Learning Symbolic Inferences with Neural Networks

Helmar Gust (hgust@uos.de)
Institute of Cognitive Science, University of Osnabrück
Katharinenstr. 24, 49069 Osnabrück, Germany

Kai-Uwe Kühnberger (kkuehnbe@uos.de)
Institute of Cognitive Science, University of Osnabrück
Katharinenstr. 24, 49069 Osnabrück, Germany

Abstract

In this paper, we will present a theory of representing symbolic inferences of first-order logic with neural networks. The approach transfers first-order logical formulas into a variable-free representation (in a topos) that can be used to generate homogeneous equations functioning as input data for a neural network. An evaluation of the results will be presented and some cognitive implications will be discussed.

Keywords: Neural Networks; First-Order Inferences; Neural-Symbolic Logic.

Introduction

The syntactic structure of formulas of classical first-order logic (FOL) is recursively defined. Therefore it is possible to construct new formulas using given ones by applying a recursion principle. Similarly semantic values of (complex) formulas can be computed by the interpretation of the corresponding parts (Hodges, 1997). Consider, for example, the following formula:

$$\forall x : human(x) \rightarrow mortal(x)$$

The semantics of the complex formula is based on the semantics of the subexpressions *human(x)*, *mortal(x)*, and the implication $\rightarrow$ connecting these subexpressions. Clearly a problem arises because of the presence of the quantifier $\forall$ and the variable x. Nevertheless it is assumed that a compositionality principle allows to compute the meaning of the complex formula using the meaning of the corresponding subformulas. [1]

On the other side, it is assumed that neural networks are non-compositional on a principal basis making it difficult to represent complex data structures like lists, trees, tables, formulas etc. Two aspects can be distinguished: The representation problem (Barnden, 1989) and the inference problem (Shastri & Ajjanagadde, 1990). The first problem states that, if at all, complex data structures can only implicitly be used and the representation of structured objects is a non-trivial challenge for connectionist networks. The second problem tries to model inferences of logical systems with neural accounts. In this paper, our primary aim is the second problem.

A certain endeavor has been invested to solve the representation problem as well as the inference problem. It is well-known that classical logical connectives like conjunction, disjunction, or negation can be represented by neural networks (Rojas, 1996). Furthermore it is known that every Boolean function can be learned by a neural network (Steinbach & Kohut, 2002). Although it is therefore straightforward to represent propositional logic with neural networks, this is not true for FOL. The corresponding problem, usually called the variable-binding problem, is caused by the usage of quantifiers $\forall$ and $\exists$, which are binding variables that occur at different positions in one and the same formula. It is therefore no surprise that there are a number of attempts to solve this problem of neural networks: Examples for such attempts are sign propagation (Lange & Dyer, 1989), dynamic localist representations (Barnden, 1989), tensor product representations (Smolensky, 1990), or holographic reduced representations (Plate, 1994). Unfortunately these accounts have certain non-trivial side-effects. Whereas sign propagation as well as dynamic localist representations lack the ability of learning, the tensor product representation results in an exponentially increasing number of elements to represent variable bindings, only to mention some of the problems.

With respect to the inference problem of connectionist networks the number of proposed solutions is rather small and relatively new. An attempt is Hitzler, Hölldobler & Seda (2004) in which a logical deduction operator is approximated by a neural network and the fixpoint of such an operator provides the semantics of a logical theory. Another approach is Healy & Caudell (2004) where category theoretic methods are assigned to neural constructions. In D'Avila Garcez, Broda & Gabbay (2002), tractable fragments of predicate logic are learned by connectionist networks. Finally in Gust & Kühnberger (2004), a procedure is given how to translate predicate logic into variable-free logic that can be used as input for a neural network. To the knowledge of the authors, the latter account is the only one that does not require hard-wired networks designed for modeling a particular theory. Rather one network topology can be used for arbitrary first-order theories. We will apply the account presented in Gust & Kühnberger (2004) to model first-order inferences of neural networks and to discuss issues relevant for cognitive science.

