Machine Learning for the Working Logician (Computer Scientist)

Katya Komendantskaya

AI meets Formal Software Development

July 2012

Outline

Outline

1. Motivation
2. First experiments

Why Machine-Learning?

- ... Digital era means most of information (in science, industries, even art!) is stored/handled in electronic form.

Why Machine-Learning?

- ... Digital era means most of information (in science, industries, even art!) is stored/handled in electronic form.
- ... Computer-generated data may not make much sense to human users; or in fact, other computers!

Why Machine-Learning?

- ... Digital era means most of information (in science, industries, even art!) is stored/handled in electronic form.
- ... Computer-generated data may not make much sense to human users; or in fact, other computers!
- The volumes of data make it infeasible to be processed and interpreted manually.

Why Machine-Learning?

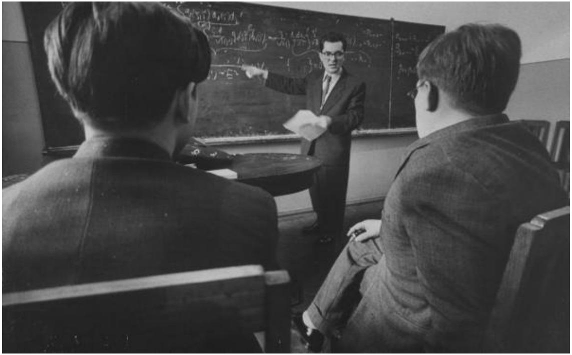
- ... Digital era means most of information (in science, industries, even art!) is stored/handled in electronic form.
- ... Computer-generated data may not make much sense to human users; or in fact, other computers!
- The volumes of data make it infeasible to be processed and interpreted manually.
- ... the only hope is, our machine-learning algorithms become fast and clever enough to do that dirty (pre-processing) work for us!

Why Machine-Learning?

- ... Digital era means most of information (in science, industries, even art!) is stored/handled in electronic form.
- ... Computer-generated data may not make much sense to human users; or in fact, other computers!
- The volumes of data make it infeasible to be processed and interpreted manually.
- ... the only hope is, our machine-learning algorithms become fast and clever enough to do that dirty (pre-processing) work for us!

So, why should we (logicians) care?

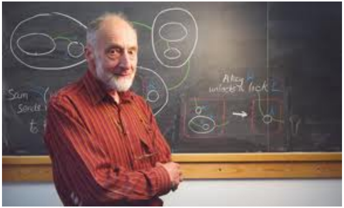
So, why should we (logicians) care?



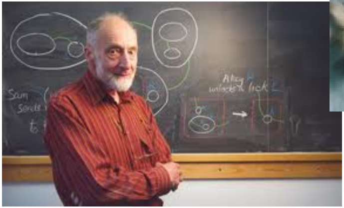
The answer is...

The answer is...

The answer is...



The answer is...



No matter what your personal choice is, . . .

- ... increasingly, theorems [be it mathematics or software/hardware verification] are proven IN automated provers.

No matter what your personal choice is, . . .

- ... increasingly, theorems [be it mathematics or software/hardware verification] are proven IN automated provers.
- ... electronic libraries may be data-mined (Learn2Reason);

No matter what your personal choice is, . . .

- ... increasingly, theorems [be it mathematics or software/hardware verification] are proven IN automated provers.
- ... electronic libraries may be data-mined (Learn2Reason);
- ... proof-search as process, in routine cases, can be statistically analysed (e.g. H.Duncan, Schulz et al.).

No matter what your personal choice is, . . .

- ... increasingly, theorems [be it mathematics or software/hardware verification] are proven IN automated provers.
- ... electronic libraries may be data-mined (Learn2Reason);
- ... proof-search as process, in routine cases, can be statistically analysed (e.g. H.Duncan, Schulz et al.).
- Manual handling of various proofs, strategies, libraries, becomes difficult.

Main applications in Automated Theorem Proving:

Where can we use ML?

ML in other areas of (Computer) Science:

Where data is abundant, and needs quick automated classification:

- robotics (from space rovers to small apps in domestic apploences, cars...);
- image processing;
- natural language processing;
- web search;
- computer network analysis;
- Medical diagnostics;
- etc, etc, ...

In all these areas, ML is a common tool-of-the-trade of the Computer Scientists, additional to their primary research specialisation.

Will this practice come to Automated theorem proving?

