Clause Features for Theorem Prover Guidance

Jan Jakubův¹  Josef Urban¹
AITP’19, Obergurgl, Austria, April 2019

¹Czech Technical University in Prague, Czech Republic
Outline

Introduction: ATPs & Given Clauses

Enigma: The story so far…

Enigma: What’s new?

Experiments: Hammering Mizar
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).
- \( T \models C \) iff \( T \cup \{\neg C\} \) is unsatisfiable.
- Translate \( T \cup \{\neg C\} \) to clauses (ex. “\( x = 0 \lor \neg P(f(x, x)) \)”).
- Try to derive a contradiction.
Basic Loop

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

\[(10 \times \text{ClauseWeight1}(10,0.1,...),
  1 \times \text{ClauseWeight2}(...),
  20 \times \text{ClauseWeight3}(...))\]
Outline

Introduction: ATPs & Given Clauses

Enigma: The story so far...

Enigma: What’s new?

Experiments: Hammering Mizar
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 (∼ a vector of weights)
- **evaluation** of a generic vector: dot product with the model

\(^1\)http://www.csie.ntu.edu.tw/~cjlin/liblinear/
Consider the literal as a tree and simplify (sign, vars, skolems).
Features are descending paths of length 3 (triples of symbols).
Collect and enumerate all the features. Count the clause features.

<table>
<thead>
<tr>
<th>#</th>
<th>feature</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((\oplus,=,a))</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>((\oplus,=,f))</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>((\oplus,=,g))</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>((=,f,\odot))</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>((=,g,\odot))</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>((g,\odot,\odot))</td>
<td>1</td>
</tr>
</tbody>
</table>
Take the counts as a **feature vector**.

```
<table>
<thead>
<tr>
<th>#</th>
<th>feature</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(⊕,=,a)</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>(⊕,=,f)</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>(⊕,=,g)</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>(=,f,ο)</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>(=,g,ο)</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>(g,ο,*)</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Horizontal Features

Function applications and arguments top-level symbols.

```
<table>
<thead>
<tr>
<th>#</th>
<th>feature</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((\oplus,=,a))</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>((f,g))</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>(f(\odot,\odot))</td>
<td>1</td>
</tr>
<tr>
<td>102</td>
<td>(g(\odot,\odot))</td>
<td>1</td>
</tr>
<tr>
<td>103</td>
<td>(\odot(\odot))</td>
<td>1</td>
</tr>
</tbody>
</table>
```
Static Clause Features

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

```
<table>
<thead>
<tr>
<th>#</th>
<th>feature</th>
<th>count/val</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>⊕(⊛)</td>
<td>1</td>
</tr>
<tr>
<td>200</td>
<td>len</td>
<td>9</td>
</tr>
<tr>
<td>201</td>
<td>pos</td>
<td>1</td>
</tr>
<tr>
<td>202</td>
<td>neg</td>
<td>0</td>
</tr>
</tbody>
</table>
```
Static Symbol Features

For each symbol, its count and maximum depth.

\[
\begin{array}{c}
\begin{array}{c}
\oplus \\
\quad \\
\quad =
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
f \\
\ast
\end{array} \\
\ast \\
\ast \\
\ast
\end{array}
\\begin{array}{c}
\begin{array}{c}
g \\
\ast
\end{array}
\end{array}
\end{array}
\]

<table>
<thead>
<tr>
<th>#</th>
<th>feature</th>
<th>count/val</th>
</tr>
</thead>
<tbody>
<tr>
<td>202</td>
<td>neg</td>
<td>0</td>
</tr>
<tr>
<td>300</td>
<td>#_{\oplus}(f)</td>
<td>1</td>
</tr>
<tr>
<td>301</td>
<td>#_{\ominus}(f)</td>
<td>0</td>
</tr>
<tr>
<td>310</td>
<td>%_{\oplus}(\ast)</td>
<td>4</td>
</tr>
<tr>
<td>311</td>
<td>%_{\ominus}(\ast)</td>
<td>0</td>
</tr>
</tbody>
</table>
Static Symbol Features

For each symbol, its count and **maximum depth**.

```
<table>
<thead>
<tr>
<th>#</th>
<th>feature</th>
<th>count/val</th>
</tr>
</thead>
<tbody>
<tr>
<td>202</td>
<td>neg</td>
<td>0</td>
</tr>
<tr>
<td>300</td>
<td>$#_\oplus(f)$</td>
<td>1</td>
</tr>
<tr>
<td>301</td>
<td>$#_\ominus(f)$</td>
<td>0</td>
</tr>
<tr>
<td>310</td>
<td>$%_\oplus(\ast)$</td>
<td>4</td>
</tr>
<tr>
<td>311</td>
<td>$%_\ominus(\ast)$</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Diagram:
```
  ⊕
  /  \\
  ☐  ☐
```

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

- Enigma classifier $\mathcal{M}$ is independent on the goal conjecture!
- Improvement: Extend $\Phi_C$ with goal conjecture features.
- Instead of vector $\Phi_C$ take vector $(\Phi_C, \Phi_G)$. 
We have Enigma model $\mathcal{M} = (\pi, w)$ and a generated clause $C$.

