DeepLogic
Towards End-to-End Differentiable Logical Reasoning

Imperial College London

Nuri Cingillioglu & Alessandra Russo
Imperial College London
`nuric@imperial.ac.uk`

Abstract

Combining machine learning with logic-based expert systems in order to get the best of both worlds are becoming increasingly popular. However, to what extent machine learning can already learn to reason over rule-based knowledge is still an open problem. We explore how symbolic logic, defined as logic programs at a character level, is learned to be represented in a high-dimensional vector space using iterative recurrent neural networks to perform reasoning.

Introduction

Deep learning [2] has emerged as a tool for learning human level tasks such as recognising objects in images, transcribing speech to text and translating between different natural languages. Logic [3] is the field of study that attempts to formalise the reasoning process and valid inference. Computational logical reasoning can take the form of first-order logic and express inferences such as "human(X)→mortal(X) ∧ human(socrates) ⊢ mortal(socrates)" to derive conclusions. Can we thus make machines learn such forms of reasoning using neural networks? DeepLogic [1] takes the first steps towards:

1. If and how neural networks learn to represent symbolic constructs from logic?
2. If and how iterative neural networks use those representations to perform reasoning over logic programs?

Dataset

To train the neural networks we generate 12 classes of normal logic program tasks that capture various reasoning processes such as logical and, logical or and negation by failure. In total, there are 20k programs per task. The programs do not contain function symbols, recursion and have symbol lengths up to 2 during training. A ground query atom Q together with the logic program C are used to provide a true or false target whether the context entails the query or not $C \vdash Q$.

$$f(C,Q) = \begin{cases} 1 & \text{if } C \vdash Q \\ 0 & \text{otherwise} \end{cases}$$

1: Facts	5: 3 Steps	8: Transitivity	10: 2 Step NBF	12: OR NBF
e(l). i(u). n(d). v(h,y). p(n).	p(P,R) :- b(R,P). b(A,L) :- a(A,L). a(W,F) :- v(F,W). v(t,i). c(V,V) :- d(V,V). l(D) :- t(D).	f(A,W) :- q(A,P) , d(P,W). q(h,t). d(t,j). q(d,m). d(n,g). s(S,F) :- x(S,A) , e(A,F).	r(C) :- -o(C). o(P) :- l(P). l(o). g(u). p(U,L) :- e(U,L). p(X,X).	y(Z) :- -e(Z). y(Z) :- b(Z). y(r). e(d). s(a). b(m).
? e(l). 1 ? i(d). 0	? p(t,i). 1 ? p(i,t). 0	? f(h,j). 1 ? f(d,g). 0	? r(u). 1 ? r(o). 0	? y(a). 1 ? y(d). 0

Table 1: Sample logic programs from the dataset. At test time, we generate 4 test sets of increasing difficulty: validation, easy, medium and hard which have up to 2, 4, 8 and 12 characters for predicates and constants as well as added number of irrelevant rules respectively.

