Rediscovering Continuous Categories

Javier Prado
jprado1@swarthmore.edu

May 15, 2006

Abstract

This paper attempts to recreate the experiment of discovering continuous categories done originally by [2]. This endeavor is worthwhile because of the power categories can and do play in the functioning of a self-governing agent. This paper attempts to recreate the learning algorithm first developed by [2]. The learning algorithm is able to take raw sensor data and generalize over it through clustering. I address the issues that concern each stage of the learning algorithm and show specific examples of a simulated Pioneer 1 mobile robot interacting within a simple simulated world.

1 Introduction

Within the realm of developmental robotics, there are several areas of research which involve trying to develop a robot to be as autonomous as possible. An ultimate goal for the developmental robotic community is the construction of a robot that incorporates an intrinsic developmental system which contain the following three critical mechanisms: abstraction, anticipation, and self motivation [1]. This experiment concerns only the first two mechanisms, abstraction and anticipation. By clustering over raw sensor data, the robot is able to abstract common events in its environment. Also, by abstracting from its basic sensor inputs, the robot is able to anticipate events in the future. This process is hierarchical in nature because the robot starts out with only its sensor inputs, and from here generalizes on those inputs to get some kind of abstraction. After this abstraction has been made, the robot can continue to generalize over these abstractions it has previously made in order to generate even more abstractions about its environment.

The learning algorithm which I implement is supposed to return a prototype for each category that it creates. The prototype is simply an average of the members of its cluster group. The learning algorithm is also supposed to produce a hierarchy of the each of the objects the robot is capable of interacting with. This is the abstraction the learning algorithm can make with using only vectors of time series.

2 Learning Algorithm

The primary accomplishment of this paper is the recreation of the learning algorithm developed by [2]. The learning algorithm that they developed consisted of four fairly distinct parts: event detection, time series comparison, sensor comparison, and sensor weighting. The motivation for the original learning algorithm was a desire for an autonomous agent to be able to abstract ideas for itself. The basic way in which the abstractions are made is by clustering on time series data.

2.1 Event Detection

Event detection is the process by which an agent in the world can recognize when an event is taking place. This is the fundamental part of the algorithm upon which everything else is based. [2] developed 4 templates which are used in conjunction with sensor data to gather a time series vector. (See Figure 1) Their argument for using such templates is since it is observable that sensorimotor agents like a mobile robots or infant humans have this intuitive ability to sense when unexpected sensor data happens upon them, then it is OK to include such an innate mechanism in forming a learning algorithm.

For this part of the learning algorithm, I decided to build up fifty-placed vectors for each of the robots sensors. This fifty-placed vector roughly equals five seconds of elapsed real time. After collecting these vectors, I do a correlation test with each vector and each template centered around the middle of the vectors, this way the time series event is centered on the event. If the correlation between the sensor vector and the template exceeds a certain threshold, then my algorithm stores the entire fifty-placed vector into a buffer of other fifty-placed vectors. Each of these fifty-placed vectors becomes an instance for cluster analysis.

2.2 Time Series Comparison

In order to compare the time series vectors, I need a clustering algorithm to do so. [2] provided a reasonably simple algorithm. The algorithm is agglomerative in nature. Simply calculate the Euclidean distance between each fifty-placed vector and chose the pair with the smallest distance. From here, average the two vectors together and replace them in the pool of time series vectors. Repeat until there is only one vector left. This should give a clustering from which to determine which sensors belong together. At a certain depth in the clustered tree, all the instances of a sensor's time series vectors should be clustered together. For example, all of the sonar sensor time series vectors should be clustered together at a certain depth. The same goes for all of the other sensors inputing data for the learning algorithm.

