Computational Linguistics
A.Y. 2018/19 – C. Chesi

Lecture 14

NEURAL NETWORKS AND NLP

References

- **Essential references**
  - Elman (1999) *Origins of language: a conspiracy theory*. In MacWhinney "The emergence of language".
  - Christiansen & Chater (1999) *Toward a Connectionist Model of Recursion in Human Linguistic Performance*. Cognitive Science 23(2):157-205
- **Extended references**
  - Elman, Bates, Karmiloff-Smith, Parisi, Plunket (1996) *Rethinking Innateness*. MIT Press (Cap. 2)
  - Elman (1993) *Learning and development in neural networks: the importance of starting small*. Cognition 48:71-99
  - Rohde & Plaut (2001) *Less is less in language acquisition*. In Quinlin "Connectionist modelling of cognitive development".
  - Kandel, Schwartz, Jessell (1999) *Fondamenti delle neuroscienze e del comportamento*. Ambrosiana (Cap. 1-4)

Today

- Which problems are better approached by using sub-symbolic methods
  - Why certain problems better suit sub-symbolic approaches
- Neural networks
  - Neurophysiology
  - Artificial Neural Networks (ANN)
  - Past tense learning
- NLP with neural networks: the case of Simple Recurrent Networks (SRN)
  - Short term memory
  - Learning a language
  - Recursive properties acquisition with SRN

What "sub-symbolic" means

- Sub-symbolic approaches use **implicit representations**
  - **Dynamic Functional System** (Luria) every behavior, or high cognitive function, is the result of a concert of activation of specific cerebral areas
  - **Parallel Distributed Processing** (PDP, Rumelhart, McClelland & al.) knowledge (or *competence*, from a linguistic perspective) is not "*memory*" but a *processing module*
  - Only **input** and **output** are coded as *discrete/symbolic* entities;
  - No explicit structure or structure building operations;
  - **System complexity** (and its apparent representations) is an emergent property simple interaction among parts

Which problems?

Problems that can hardly be decomposed in sub-problems:

- Problems **complex** to be described
- **Partial representation** of problem space
- Complex algorithmic solutions that ask for **approximations**
- **Rules** and/or **heuristics** hard to be defined
- High degree on interaction among level (**multiple constraints**)

Competence and Grammar: two definitions

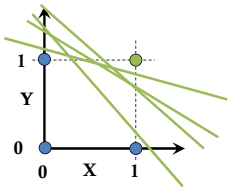
- **Symbolic** (e.g. phrase structure grammar)
  static set of rules and/or principles (explicit representation of the competence)
- **Sub-symbolic** (e.g. neural networks)
  Grammar is a processing device (implicit representation) reacting to contextual input; words are «operators» smoothly moving the system from state to state

An example

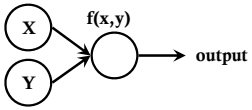
Logical operators definition

- AND from a symbolic perspective:

X	AND	Y
1	1	1
1	0	0
0	0	1
0	0	0

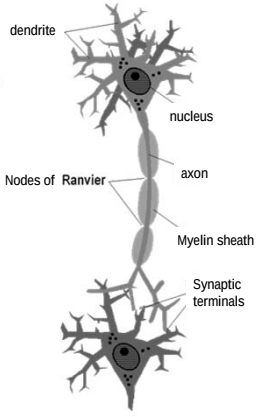


- AND from a sub-symbolic perspective:

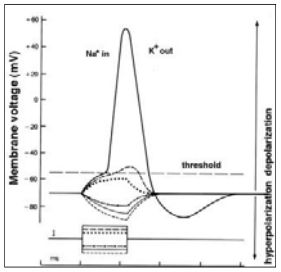


The Central Nervous System

Neurons:

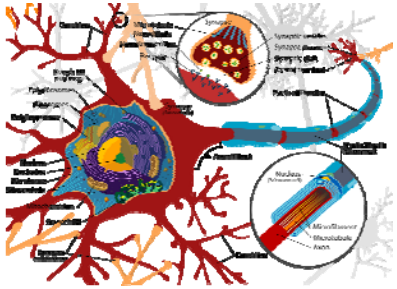


Synaptic potential:



The Central Nervous System

Synapses:

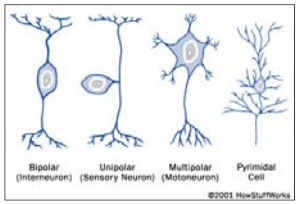


Synaptic transmission:

- **Electric** (cytoplasm continuity between neurons, direct ions transmission flow, no delay, bidirectional communication)
- **Chemical** (pre-synaptic vesicles active membrane sites + post-synaptic receptors, chemical transmitters, 0,3-5 ms delay or more, mono-directional)

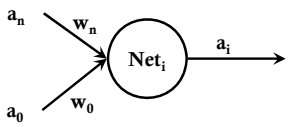
The Central Nervous System

Neuron typology:



Artificial neurons

- Nodes:



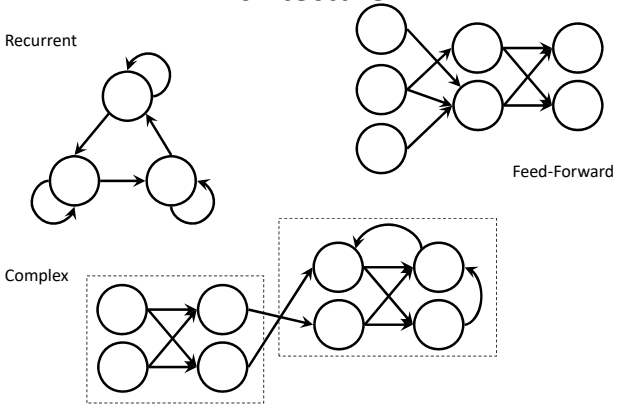
$$Net_i = \sum_{j=0}^{n} w_{ij} a_j$$

$$a_i = f(net_i)$$

- Connection:

- Fundamental idea: simple processing units linked by weighted connections. Their interaction might be extremely complex (emergent property); brain-like with respect to serial computers

Artificial Neural Network (ANN) Architecture



Classic ANN

- **Perceprton** (Rosemblatt)

- **Kohonen maps** (associative networks)

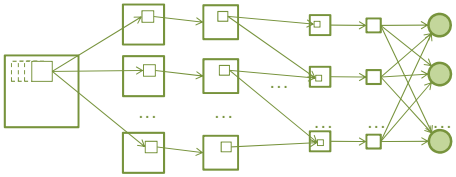
- **Pattern associator**

- **Multilayer networks**

Classic ANN

- **Recurrent networks** (Elman)

- **Convoluted networks** (Krizhevsky, Sutskever, Hinton)

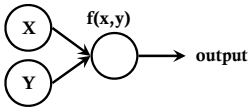


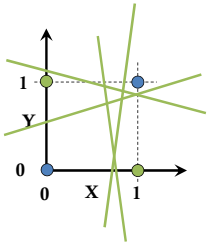
An «unsolvable» logical problem

Minsky & Papert 69:

XOR from a symbolic point of view

X	XOR	Y
1	0	1
1	1	0
0	1	1
0	0	0



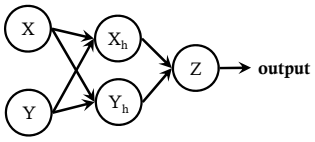


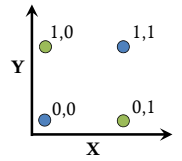
An «unsolvable» logical problem

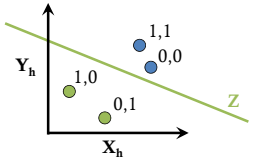
Minsky & Papert 69:

XOR from a symbolic point of view

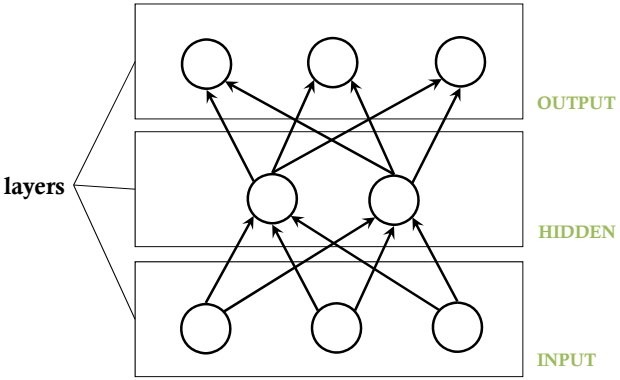
X	XOR	Y
1	0	1
1	1	0
0	1	1
0	0	0







Multilayers networks



Learning in ANN

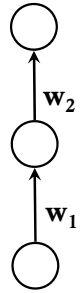
- **Supervised** learning
  we inform the network when the output is wrong or correct
- **Unsupervised** learning
  implicit learning, no information on the given/expected output
- **Hebbian learning** (Hebb 49)
  - "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth 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" (Hebb 1949:62)
  - $\Delta w_{ij} = \eta a_i a_j$
- **Perceptron Convergence Procedure** (PCP, Rosenblatt 59)
  - Calculate the difference between **real** and **expected output**.
  - Minimize error, **rewarding** or **punishing** weights on incoming connections

