Clause Features for Theorem Prover Guidance

Jan Jakubův[1] Josef Urban[1]

AITP'19, Obergurgl, Austria, April 2019

[1]Czech Technical University in Prague, Czech Republic

Outline

Introduction: ATPs & Given Clauses

Enigma: The story so far...

Enigma: What's new?

Experiments: Hammering Mizar

Outline

Introduction: ATPs & Given Clauses

Enigma: The story so far. . .

Enigma: What's new?

Experiments: Hammering Mizar

Saturation-style ATPs

- Represent axioms and conjecture in First-Order Logic (FOL).
- $T \vdash C$ iff $T \cup \{\neg C\}$ is unsatisfiable.
- Translate $T \cup \{\neg C\}$ to clauses (ex. "$x = 0 \vee \neg P(f(x, x))$").
- Try to derive a contradiction.

Basic Loop

```
Proc = {}
Unproc = all available clauses
while (no proof found)
{
   select a given clause C from Unproc
   move C from Unproc to Proc
   apply inference rules to C and Proc
   put inferred clauses to Unproc
}
```

Clause Selection Heuristics in E Prover

- E Prover has several pre-defined clause weight functions. (and others can be easily implemented)
- Each weight function assigns a real number to a clause.
- Clause with the smallest weight is selected.

E Prover Strategy

- E strategy = E parameters influencing proof search (term ordering, literal selection, clause splitting, ...)
- Weight function gives the priority to a clause.
- Selection by several priority queues in a round-robin way

```
(10  *  ClauseWeight1(10,0.1,...),
  1  *  ClauseWeight2(...),
 20  *  ClauseWeight3(...))
```

Outline

Machine Learning of Given Clause

- **Idea:** Use machine learning methods to guide E prover.
- Analyze successful proof search to obtain training samples.
- *positives:* processed clauses used in the proof
- *negatives:* other processed clauses

Enigma Basics

- **Idea:** Use fast linear classifier to guide given clause selection!
- **ENIGMA** stands for...

Enigma Basics

- **Idea:** Use fast linear classifier to guide given clause selection!
- **ENIGMA** stands for. . .

Efficient learNing-based Inference Guiding MAchine

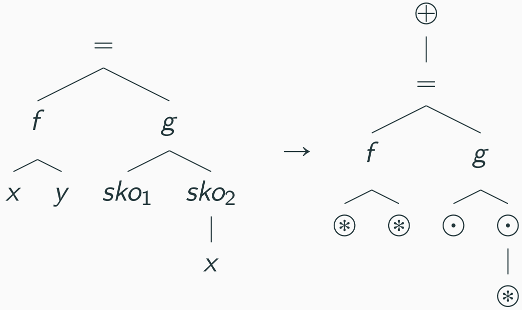
LIBLINEAR: Linear Classifier

- LIBLINEAR: open source library[1]
- **input:** positive and negative examples (float vectors)
- **output:** model ($\sim$ a vector of weights)
- **evaluation** of a generic vector: dot product with the model

[1] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

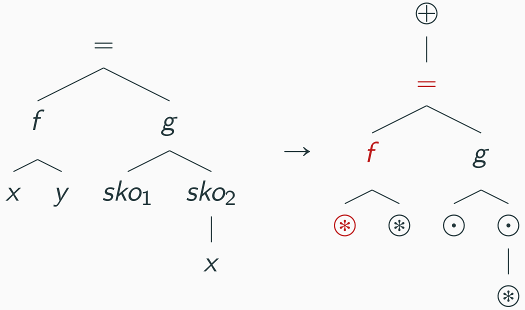
Clauses as Feature Vectors

Consider the literal as a tree and simplify (sign, vars, skolems).

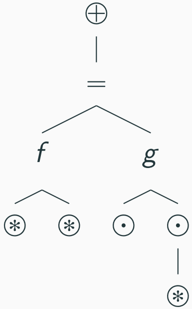


Clauses as Feature Vectors

Features are descending paths of length 3 (triples of symbols).



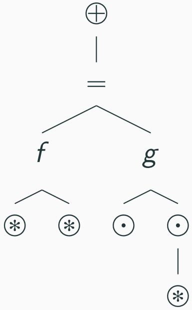
Collect and enumerate all the features. Count the clause features.



#	feature	count
1	($\oplus$,$=$,a)	0
⋮	⋮	⋮
11	($\oplus$,$=$,f)	1
12	($\oplus$,$=$,g)	1
13	($=$,f,$\circledast$)	2
14	($=$,g,$\odot$)	2
15	(g,$\odot$,$\circledast$)	1
⋮	⋮	⋮

Clauses as Feature Vectors

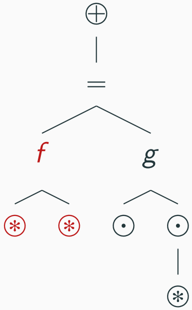
Take the counts as a feature vector.