Approach

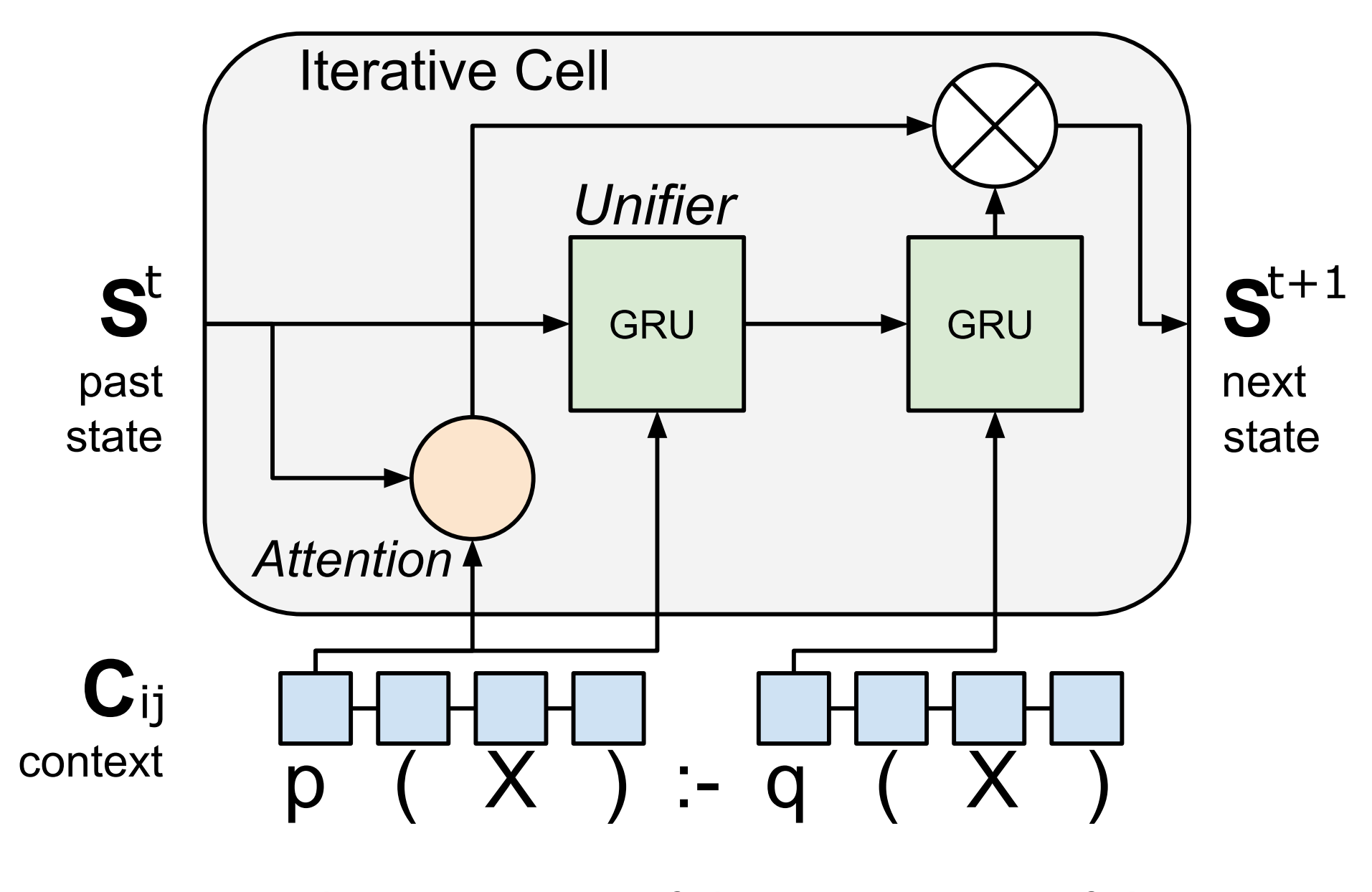

Figure 1: Graphical overview of the iterative cell of the Iterative Memory Attention (IMA) model. The context and query are processed at the character level to produce literal embeddings, then an attention is computed over the head of the rules. A weighted sum of the unifier GRU outputs using the attention, updates the state for the next iteration.

We can consider the logic program context a read-only memory and the proof state a writable memory component. In a similar fashion to backward chaining algorithm [3], we aim to have (i) a state to store information about the proof such as the query, (ii) a mechanism to select rules via attention and (iii) a component to update the state with respect to the rules. To that end, we introduce the Iterative Memory Attention (IMA) network that given a normal logic program as context and a positive ground atom as query, embeds the literals in a high dimensional vector space, attends to rules using soft attention and updates the state using a recurrent network.

Results

Model	LSTM	MAC	DMN	IMA		
Embedding	-	rule	rule	literal		lit+rule
Attention	-	sm	σ	σ	sm	sm
Easy Mean	0.57	0.81	0.79	**0.91**	0.90	0.88
Medium Mean	0.52	0.70	0.70	**0.86**	0.81	0.79
Hard Mean	0.51	0.63	0.66	**0.83**	0.75	0.72

Table 2: Mean accuracy of the models when trained all the tasks at the same time; sm stands for softmax activation.

We take each test set that consists of 10k generated logic programs per task and show the mean accuracy for the best single training run out of 3 for each model with state size $d = 64$. All models seem to degrade in performance as the difficulty of test sets increase; we speculate this stems from the fixed size state vector that needs to store more information as symbols lengths get longer at test time.

