NeSy'11

Seventh International Workshop on Neural-Symbolic Learning and Reasoning

Workshop at IJCAI-11
Barcelona, Catalonia (Spain)
Sunday, July 17th, 2011

Editors/Organizers:

Artur d'Avila Garcez, City University London, UK
Pascal Hitzler, Kno.e.sis Center, Wright State University, Dayton, OH, USA
Luis Lamb, Universidade Federal do Rio Grande do Sul, Brazil

Workshop Schedule

09:30–10:30 Keynote talk by Robert Kowalski, The Connection Graph Proof Procedure as a Logical-Connectionist Model of the Mind

10:30–11:00 Andreas Wichert, Neural Sub-Symbolic Reasoning

Coffee break

11:30–12:00 Christian Huyck, Roman Belavkin, Fawad Jamshed, Kailash Nadh, Peter Passmore, Emma Byrne and Dan Diaper. CABot3: A Simulated Neural Games Agent

12:00–12:30 Gadi Pinkas, Priscila Lima and Shimon Cohen. Compact Crossbar Variable Binding for Neuro-Symbolic Computation

12:30–13:00 Silvano Colombo Tosatto, Guido Boella, Leon Van Der Torre, Artur d'Avila Garcez and Valerio Genovese. Embedding Normative Reasoning into Neural Symbolic Systems

Lunch break

14:30–15:00 Michelangelo Diligenti, Marco Gori and Marco Maggini. Towards Developmental AI: The paradox of ravenous intelligent agents

15:00–15:30 Yoshiaki Gotou, Wataru Makiguchi and Hajime Sawamura. Extracting Argumentative Dialogues from the Neural Network that Computes the Dungean Argumentation Semantics

15:30–16:30 Posters and Demos: Leo de Penning, Visual Intelligence using Neural-Symbolic Learning and Reasoning (demo); Silvano Colombo-Tosatto, Neural-Symbolic Learning: How to play Soccer (demo); Norbert Tsopze, Engelbert Mephu Nguifo and Gilbert Tindo, Extracting both MofN rules and if-then rules from the training neural networks (poster); Ekaterina Komendantskaya and Qiming Zhang, SHERLOCK – An Inteface for Neuro-Symbolic Networks (poster)

Coffee break

17:00–18:00 Discussion (Directions for Neural-Symbolic Computation)

Workshop Organisers

Artur d’Avila Garcez, City University London, UK
Pascal Hitzler, Kno.e.sis Center, Wright State University, Dayton, OH, USA
Luis Lamb, Universidade Federal do Rio Grande do Sul, Brazil

Programme Committee

Preface

Artificial Intelligence researchers continue to face huge challenges in their quest to develop truly intelligent systems. The recent developments in the field of neural-symbolic computation bring an opportunity to integrate well-founded symbolic artificial intelligence with robust neural computing machinery to help tackle some of these challenges.

Neural-symbolic systems combine the statistical nature of learning and the logical nature of reasoning.

The Workshop on Neural-Symbolic Learning and Reasoning provides a forum for the presentation and discussion of the key topics related to neural-symbolic integration.

Topics of interest include:

- The representation of symbolic knowledge by connectionist systems;
- Learning in neural-symbolic systems;
- Extraction of symbolic knowledge from trained neural networks;
- Reasoning in neural-symbolic systems;
- Biological inspiration for neural-symbolic integration;
- Integration of logic and probabilities in neural networks;
- Structured learning and relational learning in neural networks;
- Applications in robotics, simulation, fraud prevention, semantic web, fault diagnosis, bioinformatics, etc.

Table of Contents

Invited Keynote Talk

The Connection Graph Proof Procedure as a Logical-Connectionist Model of the Mind

Robert Kowalski, Department of Computing, Imperial College London, UK

In this talk, I present an agent architecture in which thinking is modeled as activating links in a connection graph of goals and beliefs, represented in abductive logic programming (ALP) form. In this ALP agent model, beliefs are represented as logic programming clauses, and goals are represented as a variant of range-restricted FOL clauses. This clausal representation facilitates a connectionist (connection graph) implementation, in which forwards and backwards reasoning are different ways of selecting and activating links. The activation of links can also be determined in the manner of Patie Maes' activation networks, by associating different strengths with different goals and beliefs and different weights with different links, spreading the strength of activation throughout the graph in proportion to the weights on the links.

Neural Sub-Symbolic Reasoning

Andreas Wichert
Department of Informatics
INESC-ID / IST - Technical University of Lisboa
Portugal
andreas.wichert@ist.utl.pt

Abstract

The sub-symbolical representation often corresponds to a pattern that mirrors the way the biological sense organs describe the world. Sparse binary vectors can describe sub-symbolic representation, which can be efficiently stored in associative memories. According to the production system theory, we can define a geometrically based problem-solving model as a production system operating on sub-symbols. Our goal is to form a sequence of associations, which lead to a desired state represented by sub-symbols, from an initial state represented by sub-symbols. We define a simple and universal heuristics function, which takes into account the relationship between the vector and the corresponding similarity of the represented object or state in the real world.

1 Introduction

One form of distributed representation corresponds to a pattern that mirrors the way the biological sense organs describe the world. Sense organs sense the world by receptors. By the given order of the receptors the living organisms experience the reality as a simple Euclidian geometrical world. Changes in the world correspond to the changes in the distributed representation. Prediction of these changes by the nervous system corresponds to a simple geometrical reasoning process. Mental imagery problem solving is an example for a complex geometrical problem- solving. It is described by a sequence of associations, which progressively change the mental imagery until a desired solution of a problem is formed. For example, do the skis fit in the boot of my car? Mental representations of images retain the depictive properties of the image itself as perceived by the eye [Kosslyn, 1994]. The imagery is formed without perception through the construction of the represented object from memory. Symbols on the other hand are not present in the world; they are the constructs of human mind to simplify the process of problem solving. Symbols are used to denote or refer to something other than them, namely other things in the world (according to the pioneering work of Tarski [Tarski, 1956]). They are defined by their occurrence in a structure and by a formal language, which manipulates these structures [Simon, 1991; Newell, 1990]. In this context, symbols do not by themselves, represent any utilizable knowledge. They cannot be used for a definition of similarity criteria between themselves. The use of symbols in algorithms which imitate human intelligent behavior led to the famous physical symbol system hypothesis by Newell and Simon (1976) [Newell and Simon, 1976]: The necessary and sufficient condition for a physical system to exhibit intelligence is that it be a physical symbol system. We do not agree with the physical symbol system hypothesis. Instead we state that the actual perception of the world and manipulation in the world by living organisms lead to the invention or recreation of an experience, which at least in some respects, resembles the experience of actually perceiving and manipulating objects in the absence of direct sensory stimulation. This kind of representation is called sub-symbolic. Sub-symbolic representation implies heuristic functions. Symbols liberate us from the reality of the world although they are embodied in geometrical problem solving through the usage of additional heuristics functions. Without the use of heuristic functions real world problems become intractable.

The paper is organized a follows: We review the representation principles of objects by features as used in cognitive science. In the next step we indicate how the perception-oriented representation is build on this approach. We define the sparse sub-symbolical representation. Finally, we will introduce the sub-symbolical problem solving which relies on a sensorial representation of the reality.

2 Sub-symbols

Perception-oriented representation is an example of sub-symbolical representation, such as the representation of numbers by the Oksapmin tribe of Papua New Guinea. The Oksapmin tribe of Papua New Guinea counts by associating a number with the position of the body [Lancy, 1983]. The sub-symbolical representation often corresponds to a pattern that mirrors the way the biological sense organs describe the world. Vectors represent patterns. A vector is only a sub-symbol if there is a relationship between the vector and the corresponding similarity of the represented object or state in the real world through sensors or biological senses. Feature based representation is an example of sub-symbolical representation.

2.1 Feature Approach

Objects can be described by a set of discrete features, such as red, round and sweet [Tversky, 1977; McClelland and Rumelhart, 1985]. The similarity between them can be defined as a function of the features they have in common [Osherson, 1995; Sun, 1995; Goldstone, 1999; Gilovich, 1999]. The contrast model of Tversky [Tversky, 1977] is one well-known model in cognitive psychology [Smith, 1995; Opwis and Plötzner, 1996] which describes the similarity between two objects which are described by their features. An object is judged to belong to a verbal category to the extent that its features are predicted by the verbal category [Osherson, 1987]. The similarity of a category C and of a feature set F is given by the following formula, which is inspired by the contrast model of Tversky [Tversky, 1977; Smith, 1995; Opwis and Plötzner, 1996],

$$Sim(C,F) = \frac{|C \cap F|}{|C|} \quad \in [0,1] \tag{1}$$

$|C|$ is the number of the prototypical features that define the category a. The present features are counted and normalized so that the value can be compared. This is a very simple form of representation. A binary vector in which the positions represent different features can represent the set of features. For each category a binary vector can be defined. Overlaps between stored patterns correspond to overlaps between categories.

2.2 The Lernmatrix

The Lernmatrix, also simply called "associative memory" was developed by Steinbuch in 1958 as a biologically inspired model from the effort to explain the psychological phenomenon of conditioning [Steinbuch, 1961; 1971]. Later this model was studied under biological and mathematical aspects by Willshaw [Willshaw *et al.*, 1969] and G. Palm [Palm, 1982; 1990].

Associative memory is composed of a cluster of units. Each unit represents a simple model of a real biological neuron. The Lernmatrix was invented in by Steinbuch, whose goal was to produce a network that could use a binary version of Hebbian learning to form associations between pairs of binary vectors, for example each one representing a cognitive entity. Each unit is composed of binary weights, which correspond to the synapses and dendrites in a real neuron. They are described by $w_{ij} \in \{0,1\}$ in Figure 1. T is the threshold of the unit. We call the Lernmatrix simply *associative memory* if no confusion with other models is possible [Anderson, 1995a; Ballard, 1997].

The patterns, which are stored in the Lernmatrix, are represented by binary vectors. The presence of a feature is indicated by a 'one' component of the vector, its absence through a 'zero' component of the vector. A pair of these vectors is associated and this process of association is called learning. The first of the two vectors is called the *question vector* and the second, the *answer vector*. After learning, the question vector is presented to the associative memory and the answer vector is determined by the retrieval rule.

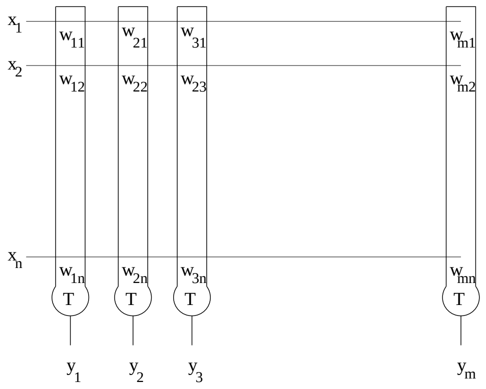


Figure 1: The Lernmatrix is composed of a set of units which represent a simple model of a real biological neuron. The unit is composed of weights, which correspond to the synapses and dendrites in the real neuron. In this figure they are described by $w_{ij} \in \{0,1\}$ where $1 \leq i \leq m$ and $1 \leq j \leq n$. T is the threshold of the unit.

Learning In the initialization phase of the associative memory, no information is stored. Because the information is represented in weights, they are all initially set to zero. In the learning phase, pairs of binary vector are associated. Let $\vec{x}$ be the question vector and $\vec{y}$ the answer vector, the learning rule is:

$$w_{ij}^{new} \begin{cases} 1 & if \;\; y_i \cdot x_j = 1 \\ w_{ij}^{old} & \text{otherwise.} \end{cases} \tag{2}$$

This rule is called the binary Hebbian rule [Palm, 1982]. Every time a pair of binary vectors is stored, this rule is used.

Retrieval In the *one-step* retrieval phase of the associative memory, a fault tolerant answering mechanism recalls the appropriate answer vector for a question vector $\vec{x}$. For the presented question vector $\vec{x}$, the most similar learned $\vec{x^l}$ question vector regarding the Hamming distance is determined and the appropriate answer vector $\vec{y}$ is identified. For the retrieval rule, the knowledge about the correlation of the components is sufficient. The retrieval rule for the determination of the answer vector $\vec{y}$ is:

$$y_i = \begin{cases} 1 & \sum_{j=1}^{n} w_{ij} x_j \geq T \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

where T is the threshold of the unit. The threshold is set as proposed by [Palm *et al.*, 1997] to the maximum of the sums $\sum_{j=1}^{n} w_{ij} x_j$:

$$T := \max_{1 \leq i \leq m} \left\{ \sum_{j=1}^{n} w_{ij} x_j \right\}. \tag{4}$$

Only the units that are maximal correlated with the question vector are set to one.

Storage capacity For an estimation of the asymptotic number of vectorpairs $(\vec{x}, \vec{y})$ which can be stored in an associative memory before it begins to make mistakes in retrieval phase, it is assumed that both vectors have the same dimension n. It is also assumed that both vectors are composed of M 1s, which are likely to be in any coordinate of the vector. In this case it was shown [Palm, 1982; Hecht-Nielsen, 1989; Sommer, 1993] that the optimum value for M is approximately

$$M \doteq \log_2(n/4) \tag{5}$$

and that approximately [Palm, 1982; Hecht-Nielsen, 1989]

$$L \doteq (\ln 2)(n^2/M^2) \tag{6}$$

of vector pairs can be stored in the associative memory. This value is much greater then n if the optimal value for M is used. In this case, the asymptotic storage capacity of the Lernmatrix model is far better than those of other associative memory models, namely 69.31%. This capacity can be reached with the use of sparse coding, which is produced when very small number of 1s is equally distributed over the coordinates of the vectors [Palm, 1982; Stellmann, 1992]. For example an optimal code is defined as following; in the vector of the dimension n=1000000 M=18 ones should be used to code a pattern. The real storage capacity value is lower when patterns are used which are not sparse or are strongly correlated to other stored patterns. Usually suboptimal sparse codes a sufficiently good to be used with the associative memory. An example of a suboptimal sparse code is the representation of words by context-sensitive letter units [Wickelgren, 1969; 1977; Rumelhart and McClelland, 1986; Bentz *et al.*, 1989]. The ideas for the used robust mechanism come from psychology and biology [Wickelgren, 1969; 1977; Rumelhart and McClelland, 1986; Bentz *et al.*, 1989]. Each letter in a word is represented as a triple, which consists of the letter itself, its predecessor, and its successor. For example, six context-sensitive letters encode the word *desert*, namely: *_de, des, ese, ser, ert, rt_*. The character "_" marks the word beginning and ending. Because the alphabet is composed of 26+1 characters, 27^3 different context-sensitive letters exist. In the 27^3 dimensional binary vector each position corresponds to a possible context-sensitive letter, and a word is represented by indication of the actually present context-sensitive letters. We demonstrate the principle of sparse coding by an example of the visual system and visual scene representation.

2.3 Sparse features

In hierarchical models of the visual system [Riesenhuber and Poggio, 1999],[Fukushima, 1980], [Fukushima, 1989], [Cardoso and Wichert, 2010] the neural units have a local view unlike the common fully-connected networks. The receptive fields of each neuron describe this local view. During the categorization the network gradually reduces the information from the input layer through the output layer. Integrating local features into more global features does this. Supposed in the lower layer tow cells recognize two categories at neighboring position, and these two categories are integrated into a more global category. The first cell is named α the second β. The numerical code for α and β may represent the position of each cell. A simple code would indicate if a cell is active or not. One indicates active, zero not active. For c cells we could indicate this information by a binary vector of dimension c. For an image of size $x \times y$ a cell covers the image X times. A binary vector that describes that image using the cell representation has the dimension $c \times X$. For example gray images of the size 128×96 resulting in vectors of dimension 12288 can be covered with:

- 3072 masks M of the size of a size 2×2 resulting in a binary vector that describes that image has the dimension $c_1 \times 3072$, $X_1 = 3072$ (see Figure 2 (a)).
- 768 masks M of the size of a size 4×4 resulting in a binary vector that describes that image has the dimension $c_2 \times 768$, $X_2 = 768$ (see Figure 2 (b)).
- 192 masks M of the size of a size 8×8 resulting in a binary vector that describes that image has the dimension $c_3 \times 192$, $X_3 = 192$ (see Figure 3 (a)).
- 48 masks M of the size of a size 16×16 resulting in a binary vector that describes that image has the dimension $c_4 \times 48$, $X_4 = 48$ (see Figure 3 (b)).

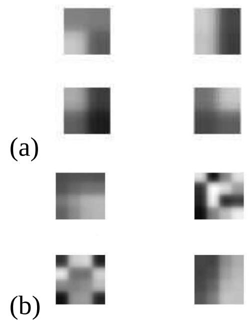


Figure 2: (a) Two examples of of squared masks M of a size 2×2. (b) Two examples of squared masks M of a size 4×4. The masks were learned using simple k-means clustering algorithm.

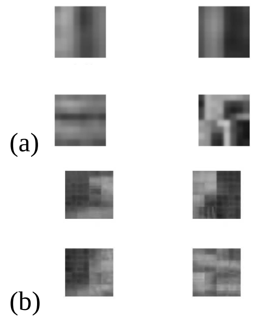


Figure 3: (a) Two examples of of squared masks M of a size 8×8. (b) Two examples of squared masks M of a size 16×16. The masks were learned using simple k-means clustering algorithm.

The ideal c value for a sparse code is related to $M \doteq \log_2(n/4)$.

$$X = \log_2(X \cdot c/4)$$
$$2^X = X \cdot c/4$$
$$c = \frac{4 \cdot 2^X}{X} \qquad (7)$$

The ideal value for c grows exponentially in relation to X. Usually the used value for c is much lower then the ideal value resulting in a suboptimal sparse code. The representation of images by masks results in a suboptimal code. The optimal code is approached with the size of masks, the bigger the mask, the smaller the value of X. The number of pixels inside a mask grows quadratic. A bigger masks implies the ability to represent more distinct categories, which implies a bigger c.

An ideal value for c is possible, if the value for $X << 100$. Instead of covering an image by masks, we indicate the present objects. Objects and their position in the visual field can represent a visual scene. A sub-vector of the vector representing the visual scene represents each object. For example, if there is a total of 10 objects, the c value is 409. To represent 409 different categories of objects at different positions resulting in 4090 dimensional binary vector. This vector could represent $\frac{409!}{(409-20)!}$ different visual states of the world. The storage capacity of the associative memory in this case would be around 159500 patterns, which is 28 times bigger as the number of the units (4090).

2.4 Problem Solving

Human problem solving can be described by a problem-behavior graph constructed from a protocol of the person talking aloud, mentioning considered moves and aspects of the situation. According to the resulting theory, searching whose state includes the initial situation and the desired situation in a problem space [Newell, 1990; **?**] solves problems. This process can be described by the production system theory. The production system in the context of classical Artificial Intelligence and Cognitive Psychology is one of the most successful computer models of human problem solving. The production system theory describes how to form a sequence of actions, which lead to a goal, and offers a computational theory of how humans solve problems [Anderson, 1995b]. Production systems are composed of if-then rules that are also called productions. A rule [contains several if patterns and one or more then patterns. A pattern in the context of rules is an individual predicate, which can be negated together with arguments. A rule can establish a new assertion by the then part (its conclusion) whenever the if part (its premise) is true. One of the best-known cognitive models, based on the production system, is SOAR. The SOAR state, operator and result model was developed to explain human problem-solving behavior [Newell, 1990]. It is a hierarchical production system in which the conflict-resolution strategy is treated as another problem to be solved. All satisfied instances of rules are executed in parallel in a temporary mode. After the temporary execution, the best rule is chosen to take action. The decision takes place in the context of a stack of earlier decisions. Those decisions are rated utilizing preferences and added to the stack by chosen rules. Preferences are determined together with the rules by an observer using knowledge about a problem.

According to the production system theory, we can define a geometrically based problem-solving model as a production system operating on vectors of fixed dimensions. Instead of rules, we use associations and vectors represent the states. Our goal is to form a sequence of associations, which lead to a desired state represented by a vector, from an initial state represented by a vector. Each association changes some parts of the vector. In each state, several possible associations can be executed, but only one has to be chosen. Otherwise, conflicts in the representation of the state would occur. To perform these operations, we divided a vector representing a state into sub-vectors. An association recognizes some sub-vectors in the vector and exchanges them for different sub-vectors. It is composed of a precondition of fixed arranged m sub-vectors and a conclusion. Suppose a vector is divided into n sub-vectors with $n > m$. An association recognizes m different sub-vectors and exchanges them for different m sub-vectors. To recognize m sub-vectors out of n sub-vectors we perform a permutation p(n,m) and verify if each permutation corresponds to a valid precondition of an association. For example, if there is a total of 7 elements and we are selecting a sequence of three elements from this set, then the first selection is one from 7 elements, the next one from the remaining 6, and finally from the remaining 5, resulting in 7 * 6 * 5 = 210, see Figure 4.

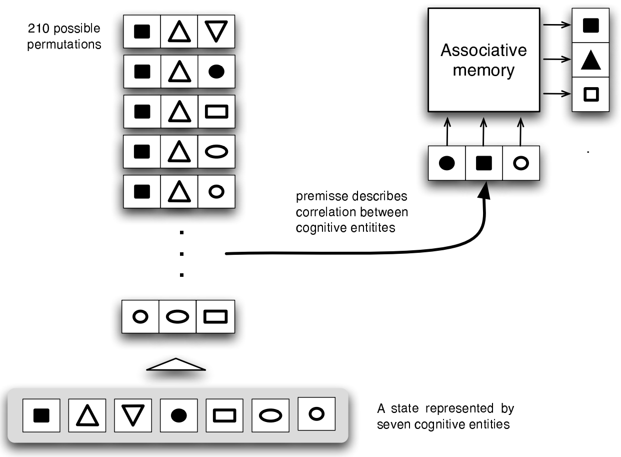

210 possible permutations
Associative memory
premisse describes correlation between cognitive entitites
A state represented by seven cognitive entities

Figure 4: To recognize one learned association permutations are formed. For example, if there is a total of 7 elements and we are selecting a sequence of three elements from this set, then the first selection is one from 7 elements, the next one from the remaining 6, and finally from the remaining 5, resulting in 7 * 6 * 5 = 210. In our example, all possible three-permutations sub-vectors of seven sub-vectors are formed to test if the precondition of an association is valid.

Out of several possible associations, we chose the one,

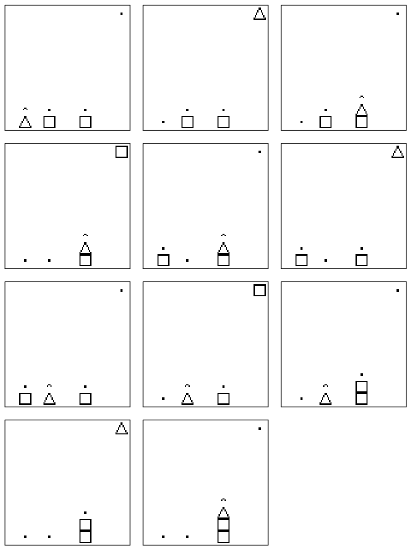


Figure 5: The simplest method corresponds to a random choice, and does not offer any advantage over simple symbolical representation. An example of visual planning of the tower building task of three blocks using the random choice is shown. The upper left pattern represents the initial state; the bottom right pattern, the desired state.

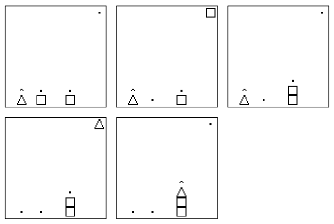


Figure 6: The computation can be improved by a simple and universal heuristics function, which takes into account the relationship between the vector and the corresponding similarity of the represented object or states in the real world as expressed by Equation 1 for binary vectors. The heuristics function makes a simple assumption that the distance between the states in the problem space is related to the distance between the sub-symbols representing the visual states. The distance between the states in the problem space is related to the distance between the visual state. An example of visual planning of the tower building task of three blocks using hill climbing using the similarity function, see Equation 1. The upper left pattern represents the initial state; the bottom right pattern, the desired state.

which modifies the state in such a way that it becomes more similar to the desired state according to the Equation 1. The desired state corresponds to the category of Equation 1, each feature represents a possible state. The states are represented by sparse features. With the aid of this heuristic hill climbing is performed. Each element represents an object. Objects are represented by some dimensions of the space and form a sub-space by themselves, see Figure 4.