The paper has the following structure: First, we will sketch the basic ideas of variable-free first-order logic using a representation of FOL induced by category-theoretic means in a topos. Second, we will present the general architecture of the system, the structure of the neural network to code variable-free logic, and the neural modeling of inferences processes.

[1] In classical logic, variables are not only used to express quantification but also to syntactically mark multiple occurrences of terms. Variable management is usually considered as a problematic issue in logic. In particular, the problem arises that algorithms have certain difficulties with quantified variables: The non-decidability of FOL is a direct consequence of this fact.

Fourth, we will discuss and evaluate in-depth an example of how logical inferences can be learned by a neural network. Last but not least, we will relate this to general issues in cognitive science and we will add some concluding remarks.

Logic Without Variables

In order to circumvent problems having to do with the undecidability of FOL a certain endeavor was invested to develop a theory of variable-free logic.[2] A prominent approach to achieve this is the usage of a topos (Goldblatt, 1984). Intuitively a topos is a category that has all limits (for example products), that allows exponents, and that has a subobject classifier. The properties of a topos induce a semantics on the logical constructions. This semantics is equivalent to the standard semantics of FOL (Goldblatt, 1984; Gust, 2000), i.e. the full expressive power of FOL is available in a topos. It is straightforward to translate FOL formulas into objects and arrows in a topos.

We give a prototypical example of a category that satisfies the properties of a topos, namely the category **SET**. The objects of **SET** are sets, connected by set theoretic functions (called arrows). A product $a \times b$ can simply be identified with the Cartesian products of sets a and b, and an exponent a^b with the set of functions $f : b \to a$.

In category theory, constructions like products allow the construction of new arrows. For example, in the case of a Cartesian product $a \times b$ the following condition holds: if arrows $f : c \to a$ and $g : c \to b$ are given, then there exists a unique arrow $h : c \to a \times b$ such that the corresponding diagram commutes. We will use the possibility of constructing new arrows – provided some other arrows are given – in the account presented in this paper. The object ! in **SET** (called the terminal object) is the one-element set $\{0\}$ with the property that for all sets a there is exactly one arrow from a into $\{0\}$. The truth value object $\Omega = \{0, 1\}$ and the subobject classifier *true*: $! \to \Omega$ mapping 0 to 1 generalizes characteristic functions and therefore interpretations of predicates. Logical terms can be interpreted as mappings from the terminal object into the universe U, and logical 2-ary connectives as mappings $\Omega \times \Omega \to \Omega$. Quantified formulas correspond to an operation mapping (complex) predicates to (complex) predicates. The topos account allows the reduction of all logical connectives to one uniform operation, namely concatenation of arrows.

Neural Learning of Formulas

The Architecture of the System

Figure 1 depicts the general architecture of the presented account in four steps: First, input data is given by a set of logical formulas (determining a partial theory) relative to a given logical language $\mathcal{L}$. The language L is considered to be a classical first-order language. Second, this set of formulas is translated into objects and arrows of a topos based on the fact that FOL can be represented by a topos. Third, a PROLOG program is generating equations in normal form $f \circ g = h$ identifying new arrows in the topos. It order to make this

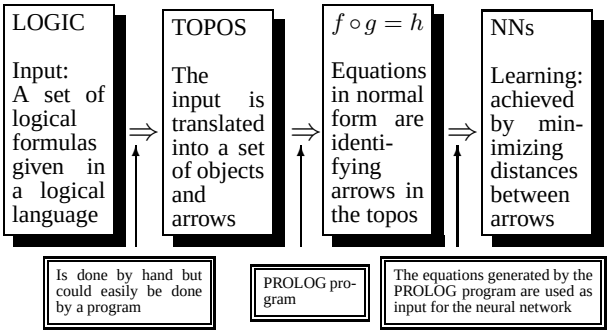

Figure 1: The general architecture of the account.

work, we developed a simple topos language $\mathcal{L}_T$ to code the definitions of objects and arrows in a way such that they can be processed by the program components. The idea is that a given arrow f can be used to generate new equations like $id \circ f = f$, $f \circ id = f$ and so on. Last but not least, these equations are used to train a neural network. The structure of the used neural network will be described below.