#	feature	count
1	$(\oplus,=,a)$	0
$\vdots$	$\vdots$	$\vdots$
11	$(\oplus,=,f)$	1
12	$(\oplus,=,g)$	1
13	$(=,f,\circledast)$	2
14	$(=,g,\odot)$	2
15	$(g,\odot,\circledast)$	1
$\vdots$	$\vdots$	$\vdots$

Horizontal Features

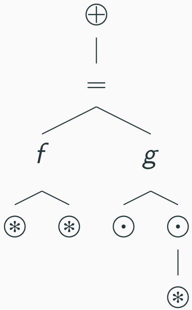
Function applications and arguments top-level symbols.



#	feature	count
1	$(\oplus,=,\mathrm{a})$	0
$\vdots$	$\vdots$	$\vdots$
100	$=(f,g)$	1
101	$f(\circledast,\circledast)$	1
102	$g(\odot,\odot)$	1
103	$\odot(\circledast)$	1
$\vdots$	$\vdots$	$\vdots$

Static Clause Features

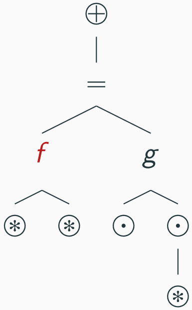
For a clause, its length and the number of pos./neg. literals.



#	feature	count/val
103	$\odot(\circledast)$	1
$\vdots$	$\vdots$	$\vdots$
200	len	9
201	pos	1
202	neg	0
$\vdots$	$\vdots$	$\vdots$

Static Symbol Features

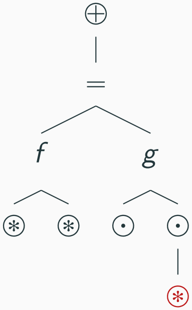
For each symbol, its count and maximum depth.



#	feature	count/val
202	neg	0
⋮	⋮	⋮
300	$\#_{\oplus}(f)$	1
301	$\#_{\ominus}(f)$	0
⋮	⋮	⋮
310	$\%_{\oplus}(\circledast)$	4
311	$\%_{\ominus}(\circledast)$	0
⋮	⋮	⋮

Static Symbol Features

For each symbol, its count and maximum depth.



#	feature	count/val
202	neg	0
⋮	⋮	⋮
300	$\#_{\oplus}(f)$	1
301	$\#_{\ominus}(f)$	0
⋮	⋮	⋮
310	$\%_{\oplus}(\circledast)$	4
311	$\%_{\ominus}(\circledast)$	0
⋮	⋮	⋮

Enigma Model Construction

1. Collect training examples from E runs (useful/useless clauses).
2. Enumerate all the features (π :: feature $\rightarrow$ int).
3. Translate clauses to feature vectors.
4. Train a LIBLINEAR classifier (w :: $\text{float}^{|\text{dom}(\pi)|}$).
5. Enigma model is $\mathcal{M} = (\pi, w)$.

Conjecture Features

- Enigma classifier $\mathcal{M}$ is independent on the goal conjecture!
- Improvement: Extend Φ_C with goal conjecture features.
- Instead of vector Φ_C take vector (Φ_C, Φ_G).

Given Clause Selection by Enigma

We have Enigma model $\mathcal{M} = (\pi, w)$ and a generated clause C.

1. Translate C to feature vector Φ_C using π.
2. Compute prediction:

$$\text{weight}(C) = \begin{cases} 1 & \text{iff } w \cdot \Phi_C > 0 \\ 10 & \text{otherwise} \end{cases}$$

Enigma Given Clause Selection

- We have implemented `Enigma` weight function in E.
- Given E strategy $\mathcal{S}$ and model $\mathcal{M}$.
- Construct new E strategy:
- $\mathcal{S} \odot \mathcal{M}$: Use $\mathcal{M}$ as the only weight function:

$$(1 \quad * \quad \texttt{Enigma}(\mathcal{M}))$$

- $\mathcal{S} \oplus \mathcal{M}$: Insert $\mathcal{M}$ to the weight functions from $\mathcal{S}$:

$$\begin{array}{rcl}(23 & * & \texttt{Enigma}(\mathcal{M}),\\ 3 & * & \texttt{StandardWeight(...)},\\ 20 & * & \texttt{StephanWeight(...)})\end{array}$$

Outline

Introduction: ATPs & Given Clauses

Enigma: The story so far. . .

Enigma: What's new?

