IMT Institute for Advanced Studies, Lucca

Lucca, Italy

The Modularity of Attention from an Artificial Intelligence perspective

PhD Program in Computer Science and Engineering

XXV Cycle

By

Nicola Catenacci Volpi

2013

The dissertation of Nicola Catenacci Volpi is approved.

Program Coordinator: Prof. Rocco De Nicola, IMT Institute for Advanced Studies, Lucca, Italy

Supervisor: Prof. Paolo Ciancarini, University of Bologna, Italy

Supervisor: Researcher Giovanni Pezzulo, ISTC/CNR, Rome, Italy - ILC/CNR, Pisa, Italy

Tutor: Prof. Marzia Buscemi, IMT Institute for Advanced Studies, Lucca, Italy

The dissertation of Nicola Catenacci Volpi has been reviewed by:

Prof. Antonio Chella, University of Palermo, Italy

,

IMT Institute for Advanced Studies, Lucca

2013

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my acknowledgements to the co-authors who contributed in writing the papers at the basis of this thesis: Giovanni Pezzulo, Jean-Charles Quinton and Laura Barca.

Vita

April 10, 1983	Born, Rome, Italy
2006	Bachelor degree in Computer Science Final mark: 103/110 University of Rome, La Sapienza, Italy
2009	Master degree in Computer Science Final mark: 110/110 University of Rome, La Sapienza, Italy

Publications

1. D. Ognibene, N. C. Volpi and G. Pezzulo "Learning to grasp information with your own hands," in *Procceeding of Towards Autonomous Robotics Systems (TAROS2011, LNCS)*, 2011, Springer.
2. J. C. Quinton, N. C. Volpi, L. Barca and G. Pezzulo "The cat is on the mat. Or is it a dog? Dynamic competition in perceptual decision making," in *IEEE Transactions on Systems Man and Cybernetics: Systems (in press)*.
3. D. Ognibene, N. C. Volpi, G. Pezzulo and G. Baldassarre "Learning Epistemic actions: a Model-Free, Memory-Free Reinforcement Learning approach," in *Proceedings of the 2nd International Conference on Biomimetic and Biohybrid Systems LIVING MACHINES 2013 (in press)*.

Abstract

The development of agents with bounded rationality is still an important challenge of artificial intelligence. Indeed, when we are facing problems with a large number of states, if we do not reason about the computational resources of our agent, it is easy to encounter exponential complexities or state explosions.

One way to solve this problem is taking inspiration from the study of attention in cognitive science. Attention can be considered as a filter in information processing that focuses only on relevant information, leaving out possibly all the useless computations. So to be focused only on a relevant subsets of the state space of a problem can be the solution to increase the quality of our algorithms. Accordingly it is important to understand how to represent relevance and being able to compute automatically what is relevant in a given situation.

In this thesis we link the study of attention with multi agent systems (MAS). The introduced methodology starts finding a way to partition an input problem. Then it specializes the agents of a MAS on the partition's subsets. Finally it finds a policy to switch the agents on or off according to the current context. This will be done allocating computational resources to the agents during the task execution.

Accordingly we will introduce two context aware methodologies based on Hopfield networks and dynamical neural fields to learn, store and recall online resource-allocation policies. This will lead to a dynamical characterization of such policies (i.e., attractors).

Finally the systems will be evaluated in distributed constraint optimization, classification and categorization tasks, underlining, when was possible, the cognitive plausibility of our proposals. Indeed the categorization task will be held in an active vision framework linking, also experimentally, our proposals with the study of attention in animal and human vision.

Chapter 1

Introduction

1.1 Bounded Rationality and Metareasoning

In artificial intelligence decision making is usually related with the maximization of a measure of expected utility in a context of complete knowledge and unlimited computational resources. The underlying formulation comes from models of microeconomics, decision theory and game theory (VNM47) (Hem65) (Ber95) (Put94) that consider an agent rational if, given a complete belief/desire system, it always optimizes its choices.

To satisfy such perfection of appropriate decisions and inferential machinery is not feasible in realistic situations, where agents have limited knowledge, limited computational power and interact with a complex and dynamic environment that imposes uncertain outcomes and strict real time constraints. Additionally we know that the notions of rationality and logical omniscience is something that does not belong even to human beings. In fact psychologists performed many experimental studies where human behavior exhibits considerably low performances compared to ideal agents (see (TK75) and (Mac87)). This departure from correctness comes from the fact that, as for humans, our machines are finite entities that have to use efficient but formally imperfect heuristic procedures. This is necessary because typically to use formally correct and complete procedures implies exponential complexities also with small-size problem instances and requires large time and memory resources.

An interesting question is to understand if rationality is all or nothing or there is a reasonable mean. One of the first efforts in this direction were done by Herbert Simon with his models of normative and empirical rationality (Sim57) (Sim82). Simon showed that optimal decision-making is not feasible in complex domains since requires to execute intractable computations within a limited amount of time. The main idea to solve this problem was to relax optimality requirements: instead of maximizing the agent only ″satisfies″ its expected utility, choosing the alternative that is good enough according to its belief/desire sets. In other words decision makers look for choices that are satisfactory in the sense of reaching a threshold of utility. This set-up of seeking specific conditions rather than unbounded optimizations can also be applied to problem solving, substituting expect utility maximization with acting to satisfy sets of goals. In addition Good introduced a distinction between ″Type 1″ rationality, that is the ordinary notion of rationality, and ″Type 2″ rationality, where decision-making takes into account the cost of deliberation (Goo52) (Goo71).

The Simon and Good distinctions had an important influence in artificial intelligence community, precisely in the areas of search, problem solving and planning (Rus97) (Hor01). A broad class of computational models that implement bounded rationality is constituted by approximate reasoning methods, used to find approximate answers to a given problem. For example in heuristic search the domain knowledge can be used to guide the search process until a not optimal satisfactory solution is found. Moreover, related with bounded rationality, there were studies on economic decisions about reasoning (Hor88) (RW91), iteratively approximate or anytime algorithms (Hor88) (DB88); studies on selective rationality (Lei76) and bounded optimality (Hor88) (RS95) that consider optimization over circumscribed sets of alternatives that abstracts the key elements and removes the useless alternatives. Finally other approaches take into account the computational cost of decision making (DB88) (Hor88) (RS95) (WD90) (Zil95).

Another important topic related with bounded rationality concerns uncertainty and informational requirements. When a decision-maker has incomplete probabilities and preferences representations the lack of information can be a problem. One possibility to deal with this issue is to consider the expected value of information (defined estimating the gain of utility in acquiring some informations) and then act to obtain the informations with highest related utility or the informations that may decrease the uncertainty (How66) (Got12) (OVP11) (OCVPB). According to this point of view also a computation can be considered as a procedure to obtain informations, with the execution time as utility. So given a model of the consequences of computations and their cost the agent should be able to decide what computations perform and when execute them on the basis of their utilities.

Often such kind of methods are referred as meta-reasoning approaches (Hor88) (RW89) (Zil08). As Dennett was saying: "primates are the only animals able to have beliefs and desires about beliefs and desires" (Den89). The term meta-reasoning comes from the same line of ideas (see figure 1): the conceptual distinctions between object-level deliberation about external entities (considering the possible moves in a game of chess) and meta-level deliberation about internal entities as beliefs or resources (thinking that it is not convenient spending much time deliberating about a certain move). Meta-reasoning makes possible to control the object-level deliberations of the agent, when to stop them to act and to choose which of them to take in consideration, all important issues for the computational complexity of decision making.

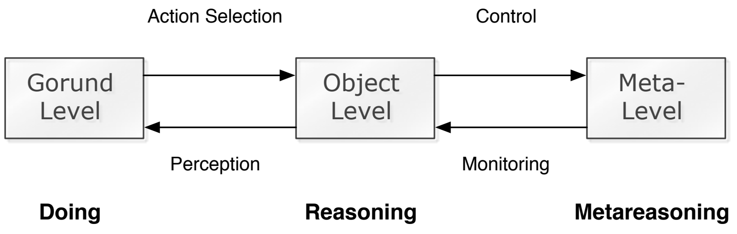


Figure 1: The hierarchy of metareasoning. Figure reproduced from (Zil08).

In (Zil08) Zilberstein reported a list of key questions to answer with respect to meta-reasoning. Below there are some of these questions that are relevant for the scope of this thesis:

1. What object-level decision-making architecture is employed? What tradeoffs it offers between computational resources and quality of results?

2. How does the metareasoning component model the objects-level reasoning process? What kind of prior knowledge is available about the efficiency and correctness of the object-level component?

3. What run time information about the state of the object-level reasoning process is being monitored? What is known about the external environment?

4. What control decisions are being made by the meta-level reasoning process? How do these decisions affect the object-level component?

5. When and how does execution switches between the object-level and the meta-level?

6. How much time is consumed by the meta-level reasoning process? How much of the metareasoning strategy is precomputed offline or computed online?

In addition metareasoning can be considered a domain independent attitude, because the meta-level deliberation does not concern with the specific object-level domain informations but with its structure that has to be extracted (RW91) (GG91).

In cognitive science metareasoning is related with metacognitive regulations, which include processes of behavior as monitoring, resource allocation, checking, planning, error detection and correction (BBFC83). Metacognitive regulation can be considered ″monitoring″ if the meta-level receives information from the ongoing object-level cognition, or ″control″ if is the meta-level to influence cognition. Finally metacognitive regulation is strongly related with meta-level knowledge: in fact to know that a task is difficult can lead an agent to allocate for it more computational resources and to monitor can lead to know which tasks are difficult and which are easy.

1.2 Attention as Solution.

Research in computer science always looked with interest at the animal brain as source of inspiration. For example the Von-Neumann architecture, base of all the modern computer architectures, was designed according to the basic structure of the information processing within the brain: the idea of disk was taken from the long-term memory of cognitive systems whereas the RAM from the working memory. If on one hand to have machines capable to mimic the basic human attitudes contributes in understanding how our brain works, on the other hand it opens important steps forward in computer science. This is possible because to see how humans or animals solve problems help us to understand how to improve our algorithms to solve the same problems.

In one of his famous paper ″elephant don't play chess″ (Bro90) R. Brooks stresses on the idea of taking inspiration from perception and motor control of natural living organisms to design intelligent machines. Indeed animals achieve incredible performances in sensorimotor tasks where robotic systems designed with a symbolic paradigm (hand-written representations) are not able to perform well also in very simple cases. So the idea of Brook was to take a step forward from the strong approach of artificial intelligence (the logical paradigm, started from the work of Turing on the basis of Fredge theories) to a bio-inspired paradigm.

In this thesis we will investigate how the study of attention can improve the design of an artificial intelligence with bounded rationality. In fact being able to focus on a suitable subset of a problem can be the solution to reduce the time complexity to solve it. In other words attention is the way how living organisms implement the same idea of many algorithms of computer science that are going against the brute-force paradigm.

About attention there is a famous sentence of one of the pioneer of psychology, William James, who in 1890 wrote: ”Every one knows what attention is. It is taking possession by the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thoughts. [. . .] It implies withdrawal from something in order to deal effectively with others” (Jam90). In cognitive science attention can be considered as a filter in information processing that focuses only on relevant information, leaving out possibly all the useless computations. Indeed, considering the real-time nature on many natural task and the bounded resources available to animal brains, it is impossible to process all the stored (covert attention) and perceived (overt attention) informations at the same time.

In a series of experiments started in the 50s on selective listening, psychologists investigated the ability of participants of listening two simultaneous speech messages. The outcomes of the experiments showed people’s *limited capacity* and *effective selectivity*: the participants were often unable to identify both messages at once and they could identify one message ignoring the other. Although the presence of selective perception, regarding the control of stimulus selection, experiments showed also the joint influence of top-down (task-driven) and bottom-up (stimuls-driven) processes (Wri68). However stimulus factors such as intensity (Bro58) or sudden onset (JY88) always contribute to the choice of which stimulus is processed, irrespective of the nature of the task.

Selective perception is implemented by an attentive dynamic process added on top of the passive elements of selection provided by the architecture of the perceptual system. In vision concurrent visual inputs compete for representation in the network of visual areas and atttended stimuli are strongly represented, while response to unwanted stimuli are suppressed (DD95). Moreover although the retina encode a wide field of view, object analysis is not uniform across the visual field but instead is concentrated in a small zone called the field of focal attention or fovea (Nei67). This is used to isolate and examine objects’ details also under conditions of interference. The mixture of information from different objects is solved in part because of the competitive, winner-take all nature of neural network convergence and in part for attentive selection (MD85) (TM96). Capacity limits of attention may vary depending upon the level of network convergence that must be reached before a discriminative decision about the set of observed objects can be achieved (MNM93).

If on one hand many computational models of attention were proposed in cognitive science (SS07) (HH05), on the other hand in artificial intelligence attention is studied mainly in the area of active vision (Bal91).

A well known proposal about bottom-up attention is the Itti model (IKN98) based on saliency maps, a data-structure that represent the probability to perform a fixation on every figure's point. The model starts identifying low level features maps (e.g. oriented edges) and then combines them to obtain an unique saliency map used to guide the gaze. A possible way to implement top-down modulation is to tune such saliency maps to look for specific objects (using suitable features template) (NI05), objects of a certain size (IK01) or within an environments with specific characteristics (NI07). Models that are totally top-down driven were considered in the context of reinforcement learning where objects discrimination can be achieved through information gain (PP00). This model was integrated with information theoretic saliency maps to produce efficient saccades able to decrease uncertainty in cluttered environments (FSPB04) (PFS05). Finally many proposals were done in the context of genetic algorithms (MFN10) (dCPvdH06), or within the area of swarm intelligence (DBT99) as (SC11).

1.3 Modularity as Solution.

Attention has been also referred as the allocation of processing resources of a cognitive system (Wic02) (Wic08) (And09). One possible way to implement this view of attention in our artificial systems is to partition our computations into specialized modules that can be activated distributing resources according to the current context. Such kind of modular representation was used in the two proposals of this thesis as in many cognitive multiple-based architectures (we will review some of them is section 2).

In artificial intelligence modularity and resource allocation are relevant issues in the area of multiagent systems (MAS) (see (CMM97) for an example about resource allocation). A MAS is an organization of coordinated agents that interact in order to achieve a common goal (KGM06). A typical design problem for MAS is the definition of a coordinated behavior of individual agents in order to achieve a system-level behavior or, more specifically, to find an effective collective state of the system given by agent interactions. A way of solving this problem consists in defining a suitable architecture for the MAS. We simply recall that while an agent architecture is a description of the internals of an agent, a multiagent architecture describes the mechanisms for controlling the interactions of societies of agents (Woo09).

Three important properties of a MAS are listed by Wooldridge in (Woo09): partial autonomy of agents; decentralization of the system's control, which is not managed by any particular agent; locality, that means each agent has a local view of the problem to solve and can interact only locally with other agents. Many issues are related with the study of MAS as negotiation, communication, multi-agent learning, organization, agent-oriented software engineering and so forth. Here we will be more interested in cooperation and coordination issues related with distributed artificial intelligence and distributed problem solving as problem subdivision, sub-problem distribution, optimization of problem solver coherence and coordination and results synthesis.

As we said in MAS we are interested in the coordinated behavior of a system of individual agents to provide a system-level behavior. Sycara (Syc98) lists six important challenges about this issue:

1. How to decompose problems and allocate tasks to individual agents.

2. How to coordinate agent control and communications.

3. How to make multiple agents act in a coherent manner.

4. How to make agents reason about other agents and the state of coordination.

5. How to reconcile conflicting goals between coordinating agents.

6. How to engineer practical MAS.

MAS methodologies can help us to accomplish these challenges. Several architectures were proposed (IGCGal99), which can extend theories coming from object oriented programming (BGPP03) (DeL99), knowledge engineering or organizational methodologies (FG98) (KGM06). Finally several applications of MAS were proposed in many fields: computer network, logistics, cognitive science, transportation, computer games, business process management, distributed sensing, information retrieval, electronic commerce, human-computer interfaces, social simulation and many others.

In the following pages we will introduce two modular architectures with bounded rationality that address covert and overt attention. The **structure of the thesis** is organized as follows.

Before introducing our proposals, in section 2 we will review some multiple model-based architectures that we think to be relevant for this thesis, with particular emphasis of the AKIRA framework (section 2.1) and the mixture of experts model (section 2.2). Then, in section 3, we will introduce our MAS architecture based on the submitted paper (CVQP). It is called attractor agent network and uses a Hopfield network to activate and coordinate agents. The AAN architecture was tested on three different applications: a distributed constraint optimizations task (section 3.9), a classification task of machine learning (section 3.10) and a categorization task of computer vision (section 3.11). Additionally, in section 4, we reported an experiment that we performed to collect behavioral data from human participants, facing the same visual categorization task, in order to investigate the cognitive plausibility of our proposals. Then in section 5 we presented the second computational proposal of the thesis that, together with the aforementioned experiment, is based on the published paper (QCVBP). The proposal concerns a dynamical neural field based architecture still used to solve the same categorization task of the previous human and computational experiments. In section 6 we will discuss about the obtained results and compare the common and different characteristics of the two proposals. Finally in section 7 we will report some of the future developments of the thesis and in section 8 we will conduct our conclusions.

Chapter 2

Multiple Model-based Architectures

Within the artificial intelligence and cognitive robotics communities many multiple-model based architectures were proposed. All these architectures have in common a modular representation and some kind of policy used to arbitrate the modules' deliberation.

In the two papers (SBR05) and (RB10) Ballard et al. proposed a computational model of human visually guided control in a reinforcement learning framework. They designed a control architecture that allocates resources to modules (called micro-behaviors) in response of task demands. Although many microbehaviors are available to address the goals of the agent, at any one time, only a small subset of those are actively engaged. Their proposal was tested on a virtual navigation task, enriched by the goals of avoiding obstacles and collecting litter. The aim of the architecture was to manage the extraction of information from visual input that is in turn mapped onto motor commands. Their computational hierarchy is defined as follow: with the behavior level the microbehaviors execute generating control signals and computing state information necessary for meeting behavioral goals (through Kalman filters); since microbehaviors share perceptual and motor resources, at the arbitration level the competition between them is solved through gaze allocations that are selected to minimize the risk of loosing reward in the set of running behaviors; finally the context level maintains an appropriate set of active microbehaviors according to current goals and environmental conditions. Other modular proposals were done in the context of Markov decisions processes (Ber95) (Put94) (SJBH11), as HORDE (SMD^{+}11) or the hidden-mode Markov decision process (CYZ01).

Another famous multiple model-based architecture is MOSAIC (modular selection and identification for control) (WK98) (Kaw99) (HWK01) (SHDK12) (SMHK12). This is an architecture for motor learning and control based on multiple pairs of forward (predictor) and inverse (controller) models. The architecture simultaneously learns the multiple inverse models necessary for control as well as how to select the set of inverse models appropriate for a given task. It combines both feedforward and feedback sensorimotor information so that the controllers can be selected both prior to movement and subsequently during movement. Among many available at any given time the system select only one inverse model (control module) according to the accuracy of prediction of its paired forward model. Indeed within a module, the forward and inverse models are tight coupled during both their acquisition and use. The forward models learn to divide up the contexts experienced so that for any given context, a set of forward models can predict the consequences of a given motor command. The predictions errors of each forward model are then used to gate the learning of its paired inverse model. This ensures that within a module, the inverse model learns the appropriate control for the context in which its paired forward model makes accurate predictions. So the selection of the inverse model is derived from the combination of the forward model's perdition errors and the sensory contextual cues, which enable MOSAIC to select controllers prior to movement. Note that to learn how to specialize pairs, and how to select them, both gradient-based learning and hidden Markov models were proposed.

Similar architectures are PASAR (model of prediction, anticipation, sensation, attention and response for artificial sensorimotor systems) (MBiBV11) and HAMMER (hierarchical attentive multiple models for execution and recognition of actions) (DK06). This was proposed on the basis of mental simulation theories. It was used for the recognition and execution of actions utilizing hierarchical and attentive multiple models. Also in this case we have pairs of inverse and forward models but arranged this time in a hierarchical and parallel manner with a top-down control of attention. This hierarchical structure induces primitive inverse models to be combined to form higher more complex motor sequences, with the eventual goal of achieving increasingly more abstract inverse models. Moreover bounded rationality is implemented through the utilization of limited computational and sensory resources.

In the following sections we will review with more details other two multiple model-based architectures: AKIRA and the mixture of experts model.

2.1 AKIRA

AKIRA is a cognitive architecture that benefits of several properties typical of cognitive systems. AKIRA takes inspiration from some properties of complex systems and biological organism as self-organization, adaptivity, learning and robustness.

Let us review shortly the components of AKIRA. First we have a server process called Pandemonium that execute and monitor many instances of computational modules called daemons. Daemons have both a symbolic component (i.e., their operations) and connectionist component as an activation variable, called energy, that represents the computational resources owned by a daemon. The global activation owned by the whole system is limited so that its utilization can be optimized. This global activation is stored is a tank called energy pool that contains a finite amount of energy available to every daemon. Finally inside the AKIRA architecture we find a global database called blackboard used by daemons to exchange messages in order to achieve coordination and communication.

AKIRA Energetic Model Daemons have a priority that is proportional to their energy (see figure 2). Moreover the energy is distributed by a weighted activation network that connects daemons called energetic network. The definition of an energetic network was introduced first in the DUAL architecture (Kok94).

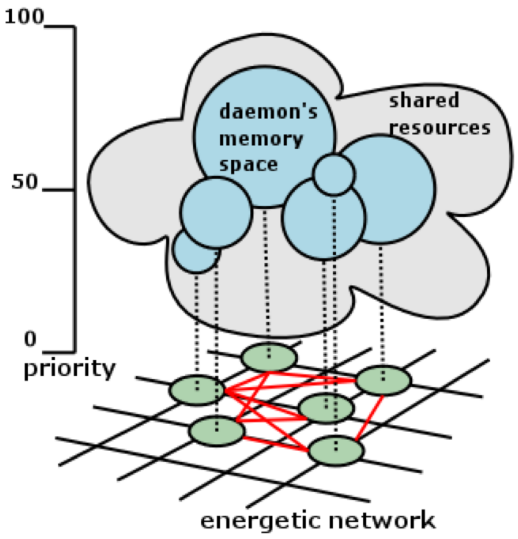


Figure 2: In AKIRA the energetic network is used to allocate computational resources within daemons. Figure taken from (PC07).

Daemons have a hybrid nature: they can be considered both as a computational unit and as a node of a connectionist network. Nodes in neural network typically identify values (so representations are totally distributed). In AKIRA nodes identify modules that execute their operations in a parallel and asynchronous manner (so both representations and functions are distributed). Moreover the connectionist component concerns the energy activation, the exchange of energy between daemons and finally the organization of daemons in spontaneous assemblies called coalitions. Coalitions are able to solve composite problems that single daemon can not solve alone.

On the other hand the procedural component concerns the set of operations that daemons can perform. Indeed every daemon is a specialized computational unit that can contain procedures of arbitrary complexity.

More daemons own energy and more their computational power is large. So every daemon has a certain amount of computational resources that are proportional to their activation level. Energy can represent the absolute relevance of a daemon and its contextual relevance in a given situation. More daemons gain energy and more they have priority to execute their operations and operations regarding the energetic network. This architecture based on the concept of energy offers promising solution to the problem of context in cognitive science and artificial intelligence because it produces contextual pressures that are able to guide the collective behavior of the system.

Inside the AKIRA energetic model we find a central tank of resources, the energy pool, that represent a global bound to the resources that daemons can access. If a daemon own an amount of energy these resources are not available to other daemons until they will be released again to the energy pool. So daemons compete for the access to the energy contained in the energy pool. Note that also if the energetic network has not negative links the architecture implements inhibition between daemons through this limited global amount of resources.

Energy is also exchanged between daemons through the energetic network. When this happens resources are effectively transferred from the original owner to the new one. This means that one daemon is gaining new computational resources that another daemon is loosing. For this reason such spread of activation is different from the one usually implemented in neural networks because there, regarding the total activation, the network is neither homeostatic nor conservative.

In AKIRA among daemons there is both a short range excitation given by positive links of the energetic network and a long range inhibition given by the energy pool. They correspond respectively to positive and negative feedbacks that taken together produces among daemons self-regulating and self-sustaining patterns of activity. So after a sufficiently long time the energy dynamics will be dominated by patterns of activity that represent the coordination between daemons necessary for the problem to solve.

The execution of daemons' operations has a cost in term of energy. So daemons have to obtain such energy from the energy pool before their

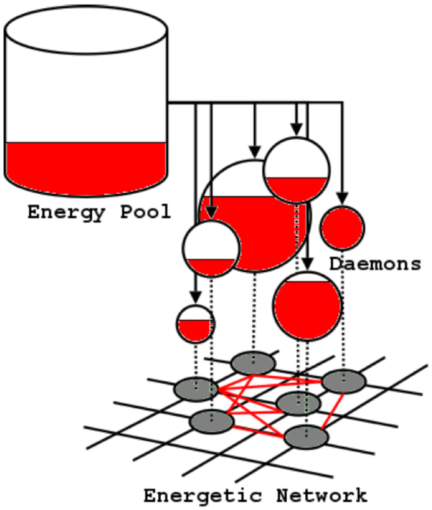


Figure 3: Akira Energetic Dynamics. Figure taken from (PC07).

execution and give it back once they have finished.

Considering that in AKIRA the total amount of resources are limited and that daemons consume resources to execute their operations only a limited number of daemons can be active at the same time. This introduces efficiency in the system because to daemons is not allowed to take control all together. This mechanism implies competition between daemons: active daemons inhibit the others because they consume the energy that everybody need.

Let us say something more about daemons. If on one hand the overall architectures is considered defined a priori on the other hand the definition of daemons' operations is left to the user. The daemons' implementation represents the only effort that the user needs to do in order to specify the architecture behavior according to the problem to solve.

The priority of the thread executing the code of a daemon is proportional with its level of energy. The possible sources of energy of a daemon are the following: first an amount defined by the user, available for every cycle, that is independent from the energy pool (we call this energy base energy); then there is the shared energy that a daemon can take from the energy pool. Both the latter and the base energy represent the absolute relevance of a daemon and is the same for each cycle. Finally we have the energy that is possible to take from the other daemons through the connections of the energetic network, which represents the contextual relevance of the daemon and is time dependent (this energy can change according the current situation and is called linked energy).

During one cycle of execution a daemon does the following steps: first it tries to take an amount of energy from the energy pool (note that it could be not available). Then, if it has the necessary energy, the daemon execute its operations. The quality of the daemon's operations result is evaluated by a test function (defined by the user) that can output a success or a failure. The success of a daemon can influence both its activation and the energetic network topology. After the execution, if the daemon succeeded, the energy used has to be given back to the energy pool, so that it will be available again to other daemons. Moreover new incoming links are established between the daemon and its neighbors (this is the way coalitions are formed) or alternatively existing links are strengthen.

