Connections and symbols

AT1

Emmanuel Dupoux
bref historique du connexionnisme et de l’IA symbolique
les années 80: la confrontation ideologique: faux débat versus vraies questions
Les nouvelles pistes
  compositionalité et produit tensoriel
  recursivité et système dynamique
  (gradualité et modèles bayesiens)
Symbols
Cognition and symbols

reasoning = computation in a formal calculus

- Euclides, Leibniz (universal language + reasoning calculus), Boole (formal system for logical and set theoretic reasoning), Frege (logisict programme for mathematics), Peano, Russell, etc

Formal system:
- A finite set of symbols (i.e. the alphabet), that can be used for constructing formulas (i.e. finite strings of symbols).
- A grammar, which tells how well-formed formulas (abbreviated wff) are constructed out of the symbols in the alphabet. It is usually required that there be a decision procedure for deciding whether a formula is well formed or not.
- A set of axioms or axiom schemata: each axiom must be a wff.
- A set of inference rules (going from wff to wff)

Examples
- first order propositional logic (A,B..C, -)
- second order logic (A,B, C, Vα, -)
- predicate logic (P(x))
Cognition and symbols (II)

Thinking = computation over symbols
computability theory (1930-) Godel, Post, Kleene, Church, Turing, Markov

Physical symbolic systems
Finite state automaton, pushdown automaton, Turing machines computational power

Power of automata:
- Finite state automata
- Pushdown automata
- Turing machines with one tape, Turing machines with several tapes, lambda calculus (eg lisp), string rewriting system (eg production), recursive functions

Church thesis:
Turing machines are on the top of the hierarchy
Cognition and symbols (III)

Language
Miller & Chomsky (1963)
  to each automaton, a class of languages
  human languages are between context free and recursively enumerable

Symbolic AI
  prolog, expert systems
  SOAR, ACT*
Newell (1980) articulated the role of the mathematical theory of *symbolic* processing.

Cognition involves the manipulation of symbols – analogous to words, concepts, schema, etc.

What are symbols?

Definition is hard to pin down.

Roughly, it’s like the values of a categorical variable (male, female, red, blue, dog, cat).

Operators on those symbols would then be things like “is-a” “a-kind-of” “purpose” “shape” “part-of” “object”
E.g. recognize a red apple

input = symbol(s) -> algorithms who work on input -> output = more symbol(s)

Input: Red(X) Round(X) ...

Program:
if (Orange(X) & Round(X) ...) then Orange(X)
if (Red(X) & Round(X) ...) then Apple(X)
...

Output: Apple(X)

Graphical:
Apple(X)

Red(X) & Round(X) ...

Color(X) -> Red(X) or Orange(X)
Shape(X) -> Round(X) or ...
Connections
McCulloch & Pitts (1943)

Neural networks as computing devices

What logical operations could neurons compute?

Five assumptions based on then-current knowledge of neurons

1. The activity of a neuron is “all-or-none” (binary coding)
2. Each neuron has a fixed threshold on the required number of synapses that need to be excited before the neuron itself will be excited. Weights are identical.
3. Synaptic action causes a time delay before firing.
4. Inhibition is absolute.
5. The physical structure of a network of neurons doesn’t change with time; connections and their strengths are static.
McCulloch/Pitts neurons

McCulloch/Pitts neurons can then be used to compute any (finite) logical function

BUT, McCulloch/Pitts networks can’t learn.
Hebb (1949)

The first rule for self-organized learning
Hebb recognized the existence of feedforward, long range lateral, and feedback connections
These cortical circuits admit self-sustaining activity that reverberate in « cell assemblies »
synapses are the fundamental computational and learning unit
activity-dependent synaptic plasticity as a basic operation

Learning in a Hebbian network

“When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some grown process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased.”

LT Potentiation (Bliss & Lomo, 1973; Kelso et al, 1986)
LT Depression (Markram et al. 1997)
The “Hebb rule”

\[ \Delta w_{ij} = \eta a_i a_j \]

where \( a' \)’s are activation values (-1 or 1), and \( \eta \) is a learning rate parameter.

Equation is applied until weights “saturate” (typically at 1) and do not keep increasing as inputs are presented.

