CHALMERS

Transparent Neural Networks, an Implementation

Master´s Thesis in Applied Information Technology

JUAN SEBASTIAN OLIER

Department of Applied Information Technology
Division of Intelligent Systems Design
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, 2012
Report No. 2012:010

REPORT NO. 2012:010

Transparent Neural Networks, an Implementation

JUAN SEBASTIAN OLIER.

Department of Applied Information Technology
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2012

Transparent Neural Networks, an Implementation

Master Thesis in Applied Information Technology

JUAN S. OLIER

ISSN: 1651-4769

Technical report no 2012:010
Department of Applied Information Technology
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Abstract

The present work is related to a research project denominated Transparent Neural Networks (Strannegård 2012); this project aims to propose a model capable of higher cognitive functions such as deductive and inductive reasoning by means of transparent, simple and interpretable structures and functionalities. This simplicity includes interactive building rules based on the manipulation of basic structures and elements. It considers characteristics of the developmental robotics and cognitive modeling.

Taking all of these concepts and goals as a basis, the main objective of this thesis is to help in the development of that model by implementing a toolbox that allows the creation and evaluation of the networks leading to conclusions and meaningful feedback that fosters proposals for further development on the model.

Contents

Introduction

The goals of the Transparent Neural Networks project, to which this thesis is directly connected, are related to achieving higher cognitive functions such as deductive and inductive reasoning, as well as automatic learning by means of transparent and interpretable structures. But all of these objectives are in turn related to problems that have been studied in different fields and which solutions imply applications of huge relevance.

In order to contextualize the concepts associated to this model an introductory description is presented depicting some approaches that aim to cope with problems similar to the ones faced here. These approaches correspond to some ideas and models emerged throughout the development of the fields of cognitive modeling, problem solving and robotics, and that are of relevance for understanding the challenges and needs addressed by this research project.

Background

The challenge of creating systems capable of mimic reasoning and cognition has been addressed by many and from different perspectives and disciplines. There are some proposals broader than others, but the main goal has been mainly related to the ability of creating concepts and manipulating them in order to draw conclusions and deliver responses.

Most of the approaches could be classified in relation to the way information is organized and processed; the general division usually is into emergentist, symbolic and statistical or probabilistic models. Among these there are certain conceptual differences that give advantages and disadvantages to each as will be shown.

In the mentioned classification the division into Symbolic and Emergentist -also called non-Symbolic- approaches is broadly used in the field. They basically differ in the way they create, represent and manipulate concepts and knowledge. The symbolic perspective in turn proposes that cognitive systems use specific symbols as a representation of knowledge and find solution by carrying out processes on these representations. Complementarily, the emergentist perspective proposes that the knowledge is represented in a distributed manner into basic elements, and processing is carried on this distributed knowledge in a complex and meaningful way. (Troy D, 2003)

Another way of classifying the models is by the top-down and bottom-up differentiation. Top-down perspectives assume that the basis of cognition lays on the symbolic abstractions and therefore only that is needed to achieve intelligent behavior, thus the relevance of whatever structure that is below the whole process can be neglected. On the other hand, bottom-up perspectives assume that intelligence and cognition emerge from the behavior of atomic components in a structure and the way they relate to each other. They argue that it is possible to achieve abstract associations from basic processes at the

bottom of a structure. (McClelland, Botvinick, Noelle, Plaut, Rogers, Seidenberg and Smith, 2010)

However these classifications are generalizations, there are also models that combine the approaches in attempts to achieve better results but they do not fit completely in any on the groups above. To better understand the characteristics of each of the approaches they will be described below by giving some notions about their utility as well as advantages, disadvantages and discrepancies among them.

Emergentist approaches

The emergentist approaches state that any behavior appears as a consequence of basic changes happening at a very low level; cognition and abstract representations are therefore seen just as a consequence of processes carried out in underlying structures. The main examples of this perspective are the connectionist approaches; in these behavior emerge from the connections among simple processing units that acquire knowledge through experience by adjusting strengths between connections or either creating or removing them. (McClelland, Botvinick, Noelle, Plaut, Rogers, Seidenberg and Smith, 2010).

The emergentist approaches propose alternatives most of the times inspired by nature, or at least by our understanding of it. The idea arises from the observation of how every structure around us emerges as patterns created by the interaction of smaller and simpler processes. Consequently, cognition also seems to be an emergent result of the interaction of simple and more understandable units.

Based on that perspective, and considering that the brain is based on the interconnection of neurons, it is assumed that is possible to have an emergent behavior from the simulation of simple units that, at least in principal, might mimic the behavior of real neurons.

This whole idea is the foundation for many connectionist approaches and specially the artificial neural networks. These lasts attempt to mimic the behavior of real neural nets by interconnecting units that share information through weighted signals and activation functions.

The artificial neural networks have been seen in different ways as a feasible model of cognition and many claim that they not only model the cognition, but also simulate the actual underlying processes (Sharkey, 2009). Nevertheless, many arguments against connectionism and artificial neural networks strongly highlight that, in comparison to real neural networks, the models are oversimplified: after all, the real biological process is not yet completely understood and it could happened that certain assumptions may lack foundations.

For instance, some other proposals on connectionism claim that the connections in a network should not rely on weights. This is the case of the HTM model proposed by Numenta. They argue that though in real neural

networks synapses might present a phenomenon similar to the one represented by the weights, their values tend to be random or volatile, therefore it cannot be safely assumed that calculations in the brain actually rely on those weights (Numenta Inc., 2011).

Nevertheless the artificial neural networks have been subject of research for many years; some have perceived them as model of cognition and others, maybe more successfully, as an optimization tool.

In the case of simulating cognition a good performance has been found based on fitting the network's response to some psychology experiments (Sharkey, 2009). However, this kind of experiments are limited to a specific task, and even when the data is fit, it cannot be said that the networks mimics the process itself or even more risky that it is comparable to actual reasoning; in fact, it is really hard to interpret the actual behavior in the network that leads to the result, but it is known that at the end it performs nothing but an error minimization task.

Moreover, traditional neural networks are dynamic systems that can accomplish very good performance on optimization and data fitting. This is why most of the development on this field has been done aiming to solve particular problems hard for traditional mathematical optimization methods; actually most of the variations of the artificial neural networks have emerged to fit particular optimization problems. But when it comes to the ability of modeling cognition or actual reasoning based on them it is not so clear that these structures possess it.

Nevertheless artificial neural networks are not the only connectionist model; some other models have been proposed based on connectionist ideas, specially aiming to create the ability of learning concepts and use them for inference. An example of a connectionist model is the Shrutí architecture (The International Computer Science Institute, 2012).

Shrutí is an architecture that focuses mainly on drawing inferences which its authors proclaim to be performed "effortlessly, spontaneously, and with remarkable efficiency". The project attempts to show how a connectionist model can be capable of encoding semantics, systematic mapping and knowledge about entities, and also be available to perform reflexive inferences in a fast and efficient manner. This is done by creating structures that represent schemas by focal cells clusters and generating inferences by the propagation of rhythmic activity over those clusters. Thus, all information processing is based on temporal synchrony throughout a structured neural representation. This fact is claimed to demonstrate how such a connectionist structure is sufficient to achieve rational processing in the brain. This model is related to different projects related to decision making, problem solving, and planning and language acquisition (The International Computer Science Institute, 2012).

In general connectionist models are capable of simplifying and generalizing data from complex inputs to more reduced spaces in the way of inductive learning. Also, some connectionist models have been merged with other approaches to achieve better capabilities as will be described further below.

Probabilistic approaches

Probabilistic models can be classified mainly as top-down approaches that relate concepts and perform selections depending on probabilities learnt through experience. The most basic and classic yet relevant example of this are the Bayesian Networks.

Probabilistic models are defended as being capable of yielding great flexibility for exploring the representations and inductive biases that underlie human cognition (Griffiths, Chater, Kemp, Perfors and Tenenbaum, 2010). That assertion is based on the assumption that whatever behavior a system displays, its causes can be easily described by means of probabilities.

This flexibility at the time of exploring inductive behaviors is a characteristic that represents an advantage when it comes to fully understand the system and what it represents. In fact, this idea has been used against Bottom-Up and some connectionist models by arguing that, even when both kind of models could successfully address similar problems, the way emergentist models solve them is not necessarily as understandable or transparent to the user as a probabilistic model could be (Griffiths, Chater, Kemp, Perfors and Tenenbaum, 2010). However, reality is that mathematics behind probabilistic inferences can easily go beyond unaided intuition, and even simple rules can become intractable as models are scaled up to fit real world problems (McClelland, Botvinick, Noelle, Plaut, Rogers, Seidenberg and Smith, 2010). That may contradict the claim of probabilistic approaches being capable to draw more understandable descriptions of reasoning and cognition.

Nonetheless, this leads to a more general topic than the one concerned to this document but that still affects the fundamentals of the Transparent Neural Networks model to be introduced. This is the more general question on how intuitive the fundamentals of cognition or reasoning could really be. As mentioned by (Chater, Tenenbaum and Yuille, 2006) people struggle not only with probability but with all the branches of mathematics, and this does not changes the fact that, for example, as hard to understand as it could be, Fourier analysis is fundamental in audition and vision in biological systems.

Therefore, it may be sound to state that analyzing the complexity behind the model or its easiness of interpretation may not be the best choice to compare performance. But regardless of which could be the best measure, it is also undoubtedly relevant to identify biases in the measures that may be favoring particular interpretations of reasoning or cognition when proposing or evaluating a model.

As the field has been broadly focused on architectures and structures such as those in connectionists or rules based models, the performance measurement has to do with their characteristics and the ideas behind them, such as logic and heuristics (Griffiths, Chater, Kemp, Perfors and Tenenbaum, 2010). This kind of biasing in the analysis may of course affect models that are based on different perspectives such as the probabilistic models. However, regardless of the

models that it could benefit or affect, this kind of biasing may lead to extremely dangerous assumptions; an example of this is treating logic almost as an equivalent to the reasoning itself. Biasing the conception of reasoning in that way is indeed such a strong assertion that may cause a tremendously narrow view on the problem.

In any case, the problem on biasing the analysis, and therefore the assessment of performance of a model may come from any perspective. In a certain way probability, or mathematical models in general, can be seen as a description of thought (Chater, Tenenbaum and Yuille, 2006), and that assertion might be useful for many problems, however one should bear in mind the it will only be a description and not thought itself either its equivalent.

To conclude, aside of the assertion on reasoning or intelligence, it can be said that probabilistic models have the abilities to infer and generalize, and that allow solving problems and creating some useful behavior that may be of interest for particular applications such as those found in data mining and machine learning.

Symbolic approaches

The symbolic approaches assume that cognition can be modeled by manipulating symbols and relations among them by means of structures and rules; in this group can be included approaches such as the logic and rule based systems. These models are mainly used as representation systems and are capable of inferences and deductive learning. However, they are criticized for lacking the ability of inductive reasoning as they normally are based on structures or concepts designed by the programmer but not learnt through experience.

From the very beginning of formal computation and the first ideas on AI, symbols and specially logic was considered as a basic mechanism by which minds work. The idea is that symbolic representations stands at the very core of how intelligence work, and therefore the focus is set on what symbolic knowledge an agent would need in order to behave intelligently (Bringsjord,2008). Then this perspective focuses not on how the knowledge arises but on how it should be used.

The symbolic approaches envision cognition as some sort of computer programs and describe aspects of cognition and their emerging results as a set of basic computational processes, claiming that this idea could produce, for example, predictions with performance comparable to humans (Lewis, 1999).

A foundation of this approach is the so called “physical symbol system hypothesis” proposed by Newell and Simon. Its idea is to use basic symbols as representational entities, combine them to form expressions and manipulate those expressions to create even new ones. Their claim stated that "A physical symbol system has the necessary and sufficient means for general intelligent action", and is an idea that has been the foundation of massive efforts in research in AI (Sun, 2001).

Many symbolic representations aim to capture and organize knowledge in the form of structures or architectures. The idea for the structures is to organize knowledge by creating relational groups of symbolic entities that may in turn contain or be contained by other groups. A well-known of these representations are the semantic networks; these networks mainly used for language representation and processing, are formed by labeled nodes representing concepts related in turn by labeled links (Sun, 2001).

An example of a symbolic architecture is the SOAR project (University of Michigan, 2012). It is a production rule system where problem solving is basically related to representing a search space. All the decisions are taken by using the interpretation of the sensory data and the compilation of relevant knowledge coming from previous experiences. The main goal is basically to create a general problem solver where every problem, regardless of its characteristics, is defined as a search space (University of Michigan, 2012).

On the other hand, many symbolic models have used logic as a representational language. For many the idea of logic has always been a part of the supreme goal of developing intelligent machines; the promises back in the 50s was on using logic as the mechanism to build computational artifacts available to even exceed human minds in terms of intelligence. The truth after many years of research is that the fundamental ideas based on formal logic have failed to accomplish the task. Nevertheless, people still believe in logic and it plays a fundamental role on many algorithms and, as mentioned before, is used to analyze performance in different kinds of models (Perlovsky, 2007).

Many go back to Aristotle to describe logic and argue how even from syllogisms intelligent responses can be described. The main element of this argument arises from the idea that Aristotle inferred that certain context-independent structures can describe and predict human thinking. However, it has to be also taken into account that these studies on logic came from the search of a supreme way of argumentation but never as a theory of mind (Bringsjord, 2008).

Thus, to talk about logic as a model that describes human thinking may be too broad in the sense that argumentation is a particular characteristic of language and its consequence. But the fact that language, and after it logic, emerge from reasoning and thinking does not necessarily mean that they are the basis from which those phenomena emerge. In other words, arguing that a consequence can be also the basis of the same process may lead to some partial understanding of the problem.

Nevertheless, approaches based on logic have been successfully implemented to solve particular problems often related to symbolic manipulation. Many problems in language processing have been addressed by these approaches, complemented in many cases by probabilistic models. Other broadly explored area is the one concerned with constrain satisfaction problems, which are addressed by logic based models and particularly bi-valuated systems.

Logic approaches have also evolved during the last decades to allow more flexibility than the formal logic, which as limited to discrete truth values runs into troubles easily. Just in 1902 Russell showed a whole in formal logic which caricature is described by this simple example: “A barber shaves everybody who does not shave himself. Does the barber shave himself?” (The Cambridge Handbook of Computational Psychology , 2008 pp. 127-169). Any possible answer to this problem -yes or no- is contradictory.

Problems like that and later more complex ones, led to the rise of concepts such as multivalued and fuzzy logic, where variables can take many values or virtually any value in an interval between the classical true and false. These more flexible approaches have allowed addressing a broader range of problems, but have also shown the need to merge logic and other approaches to achieve better results as described in the next subsection.

Hybrid models

It seems reasonable to aim for a model that includes both top-down and bottom-up ideas as they can be complementary. For example, symbolic approaches that are mainly concerned with deductive reasoning may be complemented by connectionist approaches that are mainly focused on inductive learning (d’Avila Garcez and Lamb, 2011). Thus, it is easy to advocate for the search of such a model and indeed that is not a new idea, nonetheless as easy to argue about its reasons not so easy is the task of developing it.

A hybrid model mainly concerns with merging characteristic of very heterogeneous systems such as the symbolic and connectionist models. These two approaches have very different types of representation, learning and processing; therefore, most of the proposals are architectures that attempt to use symbolic perspective for manipulation and connectionist approaches for learning. In other words, a top-down system that is fed by a bottom-up one (Troy D, 2003).

A way of seeing this, proposed by Troy (2003), is that cognition can be considered as a cognitive continuum with two ends, at a highest end the symbolic processing is carried out, which could be interpreted as the equivalence of the prefrontal cortex in the human brain. At the other end of that continuum, the lowest level is related to the most basic input processing, which in the human system could be equivalent to the reflex nerves. But still, the link between the two ends of that continuum is not yet clear.

