Towards a Connectionist Argumentation Framework

A. S. d'Avila Garcez [1] and **D. M. Gabbay** [2] and **L. C. Lamb** [3]

Abstract. While neural networks have been successfully used in a number of machine learning applications, logical languages have been the standard for the representation of legal and argumentative reasoning [6]. In this paper, we present a new hybrid model of computation that allows for the deduction and learning of argumentative reasoning. We do so by using Neural-Symbolic Learning Systems to translate argumentation networks into standard neural networks. The approach enables cummulative argumentation through learning, as the strength of the arguments change over time.

1 Introduction

Neural-Symbolic integration concerns the application of symbolic knowledge within the connectionist paradigm [2]. Neural-Symbolic Learning Systems use single hidden layer neural networks to represent and learn nonmonotonic, epistemic or temporal symbolic knowledge with the use of off-the-shelf neural learning algorithms [4, 5]. Our goal is to facilitate learning capabilities in value-based argumentation frameworks, as arguments may evolve over time, with certain arguments being strengthened and others weakened. In this paper, we use a moral debate example to show how the object language of Bench-Capon's argumentation networks can be translated into standard neural networks. Details about the translation, such as the translation algorithm and its proof of correctness, can be found in [3]. In Section 2, we briefly present the basic concepts of neural-symbolic systems. In Section 3, we discuss the moral debate example, and in Section 4, we conclude and discuss directions for future work.

2 Neural-Symbolic Learning Systems

The *Connectionist Inductive Learning and Logic Programming (*C-ILP*) System* [2] is a neural network model that integrates inductive learning from examples and background knowledge with deductive learning from logic programming. C-ILP's *Translation Algorithm* maps a logic program $\mathcal{P}$ into a single hidden layer neural network $\mathcal{N}$ such that $\mathcal{N}$ computes the least fixed-point of $\mathcal{P}$. This provides a massively parallel model for logic programming. In addition, $\mathcal{N}$ can be trained with examples using Backpropagation, having $\mathcal{P}$ as background knowledge. The knowledge acquired by training can then be extracted, closing the learning cycle [2]. Let us exemplify how C-ILP*'s Translation Algorithm* works. Each rule (r_l) of $\mathcal{P}$ is mapped from the input layer to the output layer of $\mathcal{N}$ through one neuron (N_l) in the single hidden layer of $\mathcal{N}$. Intuitively, the *Translation Algorithm* has to implement the following conditions: (c_1) The input potential of a hidden neuron (N_l) can only exceed N_l's threshold (θ_l), activating N_l, when all the positive antecedents of r_l are assigned $true$ while all the negative antecedents of r_l are assigned $false$; and (c_2) The input potential of an output neuron (A) can only exceed A's threshold (θ_A), activating A, when at least one hidden neuron N_l that is connected to A is activated.

In order to use $\mathcal{N}$ as a massively parallel model for logic programming, we just have to follow two steps: (i) add neurons to the input and output layers of $\mathcal{N}$, allowing it to be recurrently connected; and (ii) add the corresponding recurrent connections with fixed weight $W_r = 1$, so that the activation of output neuron A feeds back into the activation of input neuron A, the activation of output neuron B feeds back into the activation of input neuron B, and so on. In the case of argumentation networks it will be sufficient to consider definite logic programs (i.e. programs without negation). In this case, the neural network will contain only positive weights (W). We will then expand such a positive network to represent attacks using negative weights from the network's hidden layer to its output layer.

3 Argumentation Neural Networks

In the value-based argumentation framework defined in [1], argumentation networks are used to model arguments and counter-arguments. A typical example in the area is the following *moral debate* example. *Hal, a diabetic, loses his insulin in an accident through no fault of his own. Before collapsing into a coma, he rushes to the house of Carla, another diabetic. She is not at home, but Hal breaks into her house and uses some of her insulin. Was Hal justified? Does Carla have a right to compensation?* The following are some of the arguments involved in the example. **A**: Hal is justified, he was trying to save his life; **B**: It is wrong to infringe the property rights of another; **C**: Hal compensates Carla; **D**: Hal is endangering Carla's life; **E**: Carla has abundant insulin; and **F**: If Hal is too poor to compensate Carla he should be allowed to take the insulin as no one should die because they are poor.

