Learning Invariants through Soft Unification

Nuri Cingillioglu
nuric@imperial.ac.uk

Alessandra Russo
a.russo@imperial.ac.uk

Imperial College London

Abstract

Humans learn what variables are and how to use them at a young age. We explore whether machines can also learn and use variables solely from examples without requiring human pre-engineering. We propose Unification Networks, an end-to-end differentiable neural network approach capable of lifting examples into invariants and using those invariants to solve a given task. The core characteristic of our architecture is soft unification between examples that enables the network to generalise parts of the input into variables, thereby learning invariants. We evaluate our approach on five datasets to demonstrate that learning invariants captures patterns in the data and can improve performance over baselines.

Unification Networks

Our motivation stems from pretend play where children at a young age learn how symbols can vary to become other symbols such as how a sword can be substituted with a stick [2], also related to representative imitation [1]. We want a similar behaviour in which a model can detect and use **variables and invariants** to predict new example data points.

Algorithm 1: Unification Networks

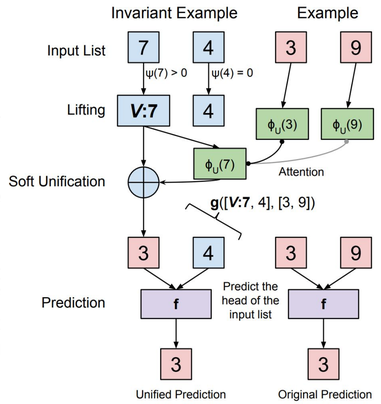
Input: Invariant I consisting of example G and variableness network ψ, example K, features network ϕ, unifying features network ϕ_U, upstream predictor network f
Output: Predicted label for example K

1 **begin** ▷ `Unification Network`
2 ⌊ **return** $f \circ g(I, K)$ ▷ `Predictions using Soft Unification` g
3 **begin** ▷ `Soft Unification function` g
4 **foreach** *symbol* s **in** G **do**
5 $\boldsymbol{A}_{s,:} \leftarrow \phi(s)$ ▷ `Features of` G, $\boldsymbol{A} \in \mathbb{R}^{|G| \times d}$
6 $\boldsymbol{B}_{s,:} \leftarrow \phi_U(s)$ ▷ `Unifying features of` G, $\boldsymbol{B} \in \mathbb{R}^{|G| \times d}$
7 **foreach** *symbol* s **in** K **do**
8 $\boldsymbol{C}_{s,:} \leftarrow \phi(s)$ ▷ `Features of` K, $\boldsymbol{C} \in \mathbb{R}^{|K| \times d}$
9 $\boldsymbol{D}_{s,:} \leftarrow \phi_U(s)$ ▷ `Unifying features of` K, $\boldsymbol{D} \in \mathbb{R}^{|K| \times d}$
10 **Let** $\boldsymbol{P} = \mathrm{softmax}(\boldsymbol{B}\boldsymbol{D}^T)$ ▷ `Attention map over symbols,` $\boldsymbol{P} \in \mathbb{R}^{|G| \times |K|}$
11 **Let** $\boldsymbol{E} = \boldsymbol{P}\boldsymbol{C}$ ▷ `Attended representations of` G, $\boldsymbol{E} \in \mathbb{R}^{|G| \times d}$
12 **foreach** *symbol* s **in** G **do**
13 $\boldsymbol{U}_{s,:} \leftarrow \psi(s)\boldsymbol{E}_{s,:} + (1 - \psi(s))\boldsymbol{A}_{s,:}$ ▷ `Unified representation of` I, $\boldsymbol{U} \in \mathbb{R}^{|G| \times d}$
14 **return** $\boldsymbol{U}$

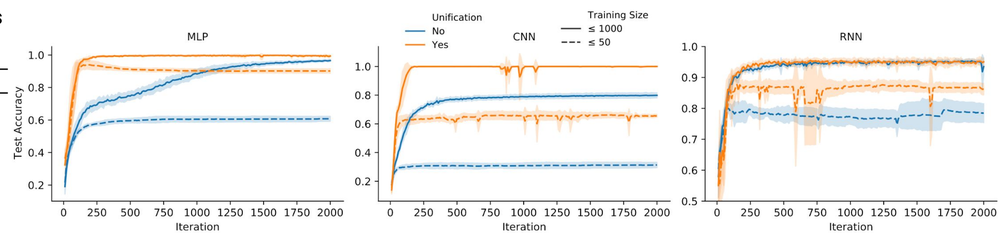
Assuming there exists a single common pattern in a dataset, we want to **predict the answer of an example K by transforming example G**:

1. We start with an example from the dataset, G, that consists of some input and output. Coupled with a variableness network ψ, the invariant predicts which of the symbols in G are variables.
2. We compute, an embedding ϕ to be used by the upstream network f and unifying features ϕ_U to capture a common latent space (lines 5-9).
3. Using the unifying features, we compute an attention map to assign new values to variable symbols, i.e. what should be the new value of the variable? (lines 10-11).
4. An intermediate unified representation is computed by interpolating between G and K based on how much each symbol is a variable in G. This representation is the result of soft unification of G towards K.
5. The unified representation is passed to an upstream predictor f.