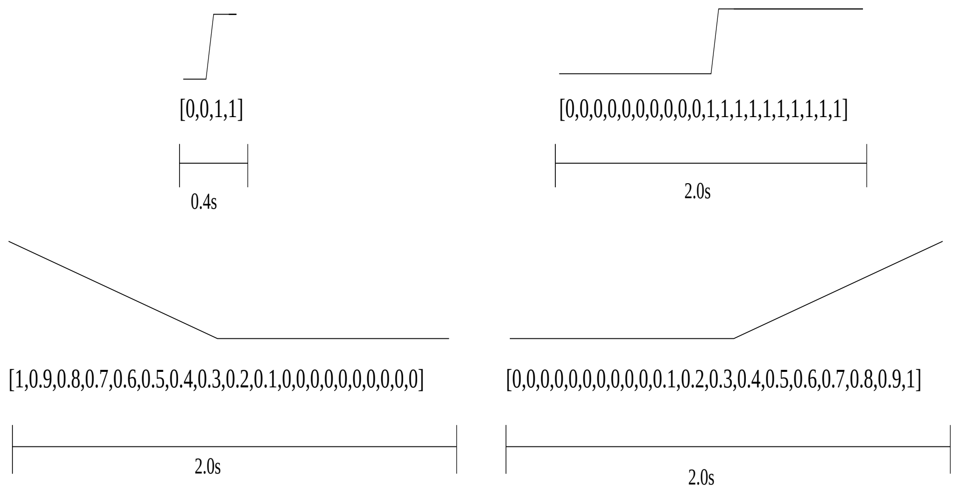


Figure 1: Figure of the templates used in detecting events in the sensor data. Because my simulation takes sensor readings roughly 10 times a second, I chose the above vectors to roughly match up with the simulation. The vectors are the exact numbers used in the correlation calculation.

2.3 Sensor Comparison

In the sensor comparison phase, the clustered sensors are used to make what is known as an alphabet. An alphabet is basically the collection of prototypes of a particular sensor at a given level of a clustering. Once again, a prototype is simply the average of all of the time series that make up a certain region of the clustering. Alphabets are useful because they allow for clustering without having to worry about what kind of sensor is being clustered over. At this stage in the learning algorithm, the robot is supposed to interact again with its environment. Using the event detection mechanism, each subsequent time series vector is compared with its respective alphabet characters. This comparison is simply the inverse of the dissimilarity or Euclidean distance used earlier in the original clustering. The comparison generates what are known as event signatures. These event signatures allow for a more general clustering of the environment.

2.4 Sensor Weighting

For this final section of the learning algorithm, I chose not to follow [2] algorithm exactly. Whereas they propose to cluster weighted event signatures, I chose to simply cluster the event signatures together. This seems reasonable since any signature that has no correlation with an object in the environment will not

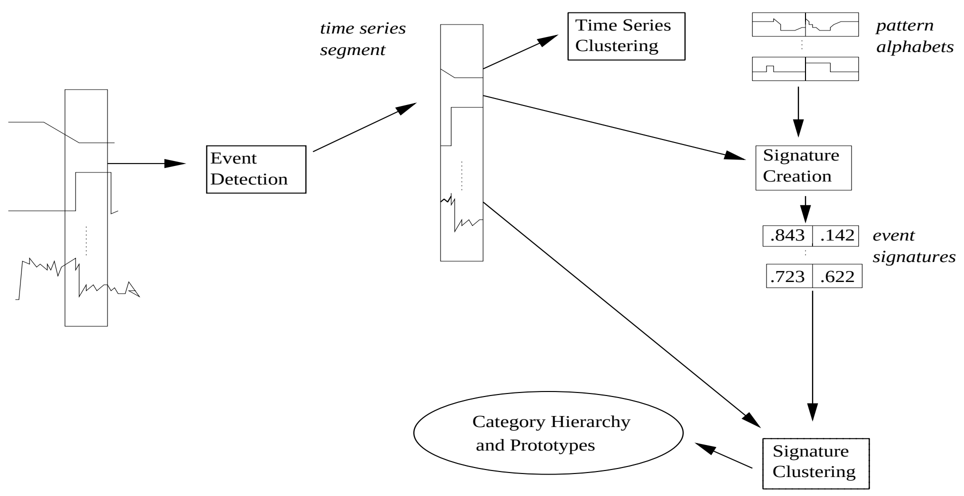


Figure 2: Figure of the general organization of the learning algorithm. Of particular note are the two instances of clustering, once done right after the time series segments have been created and once more with the event signatures. Note that both times the clustering uses the time series segments. This lends the algorithm to be used online, which unfortunately I could not do.