On the contrary if a daemon failed it spreads its energy to more pertinent daemons through the connections of the energetic network. Furthermore the incoming links are weaken. In addition the successes and failures are notified through the blackboard to other daemons to notify the quality of their behavior.

The combination of these steps constitutes the way that AKIRA uses to activate the right set of daemons. Indeed using their feedback with the environment daemons learn to synchronize each other modifying the topology of the energetic network, so that the energy is transferred from failing daemons to successful ones.

Typically in trying to solve a problem we have to choose among several possible interpretations in front of different situations that need different capabilities. In AKIRA this means to select among the different candidate daemons, to find the best ones in the current situation. The energetic dynamics assigns more computational resources to more relevant daemons in decentralized way, so that only a small number of daemons are active in a given context. This selection problem is not solved immediately but more concurrent hypothesis (active daemons) can go on together until one will dominate the others. In AKIRA dominant daemons take control of the system (they have more resources) but also less active daemons have still the possibility to become the new dominant, if the old ones start to fail too often in the current situation. In this case when a daemon stops to have success its level of activation should starts to decrease as the weights of its incoming links. On the other hand daemons that are becoming more relevant start to gain new resources.

2.2 The Mixture of Experts model

The mixture of experts model (JJNH91) was introduced to overcome the limits of the back propagation algorithm (RHW86) used to learn a large class of neural networks. This algorithm, due to the interference among the memorized patterns, often achieve poor generalizing capabilities and slow learning complexity.

The idea of the mixture of experts model is to build a modular system composed of a set of neural networks, called experts, each one specialized on different portion of the input space. This specialization is obtained through an additional neural network, called gating network, that focuses the experts' learning according to different input classes. Moreover the gating network is used to choose the most adequate experts in front a certain input data set.

Once the gating network spreads the input to all the experts, on the basis of their errors, only those that are considered more pertinent have their weights modified locally as the weights of the gating network. Doing like that we will have experts behaving locally for two reasons: first the weights of one expert will be not correlated with the weights of other experts; then to each expert is assigned only a little region of the input vector.

In the classical mixture of experts model every expert is a feed forward neural network that receives the same input and has the same number of output units. Also the gating network is a feed forward neural network too that receives the same input as the experts networks. On the other hand the gating network has a normalized output

$$p_j = \epsilon(x_j) = \frac{e^{x_j}}{\sum_i^{x_i}} \tag{2.1}$$

where x_j is the weighted input that arrives to the j-th output unit of the gating network. Once the networks processed their outputs a special unit called selector choose the output of expert j with probability p_j. See figure 4 to understand hot the different mixture of experts components interact with each other.

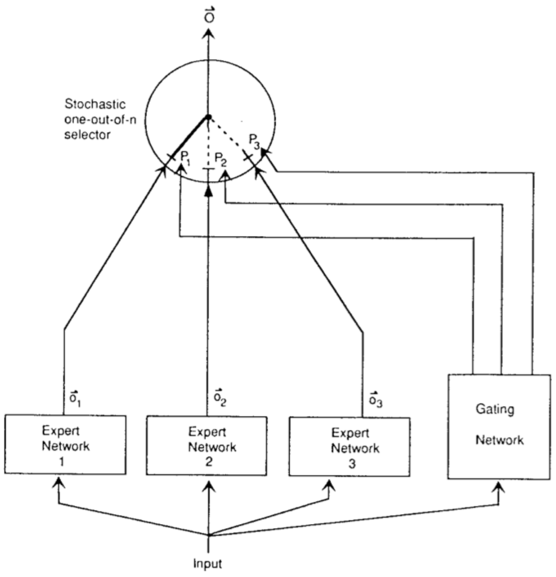


Figure 4: In the mixture of experts model we have three components: the expert networks, the gating network and a selector module. In this figure we can see how these components interact with each other. Figure taken from (JJNH91).

So a mixture of experts chooses stochastically the best expert according to the current input vector. In equation (2.2) we have the error function that has to be used: the expected value of the difference between the desired output and the actual one.

$$E^c = \langle \|\mathbf{d}^c - \mathbf{o}_i^c\|^2 \rangle = \sum_i p_i^c \|\mathbf{d}^c - \mathbf{o}_i^c\|^2 \tag{2.2}$$

In equation (2.2) $\mathbf{o}_i^c$ is the output vector of expert i on training data c and $\mathbf{d}^c$ is the desired output vector on training data c. Note that using this error function for learning we have that to modify the weights of an expert on a certain input is not directly affected by the weights of other experts.

If both the experts and the gating network are trained with a gradient descent algorithm, using E^c as error function the mixture of experts will assign to each class of input a single expert. Indeed when an expert produces an error that is smaller than the weighted average of the errors of the other experts (using the gating network's output to weight the error of each expert) its responsibility about that input class will be increased. On the contrary if its error is larger its responsibility will be decreased.

For numerical reason Jacobs and Jordan suggest to consider the following error function of equation (2.3).

$$E^c = -log(\sum_i p_i^c e^{-\frac{1}{2}\|\mathbf{d}^c - \mathbf{o}_i^c\|^2}) \tag{2.3}$$

Once we have defined the error function, to train the experts we compute its derivative respect to the expert's weights to use it in the back propagation algorithm (to descend the gradient). The same computation has to be done for the gating network.

Let us start with the derivative on the output units' weights. About the hidden units the derivative is the same as the one used in the classical back propagation algorithm. We represent with w_{ij} the weight of the link that connects the hidden unit i with the output unit j and with x_{ij} the i-th input of the j-th unit. The derivative is $\frac{\partial E^c}{\partial w_{ij}} = \frac{\partial E^c}{\partial x_j}\frac{\partial x_j}{\partial w_{ij}} = \frac{\partial E^c}{\partial x_j} x_{ij}$. According to the equivalence $\frac{\partial E^c}{\partial x_j} = \frac{\partial E^c}{\partial o_j}\frac{\partial o_j}{\partial x_j}$ and taking the sigmoid function as threshold function of the output units we compute the two partial derivatives to obtain the following equation for the expert j

$$\frac{\partial E^c}{\partial w_{ij}} = -o_j(1-o_j)\frac{p_j^c e^{-\frac{1}{2}\|\mathbf{d}^c-\mathbf{o}_j^c\|^2}}{\sum_i p_i^c e^{-\frac{1}{2}\|\mathbf{d}^c-\mathbf{o}_j^c\|^2}}(\mathbf{d}^c-\mathbf{o}_j^c)x_{ij} \tag{2.4}$$

Note that in the right term of equation 2.4 there is a weight factor that represents the quality of the expert i compared with the others. This factor can be used as a measure of the relevance of the expert i on the input class c.

The same kind of computations can be done to obtain the derivative of the error function respect to the weights of the gating network's output units links.

$$\frac{\partial E^c}{\partial w_{ij}} = -\frac{e^{-\frac{1}{2}\|\mathbf{d}^c-\mathbf{o}_j^c\|^2}}{\sum_i p_i^c e^{-\frac{1}{2}\|\mathbf{d}^c-\mathbf{o}_j^c\|^2}}\frac{e^{x_j}\sum_k e^{x_k}-e^{2x_j}}{(\sum_k e^{x_k})^2} \tag{2.5}$$

Using these error derivatives we can implement the back propagation algorithm for the mixture of expert models. It will be useful is section 3.10 where we will use it for one of our proposal. Note that the mixture of experts model was applied to extend hierarchically the model (PHC00) or to other machine learning approaches as recurrent continuous time neural networks (TNNI08) and reinforcement learning (DSKK02).

Chapter 3

First Proposal: The Attractor Agent Network

In Multiagent Systems (MAS) to decide a suitable scheduling order of agents and to determine how they coordinate are key issues (see section 1.3 to have more details about MAS). We want that the right subset of agents are active in the right span of time to cooperate to fulfill some specific goal. When the designer is not able to decide these scheduling and coordination issues, due to the complexity of the system, a machine learning approach can be used (B^+06) (Mur12). In this section we present a MAS architecture called Attractor Agent Network (AAN), which uses a Hopfield network (Hop82) (Ami89) to activate and coordinate agents (in section 3.1 we will review the main properties of Hopfield networks) .

AAN has an architectural nature: it uses a fixed machine learning methodology to achieve coordination among any kind of agents. So AAN's users have not to deal with the collective behavior of the MAS, which is already defined in the AAN and does not need to be changed. They have only to implement the particular semantics of individual agents and a function that represents their ability to achieve their goals. Once this is done AAN will automatically bring the system to an effective collective state. As we will see in the AAN the collective behavior can be defined by the designer or can be automatically learned.

The AAN approach follows the Pandemonium metaphor (Jac87) and is related to Blackboard systems (HR85). The Pandemonium involves an object that owns a set of demons. Each demon tests for a particular combination of conditions in the data that it can see. If a demon sees what it is looking for, the demon shrieks. Closer is the match and louder is the shriek. If the demon shrieks loudly enough to attract the attention of its owner, the owner may choose to act in relation to what the demon is shrieking about.

On the other hand Blackboard systems are systems with a shared knowledge base called blackboard that is iteratively updated by a diverse group of specialist knowledge sources. These specialists work together to solve the input problem. The blackboard starts with a problem specification and ends with a solution. A Blackboard system is composed by three components: the specialists, the blackboard and the control shell. Examples of Blackboard systems are Hearsay II (EHRLR80), Copycat (Hof95) and the Learning Classifier System (Hol89).

Moreover the AAN follows the Akira Energetic Model (AEM) (PC05) (PC07) that we have described in section 2.1. So agents are connected with each other through an activation network (the energetic network) that distributes a bounded amount of computational resources (called energy) among agents. Together with the constraint of a limited amount of computational resources the AEM introduces competition among agents, in order to automatically select the best subset of agents to be active at the same time.

A suitable scheduling and resource distribution are key points to achieve coordination and cohesion among agents. So, to accomplish these objectives, the main idea behind the AAN is to manage the computational resource allocation among agents using a Hopfield network. Indeed in the AAN architecture a Hopfield network is used as energetic network in order to benefit from the well defined properties coming from its definition in statistical physics. The usage of Hopfield network makes available to MAS a sophisticated mathematical language, composed of formal concepts as probability distribution of activation, attractors, energy function, correlation function and other macroscopic variables that summarize the collective behavior of agents. These can be used to analyze, predict or tune the MAS behavior. For example using the energy function of Hopfield networks we will see in section 3.7 how to measure the state of the coordination of agents. Other two very interesting ANN's features coming from the usage of a Hopfield network regard its capability to perform well also in presence of noise and when the number of agents is high.

In effect the choice of using Hopfield networks was done because with statistical physics we can formally characterize the dynamical evolution and the collective behavior of a system that consists of a high number of components. Indeed a MAS with a high number of agents belongs to the class of systems that statistical physics can successfully model[1]. Examples present in literature (such as in immunology, economy (dMM06), sociology and linguistics (CFL09)) are the Voter model (Lig99), the Naming Game (BFC$^+$05), the Minority Game (dMM06), the Brownian agents (Sch03) and many others.

We think that the AAN can be used as optimization's *metaheuristic* too (Luk09). Note that the AEM was already applied to the edge coloring problem (Pez09). In section 3.9 we will use the AAN architecture to solve an optimization task. Hopfield networks were successfully used for optimization problems too, for example the Traveling Salesman Problem (HT85). Additionally examples of famous metaheuristic algorithms are Ant Colony Optimization (Dor92), Genetic Algorithms (Hol75) and Simulated Annealing (KGV83).

To prove its ubiquity the AAN architecture has to be tested on different kind of applications, to show its ability to perform well in different contexts. We intend to perform three different kinds of experiments in the following sections. The first one, in section 3.9, concerns an optimization task related with a board game; the second experiment, in section 3.10, implements a classification task of machine learning; finally in the third one, in section 3.11, the AAN performs a distributed categorization task of computer vision.

[1] Note that a self organizing MAS with an high number of agents can be defined as a Swarm Intelligence (DBT99).

3.1 Preliminaries: Hopfield Networks

Hopfield networks are recurrent neural networks introduced by J. J. Hopfield in his famous paper ″neural networks and physical systems with emergent selective computational abilities″ of 1982 (Hop82). In this paper the author defines a model of associative memory based on a set of formal neurons (MP43; Ros62) that represents the first real mathematical model of hebbian learning. In hebbian learning weights between learning nodes are adjusted so that each weight better represents the relationship between the nodes. Nodes that tend to be positive or negative at the same time will have strong positive weights while those which tend to be opposite will have strong negative weights. Nodes which are uncorrelated will have weights near zero.

In a series of experiments made on the cerebral cortex of monkeys the biological plausibility of Hopfield networks was confirmed (YM88). There are other methodological approaches different from the experimental one used to confirm the model: several instances of the model were built using hardware and software implementations and in both cases the resulting dynamics showed the same behaviors predicted by the model (see below).

One of the main advantage that Hopfield networks offer compared with other neural networks models (e.g., feed forward neural networks) is that they give the possibility to read their internal dynamics with a well defined mathematical language. This is possible because Hopfield established a formal equivalence between his model and the Ising model of statistical physics (Isi25), used by statistical physicist to represent magnetic systems. According to this equivalence the magnetization in the Ising model corresponds to the activation in Hopfield networks.

A Hopfield network can be built closing a multi-perceptron neural network (WH$^+$60) on itself (i.e., connecting its output units with its input units, see figure 5). So we obtain a feedback mechanism. Starting from the definition of a Hopfield network taken from (Ami89), let us define the neural state $\mathbf{S}$ of a Hopfield network composed by N neurons as any possible combination of spiking and not spiking neurons, so $\mathbf{S} \in \{-\mathbf{1}, \mathbf{1}\}^{\mathbf{N}}$. We can then describe a Hopfield network as a dynamical system that

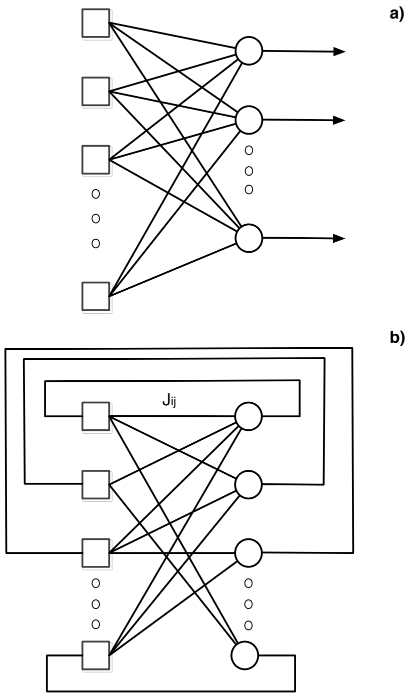


Figure 5: a) A multi-perceptron neural network. b) A Hopfield network obtained closing the multi-perceptron on itself. Figures reproduced starting from (Ami89).

evolves from one neural state to another.

The cumulated Post Synaptic Potential (PSP) at time $t+1$, $U_i(t+1)$, cumulated on neuron S_i (which has values +1/-1) of a Hopfield network of N neurons with links J_{ij} is defined as in equation (3.1).

$$U_i(t+1) = \frac{1}{2} \sum_{j=0, j\neq i}^{N} J_{ij}(S_j(t)+1) \tag{3.1}$$

Where in equation (3.1) J_{ij} is an element of the synaptic matrix, which can have both positive and negative value, corresponding to an exciting or inhibitory effect. In a dynamics without noise the relation between the neuron i with threshold T_i and its PSP it is defined as in equation (3.2).

$$S_i(t+1) = sign(U_i(t+1) - T_i) \tag{3.2}$$

It is useful here to remember the physical meaning of this expression. The argument of the *sign* function can be divided in external magnetic field h_i^e, which does not depend on the value of the other neurons, and the local magnetic field h_i, which is determined by its connection with the other neurons:

$$S_i = sign(h_i + h_i^e) \tag{3.3}$$

They are defined as in equation (3.4).

$$h_i = \sum_{j=0, j\neq i}^{N} \frac{1}{2} J_{ij}(S_j) \quad h_i^e = \sum_{j=0, j\neq i}^{N} \frac{1}{2} J_{ij} - T_i \tag{3.4}$$

As every dissipative dynamical system Hopfield networks have a set of attractors. In the case of Hopfield networks they are a set of neural states that repeat them selves indefinitely. So Hopfield networks divide input neural states in equivalence classes represented by attractors, the same attractors where the input neural states will flow soon or later.

Cognitive events are identified by the arrival to attractors of the Hopfield network. The rapid arrival to one attractor is considered as the recall of a pattern from the memory. This pattern is retrieved on the basis of the similarity with the input neural state.

To be sure that a Hopfield network has only stable fixed point attractors some assumptions are necessary (Roj95). First neurons have to be without memory, that means they have to respect the Markov property (i.e., the value of $U_i(t+1)$ depends only on the value of neurons at time step t). Moreover we have to do some additional assumptions about the synaptic matrix: first we want that each neuron is connected with every other neurons (complete graph topology); then the matrix has to be symmetric ($J_{ij} = J_{ji}$); finally the absence of self-interactions has to be assured ($J_{ii} = 0$).

Hopfield networks are associative memories. In this context memory represents the capacity to reproduce as an attractor an activation pattern that already passed through the network before.

So consider the set of not correlated attractors (memories) of equation (3.5) that we want to insert within the network.

$$\xi^{\mu} = \{\xi_1^{\mu}, \xi_2^{\mu}, \ldots, \xi_N^{\mu}\} \tag{3.5}$$

Where in equation equation (3.5) μ is an index over memories.

We want to investigate the relation between the synaptic matrix of the network and its set of attractors ξ^{μ}. First let us list two simple asymptotic properties that we would like to have for the network's dynamics: a dynamics that is not sensible to the initial conditions (i.e., not chaotic); then we want that the type of memorized attractors depends only on the values of the synaptic matrix.

Given these desirable properties, is it possible to build a synaptic matrix that guarantees that a prescribed neural states will be the attractors of the network's dynamics? Moreover, is it possible to guarantee that these will be the only attractors of the network? Indeed given a neural state exists a synaptic matrix that makes it an attractor of the network's dynamics in absence of noise. At the same time with the presence of noise (under a certain threshold, see (Ami89) for more details) with the same synaptic matrix it is assured an attractors' distribution where neural states near the memorized one have a probability different from zero to be visited. The following Hebb rule of equation (3.6) makes possible to obtain this synaptic matrix starting from the neural state that we want to memorize.

The Hebb rule is generally used to memorize a set of arbitrary patterns in a Hopfield network, and works as follow: for each known attractor $\xi^\mu \in \{+1, -1\}^N$, with $\mu \in [1, \dots, p]$, the synaptic matrix entries, initialized to zero, are modified adding the term $\Delta J^\mu_{ij} = \xi^\mu_i \xi^\mu_j$. Integrating over all attractors, the final weight's are given by:

$$J_{ij} = \begin{cases} 0 & \text{if } i = j \\ \frac{1}{N}\sum_{\mu=1}^{p} \xi^\mu_i \xi^\mu_j & \text{otherwise} \end{cases} \tag{3.6}$$

It is possible to show that the Hebb rule modifies the weights of a Hopfield network in a way able to descent the gradient of the network's energy function (see below) until a minimum is defined for each inserted pattern, which for this reason will become attractors of the network's dynamics.

Memorizing patters means that neural states as in equation (3.7), for every patterns $\mu = 1, \dots, p$, are fixed point attractors of the network's dynamics.

$$S_i = \xi^\mu_i \qquad\qquad i = 1, \dots, N \tag{3.7}$$

An interesting property of Hopfield networks concerns the possibility to assign a Lyapunov function to its dynamics. We call this function $H(\mathbf{S})$ and in physical terms it is an energy function (hamiltonian, equation (3.8)).

$$H(\mathbf{S}) = -\frac{1}{2}\sum_{i,j\ i\neq j}^{N} J_{ij} S_i S_j \tag{3.8}$$

As Lyapunov function $H(\mathbf{S})$ decreases as the dynamics of a Hopfield network evolve and it has the attractors neural states as its minima. So studying the $H(\mathbf{S})$ landscape we can find the network's attractors, as characterize them or estimate their basin of attraction.

When the dynamics of a Hopfield networks is perturbed by a gaussian noise, after a time sufficiently long, the network will relax toward a Maxwell-Boltzmann distribution (equation (3.9)).

$$P(\mathbf{S}) = \frac{e^{-\frac{H(\mathbf{S})}{T}}}{Z} \tag{3.9}$$

Where in equation (3.9) T is the temperature of the network that is related with the variance δ of the gaussian noise added to the Post Synaptic Potential U_i, for every neuron $i = 1, \ldots, N$, as $T = 2\sqrt{2}\delta$. Z is the partition function used to normalize the distribution.

$$Z = \sum_{\mathbf{S}} e^{-\frac{H(\mathbf{S})}{T}} \tag{3.10}$$

Another important result about Hopfield networks it is related with their capacity α, that is the number of patterns that can be memorized. There is a known limit from Hopfield Network theory about how many patterns we can memorize properly if we use the Hebb rule: if we have N agents this limit is $\alpha = 0.138N$. Note that this equation is is valid only when the memorized attractors are uncorrelated, but it can be considered as an upper bound for correlated patterns.

As we said Hopfield networks were used to model neurophysiological data taken from experiments made on animals, but this is not the only kind of application Hopfield networks were utilized. Hopfield networks were applied successfully for error correction, image processing, optimization and so forth. Mainly Hopfield networks can be used for memorization and recognition tasks. The idea is to store a series of patterns that the network can reconstruct once corrupted versions are presented as input neural states.

In this context applications concern the ability to reconstruct a signal transmitted through a noisy channel or speech and image recognition. For example about images we can assign to each black and white pixel a neuron of the network. Then, during the learning phase, we can store a set of images as patterns of the network. Once a distorted version of a memorized pattern is presented to the network as input neural state the dynamics will bring the network to the attractor (the original image) that is the most similar with the input .

Hopfield networks were used also to solve successfully combinatorial optimization problems (HT85) as the matching problem or the traveling salesman problem. Typically to solve such kind of problems we have to use neurons to represent the variables of the problem. The synaptic matrix has to represent the parameters to optimize and their constraints in such way that the solution of the problem will be the minima of the energy function and so the attractor of the network dynamics.

From a technological point of view Hopfield networks can have an hardware implementation too (VJ89). Neurons can be built using amplifier with capacity and synapses using resistances. This can be done to obtain faster computations. For example in image recognition a traditional method should compare all the images memorized on the disk to find the one that we want to recognize. This procedure is very slow, so neural chips based on associative memory were introduced to reduce as much as possible the disk access.

In the following sections we will present a new class of applications for which Hopfield networks can be used. Indeed to apply Hopfield networks to manage the computational resources of agents, in order to achieve coordination and selection of a MAS, is a novel idea.

3.2 Energy and Activation

Consider a Multi-Agent System where to each agent is assigned a quantity called *energy*. Agents have a priority and a computational power that are proportional to their energy, which therefore represents the access to the computational resources of the system. Let $E_i(t)$ be the energy of agent i at time t. The execution of the code of an agent has a cost in term of energy which is called *run energy* R_i. It represents the the amount of energy necessary for the agent i to become active and perform its computations.

The cost of an operation should be assigned according to its complexity and urgency, relatively to the application considered. If the agent's operations have a low cost this means that it can be easily activated. So we can consider this cost as the inverse of the operations' urgency. If on one hand urgent operations should have a low cost, on the other hand complex operations can be considered slower and so they should need more energy.

We can then define the *activation variable* $A_i \in \{-1, +1\}$ as follows:

$$A_i = \begin{cases} +1 & \text{if } E_i \geq R_i \text{ (\textbf{active})} \\ -1 & \text{otherwise (\textbf{idle})} \end{cases} \tag{3.11}$$

So each agent in the AAN architecture has its own code of execution which it will execute every time it is activated. Agents can obtain sufficient energy to switch from idle to active from two possible sources. The first one, called *base energy* or B_i, is defined by the user to represent the absolute relevance of the agent, independent of time and immediate context. It might for instance bias the selection of active agents due to external constraints (as in section 3.11.7) or reflect habituation or learning dynamics. The other source, called *linked energy* or L_i, is the energy received from other agents, and represents the contextual relevance of the agent. The sum of base energy and linked energy defines the total energy E_i received by agent i .

3.3 The Attractor Agent Network

The Attractor Agents Network (AAN) consists in a Hopfield network (Hop82) that connects agents and distributes energy between them, so that their activity can be well coordinated. The presence of a limited amount of energy guarantees only a small subset of agents will be active at the same time, by putting them into competition, thus only allowing mutually congruent agents to remain active in the end. Additionally, learning leads the system to select the most suitable agents depending on the context and problem to solve. See also (Kok94) (PC07) for related models managing the distribution of energy among components through networks that are called *energetic network*. The following paragraphs describe the functioning of such a network.

We start from the mathematical definition of a Hopfield network taken from (Ami89), where terms are translated in the AAN terminology. Table 1 makes it possible to easily transfer known results in Hopfield networks to energetic networks, and thus to AANs. In turn, equation 3.13 transposes into energy the computation of the postsynaptic potential within neuron i from the excitatory and inhibitory activities of connected neurons. Synaptic weights between neurons are encoded in matrix J, with positive values corresponding to excitatory links. The original Hopfield equation exactly corresponds to the computation of the linked energy, and is only altered to include the base energy, thus enabling the possibility to boost the energy associated to agent i when $B_i > 0$.

$$A_i(t) = sign(E_i(t) - R_i) \tag{3.12}$$

$$E_i(t+1) = B_i + L_i(t+1) = B_i + \frac{1}{2}\sum_{j=0, j\neq i}^{N} J_{ij}(A_j(t)+1) \tag{3.13}$$

Where in equation (3.13) J_{ij} are the links of the energetic network, which connect the agents with each other. A simple algebraic computation is now shown to divide the argument of the *sign* function in equation (3.12) into a time dependent component E_i^I and a not time dependent

Table 1: Hopfield to AAN terminology

Hopfield network (as in (Ami89))		AAN	
Neurons		Agents	
Symbol	Description	Symbol	Description
U_i	Post-synaptic potential	E_i	Energy
T_i	Activation threshold	R_i	Run energy
S_i	Spike	A_i	Activation
J_{ij}	Synaptic matrix	J_{ij}	Link matrix

component E_i^e. They correspond respectively to the internal magnetic field h_i and external magnetic field h_i^e defined in equation (3.4).

$$\begin{aligned} E_i(t+1) - R_i &= B_i + L_i(t+1) - R_i \\ &= B_i + \frac{1}{2}\sum_{j=0,j\neq i}^{N} J_{ij}(A_j(t)+1) - R_i \\ &= \frac{1}{2}\sum_{j=0,j\neq i}^{N} J_{ij}A_j(t) + \sum_{j=0,j\neq i}^{N} J_{ij} + B_i - R_i \\ &= E_i^I(t+1) + E_i^e \end{aligned} \tag{3.14}$$

$$E_i^I(t+1) = \frac{1}{2}\sum_{j=0,j\neq i}^{N} J_{ij}A_j(t) \tag{3.15}$$

$$E_i^e = \sum_{j=0,j\neq i}^{N} J_{ij} + B_i - R_i \tag{3.16}$$

3.4 Agents' Attractors

An Hopfield network is used to memorize patterns which are represented by the attractors of the dynamical system that it defines. Whatever the configuration of an Hopfield network, it will sooner of later converge to one of its attractors. The set of initial configurations leading to the same attractor belongs to its basin of attraction.

