Whipping Satallax

A sadistic approach to internal guidance

Michael Färber Chad Brown

6 April 2016

- Introduction
- FEMaLeCoP
- Satallax
- Evaluation

Introduction

Chad Brown a.k.a. Marquis de Sade

Figure 1: Američan v Praze.

120 days of learning – a play in 3 acts

Protagonists

- Josef Urban
- Cezary Kaliszyk
- Daniel Kühlwein
- Chad Brown

Projects

- MaLeS: Machine Learning of Strategies, invent ATP strategies automatically
- MaLeCoP & FEMaLeCoP: (Fairly Efficient) Machine Learning Connection Prover
- Satallax: an ATP for higher-order logic

FEMaLeCoP

FEMaLeCoP = leanCoP + fast ML

The three steps to learning

1. Record which contrapositives (clause + literal) are useful in which prover state
2. Create efficient classifier from learnt data
3. Rank future choices using classifier

What to influence?

tableau extension step: choice of contrapositive

How to characterise prover state?

symbols of previously chosen literals on active path

Ranking

Naive Bayes

find contrapositive l (label) with maximal probability to be useful in conjunction with path symbols $\vec{f}$ (features)

$$r(l, \vec{f}) = P(l) \prod_i P(f_i \mid l)$$

In practice (simplified)

$$r(l, \vec{f}) = \log D_l + \sum_i \log(\mathrm{idf}(f_i)) c(l, f_i)$$

$$c(l, f) = \begin{cases} \sigma & \text{if } D_{l,f} = 0 \\ \log \frac{D_{l,f}}{D_l} & \text{otherwise} \end{cases}$$

D_l is occurrence of l, and $D_{l,f}$ is co-occurrence of l with f

Satallax

Satallax 101

Basic procedure

- Based on given clause algorithm
- Uses SAT solver to find contradictions among active clauses

Vocabulary

- Priority queue: holds proof commands such as Formula Processing, Mating, Confrontation, . . .
- Priority determined by a set of flags, which form a mode
- Set of modes with runtime weight is called strategy (MaLeS used to find modes / strategy)

ML-ATP questions

Questions

- Where to influence proof search?
- How to characterise prover state?

Point of influence

- More than 90% of commands on priority queue are ProcessProp and store only a term
- Influence priority of commands (caution not to influence too much for fairness towards other commands)
- Difference to FEMaLeCoP: also remember intermediate facts $\rightarrow$ "lemma learning"

Collecting training data

When to record data?

- Data recording during proof search can considerably hurt success rate
- Solution: Save data only once proof has been found

What data to save?

- Conjecture (if given)
- Axioms (problem premises)
- Processed terms + their priorities
- Refutation terms (set of terms actually used for the proof)

Training data postprocessing

Positive / negative examples

- Positive examples: Processed terms $\cap$ refutation terms
- Negative examples: All other processed terms

Options

- Discard terms with fresh variables
- Normalise all symbols in terms, i.e. $(a + b) + c = a + (b + c)$ becomes $c_1(c_1(c_2, c_3), c_4) = c_1(c_2, c_1(c_3, c_4))$
- Normalise only fresh variables
- Only keep axiom terms (to measure "premise selection effect")

Possible features

- Axioms
- Symbols of processed terms

Naive Bayes classification with monoid occurrences

Problem

- Only positive examples à la FEMaLeCoP give bad results
- How to integrate negative examples? Multiple classifiers, ...?

Solution

- Generalised classifier to store term occurrences as monoid types
- Allows easy extension of classifier to different kinds of occurrences (e.g. neutral examples) while keeping performance high

In Code

- Before: `lbl_no : ('l, int) Hashtbl.t`
- After: `lbl_no : ('l, LabelNo.t) Hashtbl.t`, where `LabelNo` is a `Monoid`

Monoids

Commutative monoid

Commutative monoid is $(M, +)$ with a neutral element $0 \in M$ s.t.:

- $(a + b) + c = a + (b + c)$
- $a + 0 = a$
- $a + b = b + a$

Monoids as label occurrences

- 0 represents the non-occurrence of a label.
- $+$ combines label occurrences.
- Commutativity of $+$: order of learnt labels does not matter.

Pair monoid for positive/negative examples

Let $M = (\mathbb{N} \times \mathbb{N}, +_M)$, $0_M = (0, 0)$ and $+_M$ pairwise addition. The first/second pair elements store positive/negative label occurrences.

The core ranking formula

Pair monoid ranking

$$r(l) = \frac{|p - n|}{p + n}(\sigma_p p + \sigma_n n)$$

- p, n ... number of positive/negative occurrences of l
- $\sigma_p = 1, \sigma_n = -1$
- $\frac{|p-n|}{p+n}$ ... "confidence"; the less controversial a label, the higher its influence

What about features?

did not increase success rate, but incurred performance decrease

Tuning of guidance parameters

Off-line tuning via training data

- Rank all examples with classifier
- For every positive example, sum up number of preceding negative examples
- Find guidance values with minimal sum

Particle Swarm Optimization

- Run ATP with different parameters and modify them automatically depending on how many problems solved

Outcome

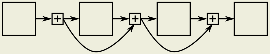
Off-line tuning fast to find initial values, but PSO more reliable

Evaluation

Evaluation

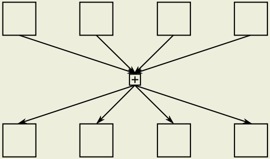
On-line learning

Learn data after each successful proof and use in all subsequent proof attempts (1x `fold`)



Off-line learning

Try all problems and save training data, then try all unsolved problems with guidance from training (2x `map`)



Results

Test set

THF version of Flyspeck from Cezary, with 14185 problems

Satallax without guidance

- 1s, auto strategy: 2717 problems
- 2s, auto strategy: 3394 problems
- 2s, auto strategy restricted to 1s modes: 2845 problems

Satallax with guidance

- On-line learning (1s): 3374 problems
- Off-line learning (1s): 3428 problems

Conclusion

When to use internal guidance?

- Satallax could be used to continually improve itself in an ITP situation with on-line learning
- When run on multiple cores, off-line learning a fast alternative

Future work

- Negative examples in FEMaLeCoP via new NB classifier with monoids
- Integrate internal guidance in ITP
- Use more training data for classifier (features . . . ?)
- Different features, e.g. TPTP