

Computational Logic and Connectionist Systems

▶ Preliminary Table of Contents:

- ▷ Introduction
- ▷ Neural Networks
- ▷ Connectionist Models
- ▷ McCulloch-Pitts Networks and Finite Automata
- ▷ Symmetric Networks and Propositional Logic
- ▷ Spreading of Activation and Reflexive Reasoning
- ▷ Recursive Networks and Model Generation

Introduction: Connectionist Systems

- ▶ Biological plausible, massively parallel, robust.
- ▶ Well-suited to learn, to adapt to new environments, to degrade gracefully.
- ▶ Many successful applications.
- ▶ Approximate functions.
 - ▷ Hardly any knowledge about the functions is needed.
 - ▷ Trained using incomplete data.
- ▶ Declarative semantics is not available.
- ▶ Recursive networks are hardly understood.
- ▶ We still observe a propositional fixation (McCarthy (1988)).
- ▶ Structured objects are difficult to represent.

Can we instantiate the power of symbolic computation within fully connectionist systems? Smolensky (1987)

Introduction: Computational Logic Systems

- ▶ Well-suited to represent and reason about structured objects and structure-sensitive processes.
- ▶ Many successful applications.
- ▶ Direct implementation of relations and functions.
- ▶ Explicit expert knowledge is required.
- ▶ Highly recursive structures.
- ▶ Well understood declarative semantics.
- ▶ But logic systems are brittle.
- ▶ Expert knowledge may not be available.

Can we instantiate the power of connectionist computation within a logic system?

Introduction: Objective

- ▶ Seek the best of both paradigms!
- ▶ Understanding the relation between connectionist and computational logic systems.
- ▶ **Contribute to the open research problems of both areas.**
- ▶ Well developed for propositional case.
- ▶ Hard problem: going beyond.

Connectionist Models

▶ **Connectionist network:**

- ▶ set of units U
- ▶ set of labelled connections $W \subseteq U \times U$; weight: $W \rightarrow \mathbb{R}$

▶ **Unit u :**

- ▶ input vector (i_1, \dots, i_m) ,
- ▶ activation function ϕ ,
- ▶ potential $p = \phi(i_1, \dots, i_n) \in \mathbb{R}$,
- ▶ output function ψ
- ▶ (output) value $v = \psi(p)$
- ▶ linear time t

▶ **State: $(v_1(t), \dots, v_n(t))$**

Connection Structure

▶ **Connections:**

- ▶ **directed and weighted connection from u_j to u_k**
 - **weight $w_{kj} \in \mathbb{R}$**
 - **input $i_k = w_{kj}v_j$**
- ▶ **higher order connection from units u_{j_1}, \dots, u_{j_m} to unit u_k**
 - **weight $w_{kj_1\dots j_m} \in \mathbb{R}$ ($j_1 < \dots < j_m$)**
 - **input $i_k = w_{kj_1\dots j_m} \prod_{l=1}^m v_{j_l}$**
- ▶ **Undirected connections**

Activation Functions

▶ Activation function ϕ of u_k :

▶ weighted sum

$$\phi(i_1, \dots, i_m, t + 1) = \sum_{j=1}^m w_{kj} v_j(t),$$

▶ sigma-pi

$$\phi(i_1, \dots, i_m, t + 1) = \sum_{j=1}^m w_{kj_1 \dots j_{m_j}} \prod_{l=1}^{m_j} v_{j_l}(t),$$

Output Functions

▶ Output function ψ of unit u_k :

▶ Binary threshold function:

$$v_k(t) = \begin{cases} 1 & \text{if } p_k(t) > \theta_k \\ 0 & \text{if } p_k(t) < \theta_k \\ v_k(t-1) & \text{if } p_k(t) = \theta_k \end{cases},$$

▶ p_k : potential,

▶ $\theta_k \in \mathbb{R}$ threshold of u_k .

▶ Sigmoidal

▶ non-decreasing,

▶ $\lim_{\lambda \rightarrow \infty} (\psi(\lambda)) = 1$,

▶ $\lim_{\lambda \rightarrow -\infty} (\psi(\lambda)) = 0$,

where $\lambda \in \mathbb{R}$ depends on the potential p_k of u_k .

Units

- ▶ **Binary threshold unit:**
 - ▷ activation function: weighted sum,
 - ▷ output function: binary threshold function.

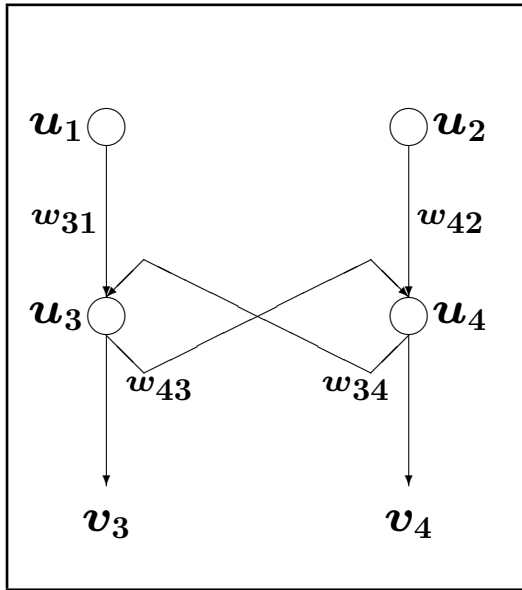
- ▶ **Sigma-pi unit:**
 - ▷ activation function: sigma-pi,
 - ▷ output function: binary threshold function.

- ▶ **Active/passive units.**

- ▶ **Flips.**

- ▶ **Input units.**

Example



$$w_{34} = -0.5$$

$$w_{43} = -0.5$$

$$w_{31} = 1$$

$$w_{42} = 1$$

$$p_i(t+1) = p_i(t) + \sum_{j=1}^4 w_{ij}v_j(t)$$

$$v_i(t) = \text{round}(p_i(t))$$

$$v_1(t) = \begin{cases} 6 & \text{if } t = 0 \\ 2 & \text{otherwise} \end{cases}$$

$$v_2(t) = \begin{cases} 5 & \text{if } t = 0 \\ 2 & \text{otherwise} \end{cases}$$

- ▶ What happens if the network is synchronously updated?
- ▶ **Exercise:** A winner-take-all network is a synchronously updated connectionist network of n units (not counting input units) such that after each unit receives an initial input at $t = 0$ eventually only the unit with the highest initial input produces a value greater than 0 whereas the value of all other units is 0. Construct a winner-take-all network of 4 units.