The same attractors are found at the core of an AAN and are called *agents' attractors*. Each attractor $A^\mu = (A_1^\mu, A_2^\mu, \dots, A_N^\mu) \in \{-1, 1\}^N$ is a configuration of active agents (that will execute their operational code) and non-active agents (that will remain silent). Each A_i^μ here represents the asymptotic activity of agent i, i.e. its activity after an indefinitely long number of energetic network updates (see equation 3.13). So agents are divided in different group of execution.

Another interesting feature of an AAN concerns its ability to perform well also in presence of noise or when the number of agents is high. Robustness to noise means that the retrieval of a agents' attractor can be done even if the network is under the influence of noise. Moreover, noise can be considered as the temperature of the system and determines the trade off between exploration and exploitation. It is thus required to escape from local minima and to destabilize spurious attractors, which are a mixture of memorized attractors (see (Ami89) to have more details).

Artificial gaussian noise is added to the agents both to prove robustness and to enable the use of the Montecarlo method to obtain faster convergence (see section 3.7). As a result, the probability of having the energy associated with agent i equals to E is:

$$Pr(E_i = E) = \frac{1}{\sqrt{2\pi\delta^2}} exp[-\frac{(E - \bar{E}_i)^2}{2\delta^2}] \tag{3.17}$$

where $\bar{E}_i$ is the statistical mean of the random variable E_i. The distribution variance δ and the temperature T are related as $T = 2\sqrt{2}\delta$.

3.5 Resource Bound

In this section we bound the amount of computational resources available to the agents (see section 1.1 for related approaches). This is useful to optimize the management of agents' memory and their access to processors (and consequently access to communication bandwidth, representation space, sensors, effectors and so forth). We want that when an agent is active the amount of computational resources used can not be available to other agents until it will end its activities. With an energy bound like this and the cost of agents' execution taken together, we can have an efficient system with only few active agents at the same time. There is also another interesting consequence: together with the local excitation and local inhibition given by the energetic network's links this resources bound introduces global inhibition too. Together they can play the role of positive feedback and negative feedback that produce self regulating and self sustaining activation patterns, which in the AAN architecture are identified by agents' attractor [2].

Storing only biased patterns (Ami89) we can force the energetic network to have a bounded number of active agents. With biased patterns we can represent agents' attractors with only a fixed average number of variables for which $A_i = 1$ (active agents). We can call this kind of agents' attractor *biased agents' attractor*. Let us define the bias probability as in equations (3.18) and (3.19).

$$P(A_i = +1) = \frac{1}{2}(1 + a) \tag{3.18}$$

$$P(A_i = -1) = 1 - P(A_i = +1) \tag{3.19}$$

Where we set $a \in [-1, 1]$, called *bias parameter*, according to the average number of agents that we want to be active.

[2] These patterns sometimes are called auto-catalytic set (Kau93).

Then we have to constraint the energetic network's dynamics on a subspace of its 2^N possible configurations. This is the subset of states with an average agents' activation given by the bias parameter a. The idea is to add the constraint of equation (3.20) to the dynamics of the visited states.

$$\frac{1}{N}\sum_{i=1}^{N} A_i = a \tag{3.20}$$

We can fix the average amount of agents' activation and implement the constraint introduced above in equation (3.21), adding a special term E_0^e to the time independent energy component E_i^e defined in equation (3.16). As in (Ami89), the value of E_0^e can be found from the relation $\tanh \frac{E_0^e}{T} = a$.

$$E_i^e = \sum_{j=0, j\neq i}^{N} J_{ij} + B_i - R_i + E_0^e \tag{3.21}$$

3.6 Learning and Execution

Once we have defined the concept of agent's attractor one important question is to understand how the energetic network can learn the right ones. We know that, as in Hopfield networks, the set of agent's attractors of the AAN are determined by the weights of the energetic network. Moreover, not only to a certain set of weights corresponds a set of agents' attractors, but also the dynamics within the energetic network used to spread energy among the agents during their execution. So one question that we will investigate in this section is how to assign the right values to the AAN links matrix.

To learn the values of the weights of the energetic network, that is to determine the set of agents' attractors of the system, we have two possible choices: to learn automatically a suitable set of agents' attractors online or to insert manually a set of agent's attractors that we already know to behave well. This choice is made according to our knowledge about the problem that we want to solve and according to the amount of adaptability needed when the solutions are changing online.

In addition an alternative online algorithm that focuses more on context awareness will be shown in section 7.2 as a future development of the work presented in this thesis.

Finally we will show how the AAN can recall the right agents' attractor online according to current context.

3.6.1 Offline Learning

If we already know the set of agents attractors that have to be stored we can use the Hopfield network s Hebb rule that we already saw in equation (3.6) of section 3.1. As we said the Hebb rule is generally used to memorize a set of arbitrary patterns in a Hopfield network.

Let us repeat how it works in the context of the AAN architecture: for each known agents' attractor $A^{\mu} \in \{+1, -1\}^N$, with $\mu \in [1, \ldots, p]$ here, the energetic network's weights, initialized to zero, are modified adding the term $\Delta J_{ij}^{\mu} = A_i^{\mu} A_j^{\mu}$. Integrating over all agents' attractors, the final weight's are given by:

$$J_{ij} = \begin{cases} 0 & \text{if } i = j \\ \frac{1}{N}\sum_{\mu=1}^{p} A_i^{\mu} A_j^{\mu} & \text{otherwise} \end{cases} \tag{3.22}$$

So we can choose the agents' attractors and then directly modify the weights of the energetic network using the Hebb rule. What we have to care about is to not store more agents' attractors than the number allowed by the networks capacity, as we showed in section 3.1.

In order to memorize p biased agents' attractors (section 3.5) we have to modify the Hebb rule of equation (3.22) as in equation (3.23).

$$J_{ij} = \begin{cases} 0 & \text{if } i = j \\ \frac{1}{N}\sum_{\mu=1}^{p} (A_i^{\mu} - a)(A_j^{\mu} - a) & \text{otherwise} \end{cases} \tag{3.23}$$

This offline approach will be tested in the experiment of section 3.11.

3.6.2 Query Algorithm and Online Learning

Once the agents' attractors are memorized (inserted manually or online, as we will see below) we still need a method to recall them during the execution of the task. In this section we will show the query algorithm (see algorithm 1) where the agents build an initial configuration trying to make the network arrive to the best agents' attractor. Starting from that initial configuration the energetic network will converge to an agents' attractor where active agents execute their operations. These two phases are called respectively *adequacy step* and *execution step*.

The energetic network is an associative memory, so the idea is to let the agents to decide locally if activate by themselves, reading their portion of the input space and using a test function that will determine if it is the right moment to be active. These local and autonomously tests will decide an initial configuration of agents, which propose themselves as active still without executing. The decision of this proposal ends the adequacy step that has the aim to produce an initial configuration that should belong to the best agents' attractor's basin of attraction. Finally when the energetic network arrives to an agents' attractor the execution step starts running all active agents.

The energetic network's weights can also be modified online by a learning algorithm that optimizes the collective behavior of the agents facing different inputs in different moments. This algorithm constitutes the *learning step* of the query algorithm, it is shown in algorithm 2 and works as follow.

According to the problem that the architecture is attempting to solve, the particular semantics of this learning process is defined by the user who has to implement the test functions of the adequacy step. As we said this function evaluates locally the quality of agents' operations and it has binary output: *success* or *failure*. When the learning algorithm is running, the success and the failure of agents determine the links' weights so that the computational resources (energy) of the system are distributed among agents. This energy distribution is developed to achieve coordination among agents: the idea is to have in the same agents' attractor a set of agents that to have success need to be synchronized. In this way they can be active and execute together in order to benefit of their operations.

The idea behind this learning algorithm is simple: we want that agents that have succeeded at the same time belong to the same agents' attractor; on the contrary when an agent has succeeded and another has failed they should belong to different agents' attractor. To implement this idea we proposed algorithm 2, which is a subroutine of algorithm 1, that uses a learning update developed taking inspiration from the Hebb rule of equation (3.22). Note that in algorithm 2 the N in the denominators of the learning updates is taken directly from the Hebb rule. Moreover the learning rate c has to depend on the number of agents N. This is true because observing the Hebb rule we find that the J_{ij} have minimum value $-\frac{p}{N}$ and maximum value $\frac{p}{N}$, where $p = 0.138N$ if the agents' attractors are totally uncorrelated.

It is possible to prove (Roj95) that imposing to the energetic network the absence of self interactions, links with symmetric weights and a complete graph topology, it is assured its convergence to stable agents' attractors. As we can see the resulting energetic network topology produced by the

Algorithm 1 Query Algorithm

```
Input: N number of agents
  activity[1 ... N] agents' activation variables, initially all set to -1
  success[1 ... N] result of the success/failure test of agents
  J[1 ... N ][1 ... N] zero-valued energetic network' links
  NumIterations number of learning cycles
  T temperature
  average_activity average number of active agents (Computational Resource Bound)
  num_active current number of active agents
  num_successes current number of active agents that had success

  ADEQUACY_STEP(parameter i) measures the adequacy of agent i and returns the result of its success/failure test
  EXECUTE_STEP(parameter i) operations executed by agent i
  LEARNING_STEP(parameters N, activity, J, c, a) updates (and returns ) with learning rate c the links J of an energetic network of N agents with activity activity and bias parameter a
Output: J[ ][ ] new links of the energetic network

  $a \leftarrow 2.0 * average\_activity - 1.0$
  Choose average_activity random agents to be active
  for $k := 1 \rightarrow NumIterations$ do
    for all active agents $n$ do
      $success[n] \leftarrow ADEQUACY\_STEP(n)$
    end for
    if $num\_successes < num\_active$ or $k = 0$ then
      $J \leftarrow LEARNING\_STEP(N, activity, J, c, a)$
    end if
    for all active agents $l$ do
      if not $success[l]$ then
        $activity[l] = -1$
      end if
    end for
    repeat
      energetic network's update (equation (3.12)) through Montecarlo method
    until energetic network reached an agents' attractor
    for all active agents $m$ do
      $EXECUTION\_STEP(m)$
    end for
  end for
```

Algorithm 2 LEARNING_STEP

```
Input: N number of agents
  activity[1 ... N] agents' activation variables
  J[1 ... N ][1 ... N] Energetic Network' links
  success[1 ... N] result of the success/failure test of agents
  c learning rate
  a bias parameter (section 3.5)

Output: J[ ][ ] new links of the Energetic Network

  for all active agents $i$ do
    for all active agents $j$ do
      if $success[i]$ and $success[j]$ then
        $J[i][j] \leftarrow J[j][i] \leftarrow J[i][j] + \frac{c-a}{N}$
      else if not $success[i]$ and not $success[j]$ then
        $J[i][j] \leftarrow J[j][i] \leftarrow J[i][j] + \frac{c-a}{N}$
      else if $success[i]$ and not $success[j]$ then
        $J[i][j] \leftarrow J[j][i] \leftarrow J[i][j] - \frac{c-a}{N}$
      else if not $success[i]$ and $success[j]$ then
        $J[i][j] \leftarrow J[j][i] \leftarrow J[i][j] - \frac{c-a}{N}$
      end if
    end for
  end for
```

proposed learning algorithm fulfills these constraints. So it is sure that after the learning step a set of stable agents' attractors have been learnt.

According to this procedure we will have positive links between agents that have succeeded at the same time and negative links between agents that have succeeded in different times. Let us try to understand why this is sufficient to make agents synchronize properly, that means to have the right set of agents in the same agents' attractor. A first possible explanation is that when two agents are connected with a positive link, when one is active the other receives energy through this link. Or when two agents are connected with a negative link when one is active the other will have less energy. But this kind of reasoning is not telling anything about the asymptotic behavior of the energetic network. To obtain this kind of information in the following section 3.7 we will focus more on the physical interpretation of the energetic network's dynamics. In addition this online approach will be tested in the experiments of sections 3.9 and 3.10.

3.7 Coordination measures and Frustration

To show that algorithm 2 learns agents′ attractors with a coordinated set of agents, in this section we give a physical interpretation of the dynamics of the energetic network. We recall that Hopfield networks are recurrent neural networks based on ideas common to those used in spin glasses (PVM87). In statistical physics the spin glass is a substance in which the atomic spins are oriented in random but fixed directions. A spin glass is a disordered material exhibiting high magnetic frustration. Here *frustration* refers to the inability of the system to remain in a single lowest energy state (the ground state). Indeed spin glasses have many ground states, that in Hopfield networks correspond to many memorized patterns. To not confuse the energy as computational resources in the AAN architecture with the energy function of physical systems we will call the latter ″hamiltonian″.

In absence of noise we can find the agents′ attractors as the configurations with minimum energetic network′s hamiltonian $H(\mathbf{A})$. The corresponding hamiltonian of equation (3.8) for Hopfield networks in the AAN context is the following equation (3.24).

$$H(\mathbf{A}) = -\frac{1}{2}\sum_{i,j,j\neq i}^{N} J_{ij}A_iA_j \tag{3.24}$$

$H(\mathbf{A})$ is the Lyapunov function of the energetic network dynamics. In the presence of noise we have to consider another Lyapunov function: the free energy. The general definition of *free energy* for a configuration $\mathbf{A} = \{+1,-1\}^N$ of active agents and silent agents is $f(\mathbf{A}) = H(\overrightarrow{A}) - TS$ where S is the entropy of the Energetic Network.

Now consider a set of people that we have to divide in two groups: the group labeled as *+1* and the group labeled as *-1*. About these people we also know that they can be connected with each other with friendship relationships or enemy relationships. We want to divide the people in a way that makes people happy as much as possible: that is to put as many friends as possible in the same group and as many enemies as possible in different groups. This problem is equivalent with the problem of finding the minimum configuration of free energy of a spin glass (PVM87). According to this interpretation to find this configuration means to maximize the happiness. Since a spin glass and a Hopfield Network are very similar in our context , what does this consideration means according to the AAN architecture?

The two groups *+1* and *-1* correspond to active and silent agents, respectively, whereas the friendship and enemy relationships correspond to positive or negative links among agents. As we said before, following its dynamics the energetic network minimizes its free energy until it arrives to a minimum where we find an agents' attractor. According to the previous interpretation this means to maximize the number of "friend" agents (agents connected with each other by a positive link) to be active and silent at the same time. That is to maximize the number of friend agents that belong to the same agents' attractor. At the same time minimizing the free energy means to maximize the number of agents that are active when their enemy are silent and vice-versa. That is to maximize the number of enemy agents that belong to different agents' attractors. 5

One interesting property is that when the system is frustrated it has not a unique state of minimum free energy, but several minimum states called *metastable states*. All these states have an equal amount of free energy, which is greater than the value of the not frustrated case.

So we can say that the energetic network's free energy is a suitable quantitative measure of global coordination among agents: the lower is its value in the stationary state and more the agents are coordinated. On the contrary the higher is its value in the stationary state and more the agents are frustrated. So, typically, to find an efficient collective state a trade-off between frustration and synchronization among the agents has to be found.

Another possibility is to measure the synchronization among two agents locally. To do this we can use the *correlation function* that measures how much two microscopic variables are correlated in different positions [3]. For

[3]In statistical physics critical phenomena, as phase transitions, are studied using the

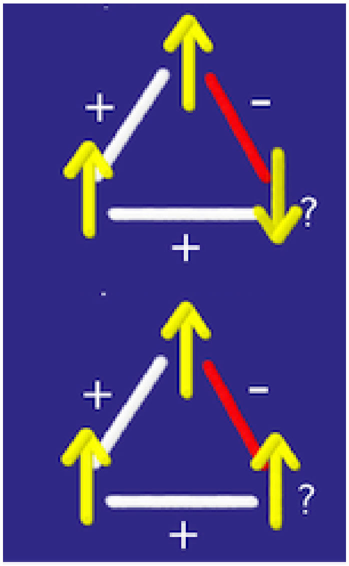


Figure 6: An example of frustrated Energetic Network. Up headed arrows are active agents and down headed arrows are silent agents. There is not an unique choice to make happy the bottom right agent. Figure taken from (oE).

the AAN these microscopic variables are the agents′ activities $A_i \; for \; i = 1, \ldots, N$.

The correlation function among two agents i and j is given by equation (3.25).

$$C_{ij} = \langle A_i A_j \rangle - \langle A_i \rangle \langle A_j \rangle \tag{3.25}$$

where the angle bracket defines a mean on the distribution given by the temperature.

Hence, after an AAN has learned how to distribute its computational resources among agents for a given problem, we can use the free energy and the correlation function to analyze the agents′ coordination globally and locally respectively.

correlation function which describes how the dependence among microscopic components of a system produces the necessary synchronization to transit in a new macroscopic state. Indeed during a phase transition typically the correlation function diverges as the system size is infinite. Indeed we have an high level of cohesion when we find a long range correlation; or in other words when the correlation function scales with the system size (scale free correlation). Here cohesion means that the information is moving really fast among agents and that the system is robust to perturbations.

3.8 Some Useful Tools

In this section we will briefly review some tools taken from Hopfield network theory that can be used to analyze the energetic network's dynamics and understand the nature of the learnt agents' attractors.

When we are using an online learning algorithm (as algorithm 2) we can not know a priori the identity and the number of the learnt agents' attractors. So to know which agents' attractors have been learnt we can study the free energy because, as we said, they are located in the minima of this function. Also if the free energy of a Hopfield Network has a known analytic expression the only way to compute these minima is numerically. This problem is known to be computationally hard because it belongs to the NP complexity class. To find an approximate solution we can use the *Branch & Bound* algorithm, a variant of the Simplex algorithm, that was already successfully used to find the free energy minima of a spin glass, as in (KH78) (HDK84) (SDJ+95). So it can be easily extended to be used for the energetic network of an AAN.

Furthermore we can measure how much the agents' attractors interfere with each other using the *signal to noise analysis* (Ami89). Indeed this is an important point because, if we want to use efficiently the capacity of the energetic network's memory, that is to let it to memorize as many agents' attractors as possible, the correlations among them should be minimum. This become even more important when we are using the Hebb rule of equation (3.22) because in this case we can check before if the memorization of the agents' attractors is feasible. As we said, if the attractors of a Hopfield Network are correlated its theoretical capacity of $0.138N$ decreases, because these correlations produce an interference between attractors that can disturb their stability. The signal to noise analysis can help us to control this phenomena. Note that if we have an high number of agents, compared with the number of the memorized agents' attractors, these can be also correlated with each other.

Another interesting measure about agents' attractors concerns their *basin of attraction*. It measures how many initial configurations will flow in a certain agents′ attractor and it can estimated numerically as follow (Tir95): initially we can assume that the energetic network is in an initial configuration $A_i(0)\ for\, i = 1, \ldots, N$, which is a perturbation of some learnt agents′ attractor $A^1 \in \{-1, +1\}^N$ as represented in equation (3.26).

$$A_i(0) = A_i^1 \xi_i \tag{3.26}$$

where ξ_i is a random variable which takes values $+1$ and -1 with the probability shown in equation (3.27).

$$P(\xi_i = 1) = 1 - q \qquad P(\xi_i = -1) = q \tag{3.27}$$

An estimate of the basin of attraction can be obtained computing the maximum value of q such that we find the convergence to the agents′ attractor A^1.

In statistical physics many macroscopic properties of a system that has reached the equilibrium can be expressed as an expectation of the canonical distribution of equation (3.9). In the AAN architecture the equilibrium is reached when the energetic network has reached an agents′ attractor. More the system is big and more this expectation corresponds with the real macroscopic observations. The mean $\langle M \rangle$ over the canonical distribution of a certain observable M related with the agents at temperature T is represented by equation (3.28).

$$\langle M \rangle = \sum_{\mathbf{A}} M(\mathbf{A}) P(\mathbf{A}) \tag{3.28}$$

An example of the expectation over the noise introduced in equation (3.17) was given to define the correlation function C_{ij} of equation (3.25).

In such kind of averages the sum is computed over the 2^N configurations representing all the possible N combinations of agents′ activities. Indeed 2^N is the cardinality of the set $\{+1, -1\}^N$. This means that if we want to compute this sum with a deterministic method it would be necessary an exponential number of steps. Moreover, when the number of agents is large, also to simulate the energetic network′s dynamics can be computationally expensive: we need $O(N^2)$ operations for each time step from the initial configuration until an agents' attractor is reached (this time is called *mixing time*). Such kind of computations can be not tractable also for a very powerful computer if the number of agents is large.

A more tractable approach consists in utilizing the *Montecarlo method*. The Montecarlo method is an importance sampling strategy: it can improve the performance of a numerical estimate visiting only the most important system's configurations, avoiding those configurations that contribute less to what we want to estimate. It was successfully used for Hopfield networks (Tir95) and so it can be used for the energetic network of the AAN architecture as well. Indeed the Montecarlo method is reported in algorithm 1 and it was implemented for the experiments of the following sections.

If the system is frustrated we have to deal with a high number of metastable agents' attractors, divided each other through high barriers in the free energy landscape, which makes difficult to go from one minimum to another. Indeed the energetic network's free energy is a very complex function with many minima and maxima that can bring the Montecarlo method to be blocked in a local minimum that is not an agents' attractor. To solve this problem we can use transitions which augment the free energy instead to diminish it, because they give a possibility to exit from a local minimum. This is a good strategy because local minimums which are not agents' attractors have a higher free energy and a smaller basin of attraction. The solution is to use an amount of temperature which is high enough to bring the energetic network out from a spurious basin of attraction and at the same time to bring it to an agents' attractor with lower free energy. Indeed observing the distribution of equation (3.9), with zero temperature $T \to 0$ the canonical distribution in concentrated on the agents' attractors. On the other hand if we consider a very high temperature, let us say $T \to \infty$, every configurations will have equal probability.

The optimal strategy is to start the Montecarlo method with a high temperature ($T = 1$) and then diminish the temperature continuously as we are getting near to the absolute minimum. This method is called

Simulated Annealing (KGV83) which was already applied successfully on Hopfield network to improve the convergence process to a memorized attractor.

With this method we have to choose for every possible value of the temperature, from higher values to lower ones, the number of cycles P which the energetic network has to wait before to decrease the temperature again. The Geman and Geman theorem (GG84) guarantees the wished result with probability 1. Indeed let n the number of steps which are used to update all the agents. If we varies the temperature according to the law

$$T(n) = \frac{C}{1 + \log n} \tag{3.29}$$

then we will reach the free energy minimum of the energetic network. The C constant can be determined doing several tests. In practical applications a exponential decay law for the temperature is used until a temperature T_0 is reached; then a logarithmic decay law is used.

3.9 First Application, Optimization: The Board Game experiment

In this section we give a simple application to explain how the introduced architecture works in practice: a simulation of the AAN architecture playing with a board game. Although simple, this example start to illustrate what is the nature of the concepts introduced above.

This board game constitutes a distributed constraint optimization problem (SLB08). As we said in section 3.3, the AKIRA energetic model was already applied to solve problems of combinatorial optimization, as the edge coloring problem (Pez09).

Hopfield networks were successfully used for optimization problems too, for example the traveling salesman problem or the matching problem (HT85). As we said in section 3.1 adopting this approach means using neurons to represent the variables of the problem and defining a hamiltonian (as in equation (3.8) of section 3.1) that has the solution of the problem as its minimum configuration. As the Hopfield network continues to update its state the hamiltonian decreases until the dynamics arrives to a minimum where the solution is represented.

Although this is a powerful approach, no explicit method exists to write such hamiltonians, which every time have to be hand-written according to the specific problem under investigation. In addition when our optimization problem is enough complex, as in the case of the board game that we are going to describe, no such hamiltonian can be found. Hence, using the AAN architecture, in this section we will see an alternative approach to solve optimization problems with Hopfield networks when we are not able to find a hamiltonian. Instead of defining the hamiltonian offline, that corresponds to defining the synaptic matrix, we will use the AAN online learning algorithm (algorithm 2 of section 3.6.2) to learn this matrix according to the success of certain values of the variables evaluated locally during the execution.

Let us consider a board with two rows with 10 cells each. The first row belongs to the Attacker player and the second to the Defender player (which is the role played by the AAN architecture). For each turn first the Attacker chooses five pieces to deploy in the first row on the board; he can choose among 4 possible kind of pieces: foot soldier, pikeman, knight and archer. Then the defender has to answer deploying his five pieces in the second row on the board with the aim to get more points than the opponent. The amount of points which each player can obtain depends both of the positions of the pieces and their type. The rules are shown in Figure 7.

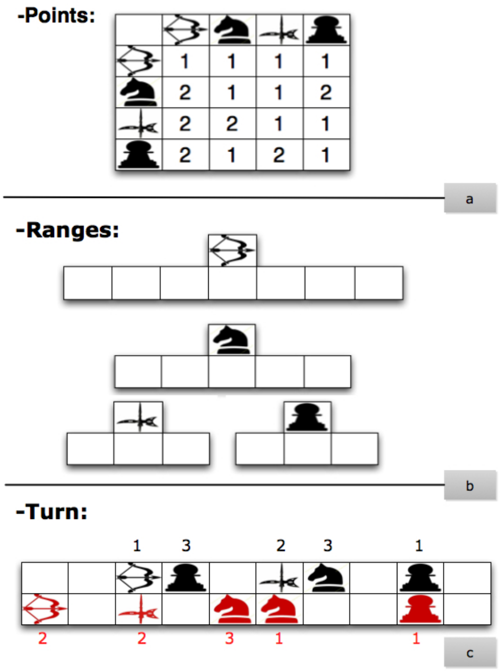


Figure 7: a) Points Table. b) Pieces ranges. c) Example turn.

