Using Self-Organizing Distinctive State Abstraction to Navigate a Maze World

Connie Li
Phil Katz

May 14, 2006

Abstract

This paper presents a developmental robotics experiment implementing self-organizing distinctive state abstraction and growing neural gas. The robot brain takes sensory input of eight sonars, a light sensor, and a stall sensor and has motor outputs of translation and rotation. The robot is placed in a maze world with a goal signified by a light source. We test whether the robot can learn to traverse the maze from increasingly distant starting points, using reinforcement learning. Although the robot does not learn to traverse the maze, it seems likely that an adaptation of our algorithm could complete this task.

1 Introduction

One of the primary goals in the field of developmental robotics is to create a robot brain that stores its own representations of its sensorimotor states; to develop *state abstraction*. It is easy to provide a robotic brain with pre-defined distinct states; however, this inserts anthropomorphic bias, as the manner in which a human experimenter divides sensorimotor space into states is most likely not the same manner in which the robot would find it most useful to be done. The challenge, of course, is to tell a robot brain how to generate states without specifying which states to generate. Robots are becoming proficient at dividing discrete sensorimotor spaces into discrete states; dividing a continuous sensorimotor state into discrete spaces proves to be even more of a challenge for robot brains. One way to develop discrete state abstraction of a continuous world is to use self-organizing distinctive state abstraction (Provost et al. 2006).

The idea of self-organizing distinctive state abstraction (SODA) is that each navigational state is composed of high-level action abstractions that allow it to approximate one sensory state. The sensory states are generated through growing neural gas (Fritzke 1995). Growing neural gas (GNG) was developed so that important topological features of a given environment could be distinguished. This algorithm is ideal for developmental robotics because it is free from anthropomorphic bias and requires few parameters in order to run. Other

dimensionality reduction techniques, such as competitive Hebbian learning, require a predefined network size and other parameters. The Growing Neural Gas algorithm allows the network to set these parameters dynamically to appropriate values. The implementation of Growing Neural Gas that we used in this experiment was modified from code written by Dan Amato.

The SODA algorithm uses the GNG units to help determine its high-level actions. A high-level SODA action is composed of two steps, trajectory following and hill climbing. For trajectory following, the algorithm finds the closest GNG unit to the current sensory input, and uses Q-learning to determine the best action. After trajectory following has led the robot to a new sensory GNG state, it executes hill climbing to fine-tune the movement executed by the trajectory following. The Q-learning algorithm uses reinforcement learning to back-propagate reward from successful trials. Because our world is continuous, we used the GNG units and the low-level actions to form the Q-table.

2 Our Environment

Our experiment placed a single simulated robot, named Katie, in a simulated world. The simulator that we used, implementing a simulated Pioneer robot, was the award-winning Pyrobot software. Katie was equipped with eight sonar sensors, arranged across her front half, a stall sensor, and a light sensor. The light sensor was actually taken from the max of two light sensors, one on her front left and one on her front right. If the light value was greater than 0.2, it was discretized to 10, so that there was a clear contrast for Katie between seeing the light and not seeing the light. These ten values were combined to create her sensory state vector at each timestep. Pioneer robots (and therefore Katie) are equipped with two independently-controlled wheels that allow free movement through a continuous world. For the SODA algorithm, it is necessary to generate a list of low-level actions for the robot to use in navigation. We gave Katie a list of sixteen actions with the intention of allowing her complete freedom of movement; nine of the actions are generally forward-moving, four are turns, and three move Katie backwards.

Our world was designed as a relatively simple maze for Katie to navigate. There is a light source in the upper-left corner that serves as the goal. At the start of the experiment, Katie is placed at the point labelled '1' on the map, very close to the goal. Every six trials, she is moved to a starting point farther and farther away from the goal. We implemented this incremental increase in difficulty so that Katie would be able to learn sections of the maze at a time, and then build on that knowledge to learn larger and larger sections.