Hence, though in hybrid architectures the sub-symbolic systems present favorable issues related to learning, the symbolic processing is still mainly related to representation and inference which transfers many of the symbolic systems flaws to the hybrid structures (Sun, 2001). This means that at the symbolic level the structures are still highly dependent of a knowledgeable user, and therefore not much is really left to learning through experience (Troy D, 2003).

Nevertheless, the usage of various approaches certainly enhances the capability of the models. Several architectures have been developed based on hybrid structures and some of them have achieved reasonable results and are known as relevant cognitive architectures, among them ACT-R is a typical example.

ACT-R (ACT- R Research Group Department of Psychology, Carnegie Mellon University, 2012) is an architecture that is born with the goal of understanding human cognition and how knowledge is organized and used to produce intelligent behavior. This architecture has been evolving for many years reaching interesting results in various fields related to cognition.

This architecture has been used by researchers to produce data on theories that can be directly compared to experiments with human participants. This allows verifying models on cognition directly by means of the architecture.

Some of the models created with ACT-R include: learning and memory, problem solving and decision making, language and communication, perception and attention, cognitive development, and individual differences.

ACT-R as a hybrid architecture has both symbolic and a sub-symbolic structures; the symbolic one is a production system that matches the state of the system to previously learnt symbols. The sub-symbolic structure is a set of parallel processes that control many of the symbolic elements through a series of equations, and in many cases in accordance with utility functions. (ACT- R Research Group Department of Psychology, Carnegie Mellon University, 2012)

ACT-R shows many of the advantages of the hybrid models, and in fields as applied psychology it has grown interest on more integrated cognitive architectures. However, it still exhibits deficiencies typical to these architectures; for instance and maybe the most important one, most of the knowledge acquired depend completely on the programmer and not on learning from the environment. (Troy D, 2003)

Another known cognitive architecture that uses a hybrid approaches is the LIDA architecture. LIDA uses both symbolic and connectionist approaches merged together. The architecture is based on a cognitive cycle that goes from perception to action.

During this cognitive cycle several aspects are taken into account but always with special emphasis on the roles of feeling and emotions. Emotions are used for conceptualization and are related by associative relations; they guide actions and what is called consciousness in the model, which affects decision making at every level. (Ramamurthy, Baars, D'Mello, Franklin, 2006)

As these two examples, many others have also shown that in general, merging approaches is a feasible way to improve performance, and that keep on generating new points of view on the overall problem solving goal. However, most of the approaches have been based on a cognitivist perspective but it

does not mean that it should be the only one, it could be reasonable to evaluate others or even hybrid approaches at that level; An example could be to focus on more behavioral models aside of the cognitive perspectives or as their complement, in a similar way as they have been opponents and complementary approaches in psychology.

Similarly, a very relevant approach to the goal on general problem solving that leaves aside the constraints of architectures aiming for specific tasks is the developmental or epigenetic robotics described below.

Epigenetic robotics

The goal of epigenetic robotics, also known as developmental robotics, is to model the development of cognition through the usage of elements from different sciences and approaches, such as robotics, neurophysiology, psychology and artificial intelligence, where the results may be a beneficial exchange among all of them (Metta. Giorgio and Berthouze. Luc, 2005).

This is carried out by the study of the development as a process in which modifications on cognitive structures lead to an overall emergence of abilities, which in human basically happens form the embryo to the fully developed adult. Here, development is seen as an open-ended adaptation process generated by means of interaction with the environment (Metta. Giorgio and Berthouze. Luc, 2005).

The whole idea emerges from the need across the cognitive sciences for models that can scale up beyond specific domains and scenarios, and that at the same time, can display a developmental trajectory and are transparent in their construction and concepts (Anthony F. Morse, Joachim de Greeff, Tony Belpeame, and Angelo Cangelosi,. 2010).

Evidently pre-programing for specific behaviors cannot give solutions to scalability problems as the systems are expected to work in too complex and unpredictable environments; that as the limitation set by constraints and assumptions made by the programmer usually fail when the systems are faced to real problems. Therefore the best is to make the systems in charge of their development by giving them the ability of verifying their own learning and the possibility of growing their cognitive structures freely towards broader goals (Stoytchev. Alexander, 2009)

However, as the systems are expected to develop by means of interacting with the environment, a clear limitation and crucial aspect in the design is the actual body of the robot in the sense that it will constraint any interaction and therefore the whole process. Body and brain cannot be separated, and at the end is the body what shapes the brain (Asada, Hosoda, Yasuo, Hiroshi, Toshio, Yoshikawa, Ogino and Yoshida, 2009), reason why in epigenetic robotics the design of the body, in terms of sensors and actuators, plays a very important role in the abilities that can be achieved and therefore great deal of the research is focused on this fact.

Epigenetic robotics is a relatively new field, but it has shown interesting results as many limitation and constraints of other approaches are overcame by means of a more freely development. Nevertheless, the main fuscous on research has been set on sensory-motor development leaving aside higher functions as the ones cognitive models and others pretend, such as inductive and deductive learning or concept formation and manipulation.

Epigenetic robotics is in fact a source of inspiration for the model related to this thesis and presented below as even when the ideas are mainly focused on sensory-motor approaches, they seem promising for applications at higher functions as the ones aimed here.

Transparent Neural Networks

The Transparent Neural Networks (TNN) model proposed by Claes Strannegård (Strannegård 2012) is being developed since 2011 at Chalmers University of Technology and the University of Gothenburg. Until the writing of the present thesis it is a theoretical model which has been presented in different conferences at the mentioned institutions as well as at the Lund University and the SewCog.

The TNN project attempts to develop a model with problem solving abilities achieved by means of transparent structures, meaning that they are as clear as possible for the user at any time. Thus the goal is to achieve transparency not only when designing a solution to a given problem but especially when the system has performed any kind of learning.

As described before, when it comes to modeling traditionally the symbolic approaches are mainly concerned with deductive reasoning, whereas emergenist are largely focused on inductive learning (d'Avila Garcez and Lamb, 2011). Therefore, one of the major goals of TNN is to include both deductive and inductive reasoning as simultaneous capabilities of the same model.

Of course there have been many different attempts to achieve that with hybrid architectures; nevertheless the fundamentals of TNN differ in the stress on the need for models that remain transparent while achieving the two kinds of reasoning by means of just one process.

It has been common that when merging approaches the architectures tend to have different structures for symbolic and sub-symbolic processing that are connected but still independent. In the case of TNN the proposal is to achieve both, the deductive and inductive capabilities, by means of a single structure and a single learning algorithm.

As mentioned the main goal with the structure proposed in TNN is to keep the transparency, also called interpretability, which refers to a model being easily understood or interpreted by its users. This fact is stress as the problem of lacking transparency is an issue that affects many models and especially those

based on connectionist approaches, which leads to great problems when interpreting and grasping the underlying process of a structure even if it solves a particular problem.

An example of that are the feed-forward artificial neural networks; in these structures there is not much transparency since it is not trivial to give a meaning to the values the weights reach after training, and actually the meaning of the activity in a particular neuron (apart of those in the inputs or outputs) is not clear for the programmer.

Transparency is a desirable characteristic for any model as it makes it easy to explain, maintain, modify and verify. Thus, this is why the TNN attempts to maintain the transparency as a crucial issue in all the building element and learning rules in the model. This is, every element in the network must have a meaning or it may be easily inferred by the programmer; equally, any parameter that is modified by learning is to represent a simple and easy to comprehend relation among elements.

An introductory description of TNN

The TNNs are networks constructed in stages by means of a small set of construction rules. The construction rules are related to the addition of nodes and connections; each node is to represent a clear function and its connections are to be easy to understand relations, this way a compositional semantics in the networks is to be ensured.

It is important to make clear that even when the ideas behind TNN are related to cognitive modeling and are partially inspired by biology, the aim is not to model any real neural system but instead the only concern of TNN is problem solving.

Again, the main goal of the model is to achieve a transparent model capable of both inductive and deductive reasoning. This transparency is to be achieved by the limitation in the construction rules that ensure the interpretability of every element. Therefore the basic elements are to represent clear concepts, and their association to others must be clear relationships.

The most basic elements of the networks are the nodes, which in the model are to represent concepts learnt by experience. This way each node in the network is a concept and is related to other by means of connections called edges.

The relations between concepts by means of the edges and the information spread through them, further called activity, allow the formation of conceptual relationships that emerge contextual meaning for each node. This permits that concepts with partial information are retrieved, or that inferences of concepts contextually connected are made even when the explicit information that elicits them is not in a given input.

To illustrate this imagine a concept representing a physical object and therefore its activation is elicited by sensing the physical characteristics of the real object

when presented as an input; However, this concept could also be related to a concept that represents the name of the object in the form of a word, which activity is elicited by the sound that corresponds to the word presented as an input. Then, even when the physical characteristics of the object are not present in the input, activity in the concept representing it may be elicited if the concept representing the word is active. That means that contextual relationships and inferences are being carried out all the time.

On the other hand, to achieve both inductive and deductive reasoning, two kinds of activity are used. One of the activities in the network is called the real activity. It is used to generate associations that may represent temporal relationships or specific concepts. It also allows achieving inductive learning through the creation of deeper associations from more basic concepts scaling up the abstraction of the concepts at each level of association.

The other kind of activity is the imaginary activity; this takes as basis the real activity and performs inductive reasoning by means of inferring causality, predicting activity in future, or deducing previous activity that could have led to the present state of the network. At the same time this activity is capable of inferring missing information in an input or deducing possible relations by using existing associations.

When it comes to the construction rules they can be related to the way the nodes are added, which can be manually done by the user, or by an automatic addition partially assisted by the user taking into account information states in the network.

The way the activities are spread through the network as well as the construction rules and the characteristics in the elements of the network are detailed in the following section.

The implementation

Given the general concepts of what TNNs are and the reasons behind them, the main goals of the present thesis is to implement a toolbox that allows its users to experiment with this kind of networks and draw conclusions about their behavior and utility.

Bearing that in mind, and the fact that at the moment of the implementation the TNN research project is just starting, the main value of this thesis becomes to generate feedback for further development on the ground concepts of the TNN.

The implementation was based on unpublished manuscripts that contained the general ideas and concepts of the TNN model, which were evolving as the work was carried out. Therefore to design the toolbox was needed to face a lack of specifications and take only as a major objective the concepts on modularity and transparency. That implied to design a tool that could be flexible enough to fit the constant changes in the theory while producing useful and fast feedback.

The focus on modularity in the design allowed making the development flexible while fulfilling the TNN's main character of being built with elements and interactions as understandable as possible.

At the same time, other important factor that led the design was to make interaction and building as easy and accessible as possible. Thus there was a special emphasis on creating a friendly and simple to use interface that gives enough information to the user and at the same time allows creating, exploring, and evaluating TNNs fast and easily.

As highlighted before, the concept is still being developed and so are the algorithms and implementation details; then all the results reported here are the outcomes of an iterative process that led both the concepts and the implementation. So, and as the research is to keep on advancing, the following description focuses in the usage and the concepts included as well as in technical details that are considered necessary for further development of the tool.

The building blocks of a TNN

The networks are built using basic elements that are related to each other and possess specific information that allows the network to work. These elements or building blocks in TNN are denominated Nodes and Edges.

Nodes

The TNNs, as implied in the name, are the interconnection of a given number of elements that share information. Thus it could be seen as a directed graph, but in this case it deals with two kinds of information that flow in opposite directions as will be explained later.

These interconnected elements in the network are called nodes; they are individual processing units that can be selected and added to the network either manually by the user, or automatically by the tool when it is specified to do so.

All the nodes in the network represent simple concepts that can be labeled by the user in order to keep the transparency. In a sense of conceptual learning every node represents a concept that basically comes from the association of either previously existing concepts, or sensor nodes. The sensor nodes are the inputs to the network and are how the network is fed and receives information from environment.

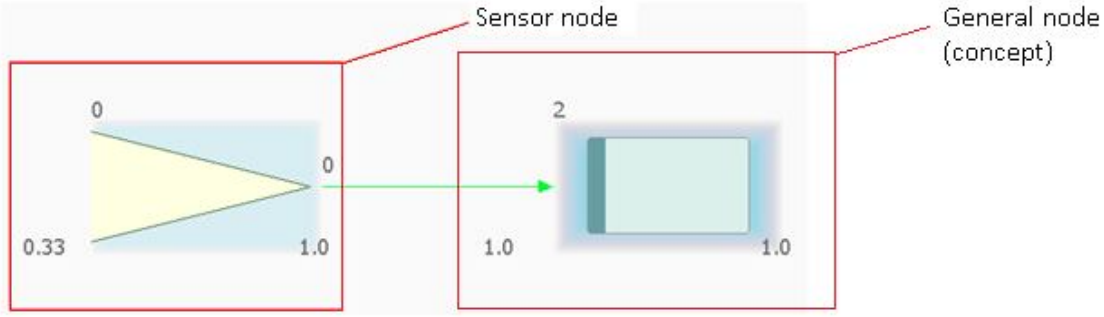


Figure 1 The two main kinds of nodes, sensor (left) and general (right), connected by an edge from the sensor to the general one.

At any time every node has twos activity parameters called real and imaginary states. These states may vary from zero to one where zero means no activity and one means full activity. The way the real state is calculated depend on the kind of node being used, each of them has a specific activation function that will always depend on the activity coming from other nodes or inputs. The only kind of nodes that changes their activity based on the inputs is the sensor node; they simply copy the environment that is, generally speaking, the input given by the user.

All the implemented types of node in relation to their activation function will be described in the Activities section.

The expression for the real activity of a given node k over time is described by:

$$r_k(t) = \mathrm{A}(\mathbf{I}(t))$$

*where A is the activation function of node k and **I** is the vector of size n containing the n inputs to the node k.*

Imaginary activity of node k over time:

$$i_k(t) = \mathrm{P}(\mathbf{PI}(t))$$

*where P is the prediction function and **PI**(t) is the prediction input vector calculated from the states of the nodes at the outputs of k at time t.*

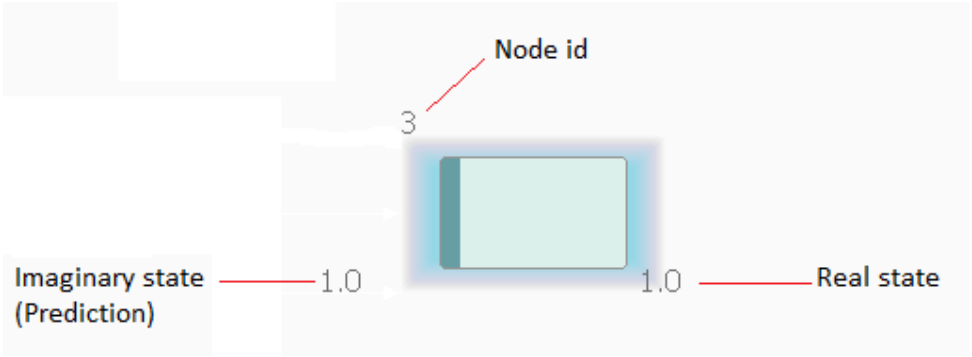


Figure 2 Description of the parameters appearing in a general node; the node id, the imaginary state and the real state.

Edges

All nodes that are not sensors must always be connected to at least other node and this connection is represented by edges.

The Edges possess different characteristics, firstly they may belong to two principal kinds depending on their direction; they can be either those that are going out of the node, or those that go into it. However, they are added by pairs, so for every node going out of a node there is one going into another, but graphically they are represented by the same connection. This is done in order to treat separately the two kinds of activity that are there in the network.