In box (a) of Figure 7 we see how many points pieces can obtain fighting against other pieces according to their roles; in (b) is shown the pieces' range of attack; finally in (c) we represented a sample turn with the amount of points taken by each player. When an Attacker's piece is in front of a

Defender's piece in the same column they are obliged to fight. Instead, if there is no opponent in front of a piece it can fight diagonally according to its range, gaining an additional point. Among all possible opponents that a piece can find in its different diagonal positions the one which maximize points is always chosen.

In order to let the AAN architecture to play the Defender role in this game we used agents to represent the idea "put a piece in a certain position". So we distribute the problem among 40 agents, where the activity of the first dozen represents the ten possible positions of archers, the activity of the second dozen represents the ten possible positions of knights, and so forth (the configurations with more than one piece in the same position are excluded from the dynamics). So for example if the agent 17 is active the AAN architecture is considering the possibility to put a knight in position 7.

What we want is to find and memorize winning defense configurations as agents' attractors. Each of them will be composed by five active agents placed in winning positions according to a certain attack configuration. In order to constraint the dynamics over five active agents I used the resource bound introduced in section 3.5. To find and memorize these agents' attractor I used the AAN online learning algorithm.

At the beginning of a match the architecture considers to deploy five random pieces on five random positions. Then inside the adequacy step of the algorithm these agents compare locally their positions and their roles with the attack configuration of the opponent. Agents that with their position make the architecture gain more points, and at the same time loose less points, are chosen to be active in the initial configuration. Then the energetic network is updated until it reaches an agents' attractor. If the energetic network already experienced a similar attack configuration then the initial configuration produced by the adequacy step is used to recall a memorized agents' attractor successful with that similar attack configuration. In the execution step of the algorithm, if this agents' attractor adds the right remaining pieces, the complete defense configuration is deployed. Otherwise if this attack configuration is unknown a new agents' attractor is learned: the topology of the energetic network is updated in order to transfer the energy from agents that represent pieces that are losing to agents that represent pieces which are gaining more points. At the end only five successful agents are active in the new learned agents' attractor which is deployed as defense configuration. In the next turn, with a new attack configuration of the opponent, the execution of a new adequacy step and a new execution step brings the energetic network to replace the current agents' attractor with another one able to win in this new situation.

The fact that agents cannot be in the same position at the same time or the fact that an agent can engage the potential enemy of other agents represent simple coordination issues. This implementation was tested successfully on several matches, showing always the ability of the AAN architecture in playing this game. In figure 8 we show a couple of defense configurations memorized as agents' attractors and deployed by the AAN architecture (red color). As we can see in the first turn the AAN architecture gets 12 points versus the 8 points of the Attacker player; in the second turn the AAN architecture gets 13 points versus the 6 points of the Attacker.

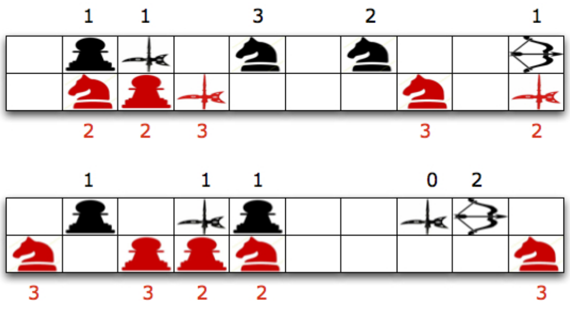


Figure 8: Two turns won by the ANN architecture.

3.10 Second Application, Classification: The Attractor Expert Network

In this section we implement and extend the ideas of the mixture of experts model (JJNH91) using the attractor agent network architecture. We have described the mixture of experts in section 2.2.

Let us consider an AAN having as agents a set of feed forward neural networks that we call experts. We will use this implementation of the AAN architecture, that we call attractor expert network (AEN), to solve a classical classification problem with a mixture of experts approach: we want to know if a point belongs to the upper or lower half plane of a complex function. The developed AAN learning algorithm exploits cooperation issues and makes similar experts belong to the same experts' attractor.

Thinking that in a Hopfield network there are not input neurons, as in a feed forward neural network for example, the input has to be codified in the whole initial neural state. In the AAN architecture until now we used the query algorithm (algorithm 1 of section 3.6.2) to build such initial activities' configurations. This approach makes the agents decide autonomously and locally when being part of the initial configuration, on the basis of their adequacy investigated online. Here we present an alternative approach to solve this problem that manages the input outside the energetic network using the gating network of the mixture of experts model. This is used both as a mapping function from the input data to the initial activities' configuration and as a procedure to specialize the experts. Thanks to this implementation of the AAN architecture we can obtain an *input module* that could be used in other further applications. Moreover this implementation offers a method to specialize our agents that until now we considered somehow already specialized.

As we can see from figure 9, what the attractor expert network adds to the mixture of expert model is a set of connections between the experts (in the figure represented in blue). The experts are connected through the energetic network that stores their contextual relations. These relations are stored using the AAN online learning algorithm (algorithm 2 of section

3.6.2).

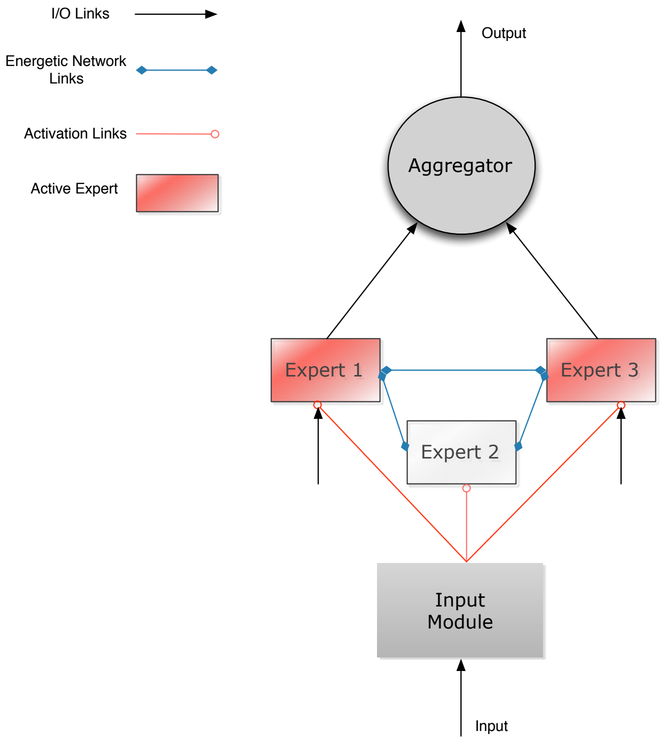


Figure 9: In this figure we can see the attractor expert network's components and how they interact with each. We have the expert networks, the input module used to indicate an initial network configuration and an aggregator module used to combine the results of the experts in the experts' attractor.

This makes the architecture able to exploit cooperation issues among the experts and not only competitive ones as in the MOE model. Indeed in the classical MOE we are usually interested in finding one leading expert (the selector choose the output of only one expert with the probability of equation (2.1) of section 2.2). On the contrary in the AEN we want to select a set of experts, related by some contextual relations, to be active at the same time. These experts should cooperate to solve a complex problem in order to benefit of the operations of the others. This set of experts is selected by the experts' attractor of the energetic network that is more similar with the initial configuration chosen by the input module. In fact as we know an experts' attractor identifies as set of active experts and a set of idle experts. The output of the attractor experts' network is obtained after the arrival to one of its attractors, combining the output of the active experts through an *aggregator module* that computes a compositional function with these outputs as arguments (in the following experiment we will use the average function).

Moreover in the MOE model all the experts receive the input and compute their outputs. Only later these outputs are filtered by the gating network to select the most appropriate expert. On the contrary in the AEN only active experts receive the input and compute their output. So in the AEN the filtering process of the gating network is moved from a post processing of all experts outputs to the preprocessing of which experts has to be active in initial configuration, with an evident gain of efficiency.

Once the experts are specialized, for each data of the training set, after the input module has identified the initial activities configuration, we spread the input to the active experts' input units. Then, to store the experts' attractors, we train the energetic network for each data point through the AAN online learning algorithm, which considers the success and the failure of every active experts. This will make the experts that are specialized on related input classes (necessary to each other or simply similar) belong to the same experts' attractors. In the AEN the success of an expert (the adequacy step of algorithm 2) is defined through the difference between the output units vector and the desired output vector (known in supervised learning for data points of the training set; not known for data points of the test set used in the evaluation phase described below). Note that in this implementation of algorithm 2 we do not modify the weights of the failing experts' links.

Once the learning phase ends we can start to classify data points of the test set spreading the input to the input module. This module will try to build the most adequate initial activities configuration. Then the energetic network dynamics will bring the system to the closer experts' attractor, where the active experts will perform their classifications (the execution step in algorithm 2) and their output will be composed in the aggregator module to obtain an unique final output.

3.10.1 The Experiment

Here we test the attractor experts network on a simple experiment: the classification of a point according to its belonging to one of the two half planes defined by a complex function. I chose a piecewise linear function to verify if the system is able to learn and recall the same experts' attractor for similar intervals.

The experts' architecture and the target function have to be chosen in order to not let an expert being able to classify points belonging to all the intervals. This can happen if the piecewise function is so simple that it can be easily learnt by a single expert. Another possible reason corresponds to having experts with an architecture enough complex to classify points of every interval.

According to these desired properties I implemented experts able to do at most a linear classification. They have an architecture with enough input neurons that together can codify the two number necessary to represent an input point (x, y) with $x, y \in [0, 65000]$. So these two numbers can be represented with 32 bits, 16 for the x and 16 for the y. Then I chose to give to the experts' architecture one hidden neuron and one output neuron. Values of the output neuron close to 1.0 represent the upper half plane and values close to 0.0 represent the lower one.

The chosen piecewise linear function can be found in the following eq. 3.30 and is represented in figure 10.

$$y = \begin{cases} 5x & \text{if } x \leqslant 15000 \\ -0.3x + 20000 & \text{if } x < 15000 \leqslant 32000 \\ -0.7x + 10000 & \text{if } x < 32000 \leqslant 47000 \\ -0.7x + 50000 & \text{if } x > 47000 \end{cases} \tag{3.30}$$

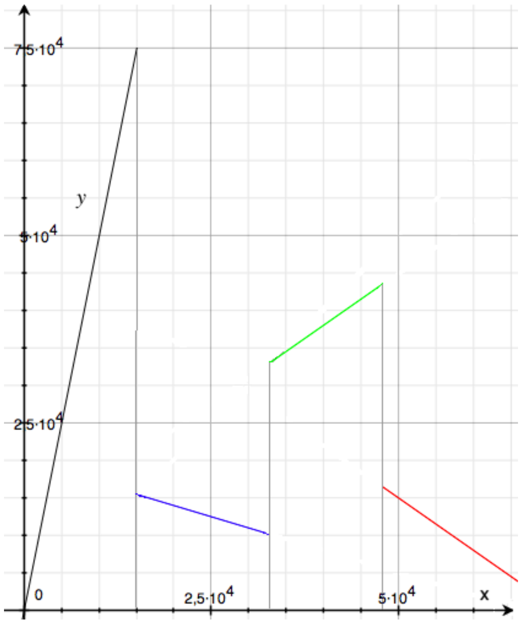


Figure 10: Piecewise target function of equation (3.30) used for the classification task.

The facts that this function is composed by four different rects and that a single expert can at most achieve linear classification should bring the system to specialize the experts.

To solve this classification problem we consider an energetic network composed of ten experts at zero temperature. The architecture chosen for the input module is composed of 32 input neurons, 6 hidden neurons and 10 output neurons.

To start showing the behavior of the attractor expert network in the described task, in the following histograms (figures 11-13) we represent the number of times that the experts were active according to the kind of input data. This will let us to evaluate the input module ability to specialize the experts and activate them for the initial activities configuration. Let us label the four domain's intervals, where the four rects are defined, with the four nominal integers 1, 2, 3 and 4.

During almost 4000 training cycles of the input module only three experts specialize their behavior, showing that only these were enough for the classification task. For this reason we will show only the histograms related to these experts.

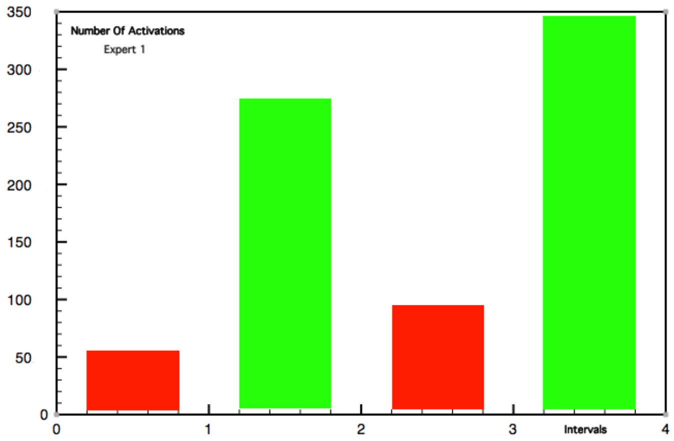


Figure 11: Histogram that represents the number of times expert 1 was active according to the kind of input data.

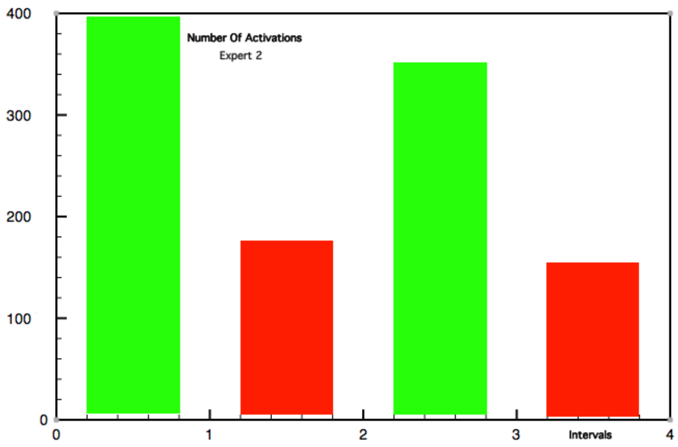


Figure 12: Histogram that represents the number of times expert 2 was active according to the kind of input data.

During its execution the AAN online learning algorithm stored two experts' attractors: in the first one only expert 2 is active, in the second one are active the experts 1 and 3.

If we compare these experts' attractors with the histograms showed below we can find an interesting result: the system specializes the two experts' attractors according to the sign of the rects' angular coefficients. So in the first experts' attractor is active one expert specialized in classifying on half planes defined by rects with positive angular coefficient. In the second experts' attractor are active two experts that cooperate in classifying on half planes defined by rects with negative angular coefficient. As is shown in figure 9 the two experts cooperate averaging their output neurons' values.

To evaluate the described system we used a test set composed of 1000 data points sampled uniformly from the whole function domain. The overall system performance consists of the 90% data points well classified. Usually the selected experts were the most adequate according to the input data.

In addition, to underline the advantages of the attractor expert network over the classical mixture of experts model, we compared the performance

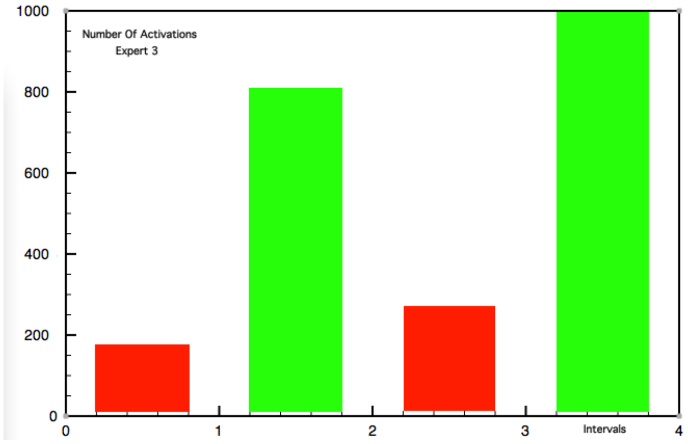


Figure 13: Histogram that represents the number of times expert 3 was active according to the kind of input data. Note the similarity with expert 2.

of the experts belonging to the second agents' attractors taken alone with the results of their coordinated behavior. Considering only the portion of the domain where these experts are specialized, the experts taken alone had performance of respectively 82% and 86% of well classified data points. On the other hand the average of their output neurons obtained a 91% of well classified data points in the restricted domain.

These results proved that the AEN is able to store similar experts in the same experts' attractor. Moreover it exploits the cooperation among the experts, which can be more convenient than selecting only one expert at the same time.

3.11 Third Application, Active Vision Categorization: The Attractor Predictor Network

In this section we will use the AAN architecture to solve a categorization task of computer vision (Pin06) (LC05). The task consists in determining whether the input image represents a giraffe, a horse, a cat or a dog. In figure 14 we have several examples of the sketched animals (cats, dogs, horses, giraffes) that have to be categorized. Note that in this data set little variations lead to different categories, making this categorization task quite challenging.

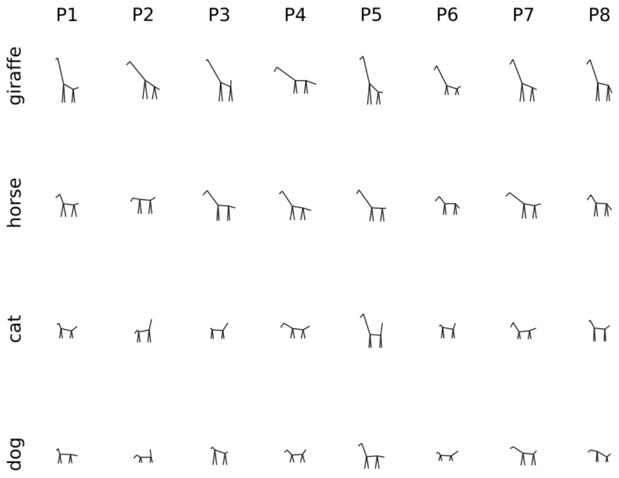


Figure 14: Stick animals used as stimuli for the categorization task of this section and sections 4 and 5. Figure taken from (SGS10a).

We will show how to build an AAN to simulate an artificial eye that can be moved to scan different areas of the input image. The choice to consider a limited filed of view was taken to investigate overt attention in an active vision framework (Bal91). To deal with situatedness and embodiment (Bar08) we want to implement an action oriented construction of categories so, accordingly, we chose to commit to an active vision approach.

In this implementation of the AAN architecture, that we call Attractor Predictors' Network (APN), the energetic network connects several sensorimotor features predictors. Each predictor is an agent specialized on a specific sensorimotor strategy involving two geometrical features and a motor command (see below). The APN merges the asynchronous received commands by different predictors to obtain the next fixation point. Then, to perform a categorization, it accumulates evidence on predictors' success during the visual exploration. Predictors that recognize their features in the filed of view, and predict correctly the relation between them through the execution of a motor command, are consider successful and so gain energy and are kept active.

We assume that a category is characterized by the combination of such sensrimotor strategies, so that the activation of a predictors' subset (i.e., predictors' attractors) can well identify a category. This boils down here to the control of eye movements during categorization tasks, with overt attention emerging from the intertwined activity of saliency based (bottom-up) and anticipatory (top-down) processes.

3.11.1 Input features

The APN implementation of our architecture includes a small foveated area that can be freely moved around the stimulus to be categorized, and that will be used as the only sensory input of the system. Due to of the reduced size of this visual area, the system has only partial knowledge of the environment and needs to explore it through saccades to obtain information and disambiguate stimuli. Actions thus consist in saccades (S) that instantaneously move the foveated area from one position to another.

We preprocessed the visual input in order to be focused on categorization and decision-making processes, instead of being focused on technical details of signal processing and computer vision, which are out of the scope of this thesis. Moreover this preprocessing keeps the system complexity low enough to be simulated in real-time on standard computer architectures.

Considering the specific class of stimuli that has to be categorized (black and white, stick-like shape and structure), we chose to equip the system with the most informative features representation. Starting from the computation of neuro-inspired saliency map over the input images (IKN98) we found the strongest saliency around the joints of the stick animals. Moreover we found that oriented Gabor filters on a single scale constitute the most sensitive feature descriptor for stick's orientations. For each eye's position only the stick animals' joints belonging to the field of view are taken as feature points (F_i), lowering the dimensionality of the input.

The complete feature descriptor is defined as follow: first we have the coordinate (u, v) of the joint point's position in the field of view; then for the sticks' orientations we have a vector representing the responses of M Gabor filters away from the joints $(o_j)_{j\in[1,M]}$ (in the following we chose $M = 16$). In other words, to obtain a fixed number of values representing the presence of an arbitrary set of orientation in the field of view we used Gaussian tuning curves with wide selectivity profiles along orientations (see equation (3.31) and figure 15).

Note that, although all the informations given by the feature descriptor, the task remain still quite challenging for the high ambiguity between categories. For example there is no way to discriminate between back and front legs without moving the field of view (saccading) from one area to another.

$$o_j = \max_l \exp - \left(\frac{M(\theta_j - \rho_l)}{2\pi} \right)^2 \tag{3.31}$$

where ρ_l is the set of angles formed by the sticks coming out from the considered joint and $\theta_j = -\pi + j\pi/M$.

The full feature descriptor is given by the vector $F_i = (u, v, o_1, \ldots, o_M)$. In addition we can compare two features using the similarity measure (F_1, F_2) defined in equation (3.32).

$$\sigma(F_1, F_2) = 1 - e^{-\frac{\|F_2 - F_1\|^2}{\sigma^2}} \tag{3.32}$$

where $\|.\|$ is a norm in R^{M+2}.

3.11.2 Predictors

Predictors are agent specialized in verifying particular sensorimotor contingencies through the exploration of the visual input. To be more specific a predictor (P_i) is defined by a triple $(F_i^{src}, S_i, F_i^{tgt})$ where F_i^{src} is the source feature of predictor i, F_i^{tgt} is its expected target feature and S_i is its motor command that moves the eye (with $S_i = (\Delta x_i, \Delta y_i)$). When a predictor observes its source feature F_i^{src} in the field of view proposes its motor command S to move it to a new area where it expects to find its target feature F_i^{tgt} (see figure 15).

The number of predictors $\{P_i\}_{i \in [1,N\ ]}$ needed to cover the representation space of all possible stick animals can be very large. Moreover the execution of a predictor takes time and has a cost in term of energy (for features recognition and motor commands). So to verify the adequacy of all the predictors for a given stick animal can be a too consuming task if the total amount of resources is limited. Luckily, to consider all the predictors is not necessary to achieve good categorization performances. Indeed, as we will show in the following, the APN is able to select the most suitable predictors, also when the input is ambiguous.

This kind of predictive representation has proponents in computational (Dre91), experimental (ON01) and theoretical (Bic01) (Pez11) communities.

3.11.3 The Attractor Predictor Network

As we said previously the success of predictors influences the accumulation of evidence in favor of one category instead of another. This is implemented by active predictors that update their activation variables according to their success: that is to predict their target feature at time $t+dt$ after moving the eye from an area with their source feature at time t. To integrate such idea within the standard energetic dynamics we introduced a new variable $\overline{A_i}$ defined in equation (3.33).

$$\overline{A_i} = \begin{cases} +1 & \text{if } A_i = 1 \text{ and } F_i^{src} \in fovea(t) \text{ and } F_i^{tgt} \in fovea(t+dt) \\ -1 & \text{otherwise} \end{cases} \tag{3.33}$$

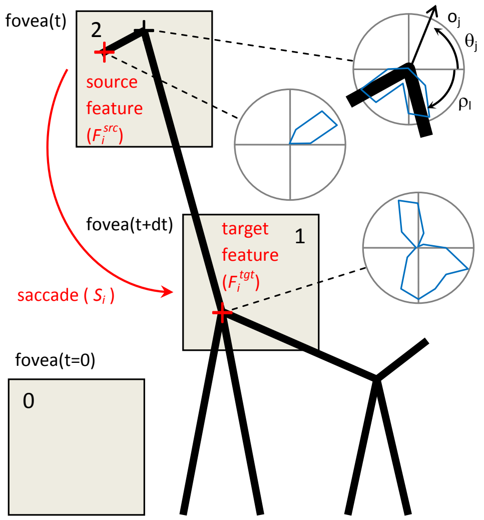


Figure 15: Here we can see a stick animal and three positions of the fovea with the corresponding features. At each fixation feature points are represented with crosses together with the orientations represented with polar diagrams. Additionally we reported the triple of predictor $(F_i^{src}, S_i, F_i^{tgt})$ between time t and $t + dt$. Figure taken from (CVQP).

Compared with the agents considered until now predictors have an active nature. Indeed their activations are not only defined by the energetic network dynamics but influence it through the feedback provided by the interactions with the environment. Substituting the variable A_i with the variable $\overline{A_i}$ in the energy update equation (equation (3.13) of section 3.3) integrates the active role of predictors in the energetic network dynamics.

$$E_i(t+1) = B_i + L_i(t+1) = B_i + \frac{1}{2}\sum_{j=0, j\neq i}^{N} J_{ij}(\overline{A_j}(t) + 1) \tag{3.34}$$

We know that predictors are specialized in relating spatially two features through a saccade. So, if we characterize a category by a set of spatial relations among features, a predictors' attractor can well identify a category. We will see that when the regularities of the input are confirmed by a set of coordinated predictors a predictors' attractor arises. But as single predictors, predictors' attractors do not constitute only a set of features, they have active capabilities too. In fact in this context predictors' attractors are used for representation, categorization and motor control.

3.11.4 Learning

In this section we will see how to store predictors' attractors within the APN. Differently from the experiments of the previous sections, here we will use the offline learning method described in section 3.6.1. In fact in this case we are able to build four predictors' attractors, one for each possible category (horse, cat, dog and giraffe), that can be used to categorize the stick figures.

