علوم شناختی

جلسه ۱۱ (ب)

شبکه‌های تک-لایه
و توابع بولی

Single-Layer Networks and Boolean Functions

کاظم فولادی قلعه
دانشکده مهندسی، دانشکدگان فارابی
دانشگاه تهران

http://courses.fouladi.ir/cogsci

CAMBRIDGE

PART 2:
MODELS AND TOOLS

CAMBRIDGE UNIVERSITY PRESS

CAMBRIDGE

Chapter 5: Neural Networks and Distributed Information Processing

CAMBRIDGE
UNIVERSITY PRESS

Chapter 5.2: Single-layer networks and Boolean functions

Overview

- Introduce single unit networks and Boolean functions
- Introduce Hebbian learning
- Introduce the perceptron convergence rule and linear separability
- Explain the limits of learning in single unit networks

Single-unit networks as logic gates

- Single-unit networks can function as logic gates
  - They can compute basic binary Boolean functions
  - Because of this, networks of single-unit networks can compute any Boolean function whatsoever

Boolean functions

- Named after the mathematician and logician George Boole - inventor of Boolean algebra, Boolean functions etc etc

- Functions from sets of truth values to truth values

  Truth values are TRUE and FALSE

- Boolean functions can be of any (finite) arity

  0-ary function (e.g. the TRUE)

  1-ary function (e.g. NOT)

  binary function (e.g. AND)

Functions

- Mappings from a domain of objects into a range
- For Boolean functions the domain is made up of (tuples of) truth values
  - Binary Boolean functions: the domain is all the different possible pairs of truth values
- For Boolean functions the range is always the same
  - The set {TRUE, FALSE}

Boolean functions

- Boolean functions can be represented by truth tables

A	B	A AND B
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

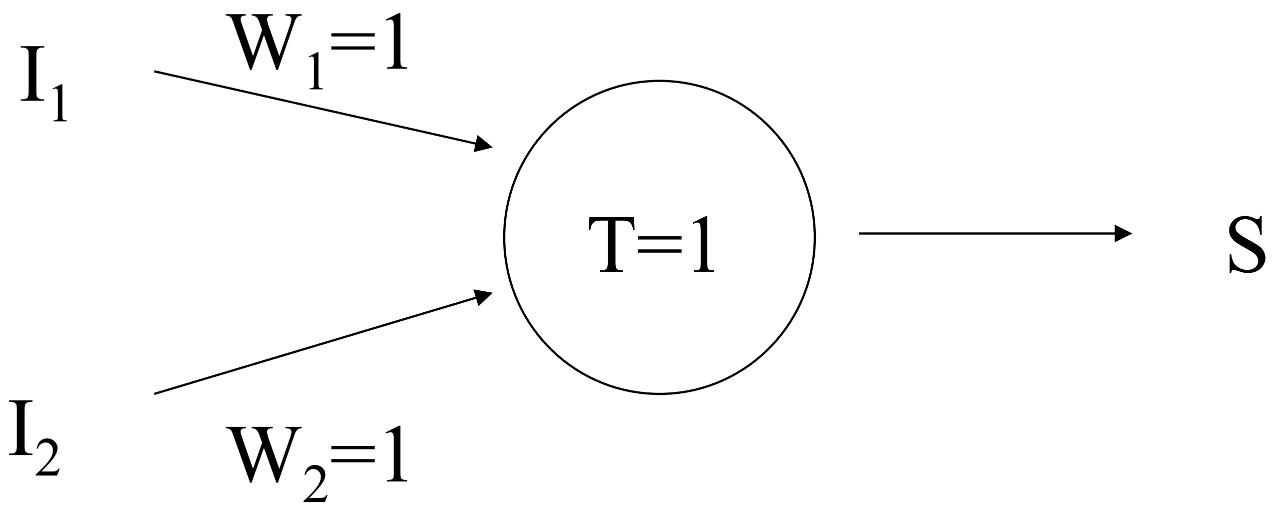
- AND, NOT, and OR are all Boolean functions

- Every Boolean function can be represented by a formula in disjunctive normal form

Single unit networks

- If we represent TRUE by 1 and FALSE by 0 then we can use single-unit networks to represent Boolean functions
  - The arity of the function is given by the number of inputs to the unit
- The weights, activation functions, and threshold need to be set so that the output is always 1 or 0
  - use a binary threshold activation function

OR-Network



The “neuron” will fire in every case except where both inputs are 0.

