Knowledge Processing with Neural Networks

Lecture 12: Revision

http://www.informatik.uni-hamburg.de/WTM/

Neural Computing: The brain as the most exciting inspiration for computing

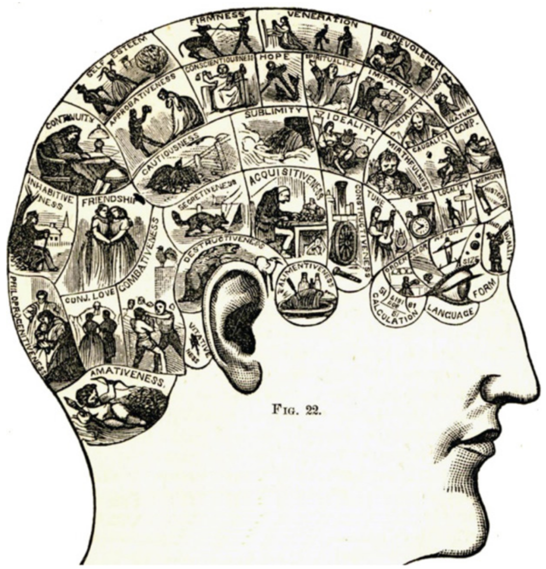


FIG. 22.

“Man is still the most extraordinary computer of all. “ [John F. Kennedy 1963]

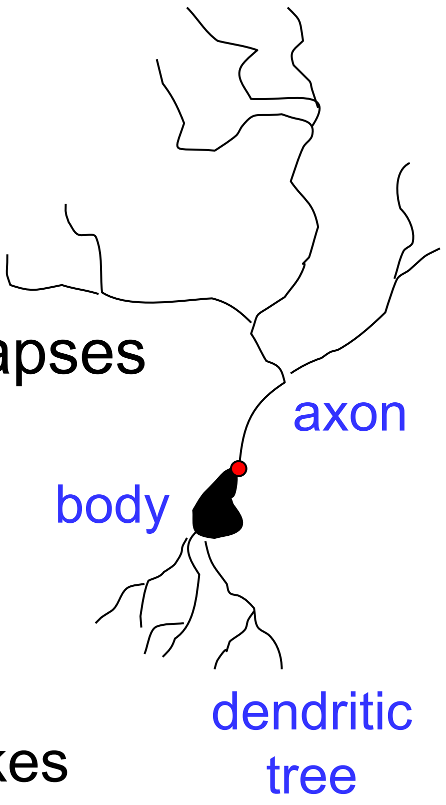
but only because of neural computing

The goals of neural computation

- To understand how the brain actually works
- To understand a new style of computation
  - Inspired by neurons and their adaptive connections
  - Very different style from sequential computation
    - should be good for things that brains are good at (e.g. vision)
    - Should be bad for things that brains are bad at (e.g. 23 x 71)
- To solve practical problems by developing novel learning algorithms
  - Learning algorithms can be very useful even if they have nothing to do with how the brain works

A typical cortical neuron

- Gross physical structure:
  - There is one axon that branches
  - There is a ***dendritic*** tree that collects input from other neurons
- Axons typically contact dendritic trees at synapses
  - A spike of activity in the axon causes charge to be injected into the post-synaptic neuron
- Spike generation:
  - There is an ***axon*** that generates outgoing spikes whenever enough charge has flowed in at synapses to depolarize the cell membrane



Three kinds of learning

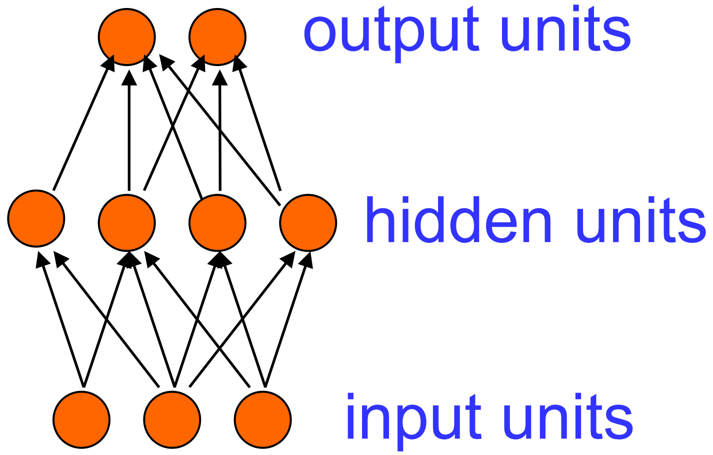
- Supervised Learning: ***this models p(y|x)***
  - Learn to predict a real valued output or a class label from an input.
- Reinforcement learning: ***tries to be successful at the end***
  - Choose actions that maximize payoff
- Unsupervised Learning: ***this models p(x)***
  - Build a causal generative model that explains why some data vectors occur and not others

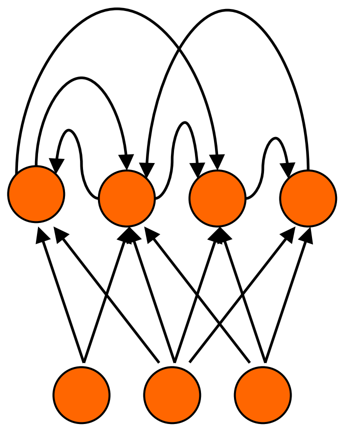
    or

  - Discover interesting features; separate sources that have been mixed together; find temporal invariants etc. etc.

More sophisticated Networks: Types of connectivity

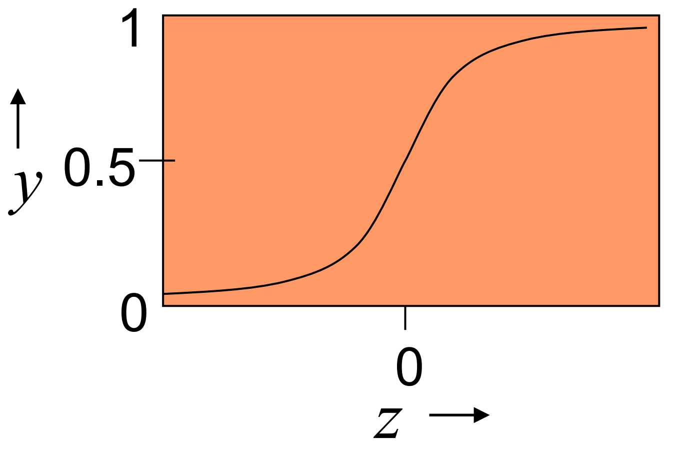
- Feedforward networks
  - These compute a series of transformations
  - Typically, the first layer is the input and the last layer is the output.
- Recurrent networks
  - These have directed cycles in their connection graph. They can have complicated dynamics.
  - More biologically realistic.





