Application of concept grounding techniques to reduce dimensionality in sensory-motor spaces

Pejman Iravani, Lucia Rapanotti and Jeffrey Johnson
The Open University, Department of Computing, Walton Hall, MK76AA, Milton Keynes, UK

Abstract. Robotic systems are characterised by large state and action spaces, which are problematic for learning and adapting techniques. This paper presents a method to alleviate learning dimensionality problems, by classifying similar states and actions into abstract concepts. Unlike previous techniques, concepts are generated based on the robot-environment interaction rather than being arbitrarily imposed. The validity of this method is demonstrated empirically using a real robot platform in a ball-targeting task.

1 Introduction

Our main goal is to develop a method that deals with the "dimensionality" problem in real robotic systems, using generalisation and abstraction techniques for concept generation. It is shown that analysing the raw data from the robot-environment interaction makes it possible to generate concepts that represent "useful" robotic behaviour. These concepts can be used to generate compound behaviours in ways that avoid combinatorial explosion. Concepts can be further used to control robots and learn new behaviours, which is the ultimate goal of this research. The theory is illustrated by experimental results.

The following framework is widely used in robotic systems. The Markov Decision Process (MDP)[10, 3] is a very general framework used to represent, control and study a wide range of dynamic systems. MDPs can be used in robotics to represent the dynamics of robot-environment interactions, and to decide or learn the most suitable control actions.

A *control policy* or *policy* is a rule that determines which action to take for a given state. Reinforcement learning (RL)[10] uses the MDP framework and an iterative process to find optimal control policies. RL suffers from the so-called "curse of dimensionality"[10, 3], that is, the complexity of the problem grows exponentially with the number of state variables and actions accessible to the robot. This is especially acute in robotics, where state variables and actions are usually a continuous range of values [5]. The method reported in this paper is a practical step towards addressing this problem in a real robot context.

The paper is organized as follows. Section 2 defines the main motivation for the work and reviews the related literature. Section 3 formally defines the framework and its elements. Section 4 explains the method applied to abstract behaviours. Section 5 empirically validates the method and presents its results. Section 6 concludes and indicates further research directions.

2 Motivation and related work

Various "engineering" approaches have been introduced to deal with the dimensionality problem, most of them relying on mathematical or statistical generalisation techniques. For example, Smart and Kaelbling [5] and Shaal and Atkenson [4] implement an RL algorithm combined with locally weighted techniques (neighbouring approximation). Tesauro's [11] TD-gammon uses a standard multilayer neural network to approximate the estimated outcome of any of the board positions in a Backgammon game. Stone and Sutton [9] use tile coding as a generalisation technique for a simulated soccer task. Smith [6] uses self-organising maps to generalise individual states and actions into areas in the self-organising map. Although these generalisation techniques show some improvements on the learning rates over the original RL definition, they do not provide an explicit representation for their generalisations. This limits their re-use in tasks different from the one used to build the generalisation itself.

This work aims to generate explicit representations of abstract elements (states, actions and behaviours), with the hypothesis that these can be used for: 1) robot control, 2) behaviour learning, 3) robot to robot communication, 4) abstract modeling and planning, all of these, with the possibility of re-using the learned abstractions in different tasks.

The approach we follow takes some inspiration from [8, 7] on the emergence of lexicon and sharing of event descriptions. In that work, autonomous agents share a common environment, and through the use of natural language and the interaction with the environment, they define a common lexicon (abstract representation of the environment) that describes the objects and events that take place in the environment. We show that it is possible to apply similar techniques to a single agent, which is embodied in the environment and learns concepts (abstractions) regarding its own behaviour. Moreover, as with language itself, it can be used in different scenarios.

3 Basic elements of the framework

This section introduces some of the notation and definitions that are used in our approach. These are grounded in the RoboCup (www.robocup.org) small-league system that we use as a test-bed. Figure 1 illustrates the proposed test-bed.

- *Objects*: elements in the physical environment, including the robots, the ball, etc. We use the following expression to designate object i, $Obj_i = (x, y, \theta, \vec{s}, w, z)$, where x and y are the cartesian coordinates of the object's centroid, θ is the object's direction with respect to the X coordinate axis. z is the size of the object. Finally, the linear and angular speeds are represented as $\vec{s}$ and w respectively. See Figure 1.

