LEARNING TO RANK FOR SYNTHESIZING PLANNING HEURISTICS

Caelan Reed Garrett, Leslie Pack Kaelbling, Tomás Lozano-Pérez
MIT CSAIL - {caelan, lpk, tlp}@csail.mit.edu
Learning a Heuristic Function

- Deterministic satisficing classical planning
  - Distribution of problems \( \{\Pi^1, \ldots, \Pi^n\} \)

- Focus on improving \textit{coverage} and runtime
  - Greedy best-first search

- Learn domain-specific heuristic function
  - But learn it in a domain-independent way

- Use statistical supervised learning
Supervised Heuristic Learning

- Assume linear function $h(x) = \phi(x)^T w$
- Gather training examples $\langle x_i^j, y_i^j \rangle$
- Specify feature representation $\phi(x)$
- Choice of loss function
- Optimize $w$ using the loss function
Gathering Training Examples

- Generate training problems (~10 problems)
- Solve and use states on plan as examples \( \langle x^i_j, y^i_j \rangle \)
  - Inputs are state and problem \( x^i_j = \langle s^i_j, \Pi^i \rangle \)
  - Outputs are costs-to-go \( y^i_j \)

\[
\begin{align*}
\Pi_1 & : s^1_0 \rightarrow s^1_* \\
\Pi_2 & : s^2_0 \rightarrow s^2_1 \rightarrow s^2_* \\
\{ & \langle x^1_0, 1 \rangle, \langle x^1_*, 0 \rangle, \langle x^2_0, 2 \rangle, \langle x^2_1, 1 \rangle, \langle x^2_*, 0 \rangle \} \\
\end{align*}
\]

- Use large-timeout timeout portfolio to produce plans
Feature Representation

- Obtain generic features from existing heuristics
  - Derive from approximate partially-ordered plans
  - FastForward (FF), Context-Enhanced Add (CEA), …

- New method for features
  - Counting *single* instance per action type
  - Counting *pairwise* instances per action types and interactions
Heuristic in a Greedy Search

- A heuristic is typically an estimate of the cost-to-go.
- A heuristic in greedy search sorts the explored state space by its estimate of cost-to-go.
- Only the ordering is important, not the numerical heuristic value.
- Ideal heuristic is any monotonically increasing function of true cost-to-go.

\[ y_j^i > y_k^i \iff h(x_j^i) > h(x_k^i) \quad \forall i, j, k \]
Measuring Ordering

- Want a statistic that counts the number of incorrect orderings
- Kendal Rank Correlation Coefficient ($\tau$)
  - Measure of monotonic correlation $\tau \in [-1, 1]$
  - Normalized number of correctly ranked pairs
- Only rank examples from the same problem instance
- Larger $\tau$ is generally better in our application
Positive Kendall Rank Correlation

\[ h(x) \]

\[ \tau = 1 \]
Zero Kendall Rank Correlation

\[ \tau \approx 0 \]
Negative Kendall Rank Correlation

\[ h(x) \]

\[ \tau = -1 \]
Optimizing the Ranking Loss Function

- Rank Support Vector Machine (RSVM)
- Optimize hinge loss relaxation of $\tau$
- Equivalent to SVM classifying ranking pairs

\[
\min_w \|w\|^2 + C \sum_{i=1}^{n} \sum_{j=1}^{m_i} \sum_{k=j+1}^{m_i} \xi_{ijk} \\
\text{s.t.} \quad (\phi(x^i_j) - \phi(x^i_k))^T w \geq 1 - \xi_{ijk}, \quad \forall y^i_j \geq y^i_k, \quad \forall i \\
\xi_{ijk} \geq 0, \quad \forall i, j, k
\]
Why Not Use Regression?

- Prior works optimize Root Mean Squared Error (RMSE) using Ordinary Least Squares Regression

- Can compromise ordering to obtain better average numerical error

- May sometimes create/widen local minima

- Particularly problematic with training noise and imperfect feature representation
1D - 2 problems with 2 states each
Ridge Regression Solution

\[ y := \nabla^2 x_1 a x_1 b x_2 b x_2 a h_{RR}(x) \]
Testing Problems Coverage

- IPC 2014 Learning Track generators - elevators, transport, parking, no-mystery
- Harder than instances used in competition

![Bar chart showing solved problems with different features and methods]
- Heuristic learning alone not effective on domains with harmful dead-ends
Eager best-first search does better but the improvement is not significant.
Comparison on IPC Instances

- RSVM lazy best-first search
  - elevators (5/5), transport (5/5), parking (5/5), no-mystery (1/5)

- RSVM eager best-first search
  - no-mystery (5/5)

- Combined coverage (20/20)
- Competition winner using portfolio (17/20)
Takeaways

- **Only ordering matters** for greedy search heuristics

- **Learning with a ranking loss function** improves performance over raw heuristic or regression heuristic

- **Pairwise features** able to encode information about approximate plan without feature extraction

- Learning most effective on large domains without harmful dead-ends
Selected References