All edges have a weight that can change over time but whether they change or not and the meaning they have depend on the kind of network being used.

The Edges also have an activity reverberation, which means that after some activity is transmitted from one node to another the activity in the edge connecting them does not disappear immediately; instead it fades down slowly according to a parameter learnt by experience called the *decay parameter*.

The reverberation activity of edge i at time t is denoted by $b_i(t)$.
Equally, a vector of reverberation activities is denoted by $\boldsymbol{B}(t)$.

Levels

As edges possess directions the way the network grows and propagates information is affected by this fact and gives rise to the concept of level. The levels work as a hierarchy, meaning that every node correspond to a higher level than all the nodes it receives information from. This can be seen as levels of abstraction since the higher the level is, the more concept have to be active and associated. The levels are labeled with increasing numbers starting at 0 which correspond to the sensors level, and up to the highest level where nodes have no outputs.

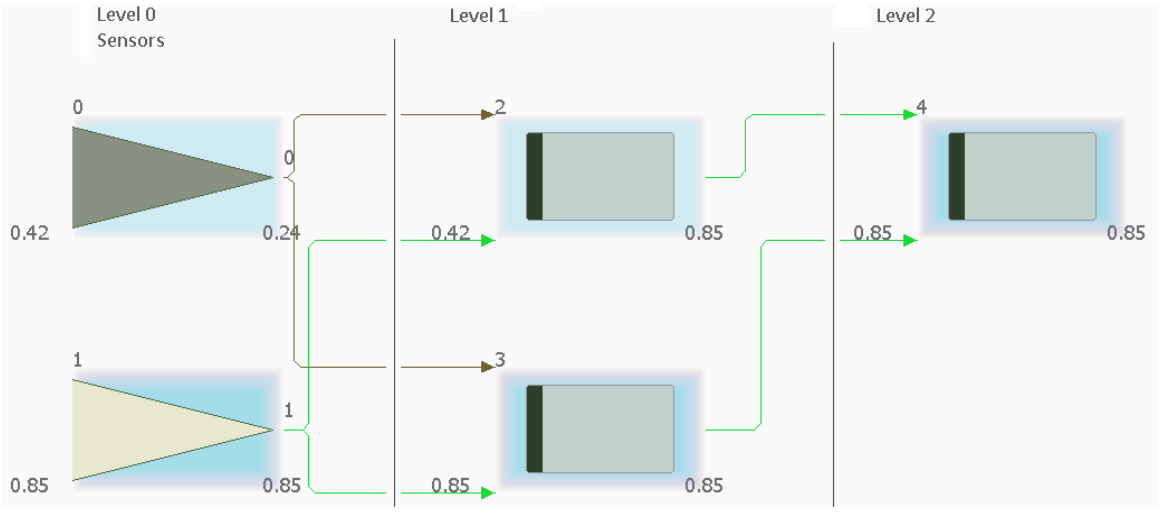


Figure 3 Example of how the nodes are shown by levels in the toolbox, and how to interpret them.

Inhibitory edges
This kind of edge inhibits the activity of a node depending on the state of a set of other nodes.

The inhibition activity over a node will corresponds to the addition of activities in the nodes inhibiting it; that addition is truncated at a maximum of 1 ensuring the inhibition over a node fits in the interval [0, 1].

The inhibition is performed after the activation has been calculated by multiplying the complement of the inhibitory addition.

> *When inhibition is applied the real activity of node k over time is modified by:*
>
> $$r_k(t) = \mathrm{A}\big(\mathbf{I}(t)\big)\big(1 - h(t)\big)$$
>
> *where $h(t)$ is the sum over all the inhibitory inputs of k at time t.*

In the interface this kind of edges is depicted as orange connection between nodes.

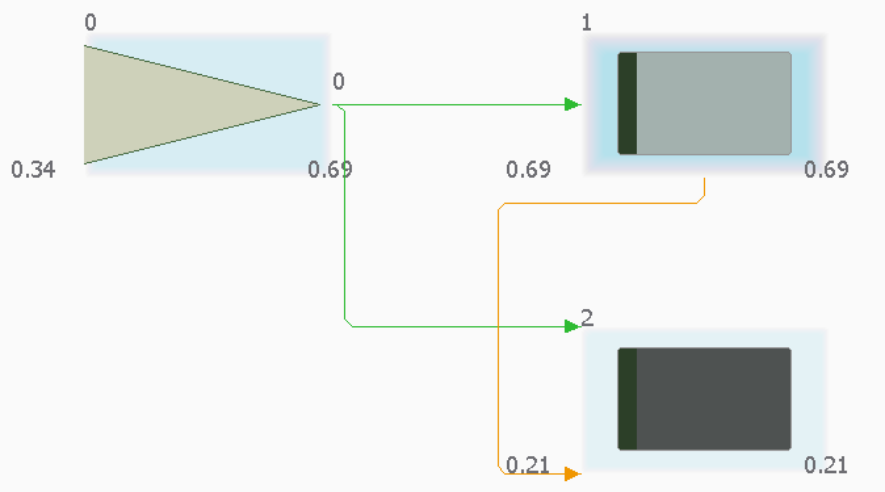


Figure 4 Depicts an inhibitory connection from the node with id 1 to the one with id 2.

Activities

As mentioned before, there are two kinds of activity that propagates in opposite directions. The main activity is called the real activity and is the one that propagates from the sensors forward till the deepest level. When propagating the real activity the activation function of all the nodes at a certain level are evaluated taking into account the activity from the nodes in the previous levels, so the activity is propagated level by level. The second kind of activity is the imaginary activity, which propagates backwards in the opposite direction than the real activity. That means that the imaginary activity starts at the deepest level and propagates back until the first one. However the imaginary activity at the deepest level, or actually at every node that has no outputs, will copy its real activity as the imaginary activity.

Each of the activities has different meaning. On one hand, the real activity is the natural response of the network to a given input, and it is also the activity taken into account for the creation of new associations or concepts. When evaluating the meaning of the real activity for each given node the amount of activity it has is related to how much of the concepts it associates were present in the inputs; that relation depends on the activation function of each node and the interpretation may vary somehow based on that function.

The activation functions depend on the goal of the node and the way the information is fed to it; the information can be the real activity of the nodes that are feeding it or the activity reverberation in its incoming edges (reverberation activity).

The types of nodes implemented regarding their real activation function is listed here:

- ***Min Nodes:*** Nodes which activity is set to the minimum at its inputs.

$$A(\boldsymbol{I}(t)) = min\{I_i(t) : i = 1, \dots, n\}$$

- ***Max Nodes:*** Nodes which activity is set to the maximum at its inputs.
-

$$A(\boldsymbol{I}(t)) = max\{I_i(t) : i = 1, \dots, n\}$$

- ***Average Nodes:*** Nodes which activity is set to the average of its inputs.

$$A(\boldsymbol{I}(t)) = \frac{\sum_{i=1}^{n} I_i(t)}{n}$$

- ***Delay Nodes***: Have only one input and set their state as the real activity at its input in the previous time step.

$$r_k(t) = \mathbf{I}(t-1);$$ *Size of vector **I** is always 1.*

- ***Buffer Nodes:*** Have only one input and copies the same state that the real activity at its input; used to bring the same activity to a deeper level.

$$r_k(t) = \mathbf{I}(t);$$ *Size of vector **I** is always 1.*

- ***Association node:*** Average of the real reverberation activity at their incoming edges.

$$A(\boldsymbol{I}(t)) = \frac{\sum_{i=1}^{n} B_i(t)}{n}$$

*where **B**(t) is the reverberation vector in the inputs of node k at time t.*

- ***Simple Gaussian node:*** Have only one input and learns by experience the average and the standard deviation of the inputs shown. The real activity is calculated by means of the parameters learnt using a bell-shaped function.

$$\mathrm{A}(\mathbf{I}(t)) = e^{-\frac{(I-\mu)^2}{2\sigma^2}}$$

where μ is the mean learnt and σ is the standard deviation.

- ***Complete Gaussian Node:*** Is a compilation n of function like the one described for the simple Gaussian node, where n is the number of inputs of the node and individual parameters are learnt for each of them. The final result is the multiplication of all of these functions.

$$\mathrm{A}(\mathbf{I}(t)) = \prod_{i=1}^{n} e^{-\frac{(I_i-\mu_i)^2}{2\sigma_i^2}}$$

- ***Sensor:*** Nodes that set their activity from the input given by the user (environment).

The second kind of activity is the imaginary; this is meant to infer or complete information from the one present at a specific moment in the input; however, imaginary activity can also be a prediction of information over time as a relation to expected concepts or inputs in both the past and the future.

The imaginary state of a node will depend on the state of those that are fed by its real activity. Nonetheless, when a node does not feed any other, or has no outputs, then it will copy its real activity as imaginary activity in order to use it as the source for inference.

To calculate the imaginary activity of a node weights at its outgoing connections are to be learnt. After the proper learning the value of these weights corresponds to the probability of the node being active, when the node at that output is active.

Every node that has connections going outwards adapts a weight for each of those edges. Then when the imaginary activity is being propagated, the imaginary activity is set to the maximum value of all the imaginary activities of the nodes at its outputs multiplied by the respective weights.

The imaginary activity is calculated by:

$$i_k(t) = \mathrm{P}\big(\mathbf{PI}(t)\big) = max\big\{WI_{gk}(t)i_g(t) : g = 1, \dots, n\big\}$$

where $WI_{gk}(t)$ *is the weight from the node* g *which is at an output of node* k, *and* $i_g(t)$ *is the imaginary activity of node* g.

In the example on Figure 5 the node at the deepest level copies its real activity as its imaginary one, but meanwhile, the nodes on the previous level have slightly different weights and therefore different imaginary activity that depends on the node at the deepest level. The sensors have weight close to 1.0, and as described they take for imaginary activity the maximum of the possible activities coming from nodes at their outputs.

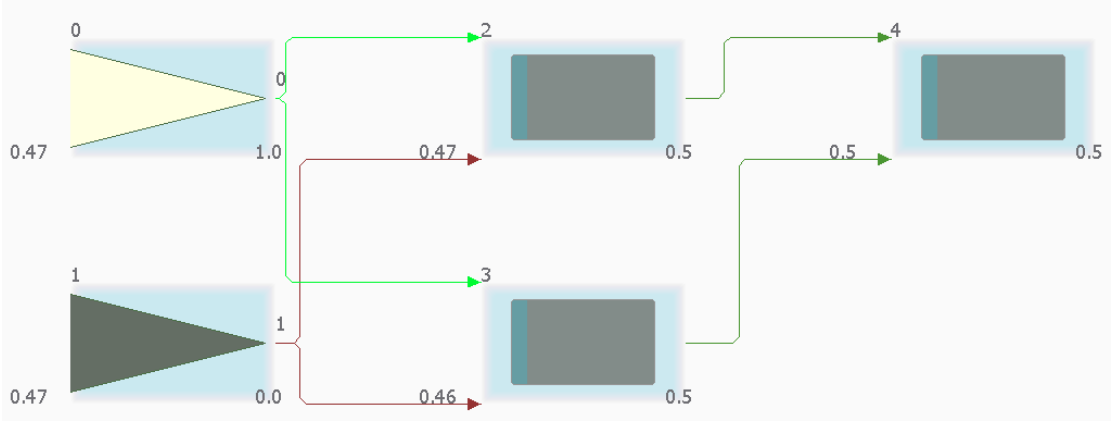


Figure 5 A simple network at a given state to depict the real and imaginary propagation. The node at the deepest level copies its real activity as imaginary, while in the others the imaginary activities are affected by the whights.

Time step

Updating the states of the network means to take an input and calculate the real and imaginary activities of all the nodes in the network. The update of the network is carried out every time a new input is presented; this is defined as a time step.

In this sense, there is no delay in between the moment the input appears and the moment the activity of all the nodes are updated. That means as well that if an input is to last longer than other, it simply has to be repeated several times in the input stream. In other words, if an input is repeated in several time steps this will keep the network in the same state after it is stable.

Networks construction modes

To build networks different construction rules can be used, but which and how are used depend on the construction mode selected. The two modes existing in the current implementation are the Manual and the Interactive modes.

Manual mode

The most basic construction mode that can be used to create networks is the manual one. The construction of a network in this mode depends completely on the user. The architecture of the networks does not change while they are being used but only as the user decides to add or remove elements.

In this mode the user chooses to add any kind of node by connecting them through edges from whatever node that already exists, unless the node added is an input. This allows a complete and easy understanding of the network, though at the same time its usage is limited to a rigid architecture. It works to evaluate and visualize how the activities spread through a network's architecture, but no automatic addition of elements is performed.

This mode is used mainly to propose anatomies and check their performance. Building a solution might require a complete understanding of the problem, reason why is not suitable for this purpose, but instead it is a good way of getting to visualize the problems and possible behaviors. Exploring the problem in this mode can help to find a reasonable starting point for the further growth based on an interactive construction.

Interactive mode

In contrast to the Manual mode in the interactive mode the network can be modified automatically by adding new nodes and connections depending on the need. However, it is not completely automatic since is the user that controls when the network should look for new associations.

The user has the ability to set the network in a recording mode and stop it when needed. In this way the network will look for associations presented in between the time the recording signal is active, but this search in only performed in the time step at which the recording signal stops.

Associations in the interactive mode

To better understand the idea with the interactive mode the definition of association has to be enhanced. An association, in the sense used in this particular mode, is the formation of a node that represents a relationship between the activities of two or more nodes limited by a maximum that can be set.

These relationships may represent a simultaneous activation of nodes, or a temporal relation among them, though in general is the same behavior.
The temporal relations refer to the situations at which one node or a group of nodes get active or increase their activity after other has done the same. This may include many steps and relate many nodes.

As mentioned the simultaneous activation of a node is just a particular case of the temporal relationships at which all the activations are presented in the same time step. In this kind of associations the order does matter; for example, given two nodes *a* and *b* that belong to the same level may have two possible temporal associations; this is, If node *b* gets activated after node *a* got activated, it is a different association than if *a* gets activated after *b*.

On the other hand, differences in time are not considered as different associations. If the order is the same, that means that if *b* gets activated one time step after *a*, it will be considered as the same association than *b* getting activated two or more time steps after *a* did so.

The activation of an association must represent how much of the actual relationship is achieved, which implies that in the case of temporal relationships the activity must relate different time steps. To achieve that, the activity reverberation of the edges going to the association, and in particular their decay parameters, are used to enclose temporal information.

The activity reverberation of an edge copies the real states of the node it comes from and decays depending on the decay parameter; this parameter is to be learnt by experience and has to do with how many time steps the whole association takes to be complete after the node sending information through the edge was first activated.

The activity reverberation depends on the decay parameter in the following way:

$$b_i(t) = \frac{b_i(t-1)^3}{1 + e^{-\sqrt{b_i(t-1)R_{ik}(t)}}}$$

where $R_{ik}(t)$is the decay parameter of the edge.

To depict the behavior of the activity reverberation in relation to time as the parameter $R_{ik}(t)$ changes is depicted in the figure 6.