Automated reasoning does not need ML applications

...where AR does not need help

- verification (unlike in Medical diagnosis)
- language parsing (unlike in NLP)

Automated reasoning does not need ML applications

...where AR does not need help

- verification (unlike in Medical diagnosis)
- language parsing (unlike in NLP)

... where we do not trust them

- new theoretical break-throughs (formulation of new theorems);
- giving semantics to data (cf. Deep learning).

So,...

where do we both need ML-tools and trust them?

So,...

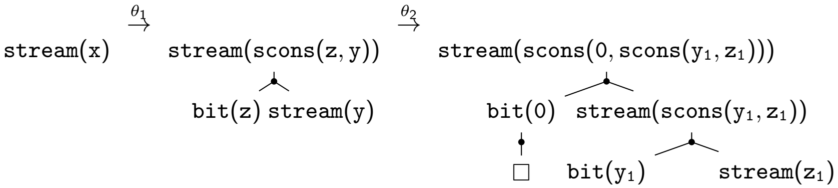
where do we both need ML-tools and trust them?

... will likely to be answered by the community of developers/practitioners, in the long run...

Outline

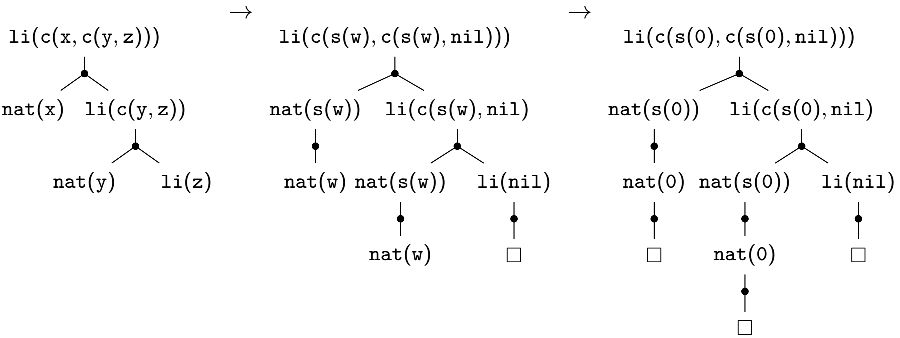
1. Motivation
2. First experiments

Data: coinductive proof-trees



Coinductive LP:

$$\begin{aligned}
\texttt{bit}(0) &\leftarrow \\
\texttt{bit}(1) &\leftarrow \\
\texttt{stream(scons (x,y))} &\leftarrow \texttt{bit(x)}, \texttt{stream(y)}
\end{aligned}$$



$$
\begin{aligned}
\mathtt{nat(0)} &\leftarrow \\
\mathtt{nat}(s(x)) &\leftarrow \mathtt{nat}(x) \\
\mathtt{list(nil)} &\leftarrow \\
\mathtt{list(cons\ (x,y))} &\leftarrow \mathtt{nat(x)}, \mathtt{list(y)}
\end{aligned}
$$

Experiment setting:

1. Big data sets of the proof trees for various programs;
2. An algorithm is used to convert the Proof-trees into feature vectors;
3. Three-layer back-propagation neural networks used for supervised learning.

All these and more details:

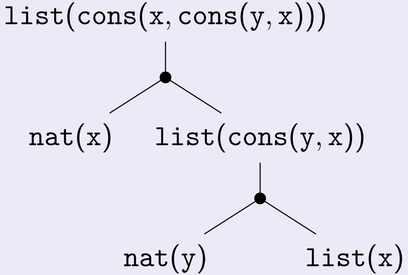
http://www.computing.dundee.ac.uk/staff/katya/MLCAP-man/

Classification task-1: well formed and ill-formed proofs

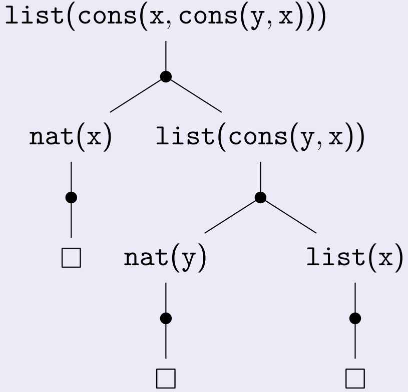
Accuracy of Neural Network recognition:

84.3% for Stream; 76.4% for List.