Analysis

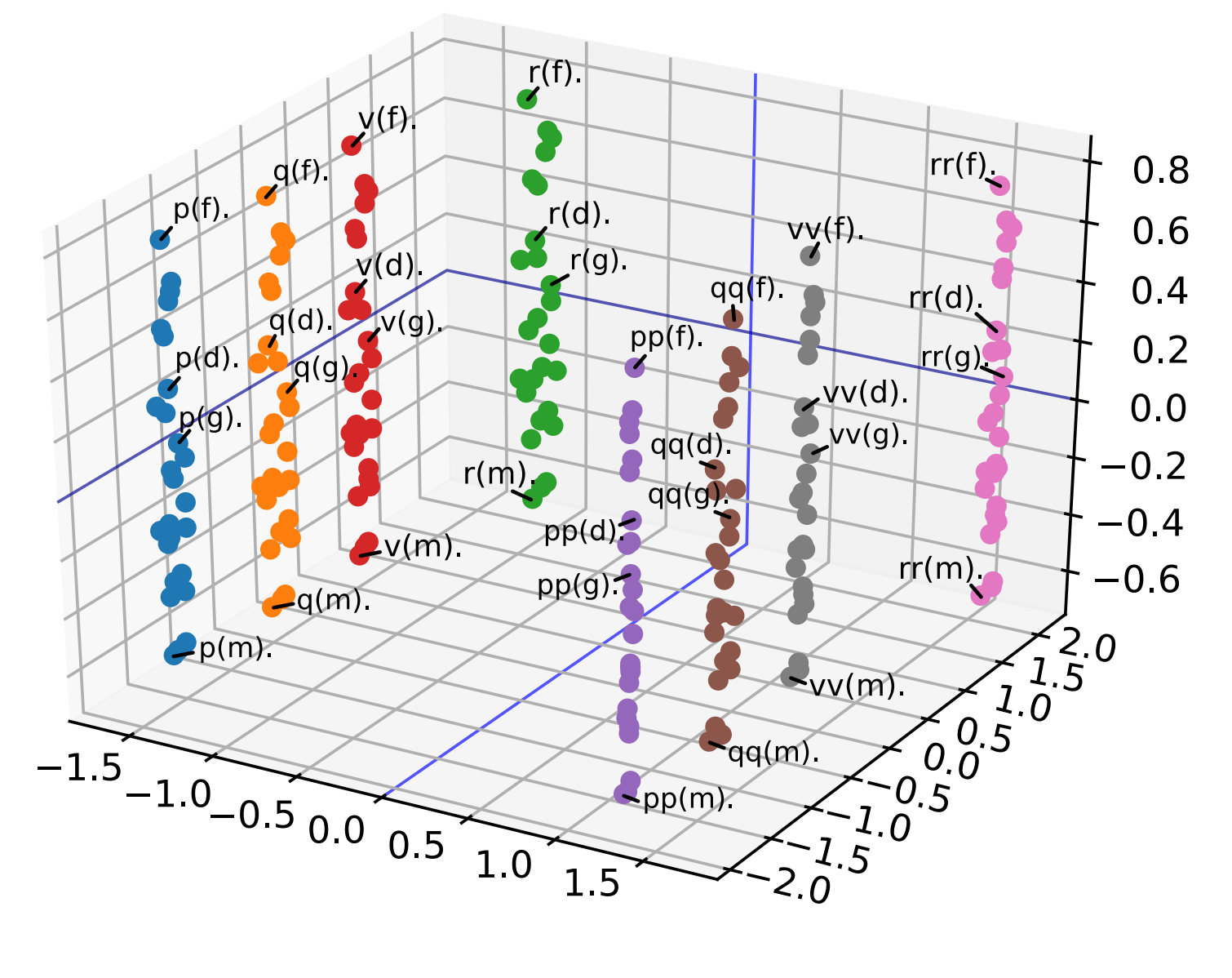

Figure 2: Structurally different literals first cluster by whether they are negated or grounded then by arity (grey lines added as visual aids).