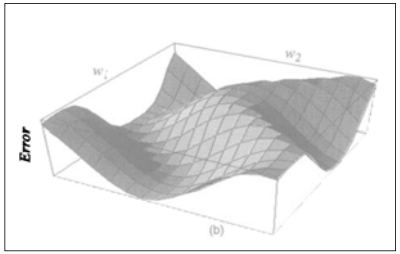
Learning in ANN

- **Back Propagation** (simplified)
  - Proportional redistribution of the errors back-ward, layer by layer, up to the input nodes
  - $\Delta w_{ij} = \eta \delta_{ip} o_{jp}$
    where
    i = out neuron
    j = in neuron
    p = activation pattern
    η = learning rate
    δ_{ip} = measure of error *i* with respect to pattern *p*
    o_{ip} = coefficient granting that the error is proportional to the activation received from *j*

Learning in ANN

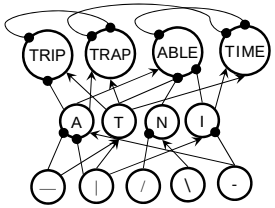
- Learning: descending gradient





How to select the best architecture?

- Heuristics are the best hint (Rumelhart & McLelland 1982)



- From a purely formal point of view, 3 layers can solve any problems (Hornik, Stinhcombe & White 1989), but how many neurons and which connection pattern?

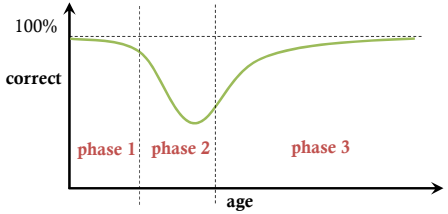
Coding input (and output)

- Information bits > number of input tokens (**distributed coding**)
  e.g. using a binary coding, we can use 2 bits, for representing 4 elements (a, b, c, d) that is, 2 input neurons (a=00, b=01, c=10, d=11)

- No similarity among inputs, orthogonal input (**localist coding**)
  1 word = 1 node
  e.g. 4 input units:
  a (0001) b (0010) c (0100) . (1000)

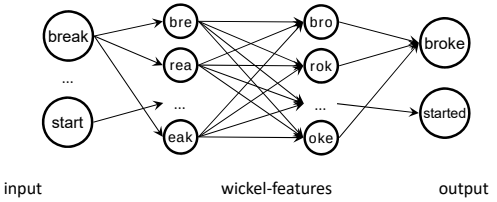
Learning linguistic properties

- Past tense (Rumelhart & McClelland, 86)
  - Clear linguistic pattern:
    - **phase 1**: few high frequency verbs (children leaned few crystallized forms)
    - **phase 2**: over-regularity (break > breaked)
    - **phase 3**: irregular verb inflection reconsidered (smooth coexistence of irregular and over-regular forms… until only correct regular forma are used)



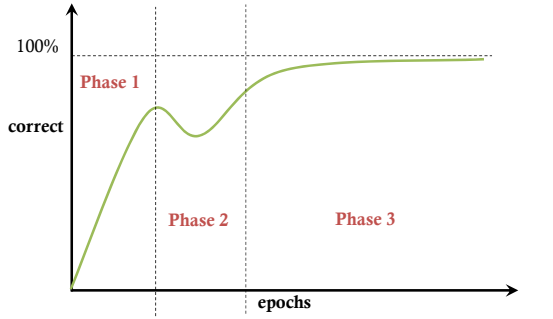
Learning linguistic properties

- Past tense (Rumelhart & McClelland, 86)
  - Human-like performance (some errors still present)
  - Phonetic input coding (Wickel-features, Wickelgren 69)
  - Network architecture (460 input and output units using wickel-features)