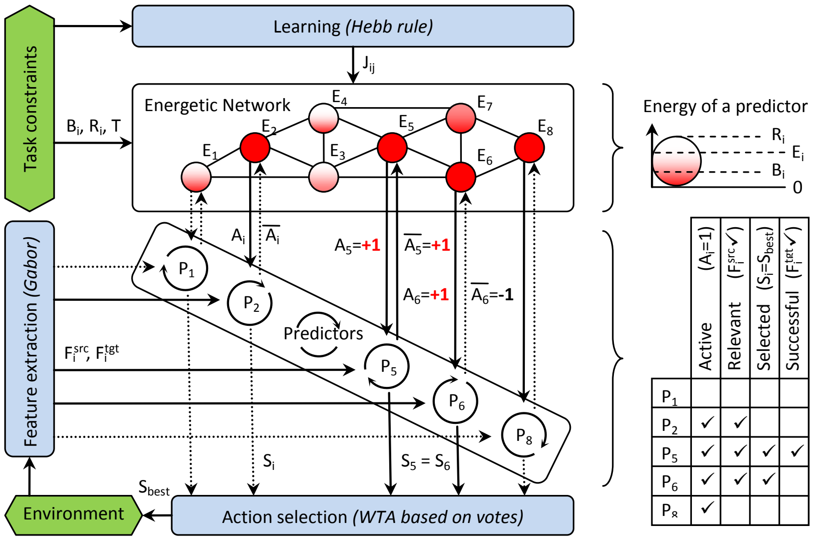
To build such predictors' attractors we started from a data set that contains several stick animals categorized by humans (see figure 14), which were originally developed by (OK04). In another experiment other authors utilized one thousand of such figures on the basis of nine parameters (e.g. torso angle, tail length), in this experiment these figures were categorized by eight human participants (SGS10b) . The prototypes considered here are averages computed over such collected responses.

Starting from the joints of each stick figure of the data set, we computed the set of all possible predictors that confirm their predictions in a given category. These are all the triples made of an existing starting feature, saccade and target feature that are confirmed for an animal that humans say to belong to that category.

Let us define four predictors' attractors $\{A^{CAT}$, A^{DOG}, $A^{GIRAFFE}$, $A^{HORSE}\}$ where the predictors that are generated by image belonging to the corresponding category are the only active predictors. Using equation (3.23) of section 3.6.1 we can store these predictors' attractors within the APN. It will be implemented in the algorithm of the following section.

3.11.5 The Categorization Algorithm

In this section we will describe a modified version of the AAN query algorithm (algorithm 1 of section 3.6.2) used here to categorize a particular



	Active (A_i=1)	Relevant (F_i^{src}✓)	Selected (S_i=S_{best})	Successful (F_i^{tgt}✓)
P_1				
P_2	✓	✓		
P_5	✓	✓	✓	✓
P_6	✓	✓	✓	
P_8	✓			

Figure 16: The Attractor Predictors Network. Once with offline learning we obtains the weights J_{ij}, we can use the energetic networks to activate predictors (*red nodes*). This happens when the energy E_i of predictor P_i is greater than its run energy R_i. When a predictor finds its source feature F_i^{src} in the field of view, and its motor command ($S_{best} = S_i$) is executed with the target feature F_i^{tgt} in its end position, a feedback is given to the energetic network through the assignment $\overline{A_i} = +1$. Note the possible states of predictors with plain arrows indicating activation, feedback and prediction success. The predictor P_5 is the winner of the election process. Figure taken from (CVQP).

stick figure through the recall of a proper predictors' attractor. Note that in this case we have also to control the eye movements within the algorithm. Algorithm 3 shows the procedure used to schedule the predictors, test their predictions and accordingly plan the saccades. After the energetic network has been updated the algorithm converges to a predictors' attractor that identify the category of the input figure. In figure 16 we have a schematic representation of the different APN components and their interactions.

Let us assume for now that all predictors have zero base energy, $B_i = 0$, and zero run energy, $R_i = 0$ (i.e., each predictor needs the same resources).

Consider at the beginning the eye positioned in a random position with the energetic network composed of few randomly chosen active predictors (N_a, see section 3.5). To belong to the basin of attraction of the agents' attractor that will arrive predictors are evaluated during each iteration in three different steps:

1. In the first step we check the presence of the source features F_i^{src} within the actual field of view. Predictors that not see a match with their sources features are set idle, the others will be considered for the next step.

2. In this step we select a new fixation according to the majority of saccades S_i that belong to the predictors considered perceptual relevant in the previous step. The most common saccade proposed by predictors, selected through a winner-take-all strategy, determines the new fixation point.

3. Once we moved the eye, according to the saccade chosen by the majority of predictors, we check if their target features F_i^{tgt} match within the field of view. If this happens these predictors will be active in the next iteration in order to be part of the basin of attraction of the future predictors' attractor.

Predictors that see these three conditions verified give the new eye position and are kept active for the following iteration in order to enrich the basin of attraction. In addition other predictors can be activated by the spreads of energy of the energetic network update, which is perturbed by a gaussian noise (i.e., the temperature) that can activate other predictors. Note that predictors generated by figures of the same category were connected by strong positive links during supervised learning, so the activation of one of them can make participate predictors of the same category. The energetic network will continue to be updated, moving the eye's fixation at each iteration, until the convergence to a predictors' attractor.

Algorithm 3 APN Categorization Algorithm

```
// Randomly position the fovea and activate $N_a$ predictors
$fovea_{pos} \leftarrow$ random position
$\{A_i\} \leftarrow$ random vector in $\{-1, +1\}^N$ where $card(A_i = 1) = N_a$
repeat
  $RP \leftarrow \emptyset$ // Relevant predictors
  for all $P_i$ do
    // Only consider active predictors with adequate context
    if $A_i = +1$ and $F_i^{src} \in fovea_{view}$ then
      $RP \leftarrow RP \cup P_i$
    end if
  end for
  // Try saccades until at least one prediction is confirmed
  $\{\overline{A_i}\} \leftarrow \{A_i\}$
  $SP \leftarrow \emptyset$ // Successful predictors
  repeat
    // Perform the saccade with the maximum number of votes
    $S_{best} \leftarrow argmax_S\{card(S_k = S | P_k \in RP))\}$
    $fovea_{pos} \leftarrow fovea_{pos} + S_{best}$
    // Test all relevant predictors
    for all $P_j \in RP$ do
      // Confirm those where action and prediction both match
      if $S_j = S_{best}$ then
        if $F_j^{tgt} \in fovea_{view}$ then
          $SP \leftarrow SP \cup P_j$
        else
          $\overline{A_j} \leftarrow -1$
        end if
        $RP \leftarrow RP \backslash P_j$ // No more selection of this predictor
      end if
    end for
    // Revert the saccade if no predictor was successful
    if $SP = \emptyset$ then
      $fovea_{pos} \leftarrow fovea_{pos} - S_{wta}$
    end if
  until $SP \neq \emptyset$ or $RP = \emptyset$
  Update the network with Montecarlo method at temperature $T$
  $\{A_i\} \leftarrow$ application of Eq. 3.12, 3.13 & 3.17
until $A \in \{A^\mu\}$ // Predictor's attractor reached
return $\mu$ // Return the associated category
```

3.11.6 Simulation 1: Categorization Perfomance

Starting from a training set of 32 categorized stick figures we used the Hebb rule (equation (3.23) of section 3.6.1) to store the four predictors' attractors. These are formed as follow: a total of 1176 predictors were extracted, each corresponding to a triple (starting feature, saccade, target feature) that matched within the figures. Note that this number is relative to the resolution used to discriminate predictors with different features and saccades. We had 319 active predictors that matched with stick figures categorized as dog in the attractor A^{DOG} , 314 that matched with figures categorized as horse in A^{HORSE}, 319 that matched with figures categorized as giraffe in $A^{GIRAFFE}$ and 224 that matched with figures categorized as cat in A^{CAT}. The number of predictors generated from figures categorized as cat are fewer because the stick cats are smaller compared with the other animals and so less predictors are needed to cover their representation space.

After the learning phase, within the test phase the APN categorized new 32 input figures of a test set using algorithm 3. Considering a temperature of $T = 0.2$ and the possible starting eye positions each stick figure was processed five times as input. At the beginning of each categorization the initial configuration is composed of 30 ($N_a = 30$) active predictors chosen randomly. At the end we have the convergence to a predictors' attractor, convergence that was accelerated using the Montecarlo method.

Comparing the results of the APN, in 160 categorizations, with the judgments of humans on the same test set of the experiments presented in (SGS10b), we got 72% (116) of correct answers. Other computational methods where tested on the same data set with the following results: the Decision Trees had a performance of 68%, Linear Discriminant Analysis 87%.

To show the evolution of the APN categorization in the following figures 17-19 we represent the dynamics of the energetic network. As similarity measure between the predictors attractors and the current energetic network configuration we use the overlap, defined in equation (3.35).

$$m^{\mu}(t) = \frac{1}{N}\sum_{i=1}^{N} A_i^{\mu} A_i(t) \qquad \mu \in \{CAT, DOG, GIRAFFE, HORSE\} \tag{3.35}$$

Where $m^{\mu}(t) \in [-1.0, 1.0] \subseteq \mathbb{R}$, with maximum value 1.0 when the predictors' attractor μ and the energetic network configuration are the same. On the contrary we have minimum value -1.0 when the only idle predictors are those active in the predictors' attractor μ.

The overlap evolution of the predictors' attractors $\{A^{CAT}$, A^{DOG}, $A^{GIRAFFE}$, $A^{HORSE}\}$ with the predictors' activities is represented in figures 17 and 18. Every two steps we reported the source and target features of the predictor that owns the last saccade executed, in order to show how decision making can be influenced by the actual field of view and its movements. This will be underlined in the following, when we will see that two different sequences of saccades made on the same animal can indeed lead to different categories, or simply differentiate the corresponding dynamics.

.

You can look at figure 17 to see that certain times the energetic network visits the neighborhood of a predictors' attractor since the early steps.

In figure 18 we can see a different case, which can be considered more ambiguous, where no unique predictors' attractor leads the competition until the end where A^{DOG} wins.

3.11.7 Simulation 2: Predictors' Influence on Saccades

In this section we will see that a categorization strictly depends on the saccades made on a stick figure. Indeed also with the same figure two different sequences of fixations can bring the system to recognize diverse categories. This can be noted comparing figure 18 with figure 19, where the categorizations were performed on the same stick animal but with different results.

This is a known phenomenon in cognitive science, as shown by the experiments performed in (KGK11). We illustrate now an additional example to investigate this issue with the APN. In figure 20 we can see an

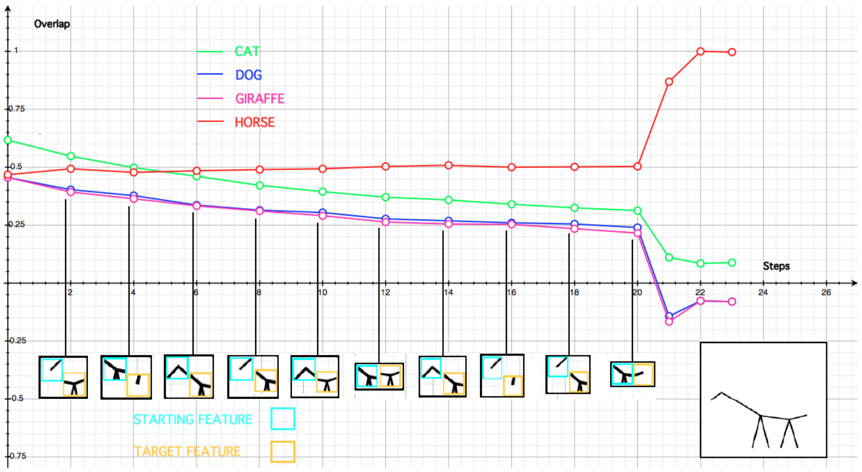


Figure 17: In this figure we can see the overlaps of the predictors' attractors giraffe in magenta, cat in green, horse in red and dog in blue. As we said source and target features of the predictor that owns the last saccade executed are represented as well in cyan and beige respectively. The convergence to the predictors' attractor A^{HORSE} (note an overlap close to 1.0) arrived after eye movements that were necessary to enrich enough the basin of attraction. As the leading attractor is the most similar with the predictors' activities during the overall categorization we can consider this input unambiguous. Figure taken from (CVQP).

ambiguous stick figure that was categorized by the APN both as a dog and as a cat. We augmented the base energies B_i of predictors extracted from figures of the same category to break this symmetry (note that until now we assumed zero base energy for all predictors). Considering the gaussian noise that we introduced with the temperature, doing this increases the probability that predictors pertaining to that category will be activated. This will confirm that according to our implementation the category of a figure is not absolute but depends on the order of visited features.

Figure 20 shows that if we increase the base energy B_i of predictors generated by figures of the category cat the system categorizes the input animal as cat. On the contrary, if we increase the base energy B_i of predictors generated by figures of the category dog the system categorizes the

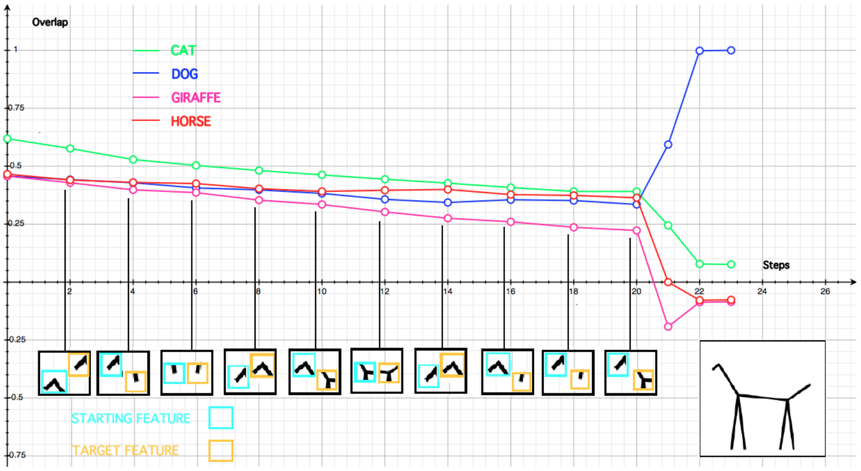


Figure 18: In this figure we can see the dynamics of a categorization with output A^{DOG}. Note the ambiguity of the input, which was already categorized by human as a cat. Here no overlap leads the dynamics clearly until the convergence. Figure taken from (CVQP).

input animal as dog. So boosted predictors can change the outcomes of the categorization.

We computed the average overlaps with the predictors' attractors dog and cat over ten simulations (each performed with different sequences of saccades). As it is possible to see in figure 20, when we set $B_i = 0 \quad \forall i \in [1, N]$ (circle data points, green for cat and blue for dog) the behavior of the system in uncertain in deciding for one category or for the other. On the other hand, if we set $\mid i \in [1, N] \ s.t. \ A_i^{DOG} = 1 \ and \ B_i = b \mid > c$, where $b = 0.23$ and $c = \frac{|saccades|}{4}$4, the system chose always for the dog category (square data points of figure 20). The same result was found for the cat category and for many other ambiguous stick figures that we decided to not report here.

[4]Note that $\mid saccades \mid$ stands for the number of fixations before convergence. Moreover the factor $\frac{1}{4}$ inside the c definition and the value 0.23 for b were hand tuned.

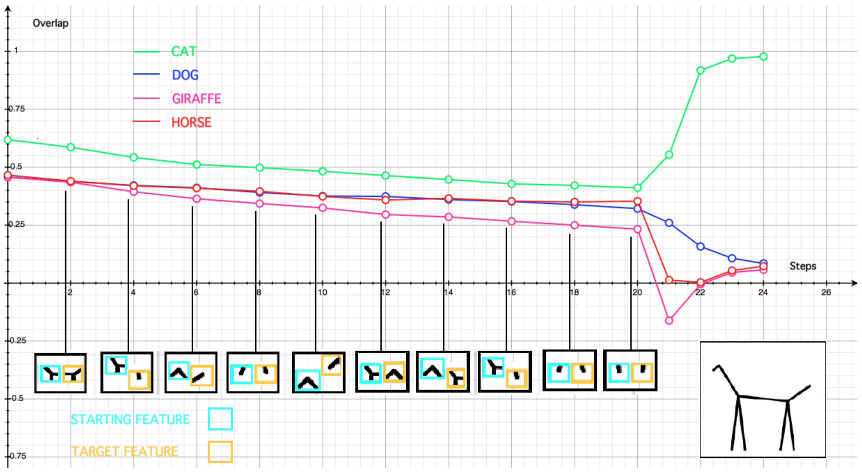


Figure 19: We can see here the overlaps' evolution when the system categorizes the same stick animal of figure 18 using different saccades. To prove that the categorization depends on the order of the visited features, we can compare the recognition of a cat chosen here with the recognition of a dog of figure 18. Figure taken from (CVQP).

3.11.8 Simulation 3: Categorization with Morphed Stimuli

In this section we want to underline the cognitive plausibility of our architecture. We start from a paper of Akrami et al. (ALTJ09) that describes an experiment concerning two primates. The authors produced nine morphed intermediate figures between each pair of images of their data set. Then the two macaques performed a 2-alternative-forced-choice delayed-match-to-sample (2AFC-DMS) task to categorize the morphed figures: the task of the macaques was to consider first a morphed figure, and then to indicate among two new figures the most similar with the previous one, where these two are the original images used to produce the first. Akrami et al. found that the number of times the macaques selected one of the original images, taken as reference category, is linearly proportional with the value of the morphing parameter used to generate the inputs (without considering the extreme values).

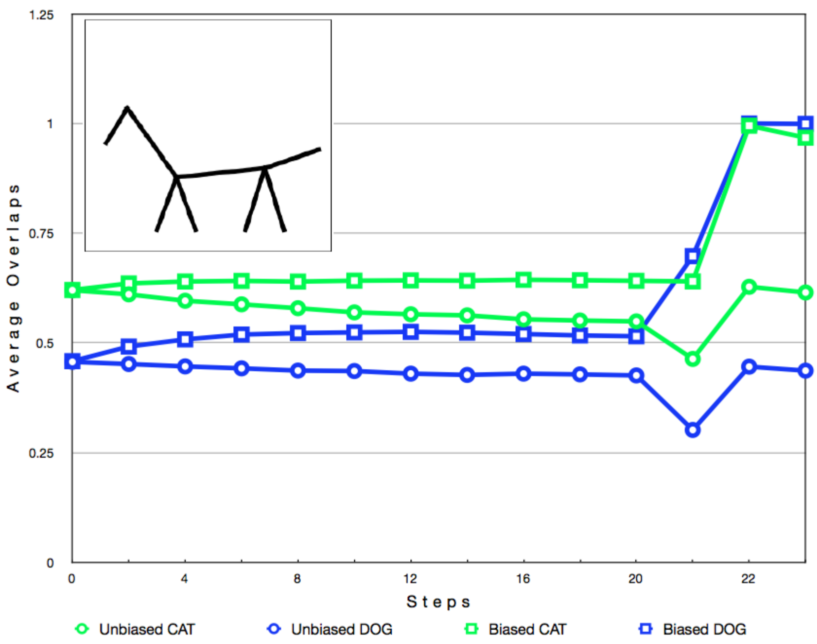


Figure 20: In this figure we have the average overlaps of ten different simulations performed over the represented animal. We can compare the behavior of the biased network when $B_i \neq 0$ for a set of predictors, with the unbiased network which has $B_i = 0$ for all predictors. In the first case boosted predictors prime the verification of contingencies typical of a certain category. Figure taken from (CVQP).

As Akrami, with our data set we generated 7 morphed stick animals for each of the 12 pairs considered. Each ambiguous figure were produced starting from two random stick animals of different categories (e.g. dog and giraffe) using a morphing function. This function has as arguments the nine parameters of the two stick animals and was used to produce 7 levels of ambiguity for each pair.

The APN was used to categorize these morphed animals to show its robustness under ambiguous condition and to see how the performance depends on the morphing parameter. Considering the two original images and their 7 variants we have a total of 9 images for each pair considered. The original images have nominal integers 0 and 8, with the latter representing the reference category. The rest of the integers from 1 to 7 are assigned to the corresponding morphed figures, respecting an identity between proximity and similarity (see figure 21). We used an APN with two stored predictors' attractors to categorize the 12 pairs of figures together with their morphed versions. For each figure we performed five simulations, for a total of 60 simulations that were used to investigate how the convergence to a category depends on the values of the morphing parameters. In figure 21 we can see the results of linear regression that confirm those found in (ALTJ09) , with a clear linear dependence between the average number of times that the APN selected the reference category and the values of the morphing parameter.

The paper of Akrami reported also the measurements, taken during the task, of the activities of neurons with an effective response to the reference category. They found, across several trials, that the average neural population response is linearly proportional with the value of the morphing parameter used to generate the inputs (without considering the extreme values).

To investigate such phenomenon we move our attention from the point of view of the categorizations performed by the whole APN, to the point of view of single predictors' activities just before the convergence to a predictors' attractor . We found that considering 60 simulations, the average number of active predictors belonging to the reference category is linearly proportional with the values of the morphing parameter (as we can see in figure 22).

The task that the APN performed for certain aspects is different from the one of the macaques. One of these aspects concerns the structure of the task. The task is our case is to select the category of a morphed figure (i.e., if it belongs to the category of figure with label 0 or the category of figure with label 8). In the case of Akrami the macaques chose between two images the most similar with a previous one morphed (a 2AFC-DMS task). Moreover the measurements of the neural activity were taken observing the response to only one reference image (the "Eff image" as they call

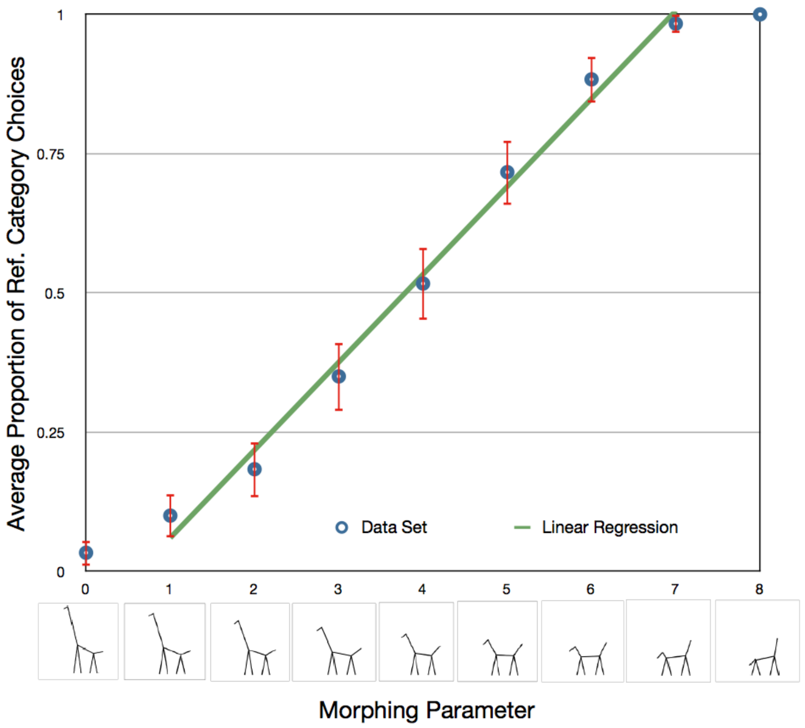


Figure 21: In this figure we can see the linear regression between the morphing parameter and number of times the APN selected the reference category (morphing parameter 8). The line within the plot has slope 0.158. Circles represent average frequencies and error bars standard deviations over the 60 simulations. We reported also an example of morphed animals with increasing morning parameter. Figure taken from (CVQP).

it in their article), making the roles played by the two original images asymmetrical. In the case of the APN both the categories of the original images were stored within the energetic network before the categorization, making the role played by the categories symmetrical.

In (ALTJ09) is considered a Hopfield network to explain the behavior and neurophysiological data of the macaques. Also the energetic network of the APN is a Hopfield-like network, but adds prediction and control of

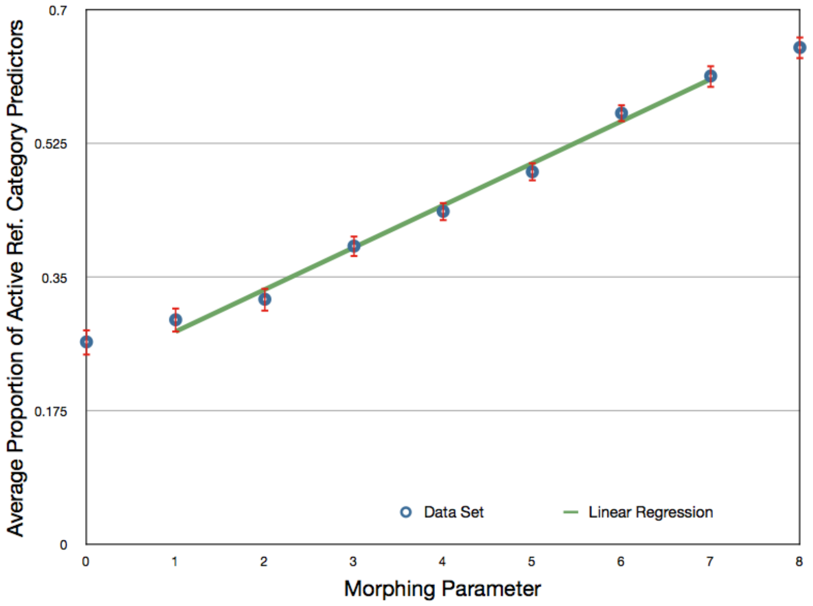


Figure 22: In this figure we can see the linear regression between the morphing parameter and the average number of active predictors belonging to the reference category (morphing parameter 8). The line within the plot has slope 0.055. Circles represent average frequencies and error bars standard deviations over the 60 simulations. Figure taken from (CVQP).

eye's movements to the typical neural networks. Moreover the anticipatory representations of predictors are used to prime the eye movements when these are active in the recognition of a certain category. Finally if on one hand the patterns memorized within the Hopfield network of Akrami are totally uncorrelated, on the other hand our predictors' attractors were produced activating the predictors that saw their strategies confirmed for the stick figures of one of the two categories.

3.11.9 Simulation 4: Probability of Categorization

In this section we will shortly characterize the probability to recognize a certain category on the basis of the saccadic sequences executed. Indeed the fixations on different features can have a diverse contribute in choosing a category. For example triples (F_s, S, F_t) that are confirmed on stick animals belonging to more than one category at the same time can be considered less informative. Because they could activate predictors of various category without promoting an unique basin of attraction. On the other hand predictors that represent triples confirmed in figures of only one category can be considered more informative, because their activation will aways promote the basin of attraction of that predictors' attractor. An example of such kind of triples is represented by the necks of the giraffes, a so long saccade, moving the eye from the torso to the head features, can be confirmed only with giraffe stick animals. On the contrary, triples codifying tails and legs of the stick animals are very similar to each other, so their activation do not help the discrimination among categories.

To better understand the role played by predictors, we investigate numerically how the probability to select a given category depends on the predictors that influenced the eye movements. Figure 23 shows that in the performed simulations the probability to choose a given category is influenced by the number of predictors of the same category that executed their saccades. Finally we found that when the proportion of these predictors is greater that one third the probability increases suddenly to values close to one, leaving this issue to further investigations.