Well-formed



Ill-formed

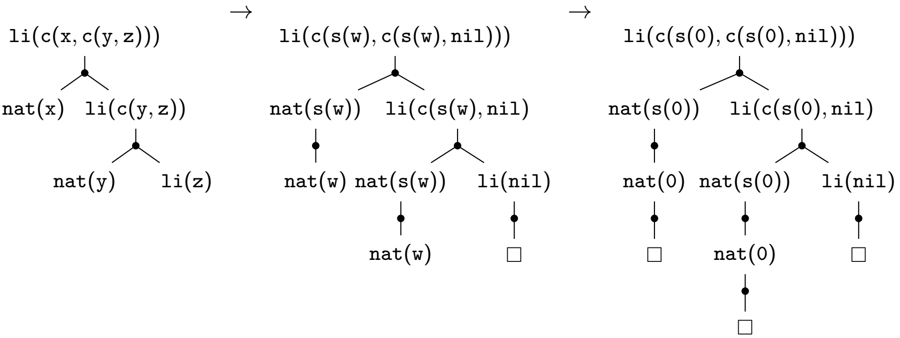


Possible use in Automated Proofs?

Classification task-2: Discovery of Proof-families (among well-formed proofs)

Accuracy of Neural Network recognition:

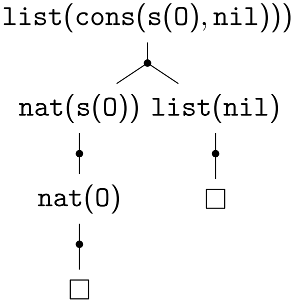
99.1% for Stream; 96.3% for List.

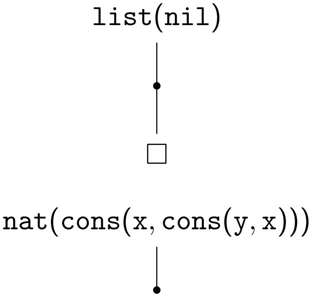


Classification task-2: Discovery of Proof-families

Negative examples

Well-formed trees on the right do not belomg to the proof family generated by the tree on the left.





Possible use in Automated Proofs?

Possible use in Automated Proofs?

- Cutting intermediate and routine proof-steps;
- Proof Speed up;
- Proof un-clattering;
- Generating suggestions (early warnings) if a certain proof step...

Possible use in Automated Proofs?

- Cutting intermediate and routine proof-steps;
- Proof Speed up;
- Proof un-clattering;
- Generating suggestions (early warnings) if a certain proof step...
  - belongs (or not) to a certain desired (previously discovered) proof family;

e.g. `Note: this proof-step belongs to the family of proofs in library N.`

Possible use in Automated Proofs?

- Cutting intermediate and routine proof-steps;
- Proof Speed up;
- Proof un-clattering;
- Generating suggestions (early warnings) if a certain proof step...
  - belongs (or not) to a certain desired (previously discovered) proof family;

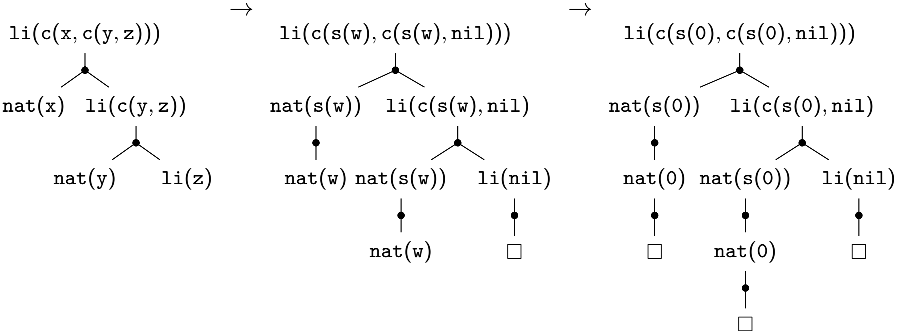
e.g. `Note: this proof-step belongs to the family of proofs in library N.`

  - belongs (or not) to a certain "bad" family discovered before;

Classification task-3: Discovery of Success Proof-families

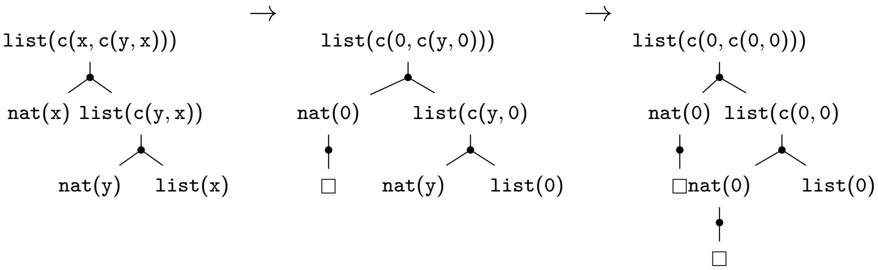
Accuracy of Neural Network recognition:

86% for List.