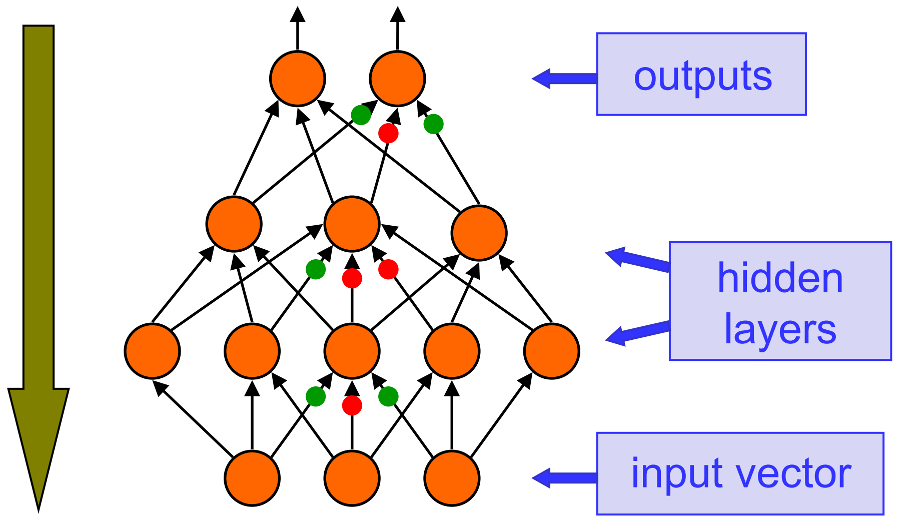
Learning by back-propagating error derivatives

- Sigmoid neurons:
  - Smooth and bounded real-valued output of their total input.
  - Typically they use the logistic function
- Back-propagation:
  - Compare outputs with ***correct answer*** to get error signal
  - Back-propagate error signal to get ***derivatives*** for learning



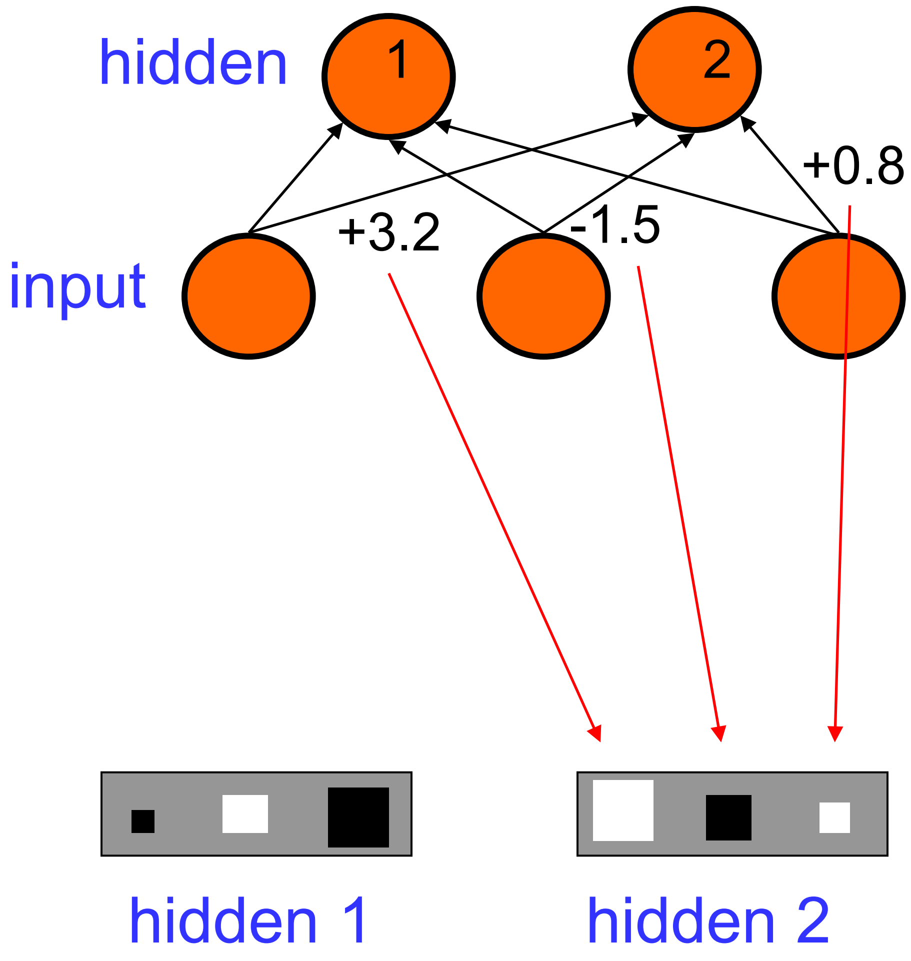
$$z = b + \sum_i x_i w_i$$

$$y = \frac{1}{1+e^{-z}}$$



How to show the weights of hidden units

- The obvious method is to show numerical weights on the connections:
  - Try showing 25,000 weights this way!
- Its better to show the weights as black or white blobs in the locations of the neurons that they come from
  - Better use of pixels
  - Easier to see patterns



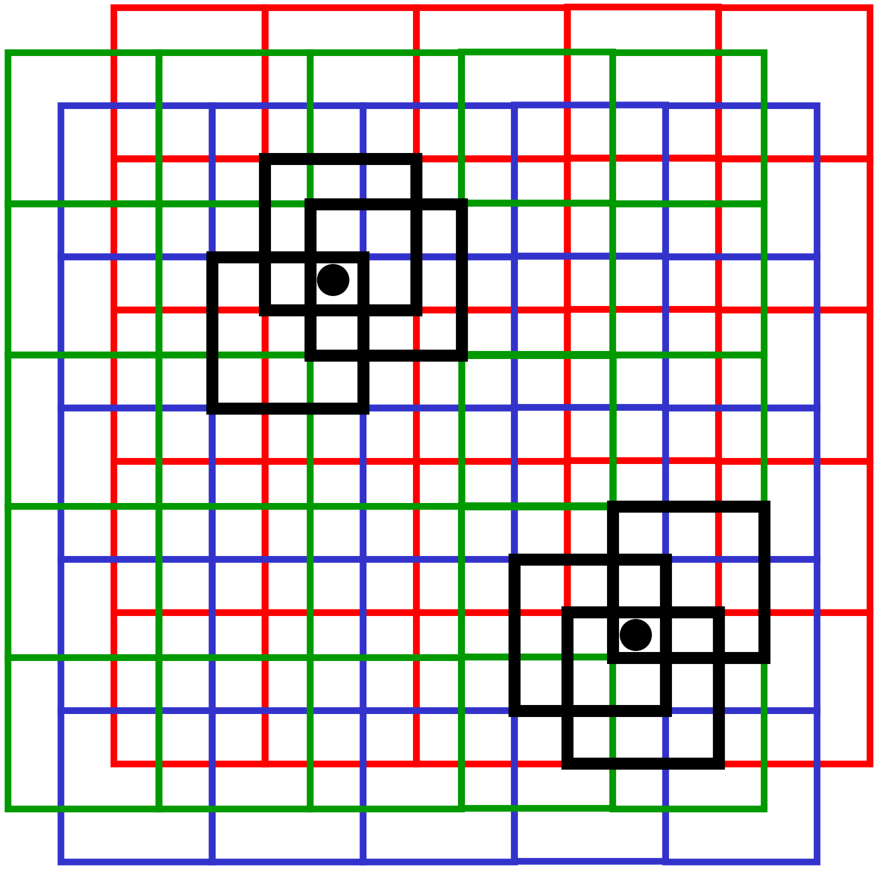
Localist and distributed representations

- Localist: simplest way to represent things with neural networks is to dedicate ***one neuron to each thing***.
  - Easy to understand, code by hand, and learn,
  - Easy to associate with other representations or responses,
  - Inefficient whenever the data has componential structure.

- Distributed: ***many-to-many*** relationship between two types of representation (such as concepts and neurons).
  - Each concept is represented by many neurons,
  - Each neuron participates in the representation of many concepts.