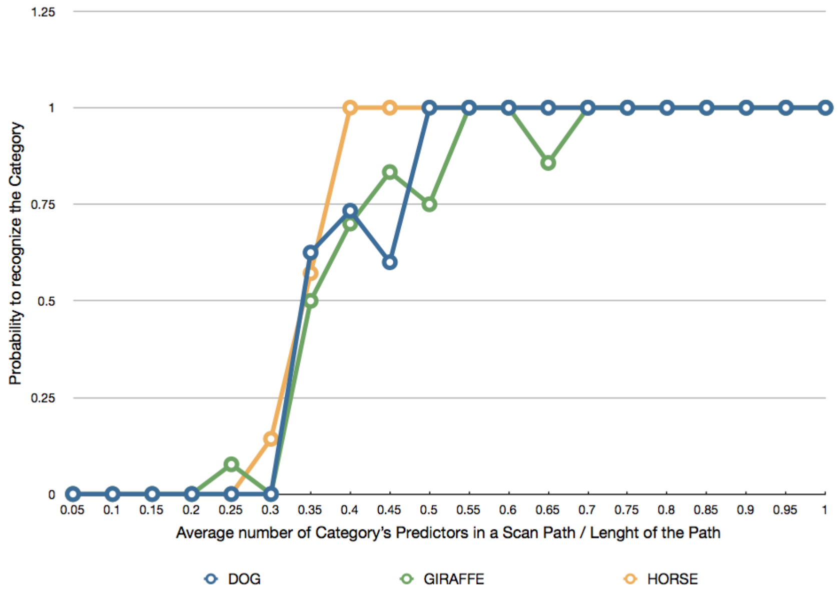


Figure 23: We can see here the probability that the APN choose a given category as a function of the proportion of predictors of the same category that contributed with their saccades. Such probability was computed using data from 160 simulations through the two following steps. First we computed the frequencies, with the number of predictors of a category that contributed in a scan path divided the length of the path. Finally, to have the probabilities, we counted the number of times that a scan path with a given frequency led to a certain category. Note that when the frequency of these predictors is greater that one third the probability increases. Figure taken from (CVQP).

Chapter 4

The Human Experiment

In this section we will describe the experiment that we performed to prove the cognitive plausibility of our proposals. Within the experiment participated 18 people with normal vision and an age going from 25 to 63 years. In the experiment we used the same stick figures described in section 3.11.4 (see figure 14), for a total of 96 stimuli. The figures were divided in two groups of 48 ambiguous stimuli and 48 unambiguous. To generate the 48 ambiguous figures we used the same procedure described in section 3.11.8. In this case we produced figures that were belonging to one category for the 25% and 75% to the other (e.g. an ambiguous dog that is also for the 25% a giraffe). In figures 24(a,b) we reported examples of unambiguous stimuli and in figure 24(c) there is an example of an ambiguous one.

The experiment is composed by several trials, each one beginning with the participant that clicks on the "start" button situated at the bottom-center of the PC screen (see figure 25 to have an idea). Once this happened two response buttons appeared on the screen together with the stick figure used as stimulus. The participant had 800 ms to identify the category of the stimulus and then clicking with the mouse on the corresponding response button. These buttons were labelled with the names of the categories involved in the trial (e.g., "horse"): we used the name of the category of the stimulus and the name of a random category in an unambiguous trial;

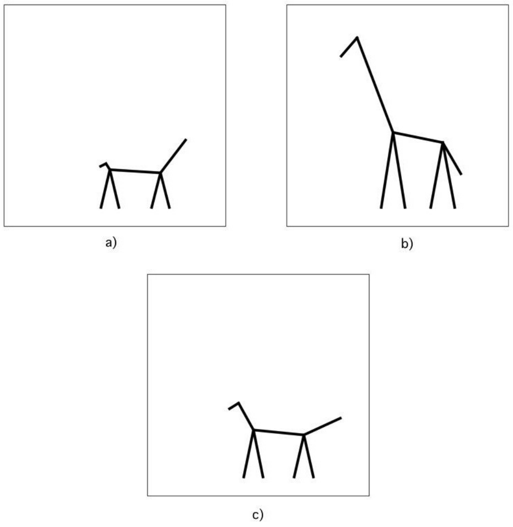


Figure 24: a) A stick cat. b) A stick giraffe. c) The cat morphed for the 25% with the giraffe. Figure taken from (QCVBP).

the names of the categories of the figures used to produce the morphed stimulus in an ambiguous trial. We considered as correct answers the category of the stimulus in the unambiguous case and the dominant category (75%) in the ambiguous one. Left or right positions for the response buttons, as the order of ambiguous and unambiguous trials, was chosen randomly. When the participant clicked on the wrong response button a red cross was visualized as feedback and when he/she took the decision too late a message ”time-out” appeared.

In each trial we recorded the mouse trajectories and the responses given by the participants utilizing the software *MouseTracker*. This is a software that can be used to record, process, and analyze mouse commands (FA10). One session of the experiment was composed by three blocks: the first one, made of 10 trials, was used to take confidence with the overall set-up; then the recording started with two blocks made of 48 trials each where the

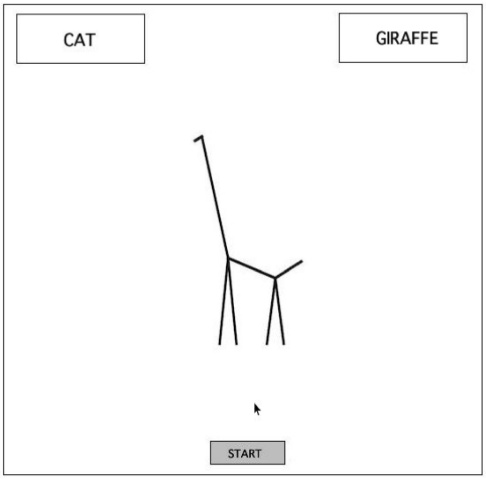


Figure 25: The set-up used within the experiment. Participants first clicked on the start button and then on the responses buttons (cat and giraffe) according to the identified category. Figure taken from (QCVBP).

96 experimental stimuli were showed. Note that stimuli were distributed among blocks randomly.

4.1 Results: Accuracy Rate and Trajectories Analysis

In this section we will describe how we analyzed the collected mouse trajectories and error rates collected in the experiment. The data were analyzed under the two ambiguous and unambiguous experimental conditions, defined according to the presence of the morphed stimuli. We found that the participants chose the wrong category in the 19% of trials and in this case data were discarded. Note that a so large number of errors can be explained considering the ambiguity of a part of the stimuli and the time constraint imposed within the trials.

To analyze how the different conditions have an influence on the response variables we used the Linear Mixed-Effects Model (LMM), used to consider and accordingly amortize the variability among subjects and stimuli (BDB08). The random-effects factors were the items and the subjects, whereas the fixed-effect factor was the ambiguity of the figures (with unambiguity taken as default condition). The analysis was done using the lm4 package for R (BM09), processing the accuracy and the trajectories independently. The p-values that we will present below were computed with Markov chain Monte Carlo simulations (BDB08).

We averaged the accuracy rate across the trials and the subjects for the two conditions separately. We found that the subjects made more mistakes to individuate the category of ambiguous figures ($mean = 0.25, standard\ deviation = 0.43$) if compared with the unambiguous ones ($mean = 0.14, standard\ deviation = 0.35$). With the mixed-effects model we were able to prove the statistical significancy of the aforementioned results, as it is possible to see with the positive contrast coefficient for the ambiguous condition ($\beta = 0.102, p_{MCMC} < 0.008$).

In addition we averaged the trajectories coordinates across all the trials and the subjects for the two conditions separately. As it is possible to see in figure 26, in the ambiguous condition we found curves more attracted toward the unselected response button (the 25% category, remember that wrong answers were discarded); whereas in the unambiguous condition we found curves that were following more the ideal straight line going to the correct response button. This difference prove that the participants' choice was still under construction during the evolution of the decision.

To measure quantitatively the spatial attraction toward the unselected alternative we used the Area Under the Curve (AUC, expressed in Mouse-Tracker units u. It is the geometrical area between the idealized straight line trajectory, going from the "start" button to the chosen response, and the participants' trajectories. Analyzing the data we found that the mean AUC was larger in the ambiguous condition ($mean = 0.92, standard\ deviation = 1.59$) than in the unambiguous one($mean = 0.74, standard\ deviation = 1.44$). As for the accuracy we used the Linear Mixed-Effects Model (LMM). The random-effects factors were the items and the subjects, whereas the

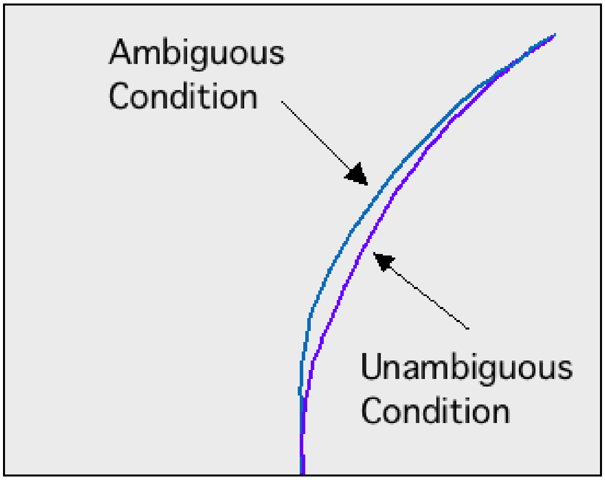


Figure 26: In this figure we can see the mean trajectories of the two conditions, represented as the correct response button was always in the right position. Figure taken from (QCVBP)

fixed-effect factor was the ambiguity of the figures (still with unambiguity taken as default condition). Finally our analysis were statistically significant also in this case ($\beta = 0.17, p_{MCMC} < 0.05$).

4.2 Discussion

In the previous section we saw that, as expected, when participants were categorizing ambiguous stimuli more trajectory curvatures and errors were encountered. To find in the ambiguous condition, compared with unambiguous one, mouse's trajectories more attracted toward the unselected response button is consistent with other categorization experiments (DKS07). Analyzing the time evolution of mouse trajectories is a good practice to instigate the dynamics of choice because we can use dynamical systems theory to investigate the competitive processes underlying the decision (Spi07). Indeed we can consider the position of the unselected response button as an attractor of the mouse trajectories and choice, which has a stronger effect when the stimulus is ambiguous. The curvature of the trajectories shows that the participants’ decision is still not finished when the action starts, on the contrary alternatives continue to compete during the course of the action influencing the outcome of its execution.

Similar phenomena were found in many studies concerning for example numerical decision, lexical decision and objects categorization (FDF11) (BP12) (SN09). In addition dynamical models were proposed to represent the competition of alternatives in perceptual decision making (SDKG10), (AMT98), (McC01).

Such dynamical interpretation of decision making is at the core of our computational and theoretical proposals. Indeed, as we saw in section 3.11 and we will see in the next section, our computational models can solve the aforementioned task integrating dynamic competition to fully capture the nature of perceptual ambiguities.

Chapter 5

Second Proposal: Neural Field-based architecture

The embodied vision of cognition underlines the close relation between cognition, the sensory and motor surfaces and the environment in which these are immersed (Bar08). Dynamical systems theory can be the right theoretical framework to formalize such embodied view of cognition. Indeed models of embodied cognition should be process models that can address the unfolding in time of cognition and the associated sensory and motor processes. Within the context of dynamical systems we can see behavior as something that emerges from underlying forces represented as vector-fields. Behavior can emerge as a stable state (i.e., attractor) from the neural network linking the sensory and motor surfaces, which together with the environment establish a dynamical system.

Neural field theory takles these issues starting from the following basic principles (Sch08):

1. Patterns of behavior are characterized by inner states, which determine the persistence over time and under changing conditions.

2. The evolution in time of these state variables is generated by neural networks linked to sensory and motor surfaces that can be modeled as dynamical systems.

3. Asymptotically stable states structure the solution of this dynamical system. Over the long run, only attractor solutions are robust and likely to be observed.

4. Only when states are released from stability behavioral flexibility arises. Release from stability takes the form of instabilities (bifurcations) in which the restoring forces around an attractor become too weak to resist change. New solutions may be reached or even created from instabilities.

In addition neural fields defines activation fields using two dimensions to represent metric information in terms of dynamical state variables. One is the metric dimension along which information is specified. To each position within the field corresponds a specific value in the metric dimension. The second dimension is an activation level defined for each such value of the metric dimension, which encodes a measure of the amount of information about that value. For example, about sensory representation, high levels of activation indicate field locations which contribute more to the actual estimate of sensory information. Whereas low levels of activation in a specific field position mark that the value of the represented dimension, related with that location, is not a probable estimate. About motor representations high levels of activation indicate that the movement represented at that location is close to being initiated and activation from that field site will be handed down to the motor control system. Whereas low levels of activation at that field position mark that the related movement is not likely to occur.

In neural field theory stable and localized peaks of activations (fixed point attractors) are units of representation. The activation level of these peaks is the strength of the representation. On the contrary a flat distribution of activation represent the absence of specific information. The activation level of a specific field position represent the system's state variable. Hence neural fields are infinite dimensional dynamical systems with activation levels that evolve continuously in time.

The corresponding field dynamics is usually defined starting from an equation similar to equation (5.1) (Sch08).

$$\tau \dot{u}(x,t) = -u(x,t) + resting\ level + input + interaction \tag{5.1}$$

Where $u(x,t)$ is the activation field defined on the metric dimension x and time t. The first three terms of the equation's left side represent the input driven regime, where attractor solutions follow the relation $u(x,t) = resting\ level + input$. The interaction stabilizes localized peaks of activation to contrast diffusion through global inhibitory interaction and to contrast decay with local excitatory interaction. Finally coefficient τ is used as rate of relaxation.

A first mathematical implementation of such ideas is the well known Amari dynamical law reported in equation (5.2) (Ama77).

$$\tau \dot{u}(x,t) = -u(x,t) + h + S(x,t) + \int dx' w(x-x')\sigma(u(x',t)) \tag{5.2}$$

Where h is a negative constant representing the resting level; $S(x,t)$ is the input function defined on time and space; $w(\delta x)$ is the interaction kernel; and $\sigma(u)$ is a nonlinear sigmoidal threshold function. The interaction term combines input from all the field positions x' where the activation is enough large. About that the interaction kernel checks if inputs from those positions are positive or negative and accordingly rises up (excitation) or down (inhibition) the activation respectively. Noe that generally the stability of localized peaks of activation is guaranteed by inhibitory inputs from all field locations and excitatory inputs from only the nearby ones.

These localized peaks of activation, sometimes called "bubbles", have strong attentional and competitive properties that are achieved though a combination of global inhibition and local excitation. Neural fields were used to have systems able to converge toward coherent stimuli in input flow (Tay99) and tracks them also in presence noise and distractors (RV06). But although these properties the classical Amari formulation remains purely reactive and strongly dependent on the input signal.

In (QGL11) (QG$^+$12) Quinton et al. introduced a new neuro-inspired model based on neural fields that extends the classical formulation with a combination of dynamic competition and normative prediction. This model was applied to solve spatiotemporal tracking where prediction was used to bias the dynamics of the field so as to constraint the selection and tracking of a target object. Indeed when applied to tracking, neural fields are typically not able to discriminate between the target and a distractor on its trajectory. On the contrary using predictive neural fields we can integrate the prediction of the trajectories within the dynamics and solve such ambiguity. Then dynamic competition was used to filter the number of goal-oriented anticipations that may be very large.

In this section we will introduce the second computational model of this thesis that is grounded on predictive neural fields. We will use our proposal to categorize the same stick figures of the computational and human experiments of sections 3.11 and 4 respectively, still through the competition of sensorimotor feature predictors (see section 3.11.2 to have a definition of predictors and section 3.11.4 to see how they were generated). A schematic view of the overall architecture is shown in figure 27. Recalling the human experiment, we added the control of the mouse movements, which was implemented following the same set-up of the Mouse Tracker software used in the previous human experiment (apart the fact that prototypical stick animals substituted words within the response buttons). This was done to compare quantitatively the results of humans with the results of the computational model on the same task.

We used the same preprocessing and features-based representation of the visual input that we used in section 3.11.1. Whereas about the output commands the architecture controls the eye movements and additionally controls the mouse trajectories. The position of the eye within the figure is represented in cartesian coordinates and its movement is immediate. As we will see to move the eye the system combines excitation, inhibition and proprioception through prediction. The mouse movements are represented in cartesian coordinates within the screen frame of reference (note that the eye and mouse are not embed in the same cartesian system). At each step the architecture generates a vector v that represents the mouse velocity on the basis of predictions' confirmations and these velocities are integrated

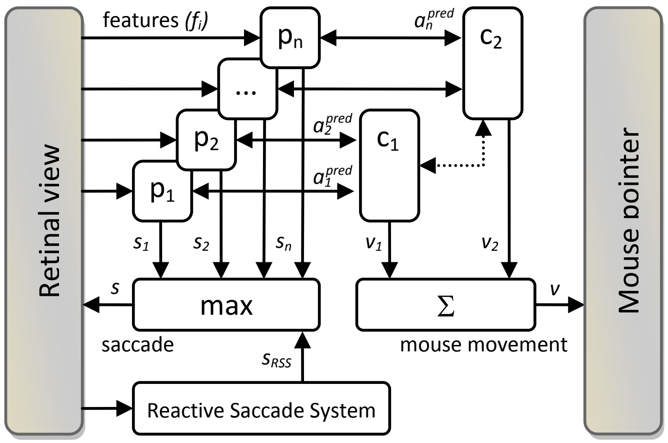


Figure 27: Schematic view of the architecture. Where plain arrows represent excitatory links and dashed arrows represent inhibition among different categories. Note that if on one hand eye movements are controlled by the leading predictor or the reactive system, on the other hand the mouse movements are computed weighting multiple vectors pointing to the targets at the same time. Figure taken from (QCVBP).

using the Euler rule producing smooth trajectories.

5.1 The Computational Model

The system is composed of a set of the same predictors (P_i), already defined is section 3.11.2 as triples $(F_i^{src}, S_i, F_i^{tgt})$ of source features, saccades and target features. As we know to each predictor is also assigned a category $c_{\{1,2\}}$, that is the category of the stick animal used to generate it. In the current proposal this information is used to move the mouse in the direction of the button associated to that category.

Moreover predictors here have a set of specific activities used to control and ameliorate the behavior of the system as we will describe below. During the execution of the task these activities are updated as shown in equation (5.3) according to the outcome of the interactions.

$$\begin{array}{rl}
a_i = & a_i^{reac} - \gamma_2 * a_i^{inhib} + \gamma_1 * c_{\{1,2\}} \\
a_i^{reac} = & \max_j\{\sigma(F_j, F_i^{src})\} \\
a_i^{pred} = & a_i^{prop} \times (\max_i\{\sigma(F_j, \overline{F_i^{tgt}})\} - \beta) \\
a_i^{prop} = & a_i^{reac} * \sigma(s_i, \overline{s}) \\
a_i^{inhib} = & \max((1-\alpha)a_i^{inhib}, a_i^{prop})
\end{array} \tag{5.3}$$

Where a_i^{reac} represents the actual perceptual relevance of the predictor, used to consider actions of predictors that have their target features within the current field of view. Then a_i^{pred} is used to verify if the prediction of a predictor become confirmed, that is if its target feature appears within the field of view after the execution of its action (this is similar with the variable $\overline{A_i}$ that we introduced in section 3.11.3). Another activity variable is a_i^{prop} that detects if the last action executed is equal with the action of the predictor. The activity a_i^{inhib} is used to inhibit the return to the last position. These two last activities were useful to avoid loopy behaviors though proprioceptive feedback. All these activities are aggregated within the equation of the variable a_i, which represents the possibility that the system will choose to execute the action of predictor i, where γ_1 is a coefficient used to tune the contribute of categories that already received consensus from other predictors.

We assumed that for the artificial system prototypical stick animals can stand for categories, as in the human experiments we assumed that the words could play the same role. For the same reason we thought that was possible to assign to a predictor a category when it was excreted from a stick figure prototypical for that category. The prototypes were constructed averaging the stick lengths and joints angles across stick figures of the same category, following the same principles of prototype theories of categorization and conceptual spaces (RMG$^+$76) (Gar00).

5.2 The Categorization Algorithm

In this section we report algorithm 4 that illustrates the procedure used to control the eye, move the mouse and perform the categorization.

In this system we also considered an external reactive saccade system

Algorithm 4 Neural Fields Algorithm

```
for all predictors ($P_i$) do
    update $a_i$, $a_i^{reac}$, $a_i^{pred}$, $a_i^{prop}$, $a_i^{inhib}$    (equation (5.3))
end for
choose the predictor with max $a_i$
move the eye to a new position $(x, y)$    (equation (5.4))
for all categories ($c_j$) do
    update $c_j$ with the best associated $a_i^{pred}$    (equation (5.5))
end for
let categories ($c_j$) compete for control
compute the mouse control    (equation (5.6))
```

that stops the eye from moving too far from the stick animal within the figure. Indeed the eye can arrive to empty areas where no features are detected if the input is unknown or its category is still under investigation. As in the Brook's subsumption architecture (Bro86), the reactive saccade system can take the control of the eye if this happens inhibiting the contribution of the predictors. To enable this system we implemented equation (5.4), where a_{RSS} is the constant activity of the reactive saccade system and S_{RSS} is its saccade.

$$S = \begin{cases} S_{argmax_i\{a_i\}} & \text{if } max_i\{a_i\} > a_{RSS} \\ S_{RSS} & \text{otherwise} \end{cases} \tag{5.4}$$

The activities (c_j) represent the current evidence in favor of the category j and, as we reported in equation (5.5), they are updated incorporating the predictors activities (a_i^{pred}). This information is used to move the mouse pointer towards the category response buttons.

$$\begin{aligned} \overline{c_i} = & \quad \lambda c_i + \max_{k \in Q_i}\{a_k^{pred}\} \\ c_i = & \quad \frac{\overline{c_i}}{\sum_i \overline{c_i}} \end{aligned} \tag{5.5}$$

In equation (5.5) Q_i represents the set of predictors that belong to the category c_i, whereas λ is an inertia coefficient used to amortize the effects of immediate change of context, with the effect of guarantee a certain level of smoothness of the generated mouse trajectories. In the equation the variable $\overline{c_i}$ is normalized to implement a limited amount of activity within the variable $c_i \in [0, 1]$ (see section 1.1 for related bounded approaches). This introduces competition between the two categories showing the capacity of neural fields model to implement robustness and attention. Similar neuro-inspired architectures with predictive representations was already introduced in (QGL11) (QG$^+$12), others were applied to categorization and sensory signals (JSS08).

To obtain the velocities of the mouse we linearly combined the vectors $v_{\{1,2\}}$, always pointing to the two response buttons, with the corresponding categories activities $c_{\{1,2\}}$ as is represented in equation (5.6). This will make the pointer arrives to the buttons according to the current evidence on the categories as we can see from figure 28.

$$v = \sum_i c_i \times v_i \tag{5.6}$$

5.3 Results

5.3.1 Simulation 1: Performance Evaulation

In this section we will describe the simulations performed to compare our artificial system with the results of the human experiment. We designed eighteen artificial participants that performed 96 simulated trials. The resultant trajectories were analyzed using the same measures of the human experiment. Note that synthetic trajectories were more equal with each other compared with human trajectories and their variability mainly depends on the initial eye position that strongly contributes in the bifurcation dynamics. The output of the analysis was the following: in the unambiguous condition we found $mean = 0.46$ AUC and $standard\ deviation = 0.27$ whereas in the ambiguous condition we found $mean = 0.30$ AUC and $standard\ deviation = 0.10$ (note that AUCs is expressed in MouseTracker units u). As in the human experiment AUC is larger for the ambiguous condition, a relation that is statistically significant as was possible to see from the results of the paired T-test on the two conditions ($p_{T-test} < 0.0001$). Additional analysis were performed about

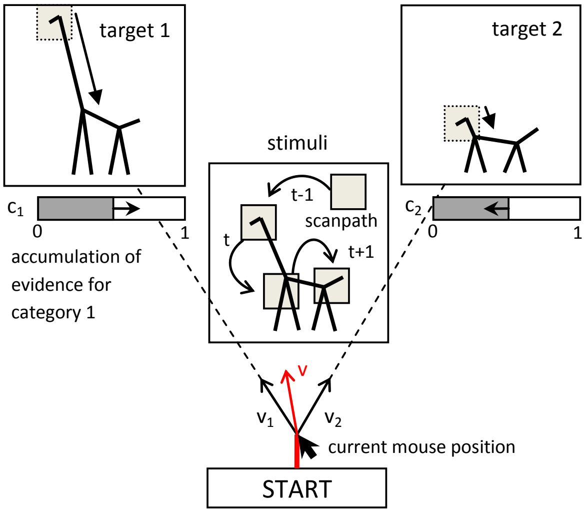


Figure 28: In this figure we can see a sequence of fixations on the input stimulus that starts from not informative features to informative ones. Indeed the first saccade does not contribute to differentiate the two prototypes. On the contrary when at time t a new saccade is performed, from the head to the torso of the stick animal, considering the different lengths of the necks of the prototypes, the system starts to favor the left one. This new evidence is collected comparing a target feature with the current field of view, it effects the mouse movements as it is possible to see from the resultant velocity vector (red arrow). Figure taken from (QCVBP).

the reaction times (RT) of the simulated participants with the following results expressed in seconds: in the unambiguous condition we found $mean = 0.93$ and $standard\ deviation = 0.027$ whereas in the ambiguous condition we found $mean = 0.960$ and $standard\ deviation = 0.08$.

As we have done in section 3.11.8 for the attractor predictor network also in this case we investigated the behavior of the artificial system varying the value of the morphing parameter μ. As it is possible to see from figure 29 the AUC, that can be considered a measure of the response accuracy, does not linearly depend on the morphing coefficient. Additionally we found a symmetry around middle values of the morphing parameter

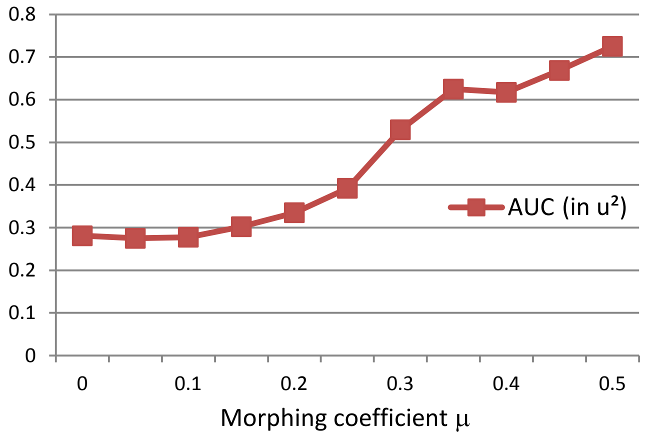


Figure 29: In this plot we can see the trials and participants' mean of the Area Under Curve (AUC) as depended variable and the morphing parameter μ as independent variable. Note how the function is more similar with a power law than with a straight line, indicating a non linear dependence on the morphing parameter, as we investigated in section 3.11.8. It is however clear how much the difficulty to categorize a figure increase with its ambiguity. Note that AUCs is expressed in MouseTracker units u. Figure taken from (QCVBP).

with the system behaving similarly for large and small values.