The motivation of the proposed solution is based on the idea that we need to transform an interpretation function I of classical logic into a function $I' : \mathbb{R}^m \to \mathbb{R}^n$ in order to make it appropriate as input for a neural network. The first step to achieve this can loosely be associated with reification and elimination of variables, both standard techniques commonly used in AI: Formulas of first-order predicate logic are interpreted as objects and arrows in a topos. The second step is motivated by the challenge to represent logical formulas as equations and finally to represent formulas as equations in a real-valued vector space. In the last step, a necessary issue is to hard-wire certain principles like the one that *true* and *false* is maximally distinct.

Figure 2 depicts the structure of a neural network that is used in order to model the composition process of evaluating terms and formulas. Each object of the topos is represented as a point in an n-dimensional real-valued unit cube. In the example used in this paper[3], we chose $n = 5$. Each arrow in the topos is again represented as a point in the n-dimensional real-valued unit cube together with pointers to the respective domain and codomain. The input of the network is represented by weights from the initial node with activation 1. This allows the backpropagation of the errors into the representation of the inputs of the network. The input of the network represents the two arrows to be composed by the following parts:

- The domain of the first arrow
- The representation of the first arrow
- The codomain of the first arrow which must be equal to the domain of the second arrow
- The representation of the second arrow
- The codomain of the second arrow

[2] In this section, we only sketch the idea of representing FOL in a variable-free logic. A detailed development can be found in Gust & Kühnberger, 2004.

[3] The choice of n depends on the number of objects and arrows which need to be represented. If n is too large, then overfitting of the network can occur. If n is too small, the network may not converge. Currently we do not know the precise correlation between the choice of n and the relative size of the logical theory.

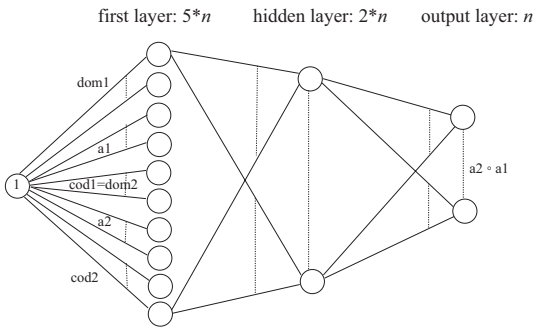

Figure 2: The structure of the neural network that learns composition of first-order formulas.

These requirements lead to a net with $5 \cdot n$ many input values (first layer in Figure 2). The output of the network is the representation of the composed arrow. In the example, we use $2 \cdot n$ many nodes for the hidden layer.

In order to enable the system to learn logical inferences, some basic arrows have static (fixed) representations. These representations correspond directly to truth values.

- The truth value *true* : $(1.0, 1.0, 1.0, 1.0, 1.0)$
- The truth value *false* : $(0.0, 0.0, 0.0, 0.0, 0.0)$

Notice that the truth value *true* and the truth value *false* are maximally distinct. All other objects and arrows are initialized with random values. The defining equations of the theory and the equations generated by categorical constructions (like products) are used to train the neural network.

The following table is an example how certain logical properties of objects and arrows of the topos can be coded in $\mathcal{L}_T$.