Coarse coding

- Use three overlapping arrays of large cells to get an array of fine cells
  - If a point falls in a fine cell, code it by activating 3 coarse cells.
- This is more efficient than using a neuron for each fine cell.
  - It loses by needing 3 arrays
  - It wins by a factor of 3x3 per array
  - Overall it wins by a factor of 3



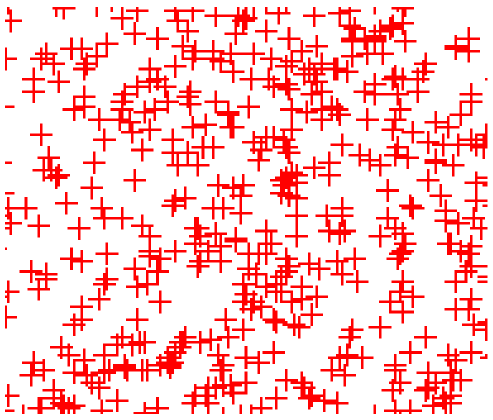
PCA and clustering

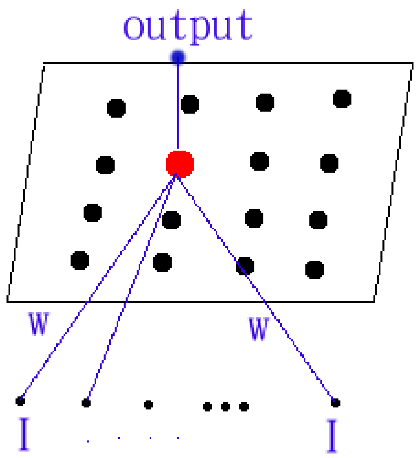
- ***Principal Components Analysis*** is
  - Powerful: uses distributed representations
  - Limited: representations are linearly related to the data
    - Autoencoders with more hidden layers are not limited this way.
- ***Clustering*** is
  - Powerful: uses very non-linear representations
  - Limited: representations are local
- We need representations that are both distributed and non-linear
  - ⇨ Unfortunately, typically very hard to learn.

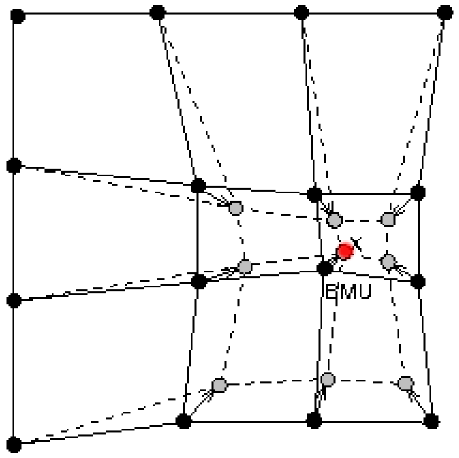
	Local	Distributed
Linear		PCA
Non-linear	clustering	what we need

Self-organizing Map

- One neural method of clustering was proposed by Kohonen
- His idea was inspired by the way that mappings are learnt in the topographic feature maps found in many brain areas
- These ***feature maps*** consist of one- and two-dimensional sheets of neurons with lateral connections


Raw Data

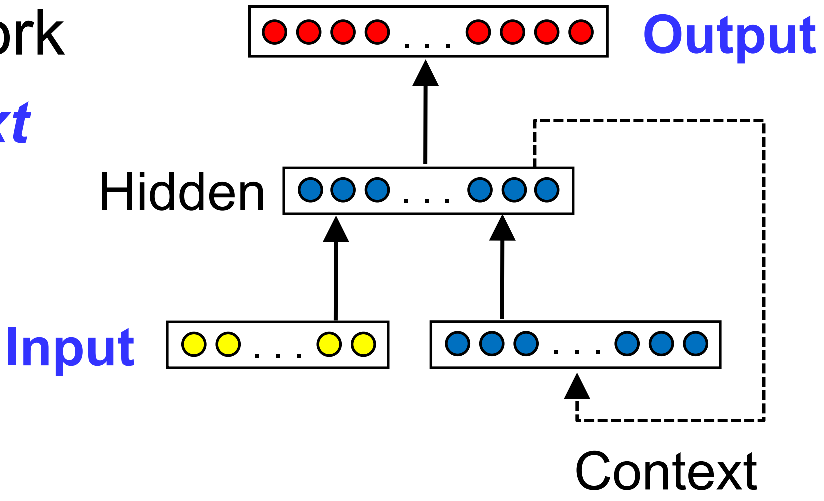




Recurrent Neural Networks

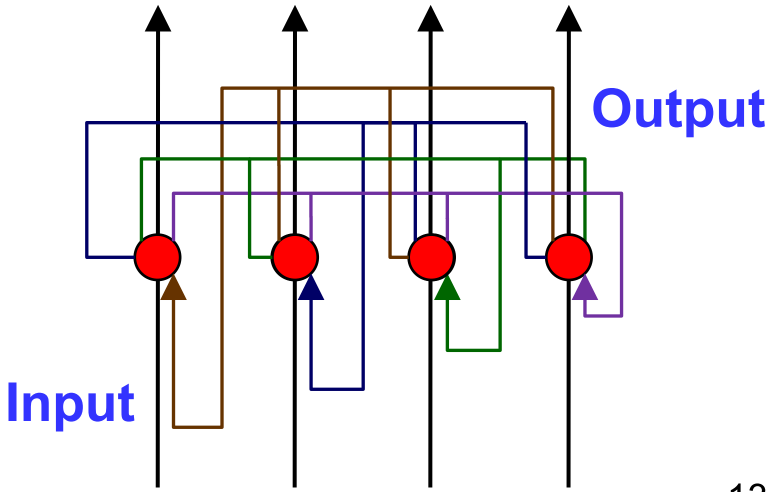
- Simple Recurrent Neural Network
  - Previous activation add ***context*** to the current activation

  Examples:
  - Elman Network
  - Jordan Network



- Fully Connected Neural Network
  - Often called ***auto-associator***

  Examples:
  - Hopfield Network (binary)
  - Boltzmann machine (stochastic)