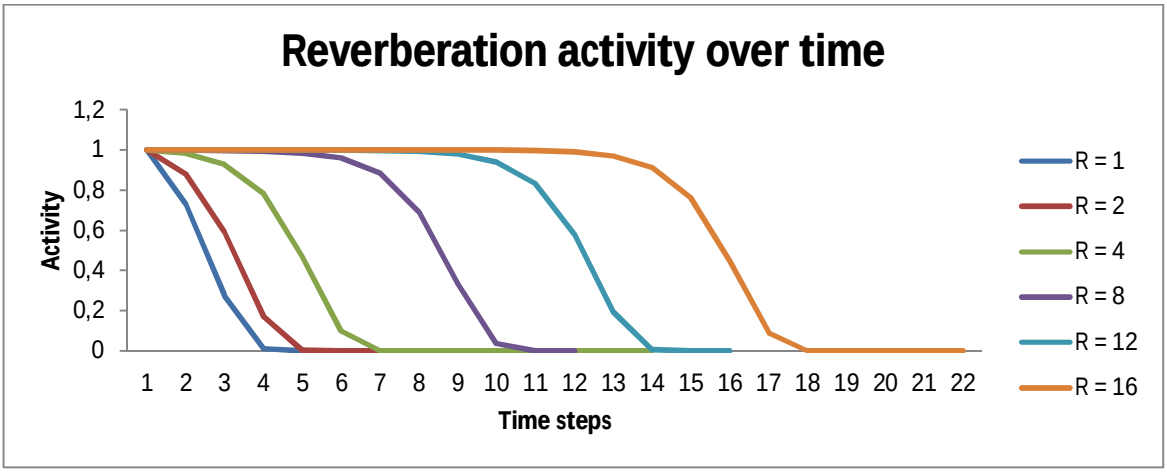


Figure 6 Reverberation activity over time for different values of the reverberation parameter $R_{ik}(t)$, denoted R.

To illustrate the temporal relationships a simple example of a sequence is shown in the figure 7. First, a sequence of three consecutive inputs is shown in the first three time steps, which happens while the recording signal is active. After, at the fourth time step, the recording signal is deactivated, thus the sequence is considered to be over and the association is formed as a new node.

This new node has three inputs and each one of the corresponding edges possess different reverberation decay parameter; the first has a parameter 3 as it takes three time steps from the activation of the corresponding node until the whole sequence is over. The same way, the second edge will have a decay parameter 2, and the last one 1.

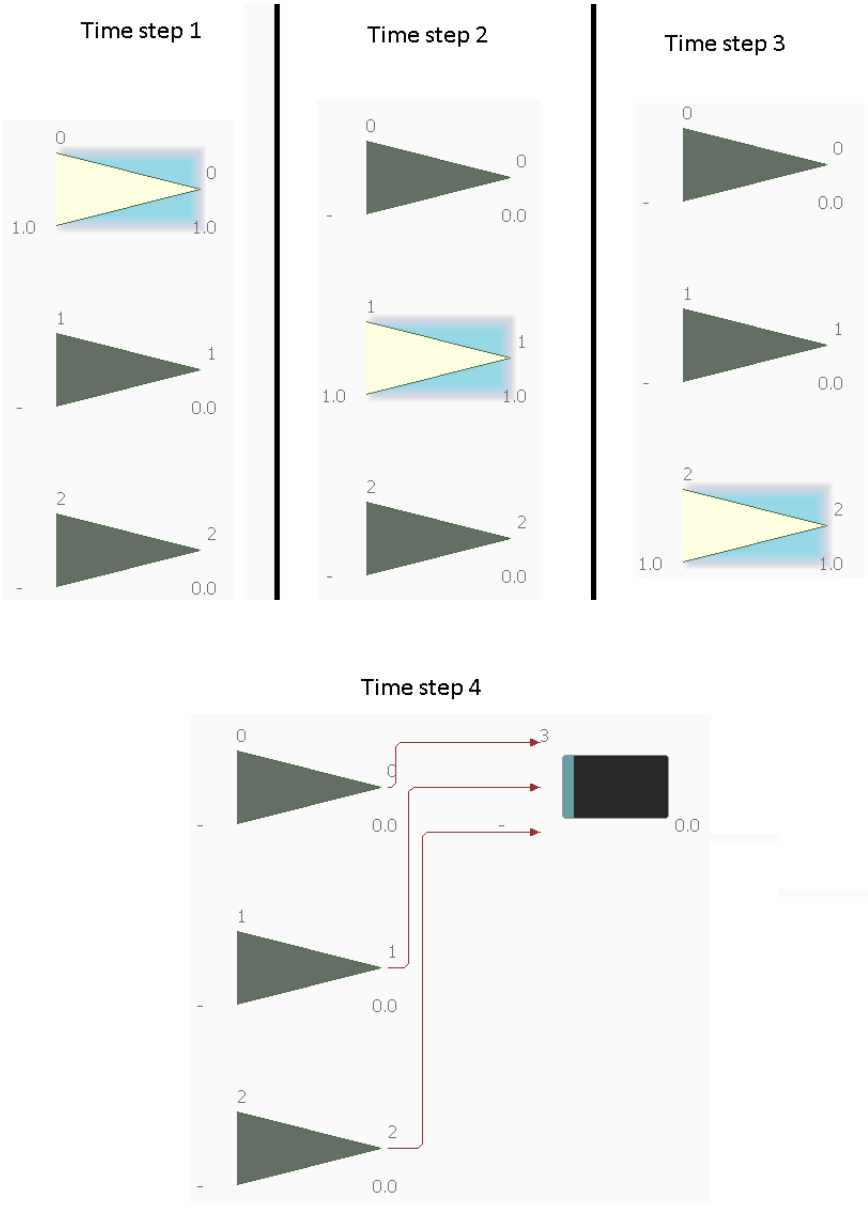


Figure 7 Example depicting the learning of a simple sequence in four timesteps, by means of an association node.

The parameters of each edge can change over time through experience, tending to be the average time that it takes for the sequence, or association, to finish since the first activation of the edge.

Gaussian growth and generalization

Other construction rule that can be used in the interactive mode is known as the Gaussian growth. This construction rules creates nodes of the class *Complete Gaussian Node* which learning allows them to build concepts by generalizing characteristics of a set of inputs.

The main idea of this construction rule is to create a new Gaussian node when a given input is far from the characteristics learnt by the existing ones. Therefore, a new class will be created from a set of sensors by means of new a new node.

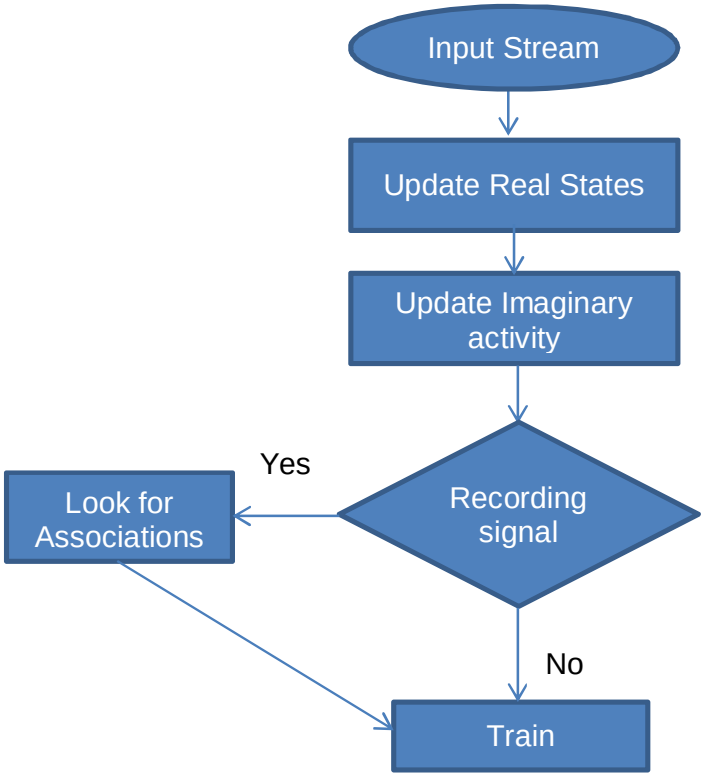
To set the sensors to be grouped in these classes, they must be selected by manually adding a *Complete Gaussian Node* fed by those sensors as the seed of the whole learning. Therefore, the Gaussian growth is only performed in the level 1 of the network fed directly by sensors, and consists on creating a new node if the ones existing do not fit the current input by the following measure.

$$D(t) = \sum_{i=1}^{n} \frac{|I_i - \mu_i|(1-\sigma_i)}{\sum_{i=1}^{n}(1-\sigma_i)\sum_{i=1}^{n}\sigma_i} > 1$$

Network working cycle

Updating the network is the process carried out at every time step. The following description depicts the implementation in General networks for both the manual and the interactive modes.

The overall process of updating is performed after reading the corresponding input array and copying it to the sensors as real activity, then the process of updating starts. First the update of the real state is performed by propagating the real activity forward. The second step is to update the imaginary activity going backwards. Once both the activities have been propagated the recording signal is checked in order to decide whether to look for associations or not, and in any case the last step is always to perform training.



-*Input format*: The input for each time step must be an array of size *n* + 1 where *n* is the number of sensors the network has. The first element of the array must be the recording signal which is to be different to zero only if the network is expected to create associations. The rest of the inputs correspond to the value of the input sent to the sensors. In the network each sensor has an Id which corresponds to the order in which they were added; these ids are the order used to update them from the input array. If the size of the input is shorter than *n*+1 the inputs given will be used to update sensors from the first id until the end of the array.

Figure 8 Input format.

Update Real states
When updating the real states all the levels are checked starting at level 1, since level 0 or sensors level is first updated by copying the values from the input. From then on the nodes are updated by using their corresponding updating function r(t) level by level.

After a whole level has been updated the inhibitions of the nodes are carried out. This is done at each level in order to avoid the propagation of activity of nodes that are to be inhibited.

Update imaginary states (Predict)
Once the real activity in the nodes has been set, the imaginary activity is propagated backwards from the last level to the sensors. The calculation is done as was described in the imaginary activities section.

Look for associations
When the *recording signal* is being used, this function performs the search for new possible associations when certain behavior in the signal is met.

-The recording signal:
The recording signal must be used to set the network in the interactive mode, meaning that associations are to be searched and automatically created when the user decides to. The recording is done for time intervals that are specified by this signal. The interval starts at the time step in which the signal changes from 0 to any higher value in the interval (0, 1], and it finishes when the signal goes back to 0.

In the interactive mode this input is referred to as the recording signal; however, in the implementation and the interface of the toolbox this parameter is known as the emotional impact of the input. In other words, the recording is performed while there is a positive emotional impact in the input.

To create associations the process is as follows: The search is performed after an interval of time steps has elapsed. During the interval an attribute of each node called *reverberation* of the node, is set to the maximum real state the node reaches within the interval. Similarly, at the time step at which the real state is found to be lower than the *reverberation* the attribute called count, starts to keep track of how many steps pass from that event until the end of the interval.

Once the interval is finished the associations search starts; it begins at the deepest level going backwards to the sensors level, but it stops wherever an association is created. The search is carried out basically by grouping all the nodes that have first, a reverberation value at the previous time step higher than

an *association threshold* given by the user; and secondly, if the predicted stated is lower than the last reverberation or the node has no outputs.

In principle the group can be of any size, but only a maximum number is associated depending on a parameter of maximum association size that is determined also by the user. The nodes of this final group are organized by the *count* parameter and so are added to a new association node. This allows differentiating associations including the same nodes but different time order.

In case it is found that an association already exists, an update is performed on the reverberation parameters of the incoming edges in the node representing that association. This update is performed taking into account the *count* parameters of the nodes feeding the association tending to the mean of all the examples seen.

Train

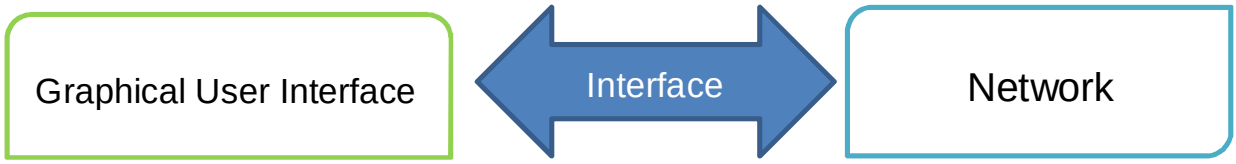
When training the network two processes can be performed, the principal one has to do with the weights training, whereas the other is carried out at each node and is related to the update of certain parameters.

In the manual mode no training is performed on the weights; meanwhile, in the interactive modes the only weights changes are those related to the imaginary activity. These are trained to represent a probabilistic relation between the activation of the nodes linked by the edge. The idea is to get a parameter that encloses how probable is that a node *a* feeding node *b*, was active during the recording interval if *b* had certain activity during the same interval.

The other kind of training occur for some nodes that have parameters to be adjusted to the inputs, the Gaussian nodes are the only ones that train parameters as they adjust their mean and standard deviation values at each time step, unless a Gaussian growth is performed.

General description of the implementation

The implementation of the toolbox is divided in three main block as depicted in the diagram below. The principal block is the network in which the functionality and algorithms as well as the structure and management of the networks are carried out. On the other end of the diagram is found the Graphical user interface which deals with all the graphics generation and interaction with the user as well as the information flow between the user and the toolbox. And finally to manage the link between these two main blocks there is an interface that deals with the communication and information flow between the network and the interface.



As the focus of this work is on the design of the network the description below is focused only on the main block Network; therefore the descriptions regarding the implementation for the Graphics and the Interface blocks are not included.

The whole system is created under an object oriented paradigm, and for the Network block basically there are three principal classes, the Networks, the Nodes, and the Edges. The Node and the Network classes are abstract classes, and the different type of nodes and networks are classes that extend the main ones implementing the abstract methods that differentiate them.

These classes are created abstract in order to allow future implementation of new kinds of nodes or networks. However, in the current description the only network used is the so called General Network, which is designed to work in both interactive and manual modes.

The figure 9 depicts a general class diagram where the main relationships and inheritances are shown. There appear all the node kinds that are available as classes that extend the abstract class Node. There are also special relationships for the nodes of classes Input, Association and Complete Gaussian.

There are special relationships because, firstly the Association and Complete Gaussian are the only kind of nodes that are added automatically, reason why there are special processes to check for the need of new nodes and the functions to create them.

Similarly, the Input nodes (sensors) need to be tracked in order to update the network correctly since these nodes are updated in a different way than the rest. There is also a need to track them in order to create groups and handle them, which makes easier to add and remove sets of sensors easily.