1. Translate $C$ to feature vector $\Phi_C$ using $\pi$.
2. Compute prediction:

$$\text{weight}(C) = \begin{cases} 
1 & \text{iff } w \cdot \Phi_C > 0 \\
10 & \text{otherwise}
\end{cases}$$
• We have implemented Enigma weight function in E.
• Given E strategy $S$ and model $M$.
• Construct new E strategy:
  • $S \odot M$: Use $M$ as the only weight function:
    \[(1 \times \text{Enigma}(M))\]
  • $S \oplus M$: Insert $M$ to the weight functions from $S$:
    \[(23 \times \text{Enigma}(M), \quad 3 \times \text{StandardWeight}(\ldots), \quad 20 \times \text{StephanWeight}(\ldots))\]
Outline

Introduction: ATPs & Given Clauses

Enigma: The story so far...

Enigma: What’s new?

Experiments: Hammering Mizar
• **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

Introduction: ATPs & Given Clauses

Enigma: The story so far…

Enigma: What’s new?

Experiments: Hammering Mizar
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 $S$ 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 $S$.
- $\mathcal{M}^n \ (n > 0)$ is trained on problems solved by $S$ and $S \odot \mathcal{M}^i \ (\text{for all } i < n)$ and $S \oplus \mathcal{M}^i \ (\text{for all } i < n)$.

<table>
<thead>
<tr>
<th></th>
<th>$S$</th>
<th>$S \odot \mathcal{M}^0$</th>
<th>$S \oplus \mathcal{M}^0$</th>
<th>$S \odot \mathcal{M}^1$</th>
<th>$S \oplus \mathcal{M}^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>14933</td>
<td>16574</td>
<td>20366</td>
<td>21564</td>
<td>22839</td>
</tr>
<tr>
<td>$S%$</td>
<td>+0%</td>
<td>+10.5%</td>
<td>+35.8%</td>
<td>+43.8%</td>
<td>+52.3%</td>
</tr>
<tr>
<td>$S+$</td>
<td>+0</td>
<td>+4364</td>
<td>+6215</td>
<td>+7774</td>
<td>+8414</td>
</tr>
<tr>
<td>$S-$</td>
<td>-0</td>
<td>-2723</td>
<td>-782</td>
<td>-1143</td>
<td>-508</td>
</tr>
</tbody>
</table>
### Solved problems: more loops

<table>
<thead>
<tr>
<th></th>
<th>$S$</th>
<th>$S \oplus M^0$</th>
<th>$S \oplus M^1$</th>
<th>$S \oplus M^2$</th>
<th>$S \oplus M^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>14933</td>
<td>20366</td>
<td>22839</td>
<td>23467</td>
<td>23753</td>
</tr>
<tr>
<td>$S%$</td>
<td>+0%</td>
<td>+35.8%</td>
<td>+52.3%</td>
<td>+56.5%</td>
<td>+58.4%</td>
</tr>
<tr>
<td>$S+$</td>
<td>+0</td>
<td>+6215</td>
<td>+8414</td>
<td>+8964</td>
<td>+9274</td>
</tr>
<tr>
<td>$S-$</td>
<td>-0</td>
<td>-782</td>
<td>-508</td>
<td>-430</td>
<td>-454</td>
</tr>
</tbody>
</table>
Solved problems: deeper trees

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

<table>
<thead>
<tr>
<th></th>
<th>$S \odot M^3_{12}$</th>
<th>$S \oplus M^3_{12}$</th>
<th>$S \odot M^3_{16}$</th>
<th>$S \oplus M^3_{16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>24159</td>
<td>24701</td>
<td>25100</td>
<td>25397</td>
</tr>
<tr>
<td>$S%$</td>
<td>+61.1%</td>
<td>+64.8%</td>
<td>+68.0%</td>
<td>+70.0%</td>
</tr>
<tr>
<td>$S+$</td>
<td>+9761</td>
<td>+10063</td>
<td>+10476</td>
<td>+10647</td>
</tr>
<tr>
<td>$S-$</td>
<td>-535</td>
<td>-295</td>
<td>-309</td>
<td>-183</td>
</tr>
</tbody>
</table>
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%).

<table>
<thead>
<tr>
<th>depth</th>
<th>error</th>
<th>real time</th>
<th>CPU time</th>
<th>size (MB)</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0.201</td>
<td>2h41m</td>
<td>4d20h</td>
<td>5.0</td>
<td>5665.6</td>
</tr>
<tr>
<td>12</td>
<td>0.161</td>
<td>4h12m</td>
<td>8d10h</td>
<td>17.4</td>
<td>4676.9</td>
</tr>
<tr>
<td>16</td>
<td>0.123</td>
<td>6h28m</td>
<td>11d18h</td>
<td>54.7</td>
<td>3936.4</td>
</tr>
</tbody>
</table>
Thank you.

Questions?