Learning in neural networks

- Neural networks are important because they allow us to model how information-processing capacities are learnt
- If we abstract away from learning, networks of single unit networks are simply implementations of symbolic systems
- Two types of learning

Supervised	[requires feedback]
Unsupervised	[no feedback]

Unsupervised learning

- Simplest algorithms for unsupervised learning are forms of Hebbian learning
  - Basic principle: Neurons that fire together, wire together
- "When an axon of a cell A is near enough to excite call B or repeatedly or persistently takes part in firing it, some growth or metabolic change takes place in both cells such that A's efficiency, as one of the cells firing B, is increased."

Hebbian learning

- Standardly used in pattern associator networks
- Very good at generalizing patterns
- Also feature in more complex learning rules (e.g. competitive learning)
- Simple formal expression: $\Delta w_{12} = \varepsilon \times a_1 \times a_2$

Perceptron convergence rule

- Also known as the delta rule

- Distinct from Hebbian learning in that training depends upon the discrepancy between actual output and intended output

$\delta =$ error measure

(Intended output – actual output)

Applying the delta rule

Delta rule gives algorithm for changing threshold and weights as a function of δ and ε (a learning rate constant)

$$\Delta T = -\varepsilon \times \delta$$

$$\Delta W_i = \varepsilon \times \delta \times I_i$$

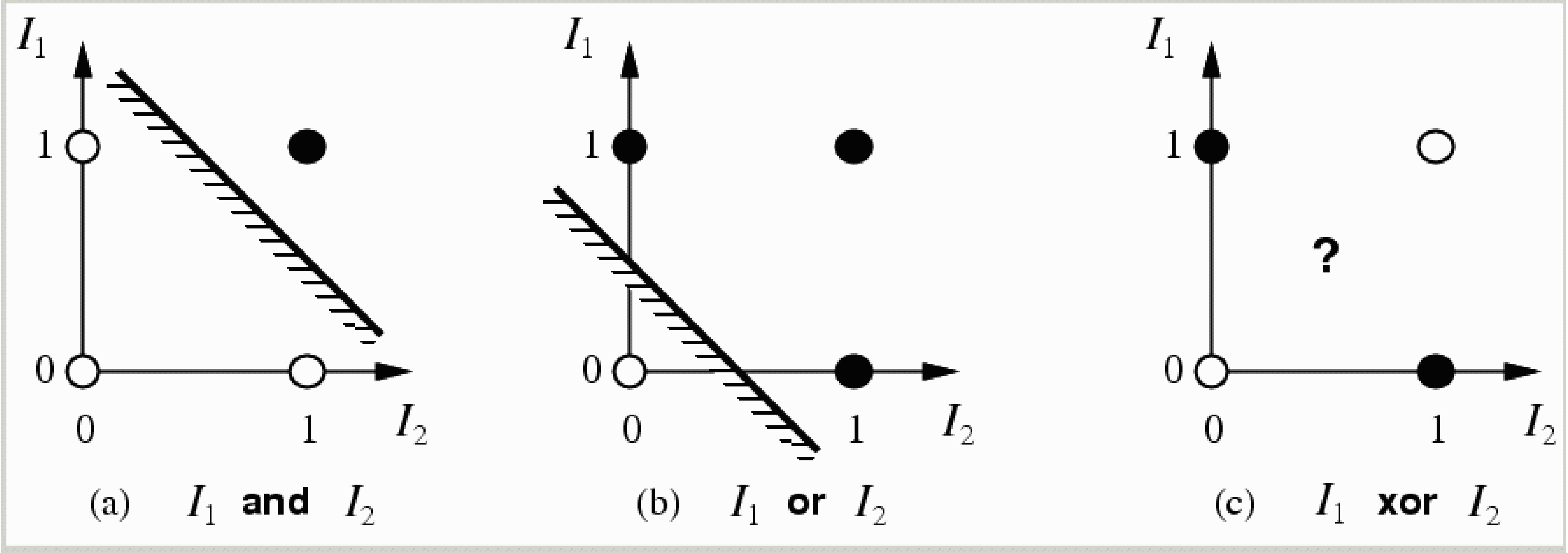
Perceptron convergence theorem

The perceptron convergence rule will converge on a solution in every case where a solution is possible

i.e. it will generate a set of weights and a threshold that will compute every Boolean function that can be computed by a perceptron (i.e. a single layer network)

But which functions are those?

Linearly separable functions



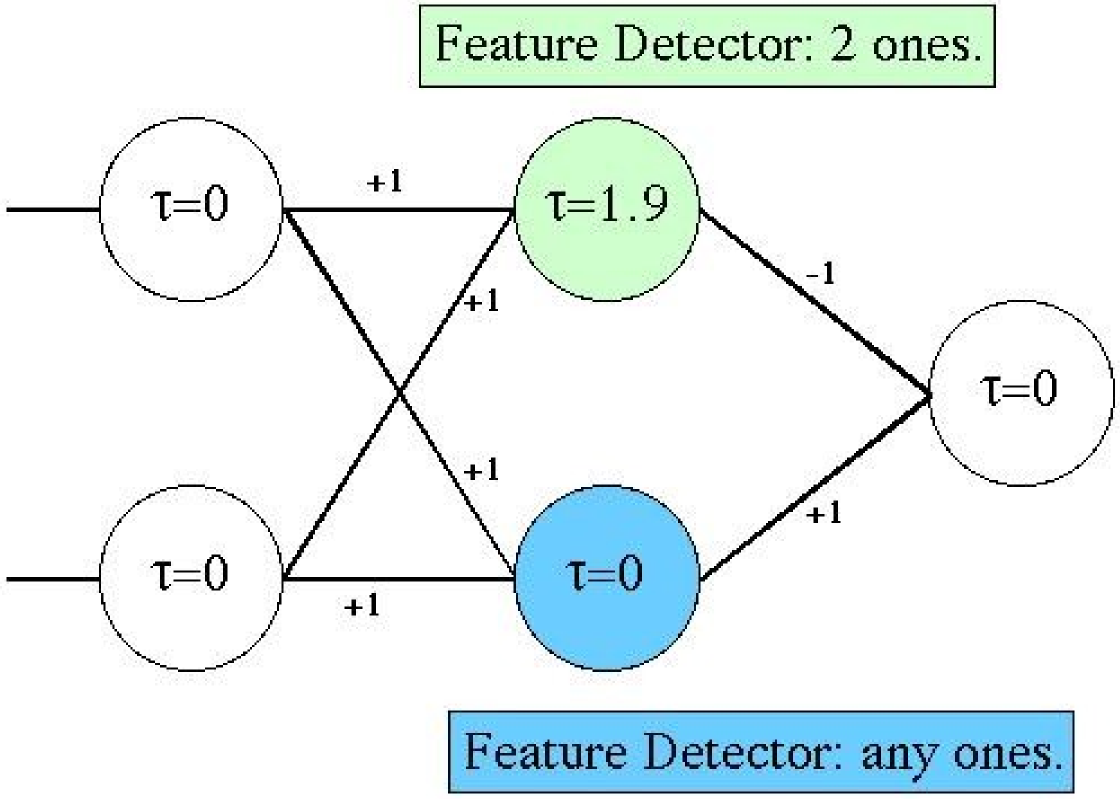
Notion of linear separability can be extended to cover *n*-ary Boolean functions for $n > 2$