The computation can be improved by a simple and universal heuristics function, which takes into account the relationship between the vector and the corresponding similarity of the represented states see Figure 5 and Figure 6. The heuristics function makes a simple assumption that the distance between the states in the problem space is related to the similarity of the vectors representing the states.

The similarity between the corresponding vectors can indicate the distance between the sub-symbols representing the state. Empirical experiments in popular problem-solving domains of Artificial Intelligence, like robot in a maze, block world or 8-puzzle indicated that the distance between the states in the problem space is actually related to the similarity between the images representing the states [Wichert, 2001; Wichert *et al.*, 2008; Wichert, 2009].

3 Conclusion

Living organisms experience the world as a simple. The actual perception of the world and manipulation in the world by living organisms lead to the invention or recreation of an experience that, at least in some respects, resembles the experience of actually perceiving and manipulating objects in the absence of direct sensory stimulation. This kind of representation is called sub-symbolic. Sub-symbolic representation implies heuristic functions. The assumption that the distance between states in the problem space is related to the similarity between the sub-symbols representing the states is only valid in simple cases. However, simple cases represent the majority of exiting problems in domain. Sense organs sense the world by receptors which a part of the sensory system and the nervous system. Sparse binary vectors can describe sub-symbolic representation, which can be efficiently stored in associative memories. A simple code would indicate if a receptor is active or not. One indicates active, zero not active. For c receptors we could indicate this information by a binary vector of dimension c with only one ”1”, the bigger the c, the sparser the code. For receptors in X positions the sparse code results in $c \times X$ dimensional vector with X ones.

Acknowledgments

This work was supported by Fundao para a Cencia e Tecnologia (FCT) (INESC-ID multiannual funding) through the PIDDAC Program funds.

References

[Anderson, 1995a] James A. Anderson. *An Introduction to Neural Networks*. The MIT Press, 1995.

[Anderson, 1995b] John R. Anderson. *Cognitive Psychology and its Implications*. W. H. Freeman and Company, fourth edition, 1995.

[Ballard, 1997] Dana H. Ballard. *An Introduction to Natural Computation*. The MIT Press, 1997.

[Bentz *et al.*, 1989] Hans J. Bentz, Michael Hagstroem, and Günther Palm. Information storage and effective data retrieval in sparse matrices. *Neural Networks*, 2(4):289–293, 1989.

[Cardoso and Wichert, 2010] A. Cardoso and A Wichert. Neocognitron and the map transformation cascade. *Neural Networks*, 23(1):74–88, 2010.

[Fukushima, 1980] K. Fukushima. Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol Cybern*, 36(4):193–202, 1980.

[Fukushima, 1989] K. Fukushima. Analisys of the process of visual pattern recognition by the neocognitron. *Neural Networks*, 2:413–420, 1989.

[Gilovich, 1999] Thomas Gilovich. Tversky. In *The MIT Encyclopedia of the Cognitive Sciences*, pages 849–850. The MIT Press, 1999.

[Goldstone, 1999] Robert Goldstone. Similarity. In *The MIT Encyclopedia of the Cognitive Sciences*, pages 763–765. The MIT Press, 1999.

[Hecht-Nielsen, 1989] Robert Hecht-Nielsen. *Neurocomputing*. Addison-Wesley, 1989.

[Kosslyn, 1994] Stephen M. Kosslyn. *Image and Brain, The Resolution of the Imagery Debate*. The MIT Press, 1994.

[Lancy, 1983] D.F. Lancy. *Cross-Cultural Studies in Cognition and Mathematics*. Academic Press, New York, 1983.

[McClelland and Rumelhart, 1985] J.L. McClelland and D.E. Rumelhart. Distributed memory and the representation of general and specific memory. *Journal of Experimental Psychology: General*, 114:159–188, 1985.

[Newell and Simon, 1976] A. Newell and H. Simon. Computer science as empirical inquiry: symbols and search. *Communication of the ACM*, 19(3):113–126, 1976.

[Newell, 1990] Allen Newell. *Unified Theories of Cognition*. Harvard University Press, 1990.

[Opwis and Plötzner, 1996] Klaus Opwis and Rolf Plötzner. *Kognitive Psychologie mit dem Computer*. Spektrum Akademischer Verlag, Heidelberg Berlin Oxford, 1996.

[Osherson, 1987] Daniel N. Osherson. New axioms for the contrast model of similarity. *Journal of Mathematical Psychology*, 31:93–103, 1987.

[Osherson, 1995] Daniel N. Osherson. Probability judgment. In Edward E. Smith and Daniel N. Osherson, editors, *Thinking*, volume 3, chapter two, pages 35–75. MIT Press, second edition, 1995.

[Palm *et al.*, 1997] G. Palm, F. Schwenker, F.T. Sommer, and A. Strey. Neural associative memories. In A. Krikelis and C. Weems, editors, *Associative Processing and Processors*, pages 307–325. IEEE Press, 1997.

[Palm, 1982] Günther Palm. *Neural Assemblies, an Alternative Approach to Artificial Intelligence*. Springer-Verlag, 1982.

[Palm, 1990] Günther Palm. Assoziatives Gedächtnis und Gehirntheorie. In *Gehirn und Kognition*, pages 164–174. Spektrum der Wissenschaft, 1990.

[Riesenhuber and Poggio, 1999] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999.

[Rumelhart and McClelland, 1986] D.E. Rumelhart and McClelland. On learning the past tense of english verbs. In J.L. McClelland and D.E. Rumelhart, editors, *Parallel Distributed Processing*, pages 216–271. MIT Press, 1986.

[Simon, 1991] Herbert A. Simon. *Models of my Life*. Basic Books, New York, 1991.

[Smith, 1995] Edward E. Smith. Concepts and categorization. In Edward E. Smith and Daniel N. Osherson, editors, *Thinking*, volume 3, chapter one, pages 3–33. MIT Press, second edition, 1995.

[Sommer, 1993] Friedrich T. Sommer. *Theorie neuronaler Assoziativspeicher*. PhD thesis, Heinrich-Heine-Universität Düsseldorf, Düsseldorf, 1993.

[Steinbuch, 1961] K. Steinbuch. Die Lernmatrix. *Kybernetik*, 1:36–45, 1961.

[Steinbuch, 1971] Karl Steinbuch. *Automat und Mensch*. Springer-Verlag, fourth edition, 1971.

[Stellmann, 1992] Uli Stellmann. *Ähnlichkeitserhaltende Codierung*. PhD thesis, Universität Ulm, Ulm, 1992.

[Sun, 1995] Ron Sun. A two-level hybrid architecture for structuring knowledge for commonsense reasoning. In Ron Sun and Lawrence A. Bookman, editors, *Computational Architectures Integrating Neural and Symbolic Processing*, chapter 8, pages 247–182. Kluwer Academic Publishers, 1995.

[Tarski, 1956] Alfred Tarski. *Logic, Semantics,Metamathematics*. Oxford University Press, London, 1956.

[Tversky, 1977] A. Tversky. Feature of similarity. *Psychological Review*, 84:327–352, 1977.

[Wichert *et al.*, 2008] A. Wichert, J. D. Pereira, and P. Carreira. Visual search light model for mental problem solving. *Neurocomputing*, 71(13-15):2806–2822, 2008.

[Wichert, 2001] Andrzej Wichert. Pictorial reasoning with cell assemblies. *Connection Science*, 13(1), 2001.

[Wichert, 2009] Andreas Wichert. Sub-symbols and icons. *Cognitive Computation*, 1(4):342–347, 2009.

[Wickelgren, 1969] Wayne A. Wickelgren. Context-sensitive coding, associative memory, and serial order in (speech)behavior. *Psychological Review*, 76:1–15, 1969.

[Wickelgren, 1977] Wayne A. Wickelgren. *Cognitive Psychology*. Prentice-Hall, 1977.

[Willshaw *et al.*, 1969] D.J. Willshaw, O.P. Buneman, and H.C. Longuet-Higgins. Nonholgraphic associative memory. *Nature*, 222:960–962, 1969.

CABot3: A Simulated Neural Games Agent

Christian Huyck, Roman Belavkin, Fawad Jamshed, Kailash Nadh, Peter Passmore,
Middlesex University
c.huyck@mdx.ac.uk
Emma Byrne
University College London
and **Dan Diaper**
DDD Systems

Abstract

CABot3, the third Cell Assembly roBot, is an agent implemented entirely in simulated neurons. It is situated in a virtual 3D environment and responds to commands from a user in that environment. It parses the user's natural language commands to set goals, uses those goals to drive its planning system, views the environment, moves through it, and learns a spatial cognitive map of it. Some systems (e.g. parsing) perform perfectly, but others (e.g. planning) are not always successful. So, CABot3 acts as a proof of concept, showing a simulated neural agent can function in a 3D environment.

1 Introduction

CABot3, the third Cell Assembly roBot, is a video game agent implemented entirely in simulated neurons. It assists a user in the game: viewing the 3D environment; processing natural language commands; making simple plans; and moving through, modifying, and learning about the environment.

As its name suggests, CABot3 makes extensive use of Cell Assemblies (CAs), reverberating circuits of neurons that are the basis of short and long-term memories [Hebb, 1949]. CABot3 represents symbolic knowledge in a neural network by CAs. Simple rules are implemented by simple state transitions, with a particular set of active CAs leading to the activation of a new set of CAs, and complex rules are implemented by variable binding combined with state transitions.

CABot3 is a virtual robot that creates and uses plans with a neural implementation of a Maes net [Maes, 1989], while natural language parsing is based around a standard linguistic theory [Jackendoff, 2002]. All agent calculations are done with Fatiguing Leaky Integrate and Fire (FLIF) neurons (see Section 2.1) and some of the network structure can be related to brain areas (see Section 4.2). The agent learns a spatial cognitive map of the rooms in the video game.

Two components of the CABots have been evaluated as cognitive models. The Natural Language Parser [Huyck, 2009] parses in human-like times, creates compositional semantic structures, and uses semantics to resolve prepositional phrase attachment ambiguities. It also learned the meaning of the verb *centre* from environmental feedback, closely related to a probability matching task [Belavkin and Huyck, 2010].

2 The Structure of CABot3

Due to space constraints, a complete description of CABot3 is not possible, though an almost complete description of an earlier version, CABot1, is available [Huyck and Byrne, 2009], and the code is available on http://www.cwa.mdx.ac.uk/cabot/cabot3/CABot3.html. A brief description of the neural model is described next, followed by a description of the subnetworks used, and a brief description of how those subnetworks are connected to generate CABot3's functionality.

2.1 FLIF Neurons

FLIF neurons are a modification of the relatively commonly used LIF model [Amit, 1989]. When a neuron has sufficient activation, it fires, and sends activation to neurons to which it is connected proportional to the weight w_{ji} of the synapse from the firing pre-synaptic neuron j to the post-synaptic neuron i. That weight can be negative. The simulations use discrete cycles, so the activation that is sent from a neuron that fires in a cycle is not collected by the post-synaptic neuron until the next cycle. If a neuron fires, it loses all its activation, but if it does not fire, it retains some, while some activation leaks away (decay); this is the leaky component and is modelled by a factor $D > 1$, where the activation is divided by D to get the initial activation at the next step. In CABot3, activation of neuron i at time t, A_{i_t} is defined by Equation 1. V_i is the set of all neurons that fired at $t-1$ connected to i.

$$A_{i_t} = \frac{A_{i_{t-1}}}{D} + \sum_{j \in V_i} w_{ji} \qquad (1)$$

Additionally, FLIF neurons fatigue. Each cycle they fire the fatigue level is increased by a constant, but when they do not fire, the fatigue level is reduced by another constant, but never below 0. The neuron fires at time t if its activity A minus fatigue F is greater than the threshold, see Equation 2.

$$A_{i_t} - F_{i_t} \geq \theta \qquad (2)$$

FLIF neurons are a relatively faithful model of neurons, though are relatively simple compared to compartmental models [Hodgkin and Huxley, 1952]. If each cycle is consider to take ten ms., it has been shown that 90% of the spikes emitted fall within one cycle of the spikes of real neurons on the same input [Huyck, 2011]. Aside from their biological

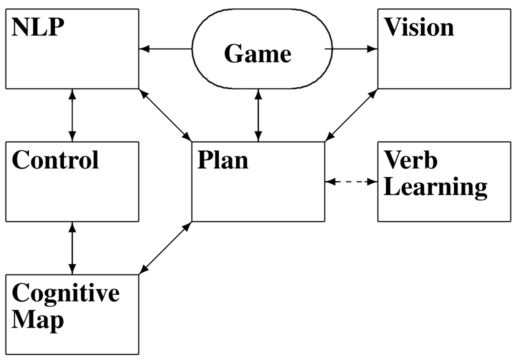


Figure 1: Gross Topology of CABot3. Boxes represent subsystems of subnets. The oval represents the environment.

fidelity, another benefit is that 100,000 FLIF neurons with a 10ms cycle can be simulated in real time on a standard PC.

Neurons are grouped into CAs, either manually by the developer, or through emergent connectivity. A given neuron may be part of one or more CAs.

2.2 SubNetworks

The FLIF neurons in CABot3 are grouped into 36 subnetworks. Each subnet is an array of neurons, and each may have different FLIF parameters and learning parameters, including no learning. In CABot3, connectivity within a subnet is always sparse, but it varies between subnets; this connectivity may have some degree of randomness, but in some cases it is tightly specified by the developer to guarantee particular behaviour. Subnets may also be connected to each other with neurons from one sending synapses to others; these types of connections vary similarly. These reflect differences, possibly caused in part by genetics, between different types of biological neuron.

Apart from biological fidelity, another advantage of subnets is that they facilitate software engineering. Tasks can be partitioned, with one developer working on one net or a set of nets for a particular subsystem. Communication with other subsystems may take place via only one subnet allowing a degree of modularity[1].

2.3 Gross Topology

CABot3 can be divided into a series of subsystems each consisting of subnets (Figure 1. Arrows show directed connections from one subsystem to another, each, aside from the game, representing a large number of synapses. Verb learning is not tested in CABot3, thus the connection is represented with a dotted line and is omitted in later diagrams. Also, for clarity in later diagrams, due to the prevalence of connections from control, connections from the control subsystems to other subsystems are omitted.

The basic subsystems are described below. Section 3.1 describes the game and the control subsystem; the game receives simple commands from the agent. Section 3.2 describes the vision subsystem; 3.3 the planning subsystem, 3.4 the natural language processing (NLP) subsystem, 3.5 verb learning, and Section 3.6 describes the spatial cognitive map learning subsystem. Connections between the subsystems are also described in these sections. Section 4 summarizes the evaluation of CABot3.

3 Subsystems

Each subsystem is explained below, concentrating on those that have not been explained elsewhere.

3.1 Communication, Control and the Game

The game was developed using the Crystal Space [Crystal Space, 2008] games engine. It is a black and white 3D environment with an agent, a user, four rooms connected by four corridors, and a unique object in each room (see Figure 4); the objects were vertically or horizontally striped pyramids or stalactites (down facing pyramids). The agent and user can move around the rooms independently. The game provides the input to the vision system using a dynamically updated picture of the game from the agent's perspective. The user issues text commands as input to the NLP system. The game also has a bump sensor, and this ignites a CA in the fact subnet in the planning system (see Section 3.3) when the agent bumps into a wall. Similarly, the game takes commands from the agent's planning system to turn left or right, or move forward or backward.

The control subsystem consists of one subnet, the control subnet, which in turn consists of five orthogonal CAs[2]. These CAs mark the state of the agent, either parsing or clearing a parse, setting a goal or clearing it, or a stub. The initial state is turned on at agent start up, and one state is always on.

In the first state, the system is waiting for input or parsing a sentence. This state has connections to most of the NLP subnets to facilitate the spread of activation. When the last grammar rule ignites, it forces the control state to move on.

Most of the CAs involved in parsing, and planning, and all of the control CAs are orthogonal oscillators. When active, they oscillate from having one half of the neurons firing to having the other half firing, then back to the first set. This allows the CA to avoid fatigue as its neurons only fire half the time. This is not biologically accurate, but enables precise behaviour with relatively few neurons.

When it has finished parsing, control moves to the clear parse state. This changes the instance counters in the NLP subsystem preparing it for the next sentence. After a few steps, activation accumulates in the set goal state causing it to ignite, and suppress the clear parse state.

In the third state, the goal in the planning system is set from the semantics of the parse via the intermediate goal set subnet. In the fourth state, information is cleared from the NLP system after the goal is met, and the fifth is a stub.

The control allows the system to parse while still processing a goal. Vision remains active at all times.

[1]Note this modularity may conflict with actual brain topology.

[2]A neuron in an orthogonal CA belongs to that and only that CA.

3.2 Vision

The visual system of CABot3 consists of six subnets: visual input, retina, V1, gratings, V1Lines, and object recognition. The retina, V1, gratings, and V1Lines share some similarities with their human counterparts, but are much simplified models. Higher-level object recognition in CABot3 is not biologically plausible and does not mimic known mechanisms in the human visual system. It does however carry out two important functions of the visual system: the simultaneous identification of *what* is seen and *where* it is in the visual field.

The visual input, retina, V1 and object recognition nets have been described elsewhere and are only slightly modified [Huyck *et al.*, 2006]. The most important modification is the addition of *grating cells* that mimic known properties of the primate visual system, in that they respond selectively to textures of a certain orientation and frequency [DeValois *et al.*, 1979].

The visual input subnet is a 50x50 network of FLIF neurons that do not fatigue. Input to this subnet is clamped to the external stimulus, thus activation is constant until the agent's point of view changes. Each neuron in the 50x50 subnet corresponds to an identically located "cell" in a 50x50 grid of light levels from the environment.

The CABot1 retina subnet contains six 50x50 grids of FLIF neurons. Each subnet contains retinotopic receptive fields of a single size and polarity: 3x3 receptive fields with single-cell centre; 6x6 receptive fields with a 2x2 cell centre and the 9x9 receptive fields with a 3x3 cell centre. For each of these sizes there is a subnet with an on-centre/off-surround polarity (neurons fire when the centre of the receptive field is stimulated and the surround is not) and an off-centre/on surround polarity.

In the V1 area of the human visual system there are neurons, known as simple cells, that are tuned to specific edge and angle orientations. These simple cells are location specific. In the CABot3 V1 and V1Lines subnets, FLIF neurons have been connected to replicate this behaviour. V1 and V1Lines were split for engineering convenience. Weighted connections feed activation from on-centre and off-centre cells in the retina subnet. There are eight orientation specific edge detectors and four angle detectors.

The edge detectors in V1Lines also have recurrent connections to grating detector subnets. Grating detector cells identify repeated patterns of edges of a given orientation and frequency. These grating detectors allow CABot3 to recognise textures in the environment. This allows CABot3 to distinguish between objects of the same shape but that are 'painted' with different textures.

The object recognition net is the least biologically plausible of the visual subnets. There are five modules in the subnet, made up of a number of overlapping cell assemblies. These specialise to recognise pyramids, stalactites, door jambs, doors, or unknown objects. The same modules also carry the "where" (position) as each subnet is a retinotopic representation of the visual field.

3.3 Planning

The planning system is basically a Maes net [Maes, 1989]. The gross topology is shown in Figure 2. All subsystems link

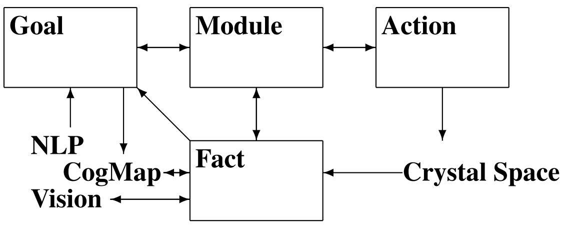


Figure 2: Gross Topology of the Planning Subsystem. Boxes represent subnets.

to the planning subsystem. Its primary entry point is from the NLP subsystem, which sets the goal. The primary outcome is to the game; the CAs in the action subnet are polled and a symbolic command is emitted to the game.

This subnet structure was used throughout CABot1, 2 and 3, and a simple example is the command, *Move forward.* When parsing is completed, the control subnet in combination with the NLP subnets cause an orthogonal oscillating CA in the goal net to ignite. This is equivalent to a goal being set in the Maes net. With a simple action, this goal CA causes the corresponding module subnet CA to ignite, which in turn causes the corresponding CA in the action subnet to ignite. The action CA is then polled to emit the command to the game. Backward inhibition extinguishes the goal and module CAs, and accumulated fatigue causes the action CA to stop.

Simple movements do not require any facts, but actions are often predicated on facts that are set by the environment. For example, an environmentally sensitive command is *Turn toward the pyramid.* In this case, the vision system ignites a fact CA expressing the target's location in the visual field, for instance, "target on left". The combination of activity from the fact net and the goal net cause the appropriate module CA to ignite, which in turn causes the appropriate action CA to ignite. This is an example of needing two (or more) CAs ignited to ignite a third. This is done by allowing the activation of the neurons in the third CA to rise, but which is below threshold when one CA is ignited. The second CA then provides enough activation to ignite the third CA.

Note that the full Maes net has a concept of Maes module activation. In CABot3, the module CAs are either on or off, and there is no activation level (but see Sections 3.4 and 5).

The system executes 21 commands, four primitives (e.g. *Turn right*), two compounds (e.g. *Move left* which executes a left then forward), turn toward pyramid or stalactite, go to seven objects, explore, stop, and move before four objects. The seven objects are door, and pyramid or stalactite either (vertically) barred, (horizontally) striped, or unspecified.

Moving to an object may require several steps. CABot3 centres the object in the visual field and then moves to it until the object fills the visual field, possibly centring again along the way. Any command can be stopped by the *Stop* command.

The most sophisticated thing the system does, in response to the *Explore* command, is to explore the four rooms and memorize the objects in the room (see Section 3.6). To test that the system has correctly memorized the map, a command such as *Move before the striped pyramid* may be used. The system then moves to the room before the striped pyramid and stops without having seen it again, showing it has memorized its location (see Section 4.1).

In all, the goal subnet contains 26 CAs, including subgoals. The fact subnet has 66 CAs, the module subnet seven, and the action subnet six including two error conditions.

3.4 Natural Language Processing

The stackless parser has been described elsewhere [Huyck, 2009]. Input is provided symbolically from Crystal Space, each word is associated with an orthogonal set of neurons in the input net, and they are clamped on when the particular word is being processed.

The subnets involved follow Jackendoff's Tripartite theory, with NLP broken into three main systems, lexicon, syntax and semantics, and the systems communicate via subsystems.

Stackless parsing is done by activation levels, with the number of neurons in a CA firing in a cycle reflecting CA activity. In practice this is done by a tightly specified topology that has the number of neurons firing in the CA decaying over time; activation levels reflect the order of items.

Semantics are handled by overlapping encoding derived from WordNet. This could be useful in resolving parsing ambiguities, though this is not implemented in CABot3.

Grammar rule CAs are selected by activation of component (lexical or higher order category) CAs. Variable binding is done with short-term potentiation [Hempel *et al.*, 2000], and this is how instances store their semantics. Noun instances represent noun phrases and verb instances, verb phrases including their arguments. A case frame is generated for each parse, and the slots are bound to other instances or to the semantics of words. These bindings are learned but decay over time. The next time they are used, two parses later, the instance frames have been erased by automatic weight decay.

3.5 Motivation and Reinforcement Learning

Hebbian learning strengthens the connections between CAs as well as within a CA. CAs are associated with some atomic propositions, and more complex propositions (such as implication rules) are represented by groups (e.g. pairs) of associated CAs. However, Hebbian rules do not differentiate between learning 'good' or 'bad' propositions. After several atomic propositions or symbols have been learnt in the form of corresponding CAs, the main problem is to learn the correct or favourable propositions from these.

