Machine Learning for Automated Theorem Proving

Bernhard Kragl

Machine Learning and Applications Course
Spring 2015
Outline

• Introduction to Automated Theorem Proving

• Machine Learning Applications
  – Strategy selection
  – (Premise selection)
Automated Theorem Proving

Reasoning question in some Logic

\[ A \implies B \]

\[ A \rightarrow B \text{ valid} \]

\[ A \land \neg B \text{ unsat} \]

Input

Set of clauses

Theorem Prover

Output

Proof (of unsat)
Expressiveness vs Automation

- **SAT** (boolean satisfiability)
  \[(x_1 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1)\]

- **SMT** (satisfiability modulo theories)
  \[x + 2 = y \rightarrow f(rd(wr(A, x, 3), y - 2)) = f(y - x + 1)\]

- **FOL** (first-order logic)
  \[\exists x \forall y \ P(x, y) \rightarrow \forall y \exists x \ P(x, y)\]

- **HOL** (higher order logic)
  \[\exists f \ (\forall y \in B)(\exists x \in A) : f(x) = y\]
Applications and Success Stories

• Proof of Robbins conjecture using theorem prover EQP (1996)
• Design and verification of Intel Core i7 processor using SAT technology (2009)
• Certified Windows 7 device drivers using SMT (2010)
• Flyspeck project: formal proof of the Kepler Conjecture using Isabelle and HOL Light proof assistants (2014)
• ...
Theorem Prover =

Inference System + Search Algorithm

input clauses

$P(X) \lor Q(X)$

search space

$\neg P(a)$

$Q(a)$

1. pick clause

2. find candidates

3. perform inferences
Theorem Prover =

Inference System + Search Algorithm

1. pick clause
2. find candidates
3. perform inferences

input clauses
search space

false

1. pick clause
2. find candidates
3. perform inferences
Theorem Prover =

Inference System + Search Algorithm

[Diagram showing Memory and Time with upward and downward arrows]
Demo (Vampire theorem prover)

• Group theory: if a group satisfies the identity \( x^2 = 1 \), then it is commutative.

\[
\forall x : 1 \cdot x = x \\
\text{Axioms of group theory} \quad \forall x : x^{-1} \cdot x = x \\
\forall x \forall y \forall z : (x \cdot y) \cdot z = x \cdot (y \cdot z) \\
\text{Assumption} \quad \forall x : x \cdot x = 1 \\
\text{Conjecture} \quad \forall x \forall y : x \cdot y = y \cdot x
\]

• Set theory: \( \cup \) is commutative.

\[
\forall A \forall B \forall x : x \in A \cup B \leftrightarrow x \in A \lor x \in B \\
\text{Axioms of set theory} \quad \forall A \forall B : (\forall x : x \in A \leftrightarrow x \in B) \rightarrow A = B \\
\text{Conjecture} \quad \forall A \forall B : A \cup B = B \cup A
\]
Options in Vampire