In [1], arguments and counter-arguments are arranged in an argumentation network, as in Figure 1(a), where an arrow from argument X to argument Y indicates that X attacks Y. For example, the fact that it is wrong to infringe Carla's right of property (**B**) attacks Hal's justification (**A**). In the argumentation network of Figure 1(a), some aspects may change as the debate progresses and actions are taken, with the strength of an argument in attacking another changing in time. This is a learning process that can be implemented using a neural network in which the weights encode the strength of the arguments as in Figure 1(b). The network is an auto-associative single hidden layer network with inputs A to F, outputs A

[1] Department of Computing, City University London, UK
[2] Department of Computer Science, King's College London, UK
[3] Instituto de Informatica, UFRGS, Porto Alegre, Brazil

to F, and hidden layer h_1 to h_6. Solid black arrows represent positive weights, and dotted gray arrows represent negative weights. Arguments are supported by positive weights and attacked by negative ones. Argument **A** (input neuron A), for example, supports itself (output neuron A) with the use of hidden neuron h_1. Similarly, argument **B** supports itself (via h_2), and so does argument **D** (via h_3). From the argumentation network, **B** attacks **A**, and **D** attacks **A**. The attacks are implemented in the neural network by the negative weights with the use of h_2 and h_4. The network of Figure 1(b) is a standard feedforward neural network that can be trained by a standard neural learning algorithm. Training would change the initial weights of the network (the initial belief on the strength of arguments and counter-arguments), according to examples of input/output patterns, i.e. examples of the relationship between arguments. If the absolute value of the weight from neuron h_1 to output neuron A is greater than the sum of the absolute values of the weights from neurons h_2 and h_4 to A, one could say that argument **A** prevails (in which case output neuron A should be *activated* in the neural network). This behaviour can be implemented by C-ILP networks [2, 3] with the help of the following transformation: given an argumentation network $\mathcal{A}$ with arguments $\alpha_1, \alpha_2, ..., \alpha_n$, make $\mathcal{P} = \{r_1 : \alpha_1 \rightarrow \alpha_1, r_2 : \alpha_2 \rightarrow \alpha_2, ..., r_3 : \alpha_n \rightarrow \alpha_n\}$, and apply C-ILP's *Translation Algorithm* to convert $\mathcal{P}$ into $\mathcal{N}$. The network can then be run, as exemplified in the sequel, to compute the prevailing arguments.[4]

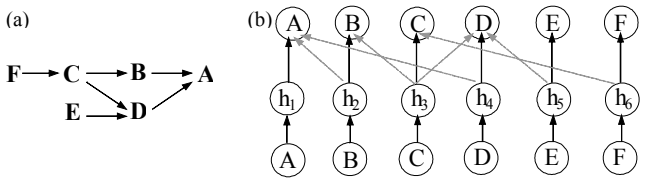

Figure 1: argumentation network (a); neural network (b).

