Top-Down Abstraction Learning Using Prediction as a Supervisory Signal

Jonathan Mugan
21CT, Inc.
Austin, TX 78730
jmugan@21ct.com

Abstract

We present a top-down approach for learning abstractions whereby a robot begins with a coarse representation of the world and incrementally finds new distinctions as they enable the robot to better predict its environment. The approach has been implemented on a simulated robot that learns new distinctions in the form of variable discretizations through autonomous exploration. This paper discusses how to generalize this approach to learning broader abstractions.

Introduction

Humans are great at ignoring irrelevant information and focusing on what is important. For instance, we know enough about bananas to be able to buy and eat them, but most of us don't understand their molecular structure because we don't need to. Possibly because we can so effortlessly understand our environment, we have been continually surprised at how difficult it has been to build that ability into a robot.

One way to think about the problem is to begin with an overwhelmingly complex sensory input and to search for ways to make it simpler. Examples of this approach include tracking blobs in computer vision, clustering, and principal components analysis. These approaches are unsupervised and bottom-up, but while such approaches can be a necessary starting point, they are not enough. Robots often need to be able to identify small distinctions that lead to big consequences. Imagine being a rat in a Skinner box where you could observe a screen full of complicated shapes. Imagine further that a small dot in the lower right corner of the screen determined whether a painful electric shock would come on the left side of the cage or the right. Bottom-up methods looking at the structure of the data without accounting for the consequences might never find this important distinction.

This paper advocates a top-down approach to learning abstractions. Instead of beginning with a fine-grained resolution of the world and learning abstractions to make it simpler, we propose beginning with a coarse representation of the world and making it finer by learning important distinctions. In learning distinctions, we seek to balance a trade-off between a representation of the world that is fine enough to be useful, but not so fine as to overwhelm computational resources. A top-down approach requires a supervisory signal, and our signal comes from trying to predict events known to the robot. In the Skinner-box example, the rat would try to predict when it would get shocked on the right side of the box, and it would search the screen for a feature that would help it to reliably make that prediction.

Abstraction Learning Using Predictive Models

Our top-down abstraction learning approach is shown in Algorithm 1. The method begins with a coarse representation consisting of a few features. Changes in feature values in the environment gives rise to events, and therefore the algorithm begins with a non-empty set of events $\mathcal{E}$. The method seeks to find regularities in the environment, and the algorithm begins with a set of predictive models $\mathcal{M}$. The algorithm requires at least one event to begin the distinction learning process, but the initial set of predictive models can be empty.

Algorithm 1 Abstraction Learning Algorithm

Require: a small, non-empty set of events $\mathcal{E}$
Require: a (possibly empty) set of models $\mathcal{M}$
while robot is still alive **do**
  sense the environment and update statistics
  for $e \in \mathcal{E}$ **do**
    if new models M can be found that predict e **then**
      $\mathcal{M} \leftarrow \mathcal{M} \bigcup M$
    end if
  end for
  for $m \in \mathcal{M}$ **do**
    if a new distinction d can be found that makes m more deterministic **then**
      convert d to set of events E
      $\mathcal{E} \leftarrow \mathcal{E} \bigcup E$
    end if
  end for
  execute some action
end while

The algorithm continually searches for new models that predict its current set of events $\mathcal{E}$. If one or more models is found, they are added to the current set of models $\mathcal{M}$. New features come from finding new distinctions, and for each

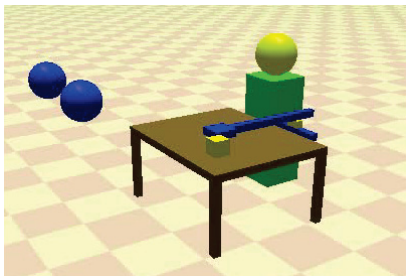

Figure 1: Robot with an arm (blue), a block (yellow), and two floating heads (blue). It perceives its environment as 46 variables (43 continuous and 3 Boolean). The robot autonomously explores to learn actions and abstractions. For example, in the process of learning to manipulate the block, the robot learns what it means for the block to be on the "left" side of its hand.

predictive model, $m \in \mathcal{M}$, the algorithm searches for some new distinction that makes m more deterministic. If such a distinction is found, it is converted to a set of events E, which is added to the total set of events $\mathcal{E}$. We make the assumption that distinctions learned to make predictive models more reliable are broadly useful to the robot, and therefore on the next iteration of the while loop the algorithm will have new events that it can learn models to predict.