Table 1: Example code of the objects and arrows

```
!.                          # the terminal object
@.                          # the truthvalue object
! x ! = !.
u.                          # the universe
static  t: ! --> @,         # true
static  f: ! --> @.         # false
not: @     --> @,           # negation
->:  @ x @ --> @.           # implication
not o t = f,
not o f = t,
-> o t x t = t,
-> o t x f = f,
-> o f x t = t,
-> o f x f = t.
```

In Table 1, elementary logical operators and equations are defined specifying the behavior of classical connectives. The coding of topos entities in $\mathcal{L}_T$ is straightforward: In the first part we define objects and arrows and in the second part we specify the defining equations. Table 2 summarizes the important constructions. Furthermore $\mathcal{L}_T$ provides a macro mechanism to allow a compact coding for complex equations (cf. the definition of ==> in Table 3). All derived objects and arrows, e.g. identities and products, are recognized by the PROLOG program and the corresponding defining equations are automatically generated.

Example

In Table 3, an extension of the premises of the classical Socrates syllogism is modeled. The represented information is not only that all humans are mortal but also that all mortals ascend to heaven and everyone in heaven is an angel. As

Table 2: The specification of $\mathcal{L}_T$

$\mathcal{L}_T$	Intended Interpretation
`!`	Terminal object !
`@`	Truth value objects Ω
`u`	The universe U
`t`	Truth value $true$
`f`	Truth value $false$
`Y x Z`	Product object of Y and Z
`y x z`	Product arrow of y and z
`! X`	Terminal arrow of X
`x: Y --> Z`	Definition of an arrow
`y o z`	Composition of arrows x and y

constants not only *Socrates* but also *robot* and *something* are introduced with the additional information that *robot* is not human. There is no knowledge about *something* available. These types of information are represented by equations. For example, the composition of the arrow *human* and *socrates* is resulting in *true* representing that "*Socrates* is human". Slightly more difficult is the representation of universally quantified expressions like $\forall x : human(x) \rightarrow mortal(x)$. The equation is constructed as follows:[4]

$$\forall(\Rightarrow \circ human \times mortal \circ d) = true$$

This is equivalent to

$$\Rightarrow \circ human \times mortal \circ d = true \circ !$$

The diagonal arrow $d : U \rightarrow U \times U$ is composed with the arrows for predicates $human : \Omega \rightarrow \Omega$ and $mortal : \Omega \rightarrow \Omega$ and the arrow for the implication $\Rightarrow: \Omega \times \Omega \rightarrow \Omega$. Notice further that the construction is only possible because a topos guarantees the existence of the relevant arrows. Finally, test equations are represented in Table 3. They correspond to the logically possible combinations of $mortal$, $human$, and $angel$ on the one hand and $Socrates$, $robot$, and $something$ on the other. All combinations can be either *true* or *false*.

The Results

The input generated by the Prolog program is feeded into the neural network. The result of an example run is then given by the errors of the test equations. These test equations query whether the composition of *angel* and *robot* is *false*, whether the composition of *angel* and *robot* is *true*, whether the composition of *angel* and *socrates* is false etc. The results of test run of our example is depicted below:

```
Tests:
angel o robot       = f         0.636045
angel o robot       = t         0.886353
angel o socrates    = f         2.197811
angel o socrates    = t         0.011343
angel o something   = f         0.053576
angel o something   = t         2.017303
heaven o robot      = f         0.454501
heaven o robot      = t         1.080864
heaven o socrates   = f         2.153396
heaven o socrates   = t         0.013502
heaven o something  = f         0.034890
heaven o something  = t         2.111497
mortal o socrates   = f         1.985289
mortal o socrates   = t         0.030935
mortal o robot      = f         0.195687
```

[4]Notice that the following expressions are expressions in the topos and not logical expressions.