- *Actions*: things a robot can do in the environment; at the lowest level of abstractions, these are voltages sent to the two motors of each robot (those controlling its left and right wheels). Let A represent the set of all possible actions (voltage pairs).

- *Environmental states*: sensor information about the environment; for instance, from a robot's viewpoint its distance from the ball or the goal post. In the experiments reported in this paper, the environmental state is composed by the measure of the distance and deviation to the target position (ball). Let S represent the set of all possible environmental states. Figure 2(c) illustrates the state representation when the robot is at two different positions.

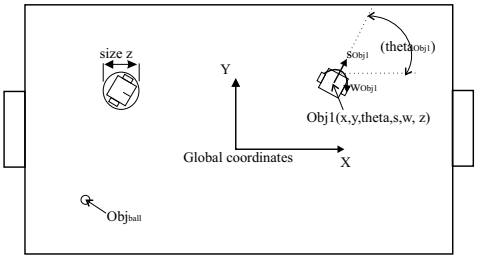

Figure 1: The test-bed with two robots and a ball.

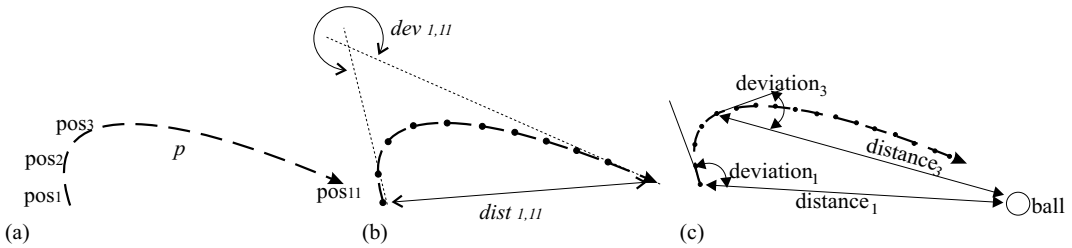

Figure 2: (a) A path p. (b) The result of the $dist$ and dev operators. (c) State representation for a ball targeting task.

- *Behaviours or policies*: behaviours express correspondences between states and actions; for instance, when the robot is close to a team player, it may pass the ball to it. Let $B : S \rightarrow A$ represent one such behaviour.

- *Robot-environment interactions*: these represent what is observable of the robot's behaviour in the environment; for instance, the robot's trajectory when it moves. In the experiments reported in this paper, we use the robot's path or trajectory to analyse its behaviour. We define a *path* as the sequence of positions described during the robot's movement, and designate it using, $p = \langle pos_1, pos_2, ..., pos_n \rangle$. Each *position* is defined as, $pos_i = (x_i, y_i, \theta_i)$, where $x_i, y_i, \theta_i \in R$, with x and y representing robot's centre, θ is the robot's orientation. Further, we have defined two operators, $dist_{ij}$ and dev_{ij} which measure the distance and deviation between the positions i, j of a path. Figure 2(a) illustrates a path P composed by the different pos_i. Figure 2(b) illustrates the result of applying the $dist_{1,11}$ and $dev_{1,11}$ operators on some of the positions of path P.

- *Goals*: these are the high-level objectives of the robotic system; for example,'scoring a goal' or 'dribbling the ball between two positions in the environment'.

4 Abstracting behaviour

This section presents our method for abstracting behaviours. Given our definitions this means abstracting actions, states and their correspondences. We discuss the approach based on a ball-targeting behaviour. Section 5 will present an experiment based on the method.

For simplicity of presentation, let us assume that we have a set of paths that correspond to a successful ball-targeting behaviour. Let $P = \{p_1, p_2, ..., p_n\}$ be such a set. Figure 3(a) illustrates the paths in P

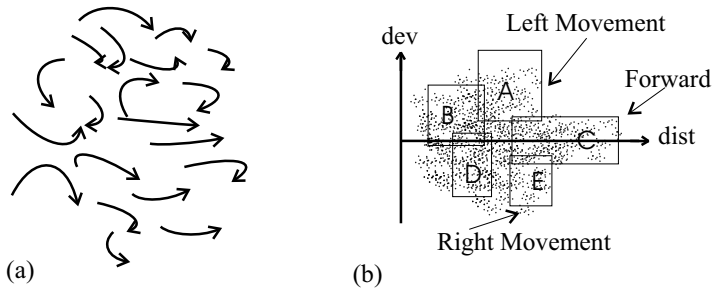

Figure 3: (a) illustrates a set of possible paths in the environment. (b) illustrates the distribution of distances and deviations of the paths shown in (a); it also shows some equivalence classes.