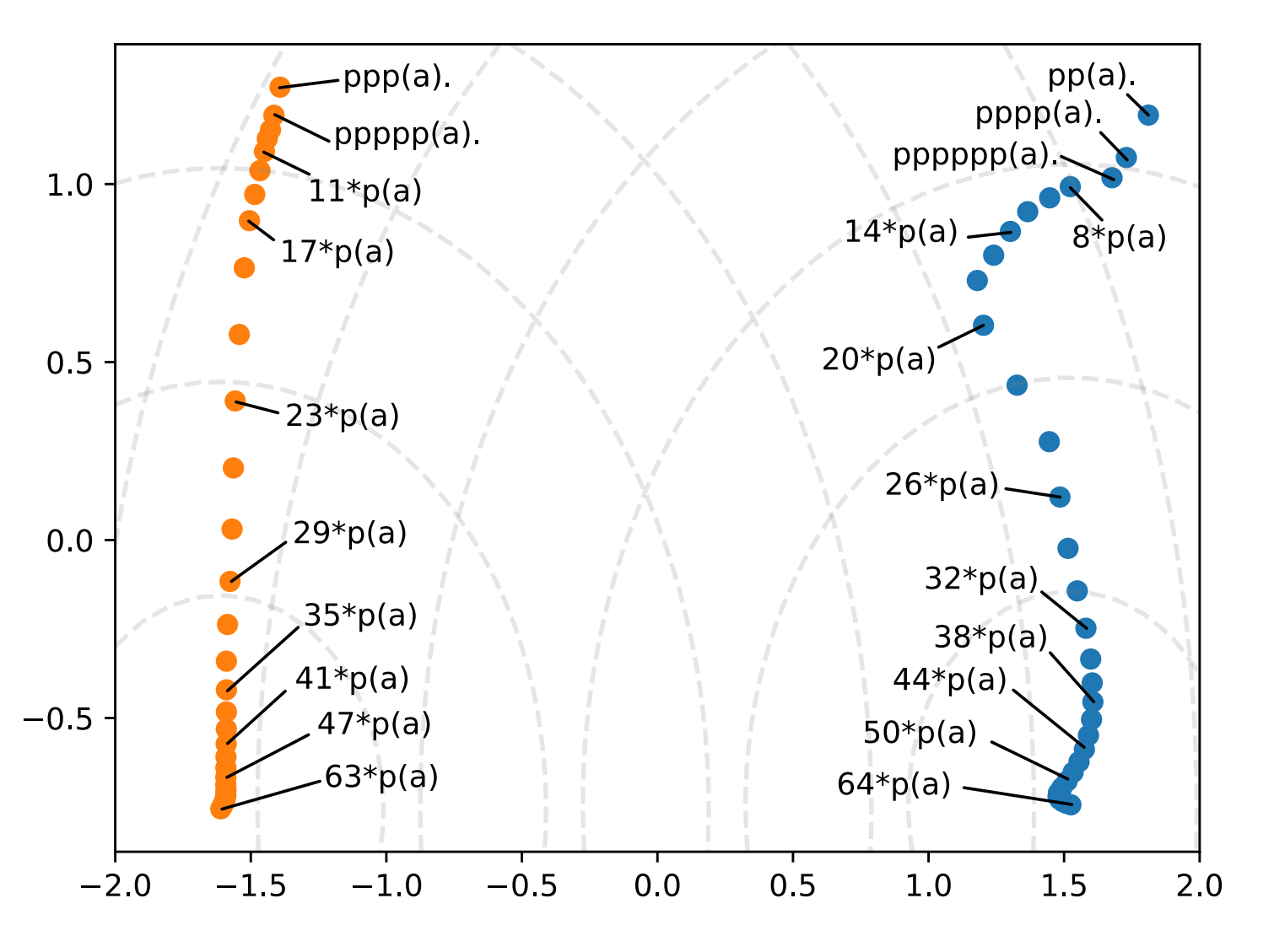

Figure 3: Repeating character predicates saturate the embedding and converge to respective points, equidistant lines are plotted in grey.