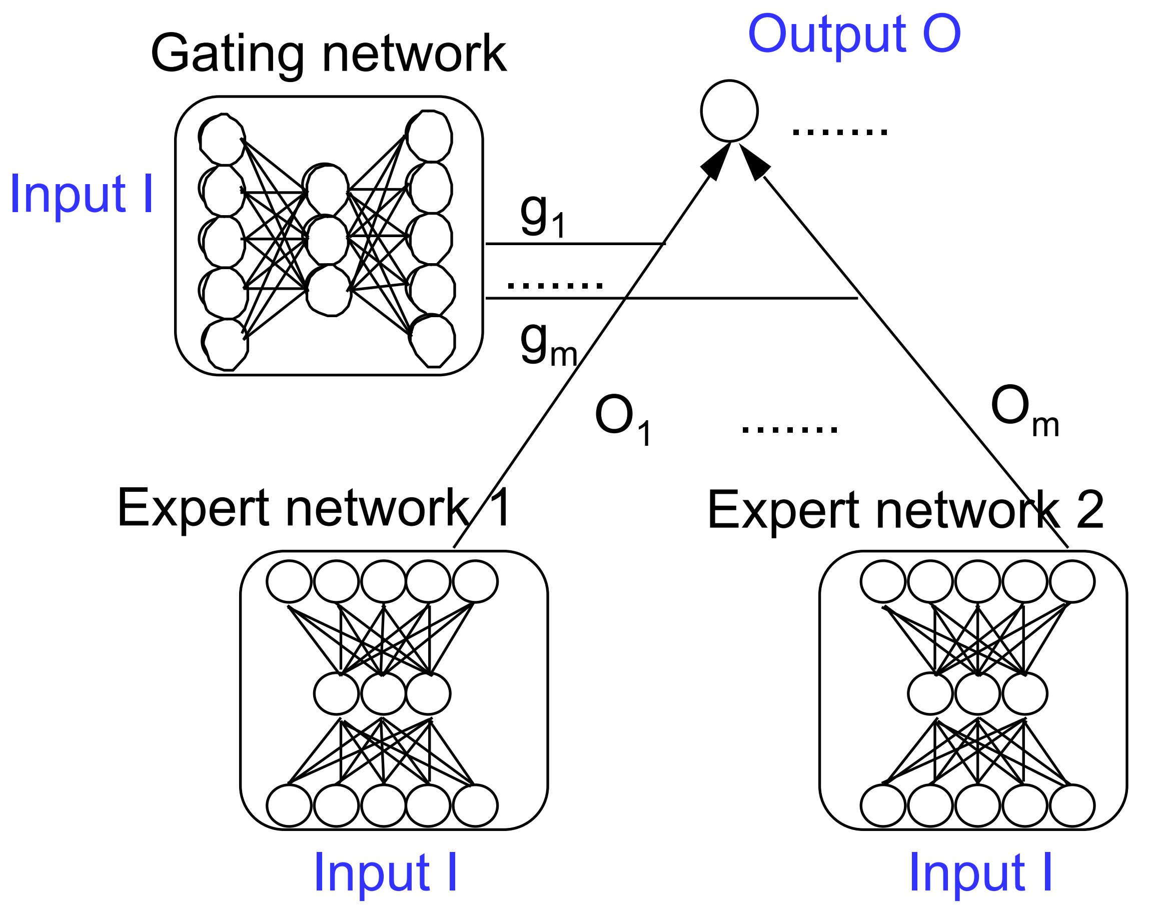


Uses of recurrent neural networks

- They can remember things for a long time.
  - The network has ***internal state***. It can decide to ignore the input for a while if it wants to.
  - But it is very hard to train a recurrent net to store information that is not used until a long time later; better for ***local context***

- They can ***model sequential data*** in a much more natural way than by using a fixed number of previous inputs to predict the next input
  - The hidden state of a recurrent net can carry along information about a ***potentially unbounded number of previous inputs***.

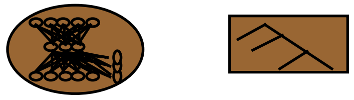
Gating: combining expert networks



Different types of hybrid architectures

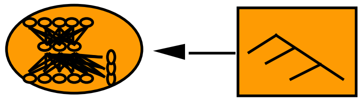
Hybrid transfer architectures

Symbolic representation
of neural representation



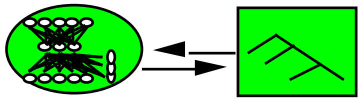
Hybrid processing architectures for hybrid realization of symbolic structures

Loose coupling



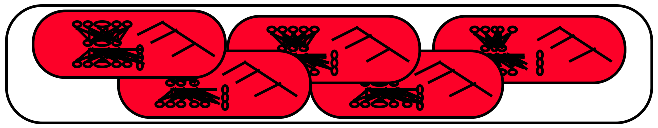
Symbolic and neural modules separate
and unidirectional communication

Tight coupling



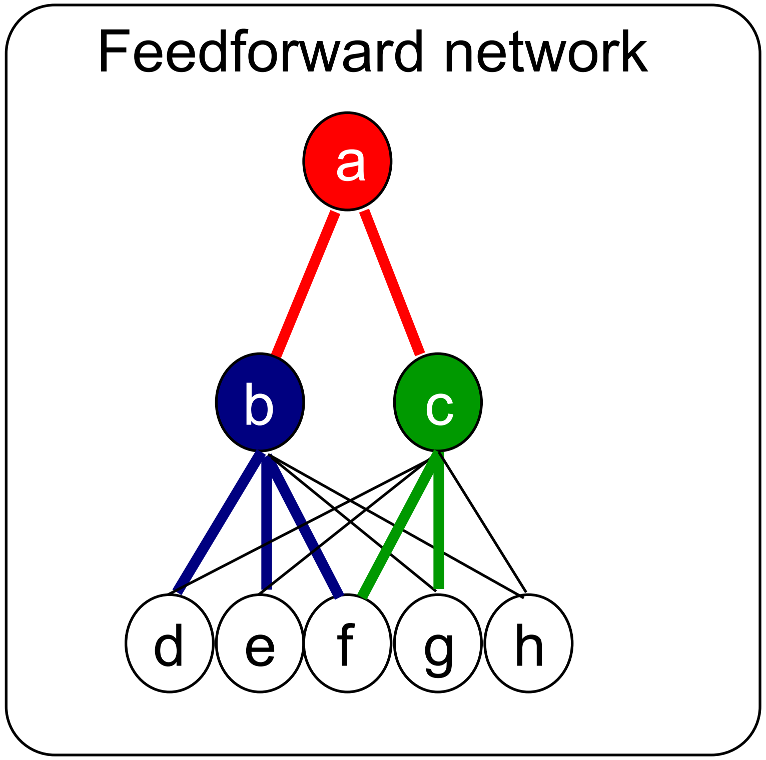
Symbolic and neural modules
separate and bidirectional communication