Note that the sets of models and events grow over time. To contain model growth, one can remove models that do not become sufficiently reliable through added distinctions (Mugan 2010). One can also consider removing distinctions (and therefore events) that no longer appear useful.

Abstraction Learning Using Dynamic Bayesian Networks and Landmarks

Algorithm 1 was implemented on the simulated robot shown in Figure 1 by Mugan and Kuipers (2012; 2010).[1] Models are represented using Dynamic Bayesian Networks (DBNs) (Dean and Kanazawa 1989), and distinctions are represented as discretizations of continuous variables.

Representing Distinctions

Mugan and Kuipers (2012; 2010) implement distinctions by discretizing continuous variables using *landmarks* (Kuipers 1994). A landmark is a symbolic name for a point on a number line. Figure 2 shows a number line for a variable X with two landmarks, which creates five distinct values for X called *qualitative values*.

To begin with a small set of events, as required by Algorithm 1, each continuous variable $\dot{X}$ that is a first derivative of another variable X is given a landmark at 0. With this initial level of abstraction, the robot cannot distinguish between the different values of X; it can only know that it is between $-\infty$ and $+\infty$. But because $\dot{X}$ has a landmark at 0, the robot can know if the value of X is increasing, decreasing, or remaining steady.

[1] A video explaining the implementation can be found at http://videolectures.net/aaai2010_mugan_qlap/.

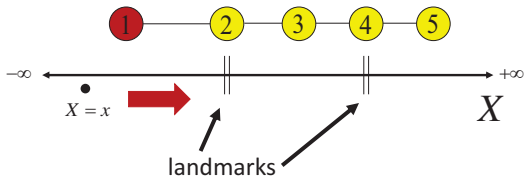

Figure 2: A number line for a variable X with two landmarks. The two landmarks partition the infinite set of values into five qualitative values. The current value of $X = x$ is shown in red. The large red arrow indicates that the value of X is increasing.

As the robot learns new landmarks, such as those shown in Figure 2, the robot can make more distinctions between the different qualitative values of variable X. Each new landmark creates two new events because the qualitative value $X = x$ can be reached from either above or below on the number line.

Representing Models

Mugan and Kuipers (2012; 2010) implement predictive models as dynamic Bayesian networks (DBNs). At the core of each DBN is a *contingency*. A contingency is a pair of events that occur together in time such that an antecedent event is followed by a consequent event. An example would be that flipping a light switch (the antecedent event) is soon followed by the light going on (the consequent event). Contingencies are a useful representation for predictive models. They are easy to learn because they only require looking at pairs of events, and they are a natural representation for planning because they indicate how events lead to other events.

Mugan and Kuipers (2012; 2010) create a model for each pair of events (e_1, e_2) where the probability of event e_2 occurring soon after event e_1 is higher than the probability of event e_2 occurring on its own. As the robot observes and explores the environment, it identifies context variables for each DBN model through *marginal attribution* (Drescher 1991) as shown in Figure 3. Marginal attribution works by iteratively adding context variables as long as each new context variable makes the DBN marginally more deterministic.

The context variables form a conditional probability table (CPT) that gives the probability of the antecedent event leading to the consequent event for each combination of values for the context variables. The robot seeks to learn deterministic models, and the level of determinism for each model is measured by the highest probability of any value in the CPT, as long as that value is less than 0.75. After that value, the level of determinism is measured by the entropy of the entire CPT. This dual method of measuring determinism is used because experiments showed that it is initially best to find some situation in which the contingency is reliably achieved, and then it is useful to find a representation of the environment that is predictable in all situations.

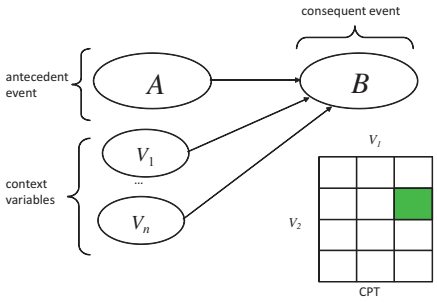

Figure 3: Dynamic Bayesian Network (DBN). The contingency that antecedent event A leads to the consequent event B forms the core of the DBN. The conditional probability table (CPT) gives the probability of the antecedent event A bringing about the consequent event B for each possible value of the context variables $V_1, \ldots, V_n$.