XOR is linearly separable

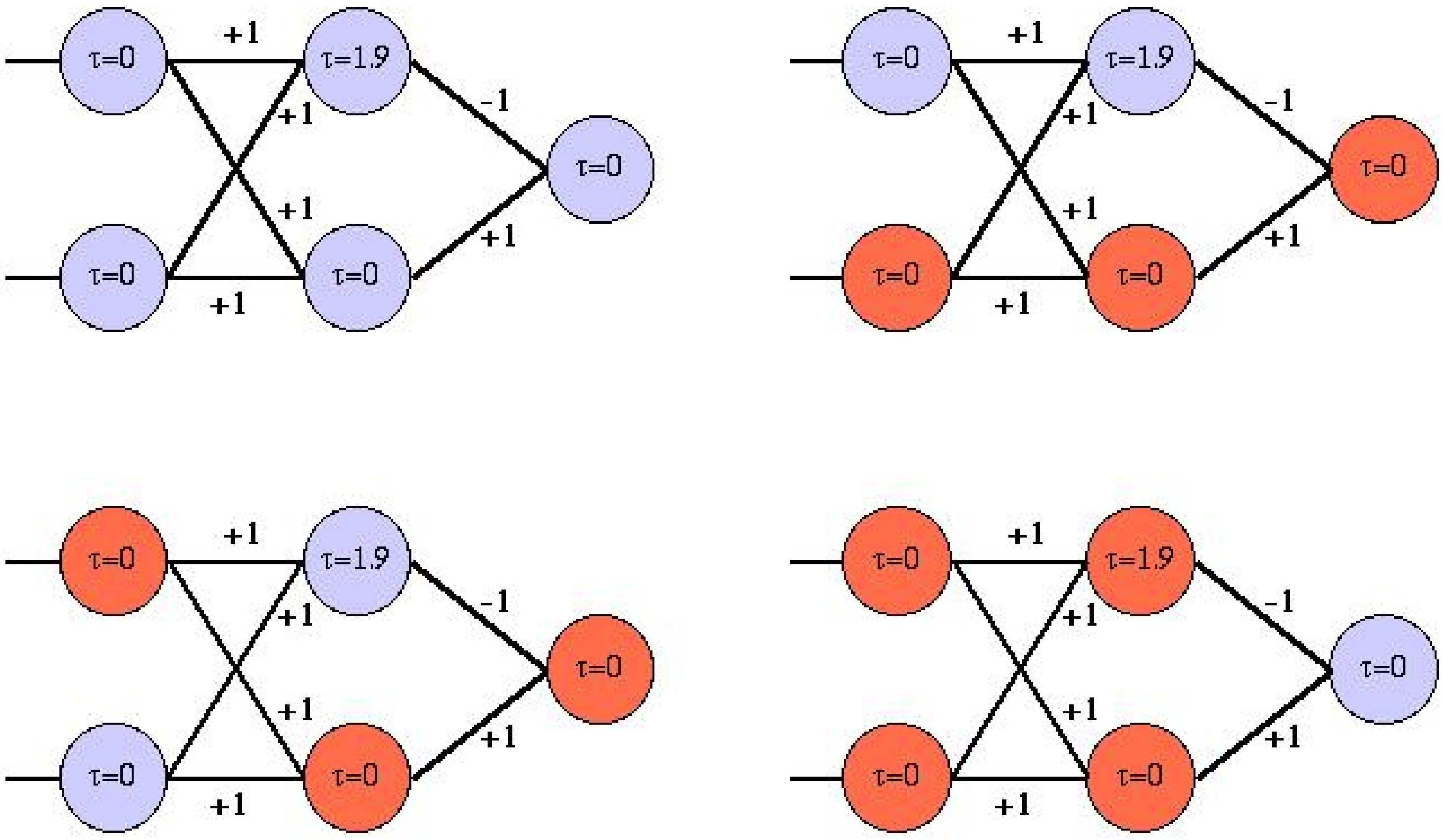
I_1	I_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