age_weight_ratio
aig_bdd_sweeping
aig_definition_introduction
aig_definition_introduction_threshold
aig_formula_sharing
aig_inliner
arity_check
backward_demodulation
backward_subsumption
backward_subsumption_resolution
bfnt
binary_resolution
bp_add_collapsing_inequalities
bp_allowed_fm_balance
bp_almost_half_bounding_removal
bp_assignment_selector
bp_bound_improvement_limit
bp_conflict_selector
bp_conservative_assignment_selection
bp_fm_elimination
bp_max_prop_length
bp_propagate_after_conflict
bp_start_with_precise
bp_start_with_rational
bp_variable_selector
color_unblocking
condensation
decode
demodulation_redundancy_check
distinct_processor
epr_preserving_naming
epr_preserving_skolemization
epr_restoring_inlining
equality_propagation
equality_proxy
equality_resolution_with_deletion
extensionality_allow_pos_eq
extensionality_max_length
extensionality_resolution
flatten_top_level_conjunctions
forbidden_options
forced_options
forward_demodulation
forward_literal_rewriting
forward_subsumption
forward_subsumption_resolution
function_definition_elimination
function_number
general_splitting
global_subsumption
horn_revealing
hyper_superposition
ignore_missing
include
increased_numerical_weight
inequality_splitting
input_file
input_syntax
inst_gen_big_restart_ratio
inst_gen_inprocessing
inst_gen_passive_reactivation
inst_gen_resolution_ratio
inst_gen_restart_period
inst_gen_restart_period_quotient
inst_gen_selection
inst_gen_with_resolution
interpreted_simplification
latex_output
lingva_additional_invariants
literal_comparison_mode
log_file
lrs_first_time_check
lrs_weight_limit_only
max_active
max_answers
max_inference_depth
max_passive
max_weight
memory_limit
mode
name_prefix
naming
niceness_option
nonglobal_weight_coefficient
nonliterals_in_clause_weight
normalize
output_axiom_names
predicate_definition_inlining
predicate_definition_merging
predicate_equivalence_discovery
predicate_equivalence_discovery_add_implications
predicate_equivalence_discovery_random_simulation
predicate_equivalence_discovery_sat_conflict_limit
predicate_index_introduction
print_classifier_premses
problem_name
proof
proof_checking
protected_prefix
question_answering
random_seed
row_variable_max_length
sat_clause_activity_decay
sat_clause_disposer
sat learnt minimization
sat learnt_subsumption_resolution
sat lingeling_incremental
sat lingeling_similar_models
sat restart fixed count
sat restart_geometric_increase
sat restart_geometric_init
sat restart_luby_factor
sat restart_minisat_increase
sat restart_minisat_init
sat restart_strategy
sat solver
sat var activity_decay
sat var selector
satisfaction_algorithm
selection
show_active
show_blocked
show_definitions
show_interpolant
show_new
show_new_propositional
show_nonconstant_skolem_function_trace
show_options
show_passive
show_preprocessing
show_skolemisations
show_symbol_elimination
show_theory_axioms
simulated_time_limit
sine_depth
sine generality_threshold
sine_selection
sine_tolerance
smtlib_consider_ints_real
smtlib_flet_as_definition
smtlib_introduce_aig_names
sos
split_at_activation
splitting
ssplitting_add_complementary
ssplitting_component_sweeping
ssplitting_congruence_closure
ssplitting_eager_removal
ssplitting_flush_period
ssplitting_flush_quotient
ssplitting_nonsplittable_components
statistics
superposition_from_variables
symbol_precedence
tabulation_bw_rule_subsumption_resolution_by_lemmas
tabulation_fw_rule_subsumption_resolution_by_lemmas
tabulation_goal_awr
tabulation_goal_lemma_ratio
tabulation_instantiate_producing_rules
tabulation_lemma_awr
test_id
thanks
theory_axioms
time_limit
time_statistics
trivial_predicate_removal
unit_resulting_resolution
unused_predicate_definition_removal
use_dismatching
weight_increment
while_number
xml_output
Current Theorem Proving Practice

• Established core algorithms, but many parameters/options
  – One particular choice of options is called a *strategy*

• Empirical observation: proof is either found very fast or never

• Conjecture: for every proof there is a strategy which finds it fast, but there is no single best strategy

• *Strategy scheduling*
  – Run many strategies with short timeouts (sequential or in parallel)

• How to find a good strategy / a good schedule?
Opportunities for machine learning

<table>
<thead>
<tr>
<th>Strategy Selection</th>
<th>Premise Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best heuristic depends on the form of the problem, however this relationship is not obvious, even to experts.</td>
<td>Proving new theorems from huge set of (often irrelevant) axioms/definitions is infeasible. However, it is a priori not clear which formulas are needed.</td>
</tr>
<tr>
<td>Can we learn it from previous proof attempts?</td>
<td>Can we learn to predict promising subsets of premises?</td>
</tr>
<tr>
<td>Problem: System is not robust; small perturbations can have large effects.</td>
<td></td>
</tr>
</tbody>
</table>
Data Sets

Many existing benchmark libraries, e.g.:

- TPTP (Thousands of Problems for Theorem Provers) library
- Archive of Formal Proofs
- Mizar Mathematical Library
- SATLIB
- SMT-LIB
Interlude: Model Selection

1. Fit model(s) on training set
2. Choose best model / parameters on validation set
3. Report performance on test set
Interlude: Model Selection

\textit{k-Fold Cross Validation for Model Selection}

\textbf{input:}
- training set \( S = (x_1, y_1), \ldots, (x_m, y_m) \)
- set of parameter values \( \Theta \)
- learning algorithm \( A \)
- integer \( k \)

\textbf{partition} \( S \) into \( S_1, S_2, \ldots, S_k \)