3 Our Algorithm

The GNG variant we used works by generating a new GNG unit, comprised of a representative vector and a list of neighboring units, every set number of

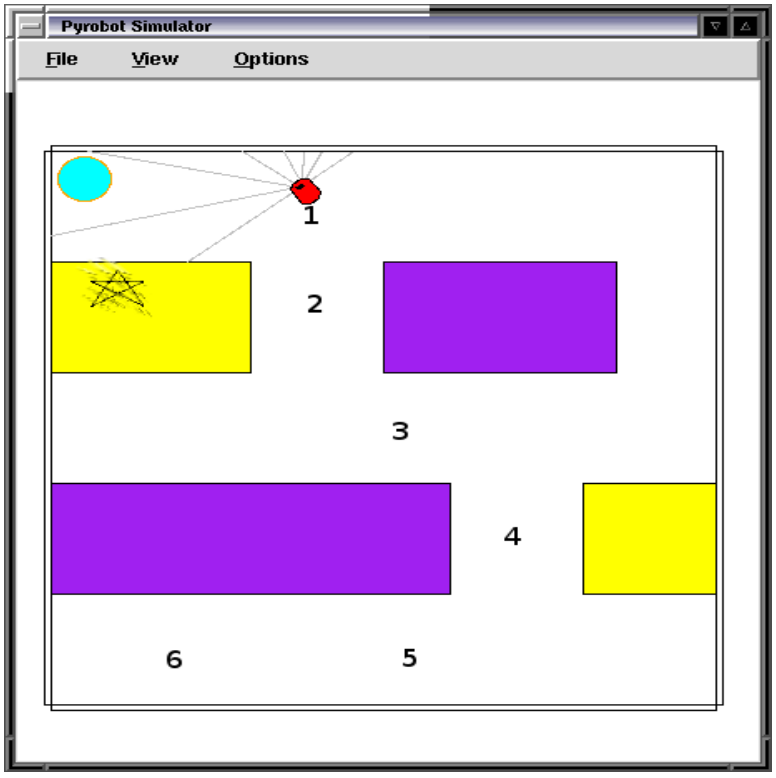

Figure 1: Katie's World. The numbers signify the sequence of starting locations that Katie uses over the course of the experiment.

timesteps. At every timestep, the GNG is given the current sensory state of the robot, and finds the closest (in Euclidean distance) unit in the network. It then slightly adjusts that unit's representative vector, sets the age of that unit to zero, and ages each neighbor of the current unit. After units reach a certain age, they are removed from the network.

Our implementation of the SODA algorithm used the low-level actions described previously. The SODA algorithm goes through a simple loop:

1. Find the closest GNG unit (as measured by Euclidean distance of the representative vector).

2. Using the Q-table, choose a low-level movement and continue to execute that movement until the closest GNG unit is no longer the same.

   - 70 percent of the time, choose the action with the highest Q-score. The other 30 percent, choose a random action.
   - The Q-scores are back-propagated through the Q-table when the goal is reached using a temporal differencing algorithm (implemented in

