Automatic Induction of Bellman-Error Features for Probabilistic Planning

Jia-Hong Wu and Robert Givan

2011.10.13

許博竣
Introduction

• Propose automatic processes for learning useful domain-specific feature sets with little or no human intervention

• Select and add features that describe state-space regions of high inconsistency in the Bellman equation during approximate value iteration
**Markov Decision Process**

- **MDP: (1950s)**
  - Provides a mathematical framework for modeling decision-making in situations where outcomes are partly random and partly under the control of a decision maker.
  - The agent has the objective of accumulating as much reward as possible.
  - Solutions: **state-value functions** assigned real numbers to states.
A Markov decision process is a 4-tuple $(S, A, P, R)$

- $S$ - a finite set of states
- $A$ - a finite set of actions (alternatively, $A_s$ is the finite set of actions available from state $s$)
- $P_a(s, s')$ - the probability that action $a$ in state $s$ will lead to state $s'$
- $R_a(s, s')$ - the expected immediate reward received after transition to state $s'$ from state $s$ with action $a$
With the state transition function $P$ and the reward function $R$, we wish to calculate the policy that maximizes the expected discounted reward. The standard family of algorithms to calculate this optimal policy requires storage for two arrays indexed by state: value $V$, which contains real values, and policy $\pi$ which contains actions. At the end of the algorithm, $\pi$ will contain the solution and $V(s)$ will contain the discounted sum of the rewards to be earned (on average) by following that solution from state $s$.

$$
\pi(s) := \arg \max_a \left\{ \sum_{s'} P_{a}(s, s') \left( R_{a}(s, s') + \gamma V(s') \right) \right\}
$$

$$
V(s) := \sum_{s'} P_{\pi(s)}(s, s') \left( R_{\pi(s)}(s, s') + \gamma V(s') \right)
$$

- $\gamma$: discount factor (typically close to 1)
Markov Decision Process

• **Value iteration:**
  - Substituting the calculation of $\pi(s)$ into the calculation of $V(s)$ gives the combined step:

  $$V(s) := \max_a \left\{ \sum_{s'} P_a(s, s')(R_a(s, s') + \gamma V(s')) \right\}.$$  

  - This update rule is iterated for all states $s$ until it converges with the left-hand side equal to the right-hand side (which is the **Bellman equation** for this problem).

• **Bellman Error:** The degree to which a given value function fails to respect action transitions
  - $B(V, s) = \text{update}(V)(s) - V(s)$

• Inducing new features on their correlation to the state-wise Bellman error
Feature Spaces

• Features: selecting a real value for each state

• Representing value functions using a linear combination of \( l \) features extracted from \( s \)

\[
\tilde{V}(s) = \sum_{i=0}^{l} w_i f_i(s)
\]

• Finding features \( f_i \) and weights \( w_i \) such that \( \tilde{V} \) closely approximates \( V^* \)
Feature Spaces

• Relational
  • Feature value is determined by the relations between objects in the domain.
  • The “number of holes” in Tetris: the number of empty squares that have some other filled squares above them
  • The ability to generalize features to large problems

• Propositional
  • An attribute-value representation
  • Can be reformulated as a related classification problem
Feature Spaces

• In relational MDPs
  • A state fact is an application $p(o_1, ..., o_n)$ of $n$-argument state predicate $p$ to object arguments $o_i$. A state is any set of state facts, representing the true facts in that state.
  • An action instance is an application $p(o_1, ..., o_n)$. The action space is the set of all action instance.

• In propositional MDPs
  • The state space is specified by providing a finite sequence of basic state properties call state attributes. A state is any vector of values for state attributes.
  • The action space is explicitly specified.
Feature Spaces

• In relational MDPs
  • A feature: a formula with one free variable, mapping each state to the number of objects in that state that satisfies the formula
  • Example:
    • $\text{on}(x, y)$ is a predicate in the Blocksworld domain that assert the block $x$ is on the top of $y$ (a block or the table). A possible feature can be described as $\exists y \text{ on}(x, y)$, which is a first-order formula with $x$ as the free variable. For $n$ block problems, the value (un-normalized) of this feature is $n$ for states with no block being held by the arm or $n-1$ for states with a block being held by the arm.
Feature Spaces

• In propositional MDPs
  • A decision tree with real number labels at the leaves is viewed as labeling all the states with real numbers, and it can be viewed as a feature.
  • A supervised classification algorithm such as C4.5 can be used.
Control Flow for Feature Learning

Initial feature vector $\tilde{\Phi}$
Initial weight vector $\tilde{w}$
Initial problem difficulty $D$

Yes

Difficulty at target level or out of time?

Final $\tilde{\Phi}$ and $\tilde{w}$
No

Increase problem difficulty $D$. Keep $\tilde{w}$ and $\tilde{\Phi}$.

Select $\tilde{w}$ approximately minimizing Bellman error of $V = \tilde{w} \cdot \tilde{\Phi}$

Yes

Performance at current difficulty meets threshold?

Rewighted value function $V = \tilde{w} \cdot \tilde{\Phi}$

Learning new feature correlating to the Bellman error for states in the training set, and add it to $\tilde{\Phi}$. Keep the current problem difficulty $D$.

Done

Generate feature training set
Learning Relational Features

• Using a beam search which selects first-order formulas as candidate features and derives new candidate features from the best scoring features.

• After new candidates have been added a fixed depth of times, the best scoring feature is selected to be added to the value-function representation.

• Scoring each candidate feature with its correlation coefficient to the Bellman error feature $f_{BE}$ as estimated by the training set.

• Bellman Error:
  - $B(V, s) = \text{update}(V)(s) - V(s)$
Learning Propositional Features

- Using only the sign of the “Bellman error feature”
- If we can identify a collection of “undervalued” states as a new feature, then assigning an appropriate positive weight to the feature will increase their value.

Classification learning
  - Generalizing the notions of overvalued and undervalued across the state space from the training sets of sample states.
Example: Tetris

• Relational
  • Rows and columns as objects
  • Three primitive predicates:
    • \texttt{fill}(c, r), \texttt{below}(r_1, r_2), \texttt{beside}(c_1, c_2)
  • Compared: Using human-specified Tetris-specific functions such as “number of holes”

• Propositional
  • 7 binary attributes representing which piece is currently being dropped, with one additional binary attributes for each grid square representing whether it is occupied
Features learned at one domain size are not generally meaningful at other domain sizes.
  • The propositional feature space varies in size as the number of objects in a relational domain is varied.

The relational approach is able to generalize naturally between different domains sizes.

We can use highly efficient, off-the-shelf classification algorithms to employ a propositional approach.
Conclusion

- Using state-wise Bellman error as criterion to select features is essential to the success of the feature learning approach.
- The relational approach produces superior results than the propositional approach.
References