\textbf{foreach} \( \theta \in \Theta \)
- \textbf{for} \( i = 1 \ldots k \)
  - \( h_{i,\theta} = A(S \setminus S_i; \theta) \)
  - \( \text{error}(\theta) = \frac{1}{k} \sum_{i=1}^{k} L_{S_i}(h_{i,\theta}) \)

\textbf{output}
- \( \theta^* = \arg\min_{\theta} [\text{error}(\theta)] \)
- \( h_{\theta^*} = A(S; \theta^*) \)
Instance-based learning (IBL)
Suitability-value (SV)

[Fuchs, Automatic Selection Of Search-Guiding Heuristics For Theorem Proving, 1998]
Notation

• $\mathcal{S} = \{s_1, \ldots, s_m\}$ set of strategies
• $\mathcal{P} = \{p_1, \ldots, p_n\}$ set of training/validation/test problems
• $\tau(p, s)$ runtime of strategy $s$ on problem $p$
• $T$ timeout for proof attempts
• $f(p) = (f_1(p), \ldots, f_k(p))$ feature vector of problem $p$
  if convenient, we only write $p$ to denote the feature vector
Setting

- Given a new problem $p$, find a schedule $S$ of strategies (i.e. a permutation $\sigma$) to run in sequence on $p$.
  \[ S = s_{\sigma(1)}, \ldots, s_{\sigma(m)} \]

- Let the $i$th strategy be the first in $S$ that succeeds on $p$
  \[ \tau(p, S) = (i - 1)T + \tau(p, s_{\sigma(i)}) \]

- Reference schedules:
  - $S_{opt}(p)$: best strategy for $p$ at first position
  - $S_{fix}$: fixed schedule such that $\sum_{p \in \mathcal{P}} \tau(p, S_{fix})$ is smallest

- Euclidean distance between two problems $p$ and $p'$
  \[ \delta(p, p') = \sqrt{\sum_{i=1}^{k} (f_i(p) - f_i(p'))^2} \]
IBL-based Approach

• Pair each training problem \( p_i \) with its best strategy \( s_i^* \)
• Let \( d_i = \delta(p, p_i) \) be the distance between \( p \) and every training problem \( p_i \)
• Find \( d_{\pi(1)} \leq \cdots \leq d_{\pi(n)} \)

• Define \( S_{IBL}(p) \) to be \( s_{\pi(1)}^*, \ldots, s_{\pi(n)}^* \) with duplicates removed and remaining strategies appended deterministically
• \( S_{IBL}^*(p) \): break distance ties by position in \( S_{fix} \)
SV-based Approach

- Drawback of IBL: only closest neighbor is taken into account
- Compute suitability value $v_j$ for every strategy $s_j$
  \[ v_j = \sum_{i=1}^{n} \Psi(d_i, \tau(p_i, s_j)) \]

\[ \Psi_e(d, t) = \begin{cases} \frac{\hat{T}}{(d+1)^e}, & t = \infty \\ \frac{t}{(d+1)^e}, & \text{else} \end{cases}; e \in \mathbb{N}, \hat{T} \geq T \]

- Find $v_{\pi(1)} \leq \cdots \leq v_{\pi(m)}$ and set $S_{SV}(p) = s_{\pi(1)}, \ldots, s_{\pi(m)}$
- Penalize timeout by the number of other strategies that also timeout $\rightarrow S_{SV}^*(p)$
  \[ \frac{\hat{T}}{(d + 1)^e} \cdot |\{s_l| \tau(p_i, s_l) = \infty\}| \]
Experimental Evaluation

• Discount theorem prover with $m = 5$ strategies
• 263 equational problems from six TPTP categories
• Timeout $T = 600s$
• Features
  • $f_1$: $|Ax_i|$, i.e., the number of axioms;
  • $f_2$: number of distinct function symbols occurring in axioms;
  • $f_3$: number of distinct constants occurring in axioms;
  • $f_4$: number of distinct unary function symbols occurring in axioms;
  • $f_5$: number of distinct binary function symbols occurring in axioms;
  • $f_6$: number of distinct ternary function symbols occurring in axioms;
  • $f_7$: number of distinct function symbols occurring in $\bar{s}_i \neq \bar{l}_i$ only;
  • $f_8$: number of distinct variables occurring in $\bar{s}_i \neq \bar{l}_i$;
  • $f_9$: total number of symbols occurring in the smallest side of $\bar{s}_i \neq \bar{l}_i$;
  • $f_{10}$: total number of symbols occurring in the biggest side of $\bar{s}_i \neq \bar{l}_i$;