have a high similarity anyway. This means that when the final clustering is done on the event signatures, those signatures that have low similarity with a particular object in the world will not be clustered together.

An overview of the whole algorithm follows: Figure 2.

3 Experiment

The simulated world in which the simulated Pioneer robot interacts is very simple. I chose this world to be simple because of my focus on the learning algorithm and its implementation. For the simulated robot itself, it has seven sensor inputs from which to inform its learning algorithm: four front sonar sensors, two gripper break beam sensors, and one internal velocity sensor. The simulated robot also utilizes a blobbified camera for the use of detecting objects within the environment. The environment itself is made up of a puck, a wall and a simulated table leg. Both the puck and the leg are small enough to fit into the robot's grippers. Both the leg and the wall will not be moved when bumped into, but the puck will. Each of the three objects is colored both internally and externally red. This is to allow the blobbified camera to seek out red blobs and approach them. The simulated robot is controlled by a simple finite state machine like brain. The robot searches for an object to interact with, and when

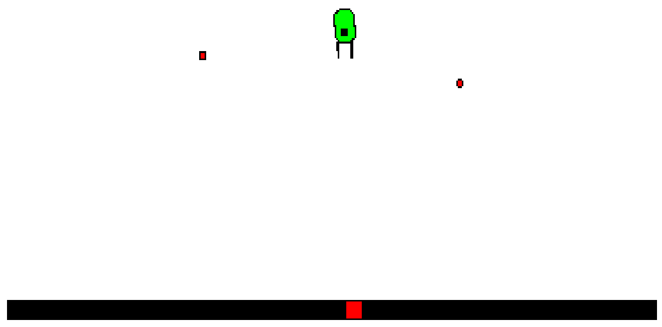
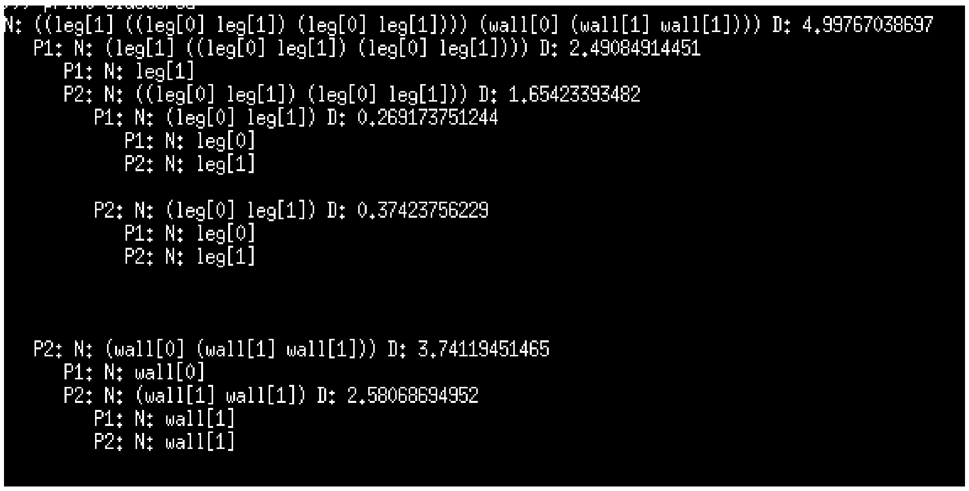