Integration



Symbolic and connectionist modules fully
embedded and integrated

Rule extraction in a hybrid transfer architecture



Symbolic rules

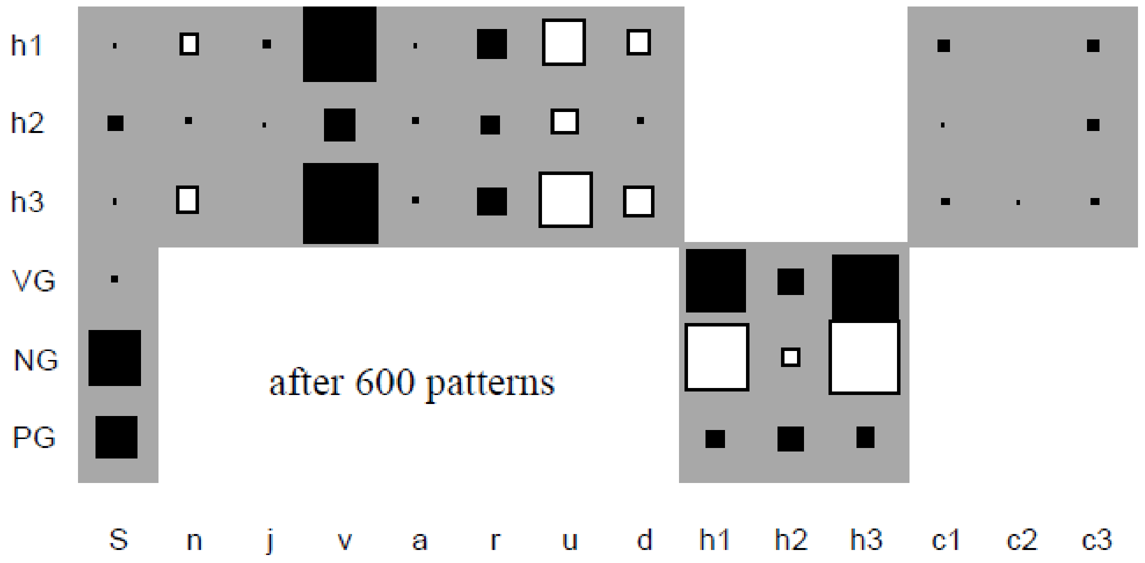
a :- b , c

b :- d , e , f

c :- f , g

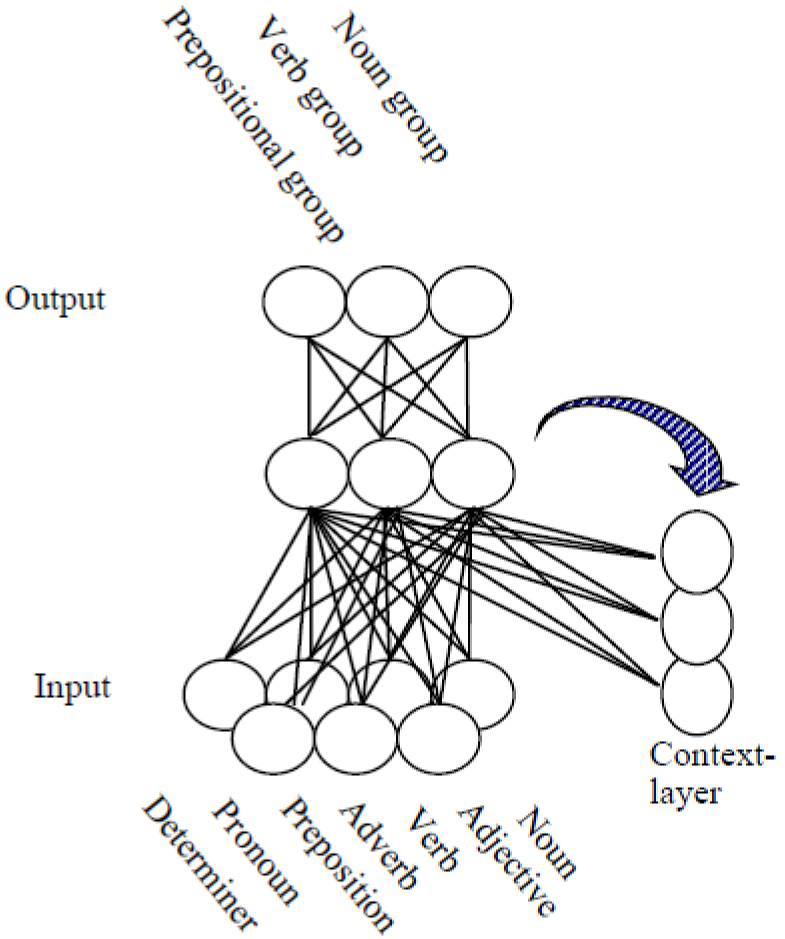
Weight analysis for knowledge extraction

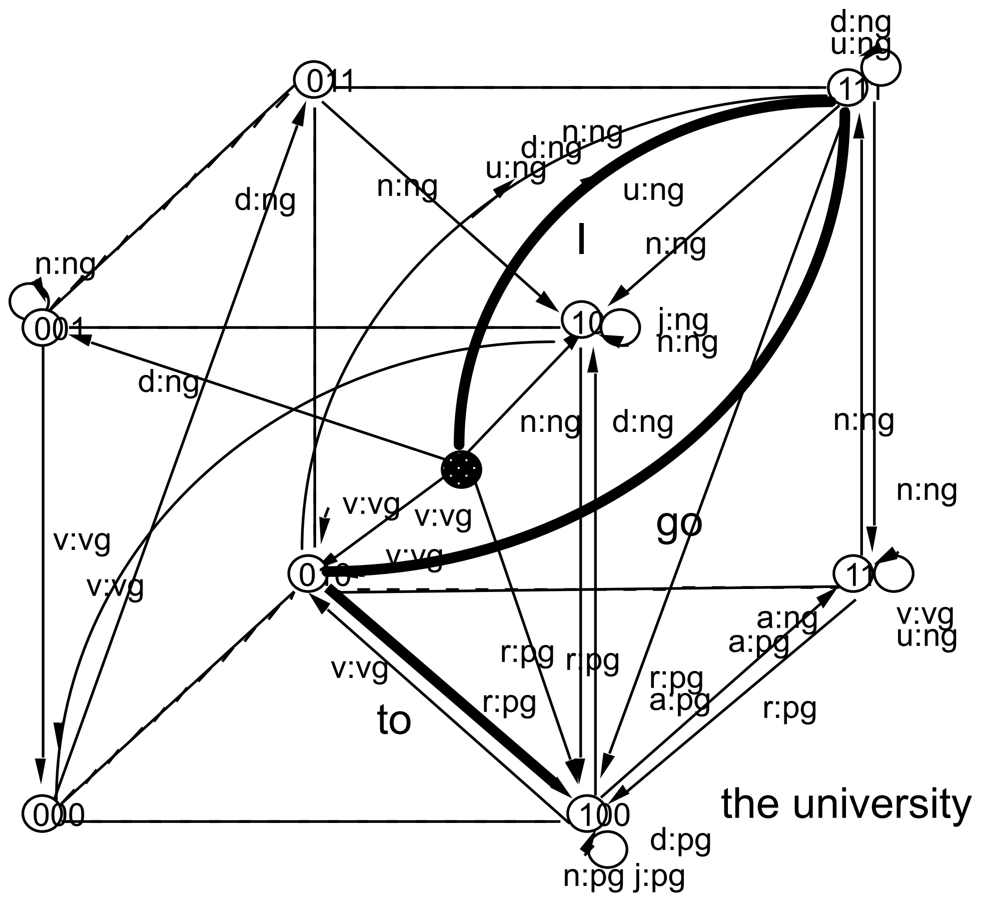
- Identifiers of the source “neurons” shown horizontally and identifiers of the goal neurons vertically.
  - Left to right: neuron (*S*), neurons for syntactic categories (*n,j,v,a,r,u,d*), internal neurons (*h1,h2,h3*), and context neurons (*c1,c2,c3*).
  - Top to bottom: internal elements (*h1, h2, h3*), and output elements representing the abstract syntactic categories (VG, NG, PG).



White boxes represent positive ***weights***, black boxes negative ***weights***; NG and VG patterns learned

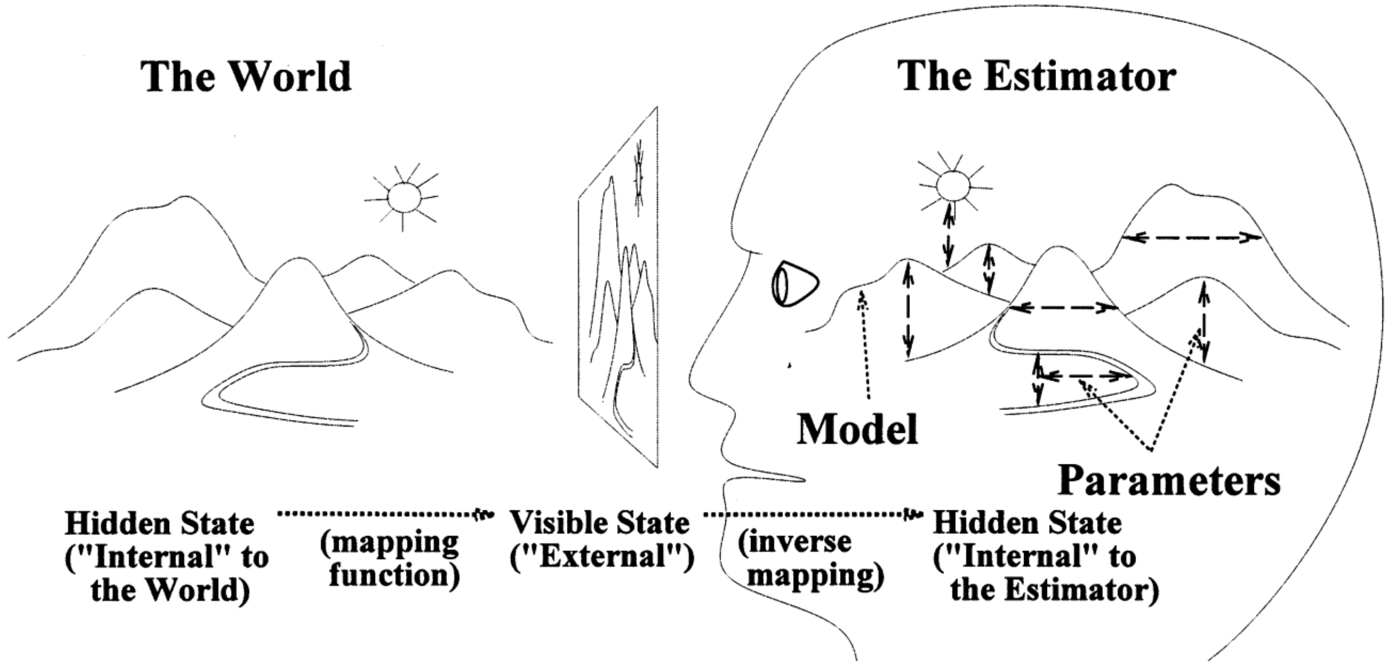
Knowledge extraction from transducer networks