Example 1 *(*moral debate*) From C-ILP* Translation Algorithm*, we know that* $A_{min} > 0$ *and* $W > 0$*. Let us take* $A_{min} = 0.5$ *and* $W = 5$ *(recall that W is the weight of solid black arrows in the network). Following [6], we reason about the problem by grouping arguments according to the features of* life, property *and* fact*. Arguments* ***A****,* ***D*** *and* ***F*** *are related to the right of life, arguments* ***B*** *and* ***C*** *are related to property rights, and argument* ***E*** *is a fact. We may argue whether* property *is stronger than* life *but facts are always the strongest. We use* $v(\alpha_i, \alpha j) = 1$ *to denote that* α_i *is stronger than* α_j*. If property is stronger than life then we have* $v(B, A) = 1$*,* $v(D, A) = 1$*,* $v(C, B) = 1$*,* $v(C, D) = 1$*,* $v(E, D) = 1$*, and* $v(F, C) = 0$*. From the* Translation Algorithm *[3], when* $v(\alpha_i, \alpha_j) = 0$ *we must have* $W' > -1.4$*, and when* $v(\alpha_i, \alpha_j) = 1$ *we must have* $W' < -12.2$*. The actual value of each attack may depend on an audience, but provided the above conditions on* W' *are satisfied, the network will compute the expected prevailing arguments, as follows: F does not defeat C, C defeats B, E defeats D and, as a result, we obtain {A,C,E} as the acceptable set of arguments. Nonetheless, if life is considered stronger than property then* $v(F, C) = 1$*.*[5] *As a result, F defeats C and, since C is defeated, it cannot defeat B, which in turn cannot defeat A (because life is stronger than property). Thus, we obtain the set {A,B,E,F} of acceptable arguments. This shows that two different lines of argumentation will provide the same answer to the question of whether Hall was justified, but two different answers to the question of whether Carla has the right to compensation.*

An interesting aspect of argumentation neural networks lies in the fact that they do not loop in the presence of circular arguments with identical strengths. For example, if argument A attacks argument B and vice-versa, both with the same strength, the network converges to a stable state (-1,-1) in which, as expected, neither of A or B prevails. For more details, see [3].

4 Conclusion and Future Work

In this paper, we have introduced a new hybrid model of computation that allows for the deduction and learning of argumentative reasoning. The model combines value-based argumentation frameworks and neural-symbolic learning systems by providing a translation from argumentation networks to C-ILP neural networks. The model works not only for acyclic argumentation networks but also for circular networks and enables cummulative argumentation through learning. Experiments on learning argumentation neural networks capable of evolving over time are currently being conducted. Complexity issues regarding the parallel computation of argumentation neural networks in contrast with standard value-based argumentation frameworks are also being investigated. We believe that a neural implementation of this reasoning process may, in fact, be advantageous from a purely computational point of view due to neural networks' massive parallel nature.

REFERENCES

[1] T. J. M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation*, 13:429–448, 2003.

[2] A. S. d'Avila Garcez, K. Broda, and D. M. Gabbay. *Neural-Symbolic Learning Systems*. Springer, 2002.

[3] A. S. d'Avila Garcez, D. M. Gabbay, and L. C. Lamb. Argumentation neural networks: Value-based argumentation frameworks as neural-symbolic learning systems. Technical report, Department of Computing, City University London, May 2004.

[4] A. S. d'Avila Garcez and L. C. Lamb. Reasoning about time and knowledge in neural-symbolic learning systems. In Proc. NIPS'03, Vancouver, MIT Press, 2004.

[5] A. S. d'Avila Garcez, L. C. Lamb, K. Broda, and D. M. Gabbay. Applying connectionist modal logics to distributed knowledge representation problems. *International Journal of Artificial Intelligence Tools*, 2004.

[6] D. M. Gabbay and J. Woods. The law of evidence and labelled deduction: A position paper. *Phi News*, 4, October 2003.

[4] Note that the program $\mathcal{P}$ obtained from an argumentation network will always have the form of a set of rules $r_i : \alpha_i \rightarrow \alpha_i$. As a result, differently from in the C-ILP *Translation Algorithm* [2], in which rules having more than one antecedent are accounted for, here there is always a single antecedent per rule (α_i). This allows us to use $A_{\min} > 0$ and $\theta = 0$ when using the algorithm to calculate positive weights W and negative weights W'. In addition, the fact that $W > 0$ and $W' < 0$ fits very well with the idea of arguments having strengths (W), and attacks also having strengths (W'). The values of W and W' could, for example, be defined by an audience using a voting system [1]. In this system, at some point, an accumulation of attacks with different strengths - neither being individually stronger than the argument being attacked - might produce a value $\sum_i W'_i$ that overcomes W. This is naturally the way that neural networks work.

[5] The complete set is $v(B, A) = 0$, $v(D, A) = 1$, $v(C, B) = 1$, $v(C, D) = 0$, $v(E, D) = 1$, and $v(F, C) = 1$.