A developmental agent for learning features, environment models, and general robotics tasks

Brandon Rohrer
Intelligent Systems, Robotics, and Cybernetics Group
Sandia National Laboratories
Albuquerque, New Mexico, USA
Email: rohrer@sandia.gov
Web: http://www.sandia.gov/rohrer

I. INTRODUCTION

BECCA, a developmental agent, is described and demonstrated performing a high-dimensional visual servoing task. BECCA learns 1) a feature representation of its state space, 2) a model of its environment, and 3) how to behave in order to receive reward. It learns these things concurrently in an on-line and incremental fashion, without any prior knowledge of its environment or the nature of its inputs and outputs. (See [4] for the full paper.)

Biological developmental agents, such as children, learn both feature representations and world models through their actions and interactions. Learning a feature representation is the act of mapping low-level inputs onto higher level perceptual symbols or categories. Learning a world model is the act of recording observed features in order to capture salient aspects of the agent’s experience. The learning of feature representations and world models that are both useful and biologically plausible are among the chief technical challenges for those seeking to create developmental agents. [1] This work is an effort to address the problems of integrated feature, model, and task learning in a unified framework.

II. METHOD

A BECCA agent interacts with the world by taking actions, making observations, and receiving reward. (See Figure 1.) Formulated in this way, natural world interaction is a general reinforcement learning (RL) problem, and BECCA is a potential solution.

BECCA’s feature creation algorithm identifies patterns in the agent’s input that are repeated and thus likely to have semantic relevance. It works by grouping the elements of the input vector into groups whose activity is somewhat correlated. In a pixel array exposed to a video stream of broadcast television, for example, the correlation between two neighboring pixels will be much higher than that occurring between distant pixels, and a small number of pixels grouped by correlation will be closely related in space. The groups of input elements form input subspaces, and unit vectors in these subspaces represent features. The feature creator creates new features by adopting novel inputs, also known as imprinting. Inputs must be sufficiently different from existing features in order to be imprinted. Returning to the pixel array example, once a small group of correlated pixels has been formed, features will be created based on patterns observed in those pixels. These may include horizontal, vertical, and diagonal edges, (as in Figure 2g) as well as uniform intensity and center-surround patterns.

Fig. 1. At each timestep, the BECCA agent completes one iteration of the sensing-learning-planning-acting loop, consisting of six major steps: 1) Reading in observations and reward. 2) Updating its feature set. 3) Expressing observations in terms of features. 4) Predicting likely outcomes based on an internal model. 5) Selecting an action based on the expected reward of likely outcomes. 6) Updating its world model.
Fig. 2. A Morris water maze-type two-dimensional visual servoing task. a) The reward signal as a function of the agent’s gaze position. It is 1 at the center of the image and falls off as $1/d$, where $d$ is the distance of the gaze position from the center of the image. b) The task environment. The white field with black square provided a visual world. The frame representing the agent’s field of view and the location of its gaze is also shown. The agent executed horizontal and vertical panning movements to adjust its gaze position. c) The agent’s gaze position history, vertical component. After approximately 12,000 time steps the gaze focused more on the high reward region. d) The agent’s gaze position history, horizontal component. e) Average reward per time step. After approximately 12,000 time steps the average reward stabilized to roughly twice its initial value. f) Receptive fields for each of the 56 groups. The inputs selected for inclusion in each group are shown here. Light-responsive inputs are gray, and dark-responsive inputs are black. The tight spatial groupings of pixels resulted from their correlation and not from any information about their location in the array. For the most part, the groups in the top two rows are of individual pixels. The groups in the lower rows are mostly higher level combinations of features from top row groups. g) Features created from the first group in panel f. These features primarily show horizontal and vertical edges at varying positions. The third feature also shows a corner. The nature of these features follow intuitively from the position of the group; it is expected that the upper-left hand corner of the field of view should be exposed to the top edge, the left edge, and the upper-left hand corner of the black box in the image.

In addition to creating and updating the feature set at each time step, the feature creator projects the inputs onto exiting features to calculate feature votes. These feature votes are then subjected to a winner-take-all operation, such as might be implemented in a neural network with mutual inhibition. A single feature in each group remains active, and the set of active features is passed on to the reinforcement learner. It is also fed back and combined with the next observation to form the input for the next time step. The recursive nature of the feature creation algorithm allows more complex features to be created from combinations of simpler ones.

The reinforcement learner takes in feature activity and reward and selects an action to execute. The reward map associates features with reward by approximating the correlation between reward and each feature. An attention filter selects the most salient feature at each time step as the attended feature. Working memory is a weighted combination of several recent attended features and any recent actions. The attended feature and working memory are used to update the model. The model is a table of cause–effect pairs, where each effect is an attended feature and each cause is the working memory from the preceding time step. The model also contains a record of the number of times each pair is observed. Rarely observed pairs are periodically removed. This table of cause–effect pairs provides a record of common transitions in feature space, as well as any actions that may have been taken to precipitate them.

III. RESULTS

A simulated task similar to the Morris water maze [2] was constructed to demonstrate BECCA in operation. (See Figure 2.) Complete MATLAB code for the simulation and BECCA implementation (version 0.3.5) can be found at [3].

ACKNOWLEDGEMENT

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.

REFERENCES