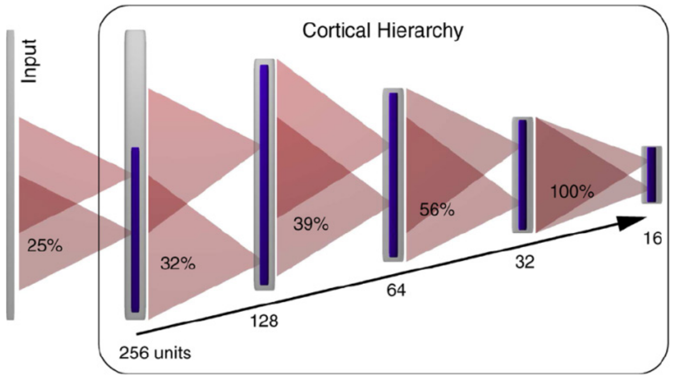
Generative models

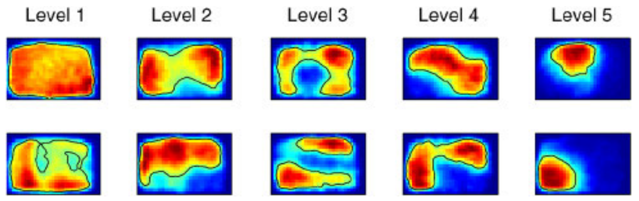
- Hierarchical visual system with growing abstraction
- Weight matrices transform the representations

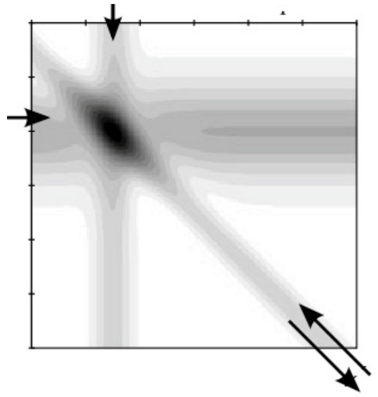

The World
The Estimator
Model
Parameters
Hidden State ("Internal" to the World)
(mapping function)
Visible State ("External")
(inverse mapping)
Hidden State ("Internal" to the Estimator)

Visual information routing

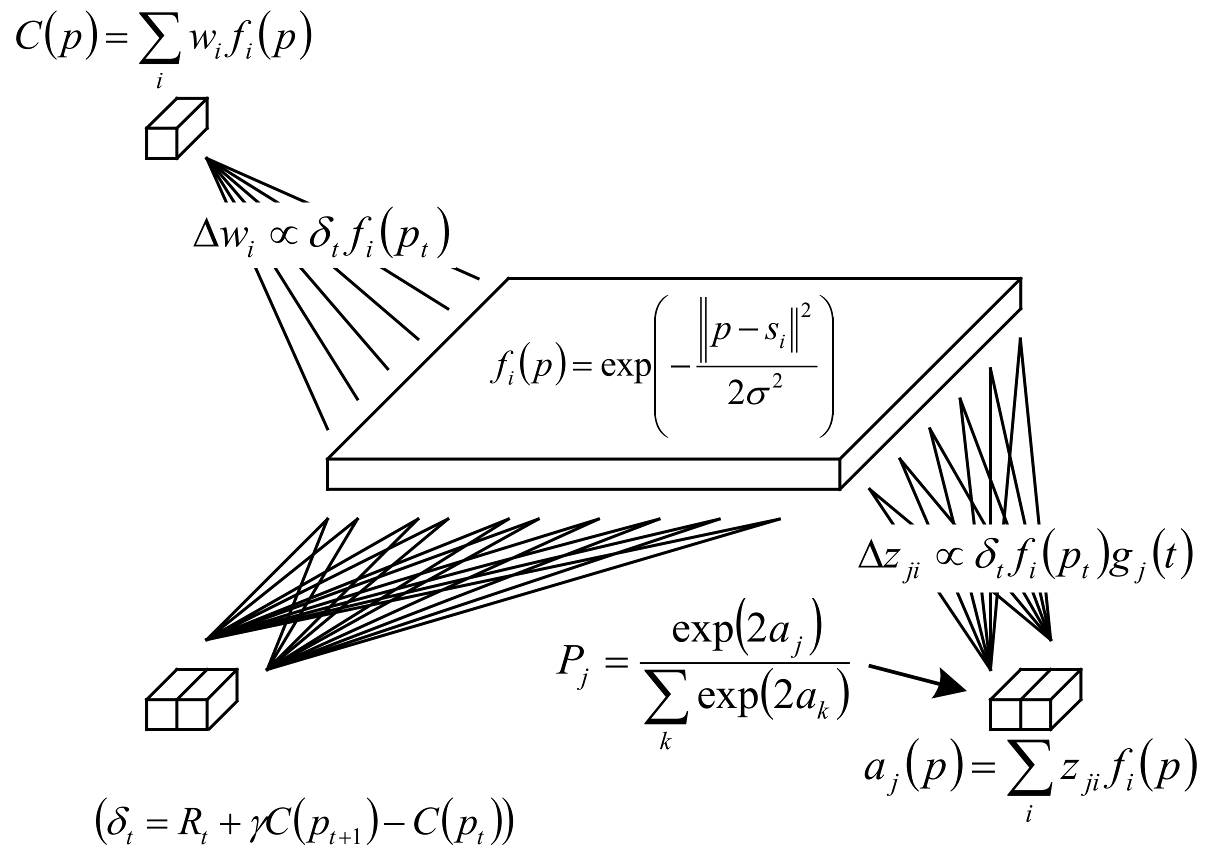
- Invariant representations
  - Inspiration: ***Place Cells***
  - Robust against transformations:
  - shift, scale, rotation, shape, ...
  - Information loss: no strict generative model
- "***What***" & "***Where***" pathways
  - Can be achieved by various models: dynamic routing, oscillation models predictive remapping, neural fields
  - Often non-linearities required for modulation/control







Guided actor-critic reinforcement learning algorithm


$C(p)=\sum_i w_i f_i(p)$
$\Delta w_i \propto \delta_t f_i(p_t)$
$f_i(p)=\exp\left(-\frac{\|p-s_i\|^2}{2\sigma^2}\right)$
$\Delta z_{ji} \propto \delta_t f_i(p_t) g_j(t)$
$P_j = \frac{\exp(2a_j)}{\sum_k \exp(2a_k)}$
$a_j(p)=\sum_i z_{ji} f_i(p)$
$(\delta_t = R_t + \gamma C(p_{t+1}) - C(p_t))$

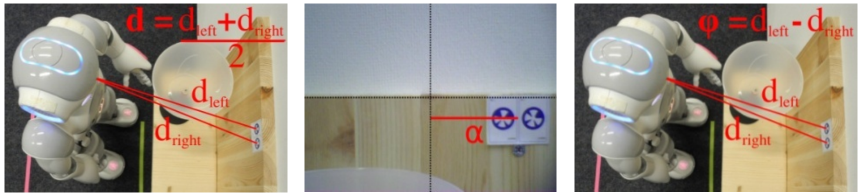
Rules for guidance:

If no reward was received from the critic:

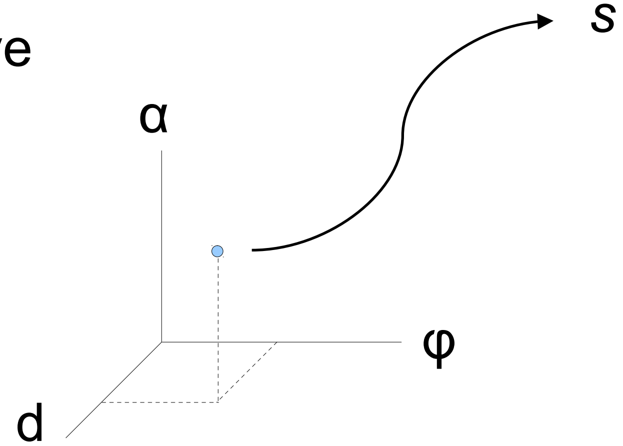
- If the agent moved closer to the goal state the Actor was rewarded
- Otherwise the Actor was punished

SARSA reinforcement learning

(1) Check where we are, also determine active state *s* and check whether we have any feedback *r* (reward or punishment)



$d = \frac{d_{left} + d_{right}}{2}$
d_{left}
d_{right}
α
$\varphi = d_{left} - d_{right}$
d_{left}
d_{right}



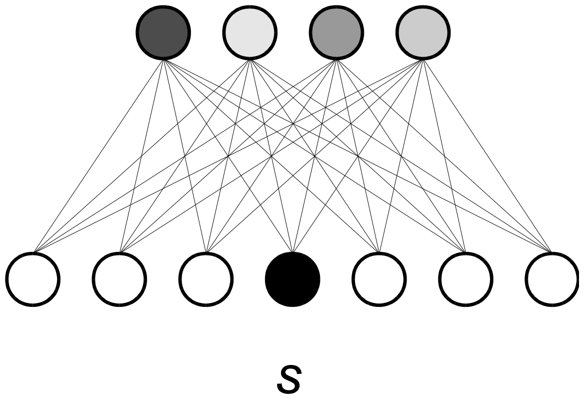
(2) Compute action strength

(3) Select action (soft-max)

(4) Current estimate

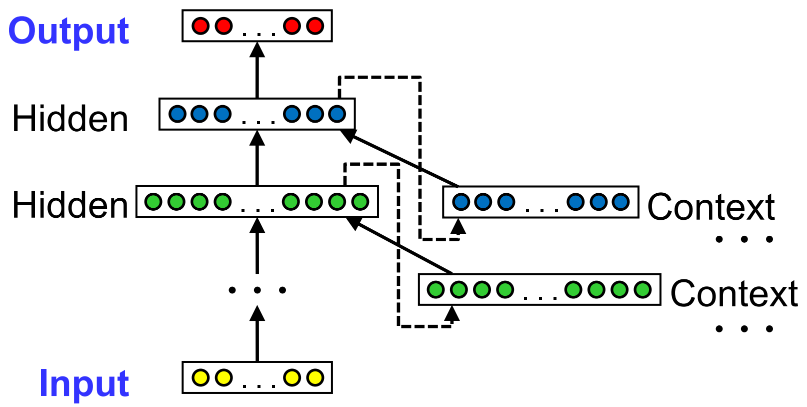
(5) Prediction error

(6) Weight update



Advanced recurrent neural architectures

- ***Arbitrary number*** of hidden & ***context layers***
  - Reflects short-context and larger-context memory
  - Very robust against noise
  - Can capture the important context independent of when it occurs in a sequence

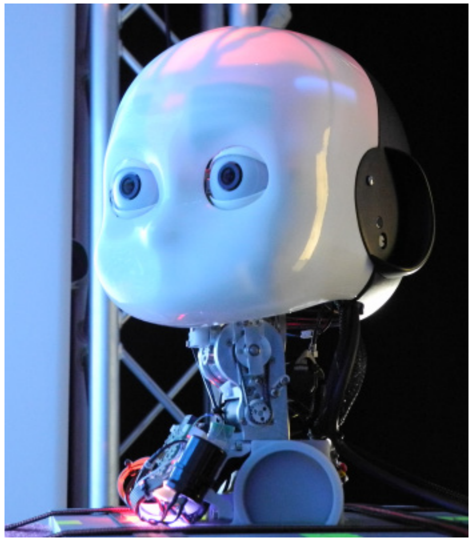


- Additional nodes which self organise a ***bias*** for a sequence
  - Network generates nonlinear mappings between the parametric bias and corresponding sequences
- Multiple context layers with ***different timescales***
  - Self organising of different hierarchical dynamics of sequences

Knowledge Processing with Neural Networks: Concluding remarks

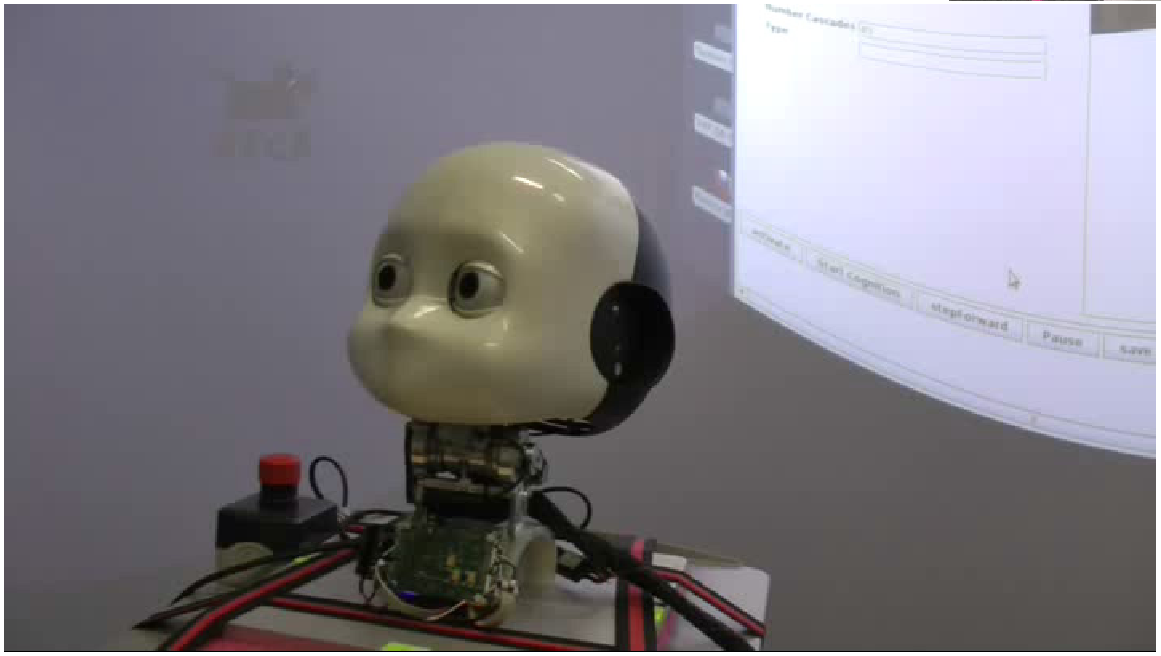
- ***Inspired*** by the brain we discovered ***neural computing***
  - Neural network ***architectures***
  - ***Learning*** with neural networks
  - ***Hybrid*** neural systems
  - Neural models for various ***cognitive capabilities***

- ... and learned to apply these capabilities to artificial intelligent systems
  - Effective ***classification*** and ***approximation***
  - ***Adaptive*** and robust learning robots



Adaptive and robust learning robots: Neural computing in the lab

Predictive docking and grasping with neural reinforcement learning



Face recognition and tracking with recurrent neural networks

Feedback on the seminar

- Be crisp and stay in time limit
- Get quick into the main research issue
- About 2 minutes for slide
- Most important ‘medium’ for the talk: the speaker!
  - Slides are just a tool to support the talk
  - Focus on the audience and avoid the wall
- Use examples, videos, illustrations, metaphors, and gestures to get your points across
- Talk freely, be confident, and have fun!

