Modular Architecture for Proof Advice

AITP Components

Cezary Kaliszyk

03 April 2016

University of Innsbruck, Austria

Talk Overview

- AI over formal mathematics
- Premise selection overview
- The methods tried so far
- Features for mathematics
- Internal guidance

AI OVER FORMAL MATHEMATICS

Inductive/Deductive AI over Formal Mathematics

- Alan Turing, 1950: *Computing machinery and intelligence*
- beginning of AI, Turing test
- last section of Turing's paper: *Learning Machines*
- Which intellectual fields to use for building AI?
  - *But which are the best ones [fields] to start [learning on] with?*
  - *...*
  - *Even this is a difficult decision. Many people think that a very abstract activity, like the playing of chess, would be best.*
- Our approach in the last decade:
  - Let's develop AI on large formal mathematical libraries!

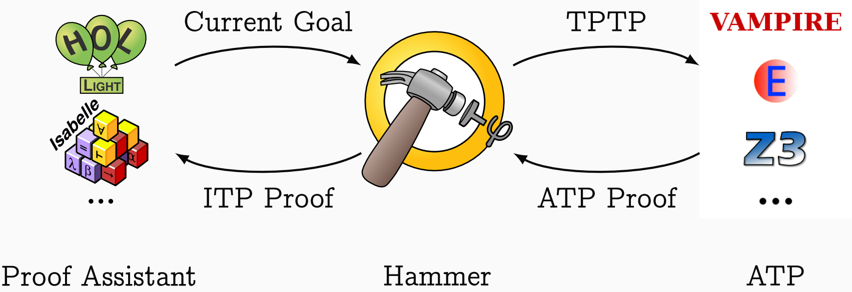
Why AI on large formal mathematical libraries?

- Hundreds of thousands of proofs developed over centuries
- Thousands of definitions/theories encoding our abstract knowledge
- All of it completely understandable to computers (*formality*)
- solid semantics: set/type theory
- built by safe (conservative) definitional extensions
- unlike in other "semantic" fields, inconsistencies are practically not an issue

Deduction and induction over large formal libraries

- Large formal libraries allow:
- strong deductive methods – Automated Theorem Proving
- inductive methods like Machine Learning (the libraries are large)
- combinations of deduction and learning
- examples of positive deduction-induction feedback loops:
- solve problems → learn from solutions → solve more problems ...

Useful: AI-ATP systems (Hammers)



AITP techniques

- High-level AI guidance:
  - *premise selection*: select the right lemmas to prove a new fact
  - based on suitable features (characterizations) of the formulas
  - and on learning lemma-relevance from many related proofs
- Mid-level AI guidance:
  - learn good ATP strategies/tactics/heuristics for classes of problems
  - learning lemma and concept re-use
  - learn conjecturing
- Low-level AI guidance:
  - guide (almost) every inference step by previous knowledge
  - good proof-state characterization and fast relevance

PREMISE SELECTION

Premise selection

Intuition

Given:

- set of theorems T (together with proofs)
- conjecture c

Find: minimal subset of T that can be used to prove c

More formally

$$\arg\min_{t \subseteq T} \{|t| \mid t \vdash c\}$$

In machine learning terminology

Multi-label classification

Input: set of samples $\mathbb{S}$, where samples are triples $s, F(s), L(s)$

- s is the sample ID
- $F(s)$ is the set of features of s
- $L(s)$ is the set of labels of s

Output: function f that predicts list of n labels (sorted by relevance) for set of features

Sample add_comm ($a + b = b + a$) could have:

- F(add_comm) = {"+", "=", "num"}
- L(add_comm) = {num_induct, add_0, add_suc, add_def}

Not exactly the usual machine learning problem

Observations

- Labels correspond to premises and samples to theorems
  - Very often same
- Similar theorems are likely to have similar premises
- A theorem may have a similar theorem as a premise
- Theorems sharing logical features are similar
- Theorems sharing rare features are very similar
- Fewer premises = they are more important
- Recently considered theorems and premises are important

Not exactly for the usual machine learning tools

Classifier requirements

- Multi-label output
  - Often asked for 1000 or more most relevant lemmas
- Efficient update
  - Learning time + prediction time small
  - User will not wait more than 10–30 sec for all phases
- Large numbers of features
  - Complicated feature relations

k-NEAREST NEIGHBOURS

Standard k-NN

Given set of samples $\mathbb{S}$ and features $\vec{f}$

1. For each $s \in \mathbb{S}$, calculate distance $d'(\vec{f}, s) = \|\vec{f} - \vec{F}(s)\|$
2. Take k samples with smallest distance, and return their labels

