Computational Intelligence

Winter Term 2009/10

Prof. Dr. Günter Rudolph
Lehrstuhl für Algorithm Engineering (LS 11)
Fakultät für Informatik
TU Dortmund

Plan for Today

- Organization (Lectures / Tutorials)
- Overview CI
- Introduction to ANN
  - McCulloch Pitts Neuron (MCP)
  - Minsky / Papert Perceptron (MPP)

Organizational Issues

Who are you?

either

studying “Automation and Robotics” (Master of Science)
Module “Optimization”
or

studying “Informatik”
- BA-Modul “Einführung in die Computational Intelligence”
- Hauptdiplom-Wahlvorlesung (SPG 6 & 7)

Who am I?

Günter Rudolph
Fakultät für Informatik, LS 11
Guenter.Rudolph@tu-dortmund.de ← best way to contact me
OH-14, R. 232 ← if you want to see me
office hours:
Tuesday, 10:30–11:30am
and by appointment
Organizational Issues

Lectures
Wednesday 10:15-11:45 OH-14, R. E23

Tutorials
Wednesday 16:15-17:00 OH-14, R. 304 group 1
Thursday 16:15-17:00 OH-14, R. 304 group 2

Tutor
Nicola Beume, LS11

Information

Prerequisites

Knowledge about
- mathematics,
- programming,
- logic
is helpful.

But what if something is unknown to me?
- covered in the lecture
- pointers to literature

... and don’t hesitate to ask!

Overview “Computational Intelligence”

What is CI?
⇒ umbrella term for computational methods inspired by nature

- artificial neural networks
- evolutionary algorithms
- fuzzy systems
- swarm intelligence
- artificial immune systems
- growth processes in trees
- ...

Our goals:
1. know what CI methods are good for!
2. know when refrain from CI methods!
3. know why they work at all!
4. know how to apply and adjust CI methods to your problem!
### Biological Prototype

- **Neuron**
  - Information gathering (D)
  - Information processing (C)
  - Information propagation (A / S)

  *human being: $10^{12}$ neurons*
  *electricity in mV range*
  *speed: 120 m / s*

- **cell body (C)**
- **dendrite (D)**
- **nucleus**
- **axon (A)**
- **synapse (S)**

### Abstraction

- **nucleus**
- **cell body**
- **axon**
- **synapse**
- **dendrites**

  - **signal**
  - **input**
  - **signal processing**
  - **signal output**

### Model

$$ f(x_1, x_2, \ldots, x_n) $$

**McCulloch-Pitts-Neuron 1943:**

- $x_i \in \{0, 1\} =: \mathbb{B}$
- $f: \mathbb{B}^n \rightarrow \mathbb{B}$

**1943: Warren McCulloch / Walter Pitts**

- Description of neurological networks
  - Modell: McCulloch-Pitts-Neuron (MCP)

- Basic idea:
  - Neuron is either active or inactive
  - Skills result from connecting neurons

- Considered static networks
  (i.e., connections had been constructed and not learnt)
McCulloch-Pitts-Neuron

n binary input signals $x_1, \ldots, x_n$
threshold $\theta > 0$

$$f(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i \geq \theta \\ 0 & \text{else} \end{cases}$$

boolean OR

\[ \Rightarrow \text{can be realized: } \]

\[ \begin{align*}
& x_1 \\
\rightarrow & x_2 \\
\end{align*} \]

boolean AND

\[ \begin{align*}
& x_1 \\
\rightarrow & x_2 \\
\end{align*} \]

\[ \sum_{i=1}^{n} x_i \geq \theta \]

\[ \Rightarrow \text{output = 1} \]

\[ \sum_{i=1}^{n} x_i = \theta \]

\[ \Rightarrow \text{output = 0} \]

in addition: m binary inhibitory signals $y_1, \ldots, y_m$

\[ f'(x_1, \ldots, x_n, y_1, \ldots, y_m) = f(x_1, \ldots, x_n) \cdot \prod_{j=1}^{m} (1 - y_j) \]

- if at least one $y_j = 1$, then output = 0
- otherwise:
  - sum of inputs $\geq$ threshold, then output = 1
  - else output = 0

Analogons

Neurons
simple MISO processors
(with parameters: e.g. threshold)

Synapse
connection between neurons
(with parameters: synaptic weight)

Topology
interconnection structure of net

Propagation
working phase of ANN
→ processes input to output

Training / Learning
adaptation of ANN to certain data

Theorem:
Every logical function $F: \mathbb{B}^n \rightarrow \mathbb{B}$ can be simulated with a two-layered McCulloch/Pitts net.