This problem was solved by a motivational system that is used to control Hebbian learning so that propositions with higher utility values or rewards are reinforced [Belavkin and Huyck, 2008]. The mechanism uses two specialised subnets: utility and explore. Neurons in the utility network output signals corresponding to a reward or payoff obtained from the environment. Neurons in the explore network output signals that represent random noise and they can be connected to any set of CAs that needs to be randomised to allow stochastic exploration of their interrelations. The utility network has inhibitory connections to the explore network so that high values of utility correspond to low level of randomness at the output of the explore network.

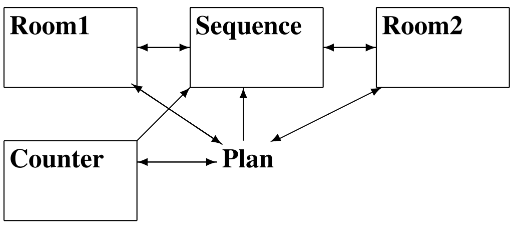


Figure 3: Subnets involved in spatial cognitive mapping.

It has been demonstrated previously that the mechanism described above can be used to learn simple sets of rules in a CA-based architecture [Belavkin and Huyck, 2008], and that it can be used to model probability matching observed in animals and people [Belavkin and Huyck, 2010]. The mechanism was used by CABot2 to learn the verb *centre* and the corresponding action associated with a visual stimulus. It is unplugged in the currently available version of CABot3.

3.6 Spatial Cognitive Map Learning

Spatial cognitive mapping is the psychological process of recording, recollecting and acting on locations and objects in a physical environment [Downs and Stea, 1973]. CABot3 implements a simple version of this complex process based on the authors' previous work [Huyck and Nadh, 2009]; the CABot3 agent explores the rooms, learns the objects, associations between them, and navigates to specific rooms.

Figure 3 shows the subnets involved. Room1 and room2 encode adjacent rooms that the agent moves through, where room1 is the prior room and room2 is the current room. The sequence net encodes the associations between the rooms, and the objects in them. The counter net supports the order.

On receiving the *Explore* command, the agent goes around the environment, room by room, learning the objects it sees. When an object is in its visual field, for instance a *striped pyramid*, the current room in association with it is encoded as a CA in Room1. The object in view is recognised from activity in the fact net, and learning lasts 200 cycles as it has been observed to be the minimum number of cycles required for CAs to be learnt. When the agent moves to the next room, the same routine happens, but as it has come from an adjacent room, the current room is also encoded in room2. The previous room CA in room1 is still active, the current room CA in room2 ignites, and the association between the two rooms learnt as a CA in the sequence net. Learning in the sequence subnet happens via co-activation with the two active room CAs in the two room nets lasting 200 cycles. This in essence creates individual CAs representing the rooms and their constituent objects in the two room nets, and the association between the rooms the agent passes through in sequence. Counter keeps track of the room the agent is currently in. When the agent is done exploring, room1 and room2 have a CA associated with the item in the fact net, and the sequence net has five CAs representing the association between each room and its adjacent room.

After exploration, when the agent is issued with a command such as *Move before the striped pyramid*, the involved fact such as "striped pyramid" ignites in fact (Figure 2). Fact in turn ignites the learnt CA in room2 representing the room with the "striped pyramid". As the sequence net has encoded the association between rooms, the active CA in room2 activates the associated room in room1, which is the room before the room in room2 that the agent entered through while exploring. Thus the agent *deduces* the target room from its simple learnt cognitive map. With the target room active, the agent starts moving, and when it reaches the target room, activity in the goal subnet informs it of task completion.

4 Evaluation

The evaluation of a CABot3 agent is a complex process. Many of the components have been evaluated separately. For the purposes of testing CABot3 itself, parsing, for example, consists of a few dozen grammar rules that it uses to parse all of the acceptable commands correctly, so as to set an appropriate goal. In parsing, all of the connections are deterministic, and the parsing subnets are insulated by layers of connections from the more stochastic areas.

The evaluation of the planning system and cognitive mapping systems are briefly described in Section 4.1. The control system is a simple finite state automata which switches states when other systems reach certain states, for example when the parser finishes, the control state changes. This system largely switches states when appropriate, but occasional errors do occur, but these are largely self correcting. However, it occasionally gets into states from which it cannot recover.

The vision system works robustly for a limited range of textures. There are two orientations and a limited range of spatial frequencies that the grating cells can accommodate due to the size and resolution of the retinal nets. Within these limitations, however, the system identifies textures reliably. Where objects are presented clearly on the retina (that is, where the viewing angles are not extreme) the visual system robustly identifies the objects in the 3D world.

4.1 Explore Evaluation

The planning system is responsible for a relatively wide range of activities. Most of these it performs entirely correctly; for example the command Turn left. always works correctly. The most sophisticated physical task the agent performs is to explore all of the rooms, making use of vision and spatial cognitive mapping (see Section 3.6). This exploration is relatively simple though it can take several hundred moves. An example is shown in Figure 4.

CABot3 initially tries to identify the room it is in by the unique object it sees. In the case of Figure 4, it sees the striped pyramid, and this is put into its spatial cognitive map. It then finds the corridor, which it can see at a distance. It moves to the front of the corridor keeping to the left edge, stopping when it bumps into the edge of the corridor. It then turns right and moves through the corridor along the edge. At the end of the corridor it turns right to see the object in the next room. It can see there is an object but the agent is not close enough to identify it. It moves toward the object, in this case the barred pyramid, until it can identify it. It then puts that in the cognitive map, and repeats the process for the next two rooms, stopping when it identifies the object in the initial room.

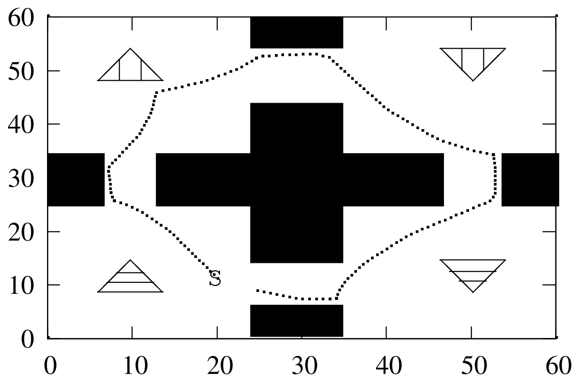


Figure 4: Forward moves of CABot3 while exploring the rooms, starting at **S** with moves marked by dots.

Explore works about half the time. It appears cognitive mapping works each time, and all of the failures are due to navigation problems.

4.2 Subnet Evaluation

The subnet topology is important both for software engineering and for relating to brain areas. From the software engineering perspective, the method has been successful. Breaking the full network into subnets has enabled development of systems to be partitioned with one developer working on one task, (e.g. vision) in isolation. The systems have then been combined to work together in the full CABot3 agent.

The brain did not evolve this way, so it is also important to see how different subnets might map to brain areas. There is a strong correlation between CABot3's early vision areas and biological vision areas, with both accounting for similar behaviour. There is a looser correlation between the explore subnet in reinforcement learning and the basal ganglia. However, in most cases the subnets have little correlation with brain areas. None the less, the basic subnet topology could be used to closely mimic known brain area topology and behaviour. As subnets still have connections to and from other subnets, so CABot3 is one large network.

5 Conclusion

Many researchers thought that implementing AI systems with simulated neurons was too complex (e.g. [Smolensky, 1988]). Perhaps this was true a few decades ago, but the authors believe that CABot3 shows that this fear has passed.

The mere implementation of a relatively simple agent may miss the point that many connectionists hope to make: that the neural level is not the correct level to study the brain. While the authors would agree that many complex behaviours, such as attractor dynamics and supervised learning, are being effectively studied with non-neural connectionist systems, this does not mean that the same problems cannot be effectively studied in neural systems.

Moreover, simulated neural systems have an important advantage over connectionist systems when it comes to studying AI: existing intelligent agents (humans and other animals) use neurons to think, and the neural and cognitive behaviour of these animals is being studied. Simulated neural systems, which match sensible intermediate behaviour, can be developed as milestones on the way to full fledged AI systems.

During the project, it was shown that in general a network of CAs, and in particular a network of FLIF neuron CAs, was Turing complete [Byrne and Huyck, 2010]. In some sense, this makes the implementation of CABot3 unsurprising. While CABot3 is obviously not a neuron by neuron simulation of a human brain, it does have a series of links to neurobiological and cognitive behaviour that increase its validity. The base neural model is a relatively accurate if simplified model of neurons. In CABot3, some subnets are reasonable approximations of brain areas. The use of CAs for long and short-term memories and as the basis of symbols is neuropsychologically supported, and provides a bridge between subsymbolic and symbolic processing. Cognitive models provide solid links to psychological behaviour from a neural system.

While it is possible to continue to program new and improved neural systems, the authors believe the key is to have the system learn its behaviour. Thus, a vast range of future work is possible such as: improving existing systems; adding new sensory modalities, for example sound detection and speech recognition; moving from virtual to physical robots; improving the fit with biological data, for example more neurons, more realistic topologies, and more accurate neural models; new and more sophisticated cognitive models; and improving computation, for example by use of specialised neural hardware. Simulated CAs themselves could also be improved so that a single CA could be learned, and persist for an appropriate duration. More radical improvements also present themselves including improved learning, for example at the CA level and in combination with variable binding, improved understanding of dual attractor dynamics, integration of attention, and experiments with agents that continue to improve over several days or longer.

CABot3 is an agent in an environment functioning in real time, implemented in simulated neurons. It is a solid step in the development of agents implemented in simulated neurons, and it is intended that more sophisticated agents will be derived from it. Building systems like this will involve trade offs between biological and psychological fidelity, and computational constraints. By building more biologically and psychologically plausible systems that perform more tasks, significant advancements in the understanding of general cognition can be made.

Acknowledgements:
This work was supported by EPSRC grant EP/D059720.

References

[Amit, 1989] D. Amit. *Modelling Brain Function: The world of attractor neural networks*. Cambridge University Press, 1989.

[Belavkin and Huyck, 2008] R. Belavkin and C. Huyck. Emergence of rules in cell assemblies of fLIF neurons. In *The 18th European Conference on Artificial Intelligence*, 2008.

[Belavkin and Huyck, 2010] R. Belavkin and C. Huyck. Conflict resolution and learning probability matching in a neural cell-assembly architecture. *Cognitive Systems Research*, 12:93–101, 2010.

[Byrne and Huyck, 2010] E. Byrne and C. Huyck. Processing with cell assemblies. *Neurocomputing*, 74:76–83, 2010.

[Crystal Space, 2008] Crystal Space. $http://www.crystalspace3d.org/main/main\_page$. 2008.

[DeValois *et al.*, 1979] K. DeValois, R. DeValois, and E. Yund. Responses of striate cortex cells to grating and checkerboard patterns. *The Journal of Physiology*, 291(1):483–505, 1979.

[Downs and Stea, 1973] Roger M. Downs and David Stea. *Cognitive maps and spatial behaviour. Process and products*, pages 8–26. Aldine, Chicago, 1973.

[Hebb, 1949] D. O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. J. Wiley & Sons, 1949.

[Hempel *et al.*, 2000] C. Hempel, K. Hartman, X. Wang, G. Turrigiano, and S. Nelson. Multiple forms of short-term plasticity at excitatory in rat medial prefrontal cortex. *Journal of Neurophysiology*, 83:3031–3041, 2000.

[Hodgkin and Huxley, 1952] A. Hodgkin and A. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

[Huyck and Byrne, 2009] C. Huyck and E. Byrne. CABot1: Technical report. Technical report, Middlesex University, 2009.

[Huyck and Nadh, 2009] C. Huyck and K. Nadh. Multi-associative memory in fLIF cell assemblies. In R. Cooper (Eds.) A. Howes, D. Peebles, editor, *9th International Conference on Cognitive Modeling - ICCM2009*, pages 81–87, Manchester, UK, 2009.

[Huyck *et al.*, 2006] C. Huyck, D. Diaper, R. Belavkin, and I. Kenny. Vision in an agent based on fatiguing leaky integrate and fire neurons. In *Proceedings of the FifthInternational Conference on Cybernetic Intelligent Systems*, 2006.

[Huyck, 2009] C. Huyck. A psycholinguistic model of natural language parsing implemented in simulated neurons. *Cognitive Neurodynamics*, 3(4):316–330, 2009.

[Huyck, 2011] C. Huyck. Parameter values for FLIF neurons. In *Complexity, Informatics and Cybernetics: IMCIC 2011*, 2011.

[Jackendoff, 2002] R. Jackendoff. *Foundations of Language: Brain, Meaning, Grammar, Evolution*. Oxford University Press, 2002.

[Maes, 1989] P. Maes. How to do the right thing. *Connection Science*, 1:3:291–323, 1989.

[Smolensky, 1988] P. Smolensky. On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1:1–22, 1988.

Compact Crossbars of Multi-Purpose Binders for Neuro-Symbolic Computation

Work in Progress Report

Gadi Pinkas
Center for Academic Studies, Israel
gadip@mla.ac.il

Priscila M. V. Lima
Universidade Federal Rural do Rio de Janeiro
Seropédica, Brazil
priscilamvl@ufrrj.br

Shimon Cohen
Center for Academic Studies, Israel
shamon51@gmail.com

Abstract

We present a compact – yet expressive – Multi-purpose, distributed binding mechanism, which is useful for encoding complex symbolic knowledge and computation, using Artificial Neural Networks (ANNs) or using Satisfiability (SAT) solvers. The technique is demonstrated by encoding unrestricted First Order Logic (FOL) unification problems as Weighted Max SAT problems and then translating the later into ANNs (or learning them). It is capable of capturing the full expressive power of FOL, and of economically encoding a large Knowledge Base either as long term synapses or as clamped units in Working Memory. Given a goal, the mechanism is capable of retrieving from the synaptic knowledge just what is needed, while creating novel, compound structures in the Working Memory. Two levels of size reduction are shown. First, we build a Working Memory, using a pool of multi-purpose binders, based on the assumption that the number of bindings that are actually needed is far less than the number of all theoretically possible bindings. The second level of compactness is due to the fact that, in many symbolic representations, when two objects are bound, there is a many-to-one relationship between them. This happens because, frequently, either only one value is pointed by variable or only one variable point to a value. A cross-bar binding network of n × k units with such restriction, can be transformed into an equivalent neural structure of size O(n log(k)). We show that, for performing unrestricted FOL unifications, the Working Memory created is only log dependent on the KB size; i.e., O(n log(k)). The variable binding technique described is inherently fault tolerant as there are no fatal failures, when some random neurons become faulty and the ability to cope with complex structures decays gracefully. Processing is distributed and there is no need for a central control even to allocate binders. The mechanism is general, and can further be used for other applications, such as language processing, FOL inference and planning.

1 Introduction

1.1 The Binding Problem

Human cognition is capable of producing combinatorial structures. The general binding problem concerns how items that are encoded in distinct circuits of a massively parallel computing device (such as the brain or ANN) can be combined in complex ways for perception, reasoning or for action [Feldman 2010]. Consider for example, a planning problem, where the task is to pick up an object and move it from its current position to another place. In order to meet a goal, a "brain"-like device, must be able to represent the object, its properties, its position and the ways to manipulate it, in such a way that the goal is achieved. The object and its properties must be bound together, and this rather complex structure should also be used in conjunction with other entities and rules, such as the action consequences (e.g., moving X from Y to Z clears position Y while occupying position Z). In another example, consider the sentence: "Sally ate": In language processing, the verb "EAT" is a predicate with at least two roles - EAT("Sally",X). The noun "Sally" should be bound to the first role, while an existentially quantified variable (representing "something") should be bound to the second role. Once we get the information that "Sally ate salad", and knowing the rule: EAT(Y,X)⇒DIGESTED(X) we should reason that "the salad is digested". In order to do that, we must bind the variable X to the noun "salad", while X must be bounded to both EAT(,X) and DIGESTED(X).

1.2 Connectionism and Variable Binding

During the years, connectionist systems have been criticized for "Propositional Fixation" [McCarthy 1988]. In [Fodor, Phylyshyn 1988] connectionism was criticized for lacking abilities to construct combinatorial representations and for performing processes that are sensitive to complex structure. Exactly how compositionality can occur is a fundamental question in cognitive science and the binding aspect of it has been identified as a key to any neural theory of language [Jackendoff 2002]. Several attempts have been made to approach the variable binding problem in a connectionist framework [Shastri, Ajjanagadde 1993], [Browne, Sun 2000], [Zimmer et al. 2006], [Van der Velde, Kamps, Kamps 2006], [Barret et al. 2008], [Velik 2010]; yet, virtually all these suggestions, have limitations, related to either limited expressiveness, size and memory requirements, central control demands, lossy information, etc.

For example, compositionality can be provided using Holographic Reduced Represenations [Plate 1995]; however, the convolution operation used, is lossy and errors are introduced as structures become more complex or as more operations are done. The BlackBoard Architecture [Van der Velde, Kamps, Kamps 2006] can form complex structures but does not manipulate those structures to perform cognition. Shastri's temporal binding has only limited FOL expressiveness and no mechanism for allocating temporal binders. Finally, all the above systems need neurons in numbers that is at best linear in the KB; while some use much more neurons than that.[1] For FOL compositionality in ANNs see [Ballard 1986], [Pinkas 1992], [Shastri 1999], [Lima 2000], [Garcez, Lamb 2006]. For partial-FOL encodings in Satisfiability, see [Domingos 2008], [Clark et al. 2001].

The ability to represent combinatorial structures and reasoning with them, still presents challenges to theories of neurocognition [Marcus 2001], while the variable binding problem is fundamental to such ability [Feldman 2010].

1.3 Unification

In conventional computing, unification is a key operation for realizing inference, reasoning, planning and language processing. It is the main vehicle for conventional symbolic systems to match rules with facts, or rules with other rules. In unification, two or more distinct hierarchical entities (terms) are merged, to produce a single, unified, tree-like structure. This unified structure adheres to the constraints of both the original entities. Formally, unification is an operation which produces from two or more logic terms, a set of substitutions, which either identifies the terms or makes the terms equal modulo some equational theory. For connectionist approaches to unification see [Hölldobler 1990], [Weber 1992], [Komendantskaya 2010].

For easiness of reading, we have chosen to demonstrate our compact variable binding mechanism on the more fundamental unification function, rather than on full FOL inference.

1.4 Artificial Neural Networks and SAT

ANNs may be seen as constraint satisfaction networks, where neuron-units stand for Boolean variables, and where the synapse weights represent constraints imposed on the variables. Any ANN may be seen as such a constraint network; yet, for ANNs with symmetric weights (e.g. Hopfield, Boltzmann Machines, MFT) a simple conversion has been shown for translating any Weighted MAX SAT problem into symmetric ANN and vice-versa [Pinkas, 1991]. Any such SAT problem could be compiled into an ANN, which performs stochastic gradient descent on an energy function that basically counts the number of unsatisfied logical constraints. The size of the generated network is linear in the size of the original formula, though additional hidden units may be required. In addition to compilation, the logical constraints of a network could be PAC learnt using Hebbian-like rule [Pinkas 1995], thus, for small-size constraints, a network can efficiently learn its weights and structure from a training set that is composed of the satisfying models. The performance efficiency of this neural mechanism can be attributed to the similarities of symmetric ANNs to stochastic local search algorithms, such as WALKSAT [Kautz et al 2004]. Due to the tight relationship between ANNs and Weighted Max SAT, our methodology is to specify an ANN designed for certain symbolic computation (e.g. unification), using a set of Boolean variables (the visible units) and a set of constraints; i.e., Boolean formulae designed for restricting the values of the visible units. The constraints specified are used to force the visible units to converge to a valid solution that satisfies as many (weighted) formulae as possible. We have written a compiler that translates such specifications into either weighted CNF (for Weighted Max SAT Solvers) or for ANN with symmetric weights.

We believe that our fault tolerant mechanism and methods for dynamically forming recursive structures will scale and be useful for both the engineering of massively parallel devices, and for modeling of high-level cognitive processes.

2 Improving CrossBar Binding

The simplest, most naïve binding techniques is CrossBar binding. The term was mentioned in [Barrett et al. 2008], yet it was intuitively used by many connectionist systems in the past [Ballard 1986], [Anandan 1989] and in many SAT reductions; e.g., [Kautz, at el 2006]. Formally, we define crossbar binding as a Boolean matrix representation of a relation between 2 sets of items, using Characteristic Matrix of the relation; i.e., if A contains m objects and B contains n objects, then the characteristic matrix R has m lines and n columns, containing m × n Boolean variables (neurons). We say that *item i is bound to item j iff R(i,j)=1.* In this naïve, binding mechanism, a neuron should be allocated for each possible binding, and all theoretic combinations of two items must be pre-enumerated as rows and columns of the matrix. A crossbar matrix, that needs to represent a complex tree or a graph, must bind together not just simple constituents, but all the compounded entities representing partial trees (or sub-graphs). It is possible to represent a FOL KB this way at the cost of using an enormous number of neurons, and with an extremely localist approach. Even more frustrating is the fact that this technique will not be suitable for dynamically creating novel, nested structures upon demand. The number of theoretic bindings, for all possible tree

[1] The BlackBoard architecture uses billions of neurons to represent thousands of atomic concepts; HRR Production systems [Stewart, Elliasmith 2008] needs about one million neurons.

structures, grows exponentially with the number of constituent items and must be computed in advance.
We improve this simplistic binding mechanism in several steps:

2.1 Using Binders as "pointers" to form Graphs

First, we introduce[2] a special kind of entities called General Purpose Binders (GPBs). GPBs are similar to pointers except for the fact that a single GPB can point to several objects, as the crossbar paradigm permits implementation of arbitrary relations between binders and objects. In the special case where binders point to binders, arbitrary directed graphs can be built. In this scenario, we can interpret each GPB as a node in the graph, and the crossbar, as specifying the arcs of the graph (adjacency matrix). In such graph interpretation, each node may be labeled using a labeling crossbar, that ties together binders, with symbols such as, predicates, functions or constants in FOL. Arcs can also be labeled, as the binder-to-binder crossbar, may have a third dimension which relates one or more labels to each arc. This enables the formation of arbitrary complex graph structures, that can be used to represent language constituents and in particular, FOL terms, predicates, literals and clauses. Unlike in the naïve crossbar approach, unrestricted graphs can be built directly out of simple constituents, with GPB as the mechanism for gluing them together.

Because the binders are general-purpose entities, we can construct a working memory out of a pool of such binders. As long as GPBs remain unallocated, they can be used for dynamic creation of novel, goal oriented structures. To do so, the "right" constraints should be embedded in the synapses, forcing binders first to be allocated and then to assume a desired structure for solving the goal. These constraints, stored at the synaptic weights, are the driving force that causes the visible units to converge to the needed graph-like structures.

Using this technique, we show that arbitrary KB of size k, can be encoded in a working Memory (WM) with O(k) binders and with a total size of $O(k^2)$. Unfortunately, when the KB tends to grow, the WM and the set of constraints may become too large for the mechanism to be used in real applications.[3]

2.2 Using a pool of binders "As Needed"

Luckily, we can reduce that size requirement, drastically, as we can assume that, at a certain time, only few binders are actually needed for the processing of a given goal. This is supported by cognitive studies [Cowan 1981] and constitutes a common assumption of several connectionist systems [Shastri, Ajjanagadde 1993], [Barrett et al 2008]. We therefore can design a Working Memory of neural units, which uses only a pool of General Purpose Binders, labeled and nested within each other; i.e., a small set of binders, for representing only those graphs that are actually needed for computing the goal. It turns out that this approach is consistent with cognitive theories, where a large KB is stored in synapses (long term memory); and a smaller size working memory is used for retrieving only few KB items at a time. Only those items that are necessary to the process[4] get to be retrieved from the synaptic KB. For example, if our purpose is to find a plan for a goal, expressed in FOL, we need to design the WM with enough binders to represent a valid plan. We retrieve the facts and rules of the world from that KB only if they are required by the plan we desire to make.