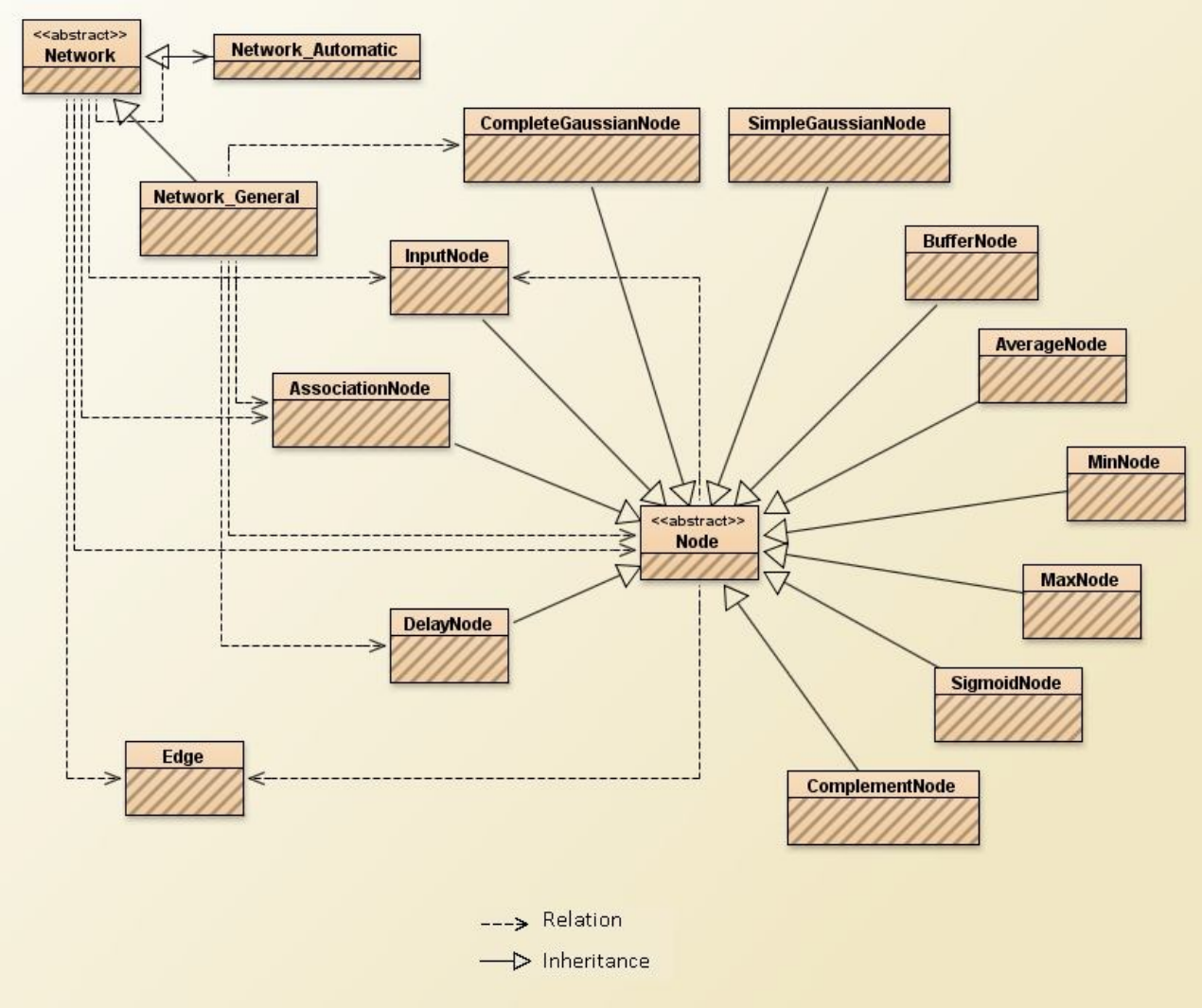


Figure 9 Classes diagram of the networks and nodes classes.

Main classes description

The network is built as a collection of nodes of different kinds with individual characteristics, behaviors and even some special functions to be handled. However, all the nodes possess a similar basic structure to which particular features are added, and therefore all of them belong to the same basic class.

The class Node

The abstract class Node has methods and attributes that are shared by all the nodes, as well as some abstract methods to be implemented in any case when creating new types. Below are listed the shared attributes and the abstract methods to be implemented for new types.

Attributes:

- ***Id:***
  It is a unique number that identifies the node and also corresponds to the index in the principal list of nodes in the network.

- ***state:***
  It is a double field that contains the real state of the node. It is updated at every time step.

- ***lastState*:**
  Retains the real state at the previous time step and is updated whenever a value is set for the atribute state.

- ***predictedState:***
  Is a double field that contains the imaginary state of the node also called predicted state. It is updated at every time step by the predict function.

- ***lastPredictedState:***
  Retains the imaginary state at the previous time step and is updated when a value for predictedState is set.

- ***reverberation:***
  It is a parameter used for creation of new association, it copies the real state as long as it's value is higher than the one stored in reverberation and the recording signal (emotional Impact) is active. Once the recording signal gets to 0, the reverberation is set to 0 again.

- ***lastReverberation:***
  Retains the value of the previous value of reverberation. It is updated each time a value for reverberation is set.

- ***count:***
  Keeps track of the number of time steps the reverberation stays in an active state (value higher than zero). It starts to count only after the real state is set to a lower value than the one stored in the reverberation. It is set to zero when reverberation changes from zero to any other value.

- ***incomingEdges:***
  It is a list of objects of the class Edge that represent the edges going to the node.

- ***outgoingEdges:***
  Is a list of objects of the class Edge that represent the edges going out of the node.

- ***depth:***
  It represents the level in the network at which the node is.

- ***lastInput:***
  It is an array containing the values received as input in the last time

step.

- ***lastPredictionInput:***
  It is an array containing the values received as input for updating the imaginary state in the last time step.

Abstract methods:

- ***double updateStateFunction (double[] statesOfInputNodes):***
  This function takes as parameter an array which must contain the information from the nodes at the inputs of the given one, that are to be used to calculate the state. The order of the states in the array is assumed to have the same order that the nodes have in the incomingEdges list. The function must perform the calculation corresponding to the node type and return the result in the interval [0, 1].

- ***void trainFunction (double relevance) :***
  This is used if the node needs to update any parameter after the update is performed at the end of every time step. It receives as parameter a number in the interval [0, 1] that may be used to indicate the relevance of the example being trained in case of being necessary. In the General Network the only kinds of nodes that train values through this function are the **simple Gaussian and the complete Gaussian nodes.**

- ***boolean addIncomingEdge(int origin, double weight, boolean inhibitory):***
  This function must return a Boolean that indicates whether the edge was added or not. There is a default function called *createIncomingEdge()* that receives the same parameters and returns a Boolean. The implementation of *addIncomingEdge()* can be just a call to *createIncomingEdge();* however, it is left abstract in order to allow certain rules for each node; for example, at a buffer node it is not possible to add more than one edge, then this function is used to add that rule; however to add the corresponding object Edge, *createIncomingEdge()* must always be used.

- ***restartFunction():***
  When a node is restarted it goes back to its initial state undoing any kind of learning and resetting default values. When a node is restarted all the edges and principal attributes are restarted; however, if more parameters are included in certain kinds of nodes this function should include the restart procedure of those parameters if needed.

- ***String getInfo():***
  It returns a string where some information about the node can be added. This info is what will be displayed in the interface when checking the information of the node. It has no relation to the actual functioning of the network.

The class Network

The same way as in Node, the abstract class Network has methods and attributes that are shared by any possible kind of network. It also has some abstract methods to be implemented when creating new types.

Attributes:

- ***Nodes:***
  It is a list of objects of the class Node where the index of each element corresponds to the id of the corresponding node.

- ***Levels:***
  It is a list containing lists of nodes. Each list of nodes is a level, and points to the nodes corresponding to that level. The level id corresponds to the index in the main list.

- ***inputNodes:***
  It is a list containing the ids of all the input nodes.

- ***inputGroups:***
  Is a list containing lists of ids; each list contains the ids of all the input nodes that belong to a group. The id of each group corresponds to the index in the main list.

- ***associationNodes:***
  Is a list of nodes of the class AssociationNode that points to all the nodes of this class, is used to keep track of the existing associations and check the existence of a particular one when looking for new possible ones.

- ***emotionalImpact :***
  Is a field updated at every time step and is used for control, it is the one used as recording signal in the interactive mode and always takes the value of the first position in the input array.

- ***lastEmotionalImpact:***
  It retains the value of the emotionalImpact at the previous time step and is updated when a value is set for *emotionalImpact*.

- ***depth:***
  It's an integer that represents the number of levels the network has.

- **associationThreshold:**
  It is used in the interactive mode. It is a number in the interval (0, 1) that indicated the minimum reverberation a node must have in order to be considered when looking for new associations.

- **maxAssociationSize:**
  Used in the interactive mode, is an integer that represents the maximum number of nodes that can be associated by a single node.

Abstract methods:

- ***void protected abstract void lookForAssociations():***
  This is the function called when associations are created automatically, the process should perform both the search for possible associations as well as the addition of the corresponding nodes.

- ***void train(double emotionalImpact):***
  This method is called at every time step while the emotional impact is higher than zero and must include all the trainings procedures at the network level. It also must call the train function of each node, which is done by calling the method *train* of the class Node.

Basic management functions

There are several basic functions that allow handling the network, those that are necessary to build a network are:

- ***addNode(Node node)***
  - This function receives as input an object of any subclass of Node, and adds it to the nodes list of the network, but also to different lists depending of the kind of node. It assigns an id to each node added taken from the index in the nodes list.

- ***removeNode(int nodeId):***
  - Receives the id of the node to be removed and performs the removal taking care of all the lists at which the node is included, and the updating of the ids of the rest of the nodes.
    The function will remove all nodes that have no input or associations with just one after the removal. So when the function is called more than one node can be actually removed.

- ***addEdge(int originNode, int destinationNode, double imaginaryWeight, double realWeight, Boolean inhibitory)***
  This function takes as parameter, first the origin and the destination nodes id. It also receives the initial weights for imaginary activity at the origin node, and real activity at the destination node. Finally, it receives a Boolean value that indicates whether the edge is inhibitory or not. When an edge is created an Edge object is added to each of the nodes, one in the outgoingEdges list of the origin node, and other in the incomingEdges of the destination node.

- ***remove edge(int originNode, int destinationNode)***
  - Given the ids of the origin node and the destination node, the edge is removed if it exists. This is done by removing the objects at the corresponding edges lists of each node.

The Tool box user's manual

In this section the main issues on the usage of the toolbox are explained. The functionality of the toolbars and menus are shown, as well as the characteristics of the interface and the way information is displayed.

The toolbar is divided in two smaller ones; the first one is file toolbar that is the one with which the basic actions over files can be performed; these actions are described below.

Figure 10 The file Tool bar in the tool box.

- New network (Ctrl+N): Creates a new network of general purpose that, depending on settings and the input stream with which is fed, can be used in manual or interactive mode.

- Open network (Ctrl+O): Loads a previously saved network in a *.TNN* file.

- Save network (Ctrl+S): Saves the network in a TNN file in a specified path, if no such a file has been specified it will open a file dialog in order to select it.

- Open input file (Ctrl+I): Opens an input file with which the network will be fed; it looks for text files (.txt).

- Add nodes (Ctrl+A): Opens the add node dialog with which the nodes to construct the network can be added.

Some of these functions are found in the File menu, plus the function *Save network as* which allows to change the destination file at which the network is saved. It also includes the Exit item (Ctrl+Q).

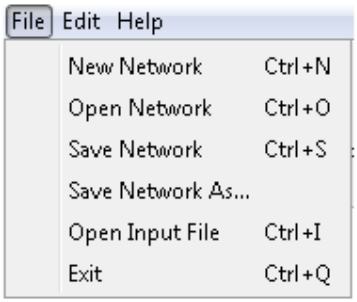


Figure 11 The file menu in the tool box.

When add nodes is called the following dialog is displayed, allowing to choose the kind of node to be added. It displays a description of the function the selected kind of node performs and the quantity of nodes to add can be selected.

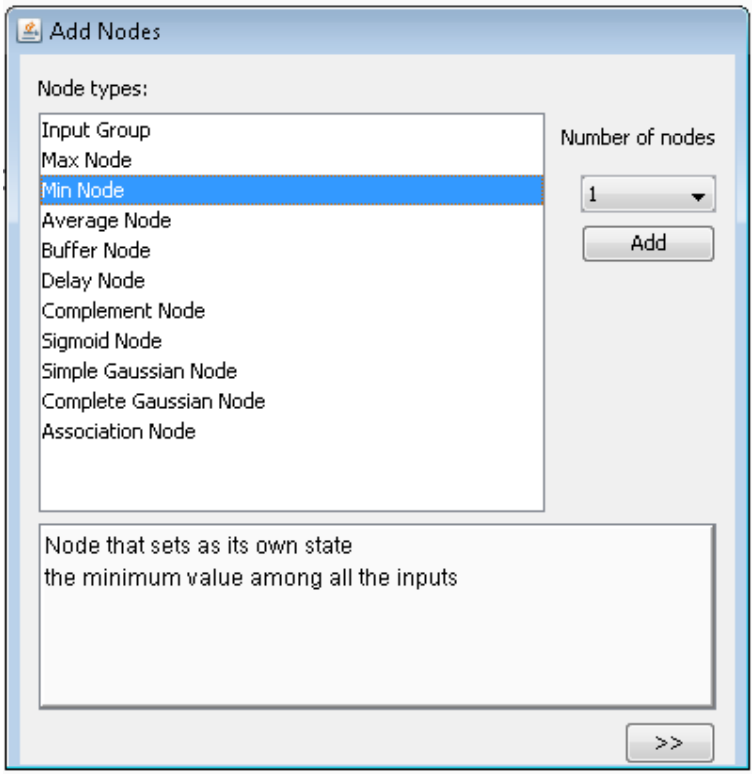


Figure 12 The add nodes dialog.

The second part in the division of the bar is the inputs toolbar which handles the way the inputs are fed into the network. It has four buttons, two of them are to read streams from the input files, and the other two are to generate random inputs.

Figure 13 The inputs tool bar in the tool box.

From left to right the buttons in this bar are:

- *Complete stream button:* This will read the whole input file feeding the network step by step. It will do that several times depending on the parameter “Number of iterations over input file” that can be set by the user in the settings dialog.

-*Step button:* This will only read one line in the input file at the time, so it goes one step at the time through the input stream.

- *Random stream button:* This will generate a random stream whit a number of steps equal to the same parameter used for the Complete stream button.

- *Random step:* It will generate a single random input.

While exploring the network and manipulating the nodes they will be shown in three different ways in relation to the mouse actions. The three states are normal, mouse over, and node selected.

Figure 14 Different states for interaction with the nodes in the tool box. From left to right, normal state, mouse over state, and selected state.

When clicking over a node this will get selected and while the mose pointer is over it its information will be shown.

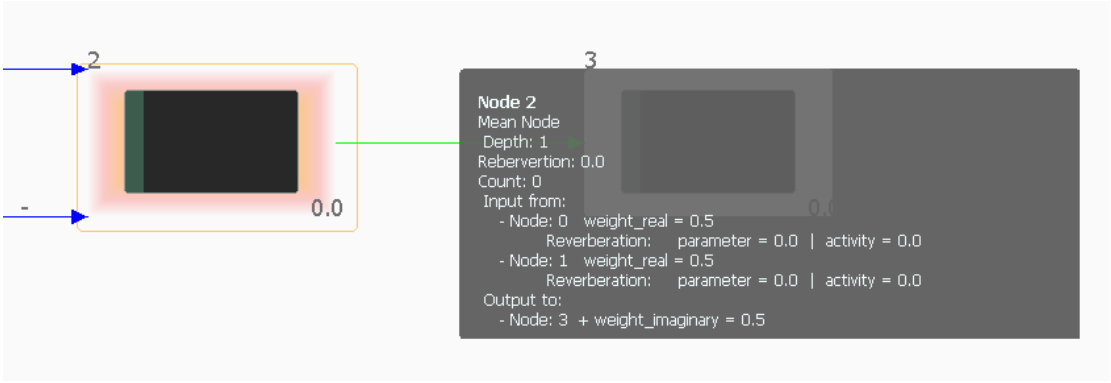


Figure 15 Information of the selected node displayed when the mose is over it.

The same way, as the node is selected it can be removed restarted or its label can be changed, this is done by right clicking on the selected node.

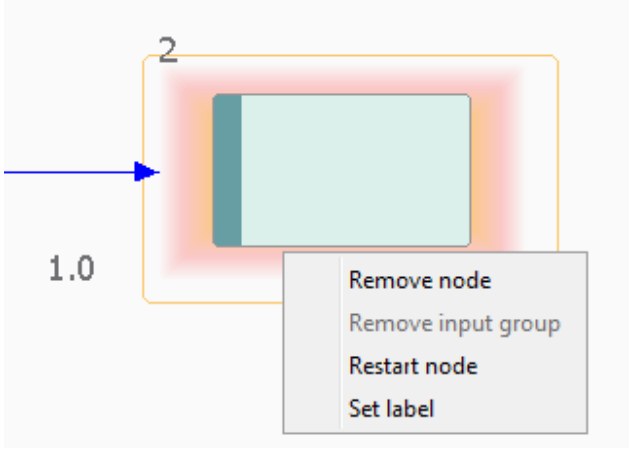


Figure 16 Edit menu for a selected node.