Table 3: Example code of an extension of the famous "Socrates inference"

```
# predicates of the theory
  human, mortal, heaven, angel:  u ---> @ .
  X ==> Y: -> o X x Y o d u = t o ! u .
  human  ==> mortal.
  mortal ==> heaven.
  heaven ==> angel.
#individuals
distinctive
  socrates, robot, something: ! ---> u.
  human o socrates = t.
  human o robot    = f.
# test the learned inferences
tests
  mortal o something = t,
  mortal o something = f,
  mortal o robot     = t,
  mortal o robot     = f,
  mortal o socrates  = t,
  mortal o socrates  = f,
  heaven o something = t,
  heaven o something = f,
  heaven o socrates  = t,
  heaven o socrates  = f,
  heaven o robot     = t,
  heaven o robot     = f,
  angel  o something = t,
  angel  o something = f,
  angel  o socrates  = t,
  angel  o socrates  = f,
  angel  o robot     = t,
  angel  o robot     = f.


mortal o robot     = t          1.488895
mortal o something = f          0.025812
mortal o something = t          2.159551
```

The system convincingly learned that *Socrates* is mortal, ascended to *heaven*, and is an *angel*. Furthermore it learned that the negations of these consequences are false. In other words, the system learned the transitivity of the implication in universally quantified formulas. With respect to *robot* the system evaluates with a high certainty that *robot* is not mortal. The other two properties are undecided. In the case of *something* relatively certain knowledge for the system is that *something* is neither in heaven nor mortal nor an angel.

We will have a closer look on how the neural network interprets queries. In the left diagram of Figure 3, the maximal error of the neural network of 10 runs with $1.6 \cdot 10^6$ many iterations is depicted. The curves show four characteristic phases: in the first phase (up to 50,000 iterations), the randomly chosen representations for the input arrows and objects remains relatively stable. During the second phase (between 50,000 and approx. 200,000 iterations) the maximal error dramatically increases due to the rearrangement of the input representations. In the third phase (between 200,000 and 600,000 iterations) the maximal error rapidly decreases which is again connected with the reorganization of the input representation. In most cases, the maximal error decreases in the fourth phase (between 600,000 and above iterations), but the input representations stay relatively stable.

The right diagram of Figure 3 shows the stability behavior of the neural network (again 10 runs with $1.6 \cdot 10^6$ many iterations). In a first phase (up to 100,000 iterations), the instability of the weights increases dramatically. In a band between approx. 100,000 and approx. 250,000 iterations the stability of the network increases and the network remains (relatively) stable in the third phase between approx. 400,000 iterations and above. Interesting are certain fluctuations (of certain runs) of the stability behavior between, for example, around 800,000 iterations or for other runs between $1,2 \cdot 10^6$ and $1,6 \cdot 10^6$ many iterations.

The two diagrams in Figure 4 show the behavior of *Socrates is an angel* (left diagram) and *The robot is an angel* (right diagram). The classification is as expected. Whereas *Socrates* is classified as an angel using the transitivity of the implication, the *robot* is classified as a non-angel. Clearly the input only specifies that the *robot* is not human. It does not follow logically that the *robot* cannot be an angel. The result of the weight distribution of the neural network with respect to the robot can be interpreted as a support of something similar to a closed-world assumption. It is interesting that the interpretation of *something* (cf. Figure 5) differs from the interpretation of *robot*, because with respect to *something* there is no clear tendency how to classify this object.

The models approximated by the network behave as expected: Test equations which are logically derivable *true* or *false* will be mapped in all models to *t* or *f* respectively. Those equations for which no logical deduction of the truth value is possible, are more or less arbitrarily distributed between f and t in the set of models. Nevertheless, the models seem to tend to realize a closed world interpretation, i.e. the truth value of $mortal(robot)$, $heaven(robot)$, and $angel(robot)$ tend to be *false*.

Consequences for Cognitive Science

The translation of first-order formulas into training data of a neural network allows, in principal, to represent models of symbolic theories in artificial intelligence and cognitive science (that are based on FOL) with neural networks.[5] In other words the account provides a recipe – and not just a general statement of the possibility – of how to learn models of theories based on FOL with neural networks. Notice that the presented approach tries to combine the advantages of connectionist networks and logical systems: Instead of representing symbols like constants or predicates using single neurons, the representation is rather distributed realizing the very idea of distributed computation in neural networks. Furthermore the neural network can be trained quite efficiently to learn a model without any hardcoded devices. The result is a distributed representation of a symbolic system.