To implement a pool of binders for FOL unification, the WM should contain three crossbar matrices: One for labeling nodes by symbols (predicates, functions, constants). The second is for nesting of the nodes in Graphs and labeling the arcs according to slots of the predicates and functions. The third crossbar is for retrieving items from the long term memory where the KB is stored (e.g., terms, literals or clauses). This third matrix ties a binder to a KB item and triggers the constraints of that item to be activated so that the binder node is forced to assume the structure of the KB item retrieved. The mechanism starts working as goal activated constraints cause some binders to be tied to KB items and activate some KB constraints. Those constraints, in turn, activate other constraints, till the WM converges to a valid solution. When we implement unification problems, the size of the WM is $O(n \times k)$ where n is the maximal number of nodes in a solution; k is the size of the KB and n<<k. This constitutes a drastic improvement, as the WM size is linear in the size of the KB, instead of being quadratic.[5] Actually, we can do even better:

2.3 Crossbars with n*log(k) size complexity

In the next size improvement, we further reduce the size of many crossbar matrices from O(n*k) to O(n* log(k)). Thus, in our unification example, a WM of O(n* log (k)) is created, where n is the maximal size of a unification tree and k is the size of the KB. This means that the WM size is only log dependent[6] on the KB size; rather than linearly as in previous section.

The key to this log-reduction, is the fact that frequently, binding relationships have *many-to-one* or one-to-many restrictions. For example, the crossbar matrix for node labeling, allows for a binder to point only to a single symbol (whereas many binders could point to the same symbol). This *many-to-one* relationship causes the rows of the crossbar labeling matrix to be *Winner-Takes-All* (WTA) arrays, where only one neural unit (if any) may fire. Normally, we need mutual exclusion constraints to force the rows of the matrix to be either all-zeros or have a single variable set to one. In such a scenario, however, we can replace each WTA

[2] The method was suggested in [Pinkas 1992] and used in [Lima 2000], [Lima 2007] for clamping a KB in Working Memory.
[3] $O(k^3)$ constraints are needed for unification in this paradigm.

[4] When an item is already in WM, no retrieving is needed.
[5] The number of *constraints* needed for unification is $O(n^2k)$; linear in the KB size, when n<<k.
[6] Even, if occurs check is used, the WM size is still linear in the KB size when n<<k.

line (with k-variables), with a much smaller size line of only O(log(k)) variables. Each such line of log(k) variables (neurons), represents an index (or a signature) to the target label. Therefore, if a binder may point to just a single object (out of k possible objects), we may use only log(k) bit signatures. Fig 1 illustrates, how one binder with WTA line that points to object 6 (out of 15 objects) is reduced to only 4 bits LOG WTA array, representing the signature of that item. This signature, once it emerges in a binder's row, activates a set of constraints associated with the bounded object. These constraints force the binder to get the retrieved item's structure and may cause a chain reaction of more constraints, retrieving more KB items and so forth.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

1	2	3	4
0	1	1	0

Figure 1. On top: A Standard WTA pointing to the 6th object; Bellow: a LOG WTA array with a binary value of 6 representing the 6th object's signature '0110'.

It should be noted that, once a LOG WTA encoding is used instead of the standard WTA, the constraints imposed on WM might need to be adjusted. [7]

3 Fault Tolerance

The variable binding mechanism suggested and its application to unification are inherently fault tolerant, if each variable is allocated a processing unit (a neuron). Small random damage to the neurons does not radically affect the unification process (if at all). For example, if a single neuron related to a binder, in one of the crossbar matrices, becomes faulty and stops firing, then the binder cannot point to a certain symbol; however, other binders from the pool can be used for pointing to that symbol if such is needed. In the meantime, this "faulty" binder may still be used, as it can be allocated to point to other symbols. Even if the faulty neuron starts firing constantly, it may still participate in the process if the symbol that is pointed by that "faulty" binder happens to be needed. The binder will simply not be used, if that symbol is irrelevant to the goal. If the damage to the WM neurons is more widespread, so that a binder cannot take part in the process, then this binder will not be allocated, and therefore will not be used in the graph construction. This may shorten the number of available GPB nodes in the largest graph but will not destroy the ability of the WTA to unify less complex terms (shallower trees).[8]

4 Conclusions

We have shown a general purpose binding mechanism that uses a pool of general purpose binders, and allocates them to KB items, only when they are necessary for achieving the goal. A large KB may be stored in long term connections rather than in the Working Memory. KB constraints are activated only upon need, and only if they are supportive for achieving the goal. We then showed that further log reduction is possible if the binding represents a many-to-one relationship. The size of a crossbar matrix is then reduced from O(n*k) to O(n*log(k)) and the number of constraints is also reduced.[9] We demonstrated the use of the suggested binding technique in ANN that performs FOL unification with size[10] that is O(n×log(k)). The mechanism is distributed since there is no central control and even binder allocation is done in a totally distributed way. It is also inherently robust, as no fatal failures occur when neurons "die". We have performed initial experiments with the GPB pool mechanism (without the LOG WTA reduction), these experiments indicate the feasibility of the approach on rather complex unification tasks including multi-instance parallel-unification and recursive occurs checking. LOG WTA and fault tolerance experiments are the subject of ongoing work. The mechanism described is general and can further be used for other applications such as: language processing, FOL inference and planning. We are working on extending the techniques, for full FOL inference and conjecture that these techniques will also improve other SAT encodings that use crossbar-like bindings, e.g. as in [Kautz, et al 2006].

References

[Ballard 1986] D. H. Ballard. Parallel logical inference and energy minimization. In Proceedings of the AAAI National Conference on Artificial Intelligence, pages 203–208, 1986.

[Barrett et al 2008] Barrett L, Feldman JA, Mac Dermed L. A (somewhat) new solution to the binding problem. Neural Computation, 20: 2361-237, 2008.

[Brown, Sun 2000] A. Browne and R. Sun. Connectionist variable binding. Springer, Heidelberg, 2000.

[Clarke et.al 2001] E. Clark, A. Bier, R. Raimi, Y. Zhu, Bounded Model Checking Using Satisfiability Solving, in Formal Methods in System Design archive, Volume 19 Issue 1, July 2001.

[7] E.g., mutual exclusion constraints - for enforcing WTA are eliminated. Long OR constraints of O(k) size, become only of log(k) length.

[8] This property may help in supporting neuro-linguistic theories that relate certain symptoms of aphasia, with loosing abilities for processing deep structures; e.g. deep syntactic tree pruning in Agrammatism and Broca's Aphasia [Friedman 2002].

[9] Note, that some constraints may grow in size complexity; while others may shrink.

[10] O(n^2 +log(k)) with occurs check.

[Cowan 2001] N Cowan, The magical number of 4 in short term memory: A reconsideration of mental storage capacity, Behavioral and Brain Sciences, 1(24), 2001.

[Domingos 2008] P. Domingos, Markov logic: a unifying language for knowledge and information management. CIKM 2008:519.

[Feldman 2010] J. Feldman, The Binding Problem(s), http://www.computational-logic.org/content/events/iccl-ss-2010/slides/feldman/papers/Binding8.pdf, accessed on May 2010.

[Fodor, Phylyshyn 1988] J. A. Fodor and Z. W. Phylyshyn, Connectionism and cognitive architecture: A critical analysis. In Pinker and Mehler (eds): Connectionism and Symbols, 3-71, MIT Press, 1988.

[Friedman 2002] Friedmann, N.Syntactic tree pruning and question production in agrammatism. Brain and Language, 83, 117-120

[Garcez, Lamb, 2006] A. S. d'Avila Garcez and L. C. Lamb. A Connectionist Computational Model for Epistemic and Temporal Reasoning. Neural Computation 18(7):1711-1738, MIT Press, July 2006.

[[Garcez, Lamb 2006] A. S. d'Avila Garcez and L. C. Lamb. A Connectionist Computational Model for Epistemic and Temporal Reasoning. Neural Computation 18(7):1711-1738, MIT Press, July 2006.

[Hölldobler 1990] S. Hölldobler, A Connectionist Unification Algorithm. International Computer Science Institute, Berkeley, TR-90-012: 1990.

[Jackendoff 2002] Jackendoff, Ray (2002). Foundations of Language: Brain, Meaning, Grammar, Evolution. Oxford: Oxford University Press. pp. 477.

[Kautz et al 2004] Henry Kautz, Bart Selman, & David McAllester, Walksat in the 2004 SAT Competition, International Conference on Theory and Applications of Satisfiability Testing, Vancouver, 2004

[Kautz at el 2006] Henry Kautz, Bart Selman, and Joerg Hoffmann. SatPlan: Planning as Satisfiability, Abstracts of the 5th International Planning Competition, 2006

[Komendantskaya 2010] E. Komendantskaya, Unification neural networks: unification by error-correction learning, Logic Jnl IGPL, 2010

[Lima, 2000] P. M. V. Lima: Resolution-Based Inference on Artificial Neural Networks. Ph.D. Thesis, Department of Computing. Imperial College London, UK (2000).

[Lima et al 2007] P. M. V. Lima, M. Mariela, M. Morveli-Espinoza and F. M. G. França, Logic as Energy: A SAT-Based Approach, Advances in Brain, Vision, and Artificial Intelligence Lecture Notes in Computer Science, 2007.

[Marcus 2001] G. F. Marcus, The algebraic mind. Cambridge, MA: MIT Press, 2001.

[McCarthy 1988] J. McCarthy, Epistemological challenges for connectionism. Behavioral and Brain Sciences, 11:44, 1988.

[Pinkas 1991] G. Pinkas, "Symmetric neural networks and logic satisfiability," Neural Computation 3, no. 2, pp.282-291, 1991.

[Pinkas 1992] G. Pinkas, "Constructing syntactic proofs in symmetric networks" in Advances in Neural Information Processing Systems – 4 (NIPS-91), pp.217-224, 1992.

[Pinkas 1995] G. Pinkas, "Reasoning, non-monotonicity and learning in connectionist networks that capture propositional knowledge", Artificial Intelligence Journal 77, (AIJ) pp. 203-247, 1995.

[Plate 1995] T. Plate Holographic Reduced Representations, IEEE Trans. On Neural Networks 6(3), 623-641

[Shastri, Ajjanagadde 1993] L. Shastri L. and V. Ajjanagadde, From associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings using temporal synchrony. Behavioral and Brain Sciences 1993;16(3):417-494.

[Stewart, Eliasmith 2008] T.C. Stewart, C. Eliasmith, Building Production Systems with Realistic Spiking Neurons", Cognitive Science Conference. Washington, DC. August, 2008

[Velik 2010] R. Velik, From single neuron-firing to consciousness- towards the true solution of the binding problem, Neuroscience Behavioral Rev., 2010, 34(7): pp. 993-1001.

[Weber 1992] V Weber, Connectionist Unification with a distributed Representation, In IJCNN-92

[Zimmer et. Al. 2006], HD Zimmer, A. Mecklinger, U. Lindenberger (eds), Handbook of binding and memory - Perspectives from cognitive neuroscience. Oxford University Press 2006.

Embedding Normative Reasoning into Neural Symbolic Systems

Guido Boella
University of Torino
guido@di.unito.it

Silvano Colombo Tosatto
University of Luxembourg
silvano.colombotosatto@uni.lu

Artur d'Avila Garcez
City University London
aag@soi.city.ac.uk

Valerio Genovese
University of Luxembourg
valerio.genovese@uni.lu

Leendert van der Torre
University of Luxembourg
leon.vandertorre@uni.lu

Abstract

Normative systems are dynamic systems because their rules can change over time.

Considering this problem, we propose a neural-symbolic approach to provide agents the instruments to reason about and learn norms in a dynamic environment.

We propose a variant of d'Avila Garcez et al. Connectionist Inductive Learning and Logic Programming(CILP) System to embed Input/Output logic normative rules into a feed-forward neural network. The resulting system called Normative-CILP(N-CILP) shows how neural networks can cope with some of the underpinnings of normative reasoning: *permissions*, *dilemmas*, *exceptions* and *contrary to duty* problems.

We have applied our approach in a simplified RoboCup environment, using the N-CILP simulator that we have developed. In the concluding part of the paper, we provide some of the results obtained in the experiments.

1 Introduction

In artificial social systems, norms and policies are mechanisms to effectively deal with coordination in normative multi-agent systems. An open problem in AI is how to equip agents to deal effectively with norms (and policies) that change over time [Boella *et al.*, 2009], either due to explicit changes by legislators, or due to the interpretation process by those agents who are in charge of applying the law (e.g, judges).

In the work of [Corapi *et al.*, 2010], they focused on refine existing knowledge about the norms by using inductive learning. Differently in game-theoretic approaches [Sen and Airiau, 2007; Boella and van der Torre, 2006; Shoham and Tennenholtz, 1997], few machine learning techniques have been applied to tackle open problems like learning and/or revising new norms in open and dynamic environments.

In this paper we use Input/Output (I/O) logic [Makinson and van der Torre, 2000], a symbolic formalism used to represent and reason about norms. We study how to represent I/O within the computational model of neural networks, in order to take advantage of their ability to learn, by addressing the following research question:

- How to define a formal framework combining I/O logic and neural-symbolic computation for normative reasoning?

Among other formalisms used in normative systems, we choose I/O logic because it presents a strong (and natural) similarity with neural networks: both have a separate specification of inputs and outputs. We exploit such similarity first to encode knowledge expressed in terms of I/O rules into neural networks, and then to use the neural network to reason and learn new norms in a dynamic environment.

Methodologically, we adopt the Neural-Symbolic paradigm of [d'Avila Garcez *et al.*, 2002] which embeds (symbolic) logical programs into feed-forward neural networks. Neural-symbolic systems provide translation algorithms from symbolic logic to neural networks and vice-versa. The network is used for robust learning and computation, while the logic provides (i) background knowledge to help learning (when the logic is translated into the neural network) and (ii) high-level explanations for the network models[1] (when the trained neural network is translated into the logic). A sound translation for the (i) step is done by using the CILP system [d'Avila Garcez *et al.*, 2002].

In normative reasoning there are some problems which have to be handled. These problems are: *permissions*, *dilemmas*, *contrary to duties* and *exceptions*. A normative agent, is an agent capable to behave within an environment regulated by norms, must be able to handle the situations listed above. A way to handle such situations is by using *priorities*. A description about how a normative agent can handle such situations with the use of priorities is described in [Boella *et al.*, 2011].

In particular, we address the following sub-questions:

- How to use priorities with I/O logic rules in order to handle normative reasoning problems?
- How to translate I/O logic into neural networks by using CILP and keeping the soundness of the logic?

We provide a description of the simulator used for testing our approach. The simulator has been written in *Java*[2] and

[1]We are not going to discuss this step in this paper.

[2]www.java.com/

using the package *Joone*[3], a framework to model neural networks.

After describing the simulator we provide the results obtained from some of the experiments made.

The paper is structured as follows. In Section 2 we introduce the neural-symbolic approach, the I/O logic and the architecture of a normative agent. In Section 3 we first describe which restrictions need to be applied to I/O rules. Then how we embed priorities within the rules and at last the role of *permissions* in our approach. In Section 4 we describe the case study used in the experiments. In Section 5 we describe the simulator and the experiments. In Section 6 we present the conclusions.

2 Related work

Neural-Symbolic approach

The main purpose of a *neural-symbolic approach* is to bring together connectionist and symbolic approaches [d'Avila Garcez *et al.*, 2002]. In this way it is possible to exploit the strengths of both approaches and to avoid their drawbacks. With such approach we are able to formally represent the norms governing the normative system. In addition we are also capable to exploit the instance learning capacities of neural networks and their massive parallel computation.

Algorithms like *KBANN*[Towell and Shavlik, 1994] and *CILP*[d'Avila Garcez and Zaverucha, 1999] provide a sound translation of a symbolic representation of the knowledge within a neural network. The advantage of CILP, is that it uses the sigmoid function for its perceptrons. This allows the use of *backpropagation* for learning. In what follows, we use a variant of CILP since we are interested in the integration of reasoning and learning capabilities.

I/O Logic

To describe the norms regulating the system we use I/O Logic [Makinson and van der Torre, 2000]. Rules used in I/O logic are defined in the shape $R_1 = (A, B)$. Both A and B represent sets of literals. The literals contained in A (or in B) can be either in conjunction or disjunction between them. A represent the antecedent of the rule, what must be considered true in order to activate the rule. Instead B is the consequent, what is considered true after the rule has been activated.

I/O logic provides some reasoning mechanisms to produce outputs form the inputs. The first of this mechanisms is the *simple-minded output*. This mechanism does not satisfy the principle of identity. Instead the simple-minded output possess other features like *strengthening input*, *conjoining output* and *weakening output*. The I/O logic also provides other reasoning mechanisms, *basic output*, *reusable output* and *reusable basic output* which allow additional features. Respectively *input disjunction* for the basic output, *reusability* for the reusable output and both for reusable basic output. A detailed description of the I/O logic mechanisms and features can be found in [Makinson and van der Torre, 2000].

In [Boella *et al.*, 2010] it is described how a connectionist approach like neural networks can embed the different features of I/O logic. In this way it is possible by using translation algorithms like KBANN or CILP to reproduce the mechanisms of I/O logic.

[3]http://sourceforge.net/projects/joone/

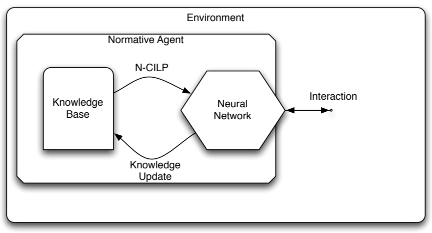


Figure 1: Neural-Symbolic Normative Agent.

Normative agent

An agent is defined as an entity that actively interacts with its surrounding environment and with other agents if we consider a multi-agent system. In this paper we will focus on a single agent, more precisely a *normative agent*. Figure 1 shows a normative agent, an entity that has to act and behave by following the norms regulating the environment where it acts. A more detailed description of what is a normative agent can be found in [Boella *et al.*, 2011].

In this paper we do not focus on which action the agent should execute in a particular situation. For *situation* we mean a particular state of the environment, including all the inputs that the agent can use to make its decisions. Instead we concentrate our efforts into deciding what an agent ought to do and can do while in a particular situation.

The normative agent must be capable to handle the problems that can arise. In normative reasoning some of this problems are *dilemmas exceptions* and *contrary to duties*. Dilemmas occurs when the agent is facing two contradictory obligations. With contradictory obligations we mean two different obligations which cannot be accomplished both. An example is the *Sartre's soldier*, which has the moral obligation to not to kill, but being the soldier he has to fight and kill his enemies. The second problem that an agent may face is the exception. An exception, like the name suggests, occurs during exceptional situations. In these exceptional situations it is possible that a rule which usually has to be applied is overridden by a different one. We can provide an example by considering the rules of a football match. The standard rule is that the players cannot play the ball with their hands. In this case we have an exception if we consider the goalkeeper. This particular player while inside its own goal area, is allowed to use its hands to play the ball. The last problem mentioned is the contrary to duty[Prakken and Sergot, 1996]. In normative reasoning the violation of a rule is not always to be considered a critical failure. In some circumstances is possible to handle the violation by fulfilling alternative obligations. As an example we can consider the situation where we are in a pub with a friend. Supposing that our friend is drinking a beer. The general rule is that we should not spill our friends beer. Considering the unfortunate situation where we accidentally (or not) spill our friend's beer. Our friend has now the possibility to severe our friendship due to our violation. In this situation we still have the possibility to repair to the violation, by considering to buy our friend a new beer.

3 Neural Networks for Norms

3.1 I/O logic for Norms

In order to use I/O logic to represent normative rules, we need to add modalities. We add two different types of modalities, the obligation ($\mathbf{O}$) used to define what the agent is ought to do or prohibited[4] and the permission ($\mathbf{P}$) use to define what is permitted to the agent. We will consider the modalities introduced are unary operators, acting over a single literal. For example $\mathbf{P}(\alpha)$ represents the permission to do α.

Considering again an I/O logic rule: $R_1 = (A, B)$ where A and B are set of literals. By unfolding the set B we can consider all the literals contained in the consequent: $B = \{\beta_1, \beta_2, \ldots, \beta_n\}$. At each literal in the consequent, can be added one of the possible modalities. By doing so what we obtain is a normative rule, a rule which does not states facts but, oughts, prohibitions and permissions for the normative agent. For example: $\mathbf{O}(\beta_1), \mathbf{P}(\beta_2), \mathbf{O}(\beta_3)$, a normative rule with such consequent, would mean that β_1 and β_3 are oughts and β_2 is a permission instead.

3.2 I/O rules restrictions

For the translation we adopt a variant of the CILP algorithm. We use N-CILP, that translates a knowledge base containing I/O logic rules in a neural network.

In order to allow N-CILP to translate I/O logic rules, we have apply some restrictions on the rules. However those restrictions are not crippling the expressivity of the logic.

1. First we need to restrict the antecedent (input) of the rule. We want that the literals in the antecedent are connected by conjunctions only. We see now how this does not harm the expressivity of the logic. Considering a I/O logic rule with a disjunction in the antecedent like the following: $(A_1 \vee A_2, B)$ where A_1 and A_2 are sets of conjuncted literals. For each disjunction we split the antecedent. In this particular case we split the starting rule into two rules with a new antecedent and the same consequent. Obtaining in this case two rules: (A_1, B) and (A_2, B) that considered together allow the same semantics of the starting rule.
2. We restrict the consequent (output) to contain a single literal. If we consider the set of consequents to be constituted by conjuncted literals, then every literal in the set produces a new rule, with itself in the consequent and the same antecedent as the starting rule.

   In this case the logic may lose some expressivity, it may happen if we need disjunctions in the consequent. Disjunctions in the consequent can be used to introduce uncertainty in the system. However due to the fact that we consider normative systems, the rules are used to describe the norms governing the system. We can safely assume that norms are meant to regulate the system and not introduce uncertainty.

[4]By prohibition we mean the obligation of a negative literal. In example we can have $\mathbf{O}(\neg\chi)$ which means the obligation to do not χ, in other words the prohibition to do χ.

3. The last restriction regards the consequent. In addition we have to restrict it to be a positive literal. We address this problem by syntactically considering a negative literal as positive. In example the consequent $\neg\beta_i$ is considered as: β'_i. The newly created literal is semantically still considered negative. Also in this case the logic does not lose expressivity.

3.3 Priorities

Priorities are used to give a partial ordering between rules. This ordering is useful because sometimes between two applicable rules we want to apply only one. This can happen when considering for example *exceptions*.

Here we explain how we encode priorities within the rules by using the *negation as failure* ($\sim$). Considering for example two rules: $R_1 = (A_1 \wedge A_2, \mathbf{O}(\beta_1))$, $R_2 = (A_1 \wedge A_3, \mathbf{O}(\beta_2))$ and a priority relation between them: $R_1 \succ R_2$, where the first rule has the priority. Knowing A_1, A_2 and A_3 are sets of conjuncted literals, we embed the priority into the rule with the lowest priority. To do so we include into the antecedent of the rule with lower priority, the negation as failure of the literals in the antecedent of the higher prioritized rule, that does not appear in the antecedent of the lower priority rule.

Considering for example the two rules given, we have to modify R_2. In this case we need to include in the antecedent of R_2 the part of the antecedent of R_1 that differs, in this case A_2. After embedding the priority within the second rule, it becomes: $R'_2 = (A_1 \wedge \sim A_2 \wedge A_3, \mathbf{O}(\beta_2))$.

3.4 Permissions

An important distinction between *oughts* and *permissions*, is that the second ones are not explicitly encoded in the neural network. In our approach we consider that something is permitted to the agent if not explicitly forbidden[5]. Due to this we consider rules with a permission in their consequent to implicitly have the priority over the rules that forbid the same action.

For example considering two rules $R_1 = (A_1, \mathbf{P}(\beta_1))$, $R_2 = (A_2, \mathbf{O}(\neg\beta_1))$. The first rule permits β_1 and the second forbids it. In this case we consider implicitly the following priority relation $R_1 \succ R_2$ to hold.

4 Case study

To test the performance of our approach to normative reasoning we use the RoboCup scenario. For simplicity we focused on the reasoning of a single robot, leaving out the multi-agent aspect of the scenario.

With our approach, the robot does not plan the sequence of the actions. Instead the robot analyzes the current situation and by taking into consideration the rules of the game[Menegatti, 2007], it knows what is ought and what is prohibited.

If we consider that the robot makes its decisions taking into account only the rules, then the robot is acting within a static environment. Because the rules does not change in the middle of the game. In order to add dynamism into the environment

[5]We consider the ought of a negative literal as a prohibition.