Feature weighting for k-NN: IDF

- If a symbol occurs in all formulas, it is boring (redundant)
- A rare feature (symbol, term) is much more informative than a frequent symbol
- IDF: Inverse Document Frequency:
- Features weighted by the logarithm of their inverse frequency

$$\mathrm{IDF}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

- This helps a lot in natural language processing
- Smothed IDF also helps:

$$\mathrm{IDF}_1(t, D) = \frac{1}{1 + |\{d \in D : t \in d\}|}$$

k-NN Improvements for Premise Selection

- Adaptive k
  - Rank (neighbours with smaller distance)

$$\mathrm{rank}(s) = |\{s' \mid d(f, s) < d(f, s')\}|$$

  - Age
- Include samples as labels
  - Different weights for sample labels
- Simple feature-based indexing
  - Euclidean distance, cosine distance, Jaccard similarity
  - Nearness

NAIVE BAYES

Naive Bayes

- For each fact f: Learn a function r_f that takes the features of a goal g and returns the predicted relevance.
- A baysian approach

$$
\begin{aligned}
& P(f \text{ is relevant for proving } g) \\
= \; & P(f \text{ is relevant} \mid g\text{'s features}) \\
= \; & P(f \text{ is relevant} \mid f_1, \ldots, f_n) \\
\propto \; & P(f \text{ is relevant}) \Pi_{i=1}^{n} P(f_i \mid f \text{ is relevant}) \\
\propto \; & \#f \text{ is a proof dependency} \cdot \Pi_{i=1}^{n} \frac{\#f_i \text{ appears when } f \text{ is a proof dependency}}{\#f \text{ is a proof dependency}}
\end{aligned}
$$

Naive Bayes: first adaptation to premise selection

$$\#f \text{ is a proof dependency} \cdot \Pi_{i=1}^{n} \frac{\#f_i \text{ appears when } f \text{ is a proof dependency}}{\#f \text{ is a proof dependency}}$$

- Uses a weighted sparse naive bayes prediction function:

$$r_f(f_1, \ldots, f_n) = \ln\, C + \sum_{j\,:\,c_j \neq 0} w_j \big(\ln\,(\pi\, c_j) - \ln\, C\big) + \sum_{j\,:\,c_j = 0} w_j\, \sigma$$

- Where $f_1, \ldots, f_n$ are the features of the goal.
- $w_1, \ldots, w_n$ are weights for the importance of the features.
- C is the number of proofs in which f occurs.
- $c_j \leq C$ is the number of such proofs associated with facts described by f_j (among other features).
- π and σ are predefined weights for known and unknown features.

Naive Bayes: second adaptation

extended features $\overline{F}(\phi)$ of a fact ϕ

features of ϕ and of the facts that were proved using ϕ
(only one iteration)

More precise estimation of the relevance of ϕ to prove γ:

$$
\begin{aligned}
&P(\phi \text{ is used in } \psi\text{'s proof}) \\
&\quad \cdot \prod_{f \in F(\gamma) \cap \overline{F}(\phi)} P\big(\psi \text{ has feature } f \mid \phi \text{ is used in } \psi\text{'s proof}\big) \\
&\quad \cdot \prod_{f \in F(\gamma) - \overline{F}(\phi)} P\big(\psi \text{ has feature } f \mid \phi \text{ is not used in } \psi\text{'s proof}\big) \\
&\quad \cdot \prod_{f \in \overline{F}(\phi) - F(\gamma)} P\big(\psi \text{ does not have feature } f \mid \phi \text{ is used in } \psi\text{'s proof}\big)
\end{aligned}
$$

All these probabilities can be computed efficiently!

Update two functions (tables):

- $t(\phi)$: number of times a fact ϕ occurs as a dependency
- $s(\phi, f)$: number of times a fact ϕ occurs as a dependency of a fact described by feature f

Then:

$$
\begin{aligned}
P(\phi \text{ is used in a proof of (any) } \psi) &= \frac{t(\phi)}{K} \\
P(\psi \text{ has feature } f \mid \phi \text{ is used in } \psi\text{'s proof}) &= \frac{s(\phi, f)}{t(\phi)} \\
P(\psi \text{ does not have feature } f \mid \phi \text{ is used in } \psi\text{'s proof}) &= 1 - \frac{s(\phi, f)}{t(\phi)} \\
&\approx 1 - \frac{s(\phi, f) - 1}{t(\phi)}
\end{aligned}
$$