Let $dist_{ij}$ and dev_{ij} be the distance and deviation between the positions i, j of a given path. Given p, let us compute the distances and deviations values starting from position $i = 1$ and $j = i + jump$, until the $j = n$, where n is the last position of the path, and $jump$ is a predefined constant. If we apply these operations on all the paths in P, the result is a distribution of distances and deviations representable in a 2D graph (see Figure 3(b)).

One of the most important ideas in this paper is that the set of distance-deviation distributions observed in the paths (Figure 3(b)) can be clustered into classes, and that a representative can be defined for each cluster. This means that many 'similar' distribution points can be represented by one unique point, thereby reducing the combinatorial complexity of the representation. In our experiments, we have used K-means [2], with a predefined number of k-centres, to cluster the distance-deviation distributions. Each k-centre is then used as the class representative.

Because of the correspondence between paths and actions, clustering the distribution of distances and deviations corresponds to clustering (abstracting) the corresponding actions. Figure 3(b) shows classes A, B, C, D, etc., which represent sets of similar observed actions. We refer to abstracted actions as *action concepts*. For example, C is a *move forward far* concept while B is a *move forward and left* concept.

The next step in the method is to abstract environmental states into *state concepts*. Here, state concepts are abstracted in relation to action concepts. At any time t, there is an active action concept, that is, the action concept suitable for the environmental state at the same time t. We use this rationale to define similar states, that is, the states in which a same action concept was used are considered equivalent or state concepts.

Finally, it remains to generate *behaviour concepts*, expressing mappings between state concepts and action concepts. Because of the state concept defined earlier, this mapping is straightforward.

5 Experiment and results

The experiment was designed to empirically validate the method introduced in the previous sections. We consider generating concepts that represent a ball-targeting behaviour, which is considered successful when it takes the robot from its initial position to a static ball.

Data gathering:

In order to generate data on successful paths, we hand-coded a control policy using a well-known potential field technique [1]. Based on distances to the target from 0 to 300

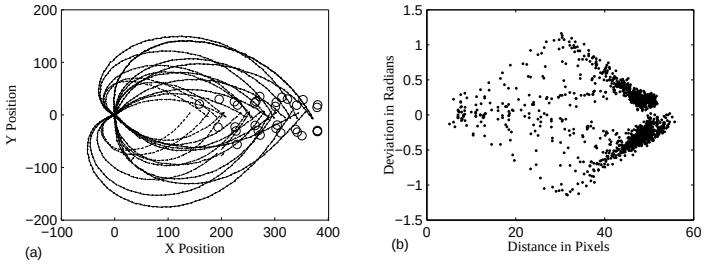

Figure 4: (a) The 30 paths, p, used in this experiment. (b) The distance and deviation distribution for all of the 30 paths (dots).

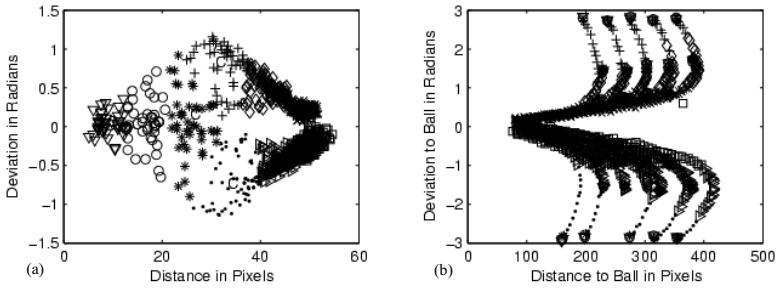

Figure 5: (a) The resulting of clustering the action space with 9 k-centres. (b) The mapping of action concepts on the environmental state space.

units and orientation to target form 0 to 360 degrees, this policy selects a pair of motor powers each ranging between 0 and 100. Thirty robot paths were generated in this way. Let $P = \{p_1, ..., p_{30}\}$ be the set of all these paths; Figure 4(a) illustrates these paths.