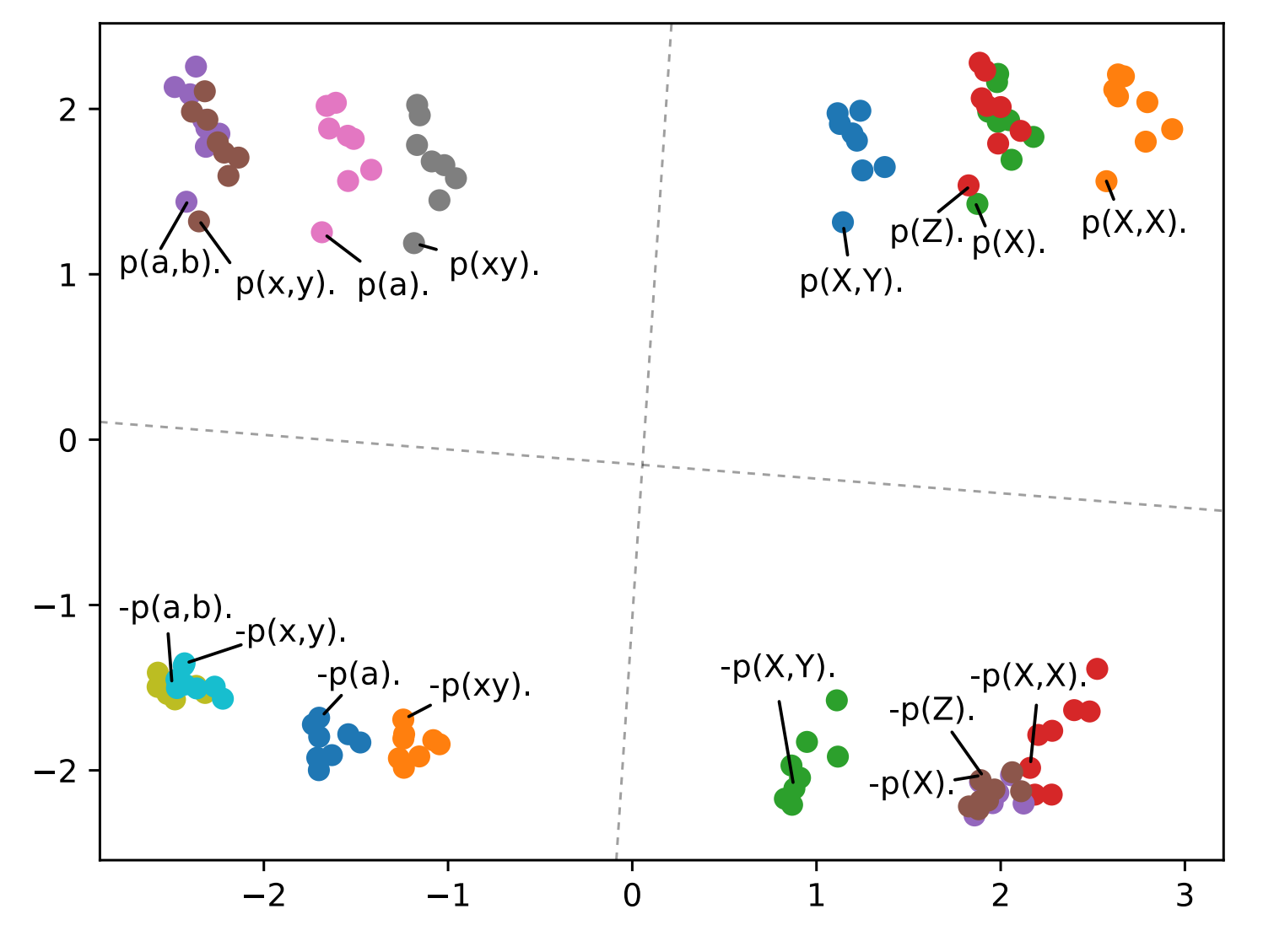

Figure 4: Structurally different literals first cluster by whether they are negated or grounded then by arity (grey lines added as visual aids).

