Learning Object Affordances from Experience: A novel approach to planning under object uncertainty

D. Abel G. Barth-Maron

October 24, 2013

1 Significance

Robots across all domains will likely encounter novel objects and environments. Therefore, a planning algorithm should involve learning about these novel objects to the robots advantage (specifically, ways in which the robot can interact with and possibly employ such objects to accomplish a particular goal, e.g. encountering new types of door handles). In this project, we will design a formal model that learns object affordances from experience. More specifically, we will present a modification to the standard MDP (Markov Decision Process) model in which the transition function is not given to the agent - it will instead be learned from experience (as affordances). We will evaluate our system by comparing its performance on a series of 10 tasks within the Minecraft game world to a standard MDP (that is given a hand crafted transition function).

2 Related Work

- "Learning High Level Planning from Text" S.R.K Branavan et al. (2012)
- "Reinforcement Learning for Mapping Instructions to Actions" S.R.K Branavan et al. (2009)
- "Planning and acting in partially observable stochastic domains" Kalblin, Littman, Cassandra (1998).
- "A Joint Model of Language and Perception for Grounded Attribute Learning" Matuszek, FitzGerald, Zettlemoyer, Bo, Fox (2012)
- "Theory of Affordances" Gibson (1977)
- "Formalizing Affordance" Steedman (2002)
- Further reading: Naive Bayes model, 'Options' paper, Epsilon Greedy and N-armed Bandits, KWIK learning, PAC learning, Model Based RL, Skill Acquisition.

3 Methodology

3.1 Model

The agent will be given the standard components of an MDP, but without T, represented as a Tuple A, S, R:

- A : set of actions $a_i = (\{\texttt{move} \mid \texttt{useItem} \mid \texttt{placeBlock}\}, (x_0, y_0, z_0), (x_a, y_a, z_a))$ where (x_0, y_0, z_0) describe the location of the agent and (x_a, y_a, z_a) indicates the coordinates of the action.
- S : set of states $s_j = \{B_c(x, y, z) \mid \forall (x, y, z) \in \mathbb{M}^3\}$ where $\mathbb{M}^3$ is the coordinate space of the Minecraft world and $B_c(x, y, z)$ describes the contents at coordinate (x, y, z).
- $R : S \times A \rightarrow r$ a reward function where $r \in \mathbb{R}$.

The transition function, T, classically included in MDPs, is the component of the model that we are learning:

$$T = p(s' \mid s, a) \tag{1}$$

We will represent each state $s_j \in S$ as a feature set consisting of five features[1]:
$\Phi(x, y, z)$

- $B_c(x, y, z)$: 6 possible values for now {Dirt, Stone, Water, Air, Gate, Bot}
- placeable((x,y,z)) : Boolean. Indicates if a block is placeable at cell (x, y, z)
- standable((x,y,z)) : Boolean. Indicates if the Bot may stand at cell (x, y, z)
- Ywalkable((x,y,z)) : Boolean. Indicates if the Bot may traverse the cell at (x, y, z) in the Y direction
- Xwalkable((x,y,z)) : Boolean. Indicates if the Bot may traverse the cell at (x, y, z) in the X direction

We use a naive Bayes model, because the independence assumption between features is reasonable within the Minecraft domain. With our feature representation of states, our model appears as follows:

$$P(s_j) = \prod_{\forall (x,y,z) \in \mathbb{M}^3} \prod_{\phi \in \Phi} Pr(\phi(x, y, z)) \tag{2}$$

is reasonable within the Minecraft domain. Thus, our goal is to find the $s' \in S$ that is most likely, given the current $s \in S$ and action $a \in A$. This is equivalent to trying to predict how the agent's action will effect the environment. We estimate T by finding the MAP estimate of T as follows:

$$\hat{T} = \arg\max_{s'} [p(s' \mid s, a)] \tag{3}$$

Using $p(s' \mid s, a) \propto p(s')p(s, a|s')$ we can rewrite the previous equation as the maximum over a prior and likelihood.

$$\arg\max_{s'} [p(s' \mid s, a)] = \arg\max_{s'} [p(s')p(s, a \mid \ s')] \tag{4}$$

We now need express both the prior distribution $p(s')$ and the likelihood $p(s, a|\ s')$ in more tractable forms. Through the use of our independence assumption as expressed in (1) we can rewrite the likelihood distribution:

$$\begin{aligned}
p(s, a|\ s') &= p(s, a|\ B'_c(0, 0, 0), B'_c(1, 0, 0), ..., B'_c(i, j, k)) && (5)\\
&= \prod_{\forall (x,y,z) \in \mathbb{M}^3} p(B_c(x, y, z), a|\ B'_c(0, 0, 0), B'_c(1, 0, 0), ..., B'_c(i, j, k)) && (6)\\
&= \prod_{\forall (x,y,z) \in \mathbb{M}^3} p(B_c(x, y, z), a|B'_c(x, y, z)) && (7)\\
&= p(B_c(x_a, y_a, z_a), a|\ B'_c(x_a, y_a, z_a), cube) && (8)
\end{aligned}$$

The last equation follows from the fact that our action a only affects one coordinate (x_a, y_a, z_a) in the Minecraft world.