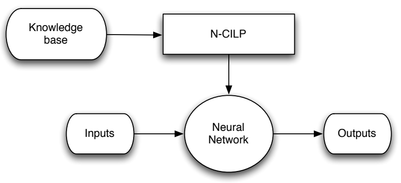


Figure 2: Neural-Symbolic simulator.

we add an additional ruling element. The first ruling element is the referee, which enforces the rules of the game. The additional ruling element is the coach, which demands to the robots to play in a specific way. The coach can introduce new rules or lift some of the existing ones during the game. In this way, a robot that acts in an environment where the coach is involved, sometimes needs to adapt its behavior.

4.1 Knowledge base structure

The knowledge base used by the robot contains both the rules of the game and the coach directions. Both the rules and the directions are shaped in I/O logic rules format. The knowledge base used in the experiments contains 29 rules, including the rules which have a permission in their consequent.

The knowledge base also contains the priority relations between the rules, which are used to resolve possible conflicts among them (like the production of contradicting oughts).

We show some of the rules contained in the knowledge base:

$R_1 : (\top^6, \mathbf{O}(\neg impact\_opponent))$

$R_2 : (\top, \mathbf{O}(\neg use\_hands))$

$R_3 : (goalkeeper \wedge inside\_own\_area, \mathbf{P}(use\_hands))$

$R_4 : (ball \wedge opponent\_approaching, \mathbf{O}(pass))$

The first rule states that a robot should never impact into an opponent. The second rule is again a prohibition, that states that a robot should not use its hands to play the ball. The third rule is different, because states that the goalkeeper is allowed to use its hands while inside its own goal area. The last rule is not from the RoboCup ruling, instead is one of the rules that the coach may have given to robots, to influence their playing behavior. The fourth rule states that if an opponent is approaching the robot handling the ball, then that robot should pass the ball.

5 Simulator and experimental results

5.1 The simulator

In Figure 2 it is shown how the simulator works. The knowledge base contains the the rules that the robot knows. We consider that the priorities are embedded within the rules as described in the previous section. The knowledge base is used as the input for the *N-CILP* translation algorithm.

[6]$\top$ means that the antecedent is always true, in other words the rule is always applied.

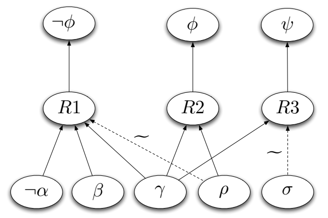


Figure 3: Example of Embedding

N-CILP

Given a knowledge base $\mathcal{KB}$ for each rule $R_k = (\alpha_{i_1} \wedge \ldots \wedge \alpha_{i_n} \wedge \sim \alpha_{i_n+1} \wedge \ldots \wedge \sim \alpha i_m, \mathbf{O}(\beta_{o_1}))$in $\mathcal{KB}$ do:

1. For each literal α_{i_j} $(1 \leq j \leq m)$ in the input of the rule. If there is no input neuron labeled α_{i_j} in the input level, then add a neuron labeled α_{i_j} in the input layer.
2. Add a neuron labeled N_k in the hidden layer.
3. If there is no neuron labeled β_{o_1} in the output level, then add a neuron labeled β_{o_1} in the output layer.
4. For each literal α_{i_j} $(1 \leq j \leq n)$; connect the respective input neuron with the neuron labeled N_k in the hidden layer with a positive weighted arc.
5. For each literal α_{i_h} $(n + 1 \leq j \leq m)$; connect the respective input neuron with the neuron labeled N_k in the hidden layer with a negative weighted arc[7].
6. Connect the neuron labeled N_i with the neuron in the output level labeled β_{o_1} with a positive weighted arc[8]

In [d'Avila Garcez *et al.*, 2002] it is shown how the weights of the resulting neural network can be calculated.

In Figure 3 we show the structure of a neural network constructed with the N-CILP algorithm from the translation of four rules. The rules are $R_1 = (\neg\alpha \wedge \beta \wedge \gamma, \mathbf{O}(\neg\phi))$, $R_2 = (\gamma \wedge \rho, \mathbf{O}(\phi))$, $R_3 = (\gamma, \mathbf{O}(\neg\psi))$ and the *permission rule* $R_4 = (\gamma \wedge \sigma, \mathbf{P}(\psi))$. Between the rules we have a priority ordering $R_2 \succ R_1$ that inhibits the activation of the first rule whenever the second is activated. This priority is embedded within the rules as described earlier in this section and as a result we obtain a new first rule: $R'_1 = (\neg\alpha \wedge \beta \wedge \gamma \wedge \sim \rho, \mathbf{O}(\neg\phi))$. The implicit priority of R_4 over R_3 embeds within the latter the negation as failure obtaining a new rule $R'_3 = (\gamma \wedge \sim \sigma, \mathbf{O}(\neg\psi))$. The neural network is built from rules R'_1, R_2 and R'_3[9], notice the dotted lines in the network which are negative weighted arcs representing the negation as failures in the rules R'_1 and R'_3, with the task to inhibit the rules if the respective negation as failure given in input is activated.

[7]The connections between these input neurons and the hidden neuron of the rule represents the priorities translated with the *negation as failure*.

[8]Each output in the rules is considered as a positive atom during the translation, this means that if we have a rule with a negative output $\neg\beta$, in the network we translate an output neuron labeled β' that has the same meaning of $\neg\beta$ but for the translation purpose can be treated as a positive output.

[9]Rule with a permission $\mathbf{P}$ in the consequent are not encoded in the neural network.

5.2 Experimental results

We describe some experiments used to test the capabilities of neural networks constructed with N-CILP. We introduce the measures used to evaluate the behavior of the networks and the parameters used.

To evaluate the evaluate the performance of the neural network, we use two distinct measures: *tot* and *part*.

$$tot = \frac{\sum_{i=1}^{n} I(\bigwedge_{j=1}^{k}(c_{ij} == o_{ij}))}{n}$$
$$part = \frac{\sum_{i=1}^{n}\sum_{j=1}^{k} I(c_{ij} == o_{ij})}{n * k}$$

n refers to the cardinality of the test set and k indicates the number of output neurons of the neural network. o_{ij} indicates the value of j-th output of the NN for the i-th test instance. c_{ij} indicates the true value (desired value) of the j-th literal of the i-th test instance. $I(\cdot)$ is the indicator, a function returning 1 if the argument is true and zero otherwise. The *tot* measure evaluates how many instances were processed entirely correctly. Instead *part* considers the number of single output neurons correctly activated.

By having 16 output neurons in the neural networks used in the test, using only *tot* to measure the accuracy could be misleading. To clarify this point we can consider an example. We can assume that by processing two instances, the neural network have produced for the first, 15 correct outputs out of the total 16. For the second it managed to return all the correct outputs. If we take into account the *tot* measure, we obtain an accuracy of 50% that does not seems a great result. Instead by considering the *part* measure, we obtain an accuracy higher than the 96%. Which better underlines that the network only missed one output out of 32 produced for the two instances given.

In our experiments we train the neural network using a *10fold cross validation*. We divide the initial data set of instances in ten distinct subsets. Each subset is then used as test set while the others are used together as training set. In this way the instances seen during training are left out of the testing phase to train ten networks and the results averaged.

In all the experiments we set the training parameters for the neural networks as follows: *learning rate*: 0.8, *momentum*: 0.3 and *training cycles*: 100 [Haykin, 1999].

Non symbolic approach comparison

We compare the learning capacity of a network built with N-CILP with a non symbolic neural network. One of the well known issues of neural networks is deciding the number of neurons to use in the hidden level. To not to put the non symbolic neural network in excessive disadvantage, we decided to adopt the same number of hidden neurons for both networks[10]. The difference between the networks involved in this test lies in their connection weights. The neural network built with N-CILP sets its weights according to the rules in the knowledge base. Instead the non symbolic network has its weights randomly initialized. One advantage of a network built with N-CILP is that even without any training, it is capable to correctly process instances by applying the rules contained in the knowledge base.

[10]The number of hidden neurons to use in the neural networks is equal to the number of rules used for the network construction with N-CILP.

The network built with N-CILP uses a starting knowledge base containing 20 rules. During the training phase the network tries to learn 9 additional rules from the instances provided. The non symbolic network during the training phase is provided with the same instances, the difference is that this network have to learn all the 29 rules applied in the instances.

The results from the experiments show that the non symbolic neural network obtains the following accuracies: *tot*: 5,13% *part*: 45,25%. Instead the network built N-CILP: *tot*: 5,38% *part*: 49,19%. We can see that under exactly the same conditions, N-CILP improves the training-set performance of the network.

Enhancing the knowledge base

The second experiment measures how the neural network performs by increasing the number of rules in the knowledge base. This test is important because the goal of a *Neural-Symbolic System*, is not only to construct a neural network capable to compute the same semantics as rule into the knowledge base. Another important objective is to exploit the learning capabilities of the neural networks, allowing the robot to increase the number of rules in its knowledge base from what it learned[d'Avila Garcez *et al.*, 2002].

The test is done incrementally. From the full set of 29 rules, the experiment first step starts with a knowledge base containing 20 rules and tries to learn the remaining 9. Successively 2 rules are incrementally added into the initial knowledge base during each step. In this way the unknown rules that the network has to learn decreases by 2 each step. In example at the second step of the experiment the starting knowledge base contains 22 rules and the network tries to learn 7 rules during the training phase.

During each step the neural network is tested over instances where the full set of rules is applied. In this way the network continues to process using the rules already known, reducing the risk to forget them and in the meantime it tries to learn of the unknown rules.

The results of this experiment are shown in Figure 4. We can see that for the first two steps of the experiment the accuracies measured quite low. instead for the last two steps the performance of the neural network increases, reaching an accuracy peak of 98,01% for the *part* measure and 91,18% for the *tot*.

From the experiment proposed we observed a direct correlation between the number of the rules in the starting knowledge base and the performance of the neural network. Another thing that can be noticed is that the smaller becomes the number of rules that the network does not know, w.r.t. the number of rules in the initial knowledge base can impact the performances of the network, also due to the fact that a network built from a larger knowledge base possesses more connections.

6 Conclusion

In this paper we presented a way to combine a connectionist and a symbolic approach that can be used for normative

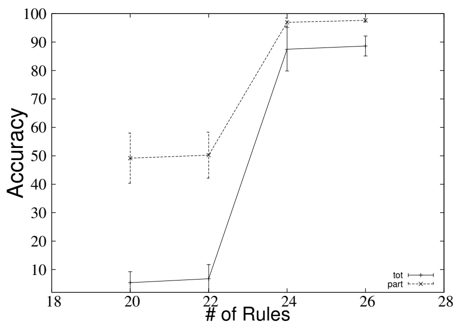


Figure 4: Accuracy of *tot* and *part* measures increasing the number of rules

reasoning. In this way agents behaving in normative environments, are able to adapt themselves to the normative evolution of the world. An important step that has not been covered by this paper concerns rules extraction. Rules extraction refers to the process where a new knowledge base is recompiled from the trained network. A method to achieve this task has already been proposed in [d'Avila Garcez *et al.*, 2002].

For a normative agent is important to be able to cope with normative problems. Here we have show how the priorities, used to achieve this task, can be embedded within the rules and translated using the N-CILP algorithm.

In the paper we provided some of the results obtained with the simulator by using our approach for a normative agent. We are aware that more experiments are needed in order to claim the validity of the approach. However we believe that the results obtained so far are promising. We show a comparison between our approach and a (not so disadvantaged) non symbolic neural network. Further comparisons with other approaches for dynamic normative system should be made. In example like comparing the pure symbolic approach used by [Corapi *et al.*, 2010], based on inductive learning, and our neural-symbolic approach.

A related line of research involves the area of Argumentation. Argumentation has been proposed, among other things, as a method to help symbolic machine learning. It would be interesting to investigate the links between the work presented here, argumentation applied to law, and the neural symbolic approach to argumentation introduced in [d'Avila Garcez *et al.*, 2005].

Acknowledgements

Silvano Colombo Tosatto and Valerio Genovese are supported by the National Research Fund, Luxembourg.

References

[Boella and van der Torre, 2006] Guido Boella and Leendert van der Torre. A game theoretic approach to contracts in multiagent systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 36(1):68–79, 2006.

[Boella *et al.*, 2009] Guido Boella, Gabriella Pigozzi, and Leendert van der Torre. Normative framework for normative system change. In *8th Int. Joint Conf. on Autonomous Agents and Multiagent Systems AAMAS 2009*, pages 169–176. IFAAMAS, 2009.

[Boella *et al.*, 2010] Guido Boella, Silvano Colombo Tosatto, Artur S. d'Avila Garcez, and Valerio Genovese. On the relationship between i-o logic and connectionism. In *13th International Workshop on Non-Monotonic Reasoning*, 2010.

[Boella *et al.*, 2011] Guido Boella, Silvano Colombo Tosatto, Artur S. d'Avila Garcez, Dino Ienco, Valerio Genovese, and Leendert van der Torre. Neural symbolic systems for normative agents. In *10th International Conference on Autonomous Agents and Multiagent Systems*, 2011.

[Corapi *et al.*, 2010] Domenico Corapi, Marina De Vos, Julian Padget, Alessandra Russo, and Ken Satoh. Norm refinement and design through inductive learning. In *11th International Workshop on Coordination, Organization, Institutions and Norms in Agent Systems COIN 2010*, pages 33–48, 2010.

[d'Avila Garcez and Zaverucha, 1999] Artur S. d'Avila Garcez and Gerson Zaverucha. The connectionist inductive learning and logic programming system. *Applied Intelligence*, 11:59–77, July 1999.

[d'Avila Garcez *et al.*, 2002] Artur S. d'Avila Garcez, Krysia B. Broda, and Dov M. Gabbay. *Neural-Symbolic Learning Systems*. Perspectives in Neural Computing. Springer, 2002.

[d'Avila Garcez *et al.*, 2005] Artur S. d'Avila Garcez, Dov M. Gabbay, and Luis C. Lamb. Value-based argumentation frameworks as neural-symbolic learning systems. *J. of Logic and Computation*, 15(6):1041–1058, 2005.

[Haykin, 1999] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[Makinson and van der Torre, 2000] David Makinson and Leendert van der Torre. Input-output logics. *Journal of Philosophical Logic*, 29, 2000.

[Menegatti, 2007] Emanuele Menegatti. Robocup soccer humanoid league rules and setup, 2007.

[Prakken and Sergot, 1996] Henry Prakken and Marek Sergot. Contrary-to-duty obligations. *Studia Logica*, 57, 1996.

[Sen and Airiau, 2007] Sandip Sen and Stephane Airiau. Emergence of norms through social learning. In *Procs. of the 20th International Joint Conference on Artificial Intelligence - IJCAI*, pages 1507–1512, 2007.

[Shoham and Tennenholtz, 1997] Yoav Shoham and Moshe Tennenholtz. On the emergence of social conventions: Modeling, analysis, and simulations. *Artificial Intelligence*, 94(1-2):139–166, 1997.

[Towell and Shavlik, 1994] Geoffrey G. Towell and Jude W. Shavlik. Knowledge-based artificial neural networks. *Artif. Intell.*, 70:119–165, October 1994.

Towards Developmental AI: The paradox of ravenous intelligent agents

Michelangelo Diligenti, Marco Gori, and Marco Maggini
DII - University of Siena

1 Introduction

In spite of extraordinary achievements in specific tasks, nowadays intelligent agents are still striving for acquiring a truly ability to deal with many challenging human cognitive processes, especially when a mutable environment is involved. In the last few years, the progressive awareness on that critical issue has led to develop interesting bridging mechanisms between symbolic and sub-symbolic representations and to develop new theories to reduce the huge gap between most approaches to learning and reasoning. While the search for such a unified view of intelligent processes might still be an obliged path to follow in the years to come, in this paper, we claim that we are still trapped in the insidious paradox that feeding the agent with the available information, all at once, might be a major reason of failure when aspiring to achieve human-like cognitive capabilities. We claim that the children developmental path, as well as that of primates, mammals, and of most animals might not be primarily the outcome of biologic laws, but that it could be instead the consequence of a more general complexity principle, according to which the environmental information must properly be filtered out so as to focus attention on "easy tasks." We claim that this leads necessarily to stage-based developmental strategies that any intelligent agent must follow, regardless of its body.

2 Developmental path and focus of attention

There a number of converging indications that most of nowadays approaches to learning and reasoning have been bouncing against the same wall (in the case of sequential information, see e.g. [Frasconi *et al.*, 1995]). This is especially clear when facing cognitive tasks that involve both learning and reasoning capabilities, that is when symbolic and sub-symbolic representations of the environment need to be properly bridged. A unified approach to embrace the behavior of intelligent agents involved in both perceptual and symbolic information is based on expressing learning data and explicit knowledge by constraints [Diligenti *et al.*, 2010]. Following that framework, let us consider tasks that can be formalized by expressing a *parsimonious* solution consistent with a given set of constraints $\mathcal{C} = \{\chi_1, \ldots, \chi_q\}$. It is worth mentioning that the in the case of supervised learning, the parsimonious satisfaction of the constraints is reduced to a finite collection of points according to the classic statistical framework behind kernel machines. We consider an agent which operates dynamically in a mutable environment where, at each time t, it is expected to access only a limited subset $\mathcal{C}_t \subset \mathcal{C}_U$ of constraints, where $\mathcal{C}_U$ can be thought of as the *universal set of constraints*. Of course, any agent of relevant interest might be restricted to acquire a limited set of constraints $\mathcal{C}$, so as $\forall t \in \mathcal{T} : \mathcal{C}_t \subset \mathcal{C} \subset \mathcal{C}_U$. Instead of following a *developmental path*, one could think of agents that acquire $\mathcal{C}$ all at once.

Definition 2.1 *A ravenous agent is one which accesses the whole constraint set at any step, that is one for which*
$\forall t \in \mathcal{T} : \mathcal{C}_t = \mathcal{C}$.

At first a glance, ravenous agents seem to have more chances to develop an efficient and effective behavior, since they can access all the information expressed by $\mathcal{C}$ at any time. However, when bridging symbolic and sub-symbolic models one often faces the problem of choosing a developmental path. It turns out that accessing all the information at once might not be a sound choice in terms of complexity issues.

The paradox of ravenous agents: *Ravenous agents are not the most efficient choice to achieve a parsimonious constraint consistency.*

To support the paradox, we start noting that hierarchical modular architectures used in challenging perceptual tasks like vision and speech are just a way to introduce intermediate levels of representation, so as to focus on simplified tasks. For example, in speech understanding, phonemes and words could be intermediate steps for understanding and take decisions accordingly. Similarly, in vision, SIFT features could be an intermediate representation to achieve the ability to recognize objects. However, when looking for deep integration of sub-symbolic and symbolic levels the issue is more involved and mostly open. We discuss three different contests that involve different degree of symbolic and sub-symbolic representations.

Developmental paths

- *Reasoning in the environment* - When thinking of circumscription and, in general, of a non-monotonic reasoning, one immediately realizes that we are addressing issues that are outside the perimeter of ravenous agents. For example, we can start from a default assumption, that is the typical "bird flies." This leads us to conclude that if a given animal is known to be a bird, and nothing else is known, it can be assumed to be able to fly. The default assumption can be retracted in case we subsequently learn that the animal is a penguin. Something similar happens during the learning of the past tense of English verbs [Rumelhart and McClelland, 1986], that is characterized by three stages: memorization of the past tense of a few verbs, application of the rule of regular verbs to all verbs and, finally, acquisition of the exceptions. Related mechanisms of retracting previous hypotheses arise when performing abductive reasoning. For example, the most likely explanation when we see wet grass is that it rained. This hypothesis, however, must be retracted if we get to know that the real cause of the wet grass was simply a sprinkler. Again, we are in presence of non-monotonic reasoning. Likewise, if a logic takes into account the handling of something which is not known, it should not be monotonic. A logic for reasoning about knowledge is the autoepistemic logic, which offers a formal context for the representation and reasoning of knowledge about knowledge. Once again, we rely on the assumption of not to construct ravenous agents, which try to grasp all the information at once, but on the opposite, we assume that the agent starts reasoning with a limited amount of information on the environment, and that there is a mechanism for growing up additional granules of knowledge.

- *Nature helps developing vision* - The visual behavior of some animals seems to indicate that motion plays a crucial role in the process of scene understanding. Like other animals, frogs could starve to death if given only a bowlful of dead flies [1], whereas they catch and eat moving flies [Lettvin *et al.*, 1968]. This suggests that their excellent hunting capabilities depend on the acquisition of a very good vision of moving objects only. Saccadic eye movements play an important role in facilitating human vision and, in addition, the autonomous motion is of crucial importance for any animal in vision development. Birds, some of which exhibit proverbial abilities to discover their preys (e.g. eagles, terns), are known to detect slowly moving objects, an ability that is likely to have been developed during evolution, since the objects they typically see are far away when flying. A detailed investigations on fixational eye movements across vertebrate indicates that micro-saccades appear to be more important in foveate than afoveate species and that saccadic eye movements seem to play an important role in perceiving static images [Martinez-Conde and Macknik, 2008]. When compared with other animals, humans are likely to perform better in static - or nearly static - vision simply because they soon need to look at objects and pictures thoughtfully. However, this comes at a late stage during child development, jointly with the emergence of other symbolic abilities. The same is likely to hold for amodal perception and for the development of strange perceptive behavior like popular Kanizsa triangle [Kanizsa, 1955]. We claim that image understanding is very hard to attack, and that the fact that humans brilliantly solve the problem might be mostly due to the natural embedding of pictures into visual scenes. Human perception of static images seems to be a higher level quality that might have been acquired only after having gained the ability of detecting moving objects. In addition, the saccadic eye movements might suggest that static images are just an illusory perception, since human eyes always perceive moving objects. As a consequence, segmentation and recognition are only apparently separate processes: They could be in fact two faces of the same medal that are only regarded as separate phases mostly because of the bias induced by years of research in pattern recognition aimed at facing specific perceptual tasks. Interestingly, animals which develop a remarkable vision system, and especially humans, in their early stages characterized by scarce motion abilities, deal primarily with moving objects from a fixed background, which facilitates their segmentation and, consequently, their recognition. Static and nearly static vision comes later in human developments, and it does not arise at all in many animals like frogs. The vision mechanisms seems to be the outcome of complex evolutive paths (e.g. our spatial reasoning inferences, which significantly improve vision skills, only emerge in late stage of development). We subscribe claims from developmental psychology according to which, like for other cognitive skills, its acquisition follows rigorous stage-based schemes [Piaget, 1961]. The motion is likely to be the secret of vision developmental plans: The focus on quickly moving objects at early stages allows the agent to ignore complex background.

- *On the bridge between learning and logic*
  Let us consider the learning task sketched in Fig. 2, where supervised examples and FOL predicates can be expressed in the same formalism of constraints. There is experimental evidence to claim that a ravenous agent which makes use of all the constraints $\mathcal{C}$ (supervised pairs and FOL predicates) is not as effective as one which focuses attention on the supervised examples and, later on, continues incorporating the predicates [Diligenti *et al.*, 2010]. Basically, the developmental path which first favors the sub-symbolic representations leads to a more effective solution. The effect of this developmental plan is to break up the complexity of learning jointly examples and predicates. Learning turns out to be converted into an optimization problem, which is typically plagued by the presence of sub-optimal solutions. The developmental path which enforces the learning from examples at the first stage is essentially a way to circumvent local minima.

[1]In addition to their infrared vision, snakes are also known to react much better to quick movements.