RANDOM FORESTS

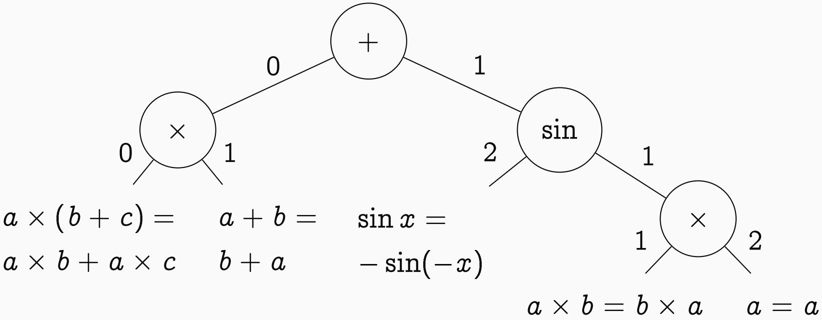
Random Forest Definition

A random forest is a set of decision trees constructed from random subsets of the dataset.

Characteristics

- easily parallelised
- high prediction speed (once trained :)
- good prediction quality (claimed e.g. in [Caruana2006])
- Offline forests: Agrawal et al. (2013)
  - developed for proposing ad bid phrases for web pages
  - trained periodically on whole set, old results discarded
- Online forests: Saffari et al. (2009)
  - developed for computer vision object detection
  - new samples added each tree a random number of times
  - split leafs when too big or good splitting features
  - features encountered first are higher up in trees: bias

Example Decision Tree



RF improvements for premise selection

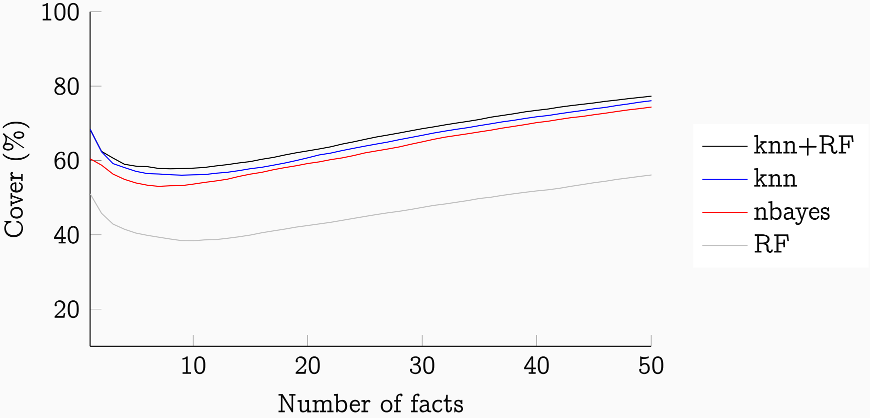
- Feature selection: Gini + feature frequency
- Modified tree size criterion
  - (number of labels logarithmic in number of all labels)
- Multi-path tree querying (introduce a few “errors”) with weighting

$$w = \prod_{d \in errors} f(d, m)$$

$$f(d, m) = \begin{cases} w & \text{simple} \\ \frac{w}{m-d} & \text{inverse} \\ w\frac{d}{m} & \text{linear} \end{cases}$$

- Combine tree / leaf results using harmonic mean

Comparison



100
80
60
40
20
Cover (%)
10
20
30
40
50
Number of facts
knn+RF
knn
nbayes
RF

Other Tried Premise Selection Techniques

- Syntactic methods
  - Neighbours using various metrics
  - Recursive: SInE, MePo
- Neural Networks (flat, SNoW)
  - Winnow, Perceptron
- Linear Regression
  - Needs feature and theorem space reduction
- Kernel-based multi-output ranking
  - Works better on small datasets

FEATURES

Features used so far for learning

- Symbols
  - symbol names or type-instances of symbols
- Types
  - type constants, type constructors, and type classes
- Subterms
  - various variable normalizations
- Meta-information
  - theory name, presence in various databases

Semantic Features

- The features have to express important semantic relations
- The features must be efficient
- In this work, features for:
  - Matching
  - Unification
- Efficiency achieved by using optimized ATP indexing trees:
  - discrimination trees
  - substitution trees
- Connections between subterms in a term
  - Paths in Term Graphs
- Validity of formulas in diverse finite models
  - semantic, but often expensive

GUIDANCE FOR ATPS

leanCoP: Lean Connection Prover (Jens Otten)