Think of Hebbian learning as picking up on correlations between features in the environment

Features that co-occur will grow strong positive weights, those that do not occur together will have grow negative weights, random pairing produces zero weights
The perceptron
(Rosenblatt, 1958, 1962)

First model for learning with a teacher (supervised learning)

McCulloch-Pitts neurons (linear-threshold) with connections that can be modified by learning

\[ y = \begin{cases} 
1 & \text{if } \sum_{i=0}^{n} w_i x_i > 0 \\
-1 & \text{otherwise}
\end{cases} \]

Perceptron Learning Rule

Start with random connections $w_i$

Error-driven learning rule (delta rule):

$$\Delta w_i = \eta (t - y) x_i$$

- $t$ is the target value (given by the teacher)
- $y$ is the perceptron output
- $\eta$ is a small constant (e.g. 0.1) called learning rate

• If the output is correct ($t=\ y$) the weights $w_i$ are not changed
• If the output is incorrect ($t\neq y$) the weights $w_i$ are changed such that the output of the perceptron for the new weights is closer to $t$ (error decreases).
• The algorithm converges to the correct classification
  • if the training data is linearly separable
  • and $\eta$ is sufficiently small
Example: learning the AND

<table>
<thead>
<tr>
<th>AND</th>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Initial weights: 0, n=0.25
Example: learning the AND

\[
\begin{align*}
W &= 0 \\
W &= 0 \\
W &= 0 \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Example: learning the AND

\[ y = \frac{x_1^2}{x_1} \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**AND**: t = 1, error = 1
Example: learning the AND

\[ y = x_1^2 = 0 \]
\[ x_1 = 1 \]
\[ x_2 = 0 \]
\[ W = 0.25 \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

AND

\[ t = 0 \]
\[ error = -1 \]
Example: learning the AND

\[ y = \begin{cases} 1 & x_1^2 = 0 \\ 0 & x_1^2 = 1 \\ x_1 = 1 & x_2 = 0 \\ 0.25 & x_1 = 0 \\ 0 & x_2 = 0 \end{cases} \]

**AND Truth Table**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Error = -1
Example: learning the AND

\[
\begin{align*}
W &= 0 \\
W &= 0.25
\end{align*}
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Error = -1
Example: learning the AND

\[
\begin{align*}
&x_1=0 \\
&x_2=1 \\
&y=1 \\
&1
\end{align*}
\]

\[
W = -0.25
\]

\[
t = 0
\]

\[
error = -1
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Example: learning the AND

\[ y = 1 \]
\[ x_1 = 0 \]
\[ x_2 = 1 \]
\[ W = -0.25 \]
\[ W = 0 \]
\[ W = 0 \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ t = 0 \]
\[ error = -1 \]

... and so on and so forth ...
Example: learning the AND

<table>
<thead>
<tr>
<th>AND</th>
<th>A</th>
<th>B</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Final weights

\[
W = \begin{cases} 
-0.25 & \text{if } \text{AND} = 0 \\
0.25 & \text{if } \text{AND} = 1 
\end{cases}
\]
Perceptron is doing something similar to linear discriminant analysis, binomial regression (or Bayes classification when the distributions are gaussian).

Decision boundary is a hyperplane when more than 2 inputs.
Generalization: Multiple outputs

- similar multinomial LDA or multinomial regression
Minsky & Papert (1969)

Presented a formal analysis of the properties of perceptrons and revealed several fundamental limitations.

Limitations

Can’t learn nonlinearly separable problems like the XOR

More…
Decision Boundary of a Perceptron

- **x₁ OR x₂**: Linearly separable
- **x₁ AND x₂**: Linearly separable
- **x₁ XOR x₂**: Non-Linearly separable

![Perceptron Decision Boundaries](image)
Limitations

So….can’t learn nonlinearly separable problems like the XOR

Although including “hidden layers” allows one to hand-design a network that can represent XOR and related problems, they showed that the perceptron learning rule can’t learn the required weights.

They also showed that even those functions that can be learned by perceptron rule learning may require huge amounts of learning time.
Fallout of
Minsky & Papert’s analysis

This paper was nearly the death of this budding field.

Subsequent research was largely done in “garages”.

i.e., only in obscure academic circles.
The revival: the 80s

multilayered feedforward networks
   Generalization of the Delta Rule: backpropagation of error
   learning of distributed representations

Symmetric recurrent networks
   winner take all
   autoassociation
The revival I: the Multi-Layer Perceptron

Activation functions

- step function
- sigmoid function
The backpropagation learning rule (I)

\[ e_i = (t_i - o_i) \]

\[ \Delta w_{ij} = o_i (1-o_i) \eta e_i h_j \]
The backpropagation learning rule (II)

\[ e'_j = \sum_i w_{ij} e_i \]

\[ \Delta w'_{jk} = h_i(1-h_i) \eta e'_j x_k \]
Expressive Capabilities of MLP

Boolean functions
Every boolean function can be represented by network with single hidden layer
But might require exponential (in number of inputs) hidden units

Continuous functions
Every bounded continuous function can be approximated with arbitrarily small error, by network with one hidden layer [Cybenko 1989, Hornik 1989]
Any function can be approximated to arbitrary accuracy by a network with two hidden layers [Cybenko 1988]
## Different Non-Linearly Separable Problems

