universität
innsbruck

Label invariant neural networks for formula embeddings

Miroslav Olšák

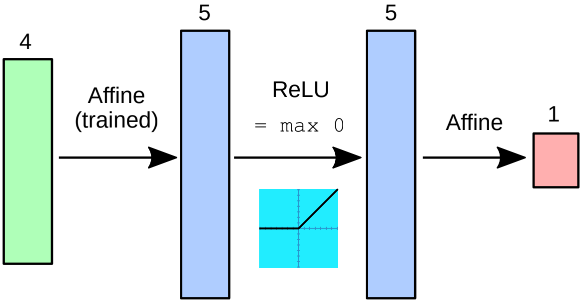
Overview

Content:

1. Graph Convolution NN
2. Formula Encoding
3. Experiments

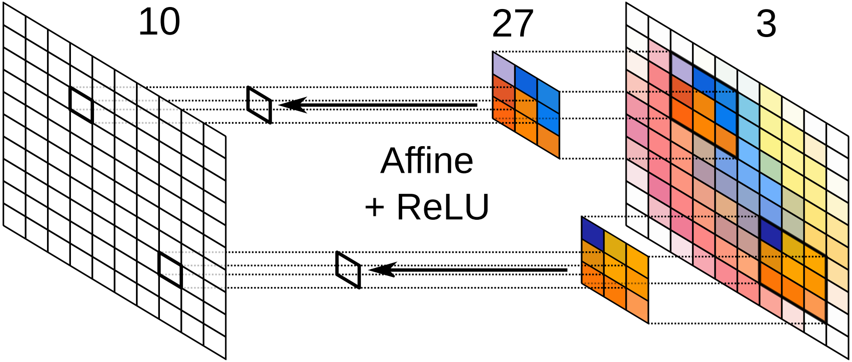
Neural networks

Basic unit flowing in a neural net is a real vector



Convolution

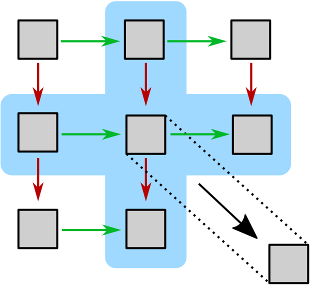
Basic layer for image processing: Picture $\rightarrow$ Picture

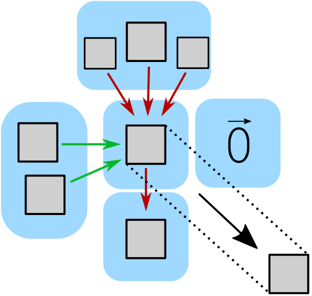


Applying the same basic layer locally around every pixel.

Convolution on a graph

Graph is a natural generalization of an image.

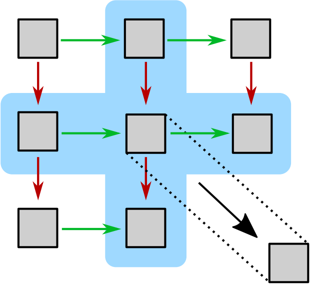


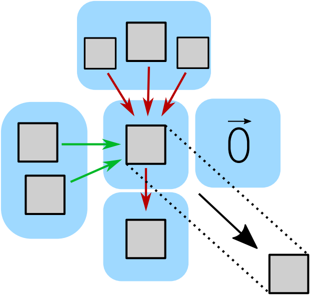


Evaluation of vertices by vectors $\to$ evaluation of the same vertices by vectors.

Convolution on a graph

Graph is a natural generalization of an image.





Evaluation of vertices by vectors $\to$ evaluation of the same vertices by vectors.

Multiple incoming edges of the same type are reduced via sum, maximum, mean, or their combination.

Mathematical formula

Formula structure

```
term = Variable of var_label
     | Application of func_label * term list;;
literal = PosTerm of term
        | NegTerm of term;;
cnf_input = literal list list;;
```

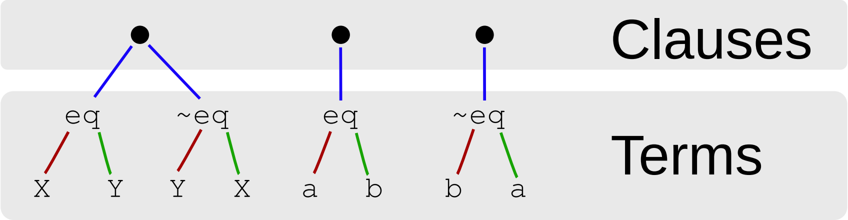
Example

```
(eq(X,Y) | ~eq(Y,X)) & eq(a,b) & ~eq(b,a)
```