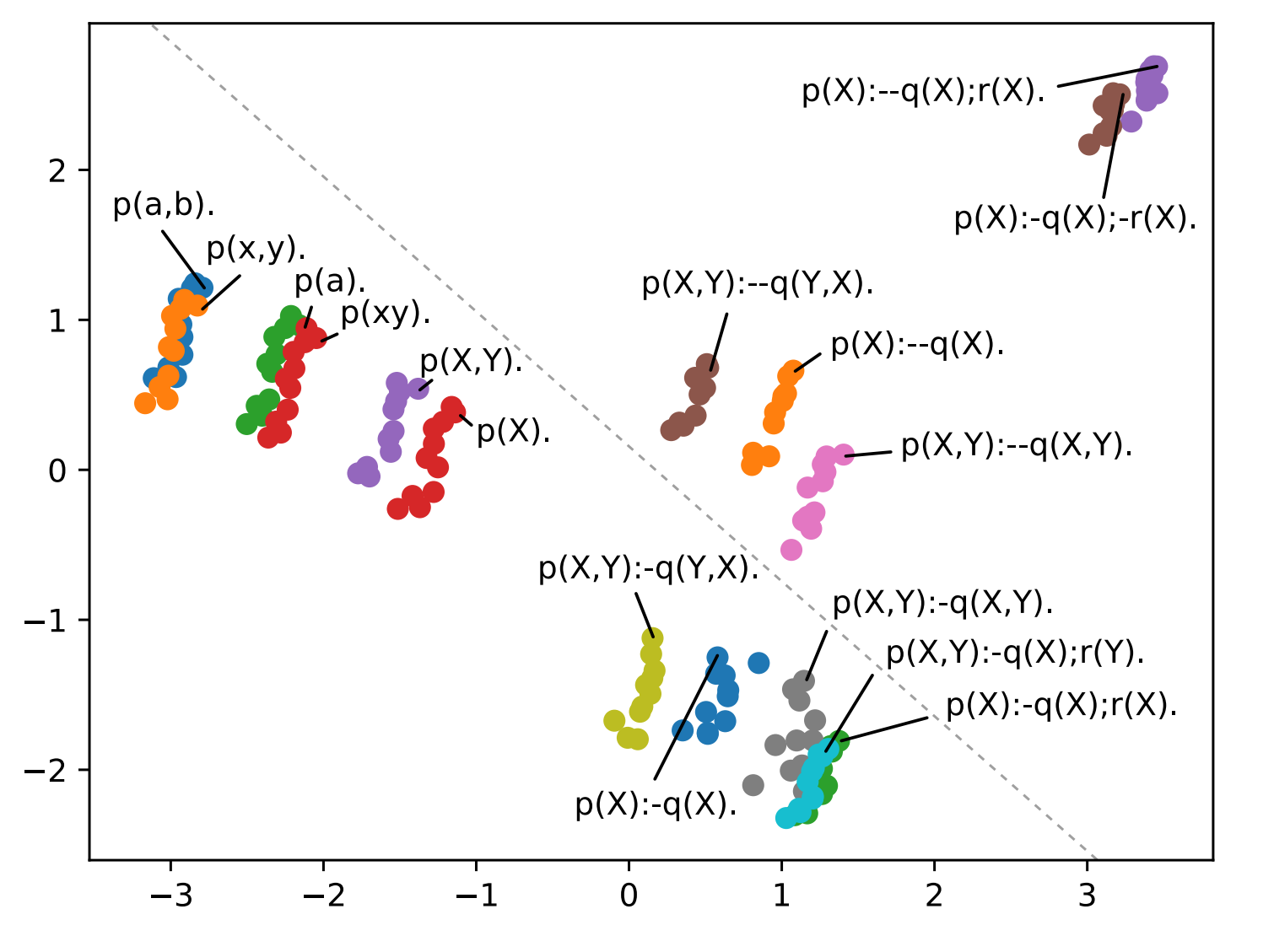

Figure 5: Rule embeddings form clusters based on their structure with a distinction between negated and positive rules (grey line as visual aid).

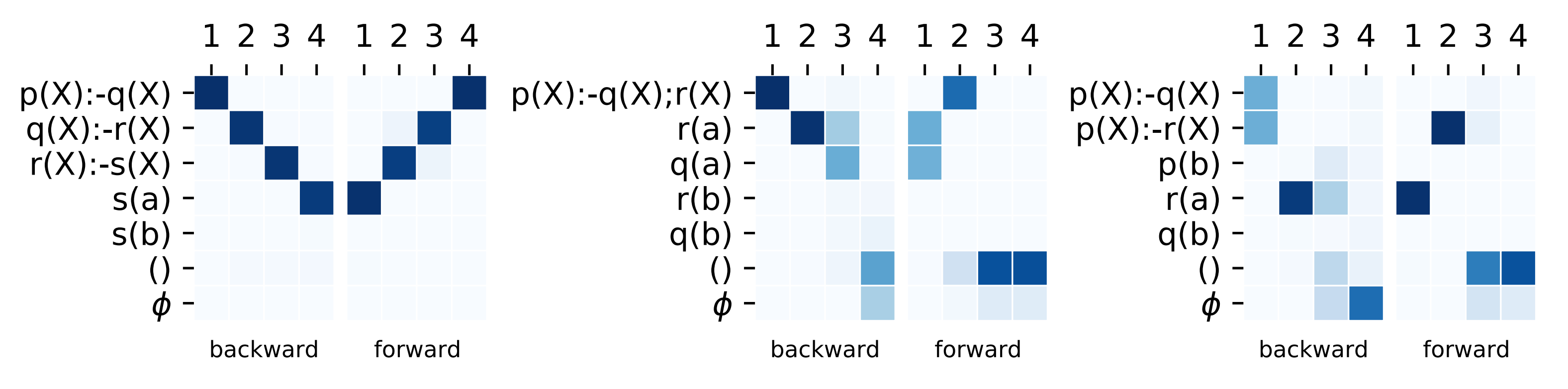

Figure 6: Attention maps produced for query $p(a)$ for IMA with softmax attention performing *backward* chaining in the left column and IMA with literal + rule embedding *forward* chaining in the right column on tasks 5 to 7.