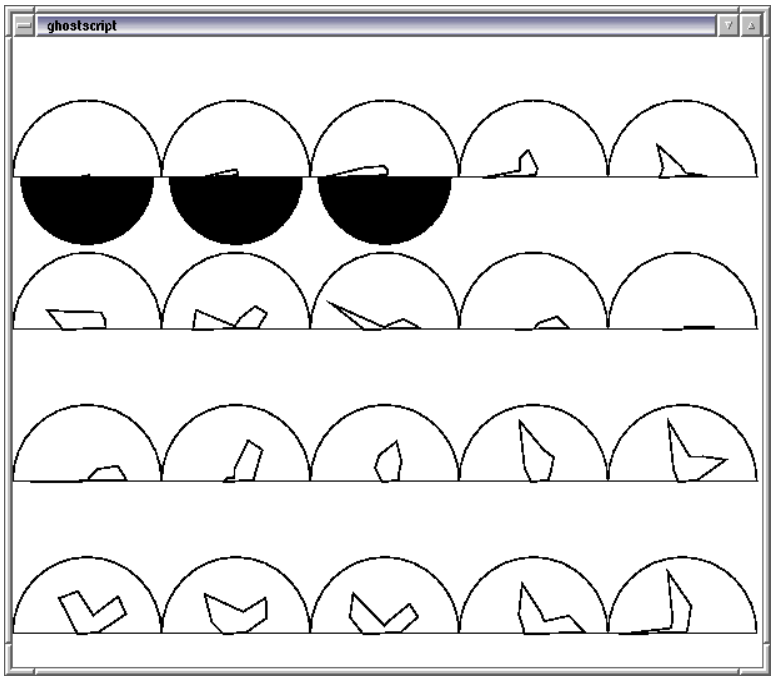

Figure 2: A set A, of 20 GNG units. These were generated using the full (non-log) sensor values. The black semicircle represents the light sensor value.

Pyrobot by Robert Casey)

- The reward for the Q-score is based on what we called a "cookie score". Katie dropped "cookies" as she travelled through the world. The cookie score was calculated by summing the amount of time that Katie spent in close proximity to a cookie. Cookies had a small decay factor so that they disappeared after a certain amount of time. The reward at the end of a trial was based on the inverse of this cookie score and on the maximum number of timesteps, so that the less time she spent visiting places she had already been, the higher her reward.
- As mentioned, the Q-table used GNG units as states. Each GNG unit that contained a light value of approximately 10 was classified as a goal state.

3. Using the same GNG unit that was used for trajectory following, commence hill climbing:

   - Execute and then reverse each low-level action to see if it would get the current sensory vector closer to the representative vector of the GNG unit you are hill climbing.

- Choose the action which will minimize the distance to the representative vector. If no action will decrease the distance, choose a new GNG unit and commence trajectory following.

4 Our Experiment

Our experiment was divided into two parts; in the first, Katie wandered randomly about the world, growing neural gas. In the second part, she ran a series of trials, during which she was rewarded for reaching the goal.

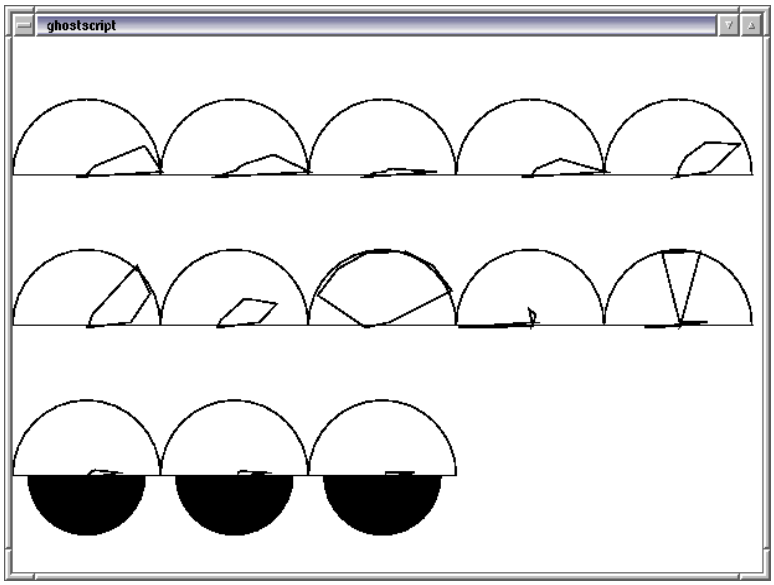

Figure 3: A subset of set B of GNG units. The full set contains 28 GNG units. These were generated by taking the log of the sensor values. The black semicircle represents the light sensor value. These images are magnified 200 percent.

Generating an appropriately sized set of GNG units required many trials and minor adjustments to the GNG algorithm. We did not want a set of GNG units with more than thirty units, because it would slow down the algorithm (particularly the hill climbing step). We also did not want a set of GNG units with less than ten units, because such a set could not encompass the variety of sensory states provided by our world. We generated two sets of GNG units that fit these parameters; one using the direct sensor data and one taking the log of the sensor values, to emphasize the differences between small values.