Also when the node is selected its relation to other node can be edited by clicking the node to be related; this action will display a pop up menu that gives the options:

- Add edge
- Add inhibitory edge
- Remove edge

If there is no edge between the two nodes an edge in purple color will show the possible connection to be created in order to visualize it easier.

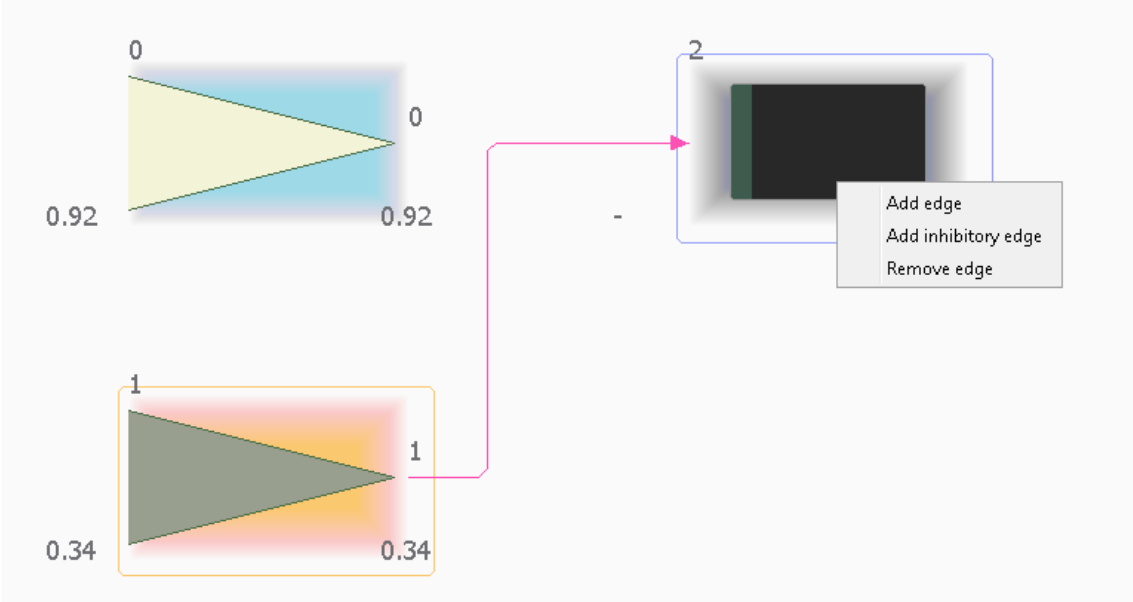


Figure 17 Connections menue deployed on a node to be related to the currently selected node.

Settings (Ctrl+T)
The settings dialog allows changing the basic parameters for the creation of nodes as well as options for the input stream reading and the random input generation.

In the upper part of the dialog the parameters for the creation of new nodes can be edited. These are the “Association Threshold” for the minimum reverberation needed on a node to be considered for new associations, and the “Maximum Association Size” which determines the maximum number of nodes admitted per association when added automatically.

It also can be selected whether or not to “Allow Gaussian growth” for the creation of new Complete Gaussian Nodes.

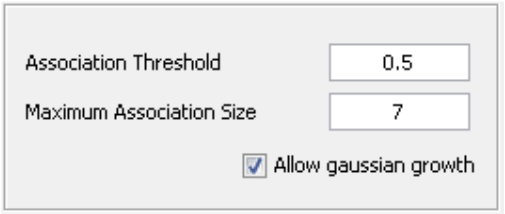


Figure 18 Association parameters in the Settings box.

The second part in the dialog allows changing the number of times the input file is read when the complete stream button or random streams are used. It can be also specified whether the randomly generated inputs are binary or not. When not selected the random inputs generated will be numbers in the interval [0, 1], otherwise they will be binary (values 0 or 1).

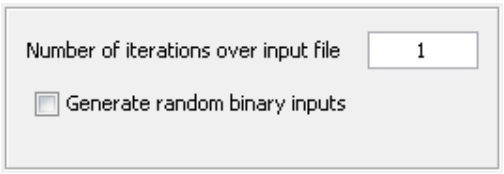


Figure 19 Inputs reading and generation parameters in the settings box.

Zooming and exploration

For zooming the zoom bar or the scroll wheel of the mouse can be used which will enlarge the size of the nodes and therefore the whole network.

To explore the network this can be moved throughout the scree by clicking at any empty space and moving the mouse while still clicking, the network will follow the movement of the mouse.

Results

The objectives of the TNN model, as stated in the description are mainly related to both the transparency and the ability to perform deductive and inductive reasoning at the same time. The transparency on one hand is a point that has been stressed during the implementation and was explained in the corresponding section of this document. On the other hand the performance of the model on the proposed abilities for reasoning has not been shown directly so far. Therefore this results section is mainly focused on showing how this model deals with these kinds of reasoning through some basic examples.

Descriptive examples

Simple associations

This example shows how a simple association is created when three inputs appear simultaneously and how partial information elicits certain prediction in form of imaginary activity.

At the first time step of the example the three inputs are completely active as well as the recording signal. At the following step the inputs all go down to 0 and the association is created (Node 3 in figure 20).

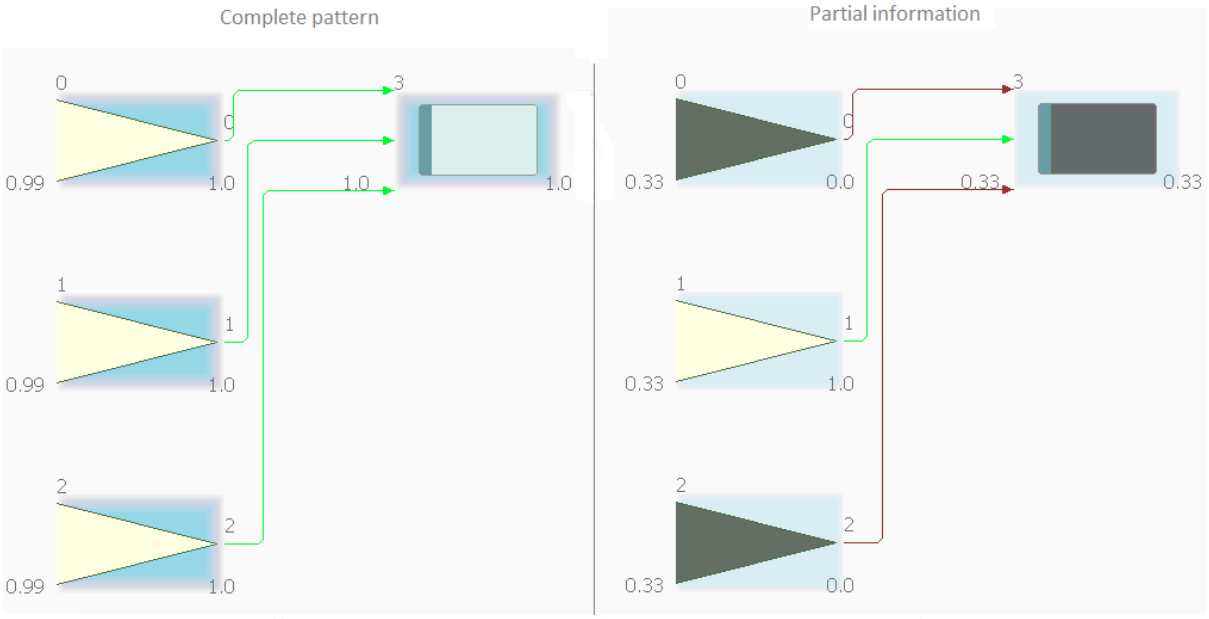


Figure 20 Depicts the differences betwwen the activation of an association node when the information is complete and when it is partial.

This simple example can be used to show how the imaginary activity of the nodes is useful to predict or infer possible inputs out of partial information. To show this the network is fed with partial information; taken into account that in this example there is only one association the predictions of the sensors will only depend on that association.

If the input activates only the second sensor then the association will be active at approximately a 33%, and as the only examples shown to the network has been the complete association including the three sensors, the prediction to each of them is going to be a third as well. Note that if the pattern is complete the prediction at every sensor will be practically 1.

Composed concepts and inferences

The following example shows how a network that has created a composed concept out of two previous ones. In this case two different concepts are shown to the network separately, each of them relates two specific sensors. Afterwards the two concepts are shown at the same time and that creates another at a deeper level.

When each concept is presented at different time the network creates an association node for each of them, called concept 1 and concept 2 in figure 21. Afterwards, when the two concepts are shown together the network uses the associations created previously to build a composed concept at a deeper level representing the two basic ones together.

That means that no concept is created including the four sensors since deeper composed concepts are preferred by the construction rules. This can be seen as an inductive learning since the network is creating more abstract and concrete associations as it finds relationships in the activation of more basic ones.

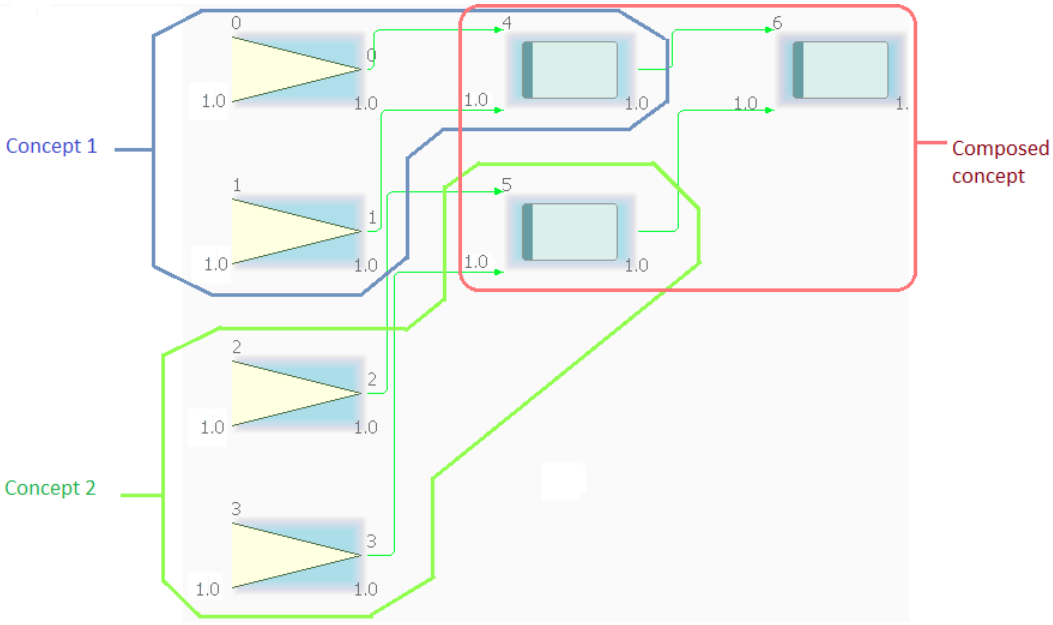


Figure 21 Depicts how to interpret the formation of a concept from two previous ones.

When the network is fed again with just one of the two basic concepts one can see how the composed concept gets activated to a 50%. This partial activation allows the network to predict possible associations that could appear by means of the imaginary activity as shown in figure 22.

The imaginary activation at node 5 representing the second concept, as well as in the two inactivated sensors, represents an inference from the known association between the two basic concepts. This activation indicates that there is a possibility of the two basic associations of appearing together, which is an association that has been learnt and is used for inference through imaginary activity in this case.

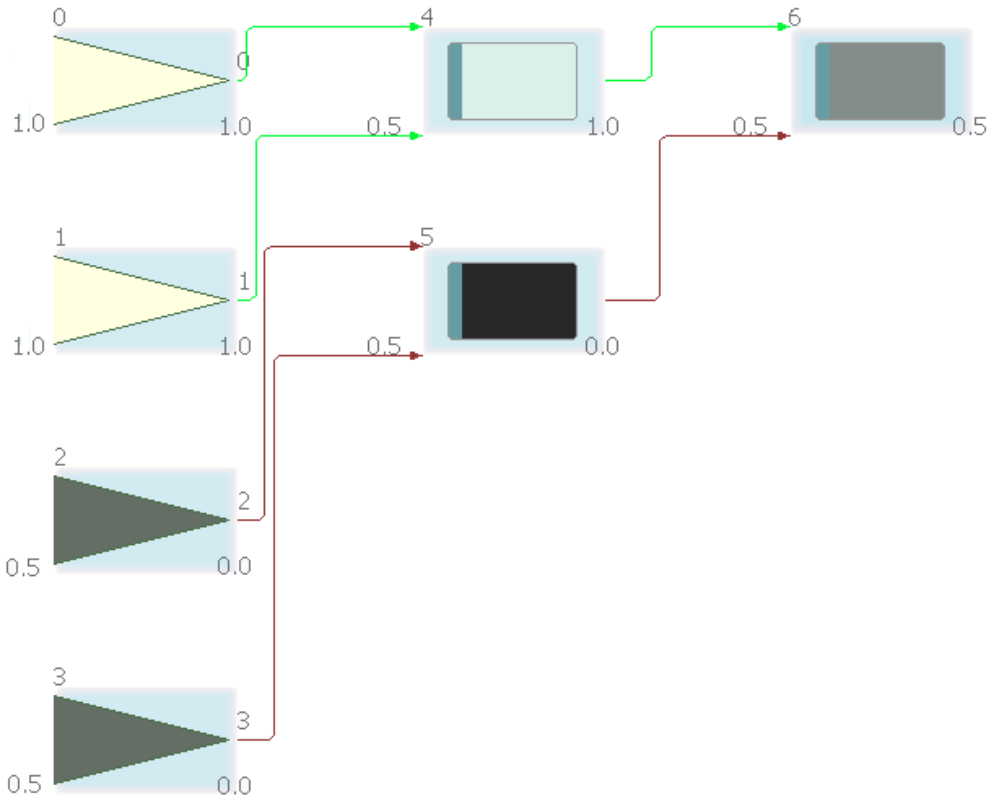


Figure 22 Partial activation of a deep concept when only one of the concepts associated is presented in the input.

Temporal associations

Similarly, association will be created when the sensors are activated sequentially, and in this case the maximum activation of the association will be reached as the sequence is completed after increasing after each time step. This means that as more information in relation to the sequence learnt more real and imaginary activities there will be in the corresponding association.

However if all the elements of the sequence are presented in different order than the one learnt, the association will increase its real activity but will never

reach the same value as in the case of the sequence being presented as it was learnt.

The sequence taught to the network in this example is simply three sensors being activated consecutively. In figure 23 after training is done one can see how the activation in the association node increases accordingly to the amount of information as the sequence learnt is shown again. This increase appears both in the real and the imaginary activities, showing how through imaginary activity prediction on future and inference on past are performed.