Experiments: Hammering Mizar

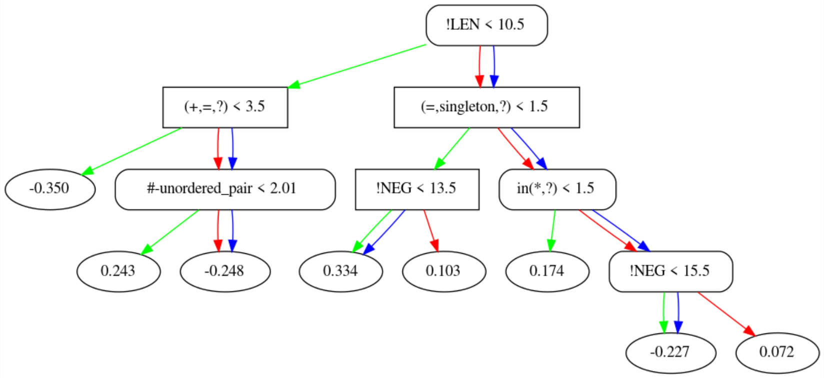
XGBoost Tree Boosting System

- **Idea:** Use decision trees instead of linear classifier.
- Gradient boosting library XGBoost.[2]
- Provides C/C++ API and Python (and others) interface.
- Uses exactly the same training data as LIBLINEAR.
- We use the same Enigma features.
- No need for training data balancing.

[2] `http://xgboost.ai`

XGBoost Models

- An XGBoost model consists of a set of decision trees.
- Leaf scores are summed and translated into a probability.



Feature Hashing

- With lot of training samples we have lot of features.
- LIBLINEAR/XGBoost can't handle too long vectors ($> 10^5$).
- Why? Input too big... Training takes too long...
- Solution: Reduce vector dimension with feature hashing.
- Encode features by strings and ...
- ... use a general purpose string hashing function.
- Values are summed in the case of a collision.

Outline

Experiments: Hammering Mizar

- MPTP: FOL translation of selected articles from Mizar Mathematical Library (MML).
- Contains 57880 problems.
- Small versions with (human) premise selection applied.
- Single good-performing E strategy $\mathcal{S}$ fixed.
- All strategies evaluated with time limit of 10 seconds.

Solved problems: one looping iteration

- Decision trees depth $= 9$.
- $\mathcal{M}^0$ is trained on problems solved by $\mathcal{S}$.
- $\mathcal{M}^n$ $(n > 0)$ is trained on problems solved by $\mathcal{S}$ and $\mathcal{S} \odot \mathcal{M}^i$ (for all $i < n$) and $\mathcal{S} \oplus \mathcal{M}^i$ (for all $i < n$).

	$\mathcal{S}$	$\mathcal{S} \odot \mathcal{M}^0$	$\mathcal{S} \oplus \mathcal{M}^0$	$\mathcal{S} \odot \mathcal{M}^1$	$\mathcal{S} \oplus \mathcal{M}^1$
solved	14933	16574	20366	21564	22839
$\mathcal{S}\%$	+0%	+10.5%	+35.8%	+43.8%	+52.3%
$\mathcal{S}+$	+0	+4364	+6215	+7774	+8414
$\mathcal{S}-$	-0	-2723	-782	-1143	-508

Solved problems: more loops

	$\mathcal{S}$	$\mathcal{S} \oplus \mathcal{M}^0$	$\mathcal{S} \oplus \mathcal{M}^1$	$\mathcal{S} \oplus \mathcal{M}^2$	$\mathcal{S} \oplus \mathcal{M}^3$
solved	14933	20366	22839	23467	23753
$\mathcal{S}\%$	+0%	+35.8%	+52.3%	+56.5%	+58.4
$\mathcal{S}+$	+0	+6215	+8414	+8964	+9274
$\mathcal{S}-$	-0	-782	-508	-430	-454

Solved problems: deeper trees

- Increase tree depth to 12 and 16.
- Train the model on the same data as $\mathcal{M}^3$.

	$\mathcal{S} \odot \mathcal{M}^3_{12}$	$\mathcal{S} \oplus \mathcal{M}^3_{12}$	$\mathcal{S} \odot \mathcal{M}^3_{16}$	$\mathcal{S} \oplus \mathcal{M}^3_{16}$
solved	24159	24701	25100	25397
$\mathcal{S}$%	+61.1%	+64.8%	+68.0%	+70.0%
$\mathcal{S}+$	+9761	+10063	+10476	+10647
$\mathcal{S}-$	-535	-295	-309	-183

Training Statistics: different tree depths

- 1.8 M features (hashed to 2^{15}).
- Vector dimension is 2^{16}.
- Input trains file is 38 GB
- ... and contains 63 M training samples (4.2M pos x 59M neg)
- ... with 5000 M non-zero values (density 0.1%).

depth	error	real time	CPU time	size (MB)	speed
9	0.201	2h41m	4d20h	5.0	5665.6
12	0.161	4h12m	8d10h	17.4	4676.9
16	0.123	6h28m	11d18h	54.7	3936.4

Thank you.

Questions?