In a second and off-line step, the distributions of distances and deviations for all the paths were calculated, with $jump = 6$. The result is illustrated by dots in Figure 4(b).

Action concepts: K-means was used to automatically generate clusters of 'similar' actions (action concepts) using the data points illustrated in Figure 4(b). These data points illustrate the distance-deviation distribution of all the segments. Because of the previously stated correlation between segments and actions, clustering the distance-deviation space is equivalent to clustering the action space (i.e. the powers to each motors).

The results of clustering the action space can be seen in Figure 5(a) using $k = 9$ (cluster centres). Note that the number of action concepts is equal to the number of clusters. Each symbol (e.g. star, square or plus) corresponds to the actions considered similar, representing the same action concept.

State concepts: As previously seen in Section 4, state concepts are defined in a straightforward manner by associating each state with its action concept. Figure 5(b) illustrates the state space where each state is decorated with the symbol representing its action concept. As we can observe, nearby areas in the state space correspond to the same action concept; the rationale behind this correspondence is that similar actions are used in similar environmental states. Finally to define the state concepts, we assume a 1-neighbour relation: that each environmental state is considered the same as its closest neighbour.

Behaviour concepts: Finally, a behaviour concept for ball-targeting is defined as a mapping between the sets of action and state concepts. When applied to each state concept, the be-

haviour concept will chose the most appropriate action concept to satisfy the ball-targeting task.

Results: In order to evaluate the accuracy of the method, we replaced the hand-coded policy with the learned behaviour concept. For the quantitative evaluation of accuracy, two metrics were used, the average speed and the accuracy at targeting the ball. Results showed that the performance in terms of speed and accuracy between the two approaches (hand-crafted policy vs. behaviour concepts) were very similar. This suggests that this approach achieves a satisfactory performance, while drastically reducing the initial system's dimensionality.

6 Conclusions and Future work

In this work, we have shown how to use clustering techniques to generalise, thus reduce, large sensory-motor spaces. In our case, the initial sensory-motor space had $> 10^9$ (100 actions per motor, 300 different distances to the target, and 360 different orientations to the target) different possibilities, while the resulting abstract space had only $< 10^2$ (9 action concepts and 9 different state concepts). This drastic reduction of the sensory-motor space will certainly have some positive repercussions on the learning rates of the system, while not necessarily reducing its performance (like shown in the experiment). The reason for this is that the "atomic" actions and states in a robotic system are very similar and the usage of any of them produces a similar outcome.

Future work includes the extension of this method to other tasks and its integration into a robotic architecture. It also remains open to fully exploit the explicit and abstract representations here generated. We believe that abstract learning, model generation and even planning are some of the new possible capabilities.

References

[1] R. Arkin, Behavior-Based Robotics, MIT Press, 1998.

[2] R. Duda and P. Hart, Pattern classification and scene analysis, John Wiley and Sons, (1978).

[3] L.P. Kaelbling and M.L. Littman and A.W. Moore, Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research, **4**:237–285, 1996.

[4] S. Schaal and C. Atkeson, Robot Juggling: impementation of Memory-Based Learning, IEEE Control Systems,**14**:57–71, 1994.

[5] W. Smart and L. Kaelbling, Practical reinforcement learning in continuous spaces, International Conference on Machine Learning, **361**:2381-2395, 2003.

[6] A. Smith, Applications of the self-organising map to reinforcement learning, Neural Networks, **15**:1107–1024, 2002.

[7] L. Steels and J. Baillie, Shared grounding of event descriptions by autonomous robots, Robotics and Autonomous Systems, **43**:163-173, 2003.

[8] L. Steels, The Origins of syntax in visually grounded robotic agents, Artificial Intelligence, **103**:133–156, 1998.

[9] P. Stone, R. Sutton, Scaling RL toward RoboCup Soccer, International Conference on Machine Learning, 2001.

[10] R. Sutton and A. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.

[11] G. Tesauro, Temporal Difference Learning and TD-Gammon, Communications of the ACM, **38**:58–68, 1995.