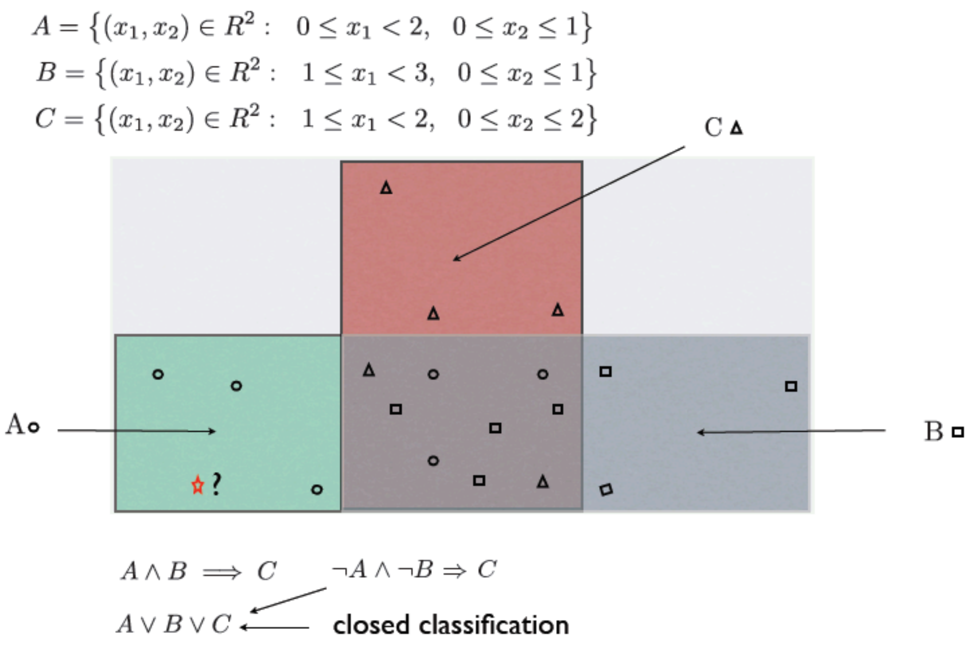

$A = \{(x_1, x_2) \in R^2 : \quad 0 \leq x_1 < 2, \quad 0 \leq x_2 \leq 1\}$
$B = \{(x_1, x_2) \in R^2 : \quad 1 \leq x_1 < 3, \quad 0 \leq x_2 \leq 1\}$
$C = \{(x_1, x_2) \in R^2 : \quad 1 \leq x_1 < 2, \quad 0 \leq x_2 \leq 2\}$
C △
A ○
B □
?
$A \wedge B \implies C$
$\neg A \wedge \neg B \Rightarrow C$
$A \vee B \vee C$
closed classification

Figure 1: The task consists of learning three classes from examples and from a set of FOL predicates. The ordering of presentation does matter.

3 Conclusions

This paper supports the position that stage-based learning, as discussed in developmental psychology is not the outcome of biology, but is instead the consequence of optimization principles and complexity issues that hold regardless of the body. This position is supposed to re-enforce recent studies on developmental AI more inspired to studies in cognitive development (see e.g. [Sloman, 2009]) and is somehow coherent with the growing interest in deep architectures and learning [Bengio *et al.*, 2009].

Acknowledgements: We thank Luciano Serafini for insightful discussions.

References

[Bengio *et al.*, 2009] Y. Bengio, J. Louradour, R. Lollobert, and J. Weston. Curriculum learning. In *26th Annual International Conference on Machine Learning*, pages 41–48, Montreal, Canada, June 2009.

[Diligenti *et al.*, 2010] M. Diligenti, M. Gori, M. Maggini, and L. Rigutini. Multitask kernel-based learning with logic constraints. In *The 19th European Conference on Artificial Intelligence*, 2010.

[Frasconi *et al.*, 1995] P. Frasconi, M. Gori, and G. Soda. Recurrent neural networks and prior knowledge for sequence processing: a constrained non-deterministic approach. *Knowledge-Based Systems*, 8(6):313–332, 1995.

[Kanizsa, 1955] G. Kanizsa. Margini quasi-percettivi in campi con stimolazione omogenea. *Rivista di Psicologia*, 49(1):7–30, 1955.

[Lettvin *et al.*, 1968] J.Y. Lettvin, H.H. Maturana, W.S. McCulloch, and W.H. Pitts. What the frog's eye tells the frog's brain. In *Reprinted from - The Mind: Biological Approaches to its functions - Eds Willian C. Corning, Martin Balaban*, pages 233–358. 1968.

[Martinez-Conde and Macknik, 2008] S. Martinez-Conde and S.L. Macknik. Fixational eye movements across vertebrates: Comparative dynamics, physiology, and perception. *Journal of Vision*, 8(14):1–16, 2008.

[Piaget, 1961] J. Piaget. *La psychologie de l'intelligence*. Armand Colin, Paris, 1961.

[Rumelhart and McClelland, 1986] D.E. Rumelhart and J.L. McClelland. On learing the pat tense of english verbs. In *Parallel Distributed Processing, Vol. 2*, pages 216–271. 1986.

[Sloman, 2009] A Sloman. Ontologies for baby animals and robots. from baby stuff to the world of adult science: Developmental ai from a kantian viewpoint. Technical report, University of Birmingham, 2009.

Extracting Argumentative Dialogues from the Neural Network that Computes the Dungean Argumentation Semantics

Yoshiaki Gotou
Niigata University, Japan
gotou@cs.ie.niigata-u.ac.jp

Takeshi Hagiwara
Niigata University, Japan
hagiwara@ie.niigata-u.ac.jp

Hajime Sawamura
Niigata University, Japan
sawamura@ie.niigata-u.ac.jp

Abstract

Argumentation is a leading principle both foundationally and functionally for agent-oriented computing where reasoning accompanied by communication plays an essential role in agent interaction. We constructed a simple but versatile neural network for neural network argumentation, so that it can decide which argumentation semantics (admissible, stable, semi-stable, preferred, complete, and grounded semantics) a given set of arguments falls into, and compute argumentation semantics via checking. In this paper, we are concerned with the opposite direction from neural network computation to symbolic argumentation/dialogue. We deal with the question how various argumentation semantics can have dialectical proof theories, and describe a possible answer to it by extracting or generating symbolic dialogues from the neural network computation under various argumentation semantics.

1 Introduction

Much attention and effort have been devoted to the symbolic argumentation so far [Rahwan and Simari, 2009][Prakken and Vreeswijk, 2002][Besnard and Doutre, 2004], and its application to agent-oriented computing. We think that argumentation can be a leading principle both foundationally and functionally for agent-oriented computing where reasoning accompanied by communication plays an essential role in agent interaction. Dung's abstract argumentation framework and argumentation semantics [Dung, 1995] have been one of the most influential works in the area and community of computational argumentation as well as logic programming and non-monotonic reasoning.

In 2005, A. Garcez et al. proposed a novel approach to argumentation, called the neural network argumentation [d'Avila Garcez *et al.*, 2005]. In the papers [Makiguchi and Sawamura, 2007a][Makiguchi and Sawamura, 2007b], we dramatically developed their initial ideas on the neural network argumentation to various directions in a more mathematically convincing manner. More specifically, we illuminated the following questions which they overlooked in their paper but that deserve much attention since they are beneficial for understanding or characterizing the computational power and outcome of the neural network argumentation from the perspective of the interplay between neural network argumentation and symbolic argumentation.

1. *Can the neural network argumentation algorithm deal with self-defeating or other pathological arguments?*
2. *Can the argument status of the neural network argumentation correspond to the well-known status in symbolic argumentation framework such as in [Prakken and Vreeswijk, 2002]?*
3. *Can the neural network argumentation compute the fixpoint semantics for argumentation?*
4. *Can symbolic argumentative dialogues be extracted from the neural network argumentation?*

The positive solutions to them helped us deeply understand relationship between symbolic and neural network argumentation, and further promote the syncretic approach of symbolism and connectionism in the field of computational argumentation [Makiguchi and Sawamura, 2007a][Makiguchi and Sawamura, 2007b]. They, however, paid attention only to the grounded semantics for argumentation in examining relationship between symbolic and neural network argumentation.

Ongoingly, we constructed a simple but versatile neural network for neural network argumentation, so that it can decide which argumentation semantics (admissible, stable, semi-stable semantics, preferred, complete, and grounded semantics) [Dung, 1995][Caminada, 2006] a given set of arguments falls into, and compute argumentation semantics via checking [Gotou, 2010]. In this paper, we are concerned with the opposite direction from neural network computation to symbolic argumentation/dialogue. We deal with the question how various argumentation semantics can have dialectical proof theories, and describe a possible answer to it by extracting or generating symbolic dialogues from the neural network computation under various argumentation semantics.

The results illustrate that there can exist an equal bidirectional relationship between the connectionism and symbolism in the area of computational argumentation. And also they lead to a fusion or hybridization of neural network computation and symbolic one [d'Avila Garcez *et al.*, 2009][Levine and Aparicio, 1994][Jagota *et al.*, 1999].

The paper is organized as follows. In the next section, we explicate our basic ideas on the neural network checking argumentation semantics by tracing an illustrative example. In Section 3, with our new construction of neural network for argumentation, we develop a dialectical proof theory induced by the neural network argumentation for each argumentation semantics by Dung [Dung, 1995]. In Section 4, we describe some related works although there is no work really related to our work except for Garcez et al.'s original one and our work. The final section discusses the major contribution of the paper and some future works.

2 Basic Ideas on the neural argumentation

Due to the space limitation, we will not describe the technical details for constructing a neural network for argumentation and its computing method in this paper (see [Gotou, 2010] for them). Instead, we illustrate our basic ideas by using a simple argumentation example and following a neural network computation trace for it. We assume readers are familiar with the Dungean semantics such as admissible, stable, semi-stable, preferred, complete, and grounded semantics [Dung, 1995][Caminada, 2006].

Let us consider an argumentation network on the left side of Figure 1 that is a graphic presentation of the argumentation framework $\mathcal{AF} =< AR, attacks >$, where $AR = \{i, k, j\}, and\ attacks = \{(i, k), (k, i), (j, k)\}$.

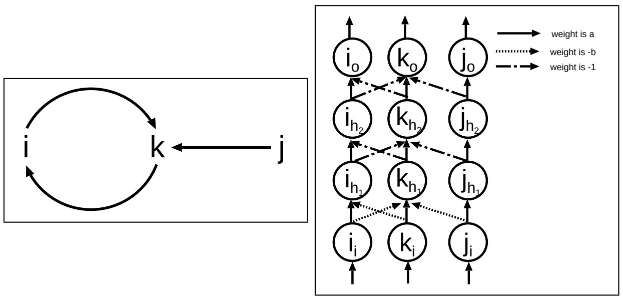


Figure 1: Graphic representation of $\mathcal{AF}$ (left) and Neural network translated from the $\mathcal{AF}$ (right)

According to the Dungean semantics [Dung, 1995][Caminada, 2006], the argumentation semantics for $\mathcal{AF}$ is determined as follows: Admissible set = $\{\emptyset, \{i\}, \{j\}, \{i, j\}\}$, Complete extension = $\{\{i, j\}\}$, Preferred extension = $\{\{i, j\}\}$, Semi-stable extension = $\{\{i, j\}\}$, Stable extension = $\{\{i, j\}\}$, and Grounded extension = $\{\{i, j\}\}$.

Neural network architecture for argumentation

In the Dungean semantics, the notions of 'attack', 'defend (acceptable)' and 'conflict-free' play the most important role in constructing various argumentation semantics. This is true in our neural network argumentation as well. Let $\mathcal{AF} =< AR, attacks >$ be as above, and S be a subset of AR, to be examined. The argumentation network on the left side of Figure 1 is first translated into the neural network on the right side of Figure 1. Then, the network architecture consists of the following constituents:

- A double hidden layer network: It is a double hidden layer network and has the following four layers: input layer, first hidden layer, second hidden layer and output layer, which have the ramified neurons for each argument, such as α_i, α_{h_1}, α_{h_2} and α_o for the argument α.
- A recurrent neural network (for judging grounded extension): The double hidden layer network like on the right side of Figure 1 is piled up high until the input and output layers converge (stable state) like in Figure 2. The symbol τ represents the pile number ($\tau \geq 0$) which amounts to the turning number of the input-output cycles of the neural network. In the stable state, we set $\tau = converging$. Then, $S_{\tau=n}$ stands for a set of arguments at $\tau = n$.
- A feedforward neural network (except judging grounded extension): When we compute argumentation semantics except grounded extension with a recurrent neural network, it surely converges at $\tau = 1$. Hence, the first output vector equals to second output vector. We judge argumentation semantics by using only first input vector and converged output vector. As a result we can regard a recurrent neural network as a feedforward neural network except judging grounded extension.
- The vectors of the neural network: The initial input vector for the neural network is a list consisting of 0 and **a** that represent the membership of a set of arguments to be examined. For example, it is $[\mathbf{a}, 0, 0]$ for $S = S_{\tau=0} = \{i\} \subseteq AR$. The output vectors from each layer take as the values only "**-a**", "0", "**a**" or "**-b**".[1] The intuitive meaning of them for each output vector are as follows:

Output layer

- "**a**" in the output vector from the output layer represents membership in $S'_\tau = \{X \in AR \mid defends(S_\tau, X)\}$[2] and the argument is not attacked by S'_τ.
- "**-a**" in the output vector from the output layer represents membership in S'^{+}_τ.[3]
- "0" in the output vector from the output layer represents the argument belongs to neither S'_τ nor S'^{+}_τ.

Second hidden layer

- "**a**" in the output vector from the second hidden layer represents membership in S'_τ and the argument is not attacked by S'_τ.
- "0" in the output vector from the second hidden layer represents membership not in S'_τ or the argument is attacked by S'_τ.

Fisrt hidden layer

- "**a**" in the output vector from the first hidden layer represents membership in S_τ and the argument is not attacked by S_τ.
- "**-b**" in the output vector from the first hidden layer represents the membership in S^{+}_τ.
- "0" in the output vector from the first hidden layer represents the others.

Input layer

- "**a**" in the output vector from the input layer represents membership in S_τ.
- "0" in the output vector from the input layer represents the argument does not belong to S.

A trace of the neural network

Let us examine to which semantics $S = \{i\}$ belongs in $\mathcal{AF}$ on the left side of Figure 1 by tracing the neural network computation. The overall visual computation flow is shown in Figure 2.

Stage1. Operation of input layer at $\tau = 0$

$S_{\tau=0} = S = \{i\}$. Hence, $[\mathbf{a}, 0, 0]$ is given to the input layer of the neural network in Figure 1. Each input neuron computes its output value by its activation function (see the graph of the activation function, an identity function, on the right side of the input layer of Figure 2). The activation function makes the input

[1] Let a,b be positive real numbers and they satisfy $\sqrt{b} > a > 0$.

[2] Let $S \subseteq AR$ and $A \in AR$. defends(S, A) iff $\forall B \in AR$(attacks(B, A) $\rightarrow$ attacks(S, B)).

[3] Let $S \subseteq AR$. $S^{+} = \{X \in AR \mid attacks(S, X)\}$.

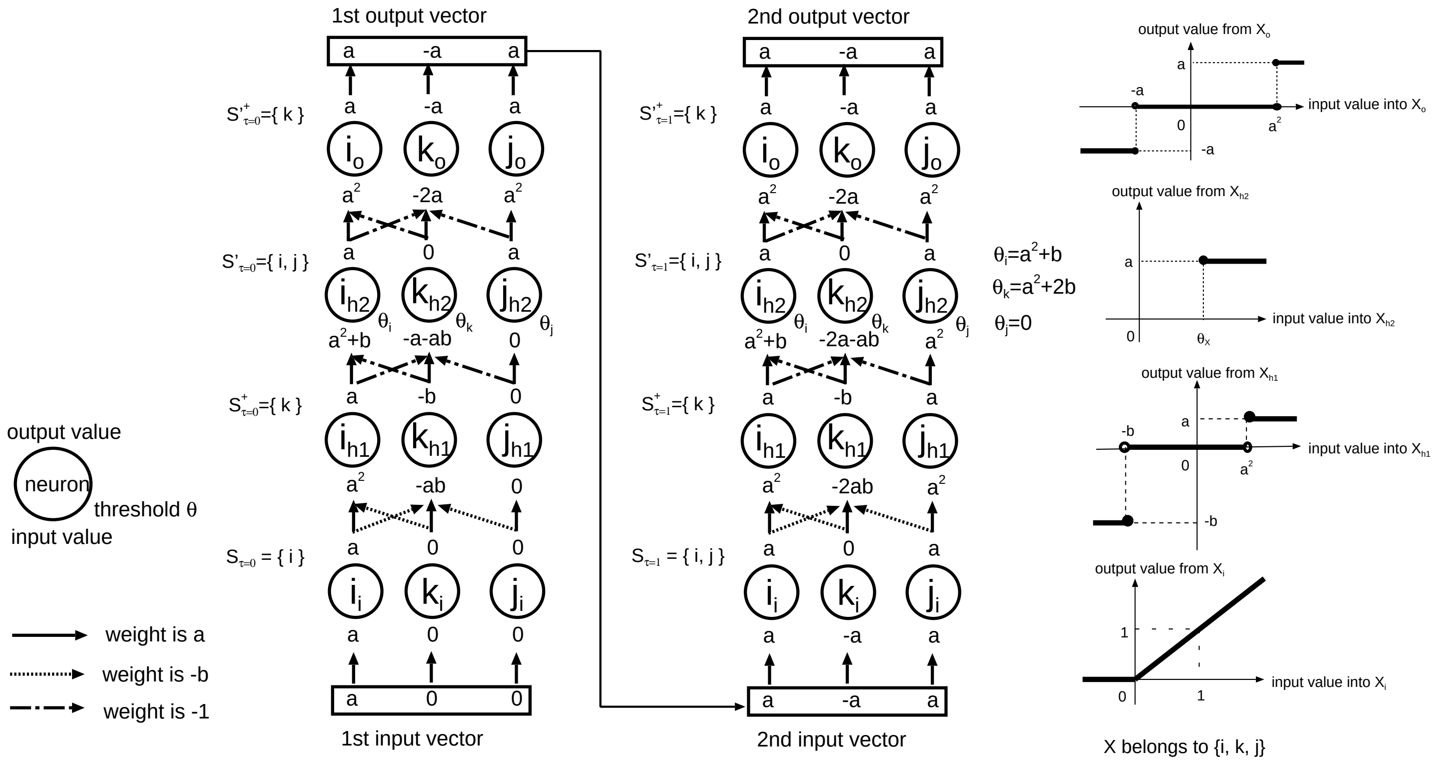


Figure 2: A trace of the neural network for argumentation with $S = \{i\}$ and activation functions

layer simply pass the value to the hidden layer. The input layer thus outputs the vector $[\mathbf{a}, 0, 0]$.

In this computation, the input layer judges $S_{\tau=0} = \{i\}$ and inputs $\mathbf{a}^2$ to i_{h_1} through the connection between i_i and i_{h_1} whose weight is $\mathbf{a}$. At the same time, the input layer inputs $-\mathbf{ab}$ to k_h through the connection between i_i and k_{h_1} whose weight is $-\mathbf{b}$ so as to make the first hidden layer know that $i \in S_{\tau=0}$ attacks k (in symbols, $attacks(i, k)$). Since the output values of k_i and j_i are 0, they input 0 to other first hidden neurons.

In summary, after the input layer receives the input vector $[\mathbf{a}, 0, 0]$, it turns out to give the hidden layer the vector [$\mathbf{a} \cdot \mathbf{a}$ + $0 \cdot (-\mathbf{b})$, $\mathbf{a} \cdot (-\mathbf{b})$ + $0 \cdot \mathbf{a}$ + $0 \cdot (-\mathbf{b})$, $0 \cdot \mathbf{a}$]= $[\mathbf{a}^2, -\mathbf{ab}, 0]$.

Stage 2. Operation of first hidden layer at $\tau = 0$

Now, the first hidden layer receives a vector $[\mathbf{a}^2, -\mathbf{ab}, 0]$ from the input layer. Each activation function of i_{h_1}, k_{h_1} and j_{h_1} is a step function as put on the right side of the first hidden layer in Figure 2. The activation function categorizes values of vectors which are received from the input layer into three values as if the function understand each argument state. Now, the following inequalitis hold: $\mathbf{a}^2 \geq \mathbf{a}^2$, $-\mathbf{ab} \leq -\mathbf{b}$, $-\mathbf{b} \leq 0 \leq \mathbf{a}^2$. According to the activation function, the first hidden layer outputs the vector $[\mathbf{a}, -\mathbf{b}, 0]$.

Next, the first hidden layer inputs $\mathbf{a}^2 + \mathbf{b}$ into the second hidden neuron i_{h_2} through the connections between i_{h_1} and i_{h_2} whose weight is $\mathbf{a}$, k_{h_1} and i_{h_2} whose weight is -1, so that the second hidden layer can know $attacks(k, i)$ with $i \in S_{\tau=0}$. At the same time, the first hidden layer inputs $-\mathbf{a} - \mathbf{ab}$ into k_{h_2} through the connections between i_{h_1} and k_{h_2} whose weight is -1, k_{h_1} and k_{h_2} whose weight is $\mathbf{a}$, so that the second hidden layer can know $attacks(i, k)$ with $k \in S^+_{\tau=0}$ and inputs 0 into j_{h_2} so that the second hidden layer can know the argument j is not attacked by any arguments with $j \notin S_{\tau=0}$.

In summary, after the first hidden layer received the vector $[\mathbf{a}^2, -\mathbf{ab}, 0]$, it turns out to pass the output vector $[\mathbf{a}^2 + \mathbf{b}, -\mathbf{a} - \mathbf{ab}, 0]$ to the second hidden neurons.

Stage 3. Operation of second hidden layer at $\tau = 0$

The second hidden layer receives a vector $[\mathbf{a}^2, -\mathbf{ab}, 0]$ from first hidden layer. Each activation function of i_{h_2}, k_{h_2} and j_{h_2} is a step function as put on the right side of the first hidden layer in Figure 2 with its threshold, $\theta_i = \mathbf{a}^2 + \mathbf{b}$, $\theta_k = \mathbf{a}^2 + 2\mathbf{b}$ and $\theta_j = 0$ respectively.

These thresholds are defined by the ways of being attacked as follows:

- If an argument X can defend X only by itself (in Figure 1, such X is i since $defends(\{i\}, i)$), then the threshold of X_{h_2} (θ_X) is $\mathbf{a}^2 + t\mathbf{b}$ (t is the number of arguments bilaterally attacking X).
- If an argument X can not defend X only by itself and is both bilaterally and unilaterally attacked by some other argument (in Figure 1, such X is k since $\neg defends(\{k\}, k) \& attacks(j, k) \& attacks(i, k)$), then the threshold of X_{h_2} (θ_X) is $\mathbf{a}^2 + \mathbf{b}(s + t)$ ($s(t)$ is the number of arguments unilaterally(bilaterally) attacking X). Note that l=m=1 for the argument k in Figure 1.
- If an argument X is not attacked by any other arguments (in Figure 1, such X is j), then the threshold of X_h (θ_{X_h}) is 0.
- If an argument X can not defend X only by itself and is just unilaterally attacked by some other argument, then the threshold of X_{h_2} (θ_X) is $\mathbf{b}s$ (s is the number of arguments unilaterally attacking X).

By these thresholds and their activation functions (step functions), if S defends X then X_{h_2} outputs $\mathbf{a}$. Otherwise, X_{h_2}

outputs 0 in the second hidden layer. As the result, the second hidden layer judges either $X \in S'_\tau$ or $X \notin S'_\tau$ by two output values ($\mathbf{a}$ and 0). In this way, the output vector in the second hidden layer yields $[\mathbf{a}, 0, \mathbf{a}]$. This vector means that the second hidden layer judges that the arguments i and j are defended by $S_{\tau=0}$, resulting in $S'_{\tau=0} = \{i, j\}$.

Next, the second hidden layer inputs $\mathbf{a}^2$ into the output neurons i_o and j_o through the connections between i_{h_2} and i_o, j_{h_2} and j_o whose weights are $\mathbf{a}$,so that the output layer can know $i, j \in S_{\tau=0}$ and $i, j \in S'_{\tau=0}$. At the same time, the second hidden layer inputs $-2\mathbf{a}$ into k_o through the connections between i_{h_2} and k_o, j_{h_2} and k_o whose weights are -1,so that output layer can know $attacks(i, k)$ and $attacks(j, k)$ with $k \in S'^{+}_{\tau=0}$.

Furthermore, it should be noted that another role of the second hidden layer lies in guaranteeing that S'_τ is conflict-free[4]. It is actually true since the activation function of the second hidden layer makes X_{h_2} for the argument X attacked by S_τ output 0. The conflict-freeness is important since it is another notion for characterizing the Dungean semantics.