The results that we reported can be explained if we consider the system's non linear dynamics and the late changes in decisions that it is possible to see from the trajectories that we represented in figure 30. The reasons of this behavior can be found in the inertia of predictors and in the evidence for both categories that is collected when the stimulus is ambiguous enough.

The system dynamics is non linear because of the feedback that its interactions with the environment produce. This feedback is related with the current field of view and with the possible confirmations of predictors' hypothesis. Predictors interact with each other through category links weighted by the $\gamma_1 * c_{\{1,2\}}$ term in equation (5.3). Moreover the action of a predictor, if executed, can favor the suitable context to make other

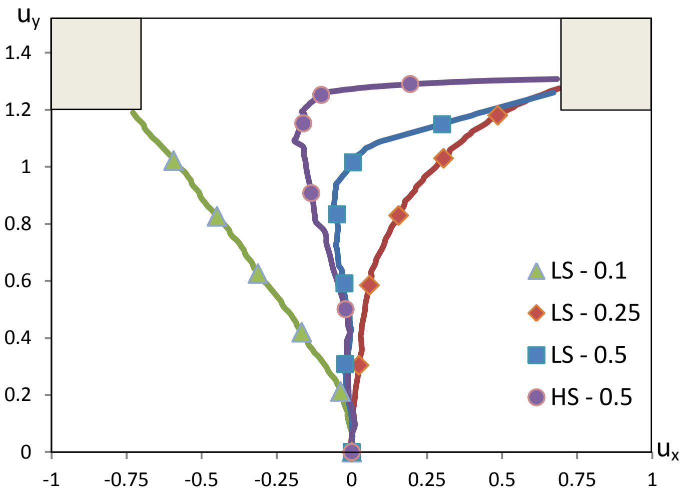


Figure 30: Trajectories produced by the agent under several conditions. At low speed (*LS*), 3 representative trajectories are provided for a morphing factor in $\{0.1, 0.25, 0.5\}$, with increasing deviation from the straight trajectory. For high speed (*HS*) and high ambiguity (morphing coefficient of 0.5), the late change in decision during the reaching movement is amplified compared to the *LS - 0.5* condition. Figure taken from (QCVBP).

predictors of the same category to operate with their commands. So the role of the prediction activity a_k^{pred} is very important, because it contributes to select saccades of the category that is collecting more evidence favoring rapid bifurcations.

In other words predictors that are supporting the same category enhance each other and contribute at the same time in moving the pointer toward their response button. Moreover predictors compete for controlling the eye. At the category level, they discriminate among figures; individually, they discriminate among different parts of a stick animal. About their source features ($F_1^{src} \neq F_2^{src}$) two predictors can be used to differentiate two different animals or two different part of an animal (e.g., head and leg). In addition if predictors differ from their saccadic commands ($S_1 \neq S_2$) only one of them will have the possibility to check its prediction;

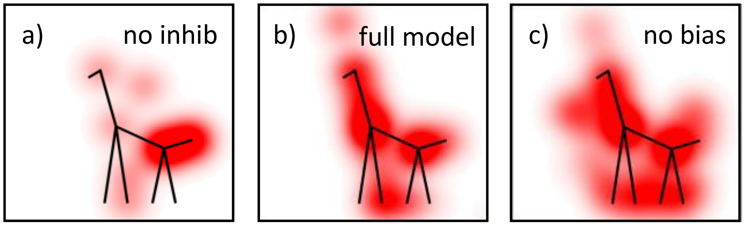


Figure 31: Heatmaps at the end of a trial, in three cases: *a)* with no inhibition of return, *b)* complete model, *c)* with no top-down modulation. *a)* Without the informativeness mechanism, the architecture has higher probability to converge on limit cycles of saccades where no clear decision can be made. *c)* When saccades are reactively selected, the system focus less on relevant features. Figure taken from (QCVBP).

this situation can be useful to discriminate within animals of different dimensions with very similar features ($F_1^{src} \simeq F_2^{src}$ and $F_1^{tgt} \simeq F_2^{tgt}$). If the two predictors have equal starting features and saccades the execution of the command is needed to identify if a target feature corresponds with the feature within the field of view, revealing informations about figures and categories.

5.3.2 Simulation 2: No Prediction

Now we want to underline the role of prediction comparing our architecture with a reactive controller that does not implement predictors. Within the reactive controller to compute a_k^{reac} features are matched one by one with the prototypes, without the verification of target features and the usage of a_k^{pred}.

As with the human experiment and the computational one, the reactive controller generates differences about the AUC values, which are: $mean = 2.13, standard\ deviation = 0.08$ for the ambiguous condition and $mean = 1.99, standard\ deviation = 0.10$ for the unambiguous condition (with $p_{T-test} < 0.0001$). Additionally for the RT the results were: $mean = 9.39$, $standard\ deviation = 5.03$ for the ambiguous condition and $mean = 5.19, standard\ deviation = 2.05$ for the unambiguous condition (with

$p_{T-test} < 0.001$). However the absence of prediction leads to a significant increase in the mean values of the AUC and RT, reflecting the difficulty and longer time required to reach a decision.

Although a large RT standard deviation, note that we already discarded a set of outliers from the simulation data to perform a meaningful analysis (when RT was larger than its mean for three times its standard deviation). These trials were useful to understand what happen when there was no clear discrimination: we observed that a system with no prediction can only compare features of different figures without exploiting their spatial relations. Moreover the features taken singularly can be very similar across different animals so, to discriminate them, we should use saccades too. Indeed predictors use saccades' amplitude to encode the stick lengths and this information is lost in this implementation without predictors. These observations were verified with a simulation having two categories' prototypes with the same features but different for the length of a single stick. In this case, where we still used the inhibition of return mechanism, the system without prediction was not able to discriminate the two stick figures.

5.3.3 Simulation 3: No Top-down bias

Within the architecture we can find a combination of bottom-up and top-down processes that make the system choose and move. The bottom-up process is identified by the predictors and their tests whereas the top-down influence is given by the category activities $c_{\{1,2\}}$. To underline the role of these activities we considered an experiment with $\gamma_1 = 0$ and so with no top-down feedback.

In this case we found a clear loss of performance that can be seen comparing figure 31b and figure 31c, where we represented heat maps showing the density of fixations on the figure. In fact to boost predictors belonging to the category with more evidence diminishes the number of fixations out of the stick figure and so increases the informativeness of saccades and consequently the categorization's efficiency.

5.3.4 Simulation 4: No Informativeness

The utility and efficiency of the inhibition of return can also understood trying to remove it. This can be done in the model by annulling γ_2, in order to delete the effect of a_k^{inhib} on a_k^{reac}.

The effects can be seen on the heatmaps comparing figure 31a and figure 31b. With no inhibition of return, the system converges on limit cycles of saccades from which it is difficult to escape by only using the top-down modulation. In fact, predictors prime each other using the environment and if they belong to the same prototype activate each other; without inhibition of return, this mutual excitation cannot be counterbalanced. Using the informativeness of the fixation points within the limit cycle attractor we are not sure that the system will be able to take a decision.

Chapter 6

Discussion and Comparisons

In this section we will compare our proposal with others within the artificial intelligence community. Moreover we will compare the AAN architecture with the neural field-based architecture.

One first common property that can be found about the two architectures is that both are based on a self-organizing approach (Kau93) (Min88). According to it the functional adequacy of the system is found in its collective behavior and it is designed among individual agents. The desired function emerges from the local interactions of agents without any centralized control. Each agent follows its own task and at the same time adapts its local interactions cooperating with other agent, in order to fulfill the general purpose of the system. Moreover the two architectures have in common an approach that combines dynamical systems and neuro-inspired models. If on one hand the simplicity and well defined formulation of Hopfield networks makes their dynamical properties usable for learning and memory representation, on the other hand the larger expressivity of neural fields made available more complex behaviors (informativeness, inhibition of return and so on). In addition if the two systems have in common the same kind of interaction feedback based on predictors' outcomes, the systems' bounded rationality had different implementation among the proposals. In fact the limit of computational resources is implement with the normalization of the category activities in the neural filed architecture, whereas in the AAN constraining the energetic network dynamics over a limited average number of agents.

In section 1.1 we represented a list of key questions to answer with respect to meta-reasoning that Zilberstein reported in (Zil08), where the author also states that metareasoning is the right framework to deal with bounded rationality. These questions can constitute a good starting point to see the issues addressed by our proposals. The object-level architecture in the AAN is constituted by the agents and is fully domain independent as we showed in three different applications where agents were problem instances, experts or predictors (automatic provers will be considered too in section 7.1), fulfilling those requirements of dealing only with the structural information of object-level deliberation (RW91) (GG91). Although the neural fields based architecture was implemented with only predictors in this thesis, Quinton et al. already presented similar proposals concerning tracking and pattern recognition (QGL11) (QG$^+$12), so the same line of domain independence could be easily achieved also in this case.

The meta-level architectures were constituted by a Hopfield network and dynamical neural fields in the two cases, used to control the allocation of the computational resources within modules of the object-level. These allocation policies are computed online or precomputed using the AAN offline learning algorithm. The correctness of the object-level components is monitored run time by the perceptual relevance and prediction success in the active vision task. In other tasks the correctness can be still monitored run time by the AAN architecture's adequacy step, which outputs the agents' success or failure. Moreover the prior knowledge about the object-level component's efficiency can be defined through the assignation of the run energy variable R_i. Within our architectures there is no switches between the object-level and the meta-level because the latter always interact with the first with a computational complexity that can be estimated as $O(N^2)$ at each step in the case of Hopfield networks connecting N agents.

About the list of important challenges in the context of MAS of section 1.3, which Sycara reported in (Syc98), the AAN architecture addresses at least those labeled as 2, 3 and 5. Indeed the resource allocation implemented by the energetic network was used to coordinate agent control. Morevoer the reconciliation of conflicting goals between the agents are solved by the adequacy tests and within the convergence of an agents' attractor where agents act in coherent manner.

From a real time systems point of view the general AAN aim is to schedule several processes (or agents) on several processors. Where we can choose the number of allowed processors with the bias parameter a introduced in section 3.5. Indeed, distributing energy between agents in different span of time, the AAN architecture is effectively implementing the scheduling of the processes. We know that this energy distribution is implemented by the energetic network, which in the case of the AAN architecture is a Hopfield network. As a matter of fact Hopfield networks have already been used to solve real-time scheduling problems (SCM97) (Mar99). But if in the case of classical scheduling problems the deadlines and release times of the tasks to be scheduled are known, whereas with the AAN architecture we solve an implicit and adaptive kind of scheduling. Indeed with the AAN we want to schedule tasks with unknown deadlines and release times, this because in our architecture they can be implicitly defined by the success or failure of the processes, which are determined online.

An adaptive methodology based on machine learning can be useful when the processes that we have to schedule are so complex that we are not able to establish their timing constraints. Or when these timing constraints are changing in so complicated manner that is not feasible to schedule them with a classical approach (due to a dynamic, noisy and not predictable environment or due to other agents behavior). Note that also if the time constraints are not known, the AAN can learn to assign the right agents to available processors in the right time, making its behavior very similar with the behavior of a scheduler. Moreover it is not difficult to extract the explicit timing constraints introduced by agents' successes and failures: when an agent has succeeded it should be active, when an agent has failed it should be not active. Also if we already proved that the AAN architecture is able to make active successful agents, it can be useful to leave for further investigation a performance evaluation with the same kind of analysis used in real time systems: as runtime overhead, schedulability analysis, robustness during transient and permanent overloads, resource sharing and aperiodic task handling.

MultiAgent Resource Allocation (MARA) is the study of how to distribute a number of resources amongst a number of agents (CDE$^+$05) (Cea06). Task allocation can be considered as a MARA problem where task are resources with negative utility (cost). To compare our energy to other kind of resources we can say that the energy is a continuous indivisible resource which determines the number of active agents, it is sharable and not static. What makes the AAN model different from the other MARA approaches is its preference representation. Indeed the AAN's preference structure is computed by the agents locally with their success/failure tests. Moreover externalities about others' resources are coded in the weights' matrix of the energetic network with an associative representation (rather than only cardinal or ordinal).

Among the several multiple models-based architectures the AKIRA framework (PC07) is the most related with the AAN architecture (see section 2.1). The main differences with the approach presented here are related with the language based on dynamical systems that we introduced using a Hopfield network as energetic network (e.g. agents' attractors). Moreover the energetic network considered here has a different topology, because the AAN's energetic network is a recurrent full graph that implements also links with negative weights (i.e., local inhibition). Additionally in AKIRA only failing agents spread their energy through the energetic network, where in the AAN architecture this happens also for the successful ones. Note also the different implementation of the resources bound, which in the AAN architecture is obtained constraining the energetic network dynamics and in AKIRA through an external tank that contains the not utilized resources (the energy pool). Finally we showed that the AAN dynamics can be also perturbed without effecting its capacity to retrieve agents' attractors, making the architecture robust to noise.

Chapter 7

Open Issues and Future Works

7.1 Future AAN Application, Distributed Automatic Reasoning: The Attractor Predictor Network

An implementation of AAN architecture in automatic reasoning has not been tested yet. We could call it attractor provers' network. According to it each agent of the AAN should be an automatic theorem prover so that the reasoning process can be distributed among several agents.

In automatic reasoning a theorem prover tries the soundness of a logic formula if it can be inferred from another set of true logic formulae. We call this set Knowledge Base (KB). Typically these problems are computationally hard, indeed a classical prover is not efficient if the set of formulae is too large. Although the complexity of this problem, to query efficiently from very large ontologies is still an important challenge today. For example it is necessary in semantic web technologies.

Usually classical provers are not able to use contextual and semantic informations, which are necessary to prove efficiently. This means they are not able to select automatically the most suitable set of formulae for the current context. On the contrary if we apply AAN to automatic reasoning we could store this kind of informations in the agents attractors of the energetic network and therefore obtain a robust and efficient prover. In fact the AAN could be used to select only a subset of provers contextually relevant reducing the overall proof's computational cost.

The idea to use an associative memory to retrieve a set of semantically related formulae (known also as *Knowledge Pattern* (CTP03)) is something different from search or data base strategies, because it deals with analogical reasoning. The AAN could be considered an abstract data type to store knowledge patterns, to retrieve them and to achieve an efficient bottom up recognition of the knowledge pattern of a particular formula.

To apply the AAN architecture to distributed automatic reasoning we have to consider all the agents as classical theorem provers, which are different each other because each of them can access only to a small subset of the KB. Therefore, if only few agents will be active, the proof will not search on the entire KB but on a its significative subset. So, to make the proof successful, only the agents that contain the necessary formulae to prove the input formula should be activated.

The idea to select an appropriate subset of formulae from the KB was already proposed in automatic reasoning (WCR65), where this set is usually called *set of support*. What we propose here is a dynamical version of the set of support, which can adapt online according to the current context. Indeed to a subset of active agents corresponds a subset of active logic formulae, and as we have seen agents' activities can adapt according to the context.

One of the first problems to solve here are: to decide how to distribute the KB across the agents; to decide how to define the topology, which determines the provers' attractors.

About the first issue, a possible solution is to make each prover have access only to formulae that involve the same concept (e.g. the same logical predicate or constant) and connect provers that are semantically related. So in this case the energetic network will be both a weighted semantic network, connecting constant and predicates, and an activation network with its own dynamics. The arrival to a provers' attractor can represent the activation of a particular semantic field, which is a good representation of the context that is necessary to not search on the entire KB.

About the second issue, following the introduced AAN methodology, provers' attractors can be inserted manually or learned automatically. On one hand, following the first approach, we can use a knowledge engineering approach, that is using our domain expertise to select which formulae should belong to the same provers' attractor. Once these are defined we can use the Hebb rule (equation (3.22) of section 3.6.1) to memorize these provers' attractors within the energetic network.

On the other hand we can let the AAN architecture to find automatically the right set of provers' attractors using the introduced online learning algorithm (algorithm 2 of section 3.6.2). Using this method provers are evaluated online according their ability to infer fruitful formulae with the success and failure tests. These tests can be based upon the same cost function used in an A* search algorithm, which could be implemented in each prover to solve its local proof: in other words a prover has success if it infers a formula with low A* cost (given by the cost to arrive to the current proof tree's node and by the inferred formula's size); otherwise it fails. In this way energy will be transferred to agents that are proving well, to let them to take the control. Then with the resource bound of the AAN architecture only few provers can be active at the same time, so the computational cost could be always limited.

Another alternative approach that could be used to learn the provers' attractors is related with the *Latent Semantic Analysis* (LSA) method (DDF$^+$90). Typically this method is known for its applications in natural language processing, but we could use similar algorithms to execute on ontologies instead of texts. According to it we can start from known ontologies, or completed proof, to compute their occurrences of each logical predicate or logical constant. The output of this process is a vectorial space, called concept space, where each logical predicate or logical constant have a coordinate according to its semantic meaning found from the occurrences analysis. In other words more two predicates or constants are semantically related more they will be close in the concept space.

The main idea of this approach is to embed the concept space within the energetic network. This can be done converting the distance between different constants and predicates into weight of energetic links among agents which are dealing with the same constants and predicates. In other words a link between two predicates will have a normalized weight proportional to the distance between these two predicates in the concept space. This embedding process produces a full connected and symmetric energetic network, properties that are enough to assure the presence of stable agents' attractor. Using this strategy we should have a remarkable set of provers' attractors, with semantic relations among formulae represented inside.

In order to cooperate provers should exchange messages through a blackboard to share the new inferred formulae. It is important to assume that the number of formulae within the blackboard will be always smaller than the number of KB's formulae, otherwise the advantages of this approach will be lost. Inside a blackboard the local results of provers could be listed to be available and used by other provers. They could add these formulae in their own local KB in order to care about the unique global proof state. This communication can be peer to peer according to semantic relations, to maintain bounded the search space, or it can be broadcast. Another utility of such kind of communications is to warn about some dead portion of the proof tree where a backtracking operation was already executed.

In addition the AAN's temperature is another useful tool that could be used to introduce non determinism in agents' activation and so across chained formulae. This can be used to avoid loopy behavior where the system can stuck during the execution of a proof. As a matter of fact simulated annealing was already applied to automatic reasoning and as we have seen AAN gives a good framework to use this method.

Once provers' attractors have been memorized we could use the AAN's query algorithm (algorithm 1 of section 3.6.2) to implement a distributed proof. Starting from the input formula, communicated to every agent through a blackboard broadcast message, each agent can locally propose itself as active in the initial configuration. This will happen according to an adequacy test that establish if its formulae are related with the input formula (e.g. they contain same predicates). To make recent reasonings meaningful, also formulae inferred during a chosen number of initial steps could play the role of new input formulae. The adequacy step go on until a sufficiently rich initial configuration is determined. Then the energetic network will be updated, with the energy going from active provers that are directly related with the input formula (e.g. Cat(bob)), to provers that have semantically related formulae (e.g. Animal(x)).

Finally the execution step will start with the arrival to a provers' attractor, with its corresponding subset of active provers that will start to execute. The new inferred formulae will be used as next input for the following iterations until the proof is completed.

This example illustrates how the AAN architecture can be used for different applications maintaining the same fixed methodology. What is still missing about the attractor provers network is a series of experiments to show that the proposal is at least competitive with other classical provers, as Otter for example (Mcc92).

A possible idea to measure our results is to start from a large collection of ontologies (as the TPTP Problem Library for Automated Theorem Proving) and an automatic prover. The attractor provers network will connect more instances of the same prover. Finally we can compare the performance of the single prover with the performance of the attractor provers network on the same set of formulae.

7.2 Context Aware AAN Learning Algorithm

In section 2.1 we saw that a key issue addressed by the AKIRA energetic model is context awareness, that is the ability to select the most suitable set of agents at current time. This topic was exploited during the previous definition of the AAN architecture. But the previous AAN learning algorithms were dealing mainly with coordination issues: to establish strong energetic links among agents which had success at the same time, so that they will be active at the same time.

In this section we consider context awareness the ability to transfer energy online from agents which had failed to agents which had succeed, without any centralized control. So that successful agents will have more resources to be active and failing agents will start to be idle, leaving their amount of energy available to other agents. Indeed less active agents should have always a possibility to substitute dominant agents if they are starting to have a bad behavior. This approach produces a selection of the most appropriate agents in the current situation using an more adaptive resource allocation strategy. According to it the energetic network should assign less or more energy to agents according to their current relevance. Here the role of an adaptive resource allocation strategy is to produce contextual pressures that optimize the computational cost of an execution.

So a new AAN learning algorithm should be developed addressing more context awareness. As in the AKIRA energetic model the general idea is on one hand to increase the weights of the incoming links of successful agents. On the other hand, if the agent is not successful, the weights of the outgoing links are increased. Like that the energy is transferred from less suitable agents to most suitable ones.

The learning rule just described needs an energetic network with asymmetric links: link from agent i to agent j has a weight different from link from agent j to agent i. Indeed to transfer activation from failing agents to successfulones this asymmetry has to be introduced, because otherwise we could transfer energy from successful agents to failing too. For symmetric Hopfield network is easier to ensure the presence of stable memorized pattern, but also asymmetric Hopfield networks can have stable memorized patterns, also if the recipe is more complex (BP97) (XHK96). It is interesting to note that in (Par86) the asymmetry of links was already considered essential to achieve online learning with Hopfield networks.

With these ideas in mind we developed Algorithm 5, reported below, that has still to be tested.

Note that the constant λ_i within the algorithm is used to maintain the constraint of equation (7.1), used to let the algorithm converge without the

Algorithm 5 *Context Aware* AAN Learning Algorithm

Input: *N* number of agents
activity[1 . . . N] activation variable of agents
success result of the success/failure test of an agent
Window number of steps during successes and failures of agents are counted
successes[1 . . . N] number of successes of agents in a number of steps *Window*
failures[1 . . . N] number of failures of agents in a number of steps *Window*
NumIteration number of learning cycles
J[ ][ ] links of the Energetic Network
λ_i constant used to keep the links' weight exiting from agent *i* bounded
EXECUTE(paramater i) operations executed by agent *i*, returns the result of the success/failure test
Output: *J[ ][ ]* new links of the Energetic Network

```
for $k = 1 \rightarrow N$ do
  $activity[k] \leftarrow RANDOM\{-1, +1\}$
end for
for $n = 1 \rightarrow NumIteration$ do
  repeat
    Energetic Network's update through Equation (5)
  until Energetic Network have reached an Agents' Attractor
  repeat
    $m \leftarrow RANDOM\{1, \dots, N\}$
    if activity[m] = +1 then
      $success \leftarrow EXECUTE(m)$
      if $success =$ true then
        $successes[m] \leftarrow successes[m] + 1$
      else
        $failures[m] \leftarrow failures[m] + 1$
      end if
    end if
  until All agents are chosen once.
  for $i = 1 \rightarrow N$ do
    for $j = 1 \rightarrow N$ do
      $J[i][j] = J[i][j] + ((successes[j] - failures[j]) - (successes[i] - failures[i]) + \lambda_i)$
    end for
    if $NumIteration \ mod \ Window = 0$ then
      $successes[i] \leftarrow failures[i] \leftarrow 0$
    end if
  end for
end for
```

divergence of the weights.

$$\sum_{k}^{N} J_{ik}^2 = 1 \tag{7.1}$$

What it should be proved is the statement of equation (7.2) about the proportionality between agent activation and success index in the stationary state of the learning update rule of equation (7.3).

$$h_i = C(successes(i) - failures(i)) \tag{7.2}$$

$$\dot{J_{ik}} = J_{ik} + \Delta J_{ik} = 0 \tag{7.3}$$

Indeed this will establish the correctness of this new AAN learning algorithm: in fact equation (7.2) states that more an agent has successes and more energy it will gain. On the contrary more an agent has failures and less energy it will gain (exactly what we wanted to implement about context awareness). To have this proportionality when equation (7.3) holds means to achieve it when the learning algorithm has reached its convergence, that happens when the links' weight are not changing anymore.

Chapter 8

Conclusions

In this thesis we presented two modular architectures that exploit attentive processes to deal with bounded rationality. Both architectures follow an approach based on dynamical systems, self-organization and neural networks. Indeed Hopfield networks and dynamical neural fields constitute the meta-levels of the architectures used to deliberate on modules' resource allocation. In this way we tackled the close link between bounded rationality, resource management and covert attention.

The first proposal concerns a MAS architecture called Attractor Agent Network (AAN). The main aim of this architecture is to achieve coordination and context awareness among agents on the basis of a machine learning approach. In the AAN a Hopfield network dynamically distributes the computational resources of the system in order to select the most suitable set of agents in a given situation, on the basis of coordination and context awareness issues. These agents' sets are called agents' attractors and are defined as minimums of the energy function that the network descends during its dynamics.

Three learning algorithms were presented to automatically find or store these agents' attractors. In addition to achieve efficiency a bias parameter was introduced to fix the average number of active agents, in order to bound the amount of computational resources used by the system. The

AAN is well suited when we have to find an efficient collective state of a MAS composed by a high number of agents, because it uses Montecarlo and simulated annealing methods to reduce the computational cost of agents's selection. Also adaptivity is a property of the architecture, which is not only obtained through its learning algorithms but also due to its robustness to noise.

Finally we tested the AAN architecture in three different applications. All these different kind of applications were useful to prove the architectural nature of the AAN. In the first one we investigated the possibility to use the AAN as a metaheuristic for distributed constraint optimization problems, a board game in our case. In the second application the architecture was used to classy points, according to the half planes defined by a piecewise linear function. This was done extending the mixture of experts approach with additional links between the experts. In the third application the AAN faced a categorization task of computer vision. We used the AAN to control an artificial eye with agents specialized in anticipating the outcome of sensorimotor strategies. We showed that a category can be characterized by a set of active agents, so that agents' attractors were representing categories. The computational performance of the AAN architecture was evaluated in all the three applications with particular emphasis in the categorization task where the performance were compared with other computational approaches with competitive results. In the future we think that it will be possible to use the AAN architecture also as a distributed automated theorem prover. In this case the AAN could let us to select only a subset of local provers contextually relevant, reducing the overall proof's computational cost.