Figure 3: Figure of the simulated world. This world simulates a Pioneer robot with three objects with which it can interact. Two of these objects fit in its gripper. The object on the left is a simulated table leg, it fits into the robot's gripper but will not be moved when run into. The object on the right is a simulated puck. It also fits into the robot's grippers but unlike the simulated leg, it does move when run into. Finally, the wall is at the bottom of the image. I inserted a red colored box to draw the robot towards the wall. Other than the coloring, the red colored box is completely flush with the surrounding wall.

it detects such an object with its blobbified camera, it seeks it out using a direct path. The robot is programmed to continue moving for another two seconds after the robot encounters an object so that the learning algorithm's event detection can take place. For my experiment, I had the robot interact with each object three times. After all of the time series vectors were collected, I clustered over them to generate the pattern alphabets for each sensor. With the alphabet patterns in hand, I had the robot interact with each object again for another two times, each time using the event detection to generate a time series vector. Each vector was compared using the inverse of the dissimilarity function used to produce the sensor signatures. Finally, with all the event signatures, I did one final clustering on the event signatures to produce the category hierarchies. See Figure 3 for an illustration of the world.



```
N: ((leg[1] ((leg[0] leg[1]) (leg[0] leg[1]))) (wall[0] (wall[1] wall[1]))) D: 4.99767038697
   P1: N: (leg[1] ((leg[0] leg[1]) (leg[0] leg[1]))) D: 2.49084914451
      P1: N: leg[1]
      P2: N: ((leg[0] leg[1]) (leg[0] leg[1])) D: 1.65423393482
         P1: N: (leg[0] leg[1]) D: 0.269173751244
            P1: N: leg[0]
            P2: N: leg[1]

         P2: N: (leg[0] leg[1]) D: 0.37423756229
            P1: N: leg[0]
            P2: N: leg[1]



   P2: N: (wall[0] (wall[1] wall[1])) D: 3.74119451465
      P1: N: wall[0]
      P2: N: (wall[1] wall[1]) D: 2.58068694952
         P1: N: wall[1]
         P2: N: wall[1]
```

Figure 4: Figure of the results from one final clustering over the event signatures. The objects which the algorithm was able to differentiate were between the leg and the wall. The number following the label of the event is simply a reminder of which numbered interaction that was between the robot and the labeled object.

4 Results

The results that I received were all very promising. My implementation of the learning algorithm was successful enough to provide some categorical generalizations. From the results of Figure 4, it can be seen that the algorithm is in fact able to differentiate between the leg and the wall. I find this to be curious because both have a similar velocity behavior with respect to running into an immovable object. The fact that the puck was not even on the list of encountered objects is also interesting. Usually at least all of each of the different kinds of sensors are set off by the puck when it is being handled by the robot.

5 Conclusion

Overall I can judge this experiment to be a success. The learning algorithm has been successfully implemented. The only issue I have with this is the enormous computational cost of clustering. I believe it is primarily my responsibility to optimize the clustering algorithm, yet I do not see how it may be possible to run the entire learning algorithm online. The number of time series vectors generated during the initial event detection phase was staggering, something on the order of fifty-thousand. This may seem like a small number but the way in which I implemented the algorithm, it took almost an hour simply to cluster the first time. Ultimately I believe this is a step in the right direction. Having a mobile robot explore its environment and generate its own abstractions is certainly what developmental robotics is striving towards. Yet because of the large computational cost I think we are still a ways off from having an implementation that can run online, removed from the engineer.

References

[1] Douglas Blank, Deepak Kumar, Lisa Meeden, and James B. Marshall. Bringing up Robot: Fundamental mechanisms for creating a self motivated, self-organizing architecture. *Cybernetics and Systems*, 36(2), 2005.

[2] Michael T. Rosenstein and Paul R. Cohen. The Playground Experiment: Task-Independent Development of a Curious Robot. 1999.