Learning linguistic properties

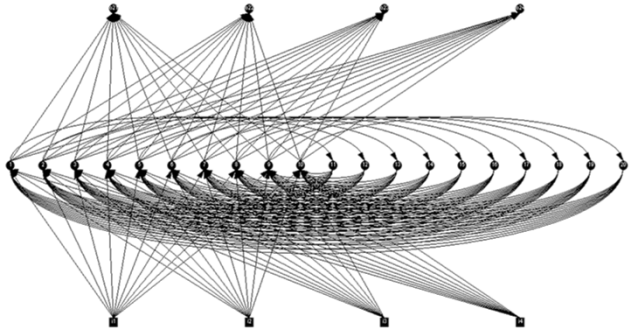
- Past tense (Rumelhart & McClelland, 86)
  - Results (after 420 turns of 200 verbs presentations = 84.000 examples)



Time in ANN

- Epochs (sweeps)
  (discrete) temporal units corresponding to one input processing
- Atemporal processing
  activation only depends on input, connections and weights
- Temporal flow simulation trick
  input divided in groups; each group a distinct temporal interval
- Context layer (Elman 1990)
  hidden layer activation is copied on a context layer that will be added to the next input activation

Simple Recurrent Networks



powered by **TLearn**

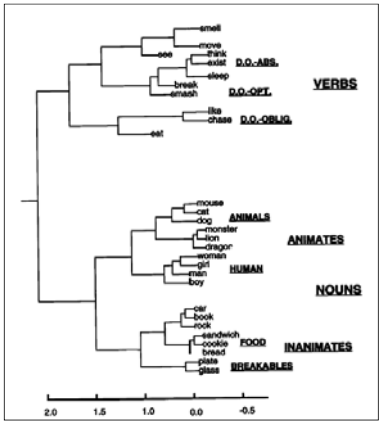
Simple Recurrent Networks

- Guessing next word paradigm
  e.g. *the house is red*
  Input = *the*
  Output = *house*

  («unsupervised» learning: **auto-supervised**)
- Psycho/neurological plausibility (Cole & Robbins 92)
  A sort of priming
- Input structure
  - *localist* (1 node = 1 concept)

Simple Recurrent Networks

- **Hierarchical clustering**: (implicit) knowledge representation
  - Comparing vector activation with respect to various inputs
  - Euclidean distance among vectors



Simple Recurrent Networks

The importance of **starting small** (Elman 93):

- **goal**: show how SRN can deal with complex and hierarchical linguistic information
- **hypothesis**: ANN learning is similar to L1 acquisition
- Linguistic properties to be learn:
  1. Subject-verb agreement
  2. Verbal subcategorization
  3. nesting

  e.g. *boys who chase dogs see girls*
  dogs see boys who cats who Mary feeds chase

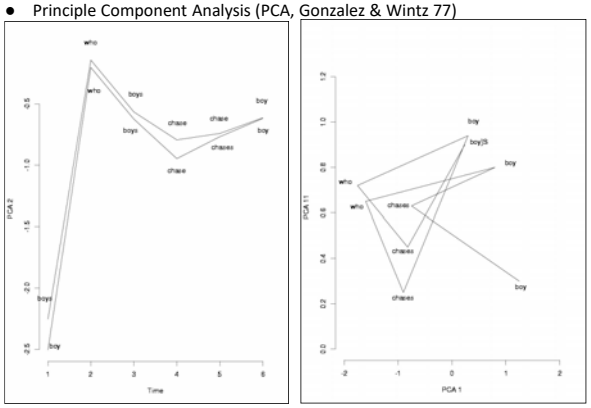
Simple Recurrent Networks

The importance of starting small (Elman 93):

- Everything since the beginning > learning failure (e.g. *the boys who the girl *chase...*)
- Complex regimen:
  1. 10.000 simple sentence (5 epochs)
  2. 7.500 simple sentence + 2.500 complex sentence (5 epochs)
  3. 50% simple sentence + 50% complex sentence (5 epochs)
  4. 25% simple sentence + 75% complex sentence (5 epochs)
  5. 10.000 complex sentence (5 epochs)
- **Limiting** memory the network learns anyway:
  1. Deleting contextual units every 3/4 words (12 epochs)
  2. Every 4/5 words (5 epochs)
  3. Every 5/6 words (5 epochs)
  4. Every 7 words (5 epochs)
  5. No deletion (5 epochs)

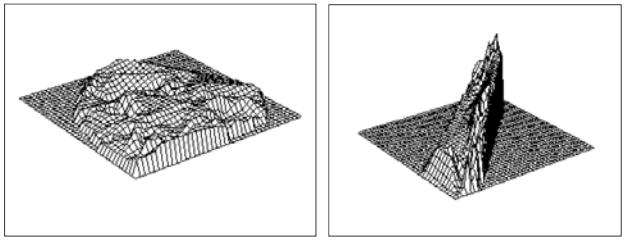
Simple Recurrent Networks

- Principle Component Analysis (PCA, Gonzalez & Wintz 77)



Simple Recurrent Networks

- How “lesioning” the network produces better results??