For the second half of our experiment, Katie was placed in the world and given a certain number of timesteps (between 1000 and 3500, depending on the distance from the starting location to the goal) to attempt to find the goal. If she reached the goal (by reaching a GNG state with a light value of 10), the

reward is back-propagated along her path. If she reaches the timestep limit and has not seen the light, she is reset to the starting position with no reward. She is placed at each of the six start positions six times before moving on to the next one.

5 Results

The results that we were hoping to see were that the cookie score and number of timesteps needed to find the goal would go down during each phase of the experiment. Because we normalized the cookie score to be divided by the maximum number of timesteps allowed, which increased with distance between the start and the goal, we also expected to see a general decrease in the cookie score through the whole experiment. We also hoped to see at worst a consistent percentage of "failed" trials; trials in which Katie did not find the goal.

When we ran the experiment with set A of GNG units, we did not find a consistent percentage of "failed" trials; in fact, from starting point #4 onwards, Katie never found the goal.

$StartPoint$	$Successes$	$AverageSteps$	$MaxSteps$
1	4	581	1000
2	3	1246	1500
3	2	1846	2000
4	0	2500	2500
5	0	3000	3000
6	0	3500	3500

This is clearly not what we were hoping to observe. Although Katie had some success in the first few trials with set A, she quickly moved beyond the point where she was capable of finding the goal and thus avoiding a very low cookie score. In the experiment with GNG set B, the results were even more drastic; Katie struggled to find the goal at the beginning, and so had no reward to propagate, and never learned to find the goal. In set B, the percentage of trials that ended with Katie stalling in a corner increased significantly as the experiment progressed.

5.1 Discussion

Although our experiment was unsuccessful, there is still much that can be learned from it. It was clearly helpful for Katie to take the log of the sensor values. This makes sense, because log-ed sensor values emphasize differences between small values, and most of the sensor values Katie encountered were relatively small (generally between 0 and 3, out of a maximum of 10). Perhaps the world was simply too complex for this algorithm; Katie might have benefitted from another layer of abstraction, or a different method of reward learning. One other concern is the use of GNG units as states for the Q-table; qualitatively observing, Katie spent a lot of time going directly to corners and getting stuck.

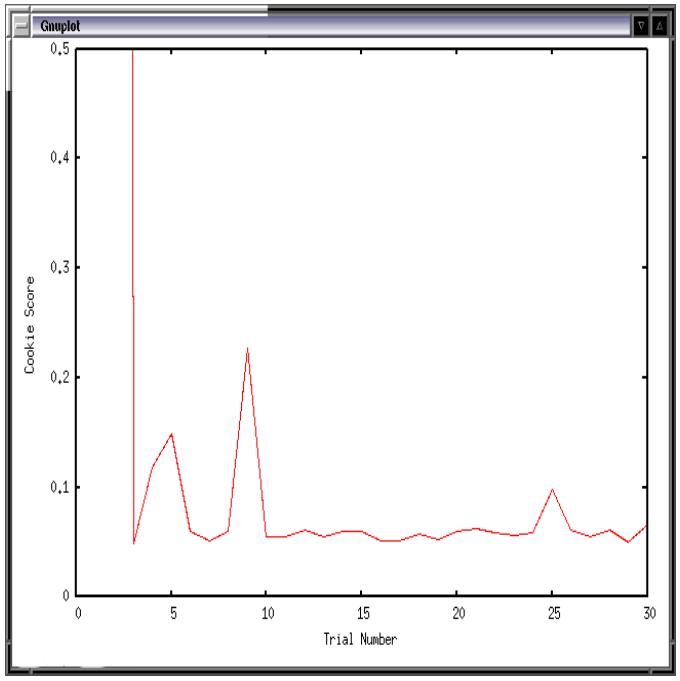

Figure 4: Cookie scores from trials with GNG set A.