Finally, we can restate the prior distribution $p(s') = \sum_{s \in S, a \in A} p(s'|s, a)$. Putting all the pieces together, the MAP estimate of T is:

$$\hat{T} = \arg\max_{s'} \left[p(B_c(x_o, y_o, z_o), a|\ B'_c(x_o, y_o, z_o)) \sum_{s \in S, a \in A} p(s'|s, a) \right] \tag{9}$$

[1]Note: this set of features will probably change over time, but will resemble something like the list included above

3.2 Policy Selection

Our model will learn the best policy by employing Value Iteration with an infinite horizon and discount factor γ.

Our goal now is to update our knowledge of the transition function T in a manner such that as our agent performs actions it gains knowledge of Affordances. An Affordance is extremely similar to our model's transition function, and we will be thinking of the two as somewhat interchangeable.

We use value iteration to determine an optimal policy based on the standard MDP framework. When calculating the reward function our agent will also take into consideration the value of the additional information gain by taking action a in state s. The exact details of this reward function are still being worked out, and is a high priority item. Our current formulation for R and IG appear as follows, though these will require some fleshing out:

$$R(s,a) = \mathbb{E}\left[IG(s,a)\right] + R_\alpha(s,a)]$$

$$\mathbb{E}\left[IG(s,a)\right] \propto \mathbb{E}\left[\Delta T\right]$$

Where $IG(s,a)$ is the information gained from taking action a in state s.

This formulation of the reward function R incorporates exploration (in terms of the expectation of gained knowledge per action) as part of planning. The details of the information gain when acting according to a in state s are still being finalized, but will rest heavily on our theory of affordances. This will involve using similarity metrics on objects in the environment to infer possible affordances based on previous experience (i.e. we can predict the affordances of an object we haven't used before).

3.3 Evaluation Metric

We will evaluate our system by running a baseline MDP planner on each task (10 total tasks), collecting data on 10 (novice) human player try to solve each task[2], and run our own system, the MDP-T. Our specific success metric will be determined by the amount of time taken to solve the task. In the Minecraft world, time taken to solve a task effectively corresponds to the optimality of the policy chosen to solve the task. More formally:

```
INPUT: A command-reward function pair.
OUTPUT: Behavior that attempts to solve the goal described in the input (i.e. a policy)
```

Below we have listed the 10 tasks that will be used for the purpose of evaluation:

1. Place X blocks of type T in the world
2. Destroy X blocks in the world
3. Destroy X blocks of type T in the world
4. Go from point A to point B (with no obstacles - just walking)
5. Go from point A to point B with obstacles (requires more advanced pathfinding)
6. Go from point A to point B with obstacles that requires destroying blocks in the world
7. Go from point A to point B with obstacles that requires placing blocks in the world (e.g. a staircase)
8. Go from point A to point B with obstacles that requires destroying blocks and placing blocks in the world
9. Go from point A to point B while interacting with a simple novel object (e.g. a door)
10. Go from point A to point B while interacting with novel objects (e.g. an elevator, a bridge, a door)

[2]Will we monitor each human task to determine time to completion

4 Schedule

Complete?	Date	Goal
	Oct 13	Finalize model of R/IG/Affordances in MDP-T for proposal 2.0
	Oct 13	Read more related work for proposal 2.0
	Oct 13	Represent state space in terms of *features*
	Oct 13	Determine if block health is possible in Minecraft API
	Oct 17	Implement Value Iteration in a grid world (Russell and Norvig example)
	Oct 21	Implement MDP-T Value Iteration in a grid world
	Oct 27	Represent Minecraft World via state space. Code up high level actions
	Nov 1	Implement Value Iteration in Minecraft (path planning)
	Nov 3	First iteration of MDP-T implemented in Minecraft for project update presentation
	Nov 5	Start data collection with humans. Start writing the paper.
	Nov 20	Final MPD-T bot implemented in Minecraft
	Nov 24	Human data collection finished
	Nov 30	Evaluation completed.
	Dec 2	Identify future work and current problems
	Dec 5	Final project presentation. Paper done.
	Dec 15	Incorporate new augmented version of affordances into MDP-T
	Dec 15 - Jan 6	"Dead in the water period"
	Jan 12	Additional Evaluations/Metrics, Bug Fixes, Versions, New tasks, etc.