- Naif idea: focusing on simpler relations first, facilitate learning (greater flexibility at the beginning than at the end)

Simple Recurrent Networks

Less is less (Rohde & Plaut 01)
show that “starting small” is useless: SRN tend to learn first simpler (local) relations.

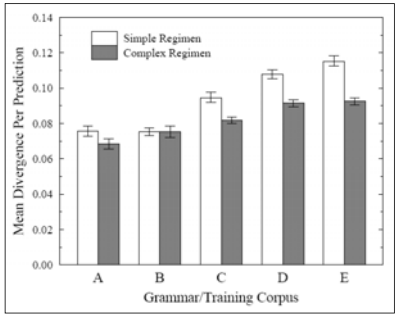
- Cleeremans & al. (89) show improved SRN learning with intervening semantic correlated (e.g. subj-verb agreement) (e.g. The dog [that **barks]** runs away)

test: **5 grammar classes**:

A. No semantic dependency between principal and embedded sentence
B. 25% of sentences with a semantic dependency
C. 50% of sentences with a semantic dependency
D. 75% of sentences with a semantic dependency
E. 100% of sentences with a semantic dependency

Simple Recurrent Networks

Less is less (Rohde & Plaut 01):



mean of the 16 training over 20 performed

ANN and recursive properties learning

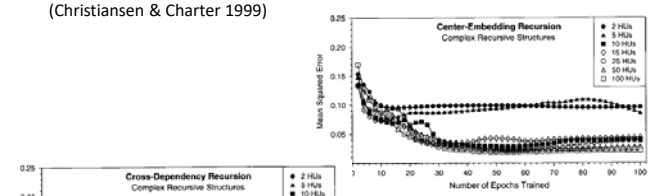
- Connectionist model of human **performance** (Christiansen & Charter 1999)
- Different from **competence idea**:
  e.g. Phrase Structure Grammars including recursive rules
  + (memory) limitations = **performance**
- Connectionist models include these limitations in a unique competence + performance processing device

ANN and recursive properties learning

- Summary of recursive properties in language
  - **Right-branching** (*iteration* rather than *recursion*)
    Mario vede il gatto <u>che</u> ha mangiato il topo <u>che</u> ha rubato il formaggio....
  - **Counting Recursion** ($a^n b^n$)
    <u>se</u> è vero che <u>se</u> Mario vince il concorso ... <u>allora</u> si sposa, <u>allora</u> anch'io ci faccio un pensierino
  - **Center embedding** (ww^R)
    il topo [che il gatto [che Mario ... vede] ha mangiato] ha rubato il formaggio
  - **Cross-serial dependencies** (ww)
    Mario, Giovanna, Giuseppe ... sono rispettivamente promosso, bocciata, promosso ...

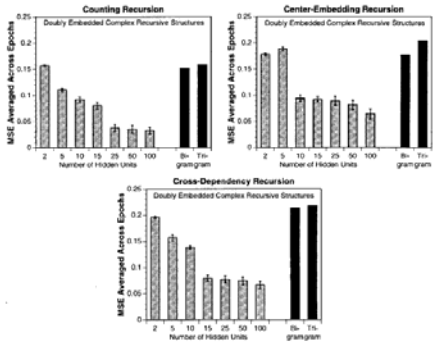
ANN and recursive properties learning

- Learning results based on different numbers of nodes in the hidden layers (Christiansen & Charter 1999)



ANN and recursive properties learning

- Learning results for different recursive properties (Christiansen & Charter 1999)



ANN and innateness

- Different kinds of «innateness» (Elman 99)
  - **representational**
    different connection patterns -> different activation states
  - **architectural**
    - unit-based (neurons typology, activation function...)
    - Local architectural constraints (density, local circuits...)
    - Global architectural constraints (constraint on different areas...)
  - **timing** (innate chronotopy)
    structural modification due to external factors (neural modified plasticity, different input processed...)

Next lecture

- **Lab (please bring your own laptop)**
  - **T-learn**
    - Neural network simulation
    - Input coding
    - training set and learning
    - Output analysis
    - Principal Component Analysis and Cluster Analysis