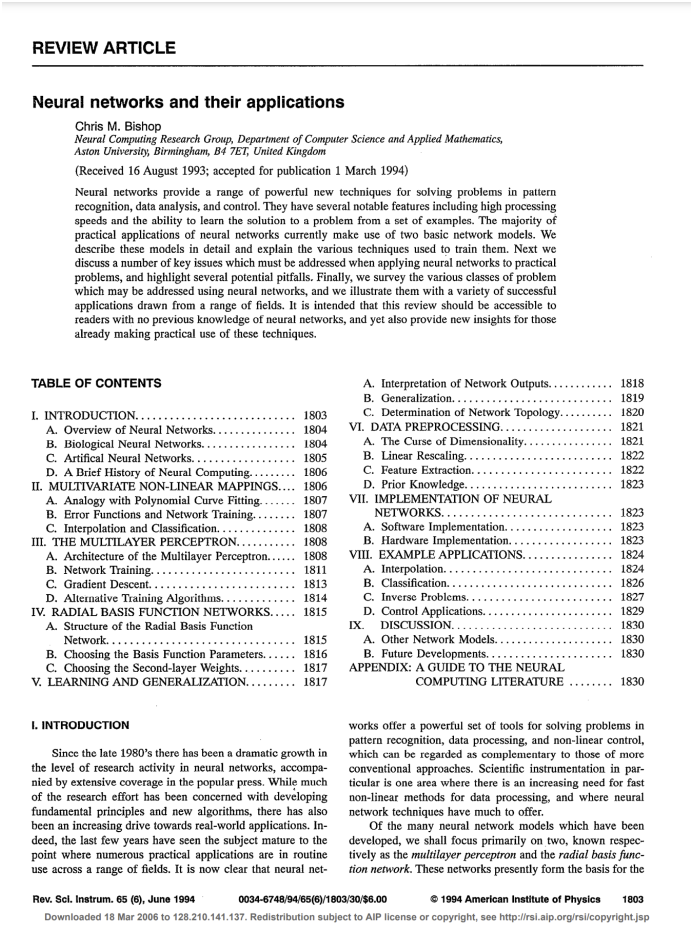
- Typical real-world network can have 10 to 20 layers with hundreds of millions of weights:
  - It can take hours, days, or months to learn on machines with thousands of cores.

References

- Poole & Mackworth, Artificial Intelligence: Foundations of Computational Agents, Chapter 7.
- Russell & Norvig, Artificial Intelligence: a Modern Approach, Chapters 18.6, 18.7
.
- Bishop, Neural Networks and Their Applications, Review of Scientific Instruments, 65(6): 1803-1832.







REVIEW ARTICLE

Neural networks and their applications

Chris M. Bishop
Neural Computing Research Group, Department of Computer Science and Applied Mathematics, Aston University, Birmingham, B4 7ET, United Kingdom

(Received 16 August 1993; accepted for publication 1 March 1994)

Neural networks provide a range of powerful new techniques for solving problems in pattern recognition, data analysis, and control. They have several notable features including high processing speeds and the ability to learn the solution to a problem from a set of examples. The majority of practical applications of neural networks currently make use of two basic network models. We describe these models in detail and explain the various techniques used to train them. Next we discuss a number of key issues which must be addressed when applying neural networks to practical problems, and highlight several potential pitfalls. Finally, we survey the various classes of problem which may be addressed using neural networks, and we illustrate them with a variety of successful applications drawn from a range of fields. It is intended that this review should be accessible to readers with no previous knowledge of neural networks, and yet also provide new insights for those already making practical use of these techniques.

TABLE OF CONTENTS

I. INTRODUCTION

Since the late 1980's there has been a dramatic growth in the level of research activity in neural networks, accompanied by extensive coverage in the popular press. While much of the research effort has been concerned with developing fundamental principles and new algorithms, there has also been an increasing drive towards real-world applications. Indeed, the last few years have seen the subject mature to the point where numerous practical applications are in routine use across a range of fields. It is now clear that neural networks offer a powerful set of tools for solving problems in pattern recognition, data processing, and non-linear control, which can be regarded as complementary to those of more conventional approaches. Scientific instrumentation in particular is one area where there is an increasing need for fast non-linear methods for data processing, and where neural network techniques have much to offer.