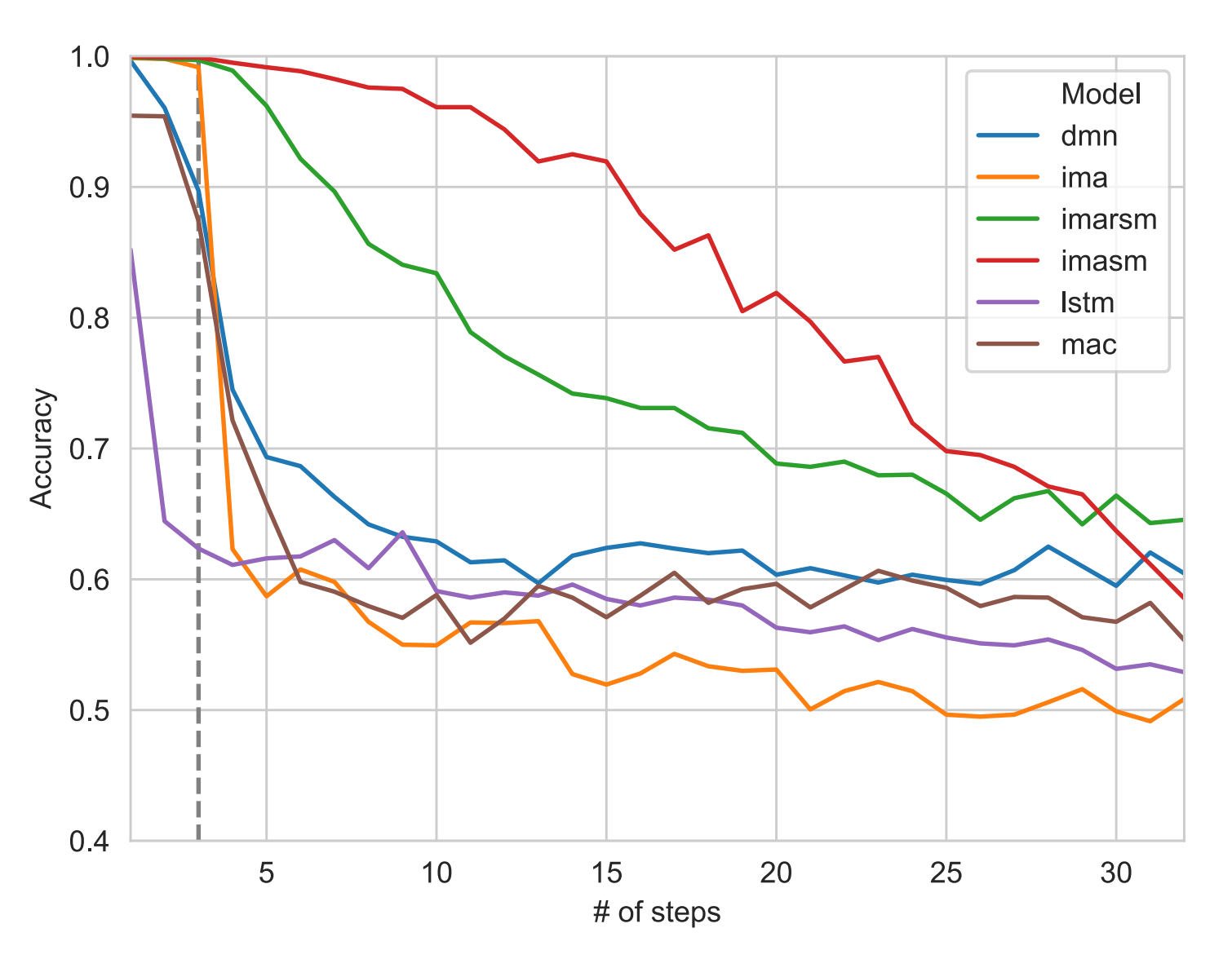

Figure 7: When models are iterated beyond the training number of steps (grey line) to perform increasing steps of deduction, the accuracy degrades for all models. We speculate this is caused by noise introduced to the state at each iteration.