In summary, after the second hidden layer received the vector $[\mathbf{a}^2 + \mathbf{b}, -\mathbf{a} - \mathbf{ab}, 0]$, it turns out to pass the output vector $[\mathbf{a}^2, -2\mathbf{a}, \mathbf{a}^2]$ to the second hidden neurons.

Stage 4. Operation of output layer at $\tau = 0$

The output layer now received the vector $[\mathbf{a}^2, -2\mathbf{a}, \mathbf{a}^2]$ from the second hidden layer. Each neuron in the output layer has an activation function as put on the right side of the output layer in Figure 2.

This activation function makes the output layer interpret any positive sum of input values into the output neuron X_o as $X \in S'_\tau$, any negative sum as $X \in S'^{+}_\tau$, and the value 0 as $X \notin S'_\tau$ and $X \notin S'^{+}_\tau$. As the result, the output layer outputs the vector $[\mathbf{a}, -\mathbf{a}, \mathbf{a}]$.

Summarizing the computation at $\tau = 0$, the neural network received the vector $[\mathbf{a}, 0, 0]$ in the input layer and outputted $[\mathbf{a}, -\mathbf{a}, \mathbf{a}]$ from the output layer. This output vector means that the second hidden layer judged $S'_{\tau=0} = \{i, j\}$ and guaranteed its conflict-freeness. With these information passed to the output layer from the hidden layer, the output layer judged $S'^{+}_{\tau=0} = \{k\}$.

Stage 5. Inputting the output vector at $\tau = 0$ to the input layer at $\tau = 1$ (shift from $\tau = 0$ to $\tau = 1$)

At $\tau = 0$, the neural network computed $S'_{\tau=0} = \{i, j\}$ and $S'^{+}_{\tau=0} = \{k\}$. We continue the computation recurrently by connecting the output layer to the input layer of the same neural network, setting first output vector to second input vector. Thus, at $\tau = 1$, the input layer starts its operation with the input vector $[\mathbf{a}, -\mathbf{a}, \mathbf{a}]$. We, however, omit the remaining part of the operations starting from here since they are to be done in the similar manner.

Stage 6. Convergence to a stable state

We stop the computation immediately after the time round $\tau = 1$ since the input vector to the neural network at $\tau = 1$ coincides with the output vector at $\tau = 1$. This means that the neural network amounts to having computed a least fixed point of the characteristic function that was defined with the acceptability of arguments by Dung [Dung, 1995].

[4]A set S of arguments is said to be conflict-free if there are no arguments A and B in S such that A attacks B.

Stage 7. Judging admissible set, complete extension and stable extension

Through the above neural network computation, we have obtained $S'_{\tau=0} = \{i, j\}$ and $S'^{+}_{\tau=0} = \{k\}$ for $S_{\tau=0} = \{i\}$, and $S'_{\tau=1} = \{i, j\}$ and $S'^{+}_{\tau=1} = \{k\}$ for $S_{\tau=1} = \{i, j\}$. Moreover, we also have such a result that both the sets $\{i\}$ and $\{i, j\}$ are conflict-free.

The condition for admissible set says that a set of arguments S satisfies its conflict-freeness and $\forall X \in AR(X \in S \rightarrow X \in S')$. Therefore, the neural network can know that the sets $\{i\}$ and $\{i, j\}$ are admissible since it confirmed the condition at the time round $\tau = 0$ and $\tau = 1$ respectively.

The condition for complete extension says that a set of arguments S satisfies its conflict-freeness and $\forall X \in AR(X \in S \leftrightarrow X \in S')$. Therefore, the neural network can know that the set $\{i, j\}$ satisfies the condition since it has been obtained at $\tau = converging$. Incidentally, the neural network knows that the set $\{i\}$ is not a complete extension since it does not appear in the output neuron at $\tau = converging$.

The condition for stable extension says that a set of arguments S satisfies $\forall X \in AR(X \notin S \rightarrow X \in S'^{+})$. The neural network can know that the $\{i, j\}$ is a stable extension since it confirmed the condition from the facts that $S_{\tau=1} = \{i, j\}$, $S'_{\tau=1} = \{i, j\}$ and $S'^{+}_{\tau=1} = \{a\}$.

Stage 8. Judging preferred extension, semi-stable extension and grounded extension

By invoking the neural network computation that was stated from the stages 1-7 above for every subset of AR, and AR itself as an input set S, it can know all admissible sets of $\mathcal{AF}$, and hence it also can know the preferred extensions of $\mathcal{AF}$ by picking up the maximal ones w.r.t. set inclusion from it. In addition, the neural network can know semi-stable extensions by picking up a maximal $S \cup S^{+}$ where S is a complete extension in $\mathcal{AF}$. This is possible since the neural network already has computed S^{+}.

For the grounded extension, the neural network can know that the grounded extension of $\mathcal{AF}$ is $S'_{\tau=converging}$ when the computation stopped by starting with $S_{\tau=0} = \emptyset$. This is due to the fact that the grounded extension is obtained by the iterative computation of the characteristic function that starts from $\emptyset$ [Prakken and Vreeswijk, 2002].

Readers should refer to the paper [Gotou, 2010] for the soundness theorem of the neural network computation illustrated so far.

3 Extracting Symbolic Dialogues from the Neural Network

In this section, we will address to such a question as if symbolic argumentative dialogues can be extracted from the neural network argumentation. The symbolic presentation of arguments would be much better for us since it makes the neural net argumentation process verbally understandable. The notorious criticism for neural network as a computing machine is that connectionism usually does not have explanatory reasoning capability. We would say our attempt here is one that can turn such criticism in the area of argumentative reasoning.

In our former paper [Makiguchi and Sawamura, 2007b], we have given a method to extract symbolic dialogues from the neural network computation under the grounded semantics, and showed its coincidence with the dialectical proof theory for the grounded semantics. In this paper, we are concerned with the question how other argumentation semantics can have dialectical proof theories. We describe a possible answer to it by extracting or generating symbolic dialogues from the neural network computation under other more complicated argumentation semantics. We would say this is a great success that was brought by our neural network approach to argumentation since dialectical proof theories for various Dungean argumentation semantics have not been known so far except only some works (e. g., [Vreeswijk and Prakken, 2000], [Dung *et al.*, 2006]).

First of all, we summarize the trace of the neural network computation as have seen in Section 2 as in Table 1, in order to make it easy to extract symbolic dialogues from our neural network. Wherein, $S_{PRO,\tau=k}$ and $S_{OPP,\tau=k}$ denote the followings respectively: At time round $\tau = k (k \geq 0)$ in the neural network computation, $S_{PRO,\tau=k} = S'_{\tau=k}$, and $S_{OPP,\tau=k} = S'^{+}_{\tau=k}$ (see Section 2 for the notations).

Table 1: Summary table of the neural network computation

		$S_{PRO,\tau=k}$	$S_{OPP,\tau=k}$
$\tau = 0$	input	S	{}
	output	...	...
$\tau = 1$	input	...	...
	output	...	...
$\vdots$	$\vdots$	...	...

Table 2: Summary table of the neural network computation in Fig. 2

		$S_{PRO,\tau=k}$	$S_{OPP,\tau=k}$
$\tau = 0$	input	{i}	{}
	output	{i, j}	{k}
$\tau = 1$	input	{i, j}	{k}
	output	{i, j}	{k}

For example, Table 2 is the table for $S = \{i\}$ summarized from the neural network computation in Fig. 2

We assume dialogue games are performed by proponents (PRO) and opponents (OPP) who have their own sets of arguments that are to be updated in the dialogue process. In advance of the dialogue, proponents have $S (= S_{\tau=0})$ as an initial set $S_{PRO,\tau=0}$, and opponents have an empty set {} as an initial set $S_{OPP,\tau=0}$.

We illustrate how to extract dialogues from the summary table by showing a concrete extraction process of dialogue moves in Table 2:

1. P(roponent, speaker): PRO declares a topic as a set of beliefs by saying $\{i\}$ at $\tau = 0$. OPP just hears it with no response {} for the moment. (dialogue extraction from the first row of Table 2)
2. P(roponent, or speaker): PRO further asserts the incremented belief $\{i, j\}$ because the former beliefs defend j, and at the same time states the belief $\{i, j\}$ conflicts with $\{k\}$ at $\tau = 0$. (dialogue extraction from the second row of Table 2)
3. O(pponent, listener or audience): OPP knows that its belief $\{k\}$ conflicts with PRO's belief $\{i, j\}$ at $\tau = 0$. (dialogue extraction from the second row of Table 2)
4. No further dialogue moves can be promoted at $\tau = 1$, resulting in a stable state. (dialogue termination by the third and fourth rows of Table 2)

Thus, we can view P(roponent, speaker)'s initial belief $\{i\}$ as justified one in the sense that it could have persuaded O(pponent, listener or audience) under an appropriate Dungean argumentation semantics. Actually, we would say it is admissibly justified under admissibly dialectical proof theory below. Formally, we introduce the following dialectical proof theories, according to the respective argumentation semantics.

Definition 1 (**Admissibly dialectical proof theory**) *The admissibly dialectical proof theory is the dialogue extraction process in which the summary table generated by the neural network computation satisfies the following condition:* $\forall A \in S_{PRO,\tau=0}\ \forall k \geq 0 (A \in S_{PRO,\tau=k})$, *where* $S_{PRO,\tau=0}$ *is the input set at* $\tau = 0$.

Intuitively, the condition says every argument in $S_{PRO,\tau=0}$ is retained until the stable state as can be seen in Table 2. It should be noted that the condition reflects the definition of 'admissible extension' in [Dung, 1995].

Definition 2 (**Completely dialectical proof theory**) *The completely dialectical proof theory is the dialogue extraction process in which the summary table generated by the neural network computation satisfies the following conditions: let* $S_{PRO,\tau=0}$ *be the input set at* $\tau = 0$.

1. $S_{PRO,\tau=0}$ *satisfies the condition of Definition 1.*
2. $\forall A \notin S_{PRO,\tau=0}\ \forall k (A \notin S_{PRO,\tau=k})$

Intuitively, the second condition says that any argument that does not belong to $S_{PRO,\tau=0}$ does not enter into $S_{PRO,\tau=t}$ at any time round t up to a stable one k. Those conditions reflect the definition of 'complete extension' in [Dung, 1995].

Definition 3 (**Stably dialectical proof theory**) *The stably dialectical proof theory is the dialogue extraction process in which the summary table generated by the neural network computation satisfies the following conditions: let* $S_{PRO,\tau=0}$ *be the input set at* $\tau = 0$.

1. $S_{PRO,\tau=0}$ *satisfies the conditions of Definition 2.*
2. $AR = S_{PRO,\tau=n} \cup S_{OPP,\tau=n}$, *where* $\mathcal{AF} =< AR, attacks >$ *and n denotes a stable time round.*

Intuitively, the second condition says that PRO and OPP cover AR exclusively and exhaustively. Those conditions reflect the definition of 'stable extension' in [Dung, 1995].

For the dialectical proof theories for preferred [Dung, 1995] and semi-stable semantics [Caminada, 2006], we can similarly define them taking into account maximality condition. So we omit them in this paper.

As a whole, the type of the dialogues in any dialectical proof theories above would be better classified as a persuasive dialogue since it is closer to persuasive dialogue in the dialogue classification by Walton [Walton, 1998].

4 Related Work

Garcez et al. initiated a novel approach to argumentation, called the neural network argumentation [d'Avila Garcez *et al.*, 2005]. However, the semantic analysis for it is missing there. That is, it is not clear what they calculate by their neural network argumentation. Besnard et al. proposed three symbolic approaches to checking the acceptability of a set of arguments [Besnard and Doutre, 2004], in which not all of the Dungean semantics can be dealt with. So it may be fair to say that our approach with the neural network is more powerful than Besnard et al.'s methods.

Vreeswijk and Prakken proposed a dialectical proof theory for the preferred semantics [Vreeswijk and Prakken, 2000]. It is similar to that for the grounded semantics [Prakken and Sartor, 1997], and hence can be simulated in our neural network as well.

In relation to the neural network construction and computation for the neural-symbolic systems, the structure of the neural network is a similar 3-layer recurrent network, but our neural network computes not only the least fixed point (grounded semantics) but also the fixed points (complete extension). This is a most different aspect from Hölldobler and his colleagues' work [Hölldobler and Kalinke, 1994].

5 Concluding Remarks

It is a long time since connectionism appeared as an alternative movement in cognitive science or computing science which hopes to explain human intelligence or soft information processing. It has been a matter of hot debate how and to what extent the connectionism paradigm constitutes a challenge to classicism or symbolic AI. In this paper, we showed that symbolic dialectical proof theories can be obtained from the neural network computing various argumentation semantics, which allow to extract or generate symbolic dialogues from the neural network computation under various argumentation semantics. The results illustrate that there can exist an equal bidirectional relationship between the connectionism and symbolism in the area of computational argumentation. On the other hand, much effort has been devoted to a fusion or hybridization of neural net computation and symbolic one [d'Avila Garcez *et al.*, 2009][Levine and Aparicio, 1994][Jagota *et al.*, 1999]. The result of this paper as well as our former results on the hybrid argumentation [Makiguchi and Sawamura, 2007a][Makiguchi and Sawamura, 2007b] yields a strong evidence to show that such a symbolic cognitive phenomenon as human argumentation can be captured within an artificial neural network.

The simplicity and efficiency of our neural network may be favorable to our future plan such as introducing learning mechanism into the neural network argumentation, implementing the neural network engine for argumentation, which can be used in argumentation-based agent systems, and so on. Specifically, it might be possible to take into account the so-called core method developed in [Hölldobler and Kalinke, 1994] and CLIP in [d'Avila Garcez *et al.*, 2009] although our neural-symbolic system for argumentation is much more complicated due to the complexities and varieties of the argumentation semantics.

References

[Besnard and Doutre, 2004] Philippe Besnard and Sylvie Doutre. Checking the acceptability of a set of arguments. In *10th International Workshop on Non-Monotonic Reasoning (NMR 2004*, pages 59–64, 2004.

[Caminada, 2006] Martin Caminada. Semi-stable semantics. In Paul E. Dunne and Trevor J. M. Bench-Capon, editors, *Computational Models of Argument: Proceedings of COMMA 2006*, volume 144 of *Frontiers in Artificial Intelligence and Applications*, pages 121–130. IOS Press, 2006.

[d'Avila Garcez *et al.*, 2005] Artur S. d'Avila Garcez, Dov M. Gabbay, and Luis C. Lamb. Value-based argumentation frameworks as neural-symbolic learning systems. *Journal of Logic and Computation*, 15(6):1041–1058, 2005.

[d'Avila Garcez *et al.*, 2009] Artur S. d'Avila Garcez, Luís C. Lamb, and Dov M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Springer, 2009.

[Dung *et al.*, 2006] P. M. Dung, R. A. Kowalski, and F. Toni. Dialectic proof procedures for assumption-based, admissible argumentation. *Artificial Intelligence*, 170:114–159, 2006.

[Dung, 1995] P.M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logics programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[Gotou, 2010] Yoshiaki Gotou. Neural Networks calculating Dung's Argumentation Semantics. Master's thesis, Graduate School of Science and Technology, Niigata University, Niigata, Japan, December 2010. `http://www.cs.ie.niigata-u.ac.jp/Paper/Storage/graguation_thesis_gotou.pdf`.

[Hölldobler and Kalinke, 1994] Steffen Hölldobler and Yvonne Kalinke. Toward a new massively parallel computational model for logic programming. In *Proc. of the Workshop on Combining Symbolic and Connectionist Processing, ECAI 1994*, pages 68–77, 1994.

[Jagota *et al.*, 1999] Arun Jagota, Tony Plate, Lokendra Shastri, and Ron Sun. Connectionist symbol processing: Dead or alive? *Neural Computing Surveys*, 2:1–40, 1999.

[Levine and Aparicio, 1994] Daniel Levine and Manuel Aparicio. *Neural Networks for Knowledge Representation and Inference*. LEA, 1994.

[Makiguchi and Sawamura, 2007a] Wataru Makiguchi and Hajime Sawamura. A Hybrid Argumentation of Symbolic and Neural Net Argumentation (Part I). In *Argumentation in Multi-Agent Systems, 4th International Workshop, ArgMAS 2007, Revised Selected and Invited Papers*, volume 4946 of *Lecture Notes in Computer Science*, pages 197–215. Springer, 2007.

[Makiguchi and Sawamura, 2007b] Wataru Makiguchi and Hajime Sawamura. A Hybrid Argumentation of Symbolic and Neural Net Argumentation (Part II). In *Argumentation in Multi-Agent Systems, 4th International Workshop, ArgMAS 2007, Revised Selected and Invited Papers*, volume 4946 of *Lecture Notes in Computer Science*, pages 216–233. Springer, 2007.

[Prakken and Sartor, 1997] H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *J. of Applied Non-Classical Logics*, 7(1):25–75, 1997.

[Prakken and Vreeswijk, 2002] H. Prakken and G. Vreeswijk. Logical systems for defeasible argumentation. In *In D. Gabbay and F. Guenther, editors, Handbook of Philosophical Logic*, pages 219–318. Kluwer, 2002.

[Rahwan and Simari, 2009] Iyad Rahwan and Guillermo R. (Eds.) Simari. *Argumentation in Artificial Intelligence*. Springer, 2009.

[Vreeswijk and Prakken, 2000] Gerard A. W. Vreeswijk and Henry Prakken. Credulous and sceptical argument games for preferred semantics. *Lecture Notes in Computer Science*, 1919:239–??, 2000.

[Walton, 1998] D. Walton. *The New Dialectic: Conversational Contexts of Argument*. Univ. of Toronto Press, 1998.

Visual Intelligence using Neural-Symbolic Learning and Reasoning

H.L.H. (Leo) de Penning
TNO Behaviour and Societal Sciences
Kampweg 5, Soesterberg, The Netherlands.
leo.depenning@tno.nl

Abstract

The DARPA Mind's Eye program seeks to develop in machines a capability that currently exists only in animals: visual intelligence. This short paper describes the initial results of a Neural-Symbolic approach for action recognition and description to be demonstrated at the 7th international workshop on Neural-Symbolic Learning and Reasoning.

Introduction

Humans in particular perform a wide range of visual tasks with ease, which no current artificial intelligence can do in a robust way. Humans have inherently strong spatial judgment and are able to learn new spatiotemporal concepts directly from the visual experience. Humans can visualize scenes and objects, as well as the actions involving those objects. Humans possess a powerful ability to manipulate those imagined scenes mentally to solve problems. A machine-based implementation of such abilities would require major advances in each of the following technology focus areas: Robust recognition, Anomaly detection, Description, Gap-filling (i.e., interpolation, prediction, and post diction). These are human intelligence-inspired capabilities, which are envisaged in service of systems to directly support humans in complex perceptual and reasoning tasks (e.g. like Unmanned Ground Vehicles).

The DARPA Mind's Eye program seeks to develop in machines a capability that currently exists only in animals: visual intelligence [Donlon, 2010]. In particular, this program pursues the capability to learn generally applicable and generative representations of action between objects in a scene, directly from visual inputs, and then reason over those learned representations. A key distinction between this research and the state of the art in machine vision is that the latter has made continual progress in recognizing a wide range of objects and their properties—what might be thought of as the nouns in the description of a scene. The focus of Mind's Eye is to add the perceptual and cognitive underpinnings for recognizing and reasoning about the verbs in those scenes, enabling a more complete narrative of action in the visual experience.

The contribution of TNO, a Dutch research institute and one of the teams working on the Mind's Eye program, is called CORTEX and is presented in this paper. CORTEX is a Visual Intelligence (VI) system and consists of a visual processing pipeline and reasoning component that is able to reason about events detected in visual inputs (e.g. from a movie or live camera) in order to; i) recognize actions in terms of verbs, ii) describe these actions in natural language, iii) detect anomalies and iv) fill gaps (e.g. video blackouts by missing frames, occlusion by moving objects, or entities receding behind objects).

Neural-Symbolic Cognitive Agent

To learn spatiotemporal relations between detected events (e.g. size of bounding boxes, speed of moving entities, changes in relative distance between entities) and verbs describing actions (e.g. fall, bounce, dig) the reasoning component uses a Neural-Symbolic Cognitive Agent (NSCA) that is based on a Recurrent Temporal Restricted Boltzmann Machine (RTRBM) (described in [de Penning et al., 2011] and presented during the IJCAI 2011 poster session). This cognitive agent is able to learn hypotheses about temporal relations between observed events and related actions and can express those hypotheses in temporal logic or natural language. This enables the reasoning component, and thus CORTEX, to explain and describe the cognitive underpinnings of the recognition task as stated in the focus of the Mind's Eye program.

The hypotheses are modelled in a RTRBM, where each hidden unit H_j in the RTRBM represents a hypothesis about a specific relation between events e and verbs v being observed in the visible layer V and hypotheses h^{t-1} that have been true in the previous time frame. Based on a Bayesian inference mechanism, the NSCA can reason about observed actions by selecting the most likely hypotheses h using random Gaussian sampling of the posterior probability distribution (i.e. $h \sim P(H|V=e \wedge v, H^{t-1}=h^{t-1})$) and calculating the conditional probability or likelihood of all events and verbs assuming the selected hypotheses are true (i.e. $P(V|H=h)$). The difference between the detected events, available ground truth and the inferred events and verbs can be used by the NSCA to train the RTRBM (i.e. update its weights) in order to improve the hypotheses.

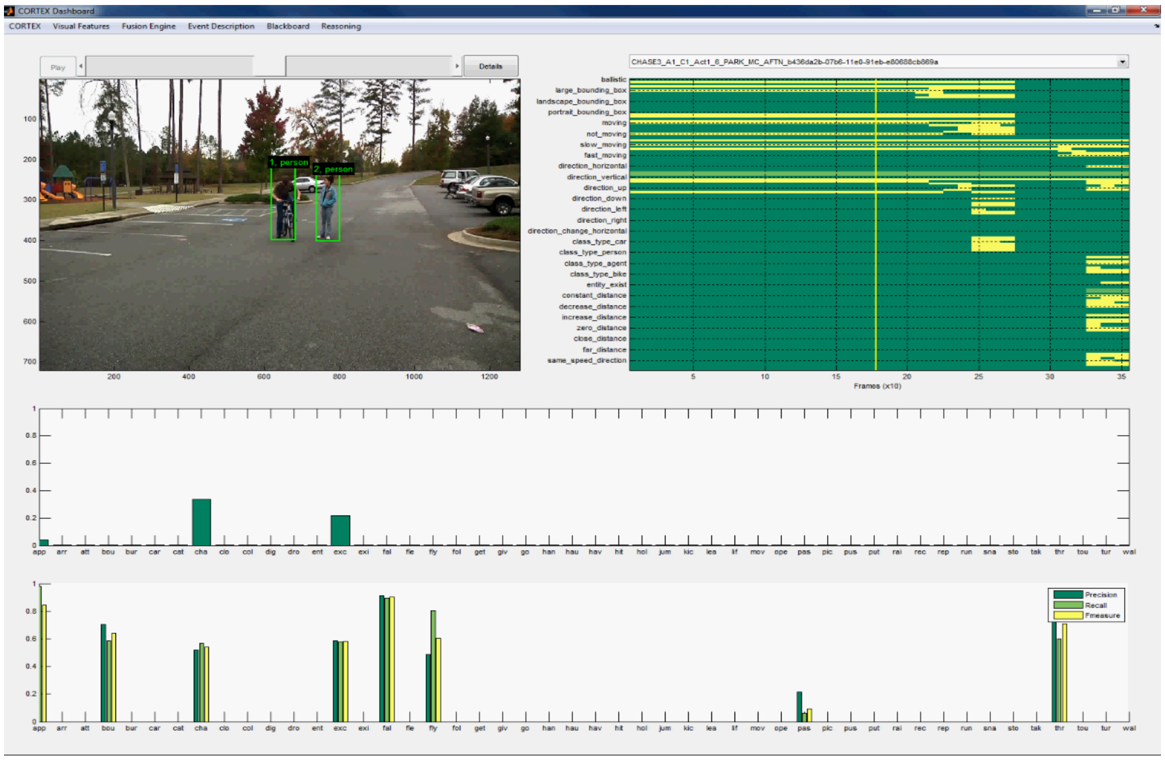


Figure 1. CORTEX Dashboard for testing and evaluation.