• Parameters $e = 4$ and $\hat{T} = 1000s$
Experimental Evaluation

training data = test data

<table>
<thead>
<tr>
<th>$P$</th>
<th>#</th>
<th>$t_{opt}^P$</th>
<th>$t_{fix}^P$</th>
<th>$t_{PBL}^P$</th>
<th>$t_{PBL}^*$</th>
<th>$t_{SV}^P$</th>
<th>$t_{SV}^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOO</td>
<td>30</td>
<td>465s</td>
<td>1494s</td>
<td>753.79s</td>
<td>1066s</td>
<td>469s</td>
<td>470s</td>
</tr>
<tr>
<td>COL</td>
<td>70</td>
<td>1270s</td>
<td>10936s</td>
<td>6152.65s</td>
<td>5098s</td>
<td>8212s</td>
<td>4189s</td>
</tr>
<tr>
<td>GRP</td>
<td>108</td>
<td>1597s</td>
<td>13802s</td>
<td>1634.92s</td>
<td>1612s</td>
<td>1621s</td>
<td>1625s</td>
</tr>
<tr>
<td>LCL</td>
<td>24</td>
<td>561s</td>
<td>1775s</td>
<td>562.85s</td>
<td>564s</td>
<td>564s</td>
<td>564s</td>
</tr>
<tr>
<td>RNG</td>
<td>22</td>
<td>165s</td>
<td>2136s</td>
<td>165s</td>
<td>165s</td>
<td>166s</td>
<td>227s</td>
</tr>
<tr>
<td>ROB</td>
<td>9</td>
<td>151s</td>
<td>200s</td>
<td>469.60s</td>
<td>151s</td>
<td>151s</td>
<td>152s</td>
</tr>
<tr>
<td>all</td>
<td>263</td>
<td>4209s</td>
<td>42678s</td>
<td>9802.46s</td>
<td>8377s</td>
<td>11224s</td>
<td>7275s</td>
</tr>
</tbody>
</table>

Robustness ratio

\[ r_X = \frac{t_{Boo}^X + t_{Col}^X + t_{Grp}^X + t_{Lcl}^X + t_{Rng}^X + t_{Rob}^X}{t_{all}^X} \]

\[ r_{fix} \approx 0.71 \quad r_X > 0.99 \text{ for } X \neq fix \]
Experimental Evaluation

90% training data

<table>
<thead>
<tr>
<th></th>
<th>worst case</th>
<th></th>
<th>average case</th>
<th></th>
<th>best case</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$t_f^P$</td>
<td>$t_{IBL}^P$</td>
<td>$t_{IBL}^*$</td>
<td>$t_{SV}^P$</td>
<td>$t_{SV}^*$</td>
<td>$t_f^P$</td>
</tr>
<tr>
<td>BOO</td>
<td>5918</td>
<td>2277</td>
<td>1670</td>
<td>4665</td>
<td>4069</td>
<td>1593</td>
</tr>
<tr>
<td>COL</td>
<td>12803</td>
<td>13772</td>
<td>11983</td>
<td>13890</td>
<td>10814</td>
<td>11134</td>
</tr>
<tr>
<td>GRP</td>
<td>13802</td>
<td>7630</td>
<td>6412</td>
<td>9421</td>
<td>8825</td>
<td>13802</td>
</tr>
<tr>
<td>LCL</td>
<td>2975</td>
<td>2364</td>
<td>1768</td>
<td>4761</td>
<td>4761</td>
<td>1787</td>
</tr>
<tr>
<td>RNG</td>
<td>3409</td>
<td>2565</td>
<td>1968</td>
<td>1529</td>
<td>6374</td>
<td>2206</td>
</tr>
<tr>
<td>ROB</td>
<td>1989</td>
<td>799</td>
<td>751</td>
<td>199</td>
<td>200</td>
<td>416</td>
</tr>
<tr>
<td>all</td>
<td>43997</td>
<td>19913</td>
<td>20231</td>
<td>20228</td>
<td>19114</td>
<td>42894</td>
</tr>
</tbody>
</table>
Support Vector Machines

[Bridge, Machine Learning and Automated Theorem Proving, 2010]