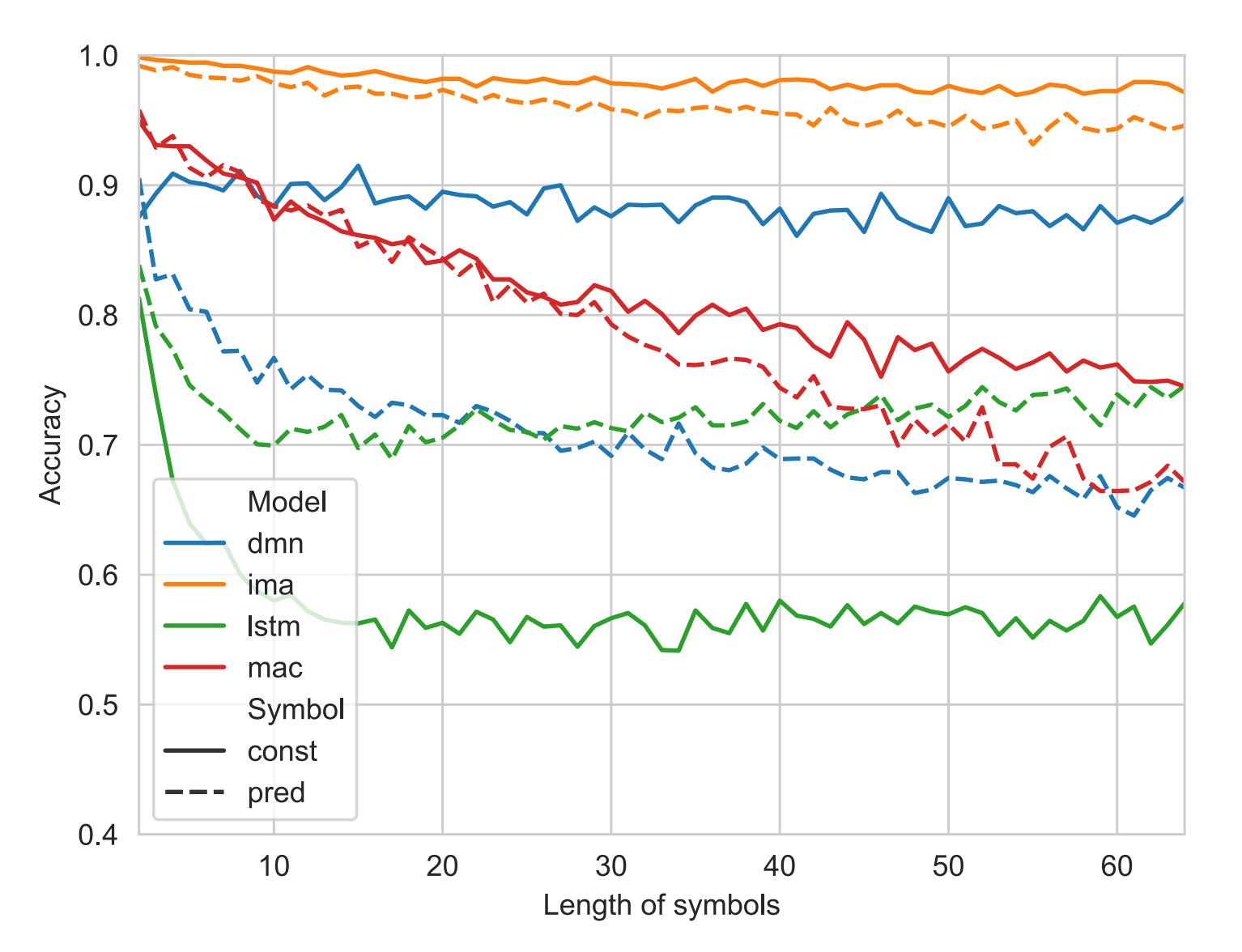

Figure 8: The models can cope, in particular IMA with literal embeddings, when predicate and constant symbols of increasing length are randomly generated. We speculate the models only look at few characters to determine uniqueness.

Conclusions

Fully differentiable models trained end-to-end have their inherent disadvantages: they seem to degrade in performance as the number of iterations is increased and the embedding space is limited in capacity. However, such networks might still hold the key to incorporate symbolic prior knowledge into a continuous space by understanding how that embedding space is organised to store symbolic information.

Forthcoming Research

Since such neural networks provide a differentiable but approximate reasoning engine over logic programs, in the future we hope to induce rules using continuous embeddings of logical rules and expand this to different mediums such as natural language.

References

[1] Nuri Cingillioglu and Alessandra Russo. Deeplogic: End-to-end logical reasoning. *AAAI-MAKE19*, 2019. `https://arxiv.org/abs/1805.07433`.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (3rd Edition)*. Pearson, 2016.