From a dual perspective, it would be desirable to find a possibility to translate the distribution of weights in a trained neural network back to the symbolic level. The symbol grounding problem could then be analyzed in detail by translating the representation levels into each other. Although we cannot give a detailed account for such a translation from the neural level to the symbolic one in this paper, we think that certain invariants on the neural correlate could be used for such a translation to the symbolic level. The development of such a theory will be dedicated to another paper.

A logical theory consists of axioms specifying facts and rules about a certain domain together with a calculus determining the "correct" inferences that can be drawn from these axioms. From a computational point of view this generates quite often problems, because inferences can be rather resource consuming. Modeling logical inferences with neural networks as presented in this paper allows a very efficient way of drawing inferences, simply because the interpretation of possible expressions is "just there", namely implicitly coded by the distribution of the weights of the network. Notice that the neural network does not only learn the input, but a model

[5] Notice that a large part of theories in artificial intelligence are formulated with tools taken from logic and are mostly based on FOL or subsystems of FOL.

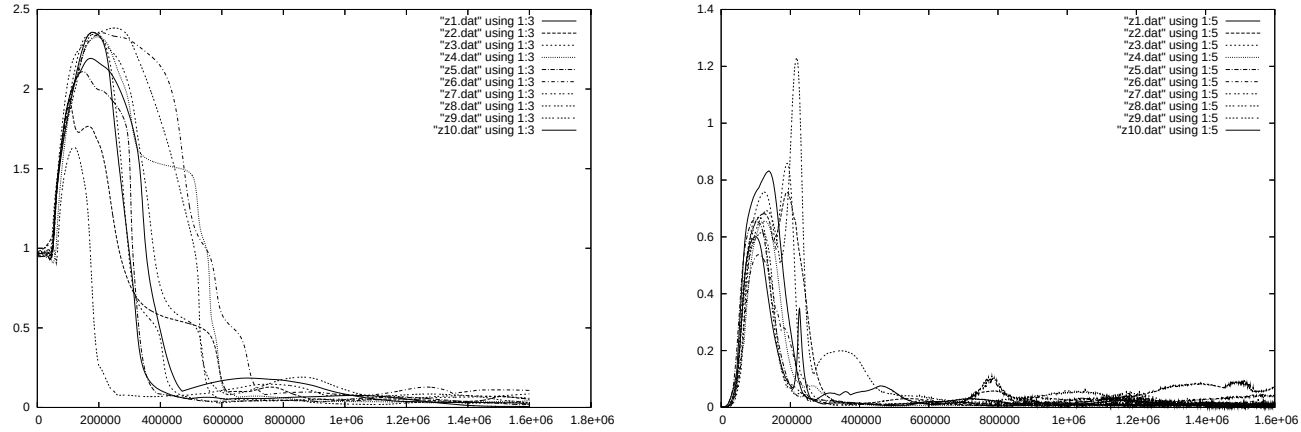

Figure 3: Stability and the maximal error of the neural network with respect to the extended Socrates example (10 runs with $1.6 \cdot 10^6$ iterations). The left diagram depicts the maximal error of the neural network. The right diagram shows the stability behavior of the network with respect to the 10 runs.