[Bridge et al., Machine Learning for First-Order Theorem Proving, 2014]
Setting

• 6118 sample problems $\mathcal{P}$ from the TPTP library
• Theorem prover E with five preselected strategies $\mathcal{S}$

• From runtimes $\tau(p, s)$ (100s timeout) build data set for every strategy $s_i$

$$\mathcal{D}_i = \{ (f(p_1), y_1^{(i)}), \ldots, (f(p_n), y_n^{(i)}) \}$$

$$y_j^{(i)} = +1/−1, \text{ depending on whether}$$

$s_i$ was fastest strategy on $p_j$ or not

• Special strategy $s_0$ and data set $\mathcal{D}_0$ denoting rejection of “too hard” problems

• Goal: given problem $p$, predict best strategy to find proof within 100s
Features

14 static features

Fraction of clauses that are unit clauses.
Fraction of clauses that are Horn clauses.
Fraction of clauses that are ground Clauses.
Fraction of clauses that are demodulators.
Fraction of clauses that are rewrite rules (oriented demodulators).
Fraction of clauses that are purely positive.
Fraction of clauses that are purely negative.
Fraction of clauses that are mixed positive and negative.
Maximum clause length.
Average clause length.
Maximum clause depth.
Average clause depth.
Maximum clause weight.
Average clause weight.

39 dynamic features

Proportion of generated clauses kept. (Subsumed or trivial clauses are discarded.)
Sharing factor. (A measure of the number of shared terms.)
$|P|/|P \cup U|$
$|U|/|A|$
Ratio of longest clause lengths in $P$ and $A$.
Ratio of average clause lengths in $P$ and $A$.
Ratio of longest clause lengths in $U$ and $A$.
Ratio of average clause lengths in $U$ and $A$.
Ratio of maximum clause depths in $P$ and $A$.
Ratio of average clause depths in $P$ and $A$.
Ratio of maximum clause depths in $U$ and $A$.
Ratio of average clause depths in $U$ and $A$.
Ratio of maximum clause standard weights in $P$ and $A$.
Ratio of average clause standard weights in $P$ and $A$.
Ratio of maximum clause standard weights in $U$ and $A$.
Ratio of average clause standard weights in $U$ and $A$.
Ratio of the number of trivial clauses to $|P|$.  
.....
Performance Measures

For a validation/test set of size $m$, define

$$P^+ / P^- \quad \# \text{true/false positives}$$

$$N^+ / N^- \quad \# \text{true/false negatives}$$

- **Accuracy**: $acc = \frac{P^+ + N^+}{m}$

- **Matthews correlation coefficient**: $M = \frac{P^+ N^- - P^- N^+}{\sqrt{(P^+ + P^-)(P^+ + N^-)(N^+ + P^-)(N^+ + N^-)}}$
  
  1...perfect prediction  
  0...like random classifier  
  -1...opposite to data

- **F1 score**: $F1 = \frac{2pr}{p+r} \in [0,1]$
  
  precision $p = \frac{P^+}{P^+ + P^-}$  
  recall $r = \frac{P^+}{P^+ + N^-}$
Support Vector Machine in Kernelized Form

• Classifier for sample $\mathbf{x}$ is given by
  \[ \text{sign} \left[ \sum_i \alpha_i k(\mathbf{x}^i, \mathbf{x}) + b \right] \]

• Regularization parameter $C$ controls trade-off between robustness and correctness
  – can be split into $C^+$ and $C^-$ to give different weights for positive and negative samples $\rightarrow$ ratio parameter $j$
  \[ j = \frac{C^+}{C^-} = \frac{\# \text{ positive samples}}{\# \text{ negative samples}} \]

• Kernels

<table>
<thead>
<tr>
<th>Linear</th>
<th>Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{x}^T \mathbf{x}'$</td>
<td>$(s \mathbf{x}^T \mathbf{x}' + c)^d$</td>
</tr>
<tr>
<td>Sigmoid tanh</td>
<td>Radial Basis</td>
</tr>
<tr>
<td>$\tanh (s \mathbf{x}^T \mathbf{x}' + c)$</td>
<td>$\exp (-\gamma |\mathbf{x} - \mathbf{x}'|^2$)</td>
</tr>
</tbody>
</table>
## Classifier performance