Experiments and Results

The CORTEX system and its reasoning component are currently being tested on a recognition task using several datasets of movies and related ground truth provided by DARPA. Figure 1 shows the CORTEX Dashboard, a user-interface for testing and evaluation of the CORTEX system. With the CORTEX Dashboard we are able to visualize the detected entities (depicted by bounding boxes in the upper-left image), probabilities on related events in each frame (depicted by intensities, green is 0 and yellow is 1, in the upper-right graph) and related verb probabilities calculated by the reasoning component (depicted by bars in the centre graph). The bottom graph shows the precision, recall and F-measure (i.e. harmonic mean of the precision and recall) for all verbs used to evaluate the output of the reasoning component. Also it can visualize the learned hypotheses and extract these in the form of temporal logic or natural language, which can be used to explain and describe the recognized actions.

Initial results show that the reasoning component is capable of learning hypotheses about events and related verbs and that it is able to reason with these hypotheses to correctly recognize actions based on detected events.
Furthermore the results show that the reasoning component is able to recognize actions that were not there in the ground truth for that specific input, but inferred from ground truth and related event patterns in other input. For example, reasoning about a movie that was trained to be recognized as a chase, resulted in some parts being recognized as fall, because one of the persons was tilting over when she started running, although fall was not part of the ground truth for this movie.

Conclusions and Future Work

With the NSCA architecture, the reasoning component is able to learn and reason about spatiotemporal events in visual inputs and recognize these in terms of actions denoted by verbs. It is also able to extract learned hypotheses on events and verbs that can be used to explain the perceptual and cognitive underpinnings of the recognition task and support other visual intelligence tasks, like description, anomaly detection and gap-filling, yet to be developed in the CORTEX system.

References

[Donlon, 2010] James Donlon. *DARPA Mind's Eye Program: Broad Agency Announcement*. Arlington, USA, 2010.

[de Penning et al., 2011] Leo de Penning, Artur S. d'Avila Garcez, Luís C. Lamb, and John-Jules C. Meyer. A Neural-Symbolic Cognitive Agent for Online Learning and Reasoning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Barcelona, Spain, 2011.

Neural-Symbolic Learning:How to play Soccer

Silvano Colombo Tosatto
University of Luxembourg
silvano.colombotosatto@uni.lu

Abstract

In the present extended abstract we describe the simulator developed. The purpose of this simulator is to test our neural symbolic approach towards normative reasoning.

After the translation process provided by the simulator I describe one of the case study used during the experiments. To be more precise, the case study regards RoboCup scenario.

1 The Simulator

The task of the simulator is to build a neural network starting from a knowledge base. For the translation, the simulator uses an approach similar to the one described in the book[d'Avila Garcez *et al.*, 2002]. More precisely the approach used is the one described in [Boella *et al.*, 2011a].

The simulator has been developed using *Java*[1]. To develop the necessaries functions for the neural networks and the networks themselves, *Joone*[2]. Joone is a object oriented neural engine, a framework written in Java that collects the functions needed to build neural networks and work with them.

2 Translation process

The translation process receives in input two *XML*[3] files. The first file contains the lists of input and output perceptrons to be included in the neural network. The second file contains the rules that have to be translated within the neural network by the *N-CILP* algorithm described in [Boella *et al.*, 2011a].

The input file containing the lists of inputs and outputs is needed for training purposes. Otherwise the neural network would be constructed using the inputs and outputs used in the rules inside the knowledge base, if this is the case, then the neural network would not be able to learn new rules containing literals which are not already known.

The knowledge base file contains I/O logic rules, shaped in XML format. The priority relationships between the rules are encoded within the rules themselves as described in [Boella *et al.*, 2011a].

[1]http://www.java.com/en/

[2]http://sourceforge.net/projects/joone/

[3]http://www.w3.org/XML/

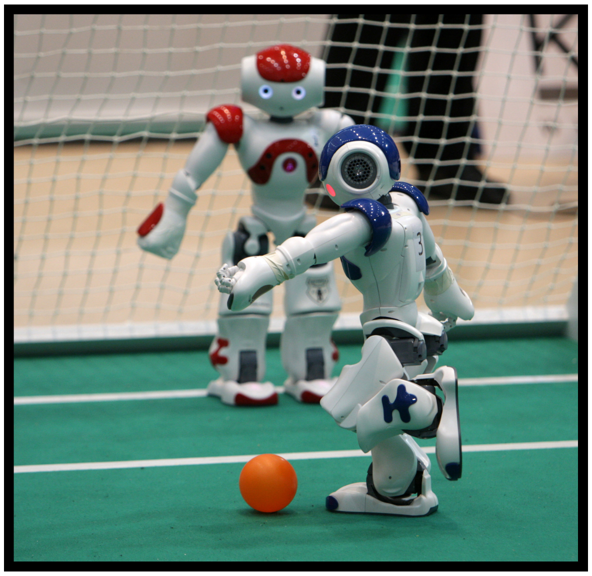


Figure 1: A snapshot of a RoboCup match.

With the mentioned files the simulator builds the neural network following the N-CILP algorithm. In addition, the simulator allows some optional translation features. The first option allows to set the weights calculated by N-CILP as untrainable. By activating this option, for the resulting neural network would be very difficult to forget the starting rules. Anyway a possible drawback of this approach is that would be difficult to forget *wrong* rules. The second option allows to randomize the weights and thresholds of the network[4]. The principal use of this option is to generate non-symbolic networks and compare their performances with the ones built with N-CILP.

3 Neural Network

The network built by the N-CILP algorithm uses a step activation function for the perceptrons in the input level, because they just need to pass the information from the inputs to the following layer. For the hidden and output level instead, the

[4]The number of hidden perceptrons is defined by the N-CILP algorithm.

perceptrons used adopts an bipolar sigmoid activation function [Karlik and Olgac, 2010].

After being built, the neural network can be used to process data. Thanks to the translation process, the neural network is capable to process the data by following the rules contained in the initial knowledge base.

In order to cope with a dynamically changing environment, the network can be trained in a supervised fashion. By instance learning the network is capable to extend its knowledge and after the training. In this way the trained neural network, should be able to correctly process data which the starting knowledge base is not.

4 Case Study

One of the case studies used with the simulator involves the RoboCup scenario (Figure 1). In our work described in [Boella *et al.*, 2011a] we focused on the normative aspect of the game. For this reason, we have took into consideration some of the rules that the robots had to follow. We used the rulings used in 2007 contained in [Menegatti, 2007].

4.1 Rules of the RoboCup

In this section we will take a look at some of the rules that the robots should follow in order to play properly. I represent the rules using the I/O logic format [Makinson and van der Torre, 2000] with modalities as described in [Boella *et al.*, 2011a].

$R_1 : (\top, \mathbf{O}(\neg impact\_opponent))$

$R_2 : (\top, \mathbf{O}(\neg use\_hands))$

$R_3 : (goalkeeper \wedge inside\_own\_area, \mathbf{P}(use\_hands))$

$R_4 : (kickoff, \mathbf{O}(\neg score))$

$R_5 : (kickoff \wedge mate\_touch\_ball, \mathbf{P}(score))$

The first rule refers to the prohibition to voluntarily impact into an opponent.

The second rule also states a prohibition, the interdiction to use the hands to play the ball. Differently, the third rule says that the goalkeeper can use its hands inside its own goal area.

The fourth rule refers to the prohibition to score from the kickoff. The last rule, the fifth, is related with the fourth. It states that it is permitted to score in a kickoff situation if a team mate touches the ball.

4.2 Adding dynamism in the system

By only considering the rules of the game, the system is static. This because the rules of the game does not change during a match. In order to include dynamism, we add to the system another ruling element[5]. The additional ruling element is represented by the *coach*.

The coach gives directions to the robot about how they should play during the match. The situation during the match can change, and the coach may want to change accordingly the strategies of the robot team. To do so, the coach can add additional rules to the knowledge base or retract some of them.

[5]The first ruling element is the *referee* which enforces the rules of the game.

In the following list we take a look at some of the possible rules that a coach may enforce.

$R_6 : (ball \wedge close\_to\_opponent\_goal, \mathbf{O}(shoot))$

$R_7 : (ball \wedge opponent\_approaching, \mathbf{O}(pass))$

$R_8 : (ball \wedge opponent\_approaching \wedge team\_mate\_marked, \mathbf{O}(\neg pass))$

$R_9 : (opponent\_shooting, \mathbf{O}(impact\_opponent))$

The sixth rule refers to the ought to try to score when a robot with the ball, is close to the opponent's goal.

The seventh rule states the obligation to pass the ball if an opponent is approaching. Differently, the eight rule states the prohibition to pass, if the same condition holds and additionally the team mate is marked by an opponent. In this case the rules can be ordered by a priority relation in order to avoid *dilemmas*[Boella *et al.*, 2011b], like $R_8 \succ R_7$.

The last rule, the ninth, states that a robot should try to prevent an opponent to scoring by impacting into him if it is shooting. This rule clearly goes against R_1 given by the referee. Even in this case is possible to enforce the decision of the robot with a priority (like $R_9 \succ R_1$) or leave at the robot both the possibilities available ($impact\_opponent$ or $\neg impact\_opponent$).

Acknowledgements

Silvano Colombo Tosatto is supported by the National Research Fund, Luxembourg.

References

[Boella *et al.*, 2011a] Guido Boella, Silvano Colombo Tosatto, Artur S. d'Avila Garcez, Valerio Genovese, and Leendert van der Torre. Embedding normative reasoning into neural symbolic systems. In *7th International Workshop on Neural-Symbolic Learning and Reasoning*, 2011.

[Boella *et al.*, 2011b] Guido Boella, Silvano Colombo Tosatto, Artur S. d'Avila Garcez, Dino Ienco, Valerio Genovese, and Leendert van der Torre. Neural symbolic systems for normative agents. In *10th International Conference on Autonomous Agents and Multiagent Systems*, 2011.

[d'Avila Garcez *et al.*, 2002] Artur S. d'Avila Garcez, Krysia B. Broda, and Dov M. Gabbay. *Neural-Symbolic Learning Systems*. Perspectives in Neural Computing. Springer, 2002.

[Karlik and Olgac, 2010] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1:111–122, 2010.

[Makinson and van der Torre, 2000] David Makinson and Leendert van der Torre. Input-output logics. *Journal of Philosophical Logic*, 29, 2000.

[Menegatti, 2007] Emanuele Menegatti. Robocup soccer humanoid league rules and setup, 2007.

Extracting both MofN rules and if-then rules from the training neural networks

Norbert Tsopze[1,4] **- Engelbert Mephu Nguifo**[2,3] **- Gilbert Tindo**[1]
[1] Department of Computer Science - Faculty of Science - University of Yaounde I, Cameroon
[2] Clermont Université, Université Blaise Pascal, LIMOS, BP 10448, F-63000 Clermont-Ferrand, France
[3] CNRS, UMR 6158, LIMOS, F-63173 Aubiére, France
[4] CRIL-CNRS UMR 8188, Université Lille-Nord de France, Artois, SP 16, F-62307 Lens, France

1 Introduction

Artificial Neural Networks classifiers have many advantages such as: noise tolerance, possibility of parallelization, better training with a small quantity of data .... Coupling neural networks with an explanation component will increase its usage for those applications. The explanation capacity of neural networks is solved by extracting knowledge incorporated in the trained network [Andrews *et al.*, 1995]. We consider a single neuron (or perceptron) with Heaviside map as activation function ($f(x) = 0\ if\ x < 0\ else\ 1$). For a given perceptron with the connection weights vector W and the threshold θ, this means finding the different states where the neuron is active (wich could be reduced to the Knapsack problem.

With the existing algorithms, two forms of rules are reported in the literature : 'If $(condition)$ then $conclusion$' form noted 'if then' rules, 'If (m of a set of $conditions$) then $conclusion$' form noted '$MofN$'

The intermediate structures that we introduce are called MaxSubset list and generator list. The MaxSubset is a minimal structure used to represent the if-then rules while the generator list is some selected MaxSubsets from which we can derive all MaxSubsets and all MofN rules. We introduce heuristics to prune and reduce the candidate search space. These heuristics consist of sorting the incoming links according to the descending order, and then pruning the search space using the subset cardinality bounded by some determined values.

2 The MaxSubsets and generators Rules extraction approach

The general form of a MofN rule is 'if $m_1\ of\ N_1 \wedge m_2\ of\ N_2 \wedge ... \wedge m_p\ of\ N_p$ then conclusion' or '$\bigwedge_i (m_i\ of\ N_i)$ then conclusion'; for each subset N_i of the inputs set, if m_i elements are verified, the conclusion is true.

The common limit of previous approaches is the exclusive form of the extracted rules. Thus we introduce a novel approach called MaxSubset from which it is possible to generate both forms of rules. The MaxSubset approach follows operations (3), (4) and (5) of figure 1; while the existing known algorithms follow the path (1) for the if-then rules and (2) for the MofN rules. The processes (3), (4) and (5) of the figure 1 are described as follows: (3) MaxSubsets and generators extraction; (4) generation of if-then rules from the MaxSubsets list and (5) generation of MofN rules from the generators list. An extended version of this work is described in [Tsopze *et al.*, 2011].

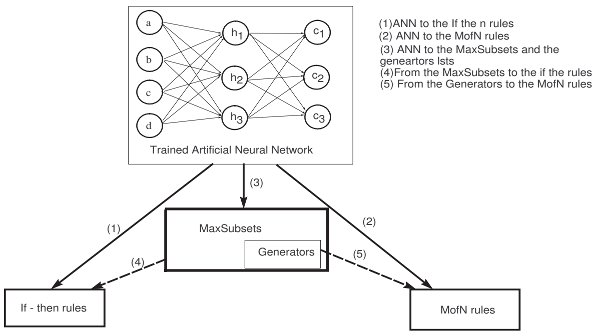

a
b
c
d
h_1
h_2
h_3
c_1
c_2
c_3
Trained Artificial Neural Network
(1)ANN to the If the n rules
(2) ANN to the MofN rules
(3) ANN to the MaxSubsets and the geneartors lsts
(4)From the MaxSubsets to the if the rules
(5) From the Generators to the MofN rules
(3)
(1)
(2)
MaxSubsets
Generators
(4)
(5)
If - then rules
MofN rules

Figure 1: Rules extraction process:

3 Conclusion

This approach consists in extracting a minimal list of elements called MaxSubset list, and then generating rules in one of the standard forms : if-then or MofN. To our knowledge, it is the first approach which is able to propose to the user a generic representation of rules from which it is possible to derive both forms of rules.

Acknowledgments

This work is partially supported by the French Embassy in Cameroon, under the first author's PhD grant provided by the French Embassy Service SCAC (Service de Coopération et d'Action Culturelle).

References

[Andrews *et al.*, 1995] R. Andrews, J. Diederich, and A. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networkss. *Knowledge-Based Systems*, 8(6):373–389, 1995.

[Tsopze *et al.*, 2011] N. Tsopze, E. Mephu Nguifo, and G. Tindo. Towards a generalization of decompositional approach of rules extraction from network. In *Proceeding IJCNN'11 international joint conference on Neural Networks*, 2011.

SHERLOCK - An Inteface for Neuro-Symbolic Networks[*]

Ekaterina Komendantskaya and Qiming Zhang
Schoool of Computing, University of Dundee, Dundee, Scotland

Abstract

We propose SHERLOCK - a novel problem-solving application based on neuro-symbolic networks. The application takes a knowledge base and rules in the form of a logic program, and compiles it into a connectionist neural network that performs computations. The network's output signal is then translated back into logical form. SHERLOCK allows to compile logic programs either to classical neuro-symbolic networks (the "core method"), or to inductive neural networks (CILP) — the latter can be trained using back-propagation methods.

1 Introduction

We take the ideas of neuro-symbolic integration to the level of software engineering and design. That is, we do not consider theoretical aspects of neuro-symbolic integration here, but take its synthetic principle to be our main software engineering principle. So, which methods could software engineering borrow from the area of neuro-symbolic integration? Here, we offer one possible answer, but see also [Cloete and Zurada, 2000].

Declarative programming languages, and especially logic programming, have one important underlying idea — they are designed to be syntactically similar to the way people reason. Logic programming, for example, is one of the easiest languages to teach students with non-technical background or general public alike. Also, it is feasible to parse natural language into logic programming syntax. Therefore, the strength of logic programming from the software engineering point of view is that it makes for a general and easily accessible interface for users with diverse backgrounds.

Neural networks, on the other hand, offer both massive parallelism and ability to adapt. However, it would seem almost impossible to imagine that a person with non-technical background easily masters neural networks as part of his working routine, alongside with a web-browser or a text editor. It is common that industrial applications of neural networks are designed and maintained by specialists, while non-specialist users do not have ways to edit the applications. This is why neural network applications are often problem-specific. Such applications could be made more general and user-friendly if the users were given a nice easy interface to manipulate neural networks at a level of natural language.

For example, consider a police officer who has just come to a crime scene and wishes to record all evidence available. To be efficient, the police officer uses a small portable computer that has a problem-solving assistant. What should this assistant be like? Neural network software would come in handy, because it can be trained as new evidence is obtained; also – it can be fast due to parallelism. On top of this neural software, though, it is best to have an easy interface allowing the officer to enter data in the form of a natural language.

We propose SHERLOCK — an application that allows the user to type in the knowledge base in the language close to the natural language, and then rely on the compiler that transforms the problem into a suitable neural network. The network will attempt to solve the problem; and once the solution is found — it outputs the answer in a logical form. Thus, SHERLOCK successfully implements the full *neuro-symbolic cycle*, [Hammer and Hitzler, 2007; d'Avila Garcez *et al.*, 2008].

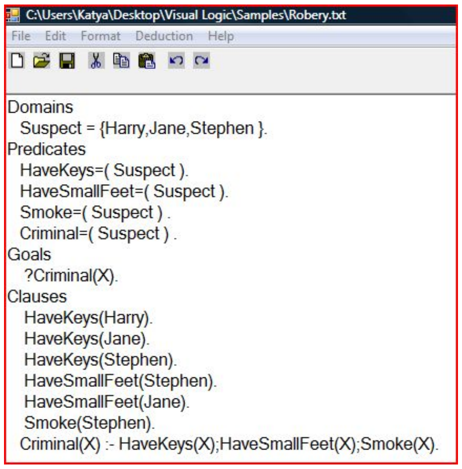
Additionally, as we show in the poster and Section 3, SHERLOCK can be embedded into a bigger knowledge-refining cycle. In this case, we rely upon the backpropagation learning that CILP (cf. [d'Avila Garcez *et al.*, 2002]) offers.

SHERLOCK software relates to the work of [Gruau *et al.*, 1995] proposing a neural compiler for PASCAL; and the programming languages AEL, NETDEF [Siegelmann, 1994] designed to be compiled by neural networks. SHERLOCK differs from the previous similar work in two respects. It is the first fully automated neural compiler for *declarative* languages we know of. Also, in the cited works the main emphasis was on building a fully functional complier for a programming language; here our emphasis is not on creating a neural compiler for PROLOG *per se*; but building a compiler sufficient to handle knowledge bases and reason over them.

2 Design of SHERLOCK

SHERLOCK provides an editor which allows to write and edit information in logical form; it then transforms the information into connectionist neural network; finally, it translates the outcome of the neural-symbolic system back to the logic programming syntax.

[*]The work was supported by EPSRC, UK; Postdoctoral Fellow research grant EP/F044046/2.


C:\Users\Katya\Desktop\Visual Logic\Samples\Robery.txt

File Edit Format Deduction Help

```
Domains
  Suspect = {Harry,Jane,Stephen }.
Predicates
  HaveKeys=( Suspect ).
  HaveSmallFeet=( Suspect ).
  Smoke=( Suspect ) .
  Criminal=( Suspect ) .
Goals
  ?Criminal(X).
Clauses
  HaveKeys(Harry).
  HaveKeys(Jane).
  HaveKeys(Stephen).
  HaveSmallFeet(Stephen).
  HaveSmallFeet(Jane).
  Smoke(Stephen).
 Criminal(X) :- HaveKeys(X);HaveSmallFeet(X);Smoke(X).
```

Figure 1: SHERLOCK's interface.

SHERLOCK consist of the following components:

1. A code editor, in which the users can write a general logic program in a prolog-like declarative language;
2. A translator, which can analyse syntax and semantics of the logic program to set up neural-symbolic systems according to the logic program;
3. A model of the "core method" neural networks [Hammer and Hitzler, 2007], and a model of CILP-neural networks [d'Avila Garcez *et al.*, 2002];
4. An interpreter;
5. An output reader.

The Figure 1 shows SHERLOCK's interface together with a data base written in syntax similar to logic programming. The answer would be all the names that satsify the rule for "Criminal".

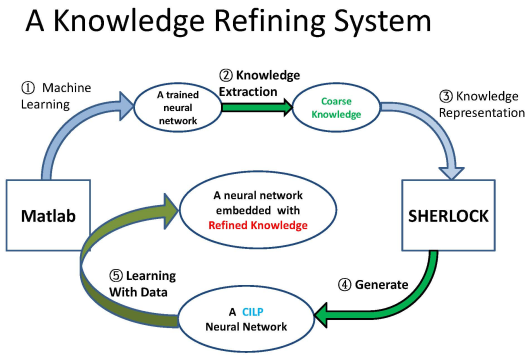
3 Knowledge Refining using SHERLOCK

Knowledge refining is one of the important features in human reasoning. We wish to insert background (or "coarse") knowledge into a neural network and obtain refined knowledge by learning with example data. CILP is suitable to do knowledge refining: it has the capability to present background knowledge into neural networks, and it can use backpropagation to get networks trained with examples.

We propose a novel approach to build knowledge refining systems based on SHERLOCK:

1. Coarse knowledge is obtained from the trained neural network using one of the standard extraction techniques.
2. Then it is expressed in the first order language in SHERLOCK.
3. A CILP neural network is obtained.
4. CILP is trained with the data, and the embedded knowledge is refined.

We test this model on the famous cancer data set from the UCI Machine Learning Repository. The final neural network



has a performance of 96.7%. The performance of the final neural network cannot be improved by setting a better training goal while a general neural network can. This implies the knowledge embedded in the CILP neural network is sensitive to certain kinds of data.

We summarise the properties of this model as follows:

1. It provides a methodology to obtain knowledge in any domain by using both induction and deduction.
2. If the knowledge obtained in Step 1 is reasonable, the final neural network will remain a clear structure, which could be interpreted to symbolic knowledge. Otherwise, the neural network is just an ordinary supervised trained neural network.
3. The final neural network has a very good performance in terms of learning. Besides, it seems that the neural network owns an ability to detect some faulty data due to the knowledge embedded in it.

Sherlock software and its user manual can be downloaded from http://www.computing.dundee.ac.uk/staff/katya/sherlock/

References

[Cloete and Zurada, 2000] I. Cloete and J. M. Zurada. *Knowledge-Based Neurocomputing*. MIT Press, 2000.

[d'Avila Garcez *et al.*, 2002] Arthur d'Avila Garcez, K. B. Broda, and D. M. Gabbay. *Neural-Symbolic Learning Systems: Foundations and Applications*. Springer-Verlag, 2002.

[d'Avila Garcez *et al.*, 2008] Arthur d'Avila Garcez, L. C. Lamb, and D. M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Cognitive Technologies. Springer-Verlag, 2008.

[Gruau *et al.*, 1995] Frédéric Gruau, Jean-Yves Ratajszczak, and Gilles Wiber. A neural compiler. *Theor. Comput. Sci.*, 141(1&2):1–52, 1995.

[Hammer and Hitzler, 2007] B. Hammer and P. Hitzler. *Perspectives of Neural-Symbolic Integration*. Studies in Computational Intelligence. Springer Verlag, 2007.

[Siegelmann, 1994] H. Siegelmann. Neural programming language. *Conf. of AAAI*, 1994.