Goal: Interpret as a graph

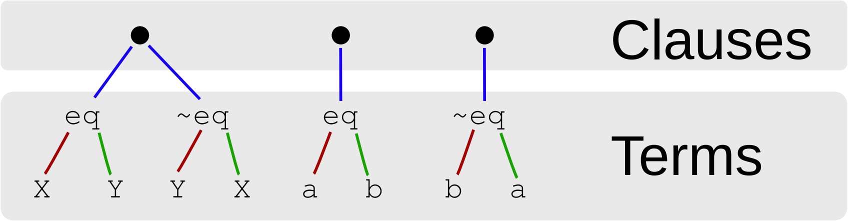
Mathematical formula

Tree

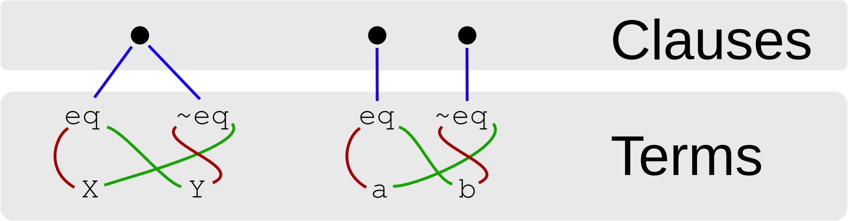


Mathematical formula

Tree



FormulaNet



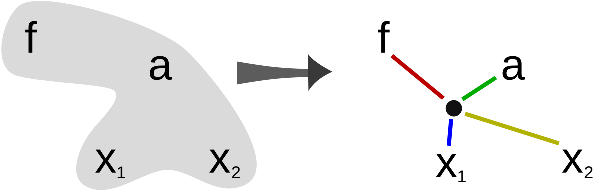
Labels and arity

- Every label is represented by a node.
- Application $a = f(x_1, x_2, \ldots, x_n)$ is represented by a set of 4-ary hyperedges $(f, a, x_1, x_2), (f, a, x_2, x_3), \ldots, (f, a, x_{n-1}, x_n)$.

Labels and arity

- Every label is represented by a node.
- Application $a = f(x_1, x_2, \dots, x_n)$ is represented by a set of 4-ary hyperedges $(f, a, x_1, x_2), (f, a, x_2, x_3), \dots, (f, a, x_{n-1}, x_n)$.

Hyperedges



Negation

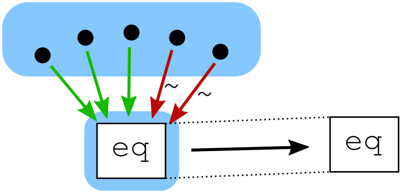
- Boolean negation is represented by real negation (multiplication by -1)

Negation

- Boolean negation is represented by real negation (multiplication by -1)
- We apply this negation to outcoming edges, if required.

Negation

- Boolean negation is represented by real negation (multiplication by -1)
- We apply this negation to outcoming edges, if required.
- How to keep incoming edges invariant under negation?
  - Reduction invariant under negation: average of min and max instead of max.
  - No biases. (Linear transformation, not affine one)
  - Odd activation function (tanh, not ReLU)
  - Labels initialized to zero.



Initialization

- Clauses:
  - three types in LeanCop (current goal, remaining goals, axioms),
  - two types in premise selection (conjecture, premises)
- Terms: constant embeddings for variables, literals, applications
- Label: Zero for relations, constant for functions

Head for MCTS

- Value:
  - Reduction of all clauses
  - Hidden layer (128)
  - Single output (sigmoid activation)
- Actions:
  - Current goal (literal) + clause + literal in clause
  - Hidden layer (128)
  - Single output for each action (softmax activation)

Network summary

- Three types of nodes: Clauses (dim 32), terms (dim 32), labels (dim 64).
- Initialization by a few learned vectors
- 5 layers of graph convolution
- Head corresponding to the task

Experiments

1. MCTS guidance for connection prover.
2. Premise selection with negative data.
3. Label guessing (same dataset as premise selection).

Monte Carlo Tree Search on LeanCop

Original experiment

- First iteration trained on 7,348 solved problems
- 1,000 new tree expansions for every bigstep
- First iteration solved 13,679 out of 31,250 problems, second 15,268.
- Last iteration solved 16,108 problems (best)

Monte Carlo Tree Search on LeanCop

Original experiment

- First iteration trained on 7,348 solved problems
- 1,000 new tree expansions for every bigstep
- First iteration solved 13,679 out of 31,250 problems, second 15,268.
- Last iteration solved 16,108 problems (best)

Our experiment

- First iteration trained on 4,595 solved problems
- Bigstep whenever root has been expanded 200-times.
- First iteration 0 solved 13,300 problems, second cca 14,000.
- Other iterations not done yet.

Premise selection

Dataset

- 32,524 queries extracted from Mizar
- Every query has balanced number of positive (useful) and negative results.
- Negative samples generated as the best scoring unuseful lemmas in k-nearest neighbors.

Premise selection

Dataset

- 32,524 queries extracted from Mizar
- Every query has balanced number of positive (useful) and negative results.
- Negative samples generated as the best scoring unuseful lemmas in *k*-nearest neighbors.

Results (on testing data)

	token sets	$\mathcal{M}_{\mathrm{lin}}$	$\mathcal{M}_{\mathrm{tree}}$	$\mathcal{M}_{\mathrm{nn}}$	label-inv
Acc	75.75%	71.41%	77.98%	79.44%	80.36%
TPR	?	80.54%	83.35%	82.00%	84.46%
TNR	?	62.28%	72.60%	76.88%	76.25%

Label guessing

Data

- 1,517,692 occurences of 13,339 symbols

21.5%	def
17.3%	skolem
2.0%	=
1.7%	m1_subset_1
1.2%	k1_zfmisc_1
1.2%	r2_hidden
1.0%	u1_struct_0
0.9%	v1_funct_1
0.9%	v1_xboole_0
0.8%	v2_struct_0
	...

Label guessing

Results

	Labels	Premise selection
Individual networks	78.40% (270)	80.36%
Combined network	74.93% (174)	80.10%
excluded “skolem” and “def”	65.49% (285)	

Thank you for your attention!

Miroslav Olšák