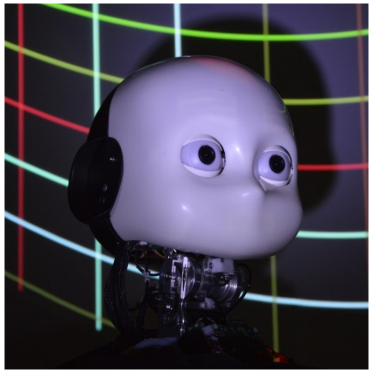
Reminder: Paper submission deadline was yesterday!

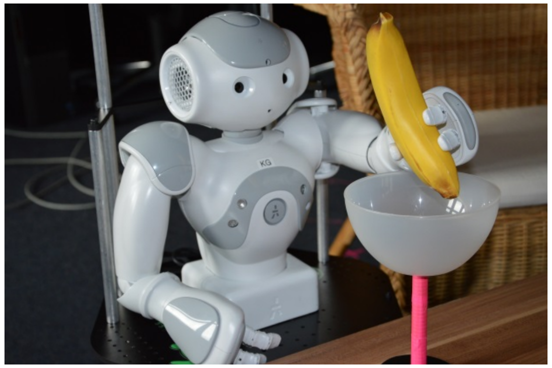
More info

- http://www.informatik.uni-hamburg.de/WTM/
- Exam: 16.07., 02.10.
  people from KPNN ot this term have preference
- Upcoming courses:
  - *Lecture* + *Seminar*: Bio-inspired Artificial Intelligence
  - *Master Project*: Human-Robot Interaction
  - *Additionally*: Oberseminar Knowledge Technology
    http://www.informatik.uni-hamburg.de/WTM/teaching/
- BSc or MSc thesis
- ***Please fill the form***

Upcoming courses …

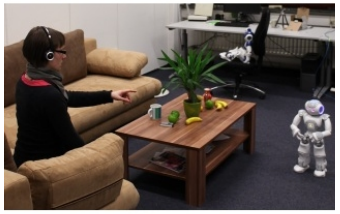
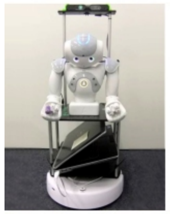
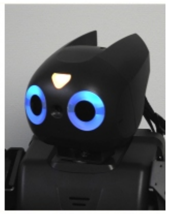
- L+S Bio-inspired Artificial Intelligence
  - Adaptation, learning, development, evolution!
  - Learn about the nature and human!
  - Learn about brain and mind!
  - Experience how to build intelligent systems and robots!



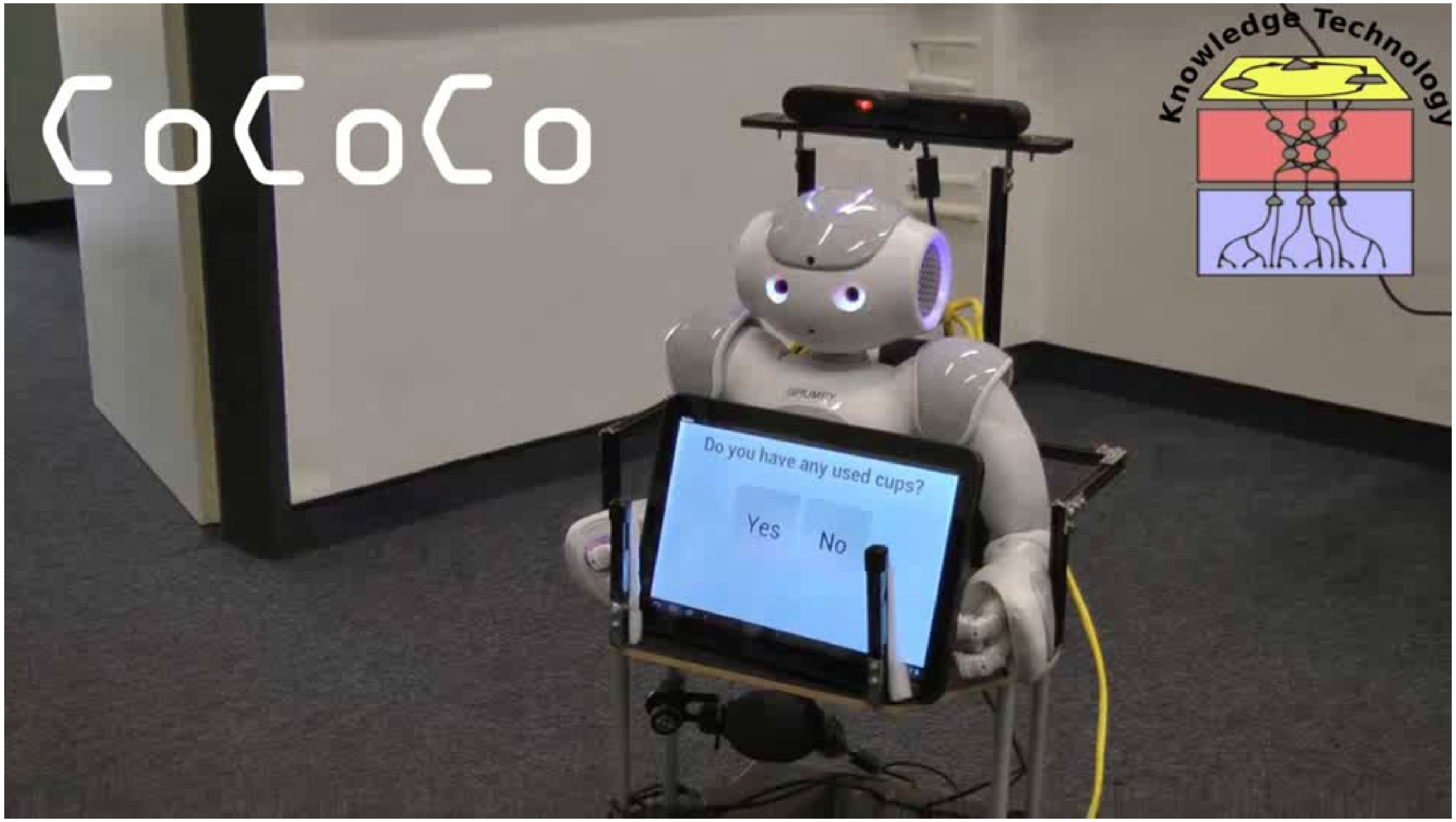


... group projects …

- MSc Project Human-Robot Interaction
  - Challenge: Robotic device capable ***of interacting with people*** as naturally as we interact with each other
  - Approach: solve a ***simple task*** in a ***complex environment***, e.g. “Serve coffee!”
  - Inspiration: RoboCup@home tasks
  - Chance: Follow up on award-winning ideas and environments of the recent student groups

– Result of last years project –



- Student project with Anne Rubruck, Azad Aminian, Jyothi Yalpi, Paul Hanzal, Sascha Winde, Sathya Kuppusami, Sohaib Younis, & Stefan Thomas
- Video is nominated for best robot video award at the AAAI 2014 video competition

… and topics for later MSc Projects

- Check for current offers: http://www.informatik.uni-hamburg.de/WTM/teaching/suggested_topics_titles.shtml
- Of course, feel free to discuss your own ideas with us
- Or contact your WTM tutors:
  heinrich@informatik.uni-hamburg.de
  jirak@informatik.uni-hamburg.de
  magg@informatik.uni-hamburg.de
  navarro@informatik.uni-hamburg.de
  weber@informatik.uni-hamburg.de
- *Additional*: Oberseminar Knowledge Technology
  http://www.informatik.uni-hamburg.de/WTM/teaching/