Classification task-3: Discovery of Success Proof-families

Negative training example:



Possible use in Automated Proofs?

This is all about a goal being potentially provable (by the statistical "look" of its proof unfolding).

Possible use in Automated Proofs?

This is all about a goal being potentially provable (by the statistical "look" of its proof unfolding). So, could be used for:

- Guiding a proof to success...
- Early warning if the proof step may contain some un-provable content...
- Guarding the proof steps...

Classification task-4: well-typed and ill-typed proofs in a family (among well-formed proofs)

Accuracy of Neural Network recognition:

85.7% for Stream.

$$\mathtt{stream}(\mathtt{x}) \xrightarrow{\theta_1} \mathtt{stream}(\mathtt{scons}(\mathtt{z},\mathtt{y})) \xrightarrow{\theta_2} \mathtt{stream}(\mathtt{scons}(0,\mathtt{scons}(\mathtt{y}_1,\mathtt{z}_1)))$$

$$\mathtt{bit}(\mathtt{z}) \quad \mathtt{stream}(\mathtt{y})$$

$$\mathtt{bit}(0) \quad \mathtt{stream}(\mathtt{scons}(\mathtt{y}_1,\mathtt{z}_1))$$

$$\Box \quad \mathtt{bit}(\mathtt{y}_1) \quad \mathtt{stream}(\mathtt{z}_1)$$

Classification task-4: well-typed and ill-typed proofs in a family

Negative examples:

$$\mathtt{stream}(\mathtt{x}) \xrightarrow{\theta_1} \mathtt{stream}(\mathtt{scons}(\mathtt{y},\mathtt{y})) \xrightarrow{\theta_2} \mathtt{stream}(\mathtt{scons}(\mathtt{scons}(\mathtt{z},\mathtt{z}),\mathtt{scons}(\mathtt{z},\mathtt{z})))$$

$$\mathtt{stream}(\mathtt{scons}(\mathtt{y},\mathtt{y})) \;\to\; \mathtt{bit}(\mathtt{y}) \quad \mathtt{stream}(\mathtt{y})$$

$$\mathtt{stream}(\mathtt{scons}(\mathtt{scons}(\mathtt{z},\mathtt{z}),\mathtt{scons}(\mathtt{z},\mathtt{z}))) \;\to\; \mathtt{bit}(\mathtt{scons}(\mathtt{z},\mathtt{z})) \quad \mathtt{stream}(\mathtt{scons}(\mathtt{z},\mathtt{z}))$$

$$\mathtt{stream}(\mathtt{scons}(\mathtt{z},\mathtt{z})) \;\to\; \mathtt{bit}(\mathtt{z}) \quad \mathtt{stream}(\mathtt{z})$$

$\ldots$

Possible use in Automated Proofs?

... same diagnostic use as with success proof families, but for cases when:

- we do not observe proof leaves: e.g., when ML-tool observes only patches of proofs, not the whole of proof-trees;
- when one works with lazy infinite proofs in coinductive cases;
- when it is on-line step-by-step diagnosis.

Classification task-5: well-typed and ill-typed proofs. (among well-formed proofs)

Accuracy of Neural Network recognition:

82.4% for Stream.

Note: membership in a proof family is not considered.

Possible use in Automated Proofs?

Rating of the classification applications

By accuracy

1. Problem 2.
2. Problem 3.
3. Problem 4.
4. Problem 5.
5. Problem 1.

By usefulness

1. Problem
2. Problem
3. Problem
4. Problem 5?
5. Problem 1.

Summary: applications in Automated Theorem Proving:

Where can we use ML?

1. Early proof dead-end diagnosis;
2. Early success identification; and thus cutting routine cases;
3. Interactive proof-hint generation?
4. Simplification of representation of proofs (by cutting routine/intermediate steps)?
5. Generalisation of proof families to strategies (Gudmund)?

More ideas?

Conclusions

More papers, prototype software, and technical reports on these topics are available on my webpage. Some papers are also uploaded on Dagstuhl meeting Wiki.

You will also find experiments/discussion of various implementation strategies.