Learning New Distinctions

For a model m that predicts event A will lead to event B, the supervisory signal comes from observing event A and then noting if event B soon follows. We call this an *application* of model m. For each application of model m, the environment replies with True if event B follows event A, and with False otherwise.

To learn new landmarks to implement line 9 of Algorithm 1, the algorithm can note the real value of each variable V_i each time model m is applied (line 2 of Algorithm 1). The algorithm can then determine if there is a landmark that, if created, would make the CPT of model m more deterministic. If so, the algorithm creates that landmark. The robot will then have two new events that it can try to predict.

Generalizing Landmark Learning

Our approach to abstraction learning is to set up a space over which the robot can search for new distinctions that make predictive models more deterministic (line 9 of Algorithm 1). This process was relatively straightforward for discretizing continuous variables as shown in Figure 4(a). We can generalize this idea using an *abstraction hierarchy*. An abstraction hierarchy is a domain-specific, user-defined hierarchy with different levels of representation at each layer.

Algorithm 2 Update Abstraction Hierarchy Statistics

Require: a set of abstraction hierarchies $\mathcal{H}$
Require: a set of models $\mathcal{M}$
1: **for** each predictive model $m \in \mathcal{M}$ **do**
2: **for** each abstraction hierarchy $h \in \mathcal{H}$ **do**
3: note the value of each abstraction instance in h at the level below the current level of abstraction each time model m is applied
4: **end for**
5: **end for**

Figure 4(b) illustrates an abstraction hierarchy for a robot agent with the task of learning to predict and control the dynamics of a computer operating system. For example, "file"

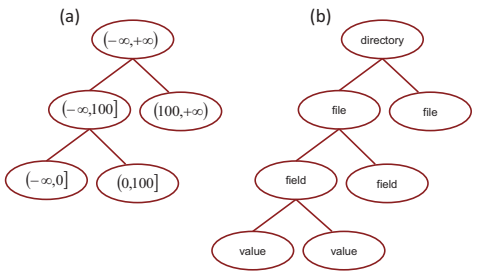

Figure 4: Two abstraction hierarchies. (a) A hierarchy over a continuous variable with two landmarks. Initially there are no landmarks. Learning a landmark at 100 splits the state space. The left state is split again when a another landmark is learned at 0. (b) A domain-specific abstraction hierarchy for configuration files in an operating system.

could be a variable that indicates if the hash of a specific configuration file has been changed. This might be the level of detail needed to predict an event that the robot cares about, such as if a mission-critical program will function correctly. By searching over this hierarchy, the robot may learn that it does not need to know how exactly the file changed, only that it was changed.

Algorithm 2 describes how the statistics for the abstraction hierarchies are maintained on line 2 in Algorithm 1.

Conclusion

Predictive models allow a robot to learn abstractions because each model can serve as a self-supervised learning problem. The abstractions learned using this top-down method will not be uniform, and this is as it should be. We want the robot to have deeper knowledge in areas that matter and less knowledge in areas that do not. If the robot can perceive the world at the right level of detail relative to its goals, the applicability of existing reasoning and planning methods can be extended.

Acknowledgments

The author wishes to thank Eric McDermid for his valuable comments on an earlier draft of this paper.

References

Dean, T., and Kanazawa, K. 1989. A model for reasoning about persistence and causation. *Computational intelligence* 5(2):142–150.

Drescher, G. L. 1991. *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*.

Kuipers, B. 1994. *Qualitative reasoning: modeling and simulation with incomplete knowledge*. MIT Press.

Mugan, J., and Kuipers, B. 2012. Autonomous learning of high-level states and actions in continuous environments. *IEEE Trans. Autonomous Mental Development* 4(1):70–86.

Mugan, J. 2010. *Autonomous Qualitative Learning of Distinctions and Actions in a Developing Agent*. Ph.D. Dissertation, University of Texas at Austin.