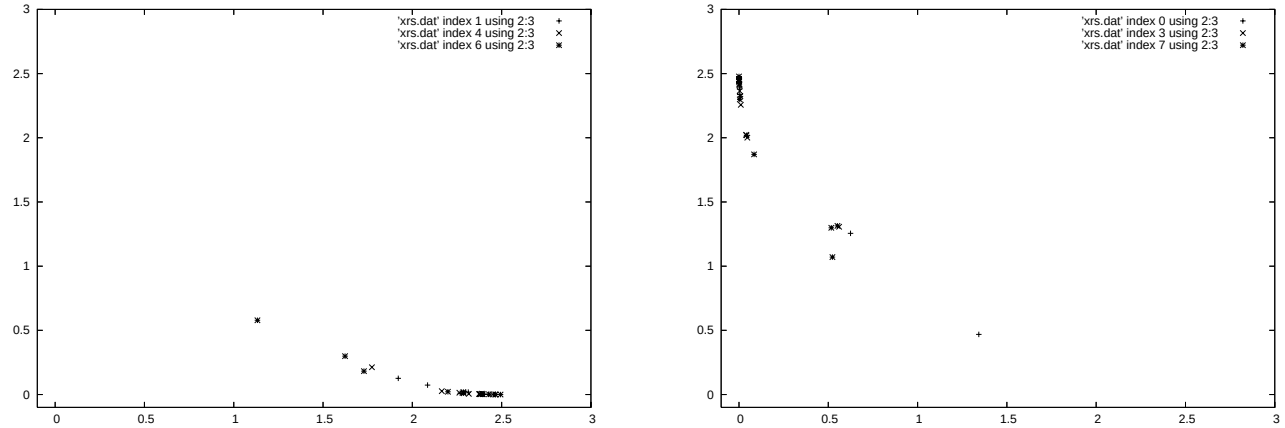

Figure 4: The distribution of the interpretation of *Socrates is an angel* (left diagram) and *The robot is an angel* (right diagram) with respect to the extended Socrates example (10 runs with $1.6 \cdot 10^6$ iterations). The left diagram shows that Socrates is quite stably classified as an angel, whereas the robot is quite stably classified as a non-angel.

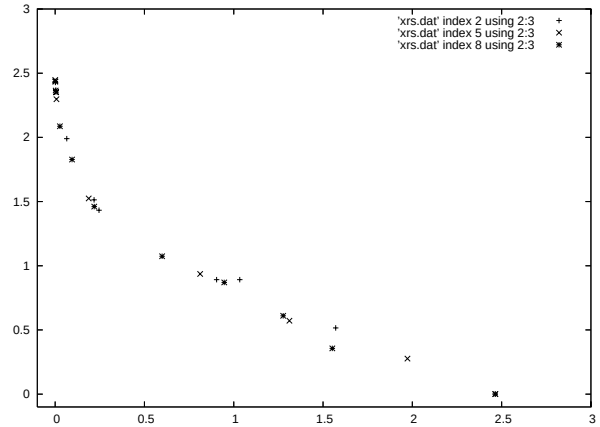

Figure 5: The distribution of the classification of *Something is an angel* with respect to the extended Socrates example (10 runs with $1.6 \cdot 10^6$ iterations). The result shows that the distribution is quite equally distributed.

making the input true. In a certain sense these models are overdetermined, i.e. they assign truth values even in those cases which are not determined by the theory. Nevertheless they are consistent with the theory. There is evidence from the famous Wason selection-task that human behavior is (in our terminology) rather model-based than theory-based, i.e. human behavior can be deductive without having an inference mechanism (cf. Gardner, 1989; Johnson-Laird, 1983). We can give an explanation of this phenomenon: Humans act mostly according to a model they have (about, for example, a situation) and not according to a theory plus an inference mechanism. The tendency of our models towards a closed-world assumption provide hints for an explanation of the empirical observations, because – as can be seen in the *robot* case – the property of the robot to be non-human propagates to the property to be a non-angel. This provides evidence for an equivalence between *The robot is human* and *The robot is an angel* in certain types of underdetermined situations.

Taking into account time limitations, for example, in real-world applications, the usage of a trained neural network in a complex system would significantly facilitate the application, because there are no inference steps that need to be computed. Furthermore the module can deal with noisy and uncertain input data which are standardly considered as a problem for applications in a rapidly changing environment. Clearly uncertain data cannot be assigned a definite truth value, but in all cases the network will generate a certain value that can be used by the system. A possibility to make the correspondence between the neural correlate on the one side and symbolic approaches under uncertainty on the other more obvious is the introduction of a truth value n (neither true nor false) and a truth value b (both true and false). Such an introduction is straightforward on the neural network side and corresponds nicely to well-known accounts of many-valued logic in symbolic systems (Urquhart, 2001).