- Network must output 1 when $I_2=1$
- So we have $1 \times W_2 > T$
- Likewise, the network must output 1 when $I_1 =1$
- So we have $1 \times W_1 > T$
- But then we must have $(1xW_1)+(1xW_2)>T$
- So the network will output 1 when I_1 and I_2 are both 1

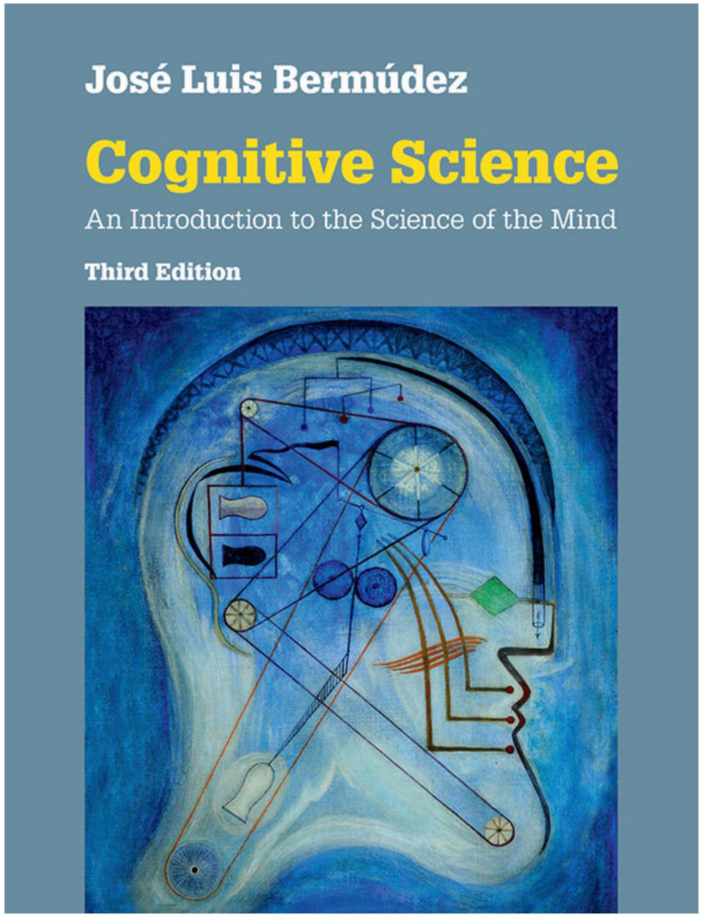
XOR network


Feature Detector: 2 ones.
τ=0
+1
τ=1.9
-1
+1
τ=0
+1
+1
τ=0
+1
τ=0
Feature Detector: any ones.