- Connected tableaux calculus
  - Goal oriented, good for large theories
- Regularly beats Metis and Prover9 in CASC
  - despite their much larger implementation
  - very good performance on some ITP challenges
- Compact Prolog implementation, easy to modify
  - Variants for other foundations: iLeanCoP, mLeanCoP
  - First experiments with machine learning: MaLeCoP
- Easy to imitate
  - leanCoP tactic in HOL Light

Internal Guidance for LeanCoP

Very simple calculus:

- Reduction unifies the current literal with a literal on the path
- Extension unifies the current literal with a copy of a clause

$$\frac{}{\{\}, M, \mathit{Path}} \quad \mathit{Axiom}$$

$$\frac{C, M, \mathit{Path} \cup \{L_2\}}{C \cup \{L_1\}, M, \mathit{Path} \cup \{L_2\}} \quad \mathit{Reduction}$$

$$\frac{C_2 \setminus \{L_2\}, M, \mathit{Path} \cup \{L_1\} \qquad C, M, \mathit{Path}}{C \cup \{L_1\}, M, \mathit{Path}} \quad \mathit{Extension}$$

FEMaLeCoP: Advice Overview and Used Features

- Advise the:
  - **selection of clause for every tableau extension step**
- Proof state: weighted vector of symbols (or terms)
  - extracted from all the literals on the active path
  - Frequency-based weighting (IDF)
  - Simple decay factor (using maximum)
- Consistent clausification
  - formula `?[X]: p(X)` becomes `p('skolem(?[A]:p(A),1)')`
- Advice using custom sparse naive Bayes
  - association of the features of the proof states
  - with contrapositives used for the successful extension steps

FEMaLeCoP: Data Collection and Indexing

- Slight extension of the saved proofs
  - Training Data: pairs (path, used extension step)
- External Data Indexing (incremental)
  - `te_num`: number of training examples
  - `pf_no`: map from features to number of occurrences $\in \mathbb{Q}$
  - `cn_no`: map from contrapositives to numbers of occurrences
  - `cn_pf_no`: map of maps of cn/pf co-occurrences
- Problem Specific Data
  - Upon start FEMaLeCoP reads
    - **only current-problem relevant** parts of the training data
  - cn_no and cn_pf_no filtered by contrapositives in the problem
  - pf_no and cn_pf_no filtered by possible features in the problem

Naive Bayes (1/2)

Estimate the relevance of each contrapositive φ by

$$\mathrm{P}(\varphi \text{ is used in a proof in state } \psi \mid \psi \text{ has features } \mathrm{F}(\gamma))$$

where $F(\gamma)$ are the features of the current path.

Naive Bayes (1/2)

Estimate the relevance of each contrapositive φ by

$$\mathrm{P}(\varphi \text{ is used in a proof in state } \psi \mid \psi \text{ has features } \mathrm{F}(\gamma))$$

where $F(\gamma)$ are the features of the current path.

Assuming the features are independent, this is:

$$P(\varphi \text{ is used in } \psi\text{'s proof})$$
$$\cdot \prod_{f \in F(\gamma) \cap F(\varphi)} P(\psi \text{ has feature } f \mid \varphi \text{ is used in } \psi\text{'s proof})$$
$$\cdot \prod_{f \in F(\gamma) - F(\varphi)} P(\psi \text{ has feature } f \mid \varphi \text{ is not used in } \psi\text{'s proof})$$
$$\cdot \prod_{f \in F(\varphi) - F(\gamma)} P(\psi \text{ does not have } f \mid \varphi \text{ is used in } \psi\text{'s proof})$$

Naive Bayes (2/2)

All these probabilities can be estimated (using training examples):

$$\sigma_1 \ln t + \sum_{f \in (\overline{f} \cap \overline{s})} i(f) \ln \frac{\sigma_2 s(f)}{t} + \sigma_3 \sum_{f \in (\overline{f} - \overline{s})} i(f) + \sigma_4 \sum_{f \in (\overline{s} - \overline{f})} i(f) \ln(1 - \frac{s(f)}{t})$$

where

- $\overline{f}$ are the features of the path
- $\overline{s}$ are the features that co-occurred with φ
- $t = cn\_no(\varphi)$
- $s = cn\_fp\_no(\varphi)$
- i is the IDF
- σ_* are experimentally chosen parameters

SUMMARY

Summary

- Formal Mathematics could be very interesting for AI
  - Easy to make arbitrarily many experiments
  - And conversely: AI is very useful
- Premise selection potential for improvement
  - Stronger techniques too slow or not precise?
- Internal guidance for Automated Theorem Proving
  - Fast learning algorithm, indexing, approximate features
- Characterization of mathematical reasoning