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>All features</th>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc</td>
<td>F1</td>
<td>Matt</td>
</tr>
<tr>
<td>0</td>
<td>0.81</td>
<td>0.77</td>
<td>0.61</td>
</tr>
<tr>
<td>1</td>
<td>0.76</td>
<td>0.48</td>
<td>0.35</td>
</tr>
<tr>
<td>2</td>
<td>0.92</td>
<td>0.30</td>
<td>0.23</td>
</tr>
<tr>
<td>3</td>
<td>0.86</td>
<td>0.47</td>
<td>0.39</td>
</tr>
<tr>
<td>4</td>
<td>0.89</td>
<td>0.40</td>
<td>0.32</td>
</tr>
<tr>
<td>5</td>
<td>0.88</td>
<td>0.39</td>
<td>0.32</td>
</tr>
</tbody>
</table>

- Only small differences between used feature sets
- Slight degradation when using dynamic features
Combining Classifiers for Heuristic Selection

Interpret value of classifier (before applying sign) as confidence in labeling input +1

→ Select strategy with largest value

<table>
<thead>
<tr>
<th></th>
<th>No H0</th>
<th>With H0</th>
<th>H0, positive margin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number</td>
<td>Time</td>
<td>Number</td>
</tr>
<tr>
<td>All</td>
<td>827/827</td>
<td>73,549/73,549</td>
<td>700/709</td>
</tr>
<tr>
<td>Static</td>
<td>833/822</td>
<td>72,845/74,350</td>
<td>726/718</td>
</tr>
<tr>
<td>Dynamic</td>
<td>810/809</td>
<td>75,268/75,286</td>
<td>667/666</td>
</tr>
</tbody>
</table>

• Learned strategies outperform every fixed strategy
• Performance with dynamic features is worse
• Allowing the rejection of problems reduces runtime drastically
Feature selection

- Usually applied to reduce huge feature sets to make learning feasible
- Here: provide useful information for strategy development

- Techniques:
  - Filter approach
  - Embedded approach
  - Wrapper approach

- Exhaustive test of all feature subsets up to size 3 → similar results can be obtained with only a few features
MaLeS - Machine Learning of Strategies

[Kühlwein et al., MaLeS: A Framework for Automatic Tuning of Automated Theorem Provers, 2013]
[Kühlwein, Machine Learning for Automated Reasoning, 2014]
MaLeS

• Parameter tuning framework
  – Automatically find set $S$ of preselected strategies
  – Construct individual strategy schedule for new problems

• Learned object: *runtime prediction functions*
  \[ \rho_S : \mathcal{P} \to \mathbb{R} \]
  – Kernel method
  – Only learn on problems that can be solved within timeout
  – Update of prediction function during runtime
Finding good search strategies

Stochastic local search algorithm

```plaintext
1: procedure FIND_STRATEGIES(Problems, tol, t_max, nS, nC)
2:     initialize Queue Q
3:     initialize dictionary bestTimes with t_max for all problems
4:     while Q not empty do
5:         s ← pop(Q)
6:         for p ∈ Problems do
7:             oldBestTime ← bestTime[p]
8:             proofFound, timeNeeded ← run_strategy(s, p, t_max)
9:             if proofFound and timeNeeded < bestTime[p] then
10:                bestTime[p] ← timeNeeded
11:                bestStrategies[p] ← s
12:         end if
13:         if proofFound and timeNeeded < bestTime[p] + tol then
14:             randomStrategies ← create_random_strategies(s, nS, nC)
15:             for r in randomStrategies do
16:                 proofFoundR, timeNeededR ← run_strategy(r, p, timeNeeded)
17:                 if proofFoundR and timeNeededR < bestTime[p] then
19:                     bestStrategies[p] ← r
20:                 end if
21:             end for
22:             if bestTime[p] < oldBestTime then
23:                 Q ← put(Q, bestStrategies[p])
24:             end if
25:         end if
26:     end while
27:     end procedure
```