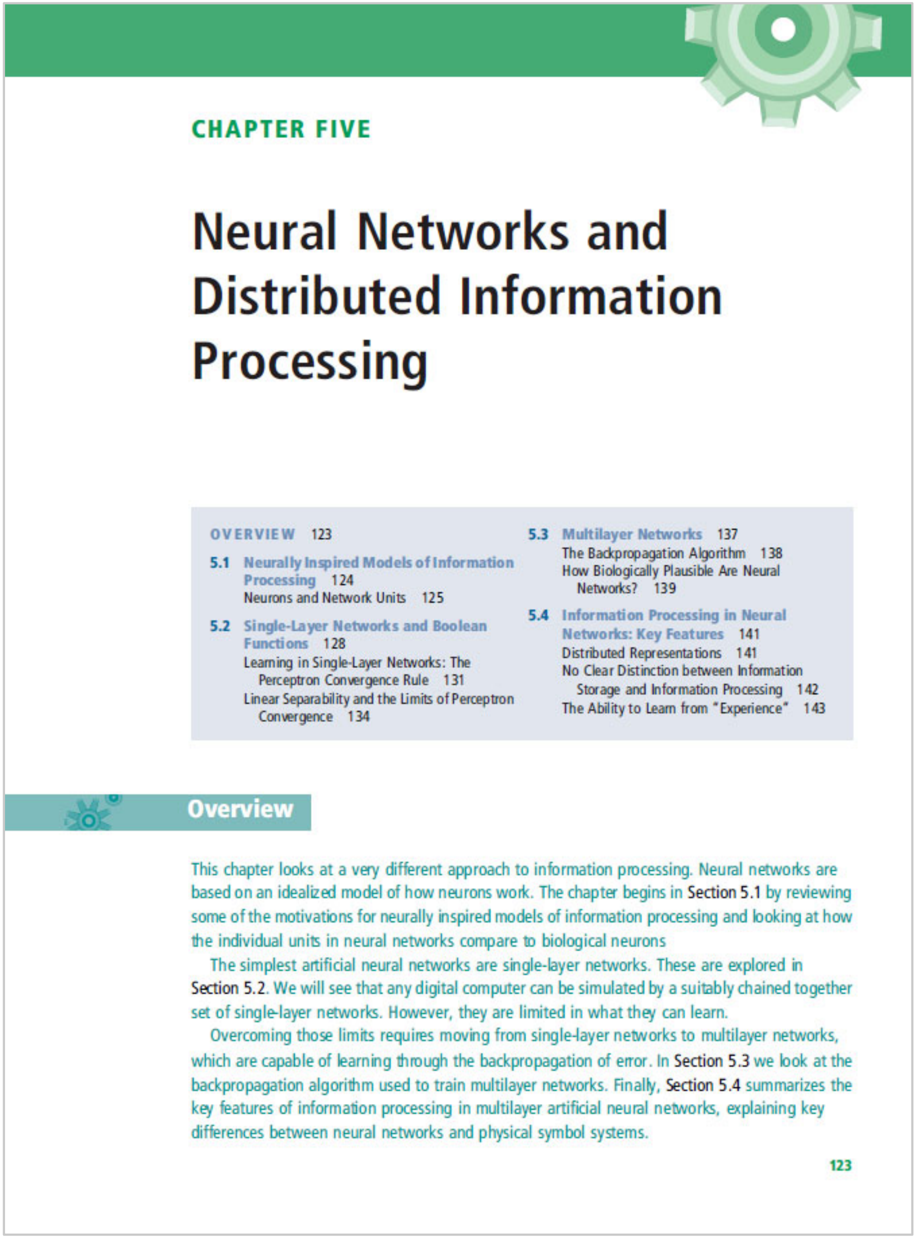
XOR network – Possible Activations



منبع اصلی



José Luis Bermúdez,
Cognitive Science:
An Introduction to the Science of the Mind,
3rd ed., Cambridge University Press, 2020.
Chapter 5 (Section 5.2)



CHAPTER FIVE

Neural Networks and Distributed Information Processing

Overview

This chapter looks at a very different approach to information processing. Neural networks are based on an idealized model of how neurons work. The chapter begins in Section 5.1 by reviewing some of the motivations for neurally inspired models of information processing and looking at how the individual units in neural networks compare to biological neurons

The simplest artificial neural networks are single-layer networks. These are explored in Section 5.2. We will see that any digital computer can be simulated by a suitably chained together set of single-layer networks. However, they are limited in what they can learn.

Overcoming those limits requires moving from single-layer networks to multilayer networks, which are capable of learning through the backpropagation of error. In Section 5.3 we look at the backpropagation algorithm used to train multilayer networks. Finally, Section 5.4 summarizes the key features of information processing in multilayer artificial neural networks, explaining key differences between neural networks and physical symbol systems.

123