<table>
<thead>
<tr>
<th>Structure</th>
<th>Types of Decision Regions</th>
<th>Exclusive-OR Problem</th>
<th>Classes with Meshed regions</th>
<th>Most General Region Shapes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Layer</td>
<td>Half Plane Bounded By Hyperplane</td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>Two-Layer</td>
<td>Convex Open Or Closed Regions</td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
<tr>
<td>Three-Layer</td>
<td></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Example: Neural network for OCR

- feedforward network
- trained using Back-propagation
Example: ALVINN

Drives 70 mph on a public highway

30 outputs for steering

4 hidden units

30x32 pixels as inputs

32 weights into one out of four hidden units
A very simple example: a winner-take all network

at each step:
- compute the total input for all units
- compute the output for all units
- iterate
The revival II: Symmetric dynamical networks

Boltzman machine (Hinton & Sejnovsky, 1983)
neurons: binary (-1, 1)
network: symmetric weights
update: stochastic (1 with probability \( \frac{1}{1+e^{-\text{sum}(\text{inputs})}} \))
dynamics: global energy minimization, basins of attraction
learning rule: tries to reproduce the distribution of its inputs (stochastically learns the basins of attraction)

Hopfield network (Hopfield 1982)
deterministic variant (‘temperature’=0)
learning rule becomes Hebb Rule.
performs pattern completion, content-addressable memory
The revival III: McClelland & Rumelhart’s (1986) Parallel Distributed Processing

Basic mechanisms
- feature discovery and competitive learning
- dynamical system and harmony theory
- learning in Boltzmann machines
- internal representation through backpropagation

Formal analysis
- linear algebra
- activation functions
- delta rule

Psychological Applications
- schemata and sequential through processes
- speech perception (TRACE)
- blackboard model of reading
- learning and memory
- learning past tense of English verbs
- sentence processing: assigning roles to constituents

Biological mechanisms
- anatomy of cortex
- place recognition and goal location
- neural plasticity and critical period
- amnesia and distributed memory
An example: early models of lexical access

Morton (1969) Logogen theory

Forster (1976) Serial search


a new PDP model

Elman & McClelland (1986)

The revival III: McClelland & Rumelhart’s (1986) Parallel Distributed Processing

Cognition involves the spreading of activation, relaxation, statistical correlation.

Represents a method for how symbolic systems might be implemented

- Hypothesized that apparently symbolic processing is an emergent property of subsymbolic operations.

Advantages

- Fault tolerance & graceful degradation
- Can be used to model learning
- More naturally capture nonlinear relationships
- Fuzzy information retrieval
- Bridges the gap with real neural processing
The critique: Pinker & Mehler (1988)

Lachter & Bever: connectionist theories are a return to associationism (Chomsky vs Skinner revisited)

Pinker & Prince: connectionist models of the capacity to derive the past tense of English verbs is inadequate
  rules: wug → wugged
  exceptions: put -> put, go->went, dig->dug

Fodor & Pylyshyn: connectionist theories are inadequate models of language and thought
Position of the problem: classical theories vs connectionism

Agree:
both classical theories & connectionism are *representationalsists* (they assign some ‘meaning’ to the elements – symbols or nodes)

Disagree
classical theory encode structural relationships and processes (eg, constituents, variables, rules)
connectionists only encode causal relationships and processes (x causes y to fire)

Arguments against connectionist systems: mental representation and processes are structure sensitive

combinatorial semantics
semantics of « J. loves M. » derived from semantics of « J. », « loves » and « M. »

productivity
the list of thoughts/sentences is not finite (I can construct new thoughts with old ones)

systematicity
I construct them in a systematic way
eg: « x loves M. » (where x can be any proper noun)
eg: If I can think « J. loves M. », I can think « M. loves J. »

recursivity & constituent structure:
If I can think « P. thinks that M. is nice » I can think « J thinks that P thinks that M is nice »

-> connectionist systems have none of the above properties
Objections to symbolic/classical systems
- rapidity of cognitive processes/neural speed
- difficulty of pattern recognition/content based retrieval in conventional architectures
- committed to rule vs exception dichotomy
- inadequate for intuitive /nonverbal behavior
- acutely sensitive to damage/noise (vs graceful degradation)
- storage in classical systems is passive
- inadequate account of gradual/frequency based application of rules
- inadequate account of nondeterminism
- no account of neuroscience

→ none of these arguments are valid or relevant

CONCLUSIONS
1. current connectionist theories are inadequate
2. if they were to be made adequate they would be mere implementation of classical architecture
Questions

les arguments de Fodor contre les modèles connectionnistes sont-ils valides

les réponses de Fodor aux arguments des connectionnistes sont-elles pertinentes

que penser de la première conclusion (les modèles connectionnistes sont inadéquats comme modèles de la pensée et du langage)

que penser de la seconde conclusion (les modèles connectionnistes qui sont adéquats ne sont que des implémentations des modèles classiques)