E-MaLeS strategy search:
- 1112 problems
- Three weeks on 64 core server
- 109 strategies selected, out of 2 million
### Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>axioms</td>
<td>Most specific class (unit, Horn, general) describing all axioms</td>
</tr>
<tr>
<td>goals</td>
<td>Most specific class (unit, Horn) describing all goals</td>
</tr>
<tr>
<td>equality</td>
<td>Problem has no equational literals, some equational literals, or only equational literals</td>
</tr>
<tr>
<td>non-ground units</td>
<td>Number (or fraction) of unit axioms that are not ground</td>
</tr>
<tr>
<td>ground-goals</td>
<td>Are all goals ground?</td>
</tr>
<tr>
<td>clauses</td>
<td>Number of clauses</td>
</tr>
<tr>
<td>literals</td>
<td>Number of literals</td>
</tr>
<tr>
<td>term_cells</td>
<td>Number of all (sub)terms</td>
</tr>
<tr>
<td>unit-goals</td>
<td>Number of unit goals (negative clauses)</td>
</tr>
<tr>
<td>unit-axioms</td>
<td>Number of positive unit clauses</td>
</tr>
<tr>
<td>horn-goals</td>
<td>Number of Horn goals (non-unit)</td>
</tr>
<tr>
<td>horn-axioms</td>
<td>Number of Horn axioms (non-unit)</td>
</tr>
<tr>
<td>eq-classes</td>
<td>Number of unit equations</td>
</tr>
<tr>
<td>ground-unit-axioms</td>
<td>Number of ground unit axioms</td>
</tr>
<tr>
<td>ground-goals</td>
<td>Number of ground goals</td>
</tr>
<tr>
<td>ground-positive-axioms</td>
<td>Number (or fraction) of positive axioms that are ground</td>
</tr>
<tr>
<td>positive-axioms</td>
<td>Number of all positive axioms</td>
</tr>
<tr>
<td>ng-unit-axioms-part</td>
<td>Number of non-ground unit axioms</td>
</tr>
<tr>
<td>max_fun arity</td>
<td>Maximal arity of a function or predicate symbol</td>
</tr>
<tr>
<td>avg_fun arity</td>
<td>Average arity of symbols in the problem</td>
</tr>
<tr>
<td>sum_fun arity</td>
<td>Sum of arities of symbols in the problem</td>
</tr>
<tr>
<td>clause_max_depth</td>
<td>Maximal clause depth</td>
</tr>
<tr>
<td>clause_avg_depth</td>
<td>Average clause depth</td>
</tr>
</tbody>
</table>

Again absolute values, instead of ratios.
Learning the prediction function

• Prediction function has the form
  \[ \rho_s(p) = \sum_{p' \in \mathcal{P}} \alpha_{p'}^s k(p, p') \text{ for some } \alpha_{p'}^s \in \mathbb{R} \]

• Kernel matrix \( K^s \in \mathbb{R}^{m \times m} \) is \( K_{i,j}^s = k(p_i, p_j) \)

• Time matrix \( Y^s \in \mathbb{R}^{1 \times m} \) is \( Y_i^s = \tau(p_i, s) \)

• Weight matrix \( A^s \in \mathbb{R}^{m \times 1} \) is \( A_i^s = \alpha_{p_i}^s \)

• Least square regression
  \[ A = \arg\min_{A \in \mathbb{R}^{m \times 1}} ((Y - KA)^T (T - KA) + CA^T KA) \]

• Theorem
  \[ A = (K + CI)^{-1} Y \]
Creating Schedules from Prediction Functions

1: procedure MALES(problem, time)
2:  proofFound, timeUsed ← run_start_strategies(problem, time)
3:  if proofFound then
4:    return timeUsed
5:  end if
6:  while timeUsed < time do
7:    times is an empty list
8:    for s ∈ S do
9:      t_s ← ρ_s(problem)
10:     times.append([t_s, s])
11:   end for
12:   ([t_{s'}, s']) ← choose_best_strategy(times)
13:   proofFound, timeNeeded ← run_strategy(s', problem, t_{s'})
14:   timeUsed += timeNeeded
15:  if proofFound then
16:    return timeUsed
17:  end if
18:  for s ∈ S do
19:    timeUsed += update_prediction_function(ρ_s, s', t_{s'})
20:  end for
21:  end while
22: return timeUsed
23: end procedure

- Delete training problems that are solved by the picked strategy within the predicted runtime
  \[ P_{train}^s := \{ p \in P_{train}^s \mid \tau(p, s') > t_{s'} \} \]
- Retrain prediction function
Evaluation
Conclusion

• Existing case studies showed applicability of machine learning techniques to strategy selection in theorem provers
  – Nearest neighbor
  – Similarity value
  – Support vector machines

• Requires many decisions $\rightarrow$ different settings
• Results often inconclusive

• Not yet a direct benefit for developers or users