Example:
$$F(x) = x_1x_2\overline{x}_3 \lor \overline{x}_1x_2\overline{x}_3 \lor x_1\overline{x}_4$$

Assumption:
inputs also available in inverted form, i.e. $\exists$ inverted inputs.
Proof: (by construction)

Every boolean function \( F \) can be transformed in disjunctive normal form
\( \Rightarrow 2 \) layers (AND - OR)

1. Every clause gets a decoding neuron with \( \theta = n \)
\( \Rightarrow \) output = 1 only if clause satisfied (AND gate)

2. All outputs of decoding neurons
   are inputs of a neuron with \( \theta = 1 \) (OR gate)

q.e.d.

Generalization: inputs with weights

\[
\begin{align*}
\text{fires 1 if} & \quad 0.2 \, x_1 + 0.4 \, x_2 + 0.3 \, x_3 \geq 0.7 \\
\Rightarrow & \quad 2 \, x_1 + 4 \, x_2 + 3 \, x_3 \geq 7 \\
\Downarrow & \quad \text{duplicate inputs!}
\end{align*}
\]

\Rightarrow \text{equivalent!}

Conclusion for MCP nets

+ feed-forward: able to compute any Boolean function
+ recursive: able to simulate DFA

- very similar to conventional logical circuits
- difficult to construct
- no good learning algorithm available
Perceptron (Rosenblatt 1958)
→ complex model → reduced by Minsky & Papert to what is „necessary“
→ Minsky-Papert perceptron (MPP), 1969

What can a single MPP do?

\[ w_1 x_1 + w_2 x_2 \geq \theta \]

isolation of \( x_2 \) yields:

\[ x_2 \geq \frac{\theta - w_1}{w_2} \]

Example:

\[ 0.9 x_1 + 0.8 x_2 \geq 0.6 \]

\( x_2 \geq \frac{3}{4} \cdot \frac{9}{8} x_1 \)

separating line in 2 classes

1969: Marvin Minsky / Seymour Papert

- book *Perceptrons* → analysis math. properties of perceptrons
- disillusioning result: perceptions fail to solve a number of trivial problems!
  - XOR-Problem
  - Parity-Problem
  - Connectivity-Problem
- „conclusion“: All artificial neurons have this kind of weakness!
  ⇒ research in this field is a scientific dead end!
- consequence: research funding for ANN cut down extremely (~ 15 years)
How to obtain weights $w_i$ and threshold $\theta$?

**as yet:** by construction

**example:** NAND-gate

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\text{NAND}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$\Rightarrow 0 \geq \theta$  
$\Rightarrow w_2 \geq \theta$  
$\Rightarrow w_1 \geq \theta$  
$\Rightarrow w_1 + w_2 < \theta$

(e.g.: $w_1 = w_2 = -2, \theta = -3$)

**now:** by “learning” / training

**Perceptron Learning**

**Assumption:** test examples with correct I/O behavior available

**Principle:**
1. choose initial weights in arbitrary manner
2. fed in test pattern
3. if output of perceptron wrong, then change weights
4. goto (2) until correct output for all test patterns

**Example**

| $P$: set of positive examples |
| $N$: set of negative examples |
| $w$: threshold as a weight: $w = (\theta, w_1, w_2)'$ |

Threshold as a weight: $w = (0, w_1, w_2)'$

**graphically:**

→ translation and rotation of separating lines
We know what a single MPP can do. What can be achieved with many MPPs?

- Single MPP ⇒ separates plane in two half planes
- Many MPPs in 2 layers ⇒ can identify convex sets

∀ a, b ∈ X: λ a + (1-λ) b ∈ X for λ ∈ (0,1)

1. How? ⇒ 2 layers!
2. Convex?⇐

Many MPPs in 2 layers ⇒ can identify convex sets
Many MPPs in 3 layers ⇒ can identify arbitrary sets
Many MPPs in > 3 layers ⇒ not really necessary!

arbitrary sets:
1. partitioning of nonconvex set in several convex sets
2. two-layered subnet for each convex set
3. feed outputs of two-layered subnets in OR gate (third layer)