Concluding Remarks

In this paper, we presented a framework for modeling FOL, in particular, its inference mechanisms with neural networks. We sketched a theory of uniformly translating axiomatic systems into a certain type of variable-free logic, which can be used for generating equations in normal form. These equations are further transformed into an input for a neural net that learns the axioms together with logical consequences. A detailed evaluation of the results were given and a discussion of consequences of this framework for cognitive science was presented. Besides the well-known advantages of stability and robustness concerning noisy data, the present account allows to learn non-trivial inferences without deduction steps.

Future work will be concerned with the modeling of a three- and four-valued logic on the network. Because the net topology is independent of the type of the underlying logic, this step is straightforward. The resulting models need to be carefully evaluated. Moreover the integration of trained neural nets into real-world robotic devices will be developed. In the case of robotics, it would be necessary to extend the framework to allow online learning. But this fits seamlessly into our approach since adding new axioms again does not change the network topology, contrary to most alternative accounts.

References

Barnden, J.A. (1989). Neural net implementation of complex symbol processing in a mental model approach to syllogistic reasoning. In Proceedings of the International Joint Conference on Artificial Intelligence, 568-573.

D'Avila Garcez, A., Broda, K. & Gabbay, D. (2002). *Neural-Symbolic Learning Systems: Foundations and Applications*. Springer-Verlag.

Gardner, H. (1989). *Dem Denken auf der Spur*. Kohlhammer.

Goldblatt, R. (1984). *Topoi, the categorial analysis of logic. Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam.

Gust, H. (2000). Quantificational Operators and their Interpretation as Higher Order Operators. M. Böttner & W: Thümmel, *Variable-free Semantics*, 132-161, Osnabrück.

Gust, H. & Kühnberger, K.-U. (2004). Cloning Composition and Logical Inferences in Neural Networks Using Variable-Free Logic, *Papers from the 2004 AAAI Fall Symposium*, Technical Report FS-04-03, pp. 25-30.

Healy, M. & Caudell, T. (2004). *Neural Networks, Knowledge and Cognition: A Mathematical Semantic Model Based upon Category Theory*, University of New Mexico, EECE-TR-04-020.

Hitzler, P., Hölldobler, S. & Seda, A. (2004). Logic programs and connectionist networks. *Journal of Applied Logic*, 2(3):245-272.

Hodges, W. (1997). *A Shorter Model Theory*. Cambridge University Press.

Johnson-Laird, P. (1983). *Mental Modes: Towards a Cognitive Science of Language, Inference, and Consciousness*, Cambridge, Mass.

Lange, T. & Dyer, M.G. (1989). *High-level inferencing in a connectionist network*. Technical report UCLA-AI-89-12.

Plate, T. (1994). *Distributed Representations and Nested Compositional Structure*. PhD thesis, University of Toronto.

Rojas, R. (1996). *Neural Networks – A Systematic Introduction*. Springer, Berlin, New York.

Shastri, L. & Ajjanagadde, V. (1990). From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences* 16: 417-494.

Smolenski, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence* 46(1–2): 159–216.

Steinbach, B. & Kohut, R. (2002). Neural Networks – A Model of Boolean Functions. In Steinbach, B. (ed.): *Boolean Problems, Proceedings of the 5th International Workshop on Boolean Problems*, 223-240, Freiberg.

Urquhart, A. (2001). Basic many-valued logic. In D. Gabbay & F. Guenthner (Eds.), *Handbook of Philosophical Logic*, 2nd ed., vol. 2, Kluwer Acad. Publ., Dordrecht, pp. 249-295.