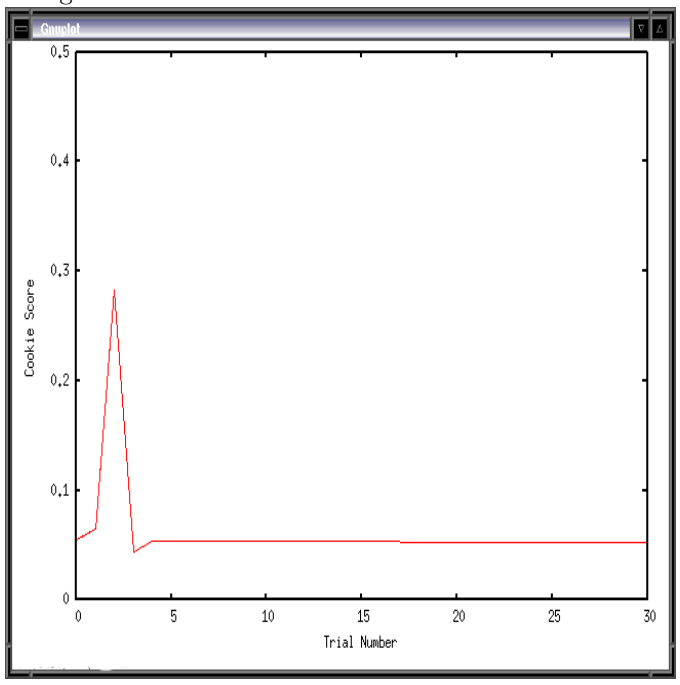

Figure 5: Cookie scores from trials with GNG set B.

Because the goal was in a corner, it is possible that the GNG states as Katie approaches the goal have the exact same sensory input state that occurs as Katie approaches a corner. This same problem could have occurred with any given feature of the map, and is indeed a fundamental concern when using sensory data for reinforcement learning states.

One solution to this problem is to implement some form of mapping. Although it is not clear to us how to implement mapping in a way that avoids anthropomorphic bias, it seems reasonable that there should be a way to have the robot brain be more aware of where it has been and where it has not been. Our cookie table was a simplified attempt at this, but it should be possible to develop a more powerful version.

It is possible that with more trials, Katie would have shown more improvement, or different learning. Due to a number of factors, we were forced to limit our experiment to six trials from each starting point; in a future experiment, we would like to try many more. It is also possible that the sets of GNG units, although they were the size we wanted, were for some reason ill-suited to the task. We generated over twenty sets of GNG units, and these were the two that seemed best, but there could potentially be significant improvement with a different set of GNG units. We generated the GNG units by placing Katie in the world and having her wander randomly. Alternatively, we could have led her on a "tour" of the world, making sure that she saw and had a chance to internalize every feature of the world.

6 Conclusions

Many interesting conclusions can be drawn from the results of our experiment. The Growing Neural Gas algorithm, as developed by Fritzke, appears to be an effective method for dimensionality reduction, even in high-diameter problems. One difficulty that we found with GNG was that although it did require few parameters, those parameters required significant fine-tuning so that we were able to generate a GNG set that was both complete and properly sized. Although the results of our experiment show that SODA did not work when using GNG units as states, we do believe that SODA could potentially solve this task, if given more information, such as our cookie score, as sensory input. Using GNG states for Q-learning creates a fundamental discrepancy between the goals of the algorithm and the complexity of the internal representation of the world.

There are numerous possibilities for future extensions of our experiment. As discussed above, there is flexibility in the GNG unit development. There is also much that could be done with the setup of the experiment. Certainly the experiment could be run with more trials, but there are also fundamental changes that could be made to the pattern of the trials. For instance, Katie could be required to stay at a given start point until she achieved a low enough cookie score or amount of timesteps to the goal. This would indicate that she has learned that portion and is ready to move on. Also, it might be interesting to test Katie on the entire maze (starting from point #6) at diverse points

throughout the experiment. We would also like to give Katie knowledge of her cookie score (or some other form of information about where she has been already) while she is making decisions, rather than solely as a magnitude on the reward.

7 Bibliography

Provost, J., Kuipers, B., Miikkulainen, R. (2006). Developing navigation behavior through self-organizing distinctive state abstraction. Connection Science 18(2).

Fritzke, B. (1995). A growing neural gas network learns topologies. Advances in Neural Information Processing Systems 7.