We follow a graphical overview of soft unification whereby an invariant sequence "**V:7** 4" (blue) correctly unifies and predicts the answer of another example "3 9" (red). Here **V:7** represents a variable with default symbol 7, i.e. the default value of the variable. The pattern is the head of a sequence and the invariant example is transformed to let f (purple) predict the correct symbol using the unifying features (green).



Experiments



We observe **higher performance in data constraint settings** for our approach (in orange) against plain counter-parts (in blue) with MLP, CNN and RNN for Sequence, Grid and Sentiment datasets respectively using the following objective function:

$$J = \overbrace{\lambda_K \mathcal{L}_{\text{nll}}(f)}^{\text{Original output}} + \lambda_I \Big[\overbrace{\mathcal{L}_{\text{nll}}(f \circ g)}^{\text{Unification output}} + \overbrace{\tau \sum_{s \in \mathbb{S}} \psi(s)}^{\text{Sparsity}} \Big]$$

Adjust lambda K and I to pre-train or serve as a baseline

Add **sparsity constraint to avoid the trivial solution**: everything is a variable and invariant gets completely replaced

Training Size	1k				50	1k				
Supervision	Weak		Strong			Weak				Strong
# Invs / Model	1	3	1	3	3	N2N	GN2N	EntNet	QRN	MemNN
Mean	18.8	19.0	**5.1**	6.6	28.7	13.9	12.7	29.6	11.3	6.7
# > 5%	10	9	3	3	17	11	10	15	5	4

Model	UMN					IMA	
Training Size	2k				100	2k	
Supervision	Weak		Strong			Weak	Strong
# Invs	1	3	1	3	3	-	
Mean	37.7	37.6	**27.4**	29.0	47.1	38.8	31.5
# > 5%	10	10	10	11	12	11	11

Competitive or higher performance against comparable baselines in bAbI (top) and logic (bottom) datasets.

Datasets

We use four synthetic datasets of increasing complexity from fixed length sequences to iterative logical reasoning tasks that involve negation by failure. We also use one real-world dataset of sentiment analysis.

Dataset	Context	Query	Answer	Training Size
Sequence	8384	duplicate	8	≤ 1k, ≤ 50
Grid	0 0 3 0 1 6 8 5 7	corner	7	≤ 1k, ≤ 50
bAbI	Mary went to the kitchen. Sandra journeyed to the garden.	Where is Mary?	kitchen	1k, 50
Logic	p(X) ← q(X). q(a).	p(a).	True	2k, 100
Sentiment A.	easily one of the best films	Sentiment	Positive	1k, 50

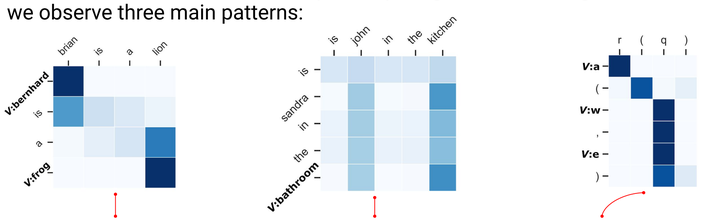
Analysis

To analyse the learned invariants, we threshold the variableness network ψ for each symbol and represent them in bold face with a **V** prefix. We observe that more likely than not symbols that contribute to the final answer become variables capturing the underlying pattern in the tasks.

***V*:john** travelled to the ***V*:office**	this ***V*:morning *V*:bill** went to the ***V*:school**
***V*:john *V*:left** the ***V*:football**	yesterday ***V*:bill** journeyed to the ***V*:park**
where is the ***V*:football**	where was ***V*:bill** before the ***V*:school**
office	park

5 8 6 4 const 2	0 ***V V***	0 1 0	0 0 1	***V*:i** (T) ← ***V*:l** (T),
***V*:8** 3 3 1 head 8	0 ***V V***	6 ***V*** 8	0 5 4	***V*:l** (U) ← ***V*:x** (U),
8 3 1 ***V*:5** tail 5	0 0 0	0 7 0	7 8 ***V***	***V*:x** (K) ← ***V*:n** (K),
***V*:1** 4 3 ***V*:1** dup 1	box	centre	corner	***V*:n** (***V*:o**) ⊢ ***V*:i** (***V*:o**)

When we look at the attention maps computing in line 10 of Algorithm 1, we observe three main patterns:



One-to-one mapping, bernhard interpolated towards brian

One-to-many, the model may *squeeze* the information to achieve sparsity

Many-to-one, upstream f learns to differentiate p(q,q) and p(q)

Limitations

- The invariants capture nothing about how the model actually solves the tasks or utilise the interpolated unified representations.
- No guarantee that the model will learn the "desired" invariant in a dataset or expose the expected common underlying pattern.

References

[1] Jean Piaget. The Psychology of Intelligence. Routledge, May 17, 2001. 126 pp. ISBN:0415254019.

[2] Joe L. Frost et al. The Developmental Benefits Of Playgrounds. Association for Childhood Education International, 2004. ISBN: 0871731649.