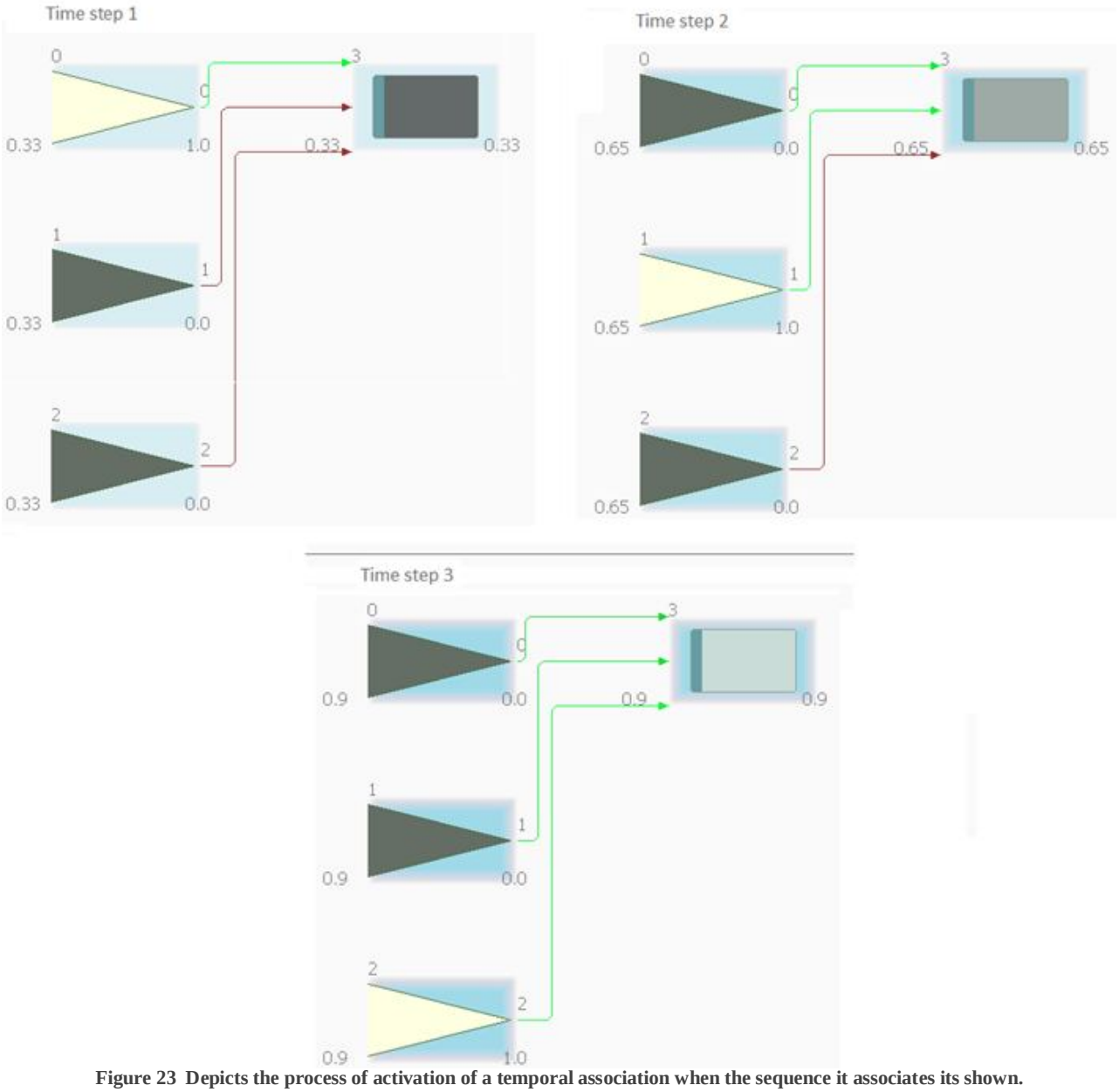


Figure 23 Depicts the process of activation of a temporal association when the sequence it associates its shown.

Predicting the most probable input form partial information

In this example two different sequences are shown to the network. Both the sequences are of a three time steps length and both include the first two sensors as the beginning of the sequence. Then, the only difference between them is the last element, being in one case the third sensor and in the other the fourth one.

These examples can resemble the two number sequences 1-2-3 and 1-2-4, which only differ on one number but one can be more probable to occur than the other.

Here is shown how the imaginary activity also represents the probability of activation of a concept or sensor given certain activation at a deeper level. To do this, the second sequence (1-2-4) is presented to the network half of the times the other one (in this case 15 times). Then the probability of occurring of each is different and in principle one must be half of the other.

When the network is fed with partial information (sequence 1-2), then the prediction on future input can be seen in the imaginary activity of the other two sensors; in this cases, as shown in figure 24, the imaginary activity of the third sensor is 0.5, whereas the one at the fourth it is 0.27. This implies that the probabilities learnt by the network into the weights of imaginary activities are tending to the actual probability of appearance of the sequences learnt.

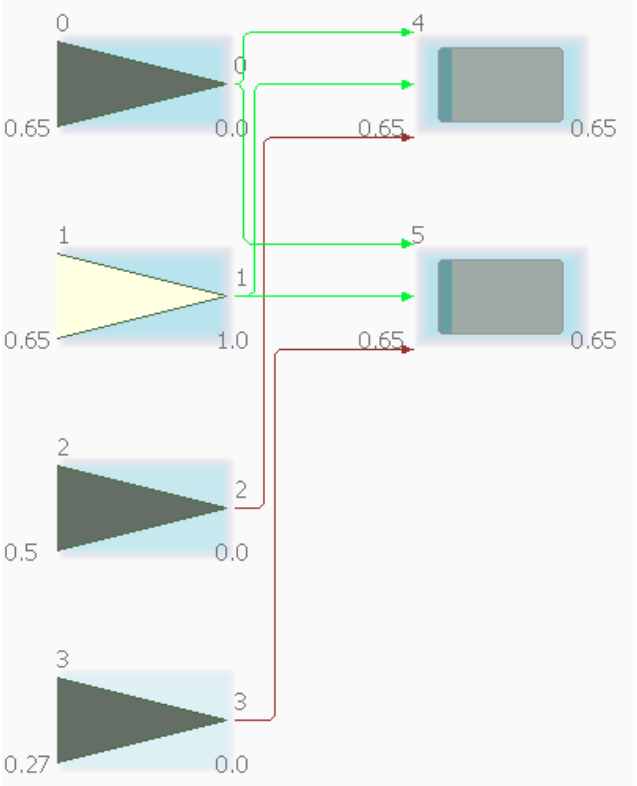


Figure 24 Imaginary activity as probabilistic inference for two different sequences that are parially equal.

Generalization

Generalization is performed by the Gaussian growth, to show how this works a simple example is shown.

In this example the network has only two sensors and the generalization is to be made over two different classes. Thus, at the beginning a Complete Gaussian Node is manually added receiving inputs from both the sensors, and afterwards examples from the two classes are shown from a distribution as the one that appears in the first table.

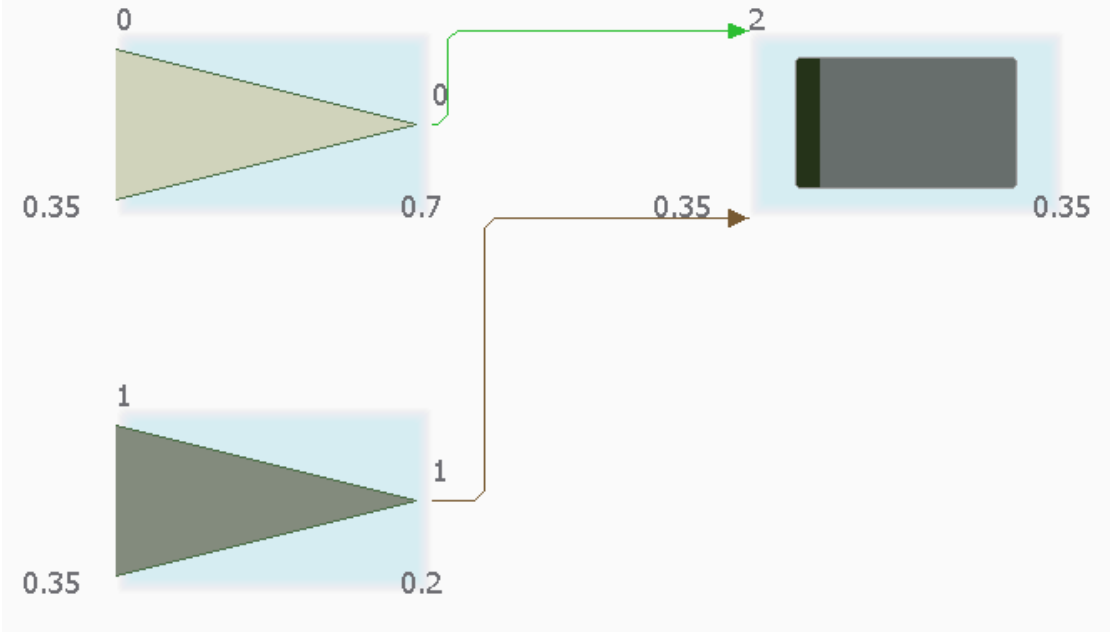


Figure 25 Generalization by means og gaussian nodes.

	Class 1		Class 2	
	Input 1	Input 2	Input 1	Input 2
Mean	0,750	0,183	0,190	0,750
Std deviation	0,041	0,062	0,070	0,041

After training is performed with a hundred inputs for each class the result is a network with two nodes, each representing one class with the following parameters

	Class 1 (Node 2)		Class 2 (Node 3)	
	Input 1	Input 2	Input 1	Input 2
Mean	0,74	0,19	0,19	0,74
Std deviation	0,063	0,081	0,086	0,063

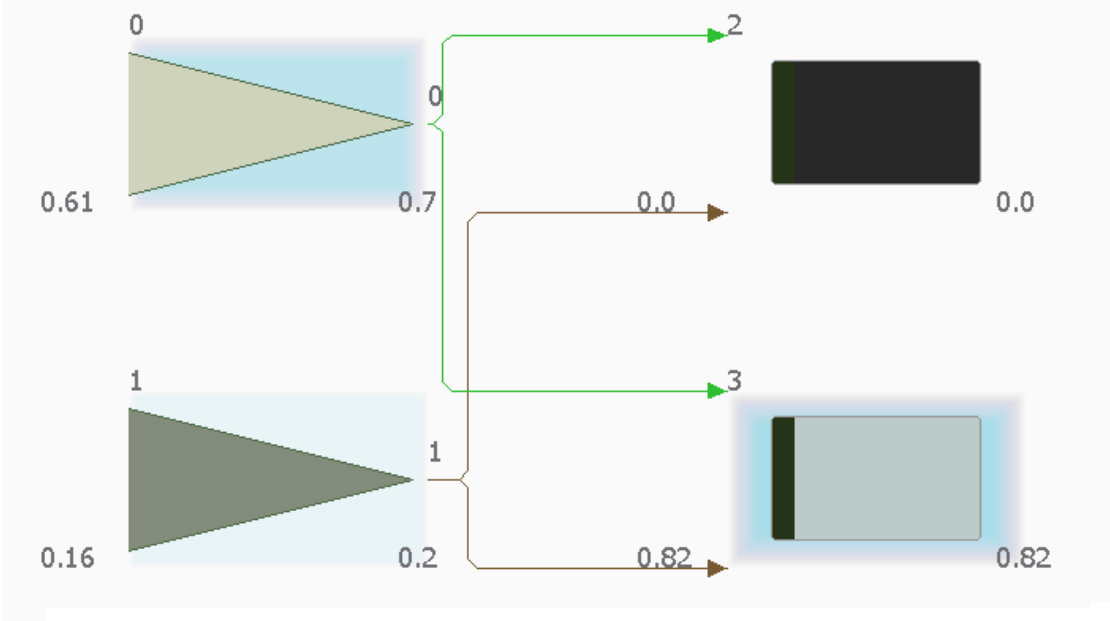


Figure 26 Generalization of two different classes by means of gaussian nodes and gaussian growth.

The same way the weights for the imaginary activities of the sensors reflect basically the same values as the means learnt at the nodes. This shows that, in this case, the imaginary activity reflects the expected value of the input from the activation of a node. In other words, if for example node 3 were to have a real state of 1.0, the imaginary activity at the inputs would be 0.19 and 0.74 respectively depicting the expected value of the inputs for each class.

Drawbacks

Through these examples it is shown the basic idea behind the attempt of TNN to achieve both kinds of reasoning in the same model while keeping the transparency. However, these are achieved when inputs are shown in a very controlled manner which can only be accomplished if the problem is known enough by the user. However, in cases where the problem is not completely known setting the basic parameters can become a complicated task, and a bad selection can easily lead the network to an explosion of redundant associations.

Redundant and unnecessary associations certainly make the network not really useful and affects the transparency as it reaches states at which the meaning of the concepts created are incomprehensible.

To give an example of this if a sequence of four sensors is shown repeatedly to a network and the association threshold is set too low (in this case 0.5), even when there is control by means of the recording signal there are at least three associations created for this sequence in the first level. Each of these associations is allowed by the constructions rules as they have different inputs.

All of these associations get activated as the sequence is presented again, and therefore will be associated again at the next level. This process is repeated over and over again as the sequence is repeated creating an endless number of levels as depicted in figure 27.

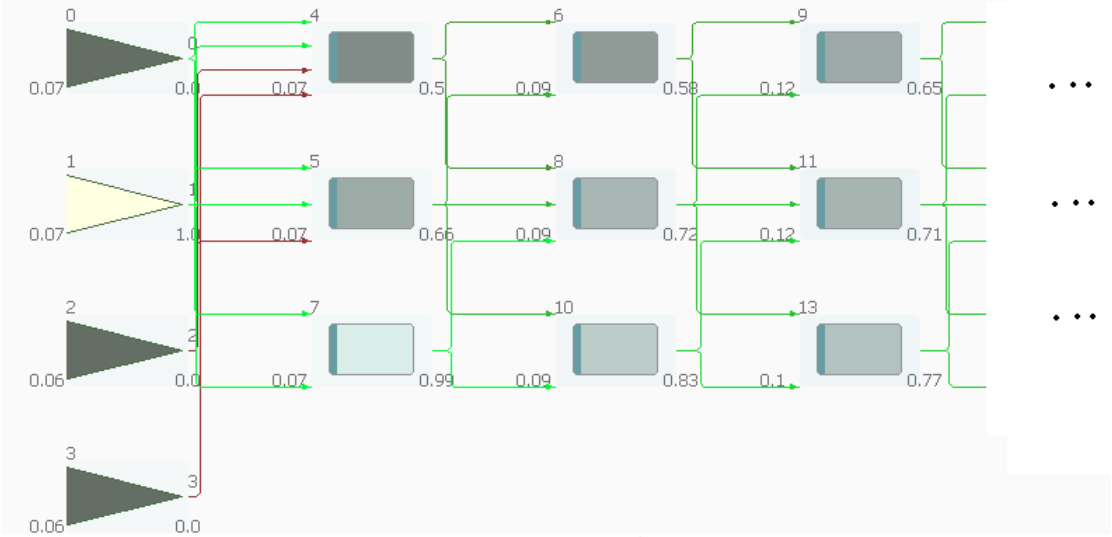


Figure 27 Explosive growth in lack of proper control.

This drawback on the controllability affects the utility of the model and its own transparency creating a need for new approaches in relation to the construction rules.

On the Toolbox

In relation to the main objectives set for the thesis in terms of the implementation, the toolbox turned out to be a very beneficial and easy to use instrument for the developing of the TNN.

The goals of modularity and flexibility of the elements was achieved by means of a simple object oriented approach, that after deployed allowed a very fast editing which represented an advantage for experimenting with many changing ideas and designs during the development of different concepts of the TNN.

The final interface fulfilled completely the desired characteristics for the application and even went beyond the requirements. The usability of the tool is based on a very simple and intuitive interface that, once the concept of TNN is clear, allows creating, training and manipulating a network in about three simple steps.

The final result offers a great deal of freedom to get information of the elements of the network as well as to manipulate and customize them stressing the point of transparency on which the whole project is based.

The ability to move freely through the network by just one click and as the zoom is easily manipulated exploring results into a very simple and helpful task, especially when the networks grow to some many nodes.

In general the feedback that the platform allowed while it was being improved permitted highlighting different drawbacks of presented approaches for the TNNs and opportunely fostered new proposal to improve the model.