The second proposal is based on dynamical field theory and its extension that integrates predictive processes. Although its weaker learning capabilities and memory representation this proposal adds interesting features compared with the AAN in the context of active vision and perceptual decision making. The architecture implements an additional motor command used to indicate the response through the movements of a mouse. In addition, about the eye control, the system avoid loopy behaviors (repetition of the same saccades) and implement a simple form of informativeness preventing the eye to saccade out of the figure. Moreover the architecture showed the importance of bounded rationality in introducing competition within task alternatives and the consequent emergence of top-down processes able to spread the current evidence within the system.

To underline the cognitive plausibility of our proposals we compared the performance of our artificial systems with the results of behavioral experiments of perceptual decision-making. Moreover we conducted an experiment with human participants categorizing the same set of stimuli used for the task of our architectures. Both the human and the artificial accuracies and mouse trajectories were analyzed measuring the uncertainty given by ambiguous stimuli and establishing a quantitative correspondence between the results in the two cases.

Bibliography

[ALTJ09] Athena Akrami, Yan Liu, Alessandro Treves, and Bharathi Jagadeesh. Converging neuronal activity in inferior temporal cortex during the classification of morphed stimuli. *Cereb Cortex*, 19(4):760–776, apr 2009. 78, 80, 81

[Ama77] Shun-ichi Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological cybernetics*, 27(2):77–87, 1977. 93

[Ami89] D. J. Amit. *Modeling Brain Function, The world of Attractor Neural Network*. Cambridge University Press, 1989. ix, 23, 26, 27, 29, 34, 35, 36, 37, 38, 49

[AMT98] P. D. Allopenna, J. S. Magnuson, and M. K. Tanenhaus. Tracking the time course of spoken word recognition using eye movements: Evidence for continuous mapping models. *Journal of Memory and Language*, 38:419–439, 1998. 90

[And09] John R Anderson. *Cognitive psychology and its implications*. Worth publishers, 2009. 7

[B$^+$06] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006. 23

[Bal91] Dana H Ballard. Animate vision. *Artificial intelligence*, 48(1):57–86, 1991. 7, 65

[Bar08] Lawrence W Barsalou. Grounded cognition. *Annu. Rev. Psychol.*, 59:617–645, 2008. 65, 91

[BBFC83] Ann L Brown, John D Bransford, Roberta A Ferrara, and Joseph C Campione. Learning, remembering, and understanding. In *In P. Mussen (Ed.), Handbook of Child Psychology*. Citeseer, 1983. 4

[BDB08] R.H. Baayen, D.J. Davidson, and M.D. Bates. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59, 2008. 88

[Ber95] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995. 1, 12

[BFC$^+$05] A. Baronchelli, M. Felici, E. Caglioti, V. Loreto, and L. Steels. Self-organizing communication in language games. In *Proceedings of the First European Conference on Complex Systems ECCS'05*, 2005. 25

[BGPP03] C. Bernon, M. P. Gleizes, S. Peyruqueou, and G. Picard. ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering. *Engineering Societies in the Agents World 3*, 2577:70–81, 2003. 9

[Bic01] Mark H Bickhard. Function, anticipation and representation. In D M Dubois, editor, *Computing Anticipatory Systems. CASYS 2000 - Fourth International Conference*, pages 459–469, Melville, NY, 2001. American Institute of Physics. 68

[BM09] D. Bates and M. Maechler. *lme4: Linear mixed-eects models using s4 classes*, 2009. http://CRAN.R-project.org/package=lme4. 88

[BP97] U. Bastolla and G. Parisi. Attractors in fully asymmetric neural networks. *Journal of Physics A: Mathematical and General*, 30:5613–5631, 1997. 114

[BP12] Laura Barca and Giovanni Pezzulo. Unfolding visual lexical decision in time. *PLoS ONE*, 7(4):e35932, 2012. 90

[Bro58] Donald Eric Broadbent. *Perception and communication*, volume 2. Pergamon press London, 1958. 6

[Bro86] Rodney A Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23, 1986. 97

[Bro90] Rodney A Brooks. Elephants don't play chess. *Robotics and autonomous systems*, 6(1):3–15, 1990. 5

[CDE^{+}05] Yann Chevaleyre, Paul E Dunne, Ulle Endriss, Jérôme Lang, Michel Lemaitre, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A Rodriguez-Aguilar, and Paulo Sousa. Issues in multiagent resource allocation. 2005. 108

[Cea06] Y. Chevaleyre and et al. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006. 108

[CFL09] Claudio Castellano, Santo Fortunato, and Vittorio Loreto. Statistical physics of social dynamics. *Rev. Mod. Phys.*, 81(2):591–646, may 2009. 25

[CMM97] Anthony Chavez, Alexandros Moukas, and Pattie Maes. Challenger: A multi-agent system for distributed resource allocation. In *Proceedings of the first international conference on Autonomous agents*, pages 323–331. ACM, 1997. 8

[CTP03] P. Clark, J. Thompson, and B. Porter. Knowledge Patterns. *Handbook of Ontologies*, pages 191–207, 2003. 110

[CVQP] Nicola Catenacci Volpi, Jean-Charles Quinton, and Giovanni Pezzulo. How active perception and attractor dynamics shape perceptual categorization: a computational model. *Neural Networks (submitted)*. x, xi, xii, 9, 69, 71, 76, 77, 78, 79, 81, 82, 84

[CYZ01] Samuel Choi, Dit-Yan Yeung, and Nevin Zhang. Hidden-mode markov decision processes for nonstationary sequential decision making. *Sequence Learning*, pages 264–287, 2001. 12

[DB88] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings of the seventh national conference on artificial intelligence*, pages 49–54, 1988. 2

[DBT99] M. Dorigo, E. Bonabeu, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, New York, 1999. 7, 25

[dCPvdH06] G de Croon, EO Postma, and HJ van den Herik. A situated model for sensory–motor coordination in gaze control. *Pattern recognition letters*, 27(11):1181–1190, 2006. 7

[DD95] Robert Desimone and John Duncan. Neural mechanisms of selective visual attention. *Annual review of neuroscience*, 18(1):193–222, 1995. 6

[DDF$^+$90] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990. 111

[DeL99] S. A. DeLoach. Multiagent Systems Engineering: A methodology and Language Designing Agent Systems. In *Proc. Int. Bi-Conf. Workshop Agent-Oriented Information Systems*, pages 45–57, may 1999. 9

[Den89] Daniel C Dennett. *The intentional stance*. MIT press, 1989. 3

[DK06] Yiannis Demiris and Bassam Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and autonomous systems*, 54(5):361–369, 2006. 12

[DKS07] Rick Dale, Caitlin Kehoe, and Michael J. Spivey. Graded motor responses in the time course of categorizing atypical exemplars. *Mem Cognit*, 35(1):15–28, Jan 2007. 89

[dMM06] A. de Martino and M. Marsili. Statistical mechanics of socio-economic systems with heterogenous agents. *Journal of Physics A: Mathematical and General*, 39(43):R465, 2006. 25

[Dor92] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, 1992. 25

[Dre91] G L Drescher. *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge, MA, 1991. 68

[DSKK02] Kenji Doya, Kazuyuki Samejima, Ken-ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369, 2002. 22

[EHRLR80] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *ACM Comput. Surv.*, 12:213–253, jun 1980. 24

[FA10] Jonathan B Freeman and Nalini Ambady. MouseTracker: software for studying real-time mental processing using a computer mouse-tracking method. *Behav Res Methods*, 42(1):226–241, feb 2010. 86

[FDF11] Jonathan B Freeman, Rick Dale, and Thomas A Farmer. Hand in motion reveals mind in motion. *Front Psychol*, 2:59, 2011. 90

[FG98] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, pages 128–135, jul 1998. 9

[FSPB04] Gerald Fritz, Christin Seifert, Lucas Paletta, and Horst Bischof. Rapid object recognition from discriminative regions of interest. In *Proceedings of the national conference in Artificial Intelligence*, pages 444–449. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004. 7

[Gar00] Peter Gardenfors. *Conceptual Spaces: The Geometry of Thought*. MIT Press, Cambridge, MA, USA, 2000. 96

[GG84] Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, nov 1984. 52

[GG91] Matthew L Ginsberg and Donald F Geddis. Is there any need for domain-dependent control information. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 452–457. Citeseer, 1991. 4, 106

[Goo52] Irving John Good. Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 107–114, 1952. 2

[Goo71] IJ Good. The probabilistic explication of information, evidence, surprise, causality, explanation, and utility. *Foundations of statistical inference*, pages 108–141, 1971. 2

[Got12] Jacqueline Gottlieb. Attention, learning, and the value of information. *Neuron*, 76(2):281–295, 2012. 3

[HDK84] A Hartwig, F Daske, and S Kobe. A recursive branch-and-bound algorithm for the exact ground state of Ising spin-glass models. *Computer Physics Communications*, 32(2):133–138, 1984. 49

[Hem65] Carl Gustav Hempel. *Aspects of scientific explanation: and other essays in the philosophy of science*. Free Press, 1965. 1

[HH05] Dietmar Heinke and Glyn W Humphreys. Computational models of visual selective attention: A review. *Connectionist models in cognitive psychology*, 1(4):273–312, 2005. 7

[Hof95] D. R. Hofstadter. *Fluid Concepts and Creative Analogies*. Basic Books, 1995. 24

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975. 25

[Hol89] J. H. Holland. Using Classifier Systems to Study Adaptive Nonlinear Networks. *Lectures in the Sciences of Complexity*, pages 463–499, 1989. 24

[Hop82] J.J. Hopfield. Neural Networks and physical systems with emergent selective computation abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558, 1982. 23, 26, 34

[Hor88] Eric Horvitz. Reasoning about beliefs and actions under computational resource constraints. *Int. J. Approx. Reasoning*, 2(3):337–338, 1988. 2, 3

[Hor01] Eric Horvitz. Principles and applications of continual computation. *Artificial Intelligence*, 126(1):159–196, 2001. 2

[How66] Ronald A Howard. Information value theory. *Systems Science and Cybernetics, IEEE Transactions on*, 2(1):22–26, 1966. 3

[HR85] Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26:251–321, aug 1985. 24

[HT85] John J Hopfield and David W Tank. neural computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985. 25, 31, 53

[HWK01] Masahiko Haruno, Daniel M Wolpert, and Mitsuo Kawato. Mosaic model for sensorimotor learning and control. *Neural computation*, 13(10):2201–2220, 2001. 12

[IGCGal99] Carlos Argel Iglesias, Mercedes Garijo, and José Centeno-Gonz a lez. A Survey of Agent-Oriented Methodologies. In *Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages*, pages 317–330, London, UK, 1999. Springer-Verlag. 9

[IK01] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3):194–203, 2001. 7

[IKN98] L Itti, C Koch, and E Niebur. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, nov 1998. 7, 67

[Isi25] Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, 31(1):253–258, 1925. 26

[Jac87] John V. Jackson. Idea for a mind. *SIGART Bull.*, pages 23–26, jul 1987. 24

[Jam90] William James. The principles of psychology, 1890. 6

[JJNH91] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 1991. ix, 19, 20, 57

[JSS08] J.S. Johnson, J.P. Spencer, and G. Schoner. Moving to higher ground: The dynamic field theory and the dynamics of visual cognition. *New Ideas in Psychology*, 26:227–251, 2008. 98

[JY88] John Jonides and Steven Yantis. Uniqueness of abrupt visual onset in capturing attention. *Perception & Psychophysics*, 43(4):346–354, 1988. 6

[Kau93] S. A. Kauffman. *The origin of order: Self-organization and selection in evolution*. Oxford University Press, New York, 1993. 37, 105

[Kaw99] Mitsuo Kawato. Internal models for motor control and trajectory planning. *Current opinion in neurobiology*, 9(6):718–727, 1999. 12

[KGK11] Tim C. Kietzmann, Stephan Geuter, and Peter Koenig. Overt visual attention as a causal factor of perceptual awareness. *PLoS One*, 6(7):e22614, 2011. 75

[KGM06] Manuel Kolp, Paolo Giorgini, and John Mylopoulos. Multi-Agent Architectures as Organizational Structures. *Autonomous Agents and Multi-Agent Systems*, 13:3–25, 2006. 8, 9

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983. 25, 52

[KH78] S Kobe and A Hartwig. Exact ground state of finite amorphous Ising systems. *Computer Physics Communications*, 16(1):1–4, 1978. 49

[Kok94] Boicho Nikolov Kokinov. The Context-Sensitive Cognitive Architecture DUAL. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pages 502–507, Citeseer, 1994. Citeseer. 14, 34

[LC05] Claire Lefebvre and Henri Cohen, editors. *Handbook of Categorization*. Elsevier, dec 2005. 65

[Lei76] Harvey Leibenstein. *Beyond economic man: A new foundation for microeconomics*. Harvard university press Cambridge, MA, 1976. 2

[Lig99] T. M. Liggett. *Stochastic Interacting Systems: Contact, Voter and Exclusion Processes*. Springer, 1999. 25

[Luk09] Sean Luke. Essentials of metaheuristics. *Lecture notes, George Mason University. Free access: http://cs. gmu. edu/˜ sean/book/metaheuristics/*, 2009. 25

[Mac87] Mark J Machina. Choice under uncertainty. *Encyclopedia of Cognitive Science*, 1987. 1

[Mar99] Jerzy Martyna. Neural network approach to design of distributed hard real-time systems. *Computational Intelligence*, pages 118–131, 1999. 107

[MBiBV11] Zenon Mathews, Sergi Bermúdez i Badia, and Paul FMJ Verschure. Pasar: An integrated model of prediction, anticipation, sensation, attention and response for artificial sensorimotor systems. *Information Sciences*, 2011. 12

[Mcc92] W.W. Mccune. Automated discovery of new axiomatizations of the left group and right group calculi. *Journal of Automated Reasoning*, 9(1):1–24, 1992. 113

[McC01] James L McClelland. The time course of perceptual choice: The leaky, competing accumulator model. *Psychological review*, 108(3):550–592, 2001. 90

[MD85] Jeffrey Moran and Robert Desimone. Selective attention gates visual processing in the extrastriate cortex. *Science*, 229(4715):782–784, 1985. 6

[MFN10] Marco Mirolli, Tomassino Ferrauto, and Stefano Nolfi. Categorisation through evidence accumulation in an active vision system. *Connection Science*, 22(4):331–354, 2010. 7

[Min88] Marvin Minsky. *Society of mind*. Simon & Schuster, 1988. 105

[MNM93] William H Merigan, Tara A Nealey, and JH Maunsell. Visual effects of lesions of cortical area v2 in macaques. *The Journal of neuroscience*, 13(7):3180–3191, 1993. 7

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biology*, 5(4):115–133, 1943. 26

[Mur12] Kevin P Murphy. *Machine learning: a probabilistic perspective*. The MIT Press, 2012. 23

[Nei67] Ulric Neisser. *Cognitive psychology*, volume 4. Appleton-Century-Crofts New York, 1967. 6

[NI05] Vidhya Navalpakkam and Laurent Itti. Modeling the influence of task on attention. *Vision research*, 45(2):205–231, 2005. 7

[NI07] Vidhya Navalpakkam and Laurent Itti. Search goal tunes visual features optimally. *Neuron*, 53(4):605–617, 2007. 7

[OCVPB] Dimitri Ognibene, Nicola Catenacci Volpi, Giovanni Pezzulo, and Giancluca Baldassarre. Learning Epistemic actions: a Model-Free, Memory-Free Reinforcement Learning approach. In *Proceedings of the 2nd International Conference on Biomimetic and Biohybrid Systems LIVING MACHINES 2013 (in press)*. 3

[oE] University of Emory. *Research on Spin Glasses*. www.physics.emory.edu/faculty/boettcher/Research/spinglasses.htm. ix, 47

[OK04] Cheryl Olman and Daniel Kersten. Classification objects, ideal observers and generative models. *Cognitive Science*, 28(2):227–239, 2004. 70

[ON01] J K O'Regan and A Noe. A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences*, 24(5):883–917, 2001. 68

[OVP11] Dimitri Ognibene, Nicola Catenacci Volpi, and Giovanni Pezzulo. Learning to grasp information with your own hands. In *Towards Autonomous Robotic Systems*, pages 398–399. Springer, 2011. 3

[Par86] G. Parisi. Asymmetric neural networks and the process of learning. *Journal of Physics A: Mathematical and General*, 19:L675, 1986. 114

[PC05] G. Pezzulo and G. Calvi. Designing and Implementing MABS in AKIRA. *Multi-Agent and Multi-Agent-Based Simulation*, pages 49–64, 2005. 24

[PC07] G. Pezzulo and G. Calvi. Designing modular architectures in the framework Akira. *Multiagent and Grid Systems*, 3(1):65–86, 2007. ix, 14, 16, 24, 34, 108

[Pez09] G. Pezzulo. Exploiting MAS Self-organization for Distributed Constraint Satisfaction Problems. *Interantional Transactions on Systems Science and Applications*, 5(3):285–295, 2009. 25, 53

[Pez11] Giovanni Pezzulo. Grounding Procedural and Declarative Knowledge in Sensorimotor Anticipation. *Mind and Language*, 26(1):78–114, 2011. 68

[PFS05] Lucas Paletta, Gerald Fritz, and Christin Seifert. Q-learning of sequential attention for visual object recognition from informative local descriptors. In *Proceedings of the 22nd international conference on Machine learning*, pages 649–656. ACM, 2005. 7

[PHC00] V Paraskevopoulos, MI Heywood, and CR Chatwin. Continuous optimal controllers using hierarchical mixtures of experts. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 4, pages 331–336. IEEE, 2000. 22

[Pin06] Axel Pinz. Object Categorization. *Foundations and Trends in Computer Graphics and Vision*, 1(4):255–353, 2006. 65

[PP00] Lucas Paletta and Axel Pinz. Active object recognition by view integration and reinforcement learning. *Robotics and Autonomous Systems*, 31(1):71–86, 2000. 7

[Put94] Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., 1994. 1, 12

[PVM87] G. Parisi, M. Virasoro, and M. Mezard. *Spin Glass Theory and beyond*. Lecture Notes in Physics. World Scientific, 1987. 45, 46

[QCVBP] Jean-Charles Quinton, Nicola Catenacci Volpi, Laura Barca, and Giovanni Pezzulo. The cat is on the mat. Or is it a dog? Dynamic competition in perceptual decision making. *IEEE Transactions on Systems Man and Cybernetics: Systems (in press)*. xii, xiii, xiv, 9, 86, 87, 89, 95, 99, 100, 101, 102

[QG$^+$12] Jean-Charles Quinton, Bernard Girau, et al. Spatiotemporal pattern discrimination using predictive dynamic neural fields. *BMC Neuroscience*, 13(Suppl 1):O16, 2012. 93, 98, 106

[QGL11] Jean-Charles Quinton, Bernard Girau, and Mathieu Lefort. Competition in high dimensional spaces using a sparse approximation of neural fields. In *From Brains to Systems: Brain Inspired Cognitive Systems 2010*. Springer-New York, 2011. 93, 98, 106

[RB10] Constantin A Rothkopf and Dana H Ballard. Credit assignment in multiple goal embodied visuomotor behavior. *frontiers in Psychology*, 1, 2010. 11

[RHW86] David E Rumelhart, Geoffrey E Hintont, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 19

[RMG$^+$76] Eleanor Rosch, Carolyn Mervis, Wayne Gray, David Johnson, and Penny Boyes-Braem. Basic Objects in Natural Categories. *Cognitive Psychology*, 8, 1976. 96

[Roj95] R. Rojas. *Neural Network: A Schematic Introduction*. Springer-Verlag, 1995. 29, 41

[Ros62] Frank Rosenblatt. Principles of neurodynamics. 1962. 26

[RS95] Stuart J Russell and Devika Subramanian. Provably bounded-optimal agents. *Journal of Artificial Intelligence Research*, 2:575–609, 1995. 2

[Rus97] Stuart J Russell. Rationality and intelligence. *Artificial intelligence*, 94(1):57–77, 1997. 2

[RV06] Nicolas P Rougier and Julien Vitay. Emergence of attention within a neural population. *Neural Networks*, 19(5):573–581, 2006. 93

[RW89] Stuart Russell and Eric Wefald. On optimal game-tree search using rational meta-reasoning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, volume 4, pages 334–340. Morgan Kaufmann, 1989. 3

[RW91] Stuart Jonathan Russell and Eric H Wefald. *Do the right thing: studies in limited rationality*. The MIT Press, 1991. 2, 4, 106

[SBR05] Nathan Sprague, Dana Ballard, and Al Robinson. Modeling attention with embodied visual behaviors. *ACM Transactions on Action and Perception*, 2005. 11

[SC11] Pedro Santana and Luís Correia. Swarm cognition on off-road autonomous robots. *Swarm Intelligence*, 5(1):45–72, 2011. 7

[Sch03] F. Schweitzer. *Brownian Agents and Active Particles: Collective Dynamics in the Natural and Social Sciences*. Springer, 2003. 25

[Sch08] G Schöner. Dynamical systems approaches to cognition, 2008. 91, 92

[SCM97] P. M. Silva, C. Carderiea, and Z. Mammeri. Solving real-time scheduling problems with hopfiled-type neural networks. *Proceedings of the 23rd EUROMICRO Conference*, pages 671–678, 1997. 107

[SDJ$^+$95] C. Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt, and G. Rinaldi. Exact ground states of Ising spin glasses: New experimental results with a branch-and-cut algorithm. *Journal of Statistical Physics*, 80(1-2):487–496, 1995. 49

[SDKG10] Michael J. Spivey, Rick Dale, Guenther Knoblich, and Marc Grosjean. Do curved reaching movements emerge from competing perceptions? a reply to van der wel et al. (2009). *J Exp Psychol Hum Percept Perform*, 36(1):251–254, Feb 2010. 90

[SGS10a] Adam N Sanborn, Thomas L Griffiths, and Richard M Shiffrin. Uncovering mental representations with Markov chain Monte Carlo. *Cognitive Psychology*, 60(2):63–106, 2010. x, 65

[SGS10b] Adam N Sanborn, Thomas L Griffiths, and Richard M Shiffrin. Uncovering mental representations with Markov chain Monte Carlo. *Cogn Psychol*, 60(2):63–106, mar 2010. 70, 74

[SHDK12] Norikazu Sugimoto, Masahiko Haruno, Kenji Doya, and Mitsuo Kawato. Mosaic for multiple-reward environments. *Neural computation*, 24(3):577–606, 2012. 12

[Sim57] H. A. Simonm. *Models of man - social and rational*. John Wiley and Sons, 1957. 2

[Sim82] Herbert Alexander Simon. *Models of bounded rationality: Empirically grounded economic reason*, volume 3. MIT Press (MA), 1982. 2

[SJBH11] BT Sullivan, LM Johnson, DH Ballard, and MM Hayhoe. A modular reinforcement learning model for human visumoto behavior in a driving task. In *Proceedings of the AISB 2011 Symposium*, pages 33–40, 2011. 12

[SLB08] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008. 53

[SMD+11] Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 761–768, 2011. 12

[SMHK12] Norikazu Sugimoto, Jun Morimoto, Sang-Ho Hyon, and Mitsuo Kawato. The emosaic model for humanoid robot control. *Neural Networks*, 29:8–19, 2012. 12

[SN09] Joo-Hyun Song and Ken Nakayama. Hidden cognitive states revealed in choice reaching tasks. *Trends Cogn Sci*, 13(8):360–366, Aug 2009. 90

[Spi07] M. Spivey. *The continuity of mind*. Oxford University Press, USA, 2007. 89

[SS07] Frederick Shic and Brian Scassellati. A behavioral analysis of computational models of visual attention. *International Journal of Computer Vision*, 73(2):159–177, 2007. 7

[Syc98] K. P. Sycara. Multiagent Systems. *AI Magazine*, 19(2):79–92, 1998. 8, 106

[Tay99] JG Taylor. Neural bubbledynamics in two dimensions: foundations. *Biological Cybernetics*, 80(6):393–409, 1999. 93

[Tir95] B. Tirozzi. *Modelli matematici di Reti Neurali*. CEDAM, 1995. 50, 51

[TK75] Amos Tversky and Daniel Kahneman. *Judgment under uncertainty: Heuristics and biases*. Springer, 1975. 1

[TM96] Stefan Treue and John HR Maunsell. Attentional modulation of visual motion processing in cortical areas mt and mst. 1996. 6

[TNNI08] Jun Tani, Ryunosuke Nishimoto, Jun Namikawa, and Masato Ito. Codevelopmental learning between human and humanoid robot using a dynamic neural-network model. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 38(1):43–59, 2008. 22

[VJ89] Michel Verleysen and Paul GA Jespers. An analog vlsi implementation of hopfield's neural network. *Micro, IEEE*, 9(6):46–55, 1989. 32

[VNM47] John Von Neumann and Oskar Morgenstern. The theory of games and economic behavior. 1947. 1

[WCR65] L. Wos, D. Carson, and G. Robinson. Efficient and completeness of the set-of-support strategy in theorem proving. *Journal of the Association for Computing Machinery*, (14):698–704, 1965. 110

[WD90] Michael P Wellman and Mark Derthick. *Formulation of tradeoffs in planning under uncertainty*. Pitman London, 1990. 2

[WH$^+$60] Bernard Widrow, Marcian E Hoff, et al. Adaptive switching circuits. 1960. 26

[Wic02] Christopher D Wickens. Multiple resources and performance prediction. *Theoretical issues in ergonomics science*, 3(2):159–177, 2002. 7

[Wic08] Christopher D Wickens. Multiple resources and mental workload. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 50(3):449–455, 2008. 7

[WK98] Daniel M Wolpert and Mitsuo Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7):1317–1329, 1998. 12

[Woo09] Michael Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley and Sons, 2009. 8

[Wri68] JM von Wright. Selection in visual immediate memory. *The Quarterly journal of experimental psychology*, 20(1):62–68, 1968. 6

[XHK96] Z. B. Xu, G. Q. Hu, and C. P. Kwon. Asymmetric Hopfield-type Networks: Theory and Applications. *Neural Networks*, 9(3):483–501, 1996. 114

[YM88] H. Chang Y. Miyashita. Neuronal correlate of pictorial short-term memory in the primate temporal cortex. *Nature*, 331(68), 1988. 26

[Zil95] Shlomo Zilberstein. Operational rationality through compilation of anytime algorithms. *AI Magazine*, 16(2):79, 1995. 2

[Zil08] Shlomo Zilberstein. Metareasoning and bounded rationality. *Metareasoning: Thinking about Thinking, MIT Press*, 2008. ix, 3, 4, 106