Of the many neural network models which have been developed, we shall focus primarily on two, known respectively as the *multilayer perceptron* and the *radial basis function network*. These networks presently form the basis for the

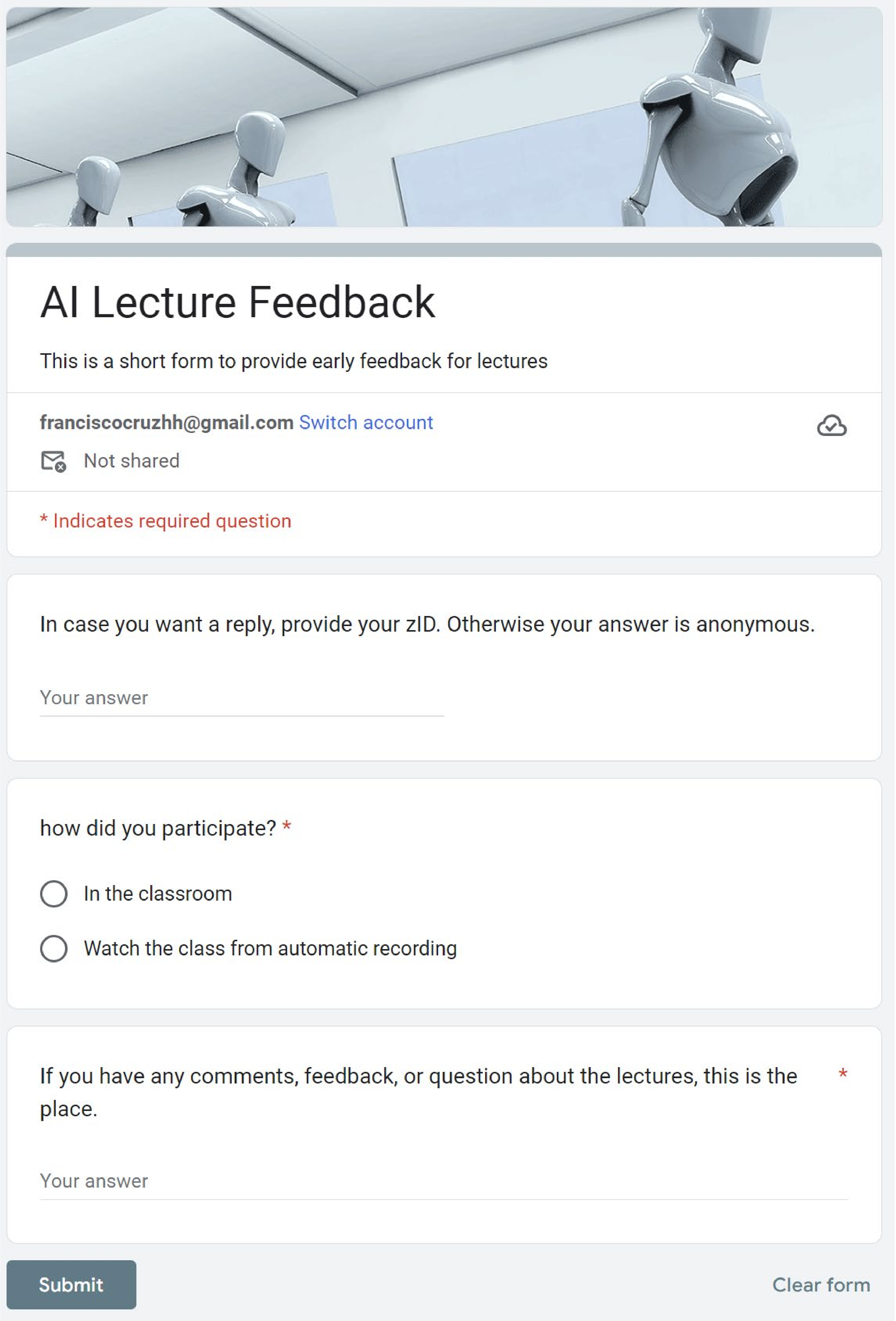
Rev. Sci. Instrum. 65 (6), June 1994 0034-6748/94/65(6)/1803/30/$6.00 © 1994 American Institute of Physics 1803

Feedback

- In case you want to provide anonymous feedback on these lectures, please visit:

- https://forms.gle/KBkN744QuffuAZLF8

Muchas gracias!



AI Lecture Feedback

This is a short form to provide early feedback for lectures

franciscocruzhh@gmail.com Switch account

Not shared

* Indicates required question

In case you want a reply, provide your zID. Otherwise your answer is anonymous.

Your answer

how did you participate? *

○ In the classroom

○ Watch the class from automatic recording

If you have any comments, feedback, or question about the lectures, this is the place. *

Your answer