Discussion

As implemented the TNN model allowed performing basic inferences, inductive learning and generalization for specific problems as depicted in the results section. All of the capabilities are achieved keeping the transparency as it is one of the main objectives, and in fact this characteristic is completely useful and understandable for the problems evaluated.

However, all of the evaluations have been performed in relation to basic, controlled and constrained problems, reason why one could ask whether the ability for induction is still feasible if the data stops being constrained. This question arises as the symbolic structures are always manipulated by the definition of a concept represented in the inputs, but induction has not been clearly achieved for inputs which meanings or behaviors are not clear for the user at the design stage.

Similarly, the growth control turns out to be a crucial issue for the equilibrium of the network, but it may get quite instable as the data becomes more complex. That fact sets a big challenge for developing automatic learning, but again it may be challenging even for problems that include an unknown behavior.

Thus, coping with the control problem is an objective to be stressed in further development as the reality is that almost any real problem may include unknown behaviors and unpredictable complexity. But it is also important to highlight that there is an apparent tradeoff between the stability, the control and the transparency that is not easy to deal whit in the current model. Not controlling leads to instability and therefore interpretability and transparency get spoiled.

When studying other models it is found that the problem on controlling growth has been addressed by many, and they always end up facing the so called combinatorial complexity or the biding problem. These problems arise when models create concepts by binding representations of intrinsic characteristics in the entities to be represented. This idea becomes problematic as the representations include more and more characteristics since the possible number of combination increases exponentially.

Therefore, in the field there has been a quest for reducing the complexity and the amount of associations created when these kinds of problems arise. One important process with which possible solutions to the biding problem in real biological systems have been described is the need for attention (Holcombe, 2009). Attention can be described in many ways and the real process is not completely known, however its possible need for solving the biding problem implies certain control on which inputs and the way they are bind at a given time.

This overall idea of attention may have direct relation to the control signals in the TNN model, which allow proper performance by stating when to associate and selecting the inputs that are to be related. In that sense the control and the

constraining of inputs proposed is arguable in terms of solving the binding problem, but then again it is not a feasible solution for automating the process. In order to achieve automatic selectivity, for associations many other features have to be included in the model.

Some other issues are also related to the biding problem beyond the complexity and growth of the structures. Typical examples are connected to the ability of assessing proper meaning to the associations and are those including relational statements of the kind “Mary loves John”. That relation could be seen as two subject or concepts bind by a relation called “love”, or a relation among three particular concepts that are bind together; however, the original statement does not imply the complementary “John loves Mary”, but when the relationship is created as described both the statements can mean the same, which is not necessarily the case.

For this example the TNN model may manage the two possible statements by means of two different associations having the possibility of interpreting them differently. This is possible if the statements are presented as different sequences, then each sequence will represent a different concept to which a distinctive meaning could be assessed. However this implies a symbolic manipulation that requires that the three concepts are clear and again the design is limited to a symbolic well understand behavior of the inputs.

In general, this discussion and the development of the model are related to a broader set of questions on the need for the development of models capable of really creating and understanding concepts and not only perform some manipulations on specific symbols to solve particular tasks.

When analyzing the existing models for cognition and problem solving one may have the sense that generally all the applications aim to solve a particular task that the researches have in mind and leave many details apart. This fact is reasonable as the goal is based on solving specific problems; however it is a very narrow perspective if the goal is to enhance the performance and capability of models, or aiming for a more general problem solving approach.

The fact is that, as mentioned by Ekbia (2010), there is a utilitarian notion of human life as being composed by a set of problems and human intelligence as nothing but a capability to solve them. The issue with this idea is that it somehow neglects that the human brain, and in fact any other brain, even when capable of solving problems by sequences of steps is rather a dynamic system with many structures shaping behavior, and the basis of its characteristics should never be confused with that particular ability of describing problems by sequences or by any other semantics.

For example, when one focuses on an specific problem and asses intelligence based on the ability of performing clear steps for reaching a desired solution, one must also think that in reality humans do not always reason in a correct way (Bringsjord,2008). In fact, psychological works by Kahneman, Tversky and colleagues suggest that human cognition might be “non-rational, non-optimal, and non-probabilistic in fundamental ways” (Chater, Tenenbaum and Yuille,

2006). Thus, it is questionable to try to emerge intelligence from a fact that is not completely related to the actual phenomena being modeled.

Nevertheless, this partial definitions on intelligence are common to many models, for instances in the SOAR architecture (University of Michigan, 2012) the ultimate goal in intelligence and complete rationality is settled as the ability to use all available knowledge to solve any problem the system encounters; but then again, if rationality is inspired by human behavior, the question is why such a crucial definition does not take into account the fact that humans never consider all the possibilities when taking a decision, but just some particular ones that depend on parameters of which we may not be even conscious at all (Overskeid, 2008).

On the other hand, there are approaches such as the epigenetic robotics that emerges from the need of robots to understand and develop in relation to their environments, and rejects more classical views of robotics in which the capabilities of robots are completely based on pre-programed behaviors that removes any possibility of concept creation and development. This approach also states the absolute need for the robots to have a body with which to explore and verify knowledge, which implies that any model to develop knowledge and intelligence must be available to interact with the environment.

In this sense the TNN model lacks crucial characteristics as it cannot interact with its environment at all, and actually its development is not related to any kind of interaction beyond the inputs it receives. In fact interaction may be needed to achieve the automatic characteristics that would remove the need for the control signals that are implemented in the current model.

Nevertheless the TNN model is still being developed, and further versions of it may take into account lessons from the present work and existing models and approaches that have faced similar problems. Therefore, as a consequence of the discussed issues and ideas, some particular future work considered relevant for the TNN is mentioned in the following section.

Conclusions

- The toolbox implemented successfully satisfied the needs and met the requirements under the constraints given by the partial development of the TNN model.

- Basic inferences, inductions and generalizations achieved are linked to the symbolic manipulation of the input grounded on previous knowledge of the problem by the user, reason why there is not enough information or evidence to claim that in general the model is capable of the two main reasoning capabilities aimed.

- The emphasis on symbolic meanings of the inputs in the problems definitions may be restricting the model into becoming completely symbolic one.

- The model still requires of great deal of control signals and thresholds definitions for proper performance, so that the scalability and creation of solutions for dynamic and complex problems is not yet feasible with the current model.

- The imaginary activity allows generating inferences and predictions by means of probabilistic relations; therefore it should be used in the learning rules as it represents big part of the knowledge in the network and may be of utility to infer the relevance of a given input.

- Despite the information given by the imaginary activity, its interpretability in some cases is still too ambiguous as it encloses different concepts in just one parameter; thus if inference and prediction or expectancy are to be interpreted separately there should be a difference in the treatment of this parameter for each of them.

- The introduction of outputs and manipulation of inputs in relation to the knowledge needs of the system and specific goals may be needed to achieve automatic learning. That is, interaction with the environment may be mandatory to reach the ambitious goals of this model.

Future work

Here some issues considered to be relevant for future development of the TNN are mentioned as well as possible hints for their solution.

- Firstly, with the model as implemented when an association is created it asses the same relevance to each of its inputs; however, it can be sound to argue that not in every case the concepts associated give the same amount of information about the concept represented.

A toy example of this is an association that represents the concept “apple”; let’s assume that this concept is formed just by biding together the concepts “apple shape” and “green color“. It is highly probable that “green color” is associated to many concepts, whereas “apple shape” may be associated only by the concept “apple” (if all the apples were green); so the concept “apple shape” gives much more information to this association than what “green color” does. In, other words, the relevance of the “apple shape” concept is much higher to the association “apple” than the color green.

To solve this it is suggested that weights for the real activity must be included in the edges. The meaning of these weights, as mentioned, would be the relevance of each input, which if managed correctly may keep the transparency of the model intact and give it better capabilities.

- Another issue, in this case related to controlling the growth of the network, is that stability depends largely on the association threshold, and a low threshold normally leads to an uncontrolled growth; but even worst, the problem is extended as the definition of “low value” in general is different for any given situation.

To partially cope with this problem an incremental threshold is proposed. The idea is that as more abstract the concepts are the more stable they should be. This means that at the bottom levels the association could be created and also deleted easily, but the deeper the concepts being related are, the more the threshold should be increased to ensure that associations created bind clearer concepts at each level. However, there will still exist a dependence on the nature of the problem being addressed.

- It was also shown that by means of the imaginary activity it is possible to achieve inferences and predictions by means of probabilistic relations; however these are related in the same way and represented by the same unique value. This unique value affects the interpretability of the results as it is hard to tell what of the possible meaning the activity has at a given point, or in fact, the meaning may always arise from a mixture of all possible interpretations, which is against the supreme goal of transparency. Therefor a different treatment for either the interpretation or the computation of the imaginary activity is suggested.

- Something also related to the amount of associations created and its control is the forgetting rules that have to be implemented. The goal of these rules is to delete certain associations that are not really relevant as may have been caused by noisy inputs or other situations.

However, though not currently implemented, the forgetting rules in the model have been proposed to be simply based on the usage frequency of the associations, meaning that if an association does not get activated in a “long” time it will be deleted.

That idea presents to main problems in practice. First, the amount of time steps that have to be elapsed in order to delete certain association is defined in a too ambiguous way. A wrong definition of this time may lead to instability, in this case because there will be forgetting of relevant information as concepts are not presented in a certain period, and also may avoid the creation of deeper concepts.

Secondly, the basic idea of this rule may be ignoring that some associations might be extremely relevant but at the same time very rare, then the importance of an association is not taken into account here.

Therefore, it is suggested that certain kind of relevance is assed to each association in order to control forgetting, which at the same time may help in controlling growth of the whole network, but without the risk of deleting curtail information because of an arbitrary definition of time thresholds for forgetting.

- In relation to the problems evaluated and the way they are presented, more emphasis on different kinds of inputs and evaluations are highly recommended as the model and its performance as it is now, is strongly directed to be a symbolic model with all what that implies. The model should be available to create representations beyond the restrictions and reach the interpretation the user wants without a previous forced definition.

- Finally, as mentioned in the previous subsection, there is a strong need for interaction in order to develop intelligent systems, then the model has to be focused more in the creation of concepts and behavior based on interaction and not only on extracting information from the inputs. Thus, a more context based and interactive learning both for the model and the implementation is suggested for a better and more interesting progress.

References

C. Strannegård, O. Häggström, J. Wessberg, C. Balkenius 2012 *Transparent Neural Networks*, paper presented at the SweCog

C. Strannegård 2011. *Transparent Neural Networks*. [manuscripts] March 2011, Chalmers University of Technology

Troy D. Kelley., 2003. *Symbolic and Sub-symbolic Representations in Computational Models of Human Cognition: What Can be Learned from Biology?*. Theory & Psychology, Vol. 13, No. 6, 2003, pp. 847–860

James L. McClelland, Matthew M. Botvinick, David C. Noelle, David C. Plaut, Timothy T. Rogers, Mark S. Seidenberg and Linda B. Smith., 2010. *Letting structure emerge: connectionist and dynamical systems approaches to cognition.* Trends in Cognitive Sciences, Vol. 14, Issue 8, August 2010, pp. 348-356

Thomas L. Griffiths, Nick Chater, Charles Kemp, Amy Perfors and Joshua B. Tenenbaum., 2010. *Probabilistic models of cognition: exploring representations and inductive biases.* Trends in cognitive Sciences, Volume 14, Issue 8, August 2010, pp. 357–364

Nick Chater, Joshua B. Tenenbaum and Alan Yuille., 2006. *Probabilistic models of cognition: Conceptual foundations.* Trends in Cognitive Sciences Volume 10, Issue 7, July 2006, pp. 287–291

Amanda J.C. Sharkey., 2009. *Artificial Neural Networks and Cognitive A Modelling.* Encyclopedia of Artificial Intelligence 2009, pp. 161-166

Inc. Numenta., 2011. *Hierarchical Temporal Memory including HTM Cortical Learning Algorithms.*

The International Computer Science Institute, 2012. *shrutí*.[online] Available at: < http://www.icsi.berkeley.edu/~shastri/shruti/> [Accessed April 2012]

Bringsjord, S.,2008. *Declarative/Logic-Based Computational Cognitive Modeling*, in Sun, R., ed., The Cambridge Handbook of Computational Psychology (Cambridge, UK: Cambridge University Press 2008), pp. 127-169

Lewis, R.L., 1999. *Cognitive modeling, symbolic*. In Wilson, R. and Keil, F. (eds.), The MIT Encyclopedia of the Cognitive Sciences. Cambridge, MA: MIT Press, 1999

R. Sun., 2001. *Artificial intelligence: Connectionist and symbolic approaches*. In: N. J. Smelser and P. B. Baltes (eds.), International Encyclopedia of the Social and Behavioral Sciences. pp.783-789. Pergamon/Elsevier, Oxford.

University of Michigan, 2012. *SOAR*. [online] Available at: <*http://sitemaker.umich.edu/soar/home*> [Accessed April 2012]

Perlovsky, L.I., 2007. *Neural Dynamic Logic of Consciousness: the Knowledge Instinct*. In Eds. L.I. Perlovsky, R. Kozma, Neurodynamics of High Cognitive Functions, Springer.

ACT-R Research Group Department of Psychology, Carnegie Mellon University., 2012., *ACT-R*. [online] Available at: < [http://act-r.psy.cmu.edu/> [Accessed April 2012]

Ramamurthy. Uma, Baars. Bernard J, D'Mello. Sidney K, Franklin. Stan., 2006. *LIDA: A Working Model of Cognition.* The 7th International Conference on Cognitive Modeling, Trieste, Italy, April 2006. (Eds: Danilo Fum, Fabio Del Missier and Andrea Stocco, p. 244-249, published by Edizioni Goliardiche, Trieste)

Metta. Giorgio and Berthouze. Luc,. 2005. *Epigenetic robotics: Modelling cognitive development in robotic systems.* Cognitive Systems Research, Volume: 6, Issue: 3, pp, 189-192

Anthony F. Morse, Joachim de Greeff, Tony Belpeame, and Angelo Cangelosi,. 2010. *Epigenetic Robotics Architecture (ERA)*. IEEE Transactions on Autonomous Mental Development, Vol. 2, Issue. 4, December 2010

Stoytchev. Alexander., 2009. *Some Basic Principles of Developmental Robotics.* IEEE Transactions on Autonomous Mental Development, Vol. 1, Issue. 2, August 2009

Asada. Minoru, Hosoda. Koh, Kuniyoshi. Yasuo, Ishiguro. Hiroshi, Inui. Toshio, Yoshikawa. Yuichiro, Ogino. Masaki and Yoshida. Chisato., 2009.*Cognitive Developmental Robotics: A Survey.* IEEE Transactions on Autonomous Mental Development, Vol. 1, Issue. 1, May 2009

d'Avila Garcez. Artur S, and Lamb. Luis C., 2011. *Chapter 18 Cognitive Algorithms and Systems: Reasoning and Knowledge Representation*., Perception-Action Cycle: Models, Architectures, and Hardware: Models, Algorithms and Systems (Springer Series in Cognitive and Neural Systems)

Holcombe, A.O., 2009. *The Binding Problem*., In E. Bruce. Goldstein (Ed.), The Sage Encyclopedia of Perception.

Ekbia, H., 2010. *Fifty years of research in artificial intelligence.* In: Cronin, B. (Ed.) Annual Review of Information Science and Technology, Volume 44. Medford, NJ: Information Today/American Society for Information Science and Technology, pp. 201-242.

Overskeid. Geir., 2008. *They Should Have Thought About the Consequences: The Crisis of Cognitivism and a Second Chance for Behavior Analysis*., The Psychological Record, 2008, Vol 58, issue 1, pp. 131–151